FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA – UNIVEM
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

CARLOS EDUARDO MARTINELLI BERGANTIN

ANÁLISE DE BOAS PRÁTICAS PARA O DESENVOLVIMENTO DE *WEB WAPPS*

MARÍLIA
2014

CARLOS EDUARDO MARTINELLI BERGANTIN

ANÁLISE DE BOAS PRÁTICAS PARA O DESENVOLVIMENTO DE *WEB APPS*

Trabalho de Curso apresentado ao Curso de Bacharelado em Sistemas de Informação da Fundação de Ensino "Eurípides Soares da Rocha", mantenedora do Centro Universitário Eurípides de Marília – UNIVEM, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador
Profº: Me. Ricardo José Sabatine

MARÍLIA
2014

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO – AVALIAÇÃO FINAL

Carlos Eduardo Martinelli Bergantin

ANÁLISE DE BOAS PRÁTICAS PARA O DESENVOLVIMENTO DE WEB WAPPS

Banca examinadora da monografia apresentada ao Curso de Bacharelado em Sistemas de Informação do UNIVEM/F.E.E.S.R., para obtenção do Título de Bacharel em Sistemas de Informação.

Nota: 10 (Dez)

Orientador: Ricardo José Sabatine

1°. Examinador: Renata Aparecida de Carvalho Paschoal

2°. Examinador: Fábio Dacêncio Pereira

Marília, 04 de dezembro de 2014.

Martinelli, Carlos Eduardo

ANÁLISE DE BOAS PRÁTICAS PARA O DESENVOLVIMENTO DE *WEB WAPPS* / Carlos Eduardo Martinelli; orientador: Profº. Me. Ricardo José Sabatine. Marília, SP: [s.n.], 2014.

76 folhas

Monografia (Bacharelado em Sistemas de Informação): Centro Universitário Eurípides de Marília.

Dedico esse trabalho primeiramente à Deus
pela realização de um sonho, a minha filha Maria Luiza
e aos meus familiares, amigos e professores pelo incentivo.

AGRADECIMENTOS

Primeiramente, à Deus pelas forças concedidas, e pela oportunidade de concluir mais uma etapa na minha vida.

À minha esposa Mariana e a minha filha que sempre estiveram ao meu lado me acalmando e me dando força para realização de mais um sonho, juntos.

Aos meu familiares e amigos que sempre me apoiarão e acreditaram nas realizações de minha vida.

A família eFinance pelos incentivos e a dedicação em me ajudar, especialmente ao Rogerio Bertini Guilen, o qual muito me auxiliou.

Aos professores e colaboradores do UNIVEM, em especial a Prof. Ms. Giuliana Marega Marques, e ao meu orientador Prof. Ms. Ricardo José Sabatine pela ajuda e dedicação para que o trabalho fosse concluído com sucesso.

Ao pessoal da sala que durante esses quatro anos mostram o quanto é importante batalharmos unidos pelos objetivos, principalmente na formação acadêmica.

Ao Grupo Nota 10 sendo eles, Jorge Pereira, Rafael Akira, Italo Inoue, Jéssica Oliveira, Fernando Mazetti, pessoas que fizeram grande parte na minha formação acadêmica, se tornando mais do que colegas de sala, e que vão continuar presentes em minha vida com certeza.

Obrigado a todas as pessoas que contribuíram para meu sucesso e para meu crescimento como pessoa. Sou o resultado da confiança e da força de cada um de vocês.

"Concentre-se nos pontos fortes, reconheça as fraquezas,
agarre as oportunidades e proteja-se contra as ameaças."
Sun Tzu

"O verdadeiro perigo não é que computadores começarão a pensar como homens,
mas que homens começarão a pensar como computadores."
Sydney J. Harris

"Julgue seu sucesso pelas coisas que você teve que renunciar para conseguir."
Dalai Lama

BERGANTIN, Carlos Eduardo Martinelli. **Análise de boas práticas para o desenvolvimento de web apps**. 2014. 73 f. Trabalho de Curso (Bacharelado em Sistemas de Informação) – Centro Universitário Eurípides de Marília, Fundação de Ensino "Eurípides Soares da Rocha", Marília, 2014.

RESUMO

Com a evolução constante das variedades de aplicativos móveis que demandam flexibilidade no acesso e troca de informações, aumenta a necessidade da qualidade e composição sobre sua arquitetura. É importante compreender quais as melhores práticas que se encaixam nas aplicações móveis a serem desenvolvidas. O presente trabalho visa estudar e analisar as boas práticas do ponto de vista dos desenvolvedores e usuários, onde uma *Web App* foi implementada, visando analisar os fatores que incluem interface com o usuário, facilidade de desenvolvimento, desempenho, infraestrutura de comunicação e manutenção. Outro objetivo, é avaliar os impactos de boas práticas de otimização de aplicações *web* amplamente utilizadas, uma vez que, em dispositivos móveis algumas práticas podem não ter os benefícios pretendidos, ou, podem até mesmo prejudicar o desempenho.

Palavras-Chave: aplicação móvel, *Web* App, sistema *web*, interface do utilizador, usabilidade.

BERGANTIN, Carlos Eduardo Martinelli. **Análise de boas práticas para o desenvolvimento de web apps**. 2014. 73 f. Trabalho de Curso (Bacharelado em Sistemas de Informação) – Centro Universitário Eurípides de Marília, Fundação de Ensino "Eurípides Soares da Rocha", Marília, 2014.

ABSTRACT

With the constant evolution of varieties of mobile applications that require flexibility in it access and swap of information, enhance the need for quality and composition of its architecture. It is important to comprise what the best practices that fit in the mobile applications to be developed. This work aims to study and to analyze good practices from the perspective of developers and users, where a Web App was implemented in order to analyze the factors that include interface with user, facility of development, performance, communication infrastructure and maintenance. Another objective is to evaluate the impacts of good optimization practices of web applications widely used, since in practice some mobile devices can not have the intended benefits, or it may even degrade performance.

Key words: mobile application, Web App, Web system, User interface, usability.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

3DES	Triple Data Encryption Standard
3G	Terceira Geração de Tecnologia de Internet Móvel
AES	Advanced Encryption Standard
API	Application Programming Interface
CSS	Cascading Style Sheets
DES	Data Encryption Standard
DOM	Document Object Model
FT	Financial Times
FTP	File Transfer Protocol
GPRS	General Packet Radio Service
GPS	Global Positioning System
HMAC	Hash-based message authentication code
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
IIS	Internet Information Services
iOS	iPhone Operation System
IP	Internet Protocol
JSON	JavaScript Object Notation
LAN	Local Area Network
MD5	Message-Digest algorithm 5
MVC	Model View Controller
NCSA	National Center for Supercomputing Applications
PDA	Personal digital assistant
REST	Representational State Transfer
SDK	Software Development Kit
SHA1	Secure Hash Algorithm 1
SMS	Short Message Service
SVG	Scalable Vector Graphics
URL	Uniform Resource Locator
W3C	World Wide Web Consortium

WHATWG	Web Hypertext Application Technology Working Group
Wi-Fi	Tecnologia para transmissão de dados por rede sem fio
XHR	XMLHttpRequest

SUMÁRIO

INTRODUÇÃO

Nos últimos anos vem ocorrendo um aumento acelerado na utilização dos dispositivos móveis, tornando-se o principal mecanismo de conexão à *internet*. Entre os vários motivos que estão levando à isso, podemos destacar os dispositivos cada vez mais poderosos em relação a capacidade de armazenamento, processamento, memória, etc.

A cada dia que passa os usuários estão buscando cada vez mais praticidade e agilidade ao acessar as informações na *web*, isso inclui tarefas simples do seu dia a dia. Um relatório apresentado pela Ericsson, prevê que no final de 2019 teremos cerca de 9,2 bilhões de conexões móveis no mundo. Sendo que 5,6 bilhões serão de smartphones [ERICSSON, 2014].

Já a Google dispõe de um relatório que informa que em 2012 a frequência do uso da *internet* móvel era de 42% e nos aparelhos estavam instalados 14 aplicativos. Em 2013 esses dados subiram para 62% e 17 aplicativos instalados. O relatório também menciona que 86% dos usuários acessam um meio comunicação, sendo rede sociais, e-mail, entre outros, e que 63% utilizam para leituras de notícias, via portais, jornais ou revistas. [GOOGLE SERVICE, 2014].

Visando esse potencial, as empresas estão investindo cada vez mais no desenvolvimento, seja para criação como para adaptação de seus *websites,* ou até mesmo sistemas *web* para dispositivos móveis. Entretanto, em alguns casos elas não estão tendo o devido sucesso, seja pela falta de planejamento ou utilização incorreta das práticas de programação.

O desenvolvimento voltado para dispositivos móveis tem as suas particularidades, tais como tamanhos, armazenamento, processamento, utilização das redes móveis, entre outros. Essas particularidades estão gerando novos desafios que requerem uma estratégia completamente diferente da qual é utilizada para computadores de mesa.

Atualmente, com a chegada de novas tecnologias como HTML5, CSS3, APIs, etc., os desenvolvedores passaram a ter mais poder ao implementarem. Em contrapartida quando utilizados muitos recursos a performance de carregamento do *website* diminui, podendo ocasionar uma insatisfação do usuário pela espera.

Pensando nas diferenças entre os dispositivos móveis e os computadores de mesa, alguns profissionais da área de desenvolvimento para *web*, estão abordando com alta frequência, a otimizações. Existem grupos como o W3C, que já disponibiliza um documento com um conjunto de boas práticas para o desenvolvimento de aplicação *web* voltada para

dispositivos móveis.

Este trabalho abordará algumas otimizações tanto no lado do cliente como no servidor, por meio de técnicas recomendadas pelo documento de boas práticas do W3C, visando minimizar o tempo de carregamento, latência da rede, números de requisições HTTP, entre outros.

Motivação e Justificativa

Tem-se como objetivo estudar e analisar algumas das melhores práticas para se desenvolver uma aplicação *web* para dispositivos móveis, mais específicamente, os smartphones. Para realizar o estudo e a análise das melhores práticas para desenvolvimento mobile, foi utilizado um framework e o documento criado pelo grupo W3C.

Contribuir com as empresas e os desenvolvedores, para que possam utilizar algumas das melhores práticas, proporcionando aos usuários uma agradável experiência na utilização da aplicação.

Objetivos

Tem-se como objetivo estudar e analisar algumas das melhores práticas, para se desenvolver uma aplicação *web* para dispositivos móveis, mais específico, os smartphones. Para realizar o estudo e a análise das melhores práticas, foi desenvolvido uma aplicação web utilizado um framework e as técnicas de boas práticas disponibilizada no documento criado pelo grupo W3C, onde mostram as melhores práticas para desenvolvimento mobile.

De modo a contribuir com as empresas e os desenvolvedores, para que possam utilizar algumas das melhores práticas, proporcionando aos usuários uma agradável experiência na utilização da aplicação.

Organização do Trabalho

A organização do trabalho encontra-se organizado da seguinte forma:

- Capítulo 1 – Estudo de Aplicações Móveis: Foi realizada uma revisão bibliográfica sobre aplicações móveis.
- Capítulo 2 – Web Mobile: Foi realizada uma pesquisa sobre as otimizações voltadas para aplicações móveis.
- Capítulo 3 – Implementação de uma Web Apps: Foi implementado um protótipo para realizar otimizações levantadas no segundo capitulo.
- Capítulo 4 – Resultados e Análises: Foram aplicadas as técnicas de otimização e verificados os resultados obtidos.
- Capítulo 5 – Conclusão: Apresentada a conclusão do trabalho.

1 ESTUDO DE APLICAÇÕES MÓVEIS

Aplicações móveis são *softwares* desenvolvidos para fornecer aos usuários serviços semelhantes a de um computador tradicional, como tarefas, funções ou até mesmo um passa tempo. As aplicações são desenvolvidas para serem executadas em dispositivos móveis, como Smartphone, Tablet, PDA, etc., combinando assim com suas tecnologias de processamento, memoria, comunicação, localização, entre outros. Os aplicativos podem ser instalados durante a fabricação do aparelho, ou por meio de *download* nas lojas *online*, como App Store, Google Play, e Windows Phone Marketplace.

A App Store foi lançada em 2008, é um serviço exclusivo para os aparelhos da Apple que utilizam como plataforma principal o iOS, tais como iPhone, iPad, iPod Touch, etc. permitindo aos usuários efetuarem os *downloads* diretamente pelos dispositivos móveis, ou via computador pela iTunes Store. Nesse caso pode ser transferido para o dispositivo na sua sincronização com o computador [APPLE, 2014].

O Google Play começou a existir da unificação do Google com a Android Market em 2012. É uma loja *online* mantida pela Google, que permite aos usuários efetuarem *downloads* diretamente nos dispositivos móveis que utilizam como plataforma o Android, ou via computador pela sua loja, podendo ser transferidos ao dispositivo pela sua sincronização [GOOGLE, 2014].

O Windows Phone Marketplace foi disponibilizada pela primeira vez em 2012, permitindo aos usuários a efetuarem os *downloads* de aplicativos móveis que utilizam como plataforma o Windows 8, Windows RT ou Windows Phone [MSDN, 2014].

Os aplicativos móveis disponibilizados nas lojas podem ser pagos ou gratuitos, dependendo dos fabricantes e são distribuídos em categorias como: Jogos, Musicas, Entretenimento, Livros, entre outros.

A área de aplicações móveis está em constante crescimento, devido à sua popularidade e tecnologia apresentada nos dispositivos móveis, tornando assim o desenvolvimento de aplicações mais visado. Porém, os números de plataformas e ambientes de desenvolvimento estão crescendo proporcionalmente, deixando mais difícil a escolha no processo para o desenvolvimento de um aplicativo, levando em consideração os recursos, o tempo, a experiência e os custos.

1.1 Tipos de aplicações móveis

Os tipos de aplicações móveis influenciam diretamente na usabilidade, performance, custo e tempo da aplicação. Com isso torna-se claro a importância de analisar os requisitos do projeto, para que a escolha da arquitetura tenha as melhores funcionalidades, facilidades e produtividades, obtendo os objetivos necessários para satisfação do cliente.

As aplicações móveis podem ser desenvolvidas em três tipos de arquitetura, conhecidas como nativa, híbrida e *web*. Cada tipo possui particularidades, vantagens e desvantagens.

1.1.1 Aplicações nativas

Aplicações Nativas (*Native App*) são aplicações desenvolvidas especificamente para uma plataforma (Sistema operacional) como Android, iOS, Windows Phone, entre outras. Assim sendo, as aplicações só funcionarão na plataforma para a qual foi desenvolvida [IBM, 2014].

Para desenvolver um aplicativo tem que conhecer a linguagem e as ferramentas especificas que são oferecida pelos fabricantes. No caso de iOS utiliza-se a linguagem de programação Objective-C, *framework* de desenvolvimento Cocoa Touch e a ferramenta Xcode. Após o seu desenvolvimento pode ser publicado na loja da Apple conhecida nos dispositivos como App Store [APPLE, 2014]. O Android utiliza a linguagem Java com o Android SDK na ferramenta Eclipse. Para ter a disponibilidade do usuário efetuar o *download* do aplicativo desenvolvido, é preciso publicar na loja do Google, conhecida como Play Store [GOOGLE, 2014]. No caso do Windows Phone a linguagem utilizada é o C# com Windows Phone SDK e com a ferramenta Visual Studio. Quando a aplicação estiver pronta, pode ser disponibilizada na loja do Windows, conhecida nos dispositivos como Windows Store, para que usuário tenha acesso ao *download* [MSDN, 2014].

Os aplicativos podem ser instalados durante a fabricação do aparelho, ou através de *download* nas lojas *online*, já mencionado acima. Eles poderão ser classificados em categorias como jogos, entretenimento, livros, músicas, notícias, etc. Para que os aplicativos possam ser publicados nas lojas *online*, precisam ter o certificado da aprovação e a licença para desenvolver na plataforma desejada. Cada loja tem seus termos e particularidades para ter aprovação do aplicativo desenvolvido.

Vantagens das aplicações nativas

- Experiência aprimorada ao usuário: Por ser um aplicativo nativo possibilita a melhor utilização dos recursos gráficos, interface, APIs do Sistemas Operacionais como câmera, bússola, GPS entre outros, proporcionando assim uma interação mais rápida e agradável do usuário.
- Armazenamento de dados: Aplicação nativa pode realizar o armazenamento local tanto em arquivos, como em banco de dados *offline*.
- Integração: A integração entre aplicativos, torna-se mais simples, o desenvolvimento mais seguro por conta dos aplicativos estarem utilizando e acessando a mesma linguagem e métodos do Sistema Operacional.

Desvantagens das aplicações nativas

- Custo alto no desenvolvimento em multiplataformas: Por existirem várias plataformas, o aplicativo requer uma versão separada para cada sistema operacional, tornando assim o tempo e custo mais elevados.
- Atualização na versão aplicativo: Para cada alteração no aplicativo, seja ele na estrutura ou apenas no conteúdo, é necessário que o desenvolvedor gere novas versões para cada sistema operacional, e assim fazendo com que o usuário efetue um nova instalação do mesmo no dispositivo. Não tem a garantia que todos os usuários estejam usando a mesma versão.
- Aprovação nas lojas *online*: Cada loja tem seus termos e particularidades no processo de aprovação dos aplicativos, tornando a publicação de um aplicativo mais complicado.

1.1.1.1 Arquitetura do aplicativo

As aplicações nativas, por serem desenvolvidas especificamente para uma única plataforma, têm a possibilidade de oferecer alta qualidade na experiência do usuário, com o acesso do dispositivo móvel, utilizando todos os recursos oferecidos pelo mesmo, como é ilustrado na Figura 1.

Figura 1 - Arquitetura do funcionamento de uma aplicação nativa

Fonte: CECCATO, 2014

1.1.2 Aplicações Híbridas

As aplicações híbridas combinam o desenvolvimento dos códigos nativos, com os códigos utilizadas na *web* (HTML, CSS, JavaScript), que são executados em conjunto, mantendo o acesso aos recursos oferecidos pelo aparelho [IBM, 2014].

Com o intuito de auxiliar e facilitar a implementação das aplicações híbridas, existem algumas ferramentas de terceiros, como PhoneGap, Titanium, Kivy, que empacota o código escrito com as tecnologia da *web*, para o formato nativo da plataforma do aparelho. Porem cada ferramenta tem a suas características e particularidades no processo de implementação [PHONEGAP, 2014; TITANIUM, 2014; KIVY, 2014]. Não são recomendados para aplicações intensivas como, jogos ou que requeiram acesso total ao aparelho.

Vantagens das aplicações híbridas

- Suporta múltiplas plataformas: Existem *frameworks* que transformam aplicações escritas na tecnologia *web* (HTML, CSS, JavaScript), para serem interpretadas como uma aplicação nativa.
- Facilidade no desenvolvimento: Por existirem alguns *frameworks* que dão suporte à diversas plataformas, não necessita a reescrita do código fonte para cada uma, já que o *framework* interpreta para a plataforma desejada.

- Custo reduzido no desenvolvimento: Os custos podem variar de acordo com o projeto, em comparação ao desenvolvimento de uma aplicação nativas o seu custo podem ser relativamente menores.

Desvantagens das aplicações híbridas

- Aprovação nas lojas: Por serem desenvolvidas com as tecnologias da *web* (HTML, CSS, JavaScript), e interpretada para linguagem nativa por ferramentas de terceiros, não se garante a aprovação do aplicativo.
- Performance e Interface: Embora as aplicações mesclarem o código nativo com a *web*, ainda tem suas limitações perante ao aparelho e a plataforma, conseguintemente tornando a performance e a interface limitada.

1.1.2.1 Arquitetura do aplicativo

Aplicações Híbridas combinam as tecnologia da *web* com as tecnologias de um aplicativo nativo, mantendo assim o acesso aos recursos oferecidos pelos dispositivos e plataformas, como ilustra na Figura 2.

Figura 2 - Arquitetura do funcionamento de uma aplicação híbrida

Fonte: CECCATO, 2014

1.1.3 Aplicações Web (Web Apps)

Aplicações *Web* mais conhecido como *Web App*, são construídas com as tecnologias da *web*, tais como HTML, CSS, Javascript, entre outras, e projetadas especificamente para

dispositivos móveis. As aplicações são interpretadas por meio de um *browser* (navegador de *internet*) desde que tenha acesso a *internet*, não sendo necessário a instalação no dispositivo para o funcionamento, tornando independente de plataformas e do dispositivo que o usuário esteja usando [OEHLMAN, BLANC, 2011].

Um grande número de empresas já estão oferecendo seus serviços nos dispositivos móveis de seus *softwares*. O aumento destes serviços, vem ocorrendo devido as melhorias que estão sendo feita pelos provedores de redes na *internet*, no aumento da velocidade da banda e na redução de custo.

Arquitetura multiplataforma é altamente recomendável para aplicações empresariais, visto que diminui o tempo necessário para chegada ao mercado. Ao utilizarmos códigos já existentes, podemos chegar aos nossos clientes de diferentes plataformas de maneira muito mais rápida e simples. [NOKIA, 2014]

Para otimizar e simplificar o processo de implementação do, existem várias ferramentas baseada nas tecnologias *web*, com o intuito de aproximar ou até mesmo igualar a interface e performance de uma aplicação nativa. As ferramentas que dispõem desses recursos são o Sencha Touch, jQuery Mobile, Dojo Mobile, AppMobi, entre outras [SENCHA, 2014; JQUERY, 2014; DOJO, 2014; APPMOBI, 2014;].

Vantagens das aplicações Web Apps [SIX REVISIONS, 2014]

- Utiliza tecnologias *web*: São escritos em HTML, CSS, JavaScript e linguagens server-side ou *framework* de aplicação *web* da escolha do desenvolvedor (PHP, Rails, Python, etc.)
- Suporta múltiplas plataformas: Por ser executado em qual quer *browser* com acesso à *internet*, funcionará independentemente da plataforma do dispositivo, acessando um número maior de usuários.
- Menor custo: Existem mais desenvolvedores com o conhecimento nas tecnologias da *web*, tornando mais fácil de encontrar mão de obra qualificada e de baixo custo.
- Tempo de desenvolvimento: As aplicação *web* se torna mais rápido no desenvolvimento, por não ter a necessidade dos conhecimentos específicos das plataformas e dos dispositivos.
- Processo de aprovação: Não há necessidade de aprovação das lojas, por não precisar efetuar o *download* da aplicação, já que o acesso é via URL de um navegador *Web*.
- Atualizações instantâneas: Por ser escritos na *web*, todos acessam a mesma aplicação com a mesma versão, atualizando uma única vez, atualiza para todos os usuários, independendo da plataforma e dispositivo.

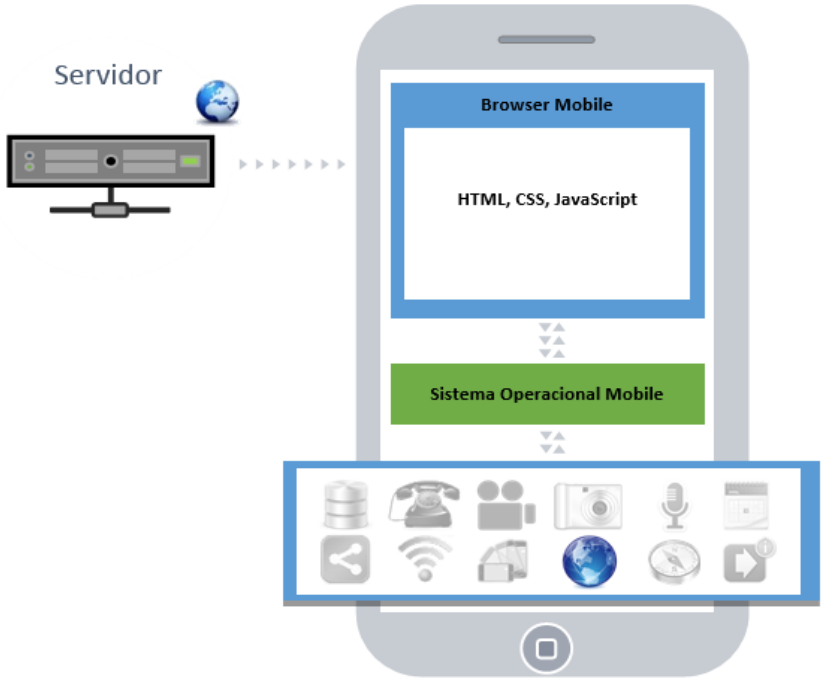
Desvantagens das aplicações Web Apps [SIX REVISIONS, 2014]

- Recursos e funcionalidades do dispositivos e plataformas: *Web app* possui acessos parciais das funcionalidades e dos recursos que os *hardwares* e *softwares* disponibilizam, por ser desenvolvida com as tecnologias da *Web*.
- Conexão: O tempo de resposta se torna mais lento pelo fato de utilizar conexão com a *internet*, que acaba sendo o responsável pelo funcionamento da aplicação.
- Localizar: A sua localização pode se tornar um pouco mais difícil devido a sua inexistência em *App stores*.
- Segurança: Por não ser avaliado, não existe um sistema de controle de qualidade para aplicativos *Web*. Os usuários não têm a mesma segurança que uma aplicação nativa.

1.1.3.1 Arquitetura do aplicativo

Aplicações *Web*, são desenvolvidas com as mesmas tecnologias da *web*, com isso as aplicações não tem acesso aos recursos que uma aplicação nativa, tornando a experiência do usuário inferior a de aplicações nativas ou híbridas, como ilustra na Figura 3.

Figura 3 - Arquitetura do funcionamento de uma aplicação *web*



Fonte: CECCATO, 2014

1.1.4 Comparações entre os tipos de aplicações

Com o intuito de facilitar o entendimento na arquitetura dos três tipos de aplicações móveis (Nativa, Híbrida e Web App) foram realizados uma breve comparação entre as mesmas.

Tabela 1 - Comparação entre os tipos de aplicações, Nativo, Híbrido e Web App

	Nativo	***Híbrido***	***Web App***
Multiplataforma	Desenvolvido para um certo tipo de dispositivo.	*	Desenvolvido para funcionar nos *browsers*, independentemente de plataforma.
Acesso à internet	Não necessita.	Necessita parcialmente.	Necessário
Recursos do dispositivo / plataforma	Utiliza recursos do dispositivo, como câmera, GPS, etc.	Utiliza recursos do dispositivo, como câmera, GPS, etc.	Possuem acesso limitado os recursos.
Performance	Usuário tem uma experiência melhor, mais rápida e fluida, por estar utilizando todos os recursos do aparelho.	*	Usuário pode não ter uma boa experiência, por depender parcialmente da internet para o seu funcionamento.
Implementação	Cada plataforma tem a sua linguagem de programação especifica Objective-C – iOS, Java – Android, Visual C++ - Windows Mobile, etc.	Mescla a linguagem de programação, *Web* com a linguagem Nativa da plataforma utilizada.	Utiliza a linguagem *web*, HTML, JavaScript, CSS, entre outras, desde que seja compatível aos navegadores.
Instalações / Atualizações	Devem ser feitas manualmente pelo usuário, fazer novo *download* e instalação do aplicativo.	*	São feitas pelo próprio servidor, sem precisar da intervenção do usuário.
Localizar	Existem lojas para ajudar a encontra o aplicativo	Existem lojas para ajudar a encontra o aplicativo.	Como não há loja de *Web App*, pode se tornar um pouco mais difícil encontra-lo.
Processo de aprovação	Sim, a maioria das lojas requerem aprovação, antes de publicar.	*	Não, basta acessar o endereço da aplicação.
Controle de versão	Alguns usuários podem optar por não atualiza, resultando em que todos os usuários não estejam na mesma versão.	*	Mantem todos os usuários nas mesmas versões.
Custo de desenvolvimento	Custo alto, por ter que lidar com mudanças de diferentes dispositivos/plataformas.	*	Baixo, por ser linguagem *web*, não precisar lidar com mudanças de diferentes dispositivos/plataformas.

Fonte: SALESFORCE, 2014; WEBMONKEY, 2014; SIX REVISIONS, 2014;

Onde temos um asterisco é porque depende de como o projeto foi definido diante ao desenvolvimento da aplicação híbrida.

1.1.5 Considerações finais

Os aplicativos Nativos, Híbridos ou Web Apps, contem suas características próprias, tendo vantagens e desvantagens em alguns aspectos, deixando indefinido qual tipo é o melhor.

A escolha de qual tipo de aplicação será utilizada, depende do que a empresa ou usuários estejam precisando. Para ter uma boa escolha no tipo que irá usar no desenvolvimento, é necessária uma análise detalhada de todos os requisitos que irão ser implementados no projeto, perante as necessidades definidas.

Algumas considerações devem ser analisadas no projeto para ter a melhor escolha do tipo de aplicação. São elas:

- Acesso as funcionalidades do dispositivo;
- Necessidade ao acesso da *internet*;
- Compatibilidade com as plataformas;
- Performance de utilização da aplicação;
- Interface com o usuário;
- Público alvo;
- Custo de desenvolvimento;

Devido as necessidades das empresas, a escolha do tipo de aplicativo acaba se tornando uma escolha difícil. Duas grandes empresas foram para caminhos distintos na criação de aplicativos:

- Facebook: Mark Zuckerberg, o CEO do Facebook, que é a maior rede social no momento, revelou que a estratégia de investir no HTML 5 em vez de aplicativos nativos, foi um dos grandes erros que ele cometeu. O fundador do Facebook decidiu mudar de estratégia que para ele estava errada, alterando o foco para os aplicativos nativos. Desde então, o número de usuários vem subindo, por conta da aplicação estar com uma melhor performance [CONSTINE, 2012].
- Financial Times: Um dos maiores jornais do mundo e tradicional do meio impresso, optou em mudar o foco do seu investimento. Antes o aplicativo para dispositivos móveis era desenvolvido especificamente com tecnologia nativa. Com isso, o FT vinha enfrentando problemas com as lojas, custos, clientes, entre outros. Acabou resolvendo mudar o foco de desenvolvimento, mudando para a tecnologia *web*, mais especificamente o HTML 5, que está proporcionando uma melhor experiência, tanto para o jornal como para seus clientes e leitores [DREDGE, 2013].

2 WEB MOBILE

A World Wide Web teve seu início em 1980 na Suíça. O precursor da ideia foi o britânico Timothy John Berners-Lee. Em 1990 o britânico Berners-Lee, um dos primeiros a utilizar o *hypertext* como forma de armazenar informações, criou o primeiro navegador *web* do mundo, o WorldWideWeb. O navegador teve esse nome por ser o único meio de acessar a *web*. Logo após um tempo o WorldWideWeb foi renomeado para Nexus, com o intuito de evitar confusões entre o navegador e a própria *web*, por terem nomes parecidos [W3C, 2014].

Entretanto, para ter um bom domínio do desenvolvimento *Web,* algumas características e técnicas são fundamentais para um bom conhecimento das tecnologias:

- URL: *Uniform Resource Locators:* são utilizadas para localizar um recurso, que esteja disponível em uma rede, seja ela intranet ou até mesmo *internet*. Utilizam alguns dos protocolos de comunicação HTTP, FTP, TELNET, entre outros, para acessar os recursos disponíveis, em tal endereço [KIOSKEA, 2014].
- HTML: *Hypertext Markup Language:* é uma linguagem de marcação utilizada para desenvolver páginas na *Web.* Permite a criação de um documento que possa ser transmitido pela *internet* e lido de qualquer computador.
- HTTP: *Hypertext Transfer Protocol:* é um protocolo de comunicação usado na *World Wide Web* para trocas de informações do *browser* ao servidor. Esse protocolo é a base da comunicação dos dados da Word Wide Web.

2.1 HTML

A linguagem “Hypertext Markup Language” (Linguagem de marcação hypertext) ou mais conhecida como HTML, teve seu início em 1991, permitindo a ligação de uma página a outra, pelas tags, que são comandos para inserir algum tipo de elemento ou características da mesma. A sua popularidade cresceu através do navegador *web* Mosaic, que foi desenvolvido no NCSA [VENKATESH, 2012].

Em 1994 surge a primeira melhoria da versão do HTML, o HTML 2.0 tendo apenas correções e padronizações das principais características da versão anterior [VENKATESH, 2012].

Após dois anos o HTML apresenta um conjunto de tecnologias como o “Cascading Style Sheets”, conhecido como CCS, que é uma linguagem de estilos, onde podem ser definidas as apresentações do documento, que são escritas em linguagem de marcação. E também o JavaScript, que é uma linguagem de programação interpretada, para que pudessem ser

executados *scripts* no lado do cliente, nos navegadores. Quando as tecnologias se juntam, podem disponibilizar vários recursos, e até mesmo podendo tornar o HTML dinâmico [VENKATESH, 2012].

Logo após, a linguagem foi reconhecida pela W3C, que é um grupo responsável por manter o padrão dos códigos. O grupo trabalhou em uma nova versão para que a linguagem tornasse uma pratica comum na *web* [W3C HTML 5, 2014].

Desde de então o HTML 3 vem sendo atualizado com as novas versões: HTML 3.2, HTML 4, HTML 4.01 e a versão atual HTML 5, trazendo novos recursos e conjuntos de tecnologias [W3C HTML 5, 2014].

O HTML foi desenvolvido de modo que funcione em todos os equipamentos, que sejam capazes de utilizar as informações na *Web*, independente do tamanho de resolução, cores, dispositivos, entre outros.

2.2 HTML 5

A criação do HTML 5 veio com a união dos grupos W3C e Web Hypertext Application Technology Working Group ou conhecido como WHATWG, onde antes a W3C estava apenas focada em criar a segunda versão do XHTML, e o grupo WHATWG estava trabalhando um uma nova versão do HTML que trazia mais flexibilidade a produção de *websites* e sistemas que são baseados na *web* [W3C HTML 5, 2014].

O HTML 5 é uma atualização baseada nas versões anteriores, com melhorias significantes, tornando algo a mais, do que simplesmente uma linguagem de marcação, fornecendo ferramentas para o JavaScript e o CSS fazerem o melhor trabalho possível em conjunto com o HTML [W3C HTML 5, 2014].

Desde da sua criação o HTML 5 vem se tornando uma das principais tecnologias na interação com a *web*, por dispor de novas funcionalidades, elementos, recursos, integrações e diversas outras novidades, criando assim novas experiências aos usuários e desenvolvedores. Esta linguagem de marcação oferece a possibilidade de criar aplicações mais complexas com os conjuntos de padrões abertos que são oferecidos.

2.2.1 Compatibilidade com os navegadores

Devido a existência de diversos dispositivos como *Tablets*, *Smartphones*, *Notebooks*, entre outros, que acessam a *internet*, torna mais difícil para os desenvolvedores, manter um

bom nível de compatibilidade com todos os navegadores, levando em conta as particularidades de cada um deles. Cada *browser* tem o seu motor de renderização que é responsável pelo processamento dos códigos da página. Os navegadores Safari e Google Chrome, utilizam o motor Webkit, já o Firefox, Mozilla e Camino utilizam o Gecko, as versões do Internet Explorer que vão do 4 ao 9, utilizam o Trident. O motor Presto é utilizado pelo Opera das versões que vão do 7 ao 10 [W3C HTML 5, 2014].

O mais interessante é fazer um código compatível com os principais motores de renderização, para que tenha uma amplitude maior nos *browsers* e consequentemente nos usuários.

O motor de renderização mais compatível atualmente com os padrões do HTML 5, é o Webkit, sendo assim os navegadores que tiverem com esse motor de renderização, é o mais indicado a utilizar as tecnologias do HTML 5 [W3C HTML 5, 2014].

Existe uma grande preocupação que deve ser levado em conta, com *websites* e aplicações *web* onde utiliza as tecnologias do HTML 5 para o seu funcionamento: a retrocompatibilidade com as versões anteriores dos *browsers*. As empresas vem cada vez mais se empenhando em atualizar os navegadores, para que sejam compatíveis com os padrões do HTML 5, proporcionado a utilização dos novos recursos e tecnologias oferecidos.

Na Tabela 2 ilustra a compatibilidade dos navegadores com alguns módulos do HTML 5.

Tabela 2 - Compatibilidade dos browsers com os principais módulos do HTML 5.

	Safari	***Chrome***	***Opera***	***Firefox***	***IE 8***	***IE 9***
Local Storage	Sim	Sim	Sim	Sim	Sim	Sim
Histórico de Sessão	Sim	Sim	Sim	Sim	Sim	Sim
Aplicações Offline	Sim	Sim	Não	Sim	Não	Não
Novos tipos de campos	Sim	Sim	Sim	Não	Não	Não
Form: Autofocus	Sim	Sim	Sim	Não	Não	Não
Form: Autocomplete	Não	Não	Sim	Não	Não	Não
Form: Required	Sim	Sim	Sim	Não	Não	Não
Video, Audio e Canvas Text	Sim	Sim	Sim	Sim	Não	Sim

Fonte: W3C HTML 5, 2014

2.2.2 Novas funcionalidades do HTML 5

O HTML 5 incluiu novidades significantes em vários aspectos que nas versões anteriores não existiam. As mudanças não foram apenas inclusões ou remoções de *tags*, mas sim melhorias para suprimir a necessidade de usar outras tecnologias para criação de certas funcionalidades. Por serem utilizadas tecnologias de terceiros, era comum encontrar erros e incompatibilidades nas funcionalidades criadas.

Alguns dos recursos que foram incluídos no HTML 5, são [W3C HTML 5, 2014]:

- Estrutura do corpo: Os *websites* tem formatos comuns, que são compostos por elementos, tais como: cabeçalho, corpo, rodapé, etc.
- *Tags* para conteúdo específicos: Antes para ser utilizado os conteúdos enriquecidos nas páginas, utilizava-se apenas um única *tag* para diversos tipos de conteúdo. Agora é possível utilizar *tags* especificas para cada tipo de conteúdo em particular, tais como áudio, vídeo, etc.
- Canvas: É um componente novo, que permite desenhar diversos tipos de formas que podem ser animadas e ter interação com o usuário, por meio de funções de uma *API*, sem necessidade de instalar nenhum *plugin*.
- Revisão ortográfica e gramatical: Oferece recursos de revisão ortográfica e gramatical aos usuários.
- Banco de dados: Por meio de uma *API*, as páginas *web*, poderão permitir o armazenamento de dados no agente do usuário, sem precisar definir a expiração dos dados. A documentação sugere que o espaço de armazenamento seja de 5MB para cada domínio.
- Aplicações *web Offline*: Permite que o usuário trabalhe na aplicação *web*, sem estar com acesso à *internet*, como se estivesse utilizando uma aplicação local.
- Geolocalização: As páginas da *web*, poderão tem acesso a localização geográfica do usuário, podendo ser de três maneiras, Geolocalização por IP, Triangulação GPRS e ao GPS do dispositivo.
- *Drag and Drop*: Permite o usuário arrastar e soltar os objetos. Antes era necessário utilizar ferramentas de terceiros, para a criação da funcionalidade de arrastar e soltar.
- Novos suporte para interface: Com o intuído de melhorar a experiência do usuário em relação a interface, a atualização do HTML veio som suporte a diversos recursos como CSS3, SVG, DOM, entre outros.
- Novos elementos de Áudio, Vídeo e *Codecs*: Teve inclusão de *tags* especificas para reprodução de áudio e vídeo sem a necessidade de utilizar mecanismos de terceiros.
- Histórico de Sessão: Permite via Javascript ter controle do histórico de navegação do cliente.

2.3 Hypertext Transfer Protocol (HTTP)

Segundo Steve Souders em seu livro High Performance Web Sites *"...somente 10-20% do tempo total do carregamento de uma página é gasto para receber o HTML do servidor para o navegador. Você precisa focar nos outros 80-90% se você quiser tornar suas páginas visivelmente mais rápidas..."* [SOUDERS, 2008].

Hypertext Transfer Protocol (HTTP) é um método utilizado para enviar e receber informações na *web*. O protocolo é baseado em requisições e respostas entre o servidor e o cliente. A mensagem HTTP é composta por uma linha de requisição, as linhas do cabeçalho e o corpo da mensagem [HYPERTEXT TRANSFER PROTOCOL, 2014].

Quando é realizada uma requisição ao servidor utilizando o protocolo HTTP é necessário informar o método, mais conhecido como "verbos" da ação a ser executada.

Os métodos mais utilizados, são o *GET* (Solicita informação ao servidor.), *POST* (Insere as informação enviadas no corpo da mensagem), *DELETE* (Remove as informações) e o *PUT* (Atualiza as informações). Existem mais 4 métodos, que não são muitos utilizados como o *HEAD, OPTIONS, TRACE e CONNECT* [HYPERTEXT TRANSFER PROTOCOL, 2014].

Toda requisição recebe um código de resposta conhecido como status. Com ele é possível saber se a operação solicitada ao servidor foi realizada com sucesso. Os códigos são formados por três dígitos, sendo que o primeiro representa a classe que pertence e o restante as condições. As classes foram classificadas em cinco tipos, para facilitar o entendimento da resposta, as classes são:

- 1xx: *Informational* (Informação) utilizada para informar ao cliente que a requisição foi recebida e está sendo processada.
- 2xx: *Success* (Sucesso) informa ao cliente que a requisição foi sucedida.
- 3xx: *Redirection* (Redirecionamento) são medidas adicionais que devem ser tomadas a fim de completar à solicitação.
- 4xx: *Client Error* (Erro no cliente) informa que provavelmente houve um erro na solicitação, impedindo o servidor que processe.
- 5xx: *Server Error* (Erro no servidor) indica que o servidor teve um erro interno, esses erros são geralmente do servidor e não da solicitação.

Os códigos de status mais comuns de encontrar ao acessar um site são o 200 (*OK*), 304 (*Not Modified*), 401 (*Unathorized*), 404 (*Not Found*), 500 (*Internal Server Error*), 502 (*Bad Gateway*) e 503 (*Service Unavailable*) [HYPERTEXT TRANSFER PROTOCOL, 2014].

No cabeçalho é possível definir algumas técnicas para otimizar o desempenho da

comunicação. O *cache* é uma das técnicas onde avisa o navegador que ele pode armazenar uma cópia daquele recurso para evitar que o *download* e/ou requisições sejam feitas novamente. Com o Cache-Control é possível definir o tempo que os recursos irão permanecer válidos no navegador [FADINO, 2014].

Outra maneira de otimizar é habilitar o gzip no servidor para comprimir os dados das respostas e reduzir a largura de banda. Porém, o uso da CPU será maior para descompactar as respostas [GOOGLE 2].

Segurança

Como relação à segurança, as conexões provêm autenticação (HMAC, MAC), integridade (MD5, SHA1) e confidencialidade (DES, 3DES, AES) das requisições e dos dados solicitados. São utilizadas quando os dados importantes são transmitidos, geralmente utilizada pelos sites de *Internet Banking* e de comércio eletrônico [CERT.BR, 2012].

O protocolo HTTP oferece esquema de autenticação de usuário para ter acesso aos recursos sigilosos. O esquema utiliza de inclusões de dados no cabeçalho (ou *header*) tanto pelas requisições como pelas respostas realizadas pelo servidor.

Autenticação HTTP Basic envia para o servidor usando a codificação *Base64* na forma de texto. Porém o tipo de autenticação Basic é considerado um dos mais inseguros, pelo fato dos dados serem trafegados na rede como texto plano, sendo possível a interceptação por terceiros e tornando acessível [ANDRADE, 2010].

Open Authorization (OAuth) é um protocolo aberto que permite autorização segura utilizando padrões de métodos simples, garantindo acesso de um usuário de um site a um aplicação externa. OAuth tornou-se mais segura e simplificada com as interações dos dados protegidos pelo cliente ou fornecedor [ANDRADE, 2010].

2.4 Tecnologias para desenvolvimento de Web Apps

Podemos observar que existem diversos *frameworks* que já utilizam as funcionalidades do HTML 5 e que oferecem soluções viáveis para o desenvolvimento de aplicações para dispositivos móveis, atendendo a um maior público, independe de plataforma ou fabricante dos dispositivos. Entre os principais podemos destacar: jQuery Mobile, Sencha Touch e AppMobi [SENCHA 2014; JQUERY 2014; APPMOBI 2014].

2.4.1 jQuery Mobile

jQuery Mobile é um *framework* de interface de usuário baseada nas bibliotecas anteriores, o jQuery e jQuery UI. A principal linguagem utilizada para construção é o JavaScript [JQUERY, 2014].

Atualmente, o *framework* encontra-se na versão 1.4.2 e com esta versão é possível encontrar diversas funcionalidades e interfaces, como *listviews*, *pop-ups*, *collapsible*, entre outras, e é totalmente gratuito [JQUERY, 2014].

2.4.2 AppMobi

Com esse *framework* é possível desenvolver aplicações com visual adequado para dispositivos móveis por meio de arquivos HTML. O seu ambiente de desenvolvimento consiste em uma extensão no navegador Chrome [APPMOBI, 2014].

O *framework* foi adquirido pela empresa Intel em 2012, onde é muito defendida a utilização de linguagens como HTML 5, CSS e JavaScript [APPMOBI, 2014]. Se destacou pela facilidade no processo de desenvolvimento, tanto para aplicação *web* como para aplicações híbridas [APPMOBI, 2014].

2.4.3 Sencha Touch

Sencha Touch foi otimizado para aplicativos móveis que trabalham em diversas plataformas, tamanhos e navegadores. Para tornar a escrita do código mais simples possível, ele fornece a utilização da arquitetura em Model View Controller (MVC). Esta arquitetura mantém o código mais limpo, testável e de fácil manutenção [SENCHA, 2014].

Atualmente o *framework* está na versão 2.3.1, utilizando as tecnologias da *web* como HTML 5, JavaScript e CSS para construção de sistemas *web* para dispositivos móveis. A maior parte de sua biblioteca é desenvolvida utilizando o JavaScript [SENCHA, 2014].

O grande destaque do Sencha Touch é a facilidade da escrita do código para sistemas *web* e a reutilização do mesmo código fonte para criar aplicações nativas. O *framework* permite aos desenvolvedores criarem aplicativos poderosos que funcionem no iOS, Android, BlackBerry, Windows Phone, entre outros. Outra vantagem do Sencha é a comunidade que vem aumentando rapidamente e que dispõe de diversos documentos, fóruns, conferencias, cursos, entre outros, para auxiliar no processo da utilização da ferramenta [SENCHA, 2014].

2.4.4 Comparativo entre as ferramentas

Para o desenvolvimento de um aplicativo *web*, existe uma grande diversidade de *frameworks* que disponibilizam as ferramentas necessárias para facilitar o desenvolvedor a implementar bons sistemas. No presente trabalho foi comparado, uns dos três maiores *frameworks* que estão disponíveis no mercado.

Pode se observar que cada *frameworks* tem suas características próprias, não definindo qual é o melhor entre eles. A Tabela 3 mostra um comparativo entre algumas características dos frameworks selecionados.

Tabela 3 - Comparação entre os *frameworks* escolhidos.

	JQuery Mobile	***AppMobi***	***Sencha Touch***
Categoria	Web Mobile	Híbrido / Web Mobile	Híbrido / Web Mobile
SDK Gratuito	Sim	Sim	Sim
Possui IDE	Sim	Sim	Sim
IDE Gratuita	Não	Sim	Não
Linguagens utilizados	HTML5, JavaScript e CSS	HTML5, JavaScript e CSS	HTML5, JavaScript e CSS
Permite PhoneGap	Não	Sim	Sim
Web mobile app	Sim	Sim	Sim
Suporte	Sim	Sim	Sim
Documentação	Sim	Sim	Sim
Nível de aprendizado	Médio	Médio	Fácil
Fórum	Sim	Sim	Sim

Fonte: AUTORIA PRÓPRIA

2.5 Melhores Práticas para desenvolvimento de Aplicação Web Mobile

Diversas aplicações *web* são desenvolvidas diariamente sem uma abordagem adequada. O desenvolvimento de aplicações *web* requer uma visão completamente diferente do que acostumamos ver em uma aplicação convencional, para computadores de mesa. Muito se discute qual a melhor prática a ser utilizada no desenvolvimento, de forma que atenda todos os requisitos de funcionalidade e proporcione ao usuário uma boa experiência de uso.

Existem alguns atributos que poderão ser levados em consideração no desenvolvimento de uma aplicação *web* mobile. São eles [MARSHALL, 2013]:

- Escalabilidade (*Scalability*): Com a necessidade do uso da aplicação se tornando cada vez maior, torna-se muito importante que a aplicação consiga suportar grandes números de usuários, sessões e operações simultâneos.
- Disponibilidade (*Availability*): Ter disponibilidade mesmo se trabalhar *offline*, para que possa ser utilizado a qualquer momento, tendo sempre um bom grau de disponibilidade do aplicativo.
- Performance (*Performance*): A performance de uma aplicação *web* se refere a realização de tarefas entre um determinado período de tempo.
- Tempo de resposta (*Responsiveness*): O tempo que o usuário leva para obter uma resposta do servidor deverá ser o mínimo possível para que tenha uma boa experiência de uso.

Os atributos fundamentais para se ter uma boa aplicação, é a escalabilidade e o tempo de resposta. As aplicações *web* que não atendem aos atributos, poderão proporcionar experiências de uso desagradáveis aos usuários, levando a diminuição do uso pela sua improdutividade no seu dia a dia [MARSHALL, 2013].

2.5.1 World Wide Web Consortium (W3C)

O grupo W3C desenvolveu um documento com um conjunto das melhores práticas de desenvolvimento voltado para *web* móvel, tendo como objetivo criar um padrão internacional, garantindo um bom desenvolvimento e a evolução das aplicações *web* para dispositivos móveis [W3C BEST PRACTICES, 2014].

O documento consiste em 5 dicas essenciais para uma boa aplicação na *web* para dispositivos móveis, são elas:

*2.5.1.1 **Design flexível** [W3C BEST PRACTICES, 2014]*

As aplicações *web* são executadas em diversos ambientes de dispositivos móveis. Sendo flexível, será possível abordar mais usuários com custo reduzido.

- Métodos de interação com a aplicação deverão ser considerados quando se projeta um interface, por existirem diversos tipos de dispositivos, os principais métodos que são utilizados, como o Cursor que navega por teclas que permite controlar o cursor no *ecrã*, Focus são os saltos de um elemento para o outro, e o Toque que são os eventos gerados diretamente pelo toque nos dispositivos.

- Fluxo de texto é importante em telas pequenas, para que não seja necessário a utilização do *scroll* horizontal, garantindo o enquadramento dos textos nas alterações do *ecrã*.
- Prefira detecção do lado do servidor quando possível, para determinar as propriedades do contexto de entrega e adaptar as respostas para o cliente antes da transferências, evitando assim a transferência de dados desnecessários ou incompatível.
- Utilizar a detecção do lado do cliente quando não for possível determinar as propriedades do contexto de entrega do lado do servidor. Uma vez obtida as informações, poderá ser utilizadas para adaptar a apresentação no momento ou enviar pedidos de alternativa para o servidor.
- Utilizar classificação do dispositivos para simplificar a adaptação do conteúdo, tendo em vista desenvolver para um grande número de dispositivos. O ideal é classificar os dispositivos em classes e construir uma versão para cada classe. Permitindo assim a explorar as capacidades de cada dispositivo com o mesmo código gerenciável.
- Ter suporte a um ambiente não JavaScript, as aplicações baseadas em *script* XHR não são compatíveis com todos os navegadores, então considere a possibilidade de utilizar uma versão em "Synchronous FROM" em vez de solicitações XHR.

*2.5.1.2 **Reduzir a utilização da rede** [W3C BEST PRACTICES, 2014]*

Utilizar os recursos de protocolo *web* apropriados e métodos para reduzir os gargalos e latência da rede.

- Utilizar a compressão de dados para tornar a transmissão mais eficiente e rápida.
- Evitar utilizar os recursos externos, as aplicações utilizam grandes quantidades de recursos como imagens, *scripts*, entre outros.
- Evitar os redirecionamentos de pedidos, normalmente são utilizados para trocar informações entre servidores. Os atrasos ocorridos pelos redirecionamentos tem um maior impactos nas redes móveis.
- Diminuir os dados e as aplicações, removendo os espaços em brancos e otimizando os códigos das páginas, tornarão o trafego e a execuções mais rápidas e confiáveis.
- Evitar a utilização dos *cookies* desnecessariamente, os recursos estáticos não necessitam de *cookies*, podendo tornar o desempenho melhor.
- Armazenar em *cache* os dados Ajax que são solicitados pelo cliente, os dados deverão ser armazenado da mesma forma que outros conteúdos.
- Utilizar o recurso de *cache* por referencias dos recursos que mudam ocasionalmente, evitando que o navegador valide novamente o cabeçalho.

- Utilizar com moderação os *cookies*, pois cada pedido será enviado para o servidor, usando uma quantidade de dados excessiva poderão afetar negativamente o desempenho, principalmente em um rede móvel.

*2.5.1.3 **Otimizar o tempo de resposta** [W3C BEST PRACTICES, 2014]*

Os mínimos detalhes são importantes em uma aplicação *web*, podendo melhorar significativamente a experiência geral da utilização dos usuários.

- Agregar imagens estáticas em um único recurso composto, melhorar a performance da aplicação. Muitas vezes as aplicações dependem de imagens estáticas para montar ícones, botões, etc.
- Incluir as imagens de fundo no CSS para evitar requisições HTTP. Utilizando a "base64" aumenta cerca de 10% do tamanho da imagem após a sua compressão no gzip.
- Manter o tamanho da memória para o Document Object Model (DOM) razoável, por ter dispositivos com limitações. As páginas grandes e complexas podem exceder o seu limite e ocasionar erros imprevisíveis.
- Otimizar a inicialização da aplicação para que o usuário tenha uma experiência agradável no começo. O HTML 5 oferece recursos para aplicações *web* se aproximarem das aplicações nativas, no que diz respeito de inicialização podendo ser utilizadas mesmo quando não há conexão com a rede.
- Minimizar a percepção da latência do utilizador é um fator importante para usabilidade geral da aplicação *web*.

*2.5.1.4 **Explorar funções especificas** [W3C BEST PRACTICES, 2014]*

Os dispositivos móveis tem funções especificas, que poderão ser utilizadas pelas aplicações *web*.

- Disponibiliza a opção de "Click-to-Call" para facilitar nas interações da aplicação com os dispositivos, algumas funções podem realizar chamadas, enviar SMS, entre outras opções, dependendo das funcionalidades do dispositivo.
- Utilize as tecnologias móveis para inicializar aplicações caso o dispositivo tenha os métodos. Os métodos permitem enviar notificações e atualizações sejam enviadas para o usuário, mesmo estando fora do contexto da aplicação.
- Utilizar o elemento do navegador para obter o tamanho do *ecrã* e exibir as páginas desenhadas para *desktop* e redesenhar para as tela pequena. Isso pode afetar as aplicações que já foram otimizadas para as telas pequenas. O "viewport meta tag" diz ao dispositivo em qual escala irá renderizar a página.

- Utiliza tecnologias apropriadas para armazenar dados no lado do cliente. Se tornou uma das melhores formas de garantir um bom tempo na inicialização e na capacidade de resposta. Se o dispositivo suportar as *APIs*, fornecerá mecanismos para armazenar dados maiores do que os *cookies*.

*2.5.1.5 **Princípios do desenvolvimento web** [W3C BEST PRACTICES, 2014]*

Os dispositivos móveis são mais uma forma de acessar as aplicações *web*. Os princípios de desenvolvimento para *web*, são aplicáveis para todos os dispositivos que acessam as aplicações *web*.

- Replicar os dados locais para um servidor para garantir a segurança das informações caso aconteça alguma coisa com o dispositivo e também para tornar disponível a outros utilizadores da mesma aplicação.
- Evitar a execução de dados JSON não confiáveis, pois existem diversos código maliciosos que poderão denegrir as informações ou até mesmo apaga-las. A técnica JSON é muito comum e útil para enviar dados a um cliente, por executar mais rapidamente do que alternativas da mesma.
- Garantir a consistência de estado entre os dispositivos, quando uma informação é atualizar em um dispositivo, deverão ser visto da mesma forma em outro dispositivo. Tornando assim uma única informação consistente na aplicação.

*2.5.1.6 **Controle ao usuário** [W3C BEST PRACTICES, 2014]*

Os diapositivos móveis são utilizados em diversos contextos no dia a dia, indo do lazer até tarefas importantes e urgentes de negócios. O controle do usuário informa as ações das aplicações no dispositivo.

- Garantir aos usuários a informação sobre atualizações de dados pessoais e do dispositivo para evitar surpresas. Os usuários deverão ser informados no primeiro acesso na aplicação ou nas informações acessadas.
- Ativar o “Sign-in” automático por ser mais difícil a introdução de texto em um dispositivo móvel do que um computador de mesa. Com o “Sign-in” ativado, as próximas sessões que serão iniciadas não irão solicitar a senha ao usuário.
- Oferecer aos usuários a escolha de interface, as decocções automáticas nem sempre são eficazes para definir qual das versão da aplicação é apropriada para o dispositivo.
- Não alterar o focus das atualizações dinâmicas da sessão na aplicação, para evitar a perca da leitura e criar confusão para o usuário. Um método do JavaScript oferece focus para mover a página na onde estava antes da atualização.

2.6 Considerações finais

Desde do surgimento do HTML 5, os *websites* e aplicações *web* estão sofrendo diversas mudanças, proporcionando aos usuários e desenvolvedores uma experiência que não se imaginava. Antes eram necessário utilizar ferramentas de terceiros para certas funcionalidades, recursos, entre outros. O HTML 5 está se tornando uma solução rápida e eficaz no processo de desenvolvimento, porém ainda existem barreiras a serem superadas, como a dependência dos usuários atualizarem os seus navegadores para versões mais recentes.

Podemos observar que muitas funcionalidades, recursos e *API's*, já estão sendo utilizadas e principalmente em aplicações *web* ou *websites*, que tem como público alvo os usuários de dispositivos móveis. No mercado encontramos vários *frameworks* que já utilizam das novas ferramentas do HTML 5, os principais são jQuery Mobile, Sencha Touch e AppMobi.

Os desenvolvedores deverão estar atentos quanto a utilização dos novos recursos, de maneira que não prejudique ou comprometa a utilização e a navegação dos usuários em seus sistemas *web* ou sites.

3 IMPLEMENTAÇÃO DE UMA WEB APPS

Um dos objetivos deste projeto era desenvolver um protótipo de uma aplicação *web* móvel, onde será analisado algumas práticas, dicas e desafios que são encontrados no processo de desenvolvimento, devido a sua dependência com a *internet*. Serão utilizadas algumas dicas e práticas presente no documento desenvolvido pelo grupo W3C, com as melhores práticas para o desenvolvimento *web mobile*.

Com o intuito de facilitar e agilizar no processo de implementação do protótipo, serão utilizados algumas ferramentas, métodos e arquitetura, tais como o *framework* Sencha Touch, API do Twitter, arquitetura MVC, entre outros.

Neste capitulo abordará o desenvolvimento do protótipo e descrever as principais funcionalidades do mesmo e aplicar algumas técnicas de otimização tanto no lado do cliente como no servidor.

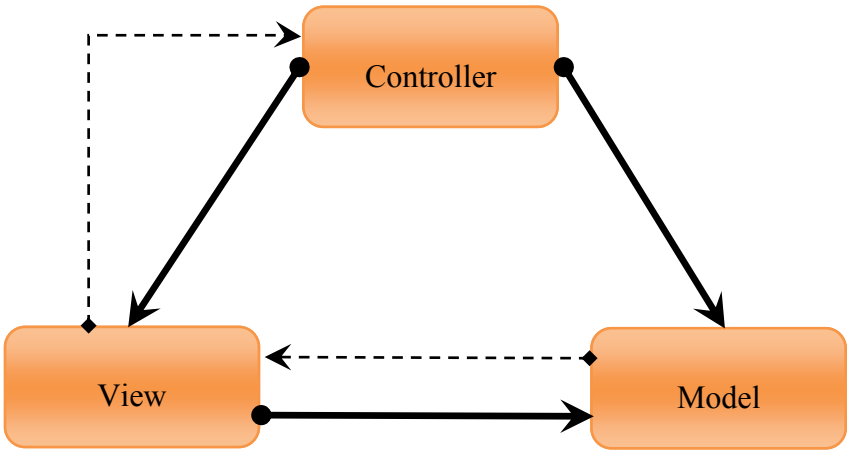
3.1 MVC

MVC é um padrão de arquitetura de *software*, que trabalha em conjunto em uma mesma estrutura, dividido em três camadas, cada camada tem suas características e responsabilidade especificas. Utilizando o padrão possibilita utilizar diversas linguagens de programação, permitindo que as equipes sejam distintas em suas atribuições. As camadas são da arquitetura MVC são separadas da seguinte forma [PADRÕES DE PROJETO, 2009]:

- Model (modelo): Representa os dados que serão acessados ou modificados pela aplicação.
- View (visão): Consiste em apresentar os dados aos usuários de forma que seja possível a interpretação e interação das informações.
- Controller (Controlador): Na camada do controlador são delegadas todas as decisões sobre o comportamento da interface, intermediando os dados provindos da camada modelo com a camada da visão.

Na Figura 44, é mostrado o relacionamento entre as camadas Model, View e Controller, como foi descrito acima.

Figura 4 - Relacionamento entre os componentes da arquitetura MVC



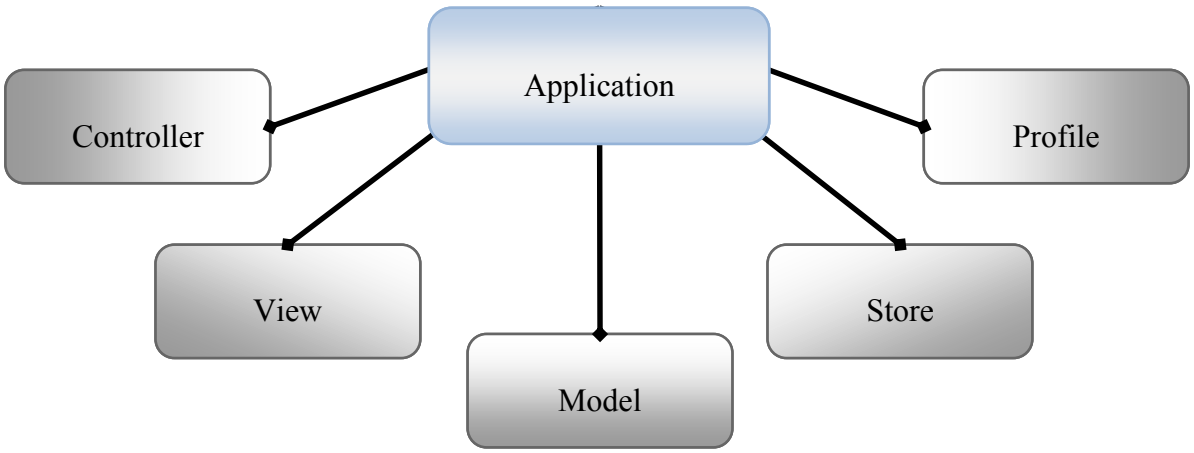
Fonte: PADRÕES DE PROJETO, 2009

3.1.1 MVC Sencha Touch

O Sencha Touch na versão 2.4 foi otimizado para a criação de aplicações móveis, que funcionarão em diversas plataformas. Com o intuito de tornar a implementação mais agradável, o *framework* fornece uma arquitetura simplificada, utilizando o padrão de modelo MVC, mantendo o código fonte limpo, testável, e de fácil manutenção. Na criação de uma aplicação usando o modelo de arquitetura MVC, é carregado os conjuntos Controller, View, Model, Store e Profile.

Na Figura 55, é mostrado a relação entre as camadas fornecidas pelo framework, com a aplicação:

Figura 5 - Camadas fornecidas pelo *framework* Sencha Touch



Fonte: SENCHA, 2014

As camada tem a suas características e importância, abaixo segue a descrição delas:

- *Controller*: são responsáveis por responder a eventos que ocorrem na aplicação. Os controladores monitora os eventos requeridos pela interface e executa uma ação com base no evento.
- *View*: embora os valores do aplicativo estarem nos modelos e controladores, os usuários interagem diretamente com a visão. As visões são responsáveis pela exibição do dados para os usuários, utilizando os componentes do próprio *framework* ou até mesmo criados na implementação.
- *Model*: representa um tipo de objeto de dados na aplicação. Na maneira mais simples o modelo é apenas um conjunto de campos e seus dados. No modelo é possível utilizar de formas diferentes, como *proxy, associations, validations e fields.*
- *Store:* é uma parte muito importante da arquitetura, por ter a função de carregar a maior parte dos dados ligado a aplicação. Os dados carregados irão alimentar os componentes, como listas, *DataViews*, entre outros.
- *Profile:* pelo *framework* funcionar em uma grande variedade de dispositivos móveis, com capacidades e tamanhos de telas deferentes, o *profile* permite personalizar a interface da aplicação, dependendo do aparelho que o usuário esteja utilizando, como *tablets*, *smartphones*, entre outros. Evitando ter que desenvolver o aplicativo várias vezes, apenas para fornecer uma interface para o usuário deferente.

Como podemos observar acima, as camadas tem suas características e funcionalidades especificas, na implementação de um *web app*. As camadas podem ser carregadas separadamente, conforme as necessidades no processo de implementação.

3.2 Descrição da aplicação proposta

Atualmente diversas empresas e desenvolvedores questionam até que ponto as recomendações oferecidas pelo grupo W3C no documento, ajudam no desenvolvimento da aplicação para dispositivos móveis na *web*.

Afim de analisar algumas recomendações do documento, foi implementado uma aplicação na *web* para dispositivos móveis, assim podendo apontar as importâncias da utilização da boas práticas estabelecidas pelo W3C.

A aplicação que foi desenvolvida, consiste em utilizar como provedor de dados, as publicações realizadas pelos usuários de uma das maiores e mais utilizada rede social do mundo, o *microblog* Twitter.

O Twitter disponibiliza uma API de serviço, com diversas funcionalidades, onde o

serviço é totalmente gratuito e documentado. Na documentação pode-se encontrar vários exemplos de desenvolvimento em diferentes tipos de linguagens de programação.

Observando a documentação e as diversas funcionalidades disponíveis na API, será utilizado a arquitetura REST no formato JSON, com o intuito de simplificar o trafego dos dados provenientes do *Web Service*.

Visando simplificar o desenvolvimento, será utilizado o *framework* Sencha Touch, onde o mesmo dispõe da arquitetura MVC e diversos recursos e funcionalidades, como citado acima. A aplicação será implementada em .Net utilizando a linguagem de programação orientada a objetos C#.

O protótipo da aplicação *web mobile* foi composto por algumas características, tais como:

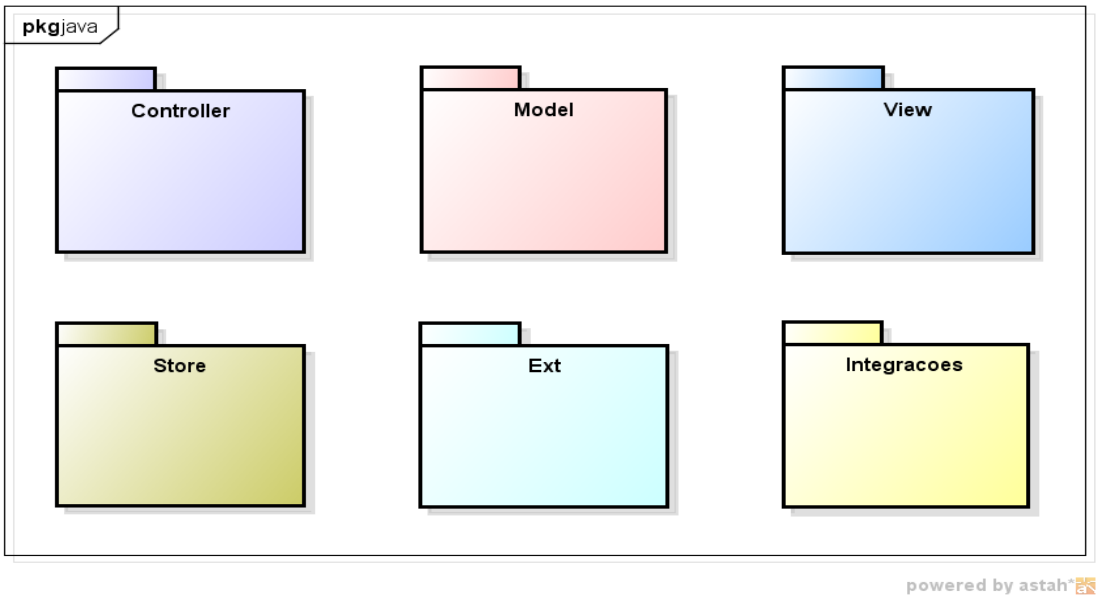
- Tela para acessar a aplicação, onde irá validar o acesso do usuário, e entrar automaticamente. Nessa tela terão os campos, Login, Senha, uma opção de ficar autenticado e o botão entrar.
- Ter um fluxo adequado dos textos, publicados pelos usuários. (Fluxo de texto e *focus*)
- Carregar as imagens em lista do perfil dos usuários. (Carregamento de imagem em lista)
- Adequação dos componentes aos diversos tamanhos de telas dos dispositivos móveis. (Designer flexível)
- Acesso a serviços externo oferecidos pelo *microblog* Twitter. (Acesso a *API* externas)
- Gerar gráfico adequado em formato de pizza, representado os dados. (Gráficos representando dados)
- Lista com textos e imagens carregadas apenas quando solicitado. (Pool - Será atualizado apenas quando efetuar a busca)
- Lista com textos e imagens carregadas automaticamente quando entrar na aplicação.

3.3 Implementação da Aplicação

Com o intuito de facilitar o entendimento da aplicação, foi desenvolvida utilizando o conceito da arquitetura MVC. O protótipo foi divido em pacotes conforme suas funções, tais como *Controller*, *Model*, *View*, *Store*, *Ext* e Integrações. Cada pacote contém classes e as classes contém seus métodos e funções.

Na Figura 66 é representado o diagrama de pacotes da aplicação.

Figura 6 - Diagrama de Pacotes da aplicação



Fonte: AUTORIA PRÓPRIA

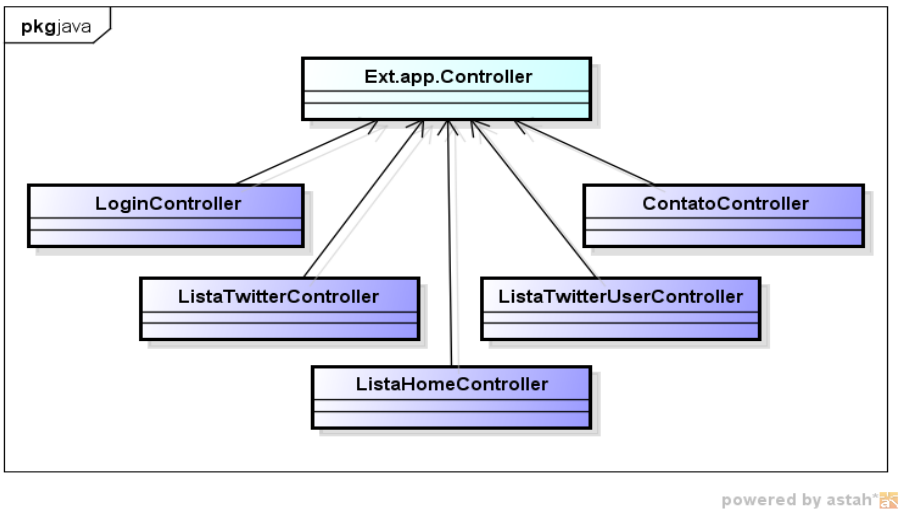
O protótipo é composto pelos pacotes Controller, Model, View, Store, Ext e Integrações, estes pacotes serão apresentados mais detalhadamente abaixo.

3.3.1 Pacote *Controller*

No pacote *controller* foram criadas cinco classes, onde serão gerenciadas as solicitações das classes dos pacotes *View* e *Store*. As classes *controller* foram separadas com o intuito de separar a parte visual da aplicação com a regra de negócios.

Na Figura 77 representa o diagrama de classes do pacote *controller*.

Figura 7 - Diagrama de classes do pacote *Controller*



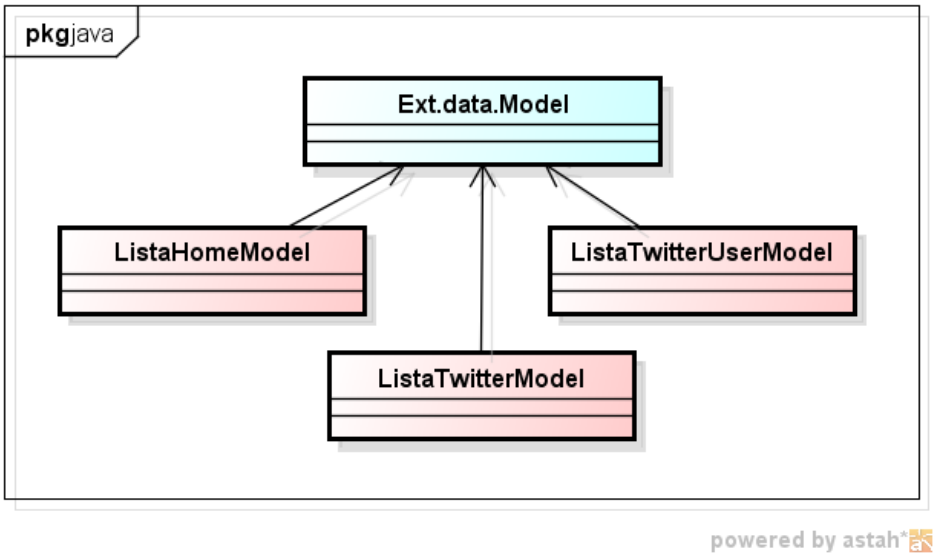
Fonte: AUTORIA PRÓPRIA

3.3.2 Pacote *Model*

As classes que estão no pacote *model* serão utilizadas quando as informações forem consultadas pelas classes dos pacotes *View* e *Store*.

Na Figura 88 representa o diagrama de classes do pacote *model*.

Figura 8 - Diagrama de classes do pacote *Model*



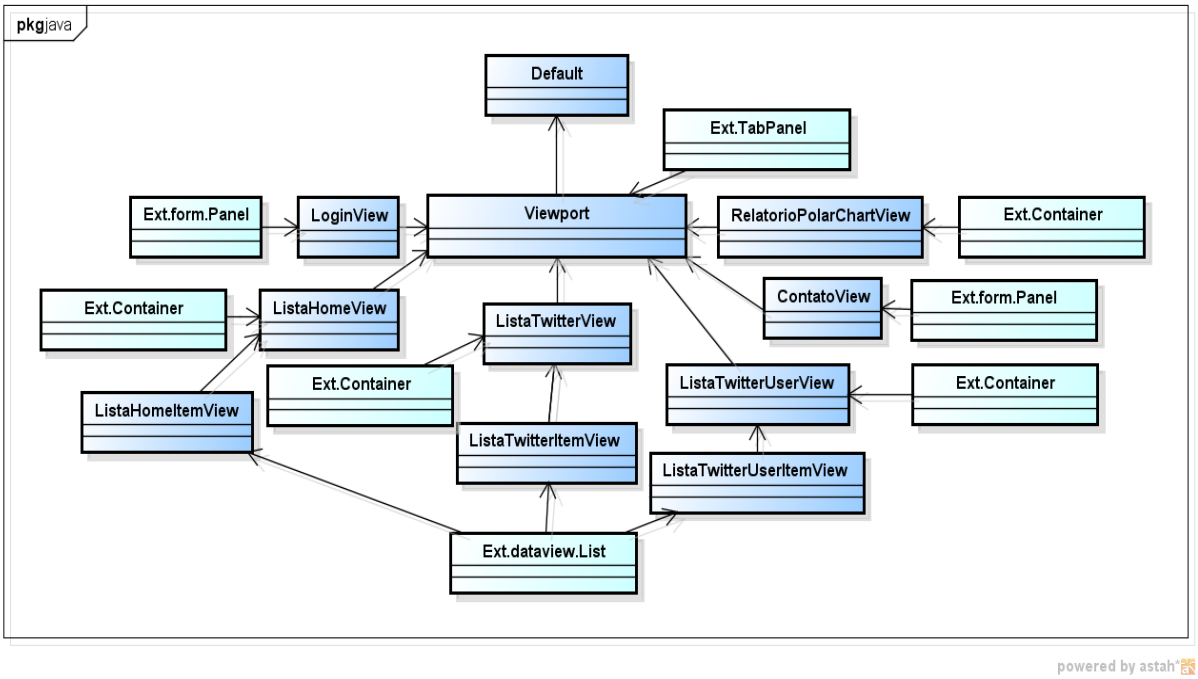
Fonte: AUTORIA PRÓPRIA

3.3.3 Pacote *View*

O pacote *view* possui as classes que irão criar as interfaces visuais, utilizando as classes do pacote *Ext*. As classes *views* foram separadas para deixar o código mais limpo e permitindo a separação entre as visões da aplicação.

Na Figura 99 representa o diagrama de classes do pacote *view*.

Figura 9 - Diagrama de classes do pacote *View*



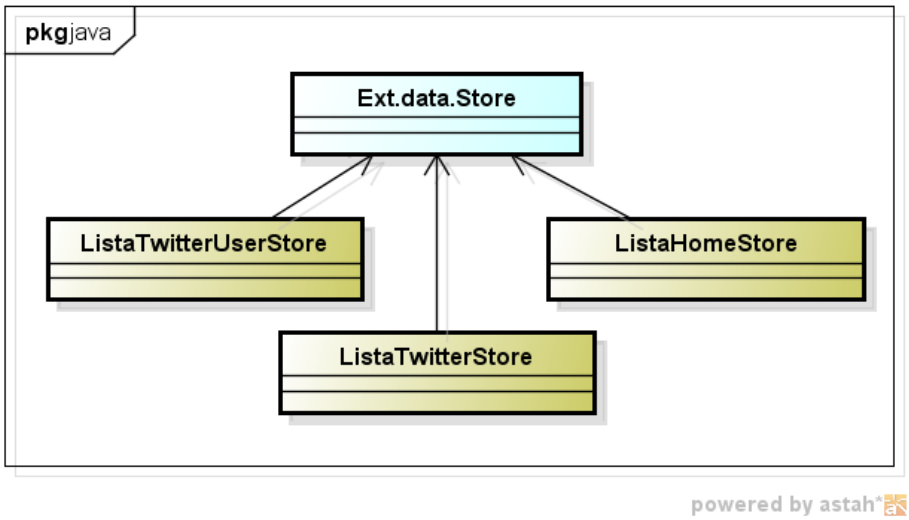
Fonte: AUTORIA PRÓPRIA

3.3.4 Pacote Store

O pacote *store* possui as classes que carregam os dados da aplicação, permitindo a conexão com as partes externas da aplicação.

Na Figura 100 representa o diagrama de classes do pacote *store*.

Figura 10 - Diagrama de classes do pacote *Store*



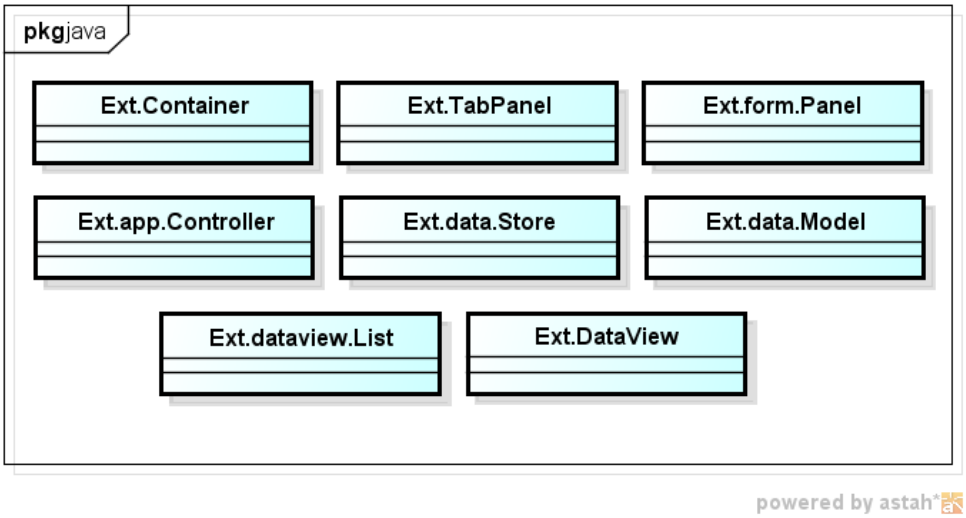
Fonte: AUTORIA PRÓPRIA

3.3.5 Pacote *Ext*

O pacote *Ext*, contém as classes que o *framework* Sencha Touch disponibiliza para criação de uma *web apps,* tanto para parte de *view, store, model, controller e profile*.

Na Figura 111 representa o diagrama de classes do pacote *store* que são utilizados na aplicação.

Figura 11 - Diagrama de classes do pacote *Ext*



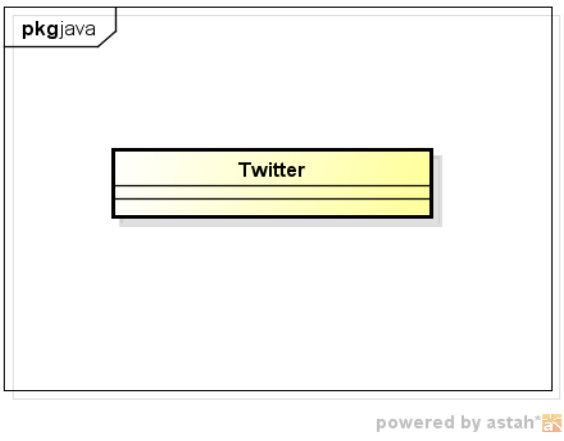
Fonte: AUTORIA PRÓPRIA

3.3.6 Pacote *Integrações*

O pacote integrações será responsável por efetuar requisições externas a um determinado *web services* como o do *microblog* Twitter.

Na Figura 122 representa o diagrama de classes do pacote.

Figura 12 - Diagrama de classes do pacote Integrações



Fonte: AUTORIA PRÓPRIA

3.4 Técnicas de otimização

A quantidade de usuários que acessa a *internet* por meio de dispositivos móveis, cresceu demasiadamente nos últimos anos. Porém os dispositivos móveis são diferentes do computador de mesa, aos quais estamos acostumados a utilizar. Questões como usabilidade, acessibilidade, flexibilidade, entre outras características, são diferentes quando se compara as duas tecnologias.

Pensando nas diferenças e visando a performance das funcionalidades, vamos adotar algumas estratégias e otimizações que poderão ser utilizadas na implementação de um *website* e *webapps*. Serão utilizadas algumas práticas que estão no documento criado pelo grupo W3C, (documento oferece as melhores práticas para o desenvolvimento voltado para *web* móvel).

3.4.1 Técnicas na Autenticação do Usuário

Geralmente as aplicações *web* móveis iniciam-se em uma tela de autenticação, para que os usuários possam ser reconhecidos e suas informações carregadas no *webapps*. Por a tela de autenticação ser utilizada frequentemente quando acessamos uma aplicação, serão aplicadas algumas boas práticas e técnicas de otimização, para facilitar e melhorar a performance das funcionalidades.

Nos dispositivos móveis se torna mais complicada a inserção de textos por serem menores e a interação do usuário com o aparelho fica mais difícil. No protótipo desenvolvido foi adicionada uma autenticação automática, mais conhecida como "*Sing-in*". Quando ativada a autenticação automática, as próximas sessões que serão iniciadas, não irá solicitar ao usuário o *login* e a senha, tornando mais simples o processo de autenticação.

Na tela de autenticação também foram utilizadas algumas recomendações do documento W3C, visando ter um designer flexível, que adequa a *ecrã* ao tamanho da resolução do dispositivo móvel, garantindo o enquadramento adequado dos textos, componentes, entre outros. Foi implementada utilizando o *cache* do navegador do dispositivo móvel para guardar as informações de autenticação.

3.4.2 Técnica de carregamento

Os conteúdo das aplicações móveis podem ser complexos ou simples, dependendo das informações a serem exibidas: textos, imagens, botões, entre outros. Geralmente essas

informações apresenta-se em formato de lista vertical ou horizontal.

A lista como um modelo de interface, é muito eficiente para exibir as informações aos usuários da aplicação. O processo de desenvolvimento se tornou muito eficaz, por utilizar um modelo de navegação com diversas técnicas de carregamento como: paginação, *polling*, *websockets*, entre outras [LEE, VALENTINO, 2005].

As técnicas de carregamento influenciam diretamente na satisfação de uso do usuário com a aplicação, pela dependência de ir ao servidor buscar as informações. As técnicas precisam ser adequadas e otimizadas para que não se torne lento o processo do carregamento, evitando assim a espera do usuário.

O protótipo desenvolvido foi implementado utilizando algumas técnicas de carregamento, como o carregamento *Pull Refresh*, onde o usuário vai ao início da lista, puxa para baixo e atualiza a lista com as informações, esse tipo de carregamento é manual e utiliza uma conexão síncrona.

A técnica de *Polling* também foi implementada. Essa técnica é baseada na automatização das atualizações entre a aplicação e o servidor. Diferente da técnica citada anteriormente, esta não precisa da interação do usuário para atualizar as informações, pois sua ação é realizada constantemente em um período de tempo estabelecido em uma função na aplicação.

Para troca de mensagens entre o cliente e o servidor foi utilizado o formato JSON, que executa mais rapidamente do que as outros formatos. Um importante ponto para tornar a usabilidade da aplicação mais agradável, é minimizar a percepção da latência ao usuário.

As técnicas citada acima que foram implementadas no protótipo, atendem algumas recomendações e boas práticas de desenvolvimento, onde visa tornar o carregamento mais satisfatório ao cliente.

3.4.3 Técnica de entrada de dados

Nas aplicações móveis, por não terem a mesma facilidade na inserção de textos do que um computador de mesa, torna-se muito importante aplicar algumas técnicas e cuidar da redução dos campos a serem preenchidos.

Existem diversas técnicas que podem beneficiar o usuário *mobile* com a utilização de componentes como: caixas *drop-down*, botões de opções, caixas de seleção, posição dos *labels* e a personalização do teclado virtual, onde aparecerá o teclado perante as necessidades do campo, tornando assim, a inserção de dados mais fácil na aplicação [LEE, VALENTINO, 2005].

Visando atender algumas recomendações de técnicas e boas práticas, o protótipo desenvolvido buscou garantir que os campos (*text*, *textarea*, *labels*, entre outros) do formulário estejam com enquadramento adequado ao *ecrã*, eliminando a necessidade de utilizar o *scroll* horizontal, garantindo também alterações da orientação do dispositivo.

Outra técnica aplicada tem como objetivo facilitar a inserção dos dados, utilizando o teclado virtual especifico para o tipo de campo, diferenciando por *e-mail*, números, letras, senha, entre outros. Um ponto importante que foi levado em consideração também, é a utilização dos *labels* acima do campo. Por conta de os navegadores darem *zoom* automaticamente ao campo do formulário que está sendo preenchido, podendo ocasionar alguns problemas quando os *labels* estão alinhados à esquerda.

3.4.4 Otimizações do lado do servidor (Otimizações do servidor web)

As aplicações para dispositivos móveis que dependem das redes *web* de dados como 2G, 3G, 4G, entre outras, são muito sensíveis a qualidade da *internet* disponível. Atualmente se tornou muito importante a otimização, tanto no lado do cliente como no servidor.

Nos servidores *web* é possível efetuar diversas configurações importantes, com o intuito de aumentar o desempenho e a performance das aplicações ou *websites* que lá estão, e reduzir a quantidade de dados transferidos entre os mesmos.

Nesta seção foram aplicadas algumas otimizações visando melhorar o desempenho e a performance do servidor *web*. As técnicas foram aplicadas em um servidor *Windows* 8.1 *Pro*, utilizando o *Internet Information Services* (*IIS*) na versão 8.5.

3.4.4.1 GZIP

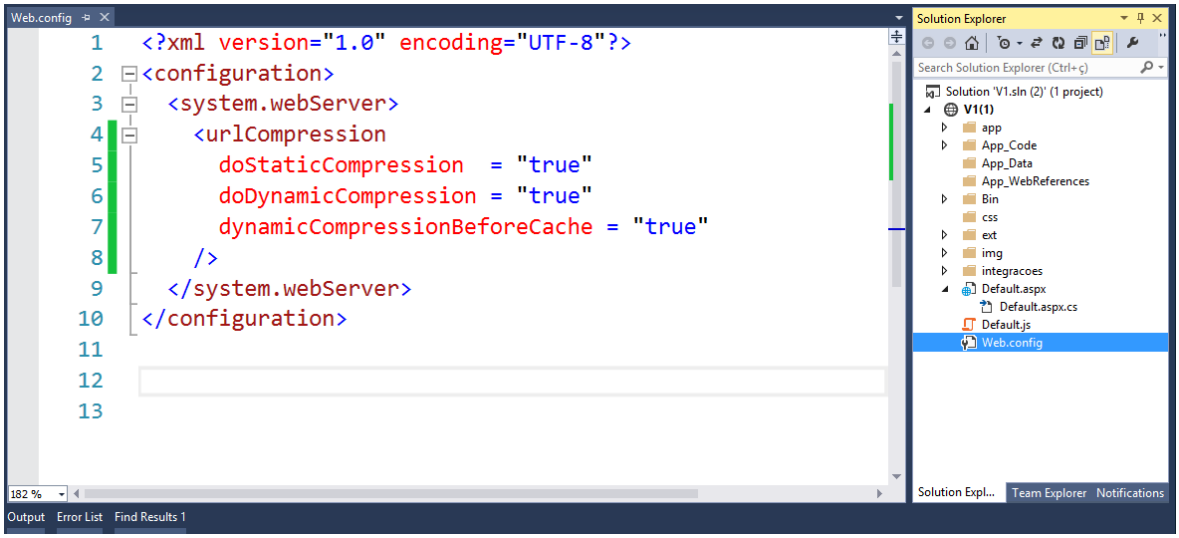
O Gzip é uma das possibilidades mais importantes e significantes para a otimização de um site para dispositivo móveis. A compressão consiste em enviar os códigos de um site compactados, de forma que ocupem menos espaço e sejam mais rápidos nas transmissões pela rede, aumentando a velocidade do site.

A maneira mais comum de enviar um arquivo comprimido é a configuração no servidor *web*, onde está hospedo o site, seja ele *ISS*, *Apache*, entre outros sistemas. O Gzip é capaz de reduzir o tamanho das respostas em cerca de 70% dos sites. O compressor é suportado pela maioria dos navegadores disponíveis atualmente [SOUDERS, 2008].

A técnica de comprimir os dados foi implementada utilizando o Gzip do servidor

citado acima. A configuração da compactação foi realizada pelo próprio código fonte, em um arquivo chamado *web.config*, este arquivo é gerado automaticamente por ser desenvolvido em *.NET*. Abaixo segue na Figura 133 do código que foi inserido para compactar os arquivos do projeto.

Figura 13 - Código fonte do arquivo web.config para compresão.



Fonte: AUTORIA PRÓPRIA

Como pode observar no código fonte do arquivo web.config, foi implementado o método *urlCompression* que define a compactação dos arquivos da aplicação. Este método atualiza a configuração do IIS automaticamente, tirando a necessidade de configurá-lo também para compactar.

3.4.4.2 Web Cache

Os *websites* estão ficando cada vez mais robustos, o que significa a utilização de mais css, *scripts*, imagens, entre outros. Com isso a página pode realizar diversas solicitações HTTP desnecessárias. Ao utilizar o *cache*, poderá reduzir o número e o tamanho das solicitações HTTP, tornando com que as páginas carreguem mais rapidamente e com muito menos requisições.

Um servidor *web* utiliza uma data de expiração no cabeçalho da resposta HTTP para informar aos navegadores quanto tempo aquele componente poderá ser armazenado em *cache*. Evita assim com que o componente não seja solicitado novamente, considerando o prazo de validade estipulado pela aplicação [YAHOO DEVELOPER NETWORK, 2014].

A aplicação desenvolvida utilizou a técnica de *web cache*, para que os arquivos que já

estão em *cache* não sejam transmitidos. Como citado acima, esta técnica utilizará o mesmo painel de configuração do IIS. A configuração do *cache* foi implementada no código fonte, como cita acima em um arquivo chamado *web.config*.

Abaixo segue na Figura 144 do código fonte que foi inserido para colocar os componentes do projeto em *cache*.

Figura 14 - Código fonte do arquivo web.config para a utilização do *web cache*.

Fonte: AUTORIA PRÓPRIA

No código fonte do arquivo web.config, foi implementado o método *staticContent* e *caching*, onde define as configurações dos arquivos que serão armazenados em *cache*. Estes métodos irão definir quais extensões de arquivos que serão armazenadas, e como serão expirados, pela data definida ou por uma notificação de alteração. Este método também atualiza a configuração do IIS automaticamente, sem precisar da intervenção no painel.

3.4.4.3 Minificar o JavaScript e CSS

Os códigos fontes que estão seguindo as recomendações e padrões de documentações, nomes de variáveis, indentação, entre outras, não irão influenciar nas execuções dos *scripts*, pois os navegadores não interpretam essas recomendações e padrões, servirá apenas para o desenvolvimento e a manutenção desses código [SOUDERS, 2008].

Minificar o código JavaScript e CSS, é uma pratica que remove todos os caracteres desnecessários, tais como comentários, linhas e espaços em branco, entre outros. Com isso reduzirá o seu tamanho e consequentemente o tempo de resposta será melhorado.

Uma pesquisa realizada pela Yahoo Developer Network nos dez principais sites norte-americano, apurou que ao minificar um *website*, o seu tamanho poderá ser reduzido em média

a 25% de seu tamanho original. Atualmente existem diversas formas de minificar o código fonte, sendo eles por *softwares*, sites, serviços, entre outros. [YAHOO DEVELOPER NETWORK, 2014].

Para minificar os códigos JavaScript e CSS da aplicação, foram utilizadas duas ferramentas criadas por uma organização *online*. As ferramentas são disponibilizadas gratuitamente e estão disponíveis como serviços ou no próprio *websites*, podendo ser acessadas em dois endereços diferentes: para minificar os códigos JavaScript acesse http://javascript-minifier.com e no caso do CSS acessar http://cssminifier.com. Ambas as ferramentas são muito simples de se utilizar, basta copiar o código fonte que deseja minificar no site e o mesmo retornara-los minificados.

3.4.4.4 Cabeçalho Keep-Alive

Os servidores que utilizam no cabeçalho de resposta HTTP o Keep-Alive, melhora o seu desempenho, pois irá reaproveitar as conexões abertas entre o cliente e o servidor. Ao abrir uma conexão com o servidor, e não tenha sido finalizado, essa conexão é reaproveitada para trafegar outras requisições. Caso não utilizar, o servidor irá abrir uma conexão nova para cada solicitação feita ao mesmo.

Abaixo segue na Figura 155 do código fonte onde foi inserido para utilizar a função de Keep-Alive em suas conexões.

Figura 15 - Código fonte do arquivo web.config para a utilização do *Keep Alive*.

Fonte: AUTORIA PRÓPRIA

No código fonte acima está no arquivo web.config, foi implementado utilizando o

método *httProtocol* e definindo pelo *allowKeepAlive*, nela poderá definir verdadeiro ou falso para que utilize a configuração ou não. Este método atualizará automaticamente as configurações do ISS, sem precisar configurá-lo também.

4 RESULTADOS E ANÁLISES

Para mostrar as vantagens ou desvantagens da utilização de técnicas e boas práticas para dispositivos móveis, testes foram realizados. Os testes foram realizados utilizando um *smartphone* com acesso ao servidor e com ferramentas que analisam o desempenho da aplicação *web* que foi proposta.

Neste capítulo será apresentada a análise dos dados coletados e seus resultados.

4.1 Cenário com Cache e sem Cache

Na aplicação desenvolvida foi implementado a técnica de *web cache* com o intuito de diminuir as transferências entre o cliente e o servidor. Como na inicialização da aplicação *web* carrega tudo de um vez, foi implementado no próprio código fonte do arquivo *web.config,* as configurações de como será utilizado o *cache* do cliente.

Após ter aplicado as configurações foram realizados alguns teste, utilizando a própria ferramenta de desenvolvedor disponível no navegador Google Chrome. Foram realizados dois testes, utilizando e sem utilizar a técnica de *cache*, com isso pode-se fazer um comparativo do uso desta técnica.

Como a aplicação está utilizando um *framework*, pode-se observar que a maior carga das requisições está logo no início, por terem que ser carregados as classes do JavaScript, CSS, entre outras. Com o intuito de realizar os teste mais eficiente, foram testados com três tipos de conexões, GPRS a 50 Kbps, 3G a 750 Kbps e por último na Wi-Fi.

Na Tabela 4, foram realizados algumas comparações relevantes, quando se trata de aplicações *web* desenvolvida para dispositivos móveis utilizando a rede GPRS.

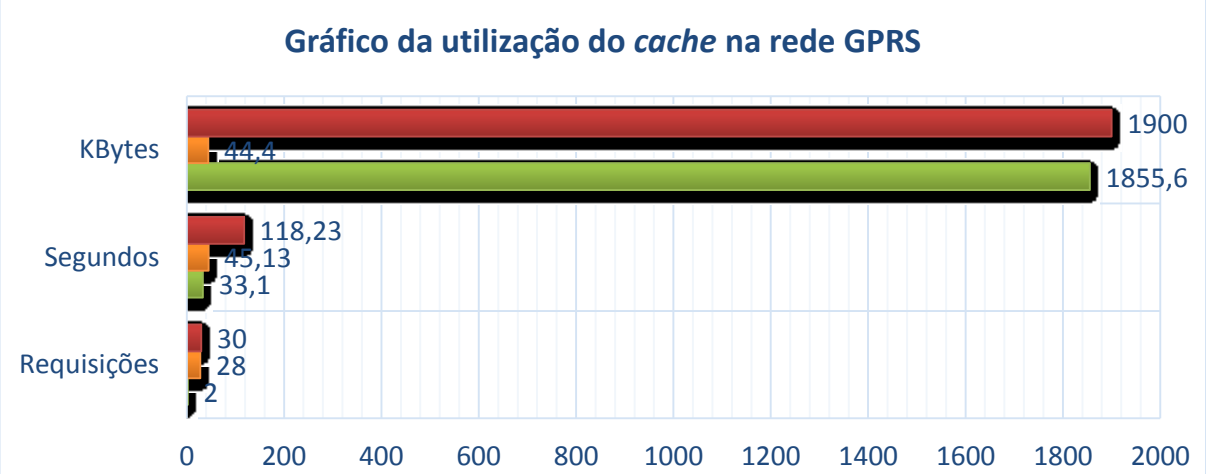
Tabela 4 - Comparação da utilização do *cache* na aplicação utilizando a rede GPRS.

	Sem cache			**Com cache**		
Telas	**Requisições**	**Segundos**	**KBytes**	**Requisições**	**Segundos**	**KBytes**
Login	30	1.18,23	1.900	28	45,13	44,4

Fonte: AUTORIA PRÓPRIA

Na Figura 16 podemos observar um gráfico que está representando os resultados dos testes quando utilizado a técnica de cache na rede GPRS.

Figura 16 - Gráfico da utilização do cache na rede GPRS



	Requisições	Segundos	KBytes
Sem cache	30	118,23	1900
Com cache	28	45,13	44,4
Redução	2	33,1	1855,6

Fonte: AUTORIA PRÓPRIA

Utilizando a rede GPRS podemos observar que o tempo foi extremamente alto por conta da baixa capacidade de transferência de informações do dispositivo móvel. Já quando utilizamos a rede 3G que é mais comum entre os dispositivos móveis, notamos uma diminuição desse tempo significante, com uma redução de 77,88%. Na Tabela 5 podemos observar essa diferença na redução do tempo entre a utilização das redes GPRS e 3G.

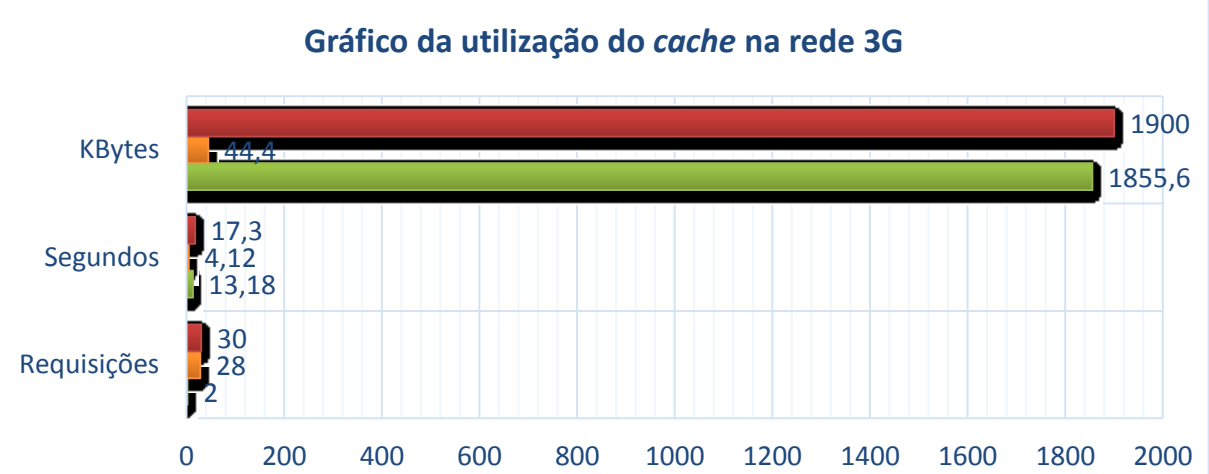
Tabela 5 - Comparação da utilização do cache na aplicação utilizando a rede 3G.

	Sem cache			**Com cache**		
Telas	**Requisições**	**Segundos**	**KBytes**	**Requisições**	**Segundos**	**KBytes**
Login	30	17,30	1.900	28	4,12	44,4

Fonte: AUTORIA PRÓPRIA

Na Figura 17 podemos observar um gráfico que está representando os resultados dos testes quando utilizado a técnica de cache na rede 3G.

Figura 17 - Gráfico da utilização do cache na rede 3G



	Requisições	Segundos	KBytes
Sem cache	30	17,3	1900
Com cache	28	4,12	44,4
Redução	2	13,18	1855,6

Fonte: AUTORIA PRÓPRIA

A rede Wi-Fi não poderá ser comparada com as redes anteriores GPRS e 3G, pois depende muito da conexão que é disponibilizada na LAN. Na Tabela 6 foram realizados as comparações utilizando a conexão Wi-Fi.

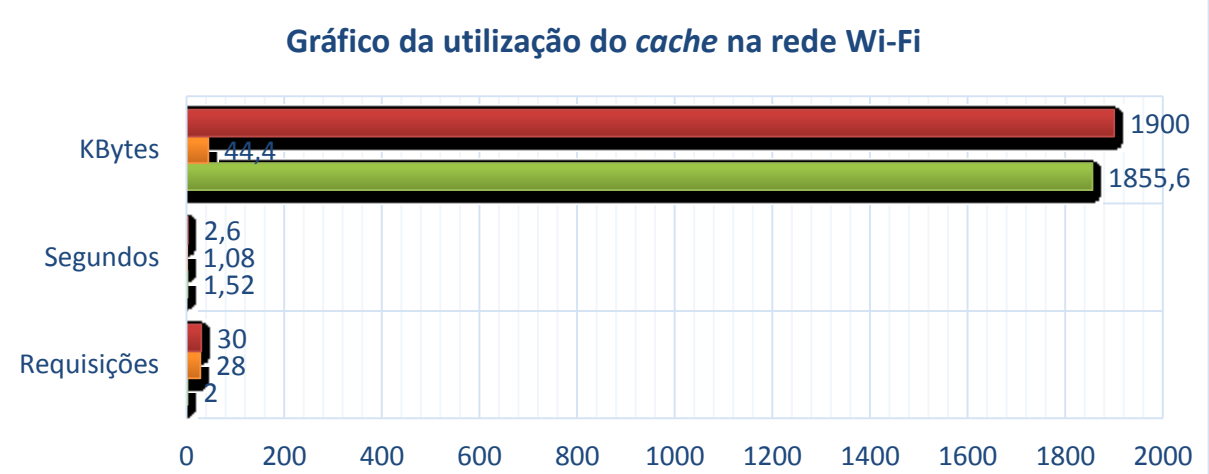
Tabela 6 - Comparação da utilização do cache na aplicação utilizando a rede Wi-Fi.

	Sem cache			**Com cache**		
Telas	**Requisições**	**Segundos**	**KBytes**	**Requisições**	**Segundos**	**KBytes**
Login	30	2,60	1.900	28	1,08	44,4

Fonte: AUTORIA PRÓPRIA

Na Figura 18 podemos observar um gráfico que está representando os resultados dos testes quando utilizado a técnica de cache na rede Wi-Fi.

Figura 18 - Gráfico da utilização do cache na rede Wi-Fi



	Requisições	Segundos	KBytes
Sem cache	30	2,6	1900
Com cache	28	1,08	44,4
Redução	2	1,52	1855,6

Fonte: AUTORIA PRÓPRIA

Como podemos observar a rede móvel tem uma grande influência no carregamento das aplicações *web*. Na tela de Login por ser a primeira quando acessamos a aplicação, tornou-se perceptível a importância da utilização dessa técnica. Ela teve uma redução de 6,66% das quantidades de requisições e 97,66% na redução do tamanho dos arquivos transferidos, tornando surpreendente a redução de 1.855,6 KBytes do seu tamanho, utilizando apenas do *Web Cache*. As demais telas não houveram requisições, por serem carregadas ao acessar a aplicação *web* móvel.

Com esta redução significante, podemos observar que é muito importante a utilização desta técnica, para evitar o uso desnecessário dos arquivos transferidos, e diminuindo o custo de utilização da bateria e da rede móvel. Com isso torna-se a experiência de uso do usuário mais agradável e eficiente, pois aplicação *web* será carregada mais rapidamente e com menos custos.

4.2 Cenário com Gzip e sem Gzip

A compreensão de dados é uma excelente técnica de otimização para aplicações *web* ou *websites*, além de ser fácil e pratico a sua implementação. Porém não são todos os arquivos que podem apresentar uma compactação adequada. As imagens e os vídeos não são recomendados compactar.

No protótipo desenvolvido foi implementado a compressão *gzip* no arquivo *web.config*, para que possa ser realizados testes na redução de seu tamanho. Foram realizados testes utilizando a ferramenta do desenvolvedor no navegador Google Chrome, nele foram realizados dois testes, com e sem a compactação do *gzip*. Como já citado acima, estamos utilizando um *framework*, onde torna-se o primeiro acesso a maior carga.

Na Tabela 7 foram realizado dois testes para comparar os benefícios da utilização da técnica de compressão gzip.

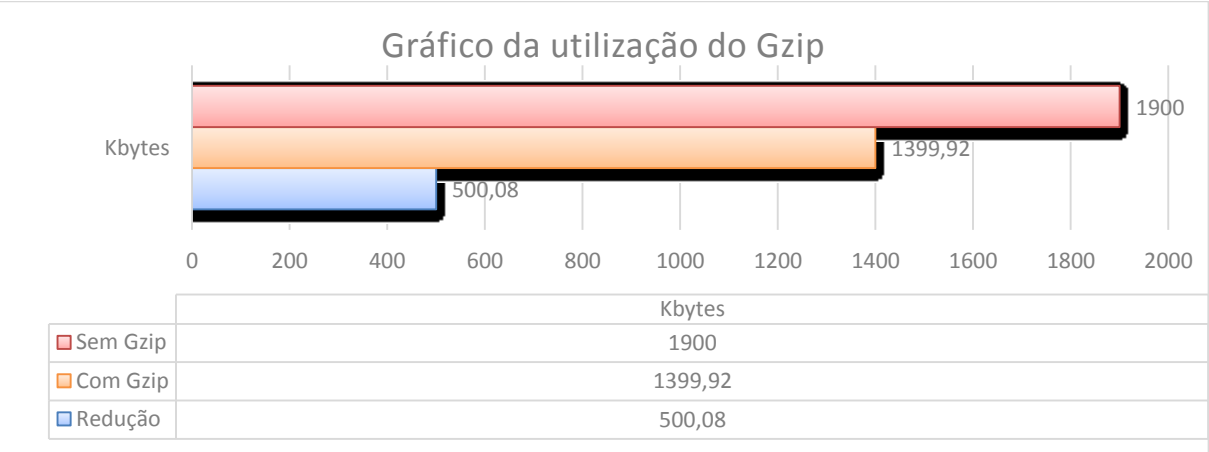
Tabela 7 - Comparação entre a utilização da técnica Gzip.

Telas	**Sem Gzip KBytes**	**Com Gzip KBytes**	**Redução KBytes**	**Redução Porcentagem**
Login	1.9	1.399,92	500,08	26,32 %

Fonte: AUTORIA PRÓPRIA

Na Figura 19 podemos observar um gráfico que está representando os resultados dos testes quando utilizado a técnica de compressão gzip.

Figura 19 - Gráfico da utilização do Gzip



Fonte: AUTORIA PRÓPRIA

Como a tela de Login é a primeira quando acessamos a aplicação, podemos observar uma redução significante em seu tamanho, após aplicar a compactação. Aplicando essa técnica, podemos observar que reduziu um quarto do seu tamanho, sendo 26,32% e 500,08 KBytes, economizando assim na transferência entre o cliente e o servidor.

Um outro ponto importante a ser observado, é que a compactação dos arquivos foram aplicadas apenas nos arquivos que estão sendo utilizando no protótipo como extensão javascript, css e aspx. Mas caso tivesse outras extensões também poderiam ser compactadas, as imagens, vídeos e arquivos menores que 1 KBytes não são recomendados utilizar essa técnica, pois não terá um efeito significativo. Com essa redução podemos diminuir a transferência entre

o cliente e o servidor cerca de 30%, simplesmente aplicando a técnica de compactação dos arquivos a serem transferidos.

4.3 Cenário da minificar do JavaScript e do CSS

No protótipo desenvolvido foi aplicado a técnica de minificar o código, onde remove os caracteres desnecessários e com isso o tamanho reduz consideravelmente, melhorando o desempenho do tempo de resposta.

A técnica foi aplicada apenas nos arquivos JavaScript e CSS, onde os mesmos tiveram uma boa redução em seu tamanho. Os testes foram realizados utilizando a própria ferramenta do desenvolvedor disponível nos navegadores do Google Chrome.

Na Tabela 8 é mostrado os resultados dos testes realizados onde teve ganho de desempenho que tivemos quando os processos de minificação são aplicados aos arquivos JavaScript.

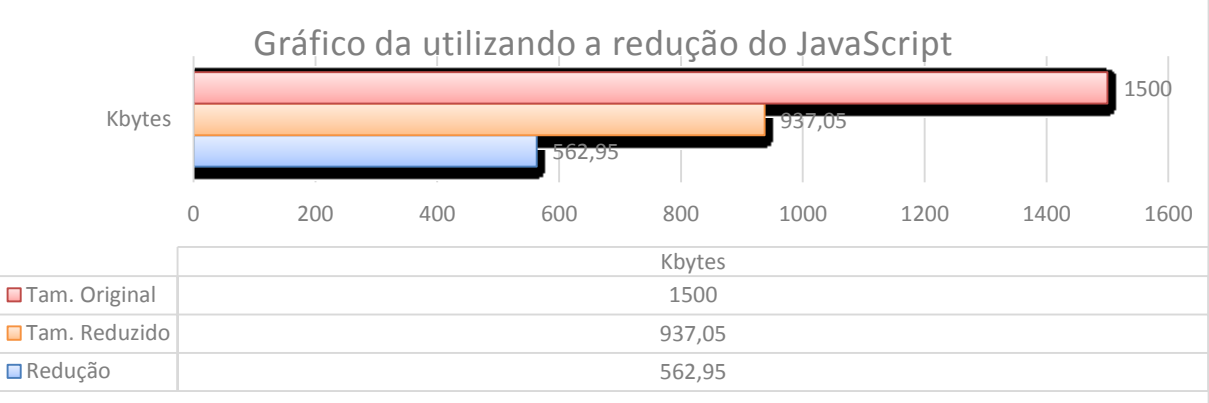
Tabela 8 - Comparação entre os arquivos JavaScript minificados.

Telas	**Tamanho original**	**Tamanho reduzido**	**Redução**	
Login	1.5 KBytes	937,05 KBytes	562, 95 KBytes	37,53 %

Fonte: AUTORIA PRÓPRIA

Na Figura 20 podemos observar um gráfico que está representando os resultados dos testes quando utilizado a técnica de minificar o JavaScript.

Figura 20 - Gráfico da utilizando a redução do JavaScript



Fonte: AUTORIA PRÓPRIA

Como já citado acima a tela de Login por ser a primeira quando acessamos o protótipo, é onde tem a maior carga e onde podemos observar que teve uma redução significante no seu tamanho. Após aplicado a técnica nos arquivos JavaScript, eles tiveram uma redução de um

terço do seu tamanho representando 562,95 KBytes e 37,53% do seu tamanho original.

Na Tabela 9 veremos os resultados quando aplicamos a técnica nos arquivos CSS, onde os mesmo também tiveram uma redução interessante.

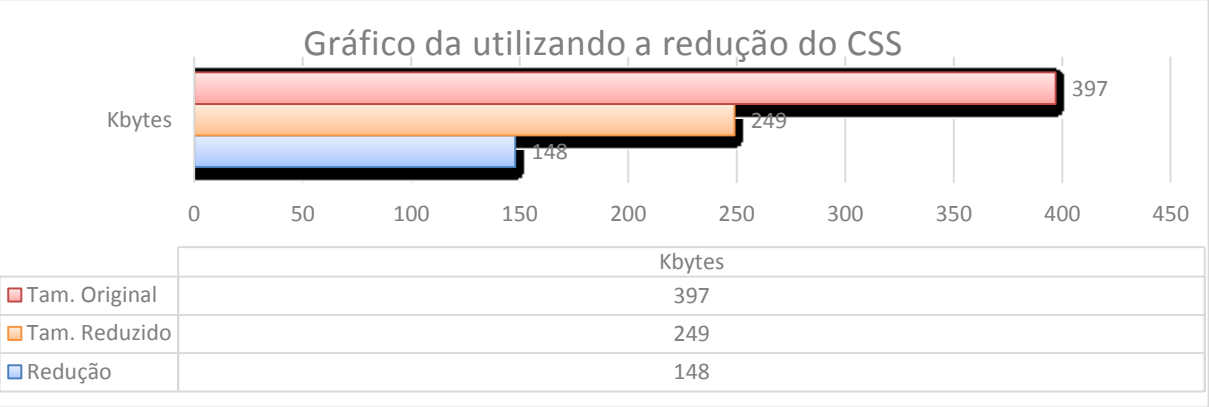
Tabela 9 - Comparação entre os arquivos CSS minificados.

Telas	**Tamanho original**	**Tamanho reduzido**	**Redução**	
Login	397 KBytes	249 KBytes	148 KBytes	37,27 %

Fonte: AUTORIA PRÓPRIA

Na Figura 21 podemos observar um gráfico que está representando os resultados dos testes quando utilizado a técnica de minificar o CSS.

Figura 21 - Gráfico da utilizando a redução do CSS



Fonte: AUTORIA PRÓPRIA

As economia dos arquivos CSS geralmente são menores em comparação ao arquivos JavaScript, pois nos CSS tem menos comentários e menos espaços em brancos. No caso do protótipo os arquivos tiveram uma redução significante por estar utilizando um framework, onde o mesmo dispõe de uma documentação bem detalhada, tornando os arquivos maiores.

Os arquivos CSS do protótipo tiveram uma redução de 148 KBytes representando 37,27% do seu tamanho original.

Recomenda-se que quando pretendemos reduzir o tamanho dos arquivos CSS é diminuir a quantidade de classes, e a utilização de abreviações, remoções de *strings* desnecessárias, como ao definir um tamanho de um componente, ao invés de utilizar “10px” utilizar apenas “10”, onde o efeito se torna o mesmo.

4.4 Cenário de resoluções diferentes de Tela

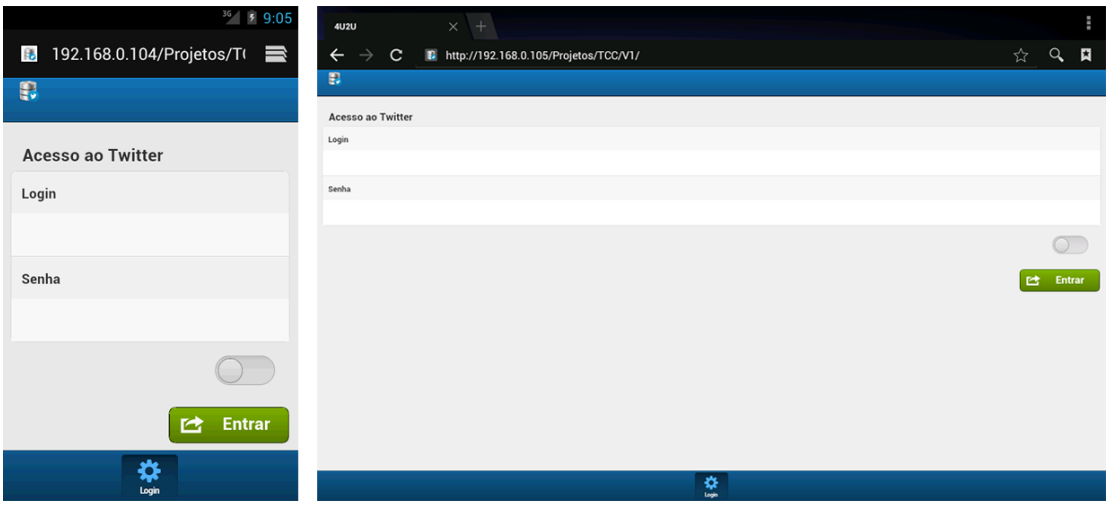
Um dos maiores desafios no desenvolvimento voltado para dispositivos móveis, é

fazer com que as telas que são implementadas se enquadrem adequadamente as diversas resoluções dos aparelhos. Visando atender essa necessidade, o framework disponibiliza o designer responsivo, onde enquadra todos os componentes da aplicação na resolução da tela que está sendo utilizada no momento.

Buscando realizar os testes adequados, no protótipo desenvolvido foram realizados dois testes em dispositivos de tamanhos diferentes. Foi utilizado um simulador para realizar os testes, sendo um *smartphone* Nexus One (3.7, 480x800 *hdpi*) e outro utilizando um *tablet* WXGA (10.1, 1280x800 *mdpi*), testando os diferentes tipos de telas implementadas com vários componentes, tais como listas, campos, botões, imagens, textos, entre outros.

Na Figura 2222 é ilustrado a tela de login onde está sendo feito uma comparação no enquadramento dos componentes entre dois tamanhos.

Figura 22 - Tela de login (A) resolução de 480x800 e (B) 1280x800.

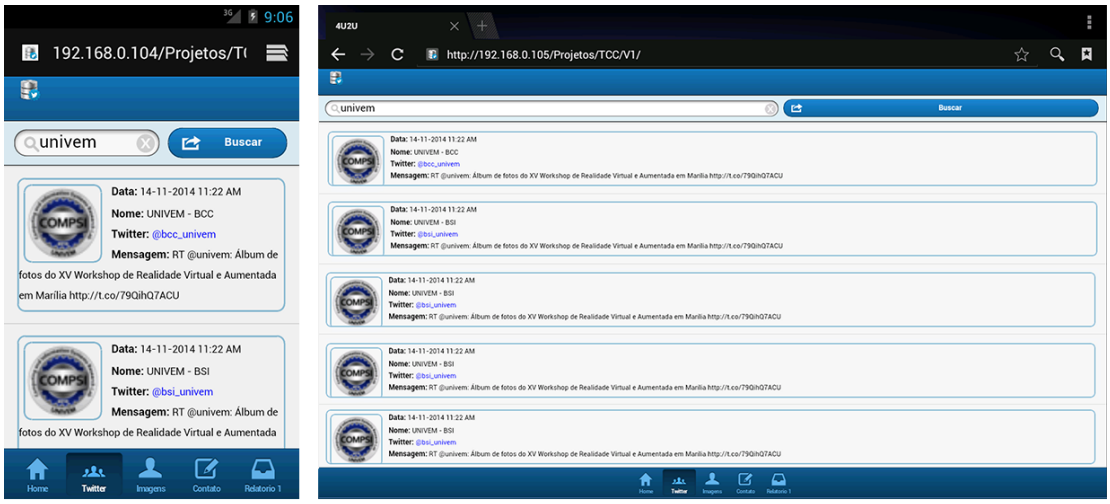


Fonte: AUTORIA PRÓPRIA

Na tela de login podemos observar que temos dois campos, um sendo de texto e outro de senha, também tem uma opção de ficar autenticado no sistema e um botão para entrar. Todos os componentes se adequaram perfeitamente ao tamanho da resolução.

Na Figura 2323 é ilustrado a tela de Twitter, onde aparece uma lista com o assunto pesquisado no Twitter.

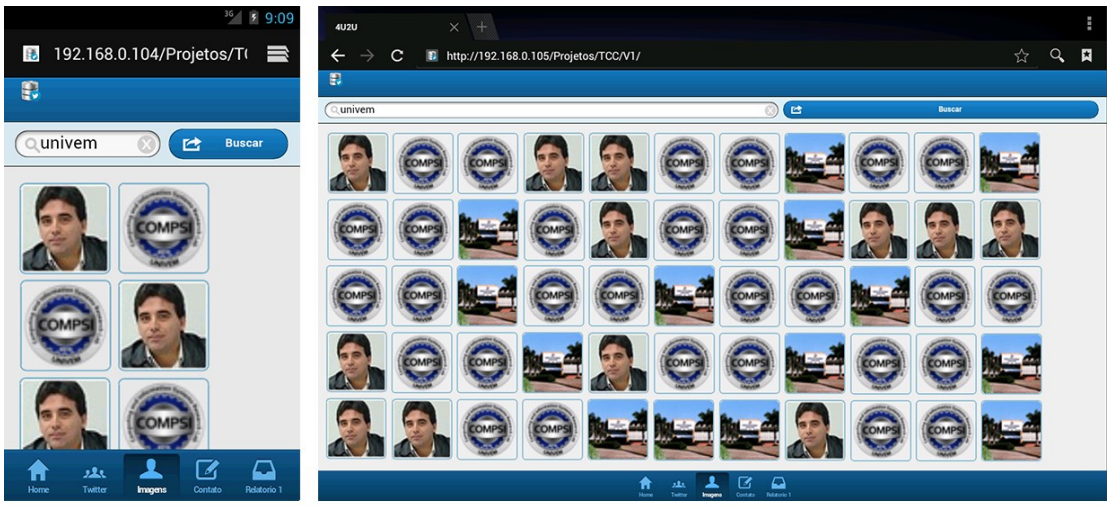
Figura 23 - Tela do Twitter (A) resolução de 480x800 e (B) 1280x800.



Fonte: AUTORIA PRÓPRIA

Na tela do Twitter tem alguns componentes, como o campos de pesquisa, o botão de buscar e a lista que representa o retorno do assunto pesquisado. Na lista, temos textos, imagens e *hiperlink*, onde os mesmos podem variar de tamanhos, pela quantidade de textos. Como podemos observar todos os componentes se enquadraram adequadamente a resolução do dispositivo móvel. Na Figura 2424 é ilustrado uma lista apenas com as imagens pesquisadas ao devido assunto no Twitter.

Figura 24 - Tela de Imagens (A) resolução de 480x800 e (B) 1280x800.

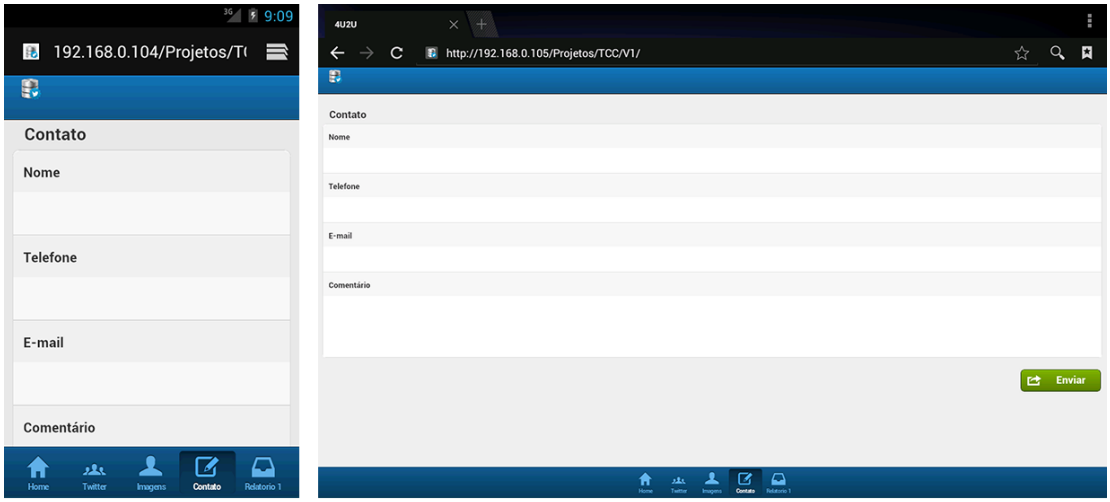


Fonte: AUTORIA PRÓPRIA

Como podemos observar essa tela é bem parecida com a anterior. Temos alguns componentes como campo para pesquisa, o botão para efetuar a busca e uma lista. Entretanto na lista aparece apenas as imagens sem os textos e os *hiperlinks*. Todos os componentes se enquadraram adequadamente a resolução.

Na Figura 255 é ilustrado um formulário para contato, com diversos campos para preenchimentos.

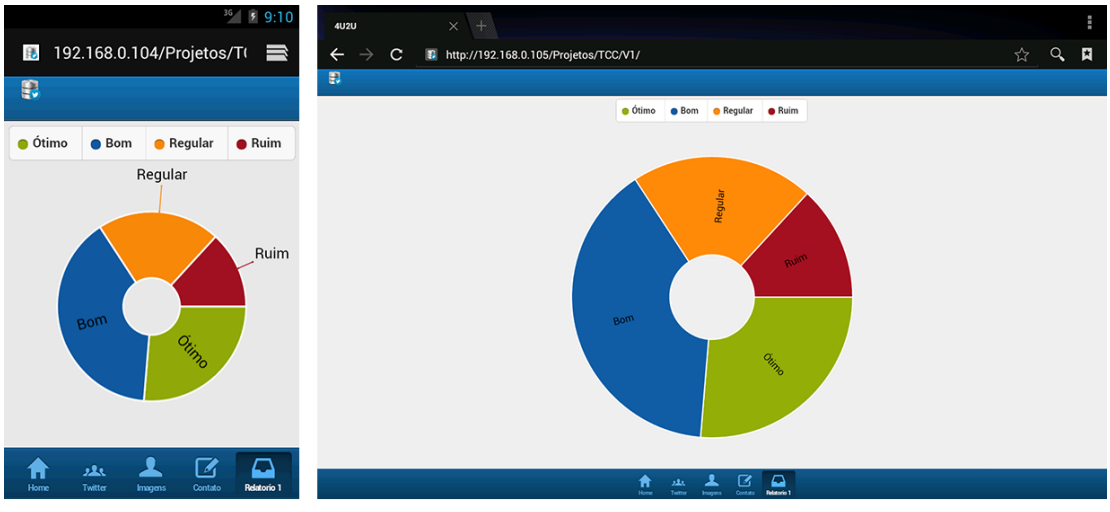
Figura 25 - Tela de Contato (A) resolução de 480x800 e (B) 1280x800.



Fonte: AUTORIA PRÓPRIA

Ao observar a tela de Contato, conseguimos notar que existem diversos componentes com entrada de dados como nome, telefone, e-mail e comentário. Esses campos são personalizados conforme suas características. Quando iniciamos a introdução do textos no campo, o teclado aparecerá conforme a necessidade do mesmo, sendo numérico, texto ou endereço de e-mails. Na Figura 266 é ilustrado uma tela com relatório gráfico em formato de pizza, onde temos quatro divisões.

Figura 26 - Tela de Relatório (A) resolução de 480x800 e (B) 1280x800.



Fonte: AUTORIA PRÓPRIA

A tela de relatório tem um gráfico no formato de pizza, onde foram divididos em quatro categorias, sendo elas: ótimo, bom, regular e ruim. Podemos observar que o gráfico diminuiu o

seu tamanho para se enquadrar adequadamente às duas resoluções de tela.

O protótipo desenvolvido buscou atender as necessidades de se enquadrar adequadamente a diversas resoluções dos dispositivos móveis. Foram realizados os testes em cima de duas resoluções 480x800 e 1280x800 visando atender as duas pontas, uma resolução menor e uma maior.

4.5 Análises e resultados

Atualmente existem diversas empresas que disponibilizam ferramentas automatizadas para realizar análises do carregamento dos sites, sistemas, entre outros. Essas ferramentas são utilizadas com frequência por quem busca um melhor desempenho, oferecendo dicas sobre o que e como melhorar suas aplicações *web* ou sites.

Nesta seção serão abordadas as duas ferramentas mais utilizadas no mercado, fornecidas por duas empresas que se tornaram referência no desenvolvimento *web*. A primeira é a PageSpeed que foi desenvolvida pela Google e a segunda é a YSlow criada pela Yahoo!

4.5.1 YSLOW

A ferramenta YSlow tem como principal característica, analisar o desempenho das páginas *web*, examinado todos os componentes da página, incluindo os criados dinamicamente usados pelo *JavaScript*. Além de examinar ele oferece sugestões de como pode ser melhorada a página [YSLOW, 2014].

A equipe de desenvolvimento da Yahoo! identificou 34 critérios que podem afetar notavelmente o desempenho das páginas. Visando diminuir os problemas a equipe criou a ferramenta *YSlow*, que se baseia em 23 dos 34 critérios. Esses critérios são listados em ordem de importância e eficácia [YSLOW, 2014].

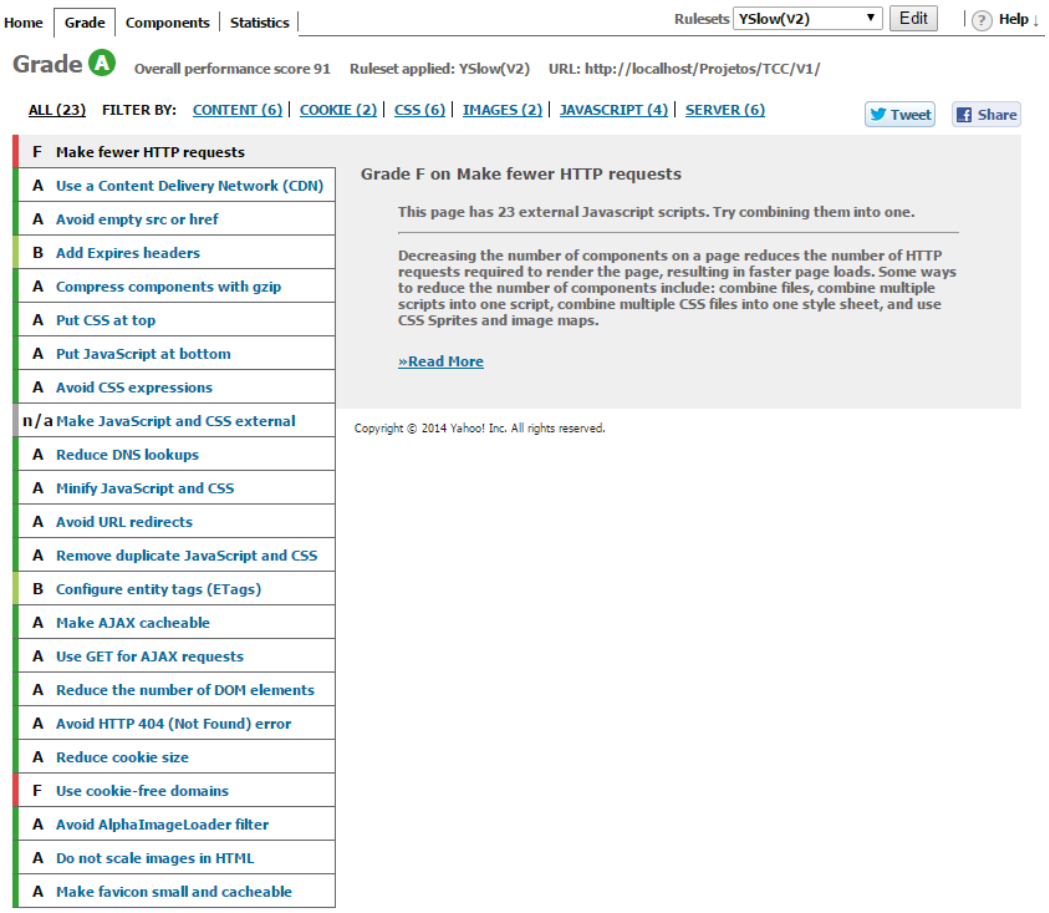
O método de análise é baseado em notas que vão da letra “A” ao “F”, e também em pontuações que vão do “0” ao “100”, conforme a avaliação dos 23 critérios. Em uma pesquisas realizada pela empresa Yahoo!, identificou que o tempo de resposta de uma página pode ser melhorado entre 25% a 50%, seguindo os critérios [YSLOW, 2014].

YSlow pode ser instalado em alguns navegadores, como o Firefox, Chrome, Opera, Safari, entre outros. Pode-se também realizar testes em dispositivos móveis, com a utilização do Bookmarklet integrando com o PhantomJS [YSLOW, 2014].

Resultados

Utilizamos o sistema YSlow no protótipo desenvolvido afim de analisarmos a nota após aplicados as técnicas de otimização de aplicações *web*. Na Figura 277 é mostrado a pontuação obtida nos testes realizados pela ferramenta.

Figura 27 - Resultado obtido pela ferramenta YSlow.



Fonte: AUTORIA PRÓPRIA

A nota que a ferramenta apresentou sobre o protótipo, foi classe A que é a maior nota entre as letras, já na pontuação numérica obteve 91. As pontuações não obtidas referem-se a utilização de arquivos do tipo javascript que estão separados por telas, divido a arquitetura inicial do protótipo, a recomendação da ferramenta é com que se reduza a quantidade de arquivos para que o carregamento seja mais rápido.

4.5.2 PAGESPEED

O PageSpeed é uma ferramenta desenvolvida pela Google, que segue a mesma linha da ferramenta oferecida pela Yahoo!, avaliando o desempenho da página *web*. Porém o Google foi mais além em sua ferramenta, adicionando alguns critérios que ao analisarem a aplicação

ou site, servirão tanto para os computadores de mesa, como para os dispositivos móveis. E utiliza apenas os critérios que são diretamente ligados as página, independentemente da qualidade na conexão de rede [PAGESPEED, 2014].

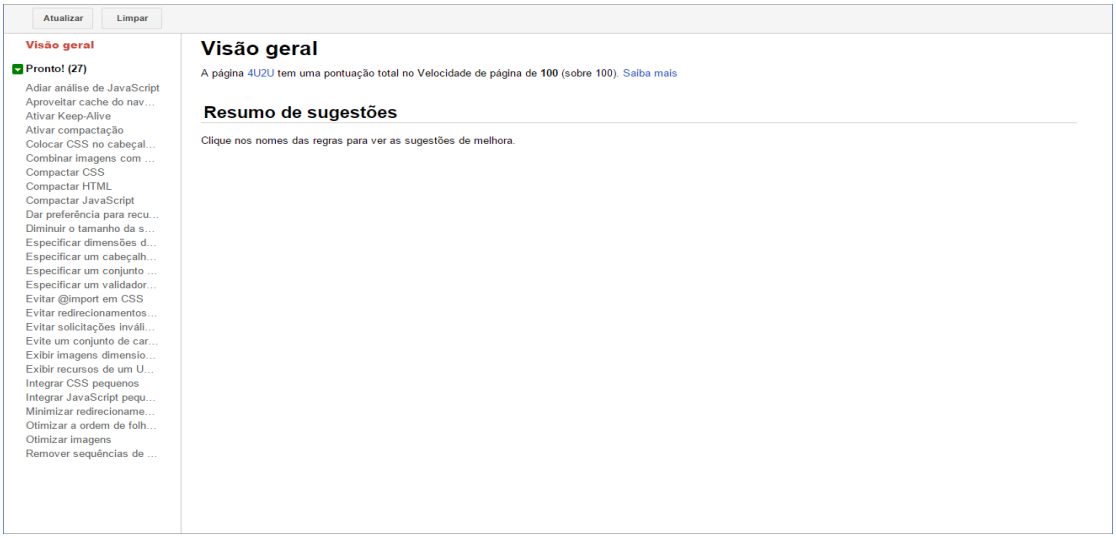
A ferramenta também utiliza o método de pontuação, para avaliar o desempenho das páginas que podem variar entre “0” a “100”. Pela ferramenta estar em constante atualizações, as pontuações poderão ser modificadas à medida que são adicionados novos critérios ou melhoria em suas análises [PAGESPEED, 2014].

Inicialmente a PageSpeed foi desenvolvida como uma extensão para o navegador Chrome, entretanto a Google percebeu uma necessidade de disponibilizar para outros navegadores, abordando um maior número de público. Atualmente a PageSpeed fornece uma análise semelhante *online*, e com extensões a os navegadores Google Chrome e Firefox [PAGESPEED, 2014].

Resultados

Utilizamos também a ferramenta PageSpeed para avaliarmos a aplicação desenvolvida. Na Figura 288 é mostrado o resultado obtido pela ferramenta desenvolvido pela Google.

Figura 28 - Resultado obtido pela ferramenta PageSpeed.



Fonte: AUTORIA PRÓPRIA

A ferramenta PageSpeed também analisou o protótipo desenvolvido, onde obteve a pontuação “100” que é a máxima, na velocidade da página. A análise contemplo 27 critérios que indica se aplicação está adequada, tanto para computadores de mesa como para dispositivos móveis, independente da conexão da rede.

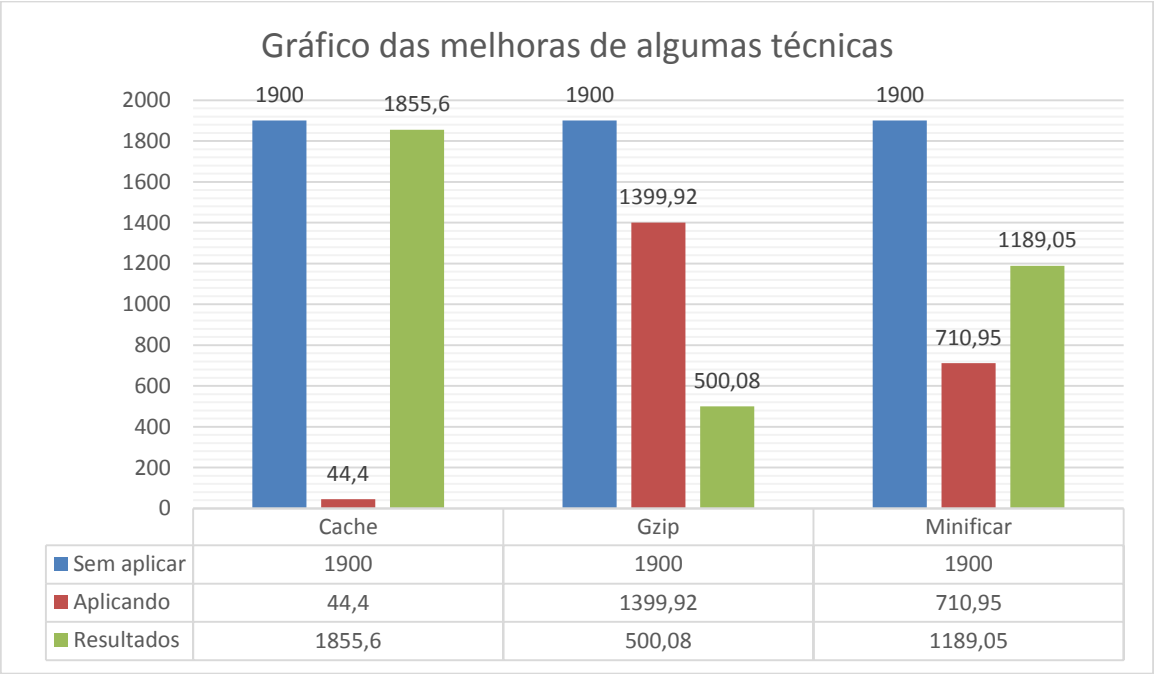
4.6 Considerações finais

Podemos observar que diversas técnicas de otimizações podem ser utilizadas nas aplicações, melhorando consideravelmente a performance de uso da mesma. Antes de aplicar qualquer técnica é recomendado que se façam testes e analises para que seu uso não se torne um equívoco, prejudicando assim a aplicação *web* móvel.

Neste capitulo demonstramos a aplicação de algumas técnicas tanto no processo de desenvolvimento da aplicação, como no próprio servidor, em busca de melhorar a performance de uso do protótipo o máximo possível.

Na Figura 29 ilustra a um comparativos aplicando as algumas técnicas de otimizações.

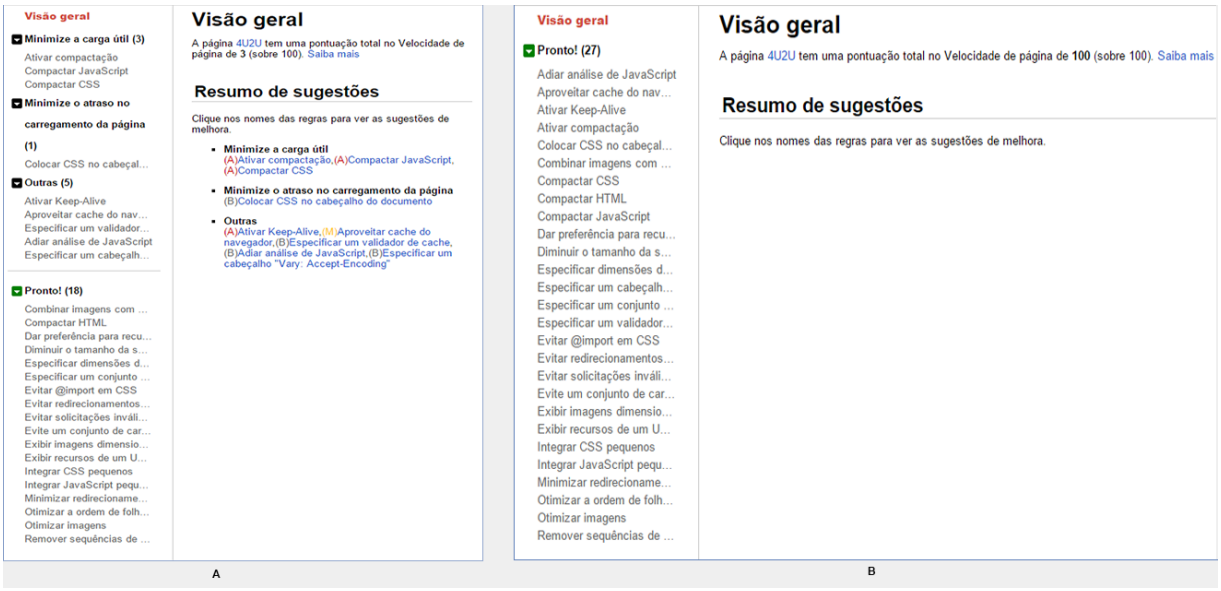
Figura 29 - Gráfico de algumas técnicas



Fonte: AUTORIA PRÓPRIA

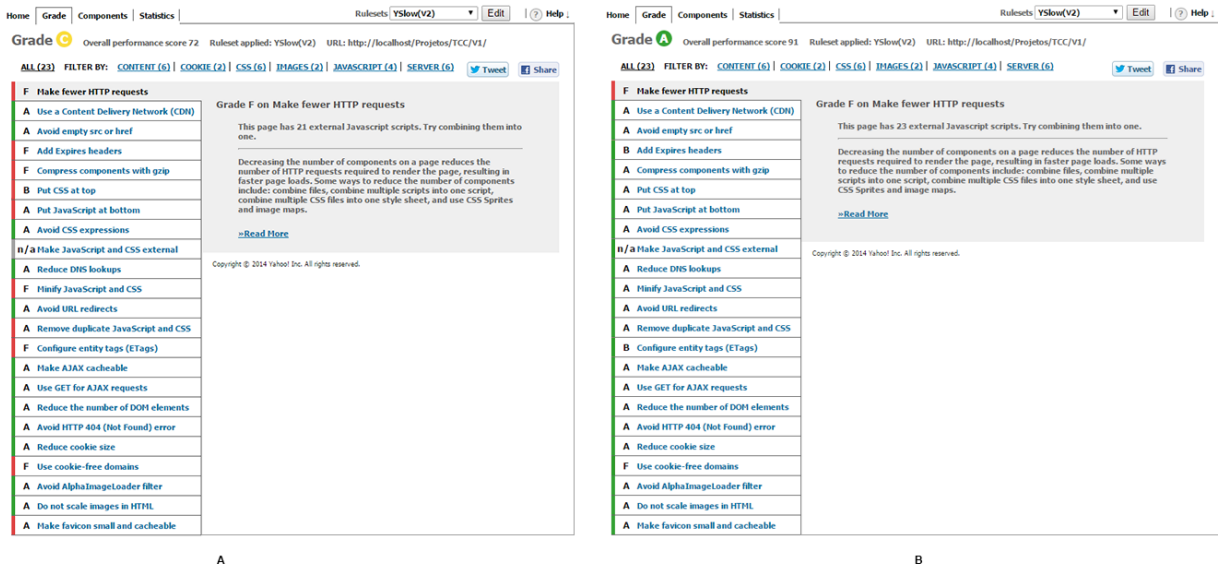
Foram ilustrados também os impactos de algumas técnicas aplicadas, utilizando algumas ferramentas para mostrar a melhoria da performance. As Figuras 30 e 31 é ilustrado a diferença nas ferramentas quando aplicado as técnicas.

Figura 30 - Comparação entre as otimização na ferramenta PageSpeed, (A) Não está otimizado (B) Está otimizado.



Fonte: AUTORIA PRÓPRIA

Figura 31 - Comparação entre as otimização na ferramenta YSlow, (A) Não está otimizado (B) Está otimizado.



Fonte: AUTORIA PRÓPRIA

Como podemos observar na imagem, tivemos uma melhoria considerável quando aplicamos as técnicas de otimizações, tanto no lado do cliente como no lado do servidor. Foram utilizadas duas ferramentas diferentes para mostrar os impactos que tiveram na aplicação.

Vimos que aplicando algumas técnicas simples podemos melhorar consideravelmente a satisfação de uso do usuário final em relação a aplicação desenvolvida para dispositivos móveis.

5 CONCLUSÃO

À medida que cresce o mercado de dispositivos móveis, aumenta também a variedade dos aplicativos que são desenvolvidos. Empresas de desenvolvimento estão criando cada vez mais aplicações que necessitam de trocas de informações pelas redes móveis. Entretanto, apesar do avanço tecnológico na área móvel, as transferências de informações via rede móvel se tornaram um fator preocupante devido a suas limitações.

No presente trabalho, foi iniciado um estudo sobre as técnicas de otimizações voltadas para dispositivo móveis que tem como principal objetivo analisar o que o mercado, fóruns e instituições de pesquisas podem oferecer para ganharmos qualidade nos apps.

O protótipo implementado nesse trabalho teve como principal finalidade, aplicar algumas das técnicas recomentadas no documento do grupo W3C, esse documento busca criar um padrão internacional, garantindo um bom desenvolvimento na evolução das aplicações voltadas para dispositivos móveis.

Ainda na fase de desenvolvimento, também foi utilizado o *framework* Sencha Touch, onde busca aproximar a aplicação *web* com a nativa. Após o termino da implementação do protótipo, foram realizados testes de performance com as ferramentas PageSpeed e YSlow, onde pontua a performance da aplicação.

Com base nos resultados dos testes e no documento da melhores práticas, verificou-se a importância de aplicar algumas otimizações na aplicação. Como podemos observar nos resultados obtidos pelas ferramentas, o protótipo teve uma avaliação com a pontuação alta podendo chegar a pontuação máxima, apenas aplicando algumas técnicas de otimização.

Lembrando que no processo de implementação algumas medidas terão de ser bem analisadas antes da criação de uma aplicação, pois as necessidades influenciam diretamente na escolha da arquitetura, tecnologias, ferramentas, banco de dados, entre outros.

A principal conclusão do projeto que podemos ressaltar, é mensurar o quando que as técnicas que foram aplicadas puderam oferecer de ganho na eficiência e na performance da aplicação *web*.

5.1 Trabalhos Futuros

Como continuidade das pesquisas realizadas no presente trabalho, é possível seguir algumas direções que poderão acrescentar mais valores nas pesquisas de otimização de aplicações *web* para dispositivos móveis.

O documento dos grupo W3C dispõe de diversas melhorias que podem ser utilizadas no desenvolvimento de aplicações, podendo ser implementadas com a utilização de outros *frameworks* que estão disponíveis no mercado.

Uma implementação interessante que poderá ser realizada, é o carregamento por *http push* mais conhecido por *websockets*, essa otimização visa diminuir as requisições desnecessárias das aplicações *web* ao servidor.

Ao meu ver todas as pesquisas que se referem a análise de boas práticas no desenvolvimento *web* voltado para dispositivos móveis, são válidas como trabalhos futuros, pois as tecnologias estão em constante melhorias.

REFERÊNCIAS

ERICSSON. **Ericsson Mobility Report**. Disponível em: <http://hugin.info/1061/r/1790097/615436.pdf>. Acessado em junho de 2014.

GOOGLE SERVICE. **Nosso Planeta Mobile Brasil: Como entender o usuário de celular**. Disponível em <http://services.google.com/fh/files/misc/omp-2013-br-local.pdf>. Acessado em novembro de 2014.

APPLE. **App Store Downloads Top 100 Million Worldwide.** Disponível em: <http://www.apple.com/pr/library/2008/09/09App-Store-Downloads-Top-100-Million-Worldwide.html>. Acessado em maio de 2014.

GOOGLE. **Introducing Google Play All Your**. Disponível em: <http://googleblog.blogspot.com.br/2012/03/introducing-google-play-all-your.html>. Acessado em maio de 2014.

MSDN. **Desenvolver aplicativos universais para Windows**. Disponível em: <http://msdn.microsoft.com/pt-br/windows/apps/br229519>. Acessado em maio de 2014.

IBM. **Dentro do Desenvolvimento de Aplicativo Móvel.** Disponível em: <http://www-03.ibm.com/software/products/Pt/category/Mobile-Application-Development>. Acessado em abril de 2014.

PHONEGAP. **PhoneGap**. Disponível em: <http://phonegap.com>. Acessado em abril de 2014

KIVY. **Kivy: cross-plataform framework**. Disponível em: <http://kivy.org>. Acessado em abril de 2014.

TITANIUM. **Titanium Mobile Application Development**. Disponível em: <http://www.appcelerator.com/Titanium>. Acessado em abril de 2014

OEHLMAN, Damon; BLANC, Sébastien. **Pro Android Web Apps. Develope for Android using HTML5, CSS3, & Java script**, Apress, 2011.

NOKIA. **Arquitetura de aplicações multiplataforma para dispositivos móveis**. Disponível em: < http://developer.nokia.com/community/wiki/Arquitetura_de_aplicações_multiplataforma_para_dispositivos_móveis>. Acessado em maio de 2014.

SENCHA. **HTML5 App Development for Desktop and Mobile. JavaScript Frameworks and Dev Tools from Sencha**. Disponível em: <http://www.sencha.com>. Acessado em abril de 2014.

JQUERY. **jQuery Mobile**. Disponível em: <http://jquerymobile.com>. Acessado em abril de 2014.

DOJO. **Unbeatable JavaScript Tools - The Dojo Toolkit**. Disponível em: <http://dojotoolkit.org>. Acessado em abril de 2014.

APPMOBI. **appMobi - Cross Platform Push Messaging, In App Purchasing & Moreappmobi**. Disponível em: <http://www.appmobi.com>. Acessado em abril de 2014.

SIX REVISIONS. **Native App vs. Mobile Web App: A Quick Comparison**. Disponível em: <http://sixrevisions.com/Mobile/Native>. Acessado em abril de 2014.

CECCATO, Leandro Martins. **IBM Worklight - IBM Mobile Foundation Overview**. Disponível em: <http://slideplayer.com.br/slide/352366>. Acessado em abril de 2014.

SALESFORCE. **Native, HTML5, or Hybrid: Understanding Your Mobile Application Development Options**. Disponível em < https://developer.salesforce.com/page/Native,_HTML5,_or_Hybrid:_Understanding_Your_Mobile_Application_Development_Options>. Acessado em maio de 2014.

WEBMONKEY. **How Do Native Apps and Web Apps Compare?**. Disponível em: <http://www.webmonkey.com/2010/08/how-do-native-apps-and-web-apps-compare>. Acessado em maio de 2014.

CONSTINE, Josh. **Zuckerberg Shows He's The Right Man For The Job. Now That Job Needs Doing**. Disponível em: <http://techcrunch.com/2012/09/11/zuckerberg-the-leader>. Acessado em maio de 2014.

DREGE, Stuart. **Financial Times: 'There Is No Drawback To Working In Html5'**. Disponível em: <http://www.theguardian.com/media/appsblog/2013/apr/29/financial-times-html5-no-drawbacks>. Acessado em maio de 2014.

W3C. **The Worldwideweb Browser**. Disponível em: <http://www.w3.org/people/berners-lee/worldwideweb.html>. Acessado em maio de 2014.

KIOSKEA. **O Que É Um Url**. Disponível em: <http://pt.kioskea.net/contents/288-url>. Acessado em Junho de 2014.

VENKATESH, C. R. Venkatesh. **Dot Com Infoway Releases Html5 Infographic**. Disponível em: <http://www.dotcominfoway.com/blog/dot-com-infoway-releases-html5-infographic>. Acessado em maio de 2014.

W3C HTML 5. **HTML 5 – Curso W3c Escritório Brasil**. Disponível em: <http://www.w3c.br/pub/cursos/cursohtml5/html5-web.pdf>. Acessado em maio de 2014.

SOUDERS, Steve. **High-performance web sites**. Communications of the ACM, v. 51, n. 12, p. 36-41, 2008.

Hypertext Transfer Protocol. Disponível Em: <http://tools.ietf.org/html/rfc2616#page-108>. Acessado em junho de 2014.

Fadino, Rodolfo. **Cache + .Net: Cache Http (Aumente A Performance De Seu Site)**. Disponível em: <http://www.rodolfofadino.com.br/2013/10/cache-net-Cache-http-aumente-a-performance-de-seu-site>. Acessado em junho de 2014.

GOOGLE 2. **Getting Started With The Youtube Data Api**. Disponível em: <https://developers.google.com/youtube/v3/getting-started?hl=pt-br>. Acessado em junho de 2014.

CERT.br. **Cartilha de Segurança para Internet**, versão 4.0 / CERT.br – São Paulo: Comitê Gestor da Internet no Brasil, 2012.

ANDRADE, Marcos Tadeu de. **Mecanismos de autenticação e autorização em redes sociais virtuais: o caso futweet**. Universidade Federal de Pernambuco, Apresentado em agosto de 2010.

MARSHALL, Sam. **Creating Offline Web Applications. Using Html 5**. Institute Of Information And Mathematical Sciences Massey University, Albany. Apres março 2013.

W3C BEST PRACTICES, **Mobile Web Application Best Practices**. Disponível em: <http://www.w3.org/tr/mwabp>. Acessado em junho de 2014.

PADRÕES DE PROJETO, **Use a cabeça – Padrões de Projeto** – 2ª Edição Revisada, Alta Books Editora – Starlin Alta Con Com LTDA 2009 – O'Reilly.

LEE, VALENTINO, Heather Schneider, and Robbie Schell. **Aplicações móveis: arquitetura, projeto e desenvolvimento**. Pearson Makron Books, 2005.

YAHOO DEVELOPER NETWORK, **Best Practices for Speeding Up Your Web Site**, Disponível em: <https://developer.yahoo.com/performance/rules.html>. Acessado em outubro de 2014.

YSLOW, **YSlow – Official Open Source Project Website**. Disponível em: <http://yslow.org>. Acessado em outubro de 2014.

PAGESPEED, **Make the Web Faster**. Disponível em: <https://developers.google.com/speed>. Acessado em outubro de 2014.

WARGO, J. M. (2012), Phonegap Essentials: Building Cross-Platform Mobile Apps, Addison-Wesley Professional.

MUNRO, J. (2012), 20 Recipes For Programming Phonegap: Cross-Platform Mobile Development For Android And Iphone, O'reilly Media.

CROWLEY, Matthew. Pro Internet Explorer 8 And 9 Development: Developing Powerful Applications Development, Appres November 2010.

Victor[1], Valdenir[2], Alexandre[3], Marconi Arouca Ribeiro[1], Tavares[2], Sztajnberg[3]. Avaliação e Redução do Tempo de Resposta de Sistemas WEB. Universidade do Estado do Rio de Janeiro, apresentado em Dezembro de 2012.

CASTLEDINE, Earle, Max Wheeler, and Myles Eftos. Build mobile websites and apps for smart devices. Sitepoint, 2011.

SÉRGIO, Lopes. A Web Mobile: Programe para um mundo de muitos dispositivos. Caso do Código, 2013.