UNIVERSIDADE DE CAXIAS DO SUL
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

HELENA BASSOTTO

DESENVOLVIMENTO DE UM PROTÓTIPO DE APLICATIVO MÓVEL UTILIZANDO FERRAMENTA MULTIPLATAFORMA

CAXIAS DO SUL
2014

HELENA BASSOTTO

DESENVOLVIMENTO DE UM PROTÓTIPO DE APLICATIVO MÓVEL UTILIZANDO FERRAMENTA MULTIPLATAFORMA

Trabalho de Conclusão do Título de Bacharel em Ciência da Computação pela Universidade de Caxias do Sul, como requisito parcial para obtenção do título de Bacharel.
Orientador Prof. Daniel Antônio Faccin.

CAXIAS DO SUL

2014

RESUMO

O desenvolvimento de aplicativos móveis está amplamente ligado às plataformas dos dispositivos. Com a diversidade de plataformas existentes e suas diferentes linguagens de programação, desenvolver um aplicativo abrangente às plataformas do mercado torna-se uma tarefa que demanda tempo e conhecimento maiores por parte dos realizadores do projeto. Uma possível solução para este cenário é o desenvolvimento de aplicativos multiplataforma híbridos. Neste tipo de aplicativo, o desenvolvimento é realizado somente uma vez, utilizando em conjunto as linguagens de programação HTML5, Javascript e CSS. Ferramentas de desenvolvimento são responsáveis por realizar o encapsulamento nativo para cada plataforma necessária. Desta forma, não seriam perdidas funcionalidades que estão disponíveis somente em aplicativos desenvolvidos no ambiente nativo de cada plataforma. Baseando-se no conceito de aplicativos multiplataforma híbridos, foi realizada uma pesquisa a respeito das ferramentas de desenvolvimento disponíveis. Foram selecionadas três ferramentas para realização de um estudo prático. A seleção baseou-se nos critérios de atendimento às plataformas Android, iOS e Windows Phone, e licenciamento gratuito. No estudo prático, as ferramentas foram avaliadas quanto ao atendimento e compatibilidade de funções nativas. Analisando os resultados obtidos, foi possível concluir que o desenvolvimento de aplicativos móveis híbridos é válido, bem como selecionar uma ferramenta para desenvolvimento de um protótipo de aplicativo híbrido. O protótipo foi construído baseando-se em características de qualidade de *software* (usabilidade, funcionalidade, confiabilidade e eficiência), com foco na portabilidade e responsividade.

Palavras-chave: desenvolvimento móvel, aplicativos móveis, aplicativos híbridos, funcionalidades nativas.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

ADT	*Android Development Tools*
AJAX	*Asynchronous Javascript and XML*
API	*Application Programming Interface*
APP	Aplicativo Móvel
CLI	C*ommand-Line Interface*
CPT	*Cross-Platform Tool*
CSS	*Cascading Style Sheets*
DOM	*Document Object Model*
DPCM	*Dots per CSS "centimeter"*
DPI	*Dots per CSS "inch"*
EM	Unidade relativa ao tamanho de fonte padrão dos dispositivos
ER	Diagrama Entidade-Relacionamento
FTP	*File Transfer Protocol*
HTML	*HyperText Markup Language*
IDE	*Integrated Development Environment*
ISO/IEC	*International Standards Organization/International Electrotechnical Commission*
JRE	*Java Runtime Environment*
JS	*Javascript*
JSON	*JavaScript Object Notation*
JSONP	*JavaScript Object Notation with padding*
MB	*Megabyte*
PC	*Personal Computer*
PCS	*Personal Communications Service*
PX	*Pixel*
SC7	Subcomitê de *Software*
SDK	*Software Development Kit*
SO	Sistema Operacional
UML	*Unified Modeling Language*
URL	*Uniform Resource Locator*
W3C	*World Wide Web Consortium*
WebApps	Aplicações baseadas na *Web*

XML *eXtensible Markup Language*

SUMÁRIO

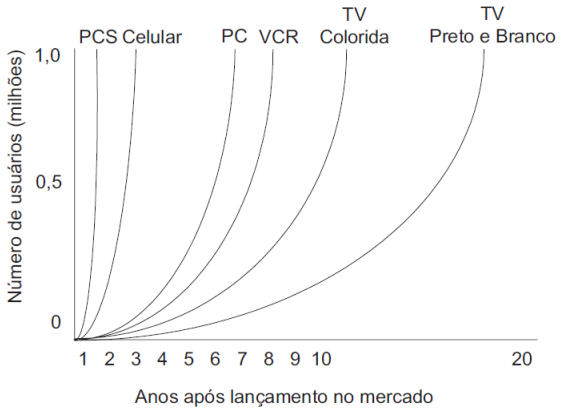
1 INTRODUÇÃO

Computação móvel representa um novo paradigma computacional. Surge como a quarta revolução na computação, antecedida pelos grandes centros de processamento de dados da década de sessenta, os terminais nos anos setenta, e as redes de computadores na década de oitenta. O novo paradigma permite que usuários tenham acesso a serviços independente de onde estão localizados. O ponto principal desta revolução é a possibilidade de mudanças de localização por parte do usuário, ou seja, mobilidade. Isso é possível graças à comunicação sem fio que elimina a necessidade do usuário manter-se conectado à infraestrutura física (MATEUS, 1998).

Segundo LAUDON (2010), *smartphones* e *notebooks* tornaram-se plataformas de computação portáteis. A comunicação sem fio foi responsável por esta evolução, permitindo a realização de tarefas antes somente vinculadas a *Personal Computers* (PCs), como acesso à *internet*, controle de *e-mails* e gerenciamento de informações pessoais. Tais funcionalidades influenciaram a aceitação da nova geração de dispositivos móveis por parte dos usuários.

A Figura 1 representa a aceitação de eletrônicos pelos usuários, considerando a quantidade de anos necessários para atingir a marca de 1 milhão de usuários. Neste gráfico, a nova geração de dispositivos móveis é chamada de *Personal Communications Service* (PCS).

FIGURA 1 - Número de usuários por ano de introdução da tecnologia



Fonte: MATEUS, 1998[1]

[1] Foi disponibilizada pelo autor em 2004, a preliminar da segunda versão da publicação. A revisão desta edição não foi finalizada e o autor solicita que a mesma não seja considerada para distribuição. Esta versão encontra-se disponível em: http://homepages.dcc.ufmg.br/~loureiro/cm/docs/cm_livro_2e.pdf.

1.1 DISPOSITIVOS MÓVEIS

Para um dispositivo ser considerado móvel deve possuir determinadas características. Conforme definido por LEE (2005), cada característica é importante em si mesma, mas nenhuma pode ser considerada definitiva. Esta variação ocorre conforme a percepção de cada usuário, que pode ser flexível em relação a alguma das características se houverem benefícios compensatórios em outra. As principais características dos Dispositivos Móveis são:

a) Portabilidade: no contexto de dispositivos móveis, o autor define portabilidade como a capacidade de um dispositivo ser facilmente transportável. Cabe ressaltar que não há relação com a portabilidade de aplicações entre plataformas. Neste sentido, dois dos fatores mais importantes que afetam a portabilidade de um dispositivo móvel são o tamanho e o peso do dispositivo, incluindo acessórios;

b) Usabilidade: um dispositivo móvel deve ser utilizável por tipos de pessoas diferentes em diversos ambientes. A usabilidade está diretamente ligada às características dos usuários: tamanho, força, flexibilidade, destreza, conhecimento e capacidade; características do ambiente: condições normais de funcionamento, ou seja, a adequação do dispositivo ao modo de trabalho do usuário, e condições extremas, por exemplo, calor, frio, umidade; e características do dispositivo: tempo de inicialização, integridade de dados, interface com o usuário e robustez/resistência, sendo estas características próprias de cada dispositivo, que podem afetar a usabilidade total;

c) Funcionalidade: a funcionalidade de um dispositivo móvel é implementada na forma de uma aplicação móvel, sendo que um dispositivo pode conter múltiplas aplicações em execução;

d) Conectividade: a função primária de um dispositivo móvel é conectar as pessoas e/ou sistemas, transmitir e receber informações. Nesta característica, é importante diferenciar móvel e sem fio. A mobilidade não está relacionada necessariamente a uma conexão sem fio. É possível funcionar de forma móvel e estar desconectado, onde informações são coletadas e transmitidas posteriormente, ao se conectar a uma rede sem fio ou ligada por cabos.

1.2 SISTEMA OPERACIONAL

Um Sistema Operacional (SO) em uma visão geral é o *software* que gerencia os componentes de *hardware* de um sistema computacional, tornando sua utilização mais simples e abstrata, tanto aos programas aplicativos como aos usuários (TANENBAUM, 2009). Assim como nos demais sistemas computacionais, os dispositivos móveis também possuem variedade em sistemas operacionais.

1.3 QUALIDADE DE *SOFTWARE*

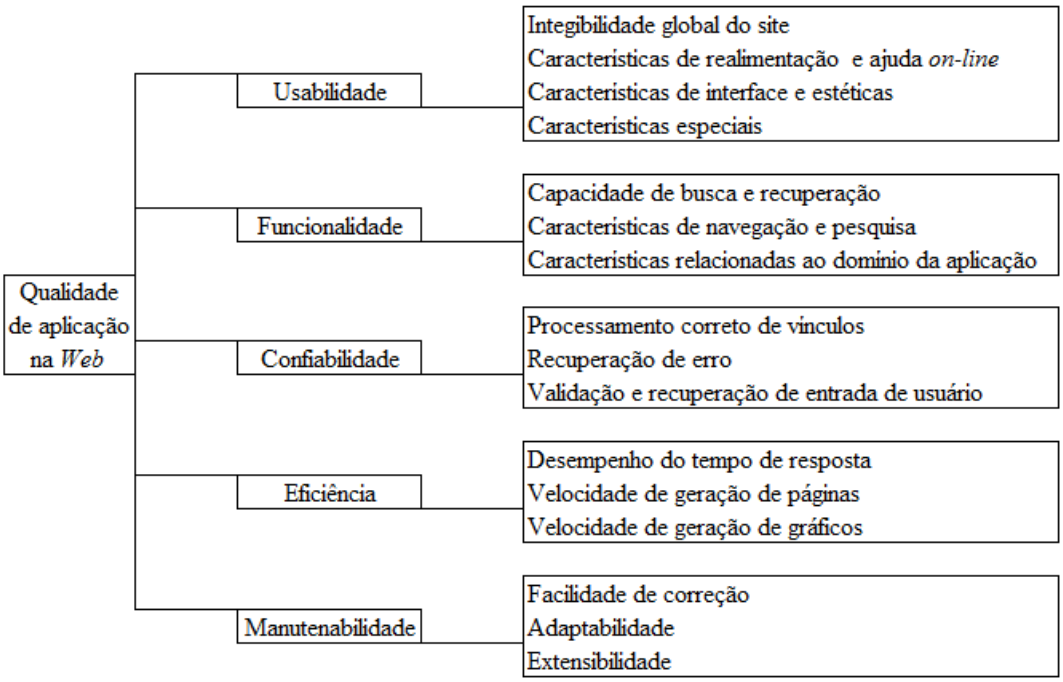
Qualidade de *software* pode ser vista como um conjunto de características que devem ser alcançadas em um determinado grau para que o produto atenda às necessidades de seus usuários. É por meio desse conjunto de características que a qualidade de um produto de *software* pode ser descrita e avaliada (ROCHA, 2001). Tais características foram definidas e padronizadas pelo Subcomitê de *Software* (SC7) da ISO/IEC em formas de normas e relatórios técnicos que abrangem as áreas de qualidade e avaliação de produtos de *software*, e teste e requisitos de qualidade em pacotes de *software*.

Na área de qualidade de produtos de *software*, a norma ISO/IEC 9126-1 define um modelo de qualidade para características externas e internas do *software* e um modelo para qualidade em uso. O modelo de qualidade para características externas e internas é composto por seis itens: Funcionalidade, Confiabilidade, Usabilidade, Eficiência, Manutenibilidade e Portabilidade. Qualidade em uso é a capacidade de o produto permitir aos usuários atingirem metas específicas em um contexto de uso determinado, e seu modelo de qualidade define quatro características: Efetividade, Produtividade, Segurança e Satisfação. Cada característica definida pode ser ampliada em subcategorias, e estas em atributos de qualidade. Para as questões abordadas neste trabalho somente serão consideradas as características principais.

Em aplicações baseadas na *Web* (*WebApps*), por questões de compatibilidade, as características para avaliação da qualidade de *software* sofreram adaptações, pois segundo ROCHA (2001), "*Web* é um ambiente complexo e, consequentemente, a avaliação de produtos de *software Web* é uma tarefa difícil dado o conjunto de características e particularidades envolvidas [...], podendo ter diferentes objetivos". Baseado nas particularidades das aplicações *Web*, PRESSMAN (2006) apresenta uma árvore de

características específicas de qualidade a serem consideradas, conforme apresentado na Figura 2. Somadas a essas características, são adicionados os atributos de Segurança, Disponibilidade, Escalabilidade e Prazo de colocação no mercado.

FIGURA 2 - Árvore de características de qualidade de Aplicações *Web*



Fonte: PRESSMAN, 2006

1.4 PROBLEMA DE PESQUISA

O desenvolvimento de aplicações móveis está, em grande parte, voltado à plataforma de Sistema Operacional em que o APP será executado. Esta restrição deve-se às particularidades de funcionamento de cada plataforma. Considerando este cenário, o desenvolvimento de um aplicativo em mais de uma plataforma demanda mais esforço, recursos e custos, pois o código terá de ser reescrito, ou no mínimo adaptado. Procurando otimizar o processo, identificam-se ferramentas de desenvolvimento multiplataforma, onde a implementação é realizada somente uma só vez, e o código gerado para todas as plataformas.

1.5 QUESTÃO DE PESQUISA

Baseado no problema de pesquisa descrito, foi criada a seguinte questão de pesquisa:

O desenvolvimento de um aplicativo móvel voltado à *Web*, utilizando uma ferramenta de desenvolvimento multiplataforma é viável, considerando a mínima perda de funcionalidades nativas?

1.6 OBJETIVOS

Para definição dos objetivos do trabalho, os mesmos foram divididos em objetivo geral e objetivos específicos.

1.6.1 Objetivo geral

O objetivo deste trabalho é analisar diferentes *frameworks* de desenvolvimento móvel multiplataforma, propondo um *layout* responsivo e desenvolvendo um protótipo de aplicativo móvel (APP) portável às diversas plataformas de dispositivos.

1.6.2 Objetivo específico

Visando a obtenção do objetivo geral, foram definidos os seguintes objetivos específicos:

1) Especificação de um *layout* responsivo, baseando-se nas características de qualidade de *software* usabilidade, funcionalidade, confiabilidade e eficiência.
2) Desenvolvimento de um protótipo de aplicativo baseado no *layout* responsivo para diferentes plataformas e diferentes dimensões, utilizando o *framework* selecionado após o estudo.

1.7 ESTRUTURA DO TEXTO

Primeiramente, este trabalho está estruturado de forma a conceituar e contextualizar os temas necessários para o seu desenvolvimento. Serão detalhados os sistemas operacionais móveis considerados, a possibilidade e utilização de HTML5, Javascript e CSS no desenvolvimento de aplicativos móveis, e diferenciação das soluções móveis disponíveis.

Na sequência foram estudadas ferramentas disponíveis no mercado para o desenvolvimento de aplicativos móveis. Três ferramentas foram selecionadas para realização da etapa de prototipação do trabalho.

Na etapa de prototipação foram desenvolvidos protótipos de aplicativos híbridos para validação de funcionalidades nativas. Os protótipos foram analisados, e a partir dos resultados obtidos foi definida a ferramenta para continuidade na segunda versão do protótipo de aplicativo.

A segunda versão do protótipo foi definida e implementada na etapa de desenvolvimento do trabalho. Foram descritos o *layout* e o novo funcionamento do protótipo desenvolvido. Ao final, foram realizados testes de compatibilidade das definições em relação à portabilidade de dispositivos e plataformas.

2 REVISÃO BIBLIOGRÁFICA

A revisão bibliográfica está dividida de forma a apresentar as principais informações para o desenvolvimento de aplicativos móveis.

2.1 SISTEMA OPERACIONAL

Conforme definido previamente na introdução do trabalho, assim como nos sistemas computacionais, os dispositivos móveis também possuem diversas opções de sistemas operacionais. Neste trabalho, serão considerados três dos SOs móveis mais utilizados atualmente: Android, iOS e Windows Phone.

2.1.1 Android

Sistema operacional desenvolvido pela Google está presente em mais de um bilhão de *smartphones* e *tablets* em todo o mundo, podendo ser encontrado em dispositivos das marcas Samsung, Sony, Motorola, LG, entre outras. É integrado com os demais aplicativos disponibilizados pela empresa, sendo os mesmos utilizados em ambientes *Desktops*. Cada versão do Android recebe um nome relacionado a um doce sendo a mais recente 4.4, chamada KitKat. A nova versão possui design mais sofisticado, novos recursos de usabilidade e desempenho aprimorado, otimizando memória e aperfeiçoando a tecnologia *touchscreen* apresentando mais rapidez e precisão. Para desenvolvedores, a empresa disponibiliza um tutorial com dicas a respeito das diversas dimensões dos dispositivos e um Kit de Desenvolvimento de *Software* (SDK), contendo as Interfaces de Programação de Aplicativos (APIs) e ferramentas necessárias para compilação, teste e *debug* dos aplicativos criados. Na Loja Virtual da Google, o Google Play, estão disponíveis aplicativos desenvolvidos pela própria empresa e por terceiros, vídeos, jogos, conteúdo para leitura, músicas e filmes (GOOGLE, 2014).

2.1.2 iOS

Sistema operacional desenvolvido pela Apple, atualmente em sua sétima versão, está presente em todos os dispositivos móveis da marca (iPhone, iPad e iPod Touch). Por ser projetado especialmente para o hardware também desenvolvido pela Apple, o iOS possui total integração com os recursos disponíveis, como processador *dual core*, os chips gráficos rápidos e antenas *wireless*. Para desenvolvedores interessados na criação de aplicativos móveis (APPs) e jogos, a Apple disponibiliza um conjunto de ferramentas e APIs voltados especialmente a cada dispositivo iOS, garantindo assim o total aproveitamento da tecnologia. Os Apps desenvolvidos por terceiros e pela própria Apple, estão disponíveis em sua Loja Virtual, a App Store, que atualmente possui mais de um milhão de opções cadastradas, nas mais diversas áreas. Como forma de segurança, antes da disponibilização de um aplicativo, a empresa realiza uma avaliação em seu conteúdo para evitar a ocorrência de *malware* (APPLE, 2014).

2.1.3 Windows Phone

Sistema operacional desenvolvido pela Microsoft, atualmente encontra-se na versão oito. Esta versão também é a última disponibilizada pela empresa para o sistema operacional Windows, voltado à PCs. Com a utilização do Windows Phone, serviços e programas disponibilizados pela Microsoft podem ser sincronizados aos dispositivos móveis. Este sistema operacional é direcionado principalmente à facilidade de uso e alta personalização pelo usuário. Atualmente pode ser utilizado em dispositivos das marcas Nokia, Samsung, HTC, entre outros. Para desenvolvedores, está disponível o Kit de Desenvolvimento de *Software* (SDK) e emuladores para testes dos aplicativos. Além destes, antes da liberação de versões, é disponibilizada uma prévia das mesmas para validações de aplicativos. Os aplicativos desenvolvidos são disponibilizados na loja virtual, Windows Phone Store, que possui mais de 175.000 aplicativos e jogos disponíveis para *download.* Todos os APPs são certificados pela Microsoft, garantindo a proteção contra *malware* e vírus (MICROSOFT, 2014).

2.2 DESENVOLVIMENTO DE APLICATIVOS MÓVEIS

O desenvolvimento de páginas *Web*, assim como o de aplicativos móveis, está diretamente ligado ao desenvolvimento *front-end* da aplicação, ou seja, a interface que será apresentada ao usuário. A base da criação da interface é realizada utilizando a linguagem HyperText Markup Language (HTML), em conjunto com as tecnologias Cascading Style Sheets (CSS) e Javascript (JS) (MAZZA, 2012).

Segundo JUNTUNEN (2013), o termo HTML além de representar a própria linguagem de programação, muitas vezes pode ser utilizado com a finalidade de generalizar sua utilização em conjunto com CSS e Javascript. Esta generalização torna-se possível pois o HTML define o conteúdo, o CSS formata sua apresentação e o Javascript controla o comportamento entre as tecnologias anteriores.

É importante considerar que os navegadores ainda não são totalmente compatíveis até mesmo com as versões atuais das linguagens (HTML5 e CSS3). Desta forma, torna-se necessária em algumas situações, a utilização de prefixos proprietários.

Os prefixos proprietários permitem que sejam utilizadas propriedades e funcionalidades ainda não totalmente implementadas e homologadas nos navegadores. O problema em ser utilizada tal prática é a replicação de código conforme a compatibilidade de cada navegador. Para minimizar o retrabalho desta prática é possível utilizar ferramentas desenvolvidas em Javascript (MAZZA, 2012).

O desenvolvimento modular, disponibilizado com o lançamento do HTML5 e o CSS3, possibilitou que alterações nas linguagens fossem liberadas em módulos menores. Tais liberações permitem que alterações nas linguagens sejam adaptadas mais rapidamente aos *browsers*, facilitando também a utilização por desenvolvedores. Por outro lado, a liberação de alterações em módulos de forma independente pode causar problemas mais frequentes de compatibilidade. Estes problemas se devem às diferenças de tratamento de características realizadas pelos *browsers*, que podem ocorrer em maior número do que quando realizadas liberações totais nos padrões das linguagens (FERREIRA, 2014).

2.2.1 HTML5

"HTML é uma linguagem para publicação de conteúdo (texto, imagem, vídeo, áudio e etc.) na *Web"*. O HTML foi desenvolvido para ser uma linguagem entendida por diversos meios de acesso, independente de plataformas ou *browsers*, distribuindo assim a informação de uma maneira global. É estruturado na forma de hipertexto, ou seja, conjuntos de elementos (nós), ligados por conexões. Estes elementos não são ligados linearmente, mas suas ligações formam comunicação de dados e organização das informações relacionadas (FERREIRA, 2014).

A última versão da linguagem HTML, o HTML5, prioriza os seguintes itens, conforme HARRIS (2011): Linguagem com núcleo simples, m*arkups* baseados na semântica, CSS utilizado para detalhes de estilo, páginas *Web* utilizadas como aplicações e Javascript central, incluindo a utilização da tecnologia Asynchronous Javascript and XML (AJAX).

Um dos principais objetivos desta versão é "facilitar a manipulação do elemento possibilitando o desenvolvedor a modificar as características dos objetos de forma não intrusiva e de maneira que seja transparente para o usuário final". Dentre as diferenças sobre as versões anteriores, está a melhor integração com ferramentas para CSS e Javascript, permitindo a manipulação de suas características; Alteração em elementos e atributos para aumento da eficácia na reutilização; Modificação na organização do código nas páginas, de forma que a reutilização da informação seja facilitada e preparada para futuros dispositivos; entre outros (FERREIRA, 2014).

O termo HTML5 é muitas vezes usado como referência para novas tecnologias e APIs. Alguns exemplos destas tecnologias incluem desenhos realizados com a *tag canvas*, novas *tags* para elementos multimídia (áudio e vídeo), funcionalidade *drag-and-drop* (arrastar e largar elementos nas interfaces gráficas), fontes incorporadas, armazenamento *off-line*, entre outros (GOLDSTEIN, 2011).

2.2.2 CSS3

O CSS é parte importante na criação de páginas *Web*, pois é responsável pela apresentação visual dos elementos do HTML, ou seja, formata a informação criada pelo

HTML. Esta informação pode ser representada por elementos como texto, imagens, áudio, vídeo ou outros elementos criados (CSS: Curso W3C Escritório Brasil, 2014).

Acompanhando o HTML5, surge a nova versão padrão do CSS, o CSS3. Esta versão apresenta novos recursos como suporte à fontes incorporado, novos seletores, quebra de elementos em colunas no *layout*, e melhorias visuais, como transparência, sombras, cantos arredondados, animações, gradientes e transformações (HARRIS, 2011). Muitas das melhorias visuais já estavam presentes anteriormente na criação de páginas, mas devido à falta de recursos nativos da linguagem, sua utilização tornava-se mais trabalhosa aos desenvolvedores, inclusive necessitando da presença de elementos extras do HTML. Com a liberação nativa destes recursos, o código torna-se mais acessível aos desenvolvedores, com melhorias na facilidade de manutenção, menores alterações na apresentação de imagens e carregamento mais rápido das páginas (GOLDSTEIN, 2011).

2.2.3 Javascript

O Javascript desempenha um papel importante no comportamento e funcionalidades de aplicações. É usado para definir as ações de cada elemento HTML. Sem a presença do JS, algumas das características mais importantes do HTML5 não estariam acessíveis. Por exemplo, a *tag canvas*, armazenamento de dados local e a funcionalidade geolocalização dependem diretamente desta linguagem de programação (HARRIS, 2011).

A interface entre os objetos do HTML e o Javascript é chamada Document Object Model (DOM). O DOM é uma estrutura em formato de árvore que permite visualizar a ligação e hierarquia entre cada um dos elementos do documento HTML (FERREIRA, 2014).

2.3 SOLUÇÕES MÓVEIS

Ao desenvolver um aplicativo é necessário considerar as funcionalidades a serem atendidas e a abrangência de plataformas requisitada. Além disso, fatores como tempo, custo, recursos disponíveis e conhecimento da equipe influenciam em sua realização. Com base nos requisitos anteriores, são apresentadas soluções para o desenvolvimento de aplicativos móveis. Tais soluções podem ser divididas em três categorias: Nativas, *WebApps* e Híbridas.

2.3.1 Aplicações Nativas

São criadas especificamente para uma plataforma móvel, não sendo compatíveis nas demais plataformas. No seu desenvolvimento a ferramenta utilizada é o SDK disponibilizado pela empresa do SO móvel.

Conforme JUNTUNEN (2013), a principal vantagem na utilização de aplicações nativas é o melhor uso dos recursos dos dispositivos, tanto de *hardware* como de *software*. Além disso, não necessitam de conexão à *internet* para o pleno funcionamento, e podem ser disponibilizados na loja de aplicativos de seu respectivo SO. A opção de distribuir o aplicativo na loja virtual possibilita que o mesmo seja mais difundido entre usuários, tornando maior sua visibilidade no mercado.

Uma desvantagem da solução nativa é a necessidade de desenvolvimento do mesmo aplicativo para cada um dos SOs a serem atendidos. Isso implica no aumento dos custos, do tempo e da quantidade de recursos para a finalização do aplicativo (JUNTUNEN, 2013). Algumas plataformas, como o iOS, necessitam de licença para o desenvolvimento e aprovação pela Apple para distribuição do APP na loja virtual (SMUTNÝ, 2012). Outro ponto negativo nesta solução é a fragmentação dos sistemas operacionais, pois conforme o SO é atualizado, os aplicativos podem tornar-se incompatíveis caso não ocorram manutenções.

Além disso, embora o desempenho dos dispositivos móveis aumente constantemente, as aplicações nativas são limitadas por restrições na arquitetura dos dispositivos, como poucos recursos de computação disponíveis e duração da bateria (JUNTUNEN, 2013).

2.3.2 *WebApps*

WebApps são aplicações desenvolvidas para *Web*, utilizando as tecnologias HTML5, CSS3 e Javascript. Segundo SIN (2012), mesmo vinculadas a *Web*, tais aplicações podem possuir funcionalidades e comportamento semelhantes às aplicações desenvolvidas de forma nativa.

A principal vantagem desta solução é a possibilidade de projetar aplicações sem vínculos com o tipo de dispositivo móvel e plataforma a serem utilizados. O acesso é realizado por meio de navegadores de *internet*, apresentando menores problemas de compatibilidade, estando seu funcionamento vinculado à capacidade de interpretação de

HTML5 e CSS3 por parte do *browser* utilizado. Apesar de não apresentar ícone de acesso (*start-up* na tela inicial do dispositivo) da mesma forma que as aplicações nativas, é possível adaptar um atalho para que o uso do aplicativo seja facilitado ao usuário (SIN, 2012).

Ao desenvolver uma *WebApps*, é importante considerar a perda de velocidade dos aplicativos *Web* em comparação com os aplicativos nativos. Com a evolução das tecnologias, esta diferença de processamento tende a diminuir, considerando melhor suporte de banda de rede e aumento do armazenamento em *cache* (ROLNITZKY, 2014). *WebApps* não são indicadas para aplicações que necessitem de processamento de imagens e jogos, incluindo jogos 3D, pois necessitam de um tempo menor de resposta. Tal perda de desempenho pode ser minimizada com um aplicativo construído de forma correta utilizando a solução *Web* (CHARLAND, 2011).

Outro ponto a ser considerado quando projetadas e desenvolvidas aplicações *Web*, é a questão da interface do aplicativo. Quando realizado um APP nativo, a interface sempre terá a mesma forma de apresentação e comportamento. Já em *WebApps*, é necessário considerar que poderão acontecer variações nas características, conforme o navegador utilizado.

Em *WebApps* o acesso aos recursos dos dispositivos é limitado. Recursos como câmera, acelerômetro, lista de endereços, e geolocalização, por exemplo, não estão disponíveis para acesso nas aplicações *Web*. Como solução alternativa, para o recurso de geolocalização, há uma API disponível para a maioria dos navegadores. Além desta API, o World Wide Web Consortium (W3C), responsável por manter os padrões na *Web*, mantém um grupo que realiza a criação de APIs para acesso a outros recursos nativos dos dispositivos (ROLNITZKY, 2014).

O suporte *off-line* também não está totalmente disponível aos aplicativos *Web*. Apesar do HTML5 possuir o recurso de trabalho *off-line*, nem todos os navegadores suportam esta opção (ROLNITZKY, 2014). Esta solução é dependente também de conexão a um servidor de aplicação, onde o acesso pode ocorrer de forma instável ou tornar-se indisponível (JUNIOR, 2013).

2.3.3 Aplicações Híbridas

Apresentam uma junção das duas soluções já apresentadas. De forma geral, aplicativos híbridos são desenvolvidos como um *WebApp*, tendo grande parte de sua interface

exibida em navegadores, e encapsulados como uma aplicação nativa, possibilitando acesso a recursos e funcionalidades não disponíveis via *Web*.

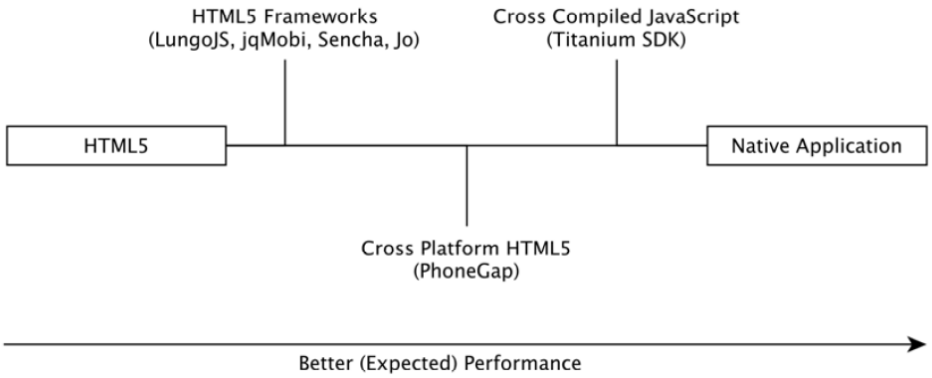
Do ponto de vista dos desenvolvedores e empresas, as soluções híbridas podem reduzir o tempo e o custo do desenvolvimento, devido a menor quantidade de código a ser replicado para as diferentes plataformas móveis. Grande parte do código é escrito em HTML, CSS e Javascript, que pode ser reutilizado em diferentes dispositivos.

Para os usuários, a diferença entre utilizar um aplicativo nativo e um aplicativo híbrido é menor se comparada a um *WebApp*. Aplicativos híbridos podem ser lançados da mesma forma como os nativos, baixados via lojas virtuais e armazenados diretamente nos dispositivos. Já *WebApps* não podem ser distribuídos nas lojas virtuais, seu armazenamento é realizado em um servidor de aplicação e o acesso é realizado via *browser* de *internet*.

Para o desenvolvimento de aplicativos híbridos, existem *frameworks* de desenvolvimento multiplataforma. Estas ferramentas criam uma arquitetura sem vínculos com plataforma específica, permitindo executar o aplicativo em diversos SOs. Seguem o princípio de escrever o código somente uma vez e realizar a compilação para várias plataformas, com o mínimo de customizações necessárias. Em determinadas situações as ferramentas não suportarão todas as funcionalidades requisitadas, sendo necessário optar por alguma em específico. São exemplos de ferramentas multiplataforma o Sencha Touch, Rhodes, Titanium e PhoneGap (LIONBRIDGE, 2014).

A Figura 3 apresenta um comparativo de desempenho entre as *WebApps* (desenvolvidas em HTML5) e os aplicativos nativos. Entre as duas soluções, são qualificados alguns dos *frameworks* multiplataforma mais utilizados na criação de aplicativos híbridos.

FIGURA 3 – Escala de desempenho entre aplicativos HTML5 e nativos



Fonte: JUNTUNEN, 2013

2.4 FRAMEWORKS PARA DESENVOLVIMENTO

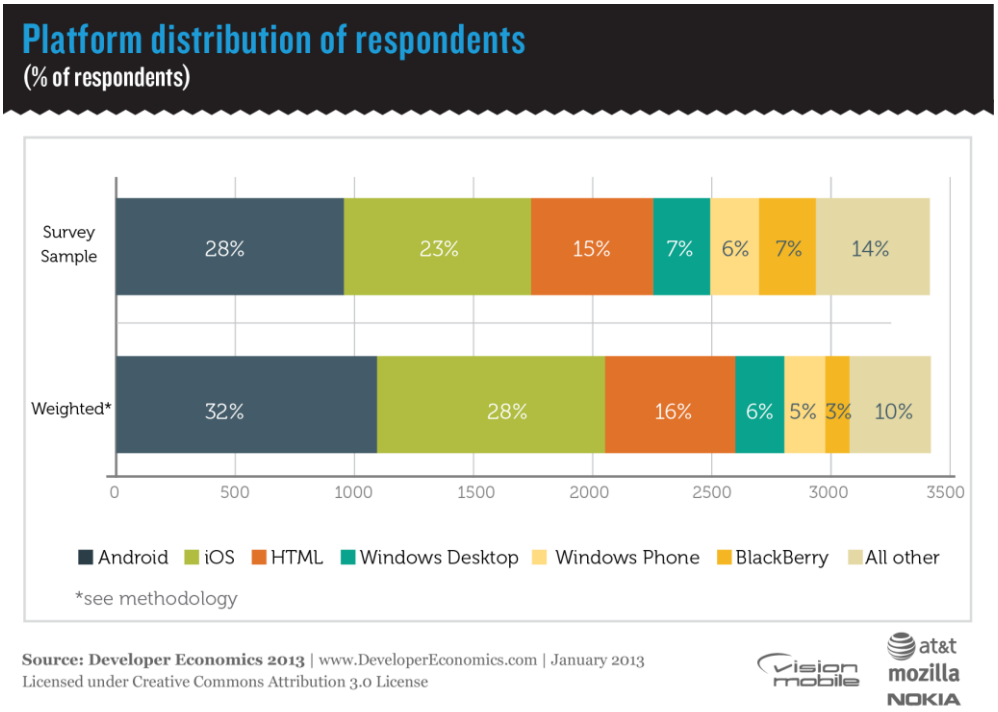
Segundo SMUTNÝ (2012), os *frameworks* de desenvolvimento móvel apresentam características em comum:

- Suporte a várias plataformas, que permite maior abrangência do aplicativo aos usuários;
- Aplicativos leves, devido à necessidade de transferência de dados via bandas de *internet* limitadas;
- Otimização para funcionalidade *touchscreen*, onde é necessário adaptar o design da aplicação para não considerar cursores de *mouse*, bem como tratar eventos especialmente para dispositivos móveis;
- Utilização de HTML5 e CSS3, pois são padrões compatíveis com a maioria dos navegadores móveis.

Uma pesquisa divulgada em 2013 apresentou as ferramentas multiplataforma, ou *Cross-Platform Tools* (CPTs), mais utilizadas entre os desenvolvedores entrevistados, e os critérios mais relevantes no momento da escolha. Realizada via questionário *on-line*, obteve cerca de 3.400 pessoas respondentes, distribuídas em 95 países.

A metodologia para cálculo dos percentuais das respostas foi baseada em proporções conforme a quantidade de desenvolvedores de cada plataforma. Este ajuste foi considerado necessário para que sejam obtidas as reais tendências do mercado. A Figura 4 apresenta os percentuais de utilização das maiores plataformas. A primeira amostra contém os reais percentuais obtidos pela pesquisa, e a segunda os percentuais reajustados para considerar a metodologia.

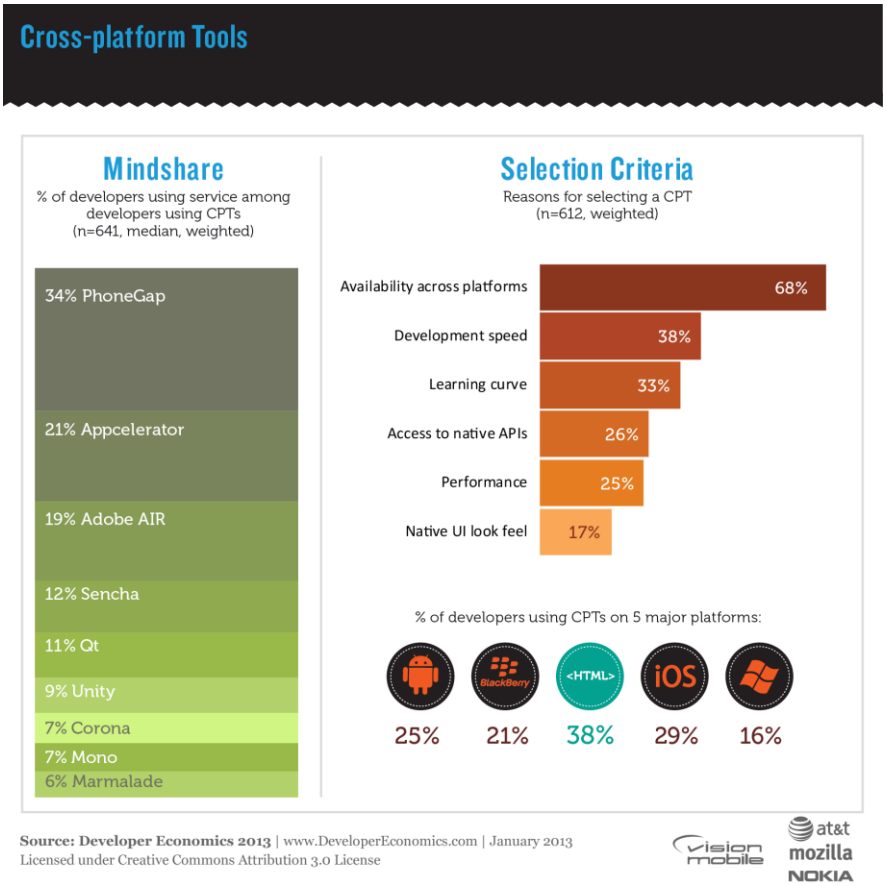
FIGURA 4 – Distribuição proporcional de desenvolvedores por plataforma



Fonte: VISION MOBILE, 2014a

Entre diversos questionamentos, foram apresentadas 100 ferramentas de desenvolvimento multiplataforma, onde cada participante indicou sua preferência de utilização (VISION MOBILE, 2014a). A lista completa das opções de ferramentas está disponível na versão de 2012 da pesquisa, e pode ser consultada no Anexo A deste trabalho. Os dados apurados referentes a ferramentas multiplataforma são apresentados a seguir:

FIGURA 5 – Dados sobre ferramentas multiplataforma apurados em pesquisa



Fonte: VISION MOBILE, 2014a

Segundo VISION MOBILE (2014a), os desenvolvedores HTML possuem mais abertura à utilização de tais ferramentas do que os desenvolvedores das quatro maiores plataformas nativas (38% contra 29% de desenvolvedores iOS, a plataforma nativa de maior percentual). Já o Windows Phone possui o menor percentual de utilização (16%). Um motivo para esta posição deve-se à restrição na quantidade de ferramentas com suporte ao SO.

A primeira prévia da pesquisa de 2014, VISION MOBILE (2014b) indica que as ferramentas multiplataforma são utilizadas por 30% do total de desenvolvedores móveis. O principal uso permanece com maior percentual entre os desenvolvedores HTML, mais especificamente o HTML5, tendo o percentual acrescido de 38% para 50% dos desenvolvedores.

A escolha de uma CPT é realizada com base em sua disponibilidade em várias plataformas, velocidade de desenvolvimento, curva de aprendizado, acesso à APIs nativas, desempenho, e aparência compatível com as plataformas nativas, respectivamente. Com isso, para a utilização de uma CPT ser válida em comparação com o desenvolvimento nativo, deve

apresentar maior facilidade e rapidez no desenvolvimento. Entre as próprias ferramentas, a diferenciação é feita por meio da capacidade de acesso às APIs nativas, melhor desempenho e compatibilidade com a interface nativa de cada ambiente.

O estudo aponta que a maioria dos desenvolvedores utiliza mais de uma CPT (1,91 em média), e a tendência é que um em cada quatro desenvolvedores utilizem mais de três ferramentas. Este cenário deve-se a pouca maturidade e falta de um padrão a ser seguido pela área, que ainda está muito focada nas plataformas Android e iOS. Entre as ferramentas disponíveis na pesquisa, foram apresentadas as nove com maior índice de utilização: PhoneGap, Appcelerator, Adobe AIR, Sencha, Qt, Unity, Corona, Mono e Marmalade, nesta ordem.

Na sequência serão apresentadas as características base de cada uma das ferramentas do *ranking*. Neste trabalho serão consideradas para o estudo prático as ferramentas que permitam a criação de aplicativos híbridos, suportem obrigatoriamente as plataformas Android, iOS e Windows Phone, e possuam licenciamento gratuito. Terão preferência também, as que possuírem como linguagem base o HTML5.

2.4.1 PhoneGap

PhoneGap, também conhecido como Cordova, é uma ferramenta *open source* para desenvolvimento multiplataforma baseada em HTML5, CSS3 e Javascript. Aplicativos móveis criados com esta ferramenta são executados simulando o ambiente nativo de cada plataforma. Esta funcionalidade deve-se à utilização de APIs que possibilitam o acesso aos recursos dos dispositivos. Para recursos específicos em certos dispositivos, que não são atendidos via APIs existentes, há a possibilidade de criação de *plugins* para comunicação entre a ferramenta e o componente nativo. Este encapsulamento das aplicações nos ambientes nativos possibilita também a distribuição dos aplicativos nas lojas virtuais.

O desenvolvimento de aplicativos utilizando o Cordova pode ser realizado em dois fluxos distintos de trabalho: centralizado em uma plataforma específica ou multiplataforma. O desenvolvimento centralizado é indicado quando o aplicativo necessita de codificação específica em funcionalidades da plataforma utilizada, tornando-se de baixo nível. Na grande maioria das vezes, esta codificação é realizada no próprio SDK da plataforma. O desenvolvimento multiplataforma não é indicado para este tipo de fluxo de trabalho, pois a

codificação torna-se específica por plataforma. Neste caso seria necessária a criação de *plugins* para integração de cada plataforma, e a realização de compilações distintas.

Para o desenvolvimento multiplataforma, a ferramenta disponibiliza um utilitário chamado Cordova CLI. O CLI permite abstrair os comandos específicos para acesso aos recursos, tornando-se uma ferramenta de programação de alto nível. As customizações necessárias são realizadas automaticamente pelo CLI. Este utilitário copia os comandos padrões para pastas específicas criadas cada plataforma, ajusta a compatibilidade do código conforme a necessidade, e gera os respectivos arquivos para execução.

Para uma plataforma ser atendida pelo CLI, é necessário que seu respectivo SDK esteja instalado no computador em que será desenvolvido o aplicativo. É importante considerar que a compilação dos programas para algumas plataformas depende diretamente do SO instalado na máquina utilizada para o desenvolvimento. A Tabela 1 exemplifica esta situação, onde são listadas as plataformas atendidas pelo CLI e SOs compatíveis para a compilação.

TABELA 1 – Relação Plataforma x Sistema Operacional atendidos pela ferramenta PhoneGap

	Sistema Operacional		
Plataforma	**Mac**	**Linux**	**Windows**
Amazon Fire OS	x	x	x
Android	x	x	x
BlackBerry 10	x	x	x
Firefox OS	x	x	x
iOS	x		
Windows 8			x
Windows Phone 7			x
Windows Phone 8			x

Fonte: PHONEGAP, 2014

Para utilização do CLI é necessária à instalação de dois outros programas: Node.js e um cliente Git (PHONEGAP, 2014). O Node.js é uma ferramenta baseada em Javascript, que permite a criação de conexões de rede não bloqueantes ao servidor. É orientada a eventos, o que a torna leve e eficiente, indicada para aplicações com grande quantidade de conexões simultâneas (NODE.JS, 2014). Git é um sistema para controle de versionamento de código.

Apresenta código aberto e é indicado para controle de projetos com grande velocidade (GIT, 2014). Por atender a todos os pré-requisitos definidos, o PhoneGap será utilizado nos testes práticos deste trabalho. A próxima ferramenta apresentada será o Appcelerator.

2.4.2 Appcelerator

Appcelerator, também conhecido como Titanium, é uma ferramenta de desenvolvimento *open source*, que permite a criação de aplicativos móveis nativos e híbridos com a utilização de HTML5. Possui suporte para os ambientes iOS, Android e BlackBerry. (APPCELERATOR, 2014). Por não atender a plataforma Windows Phone, não será considerada no estudo prático do trabalho. Na sequência será apresentada a ferramenta criada pelo Adobe.

2.4.3 Adobe AIR

O Adobe AIR é formado pelo conjunto de duas outras ferramentas, Flex e Adobe Flash Builder, adaptadas para o desenvolvimento móvel. A programação é realizada em Flex e a depuração e o encapsulamento com o Flash Builder. É uma ferramenta paga e suporta as plataformas Android, iOS e BlackBerry Tablet OS (ADOBE, 2014). Não será considerada no estudo prático do trabalho, pois não atende a plataforma Windows Phone e não é gratuita. A próxima ferramenta pode ser utilizada em conjunto com o PhoneGap já apresentado.

2.4.4 Sencha

Sencha Touch é uma ferramenta para desenvolvimento móvel baseado em HTML5, CSS3 e Javascript, que possui suporte às plataformas Android, iOS, Windows Phone, Microsoft Surface Pro e RT, e BlackBerry.

Permite o desenvolvimento de aplicativos que respondam às requisições do usuário e trocas de *layout* de forma instantânea. O tratamento visual dos aplicativos conta com animações e rolagem de tela suaves, com alta precisão de imagem. Para cada dispositivo, o

framework seleciona a melhor forma de rolagem da tela, ampliando a experiência do usuário na utilização dos aplicativos. Estão disponíveis também, temas pré-definidos para utilização nas plataformas iOS, Android, BlackBerry e Windows Phone.

Juntamente com a utilização da ferramenta, programas complementares necessitam ser instalados, atendendo assim todas as funcionalidades possíveis. São necessários o Sencha Cmd, disponibilizado pela mesma empresa, o Java Runtime Environment (JRE) para a execução do Sencha Cmd, e o Ruby para compilar informações CSS utilizadas. Quando realizado um aplicativo Android em Windows, também é necessária a instalação do Java SDK.

Sencha Cmd é uma ferramenta de linha de comando multiplataforma que fornece tarefas automatizadas e recursos para economia de tempo de desenvolvimento em aplicações criadas pelo Sencha Touch. Alguns exemplos de recursos disponíveis são: ferramenta de geração de código, servidor *Web* para manter dos arquivos do *localhost* e encapsulamento nativo, para as aplicações possuírem acesso aos recursos dos dispositivos e distribuição nas lojas virtuais. Sua utilização está disponível para Windows, Mac OS X e Linux.

Ao compilar e empacotar aplicativos desenvolvidos com o Sencha Touch há possibilidade de escolha do método mais adequado. No modo padrão da ferramenta, há a opção de compilação e criação de somente um pacote contendo os arquivos de todas as plataformas atendidas, ou realizar o processo separadamente para cada ambiente. Outra opção é a compilação e empacotamento compatíveis com os ambientes nativos, que fornece também o carregamento do simulador apropriado. O empacotamento nativo está disponível a partir da inicialização do PhoneGap ou Apache Cordova na execução do Sencha Touch.

Apache Cordova permite que sejam utilizados os recursos nativos e empacotamento para as plataformas Android, iOS, BlackBerry e Windows Phone. É a base para a criação do PhoneGap, estendendo suas funcionalidades ao *framework* (SENCHA TOUCH, 2014). A ferramenta será utilizada no estudo prático deste trabalho, pois atende os pré-requisitos definidos. Na sequência será apresentada ferramenta com desenvolvimento em C++.

2.4.5 Qt

Qt disponibiliza ferramenta específica para desenvolvimento mobile, o Qt Mobile. É uma ferramenta para desenvolvimento de aplicativos nativos multiplataforma, baseada em

C++. Atende somente as plataformas iOS e Android, e possui licenciamento pago (QT DIGIA, 2014). Devido ao não suporte à plataforma Windows Phone e o licenciamento pago, a ferramenta não será estudada neste trabalho. A próxima ferramenta apresentada é uma *engine* para desenvolvimento gráfico.

2.4.6 Unity

Unity é uma ferramenta especializada em desenvolvimento de aplicativos 2D e 3D, mais especificamente jogos. Possui suporte para aplicativos *desktop*, *Web*, Android, iOS, Windows Phone 8 e BlackBerry 10 (UNITY, 2014). Além de ser uma ferramenta paga, não será abordada, pois não engloba o enfoque deste trabalho. A ferramenta a seguir apresenta como um diferencial sua linguagem base, Lua.

2.4.7 Corona

Ferramenta indicada para o desenvolvimento de aplicativos na área de jogos 2D, educação, negócios e *eBooks*. Utiliza como padrão, a linguagem Lua. Suporta as plataformas iOS e Android. O suporte à plataforma Windows Phone está em construção. É uma ferramenta paga (CORONA, 2014). O não atendimento atual do Windows Phone e a necessidade de pagamento excluem a ferramenta do estudo prático deste trabalho. Na pesquisa da Vision Mobile, esta ferramenta obteve o mesmo percentual de utilização que a próxima a ser apresentada.

2.4.8 Mono

Mono é uma versão *open source* do *framework* .NET da Microsoft. Apresenta a linguagem C# para desenvolvimento e tem recebido apoio dos entusiastas da área para tornar-se a principal ferramenta para o desenvolvimento de aplicações Linux. Suporta as plataformas Android e iOS. Não disponibiliza encapsulamento para tornar os aplicativos híbridos (MONO, 2014). Por não possibilitar a criação de aplicativos híbridos, a ferramenta não será

considerada neste trabalho. Por fim, será a apresentada a última ferramenta considerada no ranking da Vision Mobile.

2.4.9 Marmalade

Formado por um conjunto de ferramentas para desenvolvimento móvel que permitem o desenvolvimento de aplicativos, incluindo jogos, utilizando a linguagem C/C++. Possibilita a criação de aplicativos multiplataforma para dispositivos móveis, *desktop,* e *Smart TV*. Aplicativos híbridos escritos em HTML5, CSS3 e Javascript podem ser criados utilizando uma ferramenta em específico do pacote, a Web Marmalade. O framework possui suporte as plataformas Android, iOS e Windows Phone 8, entre outros. É uma ferramenta paga (MARMALADE, 2014). O requisito de gratuidade faz com que a ferramenta não seja abordada neste trabalho. Após análise das ferramentas com maior percentual de utilização apontadas pela pesquisa da Vision Mobile, somente duas foram consideradas válidas para o estudo prático, com base nos critérios definidos para este trabalho. Foi necessário então, selecionar uma terceira ferramenta válida.

2.4.10 MoSync

MoSync é uma ferramenta *open source* para desenvolvimento de aplicativos multiplataforma. Está dividido em duas funcionalidades: MoSync SDK e MoSync Reload. Apresenta licenciamento gratuito, com a condição que os aplicativos distribuídos sejam disponibilizados para os demais usuários MoSync, com abertura de código. Para os desenvolvedores/empresas que não desejam tornar público seu código, há a opção de aquisição de licenças anuais para distribuição dos aplicativos.

MoSync SDK é uma IDE para desenvolvimento nas linguagens C/C++ ou HTML5/ Javascript, ou uma combinação das duas. Suporta nove plataformas, entre elas, Android, iOS e Windows Phone, permitindo que o código seja escrito somente uma vez e empacotado no formato de cada plataforma. A utilização de APIs permite que sejam acessados recursos nativos dos dispositivos, como gráficos, localização e câmera, entre outros.

Para aplicativos escritos em HTML5, juntamente com Javascript e CSS, a ferramenta disponibiliza emuladores para as plataformas Android e Windows Phone, onde é necessária a instalação e seus respectivos SDKs. Para o iOS, é disponibilizado um simulador que deve ser utilizado no sistema operacional OS X, com o Xcode instalado.

O MoSync Reload permite que aplicativos em HTML5, Javascript e CSS escritos em qualquer IDE ou editor de texto sejam recarregados de forma instantânea em todos os emuladores ou dispositivos em que estejam vinculados. Possui compatibilidade com as plataformas Android, iOS e Windows Phone 7. É formado pelo Reload Server, que deve ser executado na máquina em que as aplicações serão salvas, tornando-se o servidor dos aplicativos; Reload Client, um cliente que deve ser instalado em todos os emuladores e dispositivos que estarão vinculados ao servidor; e por fim o Reload Development UI, que é a interface instalada juntamente com o Reload Server, para controle das aplicações instaladas (MOSYNC, 2014).

Após compilação das informações de cada ferramenta listada anteriormente, foi possível criar a Tabela 2, comparando o atendimento aos requisitos do trabalho. Para o estudo prático, foram selecionados os ambientes PhoneGap, Sencha Touch e MoSync, pois atendem todos os critérios de seleção.

TABELA 2 – Ferramentas de desenvolvimento x Requisitos necessários no trabalho

	Requisitos					
	Plaformas			Gratuita	Aplicativos Híbridos	HTML5
Ferramenta	Android	iOS	Windows Phone			
PhoneGap	x	x	x	x	x	x
Appcelerator/Titanium	x	x		x	x	x
Adobe AIR	x	x			x	
Sencha Touch	x	x	x	x	x	x
Qt Mobile	x	x			x	
Unity	x	x	x			
Corona	x	x			x	
Mono	x	x		x		
Marmalade	x	x	x		x	x
MoSync	x	x	x	x	x	x

Fonte: AUTOR

2.5 *LAYOUT* RESPONSIVO

Layouts responsivos são aqueles que adaptam sua visualização e funcionalidades conforme o tamanho de tela do dispositivo em que são acessados. Segundo CAELUM (2014), "a ideia do responsivo é que a página se adapte a diferentes condições, em especial a diferentes resoluções". O termo *layout* responsivo faz parte do conceito de *Responsive Web Design*, criado em 2010 por Ethan Marcotte (MAZZA, 2012).

Segundo ZEMEL (2012), três elementos do CSS são essenciais para o *design* responsivo: *layout* fluído, imagens e recursos flexíveis e a utilização de *media queries.* Um *layout* fluído, imagens e recursos flexíveis são obtidos por meio da troca de tamanhos fixos ou absolutos nos objetos, pela utilização de valores relativos.

Os valores relativos indicados são percentuais para tratamento de tamanhos e *ems* para tratamento de fontes. "*Em*" é uma unidade baseada no tamanho de fonte utilizada pelo dispositivo. Desta forma, evita-se a ocorrência de barras de rolagem ou conteúdos com visualização prejudicada devido ao corte das informações na tela. O tamanho da tela de um dispositivo também é ligado à sua resolução, e não somente ao seu tamanho físico.

Media queries permitem que o *layout* da página seja ajustado, com base em alterações de orientação, resolução e tamanho de tela. A utilização de *media queries* faz com que os estilos sejam separados internamente no CSS por meio de blocos, também chamados de *breakpoints*. Há diferenças entre as propriedades do CSS e os parâmetros de *media queries*, também chamados de *media features*. Propriedades de CSS fornecem informações para a apresentação da página, e *media features* descrevem os requisitos necessários no dispositivo de saída para tornar um bloco de apresentação válido.

Media features permitem a verificação de características dos dispositivos, como largura (*width*), altura (*height*), largura física do dispositivo (*device-width*), altura física do dispositivo (*device-height*), orientação (*orientation*) e resolução (*resolution*), entre outros. Alguns parâmetros necessitam a informação de valores para validação, que podem ser definidos, por exemplo, em *Pixels* (px), "em", Pontos por polegadas (dpi) ou Pontos por centímetros (dpcm). Também é possível especificar a maioria dos *media features* por meio dos prefixos "min-" e "max-". O prefixo mínimo aplica o bloco para valores maiores ou iguais ao especificado, e o prefixo máximo define as propriedades para valores menores ou iguais ao especificado (Media Queries, 2014).

Considerando a especificação de *layouts* responsivos compatíveis com dispositivos móveis, há a possibilidade de realização de uma abordagem chamada *Mobile First.* Esta abordagem considera primeiramente os dispositivos menores para definição de características, aumentando a possibilidade de tamanhos até o maior tamanho necessário, o que pode ser chamado de melhoramento progressivo (*progressive enhancement*) (ZEMEL, 2012).

Com a utilização da abordagem *mobile first* é possível criar páginas mais simples, objetivas e com foco no conteúdo, pois o desenvolvimento é baseado na forma mais limitada de visualização e evolui conforme a necessidade. Uma forma de realização desta técnica é por meio da utilização de *media features* com os prefixos “min-”, como por exemplo, *“min-width”* (CAELUM, 2014).

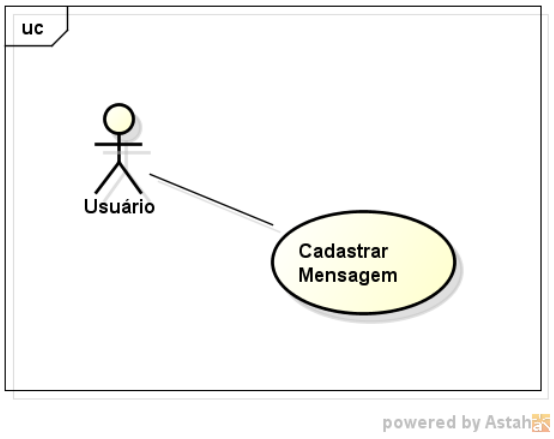
3 PROTOTIPAÇÃO

A etapa de prototipação é o início do desenvolvimento do trabalho. Nesta etapa é realizada a especificação do protótipo a ser desenvolvido para validação das ferramentas pré-selecionadas no estudo dos *frameworks* existentes. O protótipo foi baseado na tela de cadastro (inclusão) de uma mensagem. A especificação foi criada com base no Processo Unificado.

3.1 DIAGRAMA DE CASO DE USO

Os diagramas de caso de uso possibilitam o rastreamento de requisitos por meio de ações executadas por atores e o próprio sistema. A Figura 6 representa o diagrama de caso de uso criado para o protótipo.

FIGURA 6 – Diagrama de Caso de Uso



Fonte: AUTOR

3.2 DESCRIÇÃO DE CASOS DE USO E CASOS DE TESTES

A descrição de caso de uso especifica as possíveis ações a serem realizadas para a finalização do caso de uso. Completando cada caso de uso, são definidos os casos de testes, que apresentam as validações previstas para cada possível ação do usuário. A descrição dos casos de uso e os casos de teste estão detalhados na Tabela 3.

TABELA 3 – Descrição de Caso de Uso e Caso de Teste

Caso de Uso:	**Cadastrar Mensagem**
Finalidade:	Realizar inclusão de mensagem
Descrição:	Permite o usuário cadastrar mensagem inicial
Ator:	Usuário
Pré-Condições:	Usuário autenticado no sistema
Fluxo Principal:	1. Usuário informa o título da mensagem. 2. Usuário informa o corpo da mensagem. 3. Usuário confirma o cadastro. 4. Sistema salva as informações.
Fluxo Alternativo:	2. Usuário configura fonte da mensagem. 3. Usuário altera a cor de fundo da mensagem. 4. Usuário acessa a câmera do dispositivo para anexar foto na mensagem. 5. Usuário anexa arquivo na mensagem. 6. Usuário seleciona para habilitar recurso de geolocalização ao salvar a mensagem. 7. Sistema salva as informações.
Pós-Condições:	Registro de mensagem incluído no sistema
Caso de Teste:	**Cadastrar Mensagem**
Pré-Condições:	Acessar tela de inclusão de mensagem
Testes:	1. Valida habilitação dos campos. 2. Valida preenchimento do título e corpo da mensagem. 3. Valida configuração da fonte da mensagem. 4. Valida alteração da cor de fundo da mensagem. 5. Valida acesso a câmera do dispositivo e opção de anexar imagens. 6. Valida a opção de anexar arquivos diversos. 7. Valida seleção da opção de geolocalização. 8. Valida se o protótipo corresponde à orientação de tela

	do dispositivo. 9. Valida se o registro é salvo corretamente.
Resultado Esperado:	Operações realizadas com sucesso e dados cadastrados

Fonte: AUTOR

3.3 PROTÓTIPO DE TELA

O protótipo de tela é um esboço do *layout* do sistema. A Figura 7 apresenta o protótipo de tela para o cadastro da mensagem.

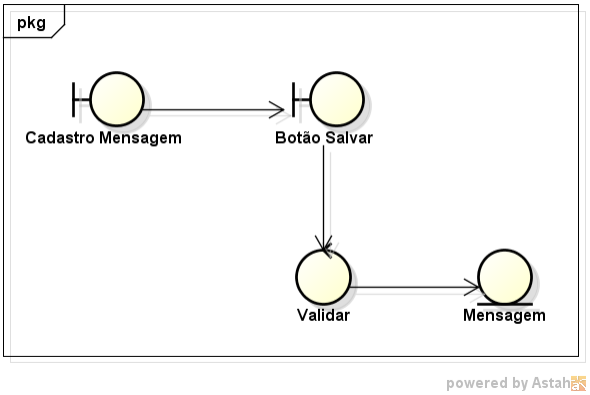
FIGURA 7 – Protótipo de tela

Fonte: AUTOR

3.4 DIAGRAMA DE ROBUSTEZ

O diagrama de robustez é uma forma especial de diagrama de colaboração da UML. Ele mostra os acontecimentos do sistema ao serem realizados eventos pelo usuário. O diagrama de robustez está representado na Figura 8.

FIGURA 8 – Diagrama de Robustez

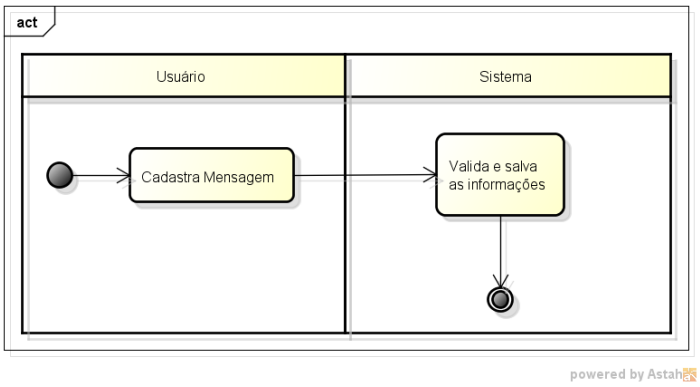


Fonte: AUTOR

3.5 DIAGRAMA DE ATIVIDADES

O diagrama de atividades é essencialmente um gráfico de fluxo. Ele mostra o fluxo entre as várias atividades que um objeto executa. A Figura 9 apresenta o diagrama de atividades criado para o protótipo.

FIGURA 9 – Diagrama de Atividades

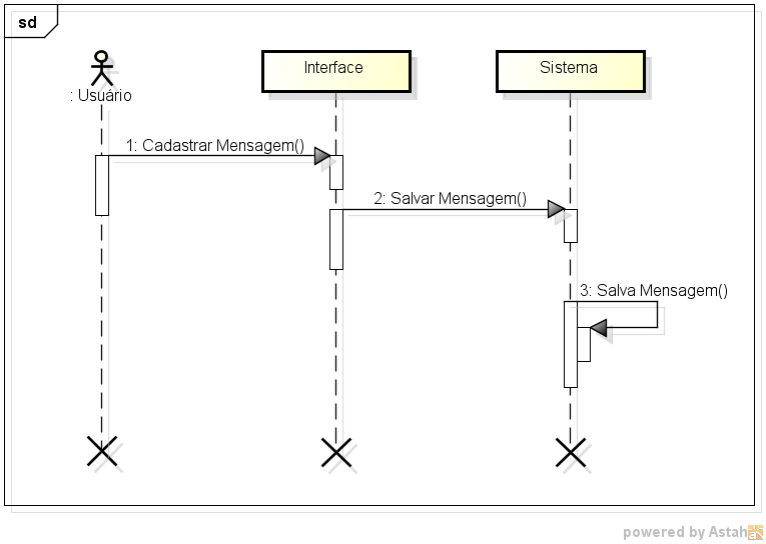


Fonte: AUTOR

3.6 DIAGRAMA DE SEQUÊNCIA

O diagrama de sequência centra-se na ordenação temporal das mensagens que são trocadas entre os objetos. O diagrama de sequência elaborado pode ser visualizado na Figura 10.

FIGURA 10 – Diagrama de Sequência

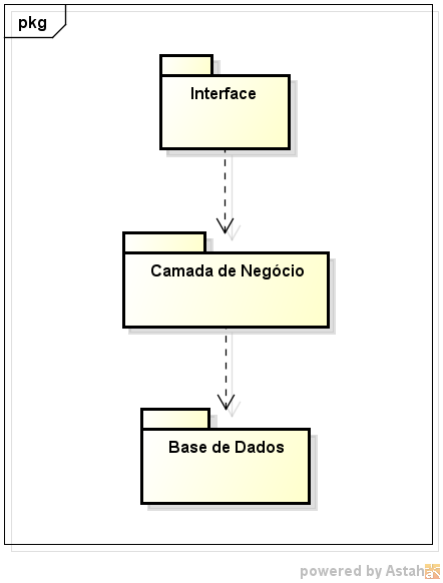


Fonte: AUTOR

3.7 DIAGRAMA DE PACOTES

Descreve os pacotes ou partes do sistema divididas em agrupamentos lógicos mostrando as dependências entre estes. Utilizado para ilustrar a arquitetura de um sistema mostrando o agrupamento de suas classes. A Figura 11 apresenta o diagrama de pacotes desenvolvido.

FIGURA 11 – Diagrama de Pacotes



Fonte: AUTOR

3.8 CONFIGURAÇÃO DOS AMBIENTES DE DESENVOLVIMENTO

O protótipo definido foi desenvolvido utilizando as três ferramentas pré-selecionadas MoSync, Sencha Touch e PhoneGap. A plataforma Android foi considerada como base para o desenvolvimento dos protótipos de aplicativos. Desta forma, os testes de desenvolvimento foram realizados nesta plataforma, e somente quando finalizado o protótipo, o código foi gerado para as demais. A conversão foi realizada com base na forma disponibilizada em cada ferramenta, sem alterações adicionais na codificação. Sendo assim, a análise de funcionalidades a ser apresentada posteriormente foi baseada no Android em comparação com as demais plataformas consideradas. A partir desta análise foi definida a ferramenta a ser utilizada como padrão na segunda parte deste trabalho.

Na sequência serão detalhados os requisitos para início do desenvolvimento em cada plataforma e ferramenta, bem como as características identificadas nos processos de desenvolvimento e validação.

3.8.1 Requisitos por plataforma

Cada plataforma móvel necessita de configurações específicas para o desenvolvimento e execução dos emuladores. Para o Android é necessária à instalação do Java (SDK e JRE) e das ferramentas para desenvolvimento Android (ADT). O ADT engloba a IDE Eclipse com *plugin* configurado para Android, gerenciadores para os SDKs da plataforma (SDK Manager) e criação dos perfis para o emulador (AVD Manager). As localizações destes arquivos devem ser informadas nas variáveis de ambiente do Sistema Operacional. Os testes desta plataforma foram realizados utilizando o sistema operacional Windows 7 Home Premium (64 bits), e a versão 4.4.2 do Android (API nível 19). O dispositivo utilizado como base do emulador foi o Google Nexus 5.

Na plataforma iOS, independente da ferramenta utilizada, a compilação de aplicativos somente pode ser realizada no sistema operacional MAC OS, em conjunto com a ferramenta para desenvolvimento Xcode. Os testes desta plataforma foram realizados utilizando uma máquina virtual configurada com o MAC OS Mountain Lion (10.8) e o Xcode versão 4.6.3, que possui o SDK iOS versão 6.1 e simulador integrados.

Para o Windows Phone, a implementação foi validada considerando a versão 7 da plataforma, pois o MoSync não possui suporte à versão mais recente da mesma, Windows Phone 8. Primeiramente foi instalado o Windows Phone SDK 7.1, e após realizado o *update* para a versão 7.8, que possui em sua instalação a ferramenta Microsoft Visual Studio 2010 Express for Windows Phone integrada. Para não ocorrer problemas na compilação dos aplicativos é necessário possuir o Framework .NET instalado e informado nas variáveis de ambiente do SO. A validação dos protótipos foi realizada utilizando o mesmo sistema operacional dos testes da plataforma Android.

3.8.2 Requisitos por ferramenta

Após a apresentação dos requisitos por plataforma, serão detalhados os requisitos necessários para utilização de cada ferramenta. Para a realização dos testes, as duas partes foram configuradas conforme descrito.

3.8.2.1 MoSync

A ferramenta MoSync, em comparação com as demais, mostrou-se a mais simples para configuração. Além dos requisitos básicos de cada plataforma, somente foi necessária a instalação do componente Java JDK também na versão 32 bits. O emulador para o Android teve de ser configurado separadamente, por meio da ferramenta Android AVD Manager e vinculado ao projeto MoSync. Desta forma o envio ao emulador é realizado automaticamente. O emulador Windows Phone foi reconhecido automaticamente somente quando utilizada a versão 7.1 do SDK. Após a atualização para a versão 7.8 o projeto teve de ser gerado para a plataforma via MoSync e aberto pela ferramenta Microsoft Visual Studio 2010 Express for Windows Phone para envio ao emulador. Para a plataforma iOS a ferramenta reconheceu automaticamente o emulador instalado.

A versão utilizada da ferramenta foi a 3.3.1. O desenvolvimento do protótipo foi realizado utilizando as linguagens de programação HTML, Javascript e CSS em conjunto.

3.8.2.2 PhoneGap

A validação do PhoneGap foi realizada por meio da ferramenta Cordova CLI. Para a utilização do CLI é necessária à instalação prévia dos programas Node.js e Git para todas as plataformas, e o Apache Ant para compilação da plataforma Android. Assim como o CLI, as ferramentas adicionais precisam ser informadas nas variáveis de ambiente para seu correto funcionamento.

A instalação do CLI é realizada via linha de comando no terminal e não apresenta IDE de desenvolvimento. Sendo assim, quando utilizados comandos Cordova, os mesmos devem ser realizados via terminal. Para a realização dos testes foram utilizadas as seguintes versões das ferramentas: Node.js (0.10.28), Git (1.9.4), Apache Ant (1.9.4) e CLI (3.5.0). Assim como o MoSync, o desenvolvimento foi realizado em conjunto com as linguagens HTML, Javascript e CSS. A criação dos arquivos foi realizada por meio de um editor de textos.

Ao realizar testes na plataforma Android, o aplicativo é enviado automaticamente ao emulador, por meio de um comando Cordova. Já no Windows Phone é necessário gerar o projeto pelo Cordova e abri-lo por meio do Microsoft Visual Studio 2010 Express for

Windows Phone para envio ao emulador. Da mesma forma, quando gerado o projeto para o iOS, o mesmo deve ser aberto com o Xcode para envio ao emulador.

3.8.2.3 Sencha Touch

Para iniciar a utilização do Sencha Touch é necessário possuir o Ruby instalado e um servidor w*eb* configurado para fornecer acesso às páginas. Foram utilizados a versão 1.9.3 do Ruby e o servidor *web* XAMPP versão 1.8.3. O servidor de arquivos foi selecionado por ser uma ferramenta gratuita e compatível com os sistemas operacionais Windows e MAC OS.

A instalação do Sencha Touch é dividida em duas partes. Uma das partes consiste em baixar os arquivos dos objetos no *site* da Sencha, e descompactá-los na pasta de arquivos configurada no Servidor *Web*. A segunda parte é a instalação do Sencha Cmd, que é o responsável por interpretar os comandos via terminal, pois assim como o PhoneGap, o Sencha Touch não possui IDE de desenvolvimento. Após as instalações é necessário incluir uma variável de sistema com a localização dos arquivos Sencha Cmd instalados. As versões das ferramentas Sencha utilizadas foram 2.3.1 para os arquivos Sencha Touch, e 4.0.4 para o Sencha Cmd.

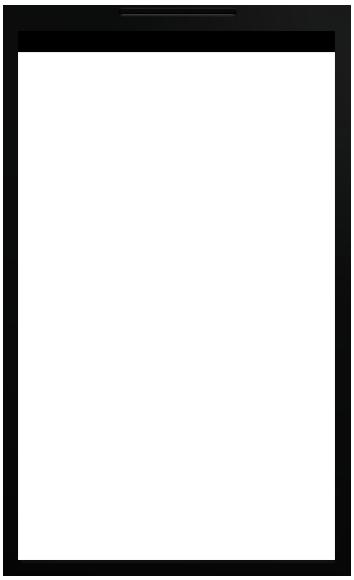
O Sencha Touch apresenta estrutura interna de arquivos diferente das demais ferramentas. Os arquivos Javascript são organizados em pastas distintas conforme a funcionalidade, por exemplo, v*iews*, c*ontrollers* e *stores*, entre outros. Essa distinção deve ser seguida para que o projeto compile corretamente. O desenvolvimento foi realizado utilizando os componentes gráficos da ferramenta ao invés do HTML para definição das telas e Javascript para a execução dos eventos. Quando criado um projeto, o arquivo CSS já é incluído automaticamente com o tema da ferramenta configurado. Estas configurações não foram alteradas, tornando o protótipo criado visualmente diferente dos demais.

A compilação de projetos Sencha para a plataforma Windows Phone é realizada somente por meio da utilização da ferramenta Cordova integrada, pois o Sencha Touch não possui suporte a esta ferramenta. Desta forma, foi utilizado o empacotamento nativo fornecido pelo Cordova para todas as plataformas.

Ao realizar testes do protótipo gerado, o mesmo é emulado diretamente para o Android. Para o iOS é necessário gerar o projeto e abrir o mesmo via Xcode, para então realizar o envio ao simulador. Para a plataforma Windows Phone também foi necessário gerar

o projeto e utilizar o Visual Studio 2010 Express for Windows Phone para envio ao simulador. Porém, ao enviar o projeto para o emulador, não foi possível validar a aplicação, pois não foi reconhecida na plataforma. Foram realizados testes com as versões 7.1 e 7.8 do emulador, e ambas retornaram uma tela em branco na execução. Esta tela pode ser visualizada na Figura 12.

FIGURA 12 – Execução do protótipo Sencha Touch no emulador Windows Phone



Fonte: AUTOR

3.9 ANÁLISE DAS FUNCIONALIDADES

Durante a realização dos testes práticos das ferramentas, foram avaliadas funcionalidades nativas específicas, para posterior comparação dos resultados. As funcionalidades desenvolvidas/testadas incluíram a possibilidade de configuração de fonte da tela (cor, tamanho e tipo da fonte), alteração da cor de fundo em objeto de entrada de dados, acesso à palete de cores, acesso à câmera do dispositivo, acesso a arquivos do dispositivo, recurso de geolocalização e troca de orientação de tela automática.

A análise das funcionalidades de cada protótipo desenvolvido foi realizada com base no agrupamento das características por ferramenta e plataforma.

3.9.1 *Layout* do protótipo

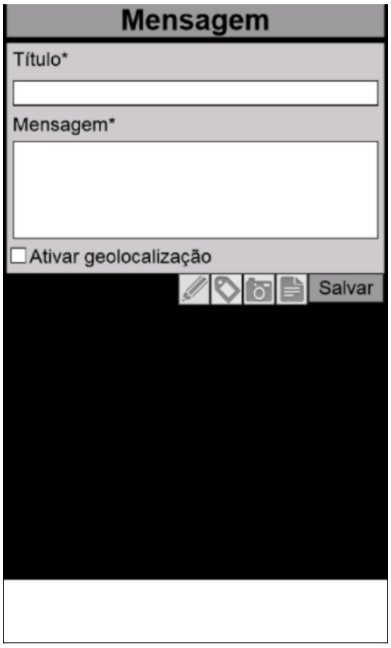
O *layout* dos protótipos MoSync e PhoneGap apresentou-se o mesmo, pois foram desenvolvidos utilizando a mesma base de CSS. Somente o protótipo Sencha Touch apresentou *layout* diferenciado devido à utilização do CSS com tema pré-configurado, conforme a Figura 13.

O Windows Phone apresentou uma particularidade devido ao tamanho da tela do emulador, pois mesmo tendo configurado o *layout* para abranger toda a altura da tela, uma faixa branca foi exibida em seu final, conforme visualizado na Figura 14.

FIGURA 13 – *Layout* dos protótipos PhoneGap e Sencha Touch no emulador Android

Fonte: AUTOR

FIGURA 14 – Protótipo PhoneGap no emulador Windows Phone



Fonte: AUTOR

3.9.2 Fonte da Mensagem

Nos protótipos criados com as ferramentas PhoneGap e MoSync foi possível definir a fonte, o tamanho e a cor da fonte apresentados no campo mensagem. Na plataforma Windows Phone, ao aumentar o tamanho da fonte o texto não é reposicionado automaticamente, sendo necessário alterar seu conteúdo para a atualização.

No protótipo desenvolvido com a ferramenta Sencha Touch, não foi possível alterar a cor da fonte da mensagem dinamicamente durante a execução do protótipo. A alteração da fonte e do tamanho realiza a alteração também no *label* do campo, conforme pode ser visualizado na Figura 16.

3.9.3 Palete de Cores

A palete de cores é utilizada na tela de configuração de Fonte e Fundo da mensagem. Nenhum dos ambientes possibilitou a criação da palete de cores de forma nativa. Para o MoSync e PhoneGap foi utilizado um componente de terceiro desenvolvido em jQuery,

chamado jquery.colorPicker (disponível em http://www.laktek.com/2008/10/27/really-simple-color-picker-in-jquery/). Um exemplo da palete de cores em funcionamento pode ser visualizado na Figura 15.

No Sencha Touch não foi possível realizar a integração com o componente jQuery pois os comandos não foram reconhecidos pela ferramenta no momento da execução. Além deste, foram realizados testes com um componente chamado ColorPicker, desenvolvido pela própria Sencha, disponível na ferramenta de desenvolvimento *desktop* da empresa, chamada Ext JS. Este componente também não foi reconhecido na execução do protótipo.

FIGURA 15 – Visualização da palete de cores no protótipo MoSync e emulador Android

Fonte: AUTOR

3.9.4 Fundo da Mensagem

Nos protótipos criados com o MoSync e o PhoneGap foi possível alterar a cor de fundo da mensagem, utilizando a seleção pelo componente de Palete de Cores. Como no Sencha Touch não foi possível utilizar a palete de cores, uma forma de validação da funcionalidade foi a implementação da geração de cores aleatórias quando confirmada a tela, para atualização da mensagem. A cor somente foi alterada no lado direito do campo, não abrangendo todo seu espaço. Além disso, quando o campo não possui conteúdo informado, a

cor não é apresentada. A atualização do fundo da mensagem no Sencha Touch pode ser verificada na Figura 16.

FIGURA 16 – Configuração de mensagem no protótipo Sencha Touch e emulador Android

Fonte: AUTOR

3.9.5 Geolocalização

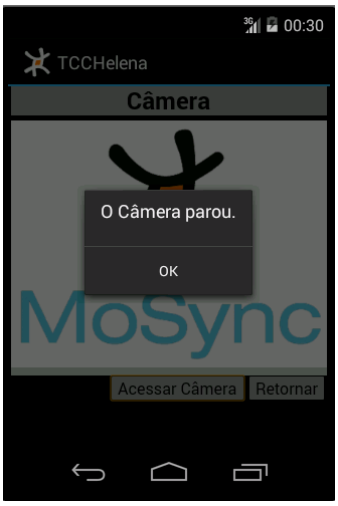
Os protótipos desenvolvidos com o PhoneGap e o MoSync apresentam a localização nos emuladores quando solicitado, com exceção do emulador Android, que não retorna a localização independente da plataforma ou ferramenta. Na execução do protótipo Sencha Touch no iOS, é solicitado ao usuário a habilitação da geolocalização, mas após a confirmação as coordenadas não são exibidas.

3.9.6 Acesso à câmera

O acesso à câmera não pode ser validado em nenhuma das ferramentas e plataformas. PhoneGap e MoSync não acessam a câmera e não apresentam erro ao usuário nas plataformas Windows Phone e iOS. Já o protótipo Sencha Touch no emulador iOS

apresenta erro no processo. No Android, todas as ferramentas acessam a câmera, mas apresentam erro no processo, conforme a Figura 17.

FIGURA 17 – Acesso à câmera no protótipo MoSync e emulador Android



Fonte: AUTOR

3.9.7 Acesso a arquivos

O acesso a arquivos foi criado para realizar a leitura em uma pasta específica do protótipo. Esta pasta foi incluída aos arquivos de cada projeto, contendo arquivos já salvos. Assim, foi possível analisar a forma de armazenamento de cada ferramenta e plataforma.

No protótipo desenvolvido no PhoneGap, plataformas Android e Windows Phone, os arquivos são buscados diretamente do cartão de memória. Para o Android foi possível incluir os arquivos na pasta de busca via ferramenta ADT. Para o Windows Phone não foi possível realizar a inclusão, pois o emulador não disponibiliza esta opção. Para a plataforma iOS, a função para busca de arquivos não foi reconhecida pelo emulador.

No protótipo desenvolvido com a ferramenta MoSync, o acesso aos arquivos permaneceu na pasta pré-configurada, tornando os mesmos visíveis nos emuladores. O protótipo Sencha Touch busca os arquivos salvos na determinada pasta, dentro do cartão de memória. Assim como PhoneGap, na execução Android foi possível incluir arquivos no cartão de memória via ferramenta ADT. Para a plataforma iOS, os arquivos não são reconhecidos e não retorna erro.

3.9.8 Orientação de tela

Todos os emuladores possuem a opção de troca de orientação, mas a tela não foi atualizada automaticamente em nenhum dos protótipos sem a realização de codificação.

Com base nos testes realizados, foi possível criar uma tabela comparativa de funcionalidades atendidas. Esta tabela está organizada por ferramenta e plataforma avaliada. Para cada funcionalidade atendida, foi atribuído um ponto para a respectiva ferramenta. Desta forma, apurou-se a pontuação final de cada ferramenta. Somente foram consideradas como atendidas as funcionalidades que obtiveram o comportamento esperado, conforme detalhado em cada item.

Para o item de acesso a arquivos somente foi considerado o atendimento da funcionalidade quando foi possível incluir arquivos na pasta de busca para serem reconhecidos pelo protótipo. Um exemplo desta situação é o PhoneGap executando no emulador Windows Phone. Neste caso, sabe-se que os arquivos são localizados a partir do cartão de memória, mas não foi possível simular esta inclusão.

A Tabela 4 apresenta os resultados obtidos, onde as ferramentas MoSync, PhoneGap e Sencha Touch apresentaram 13, 11 e 1 ponto respectivamente.

TABELA 4 – Funcionalidades atendidas nos protótipos por ferramenta e plataforma

Ferramenta		MoSync			PhoneGap			Sencha Touch		
Plataforma		Android	iOS	Windows Phone	Android	iOS	Windows Phone	Android	iOS	Windows Phone
Funcionalidade	Alteração da Fonte	x	x		x	x				
	Palete de Cores	x	x	x	x	x	x			
	Alteração do Fundo	x	x	x	x	x	x			
	Geolocalização		x	x		x	x			
	Acesso à câmera									
	Acesso a arquivos	x	x	x	x			x		
	Orientação de tela									

Fonte: AUTOR

3.10 CONSIDERAÇÕES SOBRE O CAPÍTULO

Analisando os resultados obtidos, verificou-se que grande parte das funcionalidades foram atendidas, o que torna o desenvolvimento multiplataforma híbrido válido para as funcionalidades nativas consideradas no estudo prático. O Windows Phone apresentou-se como a plataforma que possui menor compatibilidade em comparação com as demais. Esta característica também foi observada na quantidade de ferramentas que suportam esta plataforma, que está abaixo das concorrentes.

Considerando cada ferramenta individualmente, o Sencha Touch mostrou-se a ferramenta menos apropriada para a continuidade deste trabalho, apresentando somente um ponto na avaliação final. Praticamente todas as funcionalidades não foram compatíveis com as plataformas, ou foram atendidas parcialmente, além da principal incompatibilidade com o Windows Phone.

As ferramentas MoSync e PhoneGap apresentaram comportamento semelhante na maioria das funcionalidades, obtendo 13 e 11 pontos respectivamente. A diferença entre as duas ferramentas é a forma de leitura dos arquivos. Uma forma de atender esta funcionalidade no PhoneGap é a opção de programação específica para acesso ao cartão de memória, não tornando esta diferença um ponto negativo em sua utilização.

Com base neste cenário, optou-se por utilizar a ferramenta PhoneGap na continuidade do trabalho, pois apresentou bastante flexibilidade no desenvolvimento e com base na pesquisa prévia realizada, possui mais perspectiva de crescimento no mercado.

4 DESENVOLVIMENTO

A etapa de desenvolvimento é a continuação da etapa de prototipação. Nesta etapa foi realizada a definição, desenvolvimento e avaliação da segunda versão do protótipo já iniciado. Na segunda versão, o protótipo foi aprimorado para permitir a simulação de troca de mensagens entre dispositivos.

Chamado de Simulador de Mensagens, o protótipo apresenta tela de *login* no primeiro acesso realizado pelo dispositivo, e possibilidade de envio, recebimento e consulta local de mensagens. Ao realizar o envio de uma mensagem, podem ser configurados tamanho e cor de fonte do texto, e acesso à câmera e álbum de fotos para inclusão de um anexo. Usuários cadastrados e todas as mensagens trocadas são mantidos em um servidor onde os dispositivos realizam a consulta de novas mensagens.

4.1 *LAYOUT*

O *layout* do Simulador foi criado baseando-se em conceitos de telas responsivas. O desenvolvimento das telas foi realizado utilizando o *framework* de desenvolvimento *front-end* Bootstrap. Baseado em HTML, CSS e JS, e voltado para o desenvolvimento responsivo, o Bootstrap utiliza conceito *mobile first* e um sistema de colunas como base para o posicionamento dos elementos na tela. Tais colunas são tratadas por *media queries* e porcentagens de forma a manter a responsividade. O *framework* está programado para atender quatro tamanhos de tela pré-configurados, que representam os diversos tipos de dispositivos, conforme Tabela 6. Estes tamanhos podem ser alterados manualmente no site da ferramenta e seu código re-gerado com base na nova configuração (BOOTSTRAP, 2014). No desenvolvimento do protótipo foi utilizada a configuração padrão de tamanhos da ferramenta.

TABELA 6 – Tamanhos de tela pré-configurados no Bootstrap

Identificador Bootstrap	Dispositivos	Tamanho Tela	Tamanho em *Pixels*
col-xs	Celulares	Muito pequeno	Até 768px
col-sm	*Tablets*	Pequeno	Até 992px
col-md	*Desktops*	Médio	Até 1200px
col-lg	*Desktops*	Grande	1200px ou Maiores

Fonte: CAELUM, 2014

4.1.1 Qualidade de *Software*

O *layout* proposto tem como base as qualidades de *software* já mencionadas, usabilidade, funcionalidade, confiabilidade e eficiência. Características específicas são abordadas em cada item de qualidade, as quais posteriormente são avaliadas quanto à sua aplicabilidade nas diversas plataformas e dispositivos, conforme segue:

a) Usabilidade: considerada a característica que engloba Interface e Características Estéticas. O protótipo de aplicativo apresenta funções e *layout* simplificados. Todas as funções são acessadas por ícones agrupados no mesmo local, o início da página, priorizando o envio das mensagens. O registro de mensagens é apresentado de forma a diferenciar mensagens enviadas e recebidas. Considerando a visualização de imagens, é possível acessar cada imagem de forma ampliada.

b) Funcionalidade: são avaliadas características relacionadas ao Domínio da Aplicação. Mediante o contexto de envio de mensagens, foram consideradas como funcionalidades a configuração de texto da mensagem (tamanho e cor de fonte) e a possibilidade de inclusão de anexo (acesso à câmera ou álbum do dispositivo).

c) Confiabilidade: neste item de qualidade, a Validação e Recuperação de entrada de usuário são avaliadas. O envio de mensagens sem conteúdo textual ou imagem anexada não é permitido, mantendo a integridade dos dados cadastrados. Também foi verificado se as informações apresentadas no registro de mensagens são as mesmas informadas no momento do cadastro das mensagens.

d) Eficiência: neste quesito é considerado o Desempenho do tempo de resposta do protótipo de aplicativo. Esta avaliação foi baseada nas funcionalidades internas do protótipo e nas diversas formas de comunicação com o servidor de dados.

4.2 SERVIDOR

O servidor do protótipo de aplicativo foi criado por meio da hospedagem de *sites* Hostinger. O Hostinger oferece espaço em disco para armazenamento de conteúdos do *site* e banco de dados MySQL. A comunicação entre o cliente e o servidor deve ser realizada na linguagem de programação PHP. Há a possibilidade de criação de subdomínios gratuitos, que são hospedados entre os domínios disponíveis do Hostinger. Cada subdomínio gratuito criado tem disponível duas contas de FTP e duas bases de dados em MySQL, entre outros recursos (HOSTINGER, 2014).

Para este trabalho foi criado um subdomínio gratuito, disponível para acesso via URL http://tcchelena.url.ph/. A transferência de arquivos para o endereço foi realizada por meio da solução de FTP FileZilla 3.9.0.5. A comunicação entre o protótipo de aplicativo e o banco de dados do servidor foi desenvolvida por métodos jQuery. Foram utilizados os métodos GET (para recebimento) e POST (para envio), trafegando dados via JSONP.

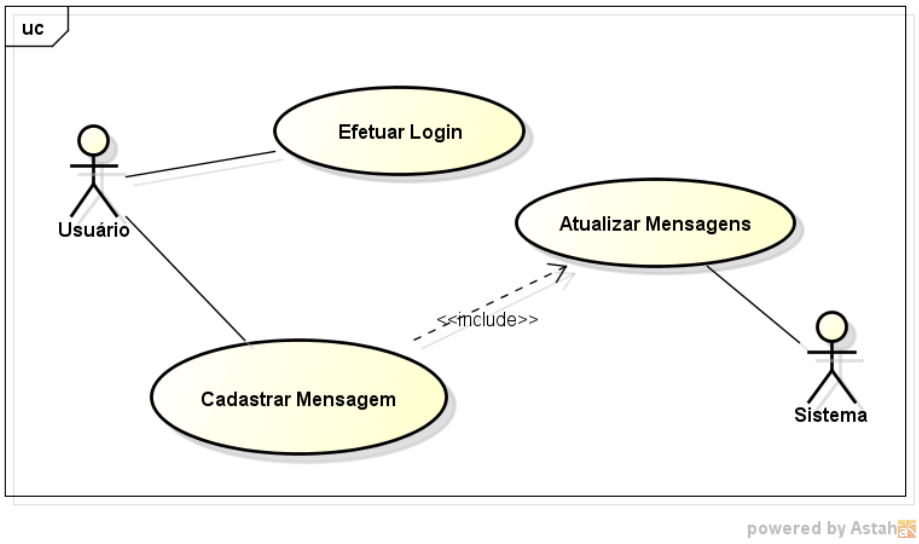
4.3 ESPECIFICAÇÃO

Os diagramas e tabelas de especificação do protótipo foram atualizados com base nas funcionalidades aprimoradas ao protótipo de aplicativo. Somente o diagrama de pacotes manteve-se conforme a primeira versão. Na sequência, cada artefato alterado será apresentado comparando-o com o anterior. Também será apresentado o Modelo Lógico de dados, não construído na primeira versão, pois o protótipo não previa acesso ao banco de dados.

4.3.1 Diagrama de Caso de Uso

No diagrama de caso de uso foi adicionado um novo ator e novas ações, conforme a Figura 18. Ao usuário foi atribuída a realização de *login* no protótipo do aplicativo. O novo ator é o respectivo sistema, e possui vinculada a atualização de mensagens no dispositivo. Esta ação também é complementar ao cadastro de mensagens, realizado pelo usuário.

FIGURA 18 – Diagrama de Caso de Uso



Fonte: AUTOR

4.3.2 Descrição de Casos de Uso e Casos de Testes

Com base no diagrama de caso de uso, a descrição dos casos de uso e casos de testes foi atualizada para contemplar o *login* no protótipo, conforme Tabela 7, o novo funcionamento do cadastro de mensagem, conforme Tabela 8, e a atualização de mensagens no dispositivo, conforme Tabela 9.

TABELA 7 – Descrição de Caso de Uso e Caso de Teste – Efetuar *Login*

Caso de Uso:	**Efetuar *Login***
Finalidade:	Realizar registro do usuário no Simulador
Descrição:	Permite ao usuário registrar-se para o envio e recebimento de

	mensagens
Ator:	Usuário
Pré-Condições:	Acessar o protótipo do Simulador
Fluxo Principal:	1. Usuário acessa o protótipo. 2. Sistema verifica se o Simulador já possui usuário logado no dispositivo. 3. Sistema não localiza usuário logado no Simulador. 4. Usuário informa o número do telefone para acesso. 5. Sistema verifica no servidor se o número já está cadastrado como um usuário. 6. Sistema realiza o *login* do usuário, e direciona para a tela do Simulador.
Fluxo Alternativo:	Fluxo Alternativo A: 3. Sistema localiza usuário logado no Simulador. 4. Sistema direciona o usuário automaticamente para a tela do Simulador. Fluxo Alternativo B: 6. Sistema não localiza o telefone informado no cadastro de usuários do servidor. 7. Sistema apresenta mensagem solicitando que o usuário seja cadastrado. 8. Sistema habilita campo Nome para digitação. 9. Usuário cadastra o nome correspondente ao número de telefone informado. 10. Sistema realiza o cadastro do novo usuário no servidor. 11. Sistema realiza o *login* do usuário no dispositivo, e direciona para a tela do Simulador.
Pós-Condições:	Usuário cadastrado no servidor; Usuário logado no Simulador
Caso de Teste:	**Efetuar *Login***
Pré-Condições:	Acessar o protótipo do Simulador
Testes:	1. Valida a busca de usuário logado no dispositivo.

	2. Valida habilitação dos campos. 3. Valida preenchimento do Telefone. 4. Valida busca de usuário no servidor. 5. Valida preenchimento do Nome do usuário quando necessário. 6. Valida cadastro do usuário no servidor. 7. Valida registro do usuário no dispositivo. 8. Valida redirecionamento da tela de *login* para a tela do Simulador.
Resultado Esperado:	Operações realizadas com sucesso, dados inseridos e *login* do usuário registrado no dispositivo

Fonte: AUTOR

TABELA 8 – Descrição de Caso de Uso e Caso de Teste – Cadastrar Mensagem

Caso de Uso:	**Cadastrar Mensagem**
Finalidade:	Realizar inclusão de mensagem e visualização de mensagens recebidas
Descrição:	Permite usuário realizar cadastro e envio de mensagem
Ator:	Usuário
Pré-Condições:	Usuário autenticado e acesso à tela do Simulador
Fluxo Principal:	1. Usuário informa o texto da mensagem. 2. Usuário realiza o envio da mensagem. 3. Sistema verifica conexão do dispositivo com a *internet*. 4. Sistema envia a mensagem ao servidor. 5. Sistema atualiza mensagens no dispositivo conforme Caso de Uso Atualizar Mensagens.
Fluxo Alternativo:	Fluxo Alternativo A: 2. Usuário configura o tamanho da fonte da mensagem. 3. Usuário altera a cor do texto da mensagem. 4. Usuário acessa a câmera do dispositivo para anexar foto na mensagem. 5. Usuário acessa o álbum do dispositivo para anexar foto

	na mensagem. 6. Usuário acessa imagem anexada para visualização ampliada. 7. Usuário realiza o envio da mensagem. 8. Sistema verifica conexão do dispositivo com a *internet*. 9. Sistema envia a mensagem ao servidor. 10. Sistema atualiza mensagens no dispositivo conforme Caso de Uso Atualizar Mensagens. Fluxo Alternativo B: 4. Sistema identifica que dispositivo não possui conexão com a *internet*. 5. Sistema exibe mensagem de aviso ao usuário, informando que a mensagem não poderá ser enviada no momento.
Pós-Condições:	Mensagem enviada ao servidor; Mensagens atualizadas no dispositivo
Caso de Teste:	**Cadastrar Mensagem**
Pré-Condições:	Usuário autenticado e acesso à tela do Simulador
Testes:	1. Valida habilitação dos campos. 2. Valida preenchimento do texto ou anexo da mensagem. 3. Valida configuração do tamanho de fonte da mensagem. 4. Valida alteração da cor do texto da mensagem. 5. Valida acesso a câmera do dispositivo e opção de anexar imagens. 6. Valida acesso ao álbum do dispositivo e opção de anexar imagens. 7. Valida visualização ampliada da imagem anexada. 8. Valida se o *layout* do protótipo corresponde à orientação de tela do dispositivo e diversos tamanhos de tela, considerando o critério responsividade. 9. Valida se o Simulador verifica corretamente o acesso

	do dispositivo à *internet*. 10. Valida se a mensagem é salva corretamente no servidor.
Resultado Esperado:	Operações realizadas com sucesso e dados cadastrados

Fonte: AUTOR

TABELA 9 – Descrição de Caso de Uso e Caso de Teste – Atualizar Mensagens

Caso de Uso:	**Atualizar Mensagens**
Finalidade:	Disponibilizar visualização de mensagens no dispositivo
Descrição:	Permite ao usuário visualizar conversas já baixadas do servidor
Ator:	Sistema
Pré-Condições:	Usuário autenticado, acesso à tela do Simulador e conexão com a *internet*
Fluxo Principal:	1. Sistema verifica se o dispositivo possui conexão com a *internet*. 2. Sistema exibe ícone informando ao usuário se o dispositivo possui conexão. 3. Sistema solicita ao servidor novas mensagens não baixadas no dispositivo. 4. Sistema atualiza as mensagens localmente no dispositivo e apresenta ao usuário. 5. Sistema repete a tentativa de atualização a cada 30 segundos.
Fluxo Alternativo:	3. Sistema identifica que dispositivo não possui conexão com a *internet*. 4. Sistema não realiza solicitação ao servidor. 5. Sistema repete a tentativa de atualização a cada 30 segundos.
Pós-Condições:	Mensagens atualizadas no dispositivo
Caso de Teste:	**Atualizar Mensagens**
Pré-Condições:	Usuário autenticado, acesso à tela do Simulador e conexão

	com a *internet*
Testes:	1. Valida se a identificação de conectividade do dispositivo está correta. 2. Valida se as mensagens são atualizadas corretamente do servidor para o dispositivo.
Resultado Esperado:	Mensagens atualizadas no dispositivo

Fonte: AUTOR

4.3.3 Protótipo de Tela

Uma nova tela foi criada, representando a realização do *login* do usuário, e pode ser visualizada na Figura 19. A prototipação de tela existente foi atualizada considerando a possibilidade de envio e visualização das mensagens na mesma tela, conforme apresentado na Figura 20. As telas foram criadas baseando-se em elementos do *framework* Bootstrap, que apresentam configurações de responsividade pré-configurados.

FIGURA 19 – Protótipo de Tela – *Login*

Fonte: AUTOR

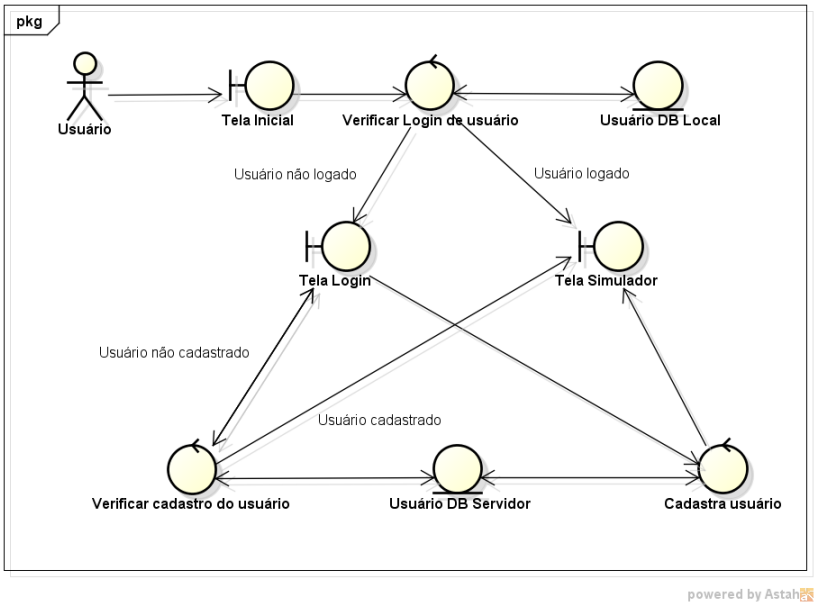
FIGURA 20 – Protótipo de Tela – Simulador

Fonte: AUTOR

4.3.4 Diagrama de Robustez

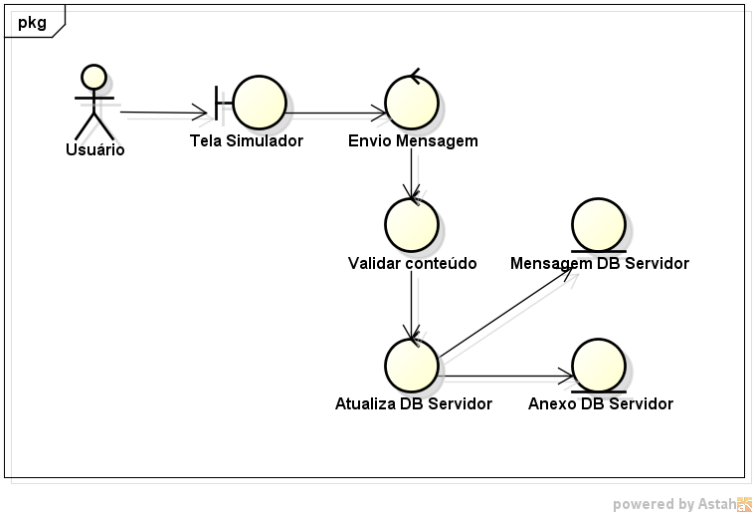
O diagrama de robustez foi atualizado para representar o *login* do usuário e as iterações do Simulador com o servidor de dados. O diagrama da Figura 21 representa o detalhamento do *login*, a Figura 22 representa o detalhamento do envio de mensagens para o servidor e a Figura 23 representa a atualização de mensagens no dispositivo.

FIGURA 21 – Diagrama de Robustez – *Login*



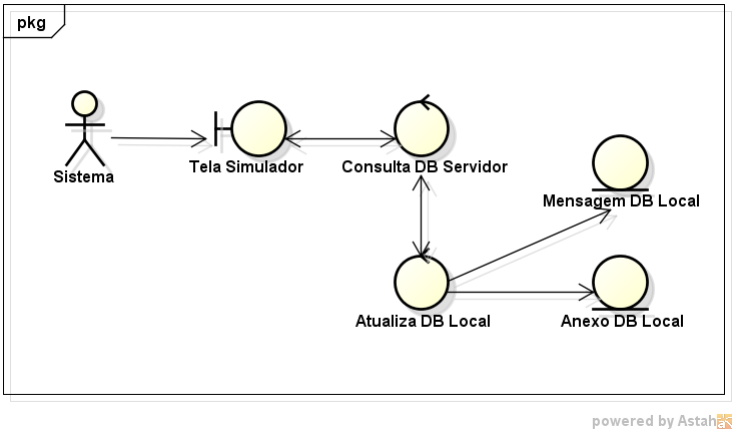
Fonte: AUTOR

FIGURA 22 – Diagrama de Robustez – Envio de mensagem



Fonte: AUTOR

FIGURA 23 – Diagrama de Robustez – Atualização de mensagens

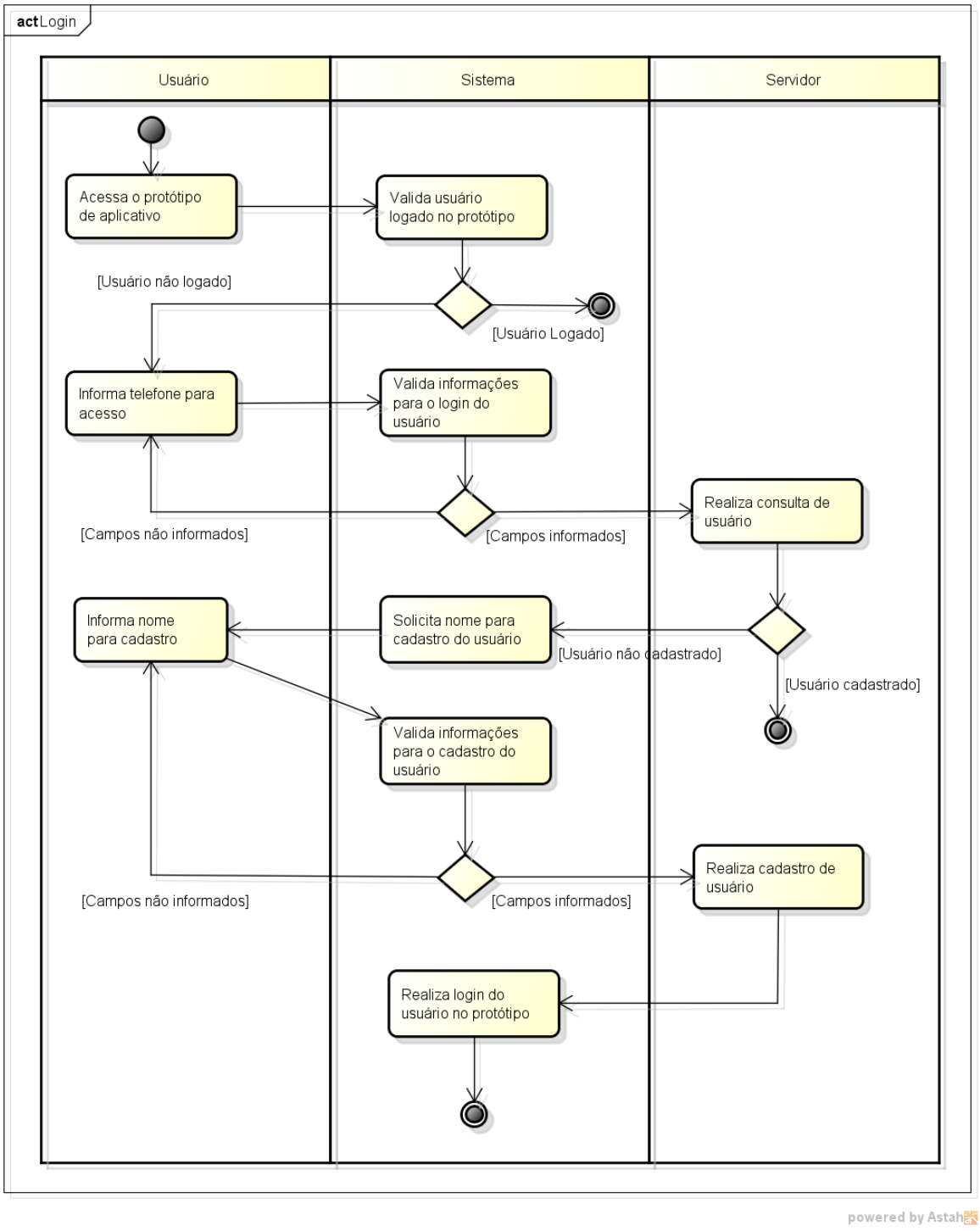


Fonte: AUTOR

4.3.5 Diagrama de Atividades

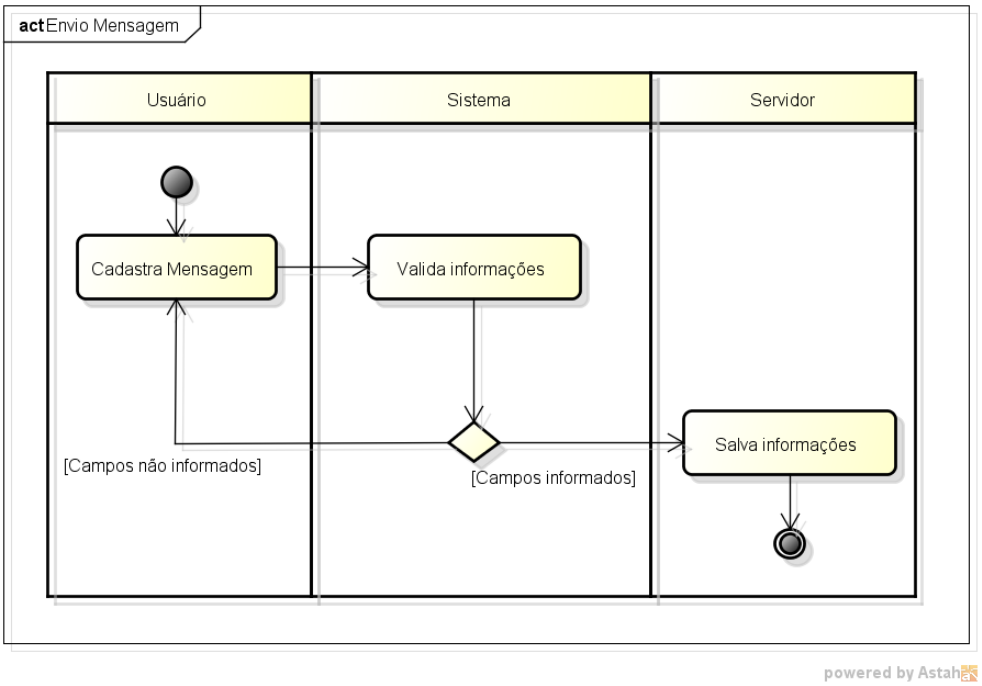
Assim como o diagrama de robustez, o diagrama de atividades da nova versão foi complementado em três novos diagramas, cada um representando ações distintas no protótipo de aplicativo. Novos diagramas foram criados com base nos fluxos de *login*, conforme Figura 24, e atualização de mensagens, conforme Figura 26. O diagrama existente para envio de mensagem foi atualizado conforme Figura 25, prevendo a comunicação com o servidor de dados.

FIGURA 24 – Diagrama de Atividades – *Login*



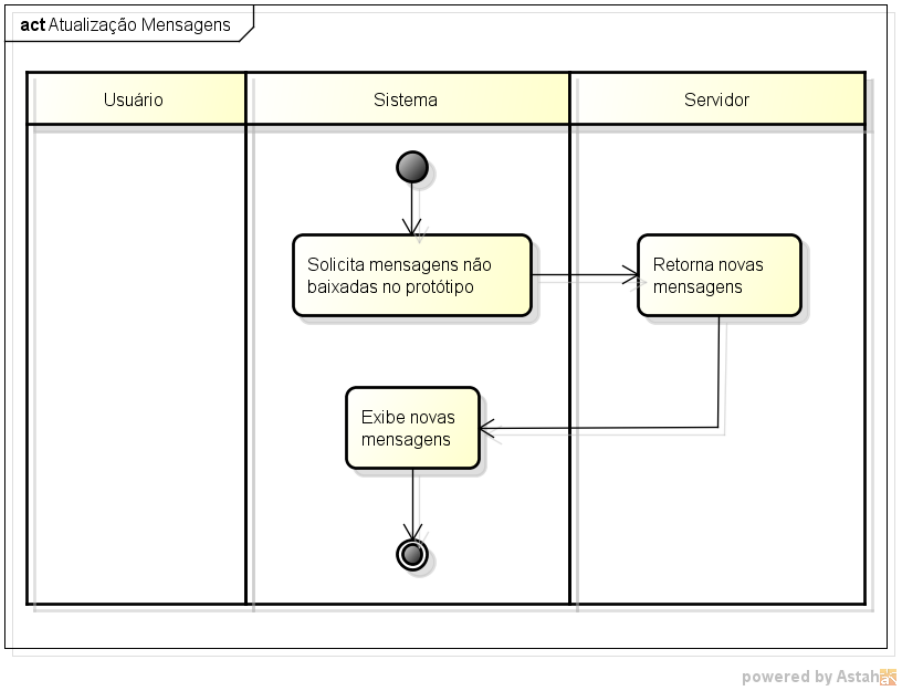
Fonte: AUTOR

FIGURA 25 – Diagrama de Atividades – Envio de mensagem



Fonte: AUTOR

FIGURA 26 – Diagrama de Atividades – Atualização de mensagens

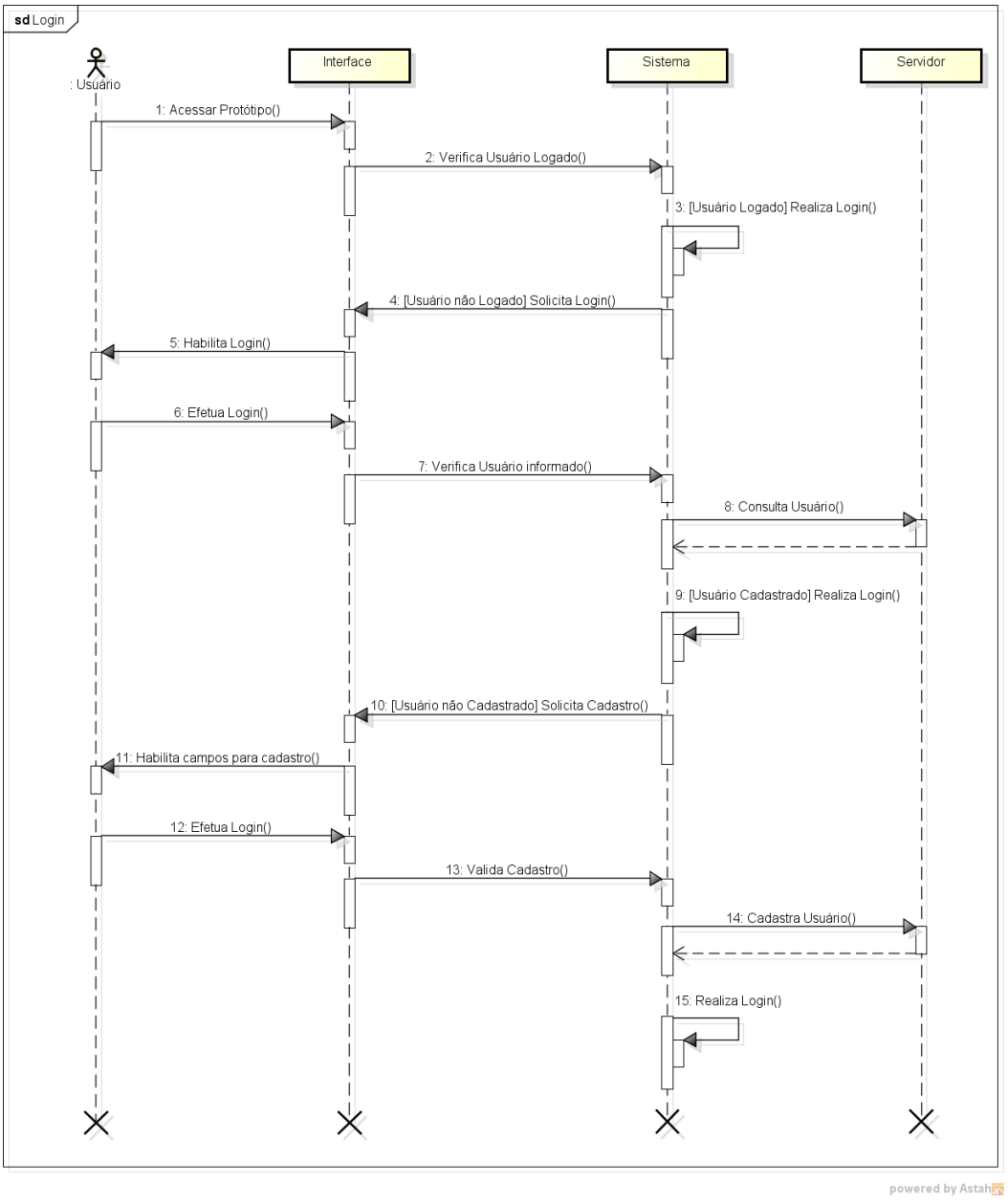


Fonte: AUTOR

4.3.6 Diagrama de Sequência

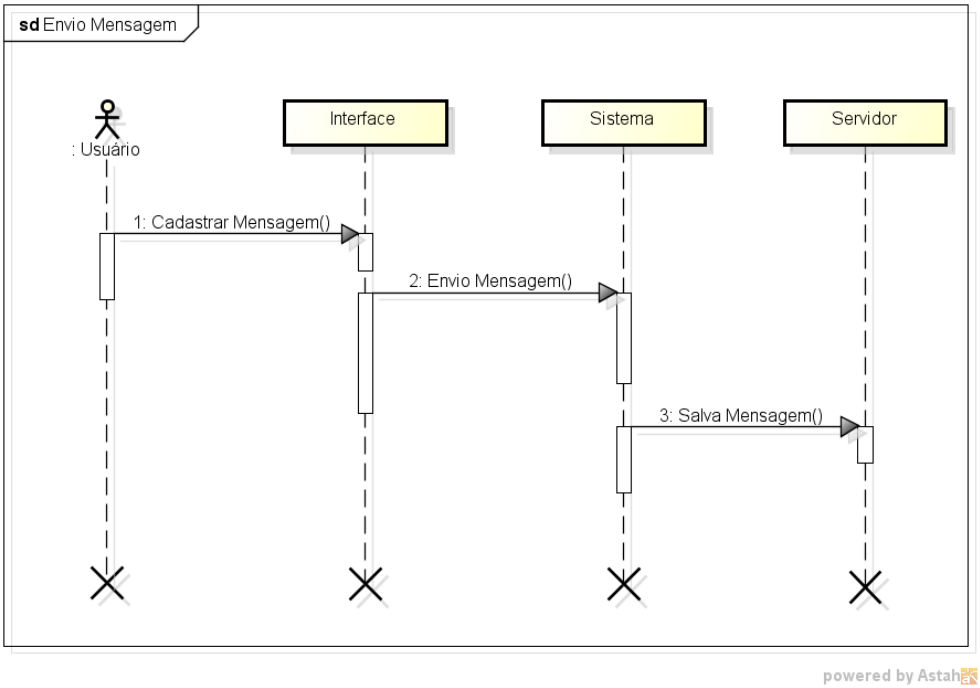
O diagrama de sequência criado para a primeira versão do protótipo foi utilizado com a mesma finalidade, o envio de mensagem. Assim como os demais artefatos, este foi complementado com a comunicação com o servidor, conforme Figura 28. Para as ações de *login* e atualização de mensagens, foram criados os diagramas disponibilizados nas Figuras 27 e 29 respectivamente.

FIGURA 27 – Diagrama de Sequência – *Login*



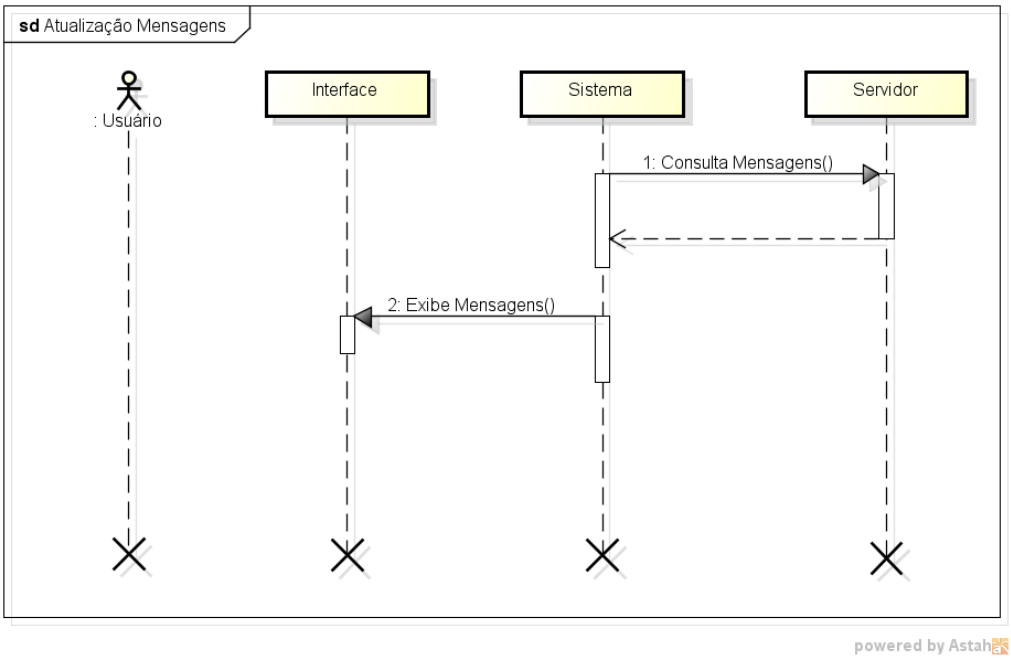
Fonte: AUTOR

FIGURA 28 – Diagrama de Sequência – Envio de mensagem



Fonte: AUTOR

FIGURA 29 – Diagrama de Sequência – Atualização de mensagens

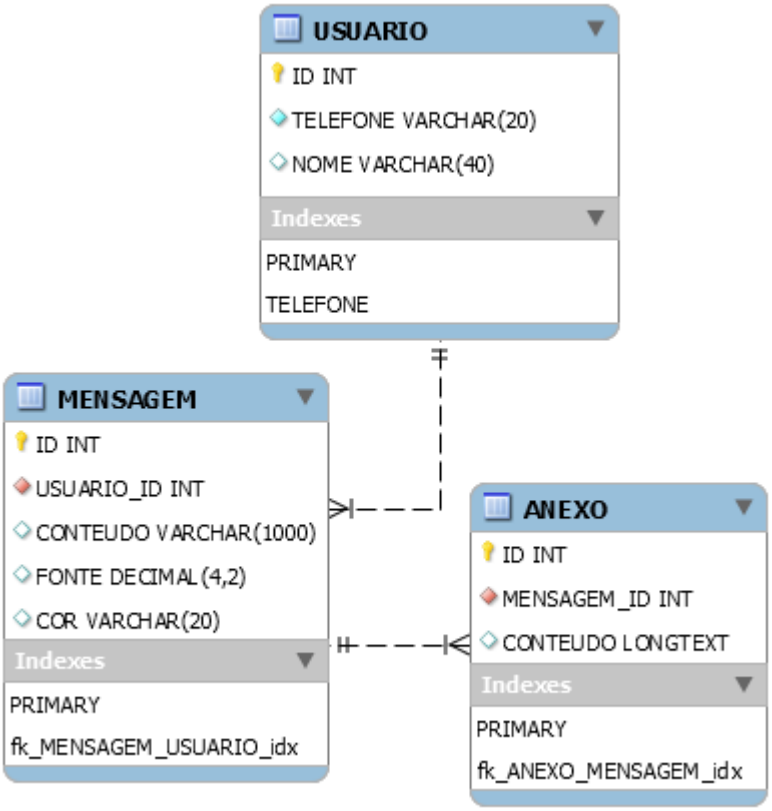


Fonte: AUTOR

4.3.7 Modelo Lógico de dados

O modelo lógico descreve o relacionamento entre tabelas em um banco de dados, bem como a estrutura de atributos e índices de cada tabela. A Figura 30 representa o Modelo Lógico do banco de dados criado como servidor do protótipo de aplicativo. Internamente em cada dispositivo, o protótipo utiliza banco de dados local com estrutura semelhante à apresentada. Este modelo lógico de dados foi criado com a utilização da ferramenta MySQL Workbench 6.2.

FIGURA 30 – Modelo Lógico de dados



Fonte: AUTOR

4.4 VALIDAÇÃO DO PROTÓTIPO

A validação da segunda versão do protótipo de aplicativo foi baseada nas qualidades de *software* apresentadas anteriormente. Assim como na primeira versão do protótipo, o desenvolvimento foi baseado na plataforma Android, e após realizada a compilação para as

demais plataformas. A seguir serão apresentadas particularidades identificadas em cada plataforma durante a validação.

4.4.1 Android

Na avaliação da segunda versão do protótipo de aplicativo no Android, foram utilizados os dispositivos emulados Nexus 7 e Nexus 10 (*Tablet*), com a versão 4.3.1 (API Level 18) da plataforma. Foi necessária a alteração da versão da plataforma para o funcionamento da câmera do emulador.

Nos testes realizados, analisando os requisitos de qualidade definidos para avaliação, foram obtidos os resultados esperados. Um detalhe a ser observado, é mesmo o *layout* tendo sido desenvolvido com técnicas de responsividade, a orientação horizontal do dispositivo apresenta pequena diferença no comportamento dos objetos na tela. O mesmo acontece com o protótipo nas demais plataformas.

4.4.2 iOS

Para plataforma iOS foi utilizada a mesma configuração de ambiente na validação da primeira versão do protótipo. O protótipo de aplicativo foi validado nos dispositivos iPhone 6.1 e iPad 6.1, por meio do iOS Simulator.

Durante dos testes realizados, foram identificadas falhas no acesso à palete de cores e câmera do dispositivo. A falha ao acesso da câmera também esteve presente na primeira validação do protótipo. Tal situação foi entendida como uma limitação do iOS Simulador, pois também não existe menu nativo para acesso à câmera. Já a palete de cores obteve funcionamento correto na primeira implementação. Acredita-se que a utilização do Bootstrap tenha influenciado no seu funcionamento.

Uma limitação do PhoneGap influenciou o teste de conectividade do dispositivo, pois o mesmo não reconhece mudanças de conectividade para o iOS. Assim, durante os testes realizados, a conectividade do dispositivo foi sempre identificada como *wi-fi*. Esta situação não prejudica o funcionamento do protótipo, pois as ações relacionadas com conectividade possuem o tratamento necessário, mas acaba omitindo a informação prévia ao usuário.

4.4.3 Windows Phone

Os testes do Windows Phone foram realizados em um dispositivo emulado na versão 7.8 da plataforma, com 512 MB de memória interna. O desenvolvimento e validação do protótipo nesta plataforma foi o de maior quantidade de particularidades em relação às demais plataformas analisadas. Em algumas situações, tais particularidades criaram a necessidade de busca de soluções alternativas, como o caso do banco de dados e troca de dados com o servidor.

O tratamento do banco de dados interno foi realizado de forma diferente das demais plataformas. O Android e o iOS possuem suporte à API de banco de dados WebSQL, uma variável do banco de dados SQL. Já o Windows Phone 7 somente possui suporte ao armazenamento de dados LocalStorage. O LocalStorage funciona como uma sessão de usuário em navegadores, com a diferença de ter maior permanência de armazenamento. Sua capacidade de dados armazenados está diretamente ligada à memória do dispositivo. Por este motivo, a visualização de mensagens para esta plataforma foi alterada, onde o protótipo passa a listar somente as 10 últimas mensagens recebidas do servidor. A visualização das imagens recebidas também foi retirada devido à quantidade de dados a serem armazenados.

A troca de dados com servidor necessitou ser reavaliada devido a uma restrição da plataforma. Enquanto as outras plataformas suportam troca de mensagens via JSON, o Windows Phone somente suporta o formato JSONP. Como as demais plataformas também reconhecem este formato, a comunicação foi mantida em JSONP. Outra particularidade no Windows Phone é a necessidade de informação de um parâmetro extra ao enviar dados para o servidor, via método POST. Esta situação é um *bug* documentado do jQuery, quando utilizado em conjunto com o PhoneGap e Windows Phone 7.

No Windows Phone, assim como no iOS, o *plugin* do PhoneGap para teste de conectividade não reconheceu a mudança de *status* do dispositivo emulado.

4.5 AVALIAÇÃO DO PROTÓTIPO

A seguir será apresentado o detalhamento das características de *software* avaliadas. Foi realizada análise qualitativa em cada plataforma, considerando o atendimento das qualidades definidas juntamente com a portabilidade do protótipo. A plataforma Android

apresentou o resultado esperado em todas as características. A Figura 31 apresenta o protótipo nesta plataforma.

FIGURA 31 – Simulador de Mensagens na plataforma Android

Fonte: AUTOR

4.5.1 Interface e Características Estéticas

A característica de interface e características estéticas engloba itens referentes ao posicionamento de ícones, registro de mensagens e ampliação das imagens. A plataforma Android obteve completo atendimento das necessidades, nas demais plataformas houve particularidades que tornaram os itens atendidos parcialmente ou não aplicáveis.

4.5.1.1 Posicionamento de ícones

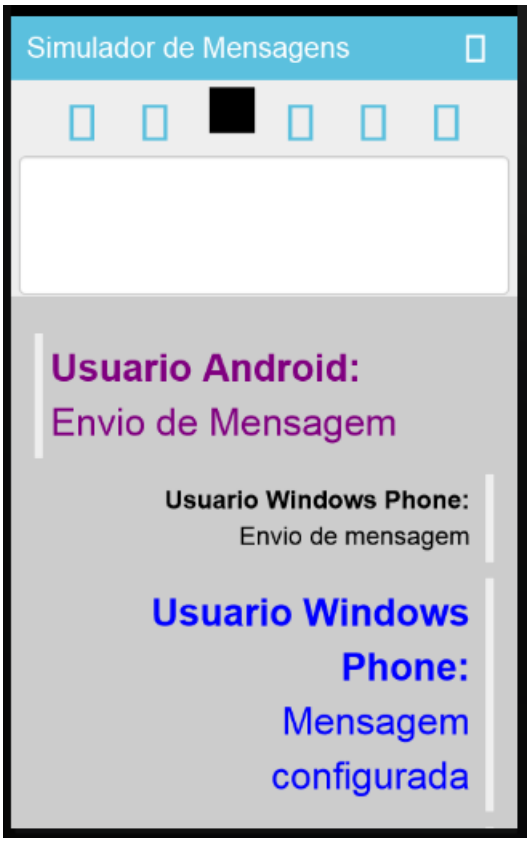
O posicionamento de ícones foi avaliado quanto ao seu agrupamento no início da página. Os mesmos devem permanecer fixos ao cabeçalho do protótipo, mesmo ao realizar a

rolagem do registro de mensagens. Os ícones foram criados por meio de elementos do Bootstrap.

Na plataforma iOS, o posicionamento teve seu funcionamento alterado quando o teclado do dispositivo está presente. Neste cenário, o cabeçalho deixa de ser fixo no topo da tela e se movimenta conforme a rolagem do registro de mensagens, tornando o item parcialmente atendido.

Na validação realizada no Windows Phone, os ícones não foram reconhecidos, conforme Figura 32. Em relação à rolagem da tela, o cabeçalho sempre moveu-se junto com o registro das mensagens. Considerando que os ícones permaneceram agrupados ao cabeçalho, o item foi considerado como parcialmente atendido.

FIGURA 32 – Visualização de ícones na plataforma Windows Phone



Fonte: AUTOR

4.5.1.2 Registro de Mensagens

O funcionamento esperado do registro de mensagens é a diferenciação de mensagens recebidas e enviadas. Este controle foi realizado utilizando o sistema de tratamento de colunas do Bootstrap e foi atendido em todas as plataformas.

4.5.1.3 Ampliação de Imagens

O quesito ampliação de imagens foi dividido entre a ampliação das imagens anexadas para envio e a ampliação de imagens apresentadas no registro de mensagens. A ampliação de imagens anexadas não pode ser validada na plataforma iOS, pois a mesma não apresenta a opção de acesso à câmera no simulador. Por este motivo a funcionalidade foi indicada como não se aplica. Nas demais plataformas o item foi atendido.

Já no Windows Phone, a ampliação de imagens no registro de mensagens não pode ser validada devido à limitação de memória do emulador. Para reduzir a quantidade de dados armazenados no dispositivo, as imagens não foram apresentadas no registro de mensagens.

4.5.2 Domínio da Aplicação

A característica de domínio da aplicação engloba as funcionalidades de configuração do texto da mensagem e inclusão de anexos.

4.5.2.1 Configuração de Texto da Mensagem

A configuração do texto da mensagem engloba a alteração de tamanho e cor de fonte em seu conteúdo. Neste quesito, somente o quesito de alteração de cor de fonte não foi atendido, na plataforma iOS, devido à incompatibilidade com a palete de cores.

4.5.2.2 Inclusão de Anexo

A inclusão de anexos foi validada considerando as opções de acesso à câmera e álbum do dispositivo. As duas opções foram atendidas nas plataformas Android e Windows Phone. Na plataforma iOS não foi possível realizar o acesso à câmera, devido à limitações do simulador, tornando o item não atendido. Considerando esta limitação, não foi possível incluir imagens no álbum do dispositivo, tornando o item de inclusão via álbum não aplicável na plataforma.

4.5.3 Validação e Recuperação de entrada de usuário

Na característica de validação e recuperação de entrada de usuário foram validados os itens de obrigatoriedade de conteúdo na mensagem e o recebimento de informações corretas, de forma a manter a confiabilidade e integridade dos dados apresentados no protótipo. Os itens foram atendidos em todas as plataformas, onde não foi possível enviar mensagens sem conteúdo ou anexo, e as informações foram apresentadas corretamente no registro de mensagens.

4.5.4 Desempenho do tempo de resposta

A característica de desempenho do tempo de resposta foi analisada separadamente considerando o desempenho interno do protótipo, e a comunicação do protótipo com o servidor. As ações internas, como rolagem das mensagens já baixadas no dispositivo, e velocidade de abertura da visualização das imagens, obtiveram resultados satisfatórios, onde a resposta do protótipo foi imediata em todas as plataformas.

O tempo de resposta em relação com a comunicação de dados com o servidor mostrou-se variável durante a execução do protótipo. A variação acontece principalmente em relação à quantidade de dados trafegados, ligados também à velocidade de conexão disponível com a *internet*. Desta forma, quando enviado anexo na mensagem, ou houver um volume maior de mensagens a serem baixadas do servidor, o aplicativo pode perder performance. Esta situação foi percebida nos protótipos de todas as plataformas, mas o Android foi a que

apresentou a maior perda de performance em relação às demais. Por este requisito variar conforme fatores externos, foi considerado como atendido.

Analisando o comportamento do protótipo de aplicativo nas plataformas estudadas, bem como as particularidades de cada plataforma mencionadas anteriormente, criou-se um quadro para visualização de cada requisito específico, conforme Tabela 10.

Este quadro está organizado por item de qualidade de *software* e plataforma. Para cada item, foi atribuído um valor de atendimento. Os valores possíveis são (A) Atendido, (AP) Atendido Parcialmente, (N) Não Atendido e (NA) Não se Aplica. O valor (NA) Não se Aplica, foi atribuído quando a plataforma não disponibiliza o item para validação.

TABELA 10 – Características de qualidade de *software* por plataforma

	Plataformas		
Qualidades de *Software*	**Android**	**iOS**	**Windows Phone**
Usabilidade			
Interface e Características Estéticas			
Posicionamento de Ícones	A	AP	AP
Registro de Mensagens	A	A	A
Ampliação de Imagens			
Anexo para Envio	A	NA	A
Registro de Mensagens	A	A	NA
Funcionalidade			
Domínio da Aplicação			
Configuração de Texto da Mensagem			
Tamanho da Fonte	A	A	A
Cor da Fonte	A	N	A
Inclusão de Anexo			
Câmera	A	N	A
Álbum	A	NA	A
Confiabilidade			
Validação e Recuperação de entrada de usuário			
Obrigatoriedade de conteúdo na mensagem	A	A	A
Recebimento de informações corretas	A	A	A
Eficiência			
Desempenho do tempo de resposta			
Interno	A	A	A
Comunicação com o servidor	A	A	A

Fonte: AUTOR

5 CONCLUSÃO

Com base nos estudos realizados ao longo deste trabalho, pode-se concluir que o desenvolvimento de aplicativos híbridos utilizando ferramenta multiplataforma é válido. Grande parte das funcionalidades analisadas mostraram-se compatíveis nas plataformas analisadas. Em alguns casos houve a necessidade de localização de opções alternativas para o completo atendimento de um requisito nas plataformas. Tais alternativas não oneram o processo de desenvolvimento. Apesar de apresentarem maior custo de tempo em relação à implementação nativa, a codificação é realizada somente uma vez.

A ferramenta PhoneGap possui diversos *plugins* que fornecem suporte a grande parte de funcionalidades nativas. Em alguns casos, estes *plugins* ainda não atendem plenamente todas as plataformas, como é caso do *plugin* para teste de conectividade do dispositivo. Com o lançamento de novas versões da ferramenta, espera-se que os mesmos sejam aprimorados e o atendimento seja pleno. Além disso, há a possibilidade de criação de *plugins* próprios, que podem ser uma alternativa válida antes da realização do desenvolvimento nativo.

Um ponto a ser considerado ao desenvolver um aplicativo com o PhoneGap, é o desempenho esperado do aplicativo. Por ser uma solução com foco na portabilidade, pode ocorrer perda de velocidade de processamento. Fatores como quantidade de dados trafegados, acessos externos ao aplicativo e velocidade de conexão do dispositivo com a *internet* influenciam diretamente o desempenho.

No desenvolvimento híbrido, a criação de telas do aplicativo pode ser um contraponto para quem não possui familiaridade com as linguagens base do *layout*, HTML e CSS. A utilização de uma ferramenta de desenvolvimento *front-end* facilita este processo. Neste trabalho foi utilizado o *framework* Bootstrap. O mesmo mostrou-se compatível com as plataformas Android e iOS, porém no Windows Phone não se obteve o resultado esperado, na qual os ícones não foram reconhecidos. Caso seja necessário disponibilizar um aplicativo nesta plataforma, cabe avaliar sua compatibilidade com outras ferramentas.

Como sugestões de melhoria e continuidade deste trabalho, há a possibilidade de agregar novas funcionalidades ao envio das mensagens; suporte a outros formatos de arquivos no anexo da mensagem; opção de exclusão de imagens recebidas para liberação de memória no dispositivo; envio de mapas por meio de geolocalização; criação de parâmetro para configuração do temporizador de novas mensagens; e complemento do cadastro do usuário.

6 REFERÊNCIAS BIBLIOGRÁFICAS

ADOBE. Disponível em: <http://www.adobe.com/devnet/devices/mobile-apps.html>. Acesso em 18/05/2014.

APPCELERATOR. Disponível em: <http://www.appcelerator.com/titanium/>. Acesso em 18/05/2014.

APPLE. **iOS 7**. Disponível em <http://www.apple.com/br/ios/>. Acesso em 31/03/2014.

BOOTSTRAP. Disponível em: <http://getbootstrap.com/>. Acesso em 18/11/2014.

CAELUM. **Desenvolvimento Web com HTML, CSS e JavaScript.** Disponível em: <http://www.caelum.com.br/apostila-html-css-javascript/>. Acesso em 25/08/2014.

CHARLAND, Andre; LEROUX, Brian. **Mobile Application Development: Web vs. Native.** Disponível em <http://queue.acm.org/detail.cfm?id=1968203>. Acesso em 03/05/2014.

CORONA. Disponível em: <http://coronalabs.com/>. Acesso em 16/05/2014.

FERREIRA, Elcio; EIS, Diego. **HTML5: Curso W3C Escritório Brasil**. Disponível em: <http://www.w3c.br/pub/Cursos/CursoHTML5/html5-web.pdf>. Acesso em 27/04/2014.

GIT. Disponível em: <http://git-scm.com/>. Acesso em 10/05/2014.

GOLDSTEIN, Alexis; LAZARIS, Louis; WEYL, Estelle. **HTML5 & CSS3 for the Real World**. Austrália: SitePoint, 2011.

GOOGLE. **Android**. Disponível em <http://www.android.com/>. Acesso em 31/03/2014.

HARRIS, Andy. **HTML5 For Dummies.** Indianapolis, Indiana: Wiley Publishing, Inc., 2011.

HOSTINGER. Disponível em: <http://www.hostinger.com.br/>. Acesso em 18/11/2014.

JÚNIOR, Venilton Falvo et al. **Uma comparação do tempo de implementação: Android vs. HTML5** in CIbSE 2013 - X Workshop Latinoamericano Ingeniería de *Software* Experimental - ESELAW 2013, 2013.

JUNTUNEN, Antero; JALONEN, Eetu; LUUKKAINEN, Sakari. **HTML 5 in Mobile Devices – Drivers and Restraints** in 46th Hawaii International Conference on System Sciences, 2013.

LAUDON, Kenneth; LAUDON, Jane. **Sistemas de informação gerenciais**. 9. ed. São Paulo: Pearson Prentice Hall, 2010.

LEE, Valentino; SCHNEIDER, Heather; SCHELL, Robbie. **Aplicações móveis: arquitetura, projeto e desenvolvimento**. São Paulo: Pearson Education do Brasil, 2005.

LIONBRIDGE. **Mobile web apps vs. mobile native apps: How to make the right choice**. Disponível em <http://www.lionbridge.com/files/2012/11/Lionbridge-WP_MobileApps2.pdf>. Acesso em 04/05/2014.

MARMALADE. Disponível em: <https://www.madewithmarmalade.com/>. Acesso em 16/05/2014.

MATEUS, Geraldo Robson; LOUREIRO, Antonio Alfredo Ferreira. **Introdução à Computação Móvel** in XI Escola de Computação - Rio de Janeiro, 1998. Disponível em <http://homepages.dcc.ufmg.br/~loureiro/cm/docs/cm_livro_1e.pdf>. Acesso em 07/05/2014.

MAZZA, Lucas. **HTML5 e CSS3: Domine a *web* do futuro**. São Paulo: Casa do Código, 2012.

MICROSOFT**. Windows Phone.** Disponível em <http://www.windowsphone.com/pt-br>. Acesso em 05/04/2014.

MONO. Disponível em: <http://mono-project.com/Main_Page>. Acesso em 16/05/2014.

MOSYNC. Disponível em: <http://www.mosync.com/>. Acesso em 25/05/2014.

NODE.JS. Disponível em: <http://nodejs.org/>. Acesso em 10/05/2014.

PHONEGAP. Disponível em: <http://phonegap.com/>. Acesso em 10/05/2014.

PRESSMAN, Roger S.. **Engenharia de *software***. 6. ed. São Paulo: McGraw-Hill, 2006.

QT DIGIA. Disponível em: <http://qt.digia.com/>. Acesso em 16/05/2014.

ROCHA, Ana Regina Cavalcanti da; MALDONADO, José Carlos; WEBER, Kival Chaves. **Qualidade de *Software*: Teoria e Prática**. São Paulo: Prentice Hall, 2001.

ROLNITZKY, David. **To mobile web app or not to mobile web app?** Disponível em: <http://www.rolnitzky.com/artifacts/mobile-v-web-app.pdf>. Acesso em 03/05/2014.

SENCHA TOUCH. Disponível em: <http://www.sencha.com/products/touch/>. Acesso em 11/05/2014.

SMUTNÝ, Pavel. **Mobile development tools and cross-platform solutions** in 13th International Carpathian Control Conference (ICCC), 2012.

TANENBAUM, Andrew S.. **Sistemas operacionais Modernos.** 3. ed. São Paulo: Pearson Prentice Hall, 2009.

UNIT. Disponível em: <http://unity3d.com/>. Acesso em 16/05/2014.

VISION MOBILE. **Cross-platform Developer Tools 2012.** Disponível em: <http://www.visionmobile.com/product/cross-platform-developer-tools-2012/>. Acesso em 21/05/2014.

VISION MOBILE. **Developer Economics 2013.** Disponível em: <http://www.visionmobile.com/product/developer-economics-2013-the-tools-report/>. Acesso em 18/05/2014.

VISION MOBILE. **Developer Economics Q1 2014.** Disponível em: <http://www.developereconomics.com/reports/q1-2014/>. Acesso em 18/05/2014.

W3C Brasil. **CSS: Curso W3C Escritório Brasil**. Disponível em: <http://www.w3c.br/pub/Cursos/CursoCSS3/css-web.pdf>. Acesso em 27/04/2014.

W3C Recommendation. **Media Queries.** Disponível em: <http://www.w3.org/TR/css3-mediaqueries/>. Acesso em 31/08/2014.

ZEMEL, Tárcio. **Web Design Responsivo: Páginas adaptáveis para todos os dispositivos.** São Paulo: Casa do Código, 2012.

ANEXO

ANEXO A – Opções de ferramentas multiplataforma disponibilizadas na pesquisa Cross-platform Developer Tools 2012 (VISION MOBILE, 2014c).

Adobe (AIR)	The Dojo Foundation (dojo toolkit)	The jQuery Project (jQuery Mobile)	RunRev (Livecode)
Adobe (Flex)	Seregon (DragonRad)	Kony (KonyOne Platform)	Sencha (Touch, jQtouch)
Innaworks (Alchemo)	Elements Interactive Mobile (EDGELIB)	Vexed Digital (Kirin, NB FOSS project)	Stonetrip (ShiVa3D)
Antenna Software (Mobility Studio)	Emo-Framework.com	Kyros (Velocity)	SIO2 Interactive (SiO2 Engine)
Antix Labs (Games Development Kit)	Enough Software (J2ME Polish)	Digital Fruit (Lime JS)	Mobinex Inc (Smartface Platform)
The Game Creators Ltd (App Game Kit)	Job and Esther Technologies Ltd (Eqela)	Service2Media (App Lifecycle Platform)	Spot Specific
Appcelerator (Titanium)	Expanz (Expanz Platform)	Didmo (Magmito)	StackMob
Geniem (Appever)	FeedHenry	Ideaworks3D Ltd (Marmalade)	Facebook (Strobe, Sproutcore)
Application Craft	SevenVal, YOC Group (Fitml.com)	Zipline Games (Moai)	Sybase (UnWired Platform)
AppMobi	Lifecycle Mobile (Fivespark)	Mobile Nation (MobileNationHQ)	Pancoda (The M Project)
Apps-Builder	Gamebuilder Inc. (Gamebuilder Studio)	Xamarin (MonoTouch, Mono for Android)	Deutsche Telekom (The Unify Project)
UX Plus Inc. (Aqua Platform)	GameSalad Inc (GameSalad)	MoSync	Exadel (Tiggr, now Tiggzi)
Battery Powered Games (BatteryTech)	Artech (GeneXus)	NeoMades (NeoMAD)	SuperWaba (TotalCross)
Software AG (Bedrock)	Gideros Mobile	Netbiscuits	Unity Technologies (Unity)
Backelite (BKrender)	SpringSource, VMWare (Grails, SpringMVC)	Octomobi	Unreal (Unreal Engine)
Qualcomm (BREW)	HaxeNME	OpenText (Mobile Wave Platform)	Uxebu (Bikeshed)
Brightcove (App Cloud)	iBuildapp Inc (iBuild App)	Oracle (ADF)	Uxebu (Aparrat.io)
Department of Behaviour and Logic (Cabana)	ITR Mobility (iFactr)	Papaya Mobile (Social Game Engine)	Vaadin
Canappi	Edhouse (IPFaces)	Adobe (PhoneGap Build)	Trigger Corp (Trigger.io)
Cellsdk.com	Radical Breeze (Illuminations)	Sideshow NetQuest (Proto.io)	IBM (Worklight)
Cocos2D	PhobosLab (impact.js)	Verivo Software (ex Pyxis)	wxWidgets
Conduit Ltd (Conduit Mobile)	FlexyCore (In-the-box)	Nokia (Qt)	XMLVM
Ansca Mobile (Corona)	iUI	Quickconnect Family	XUI.js
CoStore (Pixelspark)	JMango	Red Foundry	++ Technologies (XPower++)
DHTMLX (Touch)	Jo App	Motorola, Solutions (RhoMobile)	YoYo Games (YoYo Games Maker)