UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

DESENVOLVIMENTO DE UM PORTAL FACILITADOR AO ACESSO A CONTEÚDOS DA INTERNET MÓVEL

PEDRO COVOLAN BACHIEGA

Florianópolis
2009

PEDRO COVOLAN BACHIEGA

DESENVOLVIMENTO DE UM PORTAL FACILITADOR AO ACESSO A CONTEÚDOS DA INTERNET MÓVEL

Monografia apresentada ao Departamento de Informática e Estatística da Universidade Federal de Santa Catarina, como requisito parcial para disciplina INE5328 – Projeto em Ciência da Computação II.

Orientador: Profª. Patricia Plentz

Banca: Prof°. Frank Augusto Siqueira

Banca: Profª. Patricia Vilain

Florianópolis

2009

Agradecimentos

A minha família;

A UFSC;

A todos que fazem parte ou já passaram pela Praesto Convergence, em especial a Anderson Nielson, Eric Santos, Guilherme Lopes, Lucas Torri, Bruno Ghisi, Douglas Mendes, Diego Pereira, Caio Fonseca eTiago Cruz;

A todos meus amigos e colegas que estiveram comigo durante minha vida acadêmica.

A minha orientadora Profª. Patricia Plentz pela ajuda na formalização e documentação deste trabalho.

SUMÁRIO

RESUMO

O crescimento da Internet Móvel se dá tanto na diversidade de conteúdos disponíveis (sites) como em número de acessos a estes. Este fato possibilita o estudo de muitos aspectos relacionados à Internet Móvel. Este trabalho propõe o desenvolvimento de um portal na internet que apresenta estes conteúdos e permite a criação de uma página pessoal personalizada para dispositivos móveis através de *web services*. A arquitetura do sistema foi elaborada a partir do paradigma de Orientação a Objetos e dos conceitos MVC e REST para *web services*. A linguagem de programação utilizada é Java e a persistência dos dados é realizada com o banco de dados MySQL. O desenvolvimento deste sistema dá ao graduando a oportunidade de conhecer o cenário da internet para celulares e dispositivos móveis no Brasil, bem como mostrar algumas das tecnologias e conceitos aprendidos durante a vida acadêmica.

1 INTRODUÇÃO

A facilidade que a internet trouxe para a realização de diversas atividades, o alcance a vasta quantidade de conteúdos, juntamente da popularização dos dispositivos móveis, impulsionou o desenvolvimento da internet para estes dispositivos, a chamada Internet Móvel.

A criação de aplicações para esta área foi possível com o avanço nas tecnologias usadas nos dispositivos móveis, tanto em hardware (capacidade de processamento, de armazenamento, etc.) quanto em software (*midlewares*, sistema operacional, linguagens de programação, etc.).

Com os celulares sendo bem distintos dos computadores em vários aspectos e mesmo apesar do desejo de trazer a internet "completa" para estes pequenos dispositivos, uma internet diferente foi criada. Novas tecnologias foram desenvolvidas, sites diferentes tiveram de ser feitos e até endereços diferentes são utilizados hoje.

Mesmo com estas diferenças de tecnologias e meios, mas com a percepção da Internet como ferramenta essencial na vida das pessoas, viu-se a oportunidade para a criação de um portal facilitador do uso da Internet Móvel para os usuários do celular.

O objetivo deste trabalho é a criação de um portal facilitador de acesso a uma base de conteúdos para celulares. Os usuários podem navegar por esta base de *mobile sites* diretamente no celular ou escolher através de uma interface específica os seus preferidos, depois *web services* vão consumir estas informações para disponibilizar uma página adaptada para celulares com os conteúdos selecionados pelo próprio usuário.

1.1 CONTEXTUALIZAÇÃO

O crescimento dos portais colaborativos mantidos pela comunidade (como Wikipédia[1], por exemplo), dos blogs pessoais, das redes sociais, dos meios de comunicação e informação digital tem feito com que as pessoas utilizassem mais tempo de seu dia em frente ao computador navegando na internet, seja lendo notícias, conversando com outras pessoas e/ou utilizando serviços de todo tipo.

A Internet foi descoberta não só como um meio para entretenimento e acesso a informações gerais, mas também como meio para acesso a informações pessoais e

1 http://en.wikipedia.org/wiki/

particulares a qualquer hora e lugar que se esteja, bem como fazer atividades diversas e usar serviços do dia-a-dia.

Estas novas abordagens no uso da internet estão causando uma grande migração de conteúdos e serviços para dentro dela e deixando de lado os meios e rede locais costumadamente utilizados em casa, no trabalho ou na escola, o que acaba possibilitando o acesso de todo e qualquer lugar, livre de plataforma. Trata-se da chamada *Cloud Computing* (Computação nas Nuvens, em português).

Cloud Computing é um estilo de computação que trabalha sobre a Internet para prover serviços e recursos, onde os usuários não precisam ter conhecimento sobre as tecnologias ou estrutura suportadas. São exemplos de conceitos que utilizam desse paradigma: *Software as a Service* (SaaS), *Infrastructure as a Service* (IaaS) e *Plataform as a Service* (PaaS) (WIKIPÉDIA, 2009). Como um exemplo de SaaS é possível citar a SAP Business ByDesign[2] que é um serviço de *Enterprise Resource Planning* (ERP) para pequenas e médias empresas; um exemplo da IaaS é a Amazon S3[3] que provê armazenamento de dados; e um exemplo da PaaS é o Google App Engine[4], plataforma para desenvolvimento e hospedagem de aplicações web.

É comum hoje encontrarmos diversos sites ou portais que centralizam serviços como e-mails, páginas de notícias, blogs, etc. Essa centralização de conteúdos facilita para o usuário o acesso aos seus conteúdos preferidos num só local, agilizando as visitas e visualizações. Costumadamente estes sites têm suas interfaces compostas por *widgets* que são pequenos componentes da página que trazem funcionalidades específicas. São exemplos de alguns portais como esses o iGoogle[5] e o Netvibes[6], criados para os usuários que utilizam diversos serviços da internet e podem centralizá-los em uma única página da internet, tornando o acesso mais prático.

A Figura 1.1 ilustra alguns exemplos de *widgets* de diferentes portais que trazem conteúdos e serviços de outros sites. O primeiro (A) é um componente que fornece informações sobre as últimas notícias de um site; o segundo (B) apresenta a previsão climática fornecida por um site especializado para a cidade que o usuário escolher; e o terceiro (C) possibilita a configuração para leitura de mensagens do usuário de diversos serviços de e-mail.

2 http://www.sap.com/sme/solutions/businessmanagement/businessbydesign/index.epx

3 http://aws.amazon.com/s3/

4 http://appengine.google.com/

5 http://www.google.com/ig

6 http://www.netvibes.com/

Figura 1.1: Exemplos de *widgets*

Como comentado no começo desta seção sobre os novos usos da Internet e a importância dela como meio de acesso a vários tipos e fontes de informações e serviços, estes portais centralizadores de conteúdo têm sido amplamente utilizados como ferramenta facilitadora do dia-a-dia.

1.2 JUSTIFICATIVA

Com o avanço dos modelos de celulares e crescimento nas possibilidades de uso deles, há um bom tempo estes dispositivos não são mais utilizados simplesmente para ligações de voz. Além de bem material particular, os celulares têm sido muito utilizados como ferramenta de trabalho já que conseguem executar algumas tarefas como os computadores. Então, nessas tarefas e mesmo para uso pessoal, geralmente eles precisam de acesso aos conteúdos necessários para aquela determinada utilização.

Conteúdos podem ser entregues para os celulares por diferentes meios: *mobile sites* na internet, aplicativos de terceiros que interagem com *web services*, software

embarcado, e outros. Cada maneira de se entregar esses conteúdos tem seus pontos positivos e negativos, tornando uma melhor do que a outra para uma específica situação.

Analisando o caso de aplicativos que têm a característica de interação com serviços da internet é possível perceber vantagens e desvantagens, como apresentou Santos (2009). Entre as vantagens temos: possibilidade de uma melhor experiência ao usuário com interfaces mais ricas; acesso a recursos nativos do aparelho (como câmera e agenda); possibilidade de se ter um custo menor para o usuário no caso de transferência de dados. Entre as desvantagens estão: dificuldade perante os usuários com o processo de instalação de um novo aplicativo; difícil controle e manutenção do aplicativo pelo fabricante após o lançamento; fragmentação das linguagens nativas por causa da existência dos diversos sistemas operacionais de vários fabricantes existentes no mercado, que eventualmente obrigam a criação de diferentes aplicativos para atenderem diferentes marcas.

No caso da última desvantagem citada, o uso da linguagem Java pode ser uma solução, já que o seu lema é "*write once, run anywhere*" ("escreva uma vez, execute em qualquer lugar"). Talvez não seja a melhor por ainda não serem todos celulares que possuem a plataforma instalada, e também, nos que possuem, existem diferentes versões com implementações também diferentes, ou seja, nem sempre o mesmo código produz um aplicativo igual para todos os modelos de celular. Ainda sim, tratando-se de desenvolvimento de aplicativos, talvez a linguagem de programação Java seja a melhor opção para oferecer uma melhor experiência ao usuário e que atenda um bom número de celulares.

Talvez a ideia mais interessante para atender um grande número de pessoas e com um custo acessível aos usuários de celular seja a de criar *mobile sites* para oferecer os serviços de um portal. Os *mobile sites* são mais fáceis para se distribuir e divulgar, encontrar via sistemas de busca e até "viralizar", já que nenhum aplicativo novo precisa ser instalado para sua visualização e também pode ser facilmente repassado via *SMS-link* ou semelhantes. Além do custo reduzido para sua criação, sites para dispositivos móveis são mais fáceis de se atualizar e manter já que não ficam localmente no aparelho, diferente dos aplicativos.

Então, com a disponibilidade via internet possibilitando talvez a mais universal (e que garante a maior) forma de compatibilidade com diferentes aparelhos (se o *mobile site* for criado seguindo as boas práticas recomendadas), os sites específicos para dispositivos móveis são a melhor escolha para a entrega de conteúdos a esses dispositivos.

1.3 OBJETIVOS

Como foi visto a importância do frequente acesso a sites e serviços da internet, de forma rápida e centralizada se possível, e do celular como ferramenta em diversos contextos de uso e acesso a internet, viu-se a oportunidade de criar um portal para facilitar o acesso a Internet Móvel, já que os diversos conteúdos da rede mundial estão gradativamente sendo disponibilizados de uma melhor forma a atender as peculiaridades dos dispositivos móveis mas de uma forma pouco conhecida pelos usuários.

Baseado no exemplo dos portais agregadores de conteúdos apresentados (seção 1.1), o portal irá agrupar uma coleção de sites para dispositivos móveis onde os usuários poderão facilmente separar os seus conteúdos preferidos e acessarem todos de um só lugar na internet pelo celular.

No momento da realização deste trabalho não foi encontrado nenhum portal semelhante que possibilitava a criação de uma página personalizada. No Brasil foi encontrado apenas um portal que apresenta um diretório de sites de diversas categorias, o Portal Hands[7]. No exterior existem mais opções como PhoneFavs[8], Esty.mobi[9] e outros.

1.3.1 Objetivo Geral

- Criar um portal para facilitar o acesso a Internet Móvel.

A proposta deste trabalho de Conclusão de Curso é criar um serviço que contempla basicamente dois meios: o computador e o celular. Pelo computador o usuário pode entrar em um portal onde tem acesso a um diretório de sites já conhecidos mas em suas versões próprias para dispositivos móveis. Esses sites ficam separados em diversas categorias e o usuário pode escolher os de sua preferência para criar uma página pessoal.

1.3.2 Objetivos Específicos

- Desenvolver uma aplicação web para criação de uma página *mobile* pessoal com os sites da preferência do usuário, centralizando e facilitando o acesso a estes.
- Criar um *mobile site* que mostre um diretório de outros *mobile sites* existentes na Internet, sendo assim uma porta de entrada para estes outros sites, divulgando-os de forma a popularizar a Internet Móvel.

7 http://m.hands.com.br/

8 http://phonefavs.com/

9 http://esty.mobi/

- Conhecer tecnologias e aprimorar conhecimentos sobre o universo de desenvolvimento, especialmente web, e também de tecnologias para dispositivos móveis.

A página pessoal do usuário funcionará como uma página de *bookmarks* ou favoritos, e os sites móveis podem ser disponibilizados ou de uma forma simplificada, mostrando apenas um *link* de acesso para a página salva, ou como *widgets* que entregam o serviço ou conteúdo mais relevante de um determinado site diretamente na própria página do usuário.

Pelo celular, o usuário visita um endereço único, personalizado com o seu nome, para ter acesso a sua página anteriormente configurada pelo computador com os conteúdos de sua preferência. Uma página criada para ser utilizada facilmente pelo celular mostra então os conteúdos dos sites selecionados pelo usuário de uma forma adequada para o dispositivo.

O site do portal e a própria página do usuário, sendo acessados pelo celular, possibilitam também a navegação por toda a base de conteúdos disponíveis caso o visitante ou usuário queira acessar algum outro *mobile site* que não conheça ou que não esteja inserido em sua página pessoal.

A arquitetura do sistema está elaborada a partir do paradigma de Orientação a Objetos e dos conceitos MVC e REST para *web services*. A linguagem de programação utilizada é Java e a persistência dos dados é realizada através do banco de dados MySQL. Estes e outros conceitos e tecnologias serão apresentados no capítulo 4.

1.4 ESTRUTURA DA MONOGRAFIA

No Capítulo 1 foi feita uma breve introdução mostrando o cenário e os problemas que fomentaram a ideia do projeto que será aqui documentado. Foi realizada uma apresentação sobre a importância da Internet nos dias atuais e do surgimento dos portais agregadores de conteúdos e serviços. Mostrou-se também algumas opções de uso do celular para acesso a conteúdos bem como a justificativa sobre usar *mobile sites* no projeto. Finalmente foram apresentados os objetivos do projeto.

O Capítulo 2 faz uma apresentação sobre a Internet Móvel. Um pouco da história dos celulares e da própria Internet Móvel é contada. Também são mostradas as tecnologias envolvidas direta e indiretamente com criação de *mobile sites* e suporte a Internet Móvel.

O Capítulo 3 traz explicações sobre serviços para internet e apresenta conceitos e modelos de projetos e arquiteturas que foram utilizados para o desenvolvimento dos componentes deste projeto.

O Capítulo 4 apresenta e explica as diversas tecnologias, linguagens, ferramentas e processos usados no decorrer do desenvolvimento do projeto.

O Capítulo 5 detalha a implementação do projeto. É explicado como funcionam os diferentes componentes do projeto, as principais classes e lógicas de programação por trás do portal. É mostrado também onde os conceitos e tecnologias escolhidos são empregados.

O Capítulo 6 faz uma conclusão sobre a importância do projeto para o graduando. É repassado os resultados finais do projeto, é apresentado as matérias cursadas que ajudaram o planejamento e desenvolvimento do projeto. É apresentado também possíveis caminhos para trabalhos futuros e continuação do projeto.

2 INTERNET MÓVEL

A Internet surgiu com a necessidade de compartilhamento de hardware e software e com o propósito de facilitar a troca de conhecimentos e tornar público o acesso a conteúdos diversos. Com o passar dos anos, sua existência se dividiu em fases, desde a dominação dos grandes portais de entrada para usuários (os chamados portais provedores de conteúdos) até os portais de serviços, sites colaborativos e redes sociais que temos hoje (a chamada Web 2.0). Mas não foi só a Internet que evoluiu com o passar dos anos, mas também a forma de acessar e interagir com ela.

O século XX foi época de grandes avanços tecnológicos, do surgimento do telefone, do computador pessoal, e mais para o final do século começaram a aparecer os dispositivos móveis[10]: *notebooks*, celulares, PDA's - *Personal Digital Assistant* (Assistente Pessoal Digital), *mp3 players* e similares, etc. Com o desenvolvimento das tecnologias, esses dispositivos também se tornaram capazes de acessar a internet e seu uso tem crescido vertiginosamente.

Segundo SAARIKOSKI (2006) definir Internet Móvel não é tão simples pelos diferentes entendimentos que o termo pode gerar para as pessoas conforme sua vida pessoal e profissional, ou ainda conforme o próprio uso ou não dela, entre outros fatores. Para este trabalho, vamos considerar Internet Móvel como termo para descrever o acesso e os conteúdos específicos da Internet para dispositivos móveis.

2.1 HISTÓRICO

Os celulares, que surgiram por volta da década de 70 e só começaram a ser comercializados mais de 10 anos depois, tiveram versões de navegadores para internet só em 1997 (WIKIPÉDIA, 2009), e ainda sim, tudo era bem diferente do que se conhecia com os computadores pessoais.

Os próprios aparelhos têm diversas diferenças e limitações se comparados com os computadores, e estas ainda acabam por obrigar a se ter uma internet de certa forma diferente, não totalmente pelo seu conteúdo, mas sim na sua forma de apresentação. Alguns desses problemas são: tamanho e resolução reduzidos da tela, ausência de mouse para navegação pelo conteúdo, baixa velocidade e alto custo para tráfego de dados.

---

10 Neste trabalho consideraremos somente a classe dos celulares nas referências feitas aos dispositivos móveis.

No começo da Internet Móvel, com os recursos de rede e dos próprios dispositivos ainda muito precários, criaram-se novas tecnologias para trazer a internet para estes aparelhos. Novas linguagens para criação das páginas e protocolos de comunicação foram desenvolvidos, sistemas foram elaborados para transformarem e levarem os conteúdos da internet convencional para os celulares, mas ainda assim ela não alavancou.

Tecnologias ainda em fase inicial, pouca quantidade de conteúdos especiais para os dispositivos móveis e a forma de cobrança abusiva das operadoras foram fatores que frearam a Internet Móvel para que ela virasse parte do dia-a-dia dos usuários de celular.

Mas com o tempo novas tecnologias foram sendo criadas para tentar aproximar o uso da internet tradicional para os celulares. Dispositivos melhores e mais potentes foram fabricados, novos protocolos de comunicação e transferência de dados (bem como melhorias nas redes de telefonia móvel) também começaram a ganhar espaço, o uso de novas linguagens para criação de páginas mais próximas das linguagens já usadas normalmente foi adotado.

O surgimento de novos aparelhos celulares (com telas maiores, por exemplo), o avanço tecnológico das redes de telefonia celular (possibilitando maiores velocidades de acesso e transferência de dados), a criação de *microbowsers* que reduziam o conteúdo da internet convencional para serem visualizados nos dispositivos pequenos, salvaram, ou ressuscitaram a Internet Móvel.

Atualmente a Internet Móvel está crescendo e muito ainda precisa ser melhorado através de novas tecnologias nas áreas de redes, dispositivos e serviços. Para isso, muitas definições e padrões estão sendo discutidos e testados, como por exemplo, padrões de endereços e publicidade.

2.2 TECNOLOGIAS

A Internet Móvel chegou inicialmente aos celulares via *Wireless Application Protocol* (WAP - Protocolo para Aplicações sem Fio), que foi o padrão internacional criado para a utilização de comunicação de dados sem fio para dispositivos móveis. Seu intuito era entregar conteúdos e serviços da internet, como ler e-mails, acessar portais de notícias etc., aos navegadores WAP dos celulares (WIKIPÉDIA, 2009).

O protocolo WAP foi concebido na época da segunda geração da telefonia celular, a chamada 2G, que trouxe melhorias nos sistemas digitais da rede móvel. Essa geração é conhecida pelo surgimento dos padrões TDMA, CDMA e GSM. As tecnologias dessa geração ainda proporcionavam uma lenta taxa de transferência de dados, por volta dos 10

Kbps, então, usar a internet pelo celular não era uma boa experiência já que o serviço era bem lento.

A ideia era de existir um *gateway* ou tipo de *proxy* WAP que transformaria os conteúdos da internet tradicional para os dispositivos móveis. As requisições por conteúdos da internet providas pelo celular chegariam ao *gateway* WAP, que iria buscar as informações e as transformar de uma melhor forma para ser entregue ao celular[11]. Abaixo pode ser visto uma figura que ilustra esse processo.

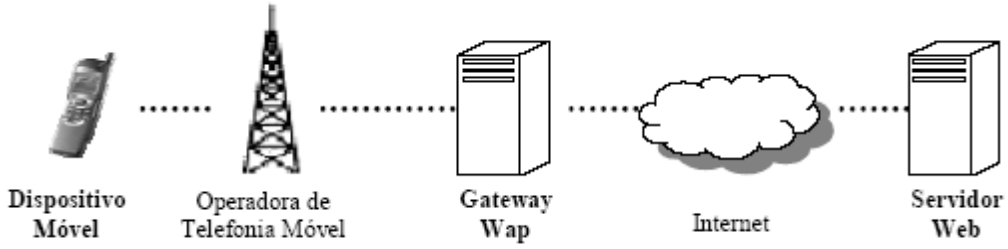

Figura 2.1 Gateway WAP
Fonte: Taffe (2002)

Para entregar os conteúdos para um celular via protocolo WAP, foi definida uma linguagem específica para criar as páginas da internet, a *Wireless Markup Language* (Linguagem de Marcação para Comunicação sem Fio). A WML é uma linguagem parecida com XML e HTML que tem características próprias no seu uso para os dispositivos móveis, com o intuito de aproximar o usuário da interface de hardware do dispositivo e possibilitar, por exemplo, que se faça ligações por uma página escrita nesta linguagem. (WIKIPÉDIA, 2009).

A linguagem WML trabalha basicamente com dois conceitos: *decks* e *cards*. Um *deck* (baralho) é definido entre as *tags* `<wml>` e `</wml>`, e compõem uma página, que pode conter vários *cards* (cartas). Um *card* é uma unidade de interação com o usuário e pode conter outras *tags* para disponibilizar elementos de texto, imagem e/ou entrada de dados.

Entretanto, o uso da internet via WAP não alavancou. Além de ser lento, o seu custo era alto, já que as operadoras tarifavam o uso do serviço pelo tempo usado. Mas as tecnologias das redes celulares evoluíram para a chamada segunda geração e meia (2,5G) com os padrões GPRS e EDGE, que proviam transmissão de dados por pacotes em uma maior velocidade. Já com essas melhorias, o uso da Internet Móvel começou a ser mais considerado.

11 No Japão, um padrão similar de protocolo de comunicação sem fio também foi criado pela operadora local NTT DoCoMo, o *i-mode*, que fez muito sucesso e garantiu o uso da Internet Móvel lá antes mesmo do Ocidente (SAARIKOSKI, 2006).

O protocolo WAP também evoluiu até o chamado WAP 2.0. A nova versão consistia do uso de uma versão da linguagem XHTML para as páginas e de comunicação direta HTTP, deixando de lado a idéia do *gateway* ou *proxy* que não surgiu como havia sido proposto, já que as operadoras ou serviços da telefonia móvel não os criaram (WIKIPÉDIA, 2009). Com o avanço das tecnologias de rede celular, o uso da WAP 2.0 se tornou mais rápido e até mais barato.

Para a criação de páginas na WAP 2.0, a versão XHTML Mobile Profile foi padronizada como melhor opção para uso nos dispositivos móveis. Esta linguagem possibilitava um melhor aprimoramento visual das páginas junto da utilização de uma versão simplificada de CSS do que a WML conseguia. Seu uso também aproximou a internet "comum" da Internet Móvel já que sua sintaxe é bem similar do que já costumadamente é utilizado.

As redes de telefonia celular continuaram a evoluir chegando a terceira geração (3G). As novas tecnologias dessa geração possibilitam uma taxa de transferência de dados de 2 Mbps, facilitando uma boa experiência no uso da internet pelo celular.

2.3 ESPECIFICIDADES DOS CONTEÚDOS PARA CELULAR

Devido a diferenças de tecnologias criadas com o passar do tempo para a Internet Móvel, os celulares ficaram divididos pelos diferentes tipos e formatos de conteúdos que cada um aceitava ou trocava com a rede.

Quando o protocolo WAP e a linguagem WML eram utilizados, era obrigatório informar o tipo de conteúdo (ou *Content-Type* como é conhecido em inglês) no formato `text/vnd.wap.wml;`. Depois os celulares começaram a aceitar outros formatos mais avançados para as linguagens HTML (`Content-Type: text/html;`) e XHTML (`Content-Type: application/xhtml+xml;`).

Além dessa diferença no tipo do conteúdo, dependendo de onde o celular era fabricado, ele suportava um tipo diferente de codificação de caracteres (*Encoding* ou *Character Encoding*), sendo os mais utilizados o europeu ISO-8859-1 e o Unicode UTF-8.

Além dessas peculiaridades no formato dos conteúdos das páginas, o detalhe da Declaração do Tipo do Documento (DOCTYPE - *Document Type Declaration*) também faz diferença conforme o suportado pelo celular e pode influenciar na maneira que o mobile site é renderizado pelo navegador.

Existem também diferenças para os conteúdos multimídia. Alguns anos atrás os celulares apenas aceitavam imagens do tipo `Content-Type: image/gif;`, mas com o passar do tempo começaram a aceitar outros formatos, como `Content-Type: image/png;`

e `Content-Type: image/jpeg;`. Para conteúdos de vídeo temos `Content-Type: video/3gpp;` ou `Content-Type: video/mpeg;` por exemplo.

Além destes detalhes nos tipos e formatos, a forma de entregar conteúdos pela Internet Móvel também é diferente, já que se trata de um dispositivo pequeno, com características diferentes de um computador, detalhes estes que devem ser considerados para proporcionar uma boa experiência ao usuário.

Diante de todas estas características, grandes empresas e entidades somam esforços na definição de melhores práticas, cuidados e atenção sobre esse assunto. Um exemplo é o W3C Mobile Web Initiative[12] (W3C MWI), grupo liderado pela W3C e patrocinado por empresas das áreas de tecnologia e telefonia, que define boas práticas para desenvolvimento e suporte da Internet Móvel.

Durante o desenvolvimento deste projeto, estas práticas foram estudadas e o máximo foi feito para garantir a entrega de conteúdos da melhor maneira, tanto no formato aceitado pelo dispositivo como pensando na usabilidade para o usuário.

12 http://www.w3.org/Mobile/

3 WEB SERVICES

Segundo a *World Wide Web Consortium* – W3C (2009), o conceito de Arquitetura *Web Service* define um sistema criado para suportar a interação entre máquinas numa rede. Juntamente do conceito de *Cloud Computing* explicado anteriormente, essas duas ideias se complementam para criar as chamadas *Web-applications* (Aplicações Web), que têm o intuito de levar a todo o mundo as funcionalidades de um serviço disponível na rede mundial de computadores.

Os *Web Services* trazem portabilidade e facilidade de comunicação entre serviços e máquinas distintas. Um de seus conceitos é de possibilitar a troca de informações ou pedidos de realização de tarefas (simples ou complexas) em sistemas diferentes, independente de sua lógica e/ou linguagem de programação. Para que exista essa independência entre as aplicações, é necessário então um formato reconhecido ou universal para a comunicação entre as partes.

Para a comunicação entre *web services*, costumadamente é utilizado o protocolo HTTP - *Hypertext Transfer Protocol* da Internet e o formato XML - *eXtensible Markup Language* nos pedidos e respostas. Com o surgimento de outras necessidades entre os serviços, novos formatos estão sendo utilizados, como é o caso do formato JSON - *JavaScript Object Notation* que é mais simples e menor no tamanho final da serialização de objetos do que o produzido com XML.

Para se construir *web services*, vários estilos de arquitetura de software podem ser empregados como plataforma a ajudar a criação de aplicações. Citando alguns exemplos, temos o protocolo SOAP - *Simple Object Access Protocol*, temos também o modelo de comunicação Cliente-Servidor, o estilo de arquitetura hipermídia REST - *Representational State Transfer*, e outros.

3.1 MODELO CLIENTE-SERVIDOR

Modelo Cliente-Servidor é um estilo de arquitetura de aplicações distribuídas que divide as tarefas ou cargas de trabalho entre o servidor e o solicitante (cliente), que frequentemente ficam separados numa rede de computadores (WIKIPÉDIA, 2009). Serviços triviais, como consultas a bases de dados, são exemplos de uso desse modelo.

Um Servidor é uma máquina de alta performance configurada para receber pedidos de outras máquinas Clientes, eventualmente de menor processamento. A ideia central do

modelo é a separação de responsabilidades, o que possibilita uma divisão do serviço em componentes, deixando a responsabilidade pela interação do lado do componente do cliente e a carga de processamento e escalabilidade do lado do componente do servidor.

A separação entre as partes cliente e servidor possibilita ainda o desenvolvimento de cada uma com tecnologias diferentes, desde que se comuniquem da forma especificada.

No caso deste projeto, o cliente (aparelho celular) não precisa executar o processamento para obtenção dos conteúdos da internet e ainda trabalhar sobre eles para separar e mostrar ao usuário só o necessário. Todo este trabalho é feito pela parte do servidor, o que agiliza o processo e diminui a transferência de dados, refletindo ainda em um menor custo ao usuário.

3.2 ARQUITETURA REST

Como Costello (2009) exemplificou, a internet é composta de recursos. Recursos são itens de interesse de um usuário. Por exemplo, a empresa Boeing pode definir o avião 747 como um recurso e as pessoas podem acessar este recurso pela URL:

`http://www.boeing.com/aircraft/747`

Visitando tal endereço, uma representação deste recurso é retornada, por exemplo a página `Boeing747.html`. Com esta representação, o cliente fica em um estado de uso na sua navegação, e a partir dai ele pode fazer outras requisições de outros recursos, mudando assim o seu estado.

Fielding (2000) explica o significado de *Representational State Transfer* (REST) em sua tese de doutorado:

> *"Representational State Transfer is intended to evoke an image of how a well-designed Web application behaves: a network of web pages (a virtual state-machine), where the user progresses through an application by selecting links (state transitions), resulting in the next page (representing the next state of the application) being transferred to the user and rendered for their use."*

Comentando a explicação, REST (Transferência de Estado Representacional) é pretendida como a imagem de como a aplicação se comportará: em uma rede de páginas que representam recursos, o usuário navega de uma página para a seguinte pelas ligações (*links*) entre elas, o que acarreta a mudança de estado dentro da aplicação.

A ideia do estilo de arquitetura REST é utilizar dos padrões já conhecidos da web para se construir aplicações para ela mesma. Através de requisições HTTP para recursos identificados por URI's, as representações destes recursos são entregues conforme o formato definido ou pedido, seja XML, HTML ou mesmo GIF, por exemplo.

Seguindo com o exemplo da Boeing, se temos o recurso avião 747, podemos querer obter, alterar, apagar ou eventualmente criar a representação dele. Para executar tais operações, o próprio protocolo HTTP já define formas de fazê-lo com os métodos GET, PUT, DELETE e POST. E como resposta da operação, a aplicação pode retornar códigos HTTP de sucesso (200 - *Ok*), de criação (201 - *Created*), ou outros.

Se o acesso a recursos é feito via URL's, então nada impede de usá-las com a melhor semântica possível para se garantir o entendimento da rede de informações. Por exemplo, se desejamos obter a lista de aviões da Boeing, uma URL intuitiva seria:

`http://www.boeing.com/aircrafts`

Se desejamos acessar o recurso 747 da lista de aviões, a URL seria:

`http://www.boeing.com/aircrafts/747`

e se desejamos ver a imagem de um avião 747, é mais fácil de entender a URL

`http://www.boeing.com/aircrafts/747/photo`

ao invés de algo do tipo:

`http://www.boeing.com/?q=aircrafts&m=747&i=photo`

Apesar do estilo de arquitetura REST parecer simples (com as características de interfaces uniformes aceitando todos os métodos HTTP GET, POST, PUT e DELETE; recursos nomeáveis e com representações interconectadas por URL's), outras características o tornam um grande estilo de arquitetura para *web services*. Algumas dessas outras características são: estar livre de estado (cada requisição do cliente para o servidor contém toda a informação para ser entendida), recursos podem ser salvos em cache (para garantir eficiência na rede) e estilo de interação cliente-servidor (WIKIPÉDIA, 2009).

3.3 ARQUITETURA MVC

Para um bom controle sobre as partes de um sistema e não permitir uma mistura de papéis entre os componentes lógicos de uma aplicação, um padrão que separa a camada de visualização da camada de negócios foi elaborado para ajudar os desenvolvedores a ter uma arquitetura de qualidade, o padrão *Model-View-controller* (MVC - Modelo-Visão-Controle).

Em um serviço web (tipo de aplicação que mais se encontra o padrão aplicado), a camada de Controle é o ponto de entrada das requisições HTTP, e ela controla as camadas de Modelo e Visualização facilitando a troca de dados entre elas. A camada de Modelo trabalha com os dados, que foram obtidos pela camada de Controle e passados para a camada de Visualização para serem renderizados de uma forma apresentável. O principal benefício dessa organização é que a camada de Modelo pode se preocupar apenas com os dados e não tem conhecimento da Visualização. Essa camada, por outro lado, não tem

conhecimento da camada de Negócios e apenas renderiza os dados repassados para ela (HEMRAJANI, 2006). Abaixo a Figura 3.1 ilustra como as camadas do padrão MVC se relacionam.

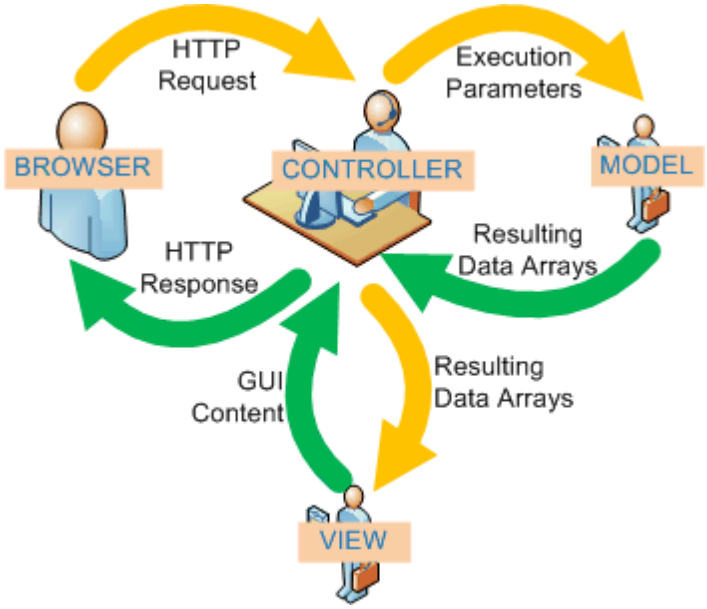

Figura 3.1 – Arquitetura e camadas do padrão MVC
Fonte: http://ash-mvc.org/

A camada de Modelo, ou algumas vezes também chamada de camada de Negócios, representa e trata os dados e lógicas da aplicação sobre esses dados.

A camada de Visualização é a responsável por como os dados devem ser apresentados ao usuário ou cliente da aplicação. Múltiplas visualizações podem ser criadas (web, *desktop*, *mobile*) para as mesmas classes de modelo.

A camada de Controle é intermediária entre a camada de Modelo e a camada de Visualização. Ela é responsável por processar os pedidos do cliente para a camada de Modelo e depois repassar os dados dessa última para a camada de Visualização.

4 TECNOLOGIAS E PROCESSO UTILIZADOS

Para o desenvolvimento do projeto, um conjunto de tecnologias e processos foi escolhido para tentar mantê-lo atualizado com as ferramentas, linguagens e metodologias mais utilizadas e/ou recomendadas na comunidade desenvolvedora de software.

4.1 JAVA

A linguagem de programação Java[13] foi criada pela Sun Microsystems com o paradigma de Orientação a Objetos e o conceito de execução em uma máquina virtual separada. Ela é extensamente utilizada em programas *desktop* e/ou *web-based* de várias áreas e está presente em computadores, servidores, dispositivos móveis e muitos outros tipos de aparelhos.

Diferente de outras linguagens tradicionais que são compiladas para a linguagem de máquina, Java é compilada para os chamados *bytecodes*, um passo intermediário que gera um código próprio para ser executado numa máquina virtual específica, a Java Virtual Machine (JVM).

Java é dividido basicamente em 3 tipos de distribuições: Java Standard Edition (JSE), Java Enterprise Edition (JEE) e Java Micro Edition (JME). Cada um desses pacotes possui conjuntos de bibliotecas, API's e código para diferentes aplicações. A JSE é a padrão e a que possui o maior número de classes e pacotes para aplicações comuns, geralmente *desktop*. A JEE contém bibliotecas que possibilitam a criação de software distribuído e multi-camada que devem rodar num servidor de aplicação. A JME possui códigos e implementações reduzidas a ponto de atender dispositivos de baixo nível de processamento e de memória - os dispositivos embarcados.

A linguagem Java e a plataforma JEE foram escolhidas para este projeto pela facilidade e abstração de vários conceitos ao se criar *web services*, que é o tipo de software do projeto. Essa escolha também se deve ao fato das diversas ferramentas, bibliotecas e *frameworks* já existentes que ajudam o desenvolvimento de aplicações. A versão utilizada foi a 1.5.

13 http://www.java.com/

4.2 ECLIPSE

O Eclipse SDK[14] é uma IDE - *Integrated Development Environment* (Ambiente de Desenvolvimento Integrado) *open source* criado pela IBM. Trata-se de uma plataforma com diversas ferramentas que facilitam a criação de aplicações. Inicialmente, ele foi criado para o desenvolvimento com a linguagem Java, mas hoje já possui extensões que dão suporte a outras linguagens.

Talvez o maior ponto favorável do Eclipse seja seu ambiente integrado. Por ser um projeto aberto, foi possível criar várias extensões e *plugins* para melhorar a ferramenta, tornando-a um ambiente centralizado de desenvolvimento. Como exemplos de *plugins*, temos: interface para diagramação UML, ferramentas de integração a bancos de dados, editores para desenvolvimento de ambientes web, e outros.

4.3 APACHE TOMCAT

Como citado brevemente sobre a necessidade de um servidor de aplicações para rodar programas JEE (seção 4.1), o container web Apache Tomcat[15] foi escolhido para suprir essa necessidade. O Tomcat é um dos mais populares e extensamente utilizado servidor web Java para aplicações Java leves.

O servidor web suporta aplicações que estejam compactadas em um único arquivo, o *Web Application Archive* (WAR - Arquivo de Aplicação Web). O WAR é um arquivo onde fica contido as classes Java, bibliotecas, e arquivos e insumos estáticos de uma aplicação web, tudo em apenas um arquivo, facilitando a distribuição e a instalação, que para ser feita é apenas colocá-lo num diretório específico do servidor web.

4.4 ADOBE FLEX

Adobe Flex[16] é uma plataforma de desenvolvimento de aplicações Flash. Seu intuito é facilitar a programação em Flash com um novo paradigma diferente do de linha de tempo do próprio Flash. A ferramenta é costumadamente utilizada para a criação de aplicações ricas (RIA's) para a internet.

Com uma linguagem MXML e XML é possível criar interfaces que possibilitam uma melhor experiência e usabilidade para o usuário. Diversos componentes gráficos (como

14 http://www.eclipse.org/

15 http://tomcat.apache.org/

16 http://www.adobe.com/products/flex/

botões, menus, caixas, *etc.*) já estão prontos e podem ser (re-)utilizados, agilizando o desenvolvimento.

4.5 JAVA PERSISTENCE API

Para facilitar a manipulação de dados de um modelo relacional, a Java Persistence API[17], também chamada simplesmente de JPA, foi elaborada como um conjunto de classes da linguagem Java para dar suporte a fácil abstração de classes num mapeamento objeto/relacional.

O *framework* se baseia em uma API, que, com apenas alguns fragmentos de código inseridos nas classes que se deseja levar para o modelo relacional, delega para outra implementação cuidar do mapeamento e da comunicação com um banco de dados relacional.

O uso da JPA facilita o desenvolvimento da camada de negócios da aplicação e persistência de dados. Junto de uma implementação sua (Hibernate, por exemplo) e da própria JPA, o desenvolvedor pode ficar despreocupado sobre as interações e comandos necessários junto ao banco de dados.

4.6 HIBERNATE

Como foi visto na seção 4.5, a JPA precisa de uma implementação para que a aplicação efetivamente se comunique com um banco de dados. Neste projeto a implementação escolhida foi a Hibernate[18], que é uma biblioteca amplamente utilizada e já bem madura junto a comunidade desenvolvedora.

Hibernate é um *framework* Java para mapeamento e persistência objeto/relacional. Um *framework* deste tipo trabalha com a transformação das classes de objetos para tabelas de dados de um banco de dados relacional e vice-versa. Ele cuida da comunicação por SQL e libera o desenvolvedor do trabalho de conversão de dados.

17 http://java.sun.com/developer/technicalArticles/J2EE/jpa/

18 https://www.hibernate.org/

4.7 MYSQL

Para a persistência de dados, foi escolhido o banco de dados relacional MySQL[19]. Além de ser gratuito, a ferramenta é amplamente utilizada pela comunidade desenvolvedora por ser leve mas com grande capacidade de performance.

4.8 SPRING

O Spring[20] é um grande *framework* que contêm diversas classes e pacotes, mas pode ser usado em partes conforme a necessidade. Ele provê funcionalidades ou pode ser usado como: um container de Inversão de Controle (*Inversion of Control* - IoC, em inglês), um *framework* web, camada de gerenciamento de transação, agendamento de tarefas, e muitas outras funcionalidades.

A principal funcionalidade do Spring é a inversão de controle e a injeção de dependência. Esses padrões mudam o fluxo de controle tradicional onde a criação, execução e finalização de componentes está toda sob o controle do programador para um fluxo onde o ciclo é executado com delegações de controle para terceiros.

O Spring foi escolhido para suprir as funcionalidades de inversão de controle, injeção de dependências e gerência de transações. Em um arquivo de configuração do *framework* são feitas as declarações de classes e seus relacionamentos de dependência. Neste arquivo também são configuradas as informações de acesso ao banco de dados e controle de transações com ele.

O *framework* fica responsável por todas as classes que estão sob seu controle de injeção. Conforme o uso destas seja necessário, ele cria instâncias delas e as injeta segundo as configurações de dependência.

Além desta manipulação sobre a instanciação, o *framework* controla a gerência sobre transações com bancos de dados. Ele fica responsável por criar novas transações, cuidar da integridade delas e dos dados trafegados, de fechá-las e reutilizá-las caso seja possível ou não.

19 http://www.mysql.com/

20 http://www.springsource.org/

4.9 STRUTS

O Struts[21] é um *framework* para ajudar o desenvolvimento de aplicações web que são construídas com base no padrão arquitetural MVC, e está fortemente ligado as camadas de Controle e Visualização deste padrão.

Suas funcionalidades principais são: um manipulador de requisições que são padronizadas em URI's para um controle sobre o fluxo de entrada da aplicação; um manipulador das respostas da aplicação que facilita o fluxo de saída encaminhando facilmente a resposta para outros tipos e formatos; uma biblioteca de *tags* para ajudar e facilitar a criação de interfaces da aplicação.

Em um arquivo de configuração, são declarados os padrões de URI que o manipulador deve tratar conforme a classe de controle responsável informada pelo desenvolvedor. Neste mesmo arquivo, é configurado também como será a manipulação da resposta de um recurso.

O *framework* trabalha como um filtro encapsulando toda a, ou uma parte da, aplicação web. Conforme as requisições chegam nela, um delegador verifica a URL requisitada e tenta encontrar um padrão semelhante no arquivo de configuração, para aí delegar o fluxo da requisição a uma classe responsável por aquele padrão.

Na camada de Controle, com as classes configuradas para manipulação de requisições, tanto na própria requisição como na resposta, o Struts trata de transformar os parâmetros enviados em atributos da classe. Essa transformação só é possível se a classe possuir métodos de acesso para esses atributos.

As classes que cuidam da camada de Controle com o *framework* Struts são conhecidas como sendo do tipo `Action`. As características delas são os métodos de acesso aos atributos de classe e os métodos sem parâmetro e com retorno de uma `String`.

Depois que a lógica da aplicação é executada, o método da `Action` retorna uma `String` que está configurada para um tipo de resposta no arquivo de configuração do *framework*. Para cada tipo de resposta é possível por exemplo repassar o fluxo para uma JSP ou um arquivo XSLT, ou ainda apenas retornar um código HTTP, conforme se queira.

21 http://struts.apache.org/

4.10 XSLT

Criada pela W3C, a XSLT[22] – *eXtensible Stylesheet Language for Transformation* é uma linguagem de transformação para documentos XML que faz parte da especificação XSL. Com ela é possível criar documentos XHTML ou outros documentos XML a partir de um documento XML.

A linguagem foi utilizada para formatar a resposta de alguns tipos de solicitações nos *web services* do projeto junto com o *framework* Struts. O próprio *framework* é responsável por repassar os objetos da requisição para um documento XSLT configurado pelo desenvolvedor, e neste documento é possível manipular as informações dos objetos para criar um documento XML da forma desejada.

4.11 JSON

JSON[23] é o acrônimo para *JavaScript Object Notation*, que é um formato texto de intercâmbio de estruturas de dados serializadas. Seu intuito é de ser fácil para os humanos lerem e escreverem, e de ser fácil para os computadores analisarem e gerarem.

É um subconjunto da notação de objetos de JavaScript e é costumadamente utilizado em aplicações web AJAX como alternativa a documentos XML, mas seu uso não requer o JavaScript exclusivamente.

Seu padrão é formado basicamente com dois tipos de estruturas: uma coleção de pares nome/valor que geralmente representam um objeto (mostrado na Figura 4.1); ou uma lista ordenada de valores (estes podendo ser objetos ou outra lista) que geralmente representa um *array* (mostrado na Figura 4.2).

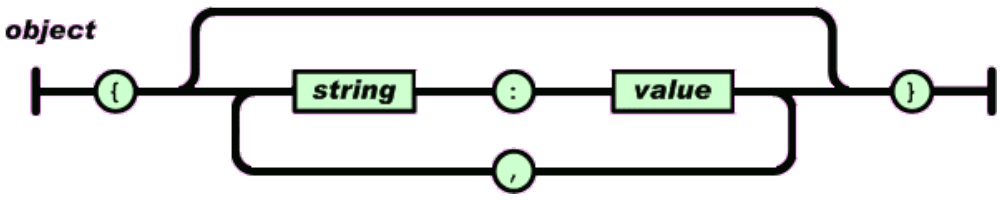

Figura 4.1 – Diagrama do formato de um objeto escrito em JSON
Fonte: http://json.org

22 http://www.w3.org/TR/xslt

23 http://json.org/

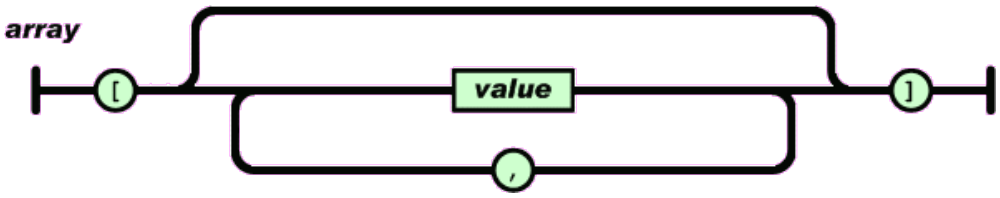

Figura 4.2 – Diagrama do formato de uma coleção escrita em JSON
Fonte: http://json.org

4.12 METODOLOGIAS ÁGEIS

Processos e práticas ágeis no desenvolvimento de um projeto ajudam a rápida entrega, foco na simplicidade do produto mas com alto valor ao usuário/cliente, e um alto grau de flexibilidade para mudanças.

No ano de 2001, renomeados desenvolvedores de software ao elaborarem o Manifesto Ágil (Agile Manifesto) concordaram sobre valorizar indivíduos e interações mais que processos e ferramentas, software funcionando mais do que documentação abrangente, colaboração com o cliente mais que negociação de contratos, e aceitar e responder as mudanças mais que seguir um plano.

As metodologias de desenvolvimento e gerenciamento de projetos eXtreme Programming, Scrum e Lean fornecem novos conceitos para criar melhores produtos visando a satisfação e a qualidade, diferente das metodologias tradicionais, como o desenvolvimento em cascata, por exemplo.

Segundo Beck (2004), eXtreme Programming é um estilo de desenvolvimento de software focado na excelente aplicação de técnicas de programação, numa comunicação clara, e no trabalho em equipe que nos permite reconhecer coisas que anteriormente não imaginávamos.

Algumas desses práticas são: redução na documentação e burocratização desnecessários, entrega contínua de pequenas versões para um rápido acompanhamento pelo cliente/usuário, desenvolvimento orientado a testes para garantir segurança e qualidade do software.

O planejamento e gerenciamento do projeto foi feito com a criação de “estórias de usuário” e priorização destas para definição de versões do produto. Com essa metodologia, um dos focos do desenvolvimento fica na entrega de funcionalidades reais ao usuário, sempre entregando algo visível e funcional para ele.

Ainda com relação ao planejamento e gerenciamento, foi definido uma periodicidade de entrega do projeto no seu estado corrente para testá-lo conforme seu

crescimento. Essa idéia de “entregas recentes, entregas frequentes” policia um desenvolvimento sempre preocupado no estado do produto completamente funcional.

Foi adotado o desenvolvimento orientado a testes (*Test Driven* Development - TDD) como processo de criação das funcionalidades do projeto. Para tal, uma série de *frameworks* (como JUnit[24], jMock[25], DbUnit[26] e Selenium[27]) foram utilizados para facilitar o desenvolvimento e ajudar a manter a qualidade do código criado.

Outra metodologia nas práticas ágeis para aumentar a produtividade e assim agregar mais valor ao produto é a automatização de tarefas repetitivas. Uma série de tarefas comuns como: configuração de ambiente de desenvolvimento e de operação, compilação dos códigos do projeto, execução de todos os testes, e outros; foram automatizadas com scripts para serem executadas utilizando-se o *framework* Apache Ant[28].

24 http://www.junit.org/

25 http://www.jmock.org/

26 http://www.dbunit.org/

27 http://seleniumhq.org/

28 http://ant.apache.org/

5 DESENVOLVIMENTO DO PROJETO

A empresa de tecnologia Praesto Convergence[29], desenvolveu alguns produtos próprios, que inicialmente se resumiam apenas a aplicativos para celulares, mas o conceito dos serviços providos pelos aplicativos evoluíram para outros meios também, como *mobile sites*. Para distribuir alguns destes produtos, a empresa criou um portal para divulgá-los, o JMobi[30]. Também com o intuito de ajudar a comunidade, o portal possuía uma seção sobre configurações de aparelhos e um blog sobre notícias e conteúdos relacionados a celular.

Com o crescimento da Internet Móvel no Brasil e melhores formas de captação de recursos com sites para dispositivos móveis do que com aplicativos, a empresa resolveu usar do portal JMobi (um veículo já conhecido como fonte de conteúdo para celular) e criar um novo portal visando a Internet Móvel, projeto este que é apresentado neste documento.

O site JMobi possuía uma grande base de usuários cadastrados, e para o novo portal, usar esta base existente e todo o sistema de cadastro e gerenciamento de contas seria de grande utilidade e facilidade por já estarem em operação. Com esta decisão evita-se também a necessidade de criação de uma nova base de usuários, seja captando novos cadastros até transferindo os já existentes.

O portal facilitador ao acesso a conteúdos da Internet Móvel JMobi é composto por uma base de conteúdos selecionados e categorizados que pode ser acessada via *web services*. Ele fornece uma interface para acesso dos usuários a essa base de conteúdos para a criação de páginas personalizadas a cada usuário. Aqueles *web services* consomem as informações dos usuários e dos seus conteúdos selecionados para disponibilizar uma página com estes conteúdos adaptada para celulares.

A arquitetura do sistema está elaborada a partir do paradigma de Orientação a Objetos e dos conceitos MVC e REST para *web services*. A linguagem de programação

---

29 A Praesto Convergence é uma empresa com foco no universo dos dispositivos móveis atuando no mercado brasileiro com o desenvolvimento de aplicações móveis, integração de aplicativos para celular com serviços web e aplicações web especialmente para dispositivos móveis (*mobile sites*). Fundada em 2005, surgiu inicialmente com uma parceria de negócios com a Fundação Certi e com projetos de TV Digital, mas se dirigiu a projetos de *mobile services*, *mobile marketing* e *mobile advertising*, se tornando uma referência nesses nichos do mercado. A empresa possui uma forte presença no mercado nacional, seja pelo reconhecimento da sua prestação de serviços de alta qualidade ou pelos seus clientes e parceiros, como Siemens, Nokia, RBS, Nextel, Unilever, D/ Araújo Comunicação, entre outros.

30 http://www.jmobi.com.br/

utilizada é Java e a persistência dos dados é realizada através do banco de dados MySQL. Estes conceitos e tecnologias foram apresentados no capítulo 4.

Os recursos do portal são alcançados com solicitações realizadas através de URL's padronizadas REST. Estas solicitações podem ser de criação de um recurso (ex.: criação da página do usuário), de obtenção de um recurso, ou seja, uma consulta (ex.: mostrar página do usuário), ou de atualização (ex.: modificar alguma informação da página do usuário).

Conforme o tipo da solicitação, ela pode ter uma resposta para entregar o recurso acessado. Esta resposta pode ser entregue via JSP's ou com a serialização do recurso com XSLT, e ela pode ser uma página HTML ou um documento XML.

O projeto foi desenvolvido com a ajuda de outros membros da equipe da empresa e foi dividido nos seguintes componentes: *jmobi-service*, *jmobi-editor* e *jmobi-mobilesite*. Está dividido assim para se ter independência e desacoplamento entre os componentes, o que facilita o entendimento e o próprio desenvolvimento, que até pode ocorrer em paralelo e por partes.

O processo de desenvolvimento era estruturado da seguinte maneira: o dono do produto (ou cliente, e, no caso, era o dono da empresa) dizia o que queria como software. Com esses pedidos eram criados “estórias de usuário” que tinham uma prioridade pela importância dada pelo próprio cliente.

Depois das estórias terem sido criadas, a equipe conversava para planejar e quebrar os pedidos do cliente e colocá-las no período da iteração (que era de uma semana). Esse planejamento era feito em uma reunião no começo da iteração.

Durante a iteração, o desenvolvimento das estórias previstas para aquele período sempre era feito com criação de testes unitários, de integração e aceitação para garantir o bom funcionamento de todo o sistema. Todo o código era mantido sob um sistema de controle de versão para que toda a equipe tivesse fácil acesso a última versão sendo trabalhada. Sempre que uma estória era completada, o sistema era instalado em um servidor de desenvolvimento que simula o servidor de operação, podendo assim verificar se não apareceriam problemas de integração.

No final da interação, a última versão estável do software estava no servidor de desenvolvimento para verificação do cliente. Assim ele poderia utilizar o sistema e fazer os seus próprios testes para depois dar um *feedack* sobre o *status*. Dependendo da avaliação, ou se avançava para novas funcionalidades ou se parava para corrigir problemas.

Na mesma reunião do começo de uma interação, a equipe fazia um breve levantamento de impedimentos e problemas que teve na interação anterior, podendo assim resolver os problemas da equipe e melhorar seus processos.

Se o cliente percebesse um estado satisfatório do sistema para ele, uma versão de todo o sistema era empacotada e colocada no servidor de operação para os usuários reais do produto.

5.1 JMOBI-SERVICE

O componente *jmobi-service* pode ser considerado o maior do projeto. Trata-se de um *web service* que disponibiliza as informações sobre as categorias e seus conteúdos, gerencia a criação e edição das páginas dos usuários, e também disponibiliza os conteúdos, sejam simples ou os especiais (também referenciados como *widgets*) criados e mantidos internamente pela empresa.

Este componente foi desenvolvido com a ajuda dos *frameworks* Spring (por causa de seu suporte a inversão de controle, injeção de dependência e gerência de transações), Struts (pelo suporte a *web services* que seguem o padrão MVC) e Hibernate (pela ajuda e simplicidade proporcionada na persistência de dados).

5.1.1 Disponibilizar categorias

A primeira parte deste componente trabalha na obtenção das informações sobre as categorias e seus conteúdos que estão salvas em banco de dados, e depois com a serialização dessas informações para serem disponibilizadas na forma de documentos XML.

Os pedidos são feitos no formato de requisições do estilo arquitetural de um *web service* REST, e para melhor ilustrar, apresentamos dois casos:

Requisição HTTP:

```
GET <HOST>/jmobi-service/categories
```

Resposta de um documento XML com a lista de categorias.

Requisição HTTP:

```
GET <HOST>/jmobi-service/category/Bancos
```

Resposta de um documento XML com a lista de conteúdos da categoria 'Bancos'

Para começar a apresentar a lógica que trata desta primeira macro funcionalidade, é mostrada a classe `CategoriesAction`, que é a responsável na camada de Controle do *web service* para tratar os pedidos pelos recursos da entidade `Category`.

Esta classe segue o padrão de uma classe do tipo `Action` do *framework* Struts, repassando o fluxo para a camada de Modelo (também chamada de camada de Negócios), no caso um DAO - *Data Access Object* (Objeto de Acesso a Dados) que busca as entidades

no banco de dados, e posteriormente repassa as entidades para a camada de Visualização, processo este feito pelo *framework* Struts.

```
// (...) imports
public class CategoriesAction {

	private ICategoryDAO categoryDAO;
	private List<Category> categories;

	public String execute() {
		categories = categoryDAO.retrieveAll();
		if (categories.size() > 0) {
			return Action.SUCCESS;
		}
		return Action.ERROR;
	}
	// (...) construtores, outros métodos, getters e setters
}
```

A classe `CategoryDAO` é a implementação do DAO para acesso a persistência da entidade `Category` no banco de dados. No fragmento de código abaixo está ilustrado a obtenção de todos objetos salvos utilizando a *Java Persistence API* (JPA).

```
// (...) imports
@Transactional
public class CategoryDAO implements ICategoryDAO {

	private EntityManager em;

	public List<Category> retrieveAll() {
		String sql = "SELECT c FROM Category c ORDER BY c.name";
		Query query = getEntityManager().createQuery(sql);
		List<Category> categories = (List<Category>) query.getResultList();

		// (...)

		return categories;
	}
	// (...) construtores, outros métodos, getters e setters
}
```

Para entregar o recurso pedido, no caso uma lista de categorias, os objetos do tipo `Category` são repassados pelo Struts para um transformador XSLT que os serializa em um documento XML. Abaixo é mostrado o fragmento de código do XSLT.

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output encoding="UTF-8" method="xml" />
<xsl:template match="result">
<result>
	<categories>
	<xsl:for-each select="categories/*">
		<category>
			<id><xsl:value-of select="id" /></id>
			<name><xsl:value-of select="name" /></name>
		</category>
	</xsl:for-each>
	</categories>
</result>
</xsl:template>
</xsl:stylesheet>
```

Uma lógica muito parecida é utilizada para a obtenção das informações dos conteúdos (entidades da classe `PageModule` no projeto) de uma determinada `Category`.

5.1.2 Gerência sobre as páginas dos usuários

A segunda parte deste componente é responsável por fornecer e salvar as informações das páginas dos usuários. Para executar tais funcionalidades o sistema sempre precisa que o usuário esteja com uma sessão aberta e, no caso da opção ser de salvar a página, é preciso que seja enviado as informações a serem guardadas.

Para fornecer as informações da página de um usuário, a classe `UserPageAction`, pertencente a camada de Controle do sistema, retira a referência do usuário na sessão e pede a camada de Negócios os objetos referentes a página daquele usuário. Se o usuário ainda não tiver nada criado, um conteúdo padrão é entregue para ele começar. Abaixo é mostrado um fragmento de código da classe responsável por essa função.

```
// (...) imports
public class UserPageAction implements ActionSupport {

	private IUserPageDAO userPageDAO;
```

```
	public String retrieve() {
		HttpServletRequest request = ServletActionContext.getRequest();
		String user = request.getRemoteUser();

		if (user == null) {
			return Action.ERROR;
		}

		userPage = userPageDAO.retrieve(user);

		if (userPage == null) {
			userPage = userPageDAO.retrieveNewDefaultPage();

			// (...)
		}

		return Action.SUCCESS;
	}
	// (...) construtores, outros métodos, getters e setters
}
```

Ainda na classe `UserPageAction`, caso a necessidade seja de salvar novas informações, uma outra lógica trata o pedido. Depois de pegar o usuário na sessão, as informações dos conteúdos que se deseja guardar são retiradas da requisição e transformadas em entidades para serem salvas no banco de dados.

O sistema espera que as informações para serem salvas sejam enviadas no formato JSON em uma coleção de entidades da classe `PageModule`. Abaixo é mostrado um exemplo de como ficariam alguns conteúdos no formato JSON.

```
{ “id” : 1 , “name” : “module” , “url” : “http://example.com/” }
```

Depois de lidas e transformadas em objetos, as entidades enviadas são repassadas para camada de Negócios para serem organizadas como página de um usuário e salvas no banco de dados.

```
// (...) imports
public class UserPageAction implements ActionSupport {

	private IUserPageDAO userPageDAO;
	private IParser jsonParser;

	public String update() {
		HttpServletRequest request = ServletActionContext.getRequest();
		String user = request.getRemoteUser();
```

```
                List<PageModule> askedPageModules = jsonParser.parse(json);

                if (askedPageModules != null) {
                        // (...)

                        userPage = userPageDAO.retrieve(user);
                        if (userPage == null) {
                                userPage = new UserPage(user);
                        }

                        userPage.setPageModules(askedPageModules);
                        userPageDAO.createOrUpdate(userPage);

                        // (...)
                }

                return Action.SUCCESS;
        }
        // (...) construtores, outros métodos, getters e setters
}
```

5.1.3 Disponibilizar conteúdos

A terceira parte do componente *jmobi-service* trata de disponibilizar conteúdos para a página do usuário em um formato adequado. No início do projeto foi decidido entregar os conteúdos de uma forma simples e que não gastasse muito tempo para ser implementada, então optou-se por apenas disponibilizar um link para os sites próprios para celular.

Para a disponibilização dos conteúdos, definiu-se um tipo de arquivo com a extensão "*.jmm*" ("*jmobi mobile module*") que contém fragmentos de código HTML, os quais são acessados pelo componente *jmobi-mobilesite* para criar as páginas dos usuários. Abaixo é possível ver o conteúdo do "*.jmm*" do Banco do Brasil, e logo depois, a Figura 5.1 mostra a visualização dele e como ficaria na página de um usuário. Todos os conteúdos disponibilizados pelo portal podem ser acessados via uma URL singular.

```
<div class="box">
        <h2>Banco do Brasil</h2>
        <div class="b_cont">
                <p><a href="http://wap.bb.com.br">Banco do Brasil Celular.</a></p>
        </div>
</div>
```

Fig 5.1 – Visualização do “.jmm” do Banco do Brasil

Como a idéia do projeto também era entregar conteúdos que agregassem resultado ao usuário instantaneamente, estes precisariam estar apresentados de uma forma mais funcional, por isso, numa segunda etapa do projeto, viu-se a oportunidade da criação de *widgets* de alguns conteúdos. Para tal, um planejamento sobre o que mostrar, formato, tamanho e outras características foi feito.

Para cada *widget*, o serviço acessa o conteúdo disponibilizado na Internet convencional ou mesmo na Internet Móvel, busca apenas as informações desejadas que foram consideradas necessárias no planejamento do *widget* e depois cria um fragmento de código HTML para ser inserido na página do usuário, similar ao padrão dos "*.jmm*".

A seguir, pode-se ver a lógica da classe `WeatherWidgetAction` responsável por acessar o portal UOL Tempo para apresentar a previsão do tempo fornecida para uma cidade desejada e depois criar um *widget* com as respostas do serviço. A Figura 5.2 mostra como é este *widget*.

```
// (...) imports
public class WeatherWidgetAction implements ActionSupport {

	private HtmlConnector htmlConnector;
	private IParser weatherParser;

	public String execute() {
		// (...)

		try {
			Response response = HtmlConnector.post(weatherServiceURL,
									attributes, headers);
			String html = response.getHtml();
			WeatherInfo weatherInfo = weatherParser.parse(html);
		} catch (Exception e) {
			return Action.ERROR;
		}

		return Action.SUCCESS;
	}
	// (...) construtores, outros métodos, getters e setters
}
```

Fig 5.2 – Visualização do “.jmm” do UOL Tempo

Até o momento da escrita deste documento, todos *widgets* do portal estavam sob o domínio do próprio portal JMobi, mas o sistema é capaz de aceitar conteúdos de terceiros desde que eles estejam no formato definido e utilizado pelo componente *jmobi-mobilesite*.

5.1.4 Pré-visualização de páginas

O componente *jmobi-service* também possui uma quarta funcionalidade que é a de fornecer uma pré-visualização de como seria uma página criada com alguns conteúdos escolhidos. Esta funcionalidade é utilizada pelo componente *jmobi-editor*, que será apresentado logo em seguida, para dar uma amostra ao usuário de como seria a sua página durante o processo de criação da mesma.

Este serviço aceita pedidos de visualização da página mobile do portal construída com conteúdos que são enviados na mesma requisição da visualização. Ele espera que a solicitação contenha uma lista de conteúdos no formato JSON (similar ao mostrado na seção 5.1.2), para que interprete essas informações, resgate os objetos correspondentes salvos no banco de dados e depois os repasse para a camada de Visualização, que vai gerar um página simulando o mobile site.

5.2 JMOBI-EDITOR

Além do acesso pelo próprio celular, o usuário tem um grande contato com a interface *desktop* do portal. Para tentar criar uma interface fácil de usar, viu-se a necessidade da criação de uma aplicação RIA - *Rich Internet Application* (Aplicação de Internet Rica). Com isso, o desenvolvimento do componente *jmobi-editor* foi realizado com a plataforma Adobe Flex.

Essa parte do projeto é a que possibilita a visualização dos conteúdos e a escolha de tais para criação da página pessoal pelo usuário. Com ela o usuário escolhe conteúdos para serem adicionados ou removidos de sua página pessoal e pode também ver uma simulação de como ficaria o resultado final.

A interface foi elaborada pensando-se basicamente em três colunas, onde na primeira é possível ver todos os conteúdos disponíveis, na segunda é mostrado os conteúdos escolhidos pelo usuário para compor a página, e na terceira coluna é possível ter uma pré-visualização do que está sendo criado.

Reafirmando o propósito de aumentar a usabilidade para o usuário, a interface foi criada com componentes e ações já costumadamente utilizados e conhecidos da área de aplicativos para computadores (como característica de uma RIA), como por exemplo: componentes de *combo box* para listas, ações de arrastar e soltar (*drag and drop*) para movimentação de itens, configuração de teclas rápidas (como *Delete*), uso de ícones, e outros. A Figura 5.3 mostra o editor com as suas áreas de configuração e visualização das páginas criadas.

Fig 5.3 – Interface do *jmobi-editor*

Há dois estados de utilização do editor: com o usuário logado ou não no sistema. As funcionalidades são as mesmas para ambos os estados, mas quando o usuário está logado em sua conta, pode editar e salvar a sua página pessoal, diferente de quando não

está logado, podendo apenas simular e visualizar como ficaria a página em um modelo de dispositivo móvel.

Quando o editor é carregado ele faz requisições ao *web service* provido pelo componente *jmobi-service* para receber as categorias e conteúdos disponíveis ao usuário. Internamente a ferramente interpreta as respostas do serviço para adaptar o conteúdo e disponibilizar as informações nos componentes da própria ferramenta.

Depois que a aplicação foi carregada, ela própria mantêm internamente todas as informações referentes as configurações do usuário. Ela só se comunica com o *web service* provido pelo componente *jmobi-service* quando o usuário escolher salvar sua página ou visualizar o resultado de sua configuração.

As ações de salvar ou visualizar a página configurada possuem o mesmo fluxo: é executada a serialização dos conteúdos escolhidos no formato JSON e então enviadas para os respectivos serviços do *jmobi-service* via requisições HTTP. Conforme a opção do pedido e a resposta do serviço, a ferramenta informa se o salvamento foi efetuado com sucesso ou não, ou então mostra uma pré-visualização dos conteúdos escolhidos na interface de *mobile site*.

5.3 JMOBI-MOBILESITE

O último componente do projeto também trata-se de um *web service* que foi desenvolvido com as ajudas do Spring, Struts e Hibernate. Ele é o responsável pela disponibilização das páginas próprias para dispositivos móveis, seja as de um usuário ou as do próprio portal para mostrar todos os conteúdos das categorias.

Independente do tipo de página a entregar, o componente trabalha de forma similar: para cada conteúdo da determinada página que será mostrada, o serviço faz requisições para obter os fragmentos de código HTML (o código dos "*.jmm*") e os insere na resposta.

Abaixo é mostrado o fragmento de código da classe `UserPageAction` que faz parte da camada de Controle do *jmobi-mobilesite* e é a responsável por entregar as páginas dos usuários. Quando o usuário acessa o recurso de sua página no sistema (através de uma URL que contém o próprio nome dele), esta classe atende pela requisição e delega para a camada de Negócios obter os conteúdos da sua página. Estes serão repassados a camada de Visualização que vai realmente "construir" o código da página do usuário.

```
// (...) imports
public class UserPageAction implements ActionSupport {
```

```
	private IUserPageDAO userPageDAO;
	private UserPage userPage;

	public String retrieve() {
		userPage = userPageDAO.retrieve(user);

		if (userPage == null) {
			return "not_founded";
		}

		return Action.SUCCESS;
	}
	// (...) construtores, outros métodos, getters e setters
}
```

Na camada de Modelo, as mesmas classes do tipo DAO pertencentes ao *jmobi-service* (e importadas para este componente) trabalham para buscar os conteúdos salvos de um usuário no banco de dados. Elas então retornam os objetos do tipo `PageModule` da página do usuário.

Na JSP de resposta que faz a criação da página do usuário, para cada objeto `PageModule` é então feita uma requisição para o endereço que o conteúdo daquele objeto está e sua resposta, que é o fragmento de HTML, é importada na própria página, como pode ser observado no fragmento de código abaixo.

```
<%@ page pageEncoding="UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.0//EN"
                                "http://www.wapforum.org/DTD/xhtml-mobile10.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="application/xhtml+xml;charset=UTF-8" />
<meta http-equiv="Cache-Control" content="max-age=3600"/>
<title>${userPage.name } - JMobi - Smart Mobile Web</title>
</head>
<body>
<div id="h_title">
	<h1>${userPage.name}</h1>
</div>
<div id="content">
	<c:forEach var="pageModule" items="${userPage.pageModules}">
	<% try { %>
		<c:set var="address" value="${fn:replace(pageModule.url,' ','%20')}"/>
```

```
                <c:import url="${address}" />
        <% } catch(Exception e) { } %>
        </c:forEach>
</div><!-- /content -->
</body>
</html>
```

A página entregue para ser vista no celular do usuário foi elaborada seguindo os padrões e melhores recomendações sobre *mobile sites*. Conforme o tipo de conteúdo que o aparelho suportava, a maneira mais adequada era mostrada para garantir uma boa visualização, como foi explicado na seção 2.3 deste documento.

O componente *jmobi-mobilesite* também possui outras páginas para mostrar todos os conteúdos das categorias, possibilitando que o usuário navegue por quaisquer conteúdos se desejar, mesmo os que não estão cadastrados em sua página pessoal. A lógica nestas páginas é similar a JSP que cria a página de um usuário, fazendo requisições para importar o fragmento de código HTML de cada conteúdo para criar a página.

5.4 OUTROS COMPONENTES

No projeto alguns outros pequenos módulos ou serviços que já existiam são utilizados, como é o caso do serviço que fornece autenticação e autorização de usuários na base JMobi. Estes componentes não estão fortemente ligados ao projeto então não foram tratados com muita atenção, apenas sendo referenciados quando utilizados.

6 CONCLUSÃO

Este trabalho sobre o portal facilitador ao acesso a conteúdos da Internet Móvel abrangeu, em geral, as áreas de serviços para Internet e uso de dispositivos móveis (em especial os celulares) como meio e ferramenta de acesso a Internet.

Seus ganhos foram tanto teórico como prático. Do ponto de vista teórico, ele possibilitou aprofundar os conhecimentos técnicos sobre diversas tecnologias, ferramentas, metodologias e processos. Do ponto de vista prático, possibilitou um aperfeiçoamento das práticas de programação com novas ferramentas e conceitos arquiteturais.

O objetivo principal do projeto é facilitar o acesso a conteúdos da Internet Móvel através de uma única página pessoal. Para tal, foi preciso criar sistemas que possibilitassem a seleção de conteúdos especiais e que, com esses, se construísse uma página específica para ser visualizada em celulares.

Durante todo projeto, desde sua concepção até o seu lançamento em operação, diversos conhecimentos aprendidos na vida acadêmica foram importantes para seu desenvolvimento. Dentre várias disciplinas, pode-se comentar que as relacionadas a programação ajudaram o desenvolvimento prático e a implementação do código do projeto; as disciplinas de Banco de Dados colaboraram para a estruturação do modelo-relacional para a persistência das informações; a disciplina de Engenharia de Software deu embasamento para os processos de concepção, desenvolvimento e implementação do software; a disciplina de Programação Web apresentou os conceitos e tecnologias para a criação de serviços e softwares para a Internet; a disciplina de Engenharia de Usabilidade ajudou a criar um produto de boa qualidade e usabilidade para ser utilizado pelas pessoas na Internet.

Numa primeira fase do projeto, foram feitas pesquisas e revisão bibliográfica a respeito dos assuntos correlacionados aos temas trabalhados. Contextualizações e referências sobre os assuntos envolvidos foram apresentadas para explicar as partes teóricas do contexto do trabalho.

Em uma segunda fase foi realizado o planejamento e estruturação do projeto. O conceito e as funcionalidades do produto foram refinados e as tecnologias, ferramentas e processos para a criação do portal foram escolhidos para posteriormente se prosseguir com o desenvolvimento.

Na fase de implementação, cada parte do produto foi criada com iterações curtas, através de testes, podendo acompanhar o avanço do produto para seu estado idealizado e final.

6.1 AVALIAÇÕES FINAIS

Como descrito no decorrer deste documento, o projeto do portal JMobi foi desenvolvido por completo. Dos desejos iniciais do produto, todas suas funcionalidades foram implementadas, sendo entregue então um portal facilitador ao acesso a conteúdos da Internet Móvel.

O portal está em funcionamento e possui por volta de 4000 páginas de usuários criadas e sendo acessado por uma média de 700 pessoas únicas por dia. Os primeiros resultados do uso do portal mostraram uma boa aceitação pelos usuários, e percebeu-se também que os conteúdos servidos como *widgets* são mais utilizados por entregarem resultado mais rápida e facilmente.

Foi criado uma aplicação com uma interface web onde o usuário pode interagir para criar uma página personalizada com conteúdos de sua preferência e própria para ser acessada por dispositivos móveis. As páginas criadas (que fazem parte da interface *mobile* da aplicação) disponibilizam os conteúdos escolhidos de diferentes formas conforme o caso de cada um. O objetivo deste trabalho foi alcançado já que todas as etapas propostas foram cumpridas com sucesso.

Como resultado positivo deste trabalho, pode-se declarar a aprendizagem e aprimoramento sobre as tecnologias envolvidas no projeto, o que possibilitou conhecer ferramentas e conceitos amplamente usados principalmente no desenvolvimento de *web services* com a linguagem de programação Java. Tal aprendizado aproxima e deixa mais preparado o graduando para o mercado de trabalho.

Utilizar a tecnologia Adobe Flex não necessariamente precise ser considerado um ponto negativo, mas talvez tenha sido uma decisão de projeto arriscada. Apesar de aprender uma tecnologia totalmente nova, ela exigiu um alto custo para o projeto em questões de tempo gasto.

6.2 TRABALHOS FUTUROS

Apesar do portal já estar em operação, melhorias e incrementos podem ser feitos no projeto como trabalhos futuros. Uma funcionalidade que logo depois da criação mostrou sua necessidade, é a de uma interface de administração para criar e inserir novos conteúdos simples (apenas um link) de uma maneira fácil. Atualmente, para se criar um novo conteúdo, é preciso alterar alguns dos componentes e colocar em operação uma nova versão, mas esse processo é demorado e arriscado, já que está sendo alterado algo que já está em operação.

Uma outra opção de nova funcionalidade seria a de dar liberdade para o usuário inserir conteúdos em sua página que não estejam na base de dados do portal. Pela interface do *jmobi-editor* o usuário poderia ter a opção de inserir o endereço de um site para dispositivos móveis, e o sistema guardaria essa informação para automaticamente criar um "*.jmm*" para entregar aquele endereço específico ao usuário.

Outro incremento que poderia ser feito ao projeto seria ter as funcionalidades de edição da página do usuário diretamente pelo celular. De uma forma simples, o usuário poderia retirar ou inserir um novo conteúdo conforme navegasse pela sua própria página pessoal ou pelas outras páginas do portal.

Uma melhoria que já está implícita na idéia do produto, é a criação de novos conteúdos especiais, os *widgets*. Esses são os conteúdos que melhor entregam resultado efetivo ao usuário, então quanto mais e melhor evoluídos os *widgets* estiverem, melhor o usuário pode interagir e ter ganhos com o portal.

Apesar da interface das páginas específicas para dispositivos móveis ser adequada para diferentes aparelhos celulares (mesmo com diferenças como a de tamanho de tela), o componente *jmobi-mobilesite* poderia reconhecer o celular que está sendo usado para visualizar as páginas e melhor adequar sua interface conforme as suas características.

Ainda uma outra melhoria ao projeto, seria criar um aplicativo Java para os celulares que traria os conteúdos do portal por um outro meio, e com este possibilitando uma interface mais rica e com mais opções de interação com o usuário. Um exemplo de produto como este é o serviço e aplicativo Widsets[31]. Estas são apenas algumas idéias e sugestões de melhorias. O conceito do portal é bem abrangente e aceita diversas novas funcionalidades.

31 http://www.widsets.com/

REFERÊNCIAS

WIKIPÉDIA. **Cloud computing.** Disponível em <http://en.wikipedia.org/wiki/Cloud_computing>. Acessado em Agosto de 2009.

SANTOS, E. S. **Desenvolver aplicativos ou mobile sites?** Disponível em <http://webinsider.uol.com.br/index.php/2009/07/13/desenvolver-aplicativos-ou-mobile-sites/ >. Acessado em Julho de 2009.

SAARIKOSKI, V. **The Odyssey of the Mobile Internet.** Faculty of Science, University of Oulu,; (Dezembro 2006)

WIKIPÉDIA. **História do telefone celular.** Disponível em <http://pt.wikipedia.org/wiki/Hist %C3%B3ria_do_telefone_celular>. Acessado em Julho de 2009.

WIKIPÉDIA. **Mobile browser.** Disponível em <http://en.wikipedia.org/wiki/Mobile_browser>. Acessado em Julho de 2009.

WIKIPÉDIA. **Wireless Application Protocol.** Disponível em <http://en.wikipedia.org/wiki/Wireless_Application_Protocol>. Acessado em Julho de 2009.

TAFFE, F. A. **Análise de Requisitos e Ferramentas para Implementação de Sites de Comércio Eletrônico Móvel.** UFSC (Março, 2002)

WIKIPÉDIA. **Wireless Markup Language.** Disponível em <http://en.wikipedia.org/wiki/Wireless_Markup_Language>. Acessado em Julho de 2009.

W3C. **Web Services Architecture.** Disponível em <http://www.w3.org/TR/ws-arch/#whatis>. Acessado em Agosto de 2009.

WIKIPÉDIA. **Cliente-server.** Disponível em <http://en.wikipedia.org/wiki/Client-server>. Acessado em Agosto de 2009.

COSTELLO, R. L. **Building Web Services the REST Way.** Disponível em <http://www.xfront.com/REST-Web-Services.html>. Acessado em Setembro de 2009.

FIELDING. **Fielding Dissertation: CHAPTER 5: Representational State Transfer (REST).** Disponível em <http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm>. Acessado em Agosto de 2009.

HEMRAJANI, A. **Agile Java Development with Spring, Hibernate and Eclipse.** Developer's Library; 1 ed. (Maio, 2006).

FREEMAN, E.; FREEMAN, E; BATES, B.; SIERRA, K. **Head First Design Patterns**. O'Reilly Media, Inc.; 1 ed. (Outubro 25, 2004).

AHONEM, TOMI T. **M-Profits: Making Money from 3G Services.** John Wiley & Sons Ltd.; 1 ed.

BECK, K.; ANDRES, C. **Extreme Programming Explained: Embrace Change.** Addison-Wesley Professional; 2 ed. (Novembro 26, 2004).

AGILE MANIFESTO **Manifesto for Agile Software Development.** Disponível em <http://agilemanifesto.org/>. Acessado em Outubro de 2009.

BROWN, D.; DAVIS, C. D.; STANLICK S. **Struts 2 in Action.** Manning Publications Co. (2008)

BAUER, C.; KING, G. **Java Persistence with Hibernate.** Manning Publications Co. (2007)

CUI, Y.; ROTO, V. **How People Use the Web on Mobile Devices.** WWW 2008 / Alternate Track: Industrial Practice and Experience. China. (Abril 2008)