

Liga de Ensino do Rio Grande do Norte
Centro Universitário do Rio Grande do Norte
Especialização em Desenvolvimento de Sistemas Corporativos

Maria Emília Eidelwein

**Eventos do MP-RN: Um projeto de introdução ao M-Government com Android e
Webservices RESTful**

Natal/RN
2012

Maria Emília Eidelwein

Eventos do MP-RN: Um projeto de introdução ao M-Government com Android e Webservices RESTful

Relatório técnico apresentado como requisito parcial para obtenção de título de especialista em Desenvolvimento de Sistemas Corporativos, no Centro Universitário do Rio Grande do Norte

Orientador: Gibeon Aquino

Natal/RN
2012

RESUMO

As instituições públicas evoluíram do modelo burocrático para o modelo gerencial de administração, buscando traçar planos estratégicos que mostrem onde a instituição deseja chegar e como alcançar as metas que foram estabelecidas. O MP-RN não é diferente dos demais órgãos públicos e visa tornar-se mais acessível à população e modernizar-se tecnologicamente. Uma maneira de atingir esses objetivos é o investimento em tecnologias para dispositivos móveis, uma tendência tecnológica e meio de comunicação que se torna cada vez mais presente e é cada vez mais utilizada. O presente trabalho tem como objetivo relatar tecnicamente o desenvolvimento do primeiro projeto para dispositivos móveis da instituição, permitindo uma prospecção para a melhoria dos serviços e da acessibilidade da instituição para a população.

Palavras-chave: Android; REST; M-Government.

ABSTRACT

The public institutions are evolving in its administration from bureaucratic model to management model, seeking to make strategic plans in order to show what the institution desires to achieve and how to accomplish the goals that had been established. MP-RN is not different from the other agencies and aims to become more available to the population and update itself and adhere to new technologies. A way to achieve those goals is to invest in mobile technologies, a technological trend and medium that becomes increasingly present and used. The current work has the objective to report technically the development of the first mobile project of the institution, allowing to prospect its services improvement and availability to benefit the population.

Keywords: Android; REST; M-Government.

LISTA DE FIGURAS

Figura 1 Parte do Sistema de Gerenciamento de Eventos do MP-RN.....	11
Figura 2 Exemplo de um QR Code Fonte: http://www.qrcode.com/en/	18
Figura 3 Arquitetura do sistema.....	20
Figura 4 Diagrama das classes de domínio.....	21
Figura 5 Classes de encapsulamento.....	22
Figura 6 Estrutura do projeto Android.....	25
Figura 7 Diagrama das classes cliente de serviço.....	26
Figura 8: Teste de Leitura de QR Code em dispositivo.....	27
Figura 9 Folha de check-in com QR Code.....	29
Figura 10: Tela principal do aplicativo.....	30
Figura 11: Lista de eventos.....	30
Figura 12: Detalhes de evento.....	31
Figura 13: Identificação via CPF é um ponto central para o sistema.....	31
Figura 14: Tela de inscrição em evento - Informações corporativas.....	32
Figura 15: Tela de Inscrição em Evento - Dados Pessoais.....	32
Figura 16 Tela "Meus eventos".....	33
Figura 17 Tela de consulta de carga horária.....	33
Figura 18 Exemplo de certificado.....	34

SUMÁRIO

1.Introdução.....	9
2. Proposta.....	10
3.Objetivo.....	12
4.Metodologia.....	12
5.Tecnologias envolvidas.....	14
6.Desenvolvimento do Projeto.....	19
7.Conclusão.....	35
REFERÊNCIAS BIBLIOGRÁFICAS.....	37

1. Introdução

M-Government ou Governo Móvel é uma área de complemento dentro do **E-Government** (Governo eletrônico) que trata do uso pelo governo de tecnologias móveis, como dispositivos móveis, *smartphones*, infraestrutura de redes *Wifi* e 3G, e que possibilita o uso de serviços do governo em qualquer lugar e hora, sem depender de um computador ou estação de trabalho [1].

Pesquisas têm revelado um crescimento cada vez maior no uso de dispositivos móveis, devido à evolução nas redes móveis e nas tecnologias para estes dispositivos, podendo o cliente, através do **M-Government** (ou Governo Móvel) ter acesso a serviços públicos em qualquer lugar e hora. O governo também cresce com o **M-Government** através da promoção de melhorias em suas atividades operacionais [2]. Disponibilização de informações ao público é uma tarefa chave do governo e a evolução das redes e tecnologias móveis criou um novo canal para fornecer informações de maneira efetiva e a baixo custo. Cidadãos necessitam de informação e o fornecimento desta pelo governo promove democracia e cria *accountability*, no que é chamado de **G2C** (Government to citizens – Governo para cidadãos) ou o inverso, **C2G**, no qual o cidadão fornece informações ao governo. [1] Um exemplo de sucesso ocorreu na Polícia Nacional das Filipinas, onde em 2003 foi implantado um sistema mobile para realizar reclamações de criminosos e ações abusivas por parte de policiais. Em um ano de funcionamento, o sistema já registrava mais de 29,000 registros de usuários [1]. Outros exemplos de sucesso são: o aplicativo dos Bombeiros de Minas Gerais para registrar e transmitir dados de segurança de um prédio, o aplicativo do Tribunal de Justiça do RJ para informar advogados da tramitação de um processo e o aplicativo do Tribunal de Justiça de Pernambuco para consultas processuais [3].

A visão do Ministério Público do Rio Grande do Norte atualmente é:

“Ser reconhecida como instituição independente, catalisadora, dinâmica, eficiente e eficaz, responsável e referente na defesa dos interesses da sociedade e na valorização efetiva de seus integrantes, com capacidade de contribuir na construção da justiça

social e da sociedade com atuação equânime em todo o Estado do Rio Grande do Norte”.

Uma visão, do ponto de vista estratégico, mostra aonde uma instituição deseja chegar. No caso do MP-RN, nota-se que este deseja ser dinâmico, eficiente e eficaz e ser capaz de contribuir para a sociedade com atuação equânime em todo o estado.

O Ministério Público do Rio Grande do Norte atualmente não possui nenhum serviço específico para **M-Government** e precisa dar o passo inicial para tornar mais acessíveis os serviços da instituição. O projeto apresentado neste trabalho visa dar o primeiro passo do MP-RN nesse sentido, buscando não só habilitar sua equipe a produzir soluções para Governo Móvel como de fato tornar a instituição mais visível, transparente, eficiente, eficaz e acessível ao público. Os eventos do MP-RN trazem informação e orientação à sociedade e se mostram como um meio de implantar o Governo Móvel no MP-RN e criar uma cultura institucional de estar sempre acessível, seja através de seu local físico, computador ou *smartphone*.

2. Proposta

A instituição vem investindo em novas tecnologias para tornar mais eficiente a execução de tarefas da área fim, buscando investir em novos sistemas corporativos. Este capítulo apresenta o contexto atual dos sistemas na instituição e o sistema que será o escopo deste trabalho.

2.1. Situação atual do MP-RN

O MP-RN, no momento, apresenta seu catálogo de sistemas voltado prioritariamente para *web*, devido à necessidade de acesso multiplataforma de suas aplicações. Todavia, o acesso aos recursos informáticos do MP-RN por meio de dispositivos móveis ainda é restrito, pois nem seu portal e a *intranet* obtiveram versões adaptadas para dispositivos móveis.

Foi pensando em ampliar o acesso móvel desta instituição, que foi elencado pela equipe um projeto de inicialização, escolhendo o Sistema de Gerenciamento de Eventos e a plataforma **Android**.

2.2. Sistema de Gerenciamento de Eventos

O Sistema de Gerenciamento de Eventos é um sistema Java para *web* que controla cadastro, inscrição e emissão de certificados para eventos promovidos pelo MP-RN. A instituição realiza anualmente diversos eventos para qualificação e avanço de servidores, aprimoramento de seus membros e para a comunidade em geral.

Os eventos têm data de início e fim, quantidade máxima de inscritos e carga horária definidos, e podem depender de prévio deferimento por parte do CEAF (departamento responsável pelos eventos) das inscrições realizadas. Ao se inscrever em um evento, o usuário preenche um formulário com seus dados pessoais e institucionais e no comparecimento do evento, caso deferido, assina uma lista que será então utilizada pelo CEAF para liberação de certificados e contabilização de carga horária.

O sistema atual (Figura 1) apresenta a parte administrativa do sistema, incluindo cadastro/edição de eventos, deferimento de inscrições e liberação de certificados. A parte visível ao público é a inscrição em eventos, consulta de carga horária e emissão de certificados, sendo o sistema desenvolvido com tecnologias **JSF** (Java Server Faces) com **Hibernate** e **JBoss Seam** rodando sobre **JBoss AS 5**.



Código da Inscrição	Evento	Período de Realização	Carga Horária (Hs.)	Classificação	Interno?	Inscrição deferida?	Participação Validada?	Opções
13094	Palestra Motivacional - Todo é Dia de Recomeçar	17/08/2012 a 17/08/2012	2	Curso Oficial	SIM	SIM	NÃO	
12530	ANÁLISE E MELHORIA DE PROCESSOS	14/08/2012 a 17/09/2012	40	Curso Oficial	NÃO	NÃO EXIGE DEFERIMENTO	NÃO	
12294	Gerência de Projetos: Teoria e Prática (turma de julho a setembro)	31/07/2012 a 03/09/2012	40	Curso Oficial	NÃO	SIM	NÃO	
12379	CURSO DE SEGURANÇA INSTITUCIONAL	08/06/2012 a 08/06/2012	4	Curso Oficial	SIM	NÃO EXIGE DEFERIMENTO	SIM	

Figura 1 Parte do Sistema de Gerenciamento de Eventos do MP-RN

3. Objetivo

Desenvolver um aplicativo para dispositivos móveis que contemple a parte cliente do Sistema de Gerenciamento de Eventos. Esse aplicativo funcionará dependendo de acesso à Internet e conterá funcionalidades de consulta de eventos, listagem dos eventos do usuário (e seu acompanhamento), consulta de carga horária do usuário, *download* de certificados e principalmente: apresentação no evento pelo usuário de uma maneira simples e sem possibilidade de fraudes.

4. Metodologia

A Tabela 1 mostra o cronograma do projeto, que iniciou-se em maio de 2012, partindo da definição da área de estudo e projeto. Escolheu-se a elaboração de um projeto para dispositivos móveis para o MP-RN. Nesse mesmo mês definiu-se o projeto e seu escopo e iniciou-se as fases de **Pesquisa Bibliográfica** e **Especificação**, esta última realizando a documentação de casos de uso e protótipos de tela do aplicativo. No mês de maio também iniciou-se a **Pesquisa de tecnologias**, escolhendo-se a plataforma **Android** para o desenvolvimento do aplicativo.

A fase **Pesquisa de Tecnologias** testou ferramentas para geração de **QRCode** e para publicação e requisição a *webservices* sobre o protocolo HTTP. Foi escolhido o estilo arquitetural **REST** e o *framework* **RETEasy** para desenvolvimento e publicação dos *webservices* **RESTful**.

Em paralelo, iniciou-se a fase de **Implementação**, que foi responsável pela implementação do código de servidor e cliente, i.e., dos serviços e do aplicativo.

Após considerável período de implementação, iniciou-se a fase de **Testes**, de maneira a validar as regras de negócio e testar o funcionamento do serviço, da comunicação entre aplicativo e serviço e do aplicativo no Emulador, em um *Smartphone* e um *Tablet*. Em paralelo com os testes, iniciou-se a **Elaboração do Relatório**. Por fim, a elaboração e exposição da **Apresentação**.

Fases do Trabalho	2012					
	MAIO	JUN	JUL	AGO	SET	OUT
Pesquisa bibliográfica						
Especificação						
Pesquisa de tecnologias						
Implementação						
Testes						
Elaboração do relatório						
Apresentação						

Tabela 1 Cronograma do projeto

5. Tecnologias envolvidas

Esta seção apresentará de maneira sucinta as tecnologias e *frameworks* escolhidos para a implementação da solução deste trabalho.

5.1. Android SDK

Dentre as opções de plataforma para dispositivos móveis, o **Android** foi criado pelo **Google®** em parceria com a **Open Handset Alliance** (OHA) – grupo formado por empresas do mercado de telefonia celular cujo objetivo é definir uma plataforma única e aberta para dispositivos móveis, sendo baseado em sistema operacional **Linux**. O ambiente operacional do **Android** traz diversos aplicativos pré-prontos, além da integração com os serviços do **Google®**. Por ter código aberto e ser livre, a plataforma pode ser customizada por qualquer usuário ou empresa, sem ônus de aquisição de código [4].

Sistema Operacional	Aquisições (milhões)	Fatia de Mercado (2011)	Crescimento Anual
Android	237.7	48.8%	244%
iOS	93.1	19.1%	96%
Symbian	80.1	16.4%	-29.1%
BlackBerry	51.4	10.5%	5%
Bada	13.2	2.7%	183.1%
Windows Phone	6.8	1.4%	-43.3%
Outros	5.4	1.1%	14.4%

Tabela 2 Mercado de dispositivos móveis em 2011. Fonte: <http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats/a#smartphoneos>

O mercado corporativo de dispositivos móveis também cresce para agilizar seus negócios e visa integrar aplicações móveis com seus sistemas informacionais, pois seus usuários buscam facilidade no acesso e usabilidade [2].

Conforme pesquisa realizada pela empresa comScore®, os dispositivos móveis obtiveram crescimento de cerca de 300% na participação em acessos à Internet entre maio de 2011 e maio de 2012, atingindo 2,4% do tráfego total [5]. Segundo a empresa Millennial Media®, no ano de 2011, o **Android** superou o **iOS** como sistema operacional para dispositivos móveis, sendo que 56% dos dispositivos possuem sistema operacional

Android contra 28% do **iOS** [6]. Pesquisas realizadas pela Canalys® afirmam quem em 2011 a fatia de mercado do **Android** foi de 48.8% com um crescimento anual de 244%, deixando claro que o **Android** tem sido o sistema operacional mais utilizado para este segmento [7], o que pode ser devido à tentativa de uniformização de plataforma estabelecida pelo **Google**® e a disponibilização de seu código fonte para uso e customização. A Tabela 2 mostra os principais sistemas operacionais para dispositivos móveis e sua participação no mercado no ano de 2011, segundo pesquisa realizada pela Canalys® [7].

O **Android** foi baseado no **Kernel 2.6** do **Linux** e apresenta, para a execução de seus aplicativos, uma máquina virtual chamada **Dalvik** que roda arquivos no formato **.dex** (Dalvik Executable). O **Android SDK**, o pacote de desenvolvimento para **Android**, foi baseado em Java, sendo os arquivos **.class** (*bytecodes*) convertidos para **.dex** quando a aplicação é compilada, sendo posteriormente compactados com os demais arquivos da aplicação (imagens, vídeos, sons, etc) em um pacote com a extensão **.apk** (Android Package File). Esse pacote pode ser publicado ou distribuído para uso nos dispositivos móveis [4].

O **Android SDK** possui diversas bibliotecas com recursos prontos, estimulando o reuso e a distribuição do conhecimento. Possui bibliotecas de integração com mapas, interface gráfica e 3D, sons, leitura de códigos de barras, conexão com serviços, entre outros, tornando-se praticamente completa e otimizando o tempo e esforço de codificação [4].

5.2. JBoss e Resteasy

O **JBoss AS** é um servidor de aplicações de código aberto que implementa **Java EE**, i.e., a versão corporativa da plataforma Java. O **JBoss** traz consigo uma série de perfis de servidores pré-configurados, desde um perfil mínimo até um perfil de servidor completo. Sendo sua arquitetura baseada em componentes, possui alta flexibilidade, permitindo realizar a publicação de *beans* (componentes corporativos), *webservices*, páginas e aplicativos em *thread*, além de possuir controle para diversos *clusters*. Assim, pode-se fornecer aplicações completas, integradas e de alto desempenho para

corporações, bastando-se realizar configurações necessárias e publicar as aplicações conforme os padrões **Java**, utilizando as bibliotecas fornecidas pela comunidade [8].

REST é um acrônimo criado por Roy Fielding para o estilo arquitetural de comunicação via rede chamada Representational State Transfer, devido à própria arquitetura da **World Wide Web** ser composta de recursos e **URLs**. O **REST** busca captar uma imagem da *web* bem estruturada e de suas melhores práticas, mas **REST** não é um padrão, embora utilize padrões como HTTP, URL, XML e MIME types [9]. **REST** apresenta elementos arquiteturais ignorando implementação e sintaxe de protocolos utilizados, focando apenas nos papéis dos elementos, que são: elementos de dados (recursos, identificadores de recursos, representações, dados de controle, etc), conectores (cliente, servidor, *cache*, tunel, etc) e componentes (servidor de origem, *gateway*, *proxy* e agente de usuário) [10].

Com isso, surgiram os *webservices* REST, que possuem algumas características essenciais como: comunicação cliente-servidor, *stateless* (não armazena estado), uso de *cache* para respostas, interface uniforme (operações HTTP GET, POST, PUT e DELETE), recursos nomeados como URLs e disposição de componentes em camadas. Desta maneira, um cliente pode utilizar um recurso disponível em um servidor invocando a URL deste recurso através da utilização de *webservices* [9].

RESTEasy é um projeto do **JBoss** que vem implementando diversas bibliotecas para auxiliar na construção de *webservices* e aplicações REST, porque implementa a **JAX-RS**, especificação para *webservices* Restful sobre a camada HTTP. **RESTEasy** pode executar tanto em um container de *servlets* como no servidor JBoss AS. **RESTEasy** faz uso de **Java Annotations** para facilitar o desenvolvimento, como exemplo: **@Path** (caminho do recurso), métodos HTTP (**@GET**, **@POST**, **@PUT** e **@DELETE**), **@Consumes** (o tipo de dados que recebe, que pode ser XML, JSON, entre outros), **@Produces** (o tipo de dados que retorna) e **@TypeParam** (parâmetro recebido) são algumas das anotações possíveis de serem utilizadas. Para a publicação basta adicionar as bibliotecas do **RESTEasy** ao projeto, criar a classe com os serviços e suas anotações

e publicar o projeto através do JBoss AS. Para acessar o serviço, basta acessar a URL definida pela sua anotação **@Path** com os devidos parâmetros requisitados [11].

5.3. HttpClient API

A **HttpClient API** faz parte da suíte de bibliotecas da apache chamada **Http Components**. Esta biblioteca apresenta implementações dos métodos do HTTP versões 1.0 e 1.1, possuindo suporte ao uso de **HTTPS**, i.e., HTTP com segurança através do **TLS**. Através dessa API pode-se realizar requisições GET e POST para páginas, *webservices* e outros recursos de arquitetura REST, bastando-se para isso definir a URL de conexão e os parâmetros. **HttpClient** apresenta alguns dos recursos que se pode encontrar em um *browser* como Gerenciador de Cookies, Gerenciador de Senhas e o Proxy, pelo qual pode ser executado o protocolo HTTP. A API apresenta algumas classes prontas para realizar as requisições e obter as respostas, como **HttpRequest**, **HttpResponse**, **Entity**, **Session** e **Cookies**, as quais encapsulam atributos e métodos de uma requisição ou objeto HTTP. Pode ser utilizada em qualquer projeto Java, seja ele Desktop, Web ou para **Android** e possui código aberto [12].

5.4. Zxing e QR Code

ZXing é uma biblioteca de código aberto para geração, leitura e processamento de diversos formatos e padrões de código de barra. Foi codificada em Java mas possui portabilidade para outras linguagens como C++, C#, JRuby, ActionScript e iPhone e suporta formatos como QR Code, UPC-A, Codabar e EAN-8. A API é dividida em diversos componentes como **core** (que realiza a codificação em imagens), **javase** (para clientes Java), **android** e **android-integration**, possibilitando-se integração com o aplicativo **Barcode Scanner** para leitura de diversos formatos [13].

O **QR Code** é um padrão de codificação de dados 2D criado em 1994 por Denso Wave (da Denso Corporation®), o qual possui uma representação gráfica no padrão da Figura 2. Enquanto o código de barras apresenta apenas dados em uma dimensão (horizontal), o QR Code apresenta dados nas duas dimensões (matricial), permitindo-se armazenar uma quantidade maior de dados por área e uma leitura de código de alta

velocidade, pois pode ser lido 360° (omnidirecional) e possui padrões de detecção de posição (em número de 3, que se localizam nos cantos superior esquerdo, superior direito e inferior esquerdo). É padronizado segundo AIM International, JIS e ISO e permite armazenar até 7,089 caracteres numéricos ou até 4,296 caracteres alfanuméricos, além de ter capacidade de restauração de até 30% dos dados caso ocorra uma falha ou rasura na representação gráfica do código. Ainda existe o Micro QR Code, para quantidades de dados menores (até 35 caracteres alfanuméricos) e que ocupe menor espaço físico, possuindo apenas um padrão de detecção de posição, localizado no canto superior esquerdo do código [14].

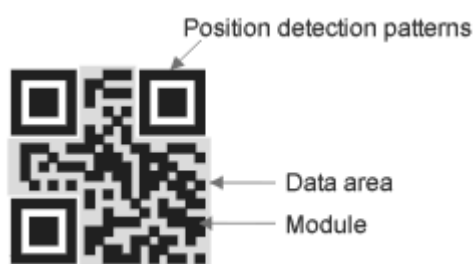


Figura 2 Exemplo de um QR Code Fonte:
<http://www.qrcode.com/en/>

6. Desenvolvimento do Projeto

6.1. Especificação

Como passo inicial, foi definido o escopo do projeto baseado no que já existia em quesito de funcionalidades do Sistema de Gerenciamento de Eventos, excluindo-se, entretanto, a parte administrativa do Sistema de Gerenciamento de Eventos. O escopo foi definido no Documento de Aprovação de Escopo, modelo definido pela equipe de desenvolvimento do MP-RN e baseado no **RUP**. Foi feito o protótipo inicial de telas do aplicativo segundo as funcionalidades descritas no escopo.

A partir da definição do escopo, foram elicitados e descritos os seguintes casos de uso: Listar eventos, Realizar inscrição em evento, Ver eventos do usuário, Realizar Comparecimento (*Check-in*), Gerar Certificado e Consultar Carga Horária. A documentação dos casos de uso utilizou o padrão definido na instituição, baseado no RUP, contendo fluxo principal, fluxos alternativos, fluxos de exceção, pré-condições, pós-condições e pontos de extensão.

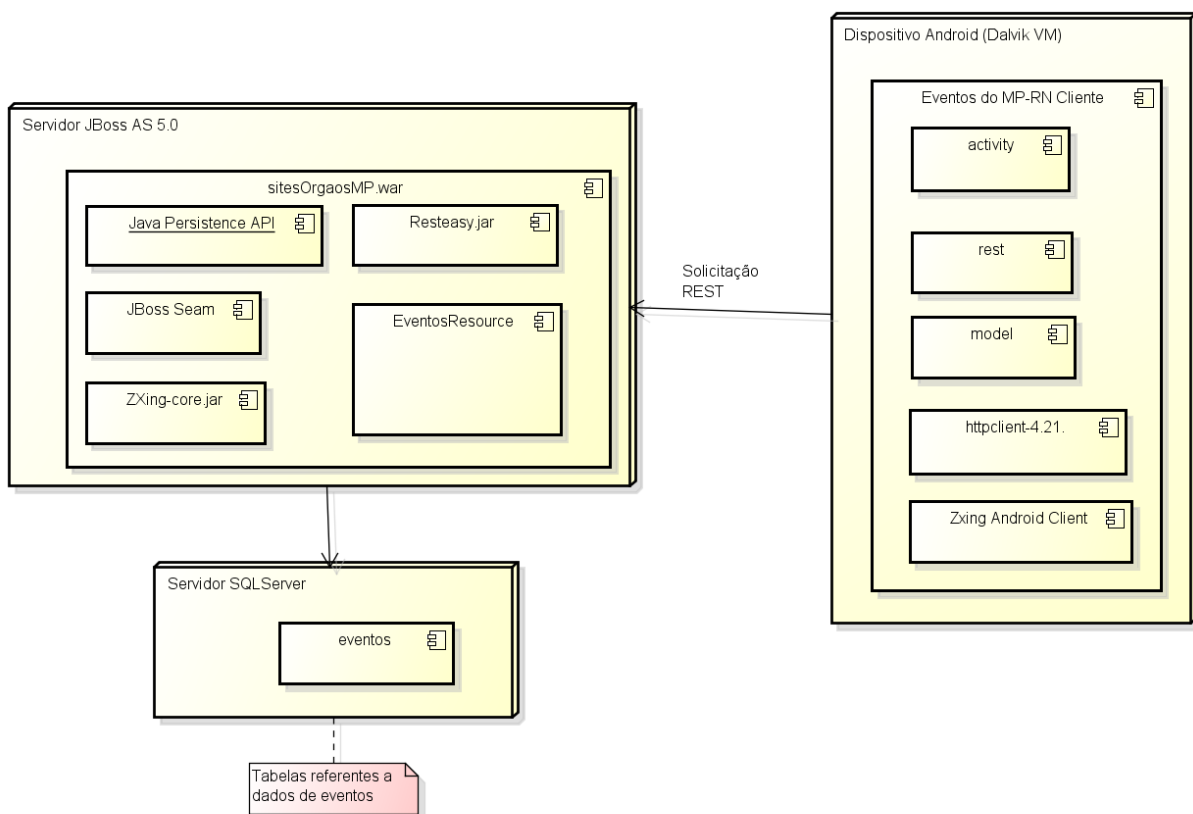
6.2. Arquitetura

Após o processo de especificação, iniciou-se a fase de definição de arquitetura e tecnologias. O banco de dados seria o mesmo utilizado para o código pré-existente do Sistema de Gerenciamento de Eventos, um *schema* no SQL Server 2008, contendo tabelas para eventos, usuários e inscrições. A plataforma *mobile* escolhida foi *Android* por ser aberta e pela experiência na equipe com Java, além de sua ampla utilização e a existência de ferramentas como a Eclipse IDE para desenvolvimento, que já eram de conhecimento da equipe de desenvolvimento. Definiu-se também o **QR Code** como padrão para código de representação, pela quantidade de dados possível de ser armazenada para realização de *check-in* em evento além da rapidez da leitura do código.

Com isso, passou-se ao estudo de quais seriam as tecnologias utilizadas para se obter e gravar dados no banco de dados. Como opção de segurança foram sugeridos

SOAP e REST, e este último foi escolhido como estilo arquitetural para criação dos *webservices* que estariam do lado servidor. Para geração de *webservices*, foram sugeridas e testadas as ferramentas **Jersey**, **CXF** e **RESTEasy**, sendo esta última escolhida por ser a biblioteca com maior compatibilidade com **JBoss**, o servidor de aplicação utilizado na instituição e que hospeda o atual Sistema de Gerenciamento de Eventos. Para o cliente **Android**, verificou-se que a melhor solução para realizar requisições HTTP seria a **HTTPClient API**, pela prévia experiência da equipe com a biblioteca e sua integração com **Android**.

Para a leitura e geração de **QR Codes**, foi escolhida a biblioteca **ZXing**, que apresentava documentação acessível e interface simples de se utilizar, além da compatibilidade com Java e **Android**, linguagens já determinadas previamente. Com isso, foi possível se desenhar a arquitetura do sistema, que pode ser visualizada na Figura 3.



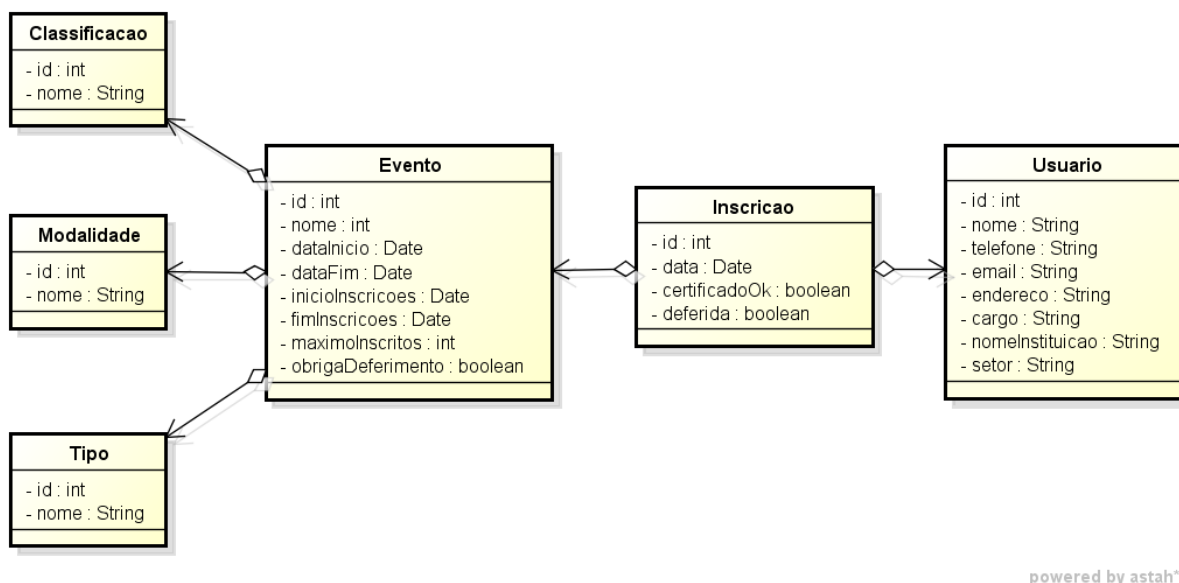
powered by astah

Figura 3 Arquitetura do sistema

6.3. Codificação

6.3.1. Classes de domínio

O sistema já possuía previamente as classes de domínio que mapeiam via **Hibernate** o *schema* de banco de dados de evento. As classes e seus relacionamentos podem ser visualizados na Figura 4. As classes centrais são **Evento**, **Inscricao** e **Usuario**, representando o domínio atual do sistema de eventos.



powered by astah

Figura 4 Diagrama das classes de domínio

6.3.2. Classes de encapsulamento

Para diminuir a complexidade dos dados e remover relacionamentos e atributos desnecessários ao desenvolvimento do aplicativo, foram criadas as classes de encapsulamento, que serão utilizadas para enviar requisições HTTP e retornar dados, ao serem convertidas em XML. Essas classes possuem a anotação **@XMLRootElement** e podem ser visualizadas na Figura 5. Uma classe de encapsulamento com menos atributos (**EventoWrapperSimples**) foi criada para listar muitos eventos apenas com seus dados básicos e também foi criada uma classe que estende **EventoWrapperSimples**, que apresenta os detalhes de evento (**EventoWrapper**). Essas classes são transformadas em XML pelo **RESTEasy** e podem também serem recebidas via XML e transformadas automaticamente em instâncias de suas respectivas classes.

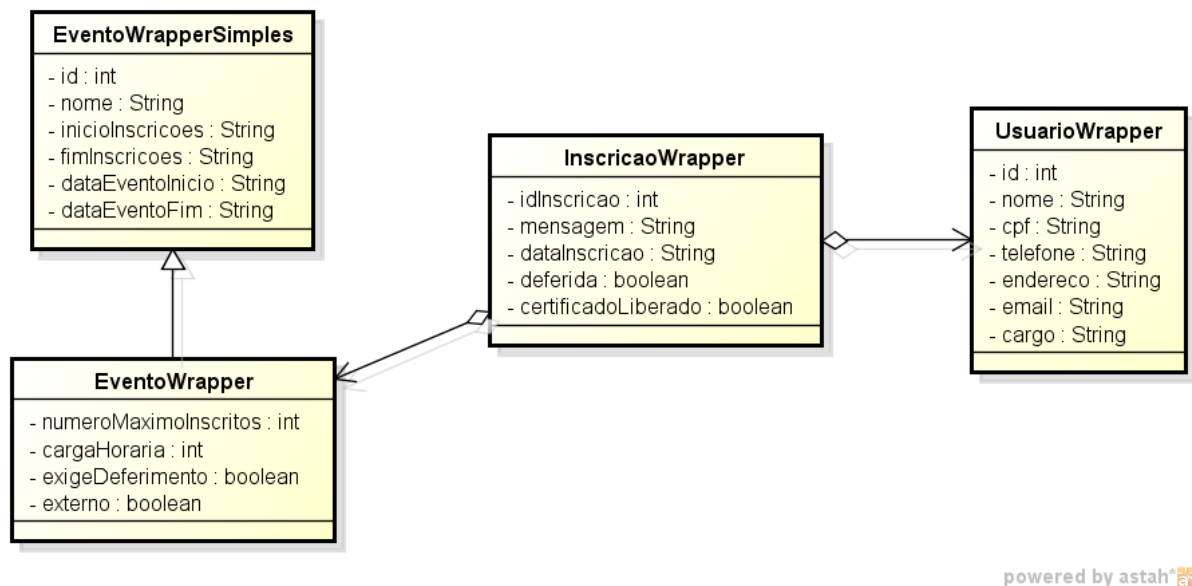


Figura 5 Classes de encapsulamento

6.3.3. Classe de serviço

A partir da definição da arquitetura, foi iniciada a codificação dos *webservices* utilizando-se **REStEasy**, escolhendo o XML como formato de metadados. Os serviços foram codificados na classe **EventosResource**, buscando definir *URLs* e o mínimo de parâmetros possível. A seguir eram publicados e testados via *URL* no navegador, depurando o código quando possível em busca de erros e para tratar exceções que aparecessem, visto que os resultados no aplicativo **Android** muitas vezes não esclarecem com precisão o que ocorreu no lado do servidor. A estrutura da classe **EventosResource** é a mostrada na Listagem 1.

```

@Path("/eventos")
@Name("eventosResource")
@Scope(ScopeType.CONVERSATION)
public class EventosResource extends ResourceHome<EventoWrapper, Long>{

    @Context
    UriInfo uriInfo;
    @Context
    Request request;
    public EventosResource() {
        setMediaTypes(new String[] {
            "application/xml", "application/json", "applic-
            ation/fastinfoset"
        });
    }
    [...]
}

```

Listagem 1: Código da Classe de Serviço

A classe **EventosResource** estende **ResourceHome** e define uma anotação **@Path** para o caminho `/eventos`. O método `setMediaTypes` define os tipos de conteúdo que o serviço pode retornar, no caso: XML, JSON e FastInfoset. Essa classe contém diversos métodos de serviço, cada um com anotação **@Path** definindo a *URL* padrão e, caso retornem XML para a requisição, a respectiva classe de encapsulamento é retornada pelo método. Exemplo de método pode ser visto no trecho de código da Listagem 2.

```
@BadgerFish
@GET
@Path("/lista_proximos_eventos/{qtd}")
@Produces("application/xml")
@Transactional
public List<EventoWrapperSimples> getProximosEventos(@PathParam("qtd")
int qtd) {

    if (qtd <= 0) {
        qtd = 3;
    }

    List<EventoWrapperSimples> eventos = new ArrayList<EventoWrapper-
    Simples>();

    EventoDao eventoDao = (EventoDao) Component.getInstance("eventoDao");

    List<Evento> eventosBanco = eventoDao.listarProximosEventos(qtd);

    SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");

    for (Evento ev: eventosBanco) {

        EventoWrapperSimples e = new EventoWrapperSimples();

        e.setNome(ev.getNome());

        e.setFimInscricoes(sdf.format(ev.getDataFimInscricoes().getTime()));

        e.setInicioInscricoes(sdf.format(ev.getDataInicioInscricoes().getTime()
        ));

        e.setId(ev.getIdEvento());

        e.setDataEventoFim(sdf.format(ev.getDataEventoFim().getTime()));

        e.setDataEventoInicio(sdf.format(ev.getDataEventoInicio().getTime()));

        eventos.add(e);
    }
    return eventos;
}
```

Listagem 2: Exemplo de método de serviço

O trecho de código da Listagem 2 retorna a lista com os “qtd” próximos eventos, onde “qtd” é um inteiro passado como parâmetro. O método utiliza **Hibernate** para consultar no banco de dados os eventos e retorna uma lista de objetos encapsulados de evento que serão retornados em formato XML, fazendo uma requisição HTTP à URL a seguir:

http://<servidor>:8080/eventos/seam/resource/rest/eventos/lista_proximos_eventos/3

6.4. Código do Aplicativo

O aplicativo teve sua implementação iniciada com a criação da interface com o usuário. As telas são descritas em arquivos *XML* de *layout* e todos os recursos podem ser definidos em arquivos de *strings* padronizados, como cores e imagens do aplicativo. A estrutura do projeto pode ser visualizada na Figura 6.

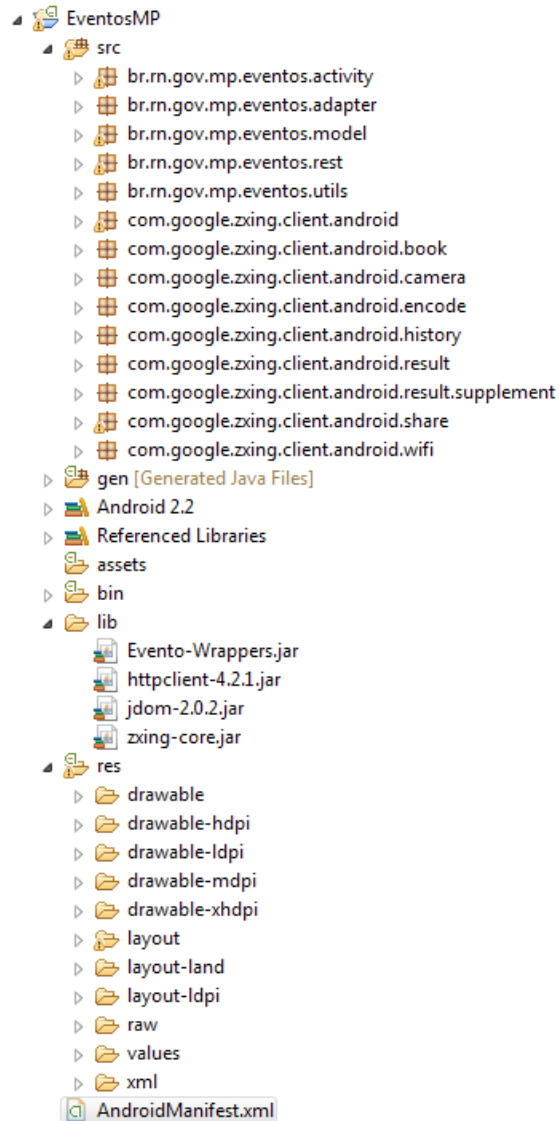


Figura 6 Estrutura do projeto Android

Na pasta *src* são postas as classes java, dentre elas as classes Activity (*br.rn.gov.mp.eventos.activity*), as classes de adaptação visual de lista (*br.rn.gov.mp.eventos.adapter*), as classes de modelo (*br.rn.gov.mp.eventos.model*), as classes de comunicação com os **webservices RESTful** (*br.rn.gov.mp.eventos.rest*), e as

classes utilitárias (br.rn.gov.mp.eventos.utils), além do código do cliente **ZXing** para **Android**, que foi baixado e compilado juntamente com a aplicação.

Para conexão com o *webservice*, foi criada a classe **EventosRestClient**, que possui métodos para formular requisições HTTP aos diversos serviços, com a construção dos parâmetros e leitura e transformação das respostas utilizando **JDOM**. Essa classe utiliza-se dos métodos **callWebService()** e **sendHttpPost()** da classe **RestClient**, que constrói requisições HTTP utilizando a biblioteca **HttpClient**. A Figura 7 mostra a relação entre classes e seus principais métodos. A classe **EventosRestClient** é, por fim, utilizada pelas classes **Activity** para trazer os dados para a visualização no aplicativo.

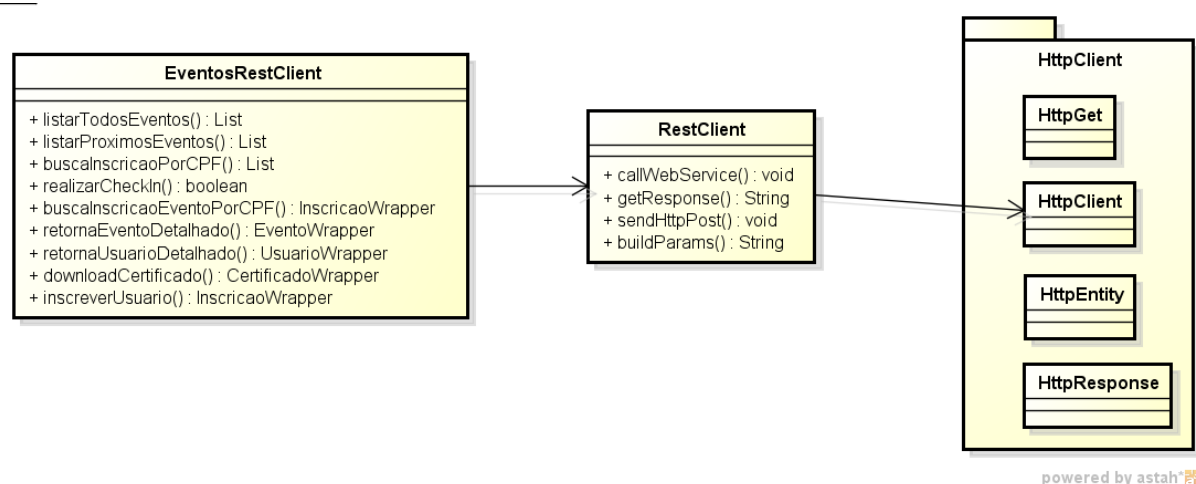


Figura 7 Diagrama das classes cliente de serviço

A pasta *gen* contém a classe **R.java**, que é gerada automaticamente pela aplicação no momento de sua compilação e contém as referências para seus recursos (imagens, *layouts*, cadeias de caracteres).

A pasta *lib* contém as bibliotecas utilizadas, que são: **Evento-Wrappers.jar**, com os objetos encapsulados; **HttpClient**, para realizar requisições HTTP aos *webservices*; **JDOM**, para leitura do XML; e **ZXing-core**, para leitura de **QR Code**.

A pasta *res* contém todos os recursos da aplicação sendo constituída dos seguintes subdiretórios: *drawable*, para inserir imagens em diversos tamanhos de tela e resolução; *layout*, arquivos XML que definem a disposição e os componentes de cada tela; *raw*, para

mídias; *values*, para padronização de cadeias de caracteres; e *xml*, que define preferências o aplicativo.

6.5. Testes

Como o aplicativo foi desenvolvido para contemplar a versão 2.2 do **Android** como a versão mínima para execução, testes foram executados no emulador versão 2.2 do **Android**.

O aplicativo também passou por testes exploratórios e de layout nos seguintes dispositivos:

- Samsung ® Galaxy Y (modelo: GT-S5360B)
- Samsung ® Galaxy Tab 10.1 (modelo: GT-P7500)

Um exemplo de teste pode ser visualizado na Figura 8, no qual se faz a leitura de um **QR Code** de evento através do dispositivo.

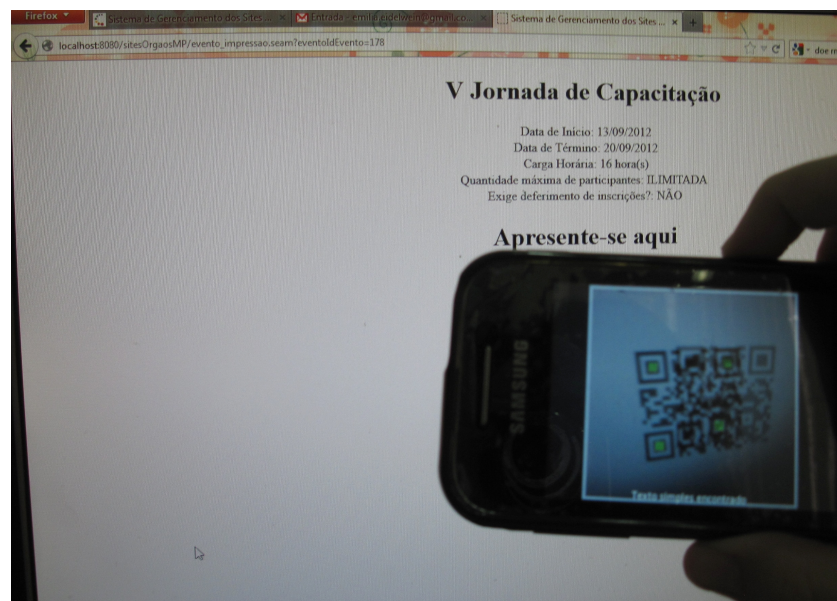


Figura 8: Teste de Leitura de QR Code em dispositivo

6.6. Interface gráfica do sistema

6.6.1. Exemplo de XML gerado no servidor

Como foi escolhido o XML como formato de metadados para comunicação de dados, o *webservice* retorna a entidade encapsulada no formato XML. Exemplificando, se fizermos uma requisição HTTP para a seguinte URL definida pelo serviço:

http://<servidor>:8080/eventos/seam/resource/rest/eventos/evento/178

; teremos a seguinte resposta em XML como na Listagem 3.

```
<eventoWrapper>
  <dataEventoFim>20/09/2012 12:00</dataEventoFim>
  <dataEventoInicio>13/09/2012 12:00</dataEventoInicio>
  <fimInscricoes>20/09/2012 12:00</fimInscricoes>
  <id>178</id>
  <inicioInscricoes>01/09/2012 12:00</inicioInscricoes>
  <nome>V Jornada de Capacitação</nome>
  <cargaHoraria>16</cargaHoraria>
  <exigeDeferimento>false</exigeDeferimento>
  <externo>false</externo>
</eventoWrapper>
```

Listagem 3: Exemplo de XML gerado.

A representação em XML acima mostra os detalhes de um evento de identificação igual a 178, sendo **eventoWrapper** a *tag* que define a entidade principal que foi encapsulada.

6.6.2. QR Code

Para cada evento, no momento de seu acontecimento, é necessário o *check-in* (apresentação) do usuário neste, para fins de emissão de certificado. Logo, o usuário deve realizar o *check-in* clicando no menu Apresentar-se no aplicativo e este abrirá uma tela para leitura de **QR Code**. Para isso, a entidade organizadora do evento deverá imprimir uma ou mais folhas contendo dados do evento e **QR Code** do mesmo e dispô-las na entrada do evento, para que os usuários possam fazer a apresentação no evento. Um modelo de folha é apresentado na Figura 9.

V Jornada de Capacitação

Data de Início: 13/09/2012
Data de Término: 20/09/2012
Carga Horária: 16 hora(s)
Quantidade máxima de participantes: ILIMITADA
Exige deferimento de inscrições?: NÃO

Apresente-se aqui



Figura 9 Folha de *check-in* com QR Code.

6.6.3. Telas do Cliente

Como pré-requisito do sistema é estar conectado à Internet, logo que se inicia o aplicativo é verificado se existe uma conexão com internet. Caso não haja conexão, uma tela de bloqueio é apresentada, impedindo o usuário de utilizar as demais funcionalidades do sistema.

A tela principal é apresentada na Figura 10 e apresenta as opções principais do sistema, como “Listar Eventos”, “Meus eventos”, “Carga Horária” e “Apresentar-se”, além de uma lista com os próximos eventos que serão promovidos na instituição.



Próximos Eventos

V Jornada de Capacitação	Aberto
De 13/09/2012 a 20/09/2012	
VI Jornada de Capacitação	Aberto
De 20/09/2012 a 30/09/2012	
Teste de Evento para Inscrição	Aberto
De 31/07/2013 a 30/08/2013	

Figura 10: Tela principal do aplicativo



Figura 11: Lista de eventos

A Figura 11 mostra a tela de listagem de todos os eventos na instituição, com opção de filtragem de eventos pelo nome através do campo e botão “Buscar”. Há duas abas auxiliares que permitem consultar somente eventos abertos e os eventos já concluídos. Um evento aberto é aquele cuja data de fim de inscrições é posterior à data

atual, i.e., o evento está aberto para inscrições. Um evento concluído não está aberto a inscrições e também já ocorreu.



Figura 12: Detalhes de evento

Ao clicar em um evento, pode-se obter detalhes deste como na Figura 12 e, se o evento estiver aberto, o botão “Realizar inscrição” estará habilitado. O usuário, clicando em “Realizar inscrição”, terá seu CPF solicitado (Figura 13) e deverá inseri-lo para prosseguir com a inscrição, devendo este ser um número de CPF válido.



Figura 13: Identificação via CPF é um ponto central para o sistema

Após o sistema validar o CPF, prossegue-se para a tela de dados pessoais (Figura 14). Se o CPF já estava cadastrado no sistema, este já retorna os campos preenchidos com os dados armazenados no banco de dados, permitindo que o usuário possa modificá-los, entretanto não pode deixar nenhum campo vazio.

The image shows a web form titled "Eventos do MP-RN". The main heading is "Inscrição - Informações Corporativas". Below this, there are several input fields: "Instituição:" with the value "Ministério Público do Rio Grande do Norte" (highlighted by an orange border), "Setor:" with the value "dti", "Cargo:" with the value "ANALISTA DO MPE", and "Área de Atuação:" with the value "DESENVOLVIMENTO DE SISTEMAS". Below these fields is a section titled "Instituição Pública?" with two radio buttons: "Sim" (unselected) and "Não" (selected, indicated by a green dot). At the bottom of the form is a button labeled "Prosseguir".

Figura 14: Tela de inscrição em evento - Informações corporativas

O usuário clica em “Prosseguir” e o sistema traz a tela da Figura 15, que apresenta informações relativas a trabalho do usuário.

Ao clicar no menu “Meus eventos”, o usuário visualizará os eventos nos quais ele se inscreveu. Para isso deverá digitar o CPF na tela conforme a Figura 13. Após inserido o CPF e validado, o sistema mostra uma tela semelhante à Figura 16, mostrando todos os eventos nos quais o usuário se inscreveu, permitindo busca pelo nome. Também lista todos os eventos cuja inscrição está pendente (não foi deferida) e todos aqueles que foram concluídos. Os eventos que possuem o selo vermelho já possuem certificados liberados, os quais podem ser baixados pelo aplicativo via PDF e serem armazenados pelo usuário em seu aparelhos ou cartão SD (Figura 18).



Figura 16 Tela "Meus eventos"

Ao clicar no menu “Carga Horária”, o usuário pode visualizar sua carga horária certificada. Para isso, também deve inserir o número de seu CPF (Figura 13) e este deve ser validado. O sistema apresenta a carga horária total e cada evento com sua respectiva carga horária (Figura 17). O usuário também pode pesquisa sua carga horária total em um período de tempo (ex: período de um ano) através da aba “por período”.



Figura 17 Tela de consulta de carga horária



Código Verificador:51ECCA3ABB51C77A924057626825F6A3

©Diretoria de Tecnologia da

Figura 18 Exemplo de certificado.

7. Conclusão

Ao finalizar este trabalho, algumas conclusões podem ser obtidas e que poderão contribuir para o desenvolvimento tanto de *webservices* como aplicativos **Android**.

7.1.1. Limitações

A aplicação e serviço desenvolvidos neste trabalho apresentam algumas limitações que deverão ser avaliadas, abordadas e tratadas posteriormente. Primeiramente, quanto à necessidade de se estar conectado à internet através do dispositivo para acessar seus dados. Uma solução pode ser criar uma versão *offline* com persistência local e desabilitação de opções que só funcionam pela Internet, tais como “Inscrever em evento” e “Realizar Check-in”.

Uma segunda restrição foi a integração do *framework* JBoss Seam com o serviço RESTful. Esse *framework* é utilizado pelo Sistema de Gerenciamento de Eventos para facilitar a administração de componentes e serviços, facilitando operações de persistência de objetos e reuso de componentes. A princípio não foi possível integrar o JBoss Seam com RESTEasy, fazendo com que muitos trechos de código tivessem que ser reescritos. Pretende-se estudar uma maneira de fazer com que os serviços RESTEasy utilizem os componentes JBoss Seam, concentrando-se assim as regras de negócio em um local apenas e reduzindo o custo de refatoração.

Outra limitação foi na parte de testes, pois não foram realizados testes de carga no serviço nem testes de usabilidade com vários usuários para o aplicativo. Estes testes deverão ser planejados e executados pela equipe tanto para buscar falhas no serviço como melhorias na interface do aplicativo.

Além disso, busca-se a implementação de segurança através de **HTTPS**, criptografia dos dados na transmissão e utilização de *tokens* para controle de sessão.

Por último, deve-se remover restrição de tecnologia de dispositivos móveis do Android e expandir para outras tecnologias como iOS, Windows Phone, Blackberry através de

treinamentos nessas tecnologias e aquisição de licenças de *software*. Uma solução alternativa para a diversidade de tecnologias é a adaptação das páginas do sistema para acesso móvel, investindo-se em usabilidade para *web*.

7.1.2. Considerações Finais

Para o trabalho foi essencial buscar tecnologias compatíveis com as tecnologias preexistentes na instituição. Exemplo disso foi a escolha de **RETEasy** para desenvolver *webservices* **RESTful**, principalmente por ser um produto da **JBoss** (servidor de aplicação utilizado na instituição) em detrimento de **Jersey** e **CXF**, as quais apresentam problemas conhecidos e documentados de compatibilidade com o servidor de aplicação.

O próximo objetivo se constitui na correção das limitações e realização da validação e homologação do produto pelo cliente. Com o aplicativo homologado, pode-se proceder à implantação do sistema em ambiente de produção. A partir desse evento, com lançamento e treinamento de usuários, dever-se-á gerar novas demandas para aplicativos *mobile* para a instituição, buscando sempre atualizar a equipe de sistemas para o desenvolvimento de novas tecnologias assim como padrões.

Esse trabalho tem o potencial de agregar à equipe um novo conhecimento tecnológico, além de experiência. A equipe se moderniza e com isso, a instituição avança em seu plano estratégico, tornando-se mais acessível à população através de mais um meio de comunicação e prestação de serviços, os dispositivos móveis. Os membros, servidores e a população em geral podem agora ter acesso ao catálogo de qualificação da instituição através de um *smartphone*, tendo acesso prático às informações, cursos e eventos disponibilizados pela instituição.

REFERÊNCIAS BIBLIOGRÁFICAS

- [1] DINIZ, Vagner. **Why mobile government? Case studies in Brazil**. Disponível em: <<http://lists.w3.org/Archives/Public/public-mw4d/2008Oct/att-0026/PDF-Presentation-M-GovBrazil.pdf>> Acesso em: 23/08/2012
- [2] GHYASI, A. Farshid. **m-Government: Cases of Developing Countries**. Disponível em: <http://www.mgovernment.org/resurces/mgovlab_afgik.pdf> Acesso em: 23/08/2012
- [3] DOS SANTOS, Mauro J. **Evolution of m-Gov in Brazil**. Disponível em: <http://www.mgov.cn/lab/Archives/EuromGov2005/PDF/37_R367JM.pdf> Acesso em: 23/08/2012
- [4] LECHETA, Ricardo R. Google Android – Aprenda a criar aplicações para dispositivos móveis com o Android SDK. Brasil, Novatec, 2010.
- [5] BLANC, LEANDRO. Participação de dispositivos móveis na internet brasileira cresceu 300% em um ano. Disponível em: <<http://i.ndigo.com.br/2012/08/participacao-de-dispositivos-moveis-na-internet-brasileira-cresceu-300-em-um-ano/>> Acesso em: 01/09/2012
- [6] FEOLA, MAURO. **Mercado de smartphones: estatísticas colocam Android à frente do iOs**. Disponível em: <<http://www.androidpit.com.br/Mercado-de-smartphones-estatisticas-colocam-Android-a-frente-do-iOS>> Acesso em: 01/09/2012
- [7] mobiThinking. **Global mobile statistics 2012**. Disponível em: <<http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats/a#smartphoneos>> Acesso em: 01/09/2012
- [8] **JBoss Application Server Documentation**. Disponível em: <<http://www.jboss.org/jbossas/docs/5-x.html>> Acesso em: 28/08/2012
- [9] COSTELLO, Roger L. **Building Web Services the REST Way**. Disponível em: <<http://www.xfront.com/REST-Web-Services.html>> Acesso em: 03/09/2012
- [10] FIELDING, Roy T. **Architectural styles and the design of network-based software architectures**. Disponível: <<http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>> Acesso em: 03/09/2012
- [11] **RETEasy – JBoss Community**. Disponível em: <<https://www.jboss.org/reteasy/>> Acesso em: 03/09/2012
- [12] **HttpClient Overview**. Disponível em: <<http://hc.apache.org/httpcomponents-client-ga/index.html>> Acesso em: 04/09/2012

[13] **Google Code: ZXing**. Disponível em: <<http://code.google.com/p/zxing/>> Acesso em: 04/09/2012

[14] **QR Code Features**. Disponível em: <<http://www.qrcode.com/en/qrcode.html>> Acesso em: 04/09/2012

[15] ORTOLANI, Luiz F. B. **M-government: cases do Paraná e potencial uso no setor público brasileiro**. Disponível em: <<http://www.batebyte.pr.gov.br/modules/conteudo/conteudo.php?conteudo=1611>> Acesso em: 23/08/2012

[16] iPNews. **Governo aumentará investimentos em dispositivos móveis**. Disponível em: <<http://www.ipnews.com.br/telefonaiip/index.php/rede/categorias-de-rede/wireless/24949-governo-aumentara-investimentos-em-aplicativos-moveis.html>> Acesso em: 03/09/2012

[17] AMORIM, Paula K. D. F.; **M-Government: complemento do governo eletrônico**. Disponível em: <<http://www.intercom.org.br/papers/regionais/norte2012/resumos/R29-0279-1.pdf>> Acesso em: 23/08/2012

[18] **Android Developers: App Components**. Disponível em: <<http://developer.android.com/guide/>> Acesso em: 02/09/2012