UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

FERRAMENTA ANDROID BASEADA EM REALIDADE AUMENTADA E SERVIÇOS BASEADOS EM LOCALIZAÇÃO USANDO NOTIFICAÇÕES

RONALDO RAMPELOTTI

BLUMENAU
2011

2011/1-32

RONALDO RAMPELOTTI

FERRAMENTA ANDROID BASEADA EM REALIDADE AUMENTADA E SERVIÇOS BASEADOS EM LOCALIZAÇÃO USANDO NOTIFICAÇÕES

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Prof. Dalton Solano dos Reis, M. Sc. - Orientador

BLUMENAU
2011

2011/1-32

FERRAMENTA ANDROID BASEADA EM REALIDADE AUMENTADA E SERVIÇOS BASEADOS EM LOCALIZAÇÃO USANDO NOTIFICAÇÕES

Por

RONALDO RAMPELOTTI

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Dalton Solano dos Reis, M. Sc. – Orientador, FURB
Membro:	Prof. Paulo César Rodacki Gomes, Dr. – FURB
Membro:	Prof. Aurélio Faustino Hoppe, M. Sc. – FURB

Blumenau, 28 de junho de 2011

Dedico este trabalho a todas as pessoas que de forma direta ou indiretamente me ajudaram na realização.

AGRADECIMENTOS

A Deus, pelo caminho ensinado.

À minha família, por toda força me dada.

A minha noiva Vanessa de Mello pela ajuda, cobrança e confiança depositada em mim.

Ao meu orientador, Dalton Solano dos Reis, por ter me dado a honra de sua orientação e acreditado em meu potencial.

Aos meus amigos Fábio Roberto Pereira, Gabriela Vasselai e Evandro Pomatti, por todo apoio.

Antes de dar comida ao mendigo, dá-lhe uma vara e ensina-lhe a pescar.

Provérbio Chinês

RESUMO

Este trabalho apresenta uma aplicação com serviços baseados em localização utilizando o conceito de *push* para dispositivos móveis da plataforma Android. A aplicação tem como cenário o ambiente universitário da FURB, disponibilizando uma ferramenta para auxiliar o cotidiano do aluno. A ferramenta torna possível verificar quais disciplinas o aluno está cursando, membros da disciplina, eventos da FURB, através dos serviços disponibilizados via REST. Também é possível identificar onde estão localizados os membros da disciplina através de um mapa. Quando um evento da FURB é adicionado no servidor, o dispositivo do aluno receberá uma notificação através do *push*. Quando o aluno passa por um perímetro próximo a um evento da FURB ele também recebe um *push*. Tais funcionalidades foram obtidas através do conceito de serviço baseado em localização. Por fim a aplicação é apresentada rodando em um dispositivo real.

Palavras-chave: Android. Mobilidade. LBS. Push.

ABSTRACT

This paper presents an application with location based services using the concept of push to mobile devices of Android platform. The application is set in the university environment of FURB, providing a tool to help the student daily. The tool makes possible to check which subjects the student is enrolled, members of the discipline, FURB events, through the REST services. It is also possible to identify where the members of the discipline are located through a map. When an event is added in the FURB's server, the student's device will receive a notification through push. When the student goes through a perimeter around the event, he also receives a push. These functionalities where obtained using the concept of location based services. Finally the application is shown running on a real device.

Key-words: Android. Mobility. LBS. Push.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

RA - Realidade Aumentada

SDK - *Software Developer Kit*

REST - *REpresentational State Transfer*

HTTP - *Hyper Text Transfer Protocol*

LBS - *Location Based Services*

YAML - *Yet Another Markup Language*

JSON - *JavaScript Object Notation*

SOAP - *Simple Object Access Protocol*

C2DM - *Cloud to Device Message*

GPS – *Global Posistioning System*

PDA - *Personal Digital Assistants*

MIT - *Massachusets Institute of Technology*

PUC - Pontifícia Universidade Católica

SOA - *Service Oriented Architecture*

J2EE – JAVA 2 *Plataform Enterprise Edition*

XML - *eXtensible Markup Language*

UML - *Unified Modeling Language*

MER - Modelo de Entidade e Relacionamento

JDBC - Java *Data Base Connectivity*

ADT - Android *Development Tools*

mHZ - *Mega Hertz*

GB - *Giga Bytes*

WCF - *Windows Communication Framework*

SUMÁRIO

1 INTRODUÇÃO

Enquanto o computador pessoal em grande parte derivou de uma “idéia” de máquina de escrever e calculadora, o celular veio da fusão do bloco de notas de bolso e do telefone individual, ou seja, algo pessoal e que nasceu em casa (FAZION, 2011).

A popularidade que os dispositivos móveis conquistaram tornou-se algo que não pode ser tratado com irrelevância. Atualmente são ferramentas de auxílio na execução de trabalhos e facilitadores de processos no dia-a-dia.

Juntamente com a mobilidade de processos empregada nos dispositivos móveis, observa-se que a necessidade de criar e aprimorar facilitadores do cotidiano são cada vez mais alvejados. Todo esforço empregado para a criação de facilitadores, resultam em mais rapidez na execução de tarefas.

Com base na constante evolução dos dispositivos móveis e colaborando para a criação de um facilitador de processos, este trabalho teve por objetivo disponibilizar uma ferramenta de uso universitário para apoiar alunos em suas atividades acadêmicas, utilizando Realidade Aumentada (RA) para localizar eventos acadêmicos, obtendo assim maior precisão na informação, pois visualizará o local como informação virtual associada à imagem real. A ferramenta ainda conta com o uso de *Location Based Services* (LBS) para alertar o usuário da proximidade de um evento. Ainda é possível visualizar onde se encontram os membros das disciplinas em comum através de um mapa, bem como comunicar-se através de e-mail.

A ferramenta construída conta também com um site para administração de conteúdo, permitindo aos professores postarem eventos, notas e freqüência de membros de uma disciplina.

Também foi empregado o uso de notificações *push*, onde o aluno receberá alerta sobre postagens efetuadas pelos professores.

Por fim, a ferramenta foi desenvolvida com a plataforma de desenvolvimento Android *Software Developer Kit* (SDK), utilizando a arquitetura *REpresentational State Transfer* (REST) como camada de serviços para troca de informação entre o cliente e o servidor.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho foi disponibilizar uma ferramenta para plataforma Android, fazendo uso de RA e serviços baseados em localização usando notificações.

Os objetivos específicos do trabalho são:

a) disponibilizar as funcionalidades do servidor para gerência de notificações do dispositivo;
b) definir uma arquitetura de comunicação entre o dispositivo e o servidor;
c) utilizar realidade aumentada para localizar pontos de interesse;
d) utilizar como cenário de aplicação a Universidade Regional de Blumenau;
e) utilizar serviços baseados em localização para interagir com membros de disciplina, cursos, seminários e palestras.

1.2 ESTRUTURA DO TRABALHO

A estrutura deste trabalho está apresentada em quatro capítulos, onde o segundo capítulo contém a fundamentação teórica necessária para o entendimento deste trabalho.

O terceiro capítulo apresenta como foi desenvolvida a ferramenta para as plataformas Android e *web*, os casos de uso da aplicação, diagramas de classe e a especificação que define a ferramenta. No terceiro capítulo são apresentadas também, as partes principais da implementação juntamente com resultados e discussões que aconteceram durante o desenvolvimento do projeto.

Por fim, o quarto capítulo refere-se às conclusões do presente trabalho e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Nas seções a seguir são detalhadas as razões e vantagens para a utilização da arquitetura REST para comunicação entre dispositivos e um servidor através de requisições do tipo *web service* e utilizando de notificações. Em seguida são apresentadas as definições de realidade aumentada e serviços baseados em localização. Por fim, são expostas algumas ferramentas com funcionalidades similares ao da ferramenta construída.

2.1 REST

Conforme Nunes e David (2005, p. 208), REST é um termo definido por Roy Fielding para descrever um estilo arquitetural de sistemas de informação distribuídos. A primeira edição do REST, originalmente designada por "HTTP *Object Model*", foi desenvolvida entre 1994 e 1995, como um meio para transmitir os conceitos que se encontram na base da web durante o desenvolvimento do protocolo *Hyper Text Transfer Protocol* (HTTP). REST representa um modelo de como a web deve funcionar. A existência deste modelo permite identificar áreas onde os protocolos existentes falham, comparar soluções alternativas e assegurar que novas extensões não violem os princípios nucleares que contribuíram para o sucesso da web.

Segundo Nunes e David (2005, p. 208), o modelo REST utiliza um conjunto de interfaces genéricas para promover interações sem estado (*stateless*) através da transferência de representações de recursos, em vez de operar diretamente sobre estes recursos. O conceito de recurso é a principal abstração deste modelo.

REST refere-se originalmente a um conjunto de princípios de uma arquitetura. Na atualidade usa-se no sentido mais amplo para descrever qualquer interface web simples que faz uso da *eXtensible Markup Language* (XML) e do *Hypertext Transfer Protocol* (HTTP); ou da *Yet Another Markup Language* (YAML) e do *JavaScript Object Notation* (JSON); ou texto puro. Sem as abstrações adicionais dos protocolos baseados em padrões de trocas de mensagem como o protocolo de serviços web *Simple Object Access Protocol* (SOAP) (WIKI, 2011).

Ou seja, conforme Nunes e David (2005, p. 209), a interação entre solicitante e

fornecedor é feita com recurso ao protocolo HTTP, utilizando um conjunto de métodos pré-definidos: `get`, `post`, `put` e `delete`.

2.2 NOTIFICAÇÕES (*PUSH*)

A maioria dos aplicativos da plataforma Android utiliza a internet para manter seus usuários atualizados. Tradicionalmente, os aplicativos utilizam *Polling* para recuperar informações periodicamente, como um cliente de e-mail que acessa o servidor em intervalos de tempo. O *Polling* é fácil de programar e funciona bem na maioria das situações. O problema do *Polling* é que ele faz acessos demasiados ao servidor, buscando informações que normalmente não estão disponíveis ainda. O *Polling* é especialmente problemático em dispositivos móveis, pois faz acessos desnecessários à internet e consome a vida útil da bateria (EMIDIO, 2010).

Em muitas situações, para as aplicações móveis que precisam reagir a alguns dados de um servidor remoto é mais eficiente os dados serem enviados para o celular, ao invés do celular ficar várias vezes requisitando ao servidor. É melhor para a aplicação móvel simplesmente aguardar o servidor informar quando ocorre uma mudança de dados (LANE, 2009). A Apple popularizou esta abordagem com o iPhone *Push Notification Service* e da mesma forma a BlackBerry fez isso com o cliente de *e-mail* disponível na plataforma. Esta última possui um SDK que permite notificações via *push* para outras aplicações.

Na Figura 1 é representado o funcionamento do *push* na plataforma BlackBerry.

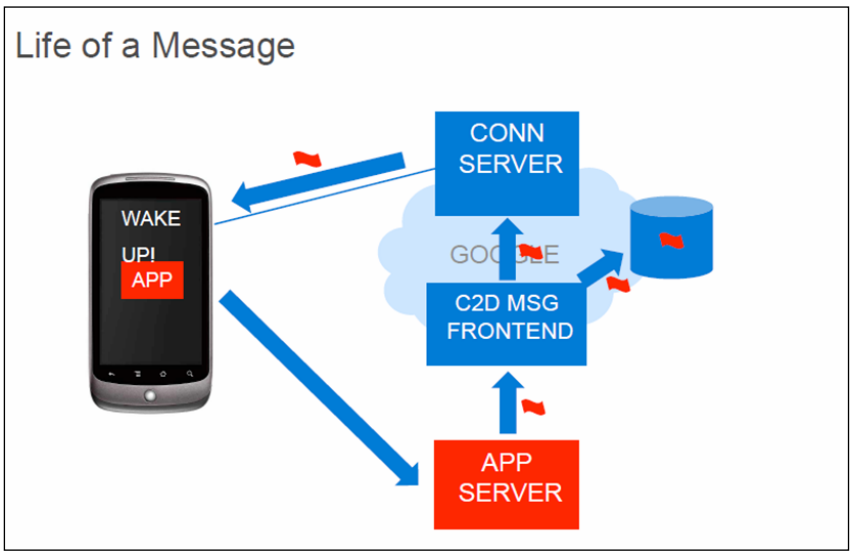
Fonte: BlackBerry (2011).

Figura 1 – Fluxo do push na plataforma BlackBerry

Na versão 2.2 do Android a Google disponibilizou o *Cloud to Device Message* (C2DM), que é um serviço que ajuda a enviar os dados dos servidores às aplicações em dispositivos Android. O serviço fornece um mecanismo simples e leve onde os servidores notificam as aplicações móveis para manter contato com o servidor, a fim de atualizar a aplicação ou dados do usuário. O serviço C2DM controla todos os aspectos de enfileiramento

e entrega de mensagens para o aplicativo em execução no dispositivo de destino (CODE, 2011).

Na Figura 2 é representado o funcionamento do *push* C2DM do Android. A aplicação solicita ao servidor Google uma identificação para troca de mensagens de notificações. Esta notificação é a chave para o envio de mensagens do servidor para a aplicação no dispositivo, após o recebimento do dispositivo da identificação, é enviada para o servidor essa identificação. Quando o servidor necessita enviar uma notificação para o dispositivo, ele envia a mensagem para o servidor Google com a identificação do dispositivo que envia a mensagem para o dispositivo.



Fonte: Androidmeup (2010).

Figura 2 – Fluxo do push na plataforma Android

2.3 REALIDADE AUMENTADA

Durante muitas décadas a sofisticação das interfaces gráficas computacionais fez com que as pessoas tivessem que se ajustar às máquinas, criando a necessidade de treinamento, uma vez que o conhecimento do mundo real já não era suficiente. Com a evolução das tecnologias de hardware e software foi possível fazer com que as máquinas se ajustassem às pessoas, possibilitando aos usuários acessarem aplicações como se estivessem atuando no mundo real, falando, apertando ou fazendo gestos (VASSELAI, 2010, p. 16).

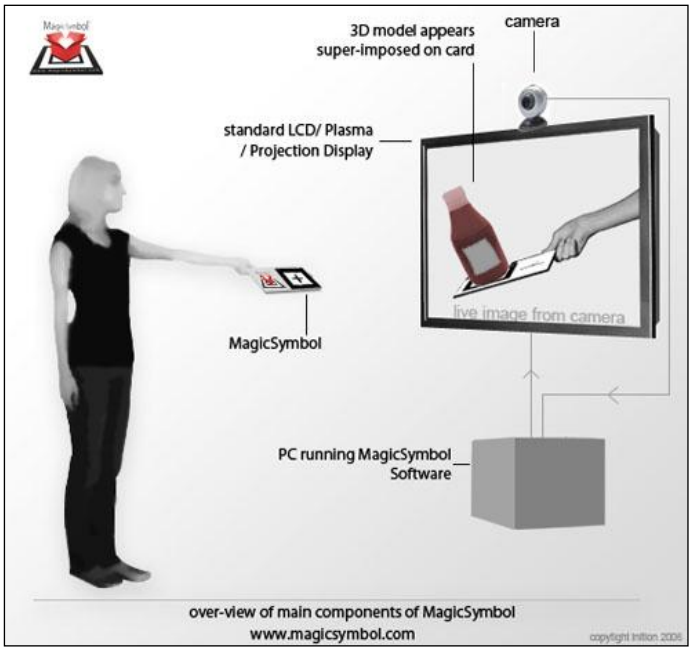
Atualmente, a maior parte das pesquisas em RA está ligada ao uso de vídeos transmitidos ao vivo, que são digitalmente processados e ampliados pela adição de gráficos criados pelo computador. Pesquisas avançadas incluem uso de rastreamento de dados em

movimento, reconhecimento de marcadores confiáveis utilizando mecanismos de visão e a construção de ambientes controlados, contendo qualquer número de sensores e atuadores (AUTOMAILS, 2010).

Conforme Azuma (1997, p. 357), RA consiste em suplementar o mundo real com objetos virtuais sobrepondo ou combinando os objetos reais, desta forma permitindo uma interação homem-máquina mais fácil e natural. Considera ainda que um sistema de RA deve possuir três características: combinar objetos reais e objetos virtuais em um ambiente real, executar interativamente em tempo real e alinhar objetos reais e virtuais entre si.

Ainda conforme Azuma (1997, p. 357), a RA amplia a percepção e interação do usuário com o mundo real. Os objetos virtuais exibem informações que o usuário não pode detectar diretamente com seus próprios sentidos. As informações transmitidas pelo objeto virtual ajudam o usuário a executar tarefas no mundo real. RA é um exemplo concreto do que Fred Brooks chama de Inteligência de Amplificação: utilizando o computador como meio para executar uma tarefa mais fácil.

Na Figura 3 é mostrada a visão com Realidade Aumentada.



Fonte: Nerddanet (2010).

Figura 3 – Visão com realidade aumentada

A figura demonstra o funcionamento de uma aplicação que utiliza realidade aumentada. Em frente ao display onde é exibida a imagem captada pela câmera que está acima da tela, o usuário segurando uma placa visualiza um objeto sendo projetado a cima da placa. Isso acontece pelo processamento realizado da imagem captada por uma aplicação que está rodando recebendo como entrada a imagem da câmera e distribuindo a imagem da

câmera somada a imagem tratada da placa.

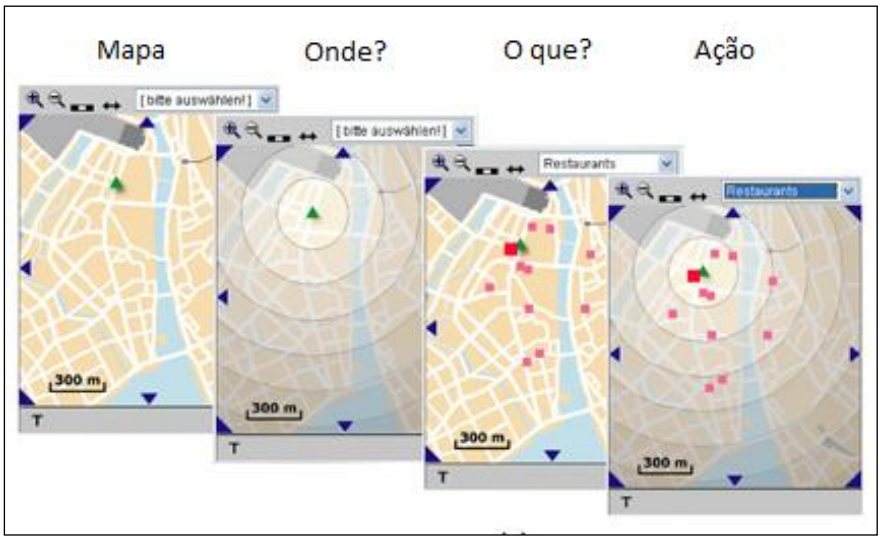
2.4 SERVIÇOS BASEADOS EM LOCALIZAÇÃO

Segundo Steiniger, Neun e Edwards (2011), os dispositivos móveis e a internet têm revolucionado a comunicação e com ela a vida das pessoas. Um número crescente de telefones celulares e *Personal Digital Assistants* (PDA) permitem as pessoas acessarem internet onde quer que estejam. A partir da internet é possível obter informações sobre eventos (cinema, shows, festas) e sobre os locais de uma cidade (restaurantes, museus, hospitais), por exemplo.

O surgimento de tecnologias capazes de dar a exata localização de um dispositivo móvel - celulares - criou possibilidades para que novos serviços surgissem também nesta área e, assim, nasceram os LBS ou, em português, serviços baseados em localização. Todos os serviços que usam o georeferenciamento podem ser incluídos entre os LBS: monitoramento de caminhões, navios, aviões, automóveis; rastreamento de veículos e pessoas (CARBONARO, 2009).

Os serviços baseados em localização respondem a três perguntas: Onde? O que há em volta? Como chegar lá? Eles determinam a localização do usuário, em seguida usam o local e outras informações para fornecer aplicações e serviços personalizados. Como exemplo, um serviço de emergência sem fio 911 que determina a localização do chamador automaticamente. Outros serviços podem combinar o local onde o dispositivo se encontra com informações sobre alojamento e entretenimento (MAHMOUD, 2004).

Na Figura 4 é mostrado o uso do conceito de LBS.



Fonte: Steineger, Neun e Edwards (2011).

Figura 4 – Usando LBS

Primeiro deve ser localizado o usuário, verificando onde ele está, depois é verificado o que tem por perto que seria de interesse do usuário e com essas informações é que é tomada alguma ação.

2.5 TRABALHOS CORRELATOS

Algumas ferramentas disponíveis no mercado possuem funcionalidades similares ao trabalho desenvolvido. Entre elas, citam-se iStanford (ITUNES, 2010a), MIT Mobile (ITUNES, 2010b) , Harvard Mobile (ITUNES, 2010c) e PUCRS Mobi (BAGUETE, 2011).

2.5.1 iStanford

Segundo iTunes (2010a), esta ferramenta foi criada pela universidade de Stanford para auxíliar os alunos. Com o iStanford o usuário pode usar os seguintes itens:

a) mapas: busca de locais no campus, consulta de rotas de ônibus e auto-localização;
b) lista: pesquisa por estudantes, professores e funcionários no diretório de Stanford, toque para chamadas ou *e-mail*, adicionar como contatos no celular;
c) cursos: busca de classes, descobrir onde e quando é oferecido, toque de chamada

ou *e-mail* do professor, inscrição em cursos, ver os cursos e graus;

d) atletismo: notícias, resultados e os horários de cada esporte do colégio Stanford;

e) eventos: informações sobre eventos que acontecem no campus;

f) notícias: fontes de notícia de Stanford;

g) biblioteca: pesquisa o conteúdo das bibliotecas de Stanford.

A Figura 5 apresenta a ferramenta.

Fonte: iTunes (2010a).

Figura 5 – Ferramenta iStanford

2.5.2 MIT Mobile

A aplicação do *Massachusets Institute of Technology* (MIT), segundo (ITUNES, 2010b) oferece entre suas funcionalidades:

a) notícias sobre pesquisa e inovação: trazendo as informações sobre as pesquisas do campus;

b) monitoramento de transporte em tempo real e alerta: trazendo ao usuário informação das principais linhas de onibus que dão acesso ao campus, bem como alertar o usuário quando configurado pelo usuário, alerta de horário da linha;

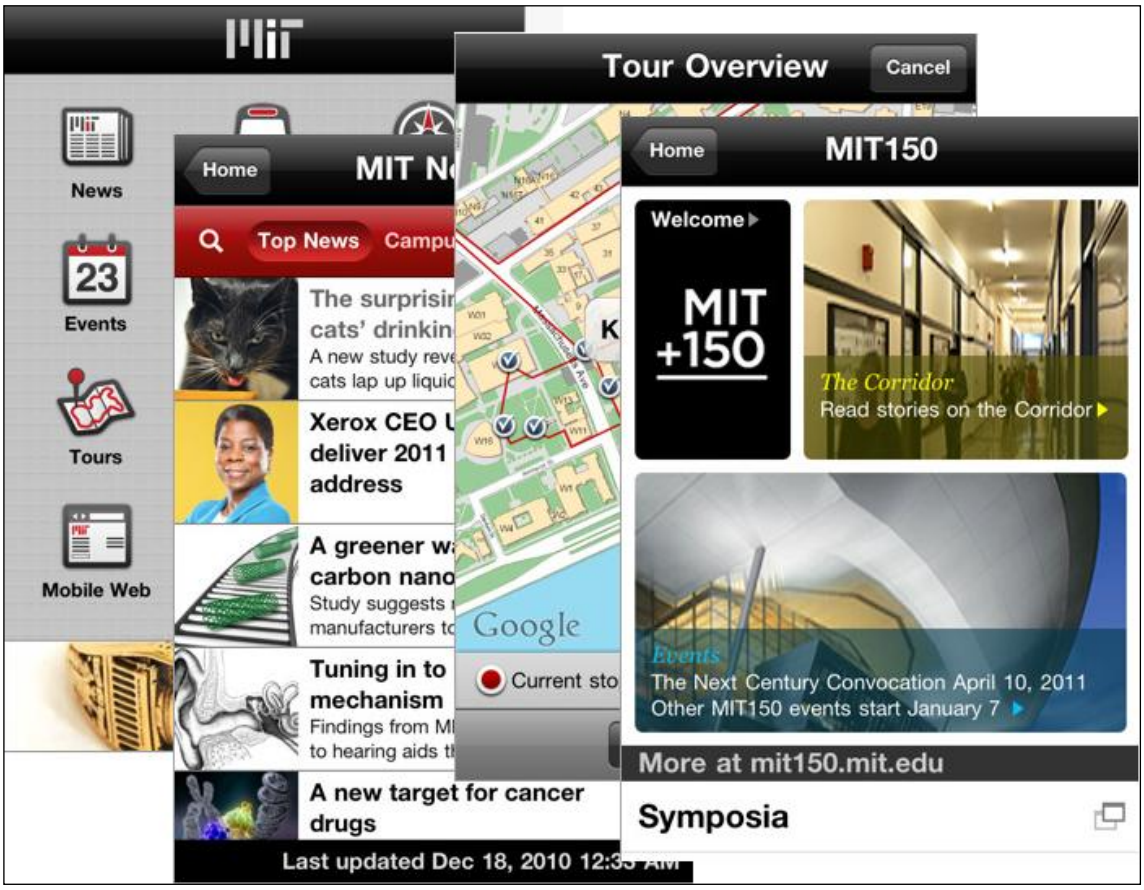
c) mapa interativo do campus: mapa com opção de tourpelo campus;

d) informações de classe e anúncios: sistema de anúncios dos alunos para venda de

bens;

e) pesquisa de pessoas: traz contatos de pessoas que trabalham no campus.

A Figura 6 mostra a ferramenta MIT Mobile.

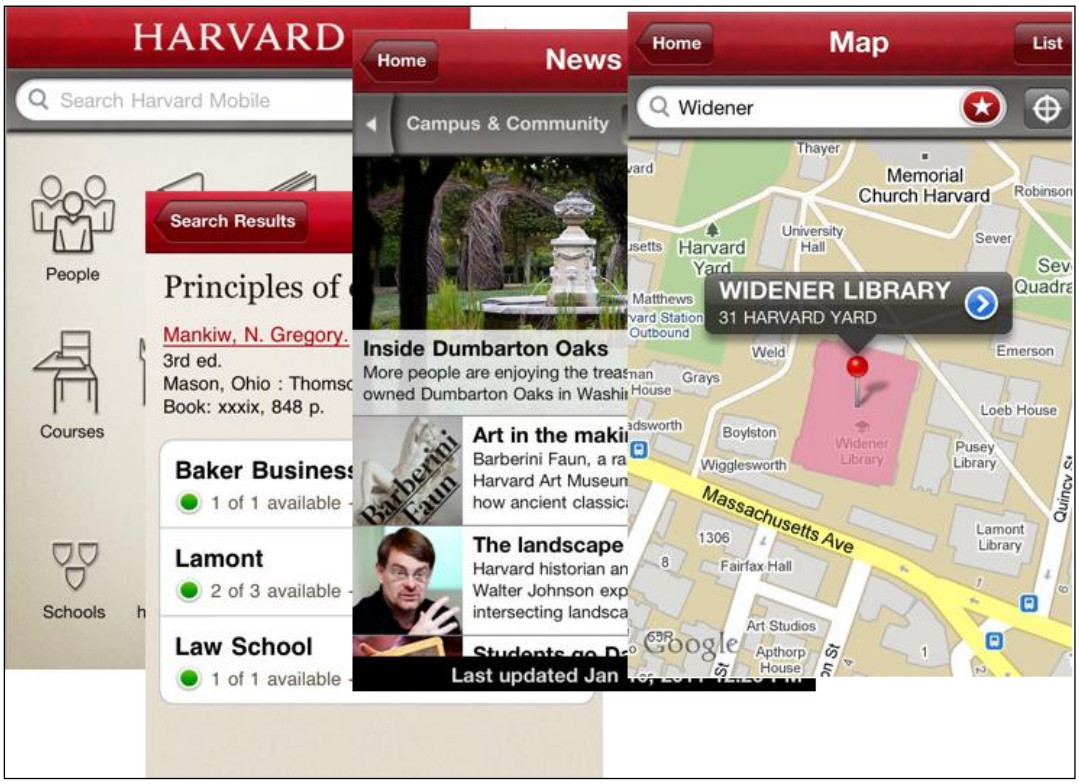


Fonte: iTunes (2010b).

Figura 6 – Ferramenta MIT Mobile

2.5.3 Harvard Mobile

Segundo iTunes (2010c), a aplicação Harward Mobile oferece notícias, transporte, mapa do campus, informações de aulas e pesquisa de alunos. A Figura 7 mostra a ferramenta Harward Mobile.



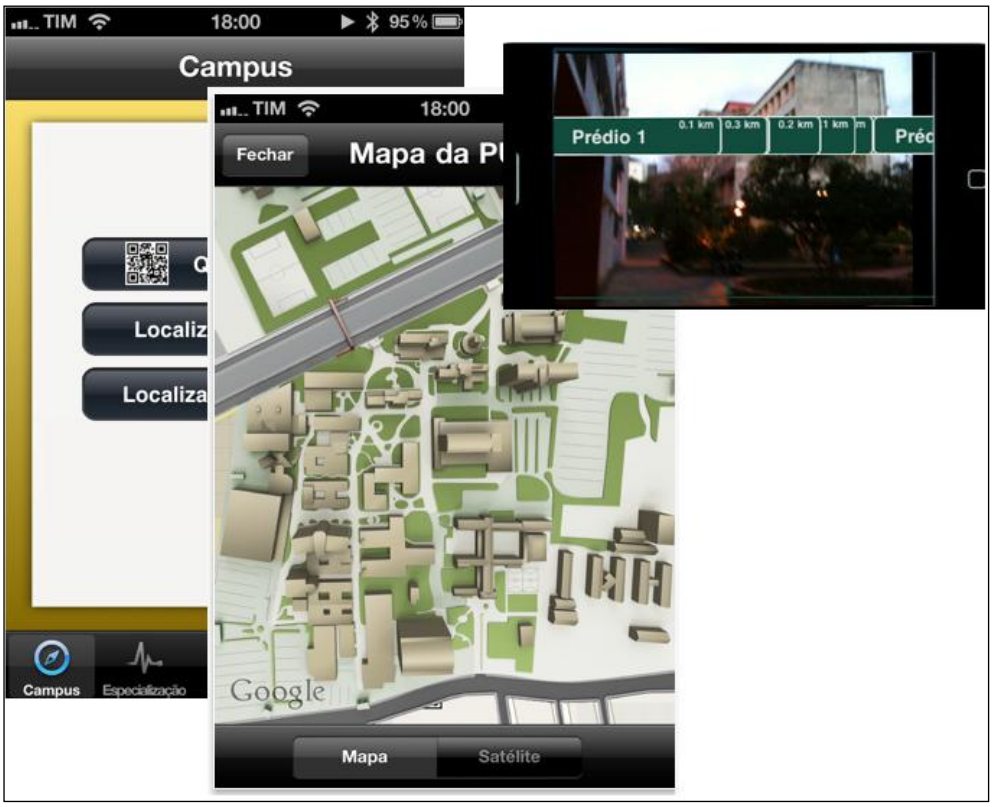
Fonte: iTunes (2010c).

Figura 7 – Ferramenta Harvard Mobile

2.5.4 PUCRS Mobi

Conforme Bagueti (2011), a ferramenta desenvolvida pela Pontifícia Universidade Catolica (PUC) do Rio Grande do Sul, teve como objetivo facilitar as atividades dentro da universidade, por meio de transmissão de informações.

Entre as principais funcionalidades a ferramenta conta com: captura de imagens com QR Codes, para utilizar como localização através de realidade aumentada ou de mapa, informações acadêmicas e serviços da universidade bem como vídeos do programa da universidade intitulado “Diário do campus”. A Figura 8 mostra a ferramenta PUCRS Mobi.



Fonte: iTunes (2011d).

Figura 8 – Ferramenta PUCRS Mobi

3 DESENVOLVIMENTO DA FERRAMENTA

Este capítulo demonstra o desenvolvimento da ferramenta construída. São apresentados os principais requisitos, especificação e implementação da ferramenta.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Para alcançar o objetivo da utilização de serviços baseados em localização e o uso de notificações a ferramenta foi construída a partir dos seguintes requisitos funcionais para o dispositivo:

a) permitir ao usuário localizar disciplinas (RF01);
b) permitir ao usuário localizar membros das disciplinas (RF02);
c) permitir ao usuário localizar algum evento (RF03);
d) permitir ao usuário localizar ponto de interesse utilizando RA (RF04);
e) permitir ao usuário se inscrever em eventos (RF05);
f) permitir comunicação entre membros da disciplina via *e-mail* (RF06);
g) permitir ao usuário consultar freqüências (RF07).

A ferramenta proposta atende os seguintes requisitos funcionais para o servidor:

a) permitir cadastro de eventos através de um *site* por um professor (RF08);
b) permitir ao professor postar materiais (RF09);
c) permitir ao professor postar notas (RF10);
d) permitir notificações para o dispositivo sobre alertas de curso, seminários e postagem de materiais (RF11);
e) permitir ao professor postar freqüências (RF12).

A ferramenta atendeu os seguintes requisitos não funcionais:

a) o sistema deve ser desenvolvido na linguagem Java (RNF01);
b) utilizar banco de dados MySQL 4.1 ou posterior (RNF02);
c) o sistema deve ser compatível com o sistema operacional Android 2.2 ou posterior (RNF03).

3.2 ESPECIFICAÇÃO

A especificação foi realizada a partir da ferramenta Enterprise Architect, utilizando a linguagem de modelagem *Unified Modeling Language* (UML). Para a modelagem foram utilizados os diagramas de casos de uso, atividade e entidade-relacionamento.

3.2.1 Diagramas de caso de uso

Os diagramas de caso de uso estão divididos em: dispositivo, servidor S*ervice Oriented Architecture* (SOA) e site. A especificação deste trabalho utilizou diagramas da UML e modelo de entidade e relacionamento (MER). Os casos de uso são apresentados através de um diagrama de casos de uso, seguidos da descrição do cenário de cada caso de uso. As classes do aplicativo cliente, site e servidor SOA são apresentadas através de diagramas de classes. Complementando a especificação, as entidades do banco de dados do servidor são demonstradas através de diagramas MER.

3.2.2 Casos de uso

A partir dos requisitos levantados para o Furbin, originou-se 33 casos de uso, divididos em quatro módulos que representam partes da ferramenta, contemplando uma arquitetura orientada a serviços.

Os casos de uso desenvolvidos são desempenhados por quatro atores. O ator `Usuário Dispositivo` representa o utilizador do sistema, interagindo através dos casos de uso que apresentam interfaces gráficas. O ator `Dispositivo` representa o aplicativo cliente, que interage através de serviços de consulta ou inserção. O ator `Professor` representa o utilizador do sistema *web*, interagindo através dos casos de uso que representam a parte *web*. Por fim o ator `Site` é a representação do aplicativo *web* que interage através de serviços de consulta e inserção.

Os diagramas de caso de uso foram desenvolvidos observando os padrões da UML. Cada caso de uso foi detalhado em cenários e vinculado a pelo menos um RF. Essa vinculação

objetiva facilitar a identificação do propósito do caso de uso e justificar sua existência. A Figura 9 contém o diagrama de casos de uso do dispositivo.

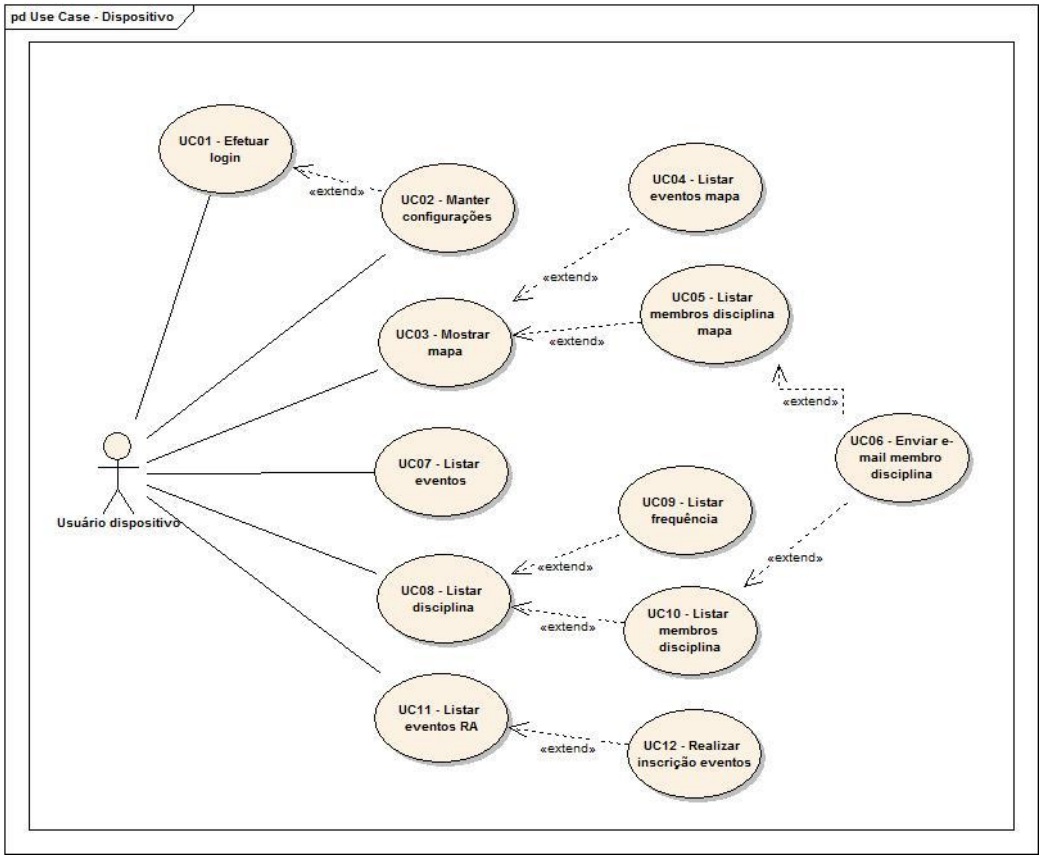


Figura 9 - Diagrama de casos de uso do dispositivo

A Figura 10 contém o diagrama de casos de uso de comunicação entre dispositivo e servidor.

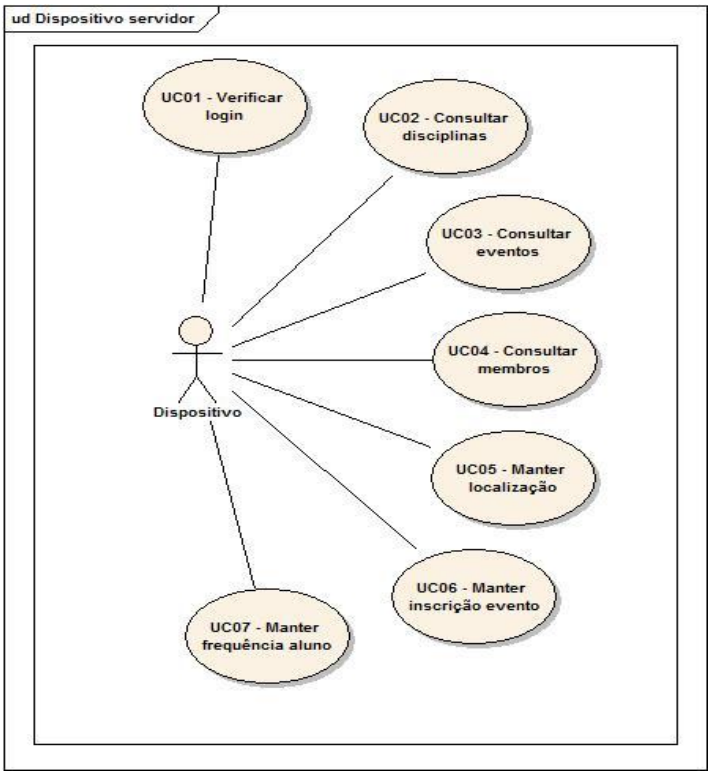


Figura 10 – Diagrama de casos de uso de comunicação entre dispositivo e servidor

A Figura 11 contém o diagrama de casos de uso de comunicação entre o *site* e o servidor.

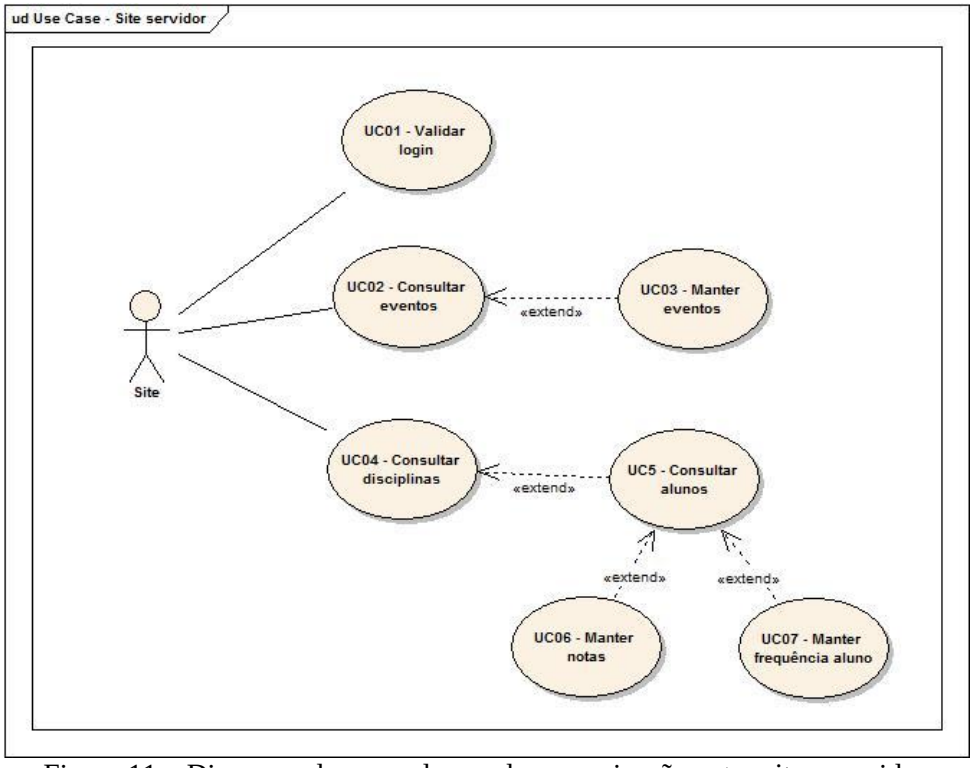


Figura 11 – Diagrama de casos de uso de comunicação entre *site* e servidor

A Figura 12 contém o diagrama de casos de uso do *site*.

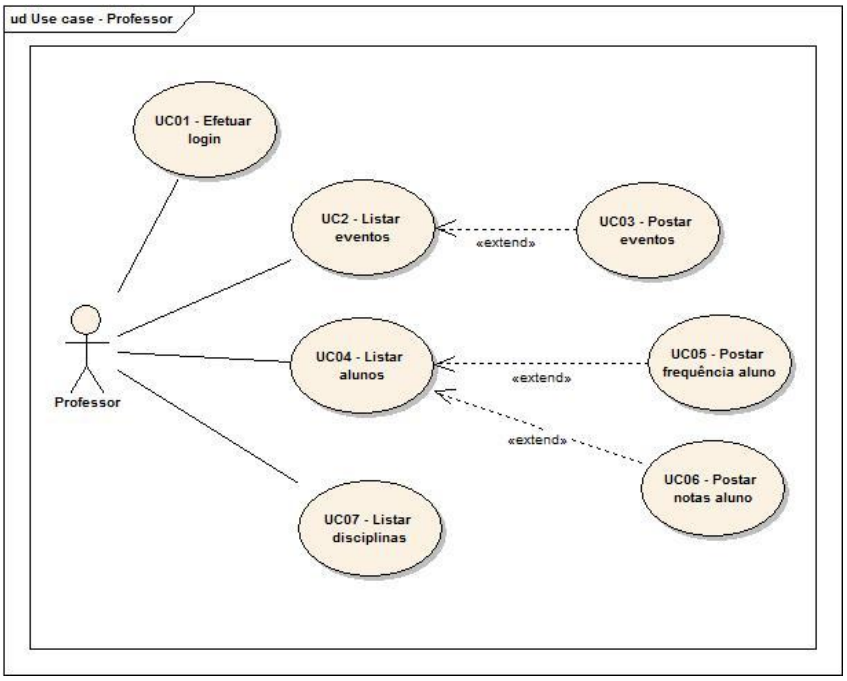


Figura 12 – Diagrama de casos de uso do *site*

3.2.2.1 Casos de uso do dispositivo

A seguir serão detalhados os principais casos de uso que correspondem ao dispositivo.

3.2.2.1.1 Efetuar login

Caso de uso obrigatório no dispositivo, pois para utilização das funcionalidades faz-se necessário estar autenticado no sistema. Esse caso de uso apresenta uma janela com campos de entrada para usuário e senha. Mais detalhes estão descritos no Quadro 1.

UC01 – Efetuar login	
Descrição	Permite ao usuário o acesso a aplicação, validando o usuário e a senha.
Requisito atendido	RF01
Pré-condições	Nenhuma
Cenário principal	1. Disponibilizar campos para entrada de valores: usuário e senha, disponibilizar um botão para autenticar; 2. Usuário informa usuário e senha; 3. Ao pressionar o botão `Logar` o sistema deve validar o usuário e a senha informados.
Fluxo alternativo 01	No passo 3 se a validação obtiver sucesso, o usuário terá acesso a tela do menu principal da ferramenta.
Exceção	No passo 3, se ocorrer um erro ao validar o usuário e a senha, será mostrada uma mensagem de erro de autenticação na tela e a aplicação permanece nesta tela.
Pós-condições	Acesso a tela de menu principal da ferramenta.

Quadro 1 - Caso de uso do dispositivo UC01

3.2.2.1.2 Lista Eventos Mapa

Neste caso de uso o usuário pode visualizar em um mapa quais eventos estão cadastrados pelas proximidades. Mais detalhes estão descritos no Quadro 2.

UC04 – Listar Eventos Mapa	
Descrição	Permite ao usuário visualizar através de um mapa regional os eventos disponíveis.
Requisito atendido	RF03
Pré-condições	Usuário deve estar autenticado no sistema e no menu principal da aplicação ter pressionado o botão `Mapa`.
Cenário principal	1. Sistema busca os eventos disponíveis no servidor; 2. Sistema faz a plotagem dos eventos no mapa; 3. Usuário visualiza no mapa os eventos disponíveis.
Fluxo alternativo 01	No passo 1 se não houverem eventos, passa para o passo 3.
Exceção	No passo 1 se ocorrer um erro de comunicação com o servidor, o sistema exibe uma mensagem para o usuário.
Pós-condições	Mapa com eventos desenhados.

Quadro 2 - Caso de uso UC04

3.2.2.1.3 Listar Membros Mapa

Neste caso de uso o usuário, através da tela de mapa, acessa na opção do menu de contexto `Localizar membros disciplina`, uma lista de disciplinas que o usuário cursa.

Após selecionar a disciplina o usuário visualiza no mapa os membros da disciplina selecionada. Mais detalhes estão descritos no Quadro 3.

UC05 – Listar Membros Mapa	
Descrição	Permite ao usuário visualizar membros de uma disciplina no mapa.
Requisito atendido	RF02
Pré-condições	Usuário deve estar autenticado no sistema e no menu principal da aplicação ter pressionado o botão `Mapa`.
Cenário principal	1. Usuário clica no botão `Menu` do dispositivo; 2. Sistema exibe o menu de contexto com a opção `Localizar membros da disciplina`; 3. Usuário clica no botão `Localizar membros da disciplina`; 4. Sistema busca as disciplinas disponíveis no servidor, exibindo uma lista com os resultados; 5. Usuário clica em uma disciplina na listagem; 6. Sistema busca membros da disciplina selecionada no servidor; 7. Sistema desenha no mapa os membros da disciplina selecionada;
Fluxo alternativo 01	No passo 4 se o servidor retornar uma lista vazia, o sistema exibe a lista sem nenhum membro.
Fluxo alternativo 02	No passo 7 somente serão desenhados os membros que tiverem o campo `ultimalocalizacao` informado.
Exceção	No passo 4 se ocorrer um erro de comunicação com o servidor, o sistema exibe uma mensagem para o usuário.
Pós-condições	Mapa com membros de disciplina desenhados.

Quadro 3 - Caso de uso do dispositivo UC05

3.2.2.1.4 Listar Disciplinas

Neste caso de uso o usuário, através do menu principal, acessa pelo botão `Disciplinas` uma tela com a lista de disciplinas que o mesmo cursa. Mais detalhes estão descritos no Quadro 4.

UC08 – Listar Disciplinas	
Descrição	Permite ao usuário visualizar as disciplinas que o mesmo se encontra matriculado.
Requisito atendido	RF01
Pré-condições	Usuário deve estar autenticado no sistema e no menu principal da aplicação ter pressionado o botão `Disciplinas`.
Cenário principal	1. Ao entrar na tela o sistema busca as disciplinas do usuário no servidor; 2. Sistema lista as disciplinas retornadas em uma lista com resultados.
Fluxo alternativo 01	No passo 1 se o servidor retornar uma lista vazia, o sistema exibe a lista sem nenhuma disciplina.
Exceção	No passo 1 se ocorrer um erro de comunicação com o servidor, o sistema exibe uma mensagem para usuário.
Pós-condições	Lista com as disciplinas do usuário.

Quadro 4 - Caso de uso do dispositivo UC08

3.2.2.1.5 Enviar E-mail Membro Disciplina

Neste caso de uso o usuário pode tanto através da tela de lista de disciplinas quanto pelo mapa, selecionar um membro da disciplina e ter a opção de enviar e-mail para o membro. Mais detalhes estão descritos no Quadro 5.

UC06 – Enviar e-mail membro disciplina	
Descrição	Permite ao usuário enviar e-mail para um membro de uma disciplina.
Requisito atendido	RF06
Pré-condições	Usuário deve estar autenticado no sistema e no menu principal da aplicação ter pressionado o botão `Disciplinas`, depois ter selecionado uma disciplina.
Cenário principal	1. Usuário seleciona um membro específico na lista de membros da disciplina; 2. Sistema exibe campos para compor e-mail com o endereço do membro da disciplina selecionada; 3. Usuário compõe um e-mail e aperta em `Enviar`; 4. Sistema envia um e-mail e volta para a lista de membros.
Fluxo alternativo 01	No passo 1 se o servidor retornar uma lista vazia, o sistema exibe a lista sem nenhum membro.
Exceção	No passo 1 se ocorrer um erro de comunicação com o servidor, sistema exibe uma mensagem para usuário.
Pós-condições	Envio de e-mail efetuado e volta para tela de seleção de membros.

Quadro 5 - Caso de uso do dispositivo UC06

3.2.2.1.6 Listar Freqüência

Neste caso de uso o usuário pode visualizar sua freqüência na disciplina selecionada. Mais detalhes estão descritos no Quadro 6.

UC09 – Listar freqüência	
Descrição	Permite ao usuário visualizar a freqüência obtida em uma disciplina.
Requisito atendido	RF07
Pré-condições	Usuário deve estar autenticado no sistema e no menu principal da aplicação ter pressionado o botão `Disciplinas`, depois ter selecionado uma disciplina.
Cenário principal	1. Ao entrar na tela sistema busca a freqüência do usuário na disciplina selecionada no servidor; 2. Sistema exibe um campo com a freqüência do usuário.
Exceção	No passo 1 se ocorrer um erro de comunicação com o servidor, o sistema exibe uma mensagem para usuário.
Pós-condições	Sistema exibe em um campo a freqüência do usuário na disciplina previamente selecionada.

Quadro 6 - Caso de uso do dispositivo UC09

3.2.2.1.7 Listar eventos RA

Neste caso de uso o usuário pode visualizar através de RA onde se localiza um evento. Mais detalhes estão descritos no Quadro 7.

UC11 – Listar eventos RA	
Descrição	Permite ao usuário visualizar através de realidade aumentada onde estão localizados os eventos.
Requisito atendido	RF04
Pré-condições	Usuário deve estar autenticado no sistema e no menu principal da aplicação ter pressionado o botão `Realidade Aumentada`.
Cenário principal	1. Ao entrar na tela o sistema busca no servidor os eventos disponíveis; 2. Com o retorno do servidor o sistema desenha na camada acima da visão da câmera os eventos; 3. Usuário visualiza os eventos.
Exceção	No passo 1 se ocorrer um erro de comunicação com o servidor o sistema exibe uma mensagem para o usuário.
Pós-condições	Usuário visualiza através de realidade aumentada os eventos.

Quadro 7 - Caso de uso do dispositivo UC06

3.2.2.1.8 Realizar inscrição

Neste caso de uso o usuário através tanto da tela de realidade aumentada quanto da tela de mapa, se inscrever em um evento. Mais detalhes estão descritos no Quadro 8.

UC12 – Realizar inscrição	
Descrição	Permite ao usuário se inscrever em eventos.
Requisito atendido	RF05
Pré-condições	Usuário deve estar autenticado no sistema e no menu principal da aplicação ter pressionado o botão "*mapa*".
Cenário principal	1. Usuário clica em um evento; 2. Sistema exibe o menu de contexto com a opção `Inscrever-se`; 3. Usuário clica no botão `Inscrever-se`; 4. Sistema envia uma solicitação de inscrição para o servidor; 5. Sistema exibe uma mensagem de sucesso na inscrição; 6. Sistema retorna ao mapa.
Fluxo alternativo 01	No passo 4 se o servidor retornar que não é possível se inscrever, o sistema exibe uma mensagem informando ao usuário e passa para o passo 6.
Exceção	No passo 4 se ocorrer erro de comunicação com o servidor o sistema exibe uma mensagem para usuário.
Pós-condições	Usuário inscrito em evento.

Quadro 8 - Caso de uso do dispositivo UC12

3.2.2.2 Casos de uso do *site*

A seguir serão mostrados os principais casos de uso que correspondem ao Site da ferramenta construída.

3.2.2.2.1 Postar eventos

Neste caso de uso o professor consegue efetuar o cadastramento de eventos no sistema através do *site*. Mais detalhes estão descritos no Quadro 9.

UC03 – Postar eventos	
Descrição	Permite ao professor postar eventos.
Requisito atendido	RF08
Pré-condições	Professor autenticado no site e ter pressionado no menu `Postar eventos`.
Cenário principal	1. Sistema exibe campos de entrada para descrição, data do evento e localização; 2. Professor preenche os campos solicitados; 3. Professor pressiona o botão `Postar`; 4. Sistema envia um evento para o servidor; 5. Sistema exibe uma mensagem de sucesso na postagem; 6. Sistema vai para a tela `Postar eventos`.
Fluxo alternativo 01	No passo 3 o sistema valida se professor preencheu todos os campos.
Exceção	N/A.
Pós-condições	Evento postado.

Quadro 9 - Caso de uso do site UC03

3.2.2.2.2 Postar notas aluno

Neste caso de uso o professor consegue através do site postar notas de um aluno. Mais detalhes estão descritos no Quadro 10.

UC06 – Postar notas aluno	
Descrição	Permite ao professor postar notas.
Requisito atendido	RF10
Pré-condições	Professor autenticado no site e ter pressionado no menu `Administrar alunos`.
Cenário principal	1. Sistema exibe as disciplinas que o professor leciona; 2. Professor seleciona uma disciplina; 3. Sistema busca no servidor os alunos da disciplina selecionada; 4. Sistema monta uma lista com o retorno do servidor; 5. Professor seleciona um aluno na lista; 6. Sistema exibe campos de entrada para: nota e freqüência; 7. Professor informa nota e clica no botão `Postar`; 8. Sistema envia a postagem para o servidor; 9. Sistema mostra uma mensagem de sucesso;
Exceção	N/A.
Pós-condições	Nota postada.

Quadro 10 - Caso de uso do site UC03

3.2.2.3 Casos de uso do servidor

A seguir serão mostrados os principais casos de uso do servidor.

3.2.2.3.1 Manter localização

O Servidor persiste em um banco de dados a localização enviada do dispositivo através de um serviço executando em *background* no dispositivo, que tem como função atualizar a localização geográfica do usuário. Mais detalhes serão mostrados no Quadro 11.

UC05 – Postar notas aluno	
Descrição	Persiste em um banco de dados a ultima localização do usuário.
Requisito atendido	RF02
Pré-condições	Dispositivo envia atual localização geográfica do usuário do sistema.
Cenário principal	1. Servidor recebe a localização; 2. Servidor abre uma conexão com o banco de dados e persiste a localização do usuário; 3. Servidor retorna um código de sucesso para o dispositivo;
Exceção	N/A.
Pós-condições	Localização atualizada.

Quadro 11 - Caso de uso do servidor UC05

3.2.2.3.2 Notificar dispositivo

O Servidor percebe que o algum usuário está próximo a um evento e dispara uma notificação para o dispositivo, o mesmo acontece se for postada nota, freqüência ou novo evento, fazendo com que o servidor mantenha o dispositivo atualizado. Mais detalhes serão mostrados no Quadro 12.

UC10 – Notificar Dispositivo	
Descrição	Servidor envia notificações caso o dispositivo esteja próximo a um evento, caso seja cadastrado algum novo evento, nota ou freqüência.
Requisito atendido	RF11
Pré-condições	Dispositivo preparado para recebimento de *push*.
Cenário principal	1. Servidor verifica se o dispositivo está próximo a algum evento; 2. Servidor percebe que dispositivo está próximo a um evento e envia notificação para o dispositivo.
Fluxo alternativo 01	1. Servidor recebe um novo cadastro de evento; 2. Servidor busca todos os usuários da ferramenta e notifica dispositivos.
Fluxo alternativo 02	1. Servidor recebe uma nova postagem de freqüência; 2. Servidor busca qual usuário recebeu atualização de freqüência; 3. Servidor envia um *push* para o dispositivo do usuário.
Fluxo alternativo 03	1. Servidor recebe uma nova inserção de nota; 2. Servidor busca qual usuário recebeu a nota; 3. Servidor envia um *push* para o dispositivo do usuário.
Exceção	N/A.
Pós-condições	Dispositivos notificados.

Quadro 12 - Caso de uso do servidor UC10

3.2.3 Diagramas de classe

A seguir serão mostrados os diagramas de classe, primeiramente uma abordagem na diagramação da aplicação no dispositivo, em seguida o servidor SOA e por fim o site.

3.2.3.1 Diagramas de classe do dispositivo

Antes de detalhar de forma completa os diagramas, o dispositivo faz uso de outro trabalho de conclusão de curso de autoria de Vasselai (2010) nomeado `Furb RA` demonstrado a seguir pela Figura 13. A ferramenta se fez necessária para cumprir o aspecto de Realidade Aumentada presente na aplicação final. Na Figura 14 é possível visualizar os pacotes que compõem a ferramenta no dispositivo.

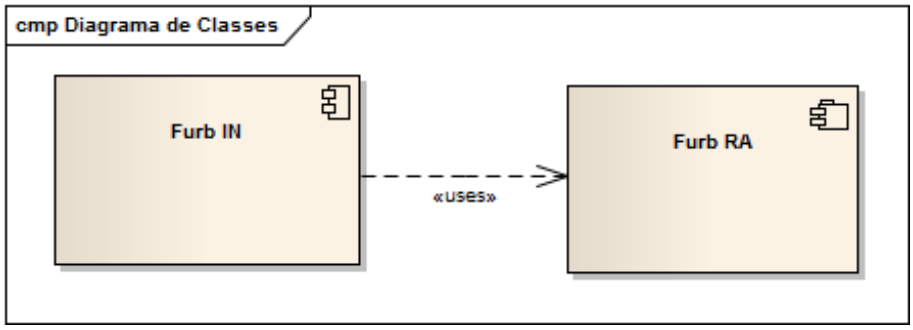


Figura 13 – Diagrama representando uso da ferramenta Furb RA

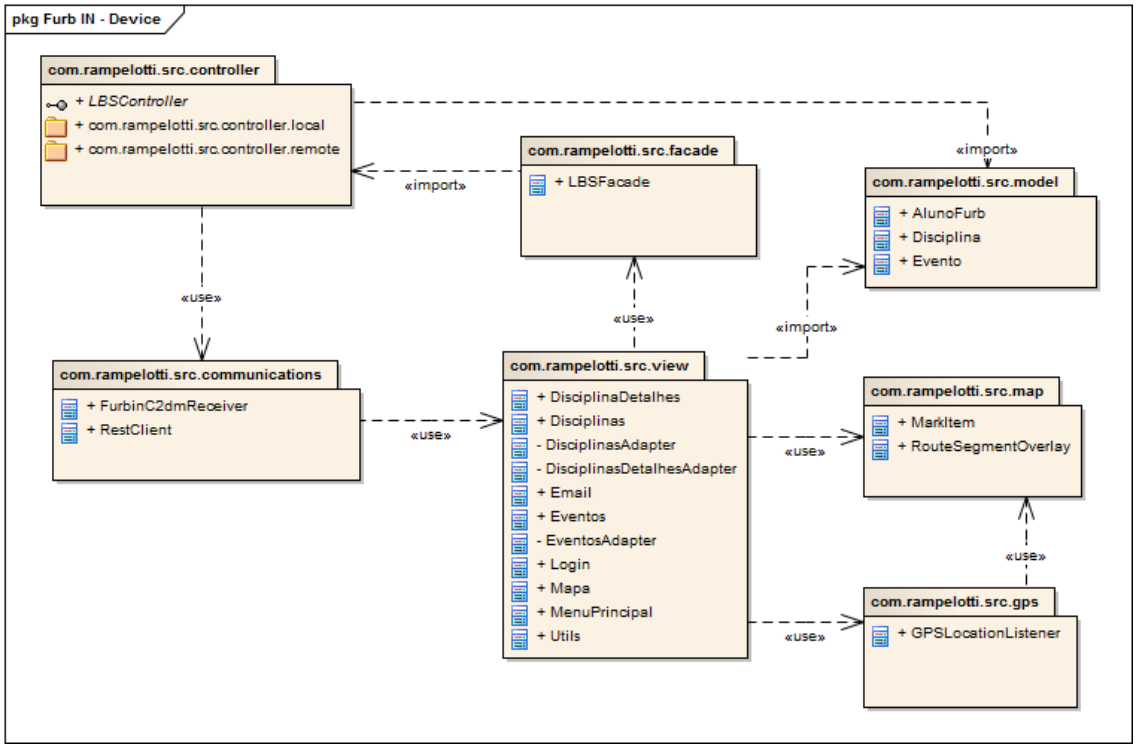


Figura 14 – Diagrama de pacotes que formam a ferramenta Furb IN

3.2.3.1.1 Pacote `com.rampelotti.src.view`

As classes que compõem este pacote fornecem telas para interação com o usuário. Na Figura 15 são mostradas as classes que compõem este pacote.

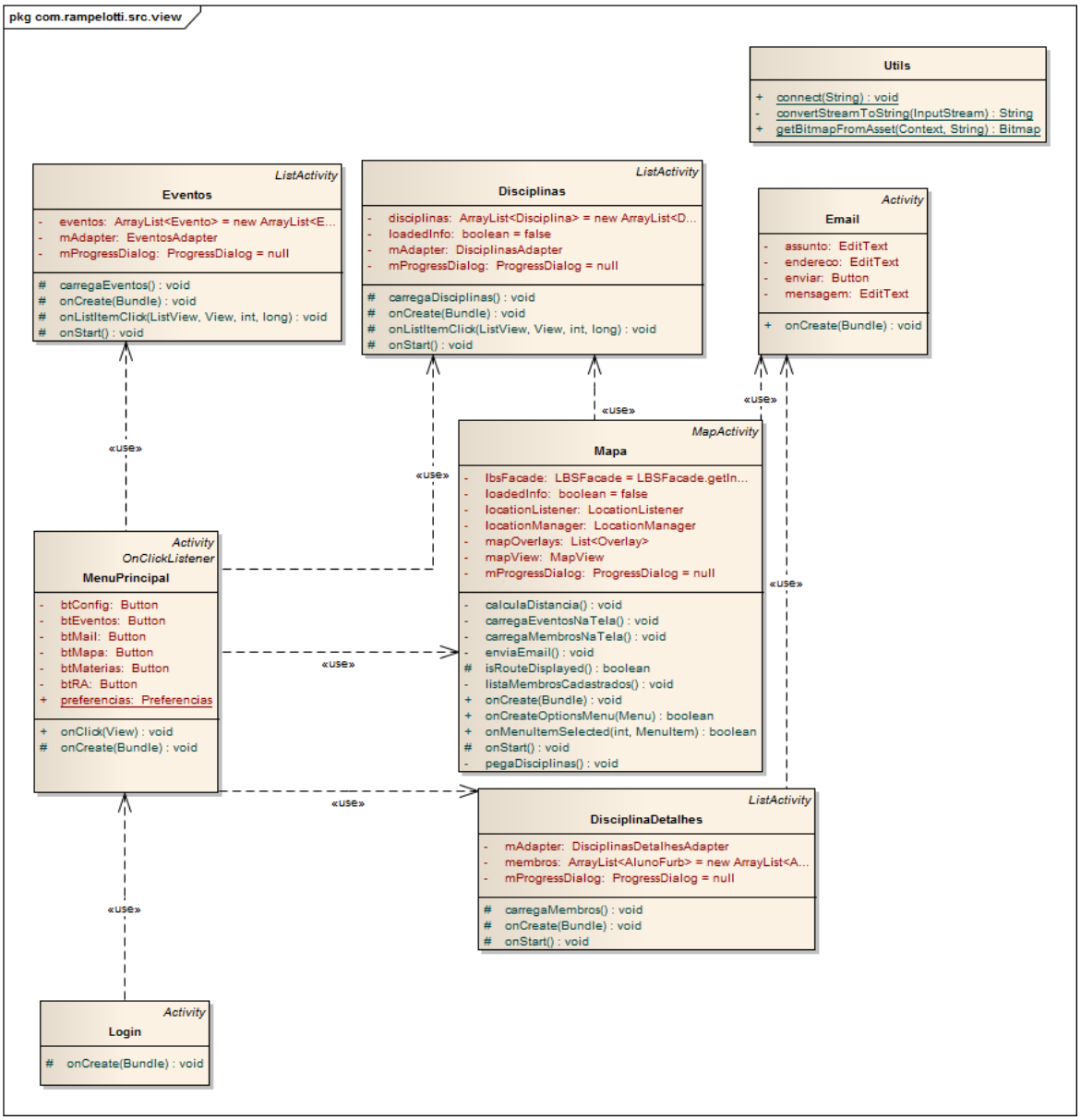


Figura 15 – Diagrama de classes do pacote `com.rampelotti.src.view`

A classe `Login` representa a primeira tela da aplicação, onde permite ao usuário efetuar acesso às demais funcionalidades da aplicação após se logar. A classe `MenuPrincipal` é a classe que disponibiliza para o usuário um menu de acesso as demais funcionalidades da aplicação. A classe `Eventos` exibe uma tela com todos os eventos cadastrados no sistema, permitindo que o usuário possa se inscrever nos mesmos. A classe `Disciplinas` exibe uma lista de disciplinas que o usuário está matriculado, permitindo que o usuário possa selecionar uma disciplina na lista exibida. A classe `DisciplinaDetalhes` exibe um detalhamento da disciplina com a freqüência do aluno e uma lista de membros que cursam esta mesma disciplina.

Este pacote conta com a classe `Mapa` que proporciona ao usuário uma ambientação através de uma figura geográfica. A tela ainda exibe os eventos em forma de símbolos, que

são dispostos em suas posições geográficas no mapa. É possível acompanhar onde o usuário está localizado geograficamente, visualizar também onde estão os membros da disciplina e interagir com eles através de envio de e-mail.

A classe `Utils` possui métodos auxiliares para utilidade na tela, como `getBitmapFromAsset` que serve para conversão de imagens que estão disponíveis para a aplicação em forma de recursos.

3.2.3.1.2 Pacote `com.rampelotti.src.controller`

Esse pacote (Figura 16) tem como finalidade manter duas classes que representam a forma de captura de dados para consumo da aplicação. Estes dados podem ser representados de duas formas: locais e remotos. Para dados locais é utilizada a classe `LBSControllerLocal` e dados remotos pela classe `LBSControllerRemoto`. Ambas as classes herdando da classe `LBSController` e implementando os métodos `popularDisciplina`(), `popularEventos`() e `popularMembros`(). Cada método carrega os dados de acordo com sua respectiva função.

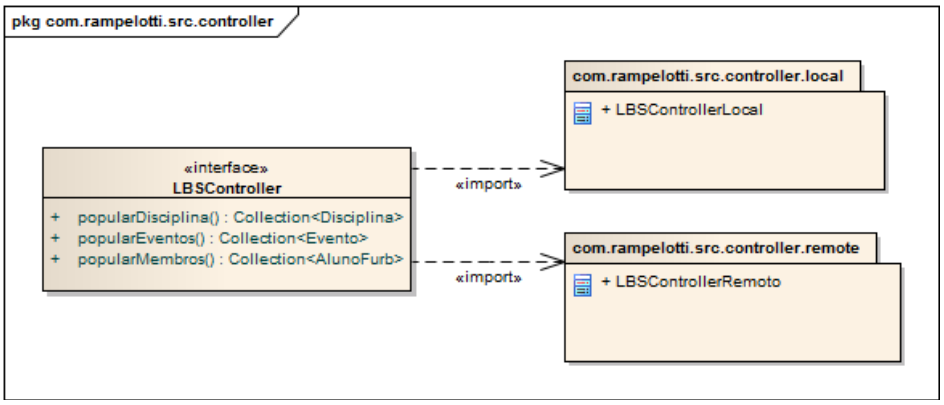


Figura 16 – Diagrama de classe do pacote `com.rampelotti.src.controller`

3.2.3.1.3 Pacote `com.rampelotti.src.facade`

Este pacote (Figura 17) é composto por uma única classe que serve para abstrair a camada de captura de dados da camada de visão.

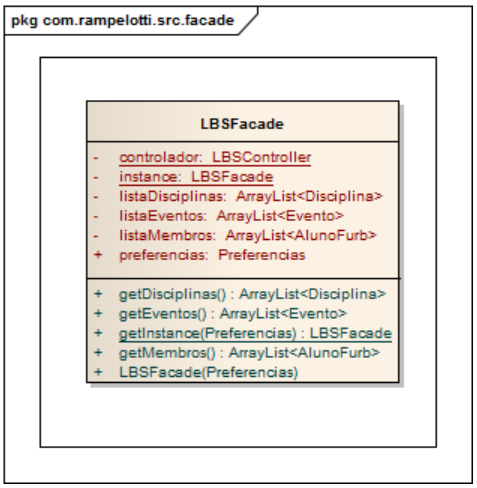


Figura 17 – Diagrama de classe do pacote `com.rampelotti.src.facade`

Essa classe utiliza-se de um mecanismo de única instancia conhecido como *singleton*. Através dessa classe obtém-se a lista de disciplinas, eventos e membros pelos seus respectivos métodos `getDisciplinas`(), `getEventos`() e `getMembros`(). Ao gerar a instância *singleton* essa classe verifica se o sistema está rodando com dados obtidos na web ou locais e instancia o atributo `controlador` de acordo com o modo de captura de dados citado.

3.2.3.1.4 Pacote `com.rampelotti.src.model`

Neste pacote (Figura 18) são dispostas as classes modelo da ferramenta criada.

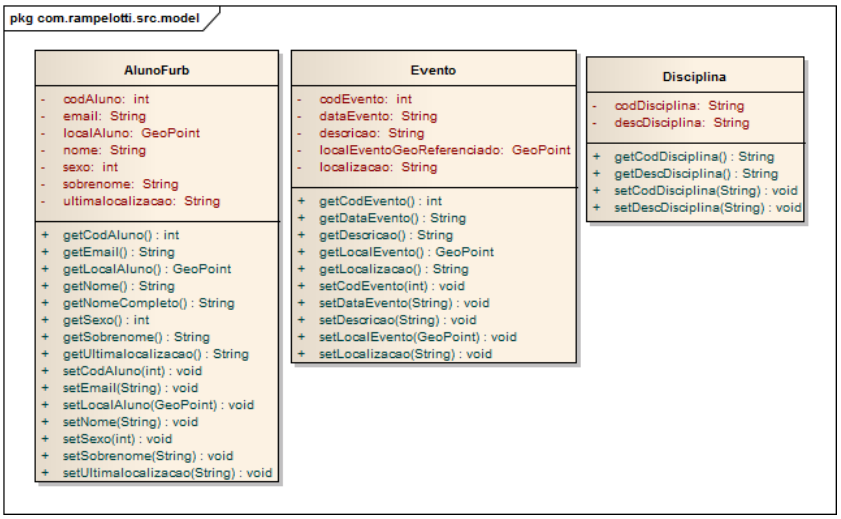


Figura 18 – Diagrama de classes do pacote `com.rampelotti.src.model`

A classe `AlunoFurb` representa os membros da disciplina, sendo possível através do atributo `localAluno` saber onde o aluno está geograficamente. Através do atributo `email` é possível enviar um e-mail para o mesmo.

A classe `Evento` representa eventos da instituição, como palestras, workshops,

atividades específicas abertas ao público, também utilizando através de seu atributo `localEvento` o lugar geograficamente situado do evento.

A classe `Disciplina` representa a disciplina do aluno, com os atributos `codDisciplina` e `descDisciplina` que representam o código e descrição da disciplina respectivamente.

3.2.3.1.5 Pacote `com.rampelotti.src.map`

A classe `MarkItem` tem como função ser um ponto de interesse do mapa, como eventos ou membros. A classe é consumida pela classe de tela `Mapa` que desenha os pontos de interesse de maneira que formem símbolos que indicam algum interesse para o usuário do sistema. A Figura 19 representa o pacote.

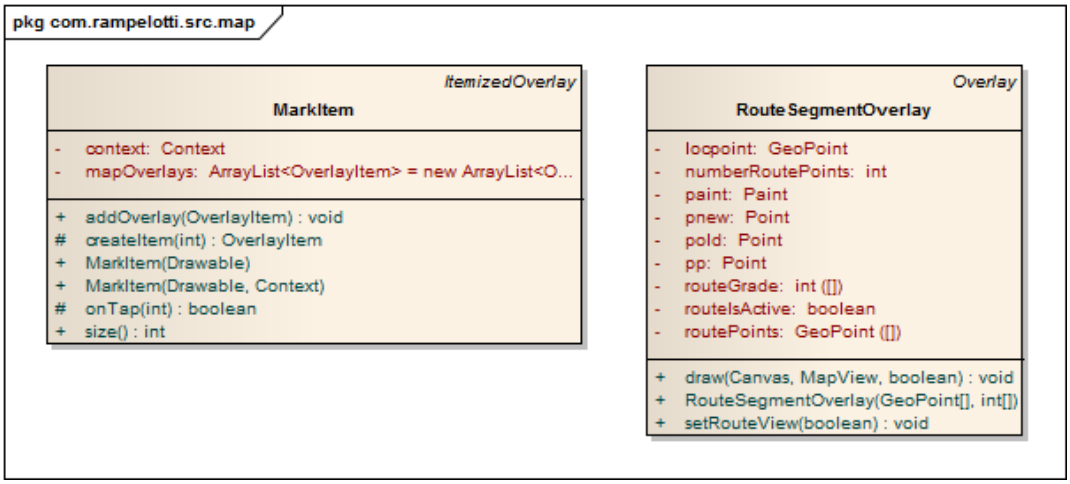


Figura 19 – Diagrama de classes do pacote `com.rampelotti.src.map`

A classe `RouteSegmentOverlay` serve para o desenho de rotas entre dois pontos, auxiliando o usuário em como chegar do ponto onde está até algum interesse específico, seja ele evento ou membro.

3.2.3.1.6 Pacote `com.rampelotti.src.gps`

Este pacote (Figura 20) possui a classe que interage com o *Global Positioning System* (GPS) do dispositivo, verificando sempre se o usuário teve algum deslocamento para poder atualizar as informações no servidor SOA.

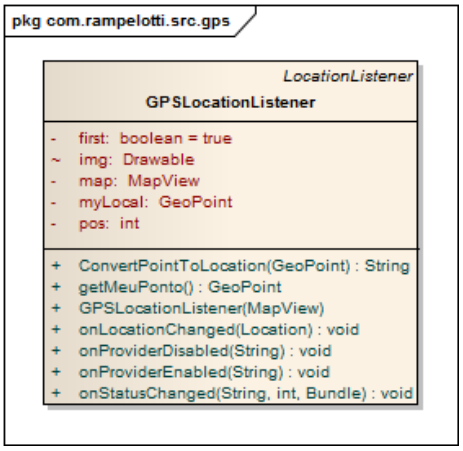


Figura 20 – Diagrama de classe do pacote `com.rampelotti.src.gps`

A classe `GPSLocationListener` serve exclusivamente para acesso as informações do GPS do dispositivo, atualiza na tela do mapa a posição do usuário com um símbolo indicando o usuário e envia em *background* para o servidor SOA qual a posição atual do usuário.

3.2.3.1.7 Pacote `com.rampelotti.src.communications`

Esse pacote possui as duas mais importantes classes do sistema para uso de notificação e comunicação utilizando serviços REST. A Figura 21 mostra o diagrama das classes deste pacote.

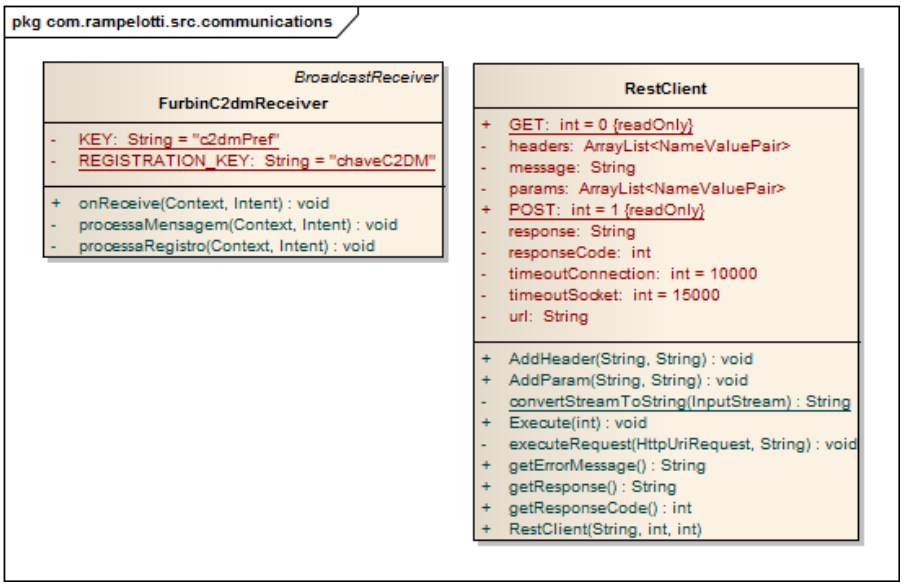


Figura 21 – Diagrama de classes do pacote com.rampelotti.src.communications

A classe `FurbinC2dmReceiver` é responsável pelo registro da aplicação no servidor Google onde deixa a aplicação elegível a receber notificações externas.

Quando o registro é feito no servidor Google, é reservado uma espécie de *ticket* que é

enviado para o dispositivo confirmando o registro. Nesse momento a classe `FurbinC2dmReceiver` intercepta o registro por ser uma classe de serviço no Android que implementa *BroadcastReceiver*, isso indica ao Sistema Operacional (SO) do Android que a classe fica verificando mensagens.

Após o recebimento da mensagem no método `onReceive`() o mesmo valida o corpo da mensagem, verificando se é uma mensagem de registro, uma mensagem de remoção de registro ou uma mensagem de notificação. Para mensagem de registro o método repassa para o método `processaRegistro`() onde pega o *ticket* recebido e envia para o servidor SOA, que mantém o *ticket* para futuras notificações. Caso a mensagem recebida seja apenas de notificação, o método repassa para o `processaMensagem`(), que processa a mensagem gerando notificações no dispositivo.

A classe `RestClient` é responsável por gerar a mensagem de requisição HTTP, montando o cabeçalho da requisição, os parâmetros de requisição e o corpo de mensagem. Após o envio da mensagem, a classe trata os códigos de retorno e separa o `response` da mensagem para consumo da classe que invocou a requisição ao servidor SOA.

3.2.3.2 Diagrama de classes do servidor SOA

Os próximos itens descrevem especificamente os pacotes que formam o servidor SOA. Primeiramente abordando o pacote de modelo e por fim o pacote de recursos de serviços disponibilizados em REST.

3.2.3.2.1 Pacote `com.rampelotti.server.model`

Pacote responsável por espelhar as entidades do banco de dados. A principal função da existência dessas classes nesse pacote é para a passagem de requisições REST em formato JSON utilizando anotações que transformam o objeto em um formato interpretável JSON. A Figura 22 mostra com mais detalhes as classes envolvidas nesse pacote.

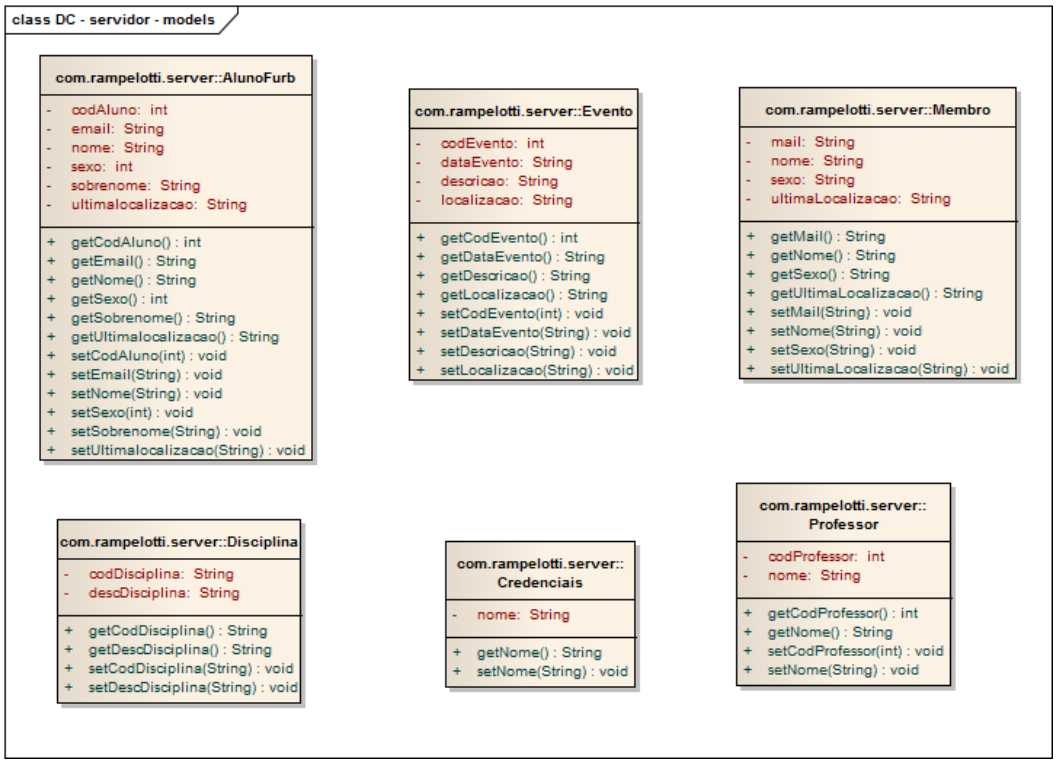


Figura 22 – Diagrama de classes do pacote `com.rampelotti.server.model`

3.2.3.2.2 Pacote `com.rampelotti.server.resources`

Este pacote contém os serviços acessados pelas aplicações que interagem com usuário, nota-se que os serviços são todos baseados em REST. A Figura 23 mostra com mais detalhes o diagrama de classes do pacote `com.rampelotti.server.resources`.

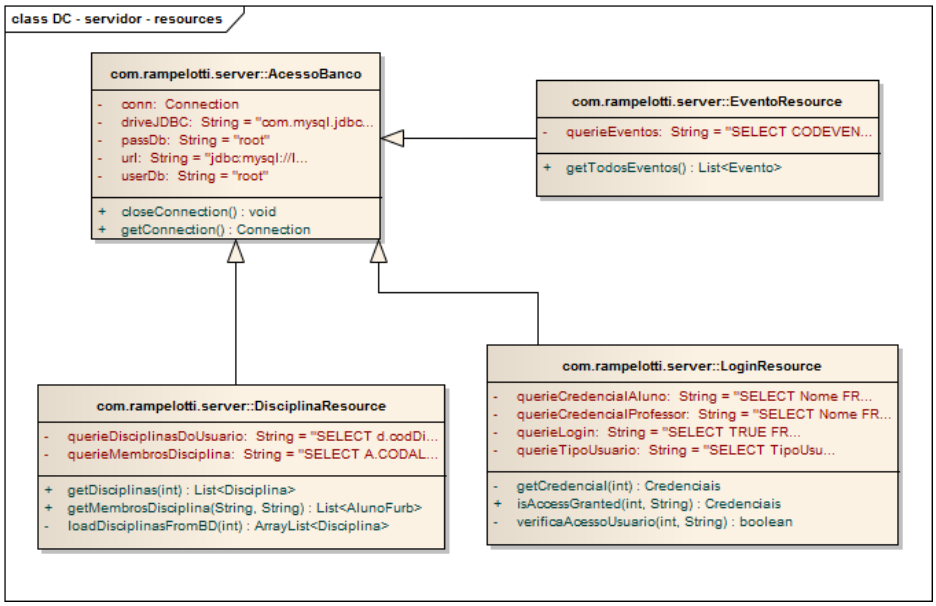


Figura 23 – Diagrama de classes do pacote `com.rampelotti.server.resources`

A classe `AcessoBanco` centraliza todas as chamadas ao banco, execução de *queries*, instâncias de *driver* para banco, possibilitando a mudança de banco de dados isolando a manutenção de código nessa classe.

A classe `DisciplinaResource` permite a execução de serviços de GET de membros de disciplina através do método `getMembrosDisciplina`() passando como parâmetro a disciplina e o código do aluno solicitante, o método executa a *query* no banco, somente colocando os parâmetros passados como filtro. O atributo de classe `querieMembrosDisciplina` representa a *query* que é executada no banco, somente deixando aberto para filtro o código da disciplina e o código do aluno solicitante, retornando uma lista de membros da disciplina em formato JSON. Outro método disponível é o `getDisciplinas`() que recebe como parâmetro o código do aluno solicitante, executa a *query* definida no atributo `querieDisciplinasDoUsuario`, aplicando filtro com o código de aluno recebido como parâmetro e retornando uma lista de Disciplinas em formato JSON.

A classe `EventoResource` permite a execução de serviços de GET e POST de eventos através de métodos como `getTodosEventos`() onde é executada uma *query* definida no atributo `querieEventos`, o método executa no banco, retornando uma lista de eventos em formato JSON, outro método é o `postEventos`() onde recebe como parâmetro os atributos do evento para persistência em banco.

A classe `LoginResource` é exclusiva para validação de *login* de usuário tanto no dispositivo como no *site*, validando o usuário do sistema via REST e retornando em formato JSON o resultado da verificação do *login*. Caso sucesso é enviado formato JSON da classe credencial.

3.2.3.3 Diagrama de classes do *site*

A seguir serão explanados os pacotes e os diagramas referentes ao site da aplicação construída. A Figura 24 mostra o diagrama dos pacotes que compõem a construção do site.

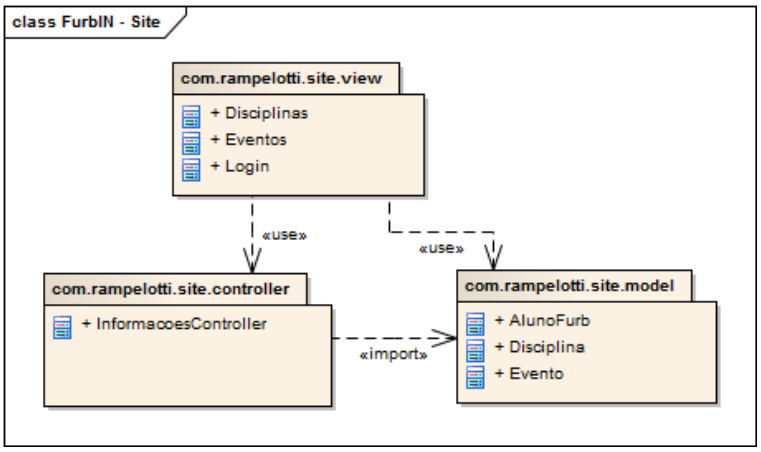


Figura 24 – Diagrama de pacotes que compõem o site da aplicação Furb IN

3.2.3.3.1 Pacote `com.rampelotti.site.model`

Este pacote (Figura 25) é composto somente por classes que representam os dados solicitados no banco para apresentação em telas.

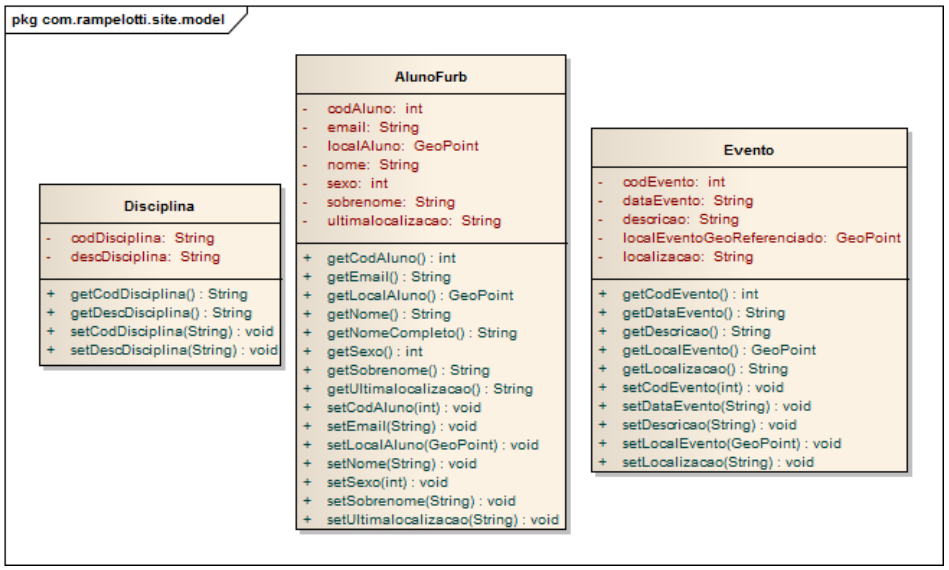


Figura 25 – Diagrama de classes do pacote `com.rampelotti.site.model`

3.2.3.3.2 Pacote `com.rampelotti.site.controller`

Neste pacote está contida a classe de comunicação com o banco de dados, para captura de dados, envio e recebimento da mesma. Na Figura 26 é demonstrada a composição deste pacote.

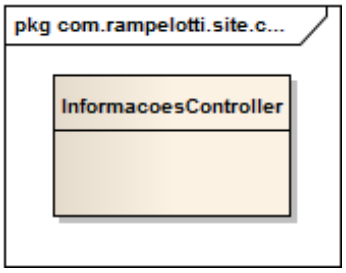


Figura 26 – Diagrama de classe do pacote `com.rampelotti.site.controller`

3.2.3.3.3 Pacote `com.rampelotti.site.view`

Este pacote (Figura 27) representa as telas de iteração com o ator `Professor`. Não será detalhado, pois assume funções similares ao dispositivo, diferenciando apenas que no *site* é possível realizar entradas de eventos e alterações de informações para membros de disciplina.

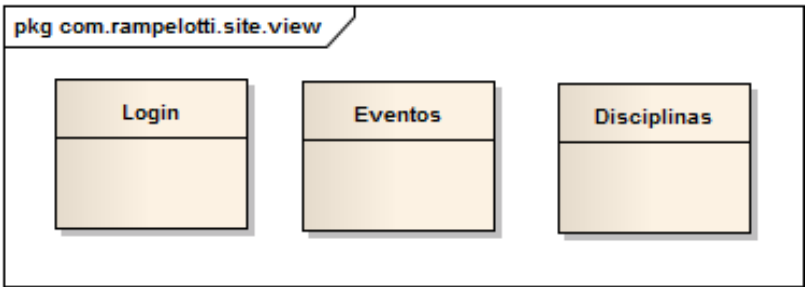


Figura 27 – Diagrama de classes do pacote `com.rampelotti.site.view`

3.3 IMPLEMENTAÇÃO

A seguir são descritas as técnicas e ferramentas utilizadas na implementação, bem como detalhes das principais classes e rotinas implementadas durante o desenvolvimento da aplicação.

3.3.1 Técnicas e ferramentas utilizadas

O desenvolvimento do sistema proposto foi efetuado em duas partes, paralelas e interligadas por uma parte de fornecimento de serviços baseados em SOA. As funcionalidades propostas foram desenvolvidas em um aplicativo cliente, a ser executado na plataforma Android. A parte que atende ao servidor de serviços baseados em localização foi feita em

J2EE, utilizando a biblioteca Jersey que permite a utilização de anotações para geração de chamadas REST para serviços *web*. No mesmo servidor executa uma aplicação em paralelo que verifica se os usuários estão próximos dos eventos.

O desenvolvimento da aplicação de serviços baseados em localização usando notificações foi feito na linguagem Java com a API de desenvolvimento do Android na versão 2.2, também conhecida por Froyo. O ambiente de desenvolvimento utilizado foi o Eclipse com o conjunto de *plugins* do Android *Development Tools* (ADT). Para a execução e a depuração da aplicação também foi utilizado um dispositivo *smartphone* da fabricante Motorola chamado Defy que possui o sistema operacional Android Froyo.

O dispositivo Defy possui processador de 800 *Mega Hertz* (mHZ), com memória de 2 *Giga Bytes* (GB) e resolução de 480x854 (MOTOROLA, 2011).

Para o servidor SOA da aplicação foi utilizada a especificação Java 2 *Platform Enterprise Edition* (J2EE) na definição de um `Servlet` que recebe requisições no protocolo HTTP com a URL de requisição baseada em REST. Os dados são obtidos a partir de um banco de dados MySQL na versão 5.2 através da interface *Java DataBase Connectivity* (JDBC). O servidor utilizado para executar a aplicação *web* foi o Apache Tomcat na versão 6.0.

3.3.2 Preparando ambiente Android para o *push*

O arquivo `AndroidManifest.xml` possui informações necessárias para as aplicações Android executarem. Para utilizar o serviço de notificações da Google, a aplicação deve manter um serviço de registro no servidor da Google. Este serviço indica ao servidor da Google que a aplicação está apta para receber notificações. Na Figura 28 é exibido trexo de código XML que informa pelo *manifest* qual serviço faz o papel de receptor de notificações.

```
<receiver android:name="com.rampelotti.src.communications.FurbinC2dmReceiver"
    android:permission="com.google.android.c2dm.permission.SEND">

    <!-- Receptor para notificação-->
    <intent-filter>
        <action android:name="com.google.android.c2dm.intent.RECEIVE" />
        <category android:name="com.rampelotti.src" />
    </intent-filter>

    <!-- Receptor para registro -->
    <intent-filter>
        <action android:name="com.google.android.c2dm.intent.REGISTRATION" />
        <category android:name="com.rampelotti.src" />
    </intent-filter>
</receiver>
```

Figura 28 – Fragmento do manifesto para indicar a classe receptora de notificação

Após indicar para o Android qual será a classe que consome as notificações, deve ser indicado que a aplicação tem permissão para registrar e receber as mensagens, e também que pode manipular a mensagem. A Figura 29 mostra como é feito no manifesto.

```
<!--
    Somente essa aplicação pode receber as mensagens e registrar resultado
-->
<permission android:name="com.rampelotti.src.permission.C2D_MESSAGE"
    android:protectionLevel="signature" />
<uses-permission android:name="com.rampelotti.src.permission.C2D_MESSAGE" />

<!--
    Essa aplicação tem permissão para se registrar e receber notificações
-->
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
```

Figura 29 – Fragmento do manifesto de permissão da aplicação para notificação

Após configurada, a aplicação deve iniciar o processo de registro no servidor C2DM da Google para receber um *ticket* que representa o dispositivo na Google e enviar esse *ticket* para o servidor SOA para envio de notificações. A Figura 30 mostra o trecho de código onde é inicializado o serviço de registro do dispositivo.

```
// Disparando registro de notificações
Intent registrationIntent = new Intent("com.google.android.c2dm.intent.REGISTER");
registrationIntent.putExtra("app", PendingIntent.getBroadcast(getApplicationContext(), 0, new Intent(), 0));
registrationIntent.putExtra("sender", "ronaldo.rampelotti@gmail.com");
getApplicationContext().startService(registrationIntent);
```

Figura 30 – Fragmento de código para disparo de registro de notificação

Com o serviço de registro iniciado, entra em ação a classe que faz o papel de receptora `FurbinC2dmReceiver`, definida no manifesto. Essa classe intercepta pelo método `onReceive` sobrescrito da classe `BroadcastReceiver`. Na Figura 31 segue o trecho de código que captura as mensagens enviadas para o dispositivo e que são redirecionadas para processar a mensagem recebida. Dentro do processamento temos: processamento de notificação e processamento de registro de notificação, o último responsável por enviar para o

servidor SOA o *ticket* do servidor Google que deve ser usado para envio de notificações *push* pelo servidor Google C2DM.

```
/**
 * Método invocando quando recebido alguma notificação.
 *
 */
@Override
public void onReceive(Context context, Intent intent) {
    // Verifica se recebeu notificação para registro senão é uma notificação simples de informação.
    if (intent.getAction().equals("com.google.android.c2dm.intent.REGISTRATION")) {
        processaRegistro(context, intent);
    } else if (intent.getAction().equals("com.google.android.c2dm.intent.RECEIVE")) {
        processaMensagem(context, intent);
    }
}
```

Figura 31 – Método de recebimento de notificações Android

O processo de registro retorna um *ticket* que deve ser redirecionado para o servidor que gerará notificações, pois o servidor necessita deste *ticket* para poder enviar notificações para o dispositivo. O processo de envio para o servidor SOA é feito pelo `LBSController` no método `atualizaPush` através de uma chamada HTTP informando ao servidor o *ticket* para envio de notificações.

O consumo de notificações no dispositivo gera um alerta na área de notificações, mostrando qual o evento que o usuário está próximo e permitindo o usuário clicar na notificação para abrir a aplicação. Na Figura 32 é demonstrado o trecho de código de geração de notificação para Android.

```
/**
 * Método de criação da notificação.
 *
 * @param ctx
 */
private void criaNotificacao(Context ctx) {
    // Recupera o serviço do NotificationManager
    NotificationManager notificationManager = (NotificationManager)  ctx.getSystemService( Context.NOTIFICATION_SERVICE);
    Notification notificacao = new Notification(R.drawable.icon, "Furbin", System.currentTimeMillis());
    notificacao.sound = Uri.withAppendedPath(Audio.Media.INTERNAL_CONTENT_URI, "6");
    // Flag que vibra e emite um sinal sonoro até o usuário clicar na notificação
    notificacao.flags |= Notification.FLAG_INSISTENT;
    // Flag utilizada para remover a notificação da toolbar quando usuário tiver clicado nela.
    notificacao.flags |= Notification.FLAG_AUTO_CANCEL;
    // PendingIntent para executar a Activity se o usuário selecionar a notifição
    PendingIntent p = PendingIntent.getActivity(ctx, 0, new Intent(ctx.getApplicationContext(), MenuPrincipal.class), 0);
    // Informações
    notificacao.setLatestEventInfo(ctx, "Evento", "Você está próximo ao evento", p);
    // Espera 100ms e vibra por 1000ms, depois espera por 1000 ms e vibra por 1000ms.
    notificacao.vibrate = new long[] { 100, 1000, 1000, 1000 };
    // Id que identifica esta notifição
    notificationManager.notify(R.string.app_name, notificacao);
}
```

Figura 32 – Geração de notificação Android

3.3.3 Envio de notificações para o dispositivo Android pelo servidor SOA

No servidor SOA foi implementado o envio de notificações, conforme regra de serviços baseados em localização definida para a aplicação, pré-requisito para o envio é

possuir o *ticket* do dispositivo. Isto é feito apenas porque o envio é feito para um servidor da Google de C2DM. Com o *ticket*, recebido pelo dispositivo, o servidor precisa primeiramente de uma sessão autenticada no servidor Google. Na Figura 33 é demonstrado um trecho de código de captura de sessão com servidor Google de C2DM.

```
/**
 * Método para pegar o Authorization google.
 *
 * @return chave para acesso google.
 */
public String getGoogleAuth() {
    String authToken = getAuthBanco();
    if (authToken != null) {
        System.out.println("Pegando token do banco: " + authToken);
        return authToken;
    }

    System.out.println("Solicitando para o servidor C2DM um token de autorização...");

    StringBuilder buf = new StringBuilder();
    HttpsURLConnection.setDefaultHostnameVerifier(new FalsoHostnameVerifier());
    HttpsURLConnection request = null;
    OutputStreamWriter post = null;
    try {
        URL url = new URL("https://www.google.com/accounts/ClientLogin");
        request = (HttpsURLConnection) url.openConnection();
        request.setDoOutput(true);
        request.setDoInput(true);

        buf.append("accountType").append("=").append((URLEncoder.encode("GOOGLE", "UTF-8")));
        buf.append("&Email").append("=").append((URLEncoder.encode(mail, "UTF-8")));
        buf.append("&Passwd").append("=").append((URLEncoder.encode(pass, "UTF-8")));
        buf.append("&service").append("=").append((URLEncoder.encode("ac2dm", "UTF-8")));
        buf.append("&source").append("=").append((URLEncoder.encode("myco-pushapp-1.0", "UTF-8")));

        request.setRequestMethod("POST");
        request.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
```

Figura 33 – Requisitando token de autenticação no servidor C2DM.

Com a autenticação recebida, agora o servidor pode mandar uma notificação para o dispositivo. A Figura 34 mostra detalhes para envio de notificação.

```
/**
 * Método de envio do push
 *
 * @param authToken
 * @param collapseKey
 * @param registrationId
 * @param message
 * @throws Exception
 */
private void enviaPush(String authToken, String collapseKey,
        String registrationId, String message) throws Exception {
    System.out.println("Enviando push...");

    HttpsURLConnection
            .setDefaultHostnameVerifier(new FalsoHostnameVerifier());
    URL url = new URL("https://android.apis.google.com/c2dm/send");
    HttpsURLConnection request = (HttpsURLConnection) url.openConnection();
    request.setDoOutput(true);
    request.setDoInput(true);

    StringBuilder buf = new StringBuilder();
    buf.append("registration_id").append("=").append((URLEncoder.encode(registrationId, "UTF-8")));
    buf.append("&collapse_key").append("=").append((URLEncoder.encode(collapseKey, "UTF-8")));
    buf.append("&data.payload").append("=").append((URLEncoder.encode(message, "UTF-8")));

    request.setRequestMethod("POST");
    request.setRequestProperty("Content-Type","application/x-www-form-urlencoded");
    request.setRequestProperty("Content-Length",buf.toString().getBytes().length + "");
    request.setRequestProperty("Authorization", "GoogleLogin auth="+ authToken);

    OutputStreamWriter post = new OutputStreamWriter(request.getOutputStream());
    post.write(buf.toString());
```

Figura 34 – Envio de notificação push para dispositivo

No envio da notificação o *ticket* é informado no parâmetro `registration_id` os dados da notificação ficam dentro do parâmetro `data.payload` e por fim para poder se comunicar respeitando a segurança do servidor da Google, é enviado o token de autorização onde deve ser requisitado junto ao servidor como já explicado anteriormente.

3.3.4 Motor de serviços baseados em localização

Este é o núcleo do processo de serviços baseados em localização, é através deste processo que é verificada a proximidade dos alunos a um evento FURB. O funcionamento deste motor possui uma *thread*, onde são verificados os eventos cadastrados e se os alunos estão próximos a estes eventos, gerando uma notificação para o aluno que esteja localizado em um raio de um quilômetro de distância do evento. Na Figura 35 é exibido um trecho de código de serviços baseados em localização.

```
// Itera sobre os alunos
if (retAlunos.first()) {
    do {
        String local = retAlunos.getString(1);
        if (null != local && (local.lastIndexOf("#") != -1)) {
            String[] longlat = local.split("#");
            double lgAluno = Double.valueOf(longlat[1])* Math.pow(10, -6);
            double ltAluno = Double.valueOf(longlat[0])* Math.pow(10, -6);
            double kmDistancia = getDistancia(Math.toRadians(ltEvento),Math.toRadians(lgEvento),Math.toRadians(ltAluno),Math.toRadians(lgAluno)) / 1000;
            // Verifica se está proximo ao evento em torno de 1 km
            if (kmDistancia <= 1) {
                String ticketGoogle = retAlunos.getString(3);
                if (ticketGoogle != null && !ticketGoogle.equals("")) {
                    try {
                        enviaPush(getGoogleAuth(), "0", ticketGoogle, retEventos.getString(3));
                    } catch (Exception e) {
                        System.out.println("Envio mal sucedido!");
                    }
                }
            }
        }
    } while (retAlunos.next());
}
```

Figura 35 – Trecho de código do motor de serviços baseados em localização

O método `getDistancia` foi feito para calcular a distancia entre dois pontos, no caso a distância entre um evento e o aluno. Para o cálculo foi considerada a fórmula de obtenção da distância entre dois pontos geográficos levando em conta o raio da terra.

3.3.5 Geração de serviços SOA utilizando REST e retornando JSON

Para a camada de serviços da aplicação, foi empregado o uso de serviços em REST que conta com anotações para a geração de assinaturas de serviços, simplificando a chamada dos serviços. Para gerar menos tráfego de banda no dispositivo móvel, foi utilizada a geração de dados de retorno dos serviços em JSON, utilizando anotações, simplificando a geração dos mesmos.

Os serviços REST são acessados diretamente pela URL do serviço formando um acesso simples e direto ao que se deseja. Para indicar como deve ser acessado o serviço pela URL, é utilizada a anotação `@Path` que recebe como parâmetro uma `String` que indica o caminho de acesso. Para definir qual vai ser o tipo de método de acesso ao serviço, podem ser usadas anotações como: `@GET`, `@POST`, `@PUT`, `@OPTION`. Por fim para determinar o tipo de retorno do serviço, deve ser indicado com a anotação `@Produces`. Um exemplo de definição de serviço REST é apresentado na Figura 36.

```
@Path("/eventos")
public class EventoResource extends AcessoBanco {

    private String querieEventos = "SELECT CODEVEN

    @GET
    @Produces("application/json; charset=utf-8")
    public List<Evento> getTodosEventos() {
```

Figura 36 – Assinatura de serviço REST

O retorno via JSON é feito de forma implícita pelo conteiner. Uma vez que utilizando anotações nas classes de modelo facilitam a geração, o único trabalho de implementação para geração JSON é a aplicação da anotação `@XmlRootElement` que indica que a classe modelo será formatada toda em nodos com valor. Essa anotação pode ser visualizada na Figura 37.

```
package com.rampelotti.models;

import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement
public class Evento {

    private int codEvento;

    private String descricao;
```

Figura 37 – Utilização de anotação

3.3.6 Operacionalidade da implementação

A operacionalidade da aplicação é apresentada na forma de funcionalidades, sendo representados através de imagens. Serão apresentadas imagens da aplicação no dispositivo Motorola Defy que foi utilizado durante o desenvolvimento do projeto, por fim será apresentada a ferramenta Web que engloba as funcionalidades de entrada de dados para o consumo dos dispositivos.

Primeiramente, para acesso à aplicação deve ser efetuada a autenticação na tela representada pela Figura 38.

Figura 38 – Tela de login

Após autenticar-se, o usuário terá acesso a tela de menu da aplicação, onde pode optar por visualizar um mapa com os eventos, as disciplinas que está cursando, os eventos disponíveis, enviar um e-mail, visualizar os eventos com auxílio de realidade aumentada e por fim configurar a aplicação. Esta tela pode ser visualizada pela Figura 39.

Figura 39 – Menu principal da aplicação

Quando o usuário clicar no botão `Mapa` a aplicação apresenta um mapa com os eventos disponíveis carregados em suas localizações geográficas, dando uma noção geográfica da relação entre local onde o usuário se encontra e o evento conforme Figura 40.

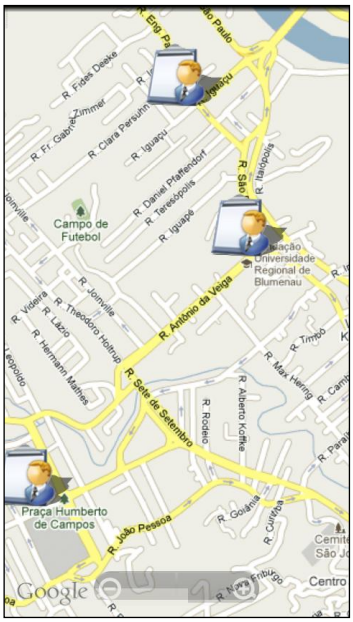


Figura 40 – Tela mapa

Na mesma tela é possível visualizar os membros de uma disciplina em comum no mapa, clicando no menu de contexto `Localizar membros da disciplina`. É exibida uma tela com as disciplinas que o usuário está matriculado, após a seleção da disciplina o mapa é carregado

agora com pontos indicando os membros da disciplina no mapa na sua última localização geográfica. A Figura 41 mostra as telas até pontos dos usuários carregados no mapa.

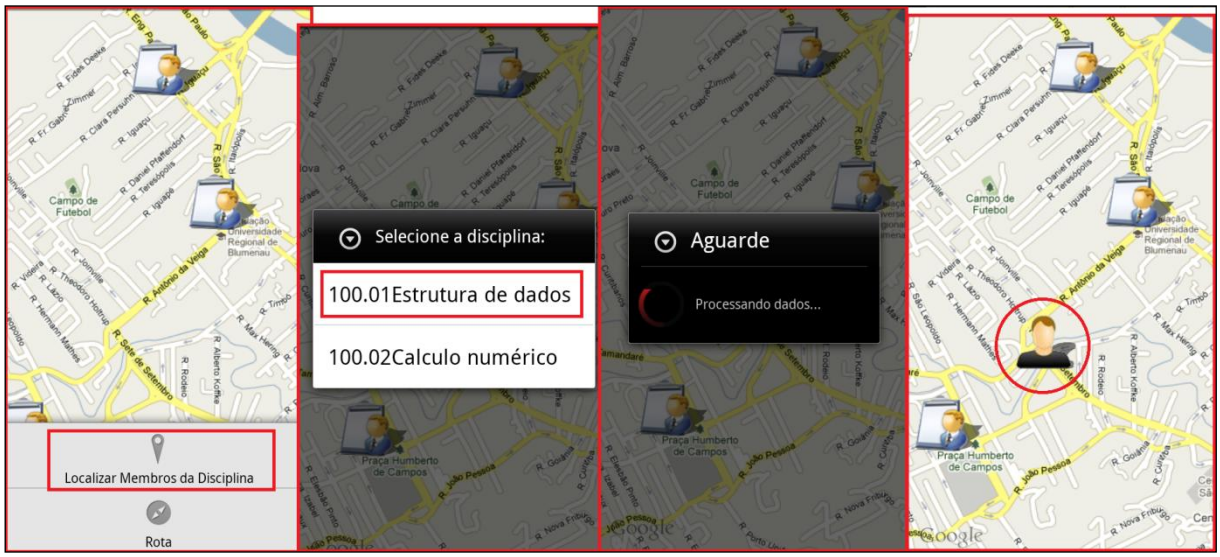


Figura 41 – Telas até plotagem de membros da disciplina no mapa

Voltando ao menu principal se o usuário clicar no botão `Matérias`, irá receber uma lista das disciplinas que ele está matriculado. Com a lista carregada, o usuário pode clicar em uma das disciplinas, podendo assim verificar sua freqüência na disciplina e também os membros da disciplina. A Figura 42 mostra o fluxo de telas até chegar à listagem de membros da disciplina e freqüência.

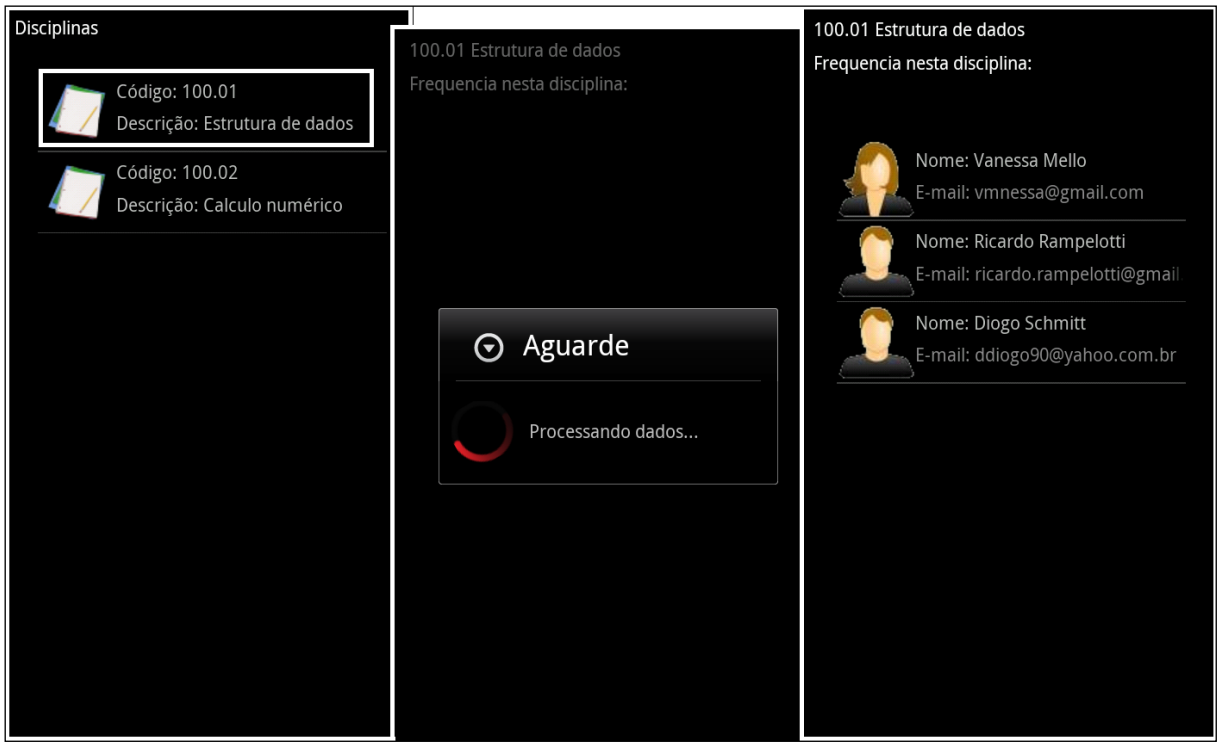


Figura 42 – Fluxo de telas até membros disciplina e freqüência

Com a lista de membros da disciplina carregada, é possível enviar um e-mail para um membro da disciplina. Isto pode ser feito clicando em um membro da lista. A aplicação questiona

se deseja ou não enviar e-mail para a pessoa selecionada. Caso confirme, é exibida uma lista de aplicações no cliente que enviam e-mail, deixando a cargo do usuário escolher qual aplicativo de e-mail ele deseja utilizar. Após a seleção da aplicação, é carregada a tela para compor o e-mail, com algumas informações já preenchidas como destinatário, assunto e parte do corpo da mensagem, como é detalhado na Figura 43.

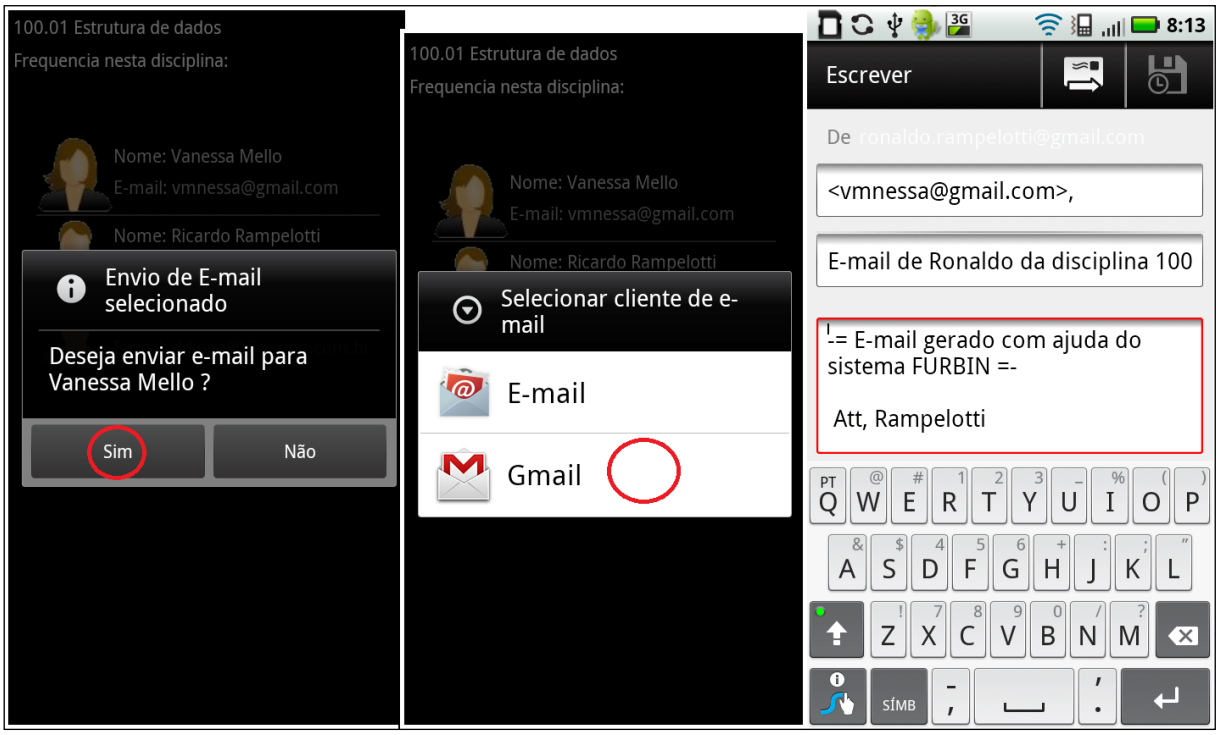


Figura 43 – Fluxo de telas para envio de *e-mail* para membro disciplina

Voltando para o menu principal, se for selecionado o botão `Eventos`, será exibida uma tela com uma lista de eventos disponíveis. Com isso é possível selecionar um evento e se inscrever no evento selecionado. A Figura 44 mostra o fluxo de telas de inscrição de evento.

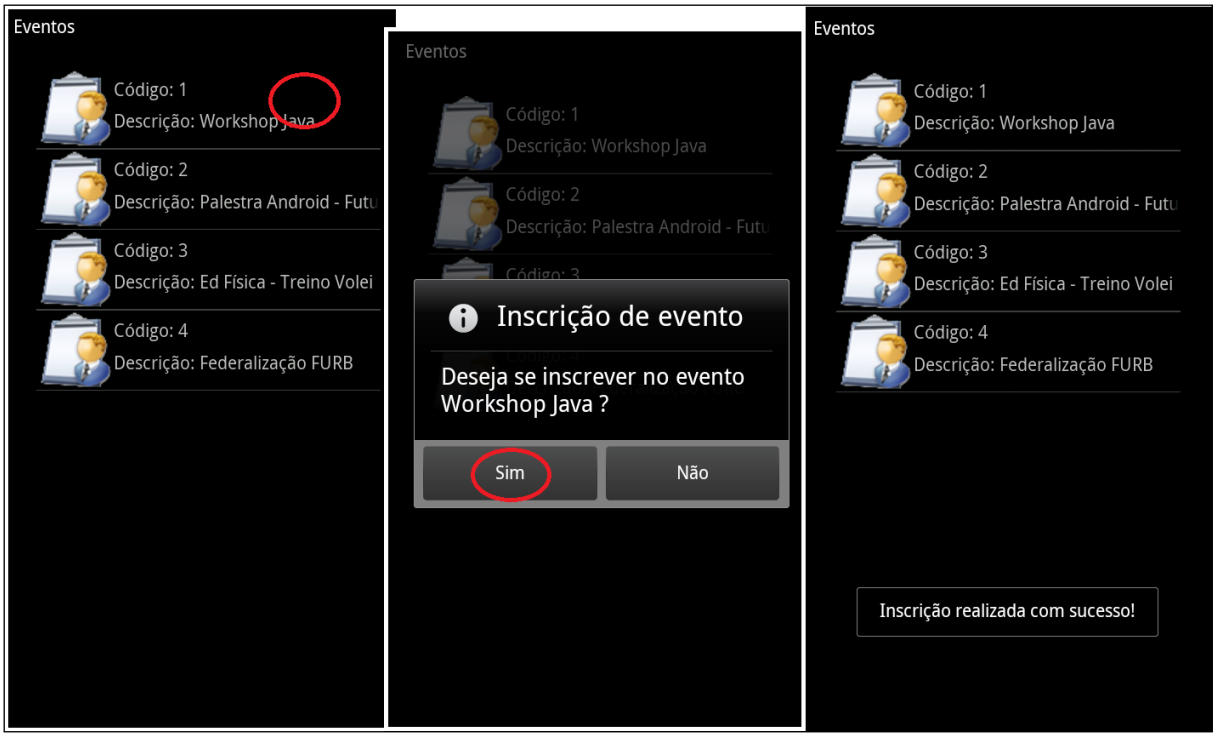


Figura 44 – Fluxo de telas para inscrição em evento

Voltando ao menu principal, clicando no botão `Email`, o usuário será questionado em qual cliente de e-mail ele gostaria de utilizar para o envio. Mesmo processo que na tela de membros da disciplina, a diferença é que não é preenchido o destinatário, deixando para o usuário escolher o mesmo.

Ainda no menu principal se o usuário selecionar o botão `Realidade Aumentada`, ele visualizará uma tela de realidade aumentada construída no trabalho de conclusão de curso de VASSELAI (2010). A Figura 45 demonstra essa tela.

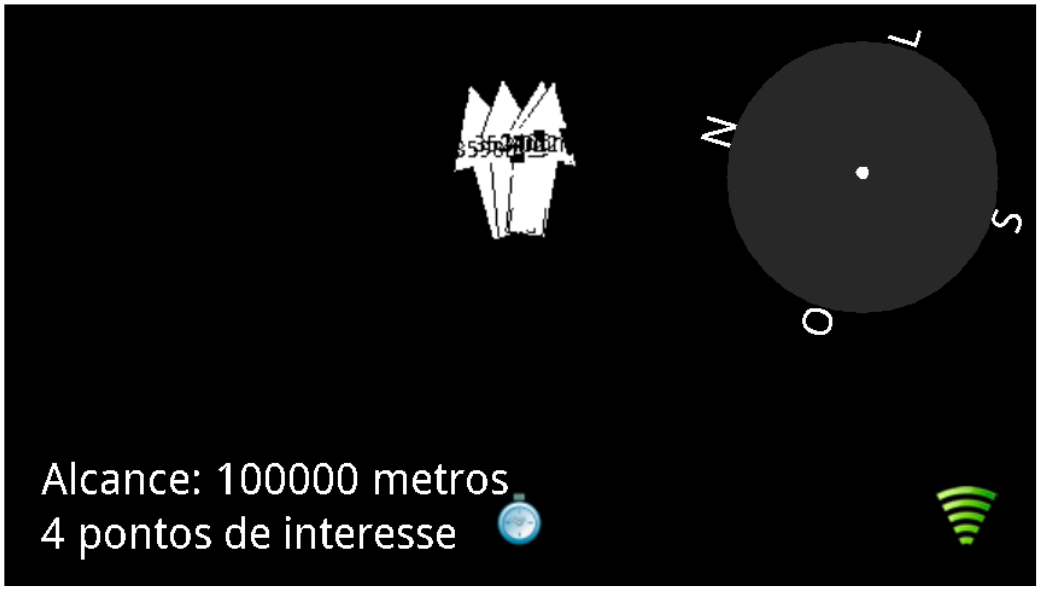


Figura 45 – Tela com Realidade aumentada

Outro comportamento da aplicação quando em uso em ambiente aberto, é o fato de receber notificações. Quando o servidor detecta que o usuário está próximo a um evento, em um raio de 1 km, o servidor envia uma notificação para o dispositivo. Essa notificação pode ser visualizada pelo usuário, através da área de notificações. Ao clicar na barra de notificações ele visualizará qual o evento que ele está próximo e pode decidir se ele irá ou não no evento. Na Figura 46 é mostrada a notificação recebida pelo dispositivo do usuário próximo a um evento. Também é enviada uma notificação para o dispositivo, quando o aluno recebe alguma atualização referente à freqüência ou nota de alguma disciplina que ele esteja cadastrado.

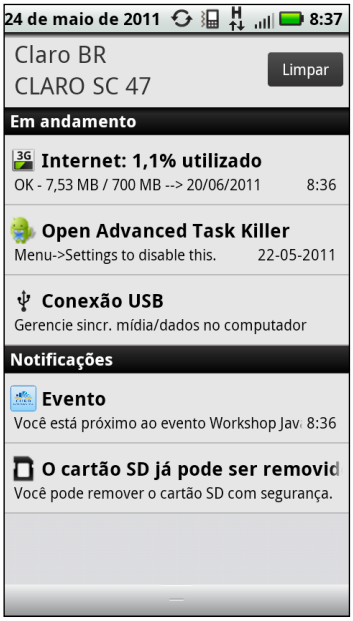


Figura 46 – Área de notificações do Android

Para o professor foi criada uma ferramenta *web* onde o mesmo pode acessar para cadastrar os eventos da universidade e também para publicar notas e freqüências dos alunos.

Para acesso ao sistema *web*, assim como no dispositivo, é necessário fazer a autenticação. Na Figura 47 é demonstrada a tela de *login*.

Figura 47 – Tela login

Após autenticar-se o professor pode utilizar as funcionalidades através do menu *web*. Caso o professor necessite publicar as notas e/ou freqüência de membros da disciplina, ele

pode fazer através do acesso ao menu `Disciplinas`, escolher a disciplina que está lecionando clicando no botão `Acessar` em seguida são carregados os membros da disciplina. Esse fluxo de telas é representado pela Figura 48.

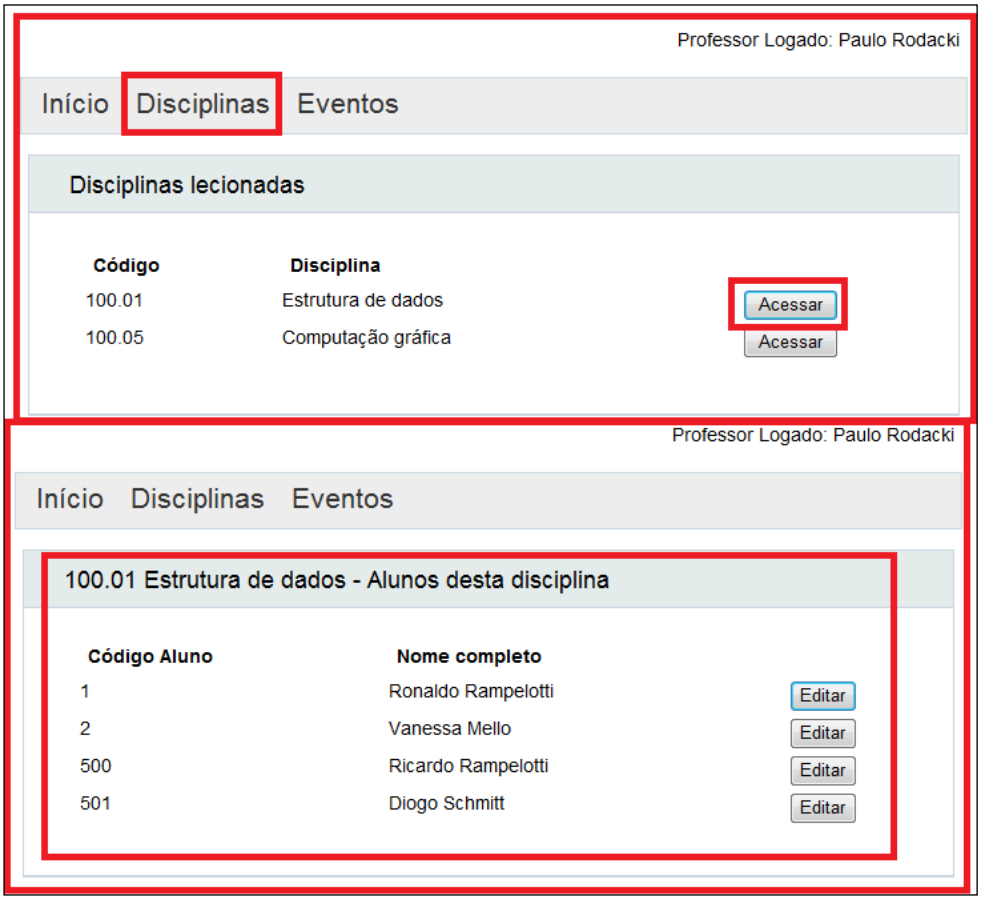


Figura 48 – Fluxo de telas site disciplinas aluno

Com os membros da disciplina carregados, é possível publicar notas e/ou freqüência clicando no botão `Editar`. Nesse momento é exibida a tela de publicação de dados para o aluno. Entrando com a nota e/ou a freqüência e clicando em `Publicar` as informações são persistidas no servidor SOA e estas são enviadas para o dispositivo do aluno através de notificação. Na Figura 49 é demonstrado esse fluxo de telas.

Figura 49 – Fluxo da tela de alunos para publicação de notas e frequência

Por fim o professor também pode cadastrar eventos e alterar existentes, através do menu de acesso `Eventos`. O professor terá acesso a uma tela de eventos carregada com uma lista de eventos previamente cadastrados e dando a possibilidade para a inserção de novos eventos. Na Figura 50 é demonstrada a tela de eventos.

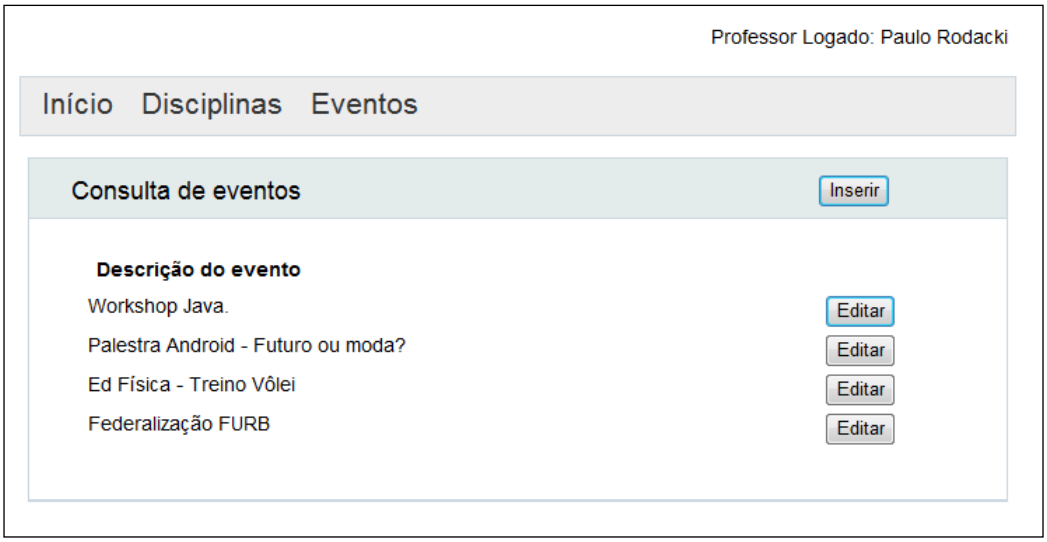


Figura 50 – Tela de eventos

Caso o professor queira adicionar um evento, ele precisa clicar no botão `Inserir`. Ao clicar nesse botão, é exibida a tela mostrada na Figura 49. Caso ele queira alterar um evento existente, precisa somente clicar no botão `Editar` do respectivo evento a ser alterado, dando acesso à tela de alteração, que também é mostrada na Figura 51.

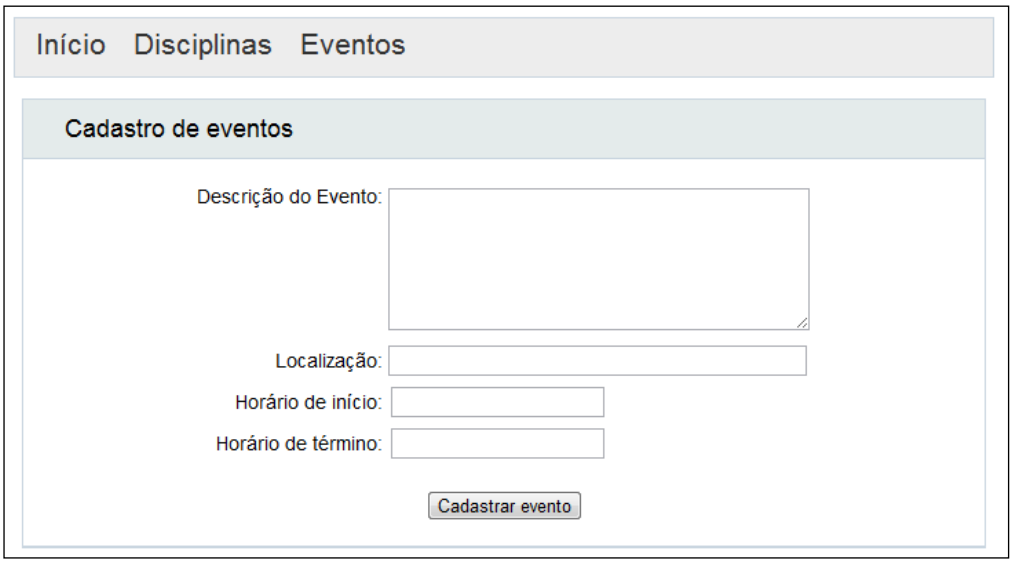


Figura 51 – Tela de cadastro e alteração de eventos

3.4 RESULTADOS E DISCUSSÃO

O presente trabalho apresentou o desenvolvimento de uma aplicação para utilização de serviços baseados em localização e notificação *push* em Android. Permitindo através do dispositivo, consultar eventos, disciplinas, membros de disciplina e freqüência na disciplina. Ainda é possível interagir com membros da disciplina através de e-mail. Também foi desenvolvida uma aplicação *web* para administração de eventos, notas e freqüências nas disciplinas pelos professores.

A ferramenta criada foi aplicada em cenário universitário, e também foram explorados trabalhos correlatos que conferem o propósito de auxiliar na universidade. Com base nisso, no Quadro 13 é feita uma comparação da ferramenta criada, com os trabalhos correlatos.

Funcionalidades / Características	Ferramenta criada	MIT Mobile	Harvard Mobile	iStanford	PUCRS Mobi
visualizar membros disciplina	sim	não	não	não	não
mapa com eventos universidade	sim	sim	sim	sim	sim
envio de e-mail para membros disciplina	sim	não	não	não	não
notificações LBS	sim	não	não	não	não
realidade aumentada	sim	não	não	não	sim

Quadro 13 – Comparação com trabalhos correlatos

Para análise das outras ferramentas foram utilizadas as funcionalidades que a

ferramenta construída possui. Logo foi detectado que funcionalidades como mapa com eventos existe em todas as soluções, porém somente uma solução atende a realidade aumentada, que é a solução da PUC, fora isso o maior diferencial seria os serviços baseados em localização. Estas ferramentas têm grandes diferenciais não aplicados na ferramenta construída, por se tratar em um estudo sobre os serviços baseados em localização e não a construção de uma solução completa com nível de detalhamento de informações como as soluções correlatas.

Ainda foram empregados alguns testes na ferramenta para demonstrar a velocidade de comunicação com o servidor SOA. Para estes testes foram realizadas consultas de membros da disciplina, onde possui o maior tráfego de informação do servidor para o dispositivo. Na Figura 52 foram demonstrados números respectivos ao tempo de consulta da aplicação e envio para dispositivo.

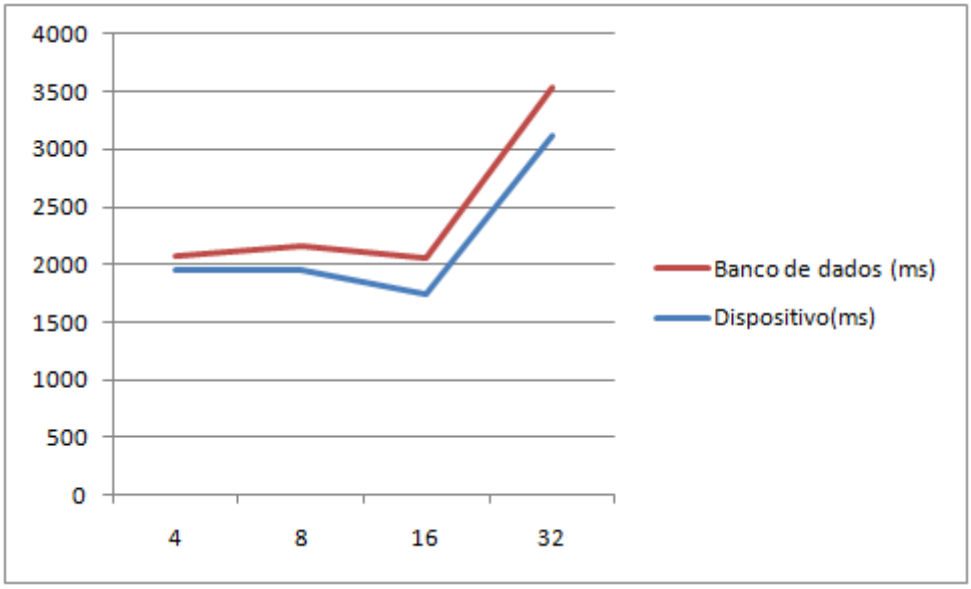


Figura 52 – Representação de tempo de requisições

Este trabalho também teve por desafio a notificação através de *push* para dispositivos. Sendo limitado o uso de *push*, somente para aparelhos que possuíssem a versão 2.2 do Android. Chegou a ser estudada uma solução alternativa através de *sockets*, rodando em um servidor paralelo, onde poderia ser iniciado um canal de comunicação e controlado via dados persistidos por um container *web*.

Outra dificuldade encontrada no trabalho, foi a exposição de serviços REST com o uso da biblioteca .Net. Chegou a ser cogitada a idéia de criar uma plataforma heterogênea. Porém, a falta de conhecimento da *Windows Communication Framework* (WCF), acabou inviabilizando a criação da mesma. Foi optado o uso do servidor em JAVA Apache Tomcat, que demonstrou muita praticidade e aplicabilidade no cenário em questão da aplicação.

Um problema que foi constatado logo no inicio do desenvolvimento no dispositivo móvel, foi a utilização de imagens em alta resolução, para deixar a aplicação com uma interface mais rica. Porém, com a utilização mais freqüente da ferramenta, a aplicação acabava sinalizando vazamento de memória. Como solução paliativa, foram deixadas de usar as imagens de alta resolução e passou-se a usar imagens de baixa resolução.

O maior e último dos problemas encontrados foi com relação ao servidor SOA. Este servidor disponibiliza os serviços utilizando a arquitetura REST. Foram criadas as chamadas e disponibilizadas tanto em métodos HTTP do tipo GET, que servem para captura de informação. Também foram criados métodos do tipo POST, que servem para inserção de dados.

O servidor foi exposto para a internet, para poder fazer uso tanto pelo aparelho, quanto pelo site, sem que seja necessário informar qual o endereço IP que o servidor possui, isso foi possível através da solução NO-IP. As consultas para os serviços GET, funcionam sem problemas, mas as inserções com métodos POST não tinham o mesmo resultado, gerando um erro de que o método não era permitido.

Depois de muitos testes com o servidor, foi constatado o erro através de uma ferramenta adicional do Mozilla Firefox chamada POSTER. Essa ferramenta por sua vez, mostra toda a rota de uma requisição e o formato da requisição. Quando passava a requisição pelo servidor do NO-IP, a requisição que ia com POST virava GET fazendo assim com que o servidor SOA retornasse um `método não permitido`. Para solução está sendo utilizado o endereço IP que o servidor recebe da internet no momento da conexão.

4 CONCLUSÕES

Este trabalho apresentou uma aplicação com uso de sistemas baseados em localização, fornecendo informações sobre eventos próximos. Fazendo com que não seja mais necessário a consulta com intenção de descobrir se existe algo próximo. Mas sim a informação de que existe indo de encontro ao usuário. Também são fornecidos para o usuário da aplicação no dispositivo, notificações sobre alterações de notas e freqüências, eliminando assim a consulta desnecessária do usuário e permitindo quase que em tempo real que a informação chegue ao interessado de maneira rápida. A aplicação conta também com um site administrativo que pode ser operado pelos professores, com a intenção de adicionar eventos para alunos da FURB e permitindo que os alunos possam acompanhar quais eventos a FURB está disponibilizando. O professor pode também publicar a freqüência e a nota do aluno na disciplina.

O presente trabalho demonstra também a utilização da plataforma Android como cliente de uma arquitetura cliente-servidor. Para isso foi utilizado um servidor SOA que faz a comunicação com o dispositivo, disponibilizando informações persistidas no banco. A comunicação foi toda baseada em requisições HTTP solicitando serviços em REST. Sendo elaborada essa arquitetura por se adequar as necessidades de um dispositivo móvel, levando em conta o menor tráfego de dados possível, trafegando menos dados do que por uma requisição SOAP. Além da adoção da arquitetura REST foi associado o uso de JSON como forma de troca de objetos pela rede, compactando muito os dados trafegados.

A adoção do servidor utilizando SOA, permitiu centralizar os esforços de maneira mais rápida e focada nas funcionalidades que a ferramenta por fim realiza. Com essa arquitetura é possível reutilizar o servidor, assim como foi feito utilizando, tanto pelo dispositivo quanto pelo site.

Com esse trabalho observou-se que a utilização de REST, trouxe um ganho significativo na produção, simplificando a construção de serviços.

Por fim a construção da solução foi facilitada pelas SDK disponíveis tanto na plataforma Android, quanto no servidor. Pela plataforma Android diversas classes auxiliaram na construção da ferramenta, tanto para utilização de mapas, quanto para notificações. Na plataforma *web* a utilização de REST com API Jersey facilitou em todo desenvolvimento de serviços.

4.1 EXTENSÕES

Como sugestões de extensões para a continuidade do presente trabalho, tem-se:

a) adicionar funcionalidade para publicação de mídia georeferenciada;
b) adicionar funcionalidade para captura de posição com triangulação de rede Wi-Fi obtendo a posição do aluno dentro de um ambiente fechado;
c) adicionar a possibilidade de comunicação entre usuário através de notificações;
d) adicionar mais serviços baseados em localização, podendo associar com mídias publicadas com georeferência;
e) adicionar funcionalidade de leitura de QRCode para acesso a informações em ambiente fechado;
f) explorar o uso de mapa alternativo, outra fonte de mapas como um desenho da própria universidade em cima do mapa Google.

REFERÊNCIAS BIBLIOGRÁFICAS

ANDROIDMEUP. **Android 2.2 cloud to device messaging.** [S.1.], [2010]. Disponível em: <http://www.androidmeup.com/articles/android-22-froyo-cloud-to-device-messaging-c2dm>. Acesso em: 18 fev. 2011.

AUTOMAILS. **Realidade aumentada.** [S.1.], [2010]. Disponível em: <http://automails.com/2010/08/realidade-aumentada/>. Acesso em: 20 fev. 2011.

AZUMA, Ronald T. A survey of augmented reality. **Presence**: Teleoperators And Virtual Environments, [S.l.], v. 6, n. 4, p. 355-385, Aug. 1997. Disponível em: <http://www.cs.unc.edu/~azuma/ARpresence.pdf>. Acesso em: 20 mar. 2010.

BAGUETI. **PUCRS lança app para iPhone e iPod.** [S.1.], [2011]. Disponível em: <http://www.baguete.com.br/noticias/telecom/18/01/2011/puc-rs-lanca-app-para-iphone-e-ipod>. Acesso em: 01 jun. 2011.

BLACKBERRY. **BlackBerry push service.** [S.1.], [2011?]. Disponível em: <http://us.blackberry.com/developers/platform/pushapi.jsp#tab_tab_works>. Acesso em: 18 fev. 2011.

CARBONARO, Tato. **LBS:** o que são os location based services? [S.1.], [2009]. Disponível em: <http://viajeaqui.abril.com.br/navegadorg4r/blog/lbs-sao-location-based-services-175340_comentarios.shtml?7725268>. Acesso em: 19 fev. 2011.

CODE. **Android cloud to device messaging framework.** [S.1.], [2011?]. Disponível em: <http://code.google.com/intl/pt-BR/android/c2dm/>. Acesso em: 18 fev. 2011.

CONCEIÇÃO, David T. **Framework para gerenciamento e disponibilização de informações multimídia geolocalizadas na plataforma android.** 2010. 102 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

EMIDIO, Jonatas. **Android cloud to device messaging.** [S.1.], [2010]. Disponível em: <http://androidpack.blogspot.com/2010/09/c2dm-android-cloud-to-device-messaging.html>. Acesso em: 19 fev. 2011.

FAZION. **Soluções móveis em ambientes corporativos**. [S.1.], 2011. Disponível em: <http://www.fazion.com.br/uploaded/File/Solucoes%20moveis%20em%20ambientes%20corporativos.pdf>. Acesso em: 19 fev. 2011.

ITUNES. **iStanford.** [S.1.], [2010a]. Disponível em: <http://itunes.apple.com/us/app/istanford/id292922029?mt=8#>. Acesso em: 20 fev. 2011.

ITUNES. **Mit mobile.** [S.l.], [2010b]. Disponível em: <http://itunes.apple.com/br/app/mit-mobile/id353590319?mt=8#>. Acesso em: 20 fev. 2011.

_____. **Harvard mobile.** [S.l.], [2010c]. Disponível em: <http://itunes.apple.com/us/app/harvard-mobile/id389199460?mt=8>. Acesso em: 20 fev. 2011.

_____. **PUCRS Mobi.** [S.l.], [2010d]. Disponível em: <http://itunes.apple.com/sn/app/pucrsmobi/id403577752?mt=8#>. Acesso em: 01 jun. 2011.

LANE, Dale. **Push notifications for mobile.** [S.1.], [2009]. Disponível em: <http://dalelane.co.uk/blog/?p=938>. Acesso em: 18 fev. 2011.

MAHMOUD, Qusay H. **J2ME and location-based services.** [S.1], [2004]. Disponível em: <http://developers.sun.com/mobility/apis/articles/location/>. Acesso em: 19 fev. 2011.

MOTOROLA. **DEFY - smartphone**. [S.l.], [2011]. Disponível em: <http://www.motorola.com/Consumers/BR-PT/Consumer-Product-Services/Mobile-Phones/ci.MOTOROLA-DEFY-BR-PT.alt>. Acesso em: 17 maio 2011.

NERDDANET. **Realidade aumentada.** [S.1.], [2010]. Disponível em: <http://nerddanet.blogspot.com/2010/07/realidade-aumentada.html>. Acesso em: 20 fev. 2011.

NUNES, Sergio; DAVID, Gabriel. Uma arquitetura web para serviços web. In: XATA 2005 – XML: APLICAÇÕES E TECNOLOGIAS ASSOCIADAS. 2005. Braga, **Actas ...** Braga: Universidade do Minho, Universidade do Porto, 2005. p.205-215. Disponível em: <https://www.fe.up.pt/si/publs_pesquisa.formview?p_id=12085>. Acesso em: 19 dez. 2011.

STEINEGER, Stefan; NEUN, Moritz; EDWARDES, Alistair. **Foundation of location based services**. [S.1.], [2011?]. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.94.1844>. Acesso em: 20 fev. 2011.

VASSELAI, Gabriela T. **Um estudo sobre realidade aumentada para plataforma Android.** 2010. 103 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

WIKI. **REST.** [S.1.], [2011]. Disponível em: <http://pt.wikipedia.org/wiki/REST>. Acesso em: 20 fev. 2011.