UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

SISTEMA MÓVEL NA PLATAFORMA PHONEGAP PARA COMPARTILHAMENTO DE GEOLOCALIZAÇÃO INTEGRADO A REDE SOCIAL

GABRIEL FELIPE BORGES DE CAMPOS

BLUMENAU
2015

2015/2-10

GABRIEL FELIPE BORGES DE CAMPOS

SISTEMA MÓVEL NA PLATAFORMA PHONEGAP PARA COMPARTILHAMENTO DE GEOLOCALIZAÇÃO INTEGRADO A REDE SOCIAL

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Ciência da Computação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Prof. Dalton Solano dos Reis, Mestre - Orientador

BLUMENAU
2015

2015/2-10

SISTEMA MÓVEL NA PLATAFORMA PHONEGAP PARA COMPARTILHAMENTO DE GEOLOCALIZAÇÃO INTEGRADO A REDE SOCIAL

Por

GABRIEL FELIPE BORGES DE CAMPOS

Trabalho de Conclusão de Curso 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, Mestre – Orientador, FURB
Membro:	Prof. Aurélio Faustino Hoppe, Mestre – FURB
Membro:	Prof. Joyce Martins, Mestre – FURB

Blumenau, 10 de dezembro de 2015

Dedico este trabalho para todas as pessoas que dedicaram tempo a compartilhar seus conhecimentos comigo.

AGRADECIMENTOS

Agradeço aos meus pais Jaime e Carmes, meus maiores exemplos de dedicação, pelo amor, incentivo e apoio incondicional.

Aos meus amigos, companheiros de trabalho, pela atenção e estimulo.

A empresa WEG S.A., por disponibilizar as ferramentas necessárias para a realização deste trabalho.

A esta universidade e todos os educadores do curso, que foram de extrema importância no desenvolvimento de minha carreira profissional.

Ao meu orientador Dalton, pelo empenho e suporte dedicado à elaboração deste trabalho.

A todos que direta ou indiretamente fizeram parte da minha jornada acadêmica e profissional.

Depois de escalar um grande morro, descobrimos apenas que há muitos outros a escalar.

Nelson Mandela.

RESUMO

Este trabalho apresenta o desenvolvimento de um aplicativo multiplataforma, que permite o compartilhamento de localização entre amigos e facilita a locomoção dos visitantes do evento Interação FURB pelo campus da universidade. O aplicativo está integrado à rede social Facebook, possibilitando que os visitantes entrem no aplicativo utilizando as informações de seu perfil e enviem convites de compartilhamento de localização para suas listas de amigos. Através do Google Maps, o aplicativo carrega um mapa virtual e apresenta a posição atual do usuário e de seus amigos dinamicamente. O gerenciamento das coordenadas geográficas dos visitantes é realizado pelo servidor de localizações, permitindo a leitura e atualização destas informações via *webservices*. Este aplicativo também foi integrado ao servidor de notificações AeroGear, proporcionando o recebimento de alertas independente do sistema operacional utilizado pelo usuário. Desenvolvido com o *framework* PhoneGap e seus *plug-ins*, o aplicativo utiliza as tecnologias *web* (HTML, CSS e Javascript) como padrão de desenvolvimento. Como resultado da implementação, o aplicativo executou muito bem na plataforma iOS, permitindo aos usuários integrar sua conta do Facebook, realizar o compartilhamento de localização entre os amigos desta rede social através do Google Maps e receber notificações de novos eventos. Porém o mesmo sucesso não pôde ser obtido com a plataforma Android. Quando gerado o projeto para esta plataforma, ocorreram conflitos de dependências impedido que o mesmo fosse testado em dispositivos Android.

Palavras-chave: PhoneGap. Dispositivo móvel. Rede social. Notificações. GPS. Compartilhamento de localização.

ABSTRACT

This work presents the development of a multi-platform application, which allows location sharing between friends and make the locomotion of event Interação FURB visitors easier within the campus university. The app integrates with the Facebook social networking enabling visitors connect the app using Facebook profile and send location sharing invitations to his friends list. Through Google Maps, the app loads up a virtual map and indicates the user and his friends current positions dynamically. The locations server made the management of visitors geographical coordinates, allowing by webservices reading and updating of these informations. This app also integrates with the AeroGear notifications server, receiving alerts independent of the operational system used by the user. Developed with PhoneGap framework and its plug-ins, the app uses the web common technologies HTML, CSS and Javascript with development tools. The result of the implementation was the success of the app at iOS platform, allowing users to integrate your Facebook account, perform location sharing between friends of this social network using Google Maps and receive notifications of new events. But the same could not be obtained in Android platform. When generated the project to this platform, occurred dependency conflicts, blocking it from being tested in Android platform.

Key-words: PhoneGap. Mobile device. Social networking. Notifications. GPS. Sharing location.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

AGPS – *Assisted Global Positioning System*

API – *Application Programming Interface*

APNS – *Apple Push Notification Service*

CSS – *Cascading Style Sheets*

EA – Enterprise Architect

FURB – Universidade Regional de Blumenau

GCM – *Google Cloud Messaging*

GPS – *Global Positioning System*

HTML – HyperText Markup Language

MPN – *Microsoft Push Notification*

REST – *Representational State Transfer*

RF – Requisitos Funcionais

RNF – Requisitos Não Funcionais

SDK – *Software Development Kit*

UC – *Use Case*

SUMÁRIO

1 INTRODUÇÃO

Atualmente a tecnologia móvel traz transformações ao comportamento das sociedades no qual ela está inserida. A todo momento, é possível perceber a mudança cultural que ocorre no mundo, através das ações humanas, dos textos que são escritos, das conversas entre amigos e dos relacionamentos sociais (SABOIA; VARGAS; VIVA, 2013). Para Soares (2012), "[...] as novas tecnologias móveis apontam para um futuro promissor: a mobilidade e com ela os aplicativos personalizados estarão cada vez mais revolucionando a sociedade e sua vida cotidiana". Tal entendimento é ratificado por Alcantara e Vieira (2011), que afirmam:

> A cada dia, um número maior de pessoas interessa-se pela mobilidade, o fácil acesso às informações em qualquer lugar, com alcance amplo a qualquer hora, se conectando de forma fácil e rápida a outros dispositivos móveis, localizando pessoas, produtos e serviços personalizados (ALCANTARA; VIEIRA, 2011, p. 2).

Os dispositivos móveis permitem que as pessoas estejam mais próximas virtualmente umas das outras, mantendo-as conectadas e substituindo a presença física. De acordo com pensamento de Calazans (2012), um fator importante para a expansão do uso da internet em dispositivos móveis, são os meios de comunicação como as redes sociais. Para Kirkpratick (2011, p. 96), "[...] as redes sociais estendem-se por todo o planeta. O Facebook é a maior delas. É raro um estudante do ensino médio ou um universitário que não use rotineiramente o Facebook [...]".

Segundo Alcantara e Vieira (2011), outro serviço amplamente utilizado e que ganhou popularidade com o auxílio dos dispositivos móveis, é o sistema de localização Global Positioning System (GPS). Existem várias empresas que disponibilizam este serviço de forma gratuita, indicando às pessoas de sua posição em tempo real, detalhando ruas, restaurantes, hotéis, pontos turísticos, hospitais e entre outras informações. A evolução do GPS permitiu que novas formas de utilização fossem criadas e não se restringindo apenas a localização pessoal. Atualmente é possível que pessoas compartilhem entre si suas localizações, permitindo que os pais saibam onde seus filhos estão ou que familiares encontrem seus entes em casos de emergência como por exemplo, um sequestro (ALCANTARA; VIEIRA, 2011, p. 5).

No meio a diversas possibilidades de utilização que a mobilidade permite, alguns novos desafios sociais surgem e necessitam da atenção. Conforme jornal O Dia (2012), os *smartphones* estão entre os presentes mais requisitados nas comemorações de final de ano, fazendo com que 18% dos jovens adultos brasileiros possuam problemas de vício com seus aparelhos. Enquanto outros 35% verificam seus dispositivos pelo menos a cada dez minutos. Desta forma, analisando o cenário universitário onde a maioria dos estudantes são jovens,

surge a necessidade de as instituições adotarem a tecnologia móvel como parte de seu cotidiano e buscar novos meios de atrair a atenção dos estudantes.

A Universidade Regional de Blumenau (FURB), ao longo de vários anos, realiza o evento chamado Interação FURB. Este tem como principal objetivo criar a oportunidade para alunos do Ensino Médio conhecer a estrutura da Universidade, participando de oficinas e atividades organizadas pelos próprios cursos de graduação. Estas atividades são elaboradas de forma que os alunos possam vivenciar a área de atuação de que possui interesse, tirar dúvidas com profissionais e acadêmicos da área e decidir sobre seu futuro acadêmico (UNIVERSIDADE REGIONAL DE BLUMENAU, 2015). A realização deste evento exige organização por parte da Universidade para receber e atender as expectativas dos estudantes da região. As atividades e oficinas oferecidas exigem espaços físicos como tendas, salas de aulas, auditórios e laboratórios. Como mostra a Figura 1, é necessário que sinalizações sejam expostas pelos corredores e que guias sejam disponibilizados para auxiliar os visitantes a se locomoverem entre os blocos do campus.

Figura 1 – Sinalização dos blocos na FURB

Alinhando a necessidade que as universidades possuem de adaptação com as tecnologias móveis para atrair a atenção dos jovens universitários, a disponibilidade de serviços gratuitos de localização, as redes sociais e o fato de que praticamente todos possuem um *smartphone*, viu-se interessante o desenvolvimento de um aplicativo para estes dispositivos no qual pudesse ser utilizado principalmente no evento Interação FURB. No caso

da Universidade Regional de Blumenau, que realiza este evento anualmente, incentivar os estudantes a explorar e interagir com universo acadêmico, auxiliando-os a se locomoverem dentro do campus, despertaria interesse por este ambiente. O compartilhamento de localização entre amigos, o recebimento de notificações de eventos acadêmicos e a disponibilização de informações de blocos, andares, salas de aula e locais públicos (biblioteca, cantina, central de impressão) também contribuiriam para a ambientação dos visitantes e novos alunos. Desta forma, as funcionalidades de compartilhamento de localização, integração aos mapas do Google Maps e o envio de notificações do trabalho desenvolvido por Kestring (2014), foram utilizadas como escopo deste trabalho.

1.1 OBJETIVOS

Este trabalho tem como objetivo disponibilizar o trabalho desenvolvido por Kestring (2014) na plataforma PhoneGap para utilização no evento Interação FURB.

Os objetivos específicos do trabalho são:

a) desenvolver um aplicativo multiplataforma que permita o compartilhamento de localização entre amigos;
b) permitir o uso da rede social Facebook para autenticação e busca de amigos no aplicativo;
c) possibilitar o recebimento de notificações e informativos emitidos pela universidade.

1.2 ESTRUTURA

Para melhor organização e entendimento do leitor, este trabalho está separado em quatro capítulos. Iniciando o primeiro capítulo com a introdução ao tema do trabalho, seguido do segundo capítulo no qual estão descritos os fundamentos básico para o entendimento das técnicas e ferramentas utilizadas. Ainda neste capítulo, são apresentados os trabalhos correlatos ao desenvolvido. O terceiro capítulo apresenta informações sobre a construção do aplicativo como diagramas, arquitetura, comunicação entre cliente e servidores e o código fonte. As últimas seções, trazem as telas do aplicativo demonstrando sua operacionalidade, os resultados obtidos após o desenvolvimento e comparação entre os trabalhos correlatos. No quarto e último capítulo estão as conclusões finais e sugestões para o desenvolvimento de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são abordados alguns assuntos relevantes para entendimento aprofundado das tecnologias utilizadas no desenvolvimento do aplicativo. Na seção 2.1 são descritos conceitos relacionados às tecnologias GPS, AGPS e Google Indoor Maps. Na seção 2.2 são apresentados conceitos sobre notificações em dispositivos móveis utilizando o serviço unificado AeroGear. Em 2.3 é apresentada a plataforma PhoneGap e seus *plug-ins* de interação com os *frameworks* do Google Maps, Facebook e *Push Notifications*. Para finalizar, a seção 2.4 demonstra os trabalhos que possuem relação ao aplicativo desenvolvido.

2.1 GPS, AGPS E GOOGLE INDOOR MAPS

Desde muitos anos atrás, o homem teve curiosidade e necessidade de saber onde estava. Este foi um dos primeiros desafios científicos em que o ser humano se preocupou em solucionar, pois devido as explorações marítimas e terrestres, era primordial que os exploradores soubessem ir e voltar de um determinado local a outro de forma segura (MONICO, 2000, p. 19). A navegação e o posicionamento geográfico, por muito tempo, basearam-se na orientação do sol, dos planetas e das estrelas e isso exigia habilidade dos navegadores (MONICO, 2000, p. 19). As condições climáticas poderiam a todo instante influenciar os pontos de referência, podendo levar a expedição ao fracasso. Ao passar dos anos, com a avanço tecnológico na eletrônica (ondas de rádios) e muito estudo científico, foram desenvolvidos vários instrumentos de medição de posicionamento e coordenadas. Porém eles apresentavam grandes margens de erro em relação ao posicionamento real e continuava a necessidade de uma solução que oferecesse boa precisão (MONICO, 2000, p. 19).

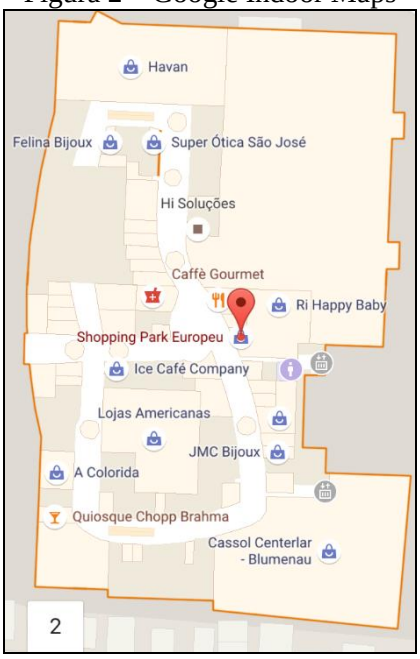
Para solucionar os problemas relacionados a localização geográfica, em 1970 surgiu a proposta do Sistema de Posicionamento Global (*Global Position System*, GPS). O funcionamento básico de navegação deste sistema consiste no cálculo da distância entre um ponto terrestre e quatro satélites apropriadamente distribuídos em um sistema de referência. Conhecendo as coordenadas dos satélites, é possível encontrar as coordenadas da antena do ponto terrestre sem que as condições adversas do clima atrapalhem a localização (MONICO, 2000, p. 21). O GPS vem auxiliando diversas atividades do cotidiano e que exigem posicionamento, pois possui abrangência global e permite que qualquer ponto da superfície terrestre seja localizado com precisão (MONICO, 2000, p. 21). Esta tecnologia fornece dados precisos quando utilizado em ambientes externos, como por exemplo estradas e rodovias. Em ambientes fechados, como *shopping centers*, museus e restaurante, o sinal do GPS acaba

sendo enfraquecido devido as estruturas metálicas pelas quais são construídos. A qualidade do sinal diminui e afeta diretamente na precisão da localização devido as interferências causados por este material (TEIXEIRA, 2014).

Hoje, facilmente encontra-se equipamentos eletrônicos que possuem antena de GPS dando suporte à esta tecnologia. Os mais comuns são os *smartphones*, que possuem vantagem em relação ao tempo de cálculo de localização por também suportarem a tecnologia *Assisted Global Positioning System* (AGPS). O AGPS foi desenvolvido para acelerar o processo de localização, fazendo uso da rede de telefonia celular para receber informações dos satélites que estão mais próximos da antena de GPS. Desta forma, evitam que todos os satélites sejam procurados pelo aparelho, aprimorando o tempo de localização (MICROSOFT, 2015). Porém a tecnologia de geolocalização fornece apenas dados de posicionamento, sendo necessário a utilização de um *software* que mostre às pessoas sua localização em relação a um mapa.

O Google Maps é uma destas ferramentas que auxiliam na localização das pessoas através de um mapa virtual. Um ponto de referência no mapa indica a posição em que ela se encontra, permitindo a navegação pelos caminhos e estradas disponíveis. Esta ferramenta fornece diversas informações de trânsito, hotéis e locais públicos para facilitar a orientação das pessoas nos lugares desconhecidos por elas. Um recurso recente disponibilizado nesta ferramenta é o chamado Google Indoor Maps. Ele permite a visualização do interior dos prédios que são considerados públicos e onde há grande circulação de pessoas. No exemplo da Figura 2, é possível ver o interior do *shopping center* Park Europeu em Blumenau.

Figura 2 – Google Indoor Maps



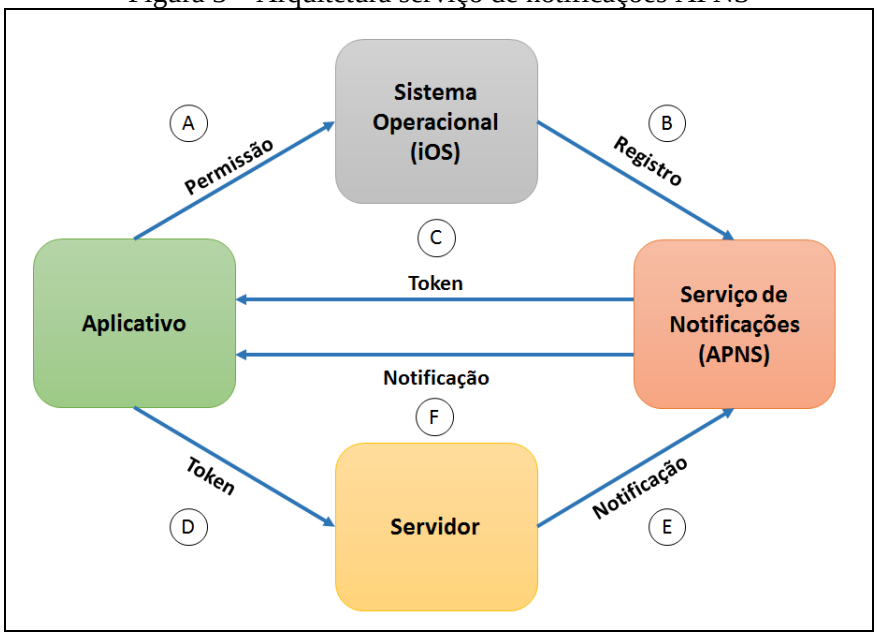
A vantagem de utilizar este tipo de visualização está na facilidade com que as pessoas podem encontrar os locais dentro de prédios e construções. No caso de um *shopping center* é possível encontrar lojas, restaurantes, banheiros e escadas rolantes. Já em uma universidade estariam disponíveis as salas de aulas, bibliotecas, auditórios, cantinas e todos os demais ambientes. Para que as construções fiquem disponíveis publicamente é necessário que cada entidade envie suas plantas baixas para o site do Google Indoor Maps. O conteúdo enviado será analisado pela equipe da Google e caso atenda a todos os requisitos legais, será publicado no Google Maps.

2.2 PUSH NOTIFICATIONS E O SERVIÇO UNIFICADO AEROGEAR

As notificações *push* são os alertas enviados pelos aplicativos para os dispositivos móveis. Estes alertas podem avisar ao usuário de que alguma novidade está disponível para ser visualizada, ou seja, uma nova mensagem, um novo convite ou novos comentários em suas publicações (LINKEDIN, 2014). Os usuários podem receber as mensagens de três modos diferentes: exibição de textos curtos, alertas de som, adição de número (badge) no ícone da aplicação, podendo ainda combinar entre elas. Segundo Tecmundo (2013), as notificações podem tornar o consumo de energia e memória dos dispositivos mais eficiente, pois são envidas apenas quando uma nova atualização está disponível e sem a necessidade do aplicativo estar aberto ou executando em segundo plano.

O envio de notificações é oferecido na forma de serviço e funciona de maneira semelhante para as plataformas iOS, Android e Windows Phone. Porém, cada plataforma possui seu próprio serviço de notificações e exigi configurações diferentes para os aplicativos que fazem uso deste recurso. Respectivamente, os serviços de notificações são: Apple Push Notification Sevice (APNS), Google Cloud Messaging (GCM) e Microsoft Push Notifications (MPN). A Figura 3 demonstra o processo para habilitar o aplicativo a receber notificações do serviço APNS.

Figura 3 – Arquitetura serviço de notificações APNS



Geralmente, as notificações são disparadas a partir do servidor que faz o gerenciamento do aplicativo cliente. Quando determinada ação ocorre neste servidor, uma notificação é envia ao serviço de notificações correspondente. Estas etapas são explicadas abaixo:

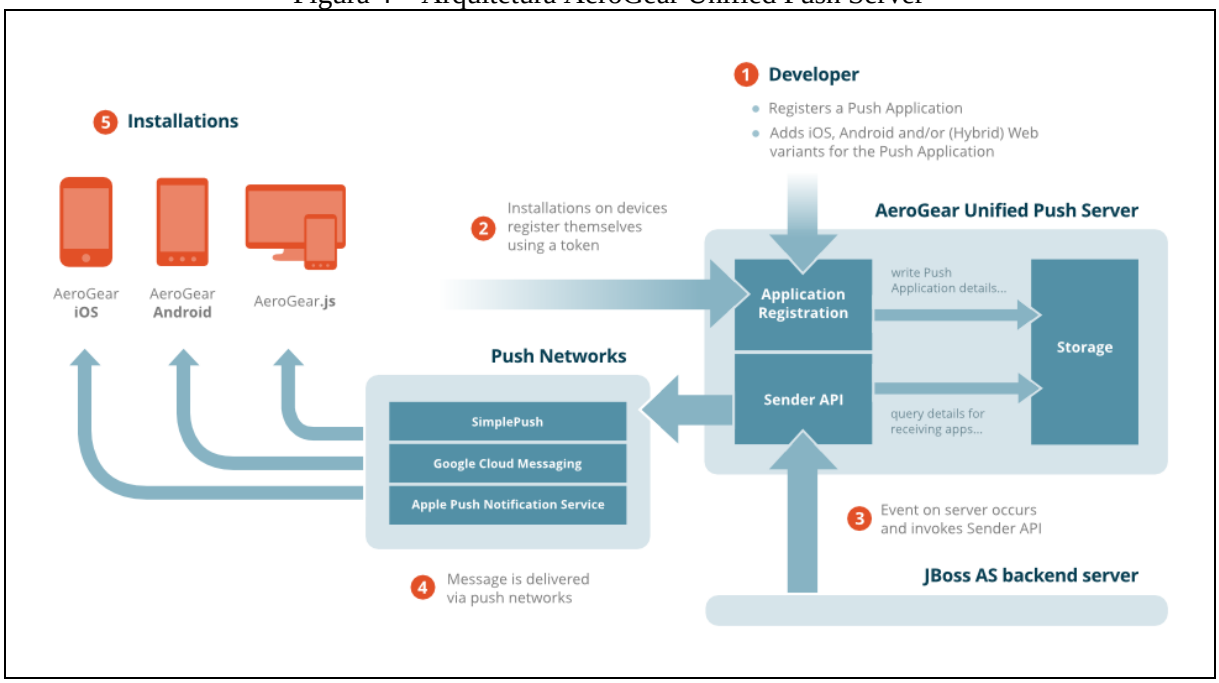
a) o aplicativo deve estar registrado com o certificado de provisionamento Apple e com permissão para receber notificações;
b) o sistema operacional solicita ao servidor APNS o *token* para o dispositivo;
c) o aplicativo recebe o *token* que será utilizado para identificar o dispositivo e a aplicação do usuário;
d) o aplicativo envia o *token* recebido para o servidor que o gerencia;
e) o servidor armazenado o *token* e quando algo interessante acontece, envia a notificação para o APNS com o *token* armazenado;
f) o serviço APNS repassa a notificação para o aplicativo do usuário.

Conforme explicação das etapas, é possível perceber que os serviços de notificações são responsáveis por realizar a intermediação entre o servidor da aplicação e o próprio aplicativo, ou seja, eles são responsáveis por transmitir as notificações para os dispositivos dos usuários. No caso do APNS não é garantido que a transmissão da mensagem seja concluída com sucesso e isso implica na confiabilidade do serviço, inviabilizando seu uso para envio de alertas críticos (GRONER, 2013).

Para realizar a comunicação e o envio das notificações do servidor da aplicação para o serviço de notificações, existem ferramentas e bibliotecas que podem facilitar esta integração.

O AeroGear Unified Push Server é uma destas ferramentas que permite o envio de notificações nativas para diferentes sistemas operacionais móveis, suportando serviços como APNS, GCM e MPN, entre outros (BLOG OPENSHIFT, 2013). A Figura 4 apresenta a arquitetura do AeroGear e pode ser comparada com a Figura 3. Neste caso, após receber o *token* do serviço de notificações, o aplicativo passa a enviá-lo para o AeroGear. O AeroGear identifica o sistema operacional do usuário para saber qual dos serviços de notificações deve ser utilizado posteriormente. O servidor da aplicação também passa a enviar as notificações ao AeroGear, desta forma evitando a comunicação direta com cada um dos serviços de notificações. O AeroGear está disponível através da plataforma de serviços OpenShift (*Platform-as-a-Service*) e também pode ser instalado em um servidor local.

Figura 4 – Arquitetura AeroGear Unified Push Server



Fonte: Blog Openshift (2013).

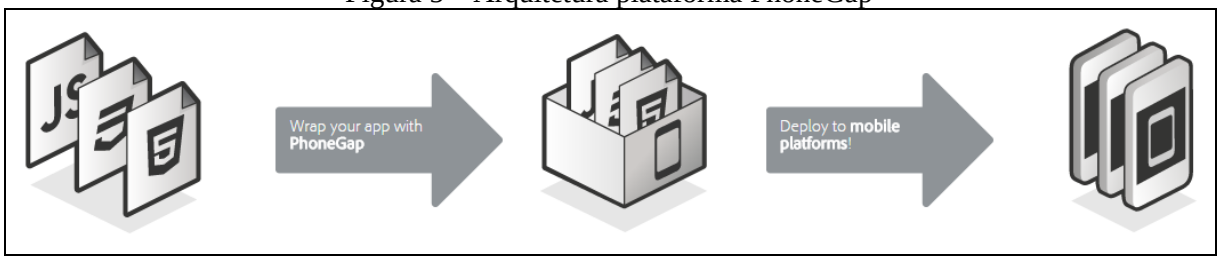
2.3 PLATAFORMA PHONEGAP E SEUS PLUG-INS (GOOGLE MAPS, FACEBOOK, AEROGEAR).

Desenvolver aplicativos para dispositivos móveis, pode não ser uma tarefa simples. Cada uma das plataformas disponíveis no mercado exige o conhecimento de *frameworks*, ferramentas e linguagens diferentes de cada fornecedor. Quando uma equipe de desenvolvedores tem o objetivo de implementar um novo aplicativo e que funcione na maioria dos dispositivos móveis, deve-se tomar a decisão de iniciar o desenvolvimento nativo ou utilizar um *framework* multiplataforma (GUINTHER, 2011). A opção de implementar um aplicativo utilizando um *framework* de desenvolvimento multiplataforma pode ser

interessante pelo fato de diminuir o tempo gasto se comparado ao desenvolvimento nativo para cada uma das plataformas *mobile*. O PhoneGap é uma destas tecnologias de desenvolvimento *mobile* multiplataforma (*open source*), que permite a criação de aplicativos utilizando os padrões de desenvolvimento *web* como o HTML, CSS e Javascript (PHONEGAP, 2012).

Aplicativos desenvolvidos com o PhoneGap são considerados híbridos, ou seja, são nativamente instalados em dispositivos móveis, porém criados com tecnologia *web*. Pense no aplicativo como um navegador da internet renderizando o conteúdo HTML, porém sem as barras de ferramentas de um navegador comum (PHONEGAP, 2012). Esta tecnologia possui uma Application Programming Interface (API) que, através do Javascript, fornece acesso às funcionalidades nativas do sistema operacional como acelerômetro, câmera, compasso, contatos, arquivos, geolocalização, notificações e banco de dados. Na Figura 5 é apresentado o processo de conteinerização do conteúdo *web* com a plataforma PhoneGap para as plataformas nativas.

Figura 5 – Arquitetura plataforma PhoneGap



Fonte: Blog PhoneGap (2012).

Empresas de software disponibilizam seus Softwares Development Kits (SDKs) para integração dos seus produtos com aplicativos desenvolvidas para as plataformas móveis. A Google, por exemplo, disponibiliza o SDK do Google Maps, com funções de incorporação de seus mapas interativos, disponibilização de ferramentas como o Google Street View, Indoor Maps, acesso a informações de estabelecimentos, locais, rotas, elevações e pontos de interesse conhecidos (GOOGLE DEVELOPERS, 2015). O Facebook também disponibiliza seu SDK para integração dos aplicativos móveis com seus produtos, permitindo, por exemplo, a utilização da funcionalidade de autenticação através da rede social. Neste caso, o usuário não precisa criar um novo cadastro para utilização do aplicativo desenvolvido, pois a mesma conta da rede social pode ser utilizada para o acesso. Este SDK também possibilita (mediante autorização do usuário) a utilização de informações pessoais, lista de amigos, fotos, compartilhamento, *likes* de conteúdo e envio de mensagens (FACEBOOK DEVELOPERS, 2015). Para complementar e tornar o desenvolvimento de aplicativos interessante, existem diversos *plug-ins* disponibilizados pela comunidade para serem utilizados com o PhoneGap.

Alguns deles são adaptações de SDKs nativos, como por exemplo o *plug-in* do Google Maps e do Facebook para uso de suas funcionalidades com esta plataforma. Outro *plug-in* utilizado neste trabalho é o do serviço AeroGear, explicado na seção 2.2, que possibilita a integração deste serviço e uso de notificações nativas de acordo com o sistema operacional utilizado pelo usuário.

2.4 TRABALHOS CORRELATOS

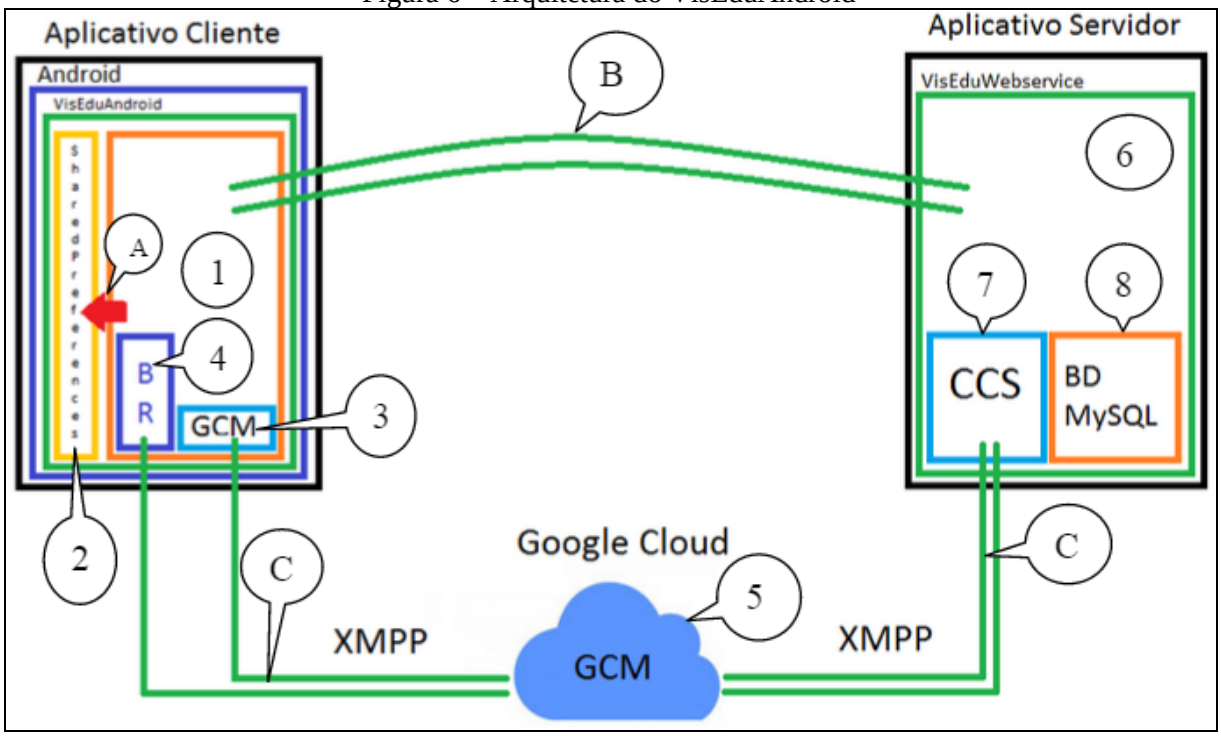
Nesta seção, são apresentados trabalhos que possuem correlação com aplicativo desenvolvido. Entre eles está o sistema desenvolvido por Kestring (2014) e o aplicativo desenvolvido por Kuehl (2012). Também são exploradas algumas funcionalidades do aplicativo Find My Friend desenvolvido pela Apple (2015) e disponível em sua loja virtual de aplicativos.

2.4.1 Sistema móvel na plataforma Android para compartilhamento de geolocalização usando mapas e notificações da Google

O sistema desenvolvido por Kestring (2014) teve como principal foco o evento realizado anualmente pela FURB, o Interação FURB. Ele é composto por um aplicativo cliente, no qual apresenta um mapa interativo e permite que pontos de interesse sejam cadastros pelos participantes do evento dentro do mapa virtual da FURB, são exemplos: os blocos e os espaços públicos. Através da geolocalização, é possível que sua localização atual em relação ao mundo real e à posição de outros participantes seja compartilhada no mapa. O sistema também possui um servidor que se comunica através de *webservices* e é responsável por responder as requisições do aplicativo cliente, como localização e pontos de interesse.

A plataforma Android foi utilizada para o desenvolvimento do trabalho, em conjunto com o *framework* de mapas Google Maps Android v2 e o serviço de notificação GCM. Na Figura 6 pode-se ter um entendimento melhor da arquitetura do projeto.

Figura 6 – Arquitetura do VisEduAndroid



Fonte: Kestring (2014, p. 23).

O aplicativo cliente, chamado `VisEduAndroid (1)`, possui uma área de recursos compartilhados `SharedPreferences (2)`, utilizada para gravar o identificador `gcmId (A)` gerado pelo serviço de notificações do lado da aplicação `(3)`. O `BroadcastReceiver (4)` é responsável por receber as notificações enviadas pelo serviço de notificações `GCM (5)` através do protocolo `XMPP (C)`. O cliente `VisEduAndroid` envia e recebe dados do servidor `VisEduWebservice (6)` através de *webservices* `(B)`, que por sua vez, grava as informações recebidas no banco de dados `MySQL (8)`. Para que a aplicação do lado do servidor consiga enviar notificações ao dispositivo móvel, através do GCM, é necessário que o serviço `Cloud Connection Server (7)` esteja implementado.

Os resultados obtidos com trabalho de Kestring (2014) mostraram-se interessantes e úteis para o desenvolvimento de novas aplicações utilizando o *framework* de mapas da Google. Ainda segundo Kestring (2014), duas dificuldades encontradas no trabalho foram a implementação das notificações, por possuir pouco material de apoio, e a utilização do Indoor Maps, que na época não permitia o cadastro de pontos de interesse no mapa, e exigindo que este recurso fosse desenvolvido na própria aplicação.

2.4.2 Protótipo de sistema móvel na plataforma Android para compartilhamento de arquivos e mensagens entre dispositivos baseado em proximidade geográfica

A tecnologia de geolocalização também foi útil para o trabalho de Kuehl (2012). Seu aplicativo, desenvolvido para a plataforma Android, exibi ao usuário uma lista das pessoas que estão geograficamente próximas a sua posição e que também possuam o aplicativo instalado em seu dispositivo. Permite que arquivos compartilhados sejam disponibilizados e que mensagens possam ser trocadas entre as pessoas da lista. O usuário pode selecionar uma pessoa visível em sua lista e requisitar um arquivo compartilhado por ela. Quando um usuário compartilha seus arquivos, eles são enviados via *webservice* e armazenados em um servidor. Ao requisitar um arquivo compartilhado por uma pessoa, outro *webservice* se encarrega de buscá-lo no servidor e enviá-lo para o dispositivo.

O aplicativo também permite que a distância limite em que usuários possam ser encontrados seja configurada, ou seja, é possível definir a área de cobertura em relação a localização atual do dispositivo. Conforme a Figura 7, esta configuração encontra-se na parte superior da tela principal do aplicativo e logo abaixo são exibidas as pessoas próximas a sua localização. Ao lado de cada pessoa estão disponíveis dois ícones, o primeiro deles para iniciar uma conversa e o segundo para visualizar os arquivos compartilhados por ela. Por último, na parte inferior, o botão “Meus Arquivos” serve para que usuário selecione os arquivos que deseja compartilhar. Assim como no aplicativo desenvolvido por Kestring (2014), as técnicas de compartilhamento de localização também foram utilizadas, possibilitando usuários encontrarem uns aos outros e permitindo que interajam entre si.

Figura 7 – Aplicativo LocalSocial

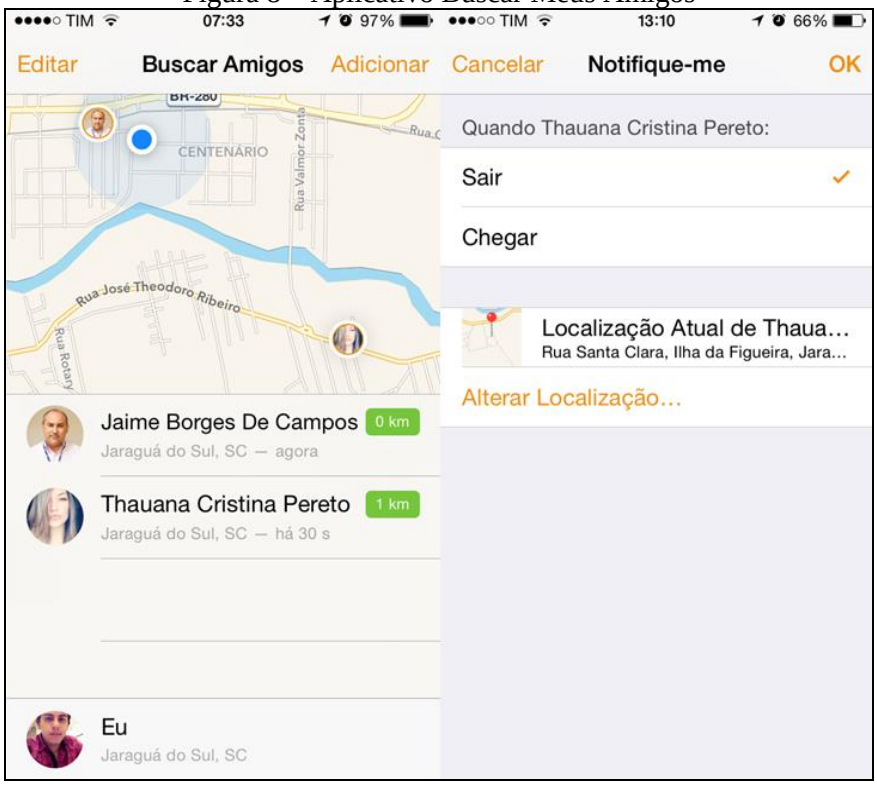
Fonte: Kuehl (2012).

2.4.3 Aplicativo Find My Friend da Apple (Buscar Meus Amigos)

Desenvolvido pela Apple (2015) e disponível para *download* em sua loja virtual, o aplicativo Find My Friends é compatível com os equipamentos que suportam o iOS. O compartilhamento de localização entre pessoas, amigos e familiares é o propósito principal desta ferramenta. Ela utiliza como base a lista de contatos do dispositivo móvel para localizar e mostrar onde as pessoas encontram-se no mapa (mediante autorização). Por exemplo, caso uma pessoa precise chegar até seus familiares que estão na areia de uma praia lotada, será uma tarefa bastante árdua para encontrá-los. Com a utilização do aplicativo a localização será revelada e rapidamente os mesmos serão encontrados (APPLE, 2015).

A ferramenta permite que novas regras de busca sejam criadas, ou seja, é possível saber quando uma pessoa da lista de contatos sai ou chega em um determinado local. Caso queira saber quando familiares chegarem na cidade ou no aeroporto, basta criar uma nova regra e quando ela for atendida, uma notificação será enviada para o dispositivo. O compartilhamento controlado de localização também pode ser atribuído a um grupo de pessoas e quando a data de expiração chegar, o mesmo é desfeito.

Figura 8 – Aplicativo Buscar Meus Amigos



Na Figura 8, é possível ver no mapa o compartilhamento da localização entre os usuários do aplicativo e a distância para chegar até eles. Ainda nesta figura é possível observar a criação de uma nova regra para o envio de notificação de acordo com a movimentação dos amigos (APPLE, 2015).

3 DESENVOLVIMENTO DO APLICATIVO

Este capítulo aborda os aspectos referente à construção do aplicativo. Na seção 3.1 encontram-se os requisitos funcionais e não funcionais. Na seção 3.2 são apresentados os diagramas de casos de usos, classes e atividades, detalhando o funcionamento e especificando o aplicativo. Ainda nesta seção, são apresentados os principais serviços que integram o Interação FURB, mostrando a comunicação entre todos os componentes. A seção 3.3 retrata toda as ferramentas, bibliotecas e *frameworks* utilizados durante o desenvolvimento. Os principais trechos de código e utilização do aplicativo também são explicados nesta seção. Para encerrar o capítulo, a seção 3.4 expõem os resultados obtidos após o desenvolvimento do aplicativo e realiza a comparação entre trabalhos correlatos apresentados na seção 2.4.

3.1 REQUISITOS

Os requisitos funcionais e não funcionais do aplicativo desenvolvido estão descritos conforme lista abaixo:

a) o aplicativo deve permitir a autenticação do usuário via Facebook (Requisito Funcional - RF);
b) o aplicativo deve permitir a visualização da localização atual do usuário através de um mapa (RF);
c) o aplicativo deve permitir o compartilhamento de localização entre amigos (RF);
d) o aplicativo deve permitir a visualização de uma lista de amigos que também possuem o aplicativo instalado (RF);
e) o aplicativo deve permitir o envio de notificações para os usuários (RF);
f) o aplicativo deve ser desenvolvido com a plataforma PhoneGap (Requisito Não Funcional - RNF);
g) o aplicativo deve ser desenvolvido com tecnologia web (HTML, CSS e Javascript) no ambiente de desenvolvimento XCode e Eclipse (RNF);
h) o aplicativo deve se comunicar com o servidor responsável por gerenciar as localizações por requisições webservice do tipo REST, utilizando a biblioteca JQuery (RNF);
i) o aplicativo deve utilizar a API do Facebook para a plataforma PhoneGap e realizar a integração com esta rede social (RNF);
j) o aplicativo deve utilizar a API do Google Maps para a plataforma PhoneGap e realizar a integração com os mapas da Google (RNF);
k) o aplicativo deve utilizar a API do AeroGear para a plataforma PhoneGap e

realizar a integração com o servidor de notificações;

l) o aplicativo deve utilizar o Indoor Maps para a visualização de espaços internos (RNF);
m) o aplicativo deve utilizar a tecnologia *Push Notification* para envio de notificações independente de sistema operacional (RNF);
n) o servidor de localizações deve ser desenvolvido na linguagem de programação Java e utilizar a ferramenta para automação de compilação Maven (RNF);
o) o servidor de localização deve utilizar a API do AeroGear para se comunicar com o servidor de notificações (RNF);
p) o servidor de localização deve suportar a implantação no servidor de aplicação WildFly (RNF);
q) o servidor de localização deve utilizar o banco de dados MySQL e realizar a persistência dos dados através do *framework* Hibernate (RNF);
r) o servidor de localização deve suportar chamadas REST através do *framework* RESTEasy (RNF);
s) o servidor de notificações deve utilizar o serviço AeroGear Unified Push Server para envio das notificações (RNF).

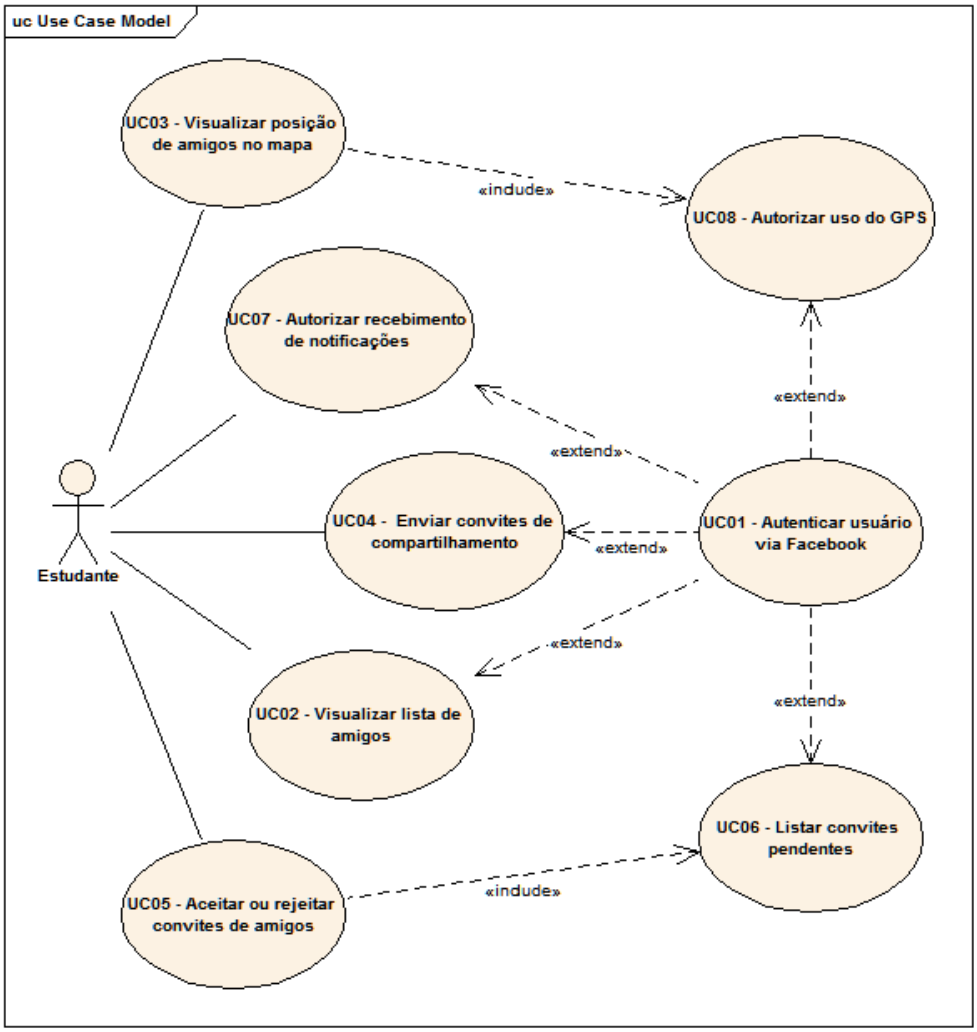
3.2 ESPECIFICAÇÃO

Para o entendimento das funcionalidades e arquitetura do aplicativo, esta seção apresenta os diagramas de casos de uso, classes e atividades, elaborados com a ferramenta Enterprise Architect (EA). No final desta seção é apresentada a arquitetura de comunicação entre o aplicativo e os servidores de localização e notificações.

3.2.1 Diagrama de casos de uso

A Figura 9 mostra os casos de usos que representam as funcionalidades disponíveis para os usuários do aplicativo.

Figura 9 – Diagrama de casos de uso



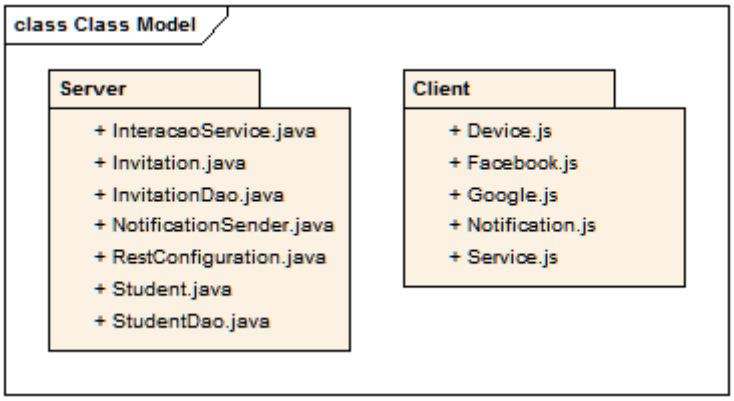
O primeiro caso de uso `UC01 – Autenticar usuário via Facebook` é ação inicial que o usuário deve executar antes de acessar as demais funções do aplicativo. Após sua autenticação, o caso de uso `UC07- Autorizar recebimento de notificações` permite que o usuário autorize o aplicativo a enviar notificações para seu dispositivo móvel. No caso de uso `UC03 – Visualizar posição de amigos no mapa` o usuário pode acompanhar a movimentação de seus amigos no campus da universidade. No entanto, isso somente será possível após o aplicativo ser autorizado a receber informações de posicionamento fornecidas pelo GPS do dispositivo, ação que está descrita no caso de uso `UC08 – Autorizar uso do GPS`. No caso de uso `UC02 – Visualizar lista de amigos`, o usuário pode visualizar os amigos que estão compartilhando sua localização. Caso ainda não exista um relacionamento ativo entre amigos, através do caso de uso `UC04 – Enviar convites de compartilhamento` é possível que convites sejam enviados para outros amigos que também possuam o aplicativo instalado em seus dispositivos. No caso de uso `UC06 – Listar convites pendentes`, após o recebimento da notificação alertando sobre novo convite de

compartilhamento, o usuário pode visualizar seus convites pendentes de decisão. Com o caso de uso `UC05 – Aceitar ou rejeitar convites de amigos`, o usuário pode tomar a decisão de compartilhar ou não suas informações de posicionamento com os novos amigos.

3.2.2 Diagrama de classes

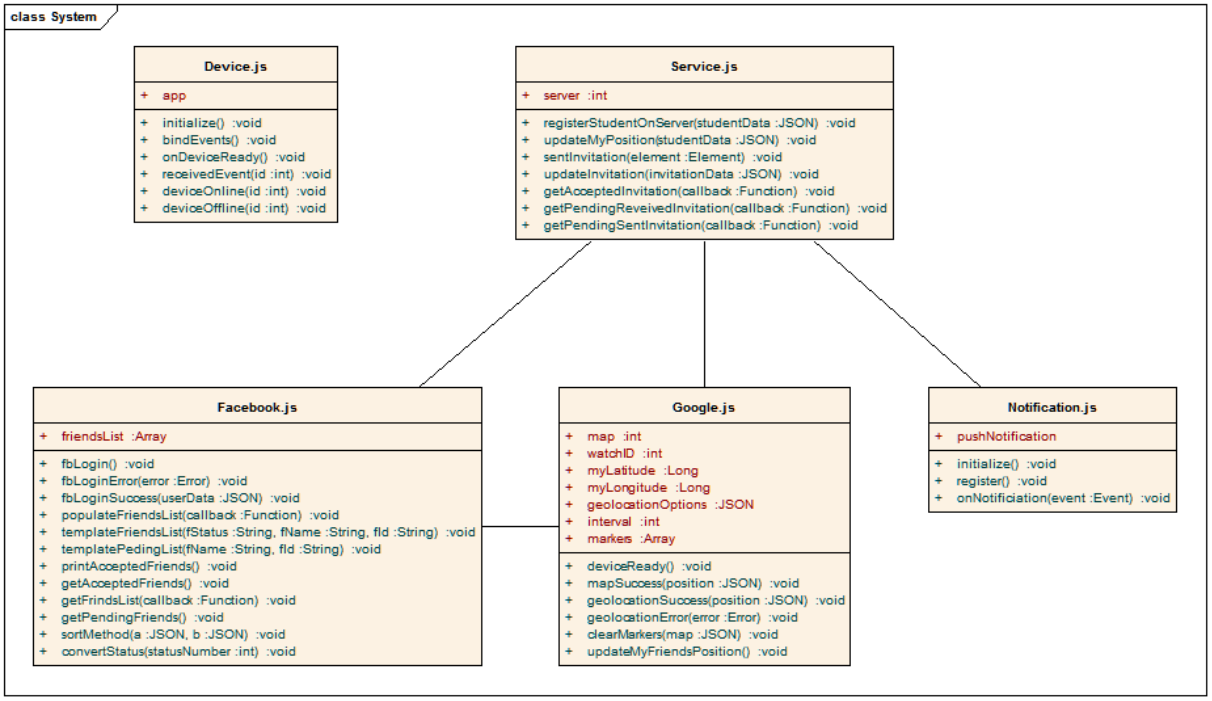
Os diagramas apresentados nesta seção detalham a estrutura de classes e objetos utilizados no desenvolvimento do aplicativo Interação FURB e do servidor gerenciador de localizações. A Figura 10 fornece uma perspectiva macro das principais classes do trabalho.

Figura 10 – Pacotes de classes



O diagrama da Figura 11 retrata a estrutura de classes do Interação FURB. A classe `Service.js` é a mais utilizada do aplicativo, pois tem a responsabilidade de intermediar e realizar as chamadas *webservice* para o servidor de localizações. Registrar novos usuários, enviar e receber convites de compartilhamento de localização e atualizar informações de coordenadas geográficas, estão entre suas principais funções. A integração com o Facebook é realizada através da classe `Facebook.js`, onde encontram-se as funções de autenticação e busca da lista de amigos. Esta classe também é responsável por gerenciar os amigos e os convites do usuário, apresentando na interface do aplicativo apenas aqueles que possuem o compartilhamento de localização ativo. Para a integração com o Google Maps foi utilizada a classe `Google.js`, que apresenta o mapa no aplicativo e atualiza a posição atual do usuário no servidor de localizações. Ainda nesta classe é realizado o gerenciamento e atualização dos marcadores que representam o usuário e seus amigos no mapa. A classe `Notification.js` tem como função registrar o dispositivo nos serviços de notificações, permitindo que o usuário receba alertas enviados para o aplicativo. Para finalizar, a classe `Device.js` tem como objetivo tratar os eventos ocorridos no dispositivo móvel, como por exemplo, a ação de bloquear o aparelho.

Figura 11 – Diagrama de classes do cliente



Na Figura 12, estão retratas as classes do servidor de localizações. A classe `InteracaoService.java` é a controladora principal, ou seja, ela possui o mapeamento de todas as chamadas *webservice* disponíveis e as redireciona para o método correspondente ao serviço solicitado. Registro de usuários, criação e busca de convites e atualização de coordenadas geográficas estão entres os principais serviços oferecidos por ela. Na classe `NotificiationSender.java` é realizada a integração com o servidor de notificações AeroGear, desta forma permitindo que alertas de eventos e atualizações sejam enviados aos dispositivos dos usuários interessados. Neste caso, por exemplo, é enviada uma notificação quando o usuário recebe um novo convite para compartilhar sua localização. As classes `Student.java` e `Invitation.java` representam os objetos que serão persistidos no banco de dados, permitindo que o compartilhamento de localização entre amigos seja gerenciável. A classe `Student.java` representa o usuário do aplicativo e possui informações de nome, geolocalização e identificador único. Já a classe `Invitation.java` representa o convite enviado a um usuário e possui as informações de remetente, destinatário, último usuário a ter alterado o *status* e o próprio *status*. Este convite é mantido no servidor para que seja possível identificar os relacionamentos ativos entre os amigos que utilizam o aplicativo. A persistência e manipulação destes objetos no banco de dados, são realizadas através das classes `StudentDao.java` e `InvitationDao.java` respectivamente.

Figura 12 – Diagrama de classes do servidor

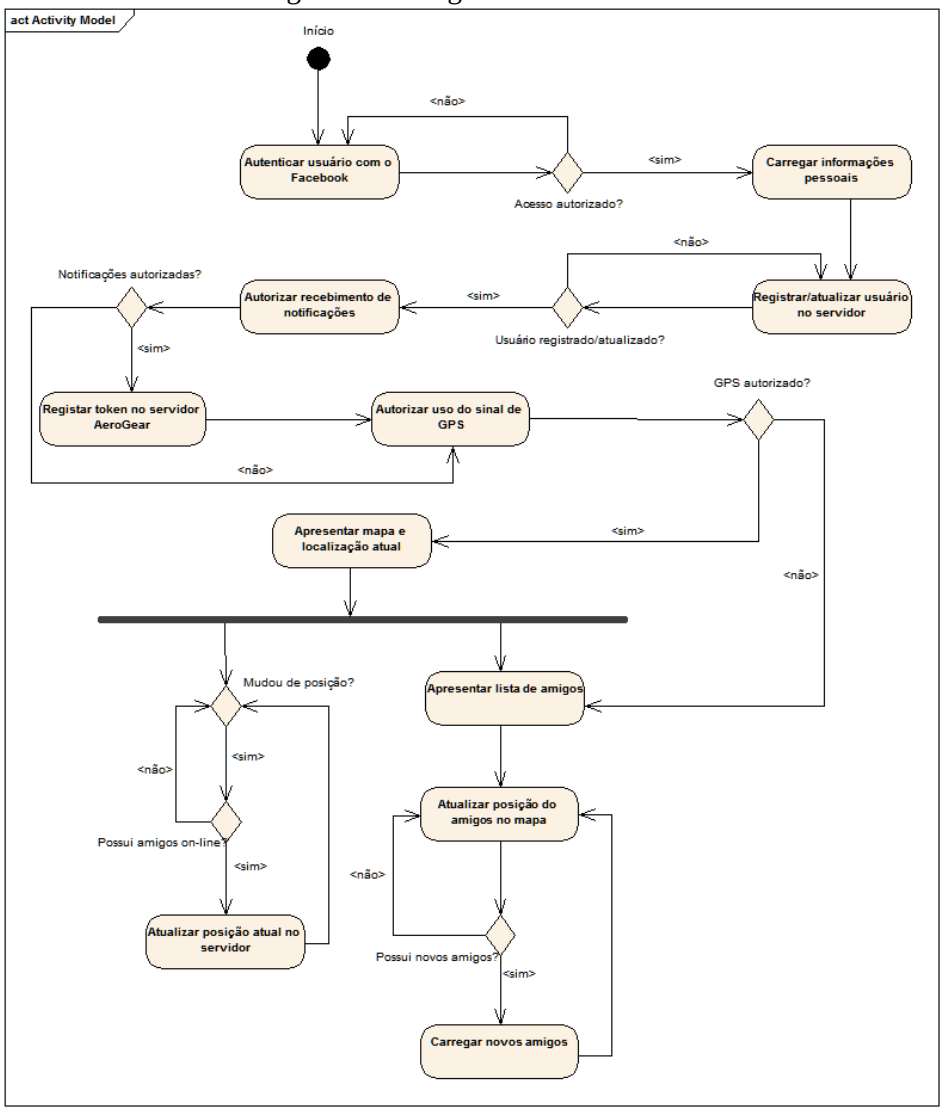
3.2.3 Diagrama de atividades

O diagrama de atividades da Figura 13, apresenta o fluxo principal de funcionamento do aplicativo desenvolvido. Para dar início ao processo, a primeira ação que usuário deve executar é realizar autenticação através de uma conta do Facebook. Após ter seu acesso autorizado, o usuário deve permitir que algumas informações desta conta (identificador, nome completo e lista de amigos) possam ser utilizadas pelo aplicativo. Obtidas estas informações, um novo registro de usuário é inserido ou atualizado no banco de dados do servidor responsável por gerenciar o compartilhamento de localização. Em sequência, o aplicativo solicita autorização para o recebimento de notificações enviadas quando um amigo solicita o

compartilhamento de localização ou quando a universidade envia alertas importantes. Caso as notificações sejam aceitas, um novo registro, contendo o token de *Push Notification* e identificador do Facebook, é inserido no servidor de notificações AeroGear.

Finalizando estas etapas, o aplicativo solicita ao usuário permissão para utilizar informações de localização fornecidas pela antena de GPS do dispositivo móvel. Se autorizado, o aplicativo irá exibir no mapa a posição de onde o usuário está localizado, carregar a lista de amigos que estão compartilhando suas localizações e exibi-los no mapa de acordo com seus posicionamentos. A cada dez segundos o ponto de referência de cada amigo no mapa é atualizando com base em informações atualizadas obtidas através do servidor de gerenciamento de localizações. Em paralelo ocorre a atualização da posição do próprio usuário no mapa, que também é enviada ao servidor de localizações assim que o evento de mudança de posição é ativado, porém somente quando for necessário e seus amigos estiverem precisando desta informação atualizada.

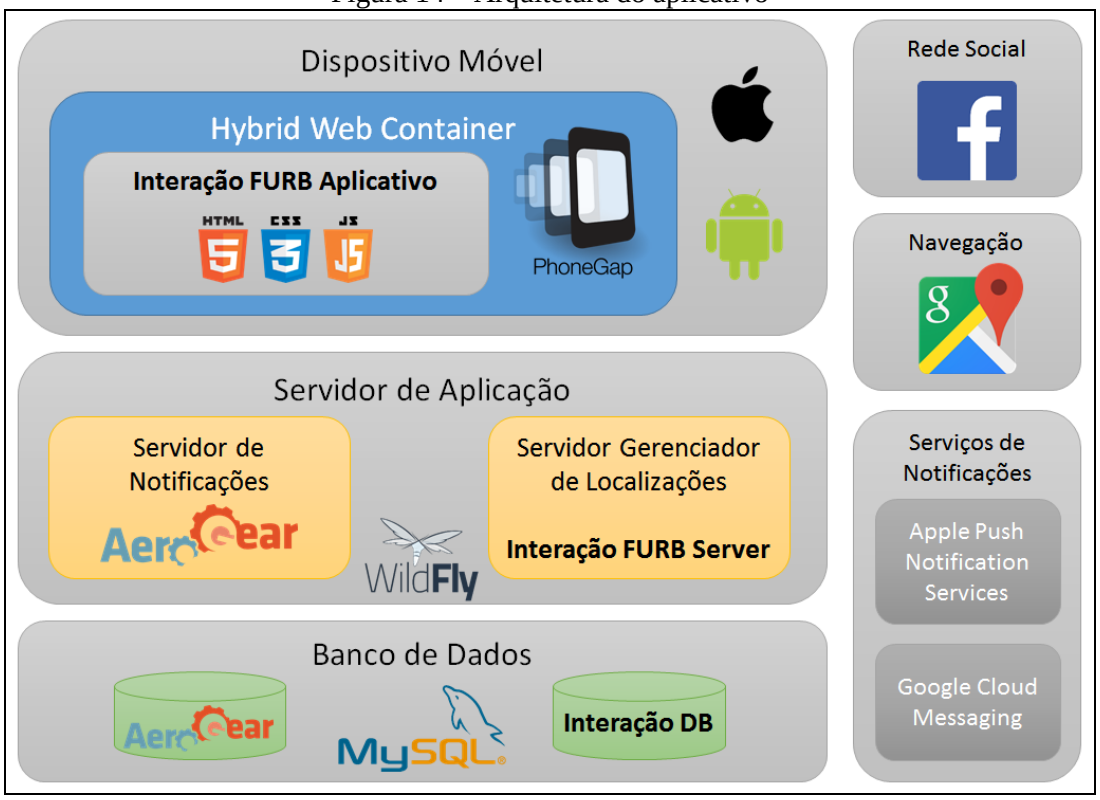
Figura 13 – Diagrama de atividades



3.2.4 Arquitetura e integrações do aplicativo

Para o funcionamento deste trabalho, vários serviços e ferramentas de terceiros foram adicionadas ao aplicativo. A Figura 14 mostra a arquitetura macro criada com a integração de todos estes serviços.

Figura 14 – Arquitetura do aplicativo

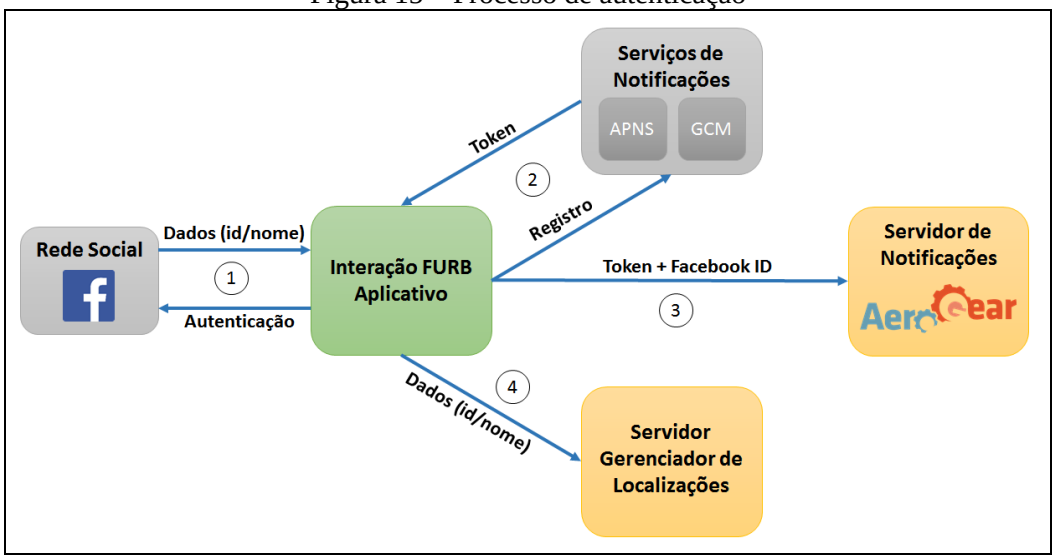


A primeira camada `Dispositivo Móvel` representa o dispositivo do usuário com o aplicativo `Interação FURB` instalado. Este, por sua vez, é um `Hybrid Web Container` desenvolvido com a plataforma PhoneGap e que pode ser executado em sistemas operacionais diferentes como Android e iOS. O aplicativo está integrado com a `Rede Social` Facebook para facilitar o procedimento de autenticação e a busca de amigos do usuário. O sistema de `Navegação` Google Maps também está integrado nele, disponibilizando o mapa que mostrará a localização do usuário e de seus amigos. Este mapa também conta com a tecnologia Google Indoor Maps que poderá exibir a planta baixa da universidade facilitando a navegação do usuário quando encontrar-se dentro dos prédios da instituição.

Na segunda camada, encontra-se o `Servidor Gerenciador de Localizações – Interação FURB Server` responsável por realizar o gerenciamento dos relacionamentos entre os usuários que compartilham suas localizações. Ainda nesta camada, está situado o `Servidor de Notificações – AeroGear` responsável por identificar e direcionar as notificações para os `Serviços de Notificações` da Apple (`APNS`) ou da Google (`GCM`)

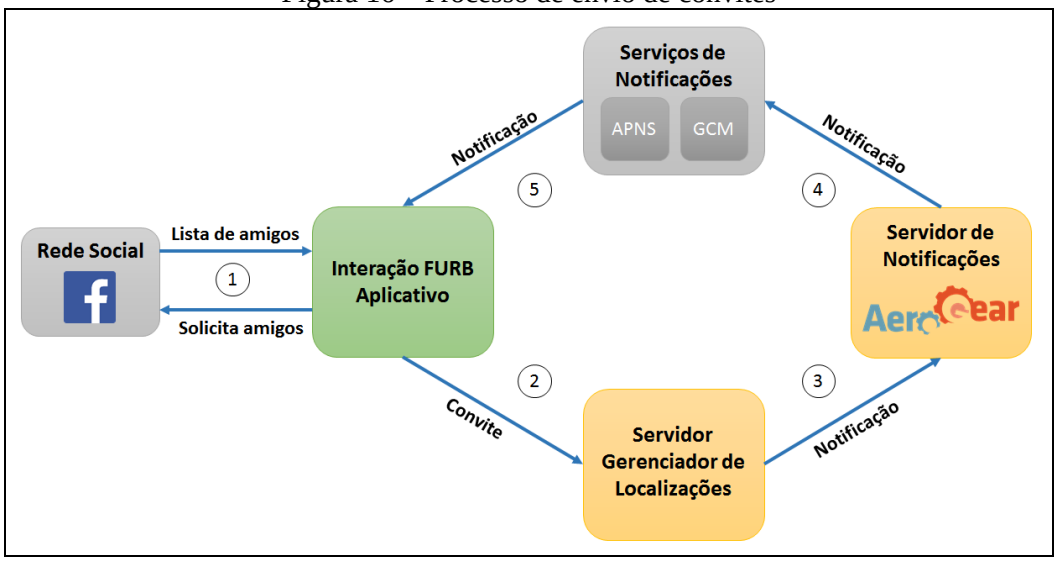
conforme o sistema operacional utilizado pelo usuário. Ambos os servidores rodam sobre o `Servidor de Aplicação` WildFly e persistem seus dados no `Banco de Dados` MySQL, conforme mostra a terceira camada. Para assimilação da comunicação entre os serviços utilizados, a Figura 15 demonstra o processo de autenticação do usuário no aplicativo `Interação FURB`.

Figura 15 – Processo de autenticação



Conforme descrito anteriormente, para iniciar o uso da aplicação é necessário que o usuário realize a autenticação através de sua conta do Facebook. Assim como mostra o passo `(1)`, após a autenticação ser realizada os dados básicos do usuário como nome e identificador são retornados ao aplicativo. No passo `(2)`, o aplicativo solicita o registro do usuário no serviço de *push notification* correspondente ao seu sistema operacional para permitir o recebimento de notificações. O serviço correspondente retorna um *token* único que durante o passo `(3)` será gravado no servidor AeroGear. O registro deste *token* será associado ao identificador da conta do Facebook do usuário na forma de um *alias*. Isso se faz necessário para que as notificações possam ser enviadas a um usuário específico e em todos os seus dispositivos registrados. Estes procedimentos são executados somente na primeira vez em que o usuário acessa a aplicação ou quando o *token* registrado não for mais válido. Para finalizar o processo de autenticação, no passo `(4)` o registro do usuário é persistido no servidor gerenciador de localizações e, caso já existam, os dados do usuário serão apenas atualizados. A seguir na Figura 16, o processo de envio de convites é explicado para reforçar o entendimento da arquitetura do aplicativo.

Figura 16 – Processo de envio de convites



Este processo é executado quando o usuário convida um de seus amigos para iniciar o compartilhamento de localização entre eles. No passo (1) o aplicativo carregada a lista de amigos da conta do Facebook do usuário e habilita o envio de convites para cada um destes elementos. Após selecionar um amigo da lista, o aplicativo executa o passo (2) persistindo o registro do convite no servidor gerenciador de localizações. Posteriormente este registro servirá para gerenciar o relacionamento de compartilhamento de localização entre os usuários do aplicativo. Ainda neste servidor, o passo (3) mostra o envio de uma notificação para o servidor AeroGear. Para que o amigo selecionado seja notificado sobre o novo convite, esta notificação é enviada para o dispositivo do destinatário. Junto à notificação, é enviada a identificação do Facebook deste amigo que representa o *alias*, permitindo que o AeroGear encontre o *token* correto a ser repassado aos serviços de notificações (APNS e GCM). No passo (4), a notificação é enviada para estes serviços que são responsáveis por realizar a entrega da mensagem ao destinatário, ou seja, o dispositivo móvel do amigo convidado. Porém as empresas que fornecem estes serviços de notificações não garantem 100% da entrega da mensagem, conforme mostra o passo (5). Desta forma, não devem ser utilizados para o envio de informações críticas.

3.3 IMPLEMENTAÇÃO

Nesta seção são apresentadas as técnicas e ferramentas utilizadas no trabalho e as etapas seguintes para o desenvolvimento do aplicativo Interação FURB e do servidor gerenciador de localizações. Para finalizar esta são serão apresentadas as telas e funcionalidades do Interação FURB.

3.3.1 Técnicas e ferramentas utilizadas

O desenvolvimento deste trabalho foi dividido em três etapas. A primeira delas é a criação do aplicativo para dispositivos móveis multiplataforma Interação FURB, construído com base no *framework* PhoneGap. Para a construção do código fonte, utilizou-se as ferramentas padrões de desenvolvimento *web* HTML, CSS e Javascript, com o auxílio do ambiente de desenvolvimento XCode da plataforma Mac OS. A biblioteca JQuery foi utilizada para facilitar a manipulação do HTML e realizar as chamadas *webservice* do tipo RESTful. Ainda nas chamadas *webservice*, o padrão de notação de objetos JSON foi utilizado para enviar e receber dados do servidor. Para a camada de apresentação, foi utilizado o *framework* Framework7, que possibilita a criação de aplicativos híbridos com o visual nativo das plataformas iOS e Android. Complementando o desenvolvimento do Interação FURB, foram utilizados quatro *plug-ins* criados para o PhoneGap: um para integração da rede social Facebook (`phonegap-facebook-plugin`); um para integração dos mapas do Google Maps (`cordova-plugin-googlemaps`); um para recebimento de notificações através do serviço AeroGear (`aerogear-pushplugin-cordova`); e o último para leitura dos dados fornecidos pela antena de GPS do dispositivo móvel (`cordova-plugin-geolocation`).

A segunda etapa foi a realização do desenvolvimento do servidor responsável por gerenciar o compartilhamento de localização entre os dispositivos móveis. Este servidor foi construído na linguagem de programação Java, no ambiente de desenvolvimento Eclipse e em conjunto com a ferramenta para automação de compilação Maven. Na camada de persistência de dados foi utilizado o *framework* Hibernate que facilitou a construção das operações de manipulação do banco de dados. Para a camada de serviço, foi utilizado o *framework* RESTeasy, permitindo através de anotações no código fonte, mapear as chamadas *webservices* para os métodos correspondentes. No que se diz respeito a o envio de notificações, foi incluída a dependência do cliente AeroGear para a linguagem Java, permitindo a integração deste servidor com os serviços de notificações.

Para finalizar a construção deste trabalho, foi realizada a configuração do servidor de notificações AeroGear, conforme explicado no Apêndice A. Ambos os servidores, de notificações e localização, foram construídos e configurados para rodar sobre o servidor de aplicação WidFly 9 e com o banco dados MySQL.

3.3.2 Etapas do desenvolvimento

Nesta seção, encontram-se trechos de códigos desenvolvidos para realizar a integração do aplicativo com os serviços de rede social, mapas e notificações. Os principais métodos do aplicativo são detalhados para o entendimento de comportamento de sua execução.

3.3.2.1 Integração com a rede social Facebook

Para realizar a integração do aplicativo com o Facebook foi utilizado o *plug-in* `phonegap-facebook-plugin`. Na linha 1 do Quadro 1, encontra-se a função `fbLogin` que é acionada ao clicar no botão `Entrar com Facebook` da Figura 17. Para realizar a autenticação via Facebook, a linha 2 chama a função `login` da classe `facebookConnectPlugin`, passando como parâmetro os tipos de dados que o aplicativo deseja utilizar do perfil acessado, como por exemplo, informações básicas do usuário e sua a lista de amigos. Ainda nesta função são passadas como parâmetro outras duas funções `fbLoginSuccess` e `fbLoginError`, processadas de acordo com o retorno da autenticação do usuário. Após o acesso ser autorizado, a função `fbLoginSuccess` da linha 5 irá armazenar as informações da conta do usuário no contexto da aplicação (semelhante aos *cookies*), conforme mostra as linhas 8, 9 e 12. Ainda no Quadro 1, a função `registerStudentOnServer` da linha 18 é acionada para realizar o registro no usuário no servidor de localizações com as informações obtidas do Facebook do usuário.

Quadro 1 – Autenticação via Facebook

```
1  var fbLogin = function (){
2      facebookConnectPlugin.login(["public_profile", "user_friends"],
3  fbLoginSuccess, fbLoginError);
4  };
5  var fbLoginSuccess = function (userData) {
6      var myFbId = userData.authResponse.userID;
7      var myFbToken = userData.authResponse.accessToken;
8      localStorage.setItem('fbId', myFbId);
9      localStorage.setItem('fbToken', myFbToken);
10     facebookConnectPlugin.api('/me', null, function(response){
11       var myFbName = response.name;
12       localStorage.setItem('fbName', myFbName);
13       var JSONdata = {
14             studentFbId: myFbId,
15             studentName: myFbName,
16             studentFbToken: myFbToken,
17             studentStatus: 1 };
18       registerStudentOnServer(JSONdata);
19     });
20 }
```

Na linha 1 do Quadro 2, é declarada a função `getFriendsList`, responsável por criar uma lista dos amigos do usuário no formato HTML e apresentá-la na interface do aplicativo.

A linha 2 chama a função `api` da classe `facebookConnectPlugin` e passa como parâmetro os tipos de dados que devem ser obtidos do Facebook. Neste caso, foi solicitada a foto e o nome dos amigos disponíveis. A execução desta função é realizada de forma assíncrona, ou seja, o programa continua sua execução e não aguarda o retorno dos dados solicitados. Para que seja possível tratar os dados obtidos, esta função permite que uma nova função de retorno seja passada como parâmetro dela, sendo possível desta forma tratar a lista de amigos recebida. Esta função de retorno encontra-se na linha 4 e tem a responsabilidade de realizar a conversão dos dados recebidos para uma lista de amigos no formato HTML. Conforme o código das linhas 9 até 12, para cada amigo obtido através da função `api`, um novo item é adicionado nesta lista HTML. O conteúdo HTML resultante é retornado para a função de um *callback*.

Quadro 2 – Busca da lista de amigos

```
1  var getFriendsList = function(callback){
2    facebookConnectPlugin.api( "me/friends?fields=picture,name",
3      ["public_profile", "user_friends"],
4      function (response){
5        var friendData = response.data.sort(sortMethod);
6        var results = '<div class="list-block media-list contacts-list
7          list-block-search searchbar-found">';
8        results +='<ul id="friends-list" class="friends-list">';
9        for (var i = 0; i < friendData.length; i++) {
10         results += templateFriendsList(friendData[i].id,
11           friendData[i].name, '');
12       }
13       results += '</ul></div>';
14       callback(results);
15     });
16 }
```

3.3.2.2 Integração com os mapas do Google Maps

A integração do aplicativo com o Google Maps foi realiza pelo auxílio dos *plug-ins* `cordova-plugin-googlemaps` e `cordova-plugin-geolocation`. O código apresentado no Quadro 3, é utilizado para adicionar o mapa na tela principal do aplicativo e iniciar o processo de atualização do georreferenciamento do usuário no servidor de localizações. Quando a tela principal do aplicativo for aberta, o evento `deviceready` será acionado, dando início a execução da função representada pela linha 1. Esta função irá cria o mapa dentro do elemento HTML que possui o identificador informado pela linha 2 e de acordo com as configurações informadas das linhas de 3 a 15. Após o mapa ser criado com sucesso, a função `onSuccess` da linha 17 será acionada para criar o evento `watchPosition` do *plug-in* de geolocalização. Este evento tem o objetivo de verificar, em um intervalo de tempo predeterminado, se houve alteração no posicionamento do usuário. Caso esta condição seja verdadeira, a função `geolocationSuccess` deste evento será acionada. As novas coordenadas do usuário serão

passadas como parâmetro desta função para que sejam atualizadas no servidor de localizações através da função `updateMyPosition` da linha 33.

Quadro 3 – Integração com o Google Maps

```
1  document.addEventListener("deviceready", function(){
2      map = plugin.google.maps.Map.getMap($("#mapCanvas"));
3      map.setOptions({
4          'controls':{
5              'compass': true,
6              'myLocationButton': true,
7              'indoorPicker': true
8          },
9          'gestures': {
10             'scroll': true,
11             'tilt': true,
12             'rotate': true,
13             'zoom': true
14         }
15     });
16     map.addEventListener(plugin.google.maps.event.MAP_READY,
17 onSuccess);
18 }, false);
19 var onSuccess = function(position) {
20     watchID = navigator.geolocation.watchPosition(geolocationSuccess,
21 geolocationError, geolocationOptions);
22 };
23 var geolocationSuccess = function(position) {
24     lat = position.coords.latitude;
25     long = position.coords.longitude;
26     var myCoordinates = new plugin.google.maps.LatLng(lat, long);
27     var studentData = {
28         studentFbId:localStorage.getItem('fbId'),
29         studentLatitude: myLatitude,
30         studentLongitude: myLongitude,
31         studentStatus: 1
32     };
33     updateMyPosition(studentData);
34 };
```

3.3.2.3 Integração com as notificações do AeroGear

O código fonte apresentado no Quadro 4, realiza a integração com o servidor de notificações AeroGear e está sendo amparado pelo *plug-in* `aerogear-pushplugin-cordova`. Este código é executado após o registro do usuário no servidor de localizações, com a chamada para a função `initialize` da linha 2. Esta função aguarda até que o evento `deviceready` seja disparado pelo dispositivo móvel e possa acionar a função `register` da linha 5. Nas linhas 7 até 19, são configurados os parâmetros para registro do dispositivo no servidor AeroGear. Esta configuração contém o endereço deste servidor, o *alias* utilizado para identificação do usuário e as informações de variantes específicas para as plataformas iOS e Android. O registro é executado na linha 21 com a chamada da função `register` disponibilizada pelo *plug-in* do AeroGear. As configurações da variável `pushConfig` e o

token gerado pelo serviço de notificação da plataforma correspondente ao sistema operacional do usuário, serão enviados para o servidor AeroGear. A execução deste registro é feita somente na primeira vez em que o usuário realiza a autenticação no aplicativo, desta forma, evitando que a mensagem para autorização do recebimento de notificações apareça todas as vezes em que o aplicativo for aberto.

Quadro 4 – Integração com o AeroGear

```
1  var pushNotification = {
2    initialize: function(){
3      document.addEventListener('deviceready', this.register, false);
4    },
5    register: function(){
6      var myId = localStorage.getItem('fbId');
7      var pushConfig = {
8        pushServerURL: "http://192.168.1.12:8080/ag-push/",
9        alias: myId,
10       ios: {
11         variantID: "a3dc497a-XXXX-XXXX-XXXX-8b481f73802b",
12         variantSecret: "7f1df0a4-XXXX-XXXX-XXXX-763ec05067b1"
13       },
14       android: {
15         sender: "37471308173",
16         variantID: "849c1cb8-XXXX-XXXX-XXXX-c44bf23a3149",
17         variantSecret: "ea5b7fbf-XXXX-XXXX-XXXX-7a2f8b173211"
18       }
19     };
20     if (typeof push !== 'undefined') {
21       push.register(pushNotification.onNotification, successHandler,
22 errorHandler, pushConfig);
23     }
24   }
25 };
```

3.3.2.4 Compartilhamento de localização

Nesta seção é explicado o código fonte do Quadro 5, responsável por gerenciar os marcadores que representam a movimentação dos amigos no mapa. Quando o usuário é redirecionado para a tela principal do aplicativo, conforme mostra a Figura 18, uma lista de amigos que possuem um relacionamento para o compartilhamento de localização ativo, é carregada no contexto da aplicação. Caso esta lista de amigos não esteja vazia, a função `map` da biblioteca Javascript Async é utiliza para realizar o processamento paralelo de cada item desta lista. Assim como mostra a linha 3, para cada amigo da lista de amigos `fList` é criada uma nova *thread* de execução para a função de criação do marcador que representará a movimentação daquele amigo no mapa. Na linha 4, a função `addMark` cria um novo marcador ainda não visível. Em seguida um novo evento é associado a este marcador, ou seja, todas as vezes em que este evento for chamado, a função corresponde a ele será executada. A linha 5 mostra a criação do evento único para alteração da posição de cada marcador criado,

conforme a função associada na linha 6. Esta função recebe como parâmetro um objeto que contém as informações das coordenadas geográficas do amigo correspondente. Estas informações são utilizadas para configurar a posição do marcador em relação ao mapa, conforme as linhas de 7 até 10. Assim como mostra a linha 11, quando o marcador estiver configurado de acordo com o posicionamento do amigo, ele poderá ser mostrado no mapa. Após todos os amigos da lista terem sido representados através de um marcador, a função `map` executa a função expressada pela linha 16. Esta última função, através da função `getAcceptedInvitation` da linha 18, tem o objetivo de buscar do servidor de localizações as informações atualizadas sobre o posicionamento de cada amigo da `fList`. O intervalo de tempo para atualização destas informações é definido pela linha 33. Quando todas as informações forem atualizadas, é disparado o evento associado em cada marcador existente. A função `trigger` da linha 31 representa o gatilho que aciona os eventos.

Quadro 5 – Atualização dos pontos de referência

```
1  function updateMyFriendsPosition(){
2      if (typeof fList !== 'undefined' && fList.length > 0){
3        async.map(fList, function(friendData, callback){
4          map.addMarker({ visible: false }, function(marker){
5            map.on("friend_change_" + friendData.studentFbId,
6              function(friend){
7                var lt = friend.studentLatitude;
8                var lg = friend.studentLongitude;
9                var position = new plugin.google.maps.LatLng(lt, lg);
10               marker.setPosition(position);
11               marker.setVisible(true);
12             });
13           markers.push(marker);
14           callback(null, marker);
15         });
16       }, function(markers){
17            interval = setInterval(function(){
18              getAcceptedInvitation(function(d){
19                fList = [];
20                for(var i in d){
21                  if(d.hasOwnProperty(i)){
22                    var sr = d[i].invitationSender.studentFbId;
23                    if(sr != localStorage.getItem('fbId'))
24                      fList.push(d[i].invitationSender);
25                    else
26                      fList.push(d[i].invitationRecipient);
27                  }
28                }
29                if(!fList){ return; }
30                friendsList.forEach(function(f, i){
31                  map.trigger("friend_change_" + f.studentFbId, f);
32                });
33           }); }, 10000);
34     });
35   }
36 }
```

3.3.2.5 Integração com servidor de localizações

O Quadro 6 exemplifica como são realizadas as chamadas *webservice* para o servidor de localizações. Neste exemplo, a função `registerStudentOnServer` da linha 1, recebe como parâmetro um objeto contendo as informações do usuário e solicita o serviço para registro dele no servidor. Nas linhas 3 até 8, é configurada a chamada `ajax` para o servidor com as informações do tipo de requisição, o endereço do serviço e tipo do conteúdo que será enviado. Neste caso, o formato JSON é utilizado em todos os serviços disponibilizados pelo servidor de localizações.

Quadro 6 – Chamada webservice para o servidor de localizações

```
1  function registerStudentOnServer(studentData){
2     $.ajax({
3        type : 'POST',
4        url : server + '/student/login',
5        crossDomain : true,
6        data : JSON.stringify(studentData),
7        dataType : 'json',
8        contentType: 'application/json; charset=utf-8',
9        success : function(data){
10          pushNotification.register();
11          window.location.href = "interacao.html";
12       }
13    });
14 }
```

O código fonte do Quadro 7 representa o mapeamento no servidor de localizações para a chamada *webservice* realizada pelo código do Quadro 6. Nas linhas 1 e 4 é possível perceber que a notação `@Path` indica que o método `login` da classe `InteracaoService` deve ser executado após a chamada para o endereço `/student/login`. A notação `@Consumes` da linha 5, está preparando o método para receber um objeto JSON e que deve convertê-lo para o objeto do tipo `Student`. Na linha 8, através do método `getStudentByFacebookId` da classe `StudentDao`, é realizada uma pesquisa no banco de dados para saber se o usuário já está registrado. Se essa informação for falsa, um novo registro de usuário será gravado, caso contrário apenas uma atualização das informações será realizada.

Quadro 7 – Mapeamento dos serviços no servidor de localizações

```
1  @Path("/student")
2  public class InteracaoService {
3     @POST
4     @Path("/login")
5     @Consumes(MediaType.APPLICATION_JSON)
6     public void login(Student student) throws Exception{
7        List<Student> studentList =
8  new StudentDao().getStudentByFacebookId(student.getStudentFbId());
9        if (studentList.isEmpty() || studentList == null){
10          new StudentDao().createStudent(student);
11       } else {
12          new StudentDao().updateStudent(student);}}}
```

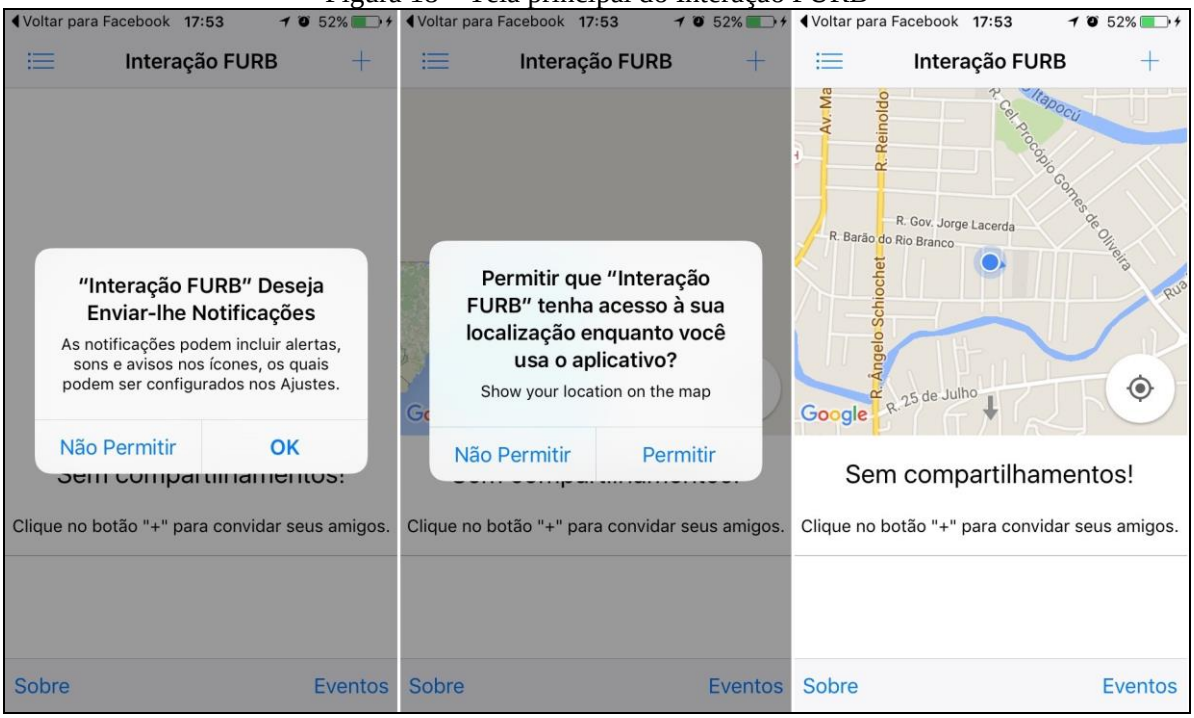
3.3.3 Operacionalidade da implementação

Nesta seção é apresentada a operacionalidade do aplicativo desenvolvido, através da interface gráfica disponível para o usuário navegar. As imagens abaixo foram obtidas de dois iPhone 5 (A e B) com o sistema operacional iOS 9.1 e mostram a interação e o compartilhamento de localização entre os dispositivos. Para iniciar os procedimentos, o iPhone A acessa a tela inicial do aplicativo conforma mostra a Figura 17. Nesta tela há um botão chamado `Entrar com Facebook`, que permite que o usuário realize sua autenticação através desta rede social.

Figura 17 – Autenticação via Facebook

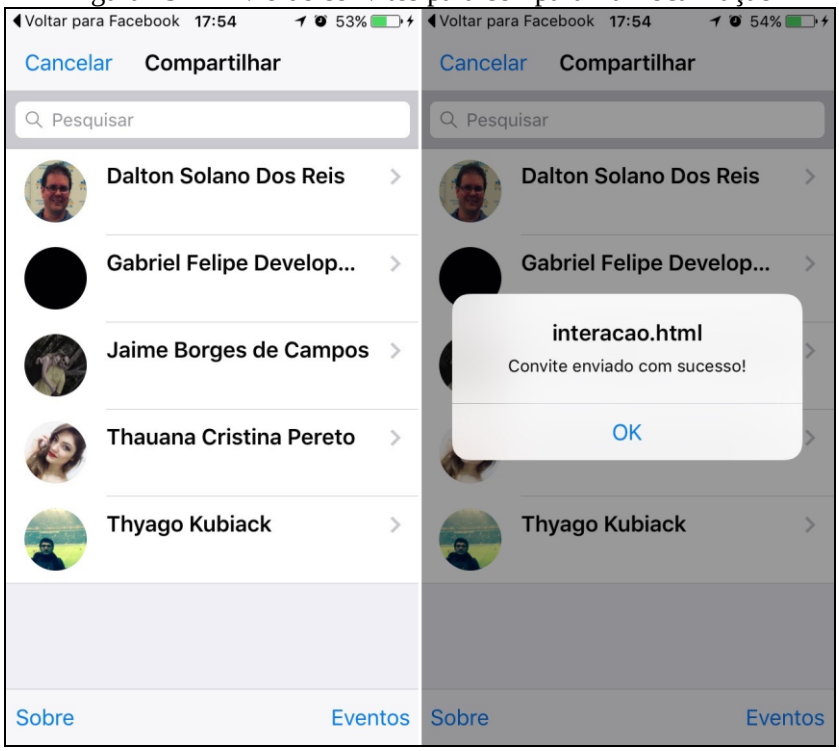
Após realizar a autenticação via Facebook e entrar na tela principal, o aplicativo solicita ao usuário permissão para receber notificações e ter acesso às informações de localização do dispositivo através dos dados fornecidos pela antena de GPS. Quando este último item for permitido, um ponto azul representando a posição atual do usuário aparece no mapa. Este ponto é atualizado dinamicamente de acordo com a movimentação do usuário, assim como apresentado na Figura 18. Ainda nesta tela uma mensagem indica que o usuário não possui compartilhamentos de localização ativos.

Figura 18 – Tela principal do Interação FURB



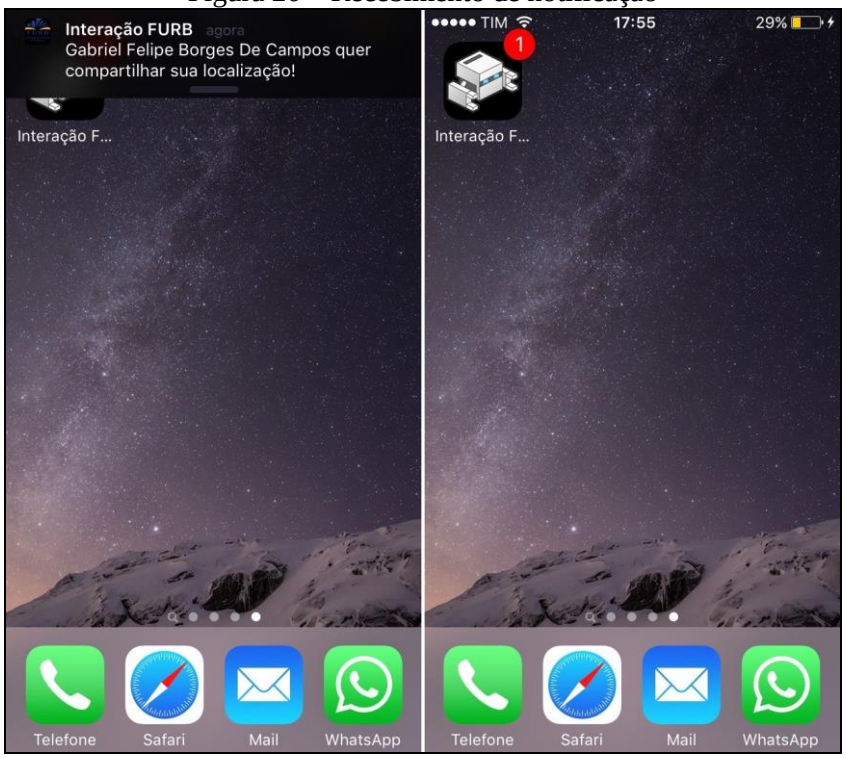
Seguindo as instruções fornecidas pelo aplicativo, o usuário deve clicar no botão +, disponível do canto superior direito, para enviar convites de compartilhamento de localização aos seus amigos. Conforme a Figura 19, após entrar na tela de compartilhamento, é carregada uma lista de amigos que também possuem o aplicativo associado em suas contas do Facebook. Neste caso, foi enviado um convite para Thauana Cristina Pereto detentora do iPhone B.

Figura 19 – Envio de convites para compartilhar localização



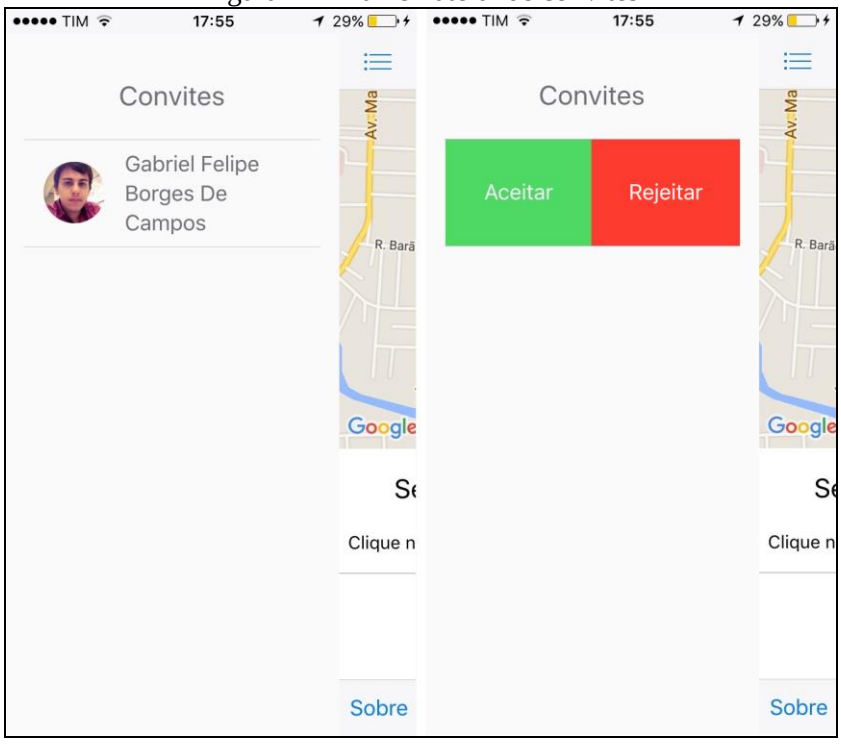
A Figura 20 mostra o recebimento da notificação no iPhone B, alertando ao usuário que um amigo lhe enviou um novo contive para compartilhar sua localização.

Figura 20 – Recebimento de notificação



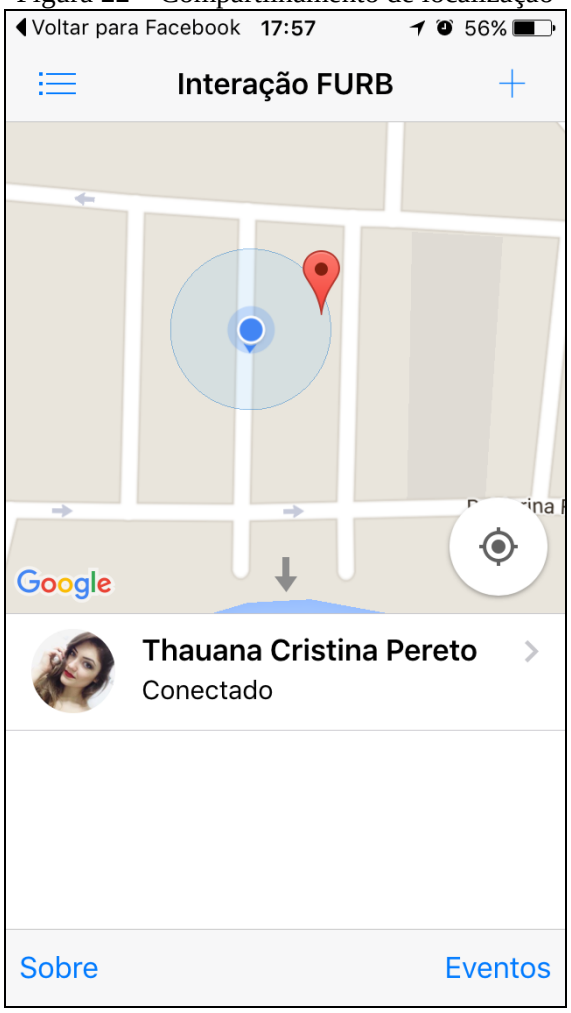
Quando o usuário convidado entra no aplicativo através do iPhone B e acessa o painel lateral, pode visualizar seus convites recebidos assim como na Figura 21. O usuário possui as opções de aceitar ou rejeitar a solicitação de compartilhamento.

Figura 21 – Painel lateral de convites



Caso o convite seja aceito, ambos os dispositivos passam a compartilhar suas localizações. Na Figura 22, é possível visualizar o compartilhamento de localização entre os amigos e conforme a movimentação deles, o marcador vermelho indicará sua posição atual. Neste caso, o iPhone A estava visualizando a posição de Thauana no mapa através do marcador vermelho. Nos cantos inferiores, direito e esquerdo, existem dois *links*. O primeiro deles `Sobre`, apresenta as informações sobre o aplicativo e o segundo `Eventos` abre uma lista dos eventos e informações registradas pela universidade.

Figura 22 – Compartilhamento de localização



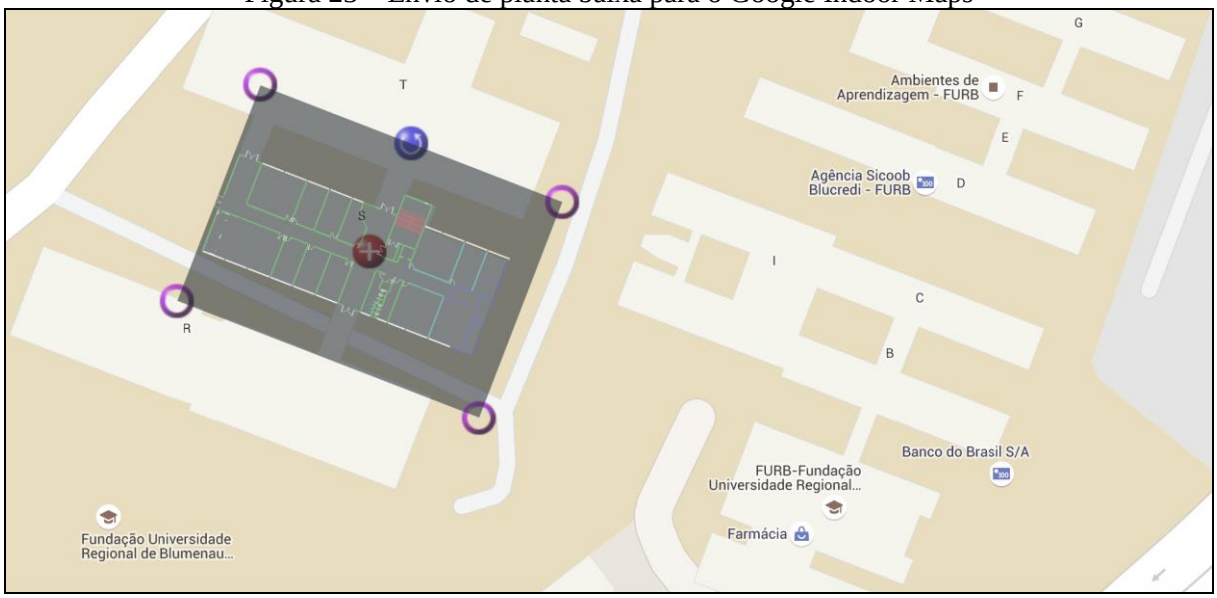
3.4 RESULTADOS E DISCUSSÕES

O principal objetivo deste trabalho era a disponibilização do trabalho desenvolvido por Kestring (2014) a partir da plataforma PhoneGap para utilização no evento Interação FURB. A intenção era criar uma nova experiência aos visitantes do evento, facilitando sua locomoção pelo campus da universidade e permitindo a identificação das salas de aula, laboratórios e auditórios através do Google Indoor Maps. Este aplicativo deveria estar integrado a rede social Facebook, aos mapas do Google Maps e suportar o recebimento de notificações

enviadas pelo serviço de notificações correspondente à plataforma do usuário (APNS, GCM e MPN). A função para o compartilhamento de localização entre os contatos do Facebook do usuário era outro requisito e que foi implementado com o auxílio do servidor gerenciador de localizações. Este servidor foi construído na linguagem de programação Java e permitiu que as coordenadas dos usuários fossem armazenadas e atualizadas constantemente de acordo com a movimentação do usuário. Através de *webservice* foram disponibilizadas as funções para registro do usuário no servidor, criação de convites de compartilhamento de localização e busca das coordenadas geográficas dos amigos.

Para que fosse possível auxiliar os visitantes a encontrar com facilidade os locais da universidade através do aplicativo, seria necessário enviar as plantas baixas de todos os blocos do campus para o Google Indoor Maps. Conforme mostra Figura 23 e o Apêndice B, durante o desenvolvimento deste trabalho, foi enviado para o site da Google a planta baixa do bloco S da FURB. Porém até o presente momento a planta baixa padronizada ainda não foi publicada no Google Maps, impossibilitando a visualização do espaço interno deste local.

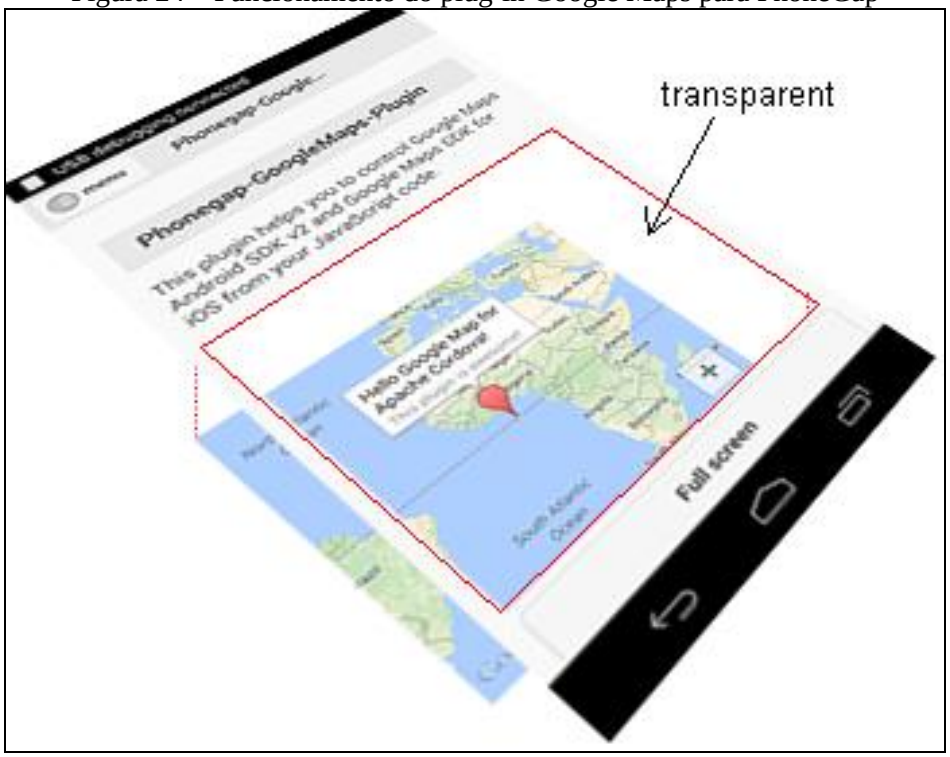
Figura 23 – Envio de planta baixa para o Google Indoor Maps



O *framework* Framework7, permitiu a criação de um aplicativo desenvolvido no padrão *web* com visual muito semelhante aos aplicativos nativos da plataforma iOS. Na Figura 8 do aplicativo Buscar Meus Amigos é possível perceber esta semelhança se comparado ao aplicativo Interação FURB da Figura 22. Porém, houve um problema na utilização deste *framework* quando integrado ao Google Maps. A Figura 24 mostra como o *plug-in* do Google Maps para PhoneGap cria o mapa no aplicativo. O mapa utiliza uma *view* nativa, ou seja, seu conteúdo não é renderizado no HTML. Porém, para que seja possível

visualizar o mapa dentro do conteúdo HTML, é criada uma espécie de janela em uma determinada área do aplicativo que está direcionada para o conteúdo do mapa.

Figura 24 – Funcionamento do plug-in Google Maps para PhoneGap



Fonte: MapsPlugin (2015).

O problema ocorre quando a lista de amigos, localizada abaixo do mapa, ultrapassa o limite da área de visualização e habilita o *scroll* para o usuário. Quando o *scroll* é utilizado, a lista de amigos rola por cima do mapa e não por de baixo como deveria ocorrer. Através de técnicas mais avançadas de desenvolvimento *web*, este problema poderia ser corrigido.

Inicialmente o aplicativo vinha sendo testado apenas com *smartphones* e *tablets* da plataforma iOS. Aproveitando a vantagem de desenvolver um aplicativo multiplataforma, foi realizado o *deploy* do Interação FURB para Android. Durante a criação do arquivo APK (extensão de aplicativo Android) para instalação no dispositivo móvel, foram encontrados problemas de conflito entre as dependências utilizadas pelos *plug-ins* instalados. Ao abrir o projeto gerado com o PhoneGap no ambiente de desenvolvimento Android Studio, foi solicitada a integração com ferramenta Gradle. Esta ferramenta faz o gerenciamento e controle de dependências dos projetos Android. Após finalizar a integração, o Gradle passou a quebrar o *build* pelo fato de que duas versões diferentes de uma mesma dependência estavam sendo utilizadas no projeto. Para tratar este problema, foi realizada uma configuração para forçar o uso de uma única versão desta dependência. Porém esta solução provocou novos erros de compilação, sendo necessária a revisão dos *plug-ins* utilizados e a busca por outras alternativas.

3.4.1 Quadro de comparação entre os trabalhos

No Quadro 8, estão sendo comparados os trabalhos correlatos apresentados na seção 2.4, ao trabalho desenvolvido. Os trabalhos que possuírem as características descritas na primeira coluna, estão marcados com “X”.

Quadro 8 – Comparação entre os trabalhos correlatos e proposto

	KESTRING (2014)	KUEHL (2012)	APPLE (2015)	Interação FURB
geolocalização de usuários	X	X	X	X
interface gráfica com mapa	X		X	X
cálculo de proximidade		X	X	
envio de notificações	X		X	X
compartilhamento de localização	X		X	X
integração com rede social				X
plataforma suportada	Android	Android	iOS	Híbrido

Conforme as características comparadas no Quadro 8, é possível perceber que todos os trabalhos fazem uso da geolocalização para localização dos usuários. Porém, apenas os trabalhos de Kuehl (2012) e Apple (2015) realizam cálculo de proximidade em relação a outros usuários e a pontos de referência no mapa. Esta é uma funcionalidade interessante pois permite o envio de notificações para diversas finalidades, como por exemplo, avisar o usuário sobre um evento que está ocorrendo próximo a sua localização. A maioria dos trabalhos apresentados possuem um mapa para visualização e referência do usuário em relação a sua posição atual. Apenas o trabalho de Kuehl (2012) não permite o compartilhamento de localização entre amigos e não possui suporte para o recebimento de notificações. As principais diferenças entre o aplicativo de Kestring (2014) e o Interação FURB está no suporte à multiplataforma e na integração com a rede social Facebook. O trabalho de Kestring (2014) foi desenvolvido exclusivamente para a plataforma Android, enquanto o Interação FURB foi desenvolvido com o *framework* PhoneGap permitindo a construção de um aplicativo híbrido. Porém, problemas relacionados a conflitos de dependências, impediram a geração do arquivo de instalação APK para o sistema Android. O Interação FURB, suporta o recebimento de notificações seja qual for o sistema operacional utilizado pelo usuário, ou seja, ele pode receber notificações dos serviços APNS, GCM e MPN, entre outras.

4 CONCLUSÕES

O objetivo de desenvolver um aplicativo multiplataforma que permitisse o compartilhamento de localização entre os visitantes do evento Interação FURB, auxiliado em sua movimentação e permitindo o recebimento de notificações sobre eventos e novas atualizações, foi parcialmente atendido. Os testes de integração do aplicativo com a rede social Facebook, com os mapas do Google Maps e com os servidores de localização e notificações, foram bem-sucedidos na plataforma iOS. Porém o mesmo não ocorreu para a plataforma Android. Na tentativa de gerar o arquivo APK (extensão dos aplicativos Android), ocorreu um erro ocasionado pelo conflito entre as dependências utilizadas pelos *plug-ins* do PhoneGap, desta forma, impossibilitando a instalação do aplicativo nos dispositivos com o sistema Android e sendo necessário testar outras alternativas de *plug-ins*.

Apesar dos problemas encontrados na compilação do código para a plataforma Android, a utilização do PhoneGap e seus *plug-ins* para a criação de aplicativos móveis, mostrou-se eficaz. A utilização do *framework* Framework7 para a criação da interface gráfica aprimorou a experiência do usuário ao proporcionar o visual de um aplicativo nativo, diminuindo a sensação de estar navegando em uma página *web*. Os *plug-ins* para PhoneGap, desenvolvidos pela comunidade, também proporcionaram agilidade de integração com os serviços de terceiros. Porém, não é verdade dizer que para o desenvolvimento de aplicativos multiplataforma não é necessário conhecer as tecnologias móveis nativas. Para fazer com que o aplicativo híbrido execute nas plataformas nativas, é necessário conhecer ao menos o básico da estrutura e funcionamento cada tecnologia. Durante o desenvolvimento deste aplicativo, diversas vezes foi necessário adicionar novas configurações ao arquivo de configuração `plist` da plataforma iOS. Devido as atualizações ocorridas no sistema operacional desta plataforma, alguns dos *plug-ins* utilizados não estavam preparados para suportar as atualizações, exigindo que as configurações fossem executadas manualmente.

Para concluir, outro ponto interessante deste trabalho foi a utilização do servidor de notificações AeroGear. Conforme os testes realizados, o mesmo se mostrou bastante estável para envio de notificações através dos serviços APNS e GCM. O interessante deste servidor é o fato de ele possuir um *plug-in* para a plataforma PhoneGap, facilitando a integração do aplicativo com este serviço. Inicialmente tentou-se criar uma instância do AeroGear através do site OpenShift, porém a mesma encontra-se com um bug e faz com que a servidor retorne o erro *Internal Server Error* 500. Desta forma, foi criado uma instância local para uso deste serviço.

4.1 EXTENSÕES

Algumas das extensões possíveis para este trabalho são:

a) realizar a integração dos equipamentos Beacons para aprimorar a geolocalização em espaços interno;
b) implementar a funcionalidade para continuar atualizando a posição do usuário mesmo quando o aplicativo esteja rodando em segundo plano;
c) implementar funcionalidade de cálculo de proximidade, permitindo identificar quando o usuário chegou ou saiu de algum lugar;
d) integrar o serviço de notificações AeroGear com os sistemas da FURB;
e) aprimorar o algoritmo para identificação do *status* dos usuários (online/offline);
f) aprimorar os marcadores utilizados para identificação dos amigos no mapa, utilizando a foto do perfil de cada amigo;
g) finalizar o envio das plantas baixas de todos os blocos da FURB para o Google Indoor Maps.

REFERÊNCIAS

ALCANTARA, Carlos A. A.; VIEIRA, Anderson L. N. **Tecnologia móvel**: uma tendência, uma realidade. Juiz de Fora, 2011. Disponível em: <http://arxiv.org/abs/1105.3715>. Acesso em: 05 abr. 2015.

APPLE. **Buscar Meus Amigos**. [S. l.], 2015. Disponível em: <https://www.apple.com/br/apps/find-my-friends>. Acesso em: 14 abr. 2015.

BLOG OPENSHIFT. **Mobile Push Simplified With The AeroGear Push Server On OpenShift**. [S. l.], 2013. Disponível em: <https://blog.openshift.com/mobile-push-simplified-with-the-aerogear-push-server-on-openshift>. Acesso em: 25 nov. 2015.

CALAZANS, José. **Como as redes sociais vão provocar a expansão das conexões móveis no Brasil**. São Paulo, 2012. Disponível em: <http://www.ibope.com.br/pt-br/conhecimento/artigospapers/Paginas/Com-as-redes-sociais-vão-provocar-a-expansão-das-conexões-móveis-no-Brasil.aspx>. Acesso em: 19 maio. 2015.

FACEBOOK DEVELOPERS. **Facebook DSK for iOS**. [S. l.], 2015. Disponível em: <https://developers.facebook.com/docs/ios>. Acesso em: 12 abr. 2015.

GOOGLE DEVELOPERS. **Documentação de desenvolvedor do Google Maps**. [S. l.], 2015. Disponível em: <https://developers.google.com/maps/documentation/?hl=pt-br>. Acesso em: 12 abr. 2015.

GRONER, Loiane. **Tutorial:** Notificações push no iOS. São Paulo, 2013. Disponível em: <http://www.loiane.com/2013/07/tutorial-notificacoes-push-no-ios>. Acesso em: 11 abr. 2015.

GUINTER, Marcelo. **Desenvolvimento Mobile Multiplataforma com Phonegap**. São Paulo, 2011. Disponível em: < http://www.mobiltec.com.br/blog/index.php/desenvolvimento-mobile-multiplataforma-com-phonegap/>. Acesso em: 25 nov. 2015.

KESTRING, Bruno A. **Sistema móvel na plataforma Android para compartilhamento de geolocalização usando mapas e notificações da Google**. 2014. 72 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

KIRKPATRICK, David. **O efeito Facebook**: Os bastidores da história da empresa que está conectando o mundo. Tradução Maria Lúcia de Oliveira. Rio de Janeiro: Intrínseca, 2011.

KUEHL, Cesar A. **Protótipo de sistema móvel na plataforma Android para compartilhamento de arquivos e mensagens entre dispositivos baseado em proximidade geográfica**. 2012. 115 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

LINKEDIN. **Notificações push em dispositivos móveis?** [S. l.], 2014. Disponível em: <https://ajuda.linkedin.com/app/answers/detail/a_id/31371/~/notifica%C3%A7%C3%B5es-push-em-dispositivos-m%C3%B3veis---resumo>. Acesso em: 25 nov. 2015.

MONICO, João Francisco G. **Posicionamento pelo NAVSTAR-GPS**: Descrição, fundamentos e aplicações. São Paulo: UNESP, 2000.

MAPSPLUGIN. **How does the plugin work?** [S. l.], 2015. Disponível em: <https://github.com/mapsplugin/cordova-plugin-googlemaps/wiki/Map>. Acesso em: 25 nov. 2015.

MICROSOFT. **O que é AGPS?** [S. l.], 2015. Disponível em: <http://www.microsoft.com/pt-br/celulares/suporte/faq/?action=singleTopic&topic=FA114913>. Acesso em: 12 abr. 2015.

O DIA. **Vício em celular atrapalha vida familiar e profissional**. Rio de Janeiro, 2012. Disponível em: <http://odia.ig.com.br/portal/cienciaesaude/v%C3%ADcio-em-celular-atrapalha-vida-familiar-e-profissional-1.528918>. Acesso em: 05 abr. 2015.

PHONEGAP. **PhoneGap Explained Visually** [S. l.], 2012. Disponível em: <http://phonegap.com/2012/05/02/phonegap-explained-visually/>. Acesso em: 25 abr. 2015.

SABOIA, Juliana; VARGAS, Patrícia L. de; VIVA, Marco Aurélio de A. O uso dos dispositivos móveis no processo de ensino e aprendizagem no meio virtual. **Revista Cesuca Virtual**, Cachoeirinha, v. 1, n. 1, jul. 2013. Disponível em: <http://ojs.cesuca.edu.br/index.php/cesucavirtual/issue/view/24>. Acesso em: 05 abr. 2015.

SOARES, Francisco G. Mobilidade novas tecnologias tornam mais próxima relação entre governos e sociedade. **Guia Das Cidades Digitais**, Rio de Janeiro, set. 2012. Disponível em: <http://www.guiadascidadesdigitais.com.br/site/pagina/mobilidade-novas-tecnologias-tornam-mais-prxima-relao-entre-governos-e-sociedade>. Acesso em: 05 abr. 2015.

TECMUNDO. **Por que as notificações de aplicativos são importantes?** São Paulo, 2015. Disponível em: < http://www.tecmundo.com.br/apps/43525-por-que-as-notificacoes-de-aplicativos-sao-importantes-.htm>. Acesso em: 05 nov. 2015.

TEIXEIRA, Fabricio. **Tudo o que você precisa saber para começar a brincar com iBeacons**. São Paulo, 2014. Disponível em: <http://arquiteturadeinformacao.com/ux-em-espacos-fisicos/tudo-o-que-voce-precisa-saber-para-comecar-a-brincar-com-ibeacons>. Acesso em: 05 abr. 2015.

UNIVERSIDADE REGIONAL DE BLUMENAU. **Interação FURB**. Blumenau, 2015. Disponível em: <http://www.furb.br/web/4263/interacao-furb/interacao-furb/interacao-furb>. Acesso em: 05 abr. 2015.

APÊNDICE A – Instalação do servidor de notificações AeroGear Unified Push Server

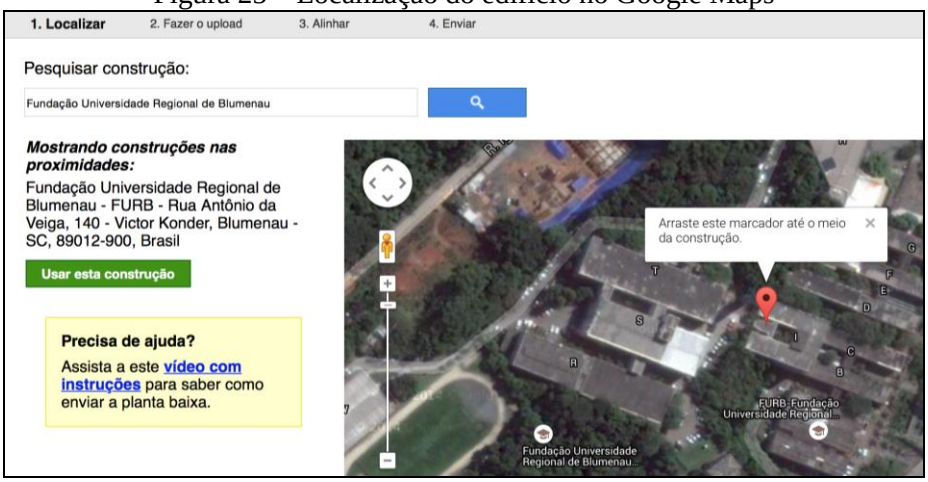
Este apêndice apresenta os comandos executados no terminal do sistema operacional OS X El Capitan para instalação e configuração do servidor de notificações AeroGear no WildFly 9 integrado ao banco de dados MySQL.

```
$cp -r <pathto>/aerogear-unifiedpush-server-
1.1.0/databases/src/main/resources/modules/com <pathto>/wildfly-
9.0.1/modules/

$mvn dependency:copy -Dartifact=mysql:mysql-connector-java:5.1.18 -
DoutputDirectory=<pathto>/wildfly-9.0.1/modules/com/mysql/jdbc/main

$mysql -u root

$create database unifiedpush default character set "UTF8" default collate u
tf8_general_ci;

$create database keycloak default character set "UTF8" default collate utf8
_general_ci;

$create user 'unifiedpush'@'localhost' identified by 'unifiedpush';

$GRANT SELECT,INSERT,UPDATE,ALTER,DELETE,CREATE,DROP,INDEX ON unifiedpush.*
 TO 'unifiedpush'@'localhost';

$GRANT SELECT,INSERT,UPDATE,ALTER,DELETE,CREATE,DROP ON keycloak.* TO 'unif
iedpush'@'localhost';

$mysql -u unifiedpush -p unifiedpush

$sudo <pathto>/wildfly-9.0.1/bin/standalone.sh --server-config=standalone-
full.xml -b localhost

$sudo vi <pathto>/mysql-database-config-wildfly.cli
connection-url="jdbc:mysql://localhost:3307/unifiedpush?
connection-url="jdbc:mysql://localhost:3307/keycloak?

$sudo <pathto>/wildfly-9.0.1/bin/jboss-cli.sh –file=<pathto>/aerogear-
unifiedpush-server-1.1.0/databases/mysql-database-config-wildfly.cli

$sudo vi <pathto>/liquibase-mysql-example.propertie
url=jdbc:mysql://localhost:3307/unifiedpush

$ cp <pathto>/aerogear-unifiedpush-server-1.1.0/migrator/unifiedpush-
migrator-1.1.0/liquibase-mysql-example.properties <pathto>/aerogear-
unifiedpush-server-1.1.0/migrator/unifiedpush-migrator-
1.1.0/liquibase.properties

$ ./bin/ups-migrator update

$ cp <pathto>/aerogear-unifiedpush-server-1.1.0/servers/unifiedpush-auth-
server.war <pathto>/wildfly-9.0.1/standalone/deployments

$ cp <pathto>/aerogear-unifiedpush-server-1.1.0/servers/unifiedpush-server-
wildfly.war <pathto>/wildfly-9.0.1/standalone/deployments
```

APÊNDICE B – Envio de plantas baixas para o Google Indoor Maps

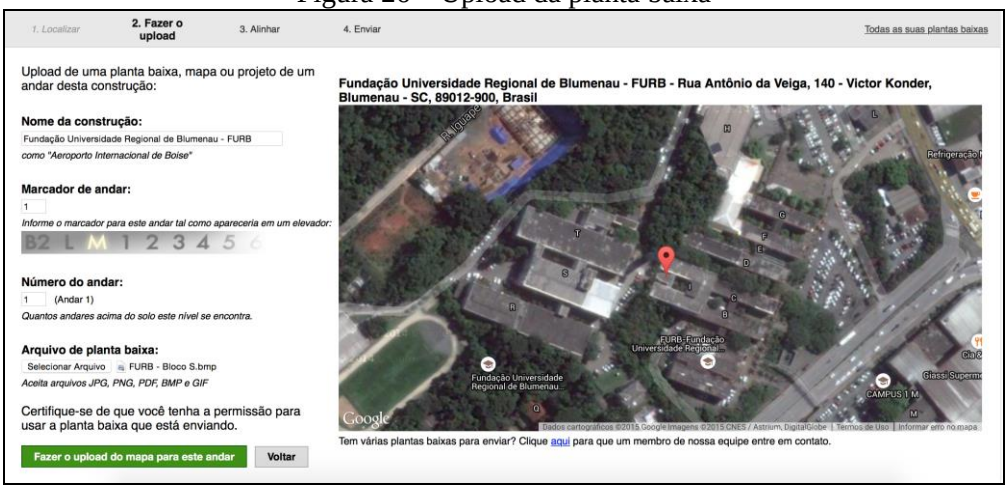
Neste apêndice encontram-se os passos para a publicação de plantas baixas no Google Indoor Maps. Estes passos são divididos em quatro etapas: localizar o edifício no mapa, realizar o *upload* da planta baixa, alinhar planta baixa com o mapa e publicar. A Figura 25 representa a primeira etapa, onde foi pesquisado pelo edifício da FURB e ajustado o marcador para o centro da construção.

Figura 25 – Localização do edifício no Google Maps



Na segunda etapa, como mostra a Figura 26, foi configurado o nome oficial para construção, informado o marcador do andar, o número do andar referente a planta baixa e selecionado a imagem contendo a planta baixa.

Figura 26 – Upload da planta baixa



Após a realizar o *upload* da planta baixo, na terceira etapa foi necessário alinhar as extremidades da imagem enviada com a construção disponível no mapa, conforme Figura 27. Neste momento percebeu-se a falta de uma única planta baixa contendo todos os blocos do campus da universidade, separada apenas pelos andares.

Figura 27 – Alinhamento da planta baixa

A última etapa (Figura 28) é o envio da planta baixa alinhada ao mapa para publicação. Neste momento é necessário aguardar a criação e disponibilização do Indoor Map pela Google.

Figura 28 – Publicação da planta baixa