Universidade Federal
do Rio de Janeiro
Escola Politécnica

REDE SOCIAL BASEADA EM EVENTOS DESENVOLVIDA PARA O SISTEMA OPERACIONAL ANDROID

João Guilherme Mattos de Oliveira Santos

Projeto de Graduação apresentado ao Curso de Engenharia Eletrônica e de Computação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientador: Flávio Luis de Mello, DSc.

Rio de Janeiro

Dezembro de 2015

REDE SOCIAL BASEADA EM EVENTOS DESENVOLVIDA PARA O SISTEMA OPERACIONAL ANDROID

João Guilherme Mattos de Oliveira Santos

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO DE ENGENHARIA ELETRÔNICA E DE COMPUTAÇÃO DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO ELETRÔNICO E DE COMPUTAÇÃO

Autor:

_____

João Guilherme Mattos de Oliveira Santos

Orientador:

_____

Prof. Flávio Luis de Mello, DSc.

Examinador:

_____

Prof Sergio Barbosa Villas-Boas, PhD.

Examinador:

_____

Prof. Aloysio de Castro Pinto Pedrosa, DSc.

Rio de Janeiro – RJ, Brasil

Dezembro de 2015

Mattos de Oliveira Santos, João Guilherme

Rede Social Baseada em Eventos para o Sistema Operacional Android/ João Guilherme Mattos de Oliveira Santos. – Rio de Janeiro: UFRJ/ Escola Politécnica, 2015. X, 75 p.: il.; 29,7 cm.

Orientador: Flávio Luis de Mello, DSc.

Projeto de Graduação – UFRJ/ Escola Politécnica/ Curso de Engenharia Eletrônica e de Computação, 2015. Referencias Bibliográficas: p. 74-75. 1. Introdução. 2. Redes Sociais. 3. Aplicações em Dispositivos Móveis. 4. Implementação e Resultados. 5. Conclusão. I. Luis de Mello, Flávio DSc. II. Universidade Federal do Rio de Janeiro, Escola Politécnica, Curso de Engenharia Eletrônica e de Computação. III. Rede Social Baseadas em Eventos.

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Escola Politécnica – Departamento de Eletrônica e de Computação
Centro de Tecnologia, bloco H, sala H-217, Cidade Universitária
Rio de Janeiro – RJ CEP 21949-900

Resumo do Projeto de Graduação apresentado à Escola Politécnica/ UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro Eletrônico e de Computação.

Rede Social baseada em Eventos desenvolvida para o sistema operacional Android

João Guilherme Mattos de Oliveira Santos

Dezembro/2015

Orientador: Flávio Mello

Curso: Engenharia Eletrônica e de Computação

O trabalho tem como objetivo apresentar os conceitos envolvidos no desenvolvimento de uma rede social baseada em eventos para o sistema operacional Android. Além disso, houve a preocupação de empregar técnicas para tornar a aplicação eficiente do ponto de vista energético. Inicialmente, é apresentada uma ideia geral da solução através de um framework explicando a arquitetura. Em seguida, so diagramas de transição de estados e classe trazem outras perspectivas para o entendimento do projeto. Por fim, para que se obtenha uma visão mais geral e sistêmica do software, é apresentado o diagrama de componentes, que mostra como os módulos do cliente e do servidor interagem entre si.

Palavras-chave: Android, Web Services, Contexto,

AGRADECIMENTO

Primeiramente, queria agradecer a minha família por estar sempre disposta a me ajudar durante todo desenvolvimento deste trabalho. Em especial, meu pai, que, além ter me ajudado com alguns aspectos tecnicos, nunca deixou que eu me acomodasse sempre me pressionando a dar o meu melhor. Também gostaria de agradecer ao meu coordenador, Flávio Mello, que durante este projeto sempre se dispôs a me ajudar com a edição do texto e o desenvolvimento do projeto a nível mais macro. Enfim, gostaria de agradecer a todos os professores da Universidade Federal do Rio de Janeiro com quem tive a oportunidade de aprender e evoluir, não somente no aspecto tecnico, mas também no pessoal e profissional e também a todos os amigos que fiz durante esta jornada na UFRJ.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas

API – Application Program Interface
VR – Virtual Reality
GPS – Global Positioning System
A-GPS – Assisted Global Positioning System
PoI – Point of Interest
P2P – Peer to Peer
CPU – Central Processing Unit
UI – User Interface
GUI – Graphic User Interface
XML – Extension Markup Language
SQL – Standard Query Language
VoIP – Voice over Internet Protocol
MAC – Media Access Control
GSM – Global System for Mobile Communications
IDE – Integrated Development Environment
HTTP – Hypertext Transfer Protocol
TCP/IP – Transmission Control Protocol / Internet Protocol
MVC – Model View Controller
REST – Representational State Transfer
SOA – Service Oriented Architecture
JSON – JavaScript Object Notation
AWS – Amazon Web Services
JVM – Java Virtual Machine
RDD – Resilient Distributed Data

Capítulo 1
Introdução

1.1 – Tema

Este projeto trata de redes sociais baseadas em eventos. Sob esta ótica, o problema a ser resolvido é o desenvolvimento de uma aplicação móvel para o sistema operacional Android, que apresente os conceitos estudados.

1.2 – Delimitação

Este projeto contém duas condições de contorno. Inicialmente, trata-se de um aplicativo móvel para sistema operacional Android, e, portanto, não está disponível para iPhones ou celulares Nokia, pois utilizam outros sistemas operacionais, respectivamente: iOS e Windows Phone. O segundo ponto limitador do projeto é o fato dos usuários necessitarem de contas no Facebook. Como o acesso é feito através desta rede social, é necessário que o usuário possua uma conta nela.

1.3 – Justificativa

As redes sociais têm evoluído rapidamente. Atualmente, essas aplicações surgem o tempo inteiro com as mais diversas finalidades. Este projeto se dedica a estudar esta área da computação não somente pelo seu dinamismo, mas também pela versatilidade de conceitos que elas abrangem. Uma aplicação móvel voltada para redes sociais agrega os seguintes conceitos:

1. Compartilhamento eficiente de mídias e streaming em tempo real;
2. Geo-localização do usuário ou de outros agentes que participam da aplicação, permitindo que se infira contexto a esses padrões de localização;
3. Modelagens complexas de banco de dados baseada em grafos, com mais de um agente, em muitos casos, e diferentes tipos de link entre esses agentes, como são apresentados no segundo capítulo;
4. Escalabilidade da aplicação, ou seja, como impedir que o alto volume de tráfego não comprometa o funcionamento do servidor.

Esses conceitos são fundamentais para o desenvolvimento de aplicações móveis. Saber explorar os sensores do smartphone corretamente agrega bastante valor à maioria das aplicações. Conseguir gerenciar o volume de dados, que essas aplicações levantam, também pode ser aplicado em diversos outros casos de desenvolvimento. Por fim, promover um mecanismo de troca de mídias entre os usuários é essencial para este paradigma da computação. Logo, o conhecimento desses conceitos é de extrema importância para quem deseja desenvolver aplicações móveis.

1.4 – Objetivos

O objetivo deste trabalho é a criação de uma aplicação móvel que permita aos integrantes de uma rede social o compartilhamento de eventos sociais baseados na informação de localização geográfica e temporal. Neste sentido, os objetivos específicos do projeto são:

- Gerenciar os eventos já criados em outras redes sociais que estão integradas com a aplicação, especificamente o Facebook, nesta versão.
- Prover funcionalidades básicas de uma rede social, como adição de amigos e criação de eventos pela rede.
- Exibir todos os eventos criados em cima do Google Maps, como balões nas cores azul ou rosa.
- Exibir uma lista com os eventos do usuário e suas informações fundamentais, como: nome, data, hora, foto do criador do evento e status do usuário no evento (informação sobre sua presença).
- Exibir uma tela para visualização de todos os detalhes do evento e interação do usuário com o evento.

1.5 – Metodologia

O desenvolvimento desta aplicação se iniciou pela análise do estado-da-arte das áreas de estudo presentes em redes sociais. Mais especificamente, os conceitos de Redes Sociais Baseadas em Eventos e em Localização foram trabalhados para se entender melhor o que poderia ser feito nesta aplicação. Além desses estudos, também foram abordados conceitos básicos de desenvolvimento de aplicações móveis, ou seja, conceitos de usabilidade e práticas de desenvolvimento de baixo consumo energético. Ambos determinam métricas de qualidade da aplicação desenvolvida. Ainda são encontrados no texto um breve entendimento

de como o sistema operacional Android está estruturado e o estado-da-arte para sistema de posicionamento global, funcionalidade imprescindível para interação do usuário com o Google Maps, responsável pela tela principal da aplicação.

A primeira etapa do projeto foi sair do campo abstrato de uma ideia para a definição dos módulos de software necessários para o projeto, avaliação das tecnologias disponíveis para criação destes módulos e por fim adoção das ferramentas que melhor se adequam ao projeto.

Em seguida, já com um entendimento melhor do sistema a ser desenvolvido, concentrou-se em se definir um escopo de funcionalidades para o projeto, de maneira, que houvesse tempo hábil para o desenvolvimento de tudo o que foi planejado.

O passo final foi a implementação do projeto. Como apresentado com maiores detalhes no capítulo 4, foram desenvolvidos diversos diagramas para se determinar os casos de uso da aplicação, quais componentes que a solução comportaria, e como esses componentes estariam desenvolvidos internamente, mostrando algumas de suas classes e o tipo de associação entre elas.

1.6 – Descrição

O texto está dividido em cinco capítulos, incluindo esta introdução, capítulo 1. O capítulo 2 apresenta uma breve história de como as redes sociais evoluíram, além do conceito de Redes Sociais Baseadas em Eventos e Localização. Ainda é apresentado, neste capítulo, uma breve os conceitos de streaming de mídias. Em seguida, no capítulo 3, são apresentadas as peculiaridades do desenvolvimento de aplicações móveis voltadas para o Sistema Operacional Android, como a usabilidade do sistema e o desenvolvimento de aplicações de baixo consumo energético. Ainda é apresentado um estudo do estado-da-arte para sistemas de posicionamento global – ítem fundamental para a implementação do projeto – e uma análise dos componentes do Sistema Operacional Android. No capítulo 4, são descritos a implementação e os resultados obtidos com este projeto. Por último, no capítulo 5 são apresentadas as conclusões e os possíveis trabalhos futuros para o projeto.

Capítulo 2
Redes Sociais

2.1 – Definição

Redes sociais são plataformas que permitem ao usuário se conectar com demais pessoas virtualmente. Elas possibilitam a criação de uma página web, chamada de perfil, que é a sua representação na rede [1]. Essas páginas web são os pontos de conexão entre os usuários. Através delas, os usuários montam a sua rede de relacionamentos, permitindo que eles interajam de diversas formas usando: troca de mensagens, compartilhamento de fotos, posts e todas as demais funcionalidades que se encaixem em um ambiente de interação virtual.

Uma definição mais formal [1] para Redes Sociais, preconiza que se tratam de serviços baseados na Web que permitem que os seus indivíduos:

- *construam um perfil público ou semi-público dentro de um sistema;*
- *criem listas de outros usuários com quem eles compartilham uma conexão;*
- *visitem e naveguem pela sua lista de conexões e por aquelas criadas pelos outros usuários dentro do sistema.*

Um aspecto que é único nas redes sociais é o fato de permitir que os usuários criem e tornem visíveis os seus relacionamentos de forma que novos relacionamentos sejam criados espontaneamente, a partir do compartilhamento de interesses comuns. Não se trata apenas de permitir que indivíduos se relacionem com indivíduos desconhecidos, mas sim a criação de relacionamentos baseados em informações em comum [1].

Por muito tempo, as redes sociais dispunham de diversos meios de interação social, como serviços de troca de mensagens, compartilhamento de fotos, vídeos, posts etc. Com o Facebook se tornando uma plataforma hegemônica no mundo e o advento dos smartphones, essa tendência acabou. Hoje em dia, as redes sociais procuram focar em um único serviço de interação social e se utilizar das APIs existentes de demais redes – Google+, Facebook, Twitter etc. – para que o usuário não precise informar todas as suas informações e conexões novamente. Esse é o caso do Instagram, Foursquare e outras redes, que surgiram após o advento do smartphone, tornando o upload de informações ubíquo, ou seja, de qualquer lugar e a qualquer hora. Neste trabalho, o interesse está exclusivamente em um caso particular

dessas redes sociais móveis: as redes sociais baseadas em eventos, como é explicado mais adiante.

A evolução da comunicação virtual é constante. O tópico a seguir explica rapidamente como elas chegaram ao estado atual, quais as previsões para um futuro próximo e o que essas redes ainda têm a acrescentar para as vidas das pessoas.

2.2 – História das Redes Sociais

2.2.1 – Surgimento das Redes Sociais

As redes sociais, avaliadas pela definição que se conhece atualmente, surgiram por volta de 1997. A primeira a ganhar um certo grau de visibilidade se chamava SixDegree.com. Este site possibilitava os usuários a criar perfis, adicionar amigos, organizar grupos e navegar pelos perfis dos demais usuários da rede. Entretanto, os usuários iniciais (*early adopters*) criticavam que não havia muito o que se fazer no site após adicionar seus amigos. Além disso, os usuários também não queriam estar de certa forma conectados a pessoas estranhas em um site. A. Weinrich, fundador da rede, a define como a frente de seu tempo [2].

Em seguida, devido ao curto sucesso do SixDegree.com e de alguns outros da época como Classmates.com, as redes sociais se proliferaram pelos Estados Unidos. De 1997 a 2001, surgiram diversas redes, por exemplo: AsianAvenue, BlackPlanet e MiGente, que permitiam a criação de perfis sociais e profissionais. Entretanto, nenhuma delas chegou a atingir grande atenção do público na época, apesar de existirem até hoje.

Em 2002, surge a rede marcada como um dos maiores fracassos da era dotcom: Friendster. Diferentemente de outros sites de encontro, Friendster propôs uma rede social entre pessoas que tinham pelo menos um amigo em comum, assumindo que desta forma o site conseguiria criar relacionamentos entre pessoas que já se conheciam, mas não tinham a oportunidade de se encontrar. A rede rapidamente se popularizou e chegou a alcançar uma quantidade expressiva de usuários. Somente pelo boca-a-boca (word-of-mouth) atingiu algo em torno de 3.000.000 [2], o que logo levantou diversas limitações técnicas: os servidores caíam o tempo inteiro, e a experiência era comprometida. Além das dificuldades técnicas, o site enfrentou outros problemas. Como os perfis dos usuários eram bloqueados para as pessoas que não eram sequer indiretamente relacionadas, ou seja, não possuíam nenhum amigo em comum, uma série de “falsos” perfis começou a ser criado, o que irritou os fundadores do site. Em resposta, eles impuseram diversas regulações aos perfis e começaram a pensar em uma versão paga da aplicação.

Devido a estas limitações, os usuários começaram a migrar para uma nova rede que estava surgindo, o MySpace. Esta rede cresceu rapidamente, e, através dela, uma nova funcionalidade das redes atuais ganhou força: os eventos. Os usuários começaram a promover eventos por essa rede social, além disso ela permitia que os usuários compartilhassem músicas e vídeos. As demais funcionalidades eram criadas de acordo com as necessidades que os usuários demandavam à plataforma. MySpace ainda permitiu que pessoas abaixo de 18 anos entrassem na rede para ouvir músicas de suas bandas prediletas, o que alavancou ainda mais o crescimento do site.

Concomitantemente ao MySpace, em 2003, surgia outra rede social com uma proposta bem diferente das demais, o LinkedIn. Com um foco mais profissional, ele propunha conectar pessoas em um outro universo de atuação, o mercado de trabalho.

Em 2005, MySpace foi vendido por 580 milhões de dólares para a News Corporation [2]. A partir deste ponto, evidenciou-se o poder econômico a que este mercado estava atrelado. Entretanto, logo após a venda, acusações de que adultos e menores interagiam de maneira inapropriada no site, levaram a uma série de processos na justiça e, por consequência, ao seu declínio, permitindo que uma nova rede se perpetuasse, o Facebook.

2.2.2 – Consolidação das redes sociais

O Facebook surgiu como uma plataforma que conectava estudantes da mesma universidade, inicialmente em Harvard, em 2004. Aos poucos a plataforma se expandiu para outras universidades e até colégios. O sucesso desta plataforma se deve não somente às simples e novas funcionalidades que ela agregou – como o botão de like – mas também ao lançamento do Facebook Platform em 2007: uma API que permitia aos desenvolvedores criar aplicações em cima do website. Isto possibilitou o surgimento de jogos na plataforma, o que aumentou consideravelmente o tempo de acesso por usuário à rede. Em certo ponto, havia tantos aplicativos desenvolvidos na plataforma do Facebook, que se tornou necessária a criação da Facebook App Store para gerenciar todas essas aplicações.

Devido ao sucesso das redes sociais, em 2007 a Google decidiu ingressar no ramo, lançando a sua própria rede, o Google+. O diferencial da rede provinha do Hangout service, que permitia os usuários se conectarem por chats de vídeos em tempo real. O Google+, em seu primeiro mês, atraiu mais de 25 milhões de pessoas e atualmente possui cerca de 550 milhões de usuários, porém muitos não podem ser considerados usuários ativos na rede [2]. Isso se deve ao fato de que um perfil na rede Google+ é criado para cada e-mail da plataforma

Gmail da Google. Logo, muitas pessoas nem sequer sabem que possuem conta na rede social Google+.

Ao longo dos últimos anos, os smartphones evoluíram exponencialmente em termos de processamento, interatividade e tamanho. O advento desta tecnologia abriu as portas para que redes sociais não convencionais – como as que se estava acostumado na era desktop – surgissem. Redes, como Instagram, Snapchat, Foursquare e infinitas outras quebraram o paradigma de que uma rede social deveria comportar diversas funcionalidades da comunicação humana de modo a se tornar hegemônica frente às demais. Essas redes começaram a se especializar em um único meio de comunicação, fosse ele: o compartilhamento de vídeos privados, fotos, localização corrente do usuário ou mesmo promover encontros com base na localização das pessoas. O usuário agora estava conectado em infinitas redes diferentes ao invés de utilizar uma única rede para imputar informações de sua vida social.

As redes sociais, que detinham a hegemonia do setor, ou seja, Facebook, Twitter, Google+ e LinkedIn, ao invés de se proteger desta competitividade, liberaram APIs para que os programadores de dispositivos móveis pudessem desenvolver suas aplicações com mais rapidez. Assim, os aplicativos de dispositivos móveis - que por sinal surgiam cada vez mais - estavam geralmente integrados a essas plataformas, sem a necessidade de que a aplicação detivesse uma base de dados exclusiva para autenticação de usuários. Eles acessavam a plataforma através de suas contas nas redes sociais de maior porte. Isso facilitou em muito o surgimento de novas plataformas, que conforme cresciam, recebiam propostas das redes maiores. O Instagram, por exemplo, foi adquirido pelo Facebook. O mesmo não se pode dizer do Snapchat, que recusou proposta bilionária desta mesma empresa.

2.2.3 – Realidade Virtual e Aumentada: O futuro das redes sociais

As redes sociais têm se desenvolvido em um ritmo extremamente elevado. A dependência e quantidade de usuários interconectadas por essas redes aumenta a uma taxa de 200 milhões de pessoas por ano. Em 2016, espera-se um total de mais de 2 bilhões de pessoas conectadas [3]. Assim, as redes têm de se manter em constante evolução de modo a melhorar a interação do usuário com os dispositivos para manter sua atual base ativa e atrair mais pessoas para este universo.

Acredita-se que a próxima evolução disruptiva dessas redes esteja atrelada aos famosos wearables, ou seja, dispositivos inteligentes vestíveis capazes de conectar pessoas de

alguma forma. Dentro desse campo, especula-se que se esteja próximo do surgimento de mais um mercado bilionário no mundo: os óculos Rift. Através de Realidade Virtual e Aumentada, pretende-se imergir o usuário em um universo virtual por onde ele consiga se comunicar em tempo real com outras pessoas como se estivesse no mesmo ambiente físico. As oportunidades nesse ramo são infinitas, aplicações em áreas, como: jornalismo, educação, medicina, engenharia e muitas outras, agregarão à sociedade vantagens, que jamais se sonhou antes. Por exemplo, o acompanhamento do médico ao doente por tele-presença, o ensino escolar remoto em ambientes multi-culturais, o acompanhamento de uma partida de futebol como se estivesse nos melhores assentos do estádio etc. [1].

Cory Bergman, co-fundador do BreakingNews, esclareceu da seguinte forma a importância que tal tecnologia pode agregar ao jornalismo, quando perguntado como seria o mundo em 20 anos:

"Quando ocorrer uma notícia no futuro, ela será coberta por uma multidão espectadores através de streaming de vídeo em tempo real. Esses streamings serão fundidos de modo a se formar um único vídeo 3D, que possibilitará o espectador experimentar virtualmente como seria estar no local do evento..." [4].

Recentemente, o Facebook adquiriu a empresa Oculus VR por 3.2 bilhões de dólares e vem desenvolvendo testes de aplicações virtuais inicialmente voltadas para games, como a própria empresa já fazia antes. No entanto, pretende-se explorar esta plataforma e trazer outros casos de imersão para o usuário mais adiante. O artigo de Dan Kaplan em [5] apresenta uma noção de como seria o quadro de notícias (news feed) do Facebook nesta plataforma. Segundo o texto, acredita-se em um quadro de notícias orientado a cards, em que cada card contemplaria o post de um amigo de sua rede. O usuário navegaria pela lista de cards através de um simples gesto no ar.

Assim, conclui-se a contextualização do tema abordado. Em seguida, são avaliados aspectos mais técnicos do estado-da-arte de aplicações móveis voltadas para redes sociais com foco em eventos e geo-localização dos usuários.

2.3 Características Observáveis em Redes Sociais

Os avanços nas tecnologias de aquisição de localização vêm influenciando o modo como as pessoas interagem virtualmente. Atualmente, é possível compartilhar fotos com inferência de localidade, como faz o Flickr, comentar em eventos diretamente de onde eles estão acontecendo, caso do Twitter, ou compartilhar instantaneamente sua localização como

no Foursquare. Esses são apenas alguns exemplos do poder que esta ferramenta agrega às redes sociais [6].

Além da forma direta de utilização desta tecnologia apresentada acima, um histórico de localizações de um usuário pode inferir muita informação sobre seu comportamento, interesses e relacionamento com seus amigos, ou seja, todo um contexto do usuário pode ser inferido, analisando somente seu padrão de localização [7] [8]. Como exemplo, se a aplicação sabe que o usuário está de segunda a sexta no período da manhã e tarde em um determinado local, ela pode inferir que aquela localidade representa a posição geográfica do local do seu trabalho. Sozinha, esta informação não possui muito valor, mas se observar o histórico deste mesmo usuário e constatar que majoritariamente das 9 da noite até 7 da manhã, ele se encontra em uma determinada localidade, pode-se inferir onde ele mora. Agora, é possível derivar destas informações uma aplicação simples que analisaria a cada dia qual a melhor rota que o usuário deve escolher para chegar ao seu local de trabalho.

Essas estruturas sociais que visam conectar pessoas através de informações extraídas de suas posições geo-espaciais, são conhecidas na literatura, como: Location-Based Social Networks. Esta área de estudo é definida formalmente em [6] do seguinte modo:

"Uma rede social baseada em localização não significa somente adicionar essa funcionalidade para uma rede social existente, de modo que os usuários possam compartilhar informação com localização embutida. Este tipo de rede consiste de uma nova estrutura social composta por indivíduos conectados pela interdependência derivada de suas localizações reais, assim como do compartilhamento de mídias com localidade embutida. A localização real não é composta somente da localização instantânea do usuário, mas também de um histórico de localizações por onde ele esteve em um certo período. Além disso, a interdependência não é determinada somente pela coexistência de duas pessoas no mesmo espaço físico ou por históricos de localidade semelhantes, também é necessário que se consiga inferir os mesmos interesses, comportamentos e atividades desses históricos e das informações com localização embutida na rede".

2.3.1 – Redes Sociais Baseadas em Localização

Os usuários de uma determinada rede social quando saem e realizam qualquer atividade estão fornecendo à rede informações para um mapeamento cronológico de suas localizações, seja mandando mensagens, tirando fotos, compartilhando vídeos, ou simplesmente com o celular no bolso. De qualquer forma, se a aplicação tem acesso ao GPS

de seu dispositivo, ela está o tempo inteiro atualizando o seu histórico de localidades. Para transformar todas essas informações em uma aplicação que realmente conecte as pessoas através de suas localizações, é necessário que se monte trajetórias dos históricos de localidade de cada usuário. Assim, com essas trajetórias, pode-se estabelecer correlações entre os usuários da rede e diversas localidades, que permitirão inferir conhecimento quanto ao grau de similaridade entre pessoas e locais. Essa explicação fica mais clara se observarmos a Figura 1.

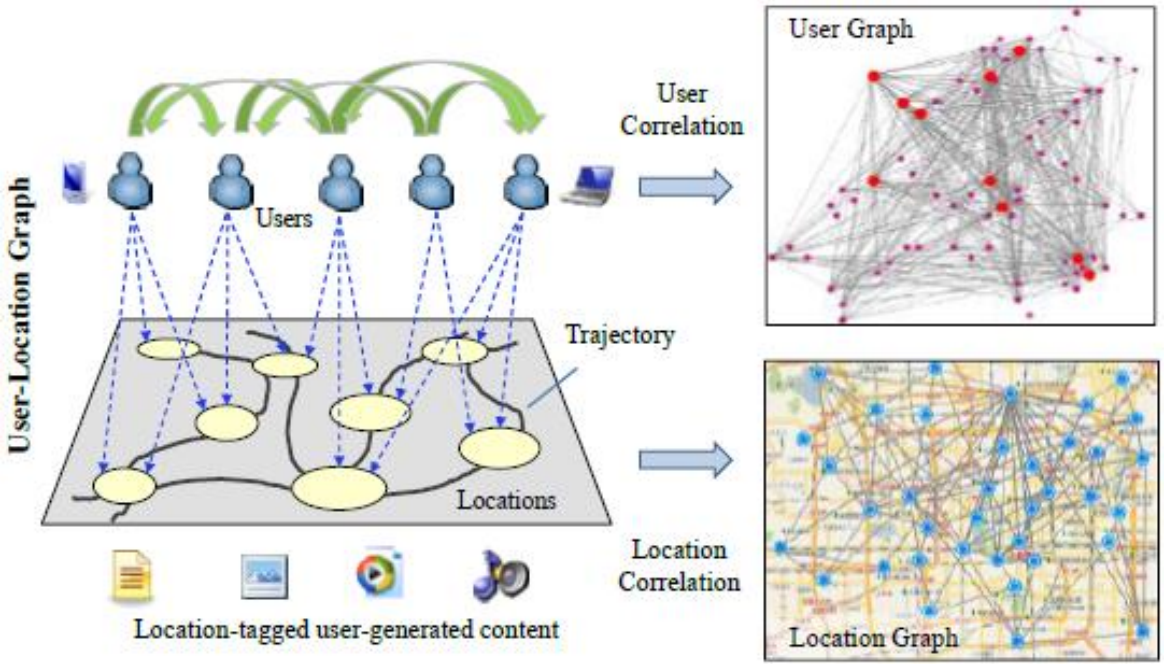

Figura 1 - Estrutura técnica de como montar uma rede baseada em localidades[27]

Algumas das funcionalidades que este tipo de rede pode prover a seus usuários, são [9]:

- Estabelecer graus de similaridade entre as pessoas.
- Descobrir quem são os experts naquela região, ou seja, os que conhecem mais locais dentro dos limites de uma região específica.
- Montar recomendações de itinerários para viagens, apontando atividades a se fazer e lugares para se visitar. Essas recomendações também podem ser personalizadas, basta pegar o escopo de pessoas que já foi ao local desejado com um grau de similaridade com o usuário superior a 0.7 por exemplo.
- Descobrir eventos ocorrendo nas regiões, sejam eles celebrações ou desastres naturais.

As complexidades para se desenvolver este tipo de rede social mostrado na Figura 1, são diversas. Além daquelas relacionadas as redes sociais tradicionais ainda temos as seguintes dificuldades adicionais que trona problema extremamente complexo:

- O grafo que representa a rede é constituído de 2 tipos de nós (usuários e localidades) e três tipos de links entre eles (usuário-usuário, usuário-localidade, localidade-localidade).
- Estas redes estão evoluindo constantemente, em geral a um ritmo mais rápido do que as redes sociais tradicionais, tanto na estrutura social quanto nas propriedades dós nós e dos elos.
- A localização tem características únicas, além das existentes em outros objetos gerenciados em uma rede social convencional. Além do relacionamento entre diferentes localizações (elos localização-localização), ainda existem relacionamentos definindo propriedades hierárquicas e sequenciais entre diferentes localizações (ex: rua A é continuação da rua B, rua A e rua B pertencem ao mesmo bairro C).

Atualmente, há diversas pesquisas sendo realizadas nessa área de conhecimento. Algumas publicações focam em entender os usuários através de seus históricos de localidade, modelando esses históricos através da trajetória do GPS, check-ins e inferindo contexto semântico às localizações [7] [10]. Outras publicações focam em mostrar estimativas para se aferir o grau de similaridade entre usuários de acordo com seus históricos geográficos [7] ou em espaços semânticos [11] [12].

No próximo tópico, são abordados os conceitos de redes sociais baseadas em eventos, o que mais se assemelha com o projeto apresentado neste trabalho. Como eventos têm uma posição geo-espacial associada, muito do que foi analisado para redes baseadas em localização, pode ser aplicado no contexto das redes baseadas em eventos, como é apresentado a seguir.

2.3.2 – Redes Sociais Baseadas em Eventos

As redes sociais baseadas em eventos (Event-Based Social Networks) têm como proposta promover que usuários interajam tanto online quanto offline. Essas redes englobam diversas funcionalidades básicas de uma rede social, porém sempre no contexto de eventos, ou seja, elas possibilitam usuários a criar e compartilhar eventos, e através deles, trocar

mensagens, localizações, fotos e comentários com os integrantes do evento. Alguns exemplos de aplicações hoje em dia utilizadas para difundir eventos sociais são o Plancast e o Meetup: o primeiro permite que os usuários sigam o calendário de atividades de outros usuários de forma unidirecional e que compartilhem, recomendem ou compareçam a eventos sociais desta pessoa; o segundo permite o compartilhamento de fotos e comentários entre usuários, além de possibilitar aos mesmos a adesão a grupos de eventos relacionados a um dado assunto. As redes sociais com foco em eventos devem considerar não somente a interação online de seus usuários, mas também a interação offline, ou seja, possibilitar encontros físicos entre seus participantes.

Apesar das tecnologias, em termos de gestão do evento, essas aplicações ainda têm muito a evoluir. Por exemplo, muitos eventos ainda dispõem de panfletos para guiar os convidados pelo local e exibir a programação disponível [13]. Entretanto, como em workshops e conferências, muitas palestras ocorrem simultaneamente, é imprescindível o surgimento de mecanismos que possibilitem o compartilhamento das informações discutidas e apresentadas em cada palestra do evento.

Além disso, os participantes ainda enfrentam um problema clássico devido a mudanças na programação do evento. Quando esta programação está impressa em banners ou folhetos, torna-se muito mais difícil comunicar alterações de última hora na agenda do evento. Novamente, se houver disponível uma ferramenta que permita o compartilhamento dessas informações de forma digital basta atualizar o novo horário no servidor, que as mudanças se propagarão pelos smartphones dos interessados naquela palestra. Ainda é possível aumentar a interação com o usuário, mapeando internamente o local do evento, de modo a otimizar o trânsito do usuário pelo local, tornando mais fácil para ele encontrar auditórios ou restaurantes. Por fim, ainda neste contexto, tais eventos têm a finalidade de promover a interação entre pessoas que se interessem pelo mesmo tema. Através de redes sociais, seria possível colocar essas pessoas em contato antes mesmo do determinado evento ocorrer, o que permitiria que elas trocassem experiências e informações, aumentando o universo de atuação do evento.

Em conjunto com a Internet das Coisas (Internet of Things) e aplicações baseadas em contexto (context-aware applications), acredita-se que organizadores de conferências, reuniões, feiras de negócios e demais sortes de encontros consigam expandir a interatividade de seus eventos, tornando a experiência dos convidados mais agradável [14]. Como elucidado no parágrafo anterior, muitos problemas decorrem da falta de uma plataforma de software capaz de englobar todos os participantes do evento.

A seguir, é apresentada uma análise de como se modelar o grafo por trás deste tipo de aplicação, seguido de um framework desenvolvido para redes sociais baseadas em um evento específico.

2.3.3 – Arquitetura das Redes Sociais Baseadas em Eventos

Conforme a definição de Ahmed et al [14], o conceito de evento em redes sociais traduz um pedaço de informação válido apenas por um determinado período de tempo. Em outras palavras, existe um campo que guarda a data exata do evento na rede [15] [16]. Eventos e usuários são criados o tempo inteiro neste tipo de aplicação, por isso elas são categorizadas como redes dinâmicas. Logo, para se detectar e descobrir agentes importantes na rede é necessário mapear, através da representação de usuários e eventos no grafo, a relação que eles mantêm entre si (usuário-usuário e evento-evento) e um com o outro (usuário-evento). Apesar disto, a maioria dos autores ainda modela esta rede considerando somente os usuários como nós do grafo [16]. Desta forma, a noção da importância de um dado nó para a rede fica limitada ao relacionamento dele com outros usuários, sem que se consiga inferir informações a respeito da relação daquele nó com os eventos.

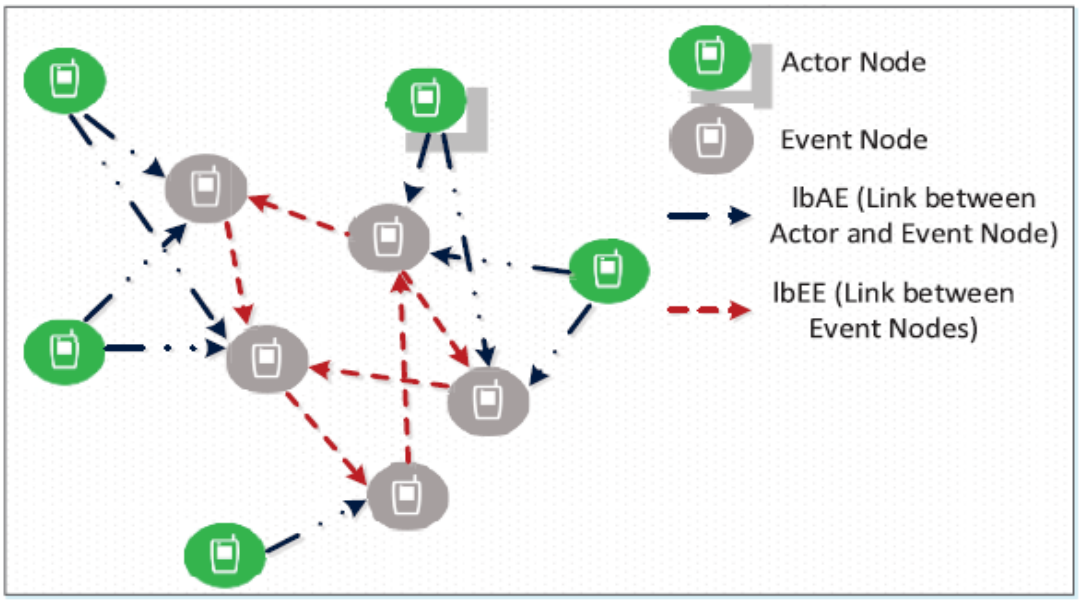

Figura 2 - Exemplo de modelagem proposta por Yuan et al [16]

Zhang et al [17] prefere modelar o grafo de forma diferente: os nós são representações tanto dos usuários como dos eventos gerenciados na rede, o que permite inferir mais informações sobre os dados. Como exemplo, deste modo, é possível relacionar os

participantes de cada evento, criar graus de similaridade entre os eventos e indicar eventos aos cadastrados no sistema.

2.3.4 – Framework para aplicações baseadas em eventos

Ahmed et al [14] sugere um framework para desenvolvimento desse tipo de rede social para um determinado evento. Pelo framework, pode-se observar que os participantes se cadastram para participar do evento. Essa etapa é gerenciada pelo módulo Membership Management and Registration Handler, que mediante aprovação dos dados do usuário, disponibiliza, através da Application Store, o aplicativo a ser utilizado durante a conferência. Além disso, esse módulo mantém ainda um banco de dados com o registro de todos os inscritos no evento, Participants Data. Tudo isso ocorre no step 1 (ver Figura 3), antes da conferência ser iniciada. No primeiro dia da conferência, os participantes têm de realizar o check-in. Durante o check-in, o módulo, Event Application Server, puxa do módulo, Membership Management and Registration Handler, a lista de cadastrados na aplicação, faz a autenticação e disponibiliza todos os dados para que o usuário inicie a aplicação de seu smartphone, ou seja, dispõe ao participante a comunidade de usuários que participará e as palestras, programações etc. Esse bloco ainda é responsável por coletar dados ao longo do evento e parar a aplicação quando ele terminar. O usuário quando adentra o local da conferência passa a fazer parte de uma rede virtual privada com toda a comunidade de pessoas que também estejam no local. Note que quaisquer mudanças de última hora são rapidamente comunicadas a todos os participantes, que possuam smartphones. Além disso, a aplicação torna a conferência ainda mais interativa.

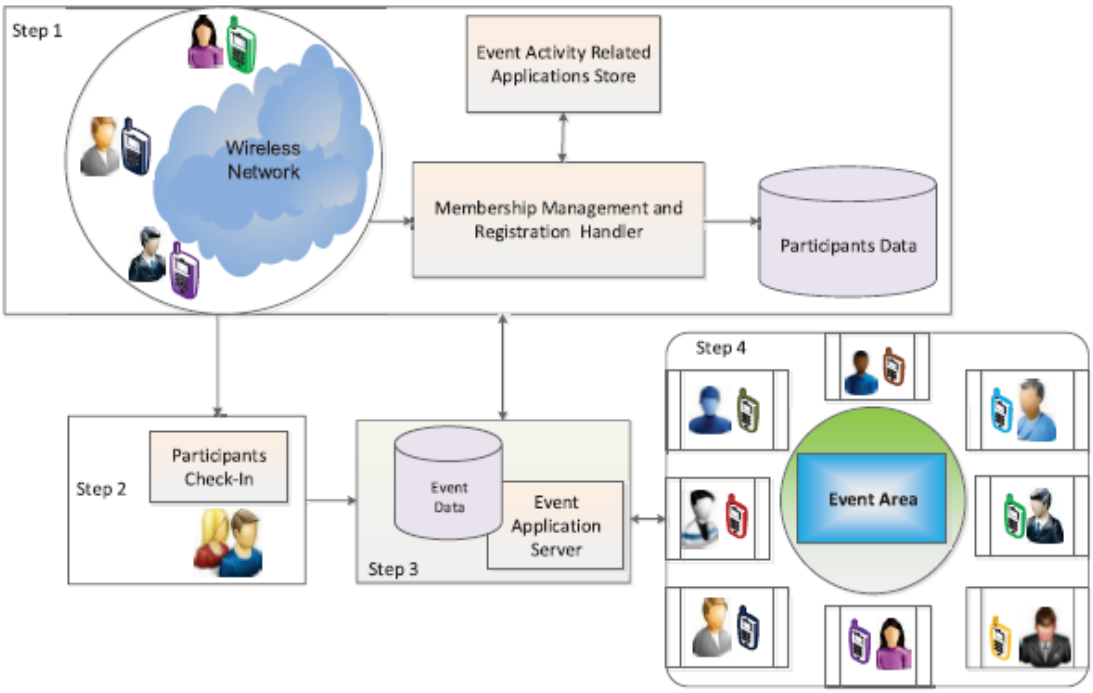

Figura 3 - Framework proposto por Ahmed et al [14]

Apesar de explicitar, de modo genérico, uma proposta de implementação de redes sociais móveis, o framework abstrai uma série de atributos que atualmente compõem o universo dessas redes. Para que esses atributos fossem contemplados, o step 4 teria de ser mais detalhado, exibindo uma arquitetura semelhante para os demais steps. Não é possível identificar um módulo que permita aos usuários compartilhar mídias com os demais integrantes da rede, por exemplo, ou mesmo o embasamento em contextos para se renderizar informações mais relevantes ao usuário de acordo com a sua posição instantânea, tema compreendido na literatura como Mobility Context-Aware. Em seguida, são esclarecidos esses conceitos e os desafios para o desenvolvimento de redes sociais, que incluam também tais funcionalidades.

2.3.5 – Mobility Context-Awareness

Diversas teorias tentam modelar os padrões de mobilidade humana, por exemplo a teoria proposta em [18], denominada Nomadic Community Mobility Model. Esta teoria considera o ser humano como um nômade, que, em sociedade, se movimenta randômica e temporariamente em torno de um ponto de referência e eventualmente - novamente em sociedade - translada para outro ponto de referência distante do primeiro, onde volta a se estabelecer, e se movimentar em torno deste ponto.

Quando considerado o universo dos eventos sociais, esta é a teoria mais aceita, pois denota um padrão de comportamento que podemos observar em workshops, conferências, festas etc. O que torna esta teoria tão fundamental para redes sociais baseadas em eventos, é que através desses padrões é possível inferir uma série de informações sobre o usuário devido ao smartphone, que ele carrega consigo. Assim, através de uma modelagem dos padrões de mobilidade humana, pode-se aferir com que frequência o usuário visita tais lugares, onde ele costuma almoçar, onde mora, qual a rota para o seu local de trabalho, além de uma série de outras informações que modelariam o contexto daquele usuário na rede. Como citado por Karamshuk et al [19], as dimensões emergentes do movimento humano são: a espacial, a temporal e a social.

Uma abordagem para os conceitos discutidos acima, considerando o espaço interno de um evento, é proposta por N. Yu Han et al [20]. Neste caso, o autor mapeia os locais onde ocorrerão palestras, workshops e outras atividades no mapa do evento. Enquanto essas atividades não ocorrem, elas estão sendo anunciadas dentro desses espaços. Esses locais são denominados pelo autor como Points of Interest (PoIs), como pode ser observado na Figura 4 a segur. Analisando a figura, também é possível aferir que os nós estão em constante movimento enquanto não adentram algum PoI do evento, por onde se mantêm por um período considerável. Isso comprova a semelhança entre a teoria estudada em [18], Nomadic Community Mobility Model, e o padrão de movimento dos participantes em uma conferência. Lógico que há particularidades de cada caso, porém a semelhança já permite aferir muito sobre o contexto do usuário no evento. Além disso, é importante notar que o mapa interno representa tanto o contexto temporal como o espacial de um evento, pois conseguimos, por exemplo, determinar o local mais visitado em um dado instante, basta olhar a configuração do mapa em um determinado instante.

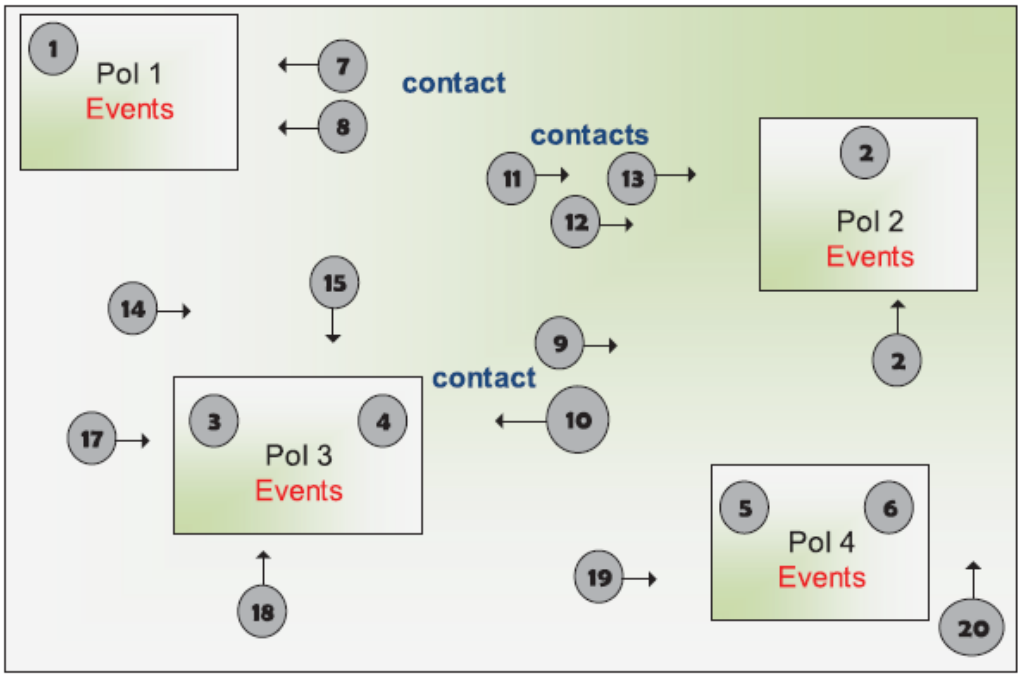

Figura 4 - Mobilidade humana analisada em eventos [14]

N. Yu Han et al [20] ainda considera que cada usuário de sua aplicação possua: um id único, um perfil de mobilidade com os contatos estabelecidos no evento, os PoIs visitados pelo usuário, os eventos desses PoIs, que o usuário participou, e, por fim, um tempo de influência, no qual ele pode influenciar os demais participantes da rede a visitar o dado PoI através das informações coletadas por seu smartphone. Essas informações desaparecem depois de um tempo. Essa proposta é interessante pois explicita uma tentativa de se inferir um contexto social ao movimento humano dentro de uma conferência. Essa é uma dimensão emergente da mobilidade humana [19].

O próximo tópico introduz alguns exemplos de funcionalidades que esta rede pode possuir. Além disso, ela mostra um framework de como poderiam ser dispostos na tela dos smartphones as diversas funcionalidades que a aplicação agrega.

2.3.6 – Experiência do usuário em redes Sociais Baseadas em Eventos

As aplicações móveis baseadas em eventos estão atraindo cada vez mais a atenção de pessoas que comparecem frequentemente a fóruns, conferências e demais sortes de eventos. Essas aplicações ainda precisam evoluir de forma a melhorar a experiência do usuário com a

aplicação. A Figura 5 a seguir mostra um padrão de interface para essas aplicações e uma série de funcionalidades que o desenvolvedor pode agregar em sua solução.

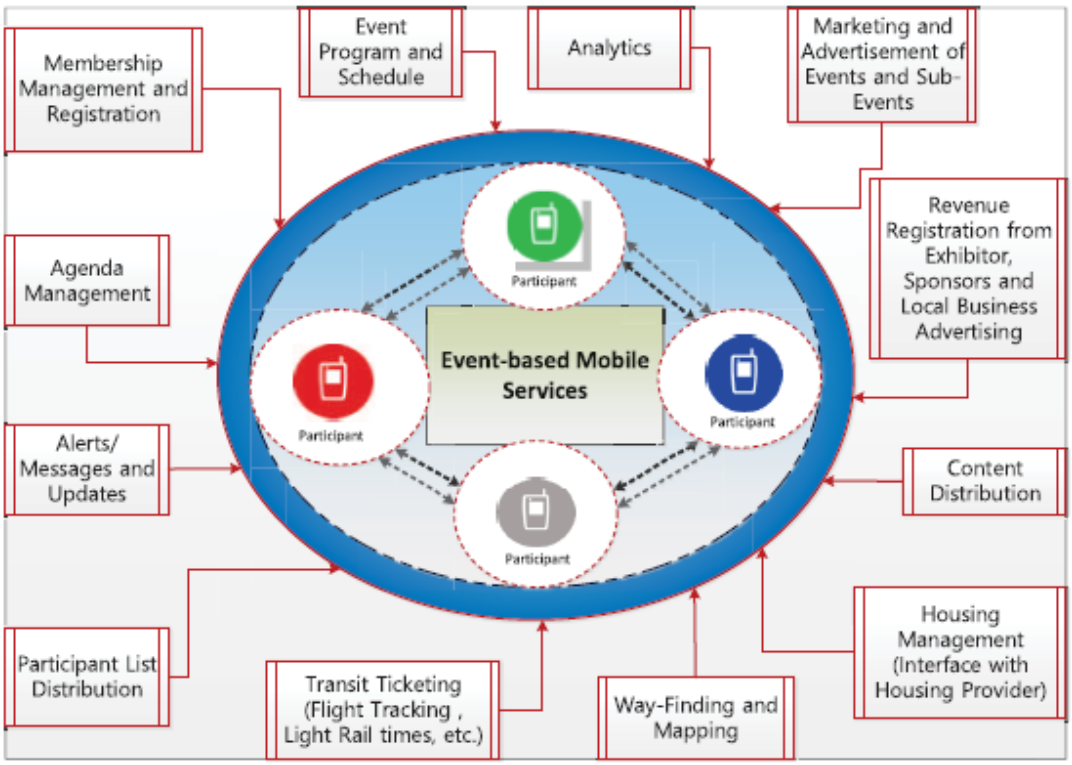

Figura 5 - UI com as funcionalidades em segundo-plano [14]

O padrão de interface da Figura 5 mostra diversas funcionalidades que uma aplicação móvel baseada em eventos pode oferecer. Com este modelo, garante-se uma boa experiência ao usuário da aplicação. Isso se deve ao fato de as funcionalidades estarem às margens da tela, nas laterais, sem comprometer a visão global da aplicação. O usuário caso precise de alguma funcionalidade, basta procurar nos botões laterais da tela. Além disso, esse padrão libera bastante espaço para eventuais patrocinadores do evento e também melhora sua usabilidade.

2.3.7 – Comparação entre aplicações móveis baseadas em eventos

Ahmed et al [14] levantou uma série de aplicações móveis que se destacam por simular redes sociais baseadas em eventos. O objetivo deste levantamento foi reconhecer quais funcionalidades já estão se tornando padrões do desenvolvimento dessas redes e por quais caminhos as aplicações têm buscado inovação. É importante ressaltar que quanto maior a interação com o usuário, mais valor o aplicativo estará agregando ao evento.

As aplicações analisadas foram: Ungerboeck, GenieConnect Grupio, CrowdCompass, Quickmobile, SeedLabs, Pathable, Zerista, EventMobi, FollowMe or Core-Apps e SherpaSolutions. A seguir é apresentada cada uma dessas aplicações seguida da tabela comparativa entre elas:

- **Ungerboeck** é uma aplicação desenvolvida pela USI International. Seu foco é aumentar a valor gerado pelo evento ao usuário, possibilitando que ele acesse a programação atualizada das sessões do evento, saiba mais sobre os palestrantes e os demais participantes e encontre outras pessoas na conferência. A aplicação roda como uma interface web em cima de qualquer dispositivo móvel, logo o suporte é multi-plataforma.
- **GenieConnect** foi desenvolvida por Genie Mobile Ltd para transformar comportamentos do evento em inteligência de negócio. Pode ser configurada pelas necessidades dos organizadores. Esta rede agrega mais valor ao usuário em workshops. As principais funcionalidades desta aplicação são: programação do evento, exibição dos perfis dos palestrantes e exibidores, links para os diversos stands ou auditórios do evento, mapa interno do local, integração com o Google Maps para informações sobre como chegar, acesso a análises sobre tendências, atividades dos exibidores e comportamento dos participantes, upload de vídeos ou mensagem de voz aos stands, sessões, exibidores ou organizadores.
- **Grupio** foi desenvolvida pelo grupo Dharanet LLC. Facilita o acesso a informações do evento, possibilita a comunicação entre todos os envolvidos, participantes ou organizadores. A plataforma roda como aplicação nativa para os sistemas operacionais: iOS, Android e Blackberry e funciona como website para os demais dispositivos.
- **CrowdCompass** visa popularizar os eventos. Permite a integração de diversos tipos de evento em sua plataforma, de modo que não seja preciso ao usuário acessar diversas plataformas para gerenciar todos os seus eventos.
- **Quickmobile** vem transformando eventos efêmeros em longas interações sociais virtuais entre seus participantes. A plataforma Quickmobile conta com quatro aplicações fundamentais para uma conferência, são elas: MobileEvent, onde os participantes são iniciados na plataforma e recebem todo o conteúdo do evento, assim como a lista de participantes e programação; MobileConnect, que permite o envio de guides do evento entre os participantes; o MobileMembership; MobileAmenites; MobileVenues

- **SeedLabs** é desenvolvida pela Seed Labs. Este aplicativo permite que os usuários criem suas agendas de acordo com a programação que os interesse. Assim, a aplicação permite que os usuários compartilhem suas programações específicas, além de fotos, biografias, áudios, mapas, vídeos etc. Ela ainda conta com um mapa interno da área do evento para guiar o usuário. Por fim, essa plataforma garante aos patrocinadores do evento espaço para propaganda e divulgação de mensagens aos participantes.
- **Pathable** proporciona que os usuários interajam antes, durante e depois do evento. Além disso, Pathable também oferece suporte para mais de um evento, ou seja, os usuários podem participar de múltiplos eventos na mesma plataforma. A aplicação carrega todos os dados para o cliente uma vez que o usuário a acesse através de alguma rede, depois ele pode navegar pela aplicação off-line.
- **Zerista** foi desenvolvida pela Zerista Company. A plataforma é nativa para os sistemas operacionais: Android e iOs. Para os demais dispositivos roda como uma aplicação web. Esta aplicação oferece a programação do evento, um serviço de mensagens entre os participantes e integração com outras redes sociais (LinkedIn, Facebook).
- **EventMobi** foi desenvolvida pela 5Touch Solutions Inc. Garante ao usuário acesso à aplicação da conferência de diversas plataformas diferentes. Não possui nenhuma funcionalidade fora do padrão.
- **FollowMe or Core-Apps** foi desenvolvida pela Core-apps LLC. Esta plataforma é utilizada por organizadores, participantes e exibidores. Ela pode ser utilizada de maneira off-line para consulta das informações do evento, possibilitando que se a utilize em lugares em que o acesso a rede é intermitente ou inexistente, como em aviões, por exemplo.
- **SherpaSolutions** desevolve um software para redes sociais baseadas em eventos. Igualmente aos demais apresenta as mesmas funcionalidades básicas. Entretanto, com a funcionalidade desenvolvida, chamada ActivMetrics, a plataforma permite inferir informações sobre o comportamento dos usuários – User Behavior. Esta ferramenta monitora, em tempo real, o tráfego das redes Wi-Fi ao longo do evento, permitindo que se tenha informações sobre como a plataforma está sendo utilizada, ou se realmente está sendo utilizada.

Tabela 1 - Análise de funcionalidades dos apps de redes sociais baseadas em eventos [14] .

Mobile Applications and Research Projects	Environmental Friendly	Agenda and Membership Management	Messaging or Alerts	Attendee List and Schedule Distribution	Content Distribution	Marketing and Advertizement of Events	Analytics	Housing Management	Revenue Registration	Transit Ticketing	Way Finding and Mapping
Ungerboeck	✓	✓	✓	✓	✓	✓	-	-	-	-	-
GenieConnect	✓	✓	✓	✓	✓	✓	✓	-	-	-	✓
Grupio	✓	✓	✓	✓	✓	✓	-	-	-	-	-
CrowdCompass	✓	✓	✓	✓	✓	✓	-	-	✓	-	-
QuickMobile	✓	✓	✓	✓	✓	✓	-	✓	-	-	-
SeedLabs	✓	✓	✓	✓	✓	✓	-	✓	-	✓	✓
Pathable	✓	✓	✓	✓	✓	✓	-	-	-	✓	-
Zerista	✓	✓	✓	✓	✓	✓	-	-	-	-	-
EventMobi	✓	✓	✓	✓	✓	✓	-	-	-	-	-
FollowMe	✓	✓	✓	✓	✓	✓	✓	-	✓	-	-
Sherpa-Solutions	✓	✓	✓	✓	✓	✓	-	-	-	-	✓

Cada aplicação apresenta uma solução diferente para os desafios que as redes sociais baseadas em eventos contêm. É impossível haver uma única plataforma que consiga agregar todas as funcionalidades inerentes deste tipo de rede e ainda apresentar um sistema plausível do ponto de vista de usabilidade. Logo, cada plataforma aposta em um escopo de funcionalidades que garantam uma boa experiência ao usuário. Por exemplo, algumas mapeiam o ambiente interno do evento para otimizar a movimentação do usuário, outras apostam em análises analíticas para se montar o perfil de uso da aplicação, e conseguir, de certa forma, mensurar o ganho que ela trouxe aos participantes, algumas apostam no suporte multi-evento, possibilitando ao usuário gerenciar mais de um evento na mesma plataforma. Enfim, existe uma infinidade de caminhos que os desenvolvedores podem seguir para tornar suas aplicações mais relevantes. Observando a Tabela 1, nota-se que as primeiras seis funcionalidades já são padrões de aplicações baseadas em eventos: distribuição da programação e lista de participantes, distribuição do conteúdo do evento, gerenciamento de agendas, alarme e mensagens etc. Essas funcionalidades já devem fazer parte de qualquer aplicação do gênero. A inovação fica por conta das demais funções da aplicação.

O próximo tópico trata especificamente de uma das funcionalidades que vem se tornando cada vez mais presente nas redes: multimedia streaming. O comportamento do usuário é dinâmico e as formas de upload de vídeo e áudio nessas redes necessitam evoluir.

2.3.8 – Multimedia Streaming

Outra funcionalidade imprescindível às redes sociais móveis é a possibilidade de usuários compartilharem mídias, como: áudios, vídeos ou fotos. Esse atributo garante melhor interatividade com a aplicação, tornando a experiência do usuário mais aprazível. A solução mais adequada a esta funcionalidade é a adoção de redes P2P para streaming de mídias, como é explicado no próximo parágrafo. Contudo, essas redes sempre representaram um problema para os provedores de serviços de Internet, que as consideravam como uma forma de consumo monstruoso de largura de banda. Sempre houve políticas agressivas para se controlar o tráfego na rede e limitar a banda consumida através deste protocolo [21] [22]. Atualmente, essa preocupação se estende aos provedores de serviços telefônicos, devido ao volume de tráfego que este protocolo tem gerado nas redes móveis.

A tecnologia P2P surgiu com o propósito de possibilitar a propagação de arquivos pesados pela rede, de modo a tornar seu download mais rápido. Essa propagação funciona da seguinte maneira, as CPUs que já tenham feito o download do arquivo, se tornam seeds – ou como denominado em [23], "fornecedores de conteúdo" – para as demais CPUs que desejam baixar o mesmo arquivo. Assim, a rede seleciona as CPUs mais próximas e com banda disponível para fornecer uma parte de seu conteúdo às demais. Desta forma, uma CPU pode variar de qual máquina estará baixando o arquivo, dependendo da disponibilidade dos seeds. Como eles são máquinas particulares de usuários e não servidores, a conexão é intermitente. Além disso, máquinas que estiverem com muitas requisições também terão mais dificuldade de "entregar" o conteúdo. Assim, a CPU em processo de download, troca de fornecedor com certa frequência. Como exemplo deste paradigma, pode-se citar o BitTorrent, que acelera o download de arquivos inserindo seus usuários em uma rede P2P [23].

Analogamente à explicação acima, esta rede pode ser implementada em um contexto de smartphones para multimídia streaming. Atualmente, stream de vídeos e áudios são compartilhados em redes P2P através do particionamento dessas mídias em pequenos blocos de dados. Cada bloco possui uma numeração de acordo com a sua posição no vídeo ou áudio. A CPU irá baixar esses pacotes de dados nos seeds mais adequados. Desta forma, conforme os blocos vão sendo baixados, eles são encaminhados a um receiver na CPU que se encarrega de ordená-los pelo campo mencionado acima. Assim, a mídia vai sendo reproduzida de acordo com ordenação dos blocos disponíveis. Outra particularidade, que torna este tipo de compartilhamento extremamente complexo – além dos problemas clássicos da arquitetura

P2P, já discutidos acima, como: latência, largura de banda limitada e intermitência das conexões –, é a própria definição de streaming: os blocos são executados em um determinado tempo na mídia, logo devem estar disponíveis antes de sua execução, senão o vídeo trava, esperando aquele pedacinho de conteúdo estar disponível.

Atualmente, a utilização desta arquitetura pelos smartphones acontece majoritariamente da seguinte forma: qualquer requisição entre smartphones é intermediada por um host estacionário, conectado a alguma rede com fio ou móvel, que alivia a sobrecarga computacional, que os celulares não teriam capacidade de lidar – pelo menos há algum tempo atrás. Esses hosts funcionam como um proxy, recebendo a requisição de um smartphone e redirecionando para o outro. Este método reduz a aplicabilidade de sistemas P2P específicos para smartphones, pois através dele, por exemplo, não se consegue inferir a posição do smartphone que enviou a mensagem.

Com o advento do protocolo Wi-Fi Direct, os celulares poderão se comunicar diretamente sem a necessidade de pontos de acesso ou roteadores intermediários. A arquitetura do Wi-Fi Direct possui uma camada física para suporte à comunicação P2P [22]. Assim, ele se encarregará da sobrecarga computacional gerada por este protocolo, quando os celulares estiverem em uma rede Wi-Fi. Entretanto, esta tecnologia funcionará até uma distância fixa.

Com a evolução do Wi-Fi Direct aumentando sua área de cobertura e um melhor gerenciamento da largura de banda consumida por celulares em uma rede P2P na rede móvel que está surgindo, o 4G. É possível que se consiga compartilhar mídias em tempo real com muito mais eficiência.

A seguir, são analisadas as tendências para o desenvolvimento de aplicações de redes sociais móveis baseadas em eventos.

2.3.9 – Tendências futuras para Redes Sociais baseadas em Eventos

O Quick Mobile 2012 [24] levantou os pontos cruciais a serem observados no desenvolvimento de aplicações de Redes Sociais baseadas em Eventos.

O texto avalia a condição dos participantes de criarem conteúdo para o evento, content creators, compartilhando informações adquiridas no transcorrer do evento, divulgando eventos semelhantes aos convidados, provendo feedbacks, através de surveys ou games, ou seja, o aplicativo proposto deve levar em consideração todas essas opções de interação com o

usuário e propor uma plataforma simples de se operar, completamente interativa e customizável aos interesses de cada um.

Outro ponto apontado é a variedade de opções que a aplicação mobile tem de gerar receita para o evento. Em 2011, por exemplo, no Mountain Travel Symposium, os organizadores venderam uma certa quantidade de push-notifications (notificações) para os patrocinadores, que, por sua vez, aproveitaram a oportunidade para divulgar a sua marca, através de merchandising, oferta de produtos e descontos através dessas notificações. Um restaurante italiano, por ter ofertado pratos 20% mais baratos durante a conferência, obteve uma capitalização muito acima dos demais. Isto evidencia o poder comercial, que as redes sociais podem exercer.

Há também no texto uma ressalva para as tendências que vêm ganhando força no mercado de software. Conceitos como o Do It Yourself (DIY) e Don't Repeat Yourself (DRY) vem sendo utilizados pelos desenvolvedores com mais frequência ultimamente. Outras tendências, que devemos observar para o futuro, são a consolidação do HTML5 para desenvolvimento mobile e a evolução dos sistemas de posicionamento indoor, permitindo inferir mais detalhes sobre o espaço mapeado.

Desta forma, esta introdução aos conceitos de Redes Sociais Móveis focadas em Eventos termina avaliando os problemas ainda em aberto para o desenvolvimento dessas aplicações. O primeiro desafio aos desenvolvedores se deve à abundância de aplicações desta sorte - cada workshop ou conferência possui uma plataforma diferente. Assim, surge a necessidade de se pensar arquiteturas de software mais robustas que possibilitem a integração com outras redes sociais da mesma natureza. A este conceito, atribui-se o termo interoperabilidade entre as redes [24]. Além deste problema, há a necessidade de arquitetar uma camada de software que aborde gerenciamento de recursos, posicionamento do usuário, mecanismos de buscas, chats online, disponibilidade de dados, inferência de contexto, compartilhamento de arquivos, vídeo streaming etc. Todos os pontos discutidos até agora devem estar contidos no design de um middleware [25]. Outro fator que trará mais relevância a este tipo de aplicação é a funcionalidade de se recomendar eventos aos usuários. Para que essas recomendações se tornem de fato relevantes, elas devem estar atreladas a eventos a que o usuário realmente compareceu. Além disso, recomendações de eventos agrega uma complexidade a mais quando comparada a recomendações de livros ou vinhos, por exemplo. Eventos têm um determinado tempo de existência [26]. Essa natureza dinâmica os tornam complexos de se monitorar, pois eles surgem e desaparecem a todo instante. Esta complicação se reflete na determinação de um universo de eventos recomendáveis ao usuário. Outra

funcionalidade que agrega um valor imensurável a este tipo de aplicação é a integração com demais sites ou apps que estão, de alguma forma, relacionados ao evento. Por exemplo, integrar um aplicativo que monitore preços de vôo, seria interessante para um usuário que mora distante da localização do evento, ou um restaurante que reserve descontos aos participantes do evento, também poderia estar integrado à plataforma.

Conclui-se nessa etapa um entendimento geral do que são de fato redes sociais móveis baseadas em eventos. Este entendimento inclui: observar as tendências e desafios técnicos, além das diversas aplicabilidades e oportunidades que estas redes proporcionam, como discutido nos casos acima. Nem todos os pontos, que compõem as dificuldades técnicas do desenvolvimento de tais aplicações, foram abordados no protótipo apresentado neste projeto, porém algumas se encaixam na arquitetura desenvolvida pelo aluno, como será visto mais adiante. Em seguida, serão abordadas as peculiaridades do desenvolvimento mobile – especificamente para o sistema operacional, Android, sistema utilizado neste projeto.

Capítulo 3
Aplicações em Dispositivos Móveis

Quando se deseja programar para um novo tipo de dispositivo, deve-se inicialmente levar em conta as principais características que o diferenciam do elemento-chave da programação clássica: o desktop. Especificamente, no caso de smartphones, há uma infinidade de particularidades a serem consideradas, que determinam a qualidade do software desenvolvido. Com isso em mente, avaliou-se o estado-da-arte nas técnicas de desenvolvimento para smartphones, sob a ótica de suas limitações e características.

Smartphones possuem baixa velocidade de processamento, curto espaço em memória e bateria de curta durabilidade. Tudo isso se deve obviamente a sua limitação de tamanho e espessura. Esses dispositivos não possuem um tamanho pré-fixado, podendo ser encontrados nos mais diversos tamanhos e resoluções. Essa variação atrapalha o desenvolvimento de uma interface gráfica única, que se adapte às diversidades apresentadas no mercado. Vale lembrar ainda, que o sistema operacional dos smartphones também é utilizado para os tablets, o que representa uma mudança drástica no tamanho da tela. Desta forma, dificilmente a mesma interface gráfica proporcionará uma boa experiência ao usuário, quando acessar o software pelos diversos dispositivos. Por fim, os smartphones ainda contam com uma série de sensores que habilitam o desenvolvedor explorar características do próprio usuário de seu sistema, como são os casos do GPS e do acelerômetro, sensores fundamentais quando se pensa em desenvolver aplicações móveis.

3.1 – Sistema Operacional Android

O sistema operacional Android é baseado em Linux e foi iniciado pela startup de Palo Alto, Android Inc. Logo, em seguida comprada pela Google que continuou o desenvolvimento do sistema. A Figura 6 mostra como a arquitetura do sistema foi montada. Uma breve explicação de cada camada com seus blocos mais importantes é apresentada em seguida [27].

Linux Kernel:

Esta camada atua como uma abstração entre o Hardware e as demais camadas de software do sistema. Serviços básicos de um sistema operacional são tratados nesta camada, como: segurança, gerenciamento de serviços, gerenciamento de memória, pilha de protocolo para acesso a redes, além de drivers para demais serviços do sistema, como pode ser observado pela Figura 6.

Figura 6 - Arquitetura do Sistema Operacional Android [27]

Libraries:

Acima do Kernel, existe uma camada com bibliotecas programadas em C/C++, que são responsáveis por diversos serviços, implementados na camada superior (Application Framework), que utilizam essas bibliotecas para prover a maioria das funcionalidades do sistema e disponibilizá-los como API para os desenvolvedores Android. A seguir, são apresentadas algumas dessas bibliotecas:

- **Surface Manager:**

  Essa biblioteca é responsável pelo gerenciamento da tela do usuário. Ela guarda as telas das diversas aplicações que estão rodando como serviços no sistema e renderiza a escolhida pelo usuário.
- **Media Framework:**

  Responsável pela experiência de áudio e vídeo do usuário. Esta biblioteca aceita os padrões IMPEG4, H.264, MP3 e AAC. Todos os arquivos de áudio e vídeo necessários para se proporcionar uma rica experiência de áudio e vídeo.
- **SQLite:**

  Banco de dados SQL. Principal responsável por guardar dados do sistema.
- **Open GL/ES:**

  Biblioteca de renderização gráfica 3D.
- **FreeType:**

  Biblioteca responsável pela renderização das fontes (escrita) na tela da aplicação.
- **WebKit:**

  Biblioteca aberta para tratamento de renderização de páginas web. Esta biblioteca permite que o Android renderize adequadamente páginas desenvolvidas para os padrões de desktop, ou seja, tamanhos e resoluções diferentes.
- **OpenSSL:**

  Biblioteca open-source, que implementa os protocolos SSL e TLS, serve para encriptar os dados enviados pela rede entre dois celulares, de modo a tornar a comunicação segura entre eles.
- **SGL:**

  Biblioteca 2D. A plataforma Android permite que uma aplicação combine elementos 2D e 3D, através do SGL e OpenGL/ES.

Android Runtime:

Cada aplicação android roda em um processo no sistema operacional. Cada processo possui uma instância da máquina virtual Dalvik. A máquina virtual Dalvik foi implementada para que o aparelho conseguisse rodar diversas instâncias dela de maneira segura. Esta máquina virtual funciona bem em situações de baixa energia e pouca memória. Ela gera os arquivos .DEX, que são criados em tempo de compilação do código do seu programa Java.

Este bloco ainda possui bibliotecas básicas da linguagem Java, que o usuário necessite utilizar em seu código, como bibliotecas de IO (Entrada e Saída), ou as Collections da linguagem.

Application Framework:

O próximo nível do sistema operacional Android apresenta o código Java base para o desenvolvimento de aplicações Android. Frameworks para se explorar as funcionalidades do celular são encontrados nesta camada.

- **Activity Manager:**
  Controla o ciclo de vida da aplicação. Mantém também uma pilha de aplicações rodando para que a navegação entre elas aconteça de forma simples e rápida.
- **Package Manager:**
  Mantém uma lista das aplicações instaladas no aparelho, além de cada funcionalidade do aparelho que elas utilizam.
- **Window Manager:**
  Gerencia as telas da aplicação. É uma abstração Java para o código C/C++ do SurfaceManager.
- **Telephony Manager:**
  Contém as APIs utilizadas para se construir aplicações relacionadas a ligações telefônicas.
- **Content Providers:**
  Possibilita o compartilhamento de dados entre duas aplicações diferentes.
- **View Systems:**
  Contém os elementos para se construir uma tela na aplicação. Todos os elementos de interface do usuário como botões, listas etc. Também gerencia os eventos de acionamento desses botões, como onButtonClick, que gera um evento para o clique em um dado botão. Ou seja, tudo relacionado a UI está contido nesta API.
- **Resource Manager:**
  Guarda as Strings, os arquivos bitmap, os arquivos XML de layout etc. Tudo que não for código é gerenciado por este bloco no sistema.
- **Location Manager:**
  São APIs para localização via GPS. Este serviço será analisado mais adiante no texto.
- **Notifications Manager:**
  APIs para gerenciamento de notificações entre os dispositivos.

- **XMPP:**

  Extensible Messaging and Presence Protocol são protocolos para troca de dados estruturados em XML instantaneamente. Eles são utilizados para serviços de troca de mensagens instantânea, VoIP, serviços de redes sociais, games, etc. Todos os serviços que envolvam troca de dados instantaneamente.

Application Layer:
Esta camada contém todas as aplicações a que o usuário final tem acesso diretamente.

Após esta introdução do sistema operacional Android é analisado no próximo tópico sensor GPS embutido no sistema. Como foi explicado na introdução este sensor tem muita importância para a aplicação, pois ele fornece as posições dos eventos ao redor da localização do usuário.

3.2 – Global Positioning System

Os satélites fornecem informações de horário e posicionamento ao smartphone quando o usuário não possui nenhuma obstrução na sua linha de visão. Cada satélite GPS emite continuamente um sinal modulado, que possui um código binário, conhecido pelos receptores de GPS (GPS receivers) – módulo instalado internamente no smartphone – além de informações sobre o horário de transmissão (na escala de tempo do satélite) e o posicionamento do satélite. Assim, o receptor calcula o tempo demorado para se receber o sinal emitido pelo satélite. Como todos os sinais se propagam com a mesma velocidade, a velocidade da luz, esta informação está diretamente relacionada à distância do satélite para o smartphone. Logo, através do processo de multilateração [28], consegue-se estimar a posição do usuário. Para que este sistema funcione corretamente, são necessários pelo menos quatro satélites diferentes: um para sincronizar o horário do aparelho com o do satélite e os demais para determinar através do processo de multilateração, a posição do usuário em três dimensões com origem no centro da terra [29].

Esse sistema possui algumas limitações. Uma delas é causada por determinadas inconsistências nas condições atmosféricas. Na troposfera, o sinal emitido pelo satélite pode ser atrasado por determinadas condições de humidade do local onde o receptor de GPS se encontra. Ainda é possível, nesta camada, detectar atrasos do sinal devido às condições de pressão e temperatura ambiente [30]. Na ionosfera, o sinal também pode ter sua velocidade de propagação alterada. Isso acontece devido a um fenômeno chamado dispersão [31]. A

velocidade de fase da onda emitida pelo satélite depende de sua frequência. Assim, internamente na ionosfera, dependendo da frequência da onda emitida, o sinal pode apresentar um atraso significativo. Os sinais de alta-frequência (30-300 MHz) sofrem uma influência menor desta camada. A segunda fonte de imprecisões na medida do sistema acontece devido a propagação da onda por múltiplos caminhos. Os sinais de rádio-frequência são refletidos por diversos objetos do ambiente terrestre, como: prédios, montanhas, água etc. Assim, os sinais refletidos são atrasados e, por isso refletem medições imprecisas da posição. Logo, quanto mais reflexões o sinal sofrer, menor a probabilidade do receptor de GPS captar o sinal que não sofreu nenhum atraso do meio.

Essas limitações do sistema de posicionamento global inspiraram o advento de outras tecnologias para auxiliar o GPS, são elas: GPS assistido, sistema de posicionamento Wi-Fi. Desta forma, surgem os sistemas de posicionamento híbridos que utilizam mais de uma tecnologia para conseguir prover a localização exata do usuário no mapa. A seguir, analisa-se algumas dessas tecnologias e como sua aplicabilidade pode auxiliar o sistema tradicional de GPS ou mesmo substituí-lo, em certos casos.

O sistema de posicionamento Wi-Fi pode ser utilizado em áreas urbanas, desde que haja diversos pontos de acesso a rede, ou em ambientes fechados, onde o sinal de GPS é bloqueado. Este sistema funciona da seguinte forma. Inicialmente, é medido a intensidade do sinal Wi-Fi dentro do ambiente. Em seguida, através de fingerprinting [31], identifica-se a rede Wi-Fi utilizada. Por último, acessa-se o banco de dados do Wi-Fi Hotspot, que correlaciona o endereço MAC da rede ou do ponto de acesso e os dados de geolocalização GPS dos celulares no mesmo espaço físico.

O GPS assistido utiliza os dados das torres telefônicas para determinar a posição do usuário. As torres telefônicas se conectam com os satélites o tempo inteiro, recebendo as informações de órbita de cada satélite conectado a torre e salvando essas informações para um banco de dados. Assim, o smartphone através de redes móveis de rádio como o GSM [32], baixa essas informações do servidor da torre telefônica. O A-GPS ou GPS assistido é utilizado quando o sinal do satélite não pode ser usado adequadamente, como em áreas urbanas, que geram diversas reflexões ao sinal, ou em condições meteorológicas adversas, também responsáveis pelo enfraquecimento do sinal. Assim, eles se valem das antenas mais altas para aquisição dos dados dos satélites, e os acessam via GSM. As prestadoras de serviços telefônicos tarifam este serviço como serviços de internet móvel [33].

O estudo das tecnologias disponíveis para aquisição de localização do usuário é importante para a aplicação. Ela mostra no mapa os eventos disponíveis ao usuário e através

dessas tecnologias ele consegue inferir qual o evento mais próximo e como chegar até ele. Em seguida, são analisadas técnicas de desenvolvimento de aplicações Android em termos de usabilidade e baixo consumo de energia.

3.3 – Recomendações para o desenvolvimento de Aplicações Móveis

No universo das aplicações móveis, um dos conceitos mais explorados e de necessário entendimento dos desenvolvedores, chama-se: Usabilidade. Este conceito é definido em [34], da seguinte forma:

"Usabilidade é um atributo da qualidade de uso da aplicação. Ela mede, pelo ponto de vista do usuário, quão eficiente, efetiva e seguramente ele consegue realizar tarefas úteis, que sejam fáceis de se aprender e memorizar, além de quão aprazível ele considera a experiência de lidar com o software desenvolvido".

Boa usabilidade é fundamental para que a aplicação seja bem-sucedida e uma das principais características que afeta este parâmetro é a variabilidade de tamanhos que as telas podem assumir. Algumas aplicações são fáceis de se manusear em determinados dispositivos, porém em outros a experiência se torna frustrante, portanto pensar meios de se evitar ao máximo essa disparidade é crucial para uma boa aplicação.

Bettina Biel e Volker Gruhn, da Universidade de Duisburg-Essen, propõem em [35] um mecanismo para mitigar os problemas causados por essa variação de resolução e tamanho dos dispositivos. Em seu artigo, eles defendem que o desenvolvedor paute o layout da aplicação em widgets, cujo tamanho e posição possam ser relativizados, e não pré-determinado em pixels. A plataforma Android permite que o programador faça isso, atribuindo às diversas widgets, o parâmetro weight, que distribui a tela através do balanceamento das views do layout. Desta forma, se um layout é orientado verticalmente com, por exemplo, cinco views diferentes, cada uma com o parâmetro weight igual a 1, têm-se que cada view ocupará ⅕ do tamanho da tela verticalmente. Este é um jeito simples de se tornar o layout da aplicação adaptável. Desta forma, é garantido que todas as views da tela aparecerão para o usuário, porém nem sempre esta distribuição proporcionará uma boa experiência. Às vezes, o conteúdo que se encaixa em uma tela grande é extenso, e quando adaptado para uma menor, a tela se torna tão poluída que usuário passa a ter dificuldade para manusear a aplicação. Neste caso, por exemplo, o conteúdo estaria melhor disposto na tela com o auxílio do scrollbar, ao invés desta forma de balanceamento. O inverso também pode

ocorrer: um conteúdo bem diagramado em uma tela pequena pode parecer muito disperso em uma tela grande, tornando a experiência novamente ruim para o usuário.

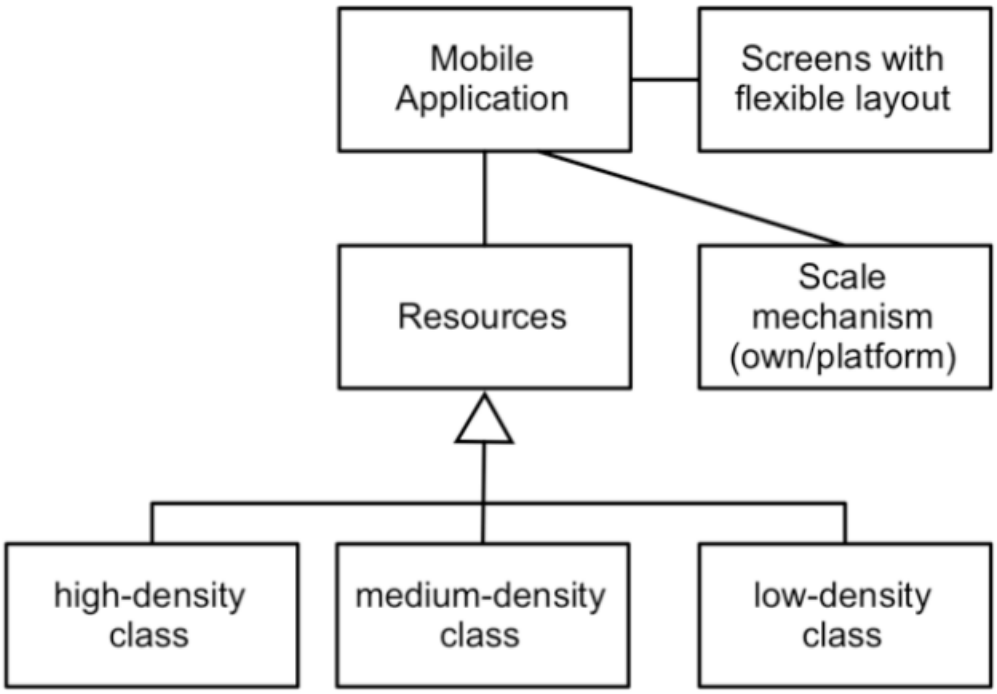

Figura 7 - Componentes da solução [35]

Betina et al [35] propõem o framework ilustrado na Figura 7 para resolver o problema enunciado. De acordo com o artigo, o programador deve focar seu desenvolvimento na tela de tamanho e resolução médias (HVGA 320x480, 3.0'' - 3.5'' diagonal, 160dpi) para reduzir a possibilidade da tela ser mal utilizada pela aplicação. Além disso, o framework defende o uso de variantes de densidade (high, medium e low). Ao invés da pasta resources da IDE utilizada possuir somente os arquivos de uma interface padrão, o usuário deve mapear, em diretórios diferentes, mudanças que sejam necessárias de acordo com a resolução da tela. O sistema operacional, Android, reconhece os parâmetros do aparelho e renderiza a interface adequada de acordo com as convenções de nome dessas pastas.

3.4 – Práticas de programação para economia de energia

Além da usabilidade, outro aspecto, que representa uma mudança de paradigma ainda mais drástica de programação, é a quantidade de energia que a aplicação requer. Muitas vezes, usuários descartam aplicativos ou deixam de usá-los com mais frequência, pois consomem muita energia. Baixo consumo de energia vem se tornando uma vantagem competitiva entre

aplicativos, e é atualmente considerado uma métrica de qualidade do software desenvolvido. Com isso, muitos estudos têm mapeado o custo energético do código da aplicação, como, por exemplo, Sahin et al [36], que desenvolveu um framework para mapear como o uso de padrões de design de software podem economizar bateria - esses padrões serão abordados mais adiante no texto.

O código para a plataforma Android é desenvolvido inteiramente em Java. Entretanto, restrições no consumo energético vão contra algumas habituais práticas da linguagem. Logo, frequentemente se estabelece um tradeoff entre a legibilidade do código e a eficiência energética da aplicação. Ding Li e William G. J. Halfond em seu texto [37] resolveram se aprofundar neste tópico. Eles analisaram o consumo de bateria em três principais situações: transmissão de dados pela rede, consumo de memória e práticas de programação de baixo-nível.

O primeiro parâmetro analisado pelos pesquisadores foram as requisições HTTP pelo dispositivo Android. Inicialmente, eles rodaram o código com o servidor "vazio", e assim mensuraram o custo energético base da conexão. Em seguida, rodaram o mesmo código diversas vezes e foram aumentando o tamanho do arquivo transmitido, de 1 até 4000 bytes. O resultado desta pesquisa é observado no gráfico da Figura 8.

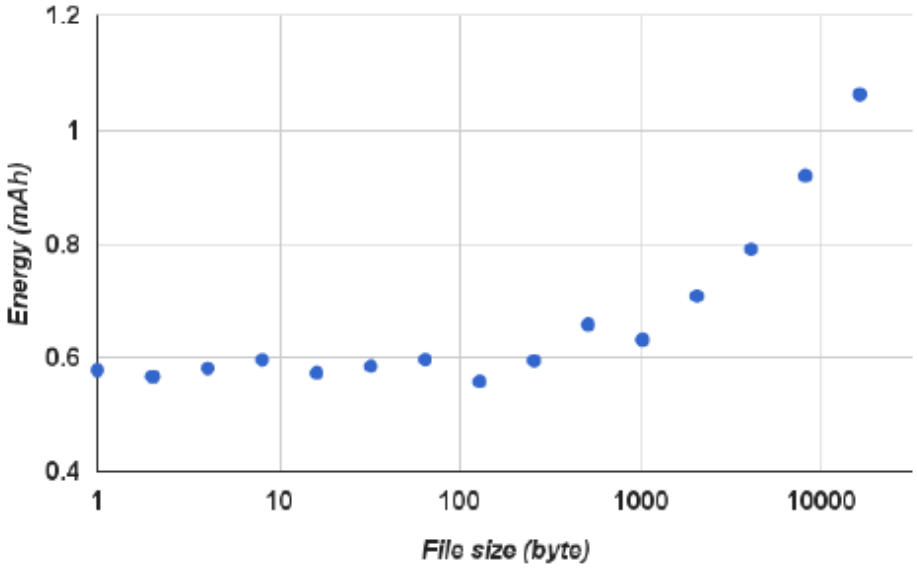

Figura 8 - Custo energético por tamanho do arquivo baixado [37]

É possível concluir, que o custo energético para transmitir um arquivo de 1 byte ou outro de 1000 bytes são equiparáveis. A explicação para essa baixa variação do custo energético dessas requisições se deve ao protocolo HTTP e suas camadas inferiores TCP/IP. Cada pacote enviado por esses protocolos à aplicação contém um cabeçalho (header), que tem

um tamanho fixo aproximado em 1024 bytes, e um controle, que depende da quantidade de pacotes enviados. Quando a informação contida no arquivo é pequena e cabe dentro do cabeçalho, não é necessário expandir o tamanho do pacote. Logo, para arquivos de até 1024 bytes, o pacote transmitido é basicamente o mesmo em termos de tamanho. A partir de 1024 bytes, a informação não cabe mais no cabeçalho e o gráfico começa a se comportar linearmente em relação ao tamanho por consumo energético. Desta forma, o programador deve evitar requisições que retornem um baixo volume de dados, ao invés disso, tentar agrupar requisições em um mesmo pacote de forma a consumir menos energia é recomendável.

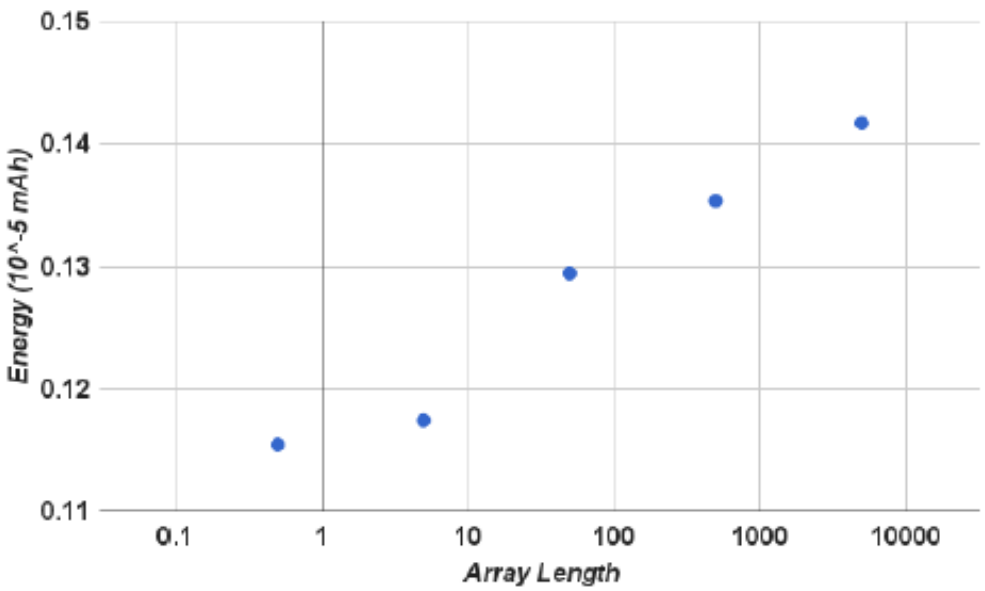

Figura 9 - Consumo de energia em diferentes níveis de uso de memória [37]

O segundo experimento deste mesmo trabalho avalia a relação memória por gasto de energia. Os pesquisadores implementaram um loop com uma simples instrução para atribuir o valor 1 ao índice do array acessado, onde variaram o tamanho do array de 512 a 5.120.000. A Figura 8 mostra a variação do consumo de energia em arrays de diferentes tamanhos. Por este gráfico, nota-se que o aumento substancial do consumo de memória não se traduz em um aumento de mesma ordem do consumo energético. Mais precisamente, pelo gráfico, infere-se que a cada 1.000.000% de aumento de memória, a energia gasta aumenta em 21,7%. Assim, conclui-se que alocar mais memória para o cache da aplicação, tiraria o peso de se acessar mais vezes o servidor, tornando a aplicação mais eficiente do ponto de vista energético.

Por fim, o estudo ainda avalia algumas técnicas de programação básica que podem causar gastos desnecessários à aplicação: acessar a cada iteração de um loop o tamanho do

array <array.length>, utilizar os métodos getters e setters para acesso de variáveis de classe e evitar a utilização de métodos estáticos desnecessariamente (ver Figura 10).

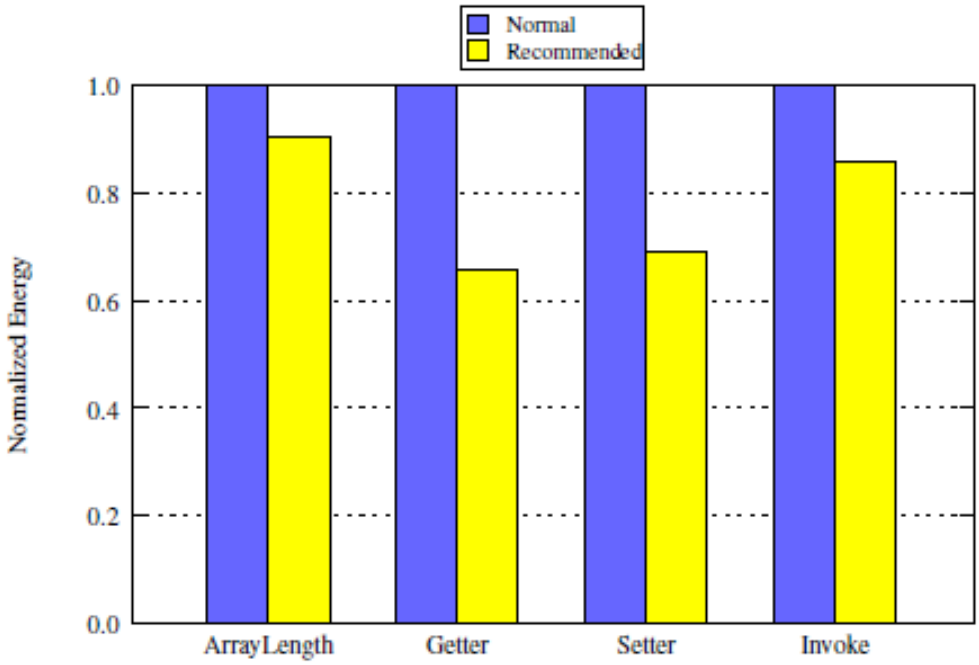

Figura 10 - Comparação energética entre práticas de desenvolvimento [37]

A Figura 10 exibe exatamente a comparação entre o uso das tradicionais práticas da linguagem – representadas em azul, com seu consumo normalizado no gráfico – e outras técnicas menos "custosas", representadas pela cor amarela. Os resultados reais são expostos abaixo:

- Observou-se um ganho energético de 10%, expresso por se referenciar o tamanho de um array fora do loop e utilizar esta variável durante as iterações, ou invés de processar o tamanho do array a cada iteração.
- Acessar diretamente os campos de uma classe, ao invés da utilização de métodos virtuais: getters e setters, apresentou uma economia energética de 31% para os setters e 35% para os getters. Isso se deve ao alto custo computacional, conhecido como overhead, que tais métodos possuem no sistema operacional Android.
- Utilizar métodos estáticos sempre que possível implica em uma redução do consumo energético de 15%. Entretanto, esta técnica representa uma péssima prática de programação.

A maioria dos recursos disponíveis na Internet para programação Android não se preocupa em gerar aplicações mais eficientes energeticamente. Logo, os desenvolvedores no geral desconhecem essas técnicas, que podem agregar muito valor a aplicação desenvolvida. Em seguida, é analisado outro estudo, que foca em melhorar o consumo energético da aplicação. Nele, são avaliados métodos e padrões da API do Android, que resultam em altos gastos energéticos, e alternativas e esses casos.

3.5 – Consumo de energia dos métodos da API do Android

O artigo de Linares-Vasquez et al [38] avaliou 55 aplicativos sob o ponto de vista energético. Neste experimento, os pesquisadores utilizaram o smartphone Nexus 4, com processador 1.5GHz quadcore Qualcomm Snapdragon S4 Pro CPU e sistema operacional, Android 4.2. Para alcançar melhores análises de performance do código, os pesquisadores utilizaram o Android Framework Tools que permite que as aplicações sejam rodadas remotamente em um computador. Através das especificações elétricas da bateria do celular (2100 mAh e 3,8V), eles chegaram a seguinte equivalência: 0.001 Joule = 3 * 10^{-5} % da bateria. O wattímetro utilizado para medir a dissipação energética do processamento computacional foi o Monsoon Power Monitor [39].

Tabela 2 – Métodos da API do Androide de alto consumo energético por categoria [38].

Category	# API Methods (%)
GUI & Image Manipulation	49 (37%)
Database	30 (23%)
Activity & Context	17 (13%)
Services	13 (10%)
Web	7 (5%)
Media & Animation	5 (4%)
Data Structure Manipulation	3 (2%)
File Manipulation	3 (2%)
Geo Location	2 (2%)
Networking	2 (2%)

Após executados os testes, obteve-se Tabela 2 que segrega por tópicos a quantidade de métodos “custosos” do ponto de vista energético mais utilizados por programadores. Como se

pode observar a maioria dos métodos diz respeito às áreas: GUI & Image Manipulation e Database. Juntos compõem 60 % das chamadas maior custo energético da API.

Em relação ao primeiro tópico, um método bastante utilizado, e que consome muita bateria, é o notifyDataSetChanged pertencente à classe ArrayAdapter. Ele notifica os observadores desta classe, que o conjunto de dados foi alterado, e por isso todas as Views, relativas aos observadores do Array, devem ser recarregadas, o que requer um alto processamento computacional, e consequentemente alto gasto energético. Este método já foi inclusive debatido no StackOverflow [40], porém, como não há alternativas diretas a sua utilização, não há também muito o que se possa fazer. Além dele, ainda há dois padrões nesta categoria que consomem bastante energia: <Toast.makeText(Context, CharSequence, int).show()> e <ProgressBar.setProgress(int)>. Ambos são widgets: a primeira exibe mensagens na tela do usuário, enquanto a segunda monitora o tempo para alguma tarefa ser executada. Respectivamente, esses padrões apresentaram um consumo de 0,007 J e 0,008 J.

Em seguida, avaliou-se as implicações das chamadas relacionadas a banco de dados. Neste tópico, utilizou-se o banco SQLite interno do sistema Android. Ao todo - desde a criação do banco com a chamada: SQLiteDatabase.openDatabase, passando pelo processo de gerenciamento, SQLiteDatabase.query, até a exclusão com ContextWrapper.deleteDatabase - foram observados mais de 30 métodos da API, cujo o custo energético se mostrou extremamente elevado. Desta forma, é mais eficiente utilizar arquivos XML para se persistir dados - pois contêm somente uma chamada custosa, XML.newSerializer, categorizada em File Manipulation -, ou utilizar o próprio arquivo SharedPreferences do sistema Android que persiste os dados em pares chave-valor. Os desenvolvedores devem ter em mente o real ganho que um banco relacional trará a sua aplicação, já que fazer uso dele é extremamente ineficiente.

Na categoria Data Structure Manipulation, um método muito utilizado nos aplicativos chamou a atenção: Bitmap.getPixel(int, int). Este método retorna somente um pixel representado na imagem por um ponto no plano cartesiano. Apesar de extremamente simples, ele se mostrou muito ineficiente energeticamente, e caracteriza o que se denomina como: energy bug, ou seja, chamadas que apresentam um injustificável consumo de energia. Como já debatido anteriormente, para evitar esta ineficiência, seria necessário tornar a variável pixel pública, de modo a possibilitar o acesso direto. O consumo é assim reduzido em torno de 30%.

Seguindo adiante, a categoria Web também possui métodos que devem ser evitados. A classe WebView contém os casos mais importante. Esta classe abre uma janela web na

aplicação em questão, entretanto quando movida para background, o desenvolvedor tem de chamar os métodos onPause/onResume para parar a execução da thread da Webview. Como esses métodos são escondidos pela classe, necessita-se da utilização de Java Reflections para acessá-los [41], o que introduz ao código mais um energy bug.

O padrão mais ineficiente energeticamente dos aplicativos avaliados pelo estudo [38] é o seguinte: <Activity.setContentView(int), Activity.findViewById(int), View.setVisibility>. O que o torna tão custoso é o simples fato do método Activity.findViewById(int) necessitar percorrer diversos arquivos XML para encontrar a View que o código requer, procurando em cada arquivo uma View que contenha o id passado para a chamada. Além disso, desenvolvedores tendem a utilizá-lo assiduamente. Em um dos aplicativo foram aferidas 57 chamadas a este método, o que levou a um consumo de 0,22 J, equivalente a $8 * 10^{-3}$ % da bateria, do celular Nexus 4, utilizado para as medições deste experimento. Vale ressaltar que, apesar de representar o consumo de um único método executado 57 vezes, os processadores dos smartphones mais modernos conseguem executar milhões de métodos em menos de um segundo. Logo, a porcentagem aparentemente irrelevante esconde um custo energético significativo. Novamente, é impossível evitar o uso deste método em sua aplicação, porém o programador pode criar variáveis globais para salvar as Views de retorno destas chamadas, limitando o seu uso.

O último padrão analisado se refere a categoria Networking: <ConnectivityManager.getNetworkInfo(int); NetworkInfo.isConnected()>. Este padrão serve para monitorar as redes disponíveis no celular do usuário. Ele consome em média 0,003 Joules. De modo a se poupar bateria, pode-se substituí-lo pelo método <ConnectivityManager.getActiveNetworkInfo()>, para cada rede monitorada [42].

Por fim, o trabalho de Linares-Vasquez et al, deriva do conjunto de aplicativos analisados cinco regras para o desenvolvimento Android com foco em eficiência energética:

- Desenvolver a estratégia de persistência de dados com muito cuidado. A ferramenta DBMS se provou extramemente ineficiente nesses casos.
- O uso do padrão Model-View-Controller (MVC) deve ser limitado, especialmente quando o aplicativo possui muitas Views, ou seja, telas. Carregar telas é uma operação comprovadamente ineficiente para a aplicação.
- Limitar o uso de widgets, como ProgressBar ou Toast. Já analisado anteriormente.
- Limitar a quantidade de telas da aplicação.

- Considerar sempre o trade-off padrões de design de aplicação e economia de energia. Apesar de muitos princípios serem extremamente custosos do ponto de vista energético, eles são, por outro lado, fundamentais para a legibilidade do código.

Assim, conclui-se um estudo sobre o desenvolvimento de aplicações móveis com baixo consumo de energia, uma métrica que vem influenciando a adoção de aplicações.

Capítulo 4
Implementação e Resultados

Este trabalho identifica a necessidade de uma plataforma que permita às pessoas gerenciar seus eventos de maneira simples. Muitas vezes, os eventos marcados através de redes sociais, que não se destinam somente a isto, são esquecidos pelos usuários. Desta forma, um aplicativo que exiba claramente todas as atividades sociais a que o usuário está convidado, torna mais prático o gerenciamento das mesmas. Além desta funcionalidade, os usuários também têm a opção de criar eventos diretamente pela plataforma, bastando preencher os dados descritivos dos eventos e publicar a seus amigos associados na plataforma. Estes amigos podem ser inicialmente carregados a partir da conta do usuário no Facebook. Futuramente o usuário poderá adicionar novos contatos de outras redes sociais.

4.1 – Arquitetura da Solução

Os estudos apresentados nos capítulos anteriores serviram como base para o desenvolvimento da aplicação a seguir. A arquitetura desenvolvida segue os conceitos de baixo consumo de energia. Mais adiante, no diagrama de classes, observa-se que não há métodos virtuais de getters e setters para algumas classes, pois, como os estudos mostram, representa um gasto excessivo de energia na aplicação. Além disso, houve uma preocupação com as requisições ao Facebook. Pelo código, observa-se um único ponto de acesso aos serviços REST do Facebook necessários para que a aplicação funcione. Essas requisições consomem muita energia, pois possuem um overhead de informações de cabeçalho e controle sempre que são utilizadas. Desta forma, opta-se por fazer uma única requisição e guardar em memória todos dados necessários à aplicação. Faz-se uso generalizado da memória do dispositivo, pois o gasto energético deste recurso é praticamente nulo.

A aplicação ainda utiliza a API do Google Maps para renderizar o mapa na tela. A estrutura Event da aplicação puxa do Facebook os dados de latitude e longitude do evento que serão convertidos em um balão em cima do mapa do Google Maps. Além disso, quando o usuário cria eventos pela aplicação, com um simples toque no mapa, ele gera a localização do evento, que, através da mesma API do Google Maps é traduzido em um endereço completo. Com o Google Places, é possível ainda inferir contexto semântico àquele endereço, ou seja,

determinar qual o tipo de estabelecimento e qual o seu nome, se é um bar, restaurante, hospital etc.

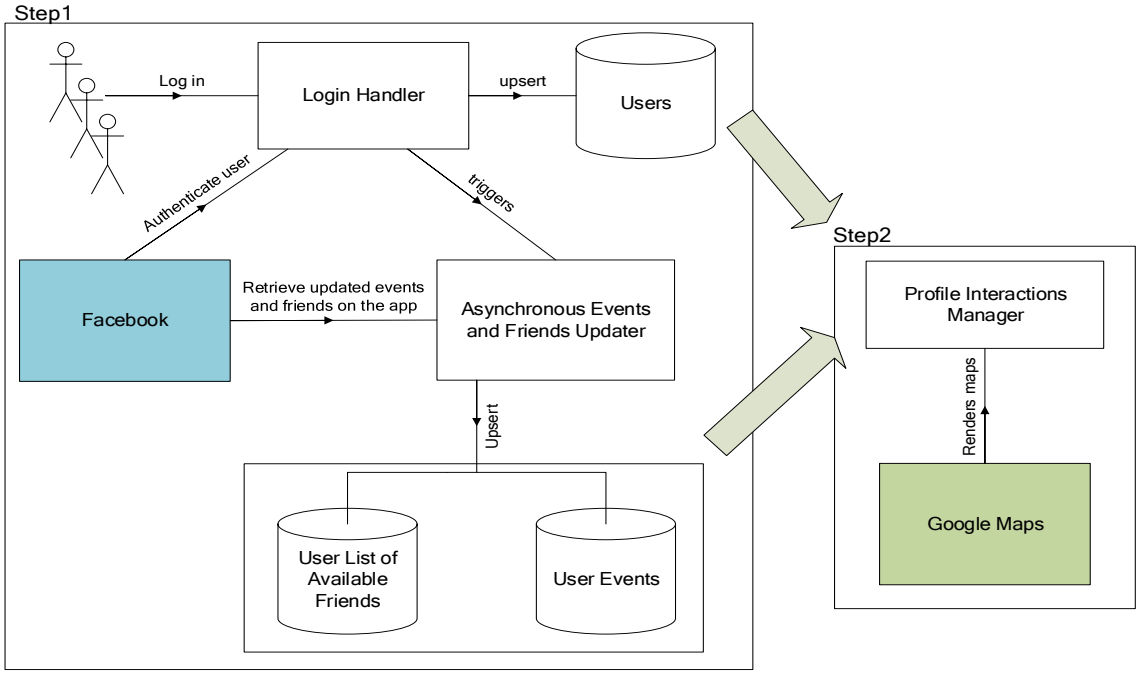

Figura 11 - Arquitetura do projeto proposto pelo aluno

A arquitetura da Figura 11 traz uma análise da solução desenvolvida. Nesta arquitetura, pode-se observar que existe um bloco, Login Handler, que é responsável pela autenticação do usuário na rede. Esse bloco faz uma requisição ao Facebook, que verifica a senha e o e-mail fornecidos pelo usuário à aplicação. Caso esteja tudo correto, o Login Handler atualiza os dados básicos do perfil do usuário no banco, como: e-mail, foto de perfil e nome. Vale lembrar que, caso o usuário não exista na aplicação, ele também será criado por este método. Além de atualizar ou criar o usuário na rede, quando ele é logado na aplicação, o Facebook fornece um Access Token para que seja possível acessar as permissões garantidas pelo usuário na página de Login. É comum todas essas permissões já serem apresentadas diretamente na etapa de Login. Assim, ele já saberá desde o início quais informações deverá compartilhar com a rede para ter acesso a todas as funcionalidades do sistema. Vale lembrar que o AccessToken informado quando o usuário é logado na rede, mantém-se por todas as sessões seguintes a não ser que ele se desconecte do Facebook dentro da aplicação. Assim, sua sessão é mantida mesmo que ele saia da aplicação, sem precisar confirmar as permissões toda hora que se logar nela.

Mesmo com os seus dados básicos persistidos não existe um mecanismo nativo de autenticação do usuário na rede. O banco de dados, User, somente atualiza as informações

dele de acordo com o Facebook. Além disso, caso o usuário pudesse logar nativamente, não haveria como puxar os eventos atualizados e a lista de amigos, que estão acessando a aplicação, pois não se teria o token de acesso para o banco do Facebook. Assim, para poder sincronizar as informações de eventos e amigos do usuário na rede, o Login Handler dispara o bloco de sincronização, Asynchronous Events and Friends Updater.

Este bloco utiliza o AccessToken gerado na primeira etapa para ter acesso a todas as permissões confirmadas no Login. Ele pega toda a lista de eventos a que o usuário está convidado, assim como seus amigos que também estão utilizando esta rede social. A estrutura de eventos recebe dados da localização do evento – alguns têm uma informação semântica embutida com o nome do que aquela localização representa, ou seja, se é um bar, restaurante etc –, data e hora, criador, status (se é público ou privado), foto de perfil do criador, descrição, nome do evento e foto do evento. Essas informações compõem a estrutura de eventos que provém do Facebook e é guardada no banco de dados.

A estrutura de usuários por outro lado é extremamente mais simples do que esta, pois somente se utiliza o nome e a foto de perfil do usuário. Por fim, estes dados são armazenados no banco, como indica a Figura 11. Deste modo, completa-se a primeira etapa do Framework proposto.

A segunda etapa fica responsável somente pela renderização das telas da aplicação, acessando os bancos e se utilizando da API do Google Maps para renderizar os eventos sobre o mapa. Quando for analisado o diagrama de estados da aplicação, na seção a seguir, será possível perceber as demais telas que compõem a segunda etapa do projeto.

4.2 – Arquitetura Orientada a Serviços (SOA)

O sistema desenvolvido é baseado nos conceitos de SOA (Service Oriented Architecture). Além de consumir os serviços do Facebook e do Google Services, o servidor da aplicação também é orientado a serviços web. Assim, os dados que chegam ao cliente para montagem da interface do usuário estão sempre no formato JSON. A figura a seguir ilustra como se montar uma arquitetura deste modo.

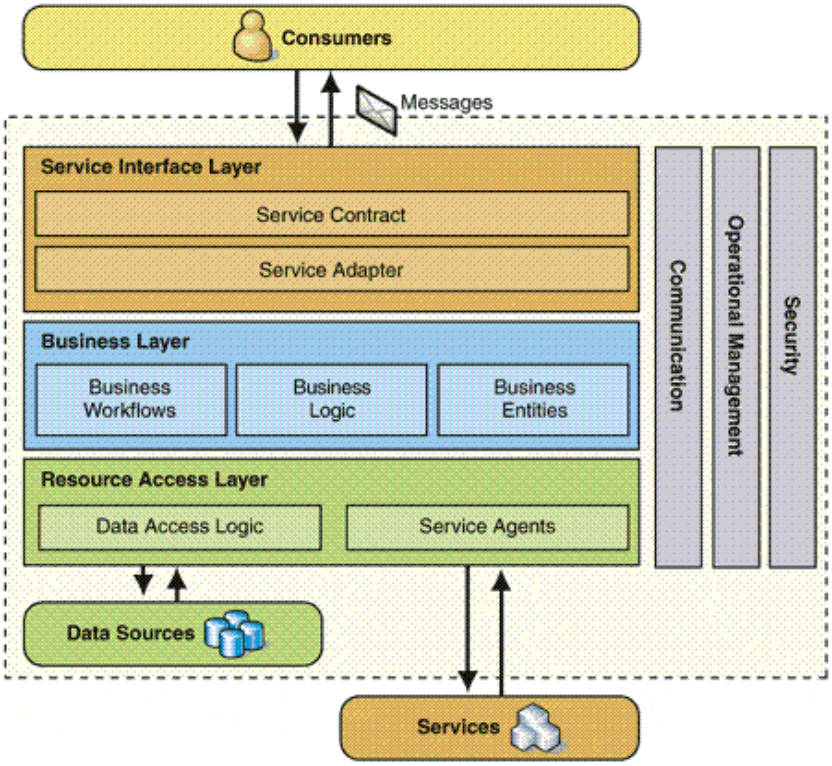

Figura 12 - Arquitetura Orientada a Serviços [46]

Observa-se que existe uma camada de acesso aos dados – Resource Access Layer – que é responsável por carregar as informações dos bancos de dados e os serviços da aplicação para o servidor. Na camada seguinte, executa-se toda a lógica de negócios em cima desses dados – Business Layer. Por fim, têm-se uma camada – Service Interface Layer – que gera o objeto a ser transferido para o cliente. Nesta aplicação, são utilizados objetos JSON para comunicar os módulos, porém, de modo a manter a independência entre a plataforma cliente e o servidor, poderia ser utilizado o WSDL por exemplo.

4.2 – Diagrama de transição de estados

Um entendimento melhor da dinâmica do sistema pode ser observado através do diagrama de transição de estados, apresentado na
Figura 13 - Diagrama de transição de estados da aplicação
. Através desse diagrama, consegue-se mapear todos os estados do sistema, analisando os eventos que ocasionam uma transição entre estados. Assim, o fluxo da aplicação se torna de fácil compreensão para todas as partes envolvidas no desenvolvimento. Esses diagramas entretanto não fornecem nenhuma informação quanto implementação desses estados ou quanto aos agentes que atuam em cada um deles, como o Google Maps, que exibe um mapa

na tela do usuário para um determinado estado, ou a API do Facebook, que é responsável pela tela de autenticação na aplicação, por exemplo. Esses estados podem ser considerados como caixas-pretas, onde somente se conhece a condição de entrada e de saída para um determinado estado. A

Figura 13 - Diagrama de transição de estados da aplicação

a seguir explicita um diagrama deste caso para a aplicação.

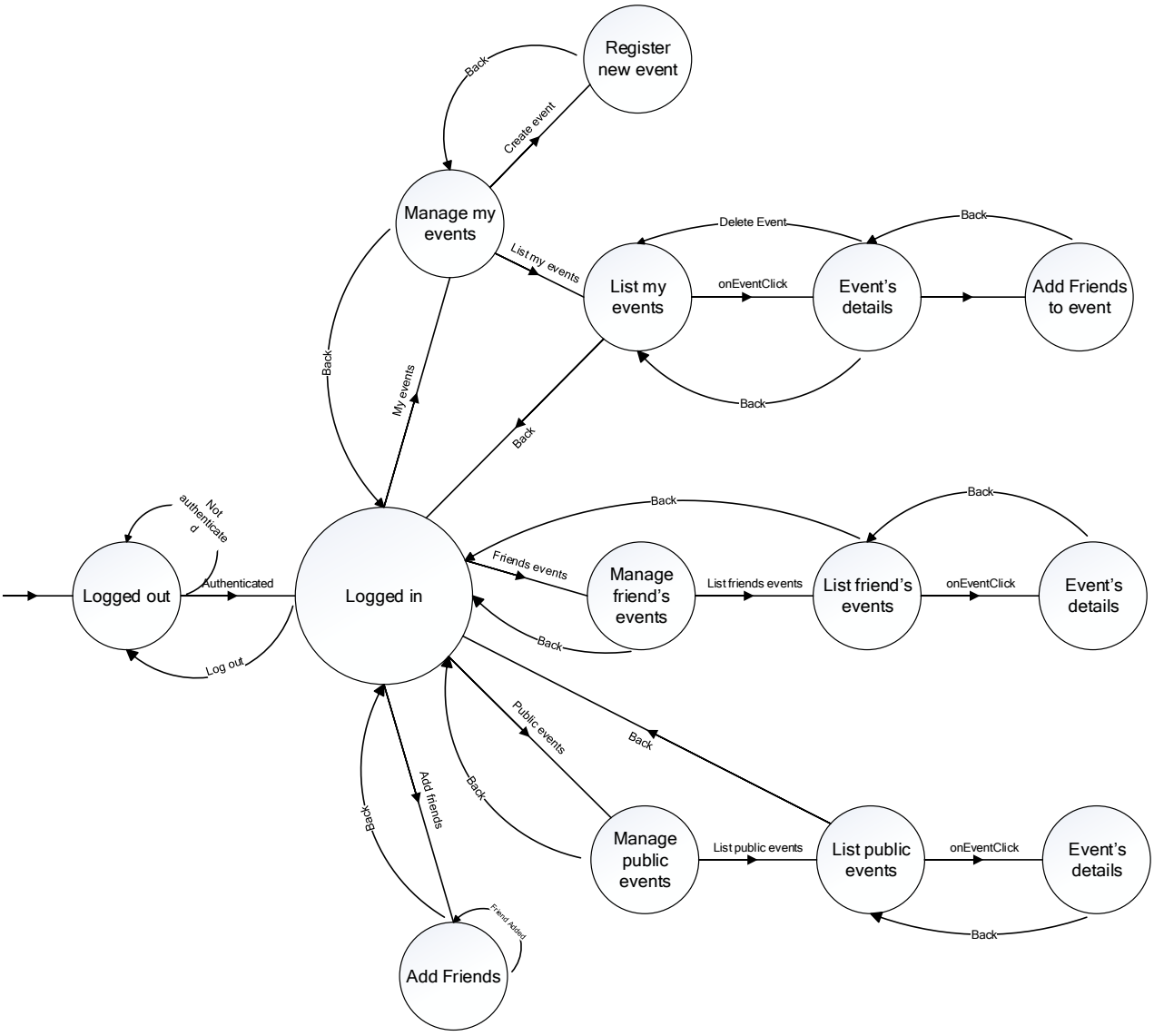

Figura 13 - Diagrama de transição de estados da aplicação

Cada estado do diagrama representa uma tela da aplicação móvel. O cenário inicial da aplicação é a página de Log in, que é representado pelo estado LOGGED OUT, pois esta tela somente aparecerá caso o usuário não tenha sido ainda autenticado pela aplicação. Nela, ele informa os dados de sua conta no Facebook. Caso a aplicação o autentique na rede do

Facebook, ele segue adiante para a tela que exibe o seu perfil em nossa rede social, representada no diagrama, pelo estado LOGGED IN. Caso não seja autenticado, continua preso na tela de Log in. A partir do estado LOGGED IN, o usuário pode seguir por diversos caminhos na aplicação: adicionar amigos, observar os mapas de eventos públicos da rede, de seus eventos privados ou dos eventos privados de seus amigos para os quais o usuário está convidado Estes mapas são fornecidos pela API do Google Maps, além deles a aplicação também utiliza os balões para identificar a localização de cada evento no mapa. Seguindo adiante, caso o usuário escolha o estado "ADD FRIENDS", ele receberá uma lista com seus amigos do Facebook que também estão conectados na rede aqui proposta. O usuário pode, então, navegar pela lista e escolher quais amigos ele pretende adicionar. A única opção de saída deste estado é apertando o botão de voltar ao estado anterior. Caso o usuário escolha qualquer um dos outros três estados: "MANAGE MY EVENTS", "MANAGE FRIENDS EVENTS" ou "MANAGE PUBLIC EVENTS", ele terá a sua disposição tanto o mapa com os eventos, como uma lista com a disposição dos eventos do caso escolhido, basta que ele passe o dedo na direção horizontal pela tela – gesto conhecido como *swipe* – para transitar entre esses estados. Dos estados seguintes aos de gerenciamento de eventos, ou seja, as listas de eventos, é possível o usuário selecionar um determinado evento para ter acesso a seus detalhes, transitando para o estado da nossa aplicação, "EVENT'S DETAILS". Para os eventos de amigos ou públicos, este é o último estado da máquina de estados. Se ele acessar os eventos criados por ele mesmo, existem mais opções de interação com o sistema: o usuário poderia excluir o evento em que está – e, desta forma, seria redirecionado para o estado anterior: a lista com seus eventos, "LIST MY EVENTS" –, ou seguir para um último estado de adição de amigos ao evento criado, "ADD FRIENDS TO EVENT".

4.3 – Diagrama de Classes

O diagrama de classes descreve a estrutura de um sistema. Ele é usado tanto para modelos conceituais da aplicação, quanto para modelos detalhados. Este diagrama descreve como as classes se relacionam umas com as outras no sistema. Cada bloco representa uma classe do sistema com seus métodos e atributos.

Nesta seção, são apresentados diversos trechos de diagrama de classe do sistema. Cada um representa um bloco da aplicação.

4.3.1 Login

O diagrama de classes da **Erro! Fonte de referência não encontrada.** apresenta um t recho relacionado com o Login da aplicação.

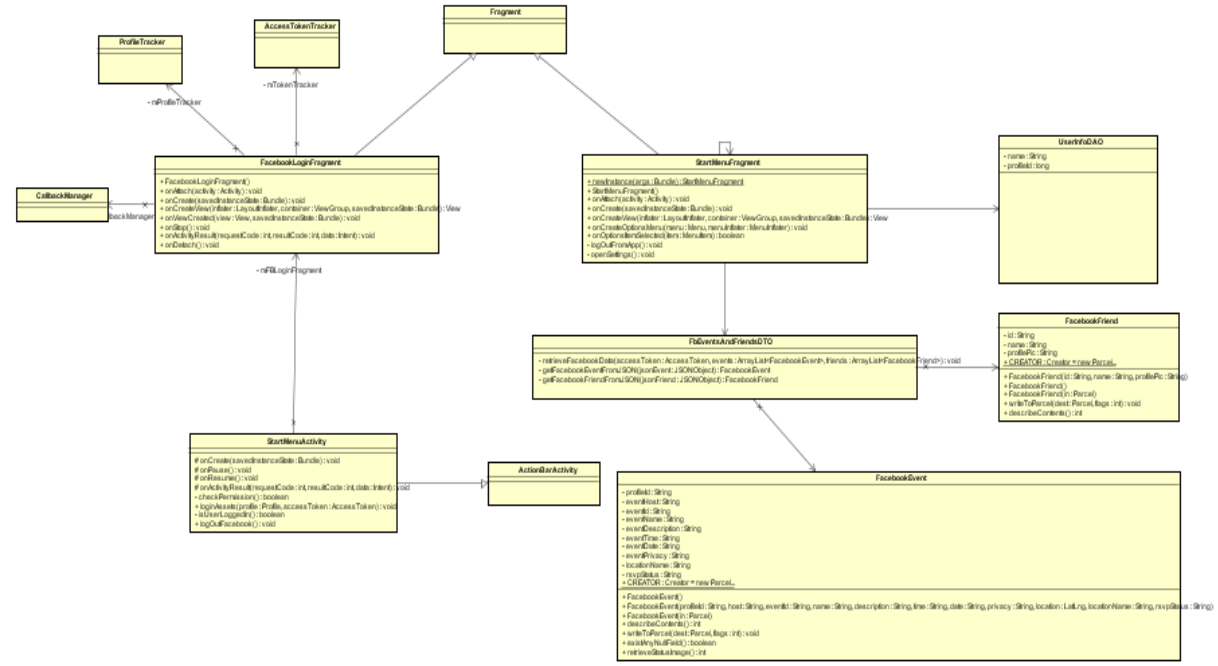

Figura 14 - Diagrama de Classes – Login

O ponto de início é a classe: StartMenuActivity. Esta classe possui o botão de login do Facebook, que é controlado pelo fragmento, FacebookLoginFragment. Caso o usuário nunca tenha logado na aplicação, ele o redireciona para a página de Login do Facebook e apresenta as permissões que deverá conceder à aplicação – elas são setadas no fragmento acima pelo método:

```
Button.setReadPermissions(Arrays.asList("user_events",
                                        "email","user_friends"));
```

Ele apresenta todas as permissões como um array de Strings. Ainda nesta classe, é definido um "callback" para tratar a validação das informações do usuário. Este callback retorna um dos três métodos: onSuccess, onCancel e onError. Desta forma, caso o login e senha sejam verdadeiro, retorna-se onSuccess, caso falso, onError e se o usuário cancelar a

operação onCancel. O AccessToken somente será gerado, caso o login seja bem-sucedido, como o snippet demonstra:

```
private FacebookCallback<LoginResult> mCallbackManager = new
FacebookCallback<LoginResult>() {
    @Override
    public void onSuccess(LoginResult loginResult) {
        AccessToken accessToken = loginResult.getAccessToken();
        Profile profile = Profile.getCurrentProfile();

        // Communicating with StartMenuActivity
        mListener.loginAssets(profile, accessToken);
    }

    @Override
    public void onCancel() {
        showAlert();
    }

    @Override
    public void onError(FacebookException e) {
        showAlert();
    }
}
```

Observando o código, nota-se que além da referência para o Profile e o AccessToken do usuário, ainda temos a chamada de um método, <mlistener.loginAssets>, que será responsável por encaminhar o usuário à aplicação, junto com as informações do perfil e da chave de acesso, que são passadas dentro de uma estrutura do tipo Bundle [43] para a próxima interface gráfica.

O usuário chega agora à tela inicial da aplicação, representada pela classe StartMenuFragment. Logo, que se inicia a classe, no método onCreate, o método <FriendsAndEventsDTO.retrieveFacebookData> é chamado. Ele irá buscar nos serviços RESTful do Facebook, todas as informações necessárias para o projeto e convertê-las do padrão JSON para classes Java, mais especificamente <FbFriends> e <FacebookEvents>. O snippet em seguida exemplifica como essa requisição é feita:

```
new GraphRequest(
  accessToken, "me/Events/attending", eventParams, HttpMethod.GET,
  new GraphRequest.Callback() {
    @Override
    public void onCompleted(GraphResponse g) {
      try {
         JSONObject jsonResponse = g.getJSONObject();
         JSONArray jsonData      = g.getJSONArray("data");
         for (int i=0; i < jsonData.length(); i++) {
           JSONObject jsonEvent = jsonData.getJSONObject(i);
           FacebookEvent event =getFacebookEventFromJSON(jsonEvent);

           // Check Events by date -> Compare to now
           if(event.getProfileId().equals(accessToken.getUserId())) {
             myFacebookEvents.add(event);
           } else {
             friendFacebookEvents.add(event);
           }
         }
      } catch (JSONException | ParseException e) {
          e.printStackTrace();
      }
    }
  }
);
```

Esse snippet representa somente uma parte do método retrieveFacebookData. Ele instancia um novo GraphRequest com as informações do accessToken, a URL do webservice, os parâmetros extras a serem passados à url, identificados por um objeto da classe Bundle, e o tipo de requisição a ser feita: POST, GET, DELETE, PATCH. Para este caso específico, são retornados somente os eventos que o usuário confirmou. Isto fica claro pela url passada: “me/events/attending”. Observe ainda que o retorno é um JSON que é mapeado através do método <FriendsAndEventsDTO. getFacebookEventFromJSON> para a classe FacebookEvent.

Esta interface ainda apresenta três botões, que expõem as funcionalidades principais do projeto: My Events, Friends Events e Add Friends. O diagrama abaixo mostra como essas classes se relacionam umas com as outras.

4.3.2 AddFriend

No diagrama abaixo da **Erro! Fonte de referência não encontrada.**, é possível observar que temos uma relação de associação entre a classe AddFriendActivity, que hospeda a View desta tela, e o AddFriendArrayAdapter, que é responsável por atualizar mudanças no conjunto de dados trazidos para a interface, neste caso o ArrayList<FacebookFriend>. Note que também há relações de associação das duas classes, AddFriendActivity e AddFriendArrayAdapter, com a classe FacebookFriend. A classe AddFriendActivity possui um array de FacebookFriend, e a AddFriendArrayAdapter recebe este array de FacebookFriend como parâmetro de seu construtor.

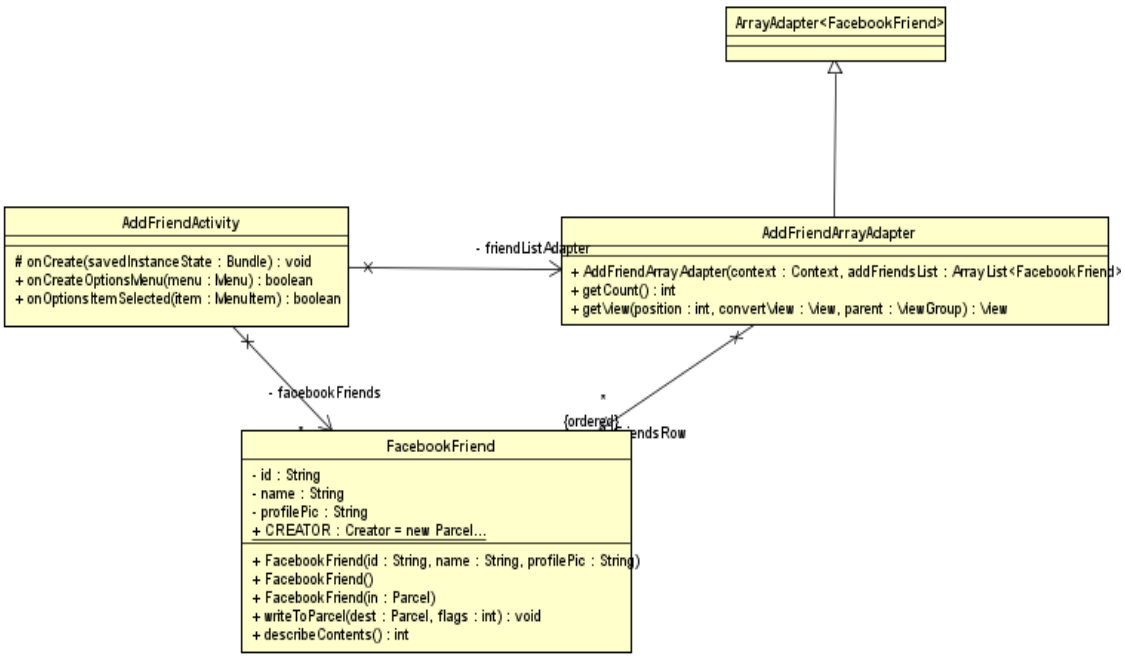

Figura 15 - Diagrama de Classes – AddFriend

4.3.3 MyEvents

O diagrama de classes para o MyEvents (Figura 16) é análogo ao do FriendsEvents. A estrutura das telas é a mesma, e o conjunto de eventos difere somente pelo criador do evento. Outro detalhe que será coberto a seguir é o fato de que na interface gráfica do MyEvents, o usuário pode criar seu próprio evento pela rede e escolher onde no mapa deseja colocá-lo.

.

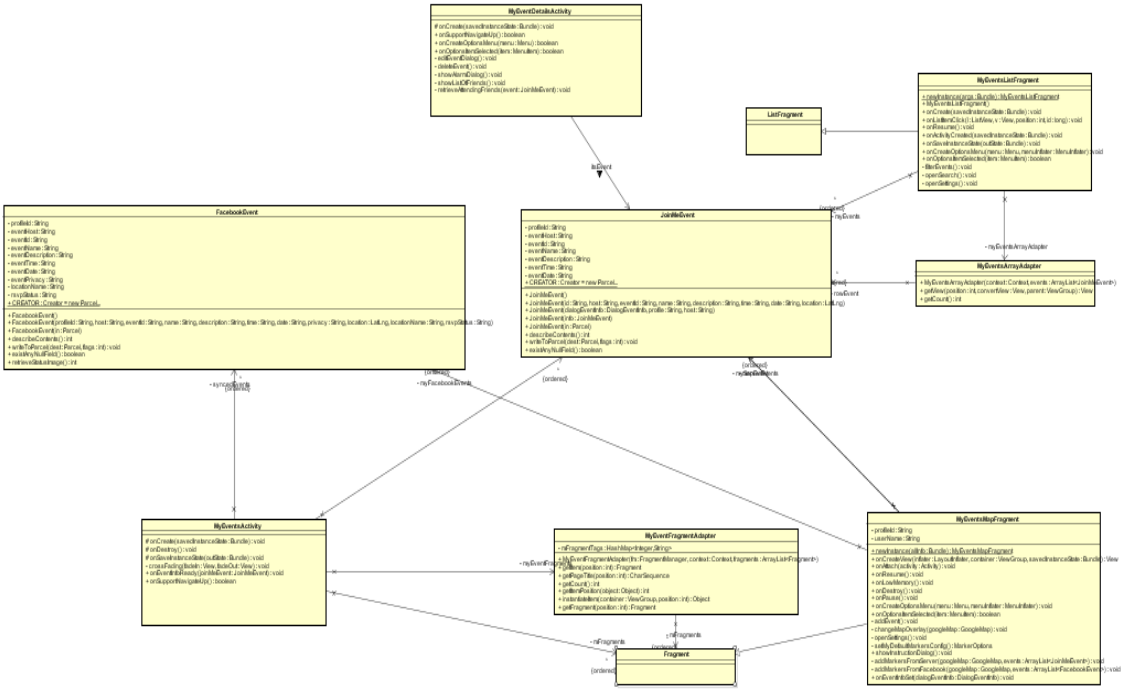

Figura 16 - Diagrama de Classes – MyEvents

Este componente da aplicação possui uma atividade que MyEventsActivity que permite fragmentos serem adicionados em um array e implementarem a interface gráfica de acordo com suas particularidades. A navegação entre os fragmentos se torna simples, basta passar o dedo de um lado da tela ao outro na posição horizontal, gesto conhecido do inglês como swiping. Neste caso, foram utilizados dois fragmentos: MyEventsMapFragment e MyEventsListFragment. Ambos fragmentos são adicionados em um ArrayList<Fragment>, como o código a seguir mostra:

```
private ArrayList<Fragment> mFragments = new ArrayList<>();

                . . . . .

mFragments.add(MyEventsMapFragment.newInstance(allInfo));
mFragments.add(MyEventsListFragment.newInstance(allInfo));
```

Assim, pode-se criar o MyEventFragmentArrayAdapter, que monitora mudanças nesses fragmentos e as reportam para a atividade, MyEventsActivity. Eles são associados no diagrama pois o adapter é um atributo da atividade, sendo instanciado no onCreate de nossa atividade da seguinte forma:

```
private MyEventFragmentAdapter myEventFragments;

                . . . . .

  myEventFragments = new MyEventFragmentAdapter(
                                getSupportFragmentManager(),
                                MyEventsActivity.this, this.mFragments);
```

Por este snippet, observa-se que o adapter somente é instanciado caso possua o array com os fragmentos que serão mostrados na aba. Desta forma, sua representação no diagrama de classes associa o adapter aos fragmentos. Esses fragmentos são associados às classes que providenciam os dados a eles: JoinMeEvent e FacebookEvent. MyEventsMapFragment está asssciado a ambas as classes, pois recebe as diferentes sortes de eventos através de um Bundle, vindo do MyEventsActivity

No entanto, na hora de se renderizar os eventos na tela do Google Maps, o evento somente informa a localização, criador e nome do evento. Na outra aba, o MyEventsListFragment agrega mais informação sobre cada evento da lista. Isso se deve porque esta aba interage, através do toque do usuário em algum evento da lista, com a classe MyEventDetailsActivity, que é responsável por mostrar todas as informações daquele específico evento clicado pelo usuário em outra interface gráfica. Esta classe está associada à classe JoinMeEvent. Os eventos da classe, FacebookEvent, são traduzidos para a definição de evento na rede, e assim, têm-se as informações importantes de cada evento, após clique do usuário, em uma nova tela. Desta forma, termina-se a explicação do diagrama de classes do MyEvents e consequentemente do FriendsEvents.

4.4 – Diagrama de Componentes

Este diagrama apresenta como os componentes de uma aplicação se relacionam estruturalmente[36]. Cada componente representado no diagrama possui um conjunto de interfaces das quais ele necessita para desempenhar sua função e expõe para os demais componentes as interfaces que ele disponibiliza para a aplicação. Assim, constrói-se o elo entre todas as partes que compõem a solução do projeto. Esta arquitetura ficará mais clara no diagrama da Figura 17 abaixo.

Os componentes do diagrama podem ser compostos por diversos esteriótipos, como: arquivos executáveis, <<executable>>; bibliotecas estáticas ou dinâmicas, <<library>>; banco de dados, <<database>>; tabela de uma base de dados, <<table>>; arquivo com código fonte ou dados, como .java, .xml, <<files>>; documentos genéricos<<documents>>. O nosso diagrama possui os estereótipos: libraries, databases, files e executables.

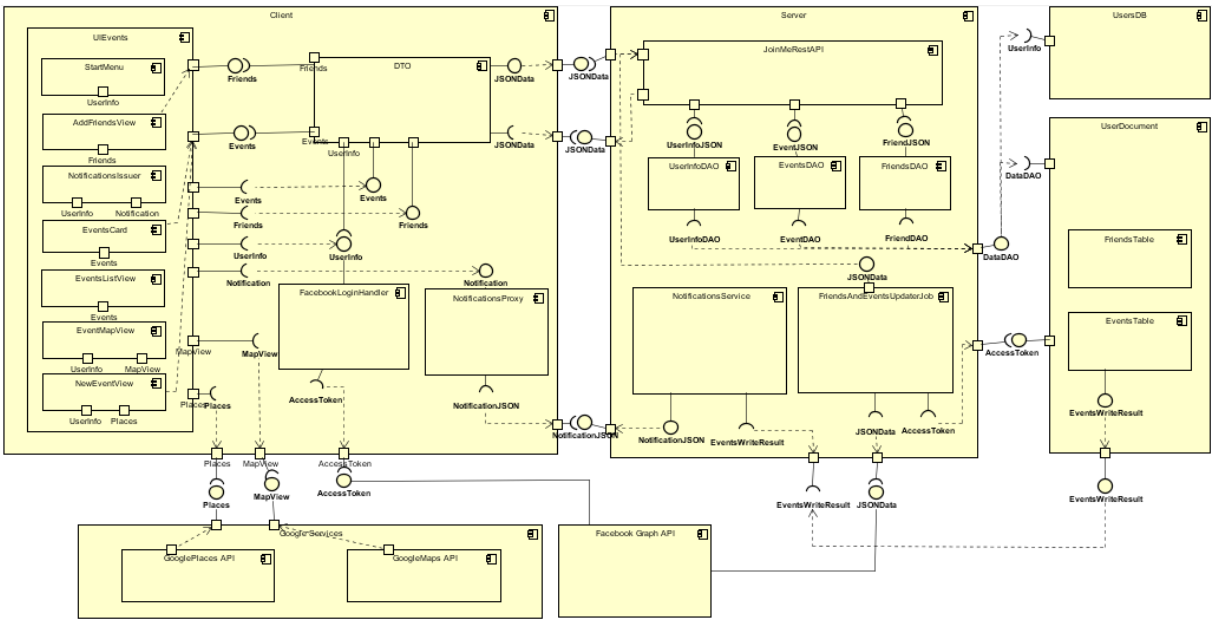

Figura 17 - Diagrama de Componentes

Esse diagrama fornece uma descrição de alto-nível da aplicação a ser desenvolvida, tornando-se fundamental para os desenvolvedores do projeto entenderem os passos iniciais do desenvolvimento. Além disso, por serem simples de se compreender e não se aprofundarem nos detalhes técnicos da aplicação, esses diagramas também são úteis para as diversas partes interessadas no projeto a ser desenvolvido[36]. Com base nisso, optou-se por utilizá-lo para esclarecer mais uma camada do desenvolvimento.

O diagrama está organizado da seguinte forma: o cliente, uma aplicação Android que roda em qualquer celular que possua uma versão do sistema operacional superior a 2.3; o servidor que será desenvolvido para a plataforma Ruby on Rails, tendo acoplado a si um banco de dados relacional – SQLite, por exemplo – para controle dos usuários do sistema e um banco NoSQL, como o Mongoid desenvolvido para ser utilizado pela plataforma Ruby on Rails; e os demais serviços web utilizados pelo sistema, como o Google Services e o Facebook. Em seguida, são explicados os componentes de cada parte do sistema.

4.4.1 Cliente

O cliente está organizado da seguinte forma: um componente responsável pelas Views da aplicação, UIEvents; Data Transfer Object (DTO), que comunica o servidor com o cliente; NotificationsProxy, que aguarda informações de notificação do servidor através de uma url; e o FacebookLoginHandler, que representa o ponto de entrada na aplicação, carregando os dados básicos do usuário: email, foto de perfil, nome, sobrenome e o AccessToken. Analisando cada um dos blocos individualmente, têm-se:

- **UIEvents**

O UIEvents depende de duas interfaces que são disponibilizadas pelos serviços do Google: o Google Maps, que disponibiliza o MapView para a renderização do mapa, e o Google Places, que auxilia o usuário durante a criação de eventos na plataforma. Além disso, as telas da aplicação ainda precisam de um objeto de notificação – utilizado pelo NotificationsIssuer, para informar o que foi alterado no conjunto de eventos armazenados no servidor –, dos dados de eventos e amigos, puxados também do servidor, e do UserInfo, atualizado sempre que o usuário alterar alguma informação de seu perfil no Facebook. Este componente contém módulos internos que representam as diversas views do sistema, são eles: StartMenu, AddFriendsView, NotificationsIssuer, EventsCard, EventsListView, EventMapView e NewEventView. Cada um, responsável por desempenhar uma função especifica de acordo com os estímulos do usuário.

- **FacebookLoginHandler**

O FacebookLoginHandler é utilizado sempre que o usuário se loga na aplicação, ele monitora os dados do perfil do usuário no Facebook e sempre que houver alguma alteração nesses dados, ele gera o objeto UserInfo já com essas alterações. Um exemplo deste tipo de alteração, seria a troca da foto de perfil do usuário no Facebook.

- **NotificationProxy**

O NotificationsProxy espera por dados enviados através de uma url pelo servidor. Estes dados novamente são objetos JSON que serão lidos e interpretados para objetos Java por este módulo. Assim o objeto gerado é enviado para a UIEvents para a criação de uma notificação para o usuário.

- **Data Transfer Object**

O Data Transfer Object (DTO) representa o ponto de comunicação com o servidor. Através dele, os dados inseridos ou alterados pelo usuário – criação, alteração e deleção de eventos, além de adição ou deleção de amigos – são transformados em um objeto JSON e publicados para uma url, que é observada pelo servidor. Da mesma forma, os dados salvos em documentos NoSQL no servidor são publicados para uma url que é observada pelo DTO. Desta forma, este componente ainda traduz objetos compreendidos pelo cliente para dados JSON e traduz esses dados JSON para objetos Java, utilizados pelo sistema operacional Android.

4.4.2 Servidor

O servidor foi desenvolvido para a plataforma Ruby on Rails. Sua função principal é armazenar os dados atualizados de cada usuário juntamente com os seus eventos e sua lista de amigos. Periodicamente, o servidor acessa os dados da plataforma do Facebook e os atualiza no sistema. Assim quando o usuário logar novamente na aplicação, só precisa acessar o servidor, agilizando o processo de aquisição de dados para o cliente. A seguir, são explicados todos os blocos que compõem o servidor.

- **JoinMeRestAPI**

Este componente é responsável por observar a entrada de dados por uma url através de um objeto JSON. Esses dados representam a interação do usuário com a plataforma, através da adição, deleção e atualização de eventos ou amigos. Esses dados são encaminhados para os objetos DAO, que acessam os banco de dados do sistema: o NoSQL(MongoDB) e o SQL (SQLite). Além desta função, este componente também publica para uma url os dados para gerar a interface do usuário no cliente. Esses dados chegam como um objeto JSON e são interpretados pelo módulo DTO, explicado no componente Cliente.

- **FriendsDAO, EventsDAO e UserInfoDAO**

Estes componentes criam objetos ruby, que podem ser utilizado para salvar dados diretamente no banco NoSQL do MongoDB. Particularmente, para o caso do UserInfoDAO é necessário que sejam salvados alguns dados no banco relacional da aplicação, como: email, nome, id e último acesso.

- **NotificationsService**

O NotificationsService monitora a função de upsert que os objetos DAO utilizam para inserir dados no banco NoSQL. Esta função avalia o conjunto de dados dos objetos e se houver algum item novo no conjunto retorna o valor 1. Ou seja, a função retorna um valor inteiro correspondente a quantidade de objetos inseridos no banco de dados. Assim, quando a função retorna algo diferente de zero, o NotificationsService gera um objeto JSON, que representa a notificação, e publica para uma url, que é acessada pelo NotificationsProxy do cliente.

- **FriendsAndEventsUpdaterJob**

O módulo FriendsAndEventsUpdaterJob é responsável por periodicamente acessar os serviços web disponibilizados pelo Facebook e salvar o que tiver sido alterado na plataforma do Facebook em termos de amigos, eventos ou informações pessoais do usuário. Assim, os dados recuperados por este componente se tornam objetos DAO e são atualizados no banco de dados. Novamente, sempre que houver um novo evento uma notificação é disparada para o cliente.

4.5 – Análise crítica sobre o sistema

O sistema integra a API do Google Maps e a API do Facebook para que se possa inferir informação de localidade aos eventos do usuário. Sempre que o usuário é iniciado na aplicação, a plataforma baixa a lista de amigos e eventos para verificar se há novas informações na base do Facebook. Desta forma, sempre é chamada assincronamente, após o Login, a tarefa que executa essas requisições, ocasionando um atraso na atualização dos dados. Enquanto a tarefa é executada, a Thread principal se mantém bloqueada por um ProgressBar, que a libera quando as chamadas terminarem de ser executadas. Assim, o usuário interage sempre com os dados atualizados das outras redes sociais associadas à aplicação – neste primeiro instante, somente o Facebook. A rede de amigos da aplicação é montada com base nos amigos que o usuário possui no Facebook. Os seus amigos, que também estejam utilizando esta aplicação, são listados através da funcionalidade Add Friends, na tela principal. Deste modo, o usuário escolhe os amigos que deseja adicionar na aplicação,

clicando em seu nome. A aplicação permite ainda que o usuário crie eventos através da própria plataforma e gerencie seus eventos do Facebook de maneira simples e rápida.

Apesar de não apresentar todas as funcionalidades de uma rede social, o sistema serve ao seu propósito, pois permite que sejam analisadas diversas características de uma rede social, como descrito no primeiro parágrafo.

No próximo capítulo, são sugeridas algumas propostas do que se deve pesquisar em próximos trabalhos para que a aplicação se torne mais robusta. Além das próprias conclusões do autor a respeito do tema estudado.

Capítulo 5
Conclusão

Este texto apresentou os conceitos para o desenvolvimento de uma rede social baseada em eventos para o sistema operacional Android. Esses conceitos foram abordados em um protótipo de rede social deste gênero. Inicialmente, avaliou-se o estado-da-arte de redes sociais baseadas em evento e localização, mostrando o que pode ser feito através das tecnologias de que se dispõe hoje sobre o assunto. Além deste estudo, outra área que se necessitou aprofundar o conhecimento para desenvolver uma aplicação mais eficiente, foi o próprio sistema operacional do Android. Explorar sua arquitetura e conhecer melhor a API e as técnicas mais eficientes em termos de consumo de energia garantiu mais segurança para o desenvolvimento de uma boa aplicação. Logicamente, o desenvolvimento também foi pautado na documentação disposta pelo Google para os desenvolvedores Android [44]. Nem sempre foi possível adotar as melhores práticas de desenvolvimento, analisadas no texto. Chamadas de alto custo energético, como findViewById, para retornar a View correta, e notifyDataSetChanged, para alterar o conjunto de dados exibidos na View, são muitas vezes necessárias, por não haver alternativas claras a estes métodos. Utilizou-se também getters e setters sempre que necessário para que o código continuasse legível. A seguir, são propostos alguns trabalhos futuros com base no que foi analisado nesse texto, porém não contemplado no protótipo desenvolvido.

O estado-da-arte de aplicações de redes sociais móveis traz conceitos de como se utilizar as informações do GPS para inferir contexto sobre cada usuário da aplicação. A aplicação desenvolvida roda em cima do Google Maps, porém não contém nenhuma forma de inteligência para inferência de contexto. Esta ideia deve ser analisada mais a fundo em próximos trabalhos. Pode-se, por exemplo, coletar do banco de dados todos os eventos a que o usuário tenha comparecido e depois, baseado nos eventos públicos disponíveis no servidor da aplicação, sugerir eventos que apresentem alguma correlação com os frequentados pelo usuário. Ou seja, escolher eventos que ocorram nos mesmos lugares que o usuário costuma frequentar, ou outros eventos criados por pessoas que o usuário conheça, ou cujas atrações se relacionem de alguma forma com aquelas dos eventos comparecidos. Além do contexto baseado em eventos, pode-se também criar um banco de dados para monitorar locais de

interesse do usuário, moldando um contexto de onde o usuário costuma estar e por onde costumar passar.

Os exemplos anteriores necessitam do processamento de alto volume de dados. Especialmente, no caso do mapeamento de contexto por localidade, ou seja, do monitoramento da informação do GPS para derivar pontos de interesse do usuário, é impossível – dada a quase infinita amostra de dados – executar tal processamento sem a utilização de uma ferramenta de processamento paralelo. Assim, a ferramenta Spark se torna fundamental para o desenvolvimento de um sistema eficiente de inferência de contextos.

O Spark necessita de um cluster com diversos workers – máquinas usadas para processar os dados – para montar seu ambiente de execução. Desta forma, é mais simples e barato montar uma arquitetura para o Spark utilizando um cluster na nuvem, um IaaS, como: AWS, Google, Digital Ocean etc. A linguagem utilizada no Spark se chama Scala e é inteiramente compatível com a linguagem Java, uma vez que, após ser compilado pelo scalac, o código Scala se transforma em bytecodes compreendidos pela Máquina Virtual Java (JVM). Isso possibilita que sejam utilizadas bibliotecas Java compiladas, ou seja, arquivos .jar dentro da aplicação Scala [45].

A arquitetura do Spark, aplicada para o projeto em questão, funciona da seguinte forma: monta-se inicialmente o SparkContext com base nas máquinas workers que o IaaS fornece para a aplicação; em seguida, carrega-se o conjunto de dados a ser processados – neste caso, esses dados vêm de bancos externos ao Spark – e os transformam em arquivos de dados RDD (Resilient Distributed Data); entra em ação o Spark Core – módulo responsável por todo gerenciamento dos workers da infra-estrutura – que distribui os RDDs entre eles, criando algumas redundância em caso de falha de algum worker ao processar seu conjunto de dados; são definidos então todo o conjunto de transformações (filtragem de dados) que serão aplicadas àqueles RDDs; avalia-se os conjuntos e o otimiza de forma a comsumir menos recursos computacionais; por fim, executa-se as tarefas paralelamente entre os workers do sistema [45]. Assim, é possível montar um sistema de inferência de contexto sobre os usuários do sistema, tornando a aplicação mais personalizada.

Além de todo o conhecimento de processamento paralelo, como demonstrado anteriormente para a ferramenta Spark, tornar a aplicação ciente a contexto também envolve o aprimoramento do modelo de banco de dados. Neste trabalho, apresentou-se por alto alguns paradigmas que os engenheiros vêm utilizando para modelar estes bancos de forma robusta o suficiente para comportar o volume de dados necessário a tais aplicações. Assim, também é

proposto como trabalho futuro um estudo mais detalhado de como se implementar bancos capazes de lidar com esse aumento substancial de dados na rede.

Por fim, este trabalho não seria possível sem a evolução da Internet, que possibilita atualmente as pessoas estarem conectadas a maior parte do tempo, compartilhando informações. Lógico que ainda há e sempre haverá restrições à conectividade, como há também com demais serviços de infraestrutura: eletricidade e água, por exemplo. Sarah Green, editora sênior associada da Harvard Business Review, define o futuro da Internet com a seguinte analogia [4]:

> *"... no futuro, a internet vai operar de forma similar a eletricidade hoje em dia, como parte de uma infraestrutura invisível que nos circula, e nós percebemos somente quando ela não está presente..."*.

Bibliografia

[1] D. M. Boyd. e N. B. Ellison, "Social Network Sites: Definition, History and Scholarship," *Journal of Computer-Mediated Communication,* vol. 13, p. 210–230, 2008.

[2] D. Trends. [Online]. Available: http://www.digitaltrends.com/features/the-history-of-social-networking/.

[3] Inc.com, "5 Predictions for the Future of Social Media," 27 May 2015. [Online]. Available: http://www.inc.com/aj-agrawal/5-predictions-of-the-future-of-social-media.html.

[4] CNBC, "11 Predictions on the future of social media," [Online]. Available: http://www.cnbc.com/2014/10/02/11-predictions-on-the-future-of-social-media.html.

[5] TechCrunch.com, "Facebook, Oculus And The Future Of Virtual Reality," 15 March 2015. [Online]. Available: http://techcrunch.com/2015/03/15/facebook-oculus-and-the-future-of-virtual-reality/.

[6] Y. Zheng, "Tutorial on Location-Based Social Networks," 2012. [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=163521. [Acesso em 2015].

[7] Y. Zheng, X. Xie, Q. Li e W. Ma, "Mining user similarity based on location history," em *ACM SIGSPATIAL GIS,* 2008.

[8] Y. Zheng, Q. Li, Y. Chen, X. Xie e W. Ma, "Understanding Mobility Based on GPS Data," em *In Proc. of UbiComp,* 2008.

[9] V. W. Zheng, . Y. Zheng, X. Xie e Q. Yang, "Learning from GPS Data for Mobile Recommendation," *Artificial Intelligence Journal,* April 2012.

[10] Ye, Y.; Zheng, Y.; Chen, Y.; Feng, J.; Xie, X.;, "Mining Individual Life Pattern Based on Location History," *IEEE,* 2009.

[11] X. Xiao, Y. Zheng, Q. Luo e X. Xie, "Finding Similar Users Using Category-Based Location History," em *In Proc. of .ACM SIGSPATIAL GIS,* 2010.

[12] X. Xiao, Y. Zheng, Q. Luo e X. Xie, "Inferring Social Ties between Users with Human Location History," *Journal of Ambient Intelligence and Humanized Computing,* 2012.

[13] B. C., "The Business Value of Mobile Applications for Meetings," 2011. [Online]. Available: http://www.corbinball.com/articles_technology/index.cfm?fuseaction=cor_av&artID=8647.

[14] A. M. Ahmed, T. Qiu, F. Xia, B. Jedary e S. Abolfazli, "Event-Based Mobile Social Networks: Services, Technologies, and Applications," *IEEE Access,* vol. 2, pp. 500-513, 2014.

[15] J. H. a. P. S. J. O'Madadhain, "Prediction and ranking algorithms for event-based network data," *ACM SIGKDD Explorat. Newslett.,* vol. 7, nº 2, pp. 23-30, 2005.

[16] S. Yuan, Q. Bai, M. Zhang e T. Win, "Discovery of core-nodes in event-based social networks," *6th IEEE Int. Conf. FSKD,* vol. 2, pp. 430-434, 2009.

[17] J. T. a. J. L. J. Zhang, "Expert Finding in a social network, in Advances in Databases: Concepts, Systems and Applications," em *Springer-Verlag,* Berlin, Germany, 2007.

[18] C. D. a. V. C. A. Gainaru, "A realistic mobility model based on social networks for the simulation of VANETs," *69th IEEEVTC,* pp. 1-5, April 2009.

[19] D. Karamshuk, C. Boldrini, M. Conti e A. Passarella, "Human mobility models for

opportunistic networks," *EEE Commun. Mag.*, vol. 49, nº 12, pp. 157-165, 2011.

[20] N. Y. a. Q. Han, "Context-aware community: Integrating contexts with contacts for proximity-based mobile social networking," *IEEE Int. Conf. DCOSS*, pp. 141-148, May 2013.

[21] "Gigaom," [Online]. Available: http://gigaom.com/2009/07/21/will-p2p-soon-be-the-scourgeof-mobile-networks.

[22] ARS, "Technica," [Online]. Available: http://arstechnica.com/tech-policy/news/2010/04/just-likecomcast-rcn-accused-of-throttling-p2p.

[23] S. -H. Gary Chan, J. Xing e W. -P. Ken Yiu, "Challenges and Approaches in Large Scale P2P Media Streaming," *IEEE Computer Society*, 2007.

[24] N. Fernando, S. W. Loke e W. Rahayu, "Mobile cloud computing: A survey," *Future Generat. Comput. Syst*, vol. 29, nº 1, pp. 84-106, 2013.

[25] P. Bellavista, R. Montanari e S. K. Das, "Mobile social networking middleware: A survey," *Pervas. Mobile Comput.*, vol. 9, nº 4, pp. 437-453, 2013.

[26] X. Liu, Q. Hey, Y. Tiany, W. -C. Lee, J. McPhersony e J. Han, "Event-based social networks: Linking the online and offline social worlds," *18th ACM SIGKDD Int. Conf. KDD*, pp. 1032-1040, 2012.

[27] B. N. Hari Krishna, *Android Architecture.*

[28] "Multilateration," [Online]. Available: https://en.wikipedia.org/wiki/Multilateration.

[29] G. P. System. [Online]. Available: https://en.wikipedia.org/wiki/Global_Positioning_System.

[30] "Navipedia: Tropospheric Delay," [Online]. Available: http://www.navipedia.net/index.php/Tropospheric_Delay.

[31] P. B. a. V. N. Padmanabhan, "RADAR: an in-building RF-based user location and tracking system," em *19th Annual Joint Conference of the IEEE Computer and Communications Societies*, Tel Aviv, Israel, 2000.

[32] "GSM," [Online]. Available: https://en.wikipedia.org/wiki/GSM.

[33] "Assisted Global Positioning System," [Online]. Available: https://en.wikipedia.org/wiki/Assisted_GPS .

[34] J. Y. R. a. H. S. Preece, *Interaction Design: Beyond Human- Computer Interaction*, 2 ed., Wiley.

[35] V. G. Bettina Biel, "Usability-Improving Mobile Application Development Patterns," Essen, Germany.

[36] C. Sahin, F. Cayci, I. Gutierrez, J. Clause, F. Kamilev, L. Pollock e K. Winbladh, "Initial Explorations on Design Pattern Energy Usage," em *1st International Workshop on Green and Sustainable Software*, June, 2012.

[37] D. L. a. W. G. J. Halfond, *An Investigation into Energy-Saving Programming Practices for Android Smartphone App Development*, Los Angeles, California.

[38] M. Linares-Vasquez, G. Bavota, C. Bernal-Cárdenas, R. Oliveto, M. Di Penta e D. Poshyvanyk, *Mining Energy-Greedy API Usage Patterns in Android Apps: An Empirical Study*, 2014.

[39] "Monsoon-Solutions," [Online]. Available: http://www.msoon.com/LabEquipment/PowerMonitor.

[40] "Stackoverflow," [Online]. Available: http://www.stackoverflow.com/questions/4715840.

[41] D. K. a. M. P. Robillard, “Detecting inefficient API usage,” *ICSE,* pp. 183-186, 2009.

[42] Google, “Determining and Monitoring the conectivity status,” [Online]. Available: http://developer.android.com/training/monitoring-device-state/conectivity-monitoring.

[43] Google, “Bundle | Android Developers,” [Online]. Available: http://developer.android.com/reference/android/os/Bundle.html.

[44] Google, “Android Developers,” [Online]. Available: http://developer.android.com/index.html.

[45]

[46] D. Modi, “https://dipenmodi.wordpress.com/,” 22 October 2007. [Online].