



Universidade Federal
do Rio de Janeiro
Escola Politécnica

ANÁLISE DOS MÉTODOS E DA INFRAESTRUTURA PARA O DESENVOLVIMENTO DA APLICAÇÃO MOBILE TIUPAR

Cristiano Mendes de Freitas

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

Rio de Janeiro

Setembro de 2016

ANÁLISE DOS MÉTODOS E DA INFRAESTRUTURA PARA O DESENVOLVIMENTO DA APLICAÇÃO MOBILE TIUPAR

Cristiano Mendes de Freitas

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:

Cristiano Mendes de Freitas

Orientador:

Prof. Flávio Luis de Mello, D. Sc.

Examinador:

Prof. Aloysio de Castro Pinto Pedroza, D. Sc.

Examinador:

Prof. Antônio Cláudio Gómez de Sousa, D. Sc.

Rio de Janeiro – RJ, Brasil

Setembro de 2016

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

Este exemplar é de propriedade da Universidade Federal do Rio de Janeiro, que poderá incluí-lo em base de dados, armazenar em computador, microfilmear ou adotar qualquer forma de arquivamento.

É permitida a menção, reprodução parcial ou integral e a transmissão entre bibliotecas deste trabalho, sem modificação de seu texto, em qualquer meio que esteja ou venha a ser fixado, para pesquisa acadêmica, comentários e citações, desde que sem finalidade comercial e que seja feita a referência bibliográfica completa.

Os conceitos expressos neste trabalho são de responsabilidade do(s) autor(es).

DEDICATÓRIA

Dedico a minha família, meus amigos e a minha linda namorada que me apoiaram nessa jornada. Sem vocês não seria possível.

AGRADECIMENTO

Agradeço ao meu pai, que me carregou até aqui praticamente sozinho. Seu esforço não foi em vão.

Também à minha mãe, que não pode estar presente, porém sempre sonhou em ter um filho formado.

À minha namorada Érika, que me fez retomar a faculdade, e tanto me apoiou nessa reta final.

Ao meu irmão Gustavo, que foi sempre solícito a me ajudar quando eu precisava. Lembro até hoje de quando você me levou ao fundão para conhecer em 2007, foi minha primeira ida a UFRJ. Ao meu irmão Caco e seus conselhos sobre estágio e sobre o curso de engenharia, sem eles eu não teria chegado aqui.

À Leila, que surgiu em minha vida como um presente, junto com a minha nova irmã Raquel e meu cunhado Jean.

À Margot por sempre me tratar como um filho.

Aos meus cinco sobrinhos que me deram tanta alegria de viver.

Aos meus amigos de colégio, Alvaro, Daniel, Fabio, Léo, Pedro Arlen, Pedro Farias, Pedro Junqueira, e principalmente ao Rodrigo, que sempre me apoiaram em todos os momentos.

Aos amigos de faculdade, especialmente Adriano, Ewerton, Leon, Mauricio, Matheus, Nilson, Niemeyer, Roxo, Silvino, Tiago, Vitão, além dos melhores amigos que alguém poderia ter, Arthur, Igor, Rafael Prallon e Timóteo.

Ao coordenador do curso de Eletrônica Casé e ao professor orientador Flávio Mello, sempre muito solícitos.

Também agradeço ao povo brasileiro que contribuiu de forma significativa à minha formação e estada nesta Universidade. Este projeto é uma pequena forma de retribuir o investimento e confiança em mim depositados.

RESUMO

O foco desse trabalho é analisar os métodos e a infraestrutura utilizados para o desenvolvimento de uma aplicação mobile, mais especificamente para o aplicativo mobile protótipo Tiupar. Neste sentido, esse projeto consiste na análise dos recursos e serviços que foram utilizados para a construção de um aplicativo mobile que liste diariamente eventos, permitindo também aos seus usuários filtragem por diferentes categorias, como a região onde o evento ocorre. Esses eventos, que são obtidos do banco de dados do Facebook, são encontros sociais previamente organizados na rede social como, por exemplo, uma festa ou show de uma banda. As informações que serão apresentadas nesse trabalho também podem ser aplicadas para outros tipos de aplicações que compartilhem das necessidades ou limitações de recursos que se apresentaram durante o desenvolvimento do aplicativo, dependendo da demanda.

Palavras-Chave: análise, aplicativo mobile, Tiupar, Facebook, eventos.

ABSTRACT

The focus of this work is to analyze methods and infrastructure for the development of a mobile application, specifically the mobile application Tiupar. In other words, this project consists in the analysis of resources and services used to build a mobile application that lists daily events and also allows users to filter these events by different categories, according to the region where the event is meant to occur. These events, which are obtained from the Facebook database, are social meetings previously organized on the social network, for example, a party or a show of a band. The information to be presented in this work can also be applied to other types of applications that share needs or resource constraints that appeared during the development of the application, depending on demand.

Keywords: analysis, mobile application, Tiupar, Facebook, events.

SIGLAS

API – Interfaces de Programação de Aplicativos

CSS – Cascading Style Sheets

EC2 – Elastic Compute Cloud

FQL – Facebook Query Language

HTML – HyperText Markup Language

HTTP – Hypertext Transfer Protocol

IDE – Ambiente de desenvolvimento integrado

JSON – JavaScript Object Notation

MySQL – Software de Banco de Dados relacional

PHP – Personal Home Page, Linguagem de Programação

RDS – Relational Database Service

SDK – Software Development Kit

Sumário

1	Introdução	1
1.1	Tema	1
1.2	Delimitação	1
1.3	Justificativa	2
1.4	Objetivos	3
1.5	Metodologia	3
1.6	Descrição	4
2	Fundamentos Teóricos	5
2.1	Desenvolvimento Mobile	5
2.1.1	Nativo para Android	6
2.1.2	Nativo para iOS	7
2.1.3	Híbrido PhoneGap	8
2.2	Facebook (API)	9
2.3	Serviços de Virtualização na Nuvem	12
2.3.1	Computação na Nuvem Privada	12
2.3.2	Computação na Nuvem Pública	13
2.4	O Objeto de Estudo	14
2.4.1	Contextualização do Problema	15
2.4.2	Funcionamento, Layout e Telas	17
2.4.3	Arquitetura do Aplicativo Protótipo	21

3	Análise	22
3.1	Nativos vs. Híbridos	22
3.2	Toolkit	24
3.2.1	Testando a Aplicação	25
3.2.2	Compilando o Aplicativo	27
3.2.3	Garantindo bom desempenho com o PhoneGap	30
3.3	Servidor	31
3.3.1	EC2	32
3.3.2	RDS	35
3.4	Utilizando a API do Facebook	26
4	Conclusão	39
4.1	Conclusão	39
4.2	Trabalhos Futuros	40
	Bibliografia	41
A	Pesquisa sobre Eventos	44

Lista de Figuras

2.1 – Distribuição de mercado para dispositivos mobile	1
2.2 – Comportamento dos entrevistados quanto à frequência de saídas para eventos	16
2.3 – Percentual de pessoas que costumam ter dificuldade para achar eventos em locais de seu interesse, excluindo os que afirmaram não irem a eventos	16
2.4 – Tela com a aba “Hoje” do protótipo	18
2.5 – Telas de ordenação dos eventos por popularidade e por região	19
2.6 – Tela com os detalhes de um evento	19
2.7 – Tela de eventos recomendados	20
3.1 – Obtendo IP para testar o projeto	27
3.2 – PhoneGap Developer App	27
3.3 – Restrições do plano gratuito do PhoneGap Build	29
3.4 – Restrições do plano gratuito de uma instância AWS EC2.	33
3.5 – Restrições do plano gratuito de uma instância AWS RDS	35
3.6 – Exemplo de requisição da listagem de eventos que ocorrerão numa casa de festas aplicada na plataforma Graph API Explorer.	37

Lista de Tabelas

2.1 – APIs suportadas pelo PhoneGap para cada dispositivo mobile com sistemas operacionais suportados	9
3.1 – Especificações Técnicas de Instâncias T2	34

Capítulo 1

Introdução

1.1 – Tema

Este trabalho tem como tema a análise dos métodos e da infraestrutura utilizadas para o desenvolvimento de uma aplicação mobile, mais especificamente para o aplicativo mobile protótipo Tiupar, desenvolvido pelo autor desse trabalho.

Neste sentido, o projeto consiste na análise dos recursos e serviços que foram utilizados para a construção de um aplicativo mobile que liste diariamente eventos na cidade do Rio de Janeiro e que permita aos usuários filtragem por diferentes categorias, como a região onde o evento ocorre. Esses eventos, que são obtidos do banco de dados do Facebook, são encontros sociais previamente organizados na rede social como, por exemplo, uma festa ou show de uma banda.

1.2 – Delimitação

Apesar de existirem vários tipos de aplicações mobile com diferentes funcionalidades, esse projeto se concentrará nos métodos, na infraestrutura e nos recursos que foram relevantes para o desenvolvimento da aplicação de listagem de eventos Tiupar.

No momento de início desse trabalho, o aplicativo Tiupar já havia sido desenvolvido pelo autor desse trabalho e já operava com usuários. Dito isso, esse trabalho não abordará as questões relativas à gestão de projetos de software que foram aplicadas em seu desenvolvimento, como o uso de práticas de engenharia de software

ou de métodos de desenvolvimento ágil. Sob esta ótica, o trabalho se limita ao estudo da infraestrutura de suporte para a aplicação previamente desenvolvida.

Essa aplicação se restringe aos eventos que podem ser obtidos diretamente pela API do Facebook, não buscando outras fontes. Desta forma, as informações que serão apresentadas nesse trabalho também podem ser aplicadas integral ou parcialmente para a construção de outros tipos de aplicações que compartilhem das necessidades ou limitações de recursos que se apresentaram durante o desenvolvimento do Tiupar, a depender de uma análise da demanda. Caso a necessidade seja muito diferente, outros métodos, recursos ou serviços podem ser mais vantajosos.

Esse trabalho também trata da obtenção de eventos do Facebook direto de seu banco, podendo ser uma funcionalidade de algum aplicativo mobile.

1.3 – Justificativa

Com a popularização de dispositivos mobile, também cresceu a demanda por aplicações mobile, e aqueles que possuem o conhecimento de como desenvolvê-los possuem uma vantagem no mercado.

Porém, existem muitas possibilidades com diferentes métodos e serviços para a construção de um aplicativo, gerando dúvidas de em quais situações eles serão mais bem empregados. Em especial, uma das situações que demandam melhor aproveitamento de recursos disponíveis é o com uma equipe enxuta, composta por apenas um desenvolvedor com poucos recursos financeiros.

Com isso, foi feita uma análise dos métodos e serviços utilizados no desenvolvimento do aplicativo mobile Tiupar, que se encaixou perfeitamente nessa demanda de utilização de recursos de forma otimizada para seu desenvolvimento, quando desenvolvido pelo autor desse projeto.

Além disso, percebeu-se que uma das funcionalidades mais importantes do Facebook, a rede social mais relevante atualmente, é o seu sistema de eventos. Por isso, também foi feita um detalhamento dos serviços utilizados para a extração periódica das informações desses eventos, podendo ser utilizadas livremente como uma funcionalidade interessante para algum novo aplicativo de eventos.

1.4 – Objetivos

O objetivo geral do trabalho é analisar os métodos e os recursos utilizados para o desenvolvimento otimizado do aplicativo Tiupar. Esse se caracteriza por ser um aplicativo que lista eventos que ocorrem na cidade do Rio de Janeiro com periodicidade diária, obtendo-os do banco de dados do Facebook, tendo sido desenvolvido de forma otimizada por apenas um desenvolvedor com poucos recursos financeiros.

Os objetivos específicos do trabalho são:

- Analisar as características e as vantagens e desvantagens dos diferentes métodos de desenvolvimento mobile;
- Avaliar o emprego de um método híbrido de desenvolvimento mobile no desenvolvimento do Tiupar;
- Analisar os tipos de serviços de provisionamento em nuvem;
- Analisar quais serviços de nuvem utilizados, sob a ótica de menor custo possível;
- Analisar qual linguagem foi utilizada para a construção do algoritmo de processamento do servidor, e qual a linguagem usada para o banco de dados;
- Estudar sobre os recursos da API do Facebook;
- Analisar como é feita a extração de eventos pela API do Facebook;
- Analisar gastos financeiros para o desenvolvimento do Tiupar;

1.5 – Metodologia

Para atingir o objetivo do projeto, primeiro foi necessário o aplicativo protótipo Tiupar, incluindo suas telas, infraestrutura e qual a motivação dele ter sido desenvolvido.

Isso levou a necessidade de analisar os métodos para desenvolvimento de aplicações mobile, sendo nativos ou híbridos, incluindo uma discussão de em quais

condições eles são mais bem aplicados, e o porque da aplicação de um método híbrido no desenvolvimento do Tiupar.

Após isso, foi preciso analisar as características do toolkit usado para o desenvolvimento da aplicação, incluindo ferramentas exclusivas do PhoneGap utilizadas para testar e compilar a aplicação.

Também foi necessário analisar os motivos que levaram ao uso de um servidor em nuvem, em especial os serviços em nuvem da Amazon. Foi analisada a função desse servidor, que é responsável por coletar eventos do banco de dados do Facebook com certa periodicidade, para que os eventos guardados no banco de dados sejam o mais relevante possível para os usuários do aplicativo mobile.

Por último, foi realizado um estudo da API do Facebook, que é o canal responsável pelo acesso as informações contidas no banco de dados do Facebook.

1.6 – Descrição

No capítulo 2 serão discutidos os fundamentos teóricos, incluindo informações sobre os métodos de desenvolvimento mobile, a API do Facebook, tipos de serviços em nuvem e uma análise sobre o problema que o aplicativo Tiupar se propõe a resolver, suas funcionalidades e suas telas para maior compreensão dos leitores.

O capítulo 3 apresenta uma discussão entre quando usar métodos de desenvolvimento mobile nativos ou híbridos, ferramentas do toolkit usado, quais os serviços usados para a construção do servidor e uma análise aprofundada sobre como a API do Facebook foi utilizada nesse projeto.

Por fim, a conclusão e trabalhos futuros serão explicitados no capítulo 4.

Capítulo 2

Fundamentos Teóricos

2.1 Desenvolvimento Mobile

O crescimento da Internet sem fio, somado à miniaturização de componentes eletrônicos, popularizou o conceito de dispositivos mobile com características de um “computador de bolso”. Anualmente, uma grande quantidade de modelos de smartphones e de tablets que chega ao mercado, incentivada por grandes fabricantes. Porém, ao analisarmos a figura 2.1, que apresenta um gráfico feito com dados [1] da distribuição do mercado para os sistemas operacionais instalados nesses dispositivos, notamos que os sistemas mobile da Apple e do Google dominam uma parcela bastante significativa do setor.

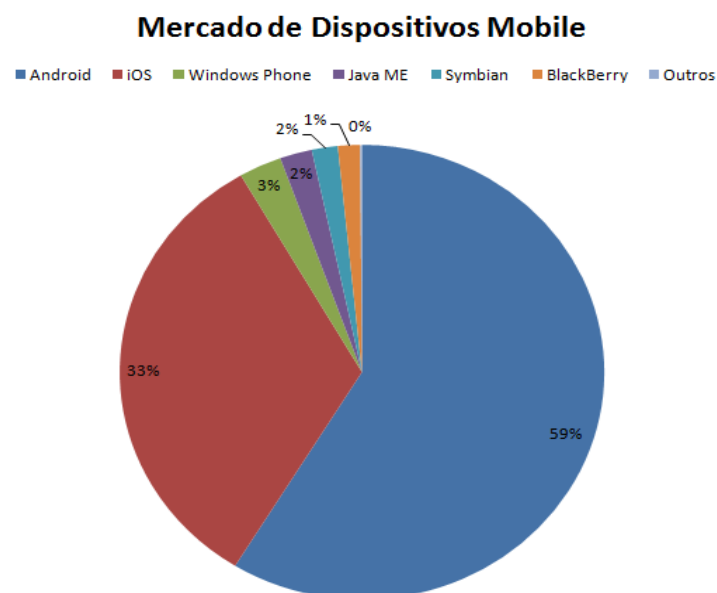


Figura 2.1 - Distribuição de mercado para dispositivos mobile

Fonte: Mobile/Tablet Operating System Market Share [1]

Podemos classificar as aplicações mobile pelo método utilizado para seu desenvolvimento. As nativas baseiam-se no desenvolvimento específico para cada sistema operacional. A principal vantagem é o desempenho elevado e a possibilidade de acessar todos os recursos do dispositivo mobile, como, por exemplo, a Câmera. Já como desvantagem, por utilizar as linguagens de desenvolvimento definidas pelas empresas proprietárias dos sistemas operacionais supracitados, é necessário que o desenvolvedor aprenda uma linguagem diferente para cada sistema operacional, além de ter que realizar a manutenção para múltiplos códigos em cada uma dessas linguagens.

No caso do Android, sua linguagem nativa padrão é a Java e seu IDE oficial é o Android Studio [2]. Para o iOS, sua linguagem nativa padrão é a Objective-C ou a Swift, e seu IDE é o XCode [3].

Já os aplicativos web são constituídos de apenas um navegador interno que carrega o conteúdo da aplicação vindo da web. Com apenas um código é possível obter o respectivo código empacotado pronto para ser instalado para cada uma das plataformas mobile existentes. Entretanto, esse método possui as desvantagens de baixo desempenho e de não permitir o acesso a recursos nativos do dispositivo.

Por último, existem os aplicativos desenvolvidos conhecidos como híbridos, que em sua maioria utilizam JavaScript, HTML e CSS. Eles são chamados assim porque não são nem aplicativos mobile nativos nem puramente aplicativos web.

Como vantagem, eles têm acesso aos recursos acessados por API de cada dispositivo, como os Nativos. Porém, em vez de utilizar interface de usuário nativa de cada sistema operacional, a renderização para exibição do layout é feita por meio de um navegador embutido no aplicativo, que interpreta seu conteúdo JavaScript, HTML e CSS, de forma similar aos aplicativos web. Isso permite desenvolvimento multiplataforma, utilizando o mesmo HTML para diferentes sistemas operacionais. Sua única desvantagem é um desempenho mais baixo que os nativos.

2.1.1 Nativo para Android

A Google, empresa de tecnologia responsável atualmente pelo desenvolvimento do Android, definiu como base no Android SDK [4] disponibilizado para os desenvolvedores que os aplicativos mobile para esta plataforma sejam desenvolvidos na linguagem de programação Java.

Cada aplicativo Android pode ser definido como uma soma de componentes distintos (cada “página” do aplicativo pode ser considerada um componente) que podem ser invocados individualmente. De um componente, é possível executar outro componente usando uma intenção (a cada mudança de “página” no aplicativo, é feita uma intenção). É possível até mesmo iniciar um componente em um aplicativo diferente, como a partir de um aplicativo de mensagens abrir um aplicativo de agenda para reservar um compromisso. Com isto temos vários pontos de entrada para um único aplicativo.

Até 2013, o IDE recomendado pelo Google para desenvolvimento de aplicativos para Android era o Eclipse [5] em conjunto com o Android Developer Tools (ADT) [6]. Porém, na conferência Google I/O de 2013, o Google promoveu o ambiente de desenvolvimento integrado Android Studio que ele vinha desenvolvendo para a versão 1.0 e anunciou o fim de todo o suporte ao Eclipse para desenvolvimento Android. Com isto, atualmente, o IDE oficial e suportado pelo Google para desenvolvimento de aplicações Android é o Android Studio.

2.1.2 Nativo para iOS

A empresa de tecnologia Apple, criadora e responsável pelo desenvolvimento do iOS, definiu que os aplicativos para esse sistema operacional devem ser desenvolvidos em Objective-C ou em Swift.

O Objective-C, que teve seu surgimento na década de 80, foi a primeira linguagem adotada para desenvolvimento iOS após o surgimento desse sistema operacional em 2007. É uma linguagem orientada a objetos com a maior parte de sua sintaxe vinda do C (definição de tipos primários de dados, declaração de funções...), adicionando sintaxe na troca de mensagens entre processos com base na linguagem Smalltalk.

Com influências do Objective-C e de outras linguagens, o Swift foi lançado em 2014 para fornecer uma alternativa apresentando uma sintaxe mais simples para os desenvolvedores. Ele possui funcionalidades interessantes como possuir telas que permitem exibir os resultados do código numa forma de animação (esse conceito se chama playground), permitindo testarmos funções ou pedaços do código instantaneamente. Além disso, possui uma performance melhor que o Objective-C, afora de ter seu uso mais intuitivo, facilitando o aprendizado por parte dos

desenvolvedores. Vale ressaltar que com o Swift é possível usar classes em Objective-C, apesar do contrario não ser verdade.

O IDE oficial, com suporte e desenvolvido pela Apple para o iOS é o XCode.

2.1.3 Híbrido PhoneGap

O PhoneGap [7] é uma plataforma de desenvolvimento que permite que desenvolvedores construam aplicativos mobile usando JavaScript, CSS3 e HTML5. O primeiro serve para tratar a lógica de programação e os recursos de dispositivos, enquanto que os dois últimos são utilizados para fazer a interface de usuário.

No caso do PhoneGap, sua estrutura é constituída por uma Interface de Programação de Aplicativos (API) base em JavaScript, servindo basicamente como uma carapaça para o aplicativo mobile. Essa carapaça instancia uma WebView, que é um navegador interno que permite a exibição do layout HTML, da mesma forma que numa página web. Esse navegador é que interpreta o CSS, HTML e JavaScript, e como resultado roda o que vem a ser a aplicação [8].

A WebView portanto também serve para realizar a conexão entre o código do aplicativo protótipo e a API do PhoneGap, que é a parte que possui acesso direto aos recursos do dispositivo mobile úteis à aplicação como GPS, rede, câmera, geolocalização, entre outros. A tabela 2.1 apresenta como o PhoneGap suporta os recursos para cada tipo de dispositivo mobile.

Esse acesso aos recursos nativos do dispositivo que podem ser necessários à alguns tipos de aplicativos é a grande vantagem de uma aplicação híbrida quando comparada com uma aplicação puramente web.

Graças aos seus recursos de JavaScript, ele pode realizar combinações com diversos scripts, incluindo JQuery.

A sintaxe do JQuery foi desenvolvida para tornar mais simples principalmente a seleção de elementos DOM (Modelo de Objeto de Documento) e a navegação do documento HTML. Utilizando esses recursos, é possível referenciar elementos do código com maior simplicidade, facilitando o desenvolvimento de aplicativos mobile complexos. Isto se mostra um recurso útil.

Tabela 2.1 - APIs suportadas pelo PhoneGap para cada dispositivo mobile com sistemas operacionais suportados. Fonte: PhoneGap Supported Features [9]

	iPhone / iPhone 3G	iPhone 3GS and newer	Android	Blackberry OS 6.0+	Blackberry 10	Windows Phone 8	Ubuntu	Firefox OS
Accelerometer	✓	✓	✓	✓	✓	✓	✓	✓
Camera	✓	✓	✓	✓	✓	✓	✓	✓
Compass	X	✓	✓	X	✓	✓	✓	✓
Contacts	✓	✓	✓	✓	✓	✓	✓	✓
File	✓	✓	✓	✓	✓	✓	✓	X
Geolocation	✓	✓	✓	✓	✓	✓	✓	✓
Media	✓	✓	✓	X	✓	✓	✓	X
Network	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Alert)	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Sound)	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Vibration)	✓	✓	✓	✓	✓	✓	✓	✓
Storage	✓	✓	✓	✓	✓	✓	✓	✓

✓ - supported feature
X - unsupported feature due to hardware or software restrictions

Em Outubro de 2011, o projeto PhoneGap teve seu código fonte doado pela Adobe para a fundação Apache sob o nome de Apache Cordova, que garantirá o futuro desenvolvimento do projeto. Com isto, a versão comercial do Cordova continua conhecida como PhoneGap [10].

2.2 Facebook (API)

Analisando historicamente o Homem, percebemos que ele pode ser definido em sua essência como um ser social, e como tal, desde o início da humanidade percebe-se a necessidade da formação das primeiras redes sociais presenciais com sua família, amigos e outros habitantes da mesma região que ele habitava.

Com o advento de novos sistemas de comunicação, em particular a Internet, surgiu o conceito de uma rede social online, que é por definição um conjunto de pessoas ou organizações, que se relacionam agora por possuírem interesses ou objetivos em comum com as outras. O aspecto principal desse tipo de rede é a horizontalidade e o caráter participativo, nos quais cada um é simultaneamente leitor e produtor de

conteúdo, e como tal não existe um conceito de hierarquia, ou seja, todos os usuários são importantes para a existência da rede.

Neste projeto trataremos de recursos de uma rede social online em particular: o Facebook. O serviço, lançado em 2004, é a maior rede social na atualidade, possuindo mais de um bilhão de usuários ativos.

Suas funcionalidades principais incluem o modelo de unir usuários em “amigos”, permitir que se personalize seu perfil de usuário com fotos e dados pessoais, possibilidade de se publicar informações numa linha do tempo que serão vistas por outros usuários conectados a você, um “sistema de eventos” para divulgar possíveis eventos ou acontecimentos criados por outros usuários e reunir pessoas que queiram participar desses, criação de paginas para divulgação de empresas ou celebridades, entre outras.

A partir destas funcionalidades, o Facebook lançou sua plataforma para desenvolvedores, em 2010, que consiste no conjunto de serviços, ferramentas e produtos fornecidos pela empresa de rede social para que desenvolvedores criem seus próprios aplicativos e servidores.

Dentre estas ferramentas se destaca a Graph API [11], uma API baseada em HTTP de baixo nível que é a principal forma de acessar e inserir dados dentro e fora da plataforma do Facebook.

A Graph API funciona com qualquer linguagem que possua uma biblioteca HTTP, podendo ser usada até mesmo diretamente no navegador (o que pode ajudar nos testes durante o desenvolvimento), a partir da sintaxe correta.

Esta API pode ser usada para tarefas que um aplicativo mobile pode precisar fazer no Facebook, como por exemplo:

- Consultar dados, já que o banco de dados do Facebook é rico em conteúdo social, incluindo, por exemplo, informações sobre o comportamento ou preferências de diferentes nichos, dados sobre eventos ocorrendo em alguma região, informações sobre locais numa determinada cidade, ou sobre relações pessoais entre determinados grupos;
- Postar conteúdo, como o Foursquare que permite postar a localização atual do usuário no Facebook;
- Enviar fotos, como o Instagram faz com sua integração ao Facebook;

- Gerenciar anúncios, como o próprio aplicativo Gerenciador de Anúncios do Facebook que serve para gerenciar os anúncios pagos feitos na plataforma;
- Gerenciar páginas, recurso utilizado pelo aplicativo Gerenciador de Páginas do próprio Facebook, que permite que a administração com mais precisão de uma página no Facebook por seu dono;

A maioria das possíveis requisições de Graph API requerem o uso de tokens de acesso que seu aplicativo mobile, aplicações internas no Facebook ou websites externos podem gerar ao implementar o sistema de Login do Facebook. Esses tokens são identificadores únicos, geralmente com um prazo de validade pré-definido, associados ao usuário ou página que os criou e que permitem um acesso a dados e funcionalidades restritos as permissões fornecidas a ele durante sua criação.

A Graph API segue um conceito de um “gráfico social”, com uma representação da informação contida no banco de dados do Facebook por:

- “Nós” - Basicamente "coisas" como um usuário, uma foto, um comentário ou uma página. Cada nó tem uma identificação única que é usada para acessá-lo;
- “Bordas” - As conexões entre estas "coisas", como fotos de uma página ou comentários de uma foto;
- “Campos” - Informações sobre essas "coisas", como o aniversário de uma pessoa, ou o nome de uma página;

O Facebook costuma lançar novas versões de sua API frequentemente, sendo que desde 2014 ele mantém uma periodicidade média de uma a cada três meses. Cada uma dessas versões apresenta novos padrões e funcionalidades, ou alterações na utilização dos recursos já implementados, sendo que elas funcionam em paralelo umas das outras até que sejam retiradas de operação, cabendo ao usuário optar por qual usar. O Facebook garante que uma versão fica em operação por no mínimo dois anos após seu lançamento [12], e no máximo por dois anos após o lançamento da versão seguinte. Depois desse período ela é retirada de vez de uso pelo Facebook, não podendo mais ser acessada.

2.3 Serviços de Virtualização na Nuvem

Com o surgimento dos computadores no século passado, o armazenamento e o processamento de dados digitalmente, outrora feito apenas manualmente em documentos físicos, passaram a ser uma possibilidade interessante e uma necessidade crescente. Esses no início ocorriam apenas localmente, pois eram feitos em redes locais conectadas fisicamente com amplitude de acesso bastante restrito. Apenas membros ligados fisicamente a essas redes poderiam acessar dados.

Com o advento da Internet, e com as possibilidades de compartilhamento a ela associada, formou-se pela primeira vez o conceito de computação na nuvem.

Esse se refere à utilização dos recursos de armazenamento, cálculo e memória de computadores e servidores compartilhados e interligados pela Internet, seguindo o princípio da computação em grade.

O armazenamento de dados é feito em bancos de dados alocados em servidores que poderão ser acessados de qualquer lugar do mundo, a qualquer hora, não havendo necessidade de instalação de programas ou de armazenar dados localmente. O acesso a serviços, programas e arquivos é feito virtualmente de forma remota, através da Internet - daí a alusão à nuvem.

Como o acesso é virtual, os usuários não precisam estar conectados fisicamente, abrindo novas possibilidades criação de projetos, gestão de empresas e nas relações atuais de trabalho entre funcionários, sendo possível, por exemplo, trabalhar de qualquer lugar desde que conectado a Internet.

O uso desse modelo pelas características descritas acima para a construção de servidores é mais viável do que o uso de unidades físicas. Esse se divide pelos seus dois modelos de implementação, chamados de privado ou público [13].

2.3.1 Computação na Nuvem Privada

Na computação na nuvem privada, o próprio cliente que compra ou aluga todo o hardware e software necessários para construir o serviço. Ela fica localizada dentro do ambiente protegido (não necessariamente fisicamente, mas virtualmente dentro da rede) do cliente e o acesso é feito virtualmente podendo ser por meio da Internet, limitado apenas a quem ele permitir.

Esse modelo de implementação se caracteriza por uma excelente autonomia na estrutura do servidor, pois o cliente possui acesso a todo o servidor implementado, podendo modificar qualquer parte da arquitetura se necessário.

Também possui é a melhor escolha para empresas ou projetos que possuam a necessidade de níveis mais rigorosos de segurança e privacidade, pois apenas a empresa dona do servidor que pode autorizar quem tem acesso a este, motivo pelo qual organizações como bancos costumam recorrer a esse tipo de serviço.

Por ultimo, dentre suas características, possui uma disponibilidade teoricamente alta, pois é como o hardware esta localizado fisicamente no próprio cliente ou em uma empresa diretamente parceira, o cliente pode garantir de perto a manutenção do servidor online o tempo todo.

2.3.2 Computação na Nuvem Pública

Já na computação na nuvem pública, os servidores são alocados em data centers externos, instalados em provedores de serviços na nuvem, e não no cliente. Toda a infraestrutura, incluindo os equipamentos e aplicações, que compõem esses servidores é compartilhada por milhares de clientes em todo o mundo, por intermédio da Internet.

O conceito principal que define esse modelo é que o usuário pague apenas pelo que utiliza dos serviços oferecidos pelo provedor a uma série de clientes.

Dentre as principais características, se encontram:

- Instalação rápida – para começar a utilizar a nuvem não é necessário se preocupar com a instalação do hardware. Somente é necessário o provedor aprovar o acesso do cliente aos serviços, feito em poucas horas ou dias;
- Custo definido apenas pelo uso - com essa modalidade de cobrança, só serão cobrados os serviços consumidos pelo tempo que forem utilizados. Com isto, o cliente não precisa se preocupar com contratos fixos de custo elevado ou com a subutilização dos recursos contratados;
- Atualizações automáticas nos sistemas e no hardware fornecido - é comum que as atualizações aconteçam com frequência e a manutenção dos servidores seja periódica. Isso acontece porque os equipamentos e estruturas são utilizados por vários usuários, e não somente por um cliente, facilitando o processo de descoberta de possíveis falhas. Além disso, constantemente os hardwares são atualizados conforme a demanda

do mercado. Os mesmos servidores são utilizados por diversas empresas e assim, uma atualização beneficia todo um ecossistema de utilizadores;

- Suporte técnico terceirizado – clientes que não possuam conhecimento especializado em hardware de servidores se beneficiam do suporte técnico terceirizado, que geralmente é oferecido pelos provedores de serviços. Com ele, poupam-se recursos relacionados a TI, que podem ser aplicados em áreas mais importantes ao projeto;
- Escalabilidade – esse modelo possui uma grande flexibilidade de atender a uma demanda crescente de uso, dependendo apenas da capacidade de infraestrutura do provedor de serviços;

Entre os líderes do mercado nesse ramo estão gigantes de tecnologia já conhecidas como o Google, Microsoft e a Amazon.

Esta última, apesar de conhecida pelos seus serviços de venda online de produtos como livros, desde 2006 criou sua plataforma de computação na nuvem conhecida como Amazon Web Services (AWS) [14].

Esta, em 2015, foi classificada no relatório “Magic Quadrant for Cloud Infrastructure as a Service” no setor de “Líderes” e como a empresa com a visão mais abrangente e com a maior capacidade de execução do setor, dentre todas as concorrentes [15]. Nesse mesmo ano atingiu um faturamento aproximado de US\$5 bilhões, com mais de um milhão de clientes.

Além disto, possui servidores em 12 regiões diferentes no mundo, garantindo assim a possibilidade do cliente obter um servidor mais próximo, e consequentemente um desempenho melhor. Um desses servidores inclusive se encontra em São Paulo, beneficiando o mercado brasileiro.

2.4 O Objeto de Estudo

A fim de auxiliar aos seus usuários na escolha de alternativas de diversão, foi concebido esse aplicativo mobile na forma de um divulgador de eventos com opções de filtros por categorias como, por exemplo, o local do evento. Com esse aplicativo, será diminuída a distância entre os frequentadores de festas e de eventos e seus respectivos

organizadores. Os potenciais usuários incluem pessoas interessadas em frequentar festas, shows ou eventos culturais, com idade média entre 18 e 30 anos no Brasil. Esse público procura um serviço que lhe transmita informações sobre eventos e que o auxilie a tomar decisões.

Nesta seção, serão apresentados todos os aspectos relacionados ao desenvolvimento do aplicativo mobile. Para tanto, num primeiro momento abordam-se os motivos que conduziram à concepção da aplicação, bem como as oportunidades de mercado correlatas. Em seguida, passa-se a descrever o aplicativo mobile: seu funcionamento, seu layout e suas telas. Por fim, é explicada a arquitetura do protótipo.

2.4.1 Contextualização do Problema

As noites brasileiras são marcadas por uma pluralidade de festas e de eventos simultâneos. Frente a essa imensa quantidade de alternativas de diversão, as pessoas muitas vezes têm dificuldade em descobrir aonde melhor aproveitar seu tempo de lazer. São pessoas que têm dinheiro, porém, por não conseguirem descobrir eventos que correspondam a seus gostos em locais que sejam do seu interesse, muitas vezes ficam em casa.

Esse problema ficou evidenciado por meio de uma pesquisa realizada pelo autor desse trabalho em outubro de 2015 (ver Anexo A). Elaborou-se um formulário online que foi enviado via e-mail para a lista de alunos da Escola Politécnica da UFRJ contendo perguntas variadas referentes aos seus comportamentos em relação a entretenimento noturno. Durante trinta dias, coletou-se 269 respostas desses estudantes universitários. Vale ressaltar que a escolha dessa lista como foco para a pesquisa se deu porque esses universitários possuem um perfil aderente com o de usuários alvo do aplicativo mobile.

Primeiramente, conforme demonstra a figura 2.2, buscou-se saber com que frequência os entrevistados saíam para festas. Descontando os 17% que responderam que não costumavam frequentar esse tipo de evento já que não representam potenciais usuários do aplicativo mobile, com alguns alegando motivos como somente encontrarem locais cheios ou caros, o total restante de entrevistados serviu de base para a análise principal: perguntou-se se essas pessoas tinham dificuldade em se informar de eventos em locais de seu interesse. O resultado demonstrado na figura 2.3 foi que dos universitários dentro do perfil desejado 44% responderam positivamente à pergunta

precedente. Isso foi considerado um percentual significativo de pessoas com dificuldade, pois isso demonstra que quase metade das pessoas desse perfil poderiam ser futuros usuários do aplicativo mobile.

Você costuma ir a eventos (festas, baladas)?

■ sim, sempre ■ sim, às vezes ■ não

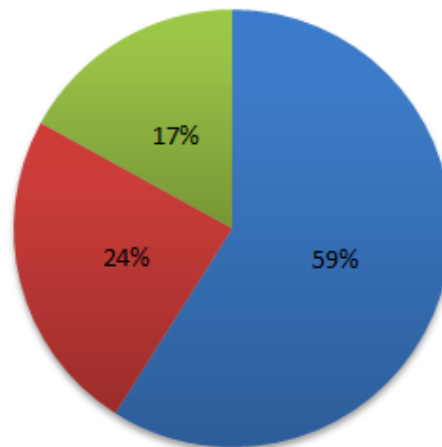


Figura 2.2 – Comportamento dos entrevistados quanto à frequência de saídas para eventos

Costuma ter dificuldade de achar eventos nos arredores ou em lugares que te interessam?

■ Sim ■ Não

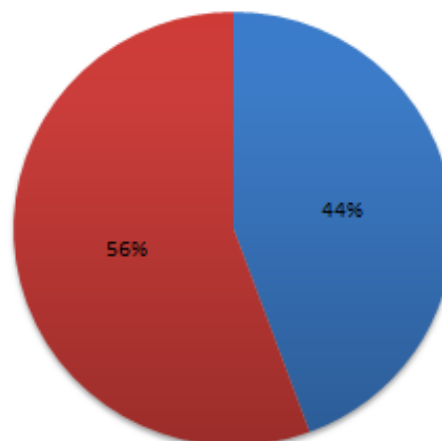


Figura 2.3 – Percentual de pessoas que costumam ter dificuldade para achar eventos em locais de seu interesse, excluindo os que afirmaram não irem a eventos

Também foi feito um levantamento quanto ao mercado de eventos. As oportunidades oferecidas pelo mercado de eventos brasileiro podem ser evidenciadas por seu enorme crescimento nos últimos anos. Segundo pesquisa [16] divulgada pela Associação Brasileira de Empresas de Eventos (ABEOC) e pelo SEBRAE, o mercado de eventos gerou, em 2013, R\$ 209,2 bilhões, num total de 590 mil eventos realizados. É um segmento que representa 4,3% do PIB do Brasil. Os estados da região Sudeste responderam por 52% do total de receita movimentado pelo setor. Sua enorme taxa de crescimento também chama a atenção. De 2002 a 2013, no Brasil, os valores movimentados pelo setor apresentaram uma expansão de 567%, passando de R\$ 37 bilhões para R\$ 209,2 bilhões.

A demanda por divulgação online é outro dado relevante. No que concerne ao uso da internet para publicidade, espera-se que em 2016 ela represente uma fatia de 27% da totalidade do mercado de anúncios [17]. Segundo estudo realizado pelo Content Marketing Institute feito em 2013, as empresas que lidam com divulgação para consumidores (B2C marketers) têm empregado as mídias sociais com mais frequência para divulgar conteúdo, com destaque para o Facebook (89%) e para o Twitter (80%), constatando-se um incremento relativo da publicidade online em detrimento da off-line [18]. Ademais, mencione-se que, no último trimestre de 2013, o Facebook registrou um ganho de US\$ 1,25 bilhão tão somente com seu serviço de anúncios, o que correspondeu a 53% do total de receitas do período [19].

Nota-se, destarte, que as conclusões da pesquisa realizada pelo autor deste trabalho com potenciais usuários, bem como os dados acima apresentados acerca da expansão do setor de eventos e da demanda crescente por publicidade digital indicam uma oportunidade de mercado para o serviço prestado por uma ferramenta online divulgadora de eventos.

2.4.2 Funcionamento, Layout e Telas

O protótipo apresenta ao usuário uma lista com todos os eventos do dia, para que ele se informe de todas as opções disponíveis. Esse conjunto de eventos é obtido diretamente do banco de dados do Facebook e, por enquanto, está restrito à cidade do Rio de Janeiro. Vale ressaltar que, como ainda não foi implementado nenhum sistema de cadastro de usuários, as informações disponibilizadas pelo aplicativo mobile não

podem ser personalizadas, de modo que a lista de eventos é uma só para todos. O aplicativo mobile possui uma versão protótipo para Android e para iOS.

A figura 2.4 mostra a tela inicial do aplicativo. Na aba “Hoje”, é exibida uma lista de todos os eventos ocorrendo no dia atual na cidade do Rio de Janeiro, por padrão enumerados por ordem de hora de início do evento. Essa aba contém informações como horário de início, total de pessoas e um cálculo do percentual de mulheres e de homens que confirmaram presença no evento do Facebook, além do local onde ele será realizado.



Figura 2.4 - Tela com a aba “Hoje” do protótipo

Conforme a figura 2.5, é possível organizar os eventos nessa aba por ordem de popularidade, ou seja, pela quantidade de pessoas que confirmaram presença no evento no Facebook. Também é possível filtrar os eventos por diferentes regiões na cidade, como zonas sul, oeste e norte, e centro.

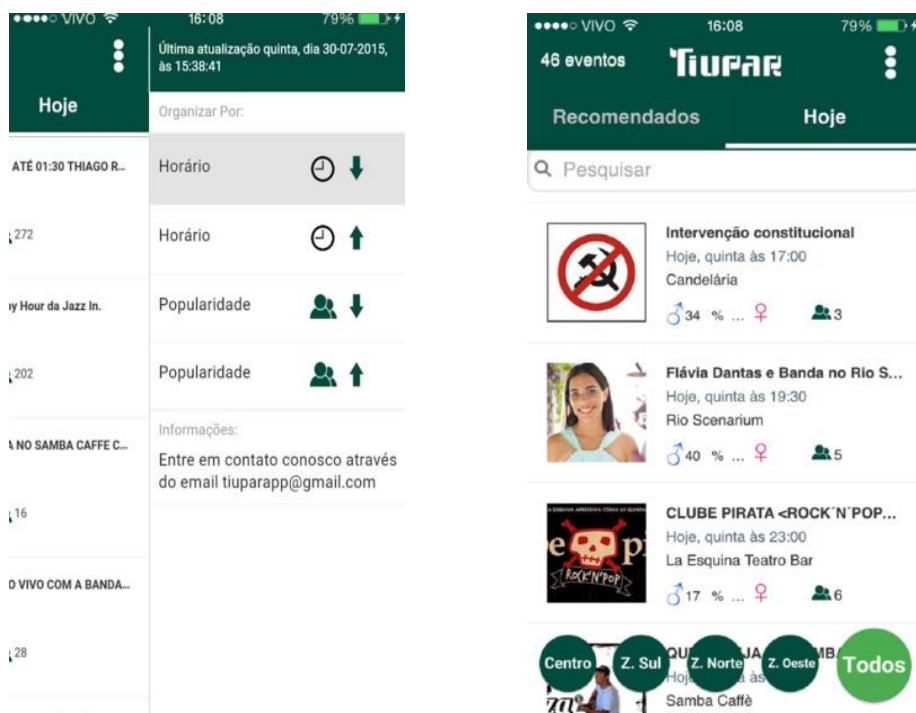


Figura 2.5 - Telas de ordenação dos eventos por popularidade e por região

A figura 2.6 mostra a tela referente aos detalhes de um evento, que se abre quando se clica num deles. Nela, estão contidas informações mais detalhadas, para além das presentes na lista da tela inicial, como o endereço do local onde o evento ocorre, bem como a descrição desse último.



Figura 2.6 – Tela com os detalhes de um evento

Durante a fase de testes, alguns produtores de festas, que souberam da existência do aplicativo e gostaram da proposta, entraram em contato, buscando aparecer com destaque no aplicativo e, assim, obter maior visibilidade para seus eventos. Desse modo, foi criada uma aba denominada “Recomendados”, apenas para eventos parceiros, conforme pode ser visto na figura 2.7.

Os referidos produtores, em troca de figurarem com destaque nesse aplicativo, acabaram por servir para a divulgação do protótipo, com benefícios aos usuários (descontos na entrada, sorteios de entradas VIP, etc.), muito embora não tenhamos recebido qualquer contrapartida financeira de nenhum deles. Graças a isso, o aplicativo contou com mais de 1000 (mil) downloads no total nas lojas online Google Play e App Store, no período de agosto de 2015 até dezembro de 2015.



Figura 2.7 - Tela de eventos recomendados

Por conseguinte, mesmo inicialmente sem um plano para monetizar o projeto, foi notado que havia uma demanda por publicidade de eventos, o que representa claramente uma grande oportunidade de mercado que pode ser futuramente aproveitada como um negócio.

2.4.3 Arquitetura do Aplicativo Protótipo

A arquitetura de um aplicativo mobile pode ser dividida em duas partes. A primeira sendo a interface do aplicativo mobile, ou seja, o pacote que pode ser baixado via as lojas virtuais App Store e Google Play que o usuário instala no seu dispositivo mobile, que foi desenvolvida utilizando a ferramenta de desenvolvimento híbrido PhoneGap.

A outra parte é composta pelo servidor, que processa e armazena todos os dados a serem utilizados pela aplicação. Nele foram utilizados alguns dos serviços de computação em nuvem da Amazon, citados a seguir:

- EC2 [20], núcleo computacional disponibilizado pela Amazon, utilizado para processar os dados automaticamente, alguns deles de forma recorrente. Ele possui acesso direto ao banco de dados, e é a interface que processa as requisições da aplicação mobile e retorna o necessário;
- RDS [21], serviço de banco de dados padrão da Amazon, foi utilizado para armazenar todas as informações referentes a todos os eventos na cidade;

Além disto, nessa parte foi utilizada a API do Facebook, como forma da instancia do EC2 acessar o banco de dados do Facebook e, portanto, extrair novos eventos que ocorram no Rio de Janeiro, ou informações atualizadas sobre esses, a serem armazenados no banco de dados criado na instância do RDS.

Capítulo 3

Análise

3.1 Nativos vs. Híbridos

Conforme introduzido no tópico 2.1, o funcionamento de uma aplicação nativa baseia-se no fato de ela ser desenvolvida na linguagem padrão do sistema operacional do dispositivo, tendo assim acesso direto a todos os seus recursos.

Já o funcionamento de uma aplicação puramente web segue o mesmo conceito básico de funcionamento de uma aplicação funcionando como uma página rodando dentro de um browser.

Por fim, o funcionamento de um aplicativo híbrido se assemelha ao do web, pois eles seguem o mesmo conceito básico de uma aplicação dentro de um navegador interno. A diferença é que esse navegador está encapsulado em um outro aplicativo nativo padrão que serve de carapaça, permitindo que a aplicação acesse também diretamente os recursos do dispositivo mobile, como fazem os nativos.

Dessa forma, por possuir ainda mais vantagens e nenhuma desvantagem em relação a aplicações Web, é possível entender os aplicativos híbridos como uma evolução deles. Por isso, para nos ater aos métodos mais relevantes, este tópico será focado na análise apenas dos métodos de desenvolvimento de aplicativos mobile conhecidos como híbridos e nativos.

Repetimos que, neste capítulo, não serão tratadas de aplicações puramente web, uma vez que elas seguem o mesmo conceito básico de funcionamento de uma aplicação que opera como uma página rodando dentro de um navegador interno, do mesmo jeito que funcionam os híbridos. A distinção é que nesses últimos, o browser está encapsulado em um outro aplicativo nativo padrão que serve de hospedeiro, permitindo que a aplicação acesse os recursos do dispositivo mobile.

Não obstante, conforme apresentado na figura 2.1, os sistemas operacionais que dominam o mercado de dispositivos mobile são, respectivamente, Android e iOS. Por conta disso, foi feita a escolha de inicialmente desenvolver o aplicativo protótipo apenas para essas duas plataformas. iOS e Android. Feita a análise, apesar do real interesse de futuramente desenvolver versões para outras plataformas a fim de conquistar mais mercado, esse tópico focará nesses dois sistemas operacionais.

A grande vantagem das aplicações nativas de um sistema operacional é a sua capacidade de comunicar-se com os dispositivos do aparelho, como câmera, acelerômetro, entre outros. Mencione-se, ainda, que o desempenho desse tipo de aplicação é mais elevado, pois o dispositivo mobile aloca 100% da memória para ela, diferentemente de uma híbrida, que é considerada pelo sistema operacional uma página dentro de um navegador interno, sendo, por isso, mais limitada em alocação de memória.

As aplicações nativas têm como vantagem ter uma comunidade maior que qualquer uma das híbridas, além de terem suporte de duas das maiores empresas de tecnologia do mundo, e por esse motivo possuem uma documentação maior também.

Quanto ao layout, as aplicações nativas são uma opção mais interessante que as híbridas, pois a Google e a Apple têm desenvolvido nos últimos anos padrões de layout muito robustos.

Os recursos de animação são muito mais eficientes nos nativos, visto que suas linguagens padrão são mais adaptadas a esses recursos. Já para as aplicações híbridas, as animações feitas em CSS costumam ser bem simples, enquanto as em JavaScript têm uma performance bastante limitada.

Por seu turno, as aplicações híbridas também apresentam vantagens. Por serem baseadas em padrões costumeiramente associados à web como JavaScript, HTML e CSS, o aprendizado é ainda mais facilitado para quem já é um web developer, profissional que conta com um mercado já há muito consolidado. Com o mesmo conhecimento é possível desenvolver sites ou aplicativos mobile, valorizando aqueles que o dominam pela flexibilidade.

Além disso, conforme mostrado anteriormente na figura 2.1, podemos ver que cerca de 8% do mercado utilizam dispositivos mobile com sistemas operacionais não tão comuns, que diferem dos dois mais utilizados (Android e iOS). Para os desenvolvedores, isso significa uma tarefa um tanto quanto árdua de criação de dispositivos nativamente para essas outras plataformas. Por sorte, algumas ferramentas

de desenvolvimento híbrido já possibilitam o empacotamento de aplicativos para que possam ser instalados nessas plataformas não tão populares, como, por exemplo, o PhoneGap, que, além de gerar aplicativos para Android e iOS, também empacota para os formatos compatíveis com sistemas operacionais como Windows Phone, Blackberry e Symbian.

Ademais, apesar de existir a possibilidade de que no futuro surjam ainda mais sistemas operacionais mobiles que utilizem outras linguagens de programação, tendo-se em consideração a forma como as comunidades responsáveis vêm desenvolvendo as ferramentas híbridas é de se esperar que essas últimas venham a suportar os novos sistemas operacionais a serem concebidos.

A conclusão é que, pelas características acima apresentadas, os métodos de desenvolvimento híbridos são recomendados geralmente para equipes de desenvolvedores pequenas, que não possuam muitos recursos para desenvolver e executar a manutenção do aplicativo. Igualmente, são mais apropriados também para equipes que tenham grande pressa em lançar suas aplicações prontas em diversas plataformas, ou ainda, com uma equipe acostumada a desenvolvimento web, pois pode-se aproveitar a expertise dos desenvolvedores.

Já os métodos nativos são mais recomendados para equipes com uma quantidade maior de desenvolvedores que possam desenvolver e dar manutenção para mais de um código ao mesmo tempo, ou para projetos que realmente necessitem de desempenho acima de qualquer outro fator.

3.2 Toolkit

Apesar de existirem várias plataformas de desenvolvimento híbrido, neste tópico vamos nos focar no PhoneGap, que foi utilizado para o desenvolvimento da aplicação mobile.

À medida que a proposta do aplicativo foi ficando clara, tornou-se necessário começar o desenvolvimento com o máximo de empenho à disposição, devido à inicial falta de recursos. Para tanto, levou-se em conta todas as restrições atuais.

Diferentemente de projetos desenvolvidos por empresas com vários funcionários e recursos financeiros mais elevados, um projeto desse tipo, feito por iniciativa de

alunos, costuma ter uma equipe bastante enxuta e pouco dinheiro em caixa para investir em ferramentas. Nesse caso, com uma equipe sendo de apenas um desenvolvedor e quase sem dinheiro, foi necessário otimizar cada recurso.

Primeiramente, ter que aprender a trabalhar em Java e Swift/Objective C para depois conseguir desenvolver versões do aplicativo para os sistemas operacionais Android e iOS, que representam a maior fatia do mercado (ver figura 2.1), parecia que iria consumir bastante tempo. Assim, conforme citado no tópico 2.1, foi muito sedutora a possibilidade de poder obter um aplicativo compatível com os dois sistemas operacionais alvo do projeto, por meio do desenvolvimento de apenas um único código numa única linguagem. Fazia muito sentido que, tendo apenas um desenvolvedor para cuidar de tudo, o uso de apenas um código seria mais eficiente.

Também, em decorrência disso, podemos analisar que a manutenção do código seria muito mais simples, assim como a construção de melhorias, por serem feitos com base num único código.

Conforme já falado no tópico 2.1, as plataformas híbridas, à símile das nativas, têm acesso a APIs dos dispositivos mobile, ou pelo menos a algumas delas.

Percebe-se, analisando a tabela 2.1, que o PhoneGap suporta todas as APIs possíveis presentes em todos os dispositivos que usam o sistema Android e naqueles com o sistema iOS, desde o modelo iPhone 3GS até o mais recente.

Essa característica por si só já mostra que o PhoneGap atende perfeitamente nesse quesito, pois, além de já se utilizar recursos como o “Storage”, “Notification” e “Network”, tem-se a garantia técnica de poder futuramente desenvolver novas funcionalidades que utilizem outros recursos caso desejado, como o de “Geolocation”, para se traçar uma rota para o evento, entre outros.

Outro ponto interessante, conforme mostrado na tabela 2.1, é que o PhoneGap permite o desenvolvimento de aplicativos para várias plataformas além de Android e iOS, que foram o alvo deste projeto, como, por exemplo, BlackBerry e Windows Phone. Isso garante a possibilidade futura de se desenvolver aplicações para outros sistemas operacionais mobile sem a necessidade de alterar a ferramenta de desenvolvimento, bem como sem precisar aprender novas linguagens.

3.2.1 Testando a Aplicação

O modo mais simples de se testar a aplicação é fazendo-o diretamente via protocolo HTTP. Para isso, pode-se usar um navegador comum como Google Chrome ou Mozilla Firefox, que suportam esse protocolo. É só acessar a página HTML do aplicativo pelo localhost. Por exemplo, se a página do aplicativo que se queira testar estiver nomeada como “pagina_inicial.html”, basta colocar no navegador <http://localhost/pagina_inicial.html>.

Esse método foi utilizado inicialmente para testar o aplicativo mobile durante seu desenvolvimento, porém teve de ser abandonado por possuir o seguinte defeito: mesmo que fosse utilizado num navegador do dispositivo mobile, não permitiria acesso de forma alguma aos recursos do aparelho. Dessa forma, não se poderia usar os recursos fundamentais como por exemplo o “storage”, necessária para guardar os dados dos eventos no “local storage”, que consiste em variáveis locais que ficam armazenadas na memória interna do dispositivo mobile do usuário.

Por conta dessa limitação se tornou necessário outra forma para testes, que foi satisfatoriamente conseguida ao utilizar para testar a aplicação o PhoneGap Developer App [22]. Criado pela Adobe para facilitar os testes de aplicativos de desenvolvedores diretamente no dispositivo desejado, permitindo assim analisar o desempenho ou, ainda, possibilitando examinar como o layout aparenta para aquele formato de tela. O PhoneGap Developer App, na verdade, emula o código do aplicativo como se fosse uma página web e, além disso, como ele já possui instaladas as APIs do dispositivo mobile, também acaba permitindo o acesso dessa aplicação emulada aos recursos do aparelho. Assim, ele “simula” diretamente o conceito do aplicativo Híbrido. Isso é interessante porque não foi necessário empacotar o código do projeto para testá-lo no dispositivo como se fosse mesmo um aplicativo.

Para usar o PhoneGap Developer App, o primeiro passo é instalá-lo em um dispositivo com sistema operacional Android ou iOS. Depois, deve-se abrir o prompt de comando do Windows e ir até a pasta em que o projeto PhoneGap foi criado. A seguir, conforme ilustrado na figura 3.1, é usado o comando “phonegap serve”, para iniciar um pequeno servidor web para esse projeto. Após, ele irá retornar um endereço IP (Internet Protocol) com a porta correta, para acessarmos esse projeto pelo dispositivo mobile.

```
CA: Prompt de Comando - phonegap serve
Microsoft Windows [versão 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Todos os direitos reservados.

C:\Users\Aldrick>cd \
C:\>cd TiuparApp
C:\TiuparApp>phonegap serve
[phonegap] starting app server...
[phonegap] listening on 169.254.177.78:3000
[phonegap] listening on 192.168.1.4:3000
[phonegap] listening on 192.168.21.1:3000
[phonegap] listening on 192.168.38.1:3000
[phonegap]
[phonegap] ctrl-c to stop the server
[phonegap]
```

Figura 3.1 - Obtendo IP para testar o projeto

Em seguida, e com a certeza de que o computador e o dispositivo móvel estejam na mesma rede local (pode-se ligar o computador ao dispositivo móvel pelo cabo), conforme mostra a figura 3.2, é inserido esse IP obtido no aplicativo no PhoneGap Developer App, que emulará o código dessa aplicação como se fosse um aplicativo do dispositivo mobile, com acesso a todas as funcionalidades requeridas.

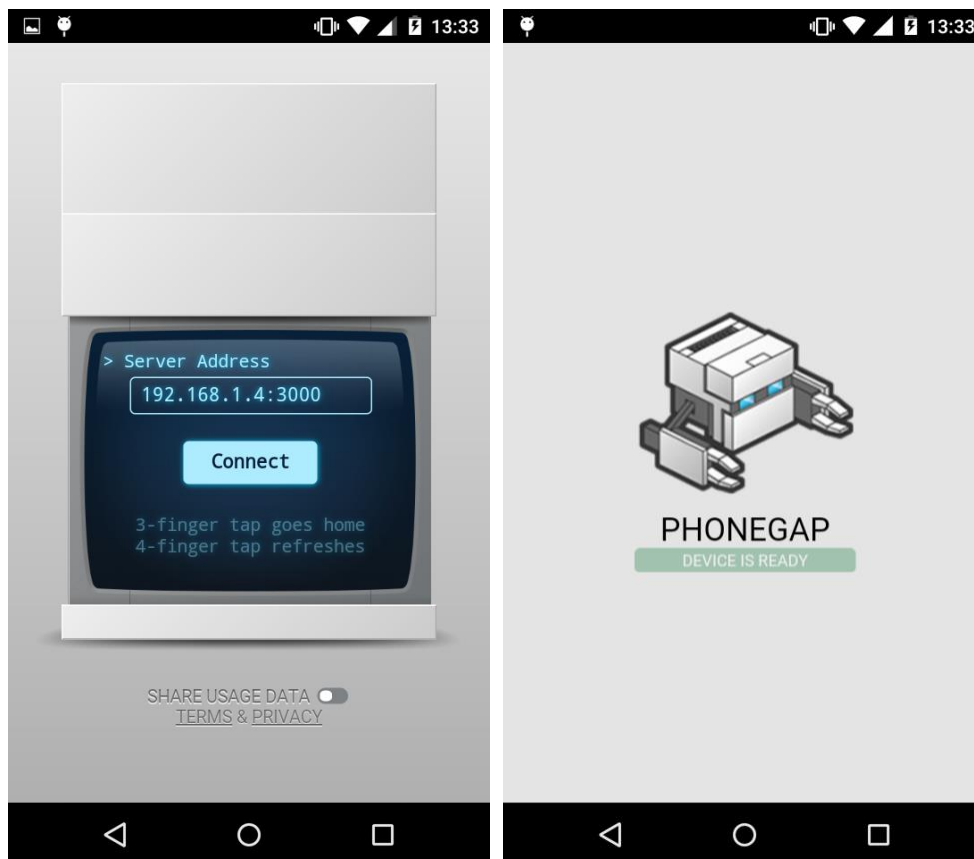


Figura 3.2 – PhoneGap Developer App

3.2.2 Compilando o Aplicativo

Para compilar um aplicativo mobile, a forma tradicional é fazer isso num computador. Para tanto, é preciso instalar os SDKs para as plataformas desejadas, ou seja, o iOS SDK [23] e o Android SDK. O problema é que Apple impõe uma limitação em que apenas é possível instalar o iOS SDK em um computador que rode sistema MacOS. Isso limita muito os desenvolvedores, pois os computadores da Apple costumam ter um preço mais elevado, sem mencionar que a única forma oficial para conseguirmos compilar um aplicativo para rodar em um dispositivo iOS é utilizando um computador da marca. O Google não impõe nenhuma limitação ao Android SDK, que pode ser instalado em ambiente Windows, Linux ou Mac.

Para quem não tem à disposição um computador com sistema MacOS nem possui recursos financeiros para comprar um, como era o caso desse projeto, o PhoneGap passa a ser uma alternativa muito interessante por possuir um serviço chamado PhoneGap Build [24]. Lançado em 2012 pela própria Adobe, esse último consiste numa plataforma web que permite enviar o código do projeto desenvolvido em PhoneGap, que é então armazenado em nuvem, e com ele é possível gerar os pacotes de instalação dos aplicativos para diferentes sistemas operacionais, incluindo iOS e Android. Com esse serviço na nuvem não é necessário instalar os SDKs, pois ele já os possui, eliminando, portanto, a obrigatoriedade de adquirir um Mac.

O referido serviço é gratuito para empacotar até um projeto, seguindo as limitações mostradas na figura 3.3. Vale destacar que esse modelo gratuito atende a todas as exigências desse projeto, pois apenas foram utilizados plug-ins nativos do PhoneGap.

	Free Plan	Paid Plan	Adobe Creative Cloud Membership
open source apps	∞ unlimited <small>must be pulled from a public Github repo</small>		
private apps	1	25	
max app size	50 MB	100 MB	1 GB
core cordova plugins <small>* a list of these plugins is here</small>	YES		
third party plugins <small>* includes plugins from npmjs.com as well as our own repository</small>	YES		
upload plugins <small>* includes plugins only you can use</small>	NO	YES	
collaborators	∞ unlimited <small>invite people to your app as either developers or testers</small>		
	completely free	starting at \$9.99/mo	sign in with your Adobe ID

Figura 3.3 – Restrições do plano gratuito do PhoneGap Build

Fonte: PhoneGap Build [24]

Contudo, mesmo com esse serviço, não é possível se esquivar de todas as limitações impostas pela Apple. É importante ressaltar que, caso o desenvolvedor queira gerar o pacote de instalação do aplicativo para um dispositivo físico com o sistema operacional iOS, é necessário adquirir uma conta de iPhone Developer, ao custo anual de US\$99.00. Com ela, é possível construir um arquivo que serve como “chave” para a aplicação (no caso do PhoneGap Build, essa “chave” deve ser enviada junto com o projeto).

Qualquer futuro desenvolvedor mobile deve estar atento a essa limitação imposta pela Apple. Essa restrição prejudica muitos desenvolvedores, pois, mesmo com recursos financeiros limitados, foi necessário pagar uma conta de iPhone Developer, cujo valor, pela alta do dólar à época, chegou perto dos R\$400,00. Sem isso, não teria sido possível instalá-lo diretamente num dispositivo com iOS.

Já o Google felizmente não impõe nenhuma restrição desse tipo para empacotar o aplicativo no formato compatível com Android, e para obter uma conta que permita publicação de aplicativos mobile em sua loja virtual custou US\$25,00 exigindo pagamento apenas uma vez, sem necessidade de renovação.

Vale ressaltar que, para publicar a aplicação na App Store, é necessário instalar um programa chamado Application Loader, que só é compatível com o sistema operacional MacOS. Ou seja, sem um computador Apple não é possível colocar o aplicativo mobile na loja virtual da Apple. Esse fato atrasou esse projeto em algumas semanas, só tendo sido resolvido com um computador Mac emprestado.

Por conclusão, todas as restrições impostas pela Apple citadas nesse tópico mostram a postura controladora da empresa. Esse comportamento, que difere de seus concorrentes diretos, limita a atuação dos desenvolvedores interessados em desenvolver aplicações para iOS. Apesar dos esforços utilizados para contornar essa necessidade, no fim se tornou impossível desenvolver e publicar em sua loja virtual um aplicativo mobile sem possuir um computador Mac em mãos.

Além disso, os gastos para aquisição de licença para desenvolvimento foram mais altos que a da plataforma concorrente, tendo de ser renovados anualmente, enquanto para a licença da loja virtual do Google não era necessário renovação. Esse custo recorrente prejudica e em alguns casos até inviabiliza desenvolvedores com baixo orçamento disponível.

3.2.3 Garantindo bom desempenho com o PhoneGap

Como visto anteriormente no tópico 2.1.3, o que caracteriza aplicações híbridas é que seu funcionamento se baseia em emular o código que foi desenvolvido para ser a aplicação como se fosse uma página web rodando em um navegador interno instanciado dentro de um aplicativo nativo.

Por padrão, ao se usar um método de desenvolvimento híbrido, o dispositivo mobile não compreende que o projeto é uma aplicação rodando; em vez disso, ele trata o aplicativo como uma simples página web dentro de um browser. Dessa forma, o dispositivo não instancia 100% de memória para a aplicação, o que pode gerar problemas de performance. Isso ocorreu com a primeira versão deste projeto, tornando praticamente inviável sua utilização por qualquer usuário. Com esse problema em vista, foi necessário pesquisar maneiras de otimizar o uso de aplicações desenvolvidas com PhoneGap, e as três melhorias de desempenho [25] utilizadas foram:

- Evitar acesso constante ao DOM, pois isso consome muito recurso da aplicação. O ideal é passar todas as variáveis de uma só vez para o DOM, deixando invisíveis os elementos que não se queira que apareçam;
- Sempre evitar utilizar recursos de animação diretamente com JavaScript, como, por exemplo, animações de mudar a posição de botões, pois, caso a aplicação esteja sobrecarregada, o desempenho da animação pode ser afetado. Já animações baseadas em CSS são geralmente tratadas em paralelo ao resto da aplicação em que layout e JavaScript são executados. Desse modo, mesmo se a aplicação estiver sobrecarregada, as animações baseadas em CSS possivelmente continuarão sem interrupções [26];
- Evitar acesso recorrente à Internet. O ideal é coletar de uma vez todos os dados necessários do servidor e colocá-los em variáveis locais. Conforme discutido no tópico 3.2.1, no caso desse projeto, logo na primeira “página” da aplicação é feito o acesso ao banco de dados na nuvem e coletados de uma só vez todos os dados dos eventos, que são guardados no “local storage”. Nenhum outro acesso ao banco é feito em outras partes da aplicação, o que melhora o desempenho do aplicativo mobile ao não precisar acessar ao servidor a cada nova mudança de tela. Além disso, essa funcionalidade foi fundamental para permitir que, mesmo com o usuário perca a conexão com a Internet durante o uso por alguma indisponibilidade, ainda consiga continuar a operar o aplicativo a partir dos dados carregados no último acesso.

3.3 Servidor

Para o desenvolvimento de um aplicativo mobile, levando em conta será instalado em dispositivos físicos com recursos de hardware limitados pela potência da bateria e focados no acesso constante à internet, o uso de um servidor é fundamental. Ele serve para armazenar os dados que servem de conteúdo a ser exibido pela aplicação num banco e, aplicar filtros e tratamentos a fim de torná-los relevantes para os usuários.

Analisando o que foi visto no tópico 2.3 e comparando com as necessidades desse projeto, um serviço de computação em nuvem pública se mostrou o mais indicado. Como não havia recursos para um investimento inicial em hardware, a

oportunidade de pagar apenas por hora em que o servidor estivesse online se mostrou bem sedutora. Além disso, como o modelo de negocio do projeto ainda não tinha se mostrado rentável, parecia ineficiente um gasto maior logo no inicio para compra de equipamentos nessa fase de validação.

Também era importante um serviço em que não fosse necessário fazer configurações no hardware, para otimizar o trabalho do único desenvolvedor que deveria se focar no desenvolvimento do aplicativo mobile, e que não possuía experiência prática em hardware.

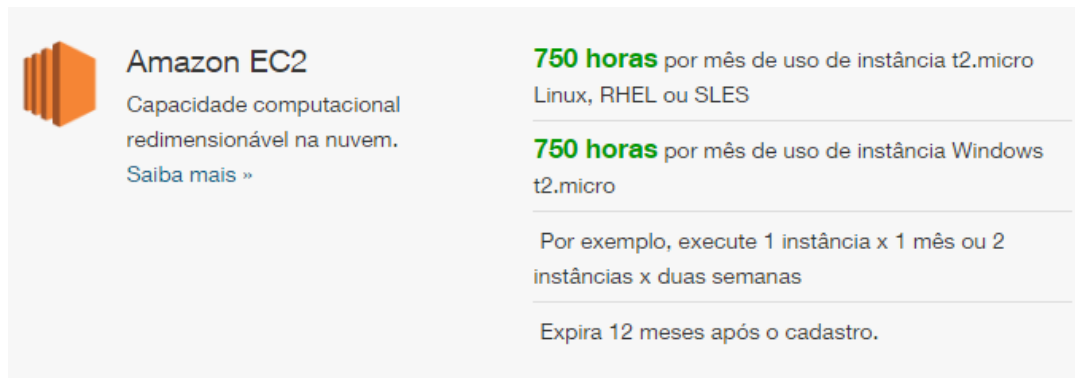
Analisando os serviços disponíveis no mercado, foi escolhida a Amazon Web Services. Ela possui um serviço de banco de dados e um serviço de processamento que era o que o projeto demandava. Seus diferenciais foram o fato dela ser a líder de mercado no setor e pela facilidade em encontrar tutoriais e cursos disponibilizados tanto por ela quanto por outros usuários de seus serviços, o que foi útil pois reduziu o tempo de curva de aprendizado e auxiliou na resolução de problemas encontrados na construção do servidor para o aplicativo mobile.

Além disso, a facilidade de integração entre seus serviços fornecida chamou atenção, pois é possível com poucos cliques dentro de sua plataforma de gerenciamento criar um grupo de segurança que limite o acesso a instancia de banco de dados RDS, tanto para leitura quanto para escrita, unicamente a instancia de processamento EC2 usada nesse servidor, o que garantiu uma camada de proteção contra ataques mal intencionados visando roubar ou corromper os dados que alimentarão o aplicativo mobile.

Outro fator que foi decisivo para a escolha da Amazon é que ela oferece um período gratuito de uso de determinadas ferramentas para novos usuários disponível por até 12 meses após a data de [cadastro](#), denominado nível gratuito [27]. Como haviam poucos recursos financeiros, manter o servidor gratuito reduziu o risco de investimento financeiro durante o período de validação da proposta do aplicativo mobile, sendo fundamental na execução desse projeto.

3.3.1 EC2

A figura 3.4 descreve as limitações de uso para uma instância EC2 dentro do nível gratuito.



The image is a screenshot of the Amazon EC2 free tier page. It features the Amazon EC2 logo and the text 'Capacidade computacional redimensionável na nuvem. Saiba mais »'. Below this, it states '750 horas por mês de uso de instância t2.micro Linux, RHEL ou SLES'. A second line states '750 horas por mês de uso de instância Windows t2.micro'. Below these, it provides an example: 'Por exemplo, execute 1 instância x 1 mês ou 2 instâncias x duas semanas'. At the bottom, it says 'Expira 12 meses após o cadastro.'

Figura 3.4 – Restrições do plano gratuito de uma instância AWS EC2

Fonte: Nível Gratuito da AWS [27]

Com uma conta rápida, percebe-se que um mês tem no máximo 24 horas vezes 31 dias, ou seja, 744 horas. A limitação de 750 horas é o suficiente para manter uma única instância do modelo t2.micro ativa por todos os meses durante um ano.

As instâncias do tipo t2 do EC2 são instâncias com capacidade de intermitência, ou seja, foram projetadas para aplicações que não precisam constantemente de níveis altos de CPU, mas se beneficiam significativamente da possibilidade de alto desempenho quando necessário. O conceito de intermitência se da por elas funcionarem com desempenho limitado do núcleo de sua CPU, e quando necessário utilizar toda a capacidade computacional desse núcleo é utilizado uma unidade conhecida como créditos de CPU, que representa um minuto utilizando 100% da potencia do núcleo da CPU. Esses créditos são distribuídos por hora, e caso não sejam utilizados acumulam por até 24 horas.

Essa característica faz com que eles tenham sido planejados para servidores como o utilizado para nossa aplicação mobile, pois o acesso não é constante durante o dia todo. Existem picos em algumas horas ou até minutos, enquanto em outros horários a taxa de acesso pode ser muito baixa para manter uma CPU em sua capacidade máxima. Esse gerenciamento de capacidade computacional faz com que o custo das instâncias t2 seja menor do que as de desempenho fixo, e no caso da t2.micro seja coberta pelo nível gratuito.

A tabela 3.1 descreve as especificações técnicas de instâncias t2. Com um processador Intel Xeon de 3.3 GHz e com 1 Gb de memória RAM, a instância t2.micro tem uma configuração semelhante ao computador usado para o desenvolvimento do aplicativo mobile, que foi cogitado como alternativa para servidor caso não conseguisse um serviço satisfatório na Amazon. Com uma taxa de 6 créditos de CPU por hora, e levando em conta que eles acumulam por até 24 horas, isso dá um total de até 2 horas e 24 minutos por dia usando 100% da capacidade da CPU, o que bastou pois conforme previsto os acessos se deram em sua maioria durante alguns períodos da noite.

Tabela 3.1 – Especificações Técnicas de Instâncias T2

Fonte: Tipos de instância do Amazon EC2 [28]

Tipo de instância	vCPU	Memória (GiB)	Armazenamento (GB)	Créditos de CPU/hora	Processador físico	Velocidade de clock (GHz)
t2.nano	1	0,5	Somente EBS	3	Família Intel Xeon	até 3.3
t2.micro	1	1	Somente EBS	6	Família Intel Xeon	Até 3.3
t2.small	1	2	Somente EBS	12	Família Intel Xeon	Até 3.3
t2.medium	2	4	Somente EBS	24	Família Intel Xeon	Até 3.3
t2.large	2	8	Somente EBS	36	Família Intel Xeon	Até 3.0

Outra vantagem é que caso o modelo t2.micro não atendesse mais as demandas do servidor, com poucos cliques é possível convertê-lo para outro modelo de instância mais robusto com mais capacidade de processamento ou com mais créditos de CPU por hora, apesar destas não estarem inclusas no nível gratuito.


É necessário definir um sistema operacional para rodar na instância. No caso do servidor usado nesse projeto, foi escolhido o sistema Linux Ubuntu, pela expertise que o autor desse trabalho possuía neste sistema operacional. Pelo mesmo motivo de conhecimento prévio foi escolhida a linguagem de programação PHP para o desenvolvimento dos algoritmos que realizam o processamento dos dados na instância EC2.

Uma instância EC2 é necessária para fins de processamento e é a única com capacidade de ler e de escrever dados diretamente no banco de dados. Ela foi utilizada de duas maneiras. Na primeira, ela recebe as requisições da aplicação mobile e retorna o conteúdo solicitado, o que significa que, no caso do protótipo deste trabalho, ela envia as informações apenas sobre os eventos do dia corrente para o aplicativo mobile a cada solicitação recebida.

Na segunda maneira empregada, a interface opera de forma automatizada, isto é, não precisa receber requisições do aplicativo mobile. Ela foi programada para, com periodicidade diária, acessar dados da API do Facebook em busca de novos eventos a serem incorporados no banco e, ainda, para atualizar automaticamente os eventos já inseridos quanto a suas informações básicas, como, por exemplo, a data e hora do evento caso seja alterada pelos organizadores do evento.

3.3.2 RDS

Conforme visto no tópico 2.4, uma instância RDS pode ser usada como banco de dados do servidor. A figura 3.4 descreve as limitações de uso para uma instância RDS dentro do nível gratuito.



The image shows the Amazon RDS Free Tier limits. On the left, there is a blue cube icon and the text 'Amazon RDS Serviço de banco de dados relacional gerenciado para MySQL, PostgreSQL, MariaDB, Oracle BYOL ou SQL Server. Saiba mais »'. On the right, there are four lines of text, each preceded by a green icon (a cube, a storage icon, a backup icon, and a document icon). The text lists the following limits: 750 horas de uso de instância db.t2.micro Single-AZ do Amazon RDS; 20 GB de armazenamento de banco de dados: qualquer combinação de propósito geral (SSD) ou magnético; 20 GB para backups (com armazenamento magnético do RDS. E/S em propósito geral [SSD] não são cobradas separadamente); and 10.000.000 de E/S. At the bottom, it says 'Expira 12 meses após o cadastro.'

750 horas de uso de instância db.t2.micro Single-AZ do Amazon RDS
20 GB de armazenamento de banco de dados: qualquer combinação de propósito geral (SSD) ou magnético
20 GB para backups (com armazenamento magnético do RDS. E/S em propósito geral [SSD] não são cobradas separadamente)
10.000.000 de E/S
Expira 12 meses após o cadastro.

Figura 3.5 – Restrições do plano gratuito de uma instância AWS RDS

Fonte: Nível Gratuito da AWS [27]

O modelo de instância que se encaixa no nível gratuito é a db.t2.micro. Além de fornecer horas suficientes para uma única instância desse formato conforme visto na conta realizada no subtópico 3.3.1, ele fornece um total de até 10 milhões de requisições de entrada e saída por ano, ou seja, de leitura e escrita no banco de dados.

Isso se mostrou suficiente para o período inicial pois conforme visto no subtópico 3.2.3 graças ao fato de evitarmos acessar o servidor toda hora no aplicativo mobile, fazendo isso apenas quando o usuário abre o aplicativo e guardando todos os dados dos eventos em variáveis locais não precisando acessar o servidor a cada mudança de tela, além de otimizarmos a instância de processamento EC2 para maior eficiência em relação aos acessos ao banco, pode-se garantir que a cada seção de um usuário só representa uma entrada/saída do banco de dados. Com isso, ficam garantidas 10 milhões de seções de usuários no aplicativo mobile totalmente gratuitas, só sendo cobradas ao se exceder esse número. Pela estimativa inicial, seria o suficiente para o escopo da aplicação.

Quanto ao armazenamento, pelo fato de serem fornecidos 20 Gb por ano foram realizadas otimizações para se guardar no banco de dados apenas dados do tipo texto, que ocupam um tamanho consideravelmente menor do que imagens. Para tanto, aproveitou-se do fato do Facebook armazenar as imagens dos eventos num servidor de imagens com acesso aberto. Então, guardou-se apenas o link em texto no banco de dados, que era passado para o aplicativo mobile que abria a imagem diretamente do serviço de imagens do Facebook, sem gerar demanda ao nosso servidor.

Apesar de ser possível com poucos cliques converter o banco de dados para outro modelo de instância mais robusto, as otimizações citadas contribuíram para manter o servidor dentro do nível gratuito, evitando gastos já que o autor desse projeto possuía recursos financeiros limitados.

O RDS fornece aos desenvolvedores vários possíveis mecanismos relacionais de banco de dados [29], e nesse caso foi feita a escolha do MySQL para rodar na instância, pelo conhecimento prévio adquirido em trabalhos anteriores.

3.4 Utilizando a API do Facebook

Conforme descrito no subtópico 3.3.1, a API do Facebook é usada para que nossa instância de processamento acesse os dados do Facebook; extraia informações de todos os eventos com teor de entretenimento como festas, baladas e eventos culturais em determinados locais dentro da região desejada, no caso a cidade do Rio de Janeiro; e insira-as no banco de dados RDS do meu servidor.

O padrão atual é usar a Graph API para acessar os dados. Para realizar as requisições é necessário um token de acesso, servindo como um código de identificação exigido a cada query realizada que pode ser obtido na plataforma em que o Facebook disponibiliza todos os recursos e informações sobre suas ferramentas para desenvolvedores, na ferramenta chamada Access Token Tool [30].

A figura 3.6 mostra um exemplo de utilização desse método, com um modelo de query usado para obter informações de eventos que ocorrerão num determinado local aplicada na ferramenta de simulação de acesso ao banco do Facebook para desenvolvedores conhecida como Graph API Explorer [31]. O código numérico 143286195735246 que consta na requisição representa um identificador único dado pelo Facebook para a casa de festa chamada Teatro Odisséia, conhecida por receber festas e shows na Lapa. Assim, essa requisição retorna uma lista com todos os eventos que ocorrerão nesse local no formato JSON.

The screenshot shows the Graph API Explorer interface. At the top, it says 'Graph API Explorer' and 'Aplicativo: [?] Graph API Explorer'. Below that, there's a 'Token de acesso:' field with a blue icon and a long alphanumeric string: 'EAACEdEose0cBAKR1cZAD2zKKMWeKZCLPmhpQXKQ7pr7C6gbGMxXIIISgYfZAQGQZCqpZAIZCFiuxjvULDWPSptzHm0jvBN1ONZC'. To the right of the token is a 'Get Token' button. Below the token, there's a 'Graph API' tab and an 'FQL Query' tab. The 'FQL Query' tab is active, showing a query: 'GET → /v2.3 /143286195735246/events/?fields=name,attending_count,start_time,end_time,description'. To the right of the query is a 'Submit' button. Below the query, there's a list of fields to include: 'name', 'attending_count', 'start_time', 'end_time', and 'description', each with a checked checkbox. Below the list is a '+ Search for a field' link. To the right of the fields, there's a 'Saiba mais sobre a sintaxe da Graph API' link. The main area shows the JSON response for the query. The response is a JSON object with a 'data' array containing one event object. The event object has fields: 'name', 'attending_count', 'start_time', 'end_time', and 'description'. The event details are: 'Esteban no Rio de Janeiro', 'attending_count': 1598, 'start_time': '2016-08-06T16:00:00-0300', 'end_time': '2016-08-06T21:00:00-0300', 'description': 'Cena Rock Produtora apresenta: Esteban Tavares Com a Tour \"Saca La Muerte de Tu Vida\" (Show Acústico) + Gaiteiro. Show de abertura: RADIOATIVA e NoTime. Ingressos: http://www.ticketplanet.com.br/evento/242-esteban. Preços: Pista Meia entrada ou Promocional com 1 kg de alimento - 45,00 Reais. PISTA + Meet & Greet - Meia entrada ou Promocional com 1 kg de alimento - 100,00 Reais'.

Figura 3.6 – Exemplo de requisição da listagem de eventos que ocorrerão numa casa de festas aplicada na plataforma Graph API Explorer

Nessa lista constam informações relevantes para os usuários como, por exemplo, nome, data e hora de início e término, quantidade de pessoas que confirmaram presença, além da descrição de cada evento.

O fato das informações serem retornadas no formato JSON é bastante interessante pela facilidade em convertê-lo para outros formatos mais simples de serem trabalhados mantendo os atributos bem definidos, sendo essa uma de suas principais características [32]. No caso desse projeto, o algoritmo de processamento foi desenvolvido em PHP que possui a função “`json_decode`” [33] que converte um JSON para um array com uma estrutura que permite acesso diretamente, sabendo apenas o nome do atributo e a posição do evento no array, a qualquer informação de qualquer um dos eventos.

Conforme visto no tópico 2.2, o Facebook lança periodicamente novas versões de sua API, retirando-as de uso no máximo dois anos após o início de operação da versão seguinte. No início do desenvolvimento desse aplicativo mobile foi utilizado um método alternativo oferecido na época para obter dados do banco do Facebook conhecido como FQL [34]. Ele foi escolhido pela sua sintaxe que se assemelhava muito com o MySQL, linguagem já dominada pelo autor desse trabalho. O problema é que ele deixou de ser suportado a partir da versão 2.1 da API [35]. Foi necessário alterar o método usado do FQL para o Graph API antes da versão 2.0 ser retirada de uso, o que ocorreu no dia 7 de agosto de 2016, exatamente dois anos após o lançamento da versão 2.1 da API.

A conclusão é que é importante o desenvolvedor ficar atento às alterações de cada nova da API, pois os recursos utilizados podem se tornar indisponíveis ou com alterações no seu uso. O fato de novos recursos surgirem também pode ser visto como uma oportunidade para novas aplicações. Apesar do trabalho extra na mudança de método, o período de dois anos para retirada de uma versão da API foi o suficiente atualizar para uma mais recente.

Capítulo 4

Conclusão

4.1 Conclusão

Ao se analisar o conhecimento passado nesse trabalho, percebe-se que todos os objetivos foram cumpridos.

As características dos métodos de desenvolvimento mobile foram caracterizadas, possuindo vantagens e desvantagens dependendo da necessidade da aplicação e dos recursos disponíveis, como equipe ou dinheiro. Percebe-se que as híbridas se sobressaem para equipes enxutas com apenas um desenvolvedor como a do Tiupar, dada sua facilidade de desenvolvimento e manutenção para múltiplos sistemas operacionais mobile.

Analisando as ferramentas que o PhoneGap possui, entende-se porque ele é uma excelente escolha para quem não possui um computador com o MacOS para compilar a aplicação, e a sua ferramenta de testes dispensa também a necessidade de um iPhone para testes. Apesar disso, um computador da Apple se faz necessário para publicar o aplicativo mobile na App Store, necessitando também de uma conta de desenvolvedor Apple, tendo um custo considerado alto para o autor desse projeto sendo necessário renovar anualmente. Isso mostra as limitações que a Apple impõe a seus desenvolvedores, que por outro lado não são impostas pela Google, que apesar de ter um custo para aquisição de uma conta de desenvolvedor que permita publicar aplicativos na Google Play, pagos apenas uma vez.

Pelos serviços de provisionamento em nuvem, percebe-se que por não possuir recursos financeiros e nem uma necessidade excessiva de medidas de segurança extras, um serviço em nuvem pública se adequou bem a situação. Nessa categoria, a Amazon se destaca por ser a líder, e seus serviços de processamento de dados EC2 e o de banco de dados RDS eram o que o servidor da aplicação demandava.

Outra vantagem da Amazon é o seu nível gratuito, o que zerou os custos de servidor pois suas restrições de uso se encaixavam na demanda do servidor do Tiupar.

Nesse tópico de servidor, foi descrita que o PHP foi a linguagem de programação utilizada para escrever o algoritmo de processamento, e o MySQL foi o banco de dados.

Quanto a API do Facebook, percebe-se que pelas constantes alterações que o Facebook tem realizado em suas atualizações é importante que o desenvolvedor fique atento às mudanças de sintaxe, aos recursos novos e aos recursos que são removidos, podendo ser tanto uma oportunidade para algum novo aplicativo quanto uma barreira para um aplicativo que necessite de alguma funcionalidade retirada.

Por fim, foi realizada uma análise de como são extraídos e convertidos em informação a ser inserida no banco de dados os eventos extraídos pela API do Facebook.

4.2 Trabalhos Futuros

Como foco de futuros trabalhos, pode-se enumerar o método de desenvolvimento híbrido AngularJS, que é baseado na estrutura do Cordova, porém já com otimizações de desempenho pré-estabelecidas. Também deve-se citar o seu framework IONIC, que possui recursos gráficos interessantes.

Na parte de servidor, é interessante pensar em estratégias para o crescimento de usuários, usando mais de uma instância do EC2 em paralelo com o uso do Elastic Load Balancing, serviço da Amazon que distribui o tráfego de acesso igualmente entre as instâncias EC2, para que nenhuma se sobrecarregue enquanto outra fique com pouca utilização.

Para o banco de dados, é interessante estudar o Aurora, tipo de banco de dados relacional da Amazon com sintaxe idêntica ao MySQL, porém com desempenho até cinco vezes superior.

Bibliografia

- [1] Mobile/Tablet Operating System Market Share, NETMARKETSHARE, < <https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1/>>, acessado em 03 de Fevereiro de 2016.
- [2] Android Studio, < <http://developer.android.com/intl/pt-br/tools/studio/index.html> >, acessado em 03 de Fevereiro de 2016.
- [3] XCode, < <https://developer.apple.com/xcode/>>, acessado em 03 de Fevereiro de 2016.
- [4] Android Software Development Kit, SDK, < <http://developer.android.com/intl/pt-br/sdk/installing/index.html?pkg=tools> >, acessado em 03 de Fevereiro de 2016.
- [5] Eclipse, < <https://eclipse.org/> >, acessado em 03 de Fevereiro de 2016.
- [6] Android Developer Tools, ADT, < <http://developer.android.com/intl/pt-br/tools/help/adt.html> >, acessado em 03 de Fevereiro de 2016.
- [7] PhoneGap, < <http://phonegap.com/> >, acessado em 04 de Fevereiro de 2016.
- [8] PhoneGap Explained Visually < <http://phonegap.com/2012/05/02/phonegap-explained-visually/>>, acessado em 07 de Fevereiro de 2016.
- [9] PhoneGap Supported Features, < <http://phonegap.com/about/feature/>>, acessado em 06 de Fevereiro de 2016.
- [10] PhoneGap About, <<http://phonegap.com/about/>>, acessado em 15 de Fevereiro de 2016.
- [11] Graph API, < <https://developers.facebook.com/docs/graph-api> >, acessado em 04 de Fevereiro de 2016.
- [12] Facebook Platform Versioning, <<https://developers.facebook.com/docs/apps/versions/>>, acessado em 07 de Julho de 2016.
- [13] Conheça as Diferenças entre Nuvem Híbrida, Pública e Privada, <<https://configr.com/blog/conheca-as-diferencas-entre-nuvem-hibrida-publica-e-privada/> >, acessado em 15 de Fevereiro de 2016.
- [14] Amazon Web Services, < <https://aws.amazon.com/pt/> >, acessado em 04 de Fevereiro de 2016.

- [15] AWS foi designada como um líder no Quadrante Mágico de IaaS pelo 4º ano consecutivo, <<https://aws.amazon.com/pt/resources/gartner-2015-mq-learn-more/>>, acessado em 04 de Fevereiro de 2016.
- [16] Crescimento do setor de eventos cria oportunidades pelo Brasil, FOLHA DE SÃO PAULO, <<http://www1.folha.uol.com.br/mercado/2014/09/1519733-crescimento-do-setor-de-eventos-cria-oportunidades-pelo-brasil.shtml>>, acessado em 04 de Fevereiro de 2016.
- [17] Participação da Internet no mercado de publicidade mundial aumenta, GO2WEB, <<http://www.go2web.com.br/pt-BR/blog/participacao-da-internet-no-mercado-de-publicidade-mundial-aumenta.html>>, acessado em 04 de Fevereiro de 2016.
- [18] B2C Content Marketing 2014, Content Marketing Institute, <http://contentmarketinginstitute.com/wp-content/uploads/2013/10/B2C_Research_2014-withlinks.pdf>, acessado em 04 de Fevereiro de 2016.
- [19] Facebook Says Its New Focuses Are Personalized Content And Spreading The Internet, Business Insider, <<http://www.businessinsider.com/in-2014-facebook-says-it-will-focus-on-personalized-content-and-spreading-the-internet-2014-1>>, acessado em 05 de Fevereiro de 2016.
- [20] Amazon Elastic Compute Cloud, EC2, <<https://aws.amazon.com/pt/ec2/>>, acessado em 05 de Fevereiro de 2016.
- [21] Amazon Relational Database Service, RDS, <<https://aws.amazon.com/pt/rds/>>, acessado em 05 de Fevereiro de 2016.
- [22] PhoneGap Developer App, <<http://app.phonegap.com/>>, acessado em 06 de Fevereiro de 2016.
- [23] iOS SDK, <<https://developer.apple.com/ios/>>, acessado em 06 de Fevereiro de 2016.
- [24] PhoneGap Build, <<https://build.phonegap.com/>>, acessado em 07 de Fevereiro de 2016.
- [25] Vale a pena usar Cordova/PhoneGap, INFOQ, <<http://www.infoq.com/br/presentations/vale-a-pena-usar-cordova-phonegap/>>, acessado em 07 de Fevereiro de 2016.
- [26] Animations and Performance, Google Developers, <<https://developers.google.com/web/fundamentals/design-and-ui/animations/animations-and-performance/>>, acessado em 08 de Fevereiro de 2016.
- [27] Nível Gratuito da AWS, <<https://aws.amazon.com/pt/free/>>, acessado em 17 de Julho de 2016.

- [28] Tipos de instância do Amazon EC2, <<https://aws.amazon.com/pt/ec2/instance-types/>>, acessado em 19 de Julho de 2016.
- [29] Detalhes do produto Amazon RDS, <<https://aws.amazon.com/pt/rds/details/>>, acessado em 26 de Julho de 2016.
- [30] Access Token Tool, <<https://developers.facebook.com/tools/accesstoken/>> acessado em 26 de Julho de 2016.
- [31] Graph API Explorer, <<https://developers.facebook.com/tools/explorer/>>, acessado em 26 de Julho de 2016.
- [32] Introducing JSON, <<http://www.json.org/>>, acessado em 27 de Julho de 2016.
- [33] PHP: json_decode - Manual, <http://php.net/manual/pt_BR/function.json-decode.php>, acessado em 27 de Julho de 2016.
- [34] Facebook Query Language Overview, <<https://developers.facebook.com/docs/technical-guides/fql/>>, acessado em 27 de Julho de 2016.
- [35] Log de alterações da Plataforma do Facebook, <<https://developers.facebook.com/docs/apps/changelog>>, acessado em 27 de Julho de 2016.

Apêndice A

Pesquisa sobre Eventos

*Item Obrigatório

Você costuma ir a eventos (festas, baladas)? *

- sim, sempre
- sim, às vezes
- não

Caso não costume ir a festas, por que não gosta/costuma ir?

Como você costuma procurar saber sobre novos eventos? *

- Amigos
- Redes Sociais
- Internet (google)

O que mais te influi na hora de escolher um evento? *

- Preço
- Localidade
- Quantidade de Pessoas do Sexo Oposto
- Quantidade de Amigos que Vão
- Outro:

Costuma ter dificuldade de achar eventos nos arredores ou em lugares que te interessam? *

- Sim
- Não

Que tipo de promoção você costuma aderir com frequência? *

- Lista Amiga
- Combos mais Baratos
- Shots de Bebida Grátis
- Outro: