UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

DIEGO MORSOLIN MARCON

Um Sistema Android para Gerenciamento de Roteiros Turísticos

Trabalho de Graduação.

Prof. Dr. Leandro Krug Wives
Orientador

Porto Alegre, janeiro de 2013.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
CSS	Cascading Style Sheets
GPS	Global Positioning System
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IDC	International Data Corporation
JSON	JavaScript Object Notation
LBS	Location-Based Service
MVC	Model-View-Controller
REST	Representational State Transfer
SDK	Software Development Kit
URL	Uniform Resource Locator
XML	eXtensible Markup Language

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Com o crescimento do mercado de dispositivos móveis e uma previsão de aumento significativo do turismo no Brasil, este trabalho apresenta a concepção e o desenvolvimento de um aplicativo Android para prover o gerenciamento de roteiros turísticos a partir de uma abordagem distribuída com o armazenamento de dados na nuvem com a plataforma Google App Engine. A partir de um estudo sobre dispositivos móveis e serviços baseados na localização e utilizando uma metodologia ágil de desenvolvimento, o sistema foi criado visando auxiliar o turista em questões dinâmicas de uma viagem, como buscas por lugares próximos e criação de rotas entre diversos pontos.

Palavras-Chave: Android, turismo, Google App Engine, LBS, dispositivos móveis.

An Android System for Managing Tourist Itineraries

ABSTRACT

With the growth of the mobile device market and the significant predicted growth in tourism in Brazil, this paper presents the design and development of an Android app to provide tourist itineraries management from a distributed approach with the data storage in the cloud using Google App Engine platform. From a study on mobile devices and location-based services (LBS) and using an agile development methodology, the system was created to assist the tourist in dynamic questions from a trip, as searches for nearby places and creating routes between various points.

Keywords: Android, tourism, Google App Engine, LBS, mobile devices.

1 INTRODUÇÃO

Cada vez mais os dispositivos móveis fazem parte da vida cotidiana das pessoas. Segundo dados de Fling (2009), mais de 1.6 bilhão de pessoas (ou cerca de 25% da população mundial) já tinha acesso a internet através de dispositivos móveis, um número que é ainda mais expressivo se comparado com o fato de que apenas 1.1 bilhão de pessoas tinha acesso a um desktop com conexão a internet.

Dentro desse cenário, dados do IDC (2012) apontam que, em agosto de 2012, a plataforma Android já detinha uma fatia de mais de 60% do mercado, representando um crescimento de quase 20% em relação ao mesmo período do ano anterior. Também é preciso destacar que a população tem cada vez mais acesso a planos de dados, e com a expansão da cobertura 3G e 4G os serviços baseados em localização (Location Based Services - LBS, em inglês) são cada vez mais utilizados.

Além da crescente demanda por aplicativos móveis, é importante destacar o impacto do mercado de turismo. No Brasil, com a iminência de uma copa do mundo de futebol sendo sediada no país, há uma grande expectativa de aumento do turismo. Segundo dados do Ministério do Turismo (2011), a estimativa é que os desembarques domésticos saltem dos 56 milhões, registrados em 2009, para 73 milhões no ano da copa, em 2014, e que a entrada de divisas internacionais salte de R$ 6,3 bilhões para R$ 8,9 bilhões no mesmo período.

Apesar do grande mercado em contínua expansão, poucas aplicações voltadas para o turismo são encontradas nas principais lojas de aplicativos para dispositivos móveis. Buscando por termos relacionados com o turismo na loja da Google para Android (a Google Play), poucas opções são oferecidas, e destas opções muitas focam apenas em uma área específica ou dispõem de pouquíssimos recursos. Além disso, as ferramentas encontradas não possuem uma forma prática de lidar com o gerenciamento de uma viagem e apenas mostram lugares baseados em algum termo de pesquisa.

O objetivo principal deste trabalho é desenvolver uma aplicação voltada para o gerenciamento de roteiros turísticos em dispositivos móveis, utilizando informações de localização geográfica do usuário disponibilizadas pelo dispositivo para buscar pontos de interesses próximos e auxiliar o turista com as questões dinâmicas de uma viagem, tais como imprevistos (um dia de chuva, por exemplo) e rotas entre locais. Essa aplicação será implementada utilizando uma das plataformas mais populares para dispositivos móveis: o Android.

Outro objetivo importante é criar um sistema que possibilite a expansão do aplicativo para outras plataformas, também móveis, e que permita uma integração com outros sistemas turísticos.

Para cumprir os objetivos apontados, este trabalho foi dividido em seis capítulos. Após a introdução, o capítulo dois apresenta os principais conceitos relacionados ao trabalho, ressaltando o conceito de serviços baseados na localização, mobilidade e dispositivos móveis e a problemática do turismo.

No capítulo três serão apresentadas as plataformas utilizadas no desenvolvimento da solução, com destaque para a plataforma Android e a plataforma Google App Engine. Também serão apresentadas as demais API utilizadas.

A modelagem e estrutura da aplicação serão apresentadas no capítulo quatro, começando pela identificação do problema. Outros aspectos abordados serão o padrão MVC, a arquitetura geral do sistema, as tabelas da base de dados e as histórias de usuário que definem as funcionalidades da aplicação.

No quinto capítulo o foco é a implementação do aplicativo. Além da justificativa para escolha das plataformas e da metodologia de desenvolvimento serão discutidos tópicos específicos do desenvolvimento como a obtenção dos dados e a armazenagem dos dados gerados pelo aplicativo. Também serão apresentadas as telas exibindo as funcionalidades implementadas, a integração do sistema com uma plataforma de planejamento de viagens e a comparação do aplicativo desenvolvido com outros aplicativos semelhantes.

Por fim, no capítulo seis, será apresentada a conclusão do trabalho.

2 CONCEITOS RELACIONADOS

Neste capítulo serão apresentados alguns conceitos fundamentais relacionados ao trabalho. Primeiramente será introduzido o conceito de serviços baseados em localização, mostrando em que áreas e como pode ser utilizado. Também serão expostos os conceitos de mobilidade e dispositivos móveis, trazendo dados da participação de diferentes plataformas no mercado, e, por fim, uma breve introdução à problemática do turismo.

2.1 Location Based Services

O termo Location Based Services (LBS) é usado em diferentes áreas, como computação e telecomunicações, e não possui apenas uma definição. Segundo Schiller e Voisard (2004), é um conceito que denota aplicações que integram localização geográfica com a noção geral de serviços. Como exemplos são citadas aplicações que incluem: serviços de emergência, sistemas de navegação de veículos e ferramentas para planejamento de passeios turísticos.

Outra definição apresentada por Ferraro (2010), afirma que o conceito de LBS ser simplesmente um serviço de informação, acessível por dispositivos móveis através da rede móvel e utilizando a habilidade de fazer uso da posição geográfica do dispositivo é desatualizado devido a nova geração de serviços móveis e web, onde a interação com o usuário e a habilidade de gerar conteúdo é fundamental para serviços e aplicações. O autor então sugere uma definição onde LBS é um serviço que é capaz de determinar a localização do usuário ("onde estou?"), retornar informações espacialmente relacionadas com o usuário ("o que posso fazer por perto?") e oferece ao usuário uma interação dinâmica com a informação do local ou conteúdo ("o que eu acho desse lugar?").

Alguns componentes-chave para um LBS são: um dispositivo móvel, um provedor de conteúdo, uma rede de comunicação e um componente de localização.

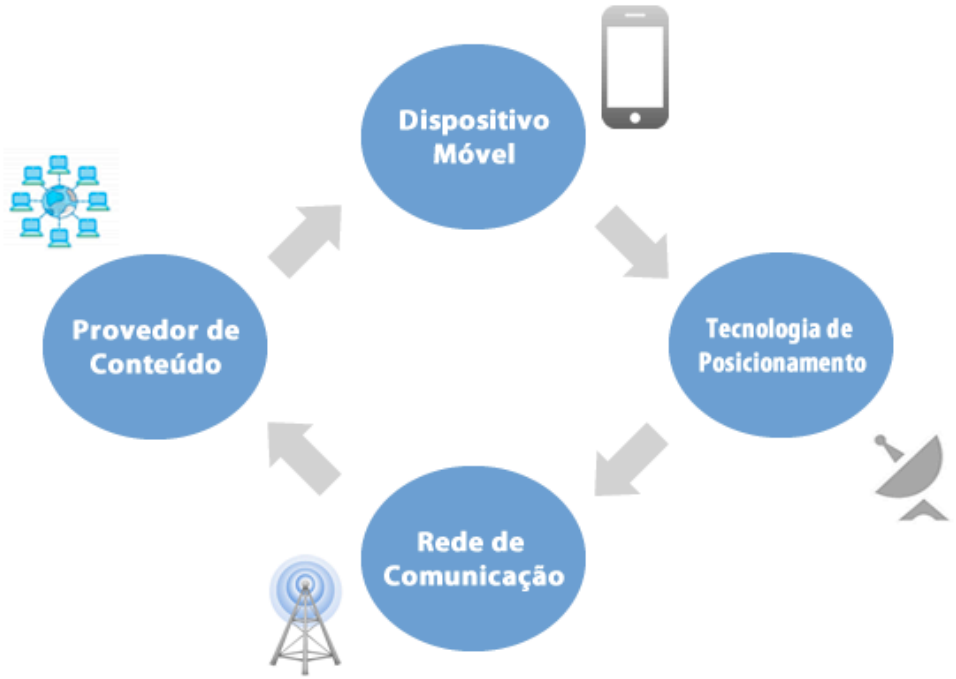


Figura 2.1 - Ilustração de um Location Based Service com seus quatro componentes chave.

Fonte: Elaborada pelo autor com base em Ferraro (2010, p.2).

2.1.1 Dispositivo Móvel

A grande variedade de sistemas para dispositivos móveis aumenta a complexidade dos sistemas baseados em localização. Mais informações sobre dispositivos móveis serão apresentados na seção 1.2.

2.1.2 Provedores de Conteúdo

São as fontes dos dados da aplicação. Ferraro (2010) afirma que os provedores de LBS normalmente não mantêm todo o conteúdo e dados que estão sendo acessados pelo usuário no dispositivo móvel, utilizando diversas fontes, como é o caso da aplicação apresentada neste trabalho, onde o usuário acessa informações de múltiplas companhias.

2.1.3 Rede de Comunicação

É a forma de comunicação do dispositivo móvel com os provedores de conteúdo. Atualmente muitos dispositivos utilizam a rede 3G para a comunicação de dados.

2.1.4 Componente de Localização

Os componentes de localização estão cada vez mais escondidos até mesmos dos desenvolvedores de aplicação, uma vez que eles estão incorporados em muitos dispositivos móveis. Ainda assim existem diversos meios de se obter a localização, como: triangulação, Cell ID, GPS e posicionamento WI-FI. Nos dispositivos que

possuem mais de uma tecnologia de localização, a combinação delas é usada para obter uma melhor precisão.

2.1.5 Tipos de Serviço

Segundo Schiller e Voisard (2004), pode-se classificar os serviços baseados em localização em dois tipos de aplicações. Os serviços do tipo *Push* implicam que o usuário recebe informações relativa a sua localização sem ter explicitamente requisitado. Em contraste, temos os serviços do tipo *Pull,* no qual o usuário ativa uma aplicação e, nesse contexto, "puxa" a informação da rede, sendo que essa informação pode conter a localização.

Na aplicação apresentada neste trabalho serão utilizadas as duas abordagens.

TABELA 2.1: Categorias e exemplos de aplicações LBS

	Serviços *Push*	**Serviços *Pull***
Orientados à Pessoa		
Comunicação	Ex. 1: Receber um alerta de uma aplicação de zona de amigos de que um amigo acabou de entrar na sua área Ex. 2: Uma mensagem é apresentada perguntando se um amigo pode localizar o usuário.	Ex. 1: O usuário faz uma requisição a um aplicativo de busca de amigos para mostrar quem está próximo.
Informação	Ex. 3: O usuário recebe um alerta de catástrofe na cidade na qual ele está.	Ex. 2: O usuário busca pelo cinema mais próximo e por instruções de navegação até lá.
Entretenimento	Ex. 4: O usuário aceita participar de um jogo de tiros baseado em localização e está sendo atacado.	Ex. 3: O usuário está jogando um jogo baseado na localização e procura alguém próximo para atacar.
Comércio e Propaganda	Ex. 5: Um cupom de desconto é enviado ao usuário por um restaurante na qual ele está próximo.	Ex. 4: O usuário busca por eventos próximos a ele.
Orientados ao Dispositivo		
Rastreamento	Ex. 6: Um alerta é enviado ao usuário por um sistema de rastreamento de que uma de suas encomendas saiu da rota prevista. Ex. 7: O usuário recebe um alerta de que seu filho saiu do parquinho.	Ex. 5: O usuário requisita informação sobre a localização da frota de caminhões no país.

FONTE: Traduzida pelo autor com base em Schiller e Voisard (2004, p.15).

2.2 Mobilidade e Dispositivos Móveis

Segundo Fling (2009), os dispositivos móveis já ultrapassaram a maioria dos meios de comunicação que são utilizados no dia a dia de uma pessoa. Hoje, mais pessoas acessam a web através de dispositivos móveis do que através de um computador, e essa diferença entre os dois meios deve crescer nos próximos anos, como mostrado na figura 2.2.

Ainda segundo Fling (2009), cerca de 25% da população mundial (aproximadamente 1.6 bilhões de pessoas) têm acesso a web através de um dispositivo móvel, o que é ainda mais impressionante se for comparado com o número de pessoas com acesso através de um computador, de "apenas" 1.1 bilhão de pessoas.

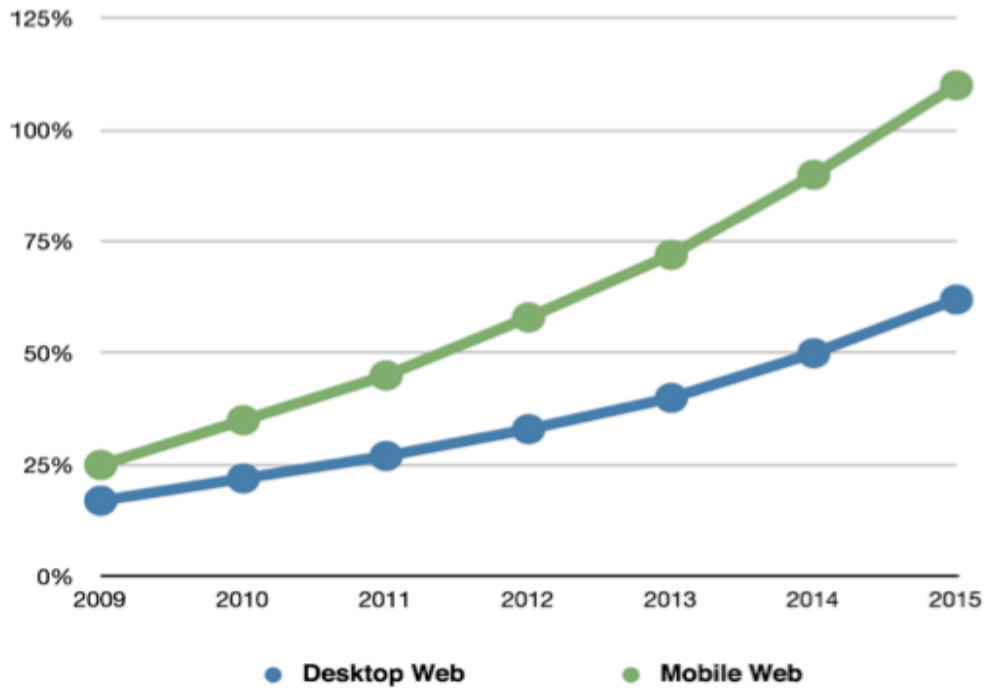


Figura 2.2 – Projeção do crescimento de acesso a Web em dispositivos móveis e computadores.

Fonte: Fling (2009, p. 33).

2.2.1 Plataformas

Fling (2009) afirma que o principal dever de uma plataforma móvel é prover ao usuário acesso às funcionalidades do dispositivo, permitindo a execução de softwares e serviços. Diversas plataformas são disponibilizadas para dispositivos móveis, sendo que as principais são a plataforma Android e a iOS, conforme IDC (2012).

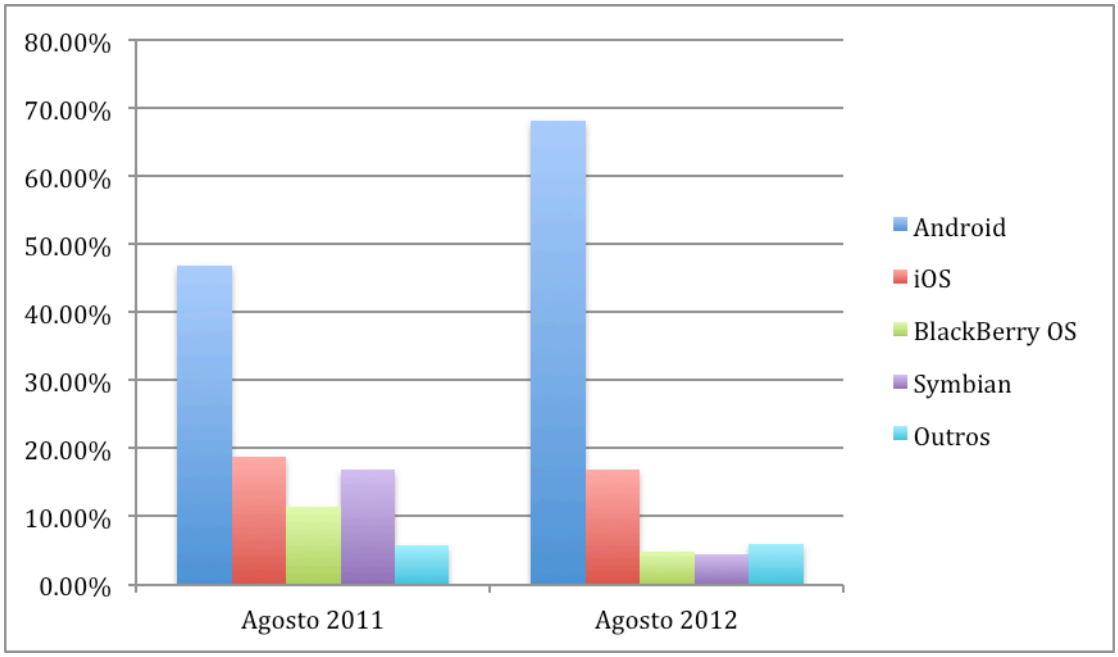


Figura 2.3 – Participação de mercado de plataformas móveis.

Fonte: elaborada pelo autor com base em dados de IDC (2012).

2.2.2 Crescimento das plataformas Android e iOS

Dados retirados de Flurry (2012) revelam que a taxa de adoção de aparelhos Android e iOS supera qualquer tecnologia de consumo na história. Se comparado com tecnologias recentes, a adoção de Smartphones é dez vezes mais rápida que a de PC's na década de 80, duas vezes mais rápida que o boom da internet na década de 90 e três vezes mais rápida que a adoção de redes sociais recentes. Flurry (2012) estima que existiam mais de 640 milhões de dispositivos Android e iOS em uso no mês de julho de 2012.

A Figura 2.4 mostra o mercado de dispositivos Android e iOS nos países em que possuem maior penetração. Há de se destacar que o Brasil já ocupa o décimo posto, com cerca de 13 milhões de dispositivos.

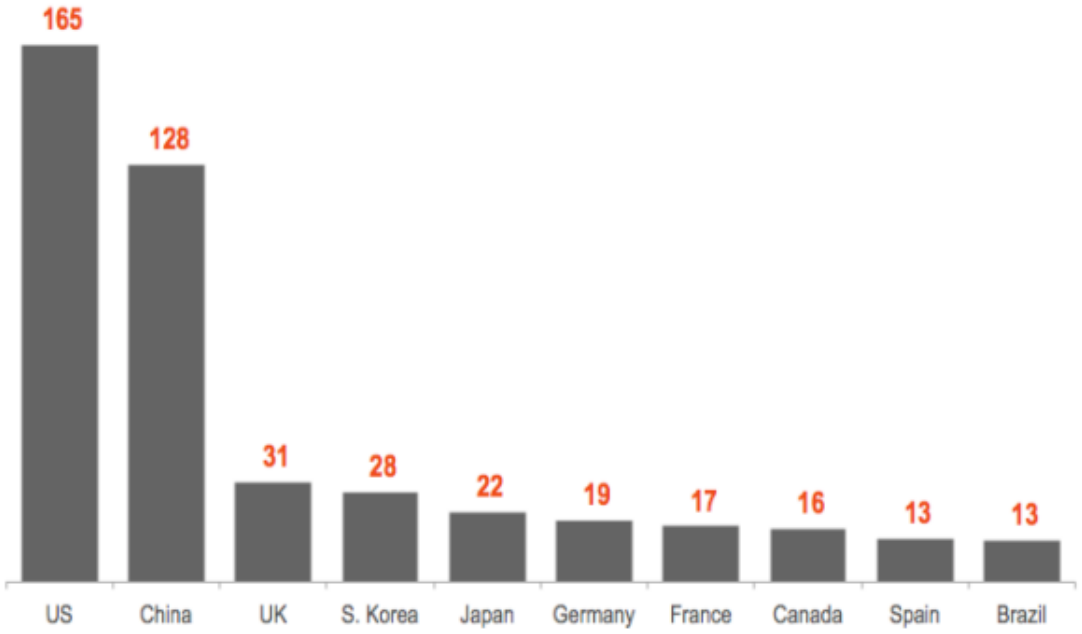


Figura 2.4 – Número de dispositivos Android e iOS (em milhões) ativos em Julho de 2012.

Fonte: Flurry (2012)

Ainda levando em conta dados fornecidos por Flurry (2012), pode-se destacar o crescimento do mercado destes dispositivos em diversos países em julho de 2012 em comparação com o mesmo período do ano anterior, sendo que no Brasil o crescimento foi de 220%, ficando atrás apenas de China e Chile.

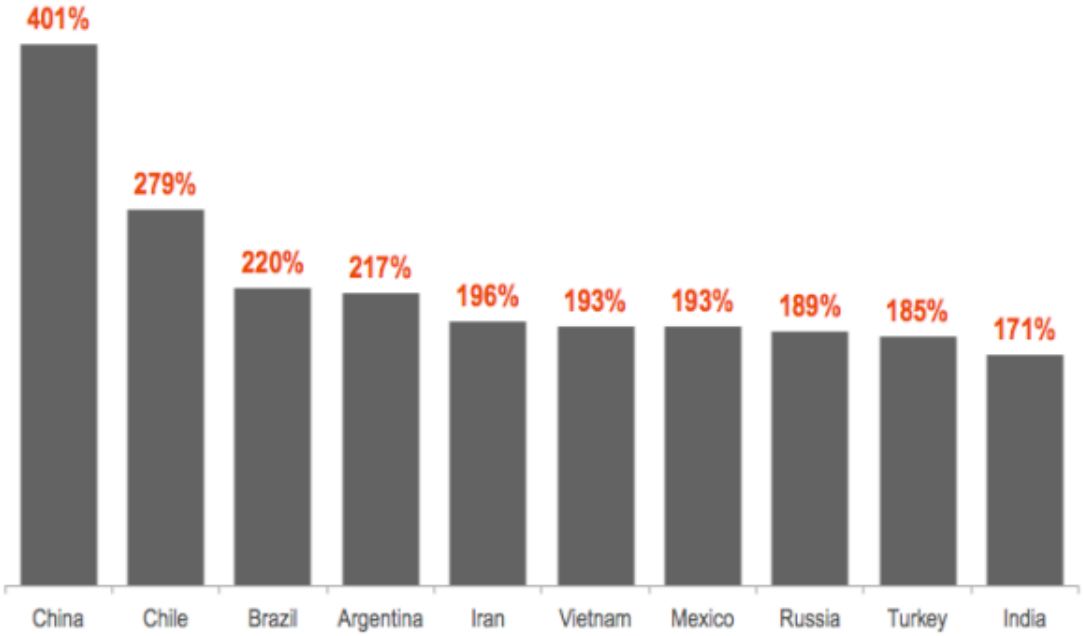


Figura 2.5 – Crescimento de dispositivos Android e iOS entre Julho de 2011 e Julho de 2012.

Fonte: Flurry (2012)

2.3 Problemática do Turismo

Brown e Chalmers (2003) afirmam que turistas gastam um tempo considerável planejando suas viagens, mas que criam os planos de uma forma não muito estruturada e específica para que possam se adaptar no caso de algum imprevisto. Esta subseção mostra, com base no trabalho dos autores acima citados, como turistas se comportam durante uma viagem.

Turistas de uma maneira geral encontram diversos desafios durante uma viagem. O primeiro ponto citado por Brown e Chalmers (2003) parece muito simples para pessoas que não estão familiarizadas com o local: o que se pode fazer naquele lugar. Essa decisão é relativa a vários fatores, como o objetivo do turista (relaxar, fazer compras, etc.), tempo para chegar em diferentes locais ou até mesmo as condições climáticas.

Outro ponto salientado pelos autores é como achar onde as coisas estão. Em uma visita a alguma cidade, muitas das atrações estão distribuídas pela cidade, criando uma necessidade de evitar o gasto excessivo de tempo se locomovendo entre lugares, buscando agrupar atrações próximas.

Duas das principais fontes de informações de um turista são publicações turísticas e os mapas, que acabam sendo utilizadas em conjunto. Mesmo assim, encontrar uma atração de uma publicação pode ser uma tarefa desafiadora. Muitas vezes os turistas possuem apenas uma vaga ideia de onde estão, e acabam usando o mapa como forma de se orientar para uma direção relativamente correta, em vez de procurar uma rota específica. O aplicativo proposto neste trabalho se utiliza de um sistema de mapas com a localização geográfica do usuário visando auxiliar o turista na sua busca por locais e também na criação de rotas.

3 PLATAFORMAS

Neste capítulo serão introduzidas as plataformas empregadas no desenvolvimento da solução proposta por este trabalho. Também serão abordadas outras API que foram utilizadas no projeto.

3.1 Plataforma Android

Nesta seção será apresentada uma visão geral da plataforma Android, na qual o aplicativo deste trabalho foi implementado. Primeiramente será introduzido um breve histórico da plataforma e suas versões, seguido de uma visão geral das características da plataforma.

3.1.1 Histórico e Versões[1]

O Android tem origem em uma StartUp homônima que fora adquirida pelo Google em 2005. Mais tarde, em 2007, o Android passou a ser mantido pela Open Handset Alliance, uma aliança formada com intuito de criar uma plataforma mobile melhor e mais aberta, que tinha como membros a T-Mobile, Motorola, Samsung e a própria Google, entre outros. Em 2011 essa aliança já possuía mais de 80 membros.

O primeiro dispositivo Android foi lançado pela T-Mobile, o T-Mobile G1, em setembro de 2008, um pouco antes de a Google tornar o código fonte disponível sob a licença open source Apache. Desde então diversas versões do Android foram lançadas.

[1] Esta subseção foi baseada em Google (2012), Ableson e Collins (2012), Hashimi, Komatineni e Maclean (2011).

[2] Esta subseção foi baseada em Burnette (2010).

[3] Esta subseção foi baseada em Roche e Douglas (2009), Zahariev (2009) e Sanderson (2012).

TABELA 3.1: Histórico de versões do Android

Versão	Codinome	API	Distribuição
1.5	Cupcake	3	0.1%
1.6	Donut	4	0.3%
2.1	Eclair	7	2.7%
2.2	Froyo	8	10.3%
2.3 – 2.3.2	Gingerbread	9	0.2%
2.3.3 – 2.3.7		10	50.6%
3.1	Honeycomb	12	0.4%
3.2		13	1.2%
4.0.3 – 4.0.4	Ice Cream Sandwich	15	27.5%
4.1	Jelly Bean	16	5.9%
4.2		17	0.8%

Fonte: Google (2012)

3.1.2 Arquitetura[2]

A arquitetura do sistema é baseada em camadas e componentes, na qual cada camada utiliza os serviços providos pela camada abaixo. Nesta seção serão discutidos alguns aspectos destas camadas.

[2] Esta subseção foi baseada em Burnette (2010).

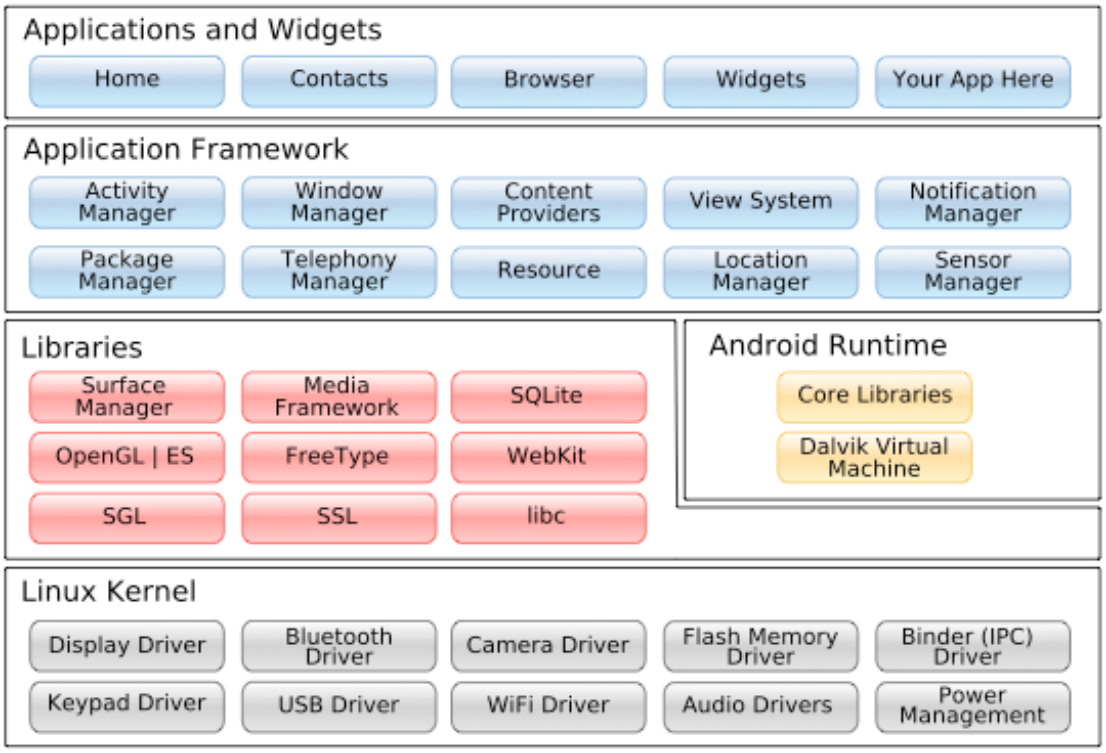


Figura 3.1: Arquitetura Android.

Fonte: Burnette (2010, p.31)

3.1.2.1 Kernel Linux

O Android é construído em cima de um kernel Linux, que é usado internamente pelo sistema para o gerenciamento de memória, gerenciamento de processos e outros serviços do sistema operacional. É importante notar que o usuário de um aparelho Android nunca verá o sistema Linux, assim como seus programas não farão chamadas Linux diretas.

3.1.2.2 Bibliotecas Nativas

A camada acima ao kernel contém as bibliotecas Android nativas. Essas bibliotecas compartilhadas são todas escritas em C ou C++, compiladas para a arquitetura específica usada pelo aparelho, e pré-instaladas pelo fabricante do mesmo. Entre as biblitecas mais importantes podemos citar: SQLite, WebKit, Media Codecs, FreeType e OpenGL.

3.1.2.3 Android Runtime

Também uma camada acima do kernel, esta camada é responsável pela execução do código compilado. Nela está a máquina virtual Dalvik, uma implementação Java do Google, otimizada para dispositivos móveis. Todo o código escrito para Android é escrito em Java e é executado nesta máquina virtual.

A Dalvik, diferentemente do Java tradicional, executa arquivos .dex, que são arquivos convertidos em tempo de compilação a partir dos arquivos padrões .class e .jar, tornando-os mais compactos e eficientes, características importantes se levado em

consideração as limitações de memória e bateria dos aparelhos Android. Outra diferença importante é que o núcleo de bibliotecas que estão no Android são diferentes do Java SE e do Java ME.

3.1.2.4 Framework de Aplicação

Esta camada, situada acima da camada de bibliotecas e de execução, é responsável por prover os programas básicos para gerenciamento do aparelho, permitindo que os desenvolvedores tenham acesso e possam tirar proveito das capacidades do hardware disponível. As partes mais importantes do framework são:

- **Activity Manager**: responsável por controlar o ciclo de vida das aplicações. (Ver seção 1.2.4);
- **Content Providers**: encapsulam os dados que precisam ser compartilhados entre aplicações, como os contatos.
- **Resource Manager**: controla os recursos do aplicativo desenvolvido que não são código.
- **Location Manager**: responsável pela localização geográfica do aparelho.
- **Notification Manager**: mostra eventos ao usuário, tais como, chegada de mensagens, compromissos e alertas.

3.1.2.5 Aplicação

A camada do topo da arquitetura Android é a de aplicações, onde estão os aplicativos que são realmente vistos pelo usuário. Nessa camada estão tanto os aplicativos que já vêm com aparelho, como o aplicativo para contatos e o navegador web, como os aplicativos adquiridos pelo usuário através de download.

3.1.3 Blocos de Construção

Para o desenvolvimento de uma aplicação Android, o desenvolvedor deve estar familiarizado com alguns objetos da SDK Android. Entre os mais importantes estão as Atividades e os Serviços.

Uma Atividade é uma tela de interface de usuário. Uma aplicação pode definir uma ou mais Atividades para lidar com as diferentes fases do programa. Já um serviço é uma tarefa que é executada em segundo plano, sem interação direta com o usuário.

3.1.4 Ciclo de Vida de uma Atividade

Durante seu tempo de vida, cada atividade de um aplicativo Android pode estar em um de diversos estados. Esse estado não é determinado pelo desenvolvedor e sim pelo sistema, porém a cada mudança de estado o aplicativo recebe uma notificação do sistema através da chamada de um método específico para cada tipo de mudança.

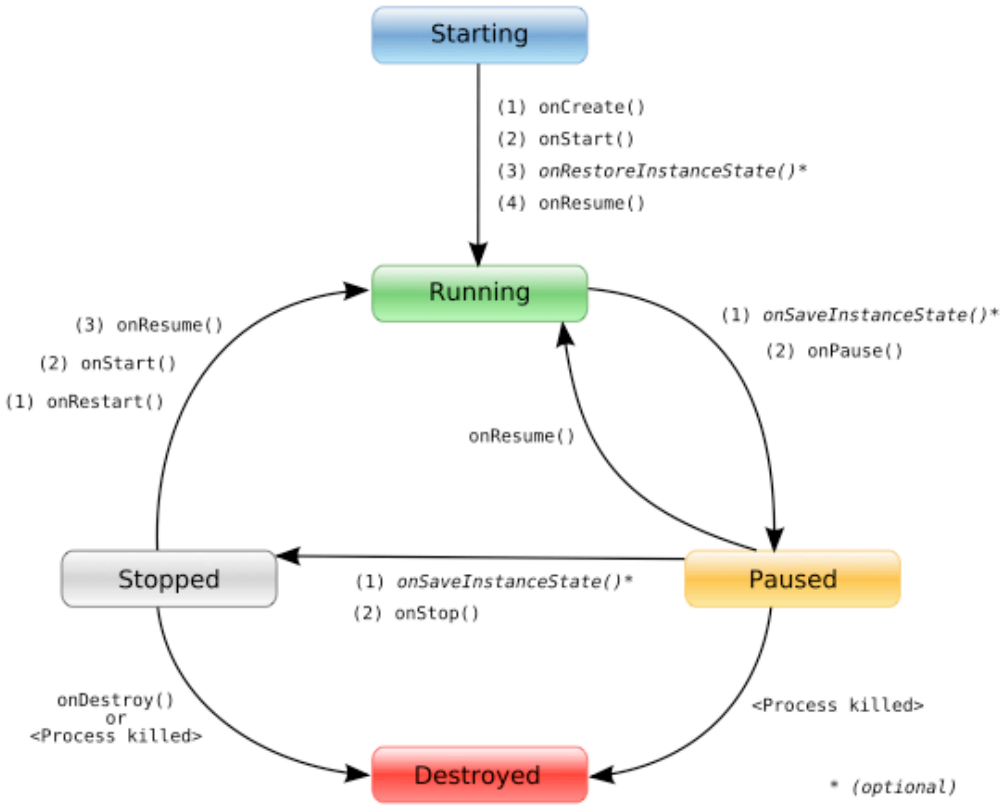


Figura 3.2: Ciclo de vida de uma aplicação Android.

Fonte: Burnette (2010, p.37)

Entre os métodos de mudança de estado mais importantes estão:

- **onCreate**: É chamado quando a Atividade é iniciada pela primeira vez. Pode ser usada para realizar inicializações que ocorrem apenas uma vez, como a interface do usuário.
- **onStart**: Indica que a atividade está prestes a ser exibida para o usuário.
- **onResume**: É chamado quando uma atividade começa a interagir com o usuário.
- **onPause**: É executado quando uma Atividade está prestes a ser movida para segundo plano. Isso acontece normalmente quando outra Atividade é lançada acima da atual.
- **onDestroy**: Chamada logo antes da atividade ser destruída.

3.2 Plataforma Google App Engine[3]

O Google App Engine é um serviço de hospedagem para aplicações web e seu ambiente é especialmente projetado para aplicações dinâmicas de tempo real.

[3] Esta subseção foi baseada em Roche e Douglas (2009), Zahariev (2009) e Sanderson (2012).

Diferentemente dos serviços de hospedagem tradicionais, no App Engine o usuário só paga pelos recursos que forem usados. Nesta seção serão introduzidas as principais características desta plataforma.

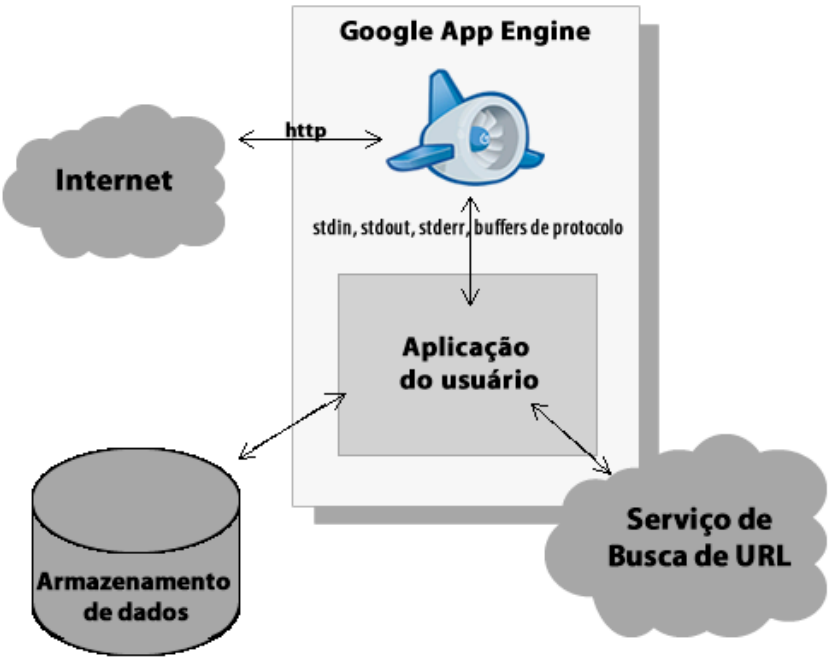


Figura 3.3: Arquitetura de uma aplicação no App Engine

Fonte: Adaptada pelo autor com base em Roche e Douglas (2009, p.8).

3.2.1 App Engine e a computação na Nuvem

Computação em nuvem é um modelo de computação na qual diferentes tarefas são atribuídas usando uma combinação de conexão, software e serviços acessados através da rede. Em palavras mais simples, usuários podem ter acesso a um grande poder computacional usando uma conexão através de um smartphone, por exemplo. O Google App Engine é um serviço que permite a utilização da infraestrutura da própria Google, a qual está abstraída no conceito de nuvem.

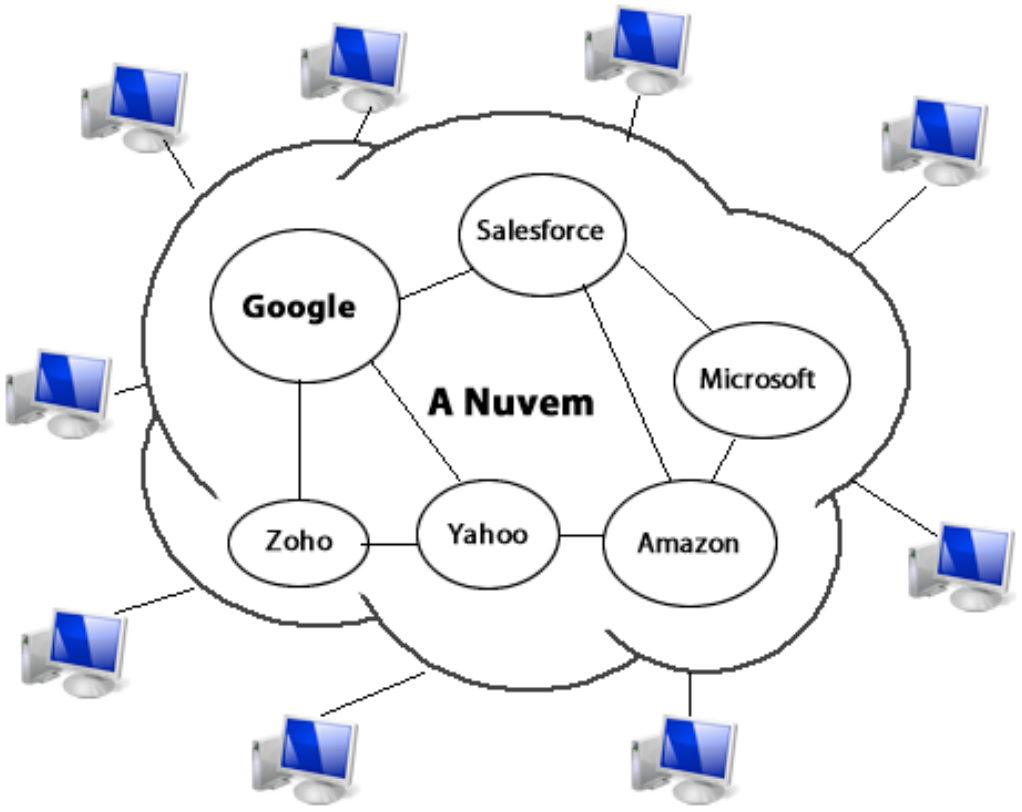


Figura 3.4: Exemplo de plataformas na nuvem.

Fonte: Adaptado pelo autor com base em Zahariev (2009, p.1).

3.2.2 O ambiente de tempo de execução

Uma aplicação App Engine responde a requisições web, que são inicializadas quando um cliente faz uma requisição HTTP. Quando a requisição é recebida, o App Engine seleciona um servidor entre muitos possíveis para lidar com a requisição, onde esta seleção busca o servidor que provavelmente terá o menor tempo de resposta para então chamar a aplicação com o conteúdo da requisição HTTP, receber os dados de resposta da aplicação e então retornar a resposta para o cliente.

Do ponto de vista da aplicação, o ambiente de tempo de execução apenas é criado quando uma requisição começa a ser tratada e desaparece quando ela acaba. Como o estado do ambiente não é mantido entre requisições, o App Engine consegue distribuir o tráfego entre vários servidores. Também é necessário ter em mente que vários métodos de armazenamento de dados são providos pelo App Engine para a persistência entre requisições, porém estes mecanismos não estão dentro do ambiente de tempo de execução.

Já do ponto de vista do sistema completo, o App Engine permite que ambientes de tempo de execução permaneçam ativos após uma requisição ser tratada e irá reutilizar ambientes o máximo possível para evitar inicialização desnecessária. Cada instância da aplicação possui uma memória local para armazenar o código importado e as estruturas de dados inicializadas. As instâncias são criadas e destruídas da forma que for necessária para acomodar o tráfego da aplicação.

O código da aplicação não pode acessar o servidor de uma maneira tradicional. A aplicação pode ler os seus próprios dados do sistema de arquivos, mas não pode escrever em arquivos, e também não pode ler arquivos que pertencem a outras aplicações.

Resumindo, cada requisição vive em sua própria "caixa de areia". Isso possibilita ao App Engine tratar uma requisição com o servidor que possa, de forma estimada, prover a resposta mais rápida. É importante salientar que não há como garantir que uma instância receberá duas requisições, mesmo se estas vierem do mesmo cliente e chegarem relativamente rápidas. Essa noção de "caixa de areia" também permite que o App Engine execute várias aplicações no mesmo servidor sem que o comportamento de uma aplicação afete outra. Além de limitar o acesso ao sistema operacional, o ambiente de tempo de execução também limita a quantidade de tempo de relógio e de memória que uma única requisição pode tomar.

A plataforma oferece três possibilidades de ambientes de tempo de execução para aplicações: um ambiente Java, um ambiente Python e um ambiente baseado na linguagem Go (linguagem desenvolvida pela Google). O aplicativo proposto neste trabalho utiliza o ambiente Java.

3.2.3 O Servidor de Arquivos Estáticos

A maior parte dos websites possuem recursos que são entregues diretamente aos navegadores e que não mudam durante a operação regular da página. Imagens, arquivos CSS, arquivos JavaScript e arquivos HTML sem componentes dinâmicos, por exemplo, são conhecidos como arquivos estáticos.

Como o envio desses arquivos não envolve o código da aplicação, o App Engine provê um conjunto separado de servidores dedicados para enviar arquivos estáticos. Estes servidores são otimizados para receber requisições deste tipo de arquivos. Para o cliente estes arquivos são vistos como qualquer outro recurso da aplicação.

3.2.4 O Armazenamento de Dados

Uma organização normal para um pequeno Website envolve um único servidor de banco de dados para o site inteiro a um ou mais servidores web que se conectam ao banco de dados para armazenar ou recuperar dados. Utilizar um banco de dados central torna fácil manter uma representação canônica dos dados, para que múltiplos usuários acessando múltiplos servidores web visualizem as mesmas e mais recentes informações. O problema de um servidor central é a dificuldade para escalar uma vez que ele chega na capacidade máxima para conexões simultâneas.

Nas últimas duas décadas a forma mais popular de armazenamento de dados foi o banco de dados relacional, com tabelas de linhas e colunas organizadas para concisão e eficiência de espaço. Outros tipos de armazenamento de dados incluem banco de dados hierárquicos e de objetos. Cada tipo tem seus prós e contras, e cada tipo tem suas próprias técnicas para escalabilidade.

O sistema de armazenamento de dados do App Engine se assemelha a um banco de dados orientado a objetos. Como no ambiente de tempo de execução, o projeto do sistema de armazenamento de dados do App Engine é uma abstração que permite ao App Engine tratar os detalhes de distribuição e escalabilidade da aplicação.

3.2.4.1 Entidades e Propriedades

Uma aplicação App Engine armazena seus dados como uma ou mais *entidades*. Uma Entidade tem uma ou mais *propriedades*, cada qual tem um nome e um valor que é de um tipo dentre diversos tipos primitivos. Cada entidade é de um *tipo* nomeado que categoriza a entidade para o propósito de pesquisa.

Este sistema pode parecer com um banco de dados relacional: entidades de um tipo são como linhas em uma tabela, e propriedades são como colunas (campos). Entretanto, há duas grandes diferenças entre entidades e linhas. Primeiro, uma entidade de um dado tipo não precisa ter as mesmas propriedades que outras entidades de mesmo tipo. Segundo, uma entidade pode ter uma propriedade do mesmo nome que outra entidade tem, mas com um valor de diferente tipo.

Cada entidade possui uma chave única que pode ser provida pela aplicação ou gerada pelo App Engine. Diferentemente de um banco de dados relacional, a chave não é um campo ou propriedade, mas um aspecto independente da entidade. Assim, é possível rapidamente recuperar uma entidade se a chave é conhecida.

3.2.4.2 Consultas e Índices

Uma consulta pode retornar zero ou mais entidades de um único tipo. Ela também pode retornar apenas as chaves das entidades que seriam retornadas por uma consulta.

Em um banco de dados relacional, as consultas são planejadas e executadas em tempo real nas tabelas de dados, as quais são armazenadas como elas foram projetadas pelo desenvolvedor, que por sua vez pode criar e manter índices em certas colunas para aumentar a velocidade de certas consultas. O banco de dados do App Engine faz algo bastante diferente: toda consulta tem um índice correspondente armazenado. Assim, quando uma aplicação faz uma consulta, o banco de dados acha o índice dela, busca a primeira linha que satisfaz a condição, e então retorna a entidade para cada linha consecutiva do índice até que a linha não satisfaça a condição.

Para isso ser possível, é necessário que o App Engine saiba quais consultas a aplicação irá fazer. Assim, o ambiente provê um conjunto de índices padrão para consultas simples, baseadas em quais propriedades existem em entidades de um tipo. Para consultas mais complexas, a aplicação deve incluir índices específicos na sua configuração.

3.2.4.3 Os Serviços

O relacionamento do sistema de armazenamento de dados com o ambiente de tempo de execução é feito através de um serviço: a aplicação usa uma API para acessar um sistema separado que gerencia de forma separada das instâncias da aplicação. A plataforma App Engine possui diversos outros serviços, nos quais se pode citar:

- A cache de memória: é um serviço de armazenamento de dados do tipo chave-valor de tempo curto, mais rápido que o armazenamento padrão, porém não é persistente entre servidores.

- Busca de URL: é por este serviço que as aplicações podem acessar outros recursos web. O serviço faz uma requisição HTTP para outros servidores na Internet como para recuperar páginas ou interagir com web services.
- Serviço de E-Mail (Mail), Serviço de armazenamento de valores grandes (Blobstore), Mensagens instantâneas, etc.

3.3 Outras API

Além das plataformas de desenvolvimento já discutidas, foram utilizadas algumas API como fonte de dados. Nesta subseção elas serão brevemente discutidas.

3.3.1 Google Places API

Segundo Google 2012, a Places API é um serviço que retorna informação sobre locais - definidos na API como estabelecimentos, localizações geográficas ou algum proeminente ponto de interesse - utilizando requisições HTTP. Estas requisições especificam localizações como coordenadas na forma latitude e longitude.

Entre as requisições disponíveis pode-se destacar:

Busca de lugares: retorna uma lista de pontos baseados na localização do usuário ou de uma consulta.

Busca por detalhes: requisição que retorna informações mais detalhadas sobre um lugar específico.

Cada uma das requisições HTTP destes serviços retorna uma resposta no formato JSON ou XML. Também é importante destacar que todas as requisições necessitam uma chave que é gerada pela Google para o aplicativo que utiliza os serviços.

3.3.2 Facebook Login

O Facebook Login é uma funcionalidade da SDK do Facebook que permite que uma aplicação obtenha uma chave parar acessar a API do Facebook em favor do usuário (FACEBOOK 2012). Com ela o usuário pode logar no sistema utilizando seu nome de usuário e senha do Facebook. Para a utilização dessa funcionalidade é necessária a criação de um aplicativo Facebook.

3.3.3 Twitter API

A API do Twitter permite a aplicativos externos recuperar informações presentes no Twitter, como os Tweets de um determinado usuário. No aplicativo proposto por este trabalho é utilizada a função de busca, que retorna uma coleção de Tweets que correspondam a uma determinada consulta.

4 MODELO DA APLICAÇÃO

Neste capítulo será apresentada a modelagem do aplicativo mobile de turismo proposto bem como a organização geral do projeto. Primeiramente será identificado o problema que o aplicativo deseja solucionar para que então possam ser apresentadas as funcionalidades do sistema.

4.1 Identificação do problema

Segundo dados apresentados por Rieger 2012, quase metade dos turistas utilizam seus dispositivos móveis durante uma viagem. Além disso, cerca de 37% fazem buscas de restaurantes através do aparelho. Com estes dados e com o estudo feito durante este trabalho sobre LBS e dispositivos móveis, foi definido que o objetivo deste trabalho é criar um aplicativo móvel para a localização de pontos de interesse baseado na localização do turista, fornecendo rotas, sugestões de restaurantes por proximidade e opiniões de outros usuários, ajudando a pessoa a escolher as atrações que deseja visitar.

4.2 Modelagem da arquitetura

Esta seção tem como objetivo introduzir a arquitetura geral da aplicação, mostrando como ela foi estruturada visando a implementação das funcionalidades descritas nas user stories. Também será apresentado o padrão MVC que foi utilizado na criação do aplicativo Android.

4.2.1 MVC

Segundo Sommerville (2009), MVC (Model-View-Controller) é um padrão arquitetural que separa a apresentação e interação dos dados do sistema. O sistema é estruturado dentro de três componentes lógicos que interagem uns com os outros. O componente Model controla os dados do sistema e associa operações com estes dados. A View é o componente que define e gerencia como os dados vão ser apresentados ao usuário. O Controller gerencia a interação com o usuário (cliques em botões, por exemplo) e passa essas informações para a View e para o Model.

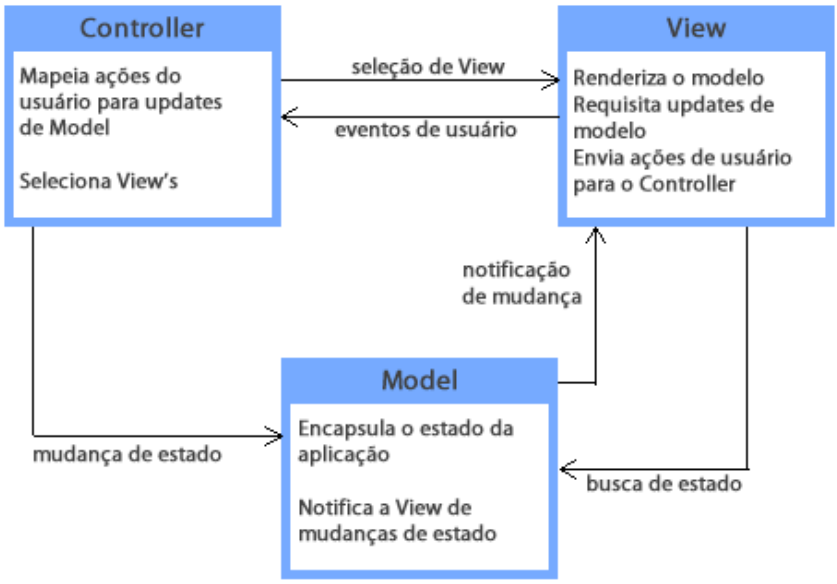


Figura 4.1: Interações entre os componentes MVC.

Fonte: Sommerville (2009, p.156)

Um aplicativo Android pode ser facilmente modelado seguindo o padrão MVC. Conforme foi apresentado na seção 3.1, uma Activity (Atividade, em português) representa um Controller, que interage com os Models representados em classes Java e com as Views, representadas por arquivos de marcação de texto XML.

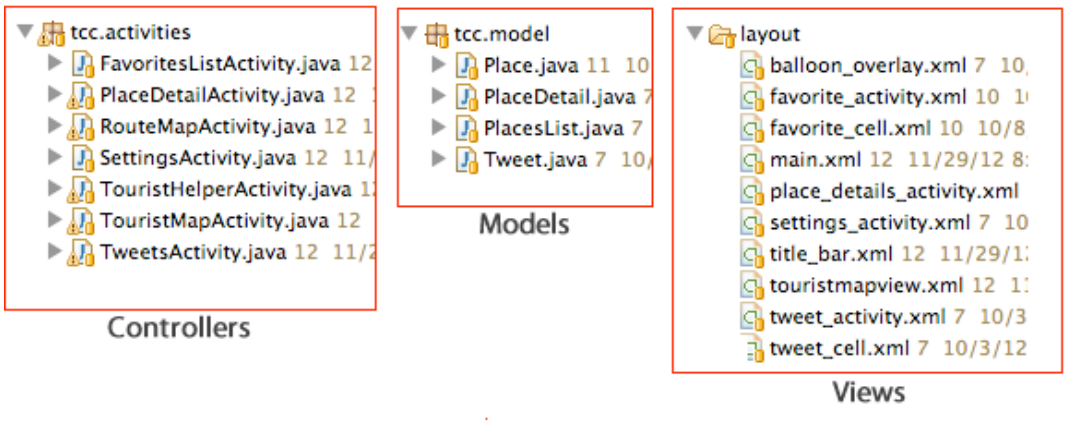


Figura 4.2: Organização MVC no projeto Android

4.2.2 Arquitetura Geral

A arquitetura do aplicativo apresentado neste trabalho foi concebida de forma que o usuário possa ter suas informações independentemente do dispositivo que ele esteja usando. Para isso ser possível, os dados do usuário são armazenados na nuvem,

utilizando a plataforma Google App Engine, ao invés de serem armazenados na memória do dispositivo. Essa abordagem também permitiria ao usuário acessar seus dados em outras plataformas em que o aplicativo fosse disponibilizado. Em outras palavras, os dados não estão ligados exclusivamente ao aparelho em que o aplicativo está instalado, mas sim ao usuário.

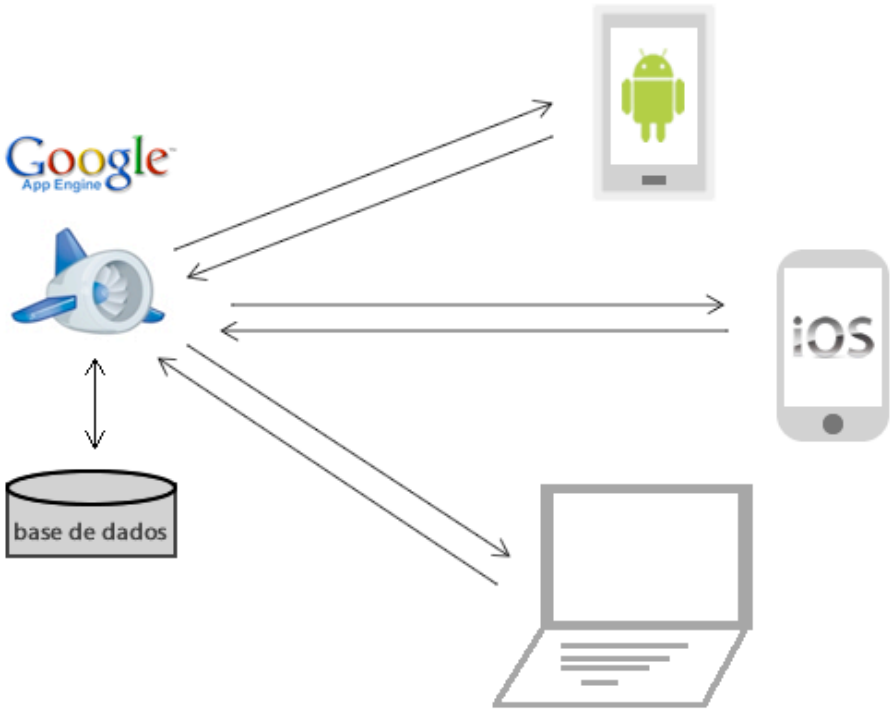


Figura 4.3: Arquitetura de comunicação entre as plataformas.

A comunicação entre os aplicativos e a base de dados acontece através de requisições web para o aplicativo desenvolvido no Google App Engine que está hospedado na nuvem.

4.3 Modelagem do aplicativo

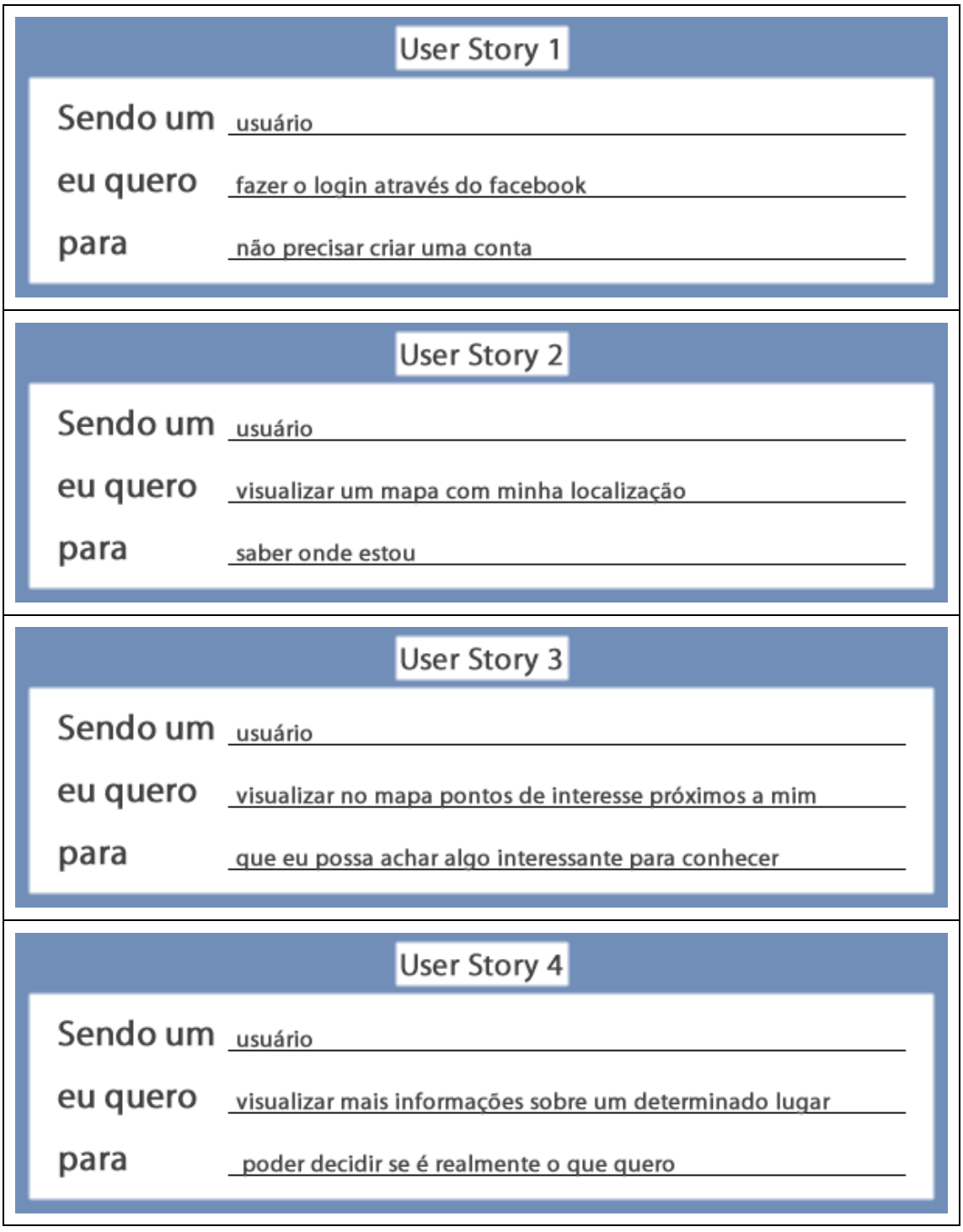
Nesta seção serão especificados os requisitos do sistema, na forma de user stories, e também a modelagem do armazenamento de dados.

4.3.1 User Stories

Segundo Nazzaro e Suscheck (2010) User Stories (histórias de usuário, no português) são narrativas de texto que descrevem uma interação de um usuário com o sistema, focando no valor que o usuário ganha do sistema, sendo uma metáfora do trabalho sendo feito. Os autores também afirmam que uma user story não é um requisito altamente documentado, mas sim um lembrete de colaboração com o tópico da user story.

O padrão de escrita das user stories deste trabalho segue o modelo proposto por Nazzaro e Suscheck (2010):

Sendo um [papel do usuário] quero [objetivo] para [razão].



User Story 5	
Sendo um	usuário
eu quero	ver o que as pessoas estão falando no twitter sobre um determinado lugar
para	que eu possa ter diferentes opiniões sobre ele

User Story 6	
Sendo um	usuário
eu quero	poder adicionar um ponto de interesse em uma lista de favoritos
para	poder visuáliza-los com mais facilidade em outro momento

User Story 7	
Sendo um	usuário
eu quero	acessar meus dados em qualquer dispositivo Android
para	que não precise criar duas contas diferentes

User Story 8	
Sendo um	usuário
eu quero	receber notificações de restaurantes próximos
para	escolher onde ir

User Story 9	
Sendo um	usuário
eu quero	visualizar rotas de carro e a pé entre meus favoritos
para	poder me locomover entre eles sem me perder

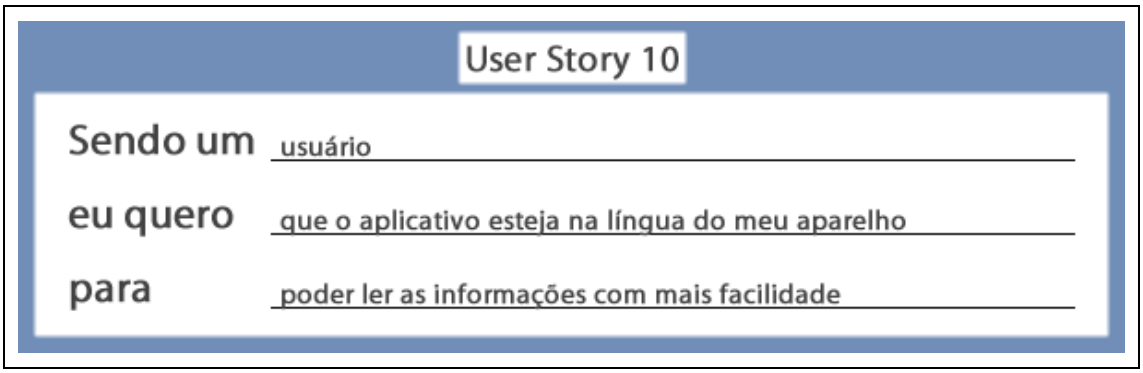


Figura 4.4: User Stories do Aplicativo.

4.3.2 Armazenamento de Dados

Como introduzido na seção 3.2, o Google App Engine possui um sistema de armazenamento baseado em entidades e propriedades. O aplicativo apresentado neste trabalho utiliza este sistema de armazenamento para salvar as informações referentes aos pontos de interesse favoritos, trips e categorias de trips do usuário.

TABELA 4.1: Modelo de uma entidade 'Favorite'

Propriedade	**Descrição**
PlaceAddress	Endereço do ponto de interesse
PlaceIcon	URL que contém o ícone para o local
PlaceId	Identificador único do local
PlaceLatitude	Latitude
PlaceLongitude	Longitude
PlaceName	Nome de apresentação do local
PlaceRating	Nota do local no sistema do Google Places
PlacePhone	Telefone
PlaceReference	Referências do local
PlaceTypes	Tipos que definem o estabelecimento
PlaceUrl	URL da página do ponto de interesse
UserId	Identificador único do usuário

TABELA 4.2: Modelo de uma entidade "Trip"

Propriedade	**Descrição**
ID	Identificador único de uma trip
TripName	Nome de exibição
TripUserId	Identificador único único do usuário

TABELA 4.3: Modelo de uma entidade “Category”

Propriedade	**Descrição**
ID	Identificador único de uma categoria
CategoryName	Nome de exibição
CategoryPlaces	Lista de referencias de pontos de interesse
CategoryTripId	Identificador que relaciona a categoria a uma trip

5 IMPLEMENTAÇÃO DO SISTEMA

Neste capítulo será elucidada a implementação da aplicação, com foco na metodologia de desenvolvimento, obtenção dos dados e o funcionamento geral do sistema. Também será explicada a integração do aplicativo com o Touristy e, por fim, será feita uma comparação com sistemas mobiles semelhantes, destacando a contribuição deste trabalho.

5.1 Escolha das Plataformas

Com base nos dados apresentados na seção 2.2 afirma-se que a plataforma móvel mais utilizada atualmente é o Android. A escolha por esta plataforma para a implementação do protótipo também se deu pela vasta documentação disponível e pela familiaridade do desenvolvedor com a linguagem Java.

Como ilustrado na Figura 4.3, o aplicativo tem como um dos objetivos ser uma solução distribuída que permita ao usuário utilizar diferentes dispositivos para acessar seus dados e também para permitir uma integração mais simples do sistema com outras plataformas (como o Touristy, que será apresentado na seção 5.6). Para atender a este requisito a escolha foi pelo Google App Engine que também possui vasta documentação de fácil acesso, possui suporte a linguagem Java e é uma solução escalável que não tem um custo inicial para utilização.

5.2 Metodologia de Desenvolvimento

O desenvolvimento do aplicativo deste trabalho foi concebido seguindo princípios ágeis, com uma modelagem menos abrangente e tendo como foco a codificação da aplicação. A escolha por esta metodologia se deu por alguns princípios presentes no manifesto ágil (presente em Cockburn, 2002), dentre eles: entregas de aplicativos funcionais frequentemente, simplicidade e priorização do software funcional mais do que documentação extensa.

Para contemplar o desenvolvimento incremental (pequenas versões do aplicativo em ciclos pequenos) foram criadas iterações para implementar funcionalidades do sistema baseada nas histórias de usuário apresentadas na seção 4.3.1. A cada iteração era obtida uma versão parcial do aplicativo. Na tabela abaixo é dada uma breve descrição das seis iterações de desenvolvimento.

TABELA 5.1: Iterações de desenvolvimento

Iteração	Descrição
1	Implementação das histórias de usuário número 2, 3 e 4 (tela de mapa com os pontos de interesse próximos e tela de detalhes)
2	Implementação das histórias de usuário número 5 e 10 (tela de tweets e internacionalização do sistema)
3	Implementação das histórias de usuário número 1, 6 e 7 (tela de login, tela de favoritos e servidor na nuvem)
4	Implementação da história de usuário número 8 (serviço de recomendação de restaurantes próximos)
5	Implementação da história de usuário número 9 (tela de rotas)
6	Implementação da integração com o sistema Touristy

5.3 Obtenção dos Dados

A implementação do aplicativo proposto neste trabalho necessita de diferentes fontes de dados para melhor atender ao usuário. Algumas das API utilizadas foram destacadas na seção 3.4, e nesta seção será exposto como os dados foram obtidos e utilizados no aplicativo.

5.3.1 Busca de Estabelecimentos

Para obter dados da API do Google Places foi necessária a criação de uma chave única de utilização de serviços do Google. Esta chave é utilizada no código fonte do aplicativo de forma com que cada requisição possa ser atribuída a um proprietário (no caso deste trabalho é o próprio autor), possibilitando a cobrança de taxas caso o número de requisições seja superior a um número determinado pelo Google.

Com a chave única em mãos, os dados podem ser obtidos através de uma requisição HTTP para a URL disponibilizada pela API, na qual são adicionados alguns parâmetros de busca:

- "Key": A chave criada
- "Location": O ponto central da busca na forma latitude e longitude.
- "Radius": O raio da busca em metros.
- "Sensor": Indica se a localização foi obtida por um sensor de localização (GPS, por exemplo).
- "Types": Parâmetro opcional que determina o tipo dos estabelecimentos da busca.

Através de uma segunda URL fornecida pela API do Google Places, é possível fazer uma busca específica por um estabelecimento visando obter mais informações sobre o mesmo. Em ambos os casos, após a requisição o aplicativo recebe um resposta no formato JSON, o qual é transformado em um objeto que contém a lista de estabelecimentos próximos.

5.3.2 Busca de Tweets

Visando buscar opiniões de diferentes pessoas sobre um determinado estabelecimento a API do Twitter foi utilizada. Ela fornece uma URL no qual os seguintes parâmetros são necessários para a busca de Tweets:

- Termo de busca: A palavra a ser buscada. No caso deste trabalho foi utilizada o nome do estabelecimento.
- Número de repostas: O número de tweets máximo a ser retornado.

Como na API do Google Places, a resposta para a requisição HTTP é no formato JSON, que é transformado em uma lista de objetos que modela o Tweet.

5.3.3 Busca de Rotas

Uma das principais funcionalidades implementada neste trabalho foi a possibilidade de criação de rotas entre mais de ponto de interesse. Para obter os dados de rota foi utilizada a API do Google Maps, que responde a uma requisição HTTP com uma String codificada com informações de rotas. Após a decodificação da resposta, é obtida uma lista de pontos na forma latitude e longitude que representam os pontos da rota. Os parâmetros necessários para a busca de uma rota entre dois pontos são:

- "saddr": Local de partida da rota na forma latitude/longitude.
- "daddr": Local de destino da rota na forma latitude/longitude.

5.4 Implementação da Base de Dados

Como definida na arquitetura da aplicação, a base de dados não é armazenada no dispositivo móvel Android, e sim na nuvem através de uma página web desenvolvida no Google App Engine. Nesta subseção serão discutidas quais funcionalidades foram desenvolvidas e como elas foram utilizadas no aplicativo.

5.4.1 Favoritos

O aplicativo desenvolvido permite ao usuário adicionar pontos de interesse que lhe interessem em uma lista de favoritos. Essa lista é armazenada na base de dados do servidor na nuvem e pode ser recuperada através da identificação do login, permitindo assim que o usuário acesse seus favoritos em diferentes dispositivos móveis. Essa comunicação entre o aplicativo Android e o servidor implementado no Google App Engine é feita através de requisições HTTP criadas na estrutura REST.

Foram implementados três métodos para o tratamento dos favoritos, que permitem a adição de um favorito, a requisição da lista de favoritos de um usuário e a deleção dos favoritos relacionados a um usuário.

5.4.1.1 Inserção de um Favorito

A adição de um favorito é feita através de uma requisição HTTP que passa como parâmetros as propriedades de um ponto de interesse, adicionado da identificação do usuário obtida através do login via Facebook. A página web processa a requisição, recuperando os parâmetros, inserindo na base de dados e retornando como resposta um booleano indicando se a inserção foi concluída ou não.

```
37    public static boolean insertFavorite(int userId, String name, String id, String reference,
38            double rating, String phone, String address, String url,
39            String icon, double lat, double lng, String types) {
40
41        boolean success = true;
42
43        DatastoreService datastore = DatastoreServiceFactory.getDatastoreService();
44
45        try {
46            //Try to add an place to a favorites list
47            Entity entity = new Entity(FavoritesKey);
48            entity.setProperty(UserIdProperty, userId);
49            entity.setProperty(PlaceNameProperty, name);
50            entity.setProperty(PlaceIdProperty, id);
51            entity.setProperty(PlaceReferenceProperty, reference);
52            entity.setProperty(PlaceRatingProperty, rating);
53            entity.setProperty(PlacePhoneProperty, phone);
54            entity.setProperty(PlaceAddressProperty, address);
55            entity.setProperty(PlaceUrlProperty, url);
56            entity.setProperty(PlaceIconProperty, icon);
57            entity.setProperty(PlaceLatitudeProperty, lat);
58            entity.setProperty(PlaceLongitudeProperty, lng);
59            entity.setProperty(PlaceTypesProperty, types);
60
61            datastore.put(entity);
62        } catch (Exception e) {
63            success = false;
64        }
65        return success;
66    }
```

Figura 5.1: Exemplo de inserção de um favorito na base de dados.

A Figura 5.1 exemplifica a inserção de dados no sistema de armazenamento provido pelo Google App Engine. Nele é criada uma entidade com a chave da tabela "Favoritos", na qual são ajustadas as propriedades relativas às informações do ponto de interesse e ao usuário, e logo em seguida é inserida no banco de dados.

5.4.1.2 Recuperar Lista de Favoritos de um Usuário

A recuperação da lista de favoritos de um usuário tem apenas como parâmetro a identificação do usuário que está buscando a lista. Com esta informação é feita uma busca na base de dados, gerando uma lista de pontos de interesse a qual é retornada para o dispositivo móvel no formato JSON, semelhante à resposta gerada pela requisição à API do Google Places.

```
@GET
@Produces({"application/json"})
@Path("/get/{userId}")
public PlacesList getFavorites(@PathParam ("userId") int userId) {
    PlacesList list = FavoritesDAO.getFavorites(userId);
    return list;
}
```

Figura 5.2: Exemplo do tratamento de uma requisição de favoritos no servidor.

5.4.2 Trips

Uma "Trip" é uma forma de organizar os pontos de interesse de forma que o usuário os associe com uma viagem. Ela é composta por categorias, que por sua vez visam acomodar os pontos de interesse seguindo critérios definidos pelo usuário. Assim como nos favoritos, as trips são armazenadas na base de dados da página web através de requisições HTTP com a estrutura REST. Também semelhante à estrutura dos favoritos foram criados os métodos de inserção de uma trip, de listagem das trips de um usuário e de deleção das trips de um usuário.

```
67  public static TripList getTrips (int userId) {
68      TripList tripList = new TripList();
69      DatastoreService datastore = DatastoreServiceFactory.getDatastoreService();
70
71      Filter filter = new FilterPredicate(TripUserIdProperty, FilterOperator.EQUAL, userId);
72
73      Query query = new Query(TripKey);
74      query.setFilter(filter);
75
76      PreparedQuery pq = datastore.prepare(query);
77
78      for (Entity result : pq.asIterable()) {
79
80          Trip trip = new Trip((String) result.getProperty(TripNameProperty));
81          trip.setId((int)result.getKey().getId());
82
83          Filter categoryFilter = new FilterPredicate(CategoryTripIdProperty, FilterOperator.EQUAL, result.getKey().getId());
84          Query categoryQuery = new Query(CategoryKey);
85          categoryQuery.setFilter(categoryFilter);
86          PreparedQuery categoryPq = datastore.prepare(categoryQuery);
87
88          for (Entity categoryEntity : categoryPq.asIterable()) {
89              TripCategory category = new TripCategory(
90                      (String) categoryEntity.getProperty(CategoryNameProperty),
91                      (String) categoryEntity.getProperty(CategoryPlacesProperty));
92              trip.addCategory(category);
93          }
94
95          tripList.addTrip(trip);
96      }
97      return tripList;
98  }
```

Figura 5.3: Exemplo de consulta de uma Trip na base de dados.

A Figura 5.3 ilustra como é feita uma consulta no sistema de armazenamento provido pela plataforma. A partir da identificação do usuário é criado um filtro que busca todas as entidades que representam trips de um usuário, e para cada trip são buscadas quais entidades categorias pertencem a aquela trip. Ao final é retornada uma lista com todas as trips do usuário.

5.5 Funcionamento do Aplicativo

Para melhor elucidar o funcionamento do aplicativo serão exibidas nesta subseção as principais telas do aplicativo e suas respectivas funcionalidades. Também serão ilustradas outras características importantes da aplicação.

5.5.1 Tela de Login

A primeira tela do aplicativo tem apenas como funcionalidade fazer a verificação do usuário através de sua conta. Quando o usuário clica no botão de Login a requisição é feita para a API do Facebook, abrindo uma janela para inserção de suas credenciais.

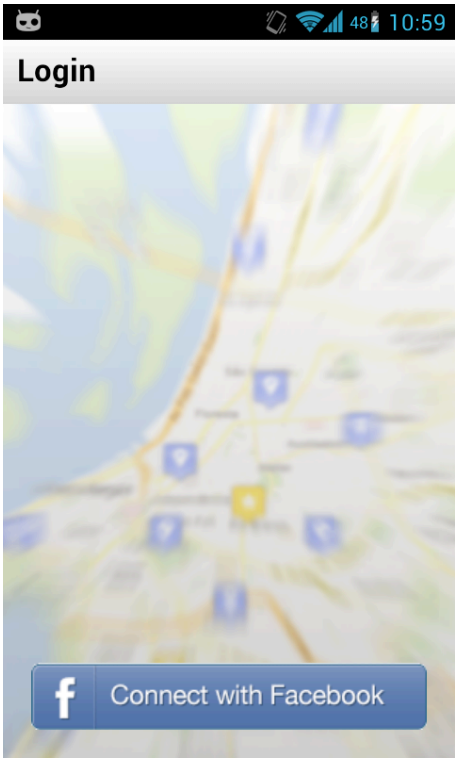


Figura 5.4: Tela de login.

5.5.2 Tela de Mapa

É a tela principal do aplicativo. Logo após o login o usuário é redirecionado para a tela do mapa, onde são exibidas a localização do usuário, os pontos de interesses retornados por um requisição à API do Google Places e os pontos favoritos do usuário recuperados por uma requisição ao servidor implementado.

A partir desta tela é possível navegar para a tela de configurações, para a tela que lista os favoritos e para a tela de detalhes de um ponto de interesse. A navegação é feita através do botão na barra superior, através do clique em um balão do mapa e através do menu que é aberto pelo botão físico dos dispositivos Android.

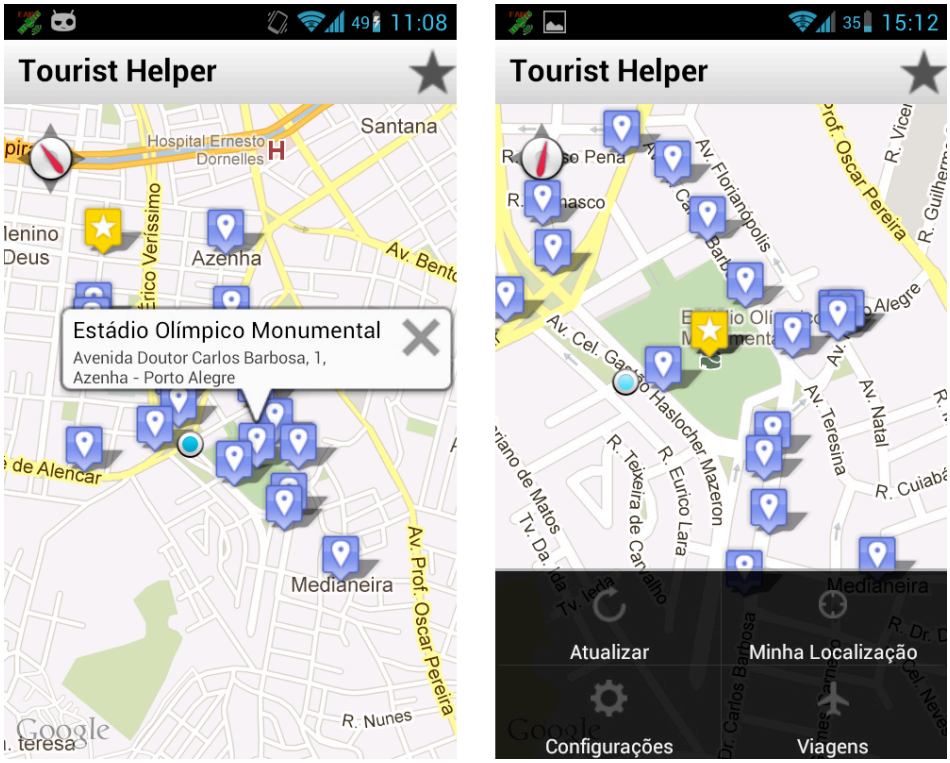


Figura 5.5: Tela de mapa.

5.5.3 Tela de Detalhes

Nesta tela são apresentadas mais informações sobre um determinado ponto de interesse. Além das informações disponíveis no balão da tela de mapa, são exibidas a URL do lugar, o telefone e uma nota disponibilizada pela API do Google Places, baseada na opinião dos usuários.

É nesta tela que o usuário adiciona um ponto na sua lista de favoritos. Quando o botão é clicado, uma requisição é gerada para o servidor implementado, inserindo na base de dados. Também é possível navegar para a tela que busca os tweets relacionados com o determinado lugar.

Figura 5.6: Tela de detalhes.

5.5.4 Tela de Tweets

A partir do nome do ponto interesse é feita uma busca através da API do Twitter, a qual retorna os tweets encontrados. Essa lista é então exibida, de forma com que o usuário possa ler opiniões sobre o que as pessoas falam sobre aquele lugar.

Figura 5.7: Tela de detalhes.

5.5.5 Telas de Favoritos, Trips e Categorias

A tela de favoritos exibe uma lista com os favoritos adicionados pelo usuário, que é buscada através de uma requisição ao servidor implementado. Nela são expostas algumas informações básicas do ponto de interesse e com um clique pode-se navegar para a tela de mapa mostrando o local no mapa. Nesta tela também é possível navegar para a tela que mostra rotas entre os pontos.

De forma parecida com os favoritos, a tela de trips faz uma requisição buscando as trips do usuário e exibe-as na forma de uma lista. Selecionando uma trip, o usuário pode visualizar as categorias relacionadas, que por sua vez quando selecionada exibe uma lista semelhante à lista de favoritos, a qual o usuário pode criar rotas e visualizar um escolhido ponto no mapa.

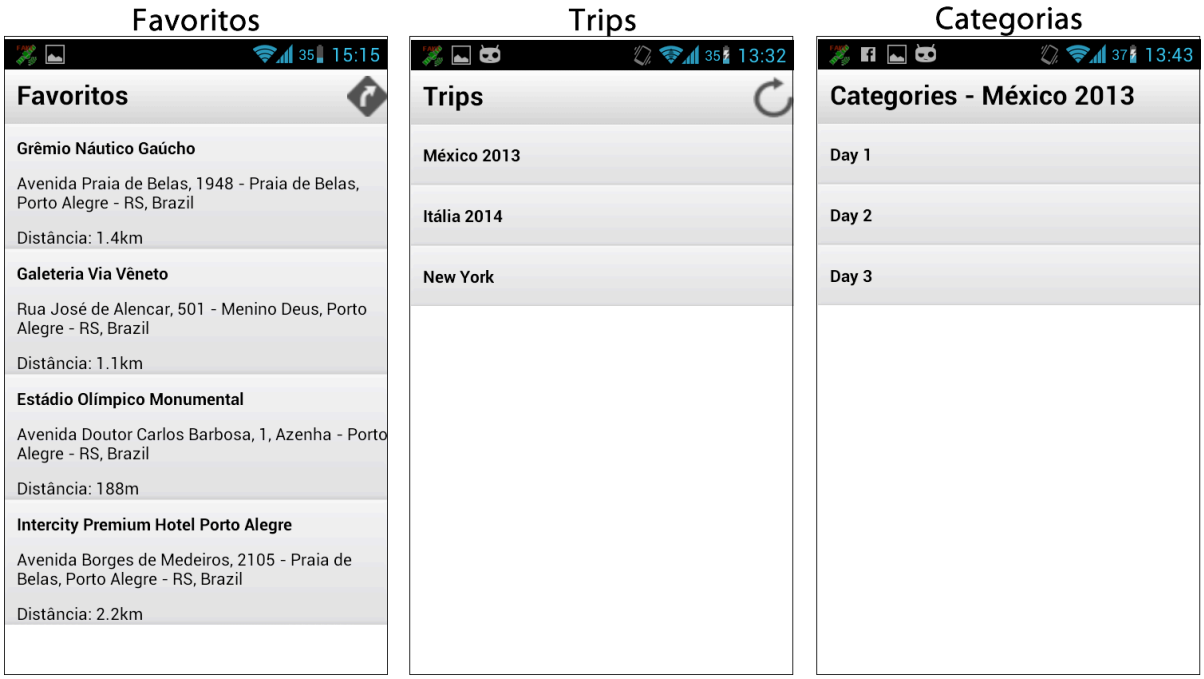


Figura 5.8: Telas de favoritos, trips e categorias.

5.5.6 Tela de Rotas

A tela de rota permite que o usuário visualize rotas entre uma lista de lugares selecionados a partir da tela de favoritos ou de uma categoria pertencente a uma trip. A rota é criada através de chamadas para o serviço web do Google Maps, que retorna uma rota entre dois determinados pontos.

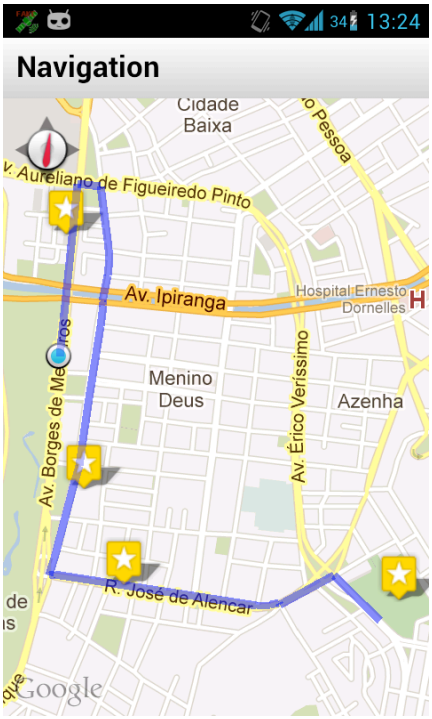


Figura 5.9: Tela de rotas

Para criar uma rota entre diversos pontos foi implementado um algoritmo simples que busca, tendo como ponto inicial a localização do usuário, qual o ponto mais próximo e gera uma rota parcial entre eles. A partir do ponto que era o destino da primeira rota é feita uma busca para o ponto mais próximo, sendo gerada uma nova rota parcial entre estes dois pontos. A iteração é repetida enquanto houver pontos na lista de pontos de interesse. Ao final, com a soma das rotas parciais, é obtida a rota completa que então é desenhada no mapa. A Figura 5.10 ilustra a implementação da criação da rota.

```
307  private void getRoute(ArrayList<Place> places) {
308      //Initialize geoPoints List
309      geoPoints = new ArrayList<GeoPoint>();
310      //Initialize a list with all the places
311      ArrayList<Place> placesList = new ArrayList<Place>(places);
312
313      GeoPoint lastGeoPoint = myLocationOverlay.getMyLocation();
314      int index;
315      Place place;
316      PlaceRequest placeRequest = new PlaceRequest();
317
318      //Initialize the startAddress and destinyAddress
319      String startAddress = lastGeoPoint.getLatitudeE6() / 1E6 + "," + lastGeoPoint.getLongitudeE6() / 1E6;
320      String destinyAddress = "";
321
322      while (!placesList.isEmpty()){
323          //Get the index in the list of the nearest point  of given geoPoint
324          index = getIndexOfNearestPlaceFromGeoPoint(placesList, lastGeoPoint);
325          place = placesList.get(index);
326          //Initialize the destinyAddress with the nearest point relative to StartAddress
327          destinyAddress = place.getLatitudeE6() / 1E6 + "," + place.getLongitudeE6() / 1E6;
328          //Update lastGeoPoint
329          lastGeoPoint = new GeoPoint(place.getLatitudeE6(), place.getLongitudeE6());
330          placesList.remove(index);
331          //Call the webservice that return the route geopoints and add them to geoPoints List
332          geoPoints.addAll(placeRequest.getGeoPointsForMapsUrl("http://maps.google.com/maps?saddr=" +
333                  startAddress + "&daddr=" + destinyAddress + "&ie=UTF8&0&om=0&dirflg=w&output=dragdir"));
334          //The destinyAddress is the startAddress of the last route
335          startAddress = destinyAddress;
336      }
337  }
```

Figura 5.10: Implementação da geração de rotas.

5.5.7 Serviço de Notificações

Como exposto na seção 3.1.3, um serviço Android é uma tarefa que é executada em segundo plano, sem interação direta com o usuário. O aplicativo apresentado neste trabalho possui um serviço que tem como objetivo gerar notificações para o usuário sem que ele explicitamente peça por elas, podendo ser caracterizado como um serviço do tipo push (definido na seção 2.1.5) e diferente das funcionalidades apresentadas nas telas onde o usuário busca informações de forma explícita, caracterizando um serviço do tipo pull.

O foco principal do serviço implementado é gerar recomendações de restaurantes durante um determinado horário do dia (que pode ser escolhido nas configurações). Este objetivo foi definido tendo em vista que, segundo Rieger (2012), 37% dos turistas buscam informações sobre restaurantes através de dispositivos móveis.

Enquanto espera o horário do dia configurado, o algoritmo fica inativo. Esta atitude foi tomada para evitar gastos desnecessários de bateria buscando informações de localização. Quando o horário é alcançado, o algoritmo implementado busca a localização do usuário e faz uma requisição à API do Google Places especificando que apenas deseja resultados que contenham os tipos restaurante ou comida. Após receber a resposta é feita uma filtragem para retirar da lista os estabelecimentos que não constam como abertos no determinado horário. Com a lista filtrada é gerada uma notificação para o usuário informando que foram encontrados estabelecimentos próximos a ele, é apresentada na bandeja de notificações do sistema Android (ver Figura 5.11).

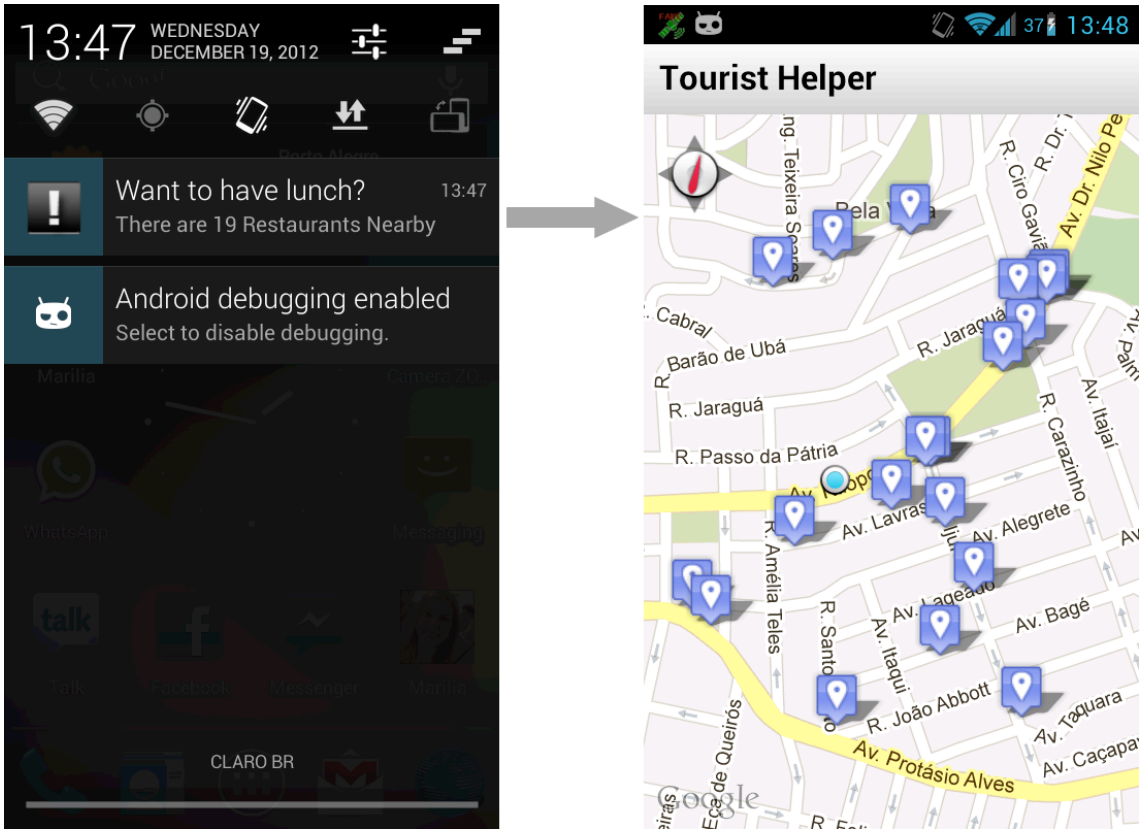


Figura 5.11: Notificação na bandeja do sistema navegando para o mapa com os pontos.

O serviço de notificações inicia sua execução em segundo plano quando o usuário acessa a aplicação pela tela de login, e pode ser desligado através da tela de configurações. É importante destacar que o serviço se mantém em execução mesmo quando o usuário está utilizando outras aplicações ou quando ele está com o dispositivo móvel na tela de descanso.

5.5.8 Internacionalização

Uma forma de atrair mais usuários é adaptar o aplicativo para diferentes línguas. Para implementar a internacionalização em um aplicativo, a plataforma Android fornece um sistema de arquivos em XML em que o desenvolvedor criar as Strings a serem utilizadas na interface com o usuário com chaves únicas. Cada língua a ser suportada possuí um arquivo XML referente a ela, e cada chave única deve conter o valor (palavra ou frase) para a determinada língua. Além disso, o desenvolvedor deve adaptar a interface de usuário para exibir as Strings definidas pelas chaves únicas, ao invés de utilizar Strings fixas no código.

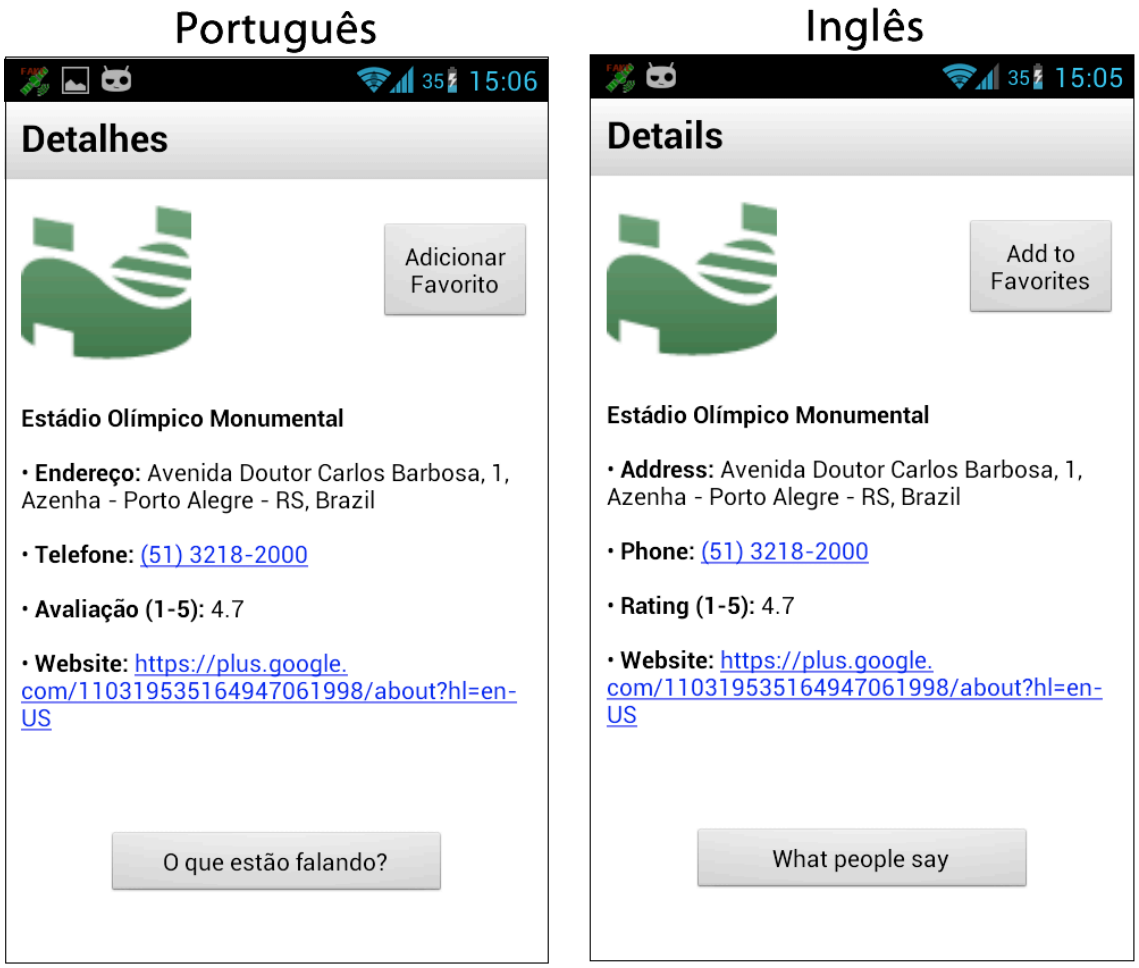


Figura 5.12: Figuras mostrando a internacionalização do sistema.

A implementação deste trabalho possui duas línguas principais: inglês e português. O suporte a novas línguas pode ser feito facilmente, sendo necessário apenas adicionar novos arquivos XML contendo as chaves já criadas com os valores correspondentes a cada língua.

Strings em Inglês

```
<string name="place_details_favorites">Add to Favorites</string>
<string name="place_details_loading">Loading Informations(…)</string>
<string name="place_details_tweets">What people say</string>
<string name="facebook_login_fail">Login Failed</string>
<string name="distance">Distance:</string>
<string name="saved_changes">Changes Saved</string>
<string name="location_not_avaiable">Location Not Avaiable</string>
<string name="restaurants_found">Restaurants found</string>
<string name="restaurants_notification_title">Want to have lunch?</string>
<string name="restaurants_notification_text1">There are</string>
<string name="restaurants_notification_text2">Restaurants Nearby</string>
```

Strings em Português

```
<string name="place_details_favorites">Adicionar Favorito</string>
<string name="place_details_loading">Carregando Informações(…)</string>
<string name="place_details_tweets">O que estão falando?</string>
<string name="facebook_login_fail">Não foi possível efetuar o Login</string>
<string name="distance">Distância:</string>
<string name="saved_changes">Mudanças Salvas</string>
<string name="location_not_avaiable">Localização não disponível</string>
<string name="restaurants_found">Restaurantes Encontrados</string>
<string name="restaurants_notification_title">Quer almoçar?</string>
<string name="restaurants_notification_text1">Existem</string>
<string name="restaurants_notification_text2">Restaurantes Próximos de Você</string>
```

Figura 5.13: Comparação entre os arquivos de XML das diferentes línguas.

5.6 Integração com o Touristy

O Touristy, proposto por Maioli (2013), é uma rede social direcionada a turistas desenvolvida para web. Com um objetivo complementar ao do sistema proposto por este trabalho (auxiliar o turista nas questões dinâmicas da viagem), o Touristy é voltado para o planejamento e compartilhamento de viagens, focando no antes de depois da viagem.

O sistema Touristy permite ao usuário criar um perfil e adicionar amigos com os quais pode trocar mensagens. Com o perfil criado, o usuário pode começar a planejar suas viagens, adicionando pontos de interesse e organizando elas em categorias de acordo com suas preferências. Para auxiliar no planejamento de uma viagem o sistema indica ao usuário pontos de interesse que seus amigos já visitaram em um determinado local, além de permitir a visualização das viagens de seus amigos através do perfil. Após uma viagem, é possível compartilhar fotos e comentários sobre os locais visitados.

Buscando criar uma experiência mais completa para o usuário, os sistemas foram integrados de forma que os dados de uma viagem planejada através do sistema Touristy possam ser visualizados em um dispositivo móvel durante a estadia na cidade de destino. Para permitir essa integração, foi implementado um web service que recebe os dados exportados pelo Touristy e os armazena na base de dados do sistema desenvolvido no Google App Engine (ver seção 5.4.2), ficando acessíveis ao aplicativo Android.

A Figura 5.14 mostra uma tela do sistema Touristy que exibe as trips de um usuário, bem como a foto do perfil e o botão para exportar uma trip para a base de dados do aplicativo deste trabalho. As telas que exibem esses dados no aplicativo Android podem ser visualizadas na Figura 5.8.

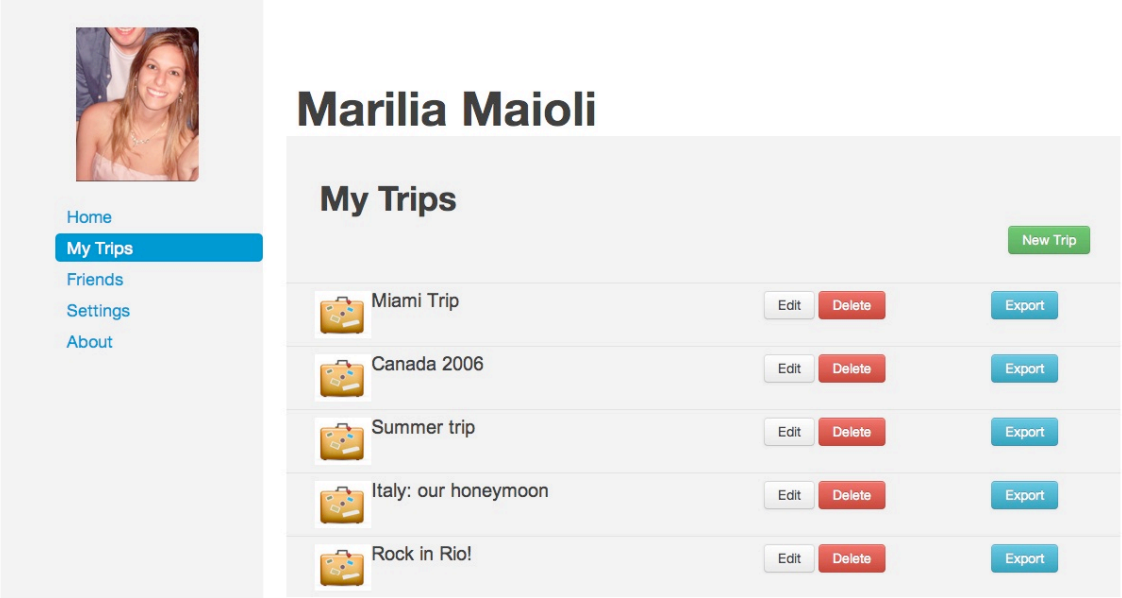


Figura 5.14: Tela exibindo a opção de exportar uma trip no Touristy.

Fonte: Maioli (2013, p.50)

5.7 Aplicativos Semelhantes

Para uma melhor compreensão da área de abrangência do trabalho desenvolvido, serão brevemente apresentados outros aplicativos Android voltados ao turismo. Os aplicativos que serão exibidos foram obtidos através de buscas na Google Play Store, a loja de aplicativos do Android.

5.7.1 Google Places[4]

O Google Places é uma ferramenta de busca de pontos de interesse. Levando em conta a localização do usuário ele lista os locais próximos, incluindo a distância do local atual até o ponto. Ele possibilita ao usuário selecionar os tipos de pontos de interesse a serem localizados (restaurantes, bares, etc.). Também é possível a criação de rotas, mas apenas até um ponto de destino.

[4] Disponível em http://www.google.com/mobile/places/. Acesso em Janeiro de 2013.

5.7.2 My Tourist Guide[5]

É um aplicativo que permite ao usuário fazer buscas por um destino. Depois de selecionado, é feito um download do conteúdo e exibido na tela, listando restaurantes, bares, pontos turísticos, o que se fazer na cidade, entre outras informações. Uma vez baixado o conteúdo de uma cidade o acesso pode ser feito de forma offline. A interface do aplicativo é bastante simples e todas as informações são exibidas em forma de texto, sem o auxílio de um mapa.

5.7.3 Wiki Tourist[6]

Um dos modos deste aplicativo é semelhante ao funcionamento do Google Places: mostra a localização do usuário em um mapa junto com os pontos de interesse próximos. Em outro modo é possível selecionar algum lugar no mapa para ver os pontos próximos à aquele lugar. Não possui opções de filtros e de rotas, e a fonte de dados é a wikiTravel.

5.7.4 WikiTravel Mobile[7]

É um guia de viagens onde as informações são adicionadas pelos usuários. Pode-se selecionar uma cidade através de uma busca e ver informações sobre ela, sugestões de locais para visitação e dicas de viagens. Também não possui um mapa para visualizar os locais.

5.7.5 TripAdvisor[8]

Entre as ferramentas encontradas foi a mais completa, permitindo ao usuário fazer busca por locais e descobrir as atrações disponíveis. Uma das opções permite buscar os pontos de interesse próximos ao local atual, porém, assim como na busca por locais, os resultados são exibidos em forma de lista. O aplicativo é integrado com o site de mesmo nome e possibilita a visualização de locais salvos como favoritos pelo usuário. Também permite ao usuário fazer busca por hotéis e passagens de avião.

5.7.6 Comparação com o Aplicativo Desenvolvido

Com exceção do Google Places, o foco principal dos aplicativos apresentados é o planejamento da viagem através da busca por pontos de interesse. O aplicativo desenvolvido neste trabalho busca uma interação diferente com o usuário, através das

---

[5] Disponível em https://play.google.com/store/apps/details?id=com.mytourguideandroid. Acesso em Janeiro de 2013.

[6] Disponível em https://play.google.com/store/apps/details?id=rayo.app. Acesso em Janeiro de 2013.

[7] Disponível em https://play.google.com/store/apps/details?id=com.exelentia.wikitravel. Acesso em Janeiro de 2013.

[8] Disponível em https://play.google.com/store/apps/details?id=com.tripadvisor.tripadvisor. Acesso em Janeiro de 2013.

notificações em segundo plano, de rotas entre os locais de uma viagem planejada anteriormente no sistema Touristy e nos locais próximos.

6 CONCLUSÃO

Ao longo do trabalho foi enfatizada a participação e importância crescente do mercado de dispositivos móveis no mundo inteiro. Também foi abordada a importância do turismo e as dificuldades que um turista passa em um lugar desconhecido. Com isso em mente foi traçado como objetivo principal do trabalho criar um protótipo de aplicação para gerenciar rotas turísticas de um turista, bem como procurar por pontos de interesse e ajuda-lo na navegação.

A partir do objetivo do trabalho foi desenvolvida uma análise sobre serviços baseados em localização e como esses serviços agregam valor ao usuário, sobre a problemática do turismo e também um estudo sobre as plataformas utilizadas para o desenvolvimento. Sobre as plataformas é importante ressaltar a integração utilizando web services, que foi facilitada pelo fato de ambas utilizarem a mesma linguagem de programação (Java).

De posse desse estudo acima citado, foi modelada uma solução distribuída para possibilitar a implementação do aplicativo em diferentes plataformas utilizando a mesma base de dados. Como exemplo de benefício dessa abordagem pode-se citar a integração com o sistema Touristy que permite ao usuário planejar suas viagens em uma página web e visualizar seus dados em seu dispositivo móvel durante a viagem.

Ao final do trabalho como resultado da implementação, foi obtido um protótipo de sistema Android para gerenciar roteiros turísticos com uma base de dados armazenada na nuvem através da plataforma Google App Engine. Há ainda muito espaço para melhorias e novas funcionalidades, porém foi demonstrado como este protótipo utilizando LBS e plataformas móveis pode facilitar a vida de turistas.

6.1 Trabalhos Futuros

Durante o desenvolvimento deste trabalho foram surgindo novas ideias e novas possibilidades para o aplicativo. A primeira delas é oriunda da arquitetura do sistema, que possibilita a criação da aplicação para outras plataformas móveis, permitindo a um usuário visualizar seus dados em um dispositivo Android e, por exemplo, um tablet Apple Ipad. Também seria possível a criação de uma página web para o planejamento de viagens, que pode ser alcançada por uma maior integração com o sistema Touristy, como, por exemplo, o envio de informações de onde o usuário já esteve e de fotos tiradas nos pontos de interesse.

Outra funcionalidade pensada foi a inclusão de um sistema de recomendação para sugestão de locais de interesse baseados no gosto do usuário. Esse sistema poderia sem incorporado no sistema web desenvolvido no Google App Engine, onde os cálculos

mais pesados poderiam ser feitos sem requerer esforço demasiado do dispositivo móvel e que poderia inclusive levar em conta a rede de amigos do usuário na rede social Touristy. Essas recomendações poderiam ser implementadas como incremento do serviço de recomendação de restaurantes, por exemplo.

REFERÊNCIAS

SOMMERVILLE, Ian. **Software engineering**. 9.ed. Boston: Addison-Wesley, 2009.

NAZZARO, William F., e SUSCHECK, Charles. **New to user stories?.** Disponível em: < http://www.scrumalliance.org/articles/169-new-to-user-stories> Acesso em: Novembro de 2012.

ROCHE, Kyle, DOUGLAS, Jeff. **Beginning Java Google App Engine**. New York: Appress, 2009.

ZAHARIEV, Alexander. **Google App Engine**. Helsink: Helsinki University of Technology, 2009.

SANDERSON, Dan. **Programming Google App Engine**. 2.ed. Sebastopol: O'Rilley, 2012.

GOOGLE. **Android Developers**. Disponível em:

<https://developer.android.com/about/dashboards/index.html>. Acesso em: Novembro de 2012.

COCKBURN, A. **Agile Software Development.** Boston, Addison-Wesley. 2002.

HASHIMI, Sayed Y., Satya KOMATINENI, e Dave MACLEAN. **Pro Android 3**. New York: Appress, 2011.

BURNETTE, Ed. **Hello, Android**. Dallas: Pragmatic Programmers, 2010.

ABLESON, W. Frank, COLLINS, Charlie, e SEN, Robi. **Unlocking Android**. Greenwich: Manning, 2012.

SCHILLER, Jochen, e Agnès VOISARD. **Location Based Services**. San Francisco: Elsevier, 2004.

FERRARO, RICHARD; AKTIHANONOG, MURAT. **Location Based Services Past, Present, and Future.** Green paper from Location Based Services Navigating through the Mobile Jungle, janeiro, 2010.

MAIOLI, Marília. **Touristy: Um Sistema Web Colaborativo para Planejamento e Compartilhamento de Viagens**. 2013, 55f. Trabalho Individual (Graduação em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

BROWN, Barry e CHALMERS, Mathew. **Tourism and Mobile Technology**. University of Glasgow, Glasgow, 2012.

FLING, Brian. **Mobile Design and Development**. Sebastopol: O'Reilly, 2009.

MINISTÉRIO DO TURISMO. **Documento Referencial Turismo no Brasil 2011-2014.** Disponível em:

<http://www.dadosefatos.turismo.gov.br/dadosefatos/outros_estudos/Documento_referencial/> Acesso em: Dezembro de 2012.

IDC. **Press Release**. Disponível em: <http://www.idc.com/getdoc.jsp?containerId=prUS23638712> Acesso em: Novembro de 2012.

FLURRY. **iOS and Android Adoption Explodes Internationally.** Disponível em: <http://blog.flurry.com/bid/88867/iOS-and-Android-Adoption-Explodes-Internationally> Acesso em: Novembro de 2012.

TWITTER. **Developers**. Disponível em: <https://dev.twitter.com/docs/api/1.1> Acesso em: Novembro de 2012.

GOOGLE. **Developers**. Disponível em: <https://developers.google.com/places/documentation/s> Acesso em: Novembro de 2012.

FACEBOOK. **Developers**. Disponível em:

<http://developers.facebook.com/docs/howtos/androidsdk/3.0/login-with-facebook/> Acesso em: Novembro de 2012.

RIEGER, Stephanie. **Technology Solutions for Tourism.** Edinburgh, 2012. Disponível em: <http://www.slideshare.net/yiibu/the-power-of-mobile-web-for-tourism>