

**UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ  
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA  
CURSO DE ESPECIALIZAÇÃO EM TECNOLOGIA JAVA E  
DESENVOLVIMENTO PARA DISPOSITIVOS MÓVEIS**

**RODRIGO FAGUNDES EGGEA**

**APLICAÇÃO ANDROID UTILIZANDO SISTEMA DE LOCALIZAÇÃO  
GEOGRÁFICA PARA DETERMINAÇÃO DE PONTOS TURÍSTICOS  
NA CIDADE DE CURITIBA**

**MONOGRAFIA DE ESPECIALIZAÇÃO**

**CURITIBA**

**2013**

**RODRIGO FAGUNDES EGGEA**

**APLICAÇÃO ANDROID UTILIZANDO SISTEMA DE LOCALIZAÇÃO  
GEOGRÁFICA PARA DETERMINAÇÃO DE PONTOS TURÍSTICOS  
NA CIDADE DE CURITIBA**

Monografia apresentada como requisito parcial à obtenção do título de Especialista em Tecnologia Java e Desenvolvimento para Dispositivos Móveis, do Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Paulo Santos Zeferino

**CURITIBA**

**2013**

## RESUMO

O Brasil sediará a Copa do Mundo em 2014 e as Olimpíadas em 2016, sendo previsto a visita de milhões de estrangeiros em diversas cidades do país, inclusive na cidade de Curitiba, que sediará jogos da Copa do Mundo. O aumento de turistas na cidade promoverá o turismo regional e a visitação aos pontos turísticos da cidade. A dificuldade na localização dos pontos turísticos pode ser um problema para os turistas, devido à falta de informações, sinalizações e pela falta de profissionais no setor hoteleiro, turístico e de serviços para atender a demanda. O uso de aplicações para dispositivos móveis vai de encontro a esta necessidade, permitindo o desenvolvimento de inúmeras aplicações turísticas. Neste contexto, este trabalho apresenta o estudo e o desenvolvimento de uma aplicação para dispositivos móveis com sistema operacional Android, que serve como um guia virtual da cidade de Curitiba. Por meio do aplicativo o turista pode se localizar no mapa, permite buscar rapidamente os pontos turísticos, ver o itinerário da linha turismo, seus pontos de parada e obter informações detalhadas de cada ponto turístico. Este aplicativo utiliza sistemas de localização geográfica, mapas, comunicação de dados, comunicação cliente-servidor e banco de dados.

**Palavras-chave:** Pontos turísticos. Linha de turismo. Curitiba. Google Maps. Android.

## ABSTRACT

The next years Brazil will host the World Cup in 2014 and the Olympic Games in 2016, and is expected to visit millions of foreigners in several cities, including the city of Curitiba, which will host matches of the World Cup. The increase of tourists in the city will promote regional tourism and visitation to the city's sights. The difficulty in locating the sights can be a problem for tourists due to lack of information, traffic signs and lack of professionals in the hospitality industry, tourism and services tend to demand. The use of mobile applications meets this need, allowing the development of numerous tourism applications. In this context, this paper presents the study and development of an application for mobile devices with Android operating system, which will serve as a virtual guide of the city of Curitiba. The application allow the tourist to locate your own position on the map, can quickly locate the sights, see the tourism line route, the bus stops and detailed information of each tourist spot. This application uses geolocation systems, maps, data communication, client-server communication and database.

**Keywords:** Tourist Attraction. Tourism line. Curitiba. Google Maps. Android.

## LISTA DE ILUSTRAÇÕES

Figura 1 - Os quatro componentes de um LBS .....	14
Figura 2 – Participação no mercado mundial de sistemas operacionais para dispositivos móveis no primeiro trimestre de 2013.....	16
Figura 3 – Arquitetura Android .....	17
Figura 4 – Aplicação utilizando Google Maps API no Android .....	20
Figura 5 – Interação entre os elementos da arquitetura de Web Services .....	22
Figura 6 – Estrutura da mensagem SOAP .....	23
Figura 7 – Exemplo de informações de livros armazenados em formato JSON .....	24
Figura 8 – Arquitetura Cliente Servidor .....	25
Figura 9 – Modelo Cliente-Cache-Servidor .....	25
Figura 10 – Modelo Cliente-Servidor sem estado de conexão .....	26
Figura 11 – Aplicativo de guia turístico de Manaus .....	27
Figura 12 – Aplicativo “Curta Curitiba” .....	28
Figura 13 – Aplicativo Mob.urb baixado do Google Play .....	29
Figura 14 – Modelo cliente-servidor .....	32
Figura 15 – Protótipo do aplicativo em <i>Evolus Pencil</i> .....	33
Figura 16 – Diagrama de Caso de Uso .....	34
Figura 17 – Diagrama de Classes .....	35
Figura 18 – Diagrama de Sequencia de <i>Login</i> .....	36
Figura 19 – Diagrama de Sequencia de acesso ao Banco de Dados .....	36
Figura 20 – Utilização do LagLng no Google Maps.....	37
Figura 21 – Itinerário da Linha de Turismo de Curitiba .....	38
Figura 22 – Edição dos ícones com o programa GIMP .....	39
Figura 23 – Modelo do Banco de Dados .....	39
Figura 24 – Implementação de um serviço web RESTful.....	41
Figura 25 – Exemplo de conversão de dados para JSON .....	42
Figura 26 – Resposta do servidor web no formato JSON .....	42
Figura 27 – Trecho do código do servlet de autenticação .....	43
Figura 28 – Trecho do código da Activity de autenticação .....	45
Figura 29 – Código XML da Interface de Mapa.....	45
Figura 30 – Trecho do código da Activity do Mapa .....	46
Figura 31 – Trecho do código que insere os marcadores no mapa .....	47
Figura 32 – Código XML do menu de configurações .....	47
Figura 33 – Tela de abertura do aplicativo Curitour .....	48
Figura 34 – Tela de autenticação de usuário .....	49
Figura 35 – Tela do aplicativo com todas camadas habilitadas .....	49
Figura 36 – Balão de informação após o marcador ser clicado.....	50
Figura 37 – Tela de informações detalhadas do ponto turístico .....	50
Figura 38 – Tela apresentando as opções do menu .....	51

Figura 39 – Tela de seleção de camadas .....	51
Figura 40 – Mapa com o percurso da linha de turismo .....	52
Figura 41 – Lista de seleção de ponto turístico .....	52

## LISTA DE SIGLAS

AJAX	<i>Asynchronous Javascript and XML</i>
API	<i>Application Programming Interface</i>
AVD	<i>Android Virtual Device</i>
CORBA	<i>Common Object Request Broker Architetura</i>
DAO	<i>Data Access Object</i>
DCOM	<i>Distributed Component Object Model</i>
GPS	<i>Global Positioning System</i>
HTTP	<i>Hypertext Transfer Protocol</i>
HTTPS	<i>Hypertext Transfer Protocol Secure</i>
JAX-RS	<i>Java API for RESTful Services</i>
JSON	<i>JavaScript Object Notation</i>
LBS	<i>Location-Based Services</i>
MIME	<i>Multipurpose Internet Mail Extensions</i>
OHA	<i>Open Handset Alliance</i>
PMC	<i>Prefeitura Municipal de Curitiba</i>
POJO	<i>Plain Old Java Objects</i>
REST	<i>Representational State Transfer</i>
RMI	<i>Remote Method Invocation</i>
RPC	<i>Remote Procedure Call</i>
SDK	<i>Software Development Kit</i>
SMS	<i>Short Message Service</i>
SOAP	<i>Simple object Access Protocol</i>
UML	<i>Unified Modeling Language</i>
URI	<i>Uniform Resource Identifier</i>
URL	<i>Uniform Resource Locator</i>
VM	<i>Virtual Machine</i>
W3C	<i>World Wide Web Consortium</i>
WSDL	<i>Web Service Description Language</i>
WWW	<i>World Wide Web</i>
XML	<i>eXtensible Markup Language</i>

## SUMÁRIO

<b>1 INTRODUÇÃO .....</b>	<b>10</b>
1.1 OBJETIVO GERAL .....	11
1.2 OBJETIVO ESPECÍFICO.....	11
1.3 JUSTIFICATIVA.....	12
1.4 ESTRUTURA DO TRABALHO .....	12
<b>2 ESTUDO DAS TECNOLOGIAS .....</b>	<b>13</b>
2.1 SERVIÇOS BASEADOS EM LOCALIZAÇÃO .....	13
2.1.1 Dispositivo Móvel ( <i>Mobile device</i> ).....	14
2.1.2 Provedor de Conteúdo ( <i>Content Provider</i> ).....	14
2.1.3 Rede de Comunicação ( <i>Communication Network</i> ) .....	15
2.1.4 Componente de localização ( <i>Positioning technology</i> ).....	15
2.2 PLATAFORMA ANDROID .....	16
2.3 ARQUITETURA ANDROID.....	17
2.4 GOOGLE MAPS API.....	19
2.5 LOCALIZAÇÃO E MAPAS NO ANDROID .....	20
2.6 WEB SERVICES.....	21
2.7 PROTOCOLO SOAP .....	22
2.8 PROTOCOLO REST.....	23
<b>3 APLICATIVOS TURÍSTICOS BRASILEIROS.....</b>	<b>27</b>
3.1 APLICATIVO REVIVA MANAUS .....	27
3.2 APLICATIVO CURTA CURITIBA.....	28
3.3 APLICATIVO MOB.URB .....	28
3.4 CONCLUSÃO DO CAPÍTULO .....	29
<b>4 DESENVOLVIMENTO.....</b>	<b>30</b>
4.1 LEVANTAMENTO DOS REQUISITOS .....	30
4.2 DESCRIÇÃO DO PROTÓTIPO .....	32
4.3 MODELAGEM.....	33
4.3.1 Diagrama de Caso de Uso.....	34
4.3.2 Diagrama de Classes.....	35
4.3.3 Diagrama de Sequência .....	36
4.3.4 Coordenadas Geográficas dos marcadores .....	37
4.3.5 Desenvolvimento dos Marcadores.....	38
4.4 IMPLEMENTAÇÃO .....	39
4.4.1 Banco de Dados .....	39
4.4.2 Servidor .....	40
4.4.3 Aplicativo Android .....	43
4.4.3.1 Biblioteca Google Maps API Android .....	43
4.4.3.2 Activities.....	44



4.4.4 Telas do aplicativo .....	48
<b>5 CONCLUSÃO.....</b>	<b>53</b>
<b>6 TRABALHOS FUTUROS.....</b>	<b>54</b>
<b>7 REFERÊNCIAS .....</b>	<b>55</b>

## 1 INTRODUÇÃO

O Brasil nos próximos anos sediará grandes eventos, como a Copa do Mundo em 2014 e as Olimpíadas em 2016. Sediar megaeventos como a Copa o Mundo e as Olimpíadas representam muito mais que eventos esportivos, mas uma grande oportunidade de negócios, que influenciam a socioeconomia dos lugares (MORGAN; SUMMERS, 2008).

Uma pesquisa realizada pela Fundação Getúlio Vargas, a pedido do Ministério do Turismo em 2009, estimou que 5,9 milhões de estrangeiros visitariam o Brasil entre os anos de 2009 a 2014, sendo que, durante a realização do torneio, 500 mil turistas são esperados e devem permanecer no país, em média 15 dias, gastando aproximadamente dez mil reais, totalizando uma receita de 6,27 bilhões de reais (NEVES et al., 2011).

Em vista de todos esses eventos e o fato da cidade de Curitiba ter conquistado o direito a ser uma das 12 cidades que terá a oportunidade de sediar os jogos da Copa do Mundo de 2014, atrairá milhares de turistas nos próximos anos. A dificuldade na localização dos principais pontos turísticos poderá ser um problema para os turistas que estarão visitando a cidade de Curitiba pela primeira vez, devido à falta de informações, placas, sinalizações e pela falta de profissionais com conhecimento na cidade, seja no setor hoteleiro, turístico e de serviços para atender a demanda.

Os telefones celulares já contam com mais de 4 bilhões de usuários no mundo e ganharam status de ferramenta indispensável. Por outro lado, a venda de *smartphones* tem superado a de *laptops*, o que indica que aqueles devem ganhar espaço como computador pessoal nos próximos anos (LECHETA, 2010).

A utilização da tecnologia de redes de dados sem fio e de celulares *smartphones* será intenso durante os próximos anos, devido à facilidade em tirar fotos, gravar vídeos, realizar compartilhamento das informações em redes sociais e assistir os jogos em tempo real (EMBRATUR, 2010).

## 1.1 OBJETIVO GERAL

O objetivo geral é o desenvolvimento de uma aplicação para dispositivos móveis que servirá como um guia turístico virtual para localizar facilmente a posição atual do dispositivo e dos pontos turísticos da cidade de Curitiba.

O aplicativo mostrará o mapa da cidade com os principais pontos turísticos da cidade como Ópera de Arame, Jardim Botânico, Passeio Público, entre outros de forma simples, clara e objetiva. O turista ao clicar no ponto turístico receberá informações relevantes sobre o local. Também será possível realizar uma busca rápida de pontos turísticos através de uma lista e poderá visualizar o itinerário da linha turismo e seus respectivos pontos de parada. Inicialmente o aplicativo estará disponível em português podendo ser estendido a outros idiomas.

## 1.2 OBJETIVO ESPECÍFICO

- Analisar as tecnologias necessárias para desenvolver o aplicativo;
- Estudar o estado da arte de aplicativos turísticos brasileiros;
- Estudar a biblioteca do *Google Maps* para Android;
- Analisar os requisitos do aplicativo;
- Modelar o aplicativo para dispositivo móvel;
- Modelar a base de dados;
- Pesquisar e inserir as informações na base de dados;
- Desenvolver o serviço *web* que realizará a comunicação com o banco de dados e será consumido pelo aplicativo Android;
- Desenvolver a aplicação para dispositivos móveis com sistema operacional Android como *tablets* e *smartphones* que possuam GPS e conexão de dados.

### 1.3 JUSTIFICATIVA

A justificativa deste trabalho é aproveitar a demanda por produtos e serviços que ocorrerá em todos os setores no Brasil nos próximos anos, devido ao aumento do turismo, e desenvolver um aplicativo que auxilie os turistas a conhecerem melhor a cidade, mesmo na ausência de um guia turístico.

O trabalho também busca mostrar a importância no desenvolvimento de aplicativos regionais, focados a uma região específica, permitindo explorar e valorizar os aspectos culturais da região.

### 1.4 ESTRUTURA DO TRABALHO

Inicialmente no capítulo 2 será realizado um estudo do estado da arte das tecnologias atualmente envolvidas no desenvolvimento de aplicativos para Android baseados em localização.

No capítulo 3 são mostrados alguns aplicativos, semelhantes ao proposto neste trabalho, voltados para o setor turístico no Brasil e suas funcionalidades.

No capítulo 4 é realizada a descrição do protótipo do aplicativo, os requisitos do sistema, sua modelagem e a estrutura do banco de dados. Após essa etapa é descrito a sua implementação no lado servidor e no cliente Android.

Por fim, o capítulo 5 apresenta a conclusão geral do trabalho, contendo a análise da aplicabilidade do aplicativo, pontos positivos e negativos, bem como as considerações finais e sugestões de implementações futuras.

## 2 ESTUDO DAS TECNOLOGIAS

Este capítulo apresenta um estudo das tecnologias empregadas no desenvolvimento de aplicativos móveis que utilizam serviços baseados em localização (*location based services*), estudo da arquitetura Android, sua biblioteca de localização e mapas (Google Maps), estudo da arquitetura de servidores *web* e dos protocolos de comunicação cliente-servidor SOAP e REST.

### 2.1 SERVIÇOS BASEADOS EM LOCALIZAÇÃO

A definição típica de serviços baseados em localização *são para* serviços de informação, acessíveis por dispositivos móveis através das redes de dados fazendo uso da posição geográfica do dispositivo.

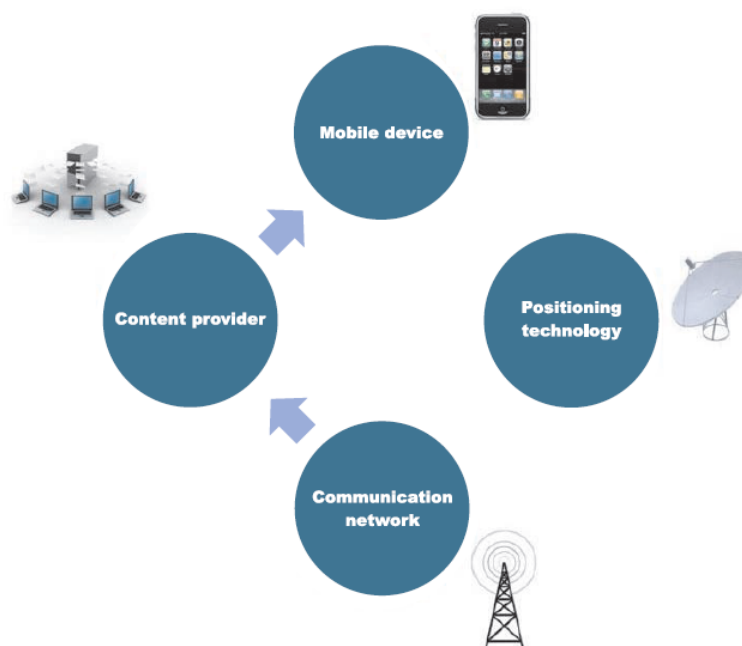
Segundo Ferraro e Aktihanoglu (2011), esta definição está desatualizada na atual geração de dispositivos móveis e *web services*. Atualmente a interação do usuário e a possibilidade de gerar conteúdo são fundamentais para os serviços e aplicações. Uma melhor definição atual de serviço baseado em localização é um serviço que:

- O usuário é capaz de determinar sua localização;
- As informações fornecidas são espacialmente relacionadas com a localização do usuário;
- É oferecido ao usuário uma interação dinâmica ou bidirecional com o provedor de conteúdo.

Desta maneira o usuário pode responder às três questões:

- Onde estou?
- O que posso fazer por perto?
- O que acho sobre este lugar?

São quatro os componentes comuns a todas as aplicações LBS: dispositivo móvel, provedor de conteúdo, rede de dados e um componente de localização, conforme mostrado na Figura 1.



**Figura 1 - Os quatro componentes de um LBS**  
**Fonte: Ferraro e Aktihanoglu (2011)**

### 2.1.1 Dispositivo Móvel (*Mobile device*)

Os *smartphones* tiveram uma importância na rápida absorção de serviços LBS, dado seu tamanho de tela, geralmente grandes e a inclusão quase que universal de tecnologias de localização (como GPS). O rápido crescimento dos *netbooks* e dos *tablets* também fizeram os desenvolvedores começarem a pensar em dispositivos móveis muito mais do que apenas telefones celulares.

### 2.1.2 Provedor de Conteúdo (*Content Provider*)

Um provedor de conteúdo é uma entidade que cria ou detém o conteúdo que pode ser fornecida aos dispositivos móveis, diretamente ou através de terceiros.

Um provedor LBS não armazena ou mantém todo o conteúdo e dados que são acessados pelo usuário no dispositivo móvel. Um exemplo são os dados de mapas, que normalmente são fornecidos por um provedor de mapas, como por exemplo pela empresa NAVTEQ. Cada vez mais os dados são disponibilizados aos usuários na forma de camadas de mapas, através de provedores de terceiros, que

normalmente podem ser ligados ou desligados pelo usuário (mostrando apenas os postos de gasolina em um mapa, por exemplo).

### 2.1.3 Rede de Comunicação (*Communication Network*)

O desenvolvedor do aplicativo LBS não tem controle direto sobre a rede de comunicação, mas pode gerenciar o tráfego dos dados utilizado pelo aplicativo, maximizando a velocidade de transferência ou minimizando a latência, bem como limitando as taxas de dados para clientes pré-pagos.

### 2.1.4 Componente de localização (*Positioning technology*)

O componente de localização refere-se à tecnologia que permite a localização do aparelho e tem a capacidade de repassar essa informação para a aplicação. Esse componente está se tornando cada vez mais escondido, até mesmo para os desenvolvedores dos aplicativos, mas ainda há um grau de escolha entre os métodos de localização, que podem ser por triangulação, Cell ID, navegação por satélite ou por *Wireless Positioning System* (WPS). Nos dispositivos que possuem mais de uma tecnologia de localização disponíveis, a utilização híbrida de posicionamento vem se tornando cada vez mais utilizada para minimizar as desvantagens do uso de uma única tecnologia.

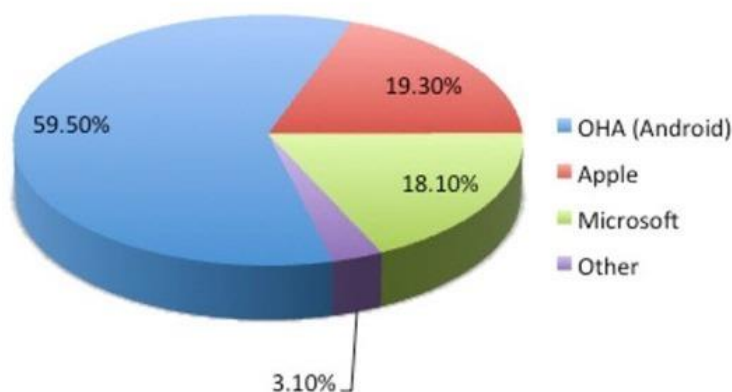
Está se tornando comum a capacidade de determinar a localização através de APIs ou componentes de software para pelo menos estabelecer uma localização aproximada. Isso é cada vez mais utilizado por navegadores de celular para, por exemplo, ser capaz de oferecer resultados de busca de imóveis restritas à área em que o dispositivo está localizado.

## 2.2 PLATAFORMA ANDROID

Atualmente há 1,5 bilhões de televisões em todo o mundo, um bilhão de pessoas com acesso a Internet e quase 3 bilhões de pessoas possuem um telefone celular, isso corresponde a mais ou menos metade da população mundial, sendo considerado um dos mais bem sucedidos produtos de consumo (OHA, 2013).

Android é uma nova plataforma de desenvolvimento para aplicativos móveis como smartphones baseados no sistema operacional *Linux* que possui uma interface visual rica, GPS, diversas aplicações pré-instaladas e um ambiente de desenvolvimento bastante poderoso e que utiliza a linguagem de programação Java. O Android foi desenvolvido pela *Open Handset Alliance* (OHA), liderada pelo Google e empresas como HTC, LG, Motorola, Samsung, Sony Ericsson, Toshiba e muitas outras, que tiveram por objetivo padronizar uma plataforma de código aberto e livre para celulares, para atender as necessidades do mercado atual (LECHETTA, 2010).

A plataforma Android atualmente é o principal sistema operacional para dispositivos móveis, seguido pelo iOS da Apple, conforme estudo de mercado produzido pela empresa Canalys (CANALYS, 2013). Este estudo mostra que durante o primeiro trimestre de 2013, a plataforma Android representa cerca de 59,5% dos smartphones vendidos em todo o mundo, a Apple, com o iOS, segue a plataforma da Google com cerca de 19,3% do mercado e por fim a Microsoft com 18,1%, conforme a Figura 2.

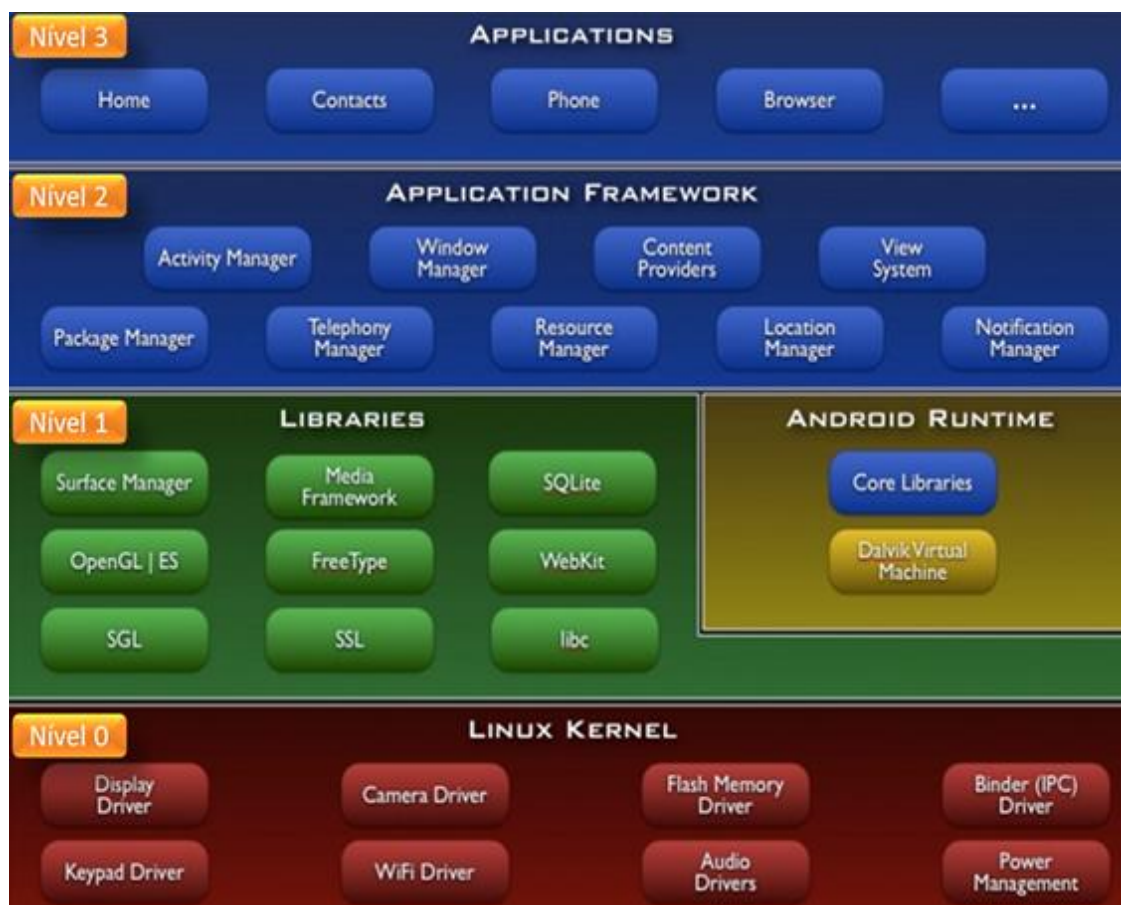


**Figura 2 – Participação no mercado mundial de sistemas operacionais para dispositivos móveis no primeiro trimestre de 2013**  
Fonte: Canalys (2013)



## 2.3 ARQUITETURA ANDROID

A arquitetura da plataforma Android é dividida em várias camadas: *Applications*, *Application Framework*, *Libraries*, *Android Runtime* e *Linux Kernel* (RABELLO, 2009). A Figura 3 apresenta cada uma dessas camadas.



**Figura 3 – Arquitetura Android**  
**Fonte: Rabello (2009)**

Na camada *Applications*, está localizada uma lista de aplicações padrões como cliente de e-mail, programa de SMS, calendário, mapas, navegador, gerenciador de contatos entre outros, todos escritos na linguagem Java.

Já na camada *Application Framework* estão os componentes que permitem com que novas estruturas sejam utilizadas para futuras aplicações, focando a reutilização de código. Fazem parte dessa camada os seguintes componentes:

- Conjunto de componentes gráficos que pode ser reutilizados para construir uma aplicação, bem como listas, *grids*, caixas de texto, botões e até um navegador *web*;
- Provedores de conteúdo que habilitam às aplicações acessarem dados de outras aplicações ou compartilhar seus próprios conteúdos;
- Gerenciar os recursos que prove acesso a recursos não-codificados como *strings*, gráficos, e arquivos de *layout*;
- Gerenciador de notificações que permite que todas as aplicações exibam mensagens de alerta personalizáveis na barra de status;
- Um Gerenciador de atividades que controlar o ciclo de vida das aplicações, liberando memória dos recursos que não estejam mais sendo utilizados.

A camada logo abaixo é subdivida no grupo das bibliotecas (*libraries*) e o ambiente de execução (*runtime*) da plataforma Android, composto pelas bibliotecas padrão e pela máquina virtual denominada *Dalvik*. No primeiro grupo estão as bibliotecas escritas em C/C++, que são compostas por uma coleção de bibliotecas que são utilizadas pela plataforma Android. Estas bibliotecas são:

- Biblioteca de sistema C: é uma implementação da biblioteca C padrão (*libc*), otimizada para dispositivos que suportam a plataforma Linux (*embedded-linux*);
- Bibliotecas de Mídias: as bibliotecas suportam execução e gravação da maioria dos formatos de áudio e vídeo, bem como exibição de imagens, incluindo MPEG4, H.264, MP3, AAC, AMR, JPG, e PNG;
- Gerenciador de Superfície: gerencia o acesso ao display do dispositivo e camadas de gráficos 2D e 3D de múltiplas aplicações;
- LibWebCore: uma moderna *engine* de navegador web que turbina tanto o navegador da plataforma Android e um outro navegador qualquer desenvolvido;
- SGL: uma *engine* de gráficos 2D;
- 3D *libraries*: uma implementação baseada na especificação OpenGL ES 1.0, a qual utiliza tanto aceleração de *hardware* 3D e um avançado e otimizado *software* para renderização de modelos tridimensionais;
- FreeType: renderização em formatos bitmaps e vetoriais de fontes;

- SQLite: uma poderosa e leve *engine* de banco de dados relacional disponível para todas as aplicações.

No que diz respeito ao ambiente de execução, a plataforma é composta pela máquina virtual *Dalvik*. Toda e qualquer aplicação em Android roda dentro de seu próprio processo, isto é, no contexto da sua instância de máquina virtual. Esta VM foi escrita para que os dispositivos possam suportar múltiplas máquinas virtuais eficientemente. A *Dalvik* executa arquivos no formato *Dalvik Executable*, com extensão *.dex*. Um arquivo *.dex* nada mais é do que uma espécie de *bytecodes* de Java (arquivos compilados em arquivos *.class*) otimizados para a Android.

Na base, está localizado o kernel *Linux*, que para a Android está na versão 2.6, fornecendo serviços do núcleo do sistema como segurança, gerenciamento de memória, gerenciamento de processos, pilhas de redes, etc.

## 2.4 GOOGLE MAPS API

A versão beta do Google Maps foi lançado em fevereiro de 2005, com sua interface inovadora, utilizando recursos do navegador nunca explorados. Esse novo estilo de desenvolvimento web foi dado o nome de Ajax (*A New Approach to Web Applications*). Google Maps se tornou não apenas uma revolução nas aplicações de mapas, mas um modelo para todas as aplicações *web*. Tim O'Reilly (fundador da O'Reilly Media) chamou este novo modelo de "Web 2.0", o que ajudou a definir a diferença entre como as aplicações costumavam ser e o novo modelo "Google Maps" (adaptado de DAVIS, 2006).

Em junho de 2005 a Google lançou a primeira versão de suas APIs, permitindo que os usuários pudessem desenvolver aplicações com mapas. Em abril de 2006 lançaram a versão 2 das APIs, com novos recursos como vários níveis de zoom, controles adicionais aos mapas e a possibilidade de criar várias camadas com imagens personalizadas aos mapas.

Atualmente a API do Google Maps está na versão 3 e foi desenvolvida a partir a do zero, como um conjunto modular de bibliotecas JavaScript focadas em

melhorar a velocidade de carregamento, especialmente para renderizar mapas em navegadores de dispositivos móveis (KATARIA, 2009).

## 2.5 LOCALIZAÇÃO E MAPAS NO ANDROID

Uma das funcionalidades do Android que chamam atenção é a integração com o Google Maps e a possibilidade de desenvolver aplicações de localização com GPS com poucas linhas de código. É possível incluir esses recursos na aplicação utilizando as classes do pacote `android.location` e o Google Maps Android API.

O componente principal do *framework* de localização é o *LocationManager system service*, que provê as APIs para determinar a localização e a direção do dispositivo. Para utilizá-lo, é necessário solicitar uma instância para o sistema, chamando o `getSystemService(Context.LOCATION_SERVICE)` que retorna um *handle* para uma nova instância do *LocationManager*.

Com as APIs de Google Maps para Android, é possível adicionar mapas nas aplicações baseados nos dados do Google Maps, como mostra a Figura 4. A principal classe da API é o *MapView*, que automaticamente acessa os servidores do Google Maps, baixa os dados, mostra os mapas e trata os eventos de toque no mapa. Também é possível utilizar a API para adicionar marcadores, polígonos, camadas e alterar a visualização do usuário de uma área específica (MAPS, 2013).

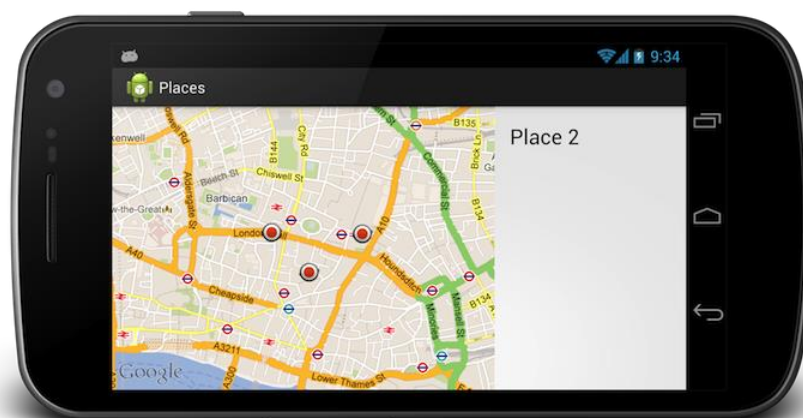


Figura 4 – Aplicação utilizando Google Maps API no Android

## 2.6 WEB SERVICES

Inicialmente as aplicações eram centralizadas em ambientes de grande porte, como os *mainframes*. As aplicações distribuídas surgiram com a evolução das redes de computadores e tem se mostrado uma importante técnica para desenvolvimento e integração de sistemas (GOMES, 2010).

Segundo definição do W3C, *Web Services* é um programa desenvolvido para permitir a interação máquina-máquina em uma rede. Ele possui uma interface descrita em um formato compreensível pela máquina (especificamente o WSDL). Outros sistemas interagem com o *Web Service* através de mensagens SOAP (*Simple Object Access Protocol*), normalmente transportados usando HTTP.

O WSDL (*Web Services Description Language*) é descrito no padrão XML, uma linguagem de marcação neutra, baseada em padrões W3C, o que a torna adequada para descrever as interfaces dos *Web Services*, permitindo sua utilização por diversas linguagens de programação. O WSDL define o conteúdo das mensagens e também onde o serviço está disponível e quais os protocolos de comunicação usados para utilizar o serviço, ou seja, define tudo que é necessário para o cliente consumir o serviço (LIMA, 2012).

Segundo Lima (2012), os três principais componentes da arquitetura de *Web Services* são:

- **Provedor de Serviços:** Responsável pela implementação e disponibilização do *Web Service*, em um formato padrão compreensível para qualquer aplicação que precise utilizar esse serviço e também publicar as características sobre o serviço em um registro central disponível.
- **Consumidor de serviços:** Qualquer um que utilize um *Web Service* criado por um provedor de serviços é chamado de consumidor de serviços. O mecanismo para ligação e funcionalidade do *Web Service* pode ser obtida através de uma pesquisa sobre o registro publicado.
- **Registro dos serviços:** Um registro de serviços é a localização central onde o provedor de serviços pode relacionar seus *Web Services*, e no qual um consumidor de serviços pode pesquisá-los. O registro dos serviços contém informações como detalhes de uma empresa, quais os serviços que ela fornece e a descrição técnica de cada um deles.

Conforme a Figura 5, o provedor de serviços define a descrição do serviço para o *Web Service* e publica esta para o consumidor de serviços no registro de serviços. O consumidor de serviços utiliza a descrição do serviço publicada para se ligar ao provedor de serviços e invocar ou interagir com a implementação do *Web Service*.



**Figura 5 – Interação entre os elementos da arquitetura de Web Services**  
Fonte: Lima (2012)

## 2.7 PROTOCOLO SOAP

O *Simple Object Access Protocol* (SOAP) é um protocolo baseado em XML para troca de informações em um sistema distribuído e descentralizado. Apesar do protocolo poder ser utilizado em qualquer protocolo de transporte, o foco inicial do SOAP é para realização de RPC (*Remote Procedure Call*) transportados por HTTP. Outros frameworks como CORBA, DCOM e Java RMI, permitem uma funcionalidade semelhante ao do SOAP, mas como as mensagens SOAP são descritas em um padrão XML à torna independente de plataforma e linguagem de programação (TUTORIALSPPOINT, 2013).

O SOAP consiste de três partes:

- Envelope: que descreve o que está na mensagem e como processá-la;
- Encoding Rules: Define regras de codificação de diversos tipos de dados como *integers*, *floats*, *double* ou *arrays*;
- RPC: Define convenção de representação de chamada remota de métodos e suas respostas.

As mensagens SOAP são documentos XML contendo os seguintes elementos:

- Envelope (obrigatório): Define o início e o fim da mensagem
- Header (opcional): Contém atributos opcionais da mensagem utilizadas no processamento da mensagem;
- Body (obrigatório): Contém os dados XML compreendendo a mensagem a ser enviada;
- Fault (opcional): Um elemento de falta que fornece informação sobre erros que ocorreram durante o processamento da mensagem.

A Figura 6 apresenta um exemplo de mensagem SOAP.

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://www.w3.org/2001/12/soap-envelope"
SOAP-ENV:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<SOAP-ENV:Header>
...
...
</SOAP-ENV:Header>
<SOAP-ENV:Body>
...
...
<SOAP-ENV:Fault>
...
...
</SOAP-ENV:Fault>
</SOAP-ENV:Body>
</SOAP_ENV:Envelope>
```

**Figura 6 – Estrutura da mensagem SOAP**  
Fonte: TutorialsPoint (2013)

## 2.8 PROTOCOLO REST

O protocolo *Representational State Transfer* (REST) foi idealizado por Roy Fielding no ano de 2000 em sua dissertação de doutorado, na busca pelas melhores práticas nos estilos de arquiteturas existentes para compor um novo estilo, o qual ficou conhecido por REST. Essa técnica de engenharia de software foi desenvolvida para sistemas hipermídia distribuídos como a *World Wide Web* (RONDON, 2013).

O REST considera cada aplicação web como um conjunto de recursos, que representam um estado particular de um aplicativo. Ao acessar um recurso é transferido o estado (conteúdo), podendo alterar seu estado.

As três principais características do REST são:

- Recursos: Os recursos individuais são identificados na requisição, como descrevendo elas nas URIs,

- Ações: É possível aplicar várias ações sobre um recurso. São oito métodos disponíveis pelo protocolo HTTP, sendo os mais utilizados o PUT (cria ou atualiza o conteúdo do recurso), GET (busca o conteúdo do recurso), DELETE (Apaga conteúdo do recurso).
- Conteúdo: É utilizado por alguns protocolos o MIME TYPES para identificar que tipo de conteúdo esta sendo negociado. A identificação geralmente é feita no cabeçalho pelo campo cujo nome é *Content-type*.

Todo recurso inclui a informação do formato referente ao conteúdo, ou seja, quando por exemplo você receber a informação da mensagem que o conteúdo (content-type) se encontra no formato JSON (application/json) você terá que processar a informação neste formato. A Figura 7 mostra um exemplo de objeto de dados codificado em JSON.

```
{
  "book": [
    {
      "id": "01",
      "language": "Java",
      "edition": "third",
      "author": "Herbert Schildt"
    },
    {
      "id": "07",
      "language": "C++",
      "edition": "second",
      "author": "E.Balagurusamy"
    }
  ]
}
```

**Figura 7 – Exemplo de informações de livros armazenados em formato JSON**  
Fonte: TutorialsPoint (2013)

Normalmente o formato dos arquivos utilizados pelos serviços RESTful são JSON, XML ou YAML (RECKZIEGEL, 2006).



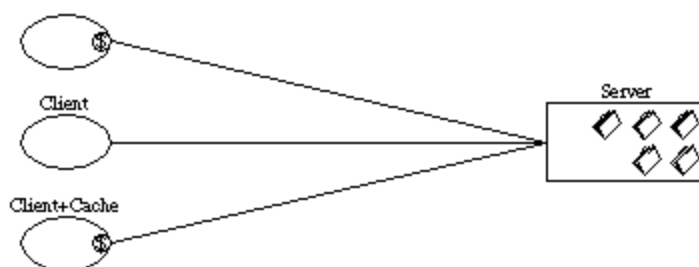
O REST é resultado das melhores práticas dos seguintes estilos:

- Cliente / Servidor: O servidor disponibiliza um conjunto de serviços e o cliente faz uso desses serviços;



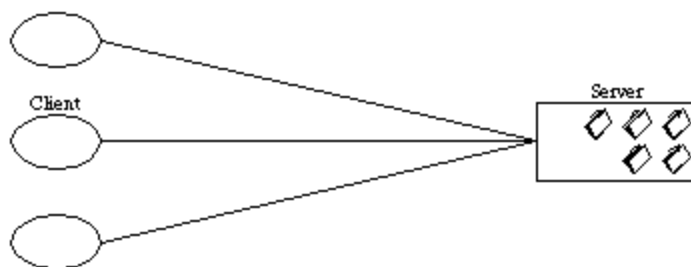
**Figura 8 – Arquitetura Cliente Servidor**  
Fonte: Rondon (2013)

- Sistema em Camadas: Cada camada conhece apenas a interface da camada superior. As camadas intermediárias podem ser utilizadas para melhorar a escalabilidade do sistema, permitindo o balanceamento de carga de serviços, através de múltiplas redes;
- *Cache*: Arquitetura que evita desperdício de banda com a utilização de *Proxy* HTTP, que armazena as páginas solicitadas pelo cliente.



**Figura 9 – Modelo Cliente-Cache-Servidor**  
Fonte: Rondon (2013)

- Sem estado de conexão (*Stateless*): Nessa arquitetura o servidor não armazena nenhuma informação do contexto. Toda informação necessária para atender a uma requisição deve estar contida nela mesma, tornando o serviço mais simples, não precisando levar em consideração o contexto atual para tomar decisões.



**Figura 10 – Modelo Cliente-Servidor sem estado de conexão**  
Fonte: Rondon (2013)

### 3 APLICATIVOS TURÍSTICOS BRASILEIROS

Neste capítulo será abordado o estudo de alguns aplicativos para dispositivos móveis, voltados para o setor turístico brasileiro, que já foram desenvolvidos até o presente momento e que também tem como objetivo auxiliar o turista à localização dos pontos de interesse em uma determinada região.

#### 3.1 APLICATIVO REVIVA MANAUS

O primeiro exemplo é o aplicativo “Reviva Manaus”, desenvolvido pela designer de interface digital Gisely Mendonça, que tendo em vista o grande potencial turístico de Manaus como cidade sede da Copa do Mundo de 2014, desenvolveu um aplicativo que funciona como guia turístico para os principais pontos turísticos de Manaus, devido ao problema da capital na transmissão de informações turísticas e falta de placas informativas bilíngues. Segundo a designer, a maior dificuldade enfrentada durante o desenvolvimento do aplicativo foi a coleta de informações dos pontos turísticos da cidade (MELO, 2013).



Figura 11 – Aplicativo de guia turístico de Manaus  
Fonte: Melo (2013)

### 3.2 APLICATIVO CURTA CURITIBA

O segundo exemplo é o aplicativo “Curta Curitiba”, desenvolvido para *tablets* e *smartphones*, foi criada em parceria do Instituto Municipal de Turismo, Curitiba *Convention & Visitors Bureau*, Associação Brasileira de Bares e Restaurantes do Paraná (ABRASEL-PR) e Sebrae-PR. Segundo Juliana Vosnika, presidente do Instituto Municipal de Turismo, o aplicativo traz informações sobre atrativos obrigatórios, serviços, hospedagem, gastronomia e orientações de locomoção na cidade. Juliana afirma que "Ter uma marca é um diferencial competitivo, uma estratégia de marketing para divulgar as potencialidades do destino. Em 2011, queremos fortalecê-la e internacionalizá-la, já pensando na Copa do Mundo de 2014" (PMC, 2013).



**Figura 12 – Aplicativo “Curta Curitiba”**  
**Fonte: Prefeitura de Curitiba (2013).**

### 3.3 APLICATIVO MOB.URB

Em Minas Gerais, a empresa SQUADRA, lançou no dia 23 de maio de 2013 o aplicativo gratuito para celular chamado de Mob.urb, que lista mais de 10 mil atrações turísticas da capital mineira. O aplicativo está disponível para os aparelhos que operam no sistema Android e iOS (iPhone), nos idiomas inglês, português e

espanhol. O ponto focal do aplicativo é um mapa turístico, similar ao que os turistas adquirem em aeroportos e hotéis, mas também permite a montagem de roteiro, com datas, horários e quais locais serão visitados (SQUADRA, 2013).

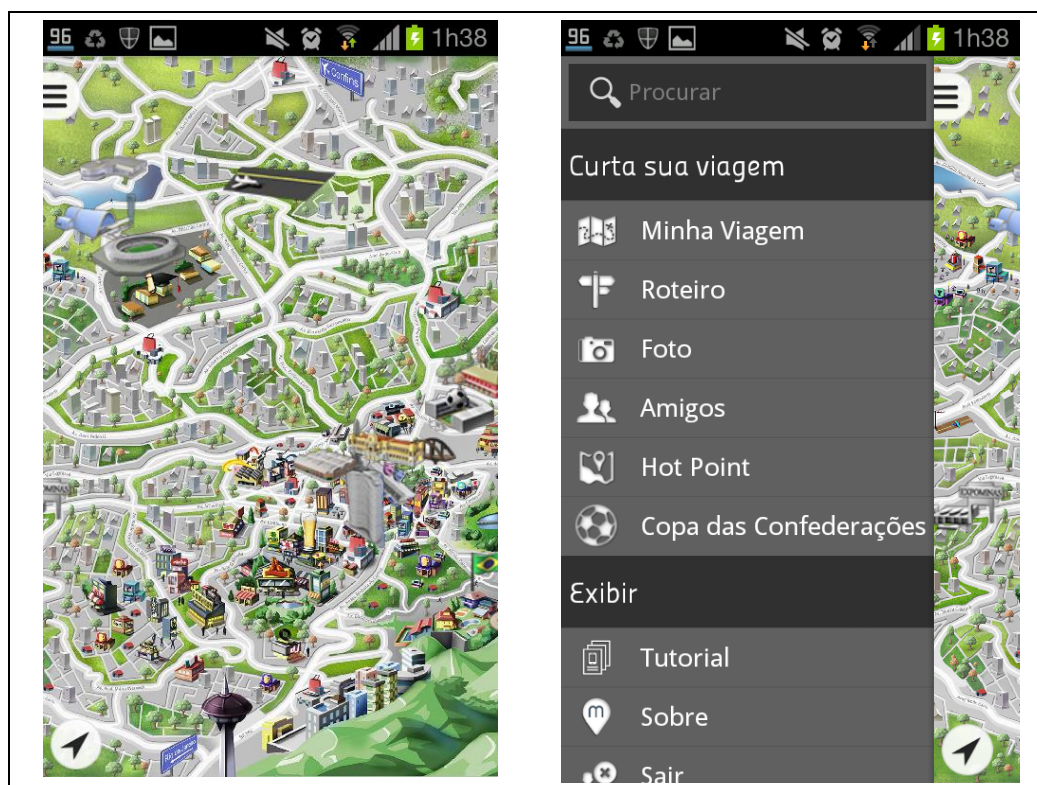


Figura 13 – Aplicativo Mob.urb baixado do Google Play

### 3.4 CONCLUSÃO DO CAPÍTULO

Muitos aplicativos estão sendo desenvolvidos para auxiliar o grande volume de turistas que visitarão o Brasil nos próximos anos, utilizando a tecnologia para suprir essa deficiência em informações e em pessoas do setor turístico em todas as cidades brasileiras. É possível observar uma característica comum destes aplicativos, que é o seu desenvolvimento para uma região específica (um estado ou uma cidade). Provavelmente ainda serão desenvolvidos diversos aplicativos turísticos, até mesmo para as mesmas regiões, o que na verdade é um aspecto positivo, pois o usuário pode utilizar diversos aplicativos simultaneamente no *smartphone*, cada um com suas finalidades e recursos específicos, permitindo ao usuário a liberdade de escolha.

## 4 DESENVOLVIMENTO

Este capítulo apresenta as diversas etapas do desenvolvimento de *software* do aplicativo denominado Curitour, que serve como um guia turístico virtual para a cidade de Curitiba. Inicialmente realizou-se o levantamento dos requisitos de *software*, seguido pela prototipagem da tela do aplicativo, modelagem UML, obtenção das coordenadas geográficas dos pontos turísticos e do itinerário da linha de turismo da cidade de Curitiba, criação dos ícones dos pontos turísticos e finalmente a implementação do serviço *web* e do aplicativo para dispositivo móvel Android. Ao final do capítulo são apresentadas as telas do aplicativo.

### 4.1 LEVANTAMENTO DOS REQUISITOS

O gerenciamento de requisitos é pré-requisito de grande importância para o sucesso do projeto e a produção de um *software* de qualidade. Todos os projetos, independentes do seu tamanho, podem se beneficiar da atenção dada aos requisitos. Os requisitos de um sistema definem os serviços que o sistema deve oferecer e as restrições aplicáveis à sua operação (CARVALHO; TAVARES, 2002).

Os requisitos funcionais são requisitos que expressam funções ou serviços que um *software* deve ou pode ser capaz de executar ou fornecer. As funções ou serviços são geralmente processos que utilizam entradas para produzir saídas. Os requisitos não funcionais são requisitos que declaram restrições ou atributos de qualidade no processo de desenvolvimento de *software* (CYSNEIROS, 2001).

Neste contexto, os requisitos funcionais (RF) do aplicativo Curitour são:

<b>Código</b>	<b>Descrição</b>
<b>RF01</b>	O sistema deverá possuir entrada de dados para autenticação através de usuário e senha
<b>RF02</b>	Deve mostrar o mapa da cidade de Curitiba com as ruas e o nome das ruas
<b>RF03</b>	O sistema deverá possuir três camadas no mapa, a camada que mostra os pontos turísticos, camada que mostra o itinerário da linha turismo e a camada de pontos de parada da linha turismo

<b>RF04</b>	O usuário poderá habilitar e desabilitar cada uma das camadas no mapa
<b>RF05</b>	O usuário poderá buscar rapidamente um ponto turístico através de uma lista ordenada por ordem alfabética com os nomes dos pontos turísticos
<b>RF06</b>	O usuário poderá obter informações detalhadas ao clicar sobre o ícone do ponto turístico
<b>RF07</b>	O aplicativo deverá permitir o controle do mapa, como mover, aproximar e afastar o mapa com movimentos de toque na tela
<b>RF08</b>	O sistema deverá mostrar a posição atual do dispositivo móvel no mapa

Os requisitos não funcionais (RNF) são:

<b>Código</b>	<b>Descrição</b>
<b>RNF01</b>	A aplicação deverá ser simples e intuitiva para o usuário
<b>RNF02</b>	A aplicação deverá responder rapidamente aos comandos na tela
<b>RNF03</b>	O sistema deverá informar o usuário que deve aguardar, através de ícone animado ou mensagem na tela, durante operações demoradas
<b>RNF04</b>	A aplicação deverá tratar erros inerentes ao sistema, como falha na conexão de dados e na localização geográfica do dispositivo
<b>RNF05</b>	A aplicação deve ser funcional em <i>smartphones</i> e <i>tablets</i>
<b>RNF06</b>	O aplicativo será executado em sistema operacional Android
<b>RNF07</b>	Campo de senha deve ser mascarado

## 4.2 DESCRIÇÃO DO PROTÓTIPO

O protótipo do aplicativo Curitour foi desenvolvido para funcionar em *smartphones* e *tablets* com plataforma mínima Android 3.1 (*Honeycomb*), sendo necessária uma conexão de dados com a Internet, para poder utilizar os mapas e também para autenticar e receber os dados do servidor do aplicativo.

O modelo adotado para o desenvolvimento da aplicação se baseia no modelo cliente-servidor, onde diversos clientes podem acessar simultaneamente o servidor, que busca as informações em um banco de dados, para obter dados atualizados sobre os pontos turísticos, rota da linha de turismo e os pontos de parada, permitindo uma maior flexibilidade na inclusão e alteração dos dados fornecidos ao usuário. A Figura 14 apresenta o modelo de aplicação cliente-servidor.

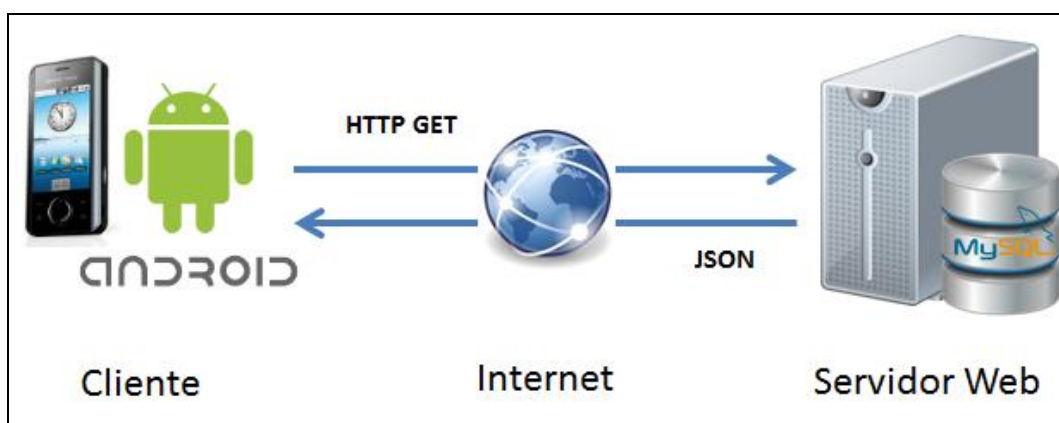


Figura 14 – Modelo cliente-servidor

O aplicativo mostra o mapa da cidade de Curitiba e seus principais pontos turísticos, através de ícones que representam os pontos de interesse. Ao clicar no ponto turístico é exibido ao usuário mais informações detalhadas sobre o local. O usuário pode obter sua localização atual no mapa através dos “Serviços de localização do Google”, como *Wi-Fi* e redes móveis, e também através do GPS, sendo recomendada a utilização deste, pois permite uma melhor localização. O aplicativo ainda permite buscar um ponto turístico rapidamente através de uma lista, que posiciona o mapa sobre o local desejado. Também é possível a exibição do itinerário da linha de turismo de Curitiba e seus pontos de parada. A utilização das novas funcionalidades do Google Maps API versão 2 para Android permite ao usuário funcionalidades como inclinação, zoom e rotação do mapa.



Este aplicativo tem o objetivo de ser simples, interativo, fácil e ao mesmo tempo útil ao turista, que tenha interesse em conhecer melhor os pontos turísticos da cidade de Curitiba sem a necessidade de um guia turístico. A Figura 15 exibe uma tela protótipo do aplicativo desenvolvido no *software Evolus Pencil*.



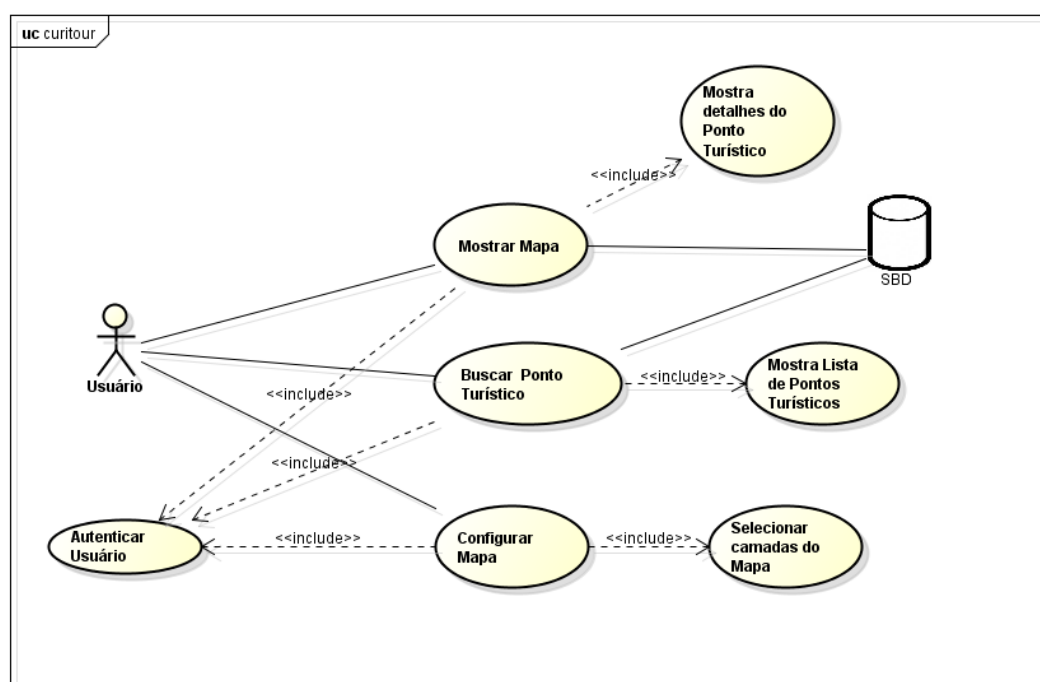
**Figura 15 – Protótipo do aplicativo em *Evolus Pencil***

### 4.3 MODELAGEM

A modelagem do sistema faz parte do processo de desenvolvimento de software, no contexto da Engenharia de Software. Utilizou-se na modelagem o padrão UML, por ser uma ferramenta que auxilia na modelagem de sistemas, do mais simples aos mais complexos, mais adequado à orientação a objetos e proporciona um padrão para a preparação de planos de arquitetura de projeto de sistemas, incluindo aspectos conceituais, como processos de negócios e funções de sistemas, além de itens concretos, como as classes escritas em determinada linguagem de programação, esquema de banco de dados e componentes de *softwares* reutilizáveis (MEDEIROS, 2004).

### 4.3.1 Diagrama de Caso de Uso

O Caso de Uso representa a interação do usuário com o sistema. A Figura 16 apresenta o Caso de Uso do protótipo do aplicativo Curitour, onde o usuário inicialmente se autentica no servidor, e posteriormente possui três opções de interações com o mapa.



**Figura 16 – Diagrama de Caso de Uso**

O fluxo de eventos abaixo mostra a sequência dos eventos do caso de uso da Figura 16:

1.	Autentica usuário
2.	Se usuário autenticado
2.1	Mostra mapa de Curitiba com as camadas selecionadas
2.2	Se clicar no ponto turístico
2.2.1	Mostra informações detalhadas do ponto turístico
2.3	Se clicar em configurações
2.3.1	Mostra lista de seleção das camadas: pontos turísticos, itinerário da linha de turismo e pontos de paradas da linha de turismo
2.4	Se clicar em “Busca”
2.4.1	Mostra lista dos pontos turístico em ordem alfabética para localização rápida no mapa

### 4.3.2 Diagrama de Classes

No diagrama de classes são apresentadas as classes utilizadas no desenvolvimento do aplicativo, a estrutura de classes, com seus atributos e métodos e a relação entre as classes, conforme a Figura 17.

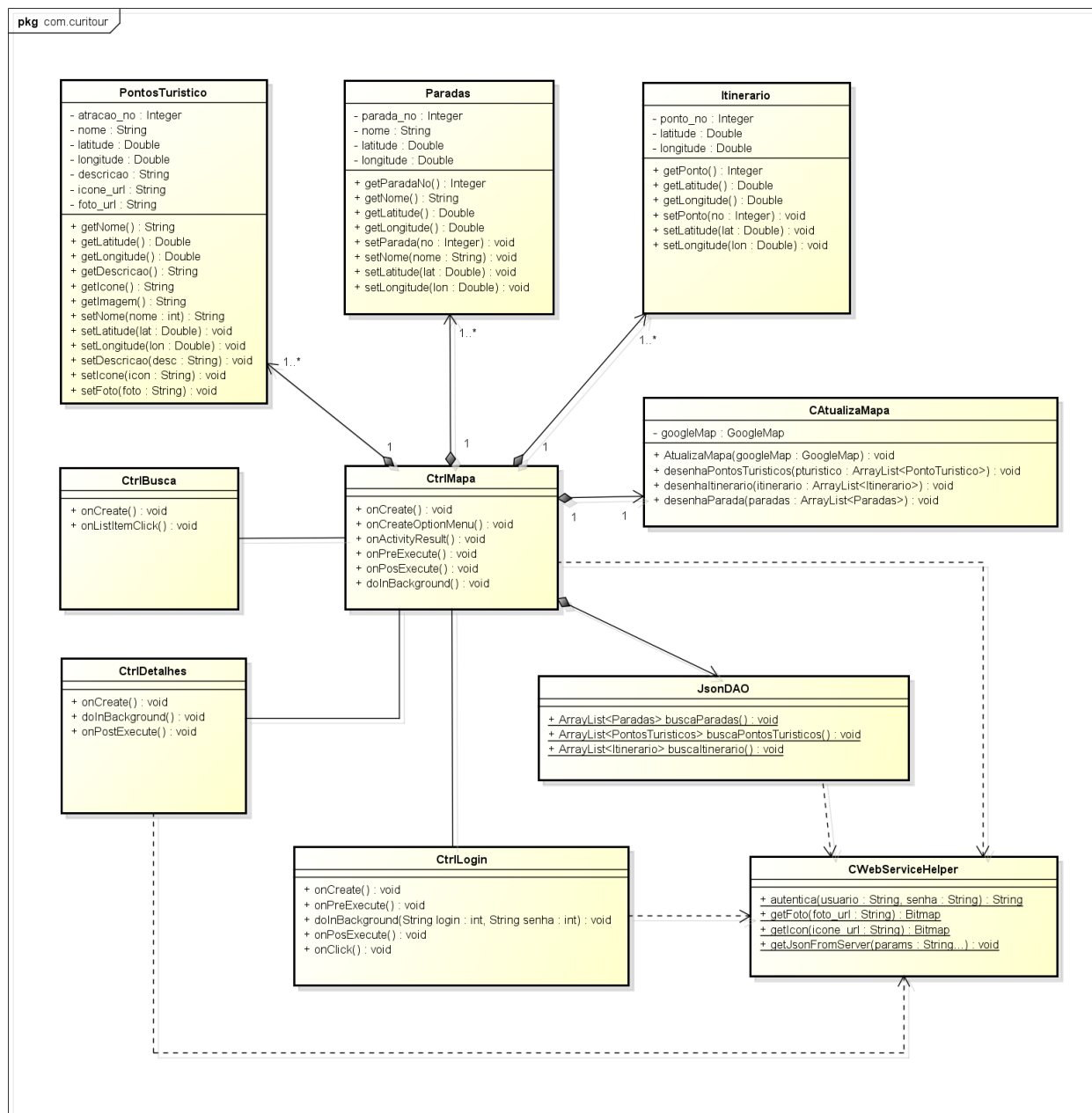


Figura 17 – Diagrama de Classes

### 4.3.3 Diagrama de Sequência

O diagrama de sequência representa as interações dos autores (usuário, aplicativo e banco de dados), o envio de mensagens de solicitação e respostas de retorno entre elas, em uma linha de tempo conforme mostra as Figuras 18 e 19.

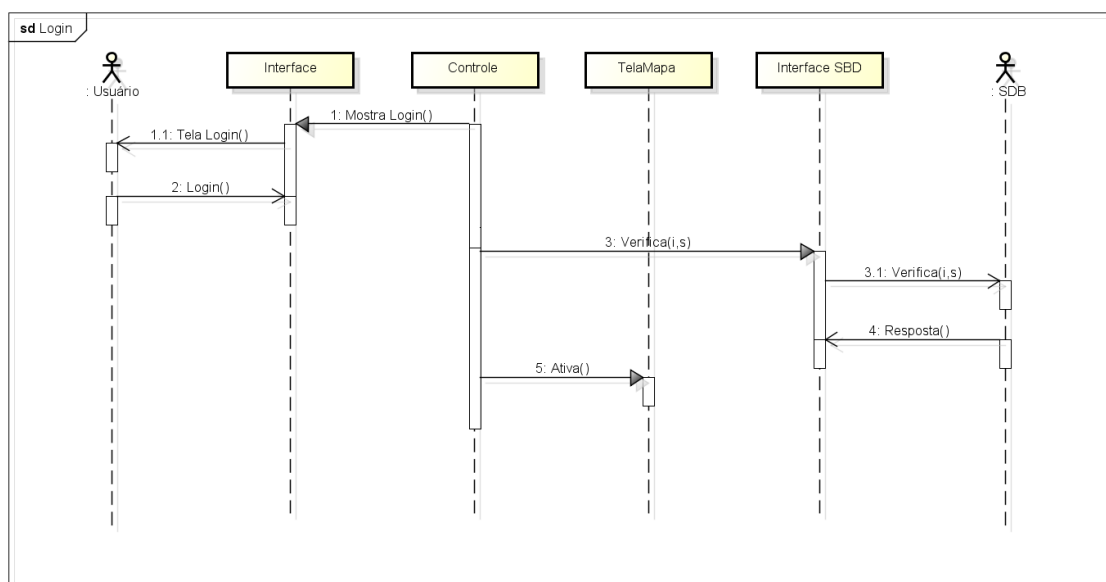


Figura 18 – Diagrama de Sequencia de *Login*

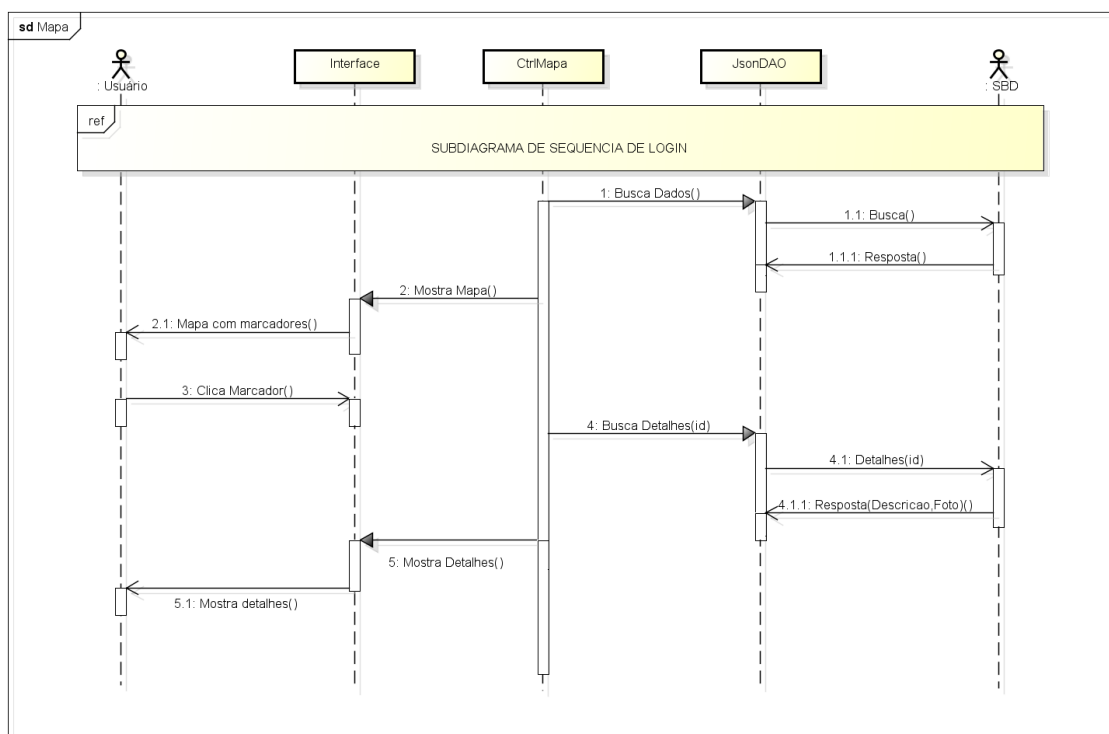
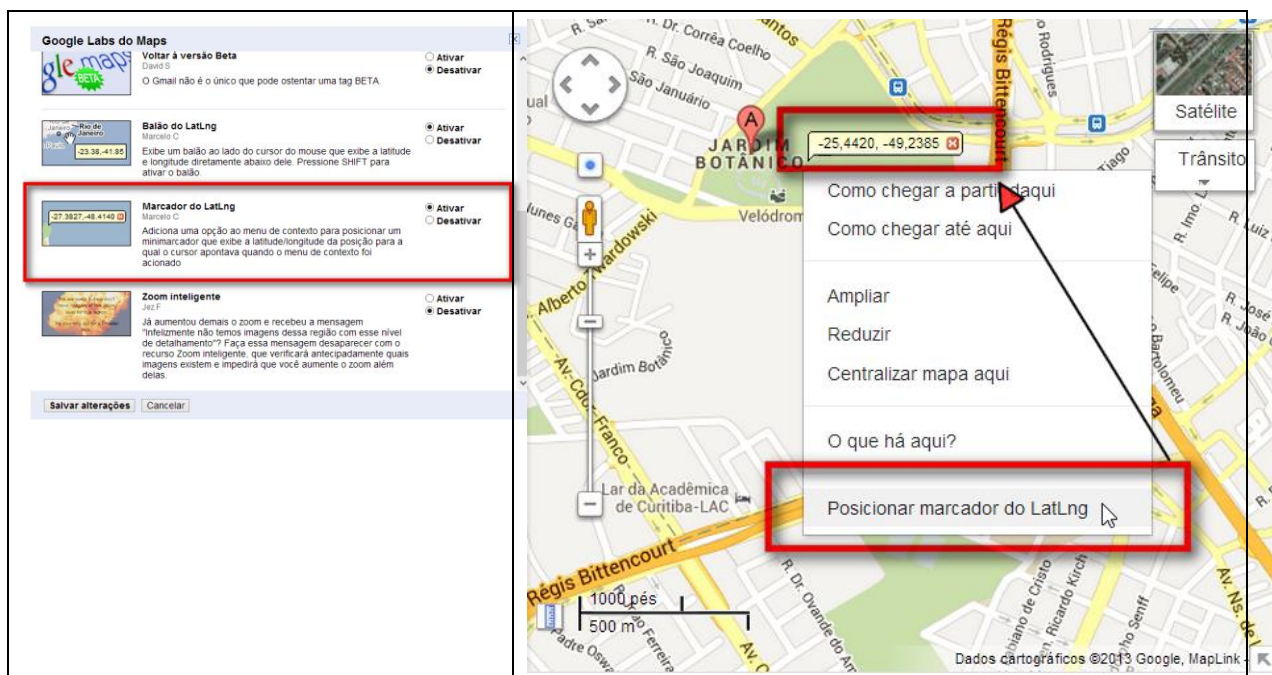


Figura 19 – Diagrama de Sequencia de acesso ao Banco de Dados

#### 4.3.4 Coordenadas Geográficas dos marcadores

Para obter os dados de latitude e longitude dos pontos turísticos, foi utilizado o Google Maps pelo Navegador com o recurso adicional “Marcador do LatLng”, que pode ser habilitada através das opções de ferramentas do Google Labs, conforme a Figura 20.



**Figura 20 – Utilização do LagLng no Google Maps**

As informações sobre o percurso percorrido pela linha de ônibus de turismo de Curitiba e seus pontos de parada foram obtidos através do sistema de Itinerários do site da URBS, mostrado na Figura 21 (URBS, 2013).



**Figura 21 – Itinerário da Linha de Turismo de Curitiba**  
Referência: URBS (2013)

Todos os dados das coordenadas geográficas dos pontos turísticos, pontos de paradas do ônibus da linha de turismo e itinerário do ônibus foram inseridos no Banco de Dados em suas respectivas tabelas.

#### 4.3.5 Desenvolvimento dos Marcadores

Os ícones dos marcadores dos pontos turísticos utilizados no aplicativo foram criados a partir de fotos reais e utilizado o software livre de edição de imagens GIMP versão 2.8 (*The GNU Image Manipulation Program*) que permite recortar, redimensionar e gerar imagens no formato PNG com efeito de transparência, como mostra a Figura 22. O efeito de transparência no ícone é importante, pois resulta em uma melhor aparência ao ser apresentado no mapa para o usuário.

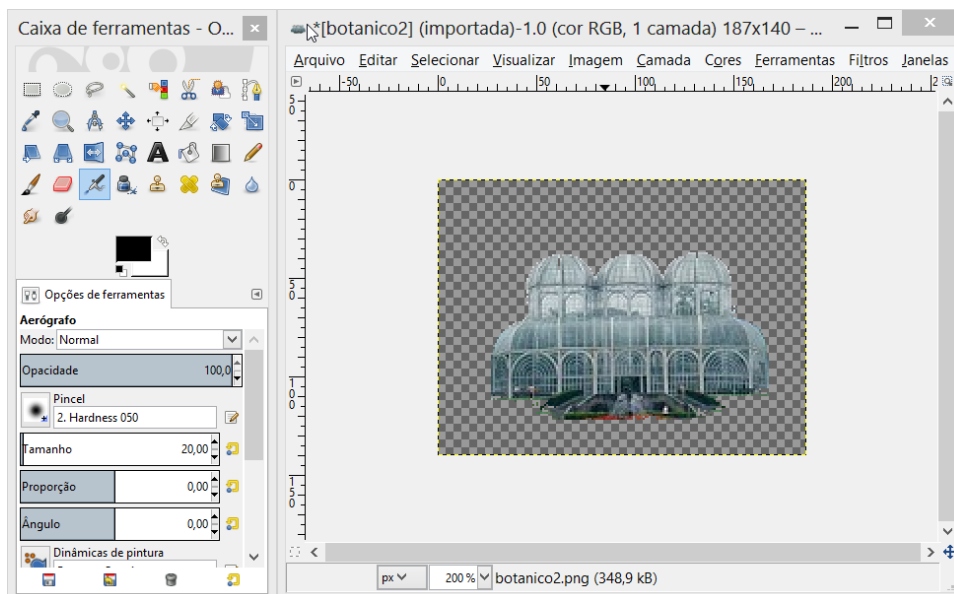


Figura 22 – Edição dos ícones com o programa GIMP

## 4.4 IMPLEMENTAÇÃO

### 4.4.1 Banco de Dados

As informações coletadas dos pontos turísticos, percurso do itinerário e as paradas da linha de turismo foram armazenadas em uma base de dados Mysql, para isto utilizou-se o Mysql Server versão 5.6.

Os dados foram organizados em quatro tabelas, conforme Figura 23.

tb_usuarios	tb_pontosturisticos	tb_paradas	tb_itinerario
id INT(11)	id INT(11)	id INT(11)	id INT(11)
login VARCHAR(45)	nome VARCHAR(45)	parada_no INT(11)	point_no INT(11)
password VARCHAR(45)	descricao LONGTEXT	parada_nome VARCHAR(45)	latitude DOUBLE
nome VARCHAR(45)	latitude DOUBLE	latitude DOUBLE	longitude DOUBLE
sobrenome VARCHAR(45)	longitude DOUBLE	longitude DOUBLE	
email VARCHAR(45)	foto VARCHAR(45)	icone VARCHAR(45)	
<b>Indexes</b>	<b>Indexes</b>	<b>Indexes</b>	<b>Indexes</b>
PRIMARY	PRIMARY	PRIMARY	PRIMARY
login_UNIQUE			

Figura 23 – Modelo do Banco de Dados

A tabela `tb_usuarios` armazena as informações de autenticação dos usuários. A tabela `tb_pontosturisticos` armazena as informações sobre os pontos turísticos, sendo que nos campos `foto` e `icone` são armazenados o nome do arquivo a ser buscado no servidor. A tabela `tb_paradas` armazena o número da parada, nome e posição geográfica das paradas da linha de turismo, e finalmente a `tb_itinerario` armazena todos os pontos necessários para desenhar o itinerário da linha de turismo no mapa.

#### 4.4.2 Servidor

Para o desenvolvimento do *web service*, utilizou-se o ambiente de desenvolvimento Eclipse Java EE Juno Release 2 e o servidor web Apache Tomcat 7.0. Para criar o serviço *Web RESTfull* foi utilizado o *framework* Jersey, que é código-livre e permite separar os detalhes de baixo nível da comunicação cliente-servidor, sendo este uma referência de implementação de JAX-RS, tornando fácil o desenvolvimento de serviços RESTfull com Java (ORACLE, 2013).

Optou-se por utilizar o formato JSON para realizar a troca de informações entre o cliente o servidor, por ser um protocolo mais simples, mais compacto que XML e por já possuir suporte nativo no Android através do pacote “org.json”.

Para utilizar este *framework* é necessário criar no Eclipse um “*Dynamic Web Project*”, copiar os arquivos de biblioteca para o diretório `/WebContent/WEB-INF/lib` da aplicação, e incluir no arquivo `web.xml` definições para redirecionar todas as requisições REST para o *servlet* Jersey do pacote java do projeto, conforme mostra a Figura 24. Também foi adicionado nas bibliotecas o pacote `mysql-connector-java` para conexão com o banco de dados MySQL.



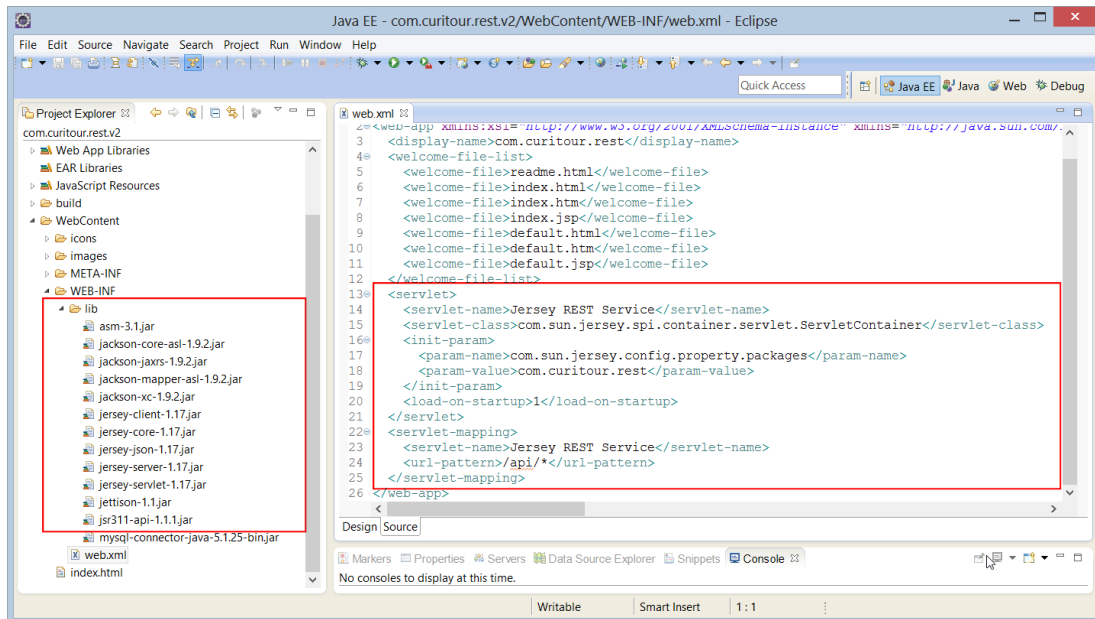


Figura 24 – Implementação de um serviço web RESTful

Os recursos são as peça-chave que compõe um serviço *web* RESTful e são identificados por Identificadores globais, normalmente utilizando URIs. Os recursos são manipulados através de requisições HTTP, através dos métodos GET, POST, PUT e DELETE (IBM, 2009). Em JAX-RS os recursos são implementados com uma anotação `@Path` que compõe um identificador. A anotação `@Get` informa que o método será acionado por uma requisição HTTP GET. A anotação `@Produces` informa o MIME type da resposta HTTP, no caso deste trabalho utiliza-se o tipo JSON. A Figura 25 mostra o trecho do código que realiza a pesquisa no o banco de dados e converte o resultado em um *JSON Array*.

```
package com.curitour.rest.getdata;
...
/* URL: http://localhost:8080/com.curitour.rest/api/v1/getmarkers */
@Path("/v1/getmarkers")
public class GetMarkers {
    @GET
    @Produces(MediaType.APPLICATION_JSON)
    public Response returnAllPcParts() throws Exception {

        PreparedStatement query = null;
        Connection conn = null;
        String returnString = null;
        Response rb = null;
```

```

try {
    // Conexão com o Banco de Dados
    conn = ConnectionFactory.getConnection();
    query = conn.prepareStatement("select * from tb_pontosturisticos");

    // Executa a Query
    ResultSet rs = query.executeQuery();

    // Cria o objeto JSON Array
    ToJSON converter = new ToJSON();
    JSONArray json = new JSONArray();

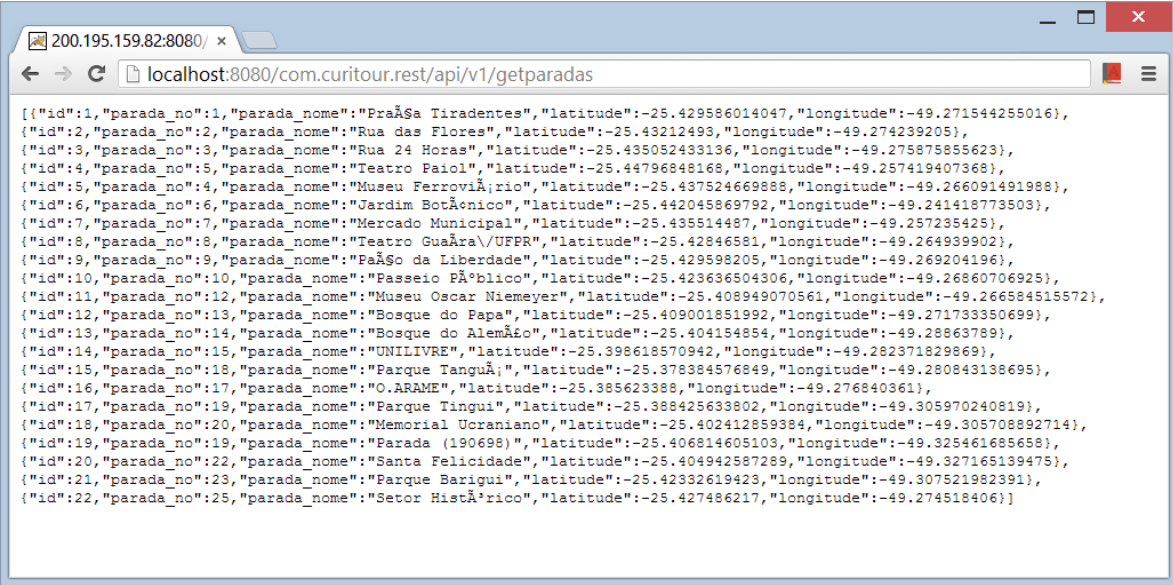
    // Converte o ResultSet em um JSON Array
    json = converter.toJSONArray(rs);
    query.close(); //close connection

    // Envia resposta ao cliente
    returnString = json.toString();
    rb = Response.ok(returnString).build();
}
...

```

**Figura 25 – Exemplo de conversão de dados para JSON**

Na Figura 26 é mostrado um exemplo de resposta do servidor no formato JSON através do navegador.



```

[{"id":1,"parada_no":1,"parada_nome":"Praça Tiradentes","latitude":-25.429586014047,"longitude":-49.271544255016},
{"id":2,"parada_no":2,"parada_nome":"Rua das Flores","latitude":-25.43212493,"longitude":-49.274239205},
{"id":3,"parada_no":3,"parada_nome":"Rua 24 Horas","latitude":-25.435052433136,"longitude":-49.275875855623},
{"id":4,"parada_no":5,"parada_nome":"Teatro Paol", "latitude":-25.44796849168,"longitude":-49.257419407368},
{"id":5,"parada_no":4,"parada_nome":"Museu Ferroviário","latitude":-25.437524669888,"longitude":-49.266091491988},
{"id":6,"parada_no":6,"parada_nome":"Jardim Botânico","latitude":-25.442045869792,"longitude":-49.241418773503},
{"id":7,"parada_no":7,"parada_nome":"Mercado Municipal","latitude":-25.435514487,"longitude":-49.257235425},
{"id":8,"parada_no":8,"parada_nome":"Teatro Guaãra/UFPR","latitude":-25.42846581,"longitude":-49.264939902},
{"id":9,"parada_no":9,"parada_nome":"Paço da Liberdade","latitude":-25.429598205,"longitude":-49.269204196},
{"id":10,"parada_no":10,"parada_nome":"Passeio Público","latitude":-25.423636504306,"longitude":-49.26860706925},
{"id":11,"parada_no":12,"parada_nome":"Museu Oscar Niemeyer","latitude":-25.408949070561,"longitude":-49.266584515572},
{"id":12,"parada_no":13,"parada_nome":"Bosque do Papa","latitude":-25.409001851992,"longitude":-49.271733350699},
{"id":13,"parada_no":14,"parada_nome":"Bosque do Alemão","latitude":-25.404154854,"longitude":-49.28863789},
{"id":14,"parada_no":15,"parada_nome":"UNILIVRE","latitude":-25.398618570942,"longitude":-49.282371829869},
{"id":15,"parada_no":18,"parada_nome":"Parque Tanguá","latitude":-25.378384576849,"longitude":-49.280843138695},
{"id":16,"parada_no":17,"parada_nome":"O. ARAME","latitude":-25.385623388,"longitude":-49.276840361},
{"id":17,"parada_no":19,"parada_nome":"Parque Tingui","latitude":-25.388425633802,"longitude":-49.305970240819},
{"id":18,"parada_no":20,"parada_nome":"Memorial Ucrâniano","latitude":-25.402412859384,"longitude":-49.305708892714},
{"id":19,"parada_no":19,"parada_nome":"Parada (190698)","latitude":-25.406814605103,"longitude":-49.325461685658},
{"id":20,"parada_no":22,"parada_nome":"Santa Felicidade","latitude":-25.404942587289,"longitude":-49.327165139475},
{"id":21,"parada_no":23,"parada_nome":"Parque Barigui","latitude":-25.42332619423,"longitude":-49.307521982391},
{"id":22,"parada_no":25,"parada_nome":"Setor Histórico","latitude":-25.427486217,"longitude":-49.274518406}
]

```

**Figura 26 – Resposta do servidor web no formato JSON**

A autenticação é realizada no *web server* através de um *servlet* Java, que verifica os parâmetros “user” e “password” recebidos na requisição HTTP GET. Após esta etapa, o *servlet* acessa o banco de dados e verifica se o usuário e senha

existem e se correspondem com o valor correto, retornando uma mensagem de “ok” confirmando a autenticação. A Figura 27 mostra o trecho do código do *servlet* que executa a verificação de usuário e senha.

```

48 protected void doGet(HttpServletRequest request,
49     HttpServletResponse response) throws ServletException, IOException {
50     String getuser = request.getParameter("user");
51     String getpassword = request.getParameter("password");
52     String dbpassword = null;
53     PreparedStatement query = null;
54     Connection conn = null;
55     response.setContentType("text/html");
56     PrintWriter pw = response.getWriter();
57
58     try {
59         conn = ConnectionFactory.getConnection();
60         query = conn
61             .prepareStatement("select password from usuarios where login = ? ");
62         query.setString(1, getuser); // Evita SQL Injection
63         ResultSet rs = query.executeQuery();
64
65         try {
66             while (rs.next()) {
67                 dbpassword = rs.getString("password");
68             }
69         } catch (SQLException e) {
70             e.printStackTrace();
71         }
72         finally {
73             rs.close();
74         }
75         if (dbpassword!=null && dbpassword.equals(getpassword)) {
76             pw.print("ok");
77         } else {
78             pw.print("invalido");
79         }
80     } catch (Exception e) {
81         e.printStackTrace();
82     }
83     query.close(); // fecha a conexão com o banco

```

Figura 27 – Trecho do código do *servlet* de autenticação

#### 4.4.3 Aplicativo Android

Para o desenvolvimento do protótipo para Android, utilizou-se o “SDK ADT Bundle for Windows”, que consiste de um pacote de *software* que contém uma versão do Eclipse adaptada para desenvolvimento Android junto com o SDK (*Software Development Kit*) do Android e o emulador AVD (*Android Virtual Devices*).

##### 4.4.3.1 Biblioteca Google Maps API Android

Para desenvolver aplicativos Android utilizando Google Maps é necessário utilizar obrigatoriamente a biblioteca do Google Maps API versão 2, pois a API versão 1 foi oficialmente removida em 03 de dezembro de 2012, não sendo mais

possível desenvolver novos projetos com a versão 1, porém os aplicativos já desenvolvidos continuarão a funcionar nos dispositivos (MAPS, 2013).

Segundo o site do *Google Developers* (MAPS, 2013), para utilizar a biblioteca do Google Maps versão 2 no Android é necessário executar algumas etapas, descritas resumidamente abaixo:

- 1) Instalar o Google Play Services no Android SDK;
- 2) Criar uma chave de acesso aos servidores do Google Maps. Para obter a chave é necessário registrar o projeto no site da Google API Console e obter um certificado de assinatura para o seu aplicativo. A obtenção da chave é gratuita e pode ser reutilizada em todas as aplicações do desenvolvedor, suportando ilimitado número de usuários;
- 3) Especificar as configurações no arquivo *AndroidManifest.xml*;
- 4) Adicionar um mapa no aplicativo;
- 5) Publicar a aplicação.

Essa nova versão da biblioteca de mapas traz inúmeros recursos adicionais, mas em contrapartida não possuía suporte ao emulador AVD (*Android Virtual Devices*) até maio de 2013, sendo necessário o desenvolvimento do aplicativo diretamente em um dispositivo real. O dispositivo utilizado para o desenvolvimento do Curitour foi um smartphone modelo Samsung Galaxy S2 I9100 com sistema operacional Android versão 4.0.4.

#### 4.4.3.2 Activities

No desenvolvimento de aplicações Android uma das classes mais importantes é a classe *Activity*, responsável por controlar os eventos da tela e definir qual *View* será responsável por desenhar a interface gráfica do usuário (LECHETA, 2010). No desenvolvimento do protótipo foram desenvolvidas algumas *Activities*, descritas abaixo:

*Activity* de Abertura: Mostra uma imagem de abertura do aplicativo.

*Activity* de Autenticação: Mostra a tela para o usuário inserir usuário e senha e um botão para efetuar a autenticação. Ao clicar no botão esta *Activity* verifica se os campos estão preenchidos corretamente, e envia os dados para validação no

servidor. Utilizou-se da classe *AsyncTask* do Android, pois permite executar operações em background e alterar os componentes da interface sem necessidade de utilização de threads ou handlers.

A Figura 28 mostra um trecho do código que realiza a autenticação no servidor, chamando o método *WebServiceHelper.autentica()*, que realiza uma conexão HTTP com o *servlet* de autenticação no *web server*, passando como parâmetros o usuário e senha do usuário.

```

LoginActivity.java
223 public class UserLoginTask extends AsyncTask<Void, Void, String> {
224
225     @Override
226     protected String doInBackground(Void... params) {
227         String response = WebServiceHelper.autentica(mUser, mPassword);
228         return response;
229     }
230
231     @Override
232     protected void onPostExecute(final String result) {
233         mAuthTask = null;
234         showProgress(false);
235
236         if (result.equals("exception")) {
237             Toast toast = Toast.makeText(getApplicationContext(),
238                 "Problema na conexão", Toast.LENGTH_LONG);
239             toast.setGravity(Gravity.CENTER | Gravity.CENTER_HORIZONTAL, 0, 0);
240             toast.show();
241         } else if (result.equals("ok")) {
242             // Chama proxima activity;
243             Intent it = new Intent();
244             it.setClass(getApplicationContext(), MapActivity.class);
245             startActivity(it);
246             finish();
247         } else {
248             mPasswordView
249                 .setError(getString(R.string.error_incorrect_password));

```

Figura 28 – Trecho do código da Activity de autenticação

*Activity* de Mapa: Esta *Activity* possui o componente do Google Maps dentro de um elemento *fragment*, onde será exibido o mapa, como mostra o arquivo XML, na Figura 29, com as configurações desta *View*.

```

1 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2   xmlns:tools="http://schemas.android.com/tools"
3   android:layout_width="match_parent"
4   android:layout_height="match_parent"
5   tools:context=".MapActivity" >
6
7   <fragment
8     android:id="@+id/map"
9     class="com.google.android.gms.maps.MapFragment"
10    android:layout_width="match_parent"
11    android:layout_height="match_parent"
12    />
13
14 </RelativeLayout>

```

Figura 29 – Código XML da Interface de Mapa

No método onCreate() desta activity é executado uma *AsyncTask* que chama a classe *JsonDAO* no seu método *doInBackground*, conforme a Figura 30.

```

318 public class LoadingTask extends AsyncTask<Void, Void, String> {
319     private ProgressDialog dialog = new ProgressDialog(MapActivity.this);
320
321     protected void onPreExecute() {
322         Log.i(">", "Pre-execução");
323         dialog.setMessage("Atualizando dados ...");
324         dialog.setCancelable(false);
325         dialog.setTitle("Aguarde");
326         dialog.show();
327     }
328
329     protected String doInBackground(Void... params) {
330         // Carrega nos objetos locais o conteúdo remoto
331         // por chamada Rest/JSON
332         pturisticos = JsonDAO.buscaPontosTuristicos();
333         itinerario = JsonDAO.buscaItinerario();
334         paradas = JsonDAO.buscaParadas();
335
336         return null;
337     }
338
339     protected void onPostExecute(final String result) {
340         Log.i(">", "Terminou a AsyncTask");
341         dialog.dismiss();
342         desenhaCamadas();
343     }
344 }

```

Figura 30 – Trecho do código da Activity do Mapa

A classe *JsonDAO* foi desenvolvida para realizar a conexão HTTP com o serviço *web* RESTful e converter os dados do formato *JSON Array* em um *ArrayList* de Objetos de Java Antigo Simples (POJO) que armazenam localmente os dados dos Pontos Turísticos, Itinerário e Paradas do ônibus. Uma vez obtido os dados do servidor, estes são mostrados no mapa em sua localização correta com seu ícone correspondente.

A Figura 31 mostra o trecho de código do método que insere os marcadores dos pontos turísticos do mapa com seu ícone personalizado.

```

27 public void desenhaPontosTuristicos(ArrayList<PontoTuristico> pturistico) {
28     for (PontoTuristico pt : pturistico) {
29         // Carrega posição do ponto turistico
30         LatLng latlng = new LatLng(pt.getLatitude(), pt.getLongitude());
31
32         // Coloca marcador no Mapa
33         if (pt.getBitmap() != null) {
34             Marker marker = googleMap
35                 .addMarker(new MarkerOptions()
36                     .position(latlng)
37                     .title(pt.getNome())
38                     .icon(BitmapDescriptorFactory.fromBitmap(pt
39                         .getBitmap()))
40                     .snippet("Clique para detalhes"));
41             pt.setMarker(marker);
42         }
43     }
44 }

```

Figura 31 – Trecho do código que insere os marcadores no mapa

Esta *Activity* possui um menu com duas opções, a de Configurações e Busca rápida.

*Activity* de Configurações: Esta *Activity* estende a classe *PreferenceActivity* para a criação de um menu de configurações através de um arquivo XML, que permite o usuário selecionar as camadas a serem mostradas no mapa: os pontos turísticos, o itinerário e/ou pontos de parada da linha de turismo. A classe *PreferenceActivity* armazena as informações de configurações de forma persistente e que podem ser acessadas por qualquer parte da aplicação através da classe *SharedPreferences*. A Figura 32 mostra o arquivo XML do menu de configurações.

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android" >
3
4     <PreferenceCategory android:title="@string/pref_title" >
5         <CheckBoxPreference
6             android:defaultValue="true"
7             android:key="prefShowMarkers"
8             android:title="@string/pref_show_markers" >
9         </CheckBoxPreference>
10        <CheckBoxPreference
11            android:defaultValue="false"
12            android:key="prefShowItinerario"
13            android:title="@string/pref_show_itinerario" >
14        </CheckBoxPreference>
15        <CheckBoxPreference
16            android:defaultValue="false"
17            android:key="prefShowParadas"
18            android:title="@string/pref_show_paradas" >
19        </CheckBoxPreference>
20    </PreferenceCategory>
21 </PreferenceScreen>

```

Figura 32 – Código XML do menu de configurações

*Activity* de Detalhes: Esta tela é chamada pela *Activity* de mapa, quando o usuário clica em mostrar detalhes do ponto turístico.

*Activity* de Busca: Esta *Activity* recebe a lista de pontos turísticos pelo nome, organizados em ordem alfabética, para localizar rapidamente a localização no mapa de um ponto turístico.

#### 4.4.4 Telas do aplicativo

A primeira tela (Figura 33) mostra a imagem de abertura do aplicativo Curitour.

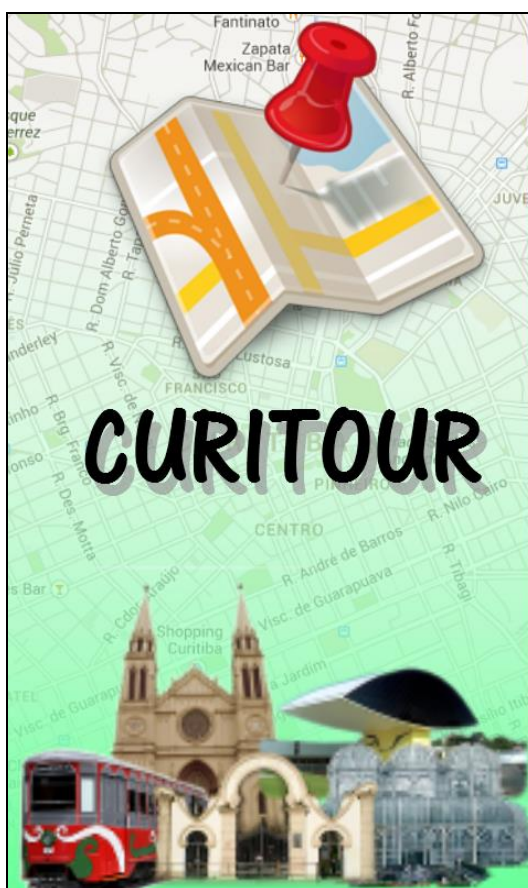


Figura 33 – Tela de abertura do aplicativo Curitour



A Figura 34 mostra a tela de autenticação do usuário.

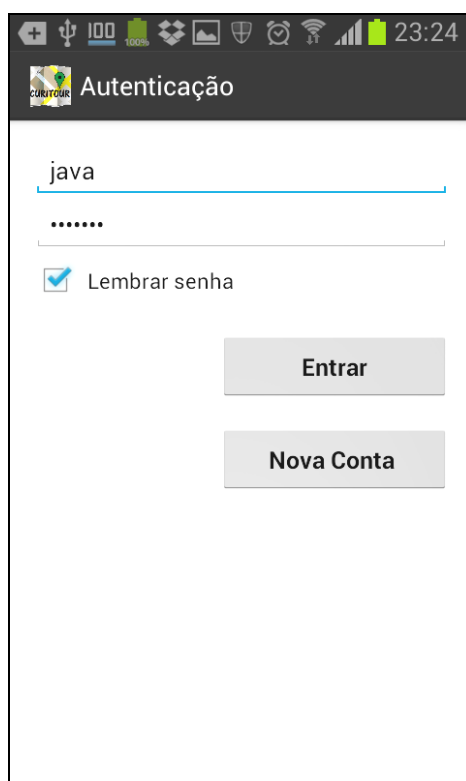


Figura 34 – Tela de autenticação de usuário

A tela principal é a do mapa exibido na Figura 35, que mostra os pontos turísticos da cidade de Curitiba, o itinerário e as paradas da linha de turismo.



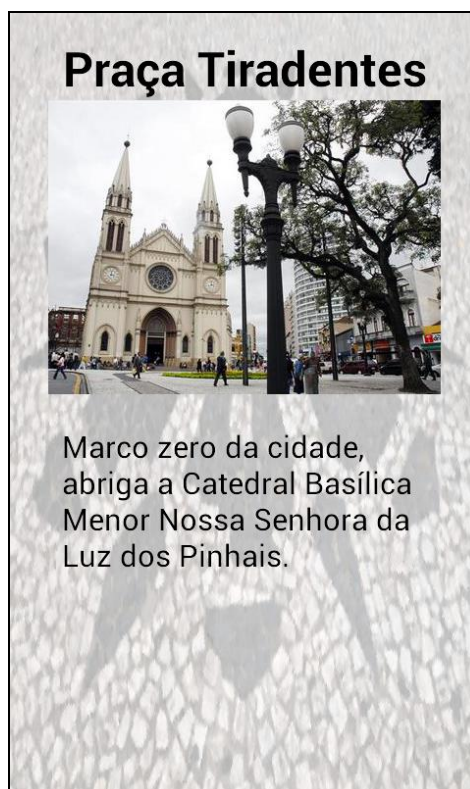
Figura 35 – Tela do aplicativo com todas camadas habilitadas

Ao clicar em um ponto turístico, é exibido seu nome conforme a Figura 36.



**Figura 36 – Balão de informação após o marcador ser clicado**

O usuário pode então obter informações detalhadas sobre o ponto turístico clicando no balão, que trará uma nova janela com a foto e informações mais detalhadas sobre o ponto de interesse, conforme visto na Figura 37.



**Figura 37 – Tela de informações detalhadas do ponto turístico**

O menu permite entrar nas configurações ou realizar uma busca rápida, conforme Figura 38.



Figura 38 – Tela apresentando as opções do menu

Nas configurações o usuário pode selecionar as camadas a serem exibidas no mapa conforme a Figura 39.

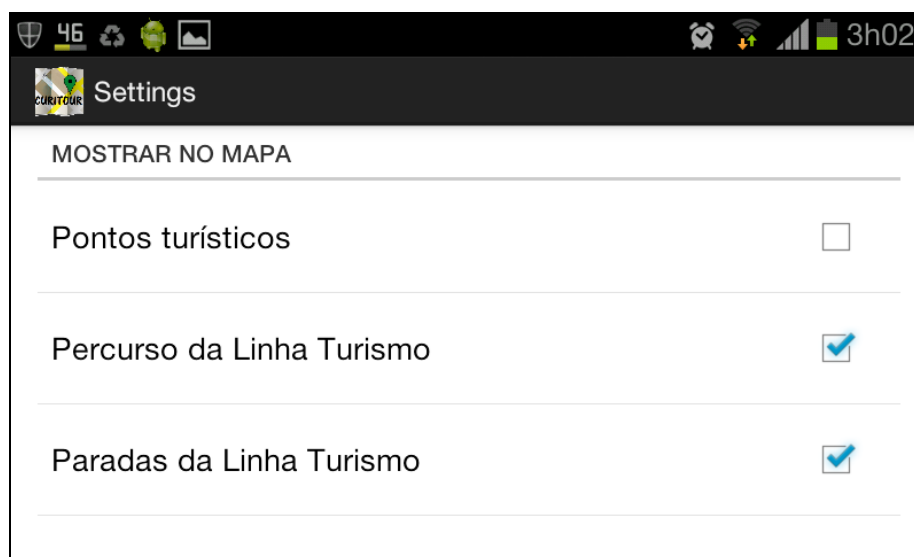
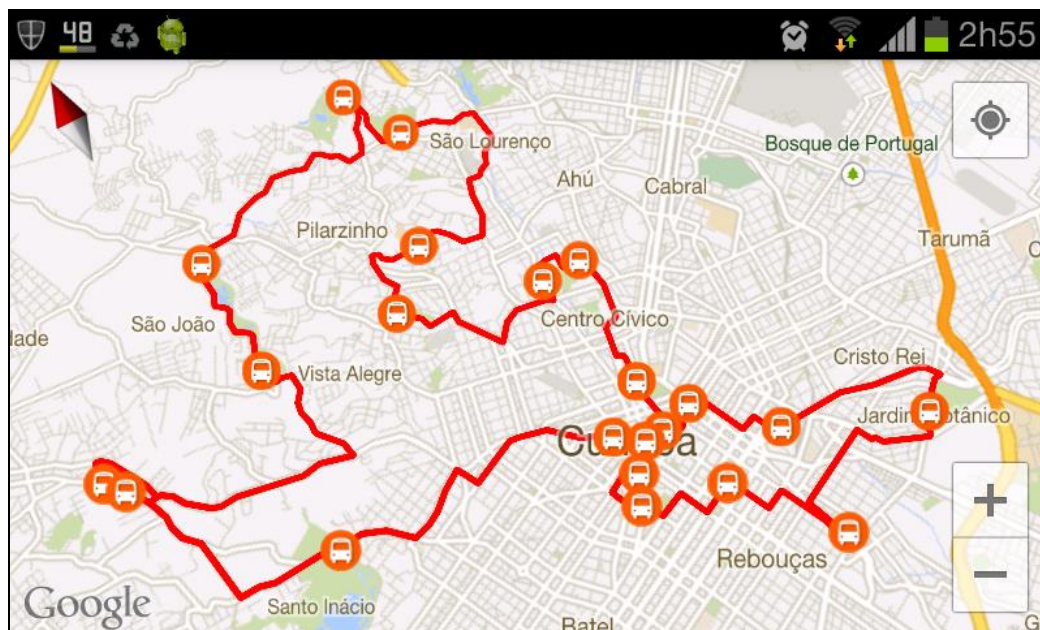


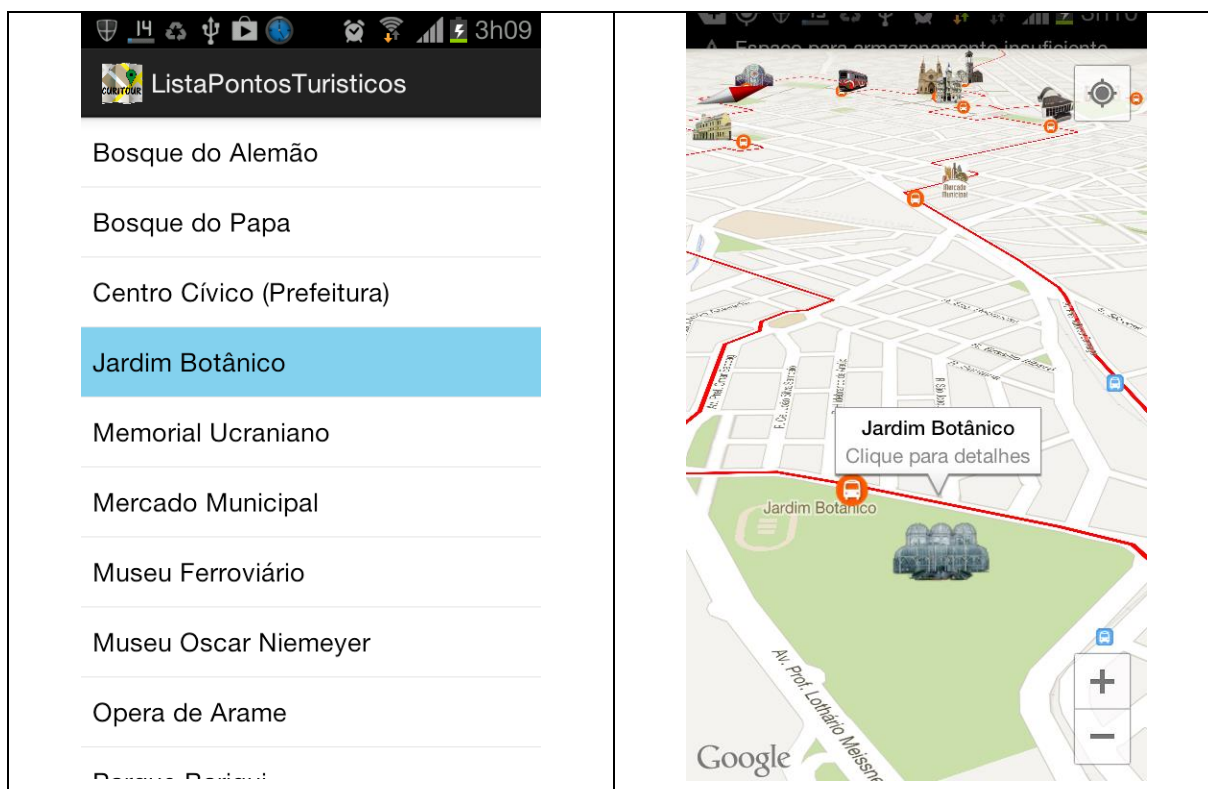
Figura 39 – Tela de seleção de camadas

Se o usuário estiver interessado apenas em ver o itinerário e os pontos de parada o mapa ficará conforme a Figura 40.



**Figura 40 – Mapa com o percurso da linha de turismo**

Selecionando a busca rápida será exibido ao usuário uma lista com os nomes dos pontos turísticos, e ao selecionar algum item na lista o mapa move-se automaticamente para o ponto desejado, conforme a Figura 41.



**Figura 41 – Lista de seleção de ponto turístico**

## 5 CONCLUSÃO

A cidade de Curitiba possui diversos pontos turísticos como parques, praças e museus, possuindo também uma linha de ônibus especial que circula nos principais pontos, mas sem o acompanhamento de um guia turístico. O protótipo do aplicativo Curitour propõe suprir essa lacuna, servir como um guia turístico virtual, através de um aplicativo que pode ser instalado em um *smartphone* ou *tablet* com sistema operacional Android.

Diversos aplicativos vêm sendo desenvolvidos para o setor turístico brasileiro, para atender a demanda nacional de turistas, mas também visando atender a demanda de turistas estrangeiros que visitarão o Brasil nos próximos anos. Esses aplicativos ajudam os turistas a obterem as informações corretas, como localização do ponto turístico, meios de acesso, horários de funcionamento, sem a necessidade de um agente de turismo, permitindo realizar seu próprio roteiro turístico.

Durante o desenvolvimento surgiram algumas dificuldades, sendo uma delas a adaptação do código desenvolvido inicialmente com a biblioteca do Google Maps para Android versão 1, que foi descontinuada, sendo substituída pela versão 2 que é incompatível com a primeira.

O sistema desenvolvido neste trabalho mostra que é possível a utilização de um aplicativo para auxiliar os turistas a obterem sua localização atual na cidade de Curitiba, que permite buscar rapidamente os pontos turísticos, visualizar o itinerário e os pontos de parada da linha de turismo, permitindo ao usuário montar seu próprio roteiro turístico sem a necessidade um guia na cidade de Curitiba.

Utilizou-se no desenvolvimento do aplicativo os conceitos aprendidos durante o curso de especialização como *web services*, programação Java, comunicação cliente-servidor, utilização dos mapas da Google e localização geográfica.

## 6 TRABALHOS FUTUROS

O protótipo do aplicativo se mostrou funcional, atendendo a todos os requisitos propostos. Este aplicativo pode ser melhorado e adicionado diversos recursos, como também pode ser uma base para o desenvolvimento de aplicativos para outras cidades brasileiras.

Segue abaixo uma lista de sugestões de melhorias para o desenvolvimento de trabalhos futuros:

- Internacionalização para outros idiomas;
- *Layout* alternativo para aproveitar a tela maior de *tablets*;
- Realizar integração do aplicativo com redes sociais tais como *Facebook* e *Twitter*;
- Permitir que o usuário seja visível a outros usuários no mapa;
- Permitir comunicação entre usuários;
- Permitir favoritar e dar nota aos pontos turísticos;
- Inclusão de novas camadas no mapa como bares, restaurantes, hotéis, etc...
- Permitir que a aplicação fale as informações sobre o ponto turístico em vários idiomas;
- Mostrar o tempo restante para chegada do próximo ônibus no ponto de parada da linha de turismo;
- Utilização dos mapas de forma *off-line*;
- Validação do aplicativo com usuários.

## 7 REFERÊNCIAS

CANALYS. **Smart mobile device shipments exceed 300 million in Q1 2013.**

Disponível em: <[http://www.canalys.com/static/press\\_release/2013/canalys-press-release-090513-smart-mobile-device-shipments-exceed-300-million-q1-2013\\_0.pdf](http://www.canalys.com/static/press_release/2013/canalys-press-release-090513-smart-mobile-device-shipments-exceed-300-million-q1-2013_0.pdf)>. Acesso em: 25 Maio 2013.

CARVALHO, Ana Elizabete; TAVARES, Helena Cristina. **Visão geral sobre requisitos.** Disponível em: <<http://www4.serpro.gov.br/imprensa/publicacoes/tema-1/tematec/2002/ttec60>>. Acesso em: 12 Out. 2013.

CYSNEIROS, Luiz Marcio. **Requisitos Não Funcionais : Da Elicitação ao Modelo Conceitual.** PUC-Rio, 2001.

DAVIS, Scott. **Google Maps API, V2 : Adding Where To Your Applications.** Raleigh: The Pragmatic Bookshelf, 2006.

EMBRATUR. **Eventos Internacionais no Brasil : Resultados 2003-2009, desafios para 2020.** Disponível em: <[http://www.turismo.gov.br/export/sites/default/turismo/o\\_ministerio/publicacoes/downloads\\_publicacoes/RELATORIO\\_EVENTOS\\_2003\\_2009.pdf](http://www.turismo.gov.br/export/sites/default/turismo/o_ministerio/publicacoes/downloads_publicacoes/RELATORIO_EVENTOS_2003_2009.pdf)>. Acesso em: 20 Maio 2013.

FERRARO, Richard; AKTIHANOGLU, Murat. **Location-Aware Applications.** Shelter Island: Manning Publications, 2011.

GOMES, Daniel Adorno. **Web Services SOAP em Java.** [s.l.]: Novatec, 2010.

IBM. **Build a RESTful Web service using Jersey and Apache Tomcat.** Disponível em: <<http://www.ibm.com/developerworks/library/wa-aj-tomcat/>>. Acesso em: 20 Jul. 2013.

KATARIA, Mickey. **Announcing Google Maps API v3.** Geo Developers Blog. Disponível em: <<http://googlegeodevelopers.blogspot.com.br/2009/05/announcing-google-maps-api-v3.html>>. Acesso em: 26 Maio 2013.

LECHETA, Ricardo. **Google Android : Aprenda a criar aplicações para dispositivos móveis com o Android SDK.** 2. ed. São Paulo: Novatec, 2010.

LIMA, Jean Carlos Rosário. **WEB SERVICES ( SOAP X REST )**. FATEC-SP, 2012.

MAPS, GOOGLE. **Android Google Maps API v2**. Disponível em: <<https://developers.google.com/maps/>>. Acesso em: 1 Jul. 2013.

MEDEIROS, Ernani Sales de. **Desenvolvendo software com UML 2.0 : definitivo**. São Paulo: Pearson Makron Books, 2004.

MELO, Tiago. **Designer projeta aplicativo como guia de pontos turísticos, em Manaus**. Disponível em: <<http://g1.globo.com/am/amazonas/noticia/2012/12/designer-projeta-aplicativo-como-guia-de-pontos-turisticos-em-manaus.html>>. Acesso em: 10 Maio 2013.

MORGAN, Melissa Johnson; SUMMERS, Jane. **Marketing Esportivo**. Thomson Le. São Paulo: [s.n.], 2008.

NEVES, Itamar Abib (UTP); SEMPREBOM, Elder (UFPR); LIMA, Andréa de Albuquerque (PUCPR). Copa 2014: expectativa e receptividade dos setores hoteleiro, gastronômico e turístico na cidade de Curitiba. *In*: **SIMPOI 2011**. São Paulo: SIMPOI, 2011, p. 16. Disponível em: <[http://www.simpoi.fgvsp.br/arquivo/2011/artigos/E2011\\_T00180\\_PCN69256.pdf](http://www.simpoi.fgvsp.br/arquivo/2011/artigos/E2011_T00180_PCN69256.pdf)>. Acesso em: 20 Jul. 2013.

OHA. **Open Handset Alliance**. Disponível em: <[http://www.openhandsetalliance.com/android\\_overview.html](http://www.openhandsetalliance.com/android_overview.html)>. Acesso em: 25 Maio 2013.

ORACLE. **RESTful Web Services in Java**. Disponível em: <<https://jersey.java.net/>>. Acesso em: 20 Jul. 2013.

PMC. **Aplicativo Curta Curitiba auxilia visitantes**. Disponível em: <<http://www.curitiba.pr.gov.br/noticias/noticia.aspx?codigo=22402>>. Acesso em: 15 Maio 2013.

RABELLO, R. R. **Android: um novo paradigma de desenvolvimento móvel**. Disponível em: <[http://www.cesar.org.br/site/files/file/WM18\\_Android.pdf](http://www.cesar.org.br/site/files/file/WM18_Android.pdf)>. Acesso em: 10 jul. 2013.



RECKZIEGEL, Mauricio. **Entendendo os WebServices**. Disponível em: <<http://imasters.com.br/artigo/4245/web-services/entendendo-os-webservices/>>. Acesso em: 7 Maio 2013.

RONDON, Thiago. **ARQUITETURA REST E O SERVIÇO WEB “RESTFUL”**. Disponível em: <<http://sao-paulo.pm.org/artigo/2010/RESTful>>. Acesso em: 7 Maio 2013.

SQUADRA. **Lançado em BH aplicativo de celular com mais de 10 mil atrações turísticas**. Disponível em: <[http://www.squadra.com.br/noticias/MobUrb\\_noticia\\_O\\_Tempo.html](http://www.squadra.com.br/noticias/MobUrb_noticia_O_Tempo.html)>. Acesso em: 1 Jun. 2013.

TUTORIALSPPOINT. **SOAP**. Disponível em: <<http://www.tutorialspoint.com/soap/>>. Acesso em: 7 Jun. 2013.

URBS. **Ininerários: Rede Integrada de Transporte**. Disponível em: <<http://www.urbs.curitiba.pr.gov.br/PORTAL/itinerarios>>. Acesso em: 9 Jul. 2013.