UNIVERSIDADE FEDERAL DO PIAUÍ
CAMPUS SENADOR HELVÍDIO NUNES DE BARROS
CURSO BACHARELADO EM SISTEMAS DE INFORMAÇÃO

RODRIGO SANTOS COSTA

UM PROTÓTIPO DE APLICATIVO REMOTO PARA CONSULTAS DE ITINERÁRIOS DE ÔNIBUS INTERMUNICIPAIS

PICOS - PI
2016

RODRIGO SANTOS COSTA

UM PROTÓTIPO DE APLICATIVO REMOTO PARA CONSULTAS DE ITINERÁRIOS DE ÔNIBUS INTERMUNICIPAIS

Trabalho de Conclusão de Curso, apresentado ao curso de Bacharelado em Sistemas de Informação do campus Senador Helvídio Nunes de Barros da Universidade Federal do Piauí – UFPI para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Prof. Me. Frank César Lopes Véras.

PICOS - PI

2016

FICHA CATALOGRÁFICA
Serviço de Processamento Técnico da Universidade Federal do Piauí
Biblioteca José Albano de Macêdo

C837p Costa, Rodrigo Santos.
Um protótipo de aplicativo remoto para consultas de intinerários de ônibus intermunicipais / Rodrigo Santos Costa. – 2016.
CD-ROM : il.; 4 ¾ pol. (58 f.)

Monografia(Bacharelado em Sistemas de Informação) – Universidade Federal do Piauí, Picos, 2016.

Orientador(A): Profº. Me. Frank César Lopes Veras.

1. Aplicativo-Android. 2. Aplicativo-Intirnerário-Ônibus 3. Dispositivo Móvel. I. Título.

CDD 005.2

UM PROTÓTIPO DE APLICATIVO REMOTO PARA CONSULTAS DE ITINERÁRIOS DE ÔNIBUS INTERMUNICIPAIS

RODRIGO SANTOS COSTA

Monografia aprovada como exigência parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Data de Aprovação

Picos – PI, 19 de fevereiro de 2016

Frank César Lopes Véras
Prof. Me. Frank César Lopes Véras
Orientador

Ivenilton Alexandre de Souza Moura
Prof. Esp. Ivenilton Alexandre de Souza Moura
Membro

Dennis Sávio Martins da Silva
Prof. Esp. Dennis Sávio Martins da Silva
Membro

Dedico este trabalho a Deus, pois sem ele nada seria possível. Dedico também aos meus pais e aos meus irmãos que sempre me incentivaram, e a todas as pessoas que de certa forma ajudaram direta e indiretamente para a conclusão deste trabalho.

AGRADECIMENTOS

Hoje, posso dizer que vivo uma realidade que mais parece um sonho, porém foi preciso muita dedicação, paciência e perseverança para conseguir chegar até aqui. Mesmo sabendo que estou apenas no começo de uma grande jornada, e que isso foi apenas o começo de grandes desafios que estão por vir, jamais poderia esquecer de agradecer a todos àqueles que me ajudaram a conseguir alcançar meu objetivo.

Sou grato a Deus por ter me proporcionado saúde, força e inspiração para realização deste trabalho.

Aos meus pais, Jussiara e Alfredo, que sempre me incentivaram a estudar e nunca mediram esforços para investir em meus conhecimentos. E aos meus irmãos Aurélio, Neuziane, e Jesus Neto; pelo incentivo, pelas orações a meu favor e por acreditarem na minha capacidade; foram fundamentais para a concretização deste sonho.

Ao meu orientador, Frank César, com quem muito aprendi, não apenas no decorrer do desenvolvimento deste trabalho, mas em todas as disciplinas em que ele ministrou, sempre buscando passar o conhecimento da melhor maneira possível.

A todos os demais professores aos quais tive a oportunidade de conhecer e receber um pouco dos seus conhecimentos: Alcilene Dalília, Allan Jheyson, Allan Rafael, Algeir Sampaio, Dennis Sávio, Flávio Henrique, Fredson Muniz, Ismael Holanda, Ivenilton Alexandre, Juliana Oliveira, Júlio César, Leonardo Sousa, Patrícia Medyna, Patrícia Vieira, Ryan Azevedo e Thiago José.

Obrigado a todos que contribuíram direta ou indiretamente para a conclusão deste trabalho.

“A única maneira de fazer um excelente trabalho é amar o que faz”.

Steve Jobs

“A maneira como você coleta, gerencia e utiliza as informações determina se você vai vencer ou perder.”

Bill Gates

RESUMO

Diariamente a movimentação nos terminais rodoviários é intensa, e principalmente nos períodos de férias escolares, em que as pessoas costumam reunir a família para visitar parentes em outras cidades, nos períodos festivos como carnaval ou feriados prolongados. E como, antes de realizar uma viagem é importante fazer um bom planejamento, pesquisando quais empresas de ônibus fazem rota para o destino desejado, valor da passagem cobrada, além é claro, de pesquisar sobre quais os horários de saídas dos ônibus. Mas nem sempre essas informações estão disponíveis facilmente para o usuário desses serviços. Nesse contexto, a principal finalidade deste projeto é desenvolver uma solução que facilite a vida das pessoas que viajam de ônibus, permitindo consultar e visualizar de forma fácil e rápida o itinerário das linhas de ônibus, a lista de empresas que trafegam para o destino desejado, horários de embarques e valor de passagens. Através de um aplicativo para dispositivos móveis, disponibilizando as informações que o usuário precisa, e proporcionando uma maior comodidade no planejamento de uma viagem, contribuindo para que as pessoas tenham mais tempo para organizar outros detalhes importantes.

Palavras-chave: Aplicativo-Android, Aplicativo-Itinerário-Ônibus, Dispositivo Móvel.

ABSTRACT

Daily drive in bus terminals is intense, and especially during the school holidays, where people usually gather the family to visit relatives in other cities in festive periods such as carnival or long holidays. And how, before making a trip is important to make good planning, researching which bus companies make the route to the desired destination, value of charged passage, and of course, the research on which the bus outputs of the times. But not always this information is readily available to the user of these services. In this context, the main purpose of this project is to develop a solution to facilitate the lives of people traveling by bus, allowing query and display easily and quickly the route of the bus lines, the list of companies that travel to the desired destination, shipping schedules and value tickets. Through an application for mobile devices, providing the information that the user needs and providing greater convenience when planning a trip, helping people have more time to organize other important details.

Keywords: *Application-Android, application-Route-Bus, Mobile Device*.

LISTA DE TABELAS

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

ADT	*Android Developer Tools*
API	*Application Programming Interface*
APK	*Android Package*
ART	*Android RunTime*
BaaS	*Backend as a Service*
GPS	*Global Positioning System*
HCE	*Host Card Emulator*
IDC	*International Data Corporation*
IDE	*Integrated Development Environment*
IoT	*Internet of Things*
JSON	*Javascript Objetct Notation*
MVC	*Model View Controller*
NFC	*Near Field Communication*
OHA	*Open Handset Alliance*
P2P	*Peer-To-Peer*
PDA	*Personal Digital Assistant*
SDK	*Software Development Kit*
USB	*Universal Serial Bus*
UX	*User Experience*
XML	*eXtensible Markup Laguage*

SUMÁRIO

1 INTRODUÇÃO

Os serviços de transporte rodoviário de passageiros no Brasil são responsáveis por uma movimentação superior a 140 milhões de usuários por ano. O grau de importância desses serviços pode ser medido quando se observa que o transporte rodoviário por ônibus é a principal modalidade na movimentação coletiva de usuários, nas viagens.

Os terminais rodoviários estão sempre lotados de pessoas, em busca de informações sobre os ônibus, compras de passagens, embarques e desembarques nos ônibus; principalmente nos períodos de férias e períodos festivos, quando as pessoas costumam viajar. E isso tem acarretado enormes filas nas rodoviárias, ocasionando uma enorme perda de tempo às pessoas que apenas precisam obter algumas informações. E, como tempo em nossas vidas é muito precioso, as pessoas tentam economizar o máximo de tempo possível, e procuram sempre opções mais ágeis e que facilitem o acesso às informações de que precisam, com o auxílio de tecnologias, principalmente os dispositivos móveis, que podem ser levado para qualquer lugar.

1.1 Computação Móvel

Segundo Coulouris et al. (2013, p.10) “computação móvel é a execução de tarefas de computação enquanto o usuário está se deslocando de um lugar a outro ou visitando lugares diferentes de seu ambiente usual”. Como exemplos de dispositivos que estão inseridos nesse contexto podemos citar: *PDAs*, *Laptops*, *Smartphones*, *Tablets*, *Smartwatches*, entre outros. A Figura 1, mostrada abaixo ilustra bem o conceito:

Figura 1 – Computação Móvel

Fonte: http://asmarterplanet.com/mobile-enterprise

Ao longo dos anos, os telefones celulares foram evoluindo, e cada novo lançamento traz consigo vários recursos inovadores. Em consequência disso, têm se tornado um item quase indispensável no cotidiano das pessoas. Tanto é, que uma grande parcela da população mundial tem se interessado pela mobilidade, facilidade de acesso às informações em qualquer lugar e em qualquer momento.

E este aspecto tem contribuído e muito para que o celular (*Smartphone*) ganhe cada vez mais funções de destaque no dia-a-dia das pessoas, pois proporcionar conectividade de forma fácil e rápida a outros dispositivos e sistemas, localizando produtos e serviços personalizados também são fatores relevantes nesse contexto. Tudo isso de forma simples, disponibilizando todas as informações que o usuário precisa na palma da sua mão.

Essa crescente busca por mobilidade têm contribuído para a grande demanda por *Smartphones*, *Tablets* e principalmente aplicativos de todos os gêneros no mercado, os quais permitam melhorar a vida das pessoas no trabalho e no seu cotidiano.

> Para a IDC, 90% do crescimento da TI entre 2013 e 2020 será relacionado à Terceira plataforma, que agrega computação em nuvem, dispositivos e aplicações móveis, tecnologias sociais e big data. “A terceira plataforma é formada por soluções, e não somente *Hardware* ou *Software*” explicou Bruno Freitas, analista de mercado da IDC Brasil. “As grandes

> oportunidades nesta plataforma estão na combinação de componentes de acordo com as necessidades do negócio. E a mobilidade passa a ser cada vez mais importante à medida que o foco dos profissionais passa a ser as atualizações em tempo real das informações relevantes para a tomada de decisão" (INTEL, 2013).

Além do mais, com o avanço das estratégias digitais em mobilidade, *Cloud Computing*, mídias sociais, *Smart Cities,* e os dispositivos inteligentes da Internet das Coisas (IoT), surge a oportunidade da integração dessas tecnologias, de forma inteligente e que proporcione maior economia de recursos e eficiência aos seus usuários.

1.2 Escopo e Delimitação do Trabalho

Levando em consideração a importância da mobilidade na vida das pessoas, e a crescente busca das pessoas por tecnologias que as auxiliem em suas necessidades cotidianas, este trabalho demonstra o desenvolvimento de um protótipo de aplicativo remoto para plataforma *Android*, utilizando-se da tecnologia de computação em nuvem[1]. Especificamente na modalidade *Backend-as-a-Service (BaaS) ou Backend* como um serviço, novo conceito de uso da computação em nuvem, no qual a parte principal da aplicação fica armazenada "na nuvem".

A proposta é possibilitar a realização de consultas online dos itinerários de ônibus intermunicipais de forma acessível e rápida, dando suporte às pessoas que costumam utilizar-se deste meio de transporte para realizar suas viagens.

1.3 Objetivos

1.3.1 Objetivo Geral

Este trabalho tem por objetivo principal projetar e desenvolver um protótipo de aplicativo para dispositivos móveis com o sistema operacional *Android.* O aplicativo em questão será utilizado para realização de consultas referentes aos Itinerários de

[1] Conjunto de recursos virtuais facilmente utilizáveis e acessíveis remotamente, tais como hardware, software, plataformas de desenvolvimento e serviços (Vaquero et al, 2009).

ônibus, tais como: dias e horários de saídas para o destino desejado, possível hora de chegada ao destino, valores de passagens.

1.3.2 Objetivos específicos

Como objetivos específicos, pretende-se cumprir as seguintes etapas:

- Elaborar um referencial teórico sobre a plataforma base do aplicativo, bem como as tecnologias e ferramentas utilizadas no seu desenvolvimento.
- Analisar as técnicas e práticas de desenvolvimento de aplicativos para dispositivos móveis e adquirir conhecimentos sobre elas.
- Codificar o aplicativo utilizando as técnicas e práticas estudadas.
- Realizar testes no aplicativo.

1.4 Estrutura da Monografia

Após a introdução que apresentou o escopo e delimitação do trabalho e objetivos, as próximas seções ficaram organizadas da seguinte forma:

- Capítulo 2 - Aborda o surgimento da plataforma móvel *Android*, trazendo um breve histórico. Também descreve sua arquitetura e seus aspectos técnicos.

- Capítulo 3 - Discorre sobre as principais ferramentas, linguagens e tecnologias utilizadas no desenvolvimento do projeto.

- Capítulo 4 - Apresenta os detalhes técnicos da arquitetura do protótipo do aplicativo, a documentação, bem como, as telas do mesmo.

- Capítulo 5 - Para finalizar, o trabalho aborda as com as conclusões e sugestões de trabalhos futuros.

2 A PLATAFORMA *ANDROID*

Nesta seção será abordado o surgimento da plataforma móvel *Android*, sua arquitetura e seus aspectos técnicos.

A tecnologia móvel vem crescendo de forma acelerada nos últimos anos, e grande parte dessa evolução se deve ao surgimento da plataforma móvel *Android*, que tem se destacado no cenário de dispositivos móveis.

Segundo pesquisas realizadas pela IDC (*Internet Data Corporation*), empresa especializada em serviços de consultoria e eventos para os mercados de tecnologia da informação, de telecomunicações e de tecnologia de consumo; a plataforma *Android* vem dominando o mercado de *smartphones* nos últimos anos (especificamente entre os anos de 2012, 2013, 2014, e 2015), e já tem alcançado uma quota de 82,8% do mercado, no ano de 2015 (IDC, 2015).

Tabela 1 – Domínio da plataforma *Android* no mercado, nos últimos quatro anos

Period	Android	iOS	Windows Phone	BlackBerry OS	Others
2015Q2	82.8%	13.9%	2.6%	0.3%	0.4%
2014Q2	84.8%	11.6%	2.5%	0.5%	0.7%
2013Q2	79.8%	12.9%	3.4%	2.8%	1.2%
2012Q2	69.3%	16.6%	3.1%	4.9%	6.1%

Fonte: http://www.idc.com/prodserv/smartphone-os-market-share.jsp

2.1 Histórico

A história do sistema operacional móvel *Android* teve seu início no ano 2003, na cidade de Palo Alto, no estado da Califórnia; onde era situada a Android Inc., empresa fundada por: *Andy Rubinera*, *Nick Sears* e *Chris White*. Inicialmente eles

tiveram a ideia de desenvolver um sistema operacional para smartphones, já visando o crescimento do mercado para esses dispositivos. Porém, a empresa não teve recursos para dar um bom andamento ao projeto, então anos mais tarde, a companhia, acabou sendo vendida para a *Google* Inc., que manteve Andy Rubinera como membro e líder de desenvolvimento de um novo sistema operacional para dispositivos móveis (GUIMARÃES, 2013).

O lançamento da primeira versão do *Android SDK* de desenvolvimento aconteceu no dia 5 de novembro do ano 2007, mesmo ano que ocorreu um marco muito importante na história do *Android*, a *Open Handset Alliance* (OHA), união entre várias empresas do ramo de tecnologia, com a liderança da *Google* (DEITEL, 2015, p. 7).

Open Handset Alliance (OHA) é formado atualmente por 84 membros, dentre eles estão: *Acer* Inc, LG *Eletronics*, *Motorola*, *Samsung Eletronics*, *Sony Ericsson*, e várias outras líderes do mercado de tecnologia (LECHETA, 2015, p. 26). Essas empresas se associaram com objetivos em comum: aperfeiçoar o projeto e transformar o *Android* em uma plataforma completa, de código fonte aberto e disponibiliza-lo gratuitamente. Além é claro, proporcionar melhorias à experiência do usuário, reduzir os custos de aquisição dos dispositivos, e facilitar o acesso dos desenvolvedores de aplicativos móveis, aos recursos de desenvolvimento do *Android*.

Figura 2 – A evolução do *Android*

Fonte: http://www.estudonasnuvens.com.br

O sistema operacional móvel vem passando por várias evoluções desde o seu surgimento, sempre trazendo melhorias na questão de suporte a novos recursos de hardware, e aprimoramento da sua interface e usabilidade. Observando a Figura 2, percebemos que cada versão do *Android* possui nome de uma guloseima, e há uma explicação para isso:

> Android é o sistema operacional que move mais de um bilhão de *smartphones* e *tablets*. Já que esses dispositivos tornam nossas vidas tão doces, cada versão do Android recebe o nome de uma sobremesa. Seja para receber rotas ou jogar na internet, cada versão do Android traz alguma novidade (ANDROID, 2015).

A primeira versão do *Android* comercialmente disponível foi a 1.5, cujo nome era *Cupcake*. O seu lançamento ocorreu no ano 2009, e tinha como novidade: suporte a *uploads* de vídeos para *youtube*, fotos no *Picasa*, correção automática nos textos (NEGREIROS, 2015). Também no ano 2009 foi lançada a versão 1.6, chamada *Donut,* trazendo algumas novidades, como: suporte a várias resoluções de telas, caixa de pesquisas rápidas a partir da tela inicial e com suporte a comandos de voz (GUIMARÃES, 2013). Ainda no mesmo ano, foi lançada a versão 2.0, chamada *Eclair,* trazendo ainda mais atrativos, como é o caso da primeira versão do *Google Maps*, suporte à navegação GPS (Sistema de Posicionamento Global) (NEGREIROS, 2015).

Próxima versão a ser lançada, foi lançada a versão 2.2 (*Froyo),* que incluía suporte a múltiplos idiomas para o teclado, suporte a armazenamento externo, entre outras melhorias (ANDROID DEVELOPER, 2015). Já no final do ano 2010 surgiu uma nova versão, que foi a 2.3 (*GingerBread*), que segundo Negreiros (2015), se tornou uma das versões mais populares, trazendo suporte a alguns recursos inovadores, como: os novos sensores de giroscópio e acelerômetro, suporte à tecnologia NFC (comunicação de campo próximo), entre outros recursos interessantes (DEITEL, 2015).

No ano seguinte foi lançada a versão 3.0 (*Honeycomb*), versão projetada especialmente para dispositivos com telas maiores, como *Tablets*. Ainda no mesmo ano do lançamento da versão 3.0, a *Google* lançou a versão 4.0 (*Ice Cream Sandwich*), com uma interface bastante reformulada e adaptada tanto para

smartphones quanto para tablets; fazendo uma mescla entre a versão (Honeycomb) e a (*Ice Cream Sandwich*) (NEGREIROS, 2015).

Já por volta do ano 2012, segundo Deitel (2015), foi lançada a versão 4.1 (*Jelly Bean*), incluindo suporte para telas externas, aperfeiçoamento na interface, e melhorias na segurança. Além da inclusão de novos recursos, tais como "*Photo Sphere"* que permite os usuários fotografar em 360° graus; e os "*Daydreams",* que são protetores de tela animados.

A *Google* dando continuidade ao desenvolvimento de novas versões lançou a versão 4.4 (*KitKat*), no ano 2013. Uma versão totalmente renovada e sofisticada incluindo aprimoramentos na segurança, acessibilidade, nos recursos gráficos e multimídia (DEITEL, 2015, p.10). Já no final do ano seguinte, ocorreu o lançamento do *Android* 5.0 (*Lollipop*), que veio com uma super-repaginada no visual, utilizando os componentes do novo padrão de design do Google, o Material Design. A versão veio projetada para funcionar não só nos *smartphones* e *tablets*, como de costume, mas também nos novos *wearables*, nas *smart TVs*, e carros inteligentes; impulsionando ainda mais, o desenvolvimento da Internet das coisas (GUIMARÃES, 2015).

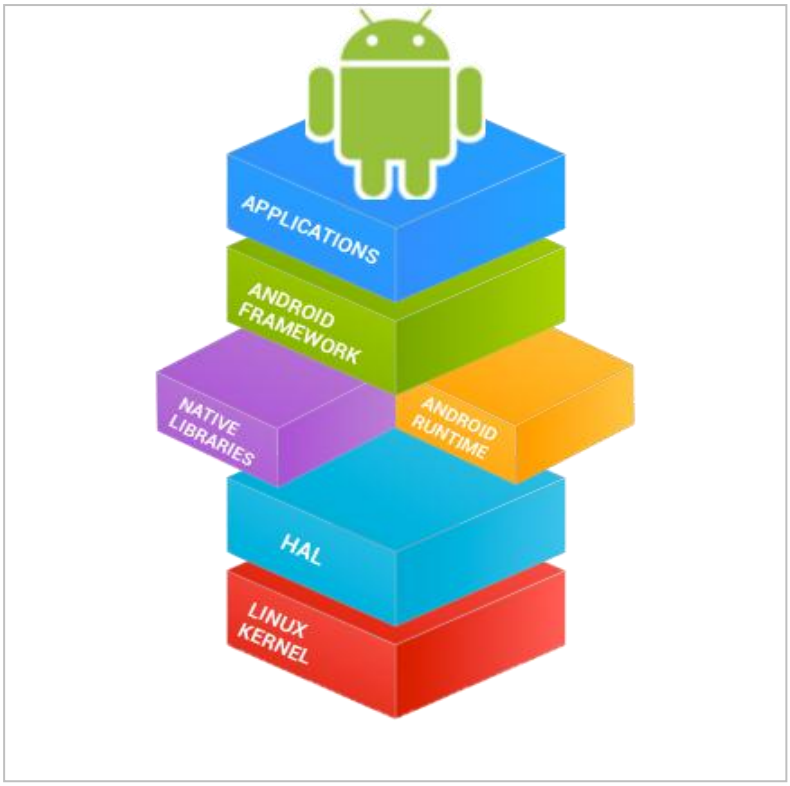
Recentemente foi anunciado o lançamento do *Android M*, a versão 6.0 do *Android* com o codinome (*Marshmallow*). Aconteceu em um evento nos EUA, no dia 29 de setembro de 2015 (NEGREIROS, 2015). A versão por sua vez, veio principalmente focada em segurança e desempenho. Trazendo como novidade o suporte a sensor biométrico, objetivando proporcionar uma maior segurança das informações armazenadas nos dispositivos, e podendo ser usado em inúmeras aplicações que necessitem de autenticação (ANDROID DEVELOPER, 2015).

2.2 Arquitetura da plataforma

Para desenvolver aplicativos para *Android*, além da necessidade de o desenvolvedor ter certo domínio em programação, se faz necessário conhecer os detalhes técnicos da arquitetura da plataforma base. Pois assim, o desenvolvedor estará apto a implementar aplicativos que tirem o maior proveito da plataforma, e consequentemente apresentem um desempenho agradável aos usuários. Vale

ressaltar ainda, que tendo plenos conhecimentos a respeito desses detalhes técnicos da arquitetura, o desenvolvedor conseguirá manipular de forma satisfatória todos os recursos de *hardware* do dispositivo.

Figura 3 – Arquitetura da Plataforma *Android*



Adaptado de: http://source.android.com/source/index.html

Na Figura 3 podemos notar que a arquitetura da plataforma *Android* é representada em forma de camadas, as quais serão explicadas em detalhes, nas seções seguintes deste trabalho.

2.2.1 Camada de Aplicações

Assim como outros sistemas operacionais, o *Android* possui uma gama de aplicativos utilitários, que por padrão já vem com o sistema (PEREIRA, 2009, p. 6). Podemos considerar que, basicamente é uma forma de os usuários testarem a plataforma, antes de instalarem outros aplicativos de terceiros.

A camada responsável pelo gerenciamento dessas aplicações é a *Aplications*, onde se encontram todos os aplicativos que por padrão são instalados

automaticamente com o sistema, tais como: *Browser*, aplicativo da câmera, relógio, calculadora, agenda de contatos, *player* de multimídia, além de todos os outros aplicativos que o usuário instala manualmente.

2.2.2 Camada do *Framework* de Aplicação

Nesta camada se encontra uma coleção de classes e interfaces que compõem o *framework* de aplicação do *Android* são todas implementadas na linguagem Java, e são responsáveis pelo gerenciamento das funções dos componentes usados nos aplicativos. Podemos citar algumas mais importantes (ANDROID DEVELOPER, 2015):

- ***Activity Manager*:** é um dos componentes mais importantes do *Framework* de aplicação, pois é responsável pelo gerenciamento da execução das atividades programadas para serem executadas pelos aplicativos. Desde discar um número em um smartphone a realizar localização no mapa.
- ***Window Manager*:** responsável pelo gerenciamento de janelas dos aplicativos.
- ***View System*:** se trata do gerenciador dos componentes gráficos do sistema.
- ***Package Manager*:** fica responsável pela gerencia dos pacotes utilizados pelo sistema.
- ***Content Providers*:** basicamente, é considerado um provedor de serviços de dados que controla a forma como os dados podem ser acessados e compartilhados entre aplicativos.

2.2.3 Bibliotecas

Essa camada dispõe de um conjunto de bibliotecas de códigos, escritos nas linguagens C/C++. As bibliotecas são utilizadas para auxiliar no trabalho com recursos do sistema. Dentre elas podemos citar (ANDROID DEVELOPER, 2015):

- ***SQLite*:** se trata de uma biblioteca que dispõe de recursos para trabalhar com o banco de dados *SQLite*, banco de dados padrão para persistência de dados em dispositivos móveis.

- ***OpenGL/ES e SGL*:** bibliotecas gráficas, responsáveis pelo suporte a geração de gráficos avançados como: 2D e 3D.
- ***Audio Manager*:** utilizada para gerenciamento de áudios no sistema.
- ***WebKit*:** se encarrega do carregamento de páginas web, no dispositivo.

2.2.4 Android Runtime

Essa camada se trata do ambiente de execução do sistema operacional *Android*, composto pela máquina virtual *Dalvik* e bibliotecas de processamento. Pereira (2009, p. 8) considera: “A Dalvik é uma máquina virtual com melhor desempenho, maior integração com a nova geração de hardware e projetada para executar várias máquinas virtuais paralelamente”. A máquina virtual *Dalvik* se encarrega da interpretação das aplicações instaladas no sistema, para isso, se faz necessário que os arquivos das aplicações sejam compilados em tempo de execução para o formato DEX (*Dalvik Executable*), formato de arquivos suportados pela *Dalvik*.

A partir da versão 4.4 do *Android* (*KitKat*), foi introduzido um novo interpretador em tempo de execução, chamado *ART* (*Android RunTime*) com melhor desempenho que o seu anterior, o *Dalvik*. O objetivo da inserção do novo interpretador é atingir um melhor desempenho e mais velocidade, tanto no carregamento de aplicativos quanto no carregamento do próprio sistema, além de consumir menos bateria e menos processamento do dispositivo (ALLEN, 2015, p. 355).

2.2.5 Biblioteca de Abstração de *Hardware*

Essa camada se encarrega do gerenciamento dos recursos de hardware do dispositivo, além de fornecer as *APIs* de suporte ao desenvolvimento de drivers controladores de periféricos do dispositivo. Basicamente, possui um conjunto de classes e interfaces programadas para se trabalhar diretamente com o *hardware* do dispositivo.

> A equipe tentou criar um padrão, um conjunto de APIs para definir como um *hardware* irá se comunicar com o dispositivo. Da mesma forma que o *Android SDK* fornece aos desenvolvedores classes e interfaces que

> possibilitam que ele faça seu próprio código e rode dentro da arquitetura, com o *hardware* vai acontecer de maneira parecida (SOUZA apud PRADY, 2008).

Enfim, essa camada se encarrega de realizar a intermediação entre as camadas superiores e o *hardware* do dispositivo.

2.2.6 *Linux Kernel*

Essa é a camada mais importante da plataforma, pois é através dela que ocorre o gerenciamento de memória, gerenciamento de processos, e de drivers. Enfim, pode ser considerado o núcleo do sistema, pois é a camada que faz a abstração de *hardware*, realizando a intermediação entre os componentes de hardware do dispositivo com o sistema. Nele ficam localizados os drivers de periféricos do dispositivo: o *driver* do *display*, câmera, áudio, teclado, entre outros.

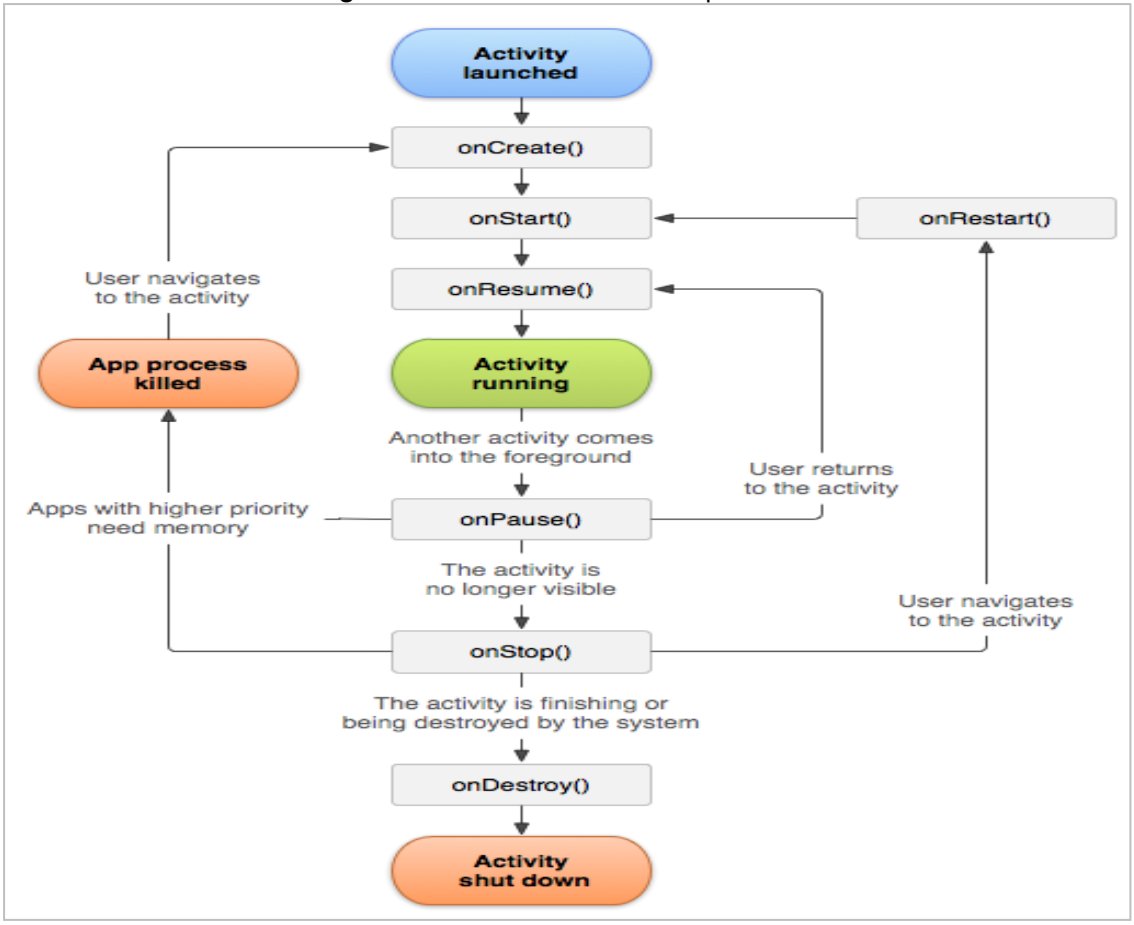
2.3 Ciclo de vida de um aplicativo *Android*

Um aplicativo *Android* é composto por *Activities*, classes que realizam atividades no aplicativo. Uma *Activity*, quase sempre estará associada a uma tela do aplicativo, ou seja, para cada tela do aplicativo deverá ser criada uma classe *Activity,* porém, há casos em que ela executará tarefas que não necessite de uma tela.

Cada *Acitivitiy* possui um ciclo de vida que começa desde o momento em que o aplicativo é iniciado e vai até o seu encerramento ou quando outro aplicativo é iniciado a partir dele.

A Figura 4, mostrada abaixo, demonstra claramente como ocorre o ciclo de vida:

Figura 4 – Ciclo de vida de um aplicativo *Android*



Fonte: <http://developer.android.com/intl/pt-br/reference/android/app/Activity.html>

Para uma *Activity* funcionar corretamente, a classe deve implementar os seguintes métodos:

- ***onCreate*()**: Esse método fica responsável pela criação do processo que realizará determinada tarefa pelo aplicativo. Assim que o método for invocado pela classe, o processo estará criado e ficará esperando até ser iniciado.

- ***onStart*()**: Já, esse método, se encarrega pela inicialização propriamente dita do processo. Quando for invocado inicia-se o processo de execução da tarefa pelo sistema.

- ***onResume*()**: Há momentos em que uma tela do aplicativo precisa ficar em segundo plano para que outra tome o controle, por exemplo: quando a tela do aplicativo possui abas, e o usuário decide mudar para a aba seguinte, então aquela aba anterior continua em execução, porém, fica em segundo plano. Quando o usuário decide retornar para anterior, o método "*onResume*()" deve

ser chamado na aplicação, para que a aba anterior possa reassumir o controle.

- ***onPause*()**: Esse método fica responsável por colocar o processo da *Activity* em segundo plano e salvar o contexto da aplicação na memória, para que em uma posterior retomada de processo, ele continue de onde foi pausado.

- ***onStop*()**: Já, esse método, deve ser chamado quando determinada *Activity* realiza a chamada de uma tela seguinte da aplicação.

- ***onRestart*()**: Esse método é chamado quando o usuário decide voltar para uma tela anterior, a qual ele já teria estado anteriormente.

- ***onDestroy*()**: Esse método se encarrega de encerrar a atividade destruí-la, neste caso não teria como o usuário voltar para essa atividade novamente.

3 METODOLOGIA

Este projeto vem sendo desenvolvido com o propósito de construir um protótipo de aplicativo móvel que irá possibilitar aos usuários de ônibus intermunicipais do estado do Piauí, consultarem fácil e rapidamente, quais empresas regulares de transporte rodoviário de passageiros, fazem rota para destino desejado pelo usuário, dentro do próprio estado. Bem como, obter as informações referentes aos horários de saída e o valor da passagem cobrada. Tudo isso de forma simples, proporcionando aos usuários a possibilidade de obter as informações que desejam através do seu dispositivo móvel. E de posse dessas informações, o usuário, poderá fazer um bom planejamento e escolha da empresa que lhe for mais conveniente.

Para a concepção deste projeto foram utilizadas algumas tecnologias e ferramentas, como: Linguagem *Java*, *XML*, a IDE *Android Studio* como ambiente de desenvolvimento do aplicativo, bem como, os recursos do Material Design para auxilio no desenvolvimento das interfaces, e o SDK de integração remota do *Parse* (conforme explicações nas próximas seções) para a integração com o serviço na nuvem do *Parse*.

3.1 Linguagem *Java*

Para o desenvolvimento de aplicativos nativos para a plataforma *Android*, a linguagem padrão utilizada é *Java* (BURTON, 2014, p.12). Uma linguagem de programação multiplataforma que pode ser utilizada tanto para desenvolvimento de aplicações *Desktops*, quanto para aplicações *Web* e aplicações móveis.

Essa linguagem possui várias características interessantes como: orientação a objetos, que é uma metodologia de desenvolvimento que garante uma boa organização de códigos e reutilização do mesmo; a portabilidade garantindo que uma mesma aplicação desenvolvida com a linguagem possa ser utilizada em diferentes sistemas operacionais, sem haver a necessidade de ter que desenvolver uma instância da aplicação para cada sistema; além da segurança que é um item fundamental a qualquer aplicação. De acordo com Gouveia (2008, p. 26), a portabilidade “é uma das características marcantes da linguagem”.

3.2 XML

A construção das interfaces dos aplicativos para a plataforma *Android* é baseada na utilização da linguagem XML.

> XML (*Extensible Markup Language*) é uma linguagem suportada em tags ou marcas (*meta-markup language*) que fornece um formato para descrever dados estruturados. Deste modo, facilita declarações mais precisas do conteúdo e resultados mais significativos de procura através de múltiplas plataformas. O XML veio permitir surgimento de uma nova geração de aplicações de manipulação e visualização de dados via internet (CARLOS, 2007, p. 50).

Para cada tela do aplicativo é necessário à criação de um arquivo no formato XML, o qual conterá um conjunto de *tags* de marcação dos componentes visuais da interface, como: *layouts*, botões, *inputTexts* para entrada de textos, textViews, esquemas de cores, entre vários outros. A Figura 5 ilustra um exemplo de código XML.

Figura 5 – Exemplo de arquivo XML

```
 9      <ImageView
10          android:id="@+id/img_logo"
11          android:layout_width="200dp"
12          android:layout_height="200dp"
13          android:layout_marginTop="140dp"
14          android:src="@drawable/logo_principal"
15          android:layout_alignParentTop="true"
16          android:layout_centerHorizontal="true" />
17
18      <TextView
19          android:layout_width="wrap_content"
20          android:layout_height="wrap_content"
21          android:textAppearance="?android:attr/textAppearanceLarge"
22          android:text="Bem-Vindo!"
23          android:id="@+id/txt_saudacao"
24          android:layout_marginTop="86dp"
25          android:layout_alignParentTop="true"
26          android:layout_centerHorizontal="true"
27          android:textStyle="bold"
28          android:textSize="30sp"
29          android:elegantTextHeight="false"
30          android:elevation="@dimen/abc_action_bar_content_inset_material"
31          android:ellipsize="start"
32          android:enabled="true"
33          android:focusable="false"
34          android:focusableInTouchMode="false"
35          android:fontFamily="@string/abc_action_bar_home_description" />
36
37  </RelativeLayout>
```

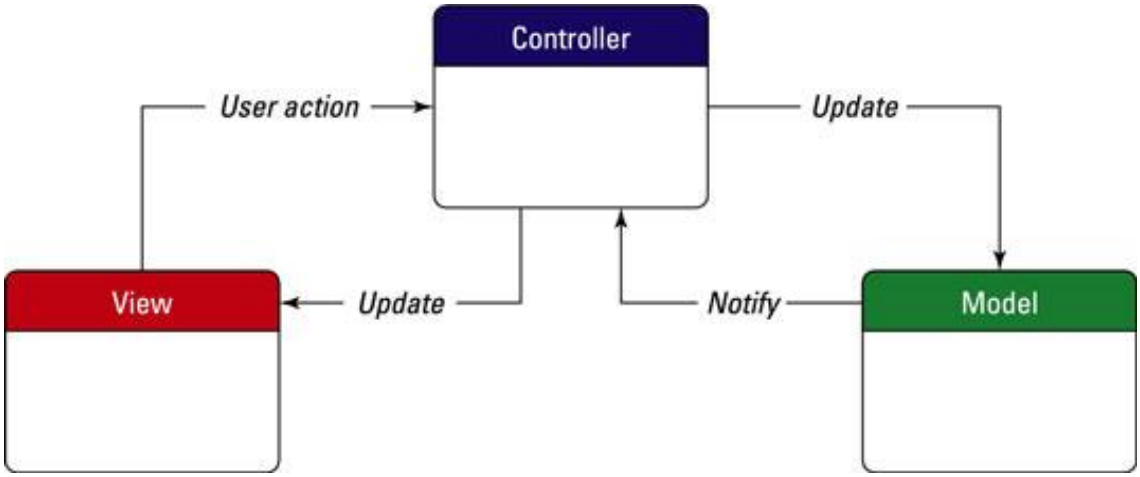
Fonte: O Autor (2016).

De acordo com Carlos (2007), a linguagem XML fornece um padrão que permite programar o conteúdo, a semântica e a esquematização de uma grande variedade de aplicações, desde as mais simples até as mais sofisticadas.

3.3 Padrão MVC

Para desenvolvimento de aplicações, existe uma abordagem muito utilizada atualmente, por grande parte dos desenvolvedores. É o padrão MVC (*Model View Controller*), que segundo Nolan et al. (2014, p. 25), “MVC é um padrão de projeto de software que separa a interface do usuário (visão) das regras de negócio e dados (modelo) usando um mediador (controlador) para conectar o modelo com a visão.”

Figura 6 – Padrão MVC



Fonte: http://www.dummies.com/how-to/content/the-modelviewcontroller-mvc-design-pattern.html

Esse padrão de desenvolvimento tem sido adotado por grande parte dos projetos de software atualmente, pois garante uma melhor organização de códigos, reutilização, e auxilia bastante o trabalho em equipe, já que a maioria dos projetos de softwares é desenvolvida em equipe.

3.4 Android Studio

Até pouco tempo atrás, o ambiente oficial para desenvolvimento de aplicativos *Android*, adotado pela *Google*, era a IDE *Eclipse* em conjunto com *plugin* ADT (*Android Developer Tools*), que por um longo período foi usado pelos desenvolvedores para desenvolver aplicativos. Recebeu vários suportes da *Google*, porém, recentemente a Google lançou uma IDE própria, e então parou de dar suporte ao *Eclipse*, e tem voltado todo seu suporte a essa nova *IDE* chamada *Android Studio*, a qual tem sido padronizada como ambiente de desenvolvimento oficial para plataforma *Android* (ANDROID DEVELOPER, 2015).

> *Android Studio* é uma nova plataforma de desenvolvimento, criada pela *Google*, para desenvolvimento para *Android*. A plataforma é semelhante ao Eclipse, com ADT *Plugin*, oferecendo as melhores ferramentas e funcionalidades aos programadores. Segundo a própria *Google*, com o *Android Studio* a programação para *Android* é mais simples e rápida. (PINTO, 2013).

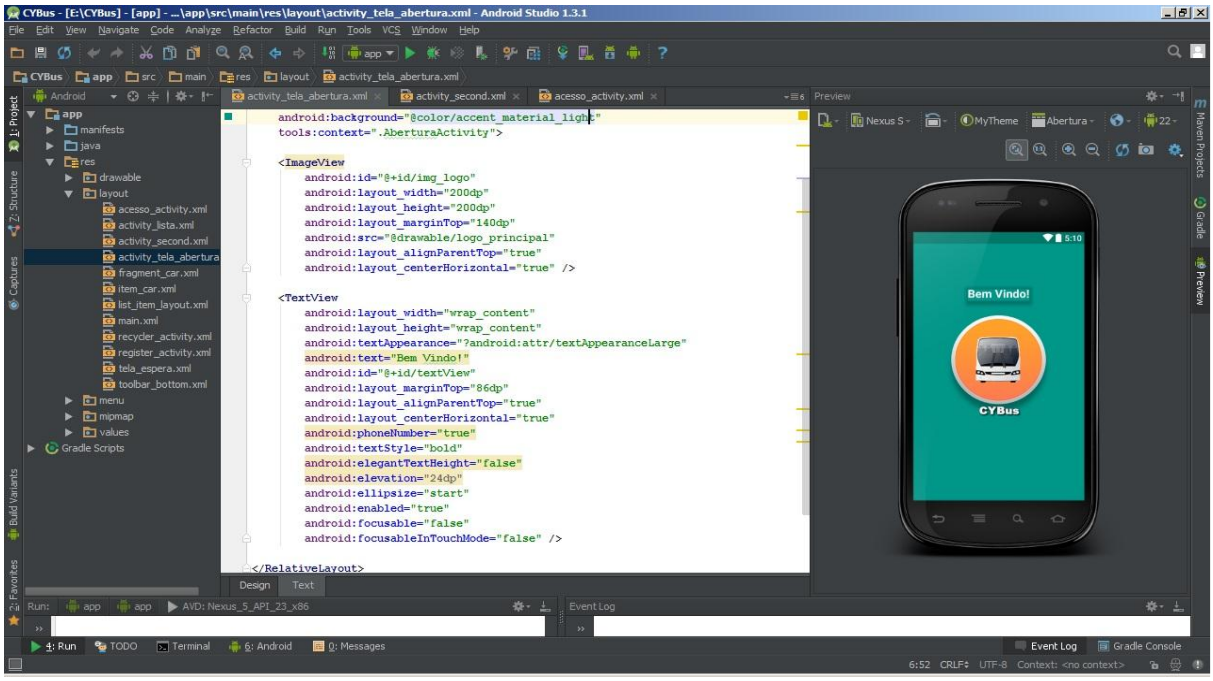
A IDE *Android Studio* foi escolhida para auxiliar no desenvolvimento do projeto, pelo fato dela ser atualmente o ambiente padrão para desenvolvimento de aplicativos *Android*, e também por possuir os melhores recursos e as ferramentas essenciais para auxiliar no desenvolvimento de aplicativos móveis, como:

- ***Gradle*:** Uma poderosa ferramenta de automatização de compilação, e gerenciamento de dependências. É responsável pelo empacotamento dos arquivos e recursos do aplicativo, resultando arquivo com a extensão APK (Pacotes de arquivos do *Android*), para distribuição e instalação em dispositivos com a plataforma *Android* (*Gradle*, 2016).
- **Editor de *layout* inteligente:** Possibilita ao desenvolvedor montar as telas simplesmente clicando nos componentes desejados, arrastando e posicionando no local desejado da tela.
- **Editor de código eficiente:** Auxilia no processo de produção, analise e refatoração dos códigos.
- **Ferramenta de *Preview*:** O desenvolvedor pode visualizar em tempo real como está ficando a organização dos componentes, em diferentes tipos de telas. Tornando o processo de desenvolvimento do design das telas muito mais fácil.

Com o *Android Studio*, desenvolver aplicativos para os mais variados modelos de dispositivos disponíveis no mercado ficou bem mais fácil, pois ele oferece suporte ao desenvolvimento para dispositivos com os mais variados tamanhos de telas, que vão desde telas bem pequenas, como dos novos relógios inteligentes até as *Smart* TVs, que podem possuir telas enormes.

Na Figura 7, mostrada abaixo, podemos visualizar o ambiente de desenvolvimento *Android Studio* em funcionamento:

Figura 7 – Android Studio



Fonte: o autor (2015).

3.5 Material Design

Material Design é um novo conceito de design para interfaces, criado pela *Google*, para uso em aplicações diversas. Basicamente, trata-se de um guia com um conjunto de boas práticas, técnicas de UX *Design*, e padrões recomendados aos desenvolvedores. A fim de construir interfaces ricas que proporcionem uma ótima usabilidade aos usuários das aplicações.

O Material Design foi inicialmente apresentado com o *Android Lollipop*, pelo chefe de experiência de uso do *Android*, em uma conferência anual realizada pela *Google*, na qual estava voltada para assuntos relacionados ao desenvolvimento de novas tecnologias, a *Google* I/O 2014. (ALECRIM, 2014).

> O Material Design nasceu com a necessidade de melhorar o design do Google. Não é segredo que o design do Google era quase inexistente, para não dizer um lixo. O Google é uma empresa de programadores e muito por causa disso não havia uma essência clara de que o design é algo importante (EIS, 2014).

O Material Design está fundamentado nos seguintes princípios, de acordo com *Google* (2014):

- **Superfície tátil:** Se baseia no estudo de papeis e tintas digitais. As superfícies e bordas dos componentes visuais devem possuir efeitos de elevação e sombra. Dando a impressão de profundidade, e oferecendo efeitos visuais que indicam que podem ser tocados e/ou movidos.

Figura 8 – Superfície tátil

Fonte: http://materialdesignblog.com/the-ultimate-guide-to-googles-material-design/

> Superfícies e bordas do Material Design fornecem dicas visuais que são fundamentadas na realidade. O uso de atributos táteis familiares ajuda os usuários a compreender *affordances* rapidamente. No entanto, a flexibilidade do Material Design cria novas *affordances* que substituem aqueles no mundo físico, sem quebrar as regras da física (*GOOGLE*, 2016).

- **Design de impressão:** Está relacionado com a forma como deve ser a apresentação dos elementos de tipografia, tamanho de fontes, imagens, cores e ícones. Reflete no conceito de uso da hierarquia de cores, foco e intenção; para proporcionar uma experiência visual impactante.

Figura 9 – Design de impressão

Fonte: http://www.a2ad.com.br/blog/9-principios-do-material-design-do-google/

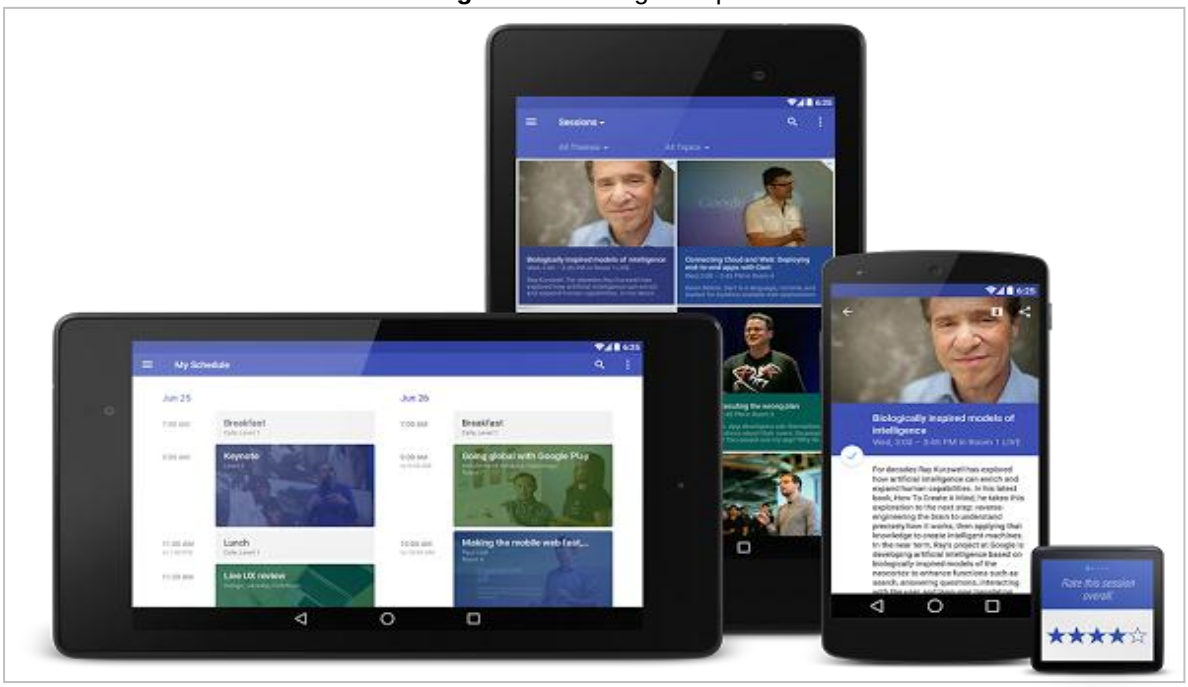
- **Animações significativas:** Utilização de recursos com efeitos de luminosidade, animações realistas e tridimensionalidade.

Figura 10 – Animações significativas

Fonte: http://www.google.com/design/spec/material-design/introduction.html

- **Design adaptável:** Consiste na definição dos componentes visuais, a fim de que sejam ajustáveis em dispositivos com tamanhos de telas e resoluções variadas.

Figura 11 – Design adaptável



Fonte:http://android-developers.blogspot.com.br/2014/10/appcompat-v21-material-design-for-pre.html

Para desenvolver aplicativos para a plataforma *Android*, que utilizem os temas do Material Design, é necessária a utilização da API nível 21, a qual faz parte da versão 5.0 do *Android* (*Lollipop*), porém, nada impede que versões anteriores do

sistema suportem aplicativos que utilizem os componentes do Material Design, já que existe uma API de compatibilidade entre versões chamada *AppCompat* v21. (BANES, 2014).

A API do Material Design possui um conjunto de classes e interfaces escritas na linguagem *Java*, para a manipulação de componentes do Material Design. A API oferece suporte desde a simples *Widgets* para cartões e listas, até animações personalizadas. Para obter mais informações a respeito de como utilizar o Material Design em seu projeto *Android*, a documentação encontra-se disponível em: <http://developer.android.com/intl/pt-br/training/material/index.html>, contendo todas as instruções necessárias para o uso da API.

3.6 *Parse*

Parse[2] é uma plataforma completa para auxiliar os desenvolvedores no processo de desenvolvimento de aplicativos remotos baseados em nuvem, fornecendo um amplo serviço de *Backend*, com grande potencial de escalabilidade e bastante robustez. Conta com uma excelente infraestrutura de armazenamento em nuvem, provendo os recursos essenciais que as aplicações distribuídas necessitam.

Além do mais, o *Parse* ainda oferece um conjunto de *SDKs* e *APIs* de código aberto, proporcionando a integração com diversas plataformas e linguagens de programação, tais como: *Android*, *iOS*, *JavaScript*, *OS X*, *Unity*, *PHP*, *.NET + Xamarin*, *Arduíno*, C padrão, *REST* API, e *Cloud Code*.

> Usando a API Parse, é possível fazer um aplicativo remoto usar a nuvem de forma rápida e com esforço mínimo. Um aplicativo remoto integrado com a API Parse pode facilmente armazenar objetos de dados e arquivos na nuvem Parse, enviar e receber notificações push, gerenciar usuários, lidar com dados de localização geográfica e usar plataformas de mídia social como Twitter e Facebook. Para aplicativos remotos que precisam de ajuste de escala, o SDK Parse oferece toda a elasticidade de uma plataforma de nuvem (ORTIZ, 2013).

A plataforma está inserida no contexto de BaaS (*Backend* como um Serviço), pois fornece recursos que geralmente precisariam ser tratados por uma variedade de tecnologias. Além disso, ela faz a junção desses recursos em uma só plataforma,

[2] https://www.parse.com/docs

provendo um serviço de armazenamento em nuvem e integração com redes sociais como *Twitter* e *Facebook*. “Um mundo de fantásticas aplicações, onde as pessoas passam mais tempo em suas ideias, em vez de sua infraestrutura.” (PARSE, 2016).

Segundo Birani (2013), “já se foram os dias em que você precisava hospedar seus próprios bancos de dados e sincronizá-los para suas aplicações. Agora todos podem ser manipulados na nuvem, memorizar e administrar os dados nunca foi tão fáceil”.

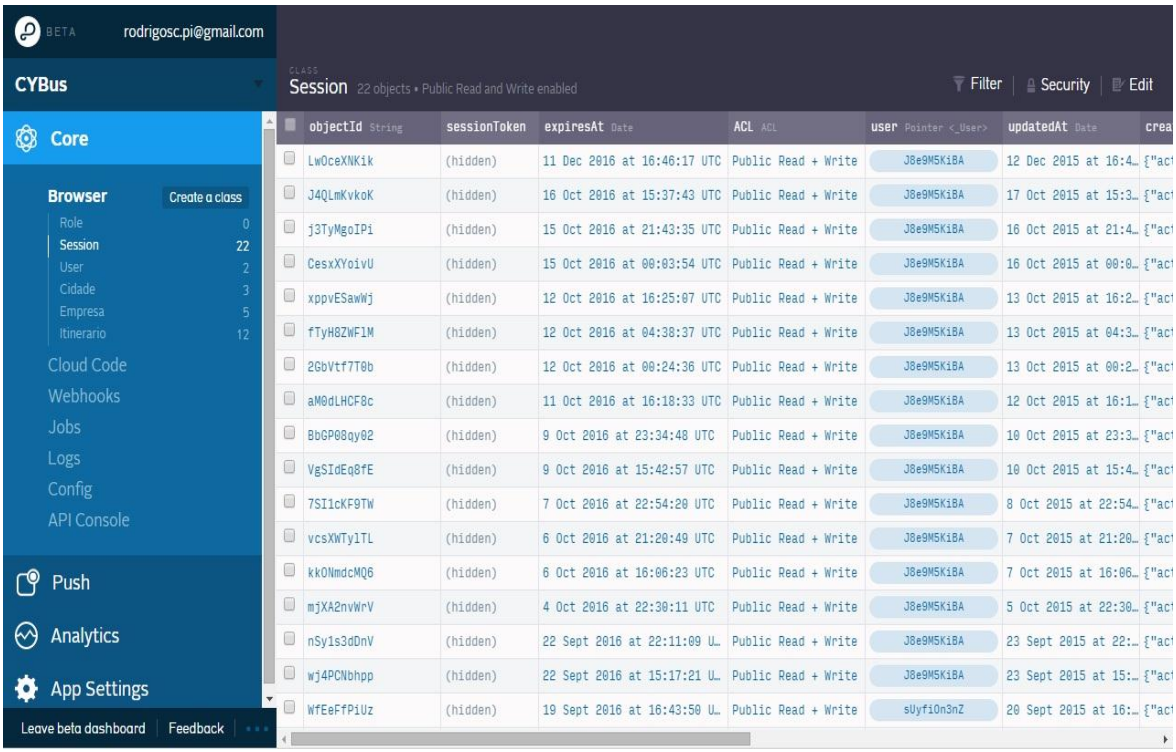
A plataforma *Parse* foi desenvolvida por uma equipe de desenvolvedores engajados na ideia de criar um conjunto de ferramentas e serviços de *Backend* para ajudar no desenvolvimento de aplicativos móveis. No ano 2013, a *Facebook* Inc. adquiriu o Parse com o objetivo de expandir ainda mais o conjunto de recursos (FERNANDEZ, 2013).

O *Parse* trabalha com o conceito de armazenamento de dados em documentos no formato JSON (Notação de Objetos *Javascript*), utilizando pares de chave-valor, mesmo modelo adotado por banco de dados não relacionais, como: *MongoDB*, Cassandra, *CouchDB*, entre outros. Steppat (2009) afirma: “essa abordagem facilita a distribuição de dados entre vários servidores onde cada servidor possui apenas uma fatia dos dados (*shard*)”, que é uma das características da escalabilidade. O armazenamento de dados no *Parse* funciona da seguinte maneira:

- **Classe:** O que seria representado como uma tabela em um banco de dados comum, no Parse é tratada como uma classe.
- **Objetos:** E o que seria tratado como uma linha da tabela, no Parse se trata de um objeto de dados Parse.
- **Campos:** Basicamente os campos são equivalentes a colunas em um banco tradicional.

Os dados das aplicações podem ser gerenciados via *browser*, através de um painel que o *Parse* disponibiliza.

Figura 12 – Painel do *Parse*



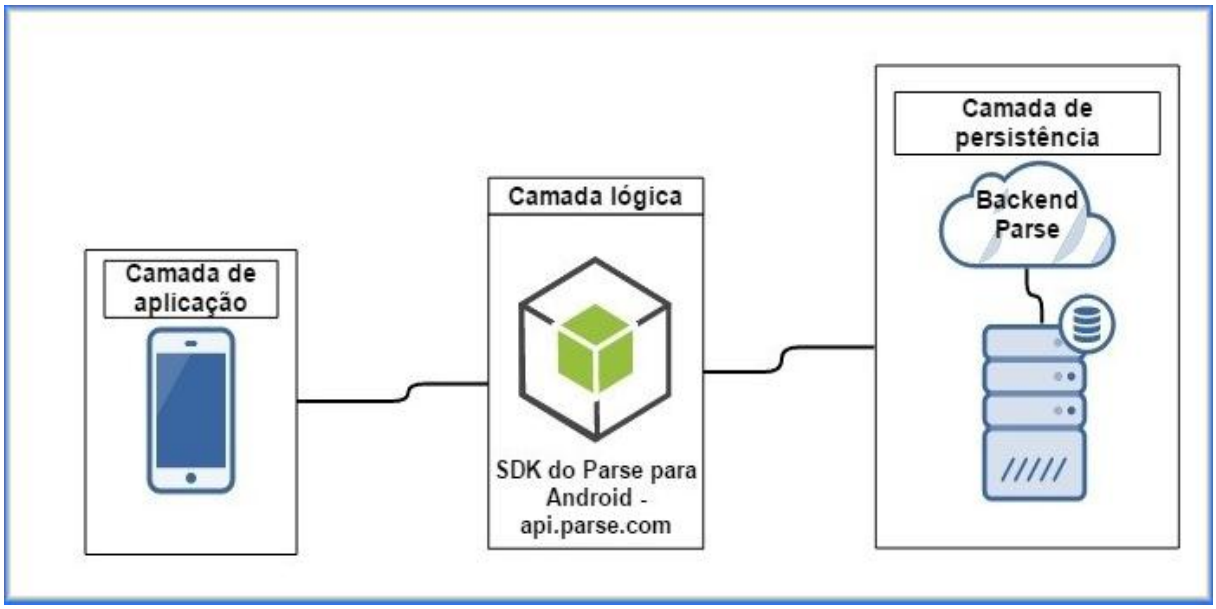
Fonte: O Autor (2016).

Por meio desse painel é possível gerenciar todas as informações que serão consumidas ou enviadas pelas aplicações integradas ao *Parse*, bem como enviar notificações *push* e analisar métricas de uso do aplicativo através de gráficos.

3.7 A Arquitetura

O projeto foi desenvolvido com base na arquitetura cliente-servidor, possibilitando alta manutenibilidade, segurança e disponibilidade das informações consultadas, sua arquitetura está subdividida no modelo de três camadas: Aplicação, Lógica e Persistência; como podemos perceber na Figura 13, mostrada abaixo.

Figura 13 – Arquitetura do aplicativo



Fonte: O Autor (2016).

A Camada de Aplicação se encarrega de prover todo conteúdo visível ao usuário, oferecendo componentes essenciais para o carregamento, manipulação e apresentação de informações. Ou seja, fica responsável pelo gerenciamento de envio das requisições à Camada Lógica, e manipulação das respostas em uma formatação que se adeque ao contexto da aplicação, para que possa ser exibida de forma satisfatória ao usuário.

Já a Camada Lógica fica encarregada pelo gerenciamento das requisições enviadas pela Camada de Aplicação e as respostas recebidas da Camada de Persistência, fazendo a ponte entre a Camada de Aplicação e a Camada de Persistência.

A Camada de Persistência se encarrega de interpretar e processar as requisições enviadas pela Camada de Aplicação, retornando as respostas esperadas de acordo com a busca do usuário, caso sejam encontradas em sua base de dados.

4 O PROTÓTIPO DO APLICATIVO DESENVOLVIDO

Neste capítulo será apresentada a análise de requisitos, arquitetura, *wireframes* de telas, e telas do protótipo.

4.1 Análises de Requisitos

A análise de requisitos é uma tarefa muito importante no processo de desenvolvimento de qualquer projeto, pois é através dela que são analisadas e definidas as funcionalidades e restrições que serão adotadas no projeto que será desenvolvido.

> A análise de requisitos resulta na especificação de características operacionais do software, indica a interface do software com outros elementos do sistema e estabelece restrições que o software deve atender. Permite ainda que você (independentemente de ser chamado de engenheiro de software, analista ou modelador) elabore mais as necessidades básicas estabelecidas durante as tarefas de concepção, levantamento e negociação, que são parte da engenharia de requisitos (PRESSMAN, 2011).

A ideia é desenvolver um aplicativo para dispositivos móveis que possuam o sistema operacional *Android* com a versão 2.3 ou uma versão superior. E para a realização da consulta precisará estar com o acesso à internet habilitada no dispositivo, pois o aplicativo precisa estar sincronizado com o servidor.

Para fins de realização deste trabalho, apenas a plataforma *Android* terá suporte a esse aplicativo, por ser uma plataforma que está mais acessível aos desenvolvedores de aplicativos, disponibilizando mais recursos para o desenvolvimento, e por estar instalada em na maioria dos dispositivos atualmente. Porém, o aplicativo poderá ser disponibilizado para outras plataformas móveis futuramente.

4.1.2 Casos de Uso

Abaixo serão apresentados alguns casos de usos importantes do aplicativo:

- **Cadastro de usuário**: O usuário precisa criar um registro no serviço para poder sincronizar com o servidor e acessar as informações, (Como pode ser observado no Quadro 2).

Tabela 2 – Caso de uso detalhado - Cadastro de usuário

Cadastro de usuário	
Ator primário	Usuário cliente do serviço.
Pré-condições	Conexão de rede ativa no dispositivo.
Fluxo principal	1. Usuário submete os seguintes dados ao formulário para cadastro: • Nome de usuário • Senha • Email 2. O sistema emite uma mensagem informando se o cadastro foi bem sucedido, e o usuário é redirecionado para a tela de *Login*. 3. Em caso de falha, uma mensagem informa que aconteceu alguma falha: por falta de conexão de rede ou algum campo faltando inserção de dados.
Requisitos funcionais	Dispositivo com Android versão 2.3 ou superior. Conexão com a internet.

Fonte: O Autor (2016).

- **Autenticação de usuário**: O usuário precisa estar autenticado para obter acesso às informações do servidor (descrito com detalhes no quadro 3).

Tabela 3 – Caso de uso detalhado - Autenticação de usuário

Autenticação de usuário	
Ator primário	Usuário cliente.
Pré-condições	Informações do servidor, usuário, senha e e-mail. Acesso à internet. Aplicativo instalado no dispositivo.

Pós-condições	Usuário autenticado no sistema.
Fluxo principal	1. Usuário submete as informações para autenticação: • Nome de usuário • Senha 2. O aplicativo emite uma mensagem de sucesso e redireciona para a tela principal. 3. O aplicativo emite uma mensagem de erro e redireciona para a tela de cadastro.
Requisitos funcionais	Conexão com a internet.

Fonte: O Autor (2016).

- **Realização de consultas**: O usuário precisa saber quais empresas de ônibus fazem linha para destino que ele pretende viajar. Então ele pega o *Smartphone*, verifica se há conexão com a internet, abre o aplicativo e se autentica, caso não tenha realizado a autenticação anteriormente. Depois escolhe a opção de busca no menu de opções, então uma tela de formulário é exibida para a inserção dos parâmetros de busca e submissão ao servidor. Depois que ele submete as informações, uma tela é exibida contendo a listagem das informações.

Tabela 4 – Caso de uso detalhado - Realização de consultas

Realização de consultas	
Ator primário	Usuário cliente.
Pré-condições	Usuário cadastrado. Usuário autenticado.
Pós-condições	Usuário de posse das informações desejadas.
Fluxo principal	1. O usuário informa os seguintes parâmetros de busca: • Sua localização atual. • Destino ao qual pretende ir. • Dia que pretende viajar. 2. O aplicativo mostra uma lista de ônibus com saídas agendadas de acordo com os parâmetros informados pelo usuário, e informando o valor da taxa cobrada pela empresa.

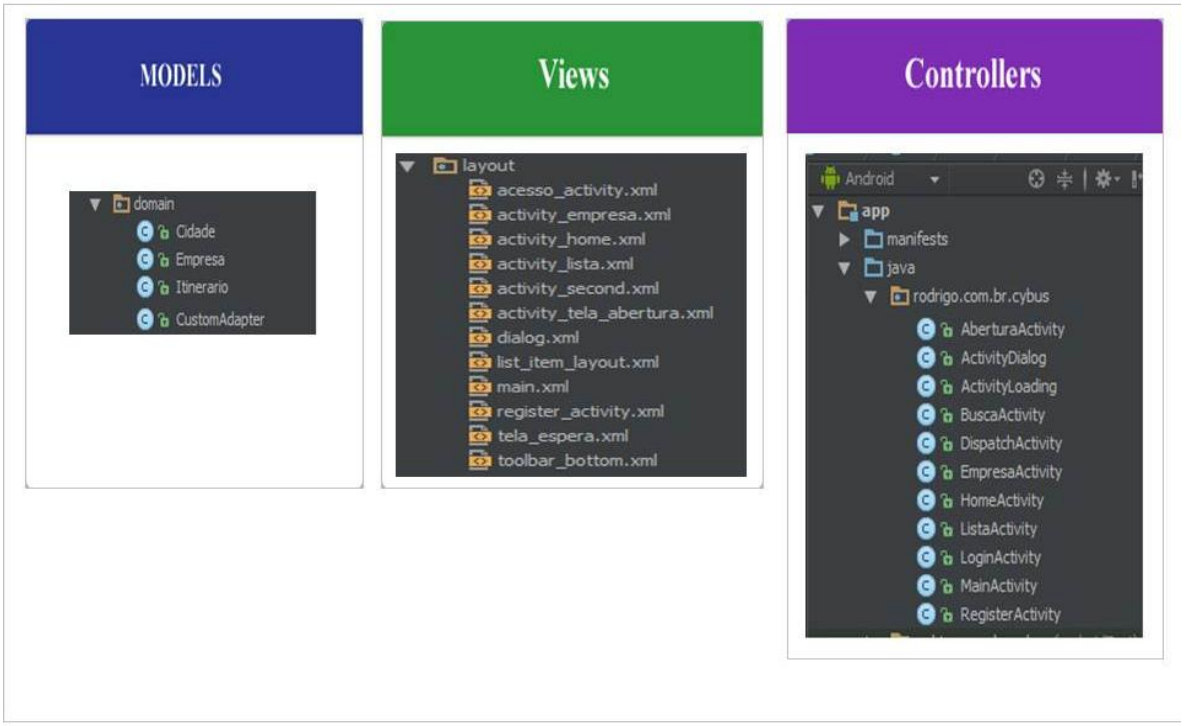
Requisitos não funcionais	Rapidez no acesso às informações.
Frequência de uso	Sempre que o aplicativo for utilizado.

Fonte: O Autor (2016).

4.2.1 Modelagem da Arquitetura

A modelagem da arquitetura do aplicativo foi desenvolvida com base no padrão MVC, como pode ser visto na Figura 14.

Figura 14 – Padrão MVC aplicado ao projeto



Fonte: O Autor (2016).

A camada *Model* é onde ficam as classes responsáveis pelo gerenciamento das regras de negócio e dos dados de domínio da aplicação, realizando acesso à base de dados para fornecer as operações necessárias para manipulação desses dados.

A camada *View* é onde ficam todos os arquivos responsáveis pela apresentação e controle dos dados, se encarregando de controlar como os dados retornados pela camada *Model* serão apresentados ao usuário. Trazendo para o

contexto desta aplicação, esses arquivos seriam basicamente os arquivos de *Layouts* de telas no formato XML (*eXtenssible Markup Language*).

A camada *Controller* é onde se localizam as classes encarregadas por realizar a interpretação das entradas de dados do usuário através dos periféricos de entrada do dispositivo, fazendo a intermediação entra a camada *View* e a camada *Model*.

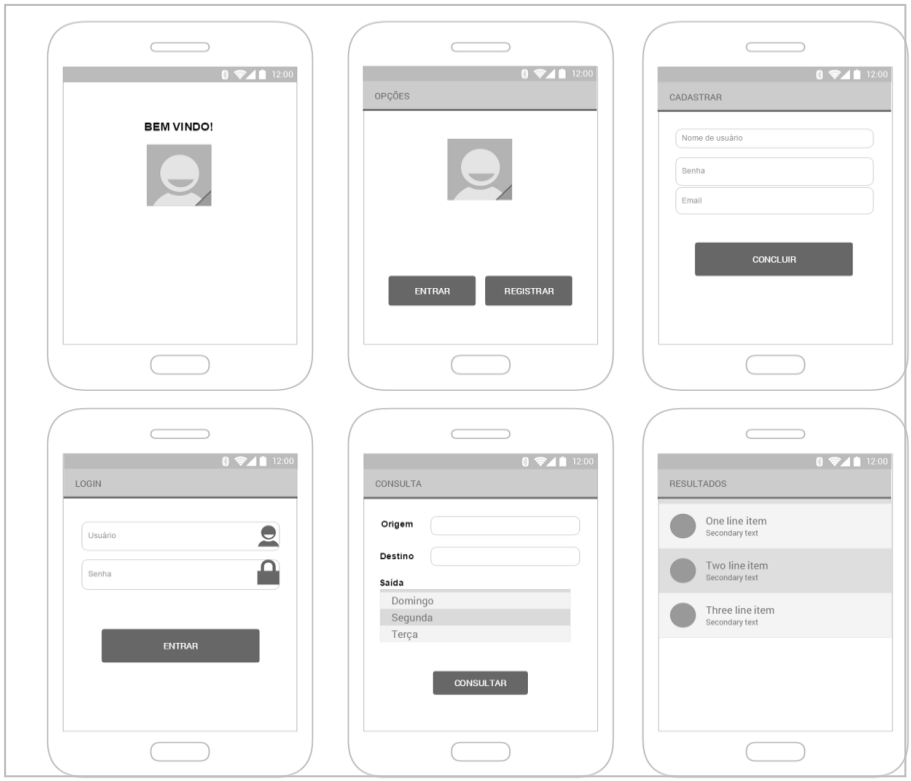
4.3 *Wireframes* das Telas

Antes de iniciar o desenvolvimento de qualquer aplicação é importante arquitetar como ficará cada componente da interface, pois é através da interface que o usuário poderá manipular aplicação com facilidade. De acordo com Zemel (2010):

> A finalidade de um *wireframe* é comunicar o layout de uma página sem se ater a cores e elementos de design. *Wireframes* são grandes 'salvadores de tempo' e ajudam todas as partes envolvidas no projeto a chegar a acordos sobre a colocação dos principais elementos de página, como cabeçalhos, áreas de conteúdo, menus de navegação e rodapés.

A Figura 15 ilustra como ficaram os *wireframes* criados para o projeto.

Figura 15 – *Wireframes* das telas



Fonte: O Autor (2015).

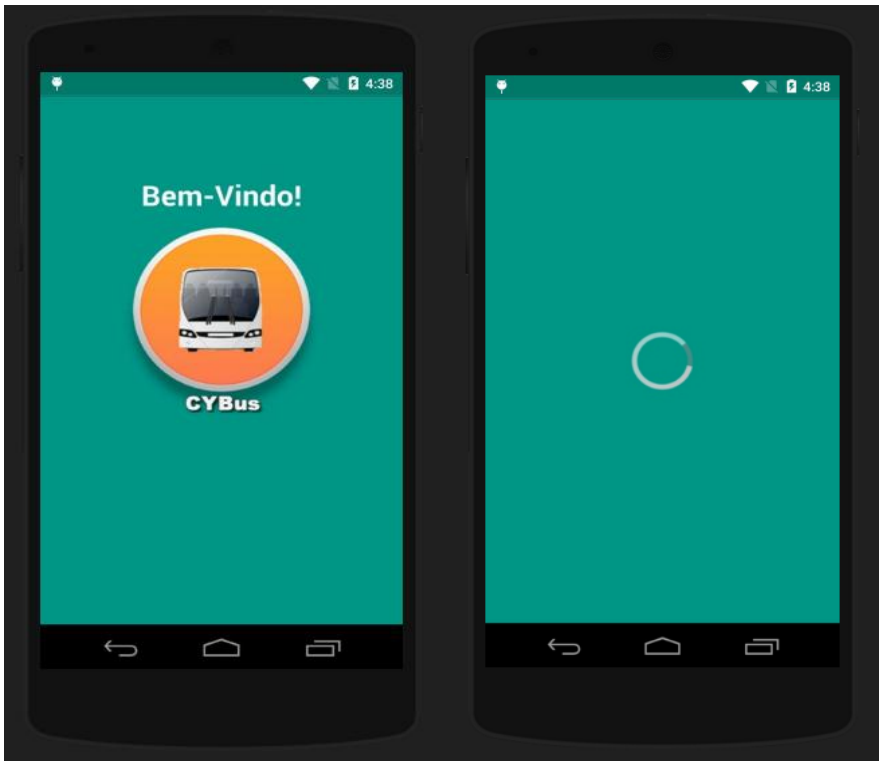
Para projetar os *wireframes* das telas do projeto, foi utilizada a ferramenta *Gliffy*, uma ferramenta online que pode ser utilizada tanto para criação de diagramas diversos, quanto para criação de *wireframes* de telas. Encontra-se disponível em: <gliffy.com>.

4.4 Telas do aplicativo

A seguir serão apresentadas algumas telas do aplicativo, e detalhes das funcionalidades.

As primeiras telas exibidas quando o aplicativo é executado são as telas de animação de abertura, mostradas na Figura 16:

Figura 16 – Tela de boas vindas e carregamento



Fonte: O Autor (2016).

As telas de aberturas servem para fazer uma breve apresentação do aplicativo, deixa-lo mais elegante e atrair a atenção do usuário. Assim que o usuário executa o aplicativo, a tela de boas vindas é exibida e fica ativa por três segundos, tempo suficiente para que o usuário possa visualizar a mensagem de boas vindas. Logo

em seguida é mostrada a tela de carregamento, informando ao usuário que o aplicativo está sendo carregado. Sempre que o usuário abrir o aplicativo essas telas serão mostradas.

Assim que o aplicativo é carregado pela primeira vez, ou caso o usuário se perca a sessão do aplicativo, uma tela para que o usuário escolha entre logar ou cadastrar é exibida:

Figura 17 – Tela de opções entre *login* ou cadastro

Fonte: O Autor (2016).

A tela contem dois botões que poderão redirecionar o usuário para a tela de *Login* ou tela de Cadastro no aplicativo. Também contem um menu de opções, na parte de baixo da tela, onde o usuário poderá acessar os canais de comunicação do aplicativo.

Logo que o usuário instala o aplicativo, como ele ainda não possui um cadastro no servidor, então se faz necessário que ele excute o cadastro. A Figura 18 ilustra o formulário de cadastro de usuários:

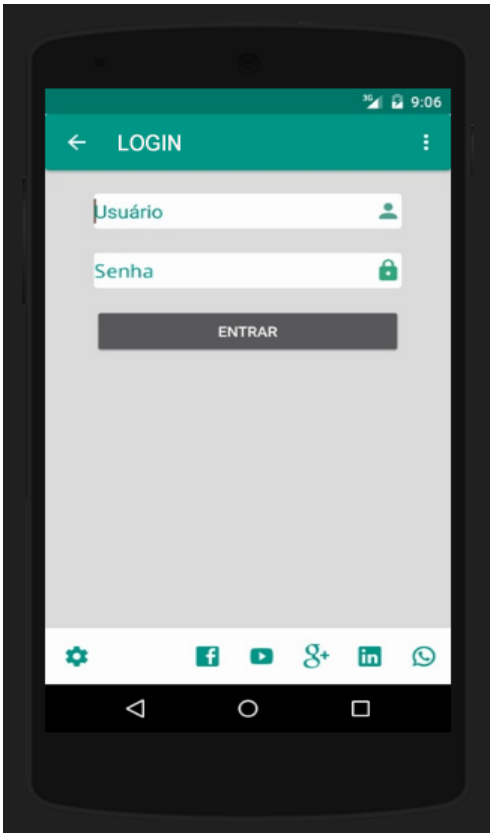
Figura 18 – Tela do formulário de cadastro

Fonte: O Autor (2016).

O formulário contem três campos de texto: um para a inserção do nome de usuário para acesso ao aplicativo, um para cadastro de senha e outro para cadastro de e-mail do usuário, para que ele possa receber notificações de novidades através do e-mail. Assim que o usuário preenche o formulário com todos os dados ele deve pressionar o botão “Concluir” para submeter às informações para o servidor.

Depois que o usuário já possui um cadastro efetuado, ele pode efetuar *login* no aplicativo. A Figura 19 mostra a tela responsável pelo gerenciamento de *login* do usuário:

Figura 19 – Tela de *Login*



Fonte: O Autor (2016).

Para manter um nível de segurança foi criada a tela de *Login*, na qual o usuário pode inserir seus dados de autenticação, e sincroniza-los com o servidor. A partir deste momento, as informações passam pelo processo de verificação de credenciais de usuário, e caso os dados sejam autênticos aos cadastrados, o usuário é direcionado para a tela principal do aplicativo. Porém, caso as credenciais de usuário não correspondam com as que estão cadastradas no servidor, então o usuário é redirecionado para a tela de cadastro, para que possa efetuar o cadastro.

Depois que o usuário efetua o *login* no aplicativo, ele é redirecionado para a tela inicial, a qual é exibida na Figura 20, mostrada abaixo:

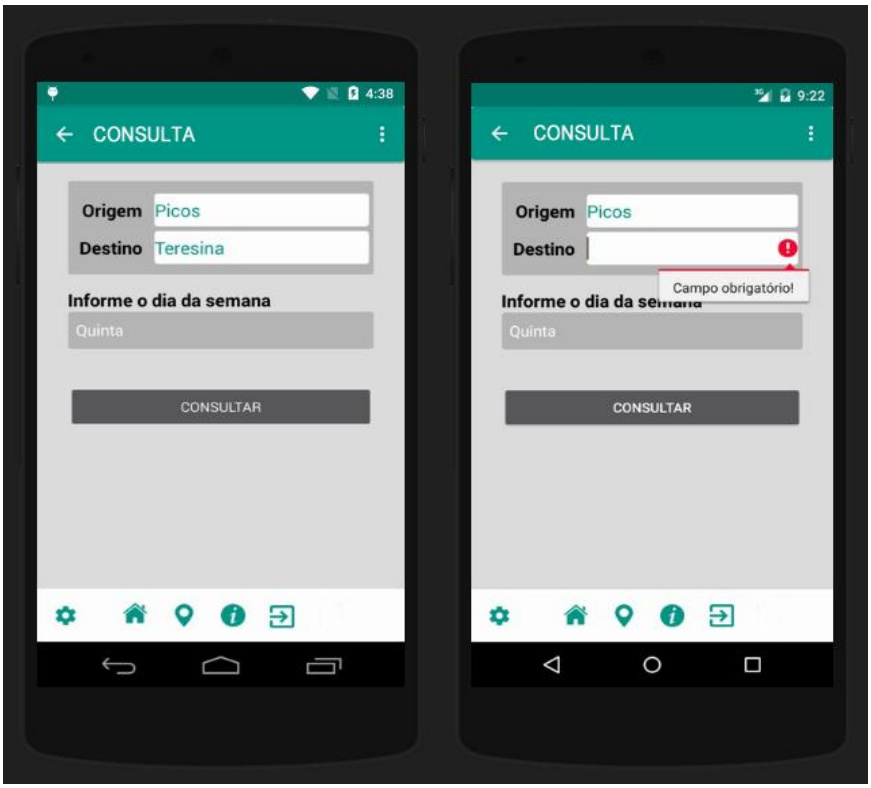
Figura 20 – Tela inicial do aplicativo

Fonte: O Autor (2016).

A tela contem um menu de opções representados por ícones, para visualizar a descrição de cada opção, basta o usuário pressionar o dedo sobre o ícone e segurar, então a descrição é mostrada flutuando sobre a tela. O ícone com a lupa representa a opção de consulta às informações das linhas de ônibus, o ícone ao lado da lupa representa a opção de carregamento do aplicativo do mapa, o ícone com o “i” representa a opção de visualização de informações sobre o aplicativo, o ícone com a seta em direção à direita representa a opção de fechamento do aplicativo.

A tela responsável por controlar o serviço principal do aplicativo é a tela de consulta, mostrada na Figura 21. Nela existe um formulário para a submissão dos parâmetros de consulta, contem um campo de texto onde o usuário pode informar o nome da cidade em que ele está no momento atual da consulta, outro campo de texto para a inserção do nome da cidade que ele pretende viajar, um componente *combobox* que contém os dias da semana onde o usuário pode escolher o dia que ele pretende viajar, e um botão para submeter os dados de solicitação ao servidor.

Figura 21 – Tela de consultas



Fonte: O Autor (2016).

Foi implementada uma função para validação dos campos de texto do aplicativo. Assim que o usuário pressionar o botão de consultar, se ele por descuido deixar algum campo sem preencher, será exibida uma mensagem informando que o campo é obrigatório.

Depois que os dados são submetidos para a consulta, o servidor, processa as informações e retorna uma lista contendo informações dos ônibus que viajam para o destino pretendido pelo usuário, a hora de saída, possível hora de chegada e valor da tarifa cobrada pela empresa, como pode ser visto na Figura 22.

Figura 22 – Tela de resultados

Fonte: O Autor (2015).

Para fins de testes, foram cadastradas no servidor, informações fictícias de algumas empresas de ônibus.

5 CONCLUSÕES E TRABALHOS FUTUROS

Através da realização deste trabalho, foi possível obter um grande conhecimento a respeito das técnicas de desenvolvimento de aplicativos para dispositivos móveis, além de vários conceitos que podem ser aplicados para desenvolvimento de *softwares* em geral, além de aplicar os conhecimentos, adquiridos nas disciplinas do curso de graduação, no desenvolvimento de um aplicativo real que poderá ser utilizado para proporcionar mobilidade às pessoas que utilizam ônibus intermunicipais.

O presente trabalho destacou o crescimento do mercado para dispositivos móveis, a importância da aplicação dessas tecnologias na vida das pessoas, proporcionando uma maior mobilidade, e a importância da plataforma *Android* para o crescimento do cenário da computação móvel atual.

Também, foram destacadas as novas ferramentas disponíveis atualmente para auxilio ao desenvolvimento de aplicativos para a plataforma *Android*. Portanto, o conteúdo disponível neste trabalho será bastante útil para as pessoas interessadas em desenvolvimento de aplicativos móveis.

Como trabalhos futuros, pretende-se implementar algumas melhorias ao projeto, como: opção de avaliação dos serviços prestados pelas empresas, opção de interesse de passagens a partir do aplicativo, e opção de agendamento.

REFERÊNCIAS

ALECRIM, Emerson. **Google Material Design**. Disponível em: <https://tecnoblog.net/158821/google-material-design/>. Acesso em: 04 de Jan. de 2016.

ALLEN, Grant. ***Beginning Android***: *Get started building apps for the Android platform*. *Fifth Edition*. *Apress*, 2015.

ANDROID. **História do *Android***. Acesso em: 05 de Nov. 2015. Disponível em: <https://www.android.com/intl/pt-BR_br/history/>.

ANDROID DEVELOPER. ***Honeycomb***. Documento eletrônico, Disponível em: <http://developer.android.com/intl/pt-br/about/versions/android-3.0-highlights.html>. Acesso em: 07 de Nov. 2015.

ADROID DEVELOPER. ***Android* Studio**. Disponível em: <http://developer.android.com/intl/pt-br/sdk/index.html>. Acesso em 27 de Dez. 2015

BANES, Chris. **AppCompat v21 - Material Design for Pre-Lollipop Devices**. Disponível em: <http://android-developers.blogspot.com.br/2014/10/appcompat-v21-material-design-for-pre.html>. Acesso em: 06 de Jan. de 2016.

BIRANI, Bhanu. ***Application Development with Parse using IOS SDK*: *Development the backendof your applications instantly using Parse iOS SDK. Packt Publishing***: Mumbai, 2013.

BURTON, Michael; FELKER, Donn. **Desenvolvimento de Aplicativos *Android* para Leigos**, Tradução da 2° Edição. Rio de Janeiro: Alta *Books*, 2014.

CARLOS, J. Costa. **Desenvolvimento para *Web***. Lisboa: Lusocrédito, 2007.

COULOURIS, George; DOLLIMORE, Jean; KINDBERG, Tim. **Sistemas Distribuídos, Conceitos e Projeto**, 5ª ed. São Paulo: Bookman, 2013.

DEITEL, Harvey. ***Android* para Programadores:** Uma Abordagem Baseada em Aplicativos, 2ª ed. São Paulo: Bookman, 2015.

DEITEL, Harvey. ***Android*: Como programar**, 2ª ed. São Paulo: Bookman, 2015.

EIS, Diego. **Anotações sobre Material Design do Google**. Disponível em: <http://tableless.com.br/anotacoes-sobre-material-design-google/>. Acesso em: 04 de Jan. de 2016.

GOOGLE. **Google Design**. Disponível em: <https://www.google.com/design/spec/style/color.html>. Acesso em: 04 de Jan. de 2016.

GOOGLE. **Material Design - Introdução**. Disponível em: <https://www.google.com/design/spec/material-design/introduction.html#introduction-principles>. Acesso em 05 de Jan. de 2016.

GOUVEIA, Daniel. ***Java* em Rede:** Programação Distribuída na Inernet. Rio de Janeiro:Brasport,2008.

GUIMARÃES, G. **História do sistema operacional *Android***. Acesso em: 5 de Novembro de 2015, disponível em: PetNews: <http://www.dsc.ufcg.edu.br/~pet/jornal/agosto2013/materias/historia_da_computacao.html>

GRADLE. ***Gradle Release Notes***. Disponível em: <https://docs.gradle.org/current/release-notes>. Acesso em: 03 de Dez.de 2015.

IDC. ***Smartphone OS Market Share*, 2015 Q2**. Disponível em: <http://www.idc.com/prodserv/smartphone-os-market-share.jsp>. Acesso em: 04 de Nov. 2015.

INTEL. **Diálogo TI**. Intel e IDC revelam tendências da mobilidade corporativa no Brasil. Acesso em: 12 de Nov. 2015. Disponível em: <http://dialogoti.intel.com/pt-br/documento/intel-e-idc-revelam-tendencias-da-mobilidade-corporativa-no-brasil>.

LECHETA, Ricardo R. **Google *Android*:** Aprenda a criar aplicações para dispositivos móveis com *Android* SDK, 4º ed. São Paulo: Novatec, 2015.

NOLAN, G.; CINAR, O.; TRUXALL, D. ***Android Best Practices***. *Apress*, 2014.

NEGREIROS, R. **Breve histórico das versões do *Android***. Acesso em: 20 de Nov. 2015. Disponível em: <http://www.estudonasnuvens.com.br/tem-doce-novo-da-google-o-marshmallow-chegou/>

ORTIZ, Henrique. **Serviços de *Parse* baseado em nuvem para aplicativos *Android***. Disponível em: <http://imasters.com.br/desenvolvimento/servicos-de-parse-baseado-em-nuvem-para-aplicativos-android/>. Acesso em: 10 de Jan. de 2016.

PARSE. ***About Parse***. Disponível em: <https://www.parse.com/about>. Acesso em: 10 de Jan. de 2016.

PEREIRA, Lúcio Camilo Oliva; SILVA, Michel Lourenço. ***Android* para desenvolvedores**. Rio de Janeiro. Brasport, 2009.

PINTO, Pedro. **O novo IDE de desenvolvimento para *Android***. Disponível em: <http://pplware.sapo.pt/smartphones-tablets/android/android-studio-ja-esta-disponivel-para-download/>. Acesso em 27 de Dez. 2015.

PRESSMAN, Roger. **Engenharia de *software*, Uma abordagem profissional**, 7 ed. São Paulo: Bookman, 2011.

SOUZA, Thiago. ***Android*, Arquitetura e Desenvolvimento**. Trabalho de conclusão de curso. Pontifícia Universidade Católica de Minas Gerais, Poços de Caldas, 2009.

STEVEN, Kin. ***Mobile Enterprise***. Disponível em:<http://asmarterplanet.com/mobile-enterprise/blog/2014/11/become-mobile-enterprise.html>. Acesso em: 07 de Nov. de 2015.

STEPPAT, Nico. **Bancos de dados não relacionais e o movimento *NoSQL***. Disponível em: <http://blog.caelum.com.br/bancos-de-dados-nao-relacionais-e-o-movimento-nosql/>. Acesso em: 10 de Jan. de 2016.

TÁRCIO, Zemel. Ferramentas para criação de *wireframes*. Disponível em: <http://desenvolvimentoparaweb.com/ux/ferramentas-criacao-wireframes/>. Acesso em: 15 de Jan. de 2016.

VAQUERO, L. M.; MERINO-RODERO, L.; CACERES, J.; LINDNER, M. ***A Break in the Clouds:*** *Towards a Cloud Definition*. *ACM SIGCOMM Computer Communication Review*, 39(1): 50-55, 2009.

APÊNDICE A – Código do *Login*

Classe responsável pelo gerenciamento de *login* do usuário, junto ao servidor.

Figura 23 – Código do *Login*

```
    /*************************************************************
     *                      Autenticação                         *
     *************************************************************/
    edt_user = (EditText) findViewById(R.id.edt_user);
    edt_senha = (EditText) findViewById(R.id.edt_senha);
    bt_entrar = (Button) findViewById(R.id.bt_logar);
    bt_entrar.setOnClickListener(new OnClickListener() {

        @Override
        public void onClick(View v) {
            /*-------------------------
            *    VALIDAÇÃO DOS CAMPOS
            **-------------------------*/
            if(edt_user.getEditableText().toString().isEmpty() && edt_senha.getEditableText().toString().isEmpty()){
              edt_user.setError("Campo obrigatório!");
              edt_senha.setError("Campo obrigatório!");
            }else if(edt_user.getEditableText().toString().isEmpty() || edt_senha.getEditableText().toString().isEmpty()){
                if(edt_user.getEditableText().toString().isEmpty()){
                    edt_user.setError("Campo obrigatório!");
                }else{
                    edt_senha.setError("Campo obrigatório!");
                }
            } else {
                login();
            }
        }
    });
   //-------------------------------------------------
}

private void login(){
    final ProgressDialog dialog = new ProgressDialog(LoginActivity.this);
    dialog.setMessage("Autenticando... aguarde");
    dialog.setIndeterminate(false);

    dialog.show();

    String usuario = edt_user.getEditableText().toString().trim();
    String senha = edt_senha.getEditableText().toString().trim();

    ParseUser.logInInBackground(usuario, senha, new LogInCallback() {
            @Override
            public void done(ParseUser user, ParseException e) {
                if(user != null){
                    dialog.dismiss();
                    Toast.makeText(LoginActivity.this, "Usuário logado com sucesso!", Toast.LENGTH_SHORT).show();
                    Intent intent = new Intent(LoginActivity.this, BuscaActivity.class);
                    startActivity(intent);
                    finish();
                }else{
                    Toast.makeText(LoginActivity.this, "Usuário incorreto ou ainda não foi cadastrado!", Toast.LENGTH_SHORT).show();
                    Intent intent = new Intent(LoginActivity.this, RegisterActivity.class);
                    startActivity(intent);
                    //finish();
                }
            }
    });
}
```

APÊNDICE B – Classe de Consulta às informações

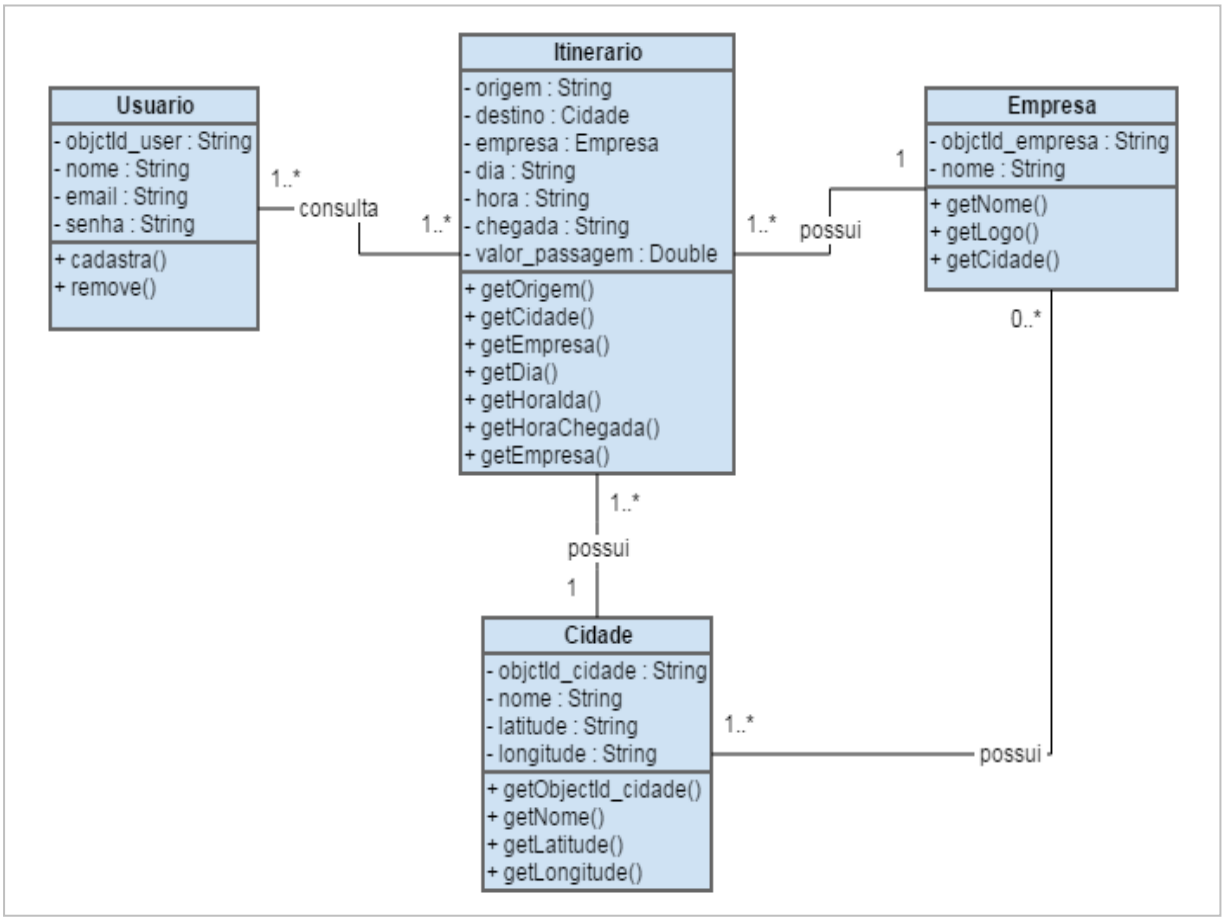
Classe que contem os códigos que realizam a consulta ao servidor para obter as informações, e carregamento das informações no *listView*.

Figura 24 – Classe de consulta aos dados

```
public class CustomAdapter extends ParseQueryAdapter<Itinerario> {

    public CustomAdapter(Context context, final String origem,final String destino, final String ida) {
        super(context, new ParseQueryAdapter.QueryFactory<Itinerario>() {
            public ParseQuery create() {

                // Consulta destino na tabela cidade
                ParseQuery cidadeQuery = ParseQuery.getQuery(Cidade.class);
                cidadeQuery.whereEqualTo("Nome", destino);

                ParseQuery query = ParseQuery.getQuery(Itinerario.class);
                //Acesse o cache primeiro e, em seguida, carregue da rede.
                query.setCachePolicy(ParseQuery.CachePolicy.CACHE_THEN_NETWORK);

                query.whereEqualTo("origem", origem);

                /*Faz a comparação entre as chaves primária da tabela Cidade
                 com a sua chave estrangeira na tabela Itinerario. */
                query.whereMatchesKeyInQuery("destino","objectId", cidadeQuery);

                query.whereEqualTo("dia", ida);
                query.include("destino");
                query.include("Empresa");
                return query;
            }
        });
    }
    // Customização do ItemView
    @Override
    public View getItemView(Itinerario object, View v, ViewGroup parent) {
        if (v == null) {
            v = LayoutInflater.from(getContext()).inflate(R.layout.list_item_layout, null);
        }

        super.getItemView(object, v , parent);

        //Carregando as imagens para a lista
        ParseImageView icone = (ParseImageView) v.findViewById(R.id.parse_image_view);
        ParseFile imageFile = object.getParseObject("Empresa").getParseFile("logo");
        if (imageFile != null) {
            icone.setParseFile(imageFile);
            icone.loadInBackground();
        }

        TextView textView = (TextView) v.findViewById(R.id.tv_emp);
        textView.setText("Viação " + object.getParseObject("Empresa").getString("Nome"));

        //Formatação do valor
        DecimalFormat df = new DecimalFormat("#,###.00");

        TextView tv_iti = (TextView) v.findViewById(R.id.tv_detalhes);

        tv_iti.setText(""+"De: "+object.getString("origem")
                        +" à "+object.getParseObject("destino").getString("Nome")
                        +"\n"+"Ida: "+object.getString("dia") +" às "+object.getString("hora")+" horas"
                        +"\n"+"Chagada: "+object.getString("chegada")+" horas"
                        +"\n" +"Valor: R$ "+df.format(object.getNumber("valor")));

        return v;
    }
}
```

APÊNDICE C – Diagrama de classes

Figura 25 – Diagrama de classes



TERMO DE AUTORIZAÇÃO PARA PUBLICAÇÃO DIGITAL NA BIBLIOTECA "JOSÉ ALBANO DE MACEDO"

Identificação do Tipo de Documento

( ) Tese

( ) Dissertação

(X) Monografia

( ) Artigo

Eu, **Rodrigo santos Costa**, autorizo com base na Lei Federal nº 9.610 de 19 de Fevereiro de 1998 e na Lei nº 10.973 de 02 de dezembro de 2004, a biblioteca da Universidade Federal do Piauí a divulgar, gratuitamente, sem ressarcimento de direitos autorais, o texto integral da publicação **Um protótipo de aplicativo remoto para consultas de Itinerários de ônibus intermunicipais**, de minha autoria, em formato PDF, para fins de leitura e/ou impressão, pela internet a título de divulgação da produção científica gerada pela Universidade.

Picos-PI, 10 de Março de 2016.

Rodrigo santos Costa

Assinatura