UNIVERSIDADE FEDERAL DE SANTA CATARINA

CENTRO TECNOLÓGICO

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Protótipo de aplicativo móvel multiplataforma para consulta de estimativas de chegada das linhas de ônibus de Florianópolis

Alan Peruch Casagrande

Vanessa Silva da Conceição

Florianópolis, dezembro de 2014

UNIVERSIDADE FEDERAL DE SANTA CATARINA

CENTRO TECNOLÓGICO

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Protótipo de aplicativo móvel multiplataforma para consulta de estimativas de chegada das linhas de ônibus de Florianópolis

Alan Peruch Casagrande

Vanessa Silva da Conceição

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

Florianópolis, dezembro de 2014

Alan Peruch Casagrande

Vanessa Silva da Conceição

Protótipo de aplicativo móvel multiplataforma para consulta de estimativas de chegada das linhas de ônibus de Florianópolis

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Prof. Dr. Leandro José Komosinski

Banca examinadora

Prof. Dr. Frank Augusto Siqueira

Profª. Dra. Patrícia Vilain

Resumo

A mobilidade urbana é um tema que vem sendo muito discutido em Florianópolis atualmente. Apesar de sua importância, o sistema de ônibus ainda tem problemas na questão de consulta de horários. Existem alguns aplicativos móveis que tornam a consulta mais fácil, porém os horários são referentes apenas às saídas dos terminais. É difícil para um passageiro em um ponto de ônibus saber quais linhas por ali passam e quanto tempo elas levam para chegar.

Essa dificuldade deu origem a este trabalho, que propõe a criação de um protótipo de aplicativo móvel multiplataforma que disponibilize as informações necessárias para os usuários do transporte coletivo de Florianópolis. Quando o usuário estiver em uma parada de ônibus o aplicativo obtém a sua localização através do GPS do dispositivo e fornece as informações das linhas de ônibus juntamente com as respectivas estimativas de chegada.

No estágio atual, o protótipo está pronto para ser testado com as estimativas pré calculadas. As estimativas em tempo real ainda necessitam de uma fonte de dados, tipicamente um serviço Web disponibilizado pelas empresas de ônibus ou prefeitura.

Palavras-chave: Aplicativo móvel, desenvolvimento multiplataforma, S*ingle Page Application*, ônibus, transporte público, mobilidade urbana.

Abstract

Urban mobility is a subject that has been discussed at length in Florianopolis of late. However, passengers continue to face major problems checking their schedules. There are a few mobile applications that make it easier to find information, but the schedules only refer to departure times from bus stations. It is difficult for a passenger at a bus stop to know which lines pass there and how long the next bus will take to arrive.

Therefore, this project aims to create a mobile application prototype to provide the necessary information for public transport passengers in Florianopolis. The application will get the user's location using the device's GPS then provide the relevant information of bus lines and estimated arrival times.

Currently, the prototype is ready for testing with the precalculated estimates. The real-time estimates still need a data source, typically a web service provided by the bus companies or municipal government.

Keywords: Mobile application, cross platform development, Single Page Application, bus schedules, estimated time of arrival (ETA), public transportation, urban mobility.

Lista de Tabelas

Lista de Ilustrações

Lista de Abreviaturas

DOM	Document Object Model
GIS	Geografic Information Systems
GPS	Global Positioning System
MVC	Model-View-Controller
MVVM	Model-View-View-Model
SPA	Single Page Application
UML	Unified Modeling Language

Sumário

1 Introdução

A motivação para o desenvolvimento deste trabalho vem da dificuldade em conseguir informações sobre linhas de ônibus e horários disponíveis para chegar em um determinado destino a partir de uma parada de ônibus em Florianópolis.

Informações sobre linhas e horários geralmente são disponibilizadas nos *sites* das empresas de ônibus, e, no caso de Florianópolis, no *site* da prefeitura de forma mais centralizada.

Entretanto, isso apresenta alguns problemas; Além de ser necessária uma conexão com a internet, esses *sites* geralmente não são preparados para serem apresentados em dispositivos móveis.

Embora já existam aplicativos que apresentam essas informações de uma forma mais amigável, elas são insuficientes para quem utiliza as linhas nas paradas de ônibus, já que é difícil estimar o tempo de chegada do ônibus a partir da saída dos terminais. Além disso, principalmente para usuários ocasionais e turistas, é comum não saber quais linhas passam em uma determinada parada de ônibus.

Para o desenvolvimento do protótipo proposto neste trabalho, a metodologia *Scrum* foi adotada para gerenciar as atividades. Com relação ao desenvolvimento de código, foram utilizadas tecnologias Web difundidas, como HTML, JavaScript e CSS. Além disso, alguns frameworks e bibliotecas também foram utilizados com intuito de facilitar o desenvolvimento, como Leaflet [11], Geolib [1], React [20], PhoneGap [19] e Vagrant [23].

1.1 Objetivos

O objetivo principal deste trabalho é o desenvolvimento de um protótipo de aplicativo móvel que informe o tempo de espera estimado para a chegada das

linhas de ônibus em uma determinada parada, baseando-se na velocidade média previamente calculada de cada trajeto.

O objetivo secundário do trabalho é realizar a estimativa em tempo real, baseando-se na informação de geolocalização de cada ônibus. Atualmente, estes dados não são públicos em Florianópolis. O protótipo irá trabalhar com a hipótese de que eles sejam disponibilizados futuramente.

Entende-se que mobilidade urbana é um assunto muito amplo e não se limita à utilização de ônibus, e que ações de melhoria têm mais chances de serem eficazes após a coleta e análise de dados através de um estudo mais aprofundado. Entretanto, não é objetivo deste trabalho analisar este assunto de forma sistêmica ou apresentar qualquer diagnóstico, mas oferecer uma solução para uma necessidade específica, identificada através da experiência de utilização do sistema de transporte público de Florianópolis.

1.2 Delimitação do problema

De acordo com Juan Carlos Munoz [21], uma pesquisa realizada em Londres em outubro de 2012 apontou que a maior parcela dos usuários utiliza o sistema de ônibus uma vez por semana.

A figura 1.1 mostra que, aproximadamente, 23% dos usuários utiliza o sistema de ônibus uma vez por semana. Resultados similares foram identificados também em Santigo, em 2010, como mostra a figura 1.2.

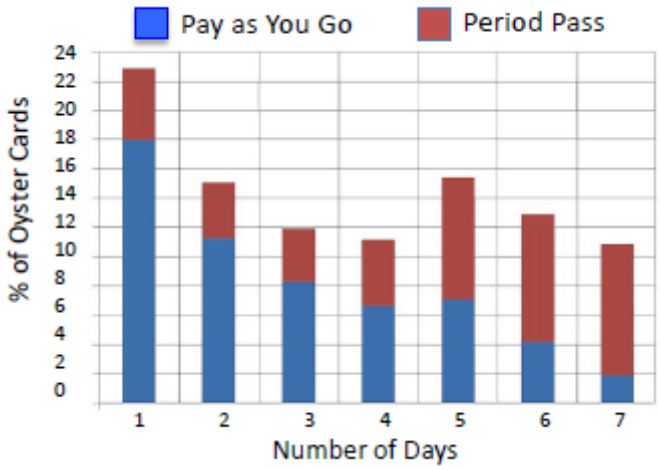


Figura 1.1 - Distribuição da utilização do sistema de ônibus por dia da semana em Londres, 2012.

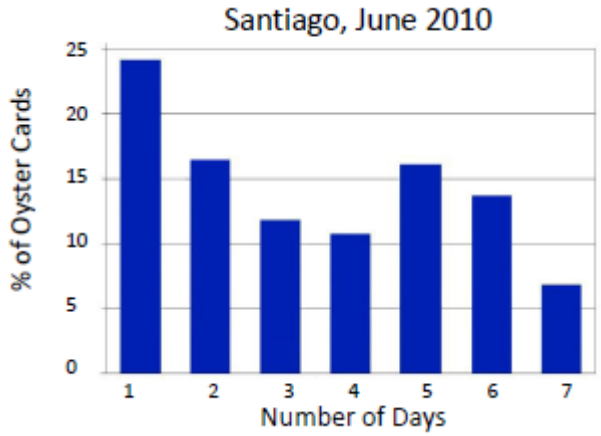


Figura 1.2 - Distribuição da utilização do sistema de ônibus por dia da semana em Santiago, 2010.

Supondo que Florianópolis tenha uma distribuição semelhante às apresentadas, se justifica a necessidade de um aplicativo que auxilie principalmente este perfil de usuário. Evidentemente, um aplicativo que informe sobre estimativas de chegada em uma parada tem mais utilidade para usuários ocasionais, que não têm conhecimento sobre linhas e horários, do que para usuários frequentes.

Atualmente é possível realizar consultas nos *sites* das empresas de ônibus. Além disso, a prefeitura disponibiliza uma página onde é possível pesquisar as linhas de todas as empresas, facilitando ligeiramente a vida do usuário. Porém, realizar esta consulta através de um dispositivo móvel enquanto aguarda o ônibus pode ser frustrante, já que tais *sites* geralmente não são pensados e preparados

para lidar com telas menores.

Para lidar com este problema, surgiram aplicativos que apresentam as informações de uma forma mais amigável, os quais serão mostrados no capítulo 3.

2 Fundamentação Teórica

2.1 Estratégias de desenvolvimento

A grande variedade de dispositivos móveis disponíveis no mercado apresenta um desafio para o desenvolvimento de novos aplicativos: Como gerar e manter aplicações para essa quantidade de dispositivos?

Segundo TRAEG (2013), a resposta para esta questão pode estar no desenvolvimento multiplataforma ou híbrido, o que acaba acarretando em um dilema: desenvolver uma aplicação que tenha melhor experiência de usuário (nativa) ou maior disponibilidade (tecnologias Web)?

2.1.1 Aplicação nativa

Aplicação nativa é aquela desenvolvida especificamente para um determinado tipo de plataforma (iOS, Android, Windows Phone, entre outras).

A principal vantagem dessa abordagem é a possibilidade de uma melhor experiência do usuário, uma vez que para cada plataforma a interface gráfica seguirá exatamente o padrão com o qual o usuário da mesma já está acostumado. Além disso, a interação com funcionalidades específicas do dispositivo, como câmera e GPS, pode apresentar um melhor desempenho do que nas aplicações multiplataforma (FOWLER, 2012).

Por outro lado, esse tipo de abordagem torna difícil a reusabilidade, já que cada plataforma adota suas convenções de interface gráfica, e utiliza uma linguagem de programação diferente das demais. Isso geralmente faz com que seja necessário criar times de desenvolvimento para cada plataforma, ou contratar

profissionais que dominem todas as linguagens necessárias, o que, além de ser raro, aumenta muito o custo por serem bastante valorizados.

Pode-se observar que essa abordagem aumenta consideravelmente os custos, uma vez que não se trata somente de implementar várias versões da aplicação, mas também de mantê-las, logo, criar ou alterar uma funcionalidade significa desenvolver e testar em cada uma das plataformas.

Esse tipo de abordagem é mais aconselhado quando se pretende fornecer a funcionalidade para uma plataforma específica, ou quando a experiência do usuário pode comprometer o sucesso do produto.

2.1.2 Aplicação multiplataforma

Aplicação multiplataforma é o caminho para distribuir o código entre diferentes plataformas sem a necessidade de criar diferentes implementações. Consiste em criar uma única aplicação utilizando as tecnologias de desenvolvimento Web (HTML, JavaScript e CSS), que através de ferramentas específicas poderá ser disponibilizada para vários dispositivos (FOWLER, 2012).

Essa abordagem reduz significativamente os custos e o tempo de criação, pois além de permitir o desenvolvimento e manutenção de apenas uma aplicação utiliza tecnologias padrões da Web, amplamente difundidas, evitando a necessidade de conhecimentos específicos de cada plataforma.

Nessa abordagem a interface da aplicação é feita em HTML, o que pode causar um certo estranhamento por parte do usuário, caso este já esteja acostumado com a interface de determinada plataforma. Entra aí a importância de uma interface intuitiva e de boa usabilidade, que siga os padrões das interfaces Web, proporcionando conforto ao usuário ao navegar pela aplicação.

2.1.3 Aplicação híbrida

Aplicação híbrida é um misto entre uma boa experiência do usuário e a praticidade do multiplataforma. Nessa abordagem é possível seguir por dois caminhos, de acordo com a intenção do desenvolvedor.

Segundo as boas práticas de desenvolvimento de software, deve-se separar completamente a interface com o usuário da lógica da aplicação. Sendo assim, é possível desenvolver de forma nativa a interface ou a implementação das funcionalidades.

Caso a experiência do usuário seja a prioridade, é possível melhorá-la criando uma interface para cada tipo de plataforma, seguindo assim as convenções de cada uma. As funcionalidades continuam sendo desenvolvidas em JavaScript, o que permite a reutilização para todas as plataformas (FOWLER, 2012).

Caso o desempenho seja o ponto central, é possível desenvolver funcionalidades, principalmente aquelas ligadas ao acesso às funções do dispositivo (câmera, GPS, mapas), utilizando o código nativo da plataforma, e mantendo a interface multiplataforma através do HTML (FOWLER, 2012).

2.2 Single Page Applications

Uma *Single Page Application* SPA é uma aplicação ou *site* Web que funciona inteiramente em apenas uma página, de forma que todas as interações do usuário são manipuladas com código executado no cliente, sem realizar transição entre documentos. Todo o código interpretado pelo navegador (HTML, CSS, JavaScript, ou um arquivo binário, no caso de um plug-in), necessário para o funcionamento da aplicação, é recuperado normalmente no primeiro acesso. A partir deste momento, as mudanças de estado, que podem envolver comunicação com o

servidor ou não, não se dão pelo carregamento de uma nova página, mas sim pela manipulação dinâmica do documento.

Com a popularização das SPAs, surgiram muitas bibliotecas oferecendo implementação de padrões como MVC e *Model-View-View-Model* (MVVM) com o seu funcionamento todo no cliente.

3 Soluções existentes

Nesta seção são apresentados alguns aplicativos que auxiliam na consulta de horários e itinerários de ônibus.

3.1 MovelBus

O MovelBus [15] é um aplicativo móvel, atualmente disponível apenas para dispositivos que utilizam o sistema Android, que permite aos usuários consultar os horários de linhas de ônibus através da seleção da empresa de ônibus e do nome ou código da linha desejada. Está disponível em diferentes versões para as cidades de Florianópolis, Blumenau, Biguaçu e Porto Alegre.

O MovelBus permite que a consulta dos horários seja feita sem a necessidade de conexão com internet, uma vez que durante a instalação do aplicativo todas essas informações são armazenadas no dispositivo.

A figura 3.1 apresenta três telas do aplicativo, a primeira é a tela inicial, onde é possível selecionar a opção “Pesquisar”, escolher o nome da empresa e a linha desejada e seguir para a segunda tela, onde o usuário deve selecionar a origem da linha (terminal) e o dia da semana desejado. Após o preenchimento desses dados a terceira tela é exibida contendo os horários de saída da linha selecionada.

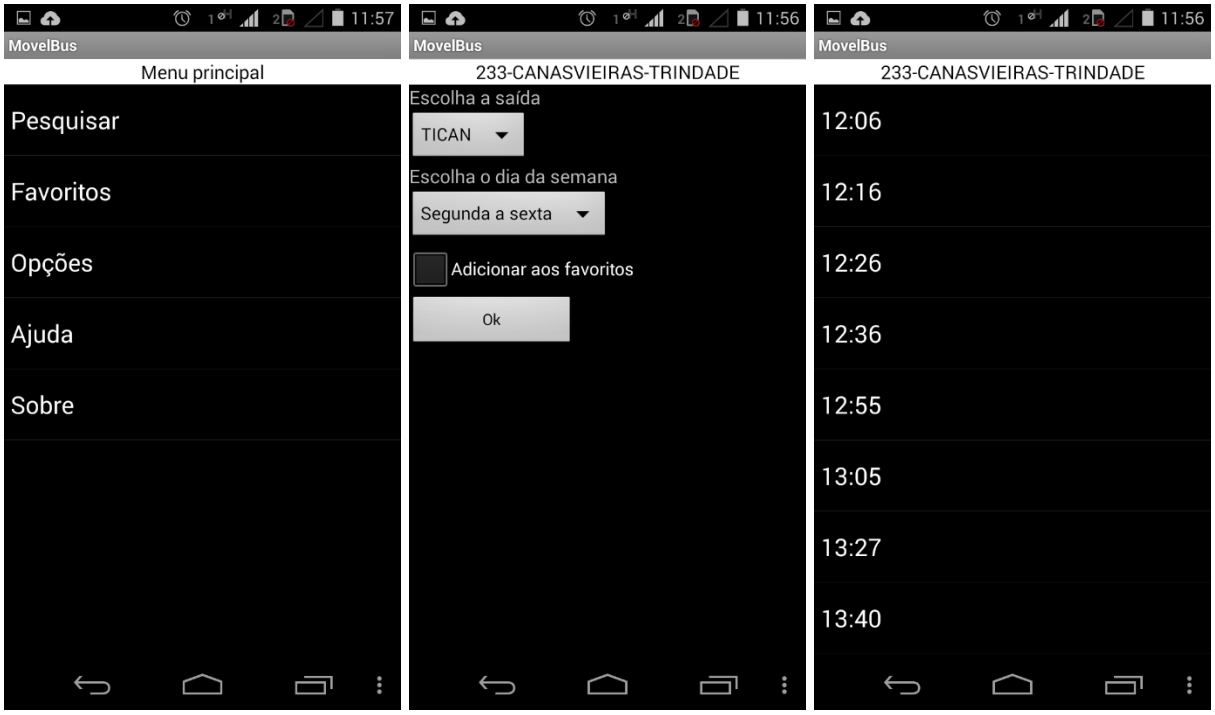


Figura 3.1 – Imagens do aplicativo MovelBus.

3.2 BusMaps

O BusMaps [24] é um aplicativo móvel, também disponível apenas para o sistema Android, que permite ao usuário visualizar a localização atual do ônibus através de um mapa, além de disponibilizar as informações dos horários das linhas.

Atualmente o aplicativo disponibiliza somente informações das linhas de Florianópolis e a consulta das mesmas pode ser feita através do código ou nome da linha desejada.

A posição do ônibus mostrada no mapa é uma aproximação, podendo ocorrer variações de acordo com as condições do trânsito, por exemplo.

Ao instalar ou atualizar o aplicativo, as informações necessárias para o funcionamento do mesmo são copiadas para o dispositivo do usuário, permitindo a utilização *off-line* do aplicativo.

Em questões de implementação, é o aplicativo que mais se assemelha ao protótipo deste trabalho, pois a estimativa é baseada nas velocidades médias

calculadas previamente para cada linha.

A figura 3.2 ilustra a pesquisa for uma linha de ônibus no BusMaps. Primeiramente o usuário digita o número ou nome da linha desejada e faz a seleção dentre as opções apresentadas. Em seguida, um mapa com a posição atual de todos os ônibus em circulação daquela linha é exibido. Caso queira saber as informações dos horários de saída dos terminais, o usuário pode clicar no ícone correspondente e as informações serão apresentadas como mostrado na terceira tela.

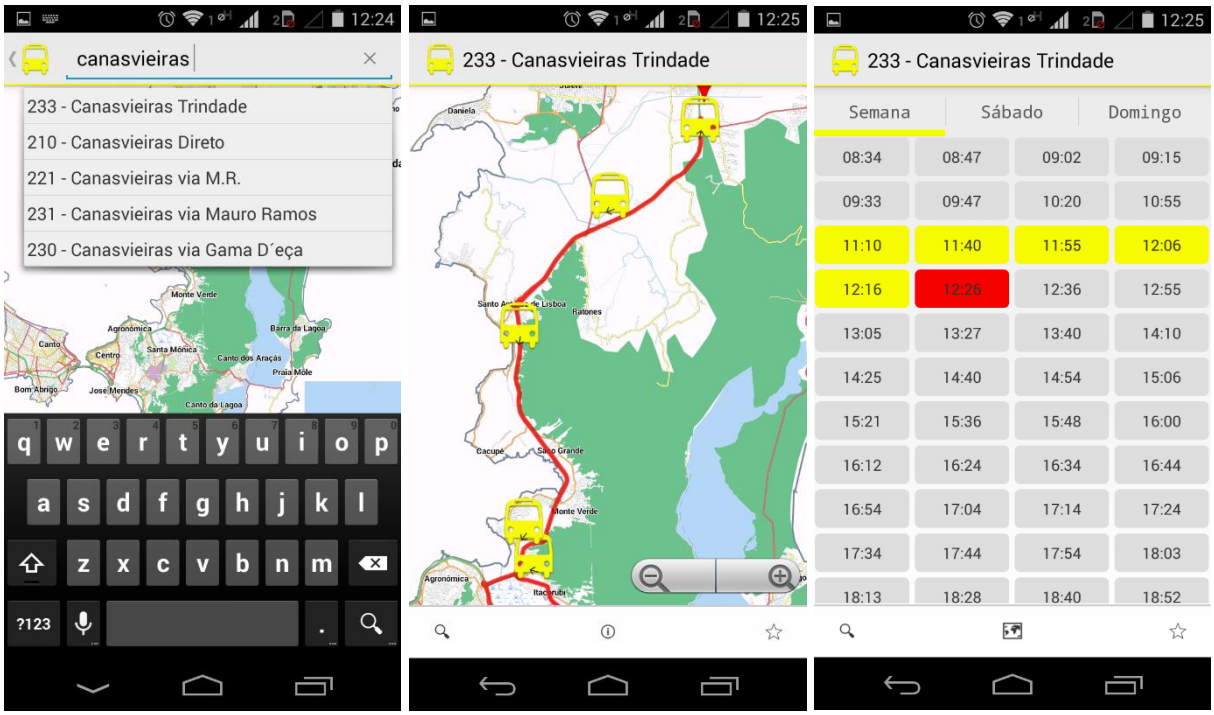


Figura 3.2 – Imagens do aplicativo BusMaps.

3.3 Floripa Ride

O Floripa Ride [5] possui, além do aplicativo móvel, uma versão Web. A versão para Android permite definir o ponto de partida e o destino desejado e apresenta as opções disponíveis para realizar o trajeto, seja ele a pé, de bicicleta, de carro ou utilizando o transporte público. A definição dos pontos de partida e destino pode ser feita através do preenchimento dos respectivos campos ou utilizando

apontamento direto no mapa. Além disso, o aplicativo obtém a localização do usuário através do GPS do dispositivo e essa informação pode ser usada como ponto de partida.

A partir das informações inseridas o aplicativo irá sugerir um trajeto, no qual o usuário poderá ver onde e qual linha de ônibus pegar, além do horário estimado da mesma.

Diferente dos aplicativos apresentados anteriormente, o Floripa Ride requer conexão com internet para sua utilização.

A figura 3.3 ilustra a pesquisa de sugestões de linhas de ônibus para o trajeto definido na primeira tela. Após fornecer as informações de origem e destino, o usuário recebe as instruções de como percorrer o trajeto desejado (segunda tela). Existe a possibilidade de alterar algumas opções relacionadas ao trajeto (terceira tela).

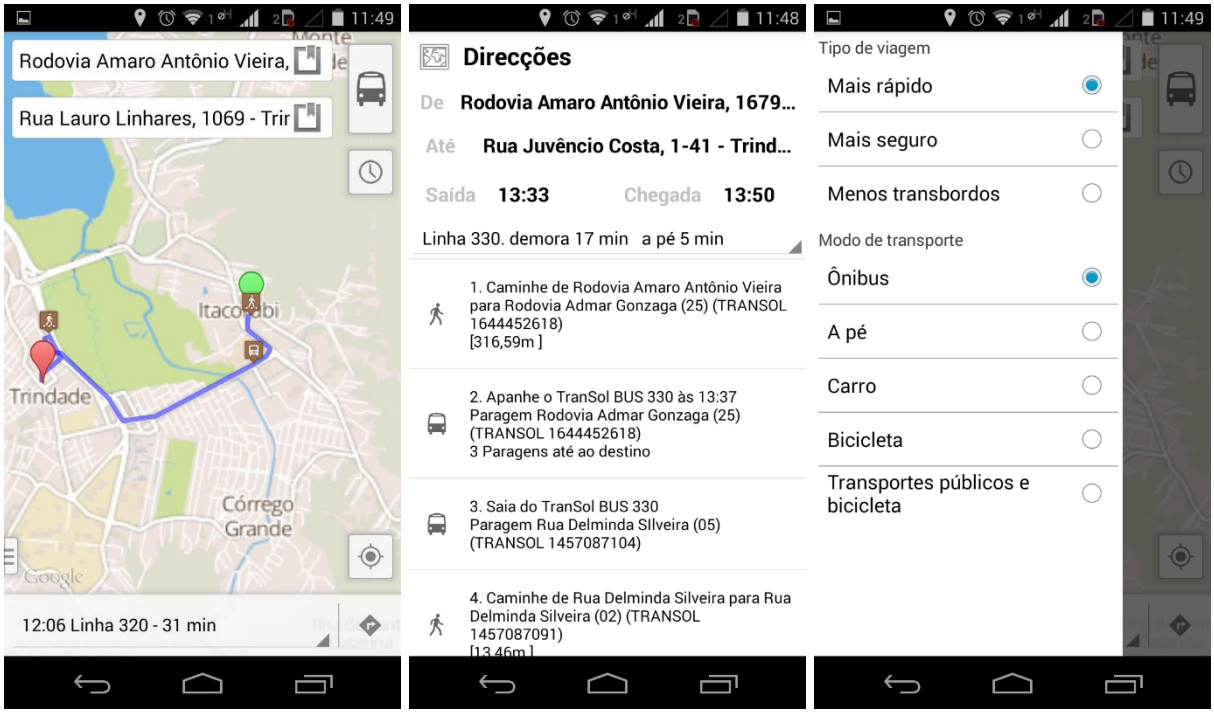


Figura 3.3 – Imagens do aplicativo Floripa Ride.

3.4 Análise das soluções existentes

Do ponto de vista deste trabalho, as soluções apresentadas são limitadas ou incompletas pois possuem uma ou mais das seguintes características:

- São aplicações nativas disponíveis apenas para o sistema operacional Android, o que restringe a base de usuários;
- O usuário não sabe quais são as linhas disponíveis em sua localização;
- Os horários são limitados à saída do ponto de origem, não incluindo o tempo estimado de chegada em cada parada. Desta forma, o usuário não sabe quanto tempo vai esperar pelo ônibus;
- Podem exigir conexão com a internet para realizar uma consulta.

A tabela 3.1 apresenta um comparativo resumido entre as funcionalidades das soluções existentes apresentadas.

Vale destacar que as características funcionais aqui comparadas são somente aquelas ligadas ao transporte coletivo. Funcionalidades dos aplicativos que não correspondem a esse contexto não serão apresentadas nessa tabela.

Aplicativos	**Compatibilidade**	**Requer conexão com Internet?**	**Possui versão web?**	**Funcionalidades**
MovelBus	Android	Não	Não	Horários de saída das linhas de ônibus; Busca por empresa de ônibus e posterior seleção da linha desejada;
BusMaps	Android	Não	Não	Visualização da localização atual aproximada do ônibus e seu trajeto em um mapa; Horários de saída das linhas de ônibus; Busca por nome ou código da linha; Suporte a mapas
Floripa Ride	Android	Sim	Sim	Sugestões de linhas de ônibus a partir da definição dos pontos de origem e

				destino; Itinerário em formato texto; Horários de saída das linhas de ônibus; Identificação da posição do usuário através do GPS; Suporte a mapas

Tabela 3.1 – Comparativo entre soluções existentes

4 Solução proposta

Este trabalho propõe o desenvolvimento de um aplicativo para dispositivos móveis capaz de, basicamente, responder duas perguntas:

1. Quais linhas passam em uma determinada parada de ônibus?
2. Quando tempo elas irão demorar para chegar?

O aplicativo deve oferecer esta informação sem precisar estar conectado à internet.

A fim de atender o maior número possível de dispositivos, foi adotada a estratégia de desenvolvimento multiplataforma baseada em tecnologias Web, que permite utilizar o mesmo código para diferentes sistemas operacionais, como Android, iOS, e Windows Phone. Este assunto vem ganhando a atenção gradual dos desenvolvedores nos últimos anos, de forma que novas práticas e tecnologias surgem e são aperfeiçoadas com frequência.

O protótipo é orientado pela simplicidade e facilidade de utilização. A ideia é fornecer a informação certa com a menor quantidade de interação possível. Para isso, não optou-se pela visualização padrão deste tipo de aplicativo que é o mapa com as rotas traçadas. Em vez disso, o aplicativo identifica a localização do usuário e mostra as linhas do ponto de ônibus mais próximo.

4.1 Requisitos funcionais

4.1.1 Identificar o ponto de ônibus mais próximo

O primeiro passo é identificar a posição atual do usuário através do GPS do dispositivo. Com isso, é possível identificar a parada de ônibus mais próxima e selecioná-la automaticamente.

4.1.2 Listar linhas de ônibus

Após a definição do ponto de ônibus, devem ser exibidas as linhas que por ali passam com a estimativa de chegada de cada uma.

4.1.3 Listar próximos horários estimados de chegada

Selecionando uma linha é possível ver as próximas estimativas de chegada, caso o usuário esteja planejando não pegar o próximo ônibus.

4.1.4 Diagrama de caso de uso

A figura abaixo mostra a representação dos requisitos funcionais em um diagrama de caso de uso em linguagem *Unified Modeling Language* (UML):

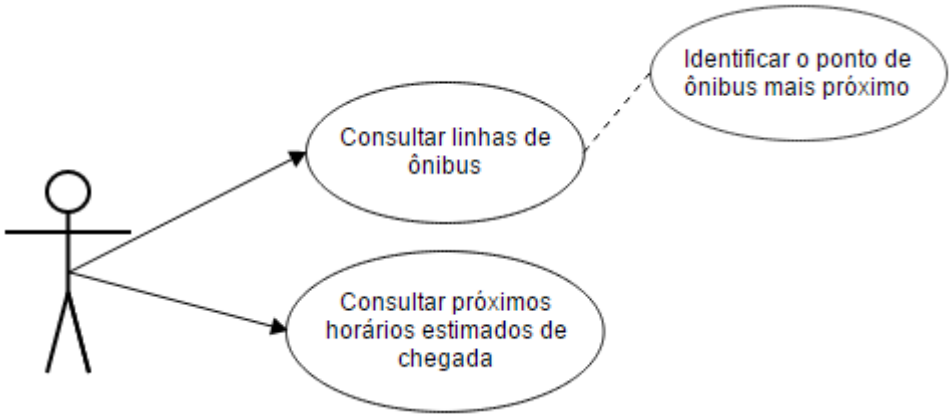


Figura 4.1 - Diagrama de caso de uso

4.2 Requisitos não funcionais

- O protótipo deve funcionar sem conexão com a internet.
- O protótipo pode ser suportado nas três plataformas principais: Android, iOS, e Windows Phone.

4.3 Arquitetura da solução

As figuras a seguir esquematizam o desenvolvimento e a utilização da solução proposta.

A figura 4.2 demonstra a utilização do aplicativo. Quando o usuário consulta as estimativas, o aplicativo tenta recuperar a posição atual dos ônibus através de um serviço. Caso não consiga, seja por falta de conexão com a internet ou porque a posição realmente não está disponível, o aplicativo exibe as estimativas calculadas previamente na etapa de desenvolvimento.

A figura 4.3[1] demonstra a etapa de desenvolvimento, quando os dados são extraídos de fontes externas (OpenStreetMap e MObFloripa) e transformados em estimativas armazenadas em arquivos, entregues junto com o aplicativo.

Figura 4.2 – Esquema de utilização do protótipo.

[1] Apesar de apresentar elementos de um diagrama de fluxo, a figura 4.3 utiliza uma notação livre, apenas para fim de ilustração.

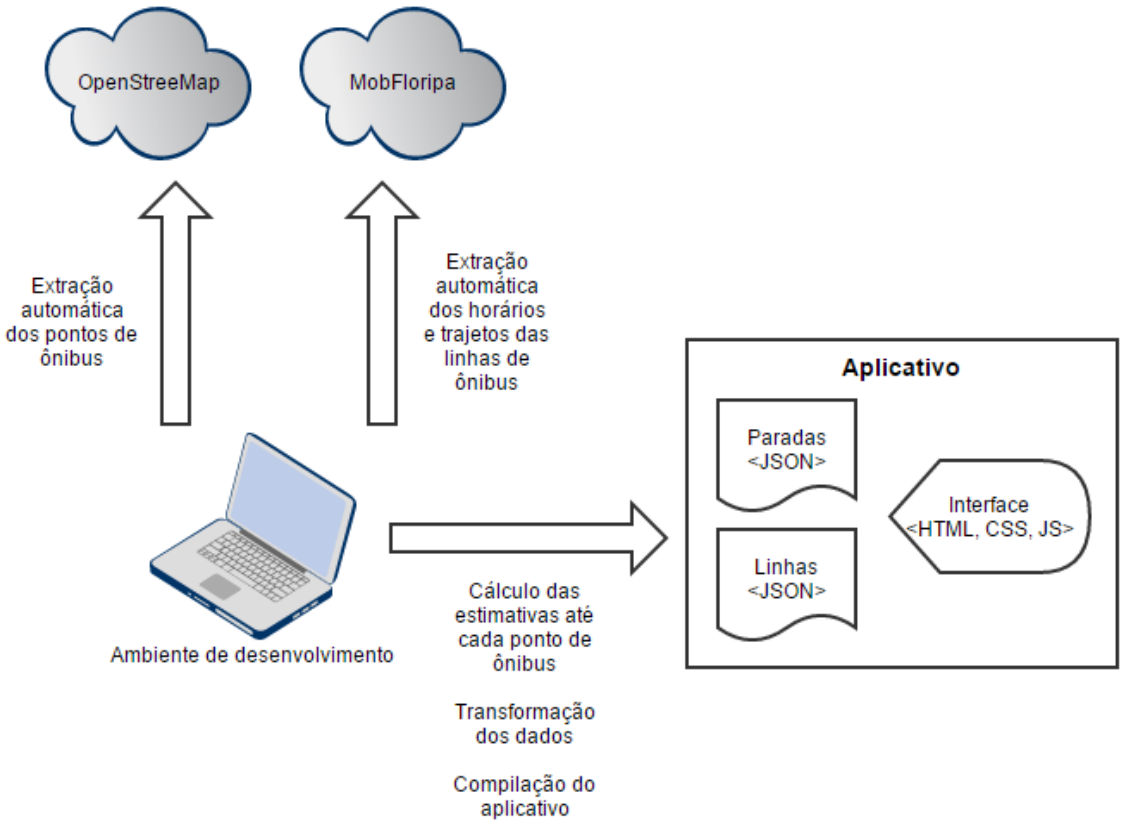


Figura 4.3 – Esquema de desenvolvimento do protótipo.

A etapa prévia de extração e transformação dos dados para o formato JSON torna mais simples a implementação do protótipo. As estimativas de chegada são calculadas para cada linha até cada ponto de ônibus, considerando a velocidade média de cada trajeto. A velocidade média da linha é calculada com as informações extraídas de distância e tempo de duração.

Os componentes de interface apenas consomem os dados transformados, comparando o horário corrente com os horários disponíveis e estimativas de tempo de chegada para informar o tempo restante de espera.

4.4 Metodologia

A metodologia adotada para o desenvolvimento do protótipo foi a de um processo iterativo baseado no *Scrum* [27]. Uma vez levantada as necessidades, foram escritas histórias de usuário e tarefas técnicas para alcançar os objetivos

descritos na seção de funcionalidades. O próximo passo foi priorizar os cartões e iniciar ciclos com duração de uma semana cada até a data de entrega do trabalho.

A cada semana foi feita uma revisão para avaliar o que foi feito e decidir se os cartões ainda estavam coerentes com o objetivo do trabalho.

Para organizar a distribuição das histórias e registrar o progresso foi utilizada a ferramenta *Trello* [7] como mostra a figura 4.4.

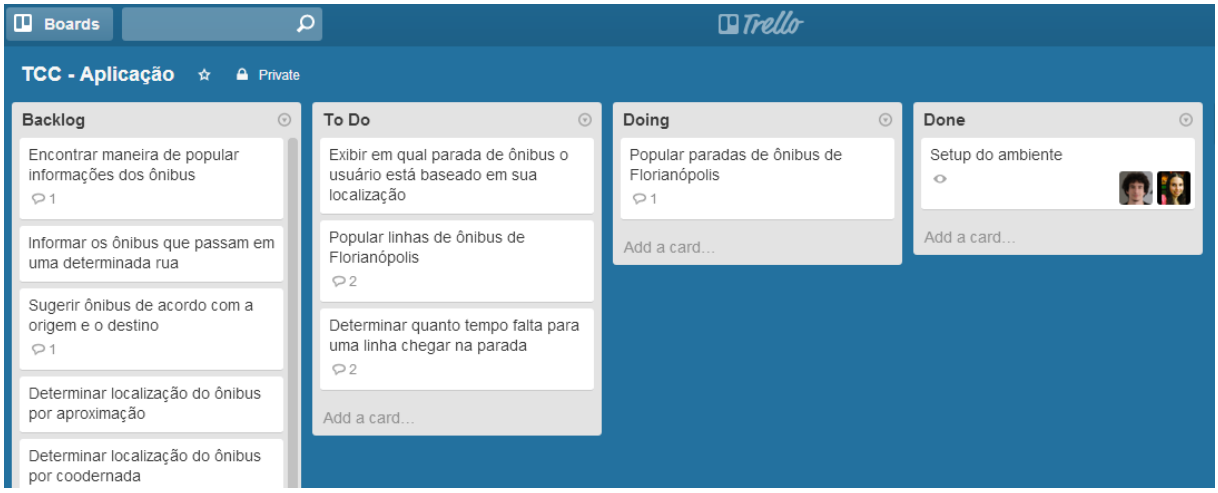


Figura 4.4 – Amostra do quadro de histórias de usuário escritas para o desenvolvimento do protótipo.

4.5 Escolha da estratégia de desenvolvimento

Com o objetivo de tornar a experiência do usuário a mais próxima possível da experiência oferecida por aplicativos nativos, foi utilizado um padrão de desenvolvimento Web SPA.

Como o protótipo não está focado em fornecer uma experiência nativa, mas sim em atingir o máximo de usuários possível com o menor esforço, foi adotada a estratégia de desenvolvimento multiplataforma baseada em JavaScript.

Como o desenvolvimento é baseado completamente em tecnologias Web, é perfeitamente possível distribuir o protótipo como um *site*, para que seja utilizado tanto nos navegadores de dispositivos móveis quanto *notebooks*/*desktops*, sem necessidade de instalação do aplicativo.

5 Desenvolvimento

5.1 Tecnologias utilizadas

Além das tecnologias comuns para desenvolvimento Web (HTML5, JavaScript, e CSS3), também foram utilizadas tecnologias auxiliares para a construção da interface e para a conversão em um aplicativo móvel, bem como tecnologias para melhorar o ambiente de desenvolvimento. Esta seção descreverá brevemente cada uma.

5.1.1 Leaflet

O Leaflet [11] é uma biblioteca JavaScript de código aberto usada para a criação de aplicações Web de mapeamento.

Criada em 2011, esta biblioteca trabalha de forma muito eficiente com as tecnologias web utilizadas para desenvolvimento do protótipo, como HTML5 e CSS3. Juntamente com as bibliotecas OpenLayers [16] e Google Maps API [9] é considerada uma das ferramentas JavaScript de mapeamento mais populares.

Por ser de simples utilização e bem documentada, permite que desenvolvedores sem conhecimento prévio de *Geografic Information Systems* (GIS) consigam, com certa facilidade, criar mapas web e hospedá-los em um servidor público.

Neste trabalho o Leaflet foi utilizado, em conjunto com o Mapbox [13] e o Geolib [1], no desenvolvimento da ferramenta para o mapeamento das paradas de ônibus. Esta ferramenta será detalhada na seção 5.2.1.3 - Mapeamento das paradas de cada linha de ônibus.

5.1.2 React

A biblioteca React [20] é um framework JavaScript utilizado para a construção de interfaces com o usuário, que tem como diferencial o fato de trabalhar com um pseudo *Document Object Model* (DOM) em memória, tratando o problema do gargalo da manipulação do DOM. Ele é comumente considerado como a *View* do padrão *Model-View-Controller* (MVC).

Sabe-se que um dos principais problemas de performance em *sites* e aplicações Web é a manipulação do DOM. O React propõe como solução para este problema a criação de um DOM próprio. Dessa forma, sempre que um elemento precisa ser redesenhado, a biblioteca identifica as alterações comparando o elemento no DOM e na memória e modifica somente aquilo que foi alterado, sem a necessidade de redesenhar todo o elemento.

A criação de componentes reutilizáveis é outro ponto forte do React. Os componentes React possuem dois atributos principais: estado e propriedades. Sempre que o estado de um componente é alterado o React faz uma comparação entre o estado atual e o novo estado, redesenhando somente o que foi alterado. Essa prática é conhecida como JavaScript reativo.

Devido às características mencionadas, o React foi escolhido para a implementação deste protótipo.

5.1.3 Geolib

O Geolib [1] é uma biblioteca que fornece operações geoespaciais básicas. Possui diversas funcionalidades, dentre elas:

- Calcular a distância, em metros, de um determinado caminho através de uma lista de coordenadas;

- Dada uma lista de coordenadas, encontrar a mais próxima de determinado ponto;
- Ordenar uma lista de coordenadas de acordo com a distância até determinado ponto.

Esta biblioteca foi utilizada no desenvolvimento do protótipo e na implementação da ferramenta de mapeamento de paradas de ônibus.

5.1.4 PhoneGap

O PhoneGap [19] é um framework para a criação de aplicações móveis multiplataforma. Ele permite escrever a aplicação uma única vez utilizando as tecnologias web já conhecidas, tais como HTML, JavaScript e CSS, e implantá-la em uma variedade de dispositivos móveis sem perder as funcionalidades de uma aplicação nativa.

O PhoneGap foi utilizado com objetivo de disponibilizar o protótipo para dispositivos que possuam como sistema operacional Android, iOS ou Windows Phone.

5.1.5 Vagrant

O Vagrant [23] é uma ferramenta para criação e configuração de ambientes de desenvolvimento virtuais. Com ele é possível gerenciar máquinas virtuais através de uma interface de linha de comando.

Desde a versão 1.1 ele não é mais atrelado ao VirtualBox [18] e consegue trabalhar com outros softwares de virtualização, como o VMware [25].

O Vagrant permite a criação de uma máquina virtual baseada em uma imagem previamente disponibilizada, como, por exemplo, o Ubuntu 14.04. Quando a

máquina virtual inicia, o Vagrant estabelece uma conexão entre ela e a máquina física através de uma pasta compartilhada (diretório raiz do projeto). Desta forma, é possível utilizar o sistema operacional e editor de código de sua preferência, mas executar o código em um ambiente virtual e controlado, sem a necessidade de instalar dependências na máquina física.

Além disso, é possível automatizar o provisionamento da máquina virtual com uma ferramenta chamada Chef [2], que permite instalar *runtime*, banco de dados, e outras dependências automaticamente quando a máquina é iniciada. Desta forma, todos os desenvolvedores do projeto podem executar o código em um ambiente com as mesmas características.

5.2 Implementação

5.2.1 Aquisição dos dados

Esta seção descreve o processo da aquisição dos dados, descrevendo em detalhes o que cada *script* faz. Para executá-los todos de uma só vez e atualizar paradas e linhas de ônibus, basta executar o comando “node data/update_data.js” dentro do diretório raiz do código fonte.

5.2.1.1 Terminais e paradas de ônibus

Os dados referentes às paradas de ônibus e terminais foram extraídos do projeto OpenStreetMap [17], uma plataforma aberta e colaborativa de mapeamento urbano. Existe uma iniciativa de um grupo de usuários [4] de Florianópolis para mapear tudo o que for relevante à cidade, incluindo informações de transporte público.

No *site* é possível selecionar o mapa de transporte público e exportar

manualmente todos os elementos de uma determinada área no formato XML, como mostra a figura 5.1. Este arquivo contém diversos tipos de dados, como paradas, terminais, linhas, e trajetórias. O *script* **data/extract_bus_stops.js** faz o *download* do arquivo automaticamente através da API fornecida pelo OpenStreetMap, e torna os dados utilizáveis transformando-os para o formato JSON. A URL utilizada é <http://overpass-api.de/api/map?bbox=-48.6186,-27.8634,-48.3275,-27.3742>, onde o parâmetro *bbox* representa as coordenadas aproximadas da área de Florianópolis.

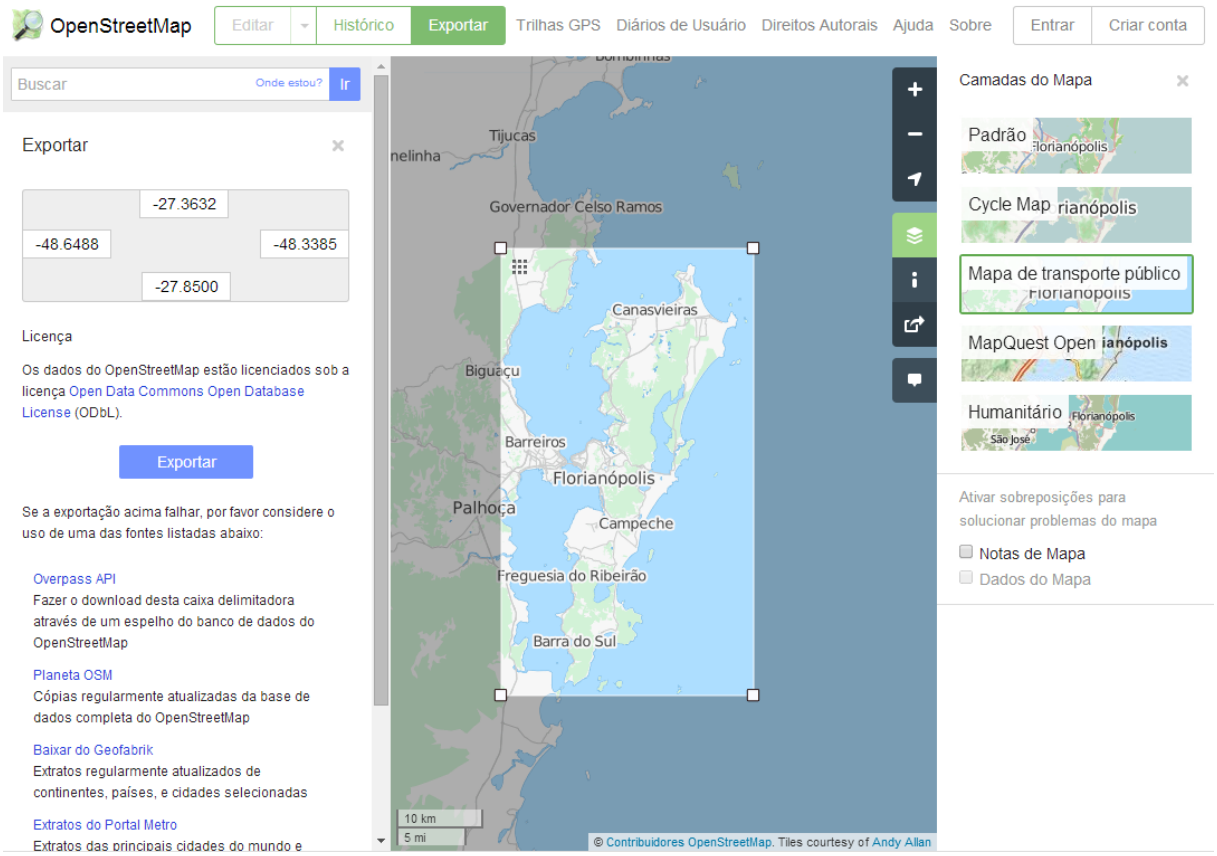


Figura 5.1 – Ferramenta de extração de dados do OpenStreetMap.

Para paradas foram extraídas as informações de nome e localização através da latitude e longitude. Na notação do OpenStreetMap, uma parada de ônibus é um *node* (ponto no mapa).

```xml
<node id="872980957"
  lat="-27.5822448"
  lon="-48.5262412"
  version="3"
  timestamp="2011-12-10T13:25:13Z"
  changeset="10080944"
  uid="506085"
  user="Koyaani">

  <tag k="bench" v="yes"/>
  <tag k="highway" v="bus_stop"/>
  <tag k="name" v="Rua Lauro Linhares (16)"/>
  <tag k="shelter" v="yes"/>
  <tag k="source" v="Bing"/>
  <tag k="source:name" v="survey"/>
</node>
```

Figura 5.2 – Amostra dos dados de uma parada de ônibus no formato XML.

Apesar de na notação do OpenStreetMap o terminal ser um *way* (caminho no mapa) composto por várias coordenadas representadas pelos elementos *nd*, utilizou-se apenas a primeira delas pois o caminho não é relevante para a aplicação.

```xml
<way id="72944878" version="7" changeset="21090086" uid="463504">
  <nd ref="865711626"/>
  <nd ref="865711627"/>
  <nd ref="865711629"/>
  <nd ref="1877678136"/>
  <nd ref="1877678137"/>
  <nd ref="865711636"/>
  <nd ref="865711626"/>
  <tag k="addr:housenumber" v="s/n"/>
  <tag k="addr:street" v="Avenida Professor Henrique da Silva Fontes"/>
  <tag k="amenity" v="bus_station"/>
  <tag k="landuse" v="commercial"/>
  <tag k="name" v="TITRI - Terminal Trindade"/>
  <tag k="short_name" v="TITRI"/>
</way>
```

Figura 5.3 – Amostra dos dados de um terminal de ônibus no formato XML.

O mapeamento das paradas de ônibus de Florianópolis é um trabalho em progresso, de modo que podem existir paradas que não estejam disponíveis no aplicativo.

5.2.1.2 Linhas de ônibus, horários e rotas

O projeto OpenStreetMap já possui linhas de ônibus mapeadas com seus respectivos pontos de parada. Entretanto, o número é muito restrito. Atualmente,

apenas 29 das 210 linhas estão mapeadas, sem previsão de avanço. Além disso, algumas linhas estão fora do padrão estabelecido na *wiki* do projeto (linhas circulares mapeadas como dois ou mais trajetos diferentes, por exemplo)

Deste modo, decidiu-se extrair do *site* MObFloripa [14] as informações das linhas, horários, e coordenadas dos trajetos, através do *script* localizado em **data/extract_bus_lines.js**.

A extração ocorre em várias etapas. Primeiramente, o *script* navega até a página <http://www.mobfloripa.com.br/onibus.php> e extrai toda a lista de linhas de ônibus. Em seguida, navega na página de cada linha, através da URL <http://www.mobfloripa.com.br/linha_det.php?codigo=[codigo]> e extrai as informações de horário para dias de semana, sábados e domingos, assim como as informações de distância e tempo estimado para a conclusão do percurso. Com isso é possível calcular a velocidade média do ônibus, que pode variar bastante dependendo do trajeto.

Em seguida, o *script* navega até o trajeto desenhado no Google Maps, figura 5.4, e baixa o arquivo no formato KML, através da URL. O arquivo contém apenas uma lista de coordenadas e a cor utilizada para representá-las, como mostrado na figura 5.5. O *script* é encerrado quando todas as linhas são processadas e o arquivo **data/files/bus_lines.json** é criado. Somente na etapa seguinte, descrita na próxima seção, é que determinamos para cada linha qual cor representa a ida e qual cor representa a volta.

Figura 5.4 – Trajeto de uma linha de ônibus desenhado no Google Maps pelo *site* MObFloripa.

```
<Style id="style8">
  <LineStyle>
    <color>73FF0000</color>
    <width>5</width>
  </LineStyle>
</Style>
<Placemark>
  <name>Rua Proc. Abelardo Gomes</name>
  <description><![CDATA[]]></description>
  <styleUrl>#style8</styleUrl>
  <LineString>
    <tessellate>1</tessellate>
    <coordinates>
      -48.553497,-27.599380,0.000000
      -48.553280,-27.599541,0.000000
      -48.552780,-27.599945,0.000000
      -48.552666,-27.600042,0.000000
      -48.552494,-27.600121,0.000000
      -48.552349,-27.600157,0.000000
      -48.552208,-27.600157,0.000000
      -48.552006,-27.600140,0.000000
      -48.551815,-27.600077,0.000000
      -48.551708,-27.600039,0.000000
      -48.551498,-27.599876,0.000000
      -48.551147,-27.599485,0.000000
      -48.550995,-27.599483,0.000000
      -48.550808,-27.599518,0.000000
      -48.550663,-27.599562,0.000000
    </coordinates>
  </LineString>
</Placemark>
```

Figura 5.5 – Amostra de um arquivo KML que representa o trajeto de uma linha de ônibus.

5.2.1.3 Mapeamento das paradas de cada linha de ônibus

Com os dados das linhas devidamente transformados, o próximo passo é estabelecer as paradas de cada linha. Para isso, foi desenvolvida uma ferramenta especificamente para este projeto, a figura 5.6 mostra a interface com o usuário da mesma.

A ferramenta exibe todos os pontos de ônibus de Florianópolis em um mapa. Escolhendo uma das linhas, o trajeto (ida/volta ou circular) é traçado e é possível escolher as paradas correspondentes. Os terminais também são considerados como paradas de ônibus.

A velocidade média do ônibus é calculada com as informações de distância e duração de cada trajeto extraídas anteriormente. Quando uma parada é selecionada, a distância do terminal até a mesma é calculada, e, através da velocidade média, é possível determinar quanto tempo o ônibus leva para chegar naquela parada.

Quando o mapeamento de uma linha está completo, é possível copiar os dados em texto no formato JSON manualmente. Os dados são armazenados manualmente no arquivo **data/files/mapped_routes.json**, ilustrado parcialmente na figura 5.7, que é usado para atualizar os arquivos **data/files/bus_lines.json** e **data/files/bus_stops.json**. Para isso, é preciso executar o *script* **data/merge_mapped_routes_and_bus_stops.js**, que também otimiza e copia os arquivos para **app/data/bus_lines.json** e **app/data/bus_stops.json** para serem compilados junto com o aplicativo.

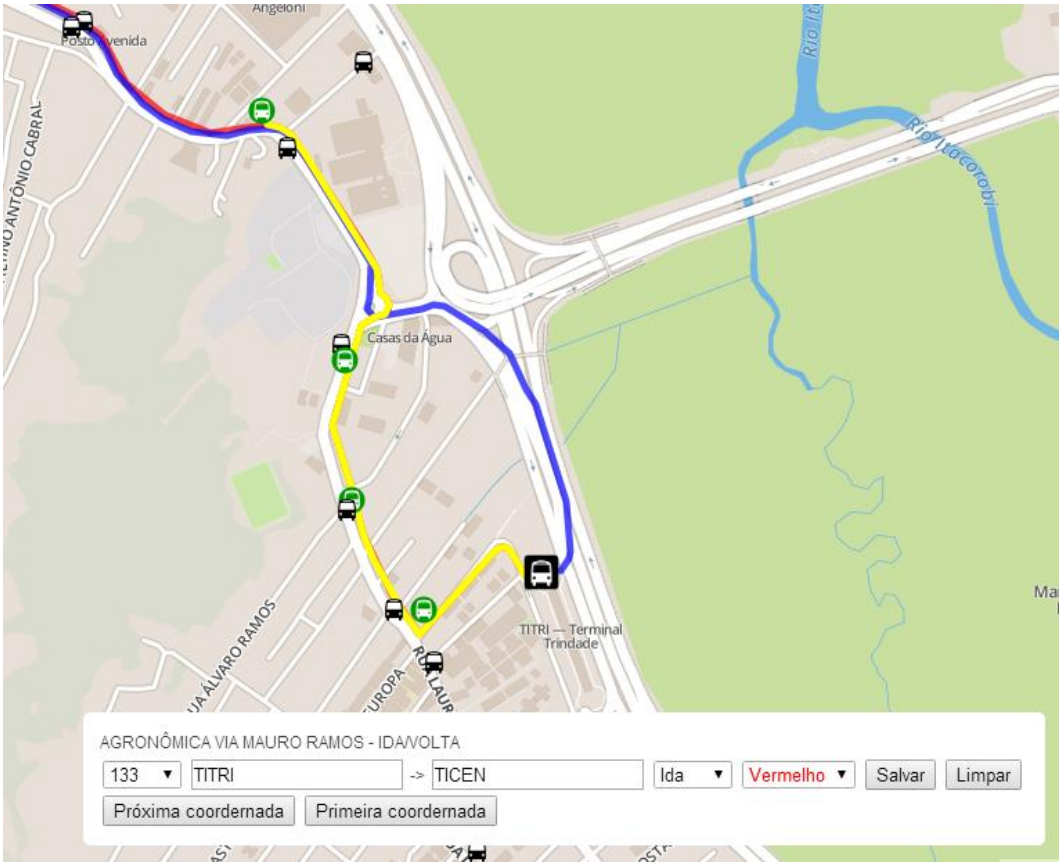


Figura 5.6 – *Interface* da ferramenta para mapeamento das paradas de ônibus.


```
{
  "number": "133",
  "direction": "going",
  "color": "red",
  "stops": [
    {
      "osmId": "2177867806",
      "order": 0,
      "distance": 0.299,
      "estimative": 0.9583333333333333
    },
    {
      "osmId": "872980957",
      "order": 1,
      "distance": 0.534,
      "estimative": 1.7115384615384617
    },
    {
      "osmId": "1639803069",
      "order": 2,
      "distance": 0.723,
      "estimative": 2.3173076923076925
    },
    {
      "osmId": "1457087092",
      "order": 3,
      "distance": 1.208,
      "estimative": 3.8717948717948723
    }
  ]
}
```


Figura 5.7 – Amostra dos dados de uma linha com as paradas mapeadas.

5.2.1.4 Descrição dos dados

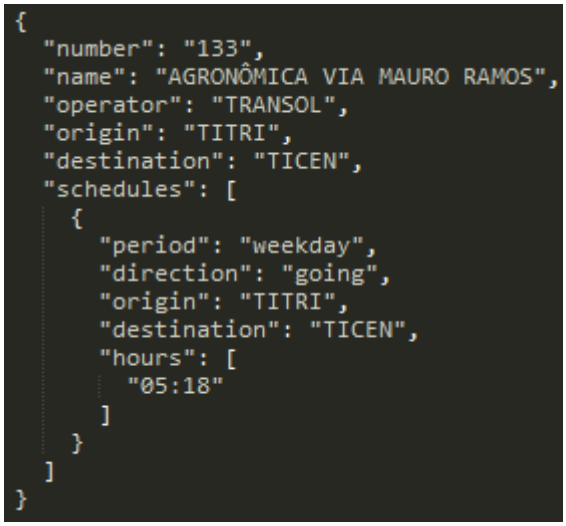
Os componentes de interface esperam que os dados estejam nos formatos descritos nas tabelas 5.1 e 5.2.

Nome	Tipo	Descrição	Exemplo
number	String	Número da linha	233
name	String	Nome da linha	Beira-mar norte
operator	String	Operador da linha	Transol
origin	String	Origem	TICEN
destination	String	Destino	TITRI
schedules	Array<Object>	Horários	
schedules.period	String	Período do horário	weekday, saturday, sunday
schedules.direction	String	Direção do horário	going, returning
schedules.origin	String	Ponto de origem do horário	TICEN
schedules.destination	String	Ponto de destino do horário	TITRI
schedules.hours	Array<String>	Horários de saída	["12:37", "13:05"]

Tabela 5.1 – Formato dos dados de uma linha de ônibus.

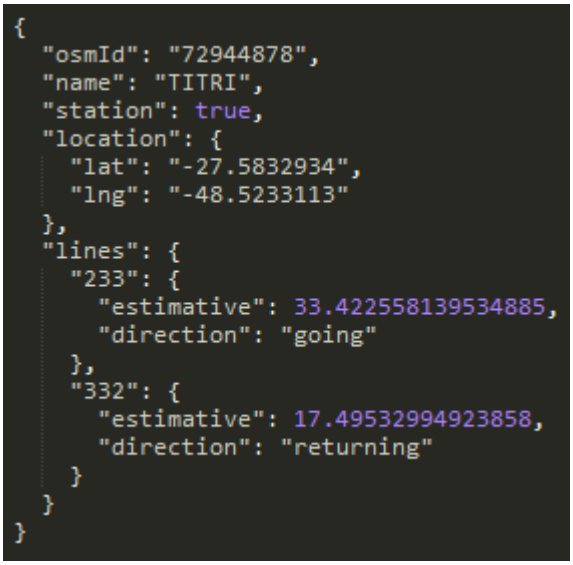
Nome	Tipo	Descrição	Exemplo
osmId	String	Identificador único	65972
name	String	Nome	Rua Lauro Linhares (7)
station	Boolean	Indica se é ou não um terminal	false
location	Object	Coordenadas do ponto	
location.lat	String	Latitude	-27.5982286
location.lng	String	Longitude	-48.5546876
lines	Object	Linhas de ônibus que passam na parada	
lines.<number>.esimative	Number	Tempo estimado de chegada (minutos)	8,5 (8 minutos e 30 segundos)
lines.<number>.direction	String	Direção da linha	going

Tabela 5.2 – Formato dos dados de um ponto de ônibus.



```
{
  "number": "133",
  "name": "AGRONÔMICA VIA MAURO RAMOS",
  "operator": "TRANSOL",
  "origin": "TITRI",
  "destination": "TICEN",
  "schedules": [
    {
      "period": "weekday",
      "direction": "going",
      "origin": "TITRI",
      "destination": "TICEN",
      "hours": [
        "05:18"
      ]
    }
  ]
}
```


Figura 5.8 – Exemplo de uma linha de ônibus no formato JSON.



```
{
  "osmId": "72944878",
  "name": "TITRI",
  "station": true,
  "location": {
    "lat": "-27.5832934",
    "lng": "-48.5233113"
  },
  "lines": {
    "233": {
      "estimative": 33.422558139534885,
      "direction": "going"
    },
    "332": {
      "estimative": 17.49532994923858,
      "direction": "returning"
    }
  }
}
```


Figura 5.9 – Exemplo de um ponto de ônibus no formato JSON.

5.2.2 Aplicativo

Como mencionado anteriormente, o protótipo foi desenvolvido como uma SPA, com o objetivo de causar a impressão de que o aplicativo é nativo da plataforma, e não uma página Web. Durante muito tempo, neste tipo de desenvolvimento, se aplicou a ideia de que o *template* (HTML) e o comportamento (JavaScript) devessem ficar separados, a fim de separar a lógica da apresentação.

A biblioteca React, usada no protótipo para a construção da interface, questiona essa ideia e argumenta que HTML e JavaScript são partes do mesmo

componente, e devem permanecer juntos (no mesmo arquivo). É uma ideia controversa e polêmica, mas que traz benefícios quando aplicada junto com a filosofia de desenvolvimento da biblioteca.

A filosofia de desenvolvimento do React enxerga a aplicação como um conjunto de componentes simples que se modificam de acordo com o seu estado. Cada mudança faz com o que o componente se redesenhe, a fim de refletir o novo estado. Os atributos que não mudam no decorrer do tempo, são considerados propriedades do componente.

Idealmente, a aplicação deve ser separada em componentes de forma que cada um tenha apenas uma responsabilidade. Além disso, a documentação do React recomenda como boa prática que os componentes tenham a menor quantidade possível de variáveis que influenciem o seu estado, e que, quando possível, o estado seja gerenciado pelos componentes de maior nível. Isto garante que os componentes permaneçam simples e fáceis de testar unitariamente.

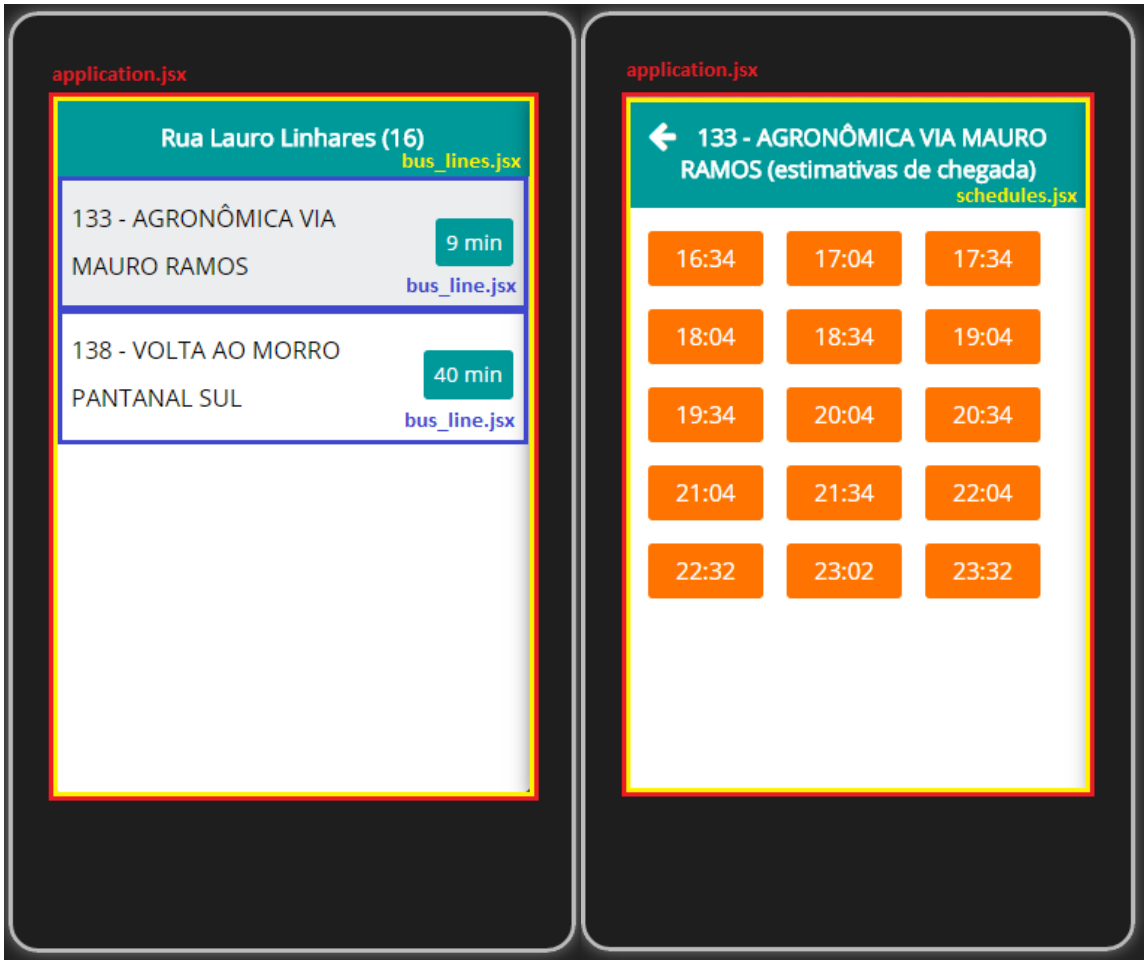


Figura 5.10 – Componentes do aplicativo.

A figura 5.10 mostra as telas do aplicativo e seus componentes. O código fonte que acompanha este documento pode ser consultado para mais detalhes sobre a implementação.

É possível perceber que a aplicação possui um componente de nível mais alto chamado application.jsx. Este componente possui basicamente três variáveis que determinam o seu estado:

1. Parada de ônibus onde se encontra o usuário (Rua Lauro Linhares 16): determina quais linhas de ônibus serão listadas pelo componente bus_lines.jsx.

2. Hora atual: atualizada a cada 1 minuto para recalcular as estimativas.

3. Linha de ônibus: se houver alguma linha selecionada, o componente schedules.jsx, que informa os próximos horários, é exibido ao invés do componente bus_lines.jsx.

O componente bus_lines.jsx não possui estado. Ele apenas recebe a parada de ônibus e decide quais linhas serão exibidas. Para cada linha, uma instância do componente bus_line.jsx é criada. Este componente se preocupa apenas em desenhar uma única linha, e, portanto, também não possui estado.

```
define([
  'react',
  'src/next_arrivals'
],
function (React, getNextArrivals) {

  return React.createClass({
    render: function() {
      var now = this.props.now;
      var busStop = this.props.busStop;
      var busLine = this.props.busLine;
      var onClick = this.props.onClick;

      var nextArrivals = getNextArrivals(now, busLine, busStop, true);
      var nextArrival = nextArrivals[0] || { arrival: 'sem horário' };

      return (
        <div className="bus-line" onClick={onClick.bind(this, busLine)}>
          <div className="bus-name">{busLine.number} - {busLine.name}</div>
          <div className="next-arrival">
            <div>
              {nextArrival.arrival}
              {nextArrival.isRealTime && <span className='fa fa-bus'></span>}
            </div>
          </div>
        </div>
      );
    }
  });
});
```

Figura 5.11 – Implementação do componente que desenha uma linha de ônibus.

5.3 Mapeamento das linhas

Atualmente, as seguintes linhas de ônibus estão mapeadas:

Número	Nome
133	AGRONÔMICA VIA MAURO RAMOS
233	CANASVIEIRAS/TRINDADE
332	SANTO ANTÔNIO VIA BEIRA MAR
138	VOLTA AO MORRO PANTANAL SUL
137	VOLTA AO MORRO PANTANAL NORTE
185	UFSC - SEMIDIRETO
154	UFSC - SEMIDIRETO - SAÍDA SUL

Tabela 5.3 – Linhas de ônibus mapeadas.

Por se tratar de um protótipo, não há necessidade de mapeamento de todas as linhas existentes para demonstrar o seu funcionamento.

5.4 Desafios enfrentados

5.4.1 Terminais

Da forma como a solução foi modelada, os terminais são considerados como pontos de ônibus, de forma que, sem tratamento, apenas os ônibus que passam pelo terminal com origem de outro lugar seriam mostrados.

Como solução para este problema, o atributo *station* (apresentado no modelo de dados) é verificado para determinar se as linhas com origem no terminal devem também ser mostradas na lista.

5.4.2 Feriados

Em caso de feriados, os horários considerados para as estimativas devem ser os praticados no domingo. Para tanto, a aplicação precisa ter conhecimento dos feriados nacionais e municipais para aplicar as estimativas de forma correta.

A melhor solução para essa questão seria, provavelmente, a criação de um serviço Web responsável por fazer a atualização dos feriados quando o usuário estivesse conectado à internet.

Porém, como não é objetivo deste trabalho criar uma solução para a questão dos feriados e existem apenas quatro feriados nacionais que sofrem variação de data, optou-se por criar uma lista simples, no formato JSON, contendo todas as datas dos feriados nacionais e municipais para o ano corrente. Esta lista deve ser atualizada anualmente com as novas datas para os feriados não fixos.

5.4.3 Meia-viagem

As linhas de ônibus são categorizadas geralmente como circular (horários

apenas de ida) ou radial (horários de ida e de volta). Algumas linhas, entretanto, possuem os dois comportamentos: possuem horários de ida e de volta, mas alguns horários de ida realizam a volta completa. Estas linhas são categorizadas como “circular-radial”.

Isto afeta diretamente a estimativa de chegada, pois o aplicativo precisa estar ciente de que alguns horários não irão passar por todas as paradas do trajeto.

A forma como o aplicativo lida com isso é considerando dois trajetos diferentes, como em uma linha radial. Um ponto de ônibus que aguarda os horários de saída do bairro, precisa verificar também os horários de saída do terminal que fazem a volta completa. No último caso, a estimativa de chegada é calculada somando-se o tempo de chegada do terminal até o bairro, e do bairro até o ponto de ônibus. No caso da estimativa em tempo real, os dois trajetos são mesclados para calcular distância percorrida e a velocidade média.

5.5 Problemas não tratados e requisitos adicionais

5.5.1 Precisão ao determinar posição do ônibus

O trajeto percorrido por um ônibus é representado por uma lista de coordenadas, que formam um caminho. No caso da estimativa baseada na informação de geolocalização do ônibus, para descobrir onde o ônibus se encontra neste caminho é necessário descobrir qual coordenada da lista está mais próxima da coordenada reportada pelo ônibus.

Isto é um problema em linhas circulares que passam pela mesma rua/avenida/rodovia na ida e na volta, pois a coordenada mais próxima na lista pode ser a do lado oposto da rua/avenida/rodovia. Isto acontece porque, nas retas, os pontos ficam muito distantes um do outro, já que são necessários apenas dois

pontos para representá-las.

É possível aumentar a precisão aumentando o número de coordenadas do trajeto. Uma maneira de fazer isso é iterando nas coordenadas existentes e criando uma nova coordenada a cada duas usando: “nova_latitude = (latitude1 + latitude2) / 2” e “nova_longitude = (longitude1 + longitude2) / 2”.

Entretanto, este problema foi considerado como fora do escopo do projeto e deve ser tratado em trabalhos futuros.

5.5.2 Implantação em Windows Phone e iOS

O aplicativo não foi implantado e testado nas plataformas Windows Phone e iOS devido à impossibilidade de conseguir os dispositivos necessários. Entretanto, é possível fazê-lo com facilidade seguindo as instruções para cada plataforma no *site* oficial do projeto PhoneGap.

5.5.3 Selecionar ponto de ônibus manualmente

É possível que o usuário queira ver as linhas de outro ponto de ônibus que não seja o mais próximo. Nesse caso a alternativa seria possibilitar a seleção de um ponto diferente no mapa. Através de um novo requisito funcional, o aplicativo poderia mostrar a localização do usuário e os pontos de ônibus disponíveis para seleção.

6 Conclusão

Durante o desenvolvimento deste trabalho foram observadas algumas dificuldades que não haviam sido previstas na etapa de planejamento. A principal delas está liga à aquisição de dados, uma vez que os mesmos são a base para o desenvolvimento do protótipo e não são disponibilizados pelas empresas de ônibus e nem pela prefeitura de maneira uniforme e centralizada, demandando um grande esforço para coletá-los de diferentes fontes, verificar a autenticidade e padronizá-los.

Constatou-se também que a estratégia de desenvolvimento multiplataforma é vantajosa quando não há preocupação em manter os padrões e convenções de *layout* e *design* específicos de cada plataforma. O fato de se utilizar tecnologias Web também é um fator positivo, devido à facilidade para encontrar documentação, soluções para problemas comuns, e padrões de projeto.

A tabela 6.1 foi apresentada no capítulo 3 em uma comparação entre as soluções existentes no mercado que fornecem informações sobre horários, linhas e trajetos dos ônibus. Com intuito de comparar essas soluções com o protótipo desenvolvido ela é novamente apresentada com informações adicionais referentes às funcionalidades implementadas.

Aplicativos	Compatibilidade	Requer conexão com Internet?	Possui versão web?	Funcionalidades
MovelBus	Android	Não	Não	Horários de saída das linhas de ônibus; Busca por empresa de ônibus e posterior seleção da linha desejada;
BusMaps	Android	Não	Não	Visualização da localização atual aproximada do ônibus e seu trajeto em um mapa; Horários de saída das linhas de ônibus; Busca por nome ou código

				da linha; Suporte a mapas
Floripa Ride	Android	Sim	Sim	Sugestões de linhas de ônibus a partir da definição dos pontos de origem e destino; Itinerário em formato texto; Horários de saída das linhas de ônibus; Identificação da posição do usuário através do GPS; Suporte a mapas
Protótipo	Android, iOS, Windows Phone	Não	Sim	Cálculo das estimativas de chegada dos ônibus em uma parada baseado na sua velocidade média; Suporte para estimativas de chegada de acordo com a localização atual do ônibus (GPS); Identificação da posição do usuário através do GPS do dispositivo;

Tabela 6.1 – Comparativo entre soluções proposta e existentes

Através da análise da tabela é possível identificar que o protótipo desenvolvido preenche uma lacuna deixada pelos demais, oferecer ao usuário as informações de estimativa de horário de chegada do ônibus.

Espera-se deste protótipo, com o mapeamento das linhas disponíveis, um impacto positivo na experiência de utilização do sistema de ônibus em Florianópolis, tornando-o mais conveniente para o usuário. Espera-se também que este protótipo estimule outros desenvolvedores a melhorá-lo, e que possa ser utilizado como base para trabalhos futuros.

Destaca-se aqui a importância da publicação de dados da gestão pública, especialmente sobre o transporte coletivo, objeto de interesse deste trabalho. Esta iniciativa já acontece em algumas cidades brasileiras como Rio de Janeiro, e é muito comum em outros países. Desta forma, estimula-se o uso da tecnologia para tornar mais fácil e conveniente a vida da população.

6.1 Trabalhos Futuros - Modelo preditivo para estimativas de chegada

As estimativas apresentadas pelo protótipo desenvolvido neste trabalho se baseiam em uma velocidade média previamente calculada, não considerando fatores externos como trânsito e pico de utilização.

Com dados históricos sobre os trajetos percorridos das diferentes linhas de ônibus em cada horário, é possível treiná-los para gerar um modelo de predição a fim de prever os tempos de chegada em cada horário. Desta forma é possível identificar padrões como congestionamentos em determinados horários, tornando possível o ajuste das estimativas.

Referências

1. BIEH, Manuel. **Geolib**. Disponível em: <https://github.com/manuelbieh/Geolib>. Acesso em: 01 out 2014.

2. CHEF. **How Chef Works**. Disponível em: <https://www.getchef.com/chef>. Acesso em: 01 out 2014.

3. FERREIRA, Davi. **React: JavaScript reativo**. Disponível em: <http://tableless.com.br/react-javascript-reativo/>. Acesso em: 01 out 2014.

4. **Florianópolis**. Disponível em: <http://wiki.openstreetmap.org/wiki/Florian%C3%B3polis>. Acesso em 15 fev 2014.

5. **Floripa Ride**. Disponível em: <http://floriparide.com.br/>. Acesso em: 18 jun 2014.

6. FOWLER, Martin. **Developing Software for Multiple Mobile Devices**. Disponível em: <http://martinfowler.com/articles/multiMobile/>. Acesso em: 06 mar 2014.

7. FOG CREEK SOFTWARE. **Trello**. Disponível em: <https://trello.com/>. Acesso em: 20 ago 2013.

8. GLANZNER, Rafael Audy. **Porque programar para Mobile? Apps nativos ou multiplataforma?** Disponível em: < http://www.devecletico.com.br/porque-programar-para-mobile-apps-nativos-ou-multiplataforma-5/ >. Acesso em: 23 ago 2013.

9. GOOGLE. **API do Google Maps**. Disponível em: <https://developers.google.com/maps/>. Acesso em: 01 out 2014.

10. KOETSIER, John. **HTML5 vs. native vs. hybrid mobile apps: 3,500 developers say all three, please**. Disponível em: <http://venturebeat.com/2013/11/20/html5-vs-native-vs-hybrid-mobile-apps-3500-developers-say-all-three-please/?utm_source=html5weekly&utm_medium=email#vb-gallery:2:862914>. Acesso em: 27 nov 2013.

11. LEAFLET. **Leaflet**. Disponível em: <http://leafletjs.com/>. Acesso em: 01 out 2014.

12. MACLEAN, Malcolm. **Leaflet Tips and Tricks**. Disponível em: <https://leanpub.com/leaflet-tips-and-tricks/read#leanpub-auto-what-is-leafletjs>. Acesso em: 01 out 2014.

13. MAPBOX. **Mapbox**. Disponível em: <https://www.mapbox.com/>. Acesso em: 01 out 2014.

14. MOBFLORIPA. **MObfloripa**. Disponível em: <http://www.mobfloripa.com.br/quemsomos.php>. Acesso em: 24 mai 2014.

15. **MovelBus**. Disponível em: <http://www.movelbus.com.br/>. Acesso em: 18 jun 2014.

16. **OpenLayers 3**. Disponível em: <http://openlayers.org/>. Acesso em: 01 out 2014.

17. **OpenStreetMap**. Disponível em <http://www.openstreetmap.org/about>. Acesso em 15 fev 2014.

18. ORACLE. **Virtual Box**. Disponível em: <https://www.virtualbox.org/>. Acesso em: 18 jun 2014.

19. PHONEGAP. **About the Project**. Disponível em: <http://phonegap.com/about/>. Acesso em: 01 out 2014.

20. REACT. **React**. Disponível em: <http://facebook.github.io/react/>. Acesso em: 01 out 2014.

21. SINCROBUS - SISTEMA DE INFORMAÇÃO E CONTROLE EM TEMPO REAL DE ÔNIBUS URBANOS. 1, 2014, Florianópolis. **Oportunidades para Melhorar a Operação de Sistemas Integrados de Transportes – O caso de Santiago**.

22. TRAEG, Peter. **Best Of Both Worlds: Mixing HTML5 And Native Code**. Disponível em: <http://mobile.smashingmagazine.com/2013/10/17/best-of-both-worlds-mixing-html5-native-code/>. Acesso em: 06 mar 2014.

23. VAGRANT. **About Vagrant**. Disponível em: <https://www.vagrantup.com/about.html>. Acesso em: 01 out 2014.

24. VILLELA, Matheus. **BusMaps Florianópolis**. Disponível em: <https://play.google.com/store/apps/details?id=com.matheusvillela.busmaps>. Acesso em: 18 jun 2014.

25. VMWARE. **Virtualização de desktops e aplicativos**. Disponível em: <http://www.vmware.com/br/products/desktop-virtualization.html>. Acesso em: 03 nov 2014.

26. WIKIPEDIA. **Leaflet (software)**. Disponível em: <http://en.wikipedia.org/wiki/Leaflet_(software)>. Acesso em: 01 out 2014.

27. WIKIPEDIA. **Scrum (software development)**. Disponível em: <http://en.wikipedia.org/wiki/Scrum_(software_development)>. Acesso em: 18 jun 2014.