

**UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
ESPECIALIZAÇÃO EM DESENVOLVIMENTO WEB**

RENNAN MARTINI RODRIGUES

**CONSTRUINDO UMA API PARA EXIBIR HORÁRIOS DE
TRANSPORTE PÚBLICO**

MONOGRAFIA

LONDRINA

2015

RENNAN MARTINI RODRIGUES

**CONSTRUINDO UMA API PARA EXIBIR HORÁRIOS DE
TRANSPORTE PÚBLICO**

Trabalho de Conclusão de Curso apresentado a Diretoria de Pesquisa e Pós-Graduação da Universidade Tecnológica Federal do Paraná do Câmpus Londrina, como requisito parcial à obtenção do grau de especialista em Desenvolvimento Web.

Orientador: Prof. Ms. Thiago Prado de Campos

LONDRINA

2015



TERMO DE APROVAÇÃO

Título da Monografia

CONSTRUINDO UMA API PARA EXIBIR HORÁRIOS DE TRANSPORTE PÚBLICO

por

Rennan Martini Rodrigues

Esta monografia foi apresentada às 10h10 do dia **13 de novembro de 2015** como requisito parcial para a obtenção do título de ESPECIALISTA EM DESENVOLVIMENTO WEB. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho APROVADO.

Prof. Me. Thiago Prado de Campos
(UTFPR)

Prof. Me. Eidy Leandro Tanaka
Guandeline
(UTFPR)

Prof. Dr. Alessandro Botelho Bovo
(UTFPR)

Visto da coordenação:

Prof. Me. Thiago Prado de Campos
Coordenador da esp. em Desenvolvimento Web

Prof. Me. José Luis Dalto
Coordenador de Pós-Graduação Lato Sensu

AGRADECIMENTOS

Agradeço ao meu orientador Prof. Me. Thiago Prado Campos, pela sabedoria com que me guiou nesta trajetória.

Aos meus colegas Diego e Bruno por contribuírem com ideias e disseminação de conhecimento sobre as tecnologias envolvidas no projeto.

Gostaria de deixar registrado também, o meu reconhecimento à minha família, pois acredito que sem o apoio deles seria muito difícil vencer esse desafio.

Enfim, a todos os que por algum motivo contribuíram para a realização desta pesquisa.

RESUMO

RODRIGUES, Rennan Martini. **Construindo uma API para Exibir Horários de Transporte Público.** 2015. 106 páginas. Monografia (Especialização em Desenvolvimento Web) - Universidade Tecnológica Federal do Paraná. Londrina, 2015.

Projeto de documentação e implementação de uma Interface de Programação de Aplicações para exibir dados de transportes públicos inicialmente para televisores e demais canais de comunicação dentro da Universidade Federal Tecnológica do Paraná, e posteriormente encorajado para ser utilizado fora da instituição, bem como diversas cidades com características comuns de prestação de serviços de transporte público.

Palavras-chave: API. Documentação. Transporte Coletivo. Serviços na Rede.

ABSTRACT

RODRIGUES, Rennan Martini. **Building an API to View Public Transport Timetables.** 2015. 106 páginas. Monografia (Especialização em Desenvolvimento Web) – Federal Technology University - Parana. Londrina, 2015.

Design of documentation and implementation of an Application Program Interface to retrieve informations of public transport at first glance at TVs and other communication channels inside of Federal Technology University – Parana, encouraging later to keeping used outside college, as well in several cities with

Keywords: API. Documentation. Public Transport. Web Services

LISTA DE ILUSTRAÇÕES

Figura 1 Representação simplificada de um Sistema de Banco de Dados (Milani, 2007)	23
Figura 2 Exemplo de uma planilha do site da CMTU com relação horários de uma linha.....	26
Figura 3 Formulário de pesquisa do site TCGL.....	28
Figura 4 Resultado do formulário de pesquisa do site TCGL	29
Figura 5 Mapa com marcação de linha de ônibus no site TCGL.....	29
Figura 6 Arquivo compactado enviado pela CMTU com a relação de horários das linhas de ônibus	31
Figura 7 Primeira etapa para selecionar uma linha de ônibus no site TCGL	32
Figura 8 Percurso realizado para mostrar horários de uma linha.....	33
Figura 9 Exemplo ao recarregar a página de exibição de horários de uma linha	33
Figura 10 Tela do aplicativo UTFbus.....	34
Figura 11 Tela do aplicativo Ônibus Uel.....	35
Figura 12 Tela com relação de horários do aplicativo Ônibus UEL.....	36
Figura 13 Tela do aplicativo Moovit.....	37
Figura 14 Diagrama Entidade-Relacionamento da API de Transporte Público	40
Figura 15 Componentes envolvidos em um projeto de API RESTful	42
Figura 16 Modelo de uma requisição HTTP (Devmedia 2014)	43
Figura 17 Documentação em Markdown na ferramenta API Apiary.....	46
Figura 18 Resultado da documentação no modo de requisições.....	47
Figura 19 Código SQL para criar tabela de Horários de ônibus	51
Figura 20 Aplicação inicial gerada pelo Express.js	52
Figura 21 Configuração do arquivo package.json	54
Figura 22 Tela de pesquisa de horários	57
Figura 23 Tela com horários da linha pesquisada.....	57
Figura 24 Tela com itinerário da linha pesquisada	58
Figura 25 Teste de requisição usando a ferramenta POSTMAN	59
Figura 26 Classificação dos Circulares que operam na região metropolitana de Curitiba - PR.....	67
Figura 27 Horários de uma linha de ônibus no site URBS	68
Figura 28 Tabela de horários em arquivo .pdf no site SETRANS	70
Figura 29 Relação de horários no site TCCC.....	71
Figura 30 Tabela de linhas de ônibus da cidade de Ponta Grossa - PR.....	73
Figura 31 Tabela de horários no site VCG	74
Figura 32 Listagem de linhas de ônibus do site CETTRANS	76
Figura 33 Listagem de linhas de ônibus do site FOZTRANS	77
Figura 34 Tabela de horários da linha 120 Parque Nacional de Foz do Iguaçu - PR.....	78

LISTA DE TABELAS

Tabela 1 Características identificadas sobre transporte público nas cidades do estado do Paraná	38
Tabela 2 Convenções utilizadas para elaboração da API	44
Tabela 3 Exemplo de respostas para o web service “Adicionar um novo onibus”	48
Tabela 4 Exemplo de respostas para o web service “Obter informações de uma via de onibus”	49
Tabela 5 Web service para Obter horários de uma via	49
Tabela 6 Exemplo de respostas para o web service “Adicionar uma nova parada do itinerário do onibus”	50
Tabela 7 Exemplo de respostas para o web service “Editar um registro de parada do itinerário do onibus”	50
Tabela 8 Web service para Deletar um registro de parada do itinerário do onibus ...	50
Tabela 9 Script para criar rota dos horários de uma via de ônibus	55
Tabela 10 Exemplo de requisição da API SPTrans.....	61

LISTA DE SIGLAS

BDD	Banco de Dados Distribuídos
CETTRANS	Companhia de Engenharia de Transporte e Trânsito
CMTU	Companhia Municipal de Transito e Urbanização
FOZTRANS	Instituto de Transportes e Trânsito de Foz do Iguaçu
NPM	Node Package Manager
MVC	Model, View, Controller
PDF	Portable Document Format
RIT	Rede Integrada de Transporte Coletivo de Curitiba
SETRANS	Secretaria do Transito e da Segurança
SGBD	Sistema Gerenciador de Banco de Dados
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
TCCC	Transportes Coletivos Cidade Canção
TCGL	Transportes Coletivos Grande Londrina
TTU	Terminal de Transporte Urbano
URBS	Urbanização de Curitiba S/A
URL	Uniform Resource Locator
UTFPR	Universidade Federal Tecnológica do Paraná
VCG	Viação dos Campos Gerais
BDD	Banco de Dados Distribuídos
CETTRANS	Companhia de Engenharia de Transporte e Trânsito
CMTU	Companhia Municipal de Transito e Urbanização
FOZTRANS	Instituto de Transportes e Trânsito de Foz do Iguaçu
MVC	Model, View, Controller
PDF	Portable Document Format
RIT	Rede Integrada de Transporte Coletivo de Curitiba
SETRANS	Secretaria do Transito e da Segurança
SGBD	Sistema Gerenciador de Banco de Dados
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
TCCC	Transportes Coletivos Cidade Canção
TCGL	Transportes Coletivos Grande Londrina

TTU	Terminal de Transporte Urbano
URBS	Urbanização de Curitiba S/A
URL	Uniform Resource Locator
UTFPR	Universidade Federal Tecnológica do Paraná
VCG	Viação dos Campos Gerais

LISTA DE ACRÔNIMOS

API	Application Programming Interface
CMS	Content Management System
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
REST	Representational State Transfer
URI	Uniform Resource Identifier
XML	eXtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO	13
1.1 O PROBLEMA E SUA IMPORTÂNCIA	13
1.2 OBJETIVOS	14
2 APLICAÇÕES DESENVOLVIDAS PARA WEB	16
2.1 PONTOS POSITIVOS E NEGATIVOS DAS APLICAÇÕES WEB	16
3 TECNOLOGIAS	18
3.1 CMS	18
3.2 API	19
3.3 JAVASCRIPT	19
3.4 NODEJS E EXPRESS	19
3.5 BANCO DE DADOS	20
3.5.1 TIPOS DE BANCO DE DADOS	20
3.5.2 MYSQL	22
4 O TRANSPORTE COLETIVO DE LONDRINA	24
4.1 SERVIÇOS DISPONÍVEIS NO MEIO DIGITAL	24
4.1.1 CMTU	24
4.1.2 TCGL	27
4.2 PROBLEMAS ENCONTRADOS NOS SERVIÇOS DISPONÍVEIS	30
4.3 OUTROS SERVIÇOS E PROJETOS	34
5 TRANSPORTE COLETIVO DE OUTRAS CIDADES	38
6 A ELABORAÇÃO DA API	39
6.1 MODELAGEM DE DADOS	39
6.2 WEB SERVICES	40
6.3 REST E RESTFUL	41
6.4 DOCUMENTAÇÃO DOS SERVIÇOS	44
6.4.1 CIDADES	48
6.4.2 EMPRESAS	48
6.4.3 ÔNIBUS	48
6.4.4 VIAS	49
6.4.5 HORÁRIOS	49
6.4.6 ITINERÁRIOS	50
6.5 IMPLEMENTAÇÃO DAS TABELAS	51
6.6 IMPLEMENTAÇÃO DOS SERVIÇOS	51
6.7 TESTES DOS SERVIÇOS IMPLEMENTADOS	55
6.8 MELHORIAS FUTURAS	59
7 CONCLUSÃO	62
REFERÊNCIAS	63
APÊNDICE A - Questionário aplicado a TI da empresa TCGL para Levantamento de Requisitos	65

APÊNDICE B - O Transporte Público de Curitiba.....	67
APÊNDICE C - O Transporte Público de Maringá.....	70
APÊNDICE D - O Transporte Público de Ponta Grossa	72
APÊNDICE E - O Transporte Público de Cascavel	75
APÊNDICE F - O Transporte Público de Foz Do Iguaçu	77
APÊNDICE G - Script SQL para criar o Banco de Dados	79
APÊNDICE H - Script SQL para criar a tabela Cidades	80
APÊNDICE I - Script SQL para criar a tabela Empresas.....	81
APÊNDICE J - Script SQL para criar a tabela Circulares	82
APÊNDICE K - Script SQL para criar a tabela Vias.....	83
APÊNDICE L - Script SQL para criar a tabela Horários	84
APÊNDICE M - Script SQL para criar a tabela Itinerários.....	85
APÊNDICE N - Código-fonte da API /cidades	86
APÊNDICE O - Código-fonte da API /empresas.....	89
APÊNDICE P - Código-fonte da API /onibus.....	93
APÊNDICE Q - Código-fonte da API /vias.....	97
APÊNDICE R - Código-fonte da API /itinerarios	101
APÊNDICE S - Código-fonte da API /horarios	104
APÊNDICE T - Código-fonte para buscar horários	109
APÊNDICE U - Código-fonte para mostrar os horários de uma linha de ônibus	110
APÊNDICE V - Código-fonte para mostrar o itinerário de uma linha de ônibus	111

1 INTRODUÇÃO

1.1 O PROBLEMA E SUA IMPORTÂNCIA

O crescimento da população brasileira e sua necessidade de consumo de serviços públicos acarretaram no aumento das demandas públicas, sejam elas sociais, tributárias, políticas, etc. Hoje já não é possível administrar uma cidade utilizando apenas ferramentas manuais, há a necessidade de se informatizar o serviço público, seja ele municipal, estadual e/ou federal. O uso das Tecnologias da Informação e Comunicação (TIC) pode promover a melhoria dos processos da administração pública – uma vez que Denhardt (2008) afirma que a administração pública “está interessada na gestão dos processos de mudança que buscam lograr os valores societários publicamente definidos”.

Muitas tecnologias para construção de aplicações na Internet têm surgido e evoluído nos últimos anos, nas quais, o usuário obtém o que deseja a partir de uma tela de opções, tornam-se cada vez mais sofisticadas e elegantes. Porém, com o emprego maciço da Web, descobriu-se que esta poderosa ferramenta de comunicação também poderia ser usada para transações no atacado entre corporações, governos e seus parceiros.

Para Davenport (1994) a inserção de tecnologias mudou radicalmente a forma de trabalho, pois “os computadores apressam o ritmo de muitas atividades de trabalho e, ao mesmo tempo, reduzem drasticamente a necessidade de mão de obra”, encurtando a noção de tempo e espaço de forma que as atividades podem ser realizadas independente do horário e da localização geográfica das pessoas.

O termo “cidades inteligentes” refere-se a um dispositivo estratégico para o planejamento e gestão inteligente de cidades. O conceito também é visto por alguns autores como algo associado à atração de capital humano [Glaeser e Berry 2006], ou mesmo a combinação deste aspecto com a qualidade de vida [Shapiro 2006] na tentativa de explicar o rápido crescimento de cidades que abrigam pessoas altamente capacitadas. Entretanto, o termo “cidades inteligentes” tem sido, cada vez mais, relacionado ao emprego eficiente de Tecnologias de Informação e Comunicação como uma ferramenta para melhorar a infraestrutura e serviços da cidade, conseqüentemente trazendo melhor qualidade de vida.

Nesse sentido, Gama et al (2012, p. 1) destaca que “No contexto de cidades inteligentes, diversas TICs podem ser utilizadas, tais como Internet das Coisas [Ahson 2008]; sensores diversos (ex: *RFID* e *ZigBee*); sistemas de informação; computação em nuvem para armazenar e aumentar a eficiência de aplicações; mobilidade com aplicações para smartphones e *tablets*; *business intelligence* como forma de minerar dados para apoio à decisão, dentre outros, tornando possível a coleta, processamento, distribuição e análise que facilitem a tomada de decisão estratégica governamental ou da população.”

Por outro lado, é necessária a criação de sistemas capazes de lidar com toda a massa de informação criada, seja através de sistemas complexo para análise dos dados seja através de infraestrutura para interligar estes sistemas. Níveis de maturidade tecnológica serviriam como guia gradativo rumo à construção de uma cidade inteligente, auxiliando na tomada de decisão, gestão e planejamento estratégico das cidades.

1.2 OBJETIVOS

O objetivo geral deste trabalho é propor uma Interface de Programação de Aplicações para fornecer e exibir horários de transportes públicos e informações associadas, inicialmente para uso interno na UTFPR do campus Londrina.

Dentre os objetivos específicos destacam-se:

- Utilizar os recursos da linguagem *JavaScript* e seus *frameworks* para fomentar a construção da aplicação;
- Documentar a arquitetura dos frameworks e as ferramentas auxiliares usadas disseminando o conhecimento e, por conseguinte, auxiliando as manutenções evolutivas da aplicação.

Este trabalho está estruturado da seguinte forma:

A primeira parte descreve a Fundamentação Teórica, que conceitua algumas definições que podem não ser do conhecimento de todos, os frameworks e ferramentas utilizadas na construção da aplicação através de bibliografias;

A segunda parte descreve o problema dos recursos digitais na prestação de serviços de transporte coletivo da cidade de Londrina. Demonstra também estudos que já existem, atualmente, e que podem auxiliar na resolução do problema;

A terceira e quarta parte descrevem a proposta de elaboração da API, desde a sua fase inicial (levantamento de requisitos) até sua fase final (desenvolvimento);

E, por conseguinte, as Considerações Finais e Apêndices que autorizaram e contribuíram para a construção do sistema proposto.

2 APLICAÇÕES DESENVOLVIDAS PARA WEB

Um sistema ou aplicação web é um programa armazenado em um servidor remoto e acessado via internet pelo usuário através de um navegador, que é reconhecido como um cliente fazendo a conexão entre o ambiente do sistema, na maioria das vezes distante, ao usuário (ROUSE, 2011).

A complexidade de um sistema web pode ser mínima, caso seja um aplicativo simples como um livro de visitas online ou um mural de recados e também pode ser extremamente complexa com aplicativos que podem substituir programas do próprio desktop, como um editor de texto avançado ou um editor de planilha (NATIONS, 2013).

2.1 PONTOS POSITIVOS E NEGATIVOS DAS APLICAÇÕES WEB

Uma aplicação *web* tira a responsabilidade de uma equipe desenvolver em relação a compatibilidade com sistemas operacionais ou diferentes tipos de computadores pois o sistema é acessado e executado diretamente no *browser* do usuário. Embora alguns aplicativos necessitem de um browser específico para que o seu funcionamento seja de forma adequada, a maioria funciona sem problemas em grande parte dos browsers disponíveis atualmente no mercado (NATIONS, 2013).

No desenvolvimento de aplicações para a web, os desenvolvedores comumente utilizam combinações de linguagens *client-side* e *server-side*, onde a *client-side* é responsável pela apresentação da informação enquanto a *server-side* trabalha com o processamento mais pesado, como o armazenamento das informações e a consulta de dados (NATIONS, 2013).

Apesar das vantagens, as aplicações *web* possuem também alguns pontos que devem ser analisados com muita atenção. Sérios problemas de vulnerabilidades no código permitem que *hackers* obtenham o acesso direto a informações armazenadas em bancos de dados restritos. Um outro ponto que deve ser analisado é a questão da disponibilidade do sistema, um bom site e uma boa aplicação devem estar disponíveis para acesso 24 horas por dia, 7 dias por semana para prover o

acesso requisitado por funcionários, clientes, fornecedores e outros relacionados (ACUNETIX, 2013).

3 TECNOLOGIAS

A escolha da tecnologia é de suma importância para o desenvolvimento de sistema de informação, e hoje, a alta disponibilidade, o fácil acesso, etc. são considerados necessidades básicas. Com isso os aplicativos são excelentes, pois permitem a disponibilização e o processamento eficiente de uma grande quantidade de informações, com um diferencial: a independência geográfica, gerenciando informações de diferentes locais a baixo custo como relata Andreto e Amaral (2006, p. 158).

Para propor uma solução que possa fornecer informações de horários de ônibus ao CMS da Universidade, será proposto um estudo de elaboração de uma ferramenta que poderá consumir recursos de serviços *web*.

3.1 CMS

Um Sistema de Gerenciamento de Conteúdo, ou CMS, do acrônimo em inglês *Content Management System*, é uma aplicação web projetada para ser fácil para usuários não-técnicos poder adicionar, editar e gerenciar um website. Não somente fazer o gerenciamento do conteúdo mas ajudar a cuidar de vários trabalhos "por trás das cenas", como: gerar navegação automática de elementos; fazer o conteúdo ser pesquisável e indexável; manter contato com usuários, suas permissões e configurações de segurança e muito mais (Plone Foundation, 2000).

A Universidade Federal Tecnológica do Paraná, campus Londrina, utiliza um CMS, desenvolvido e mantido pelo Prof. Me. Thiago Prado de Campos, para notificar alunos, professores e colaboradores informações que circulam na universidade como, por exemplo, feiras de ciências e tecnologias, cursos de extensão, datas de feriados e recessos, etc. Uma das ideias que o Professor Thiago tem é de incluir novos recursos como comunicados em geral, previsão do tempo, horário de ônibus, aniversariantes, vídeos, galeria de fotos, etc.

3.2 API

De acordo com o dicionário FOLDOC (Free On-Line Dictionary Of Computer), o termo Interface de Programação de Aplicações, cujo acrônimo API provém do Inglês *Application Programming Interface*, é um conjunto de rotinas e padrões estabelecidos por um software para a utilização das suas funcionalidades por aplicativos que não pretendem envolver-se em detalhes da implementação do software, mas apenas usar seus serviços.

A proposta inicial deste trabalho será prover informações sobre horários e itinerários dos ônibus da cidade de Londrina e que poderá ser utilizada pelo CMS já existente na universidade.

3.3 JAVASCRIPT

JavaScript é uma linguagem de programação interpretada. Foi originalmente implementada em 1995 por Brendan Eich como parte dos navegadores para que scripts pudessem ser executados do lado do cliente e interagissem com o usuário sem a necessidade deste script passar pelo servidor, controlando o navegador, realizando comunicação assíncrona e alterando o conteúdo do documento exibido (NEGRINO, SMITH, 1999).

Com o avanço da *web* e a ascensão do *HTML5*, a linguagem *JavaScript* se tornou poderosa, indo muito além da premissa inicial de ser somente utilizada como a camada de comportamento de páginas. O resultado destes avanços originou o *V8*, um interpretador *JavaScript* desenvolvido pela *Google* e utilizado em seu navegador *Google Chrome*, que acelera o desempenho de uma aplicação compilando o código *JavaScript* para o formato nativo de máquina antes de executá-lo, permitindo que rode a velocidade de um código binário compilado.

3.4 NODEJS E EXPRESS

Node.js é um interpretador de código *JavaScript* que funciona do lado do servidor. Seu objetivo é ajudar programadores na criação de aplicações de alta escalabilidade (como um servidor *web*), com códigos capazes de manipular dezenas

de milhares de conexões simultâneas, numa única máquina física. O *Node.js* é baseado no interpretador V8. Foi criado por Ryan Dahl em 2009, e seu desenvolvimento é mantido pela empresa Joyent, onde Dahl trabalha.

Foi escolhido a ferramenta Express.js, um *framework* para *Node.js* que é minimalista, flexível e contém um robusto conjunto de recursos para desenvolver aplicações *web*, como um sistema de *Views* intuitivo (MVC), um robusto sistema de roteamento, um executável para geração de aplicações e muito mais.

3.5 BANCO DE DADOS

Um sistema de banco de dados é basicamente um sistema computadorizado de manutenção de registros. O banco de dados, por si só, pode ser considerado como o equivalente eletrônico de um armário de arquivamento; ou seja, ele é um repositório ou recipiente para uma coleção de arquivos de dados computadorizados. Os usuários de um sistema podem realizar (ou melhor, solicitar que o sistema realize) diversas operações envolvendo tais arquivos - por exemplo:

- Acrescentar novos arquivos ao banco de dados
- Inserir dados de arquivos existentes
- Buscar dados de arquivos existentes
- Excluir dados de arquivos existentes
- Alterar dados de arquivos existentes
- Remover arquivos existentes do banco de dados

Estas operações são executadas através de requisições, expressas em uma linguagem chamada *SQL*, linguagem essa de consulta estruturada.

3.5.1 TIPOS DE BANCO DE DADOS

Os bancos de dados podem ser classificados de várias formas, uma delas é quanto à arquitetura do banco, isto é, como ele foi projetado no ambiente computacional para receber, armazenar os dados e dar respostas aos clientes

solicitantes. Dentre os tipos mais comuns existem Banco de dados Distribuídos, Banco de dados Relacional e Banco de dados Não-Relacional.

Banco de Dados Distribuído (BDD) é uma coleção de várias bases de dados logicamente inter-relacionadas, distribuídas por uma rede de computadores. Num banco de dados distribuídos os arquivos podem estar replicados ou fragmentados, esses dois tipos podem ser encontrados ao longo dos nós do sistema de *BDD's*.

Um Banco de Dados Relacional é um banco de dados que modela os dados de uma forma que eles sejam percebidos pelo usuário como tabelas, ou mais formalmente relações. Os bancos de dados relacionais foram desenvolvidos para prover acesso facilitado aos dados, possibilitando que os usuários utilizassem uma grande variedade de abordagens no tratamento das informações. Há muitos Sistemas de Gerenciamento de Banco de Dados (SGBDs) que utilizam o conceito de dados relacionais, como *Oracle*, *IBM DB2* e *Microsoft SQL Server*, *MySQL*, *PostgreSQL*, entre outros.

Os bancos de dados relacionais têm sido a escolha padrão para persistência de dados corporativos. No entanto, as grandes empresas e aplicações web estão mudando ao longo dos últimos anos. Com o aumento da quantidade e do fluxo de informações e a certeza de que sempre haverá mais dados para armazenar, o tradicional modelo relacional começa a sofrer limitações de escalabilidade. Neste cenário surge uma nova geração de banco de dados, não-relacional, como uma maneira de lidar com o crescente volume de dados (PEREIRA, 2011).

O *NoSQL (Not-only SQL)* atende aos requisitos do ambiente de computação distribuída em larga escala, o que permite escalabilidade, alta disponibilidade, alto desempenho e confiabilidade. A grande motivação para o *NoSQL* é resolver o problema de escalabilidade dos bancos tradicionais. Contudo, se faz necessário analisar todas as suas características para se ter uma visão geral de seus prós e contras quando comparado ao modelo relacional. Entretanto, este é um tema que não está planejado ser discutido e analisado no presente trabalho, cuja a escolha do tipo de banco de dados se dá pelo modelo relacional chamado *MySQL*.

3.5.2 MYSQL

O *MySQL* é um servidor e gerenciador de banco de dados (SGBD) relacional, de licença dupla (sendo uma delas de software livre), projetado inicialmente para trabalhar com aplicações de pequeno e médio portes, mas hoje atendendo a aplicações de grande porte e com mais vantagens do que seus concorrentes. Possui todas as características que um banco de dados de grande porte precisa, sendo reconhecido por algumas entidades como o banco de dados de código-aberto com maior capacidade para concorrer com programas similares de código fechado, tais como *SQL Server* (da *Microsoft*) e *Oracle*.

A escolha deste SGDB se deu por ser o banco de dados de código-aberto mais popular no mundo, e que conta com diversas vantagens, entre elas: a portabilidade entre plataformas, compatibilidade com diversos Sistemas Operacionais, excelente desempenho e estabilidade e pouco exigente quanto a recursos de novos hardwares, bem como a facilidade no manuseio.

De acordo com Milani (2007) o *MySQL* foi projetado para trabalhar com aplicações de pequeno a médio porte, algo em torno de 100 milhões de registros por tabela, tendo como tamanho médio aproximadamente 100MB por tabela. Contudo, esses números eram os recomendados para as primeiras versões da ferramenta. Atualmente os limites e capacidades do *MySQL* ultrapassaram essas fronteiras inúmeras vezes.

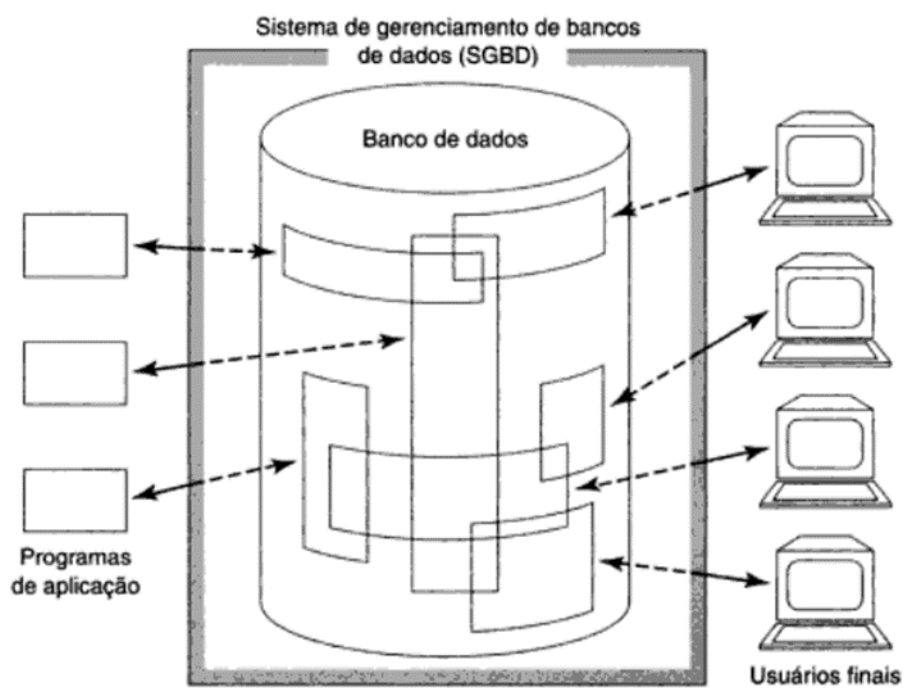


Figura 1 Representação simplificada de um Sistema de Banco de Dados (Milani, 2007)

4 O TRANSPORTE COLETIVO DE LONDRINA

A cidade de Londrina possui um moderno e funcional sistema de transporte coletivo, o qual é gerido pela CMTU - Companhia Municipal de Trânsito e Urbanização de Londrina. É possível tomar um coletivo em qualquer bairro ou distrito e, com a mesma passagem, chegar a qualquer ponto da cidade, graças ao sistema integrado de transporte. O serviço é prestado por três empresas: Francovig & Cia Ltda., Expresso Nordeste Ltda. e Transportes Coletivos Grande Londrina Ltda., também conhecida na cidade como Londrisul.

As diversas linhas do Sistema de Transporte Coletivo de Londrina possuem horários em dias úteis e finais de semana. Além disso, nos feriados prolongados e pontos facultativos são criadas tabelas especiais visando atender a demanda esporádica daquele dia.

De acordo com a CMTU (2014), a cidade possui uma excelente rede de itinerários de linhas de ônibus, pois atende algumas regiões isoladas e distantes, de forma que a maioria dos municípios são atendidos pelo Transporte Coletivo. Além disso, a distância dos pontos de ônibus não ultrapassa a 300 metros, permitindo que o usuário do sistema não tenha grande deslocamento para acessar a rede de transportes.

4.1 SERVIÇOS DISPONÍVEIS NO MEIO DIGITAL

Das empresas que prestam serviço de transporte coletivo em Londrina, apenas duas possuem o recurso de exibir na internet as informações sobre horários e itinerários das linhas de ônibus: a CMTU e a Grande Londrina.

4.1.1 CMTU

Em cumprimento às disposições do Código de Trânsito Brasileiro, que municipalizou a fiscalização do trânsito, a Companhia Municipal de Urbanização passou a ter competência de autoridade de trânsito no âmbito municipal, sendo então denominada Companhia Municipal de Trânsito e Urbanização – CMTU-LD,

por meio da Lei Municipal nº 8191 de 19 de junho de 2000. Esta lei atribui à Companhia a tarefa de gerenciar e fiscalizar o trânsito no que lhe couber no âmbito do Município de Londrina.

No site da CMTU¹ há disponibilização do preço da passagem de ônibus, bem como a relação de linhas e itinerários. Fazendo uma análise de funcionalidade no site foi possível deduzir como funcionava o processo de lançamento dos horários, mas para ter compreensão de como ocorre o lançamento destas informações, foi realizada uma visita à Companhia.

Os problemas encontrados no site da CMTU foram esboçados na próxima subseção do presente trabalho.

¹ <http://www2.londrina.pr.gov.br/cmtu>

113 - Pioneiros								
DIAS ÚTEIS - SÁBADO - DOMINGO					ITINERÁRIO			
DIAS ÚTEIS - SAÍDA DO TERMINAL CENTRAL								
6:05	6:45	7:05	7:25	7:45	8:05	8:26	8:47	9:08
9:29	9:50	10:11	10:32	10:53	11:14	11:40	12:03	12:27
12:50	13:14	13:37	14:01	14:24	14:48	15:11	15:35	16:00
16:25	16:50	17:15	17:40	18:05	18:30	18:55	19:45	20:32
21:23	<i>21:43</i>	22:04	<i>22:22</i>	22:45	23:20			
PARQUE TAUÁ					<i>COMÉRCIO ABERTO ATÉ 22H</i>			
EXPRESSO UTFPR / BOULEVARD - SAÍDA DO TERMINAL CENTRAL								
18:20	22:42							
DIAS ÚTEIS - SAÍDA DO BAIRRO (UTFPR)								
6:25	7:01	7:21	7:41	8:01	8:21	8:42	9:03	9:24
9:45	10:06	10:27	10:48	11:11	11:34	11:58	12:21	12:45
13:08	13:32	13:55	14:19	14:42	15:06	15:29	15:53	16:20
16:45	17:10	17:35	18:00	18:25	18:50	19:15	20:03	20:50
21:39	<i>21:59</i>	22:20	23:05	23:35				
PARQUE TAUÁ					<i>COMÉRCIO ABERTO ATÉ 22H</i>			
EXPRESSO UTFPR / BOULEVARD - SAÍDA DA UTFPR								
17:00	17:20	18:35	20:00	22:55				
SÁBADO-SAÍDA DO TERMINAL CENTRAL								
6:55	7:40	8:25	12:20	12:56	13:32	16:40		
SÁBADO-SAÍDA DO BAIRRO								
7:20	12:00	12:36	13:12	17:05	18:05			
DOMINGO - SAÍDA DO TERMINAL CENTRAL								
7:40	8:25	16:40						

Published by Google Sheets – Denunciar abuso – 5Atualizado automaticamente a cada minuto

Figura 2 Exemplo de uma planilha do site da CMTU com relação horários de uma linha

4.1.2 TCGL

A TCGL – Transportes Coletivos Grande Londrina - é uma empresa de transporte coletivo que atende de forma privada o sistema urbano da cidade de Londrina há mais de 20 anos.

No site da TCGL² também foi realizada uma análise de funcionalidades e coerência das informações. Constatou-se que não possui informações sobre o custo da passagem de ônibus, entretanto, o recurso de listar horários e itinerário de uma linha é organizado em relação ao site da CMTU. Na primeira etapa, o usuário tem a opção de escolher uma das linhas pré-definidas em um campo de seleção, podendo escolher o tipo de relação de horários (semanal, sábado ou domingo) e a origem/destino da linha; Na segunda etapa, o usuário é direcionado à tela resumida com as informações previamente selecionadas e também uma tabela com a relação de todos horários que a linha irá operar no dia. Ainda nesta tela, existe um recurso para exibir o deslocamento dos ônibus da empresa. Este recurso é desenvolvido por uma empresa terceirizada, com o auxílio da ferramenta *Google Maps* – que não funciona a todo momento e com qualquer endereço; é possível interpretar através de legendas na página qual é a localização dos terminais urbanos, dos ônibus que estão em deslocamento e em qual sentido acontece o deslocamento (terminal ou bairro). O recurso não acontece em tempo real, tem um atraso de 15 segundos, que são informados pela própria companhia no rodapé do mapa.

² <http://site.tcgrandelondrina.com.br:8082>

CONSULTA DE HORÁRIOS E ITINERÁRIOS

PSIU CONVENCIONAL

Selecione a linha ? Tipo

113 - PIONEIROS SEMANA

Origem Destino

113 - PIONEIROS 113 - PIONEIROS

CONSULTAR

Figura 3 Formulário de pesquisa do site TCGL

A Transportes Coletivos Grande Londrina disponibiliza uma frota de 362 ônibus para o atendimento da malha urbana 24 horas por dia. Confira os itinerários e horários abaixo.

113 - PIONEIROS
Dia : Semana - Saída : 113 - PIONEIROS - Chegada : 2 - TC-CENTRAL

Horários			Itinerários		
Saída	Chegada	Via	Saída	Chegada	Via
06:25	06:40		15:08	15:30	
07:01	07:20		15:29	15:55	
07:21	07:40	PQ.TAUÁ	15:53	16:20	
07:41	08:00		16:20	16:45	
08:01	08:21	PQ.TAUÁ	16:45	17:10	
08:21	08:42		17:00	17:15	
08:42	09:03		17:10	17:35	
09:03	09:24		17:15	17:30	
09:24	09:45		17:25	17:50	

Mostrar/Ocultar Itinerários Sua localização

Terminais Urbanos | Ônibus | Sentido Bairro | Sentido Terminal

O mapa é atualizado automaticamente a cada 15 segundos.

Figura 4 Resultado do formulário de pesquisa do site TCGL

Mostrar/Ocultar Itinerários Sua localização

Terminais Urbanos | Ônibus | Sentido Bairro | Sentido Terminal

O mapa é atualizado automaticamente a cada 15 segundos.

Dados cartográficos ©2015 Google Termos de Uso Informar erro no mapa

Figura 5 Mapa com marcação de linha de ônibus no site TCGL

Foi possível perceber que a empresa utilizou no desenvolvimento deste serviço o recurso de SOAP, um protocolo para troca de informações estruturadas em uma plataforma descentralizada e distribuída. Para entender melhor como os

serviços web foram utilizados no site, foi realizado diversos contatos com o departamento responsável e aplicado um questionário técnico sobre funcionamento do site. Este questionário está anexado aos apêndices deste trabalho.

4.2 PROBLEMAS ENCONTRADOS NOS SERVIÇOS DISPONÍVEIS

Foi realizado uma análise técnica no site da CMTU. Tarefa esta que visou compreender não só a avaliação da interface e usabilidade do site, mas também quais recursos de navegabilidade são possíveis até realizar um objetivo específico. Ainda neste contexto foram levantados alguns questionamentos³.

O primeiro problema que pode ser encontrado no site da CMTU é a tarefa de atualizar os horários manualmente e compartilhar em uma planilha. Para investigar melhor este caso, durante o processo de pesquisa e análise de requisitos do presente projeto, foi realizado uma visita presencial ao departamento administrativo da CMTU. A visita não foi previamente agendada e o pedido de informações do serviço urbano foi solicitado sem justificativa da parte solicitante. Este recurso foi possível graças à Lei de Acesso à Informação⁴. Descrita e classificada como prestadora de serviço municipal, a Companhia Municipal de Trânsito e Urbanização de Londrina forneceu, por escrito, um protocolo de comprovante de atendimento e um prazo de 7 dias corridos para o fornecimento das informações.

De acordo com o material fornecido pela CMTU, foi possível perceber que a empresa utiliza um fluxo para divulgar as informações e que ocorre da seguinte forma:

- Preenche em planilhas estáticas a relação de horários e itinerário de todas as linhas que operam na cidade de Londrina;
- Essa informação é transferida para o aplicativo Google Drive;

³ Qual caminho é necessário que o usuário faça para chegar até a relação de horários de uma linha de ônibus? Essa informação será fácil de ser recuperada posteriormente?

⁴ A Lei de nº 12.527/2011 que regulamenta o direito constitucional de acesso às informações públicas. Essa norma entrou em vigor em 16 de maio de 2012 e criou mecanismos que possibilitam, a qualquer pessoa, física ou jurídica, sem necessidade de apresentar motivo, o recebimento de informações públicas dos órgãos e entidades.

- Posteriormente é gerado um embed - um tipo de tag usada para incorporar arquivos multimídia na internet - de cada tabela.

O usuário não consegue recuperar a informação do último acesso nos horários por conta da *url* da página ser única e embutida.

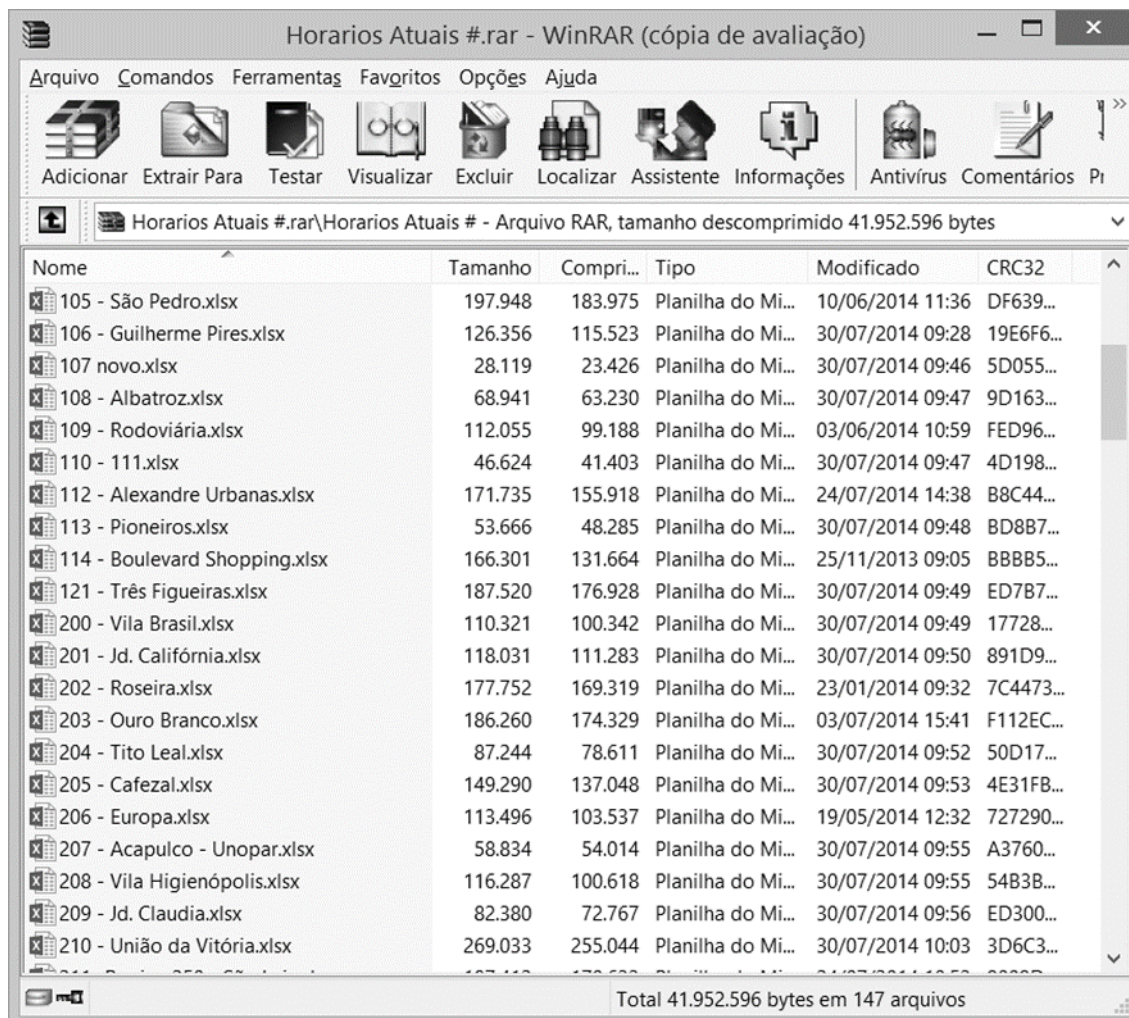


Figura 6 Arquivo compactado enviado pela CMTU com a relação de horários das linhas de ônibus

No site da empresa Transportes Coletivos Grande Londrina também foi realizado uma avaliação técnica das condições de usabilidade e navegabilidade. Na primeira etapa, foi possível perceber que o recurso de localizar uma linha de ônibus restringe o usuário a depender de opções pré-definidas que ficam listadas em uma caixa de seleção (combo box). O usuário também tem a obrigação de pesquisar a linha utilizando a barra de rolagens dentro da pequena janela de navegação embutida na caixa de seleção.

Figura 7 Primeira etapa para selecionar uma linha de ônibus no site TCGL

Após escolher a linha de ônibus, deve ser selecionado o Tipo de horário a ser listado e em seguida escolher a Origem e Destino da linha. Por exemplo, se a linha parte do Terminal Central para o Bairro ou o percurso inverso.

Na segunda etapa, o usuário então é direcionado ao trajeto da linha escolhida, onde é possível ter acesso detalhado de forma mais eficiente que o site da CMTU. As seguintes informações estão disponíveis: Resumo das escolhas realizadas na etapa anterior, uma tabela com duas abas de detalhamento: na primeira, a relação dos horários disponíveis, na segunda, a ordem das paradas do itinerário da linha. Abaixo da tabela existe a opção de ver um mapa, projetado em tempo real, com o deslocamento do seu itinerário.

Um dos problemas encontrados é que todo percurso de ações executadas pelo usuário acontece através de requisições Ajax. Estas requisições são disparadas para chamar os web services no servidor da aplicação, que por sua vez retorna as informações e manipula o resultado diretamente no client-side (navegador). Uma característica dessa arquitetura é não acontecer a mudança no endereço da página, e a requisição pode ser perdida caso a página seja atualizada, logo, e o usuário quiser buscar novamente outra linha, não é possível e ele deverá fazer todo o percurso novamente.



Consultas de Horários x

site.tcgrandelondrina.com.br:8082/soap/BuscaHorarios

COMPROMISSO COM VOCÊ!

Entre para pesquisar...

EMPRESA - SERVIÇOS - VENDA DE CRÉDITO INFORMATIVO TRABALHE CONOSCO FALE CONOSCO

Home > Consultas de Horários > Linha Psiu

CONSULTA DE HORÁRIOS

A Transportes Coletivos Grande Londrina disponibiliza uma frota de 362 ônibus para o atendimento da malha urbana 24 horas por dia. Confira os itinerários e horários abaixo.

113 - PIONEIROS
Dia : Semana - Saída : 113 - PIONEIROS - Chegada : 2 - TC-TCENTRAL

Figura 8 Percurso realizado para mostrar horários de uma linha



Consultas de Horários x

site.tcgrandelondrina.com.br:8082/soap/BuscaHorarios

CONFIRMAR REENVIO DO FORMULÁRIO

A página que você está procurando usou as informações inseridas. Voltar à essa página poderá fazer com que todas as ações realizadas antes sejam repetidas. Deseja continuar?

Continuar Cancelar

Entre para pesquisar...

EMPRESA - SERVIÇOS - VENDA DE CRÉDITO INFORMATIVO TRABALHE CONOSCO FALE CONOSCO

Home > Consultas de Horários > Linha Psiu

CONSULTA DE HORÁRIOS

A Transportes Coletivos Grande Londrina disponibiliza uma frota de 362 ônibus para o atendimento da malha urbana 24 horas por dia. Confira os itinerários e horários abaixo.

113 - PIONEIROS
Dia : Semana - Saída : 113 - PIONEIROS - Chegada : 2 - TC-TCENTRAL

Figura 9 Exemplo ao recarregar a página de exibição de horários de uma linha

4.3 OUTROS SERVIÇOS E PROJETOS

Como já foi evidenciado alguns problemas nos atuais meios de serviços digitais, algumas pessoas e empresas pensaram em soluções que contornam a situação de complexidade para saber que horas um ônibus passa.

O aplicativo UTFbus foi desenvolvido pelos alunos Lucas Jansen, Rafael Breus, Robson Penteado e Patrick Staroin, todos do primeiro período do curso de Ciência da Computação, durante a disciplina de introdução à Ciência da Computação da UTFPR campus Ponta Grossa. Ao abrir o aplicativo, o aluno tem a hora aproximada que o próximo ônibus irá chegar no câmpus, e todos os próximos ônibus a partir do horário atual do smartphone.



Figura 10 Tela do aplicativo UTFbus

O aplicativo Ônibus UEL (Google Play, 2015) também é um outro exemplo que atende às necessidades dos estudantes da Universidade Estadual de Londrina. O estudante deve apenas selecionar qual item deseja acessar e em seguida ele obtém acesso à informação.



Figura 11 Tela do aplicativo Ônibus Uel



Saída	Chegada
05:53	06:11
06:18	06:36
06:35	06:55
07:02	07:20
07:25	07:39
07:43	08:04
08:05	08:23
08:30	08:48
08:49	09:07
09:14	09:32
09:33	09:51
09:58	10:16
10:17	10:35
10:42	11:00
11:01	11:19

Figura 12 Tela com relação de horários do aplicativo Ônibus UEL

O *Moovit* é um aplicativo existente em diversos países que oferece cobertura em centenas de cidades, incluindo Londrina. O aplicativo serve para traçar rotas considerando apenas opções de transporte público, enviar aos usuários informações em tempo real até sobre quando devem descer dos ônibus ou do metrô. Esses avisos serão feitos por meio da função “Em trânsito”. Ela informa também a hora prevista de chegada, baseada no congestionamento e em alterações ao longo do caminho. Também envia informações sobre o destino, como a quantidade de paradas até lá. O objetivo do aplicativo é tornar-se o mais interativo e ágil possível para lidar com situações que ocorrem enquanto os usuários estão em movimento.

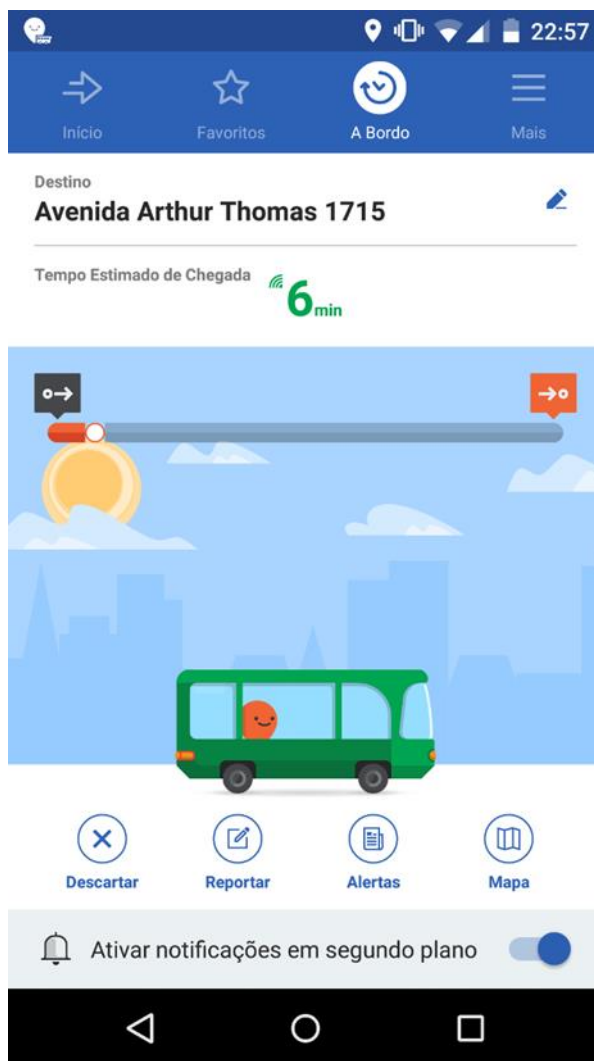


Figura 13 Tela do aplicativo Moovit

5 TRANSPORTE COLETIVO DE OUTRAS CIDADES

A ideia de conhecer como funciona o serviço de transporte coletivo de outras cidades, inicialmente do estado do Paraná, foi encorajada, para tornar proveitosa a elaboração da API afim de especificar um formato comum para horários de transportes públicos e informações geográficas associadas.

Foi realizado uma análise sucinta no Sistema de Transporte Coletivo das seguintes cidades: Curitiba, Maringá, Ponta Grossa, Cascavel e Foz do Iguaçu. Durante a análise tornou-se necessário evidenciar alguns fatos sobre cada município, por conseguinte poder analisar o serviço de transporte coletivo em extensão digital, bem como o percurso necessário que o usuário deve realizar para obter a relação de horas e itinerários sobre o destino planejado.

A tabela abaixo mostra o resumo das características comuns exploradas e identificadas na prestação de serviços de transporte público das cidades.

Cidade	Horários e itinerários estão disponíveis na internet?	Tipo de serviço disponibilizado
Londrina	Sim	Web service e tabelas do Google Sheets
Curitiba	Sim	Tabelas no site
Maringá	Sim	Tabelas em arquivo .pdf
Ponta Grossa	Parcialmente	Tabelas em arquivo .jpg
Cascavel	Parcialmente	Tabelas em arquivo .pdf
Foz do Iguaçu	Sim	Tabelas no site

Tabela 1 Características identificadas sobre transporte público nas cidades do estado do Paraná

A descrição detalhada do que foi analisado em cada cidade está anexada nos Apêndices deste trabalho.

6 A ELABORAÇÃO DA API

Através de dados e as análises discutidas no contexto do capítulo anterior, pôde-se perceber que nenhum dos serviços de Transporte Coletivo dos municípios do Paraná possuem um serviço web. Neste sentido, torna-se interessante o uso da API para que os sites ou quaisquer outros recursos possam consumir informações da ferramenta.

Para propor a elaboração de uma API tornou-se necessário realizar a modelagem de dados, escolha da arquitetura, estruturação dos serviços e ferramentas que irão garantir o funcionamento da mesma. Estas etapas serão detalhadas nos subcapítulos a seguir.

6.1 MODELAGEM DE DADOS

A modelagem do sistema é fundamental para a compreensão do desenvolvedor acerca do sistema a ser desenvolvido e facilitar a correção de problemas antes mesmo de a implementação iniciar. Esta técnica permite realizar a especificação das regras de negócios e as estruturas de dados de um banco de dados. Ela faz parte do ciclo de desenvolvimento de um sistema de informação e é de vital importância para o bom resultado do projeto. Modelar dados consiste em desenhar o sistema de informações, concentrando-se nas entidades lógicas e nas dependências lógicas entre essas entidades (Teorey, 2014).

O diagrama a seguir foi desenvolvido para demonstrar a utilização da API e suas principais funções.

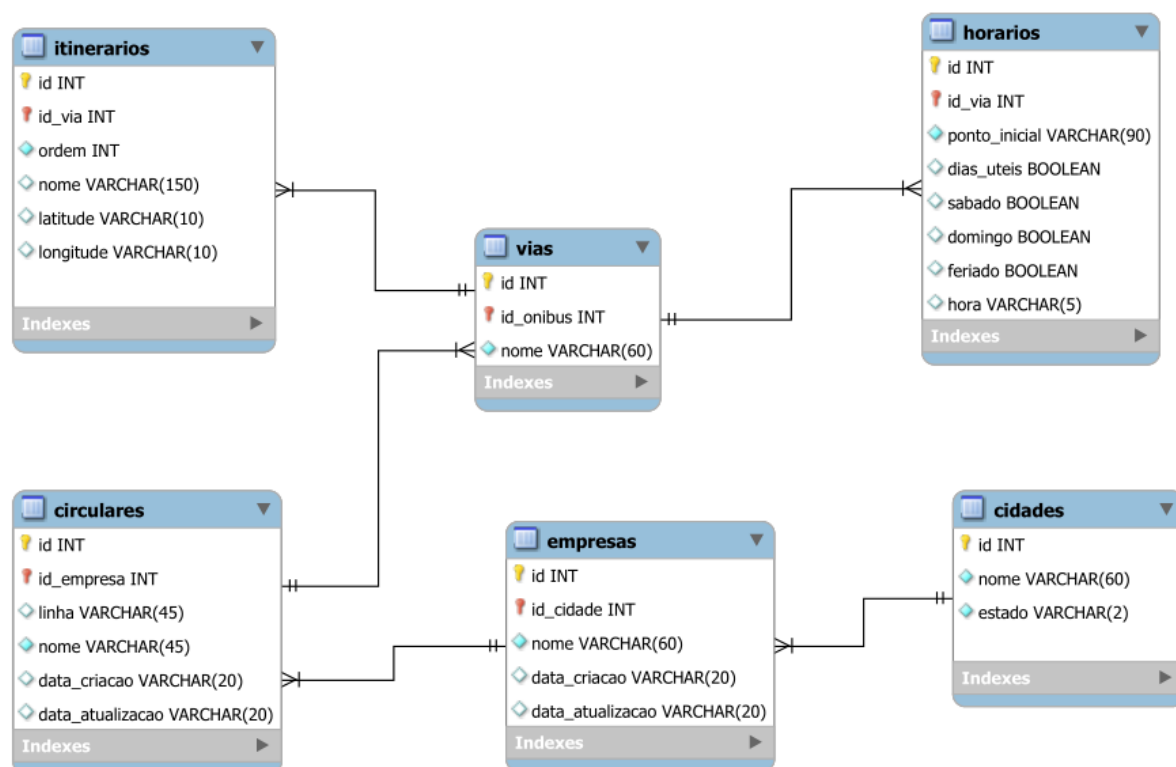


Figura 14 Diagrama Entidade-Relacionamento da API de Transporte Público

6.2 WEB SERVICES

Para Abinader e Lins (2006) a tecnologia Web Services pode oferecer duas visões, que refletem os pontos de vista técnico e conceitual: do ponto de vista técnico, Web Services constituem-se em software de baixo acoplamento, reutilizáveis, com componentes feitos para serem facilmente acessados pela Internet. Na ótica conceitual, o emprego de Web Services representa um modo para integrar tarefas que compõem um processo de negócio através da Internet, em uma cadeia de valor na qual procedimentos estão interligados e são interdependentes para atingir um resultado concreto final.

Em suma, o que os autores definem é que Web Service é uma aplicação que dispõe e publica uma API na Web. Essa API suporta a comunicação programa-para-programa, o que permite que aplicações se comuniquem. Temos aqui a simplicidade da tecnologia, pois existe o serviço que pode ser acessado pela Internet como se fosse uma caixa preta, cabendo ao cliente saber apenas que serviço deseja, e como empregar tal serviço.

6.3 REST E RESTFUL

Para que seja possível compreender como RESTful irá ajudar neste trabalho, é necessário inicialmente aprender o que é REST.

Representational State Transfer, em tradução livre, Transferência de Estado Representativo é uma técnica de engenharia de software para sistemas hipermídia como a web.

O termo REST é muito amplo e diversas representações podem ser ligadas a ele. Dessa forma, surgiu um conceito mais específico ligado às requisições web, o qual se chama RESTful, que obedece às regras do REST e implementa uma interface simples de acesso e resposta a uma requisição web, que pode ser realizada por meio de XML e/ou JSON. Como o JSON é uma forma mais leve de representar um estado, é mais comum utilizar RESTful com JSON em vez do XML.

Outra forma de entender a tecnologia é imaginar que RESTful seja uma forma universal de "trocar" dados entre o cliente e o servidor, algo muito parecido com web services que já foi abordado nas tecnologias web. Utilizar RESTful significa que o acesso ao servidor é usado para obter dados de forma stateless, ou seja, o sistema cliente não conhece o estado do servidor e vice-versa.

Antes de implementar uma API RESTful, faz sentido compreender detalhadamente como a prototipação da tecnologia funciona. Uma API não existe de forma isolada. Em vez disto, uma API expõe as funcionalidades de uma aplicação ou serviço que existe independentemente da própria API.

Durante o processo de projetar a API é importante evidenciar duas coisas:

- Compreender a importância dos detalhes da aplicação para qual a API será criada, para quando o cliente fizer a requisição a funcionalidade possa ser exposta, como ele precisa ser exposta, e qual(is) funcionalidade(s) pode ser deixado de fora.
- Modelar as funcionalidades em uma API que aborda todos os casos de uso que surgem no mundo real, seguindo os princípios RESTful, tanto quanto possível.

Existem três componentes distintos envolvidos em um projeto de API em RESTful: a aplicação, o código da API e o cliente. A imagem a seguir ilustra como esses três componentes interagem.

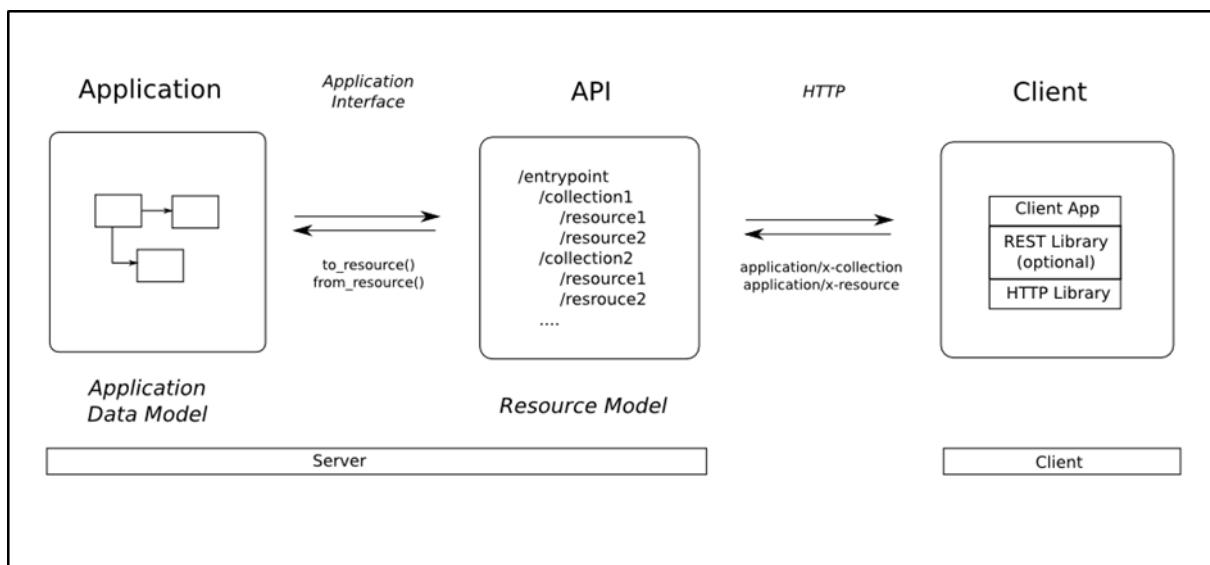


Figura 15 Componentes envolvidos em um projeto de API RESTful

A aplicação para qual uma API é fornecida deve existir independentemente da API. Pode ser uma aplicação com interface gráfica com componentes de interação (site com formulários) ou um aplicativo que acessa somente os serviços da API em segundo-plano.

O trabalho do código da API é acessar o estado do aplicativo, bem como as operações daquele estado, através da interface da aplicação, e expô-la como uma API RESTful. No meio da interface do aplicativo e a API RESTful, há uma etapa de transformação que se adapta o modelo de dados da aplicação e torna respeitar o estilo de arquitetura RESTful.

O resultado desta transformação seria recursos RESTful, operações sobre esses recursos, e as relações entre os recursos. Todos estes são descritos pelo que chamamos o modelo de recurso RESTful.

O cliente consome a API RESTful através do protocolo HTTP padrão. O cliente se conecta a uma porta do servidor, geralmente a porta 80 e envia um pedido ao servidor, o servidor processa o pedido e envia uma resposta de volta ao cliente, que inclui um código de status indicando se o processamento foi bem sucedido ou não. Estas etapas constituem uma transação HTTP, conforme Figura 25.

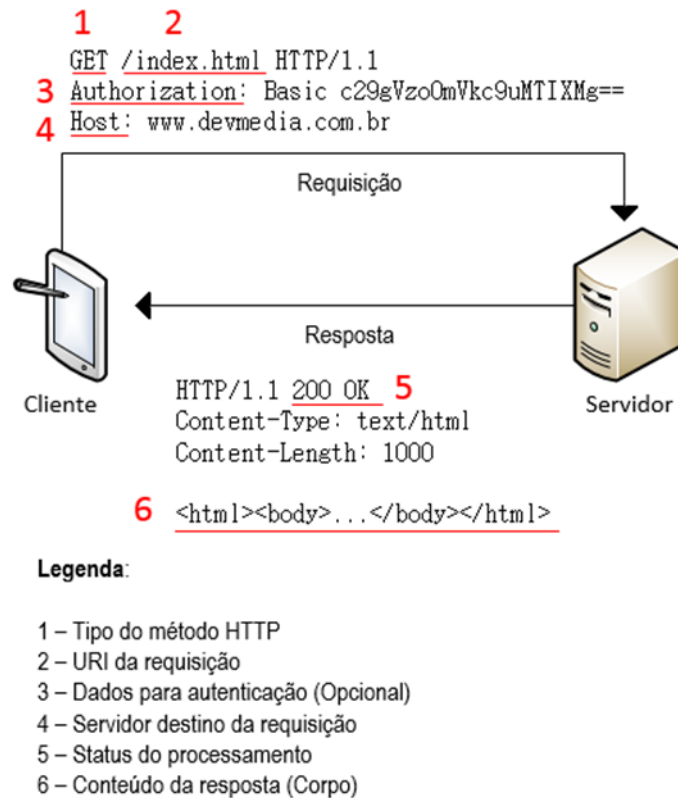


Figura 16 Modelo de uma requisição HTTP (Devmedia 2014)

Os códigos de status iniciam em 100 e vão até 599, podendo ser divididos em cinco categorias:

- 1xx: Informativos
- 2xx: Sucesso
- 3xx: Redirecionamento
- 4xx: Erro no cliente
- 5xx: Erro no servidor

Os servidores Web hospedam recursos que são identificados exclusivamente por meio de um identificador chamado de *Uniform Resource Identifier* (URI).

Para o funcionamento da tecnologia com todos os conceitos explorados, a API deve ser hospedada em um servidor onde a própria aplicação irá escutar uma porta no servidor para receber as requisições.

6.4 DOCUMENTAÇÃO DOS SERVIÇOS

Os serviços serão abordados por meio do conceito básico de RESTful, dividindo-o em entrada e saída. Todo acesso a um servidor web resume-se a uma requisição (entrada) e a uma saída (resposta). Nos serviços, a requisição e a resposta podem ser configuradas da seguinte forma:

A entrada é definida pela URL de acesso ao servidor, e essa URL obedece aos métodos HTTP: GET, POST, PUT e DELETE; Basicamente, possui a seguinte convenção:

- GET é usado para listar dados, listar um registro ou um cálculo.
- POST é usado para adicionar dados.
- PUT é usado para editar dados.
- DELETE é usado para deletar dados.

Na prática, utiliza-se as convenções mostradas na tabela a seguir:

Tabela 2 Convenções utilizadas para elaboração da API

Método	URI	Descrição
Cidades		
GET	/cidades	Obter todas as cidades
POST	/cidades	Adiciona uma nova cidade
GET	/cidades/:id	Obter informações de uma cidade
PUT	/cidades/:id	Editar um registro de uma cidade
DELETE	/cidades/:id	Deletar um registro de uma cidade
Empresas		
GET	/empresas	Obter todas as empresas
POST	/empresas	Adiciona uma nova empresa
GET	/empresas/:id	Obter informações de uma empresa
PUT	/empresas/:id	Editar um registro de uma empresa
DELETE	/empresas/:id	Deletar um registro de uma empresa
Ônibus		
GET	/ônibus	Obter todos os ônibus
POST	/ônibus	Adiciona um novo ônibus
GET	/ônibus/:id	Obter informações de um ônibus
PUT	/ônibus/:id	Editar um registro de um ônibus

DELETE	/onibus/:id	Deletar um registro de um ônibus
Vias		
GET	/vias	Obter todas as vias de ônibus
POST	/vias	Adiciona uma nova via de ônibus
GET	/vias/onibus/:id_onibus	Obter informações de todas vias de ônibus
GET	/vias/:id	Obter informações de uma via de ônibus
PUT	/vias/:id	Editar um registro de via de ônibus
DELETE	/vias/:id	Deletar um registro de via de ônibus
Horários		
GET	/horarios/via/:id_via	Obter informações de horários de uma via
GET	/horarios/via/:id_via/dias-uteis	Obter informações de horários de uma via em dias úteis
GET	/horarios/via/:id_via/sabado	Obter informações de horários de uma via aos sábados
GET	/horarios/via/:id_via/domingo	Obter informações de horários de uma via aos domingos
GET	/horarios/via/:id_via/feriados	Obter informações de horários de uma via aos feriados
POST	/horarios	Adiciona um novo horário de ônibus
PUT	/horarios/:id	Editar um registro de horário de ônibus
DELETE	/horarios/:id	Deletar um registro de horário de ônibus
Itinerários		
GET	/itinerarios/via/:id_via	Obter todas paradas do itinerário de um ônibus
POST	/itinerarios	Adiciona uma nova parada do itinerário do ônibus
PUT	/itinerarios/:id	Editar um registro de parada do itinerário do ônibus
DELETE	/itinerarios/:id	Deletar um registro de parada do itinerário do ônibus

Hoje, para o desenvolvimento de qualquer aplicação existem facilitadores que foram desenvolvidos por terceiros, pagos ou não, que auxiliam e diminuem o tempo de desenvolvimento de uma aplicação (FERRARI, 2014).

Foi escolhida para a etapa de documentação dos serviços a ferramenta *Apiary*: uma ferramenta de prototipação que interpreta comandos da linguagem de marcação *Markdown* e reproduz o resultado através de uma aplicação *RESTful* totalmente navegável e funcional. É possível ter acesso à documentação dos serviços através do site da ferramenta⁵.

⁵ <http://docs.apitransportepublico.apiary.io>

```

## Horários de vias de uma linha [/api/horarios/linha/:linha]
### Obter todos horários das vias de uma linha de ônibus [GET]

+ Request (application/json)

    {
      "linha": "113"
    }

+ Response 404 (application/json)

    {
      "status": false,
      "message": "Não existe horários para esta linha."
    }

+ Response 200 (application/json)

    {
      "status": true,
      "horarios": [
        {
          "linha": "113",
          "nome": "Pioneiros",
          "via": "Linha Convencional",
          "hora": "06:05",
          "local": "Terminal Central"
        },
        {
          "linha": "113",
          "nome": "Pioneiros",
          "via": "Parque Tauá",
          "hora": "17:10",
          "local": "UTFPR"
        }
      ]
    }

```

Figura 17 Documentação em Markdown na ferramenta API Apiary

Horários de vias de uma linha

Obter todos horários das vias de uma linha de ônibus

GET /api/horarios/linha/:linha Add Comment

Example Debugger Comments Try It

Show code sample none (raw data)

Request raw

```
1 GET /api/horarios/linha/:linha HTTP/1.1
2 Content-Type: application/json

1 {
2   "linha": "113"
3 }
```

Response

```
1 404 (Not Found)
2 Content-Type: application/json

1 {
2   "status": false,
3   "message": "Não existe horários para esta linha."
4 }
```

Response

```
1 200 (OK)
2 Content-Type: application/json

1 {
2   "status": true,
3   "horarios": [
4     {
5       "linha": "113",
6       "nome": "Pioneiros",
7       "via": "Linha Convencional",
8       "hora": "06:05",
9       "local": "Terminal Central"
10    },
11    {
12     "linha": "113",
13     "nome": "Pioneiros",
14     "via": "Parque Tauá",
15     "hora": "17:10",
16     "local": "UTFPR"
17    }
18  ]
19 }
```

Figura 18 Resultado da documentação no modo de requisições

Os serviços serão representados por recursos. No sistema REST, cada recurso é unicamente direcionado através da sua URI.

6.4.1 CIDADES

Uma coleção de objetos para Cidades; os objetos deste recurso são utilizados para representar as cidades que possuem transporte público. Possui os seguintes atributos obrigatórios: “id” – atribuído automaticamente, “nome” e “estado”.

6.4.2 EMPRESAS

Uma coleção de objetos para Empresas; os objetos deste Recurso são utilizados para representar as empresas que oferecem o transporte coletivo nas cidades. Possui os seguintes atributos obrigatórios: “id” – atribuído automaticamente, “id_cidade” e “nome”. Cadastrar ou editar objetos existentes criará um histórico de edição de registros com os atributos “data_criacao” e “data_atualizacao”.

6.4.3 ÔNIBUS

Uma coleção de objetos para os Ônibus; os objetos deste Recurso são utilizados para representar as linhas de ônibus que trafegam em uma cidade. Possui os seguintes atributos obrigatórios: “id” – atribuído automaticamente, “id_empresa” e “nome”. Cadastrar ou editar objetos existentes criará um histórico de edição de registros com os atributos “data_criacao” e “data_atualizacao”.

Tabela 3 Exemplo de respostas para o web service “Adicionar um novo onibus”

URI	Método	Response code	Status	Mensagem / Response
		200	false	Não existe empresa com esse id.
/onibus	POST	200	false	Os campos "id_empresa" e "nome" são obrigatórios.
		201	true	Ônibus cadastrado com sucesso.

6.4.4 VIAS

Serviços para acessar os recursos das vias das linhas de ônibus. Um ônibus pode possuir uma ou mais linhas. Por convenção, os ônibus possuem uma linha normal, que faz o itinerário com as paradas em uma ordem pré-determinada. Pode haver outras vias, como a Via Expressa, que faz o itinerário do ponto de origem até o ponto destino sem realizar nenhuma parada. Possui os seguintes atributos obrigatórios: “id” – atribuído automaticamente, “id_onibus” e “nome”.

Tabela 4 Exemplo de respostas para o web service “Obter informações de uma via de onibus”

URI	Método	Response code	Status	Mensagem / Response
		400	false	Erro desconhecido. Por favor tente novamente.
/vias/:id	GET	200	false	Não existe(m) via(s) para o ônibus.
		200	true	Array de objeto com o ônibus.

6.4.5 HORÁRIOS

Uma coleção de objetos para os Horários; os objetos deste Recurso são utilizados para representar horários que ocorrem as saídas e chegadas de ônibus de um ponto de origem até um ponto de destino. Possui os seguintes atributos obrigatórios: “id” – atribuído automaticamente, “id_via”, “ponto_inicial” e “hora”. Possui os seguintes atributos opcionais: “dias_uteis”, “sabado”, “domingo” e “feriado”.

Tabela 5 Web service para Obter horários de uma via

URI	Método	Response code	Status	Mensagem / Response
		400	false	Erro desconhecido. Por favor tente novamente.
/horarios/via/:id_via	GET	200	false	Não existe(m) horário(s) cadastrado(s) para o id desta via.
		200	true	Array de objetos com os horários da via

6.4.6 ITINERÁRIOS

Uma coleção de objetos para os Itinerários; os objetos deste Recurso são utilizados para representar as paradas que ocorrem durante o itinerário de uma via de ônibus. Possui os seguintes atributos obrigatórios: “id” – atribuído automaticamente, “id_via”, “ordem” e “nome”. Possui os seguintes atributos opcionais: “latitude”, e “longitude”.

Tabela 6 Exemplo de respostas para o web service “Adicionar uma nova parada do itinerário do onibus”

URI	Método	Response code	Status	Mensagem / Response
/itinerarios	POST	200	false	Não existe via com esse id.
		200	false	Os campos "id_via", "ordem" e "nome" são obrigatórios.
		201	true	Itinerário cadastrado com sucesso.

Tabela 7 Exemplo de respostas para o web service “Editar um registro de parada do itinerário do onibus”

URI	Método	Response code	Status	Mensagem / Response
/itinerarios/:id	PUT	200	false	Não existe via de ônibus com esse id
		200	false	Os campos "id_via", "ordem" e "nome" são obrigatórios.
		200	false	Não existe parada de itinerário com esse id.
		200	true	Parada do itinerário editada com sucesso.

Tabela 8 Web service para Deletar um registro de parada do itinerário do onibus

URI	Método	Response code	Status	Mensagem / Response
/itinerarios/:id	DELETE	400	false	Erro desconhecido. Por favor tente novamente.
		200	false	Não existe parada de itinerário com esse id.
		200	true	Parada do itinerário removida com sucesso.

6.5 IMPLEMENTAÇÃO DAS TABELAS

Para criar as tabelas de banco de dados foi utilizada a ferramenta MySQL Workbench. É possível especificar os formatos para cada coluna nas tabelas do banco de dados e criar relacionamentos entre as entidades (tabelas). Por fim, é possível gerar um script para implementar automaticamente o banco de dados e as respectivas tabelas.

```
CREATE TABLE IF NOT EXISTS `db_api_transporte_coletivo`.`horarios` (  
  `id` INT NOT NULL AUTO_INCREMENT,  
  `id_via` INT NOT NULL,  
  `ponto_inicial` VARCHAR(90) NOT NULL,  
  `dias_uteis` TINYINT(1) NULL,  
  `sabado` TINYINT(1) NULL,  
  `domingo` TINYINT(1) NULL,  
  `feriado` TINYINT(1) NULL,  
  `hora` VARCHAR(5) NULL,  
  PRIMARY KEY (`id`, `id_via`),  
  INDEX `fk_horarios_vias1_idx` (`id_via` ASC),  
  CONSTRAINT `fk_horarios_vias1`  
    FOREIGN KEY (`id_via`)  
    REFERENCES `db_api_transporte_coletivo`.`vias` (`id`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

Figura 19 Código SQL para criar tabela de Horários de ônibus

6.6 IMPLEMENTAÇÃO DOS SERVIÇOS

Para implementar o sistema foi utilizado o *Express.js*: ferramenta minimalista capaz de escrever rápidas aplicações com Node.js.

Através de poucos comandos é possível criar uma estrutura inicial de aplicação:

```
$ npm install express-generator -g
```

Npm é o nome reduzido de *Node Package Manager* (Gerenciador de Pacotes do *Node*). A *npm* é duas coisas: Primeiro, e mais importante, é um repositório online para publicação de projetos de código aberto para o *Node.js*; segundo, ele é um utilitário de linha de comando que interage com este repositório online, que ajuda na instalação de pacotes, gerenciamento de versão e gerenciamento de dependências.

O comando a seguir cria uma aplicação nomeada *myapp* no diretório corrente:

```
$ express myapp
```

```
create : myapp
create : myapp/package.json
create : myapp/app.js
create : myapp/public
create : myapp/public/javascripts
create : myapp/public/images
create : myapp/routes
create : myapp/routes/index.js
create : myapp/routes/users.js
create : myapp/public/stylesheets
create : myapp/public/stylesheets/style.css
create : myapp/views
create : myapp/views/index.jade
create : myapp/views/layout.jade
create : myapp/views/error.jade
create : myapp/bin
create : myapp/bin/www
```

Figura 20 Aplicação inicial gerada pelo Express.js

É necessário especificar no arquivo *package.json* as dependências que serão utilizadas. Estas dependências são módulos que são acopladas à aplicação para executarem determinados serviços quando requisitado.

Em resumo:

- *body-parser*: permite a construção de objetos *JSON* a partir da submissão de dados de um formulário *HTML*

- *cookie-parser*: analisa o cabeçalho de um cookie e popula com um objeto literal contendo nome e valor
- *debug*: realiza a depuração de todos os processos, serve para encontrar e reduzir defeitos na aplicação
- *express-myconnection*: ferramenta necessária para realizar fornecer informações do banco de dados para a aplicação
- *jade*: framework para renderizar as views (telas) da aplicação
- *moment*: biblioteca para trabalhar com formatações de data e hora
- *morgan*: ferramenta auxiliar na depuração, juntamente com o debug atua na varredura de métodos e retornos dos web services
- *mysql*: dependência responsável por fazer a conexão da aplicação com o banco de dados
- *serve-favicon*: ferramenta para criar um ícone e identificar a aplicação no navegador

```
{
  "name": "api",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
    "body-parser": "~1.13.2",
    "cookie-parser": "~1.3.5",
    "debug": "~2.2.0",
    "express": "~4.13.1",
    "express-myconnection": "^1.0.4",
    "jade": "~1.11.0",
    "moment": "^2.10.6",
    "morgan": "~1.6.1",
    "mysql": "^2.9.0",
    "serve-favicon": "~2.3.0"
  }
}
```

Figura 21 Configuração do arquivo package.json

Os dois próximos comandos servem para executar a aplicação no servidor e escutar uma porta de conexão com o recebimento de web services.

```
$ cd my app
```

```
$ set DEBUG=myapp:* & npm start
```

Com os comandos anteriores é possível executar a aplicação em um servidor⁶. Por conseguinte, todas as implementações da aplicação serão realizadas no diretório *routes*.

⁶ <http://localhost:3000>

Uma rota consiste em um método HTTP, um caminho e uma função de retorno. O método HTTP refere-se às URIs para um aplicativo e como ele responde às solicitações do cliente.

Os objetos *req* e *res* são objetos que são passados por parâmetro para fazer tratamento de requisição e resposta de cada rota.

Tabela 9 Script para criar rota dos horários de uma via de ônibus

```
router.route('/:id_via/dias-uteis')
  // Obter informações de horários de uma via em dias úteis
  .get(function(req, res) {
    var id_via = req.params.id_via;
    req.getConnection(function(err, connection) {
      var query = connection.query('SELECT * FROM horarios WHERE
id_via = ? AND dias_uteis = 1', id_via, function(err, rows) {
        if (err) {
          res.status(400).json({
            status: false,
            message: 'Erro desconhecido. Por favor tente
novamente.'
          });
        } else {
          if (rows.length > 0) {
            res.json({
              status: true,
              horarios: rows
            });
          } else {
            res.status(200).json({
              status: false,
              message: 'Não existe(m) horário(s)
cadastrado(s) ou para o id desta via.'
            });
          }
        }
      });
    });
  });
});
```

6.7 TESTES DOS SERVIÇOS IMPLEMENTADOS

Durante as etapas de implementação, foi desenvolvida uma aplicação web com páginas que podem exibir as informações através de requisições à API. Utilizou-se o framework front-end Bootstrap 3.3.5 (Twitter, 2015) para confeccionar

toda a estrutura da aplicação, desde a camada de apresentação das informações⁷ até a camada de comportamento⁸.

Uma vez que o comando para inicializar a API é executado, é possível também executar a aplicação. Este funcionamento é permitido graças à arquitetura do framework Express.js que permitiu customizar através de rotas a separação do web service da aplicação web. Assim, a aplicação web pode consumir dados do web service quando necessário e, se houver problemas⁹, os web services responderão com notificações de erro para a aplicação.

Para o presente projeto foi desenvolvido na aplicação web apenas funcionalidades mais simples, porém consistentes e essenciais, que permitiram o autor a atingir o objetivo proposto. As funcionalidades implementadas foram:

- Visualizar cidades cadastradas
- Visualizar empresas cadastradas
- Visualizar linha de ônibus cadastradas
- Visualizar vias de ônibus cadastradas
- Pesquisar horários e itinerários das vias de ônibus.
- Filtrar horários por dias úteis, sábado, domingo e feriados.

Nas figuras a seguir é possível buscar por uma linha de ônibus. O usuário deve informar o número da linha e clicar no botão “Buscar”. Os web services receberão parâmetros e caso exista resultado, retornará uma tabela contendo as vias que estão atreladas à linha. É possível ainda exibir os horários e itinerários da via. O usuário precisa apenas de clicar no ícone de relógio para exibir os horários ou no ícone de lista para exibir o itinerário.

⁷ HTML e CSS

⁸ JavaScript

⁹ Mal funcionamento: queda de conexão, perda de referência numérica, etc.

API Cidades Empresas Linhas Vias **Horários**

Horários

Você pode pesquisar os horários de uma via pesquisando o código da linha de ônibus ou digitando o nome da linha. Basta apenas selecionar o modo de busca e digitar o texto.

Selecione o modo de busca **Digite o texto**

Por número da linha

Buscar

Linha	Ônibus	Via
113	Pioneiros	Linha Convencional
113	Pioneiros	Parque Tauá
113	Pioneiros	Expresso UTFPR - Boulevard

Figura 22 Tela de pesquisa de horários

API Cidades Empresas Linhas Vias **Horários**

Horários

Você pode pesquisar os horários de uma via pesquisando o código da linha de ônibus ou digitando o nome da linha. Basta apenas selecionar o modo de busca e digitar o texto.

Selecione o modo de busca **Digite o texto**

Por número da linha

Buscar

Linha

- 113
- 113
- 113

113 Pioneiros - Linha Convencional

Consulta de horários dos ônibus metropolitanos.

Dias úteis **Sábado** Domingo Feriados

Local	Horário
Terminal Central	06:05
UTFPR	06:25
Terminal Central	06:45
UTFPR	07:01
Terminal Central	07:45
UTFPR	08:01
Terminal Central	08:26
UTFPR	08:42

* Esporadicamente horários de saída/chegada podem sofrer atrasos de alguns minutos.

Figura 23 Tela com horários da linha pesquisada

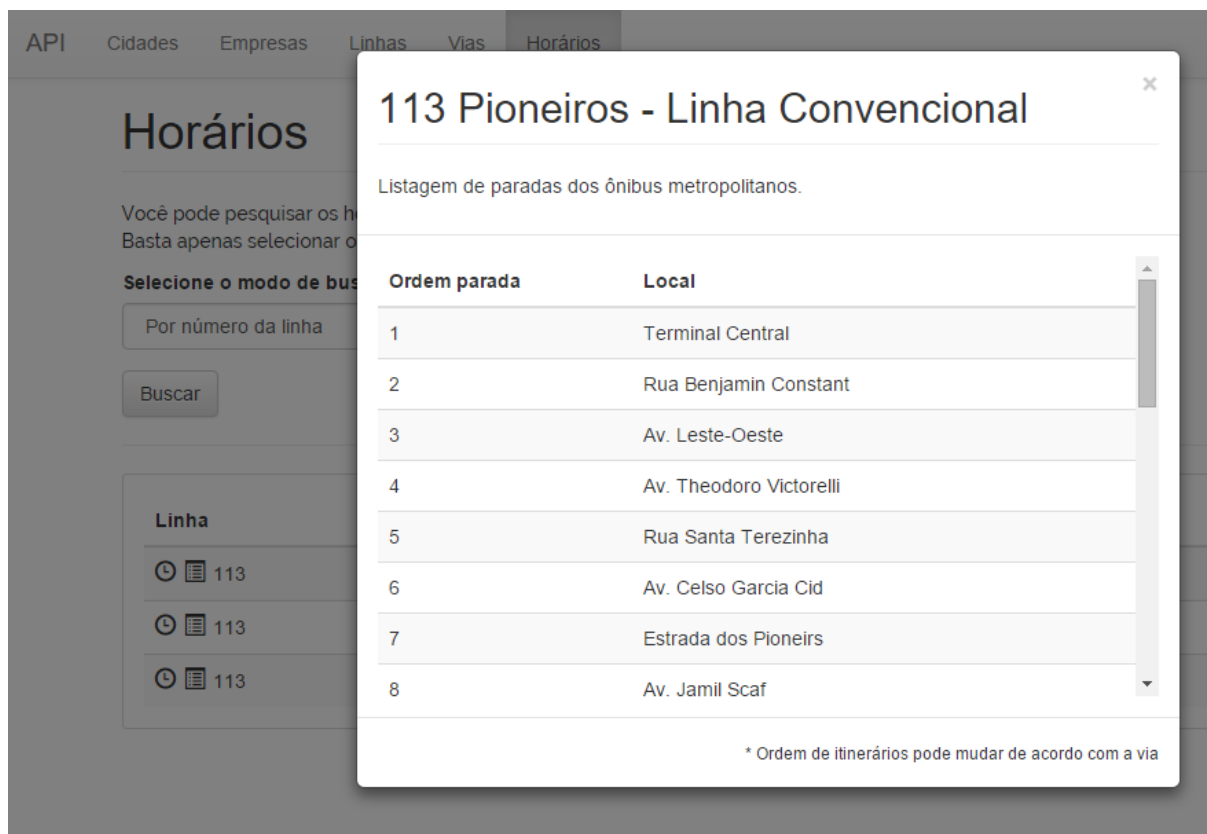


Figura 24 Tela com itinerário da linha pesquisada

Na etapa de testes utilizou-se o POSTMAN, uma ferramenta extremamente eficiente para realizar testes de API. É utilizada por empresas de grande porte como Microsoft, Cisco, Oracle etc.

Dentre as características da ferramenta é possível:

- Fazer requests simples ou complexos com agilidade
- Salvar requisições
- Utilizar, organizar e automatizar Coleções

A ferramenta é um aplicativo do navegador Google Chrome¹⁰, com suporte aos sistemas operacionais Windows, Linux e OS X.

Na figura 36 contém uma Coleção de web services utilizados para gerenciar recursos em Cidades, onde os métodos para listar, editar, cadastrar e deletar são testados e validados, por fim armazenados em uma coleção.

¹⁰ <https://www.getpostman.com>

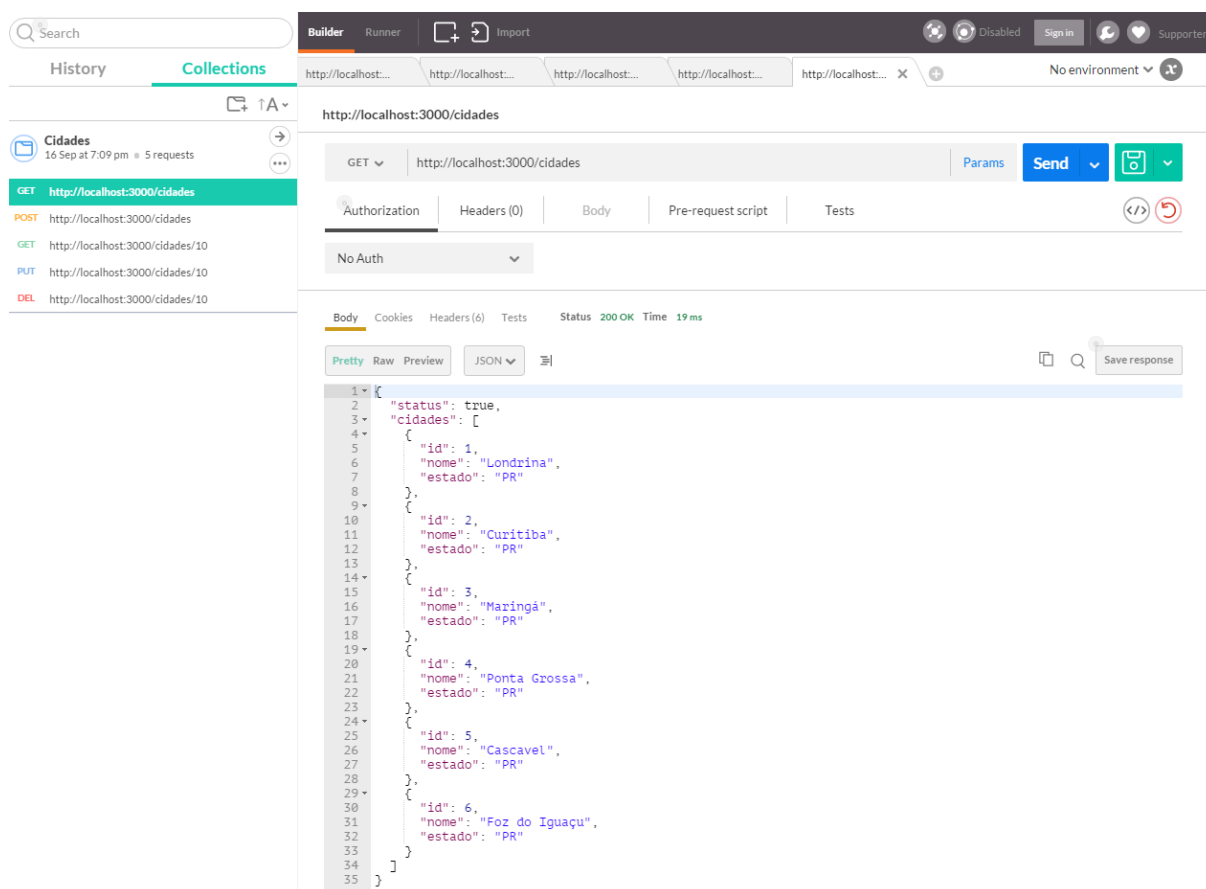


Figura 25 Teste de requisição usando a ferramenta POSTMAN

6.8 MELHORIAS FUTURAS

O presente projeto visa inicialmente atender as necessidades da Universidade Federal Tecnológica do Paraná, do Câmpus de Londrina – que irá consumir as informações da API de forma privada – e somente exibir os horários da linha de ônibus para os acadêmicos da instituição. Assim sendo, não foram abordados aspectos de segurança de informação e o controle da segurança fica totalmente implementado pela infraestrutura. Restrições de acesso por IP implementados em roteadores, firewalls e balanceadores de carga; Restrição também de protocolos de comunicação permitidos e portas, logging detalhado dos acessos realizados são exemplos que poderão ser implementados na infraestrutura para proteger a informação.

A estrutura do projeto favorece o progresso do sistema; futuramente pode ser utilizada por empresas, serviços públicos, alunos, programadores ou qualquer membro da comunidade que possuir interesse em alimentar com informações reais

os dados da API. Como os dados serão sensíveis e importantes para a sociedade, é interessante elaborar uma proteção nos serviços para que não haja brecha em nenhum mecanismo.

Para integridade e bom funcionamento dos serviços sugere-se implementar uma autenticação/autorização com a técnica chamada *Token-based authorization*. Uma forma simples e segura de controlar autenticação/autorização de serviços entre servidores, embora não seja um padrão. As etapas desta técnica são:

1. Aplicação cliente autentica-se no servidor informando suas credenciais de acesso;
2. Aplicação servidora valida credenciais e em caso de sucesso guarda um token de autorização, vinculando-o ao IP de origem do cliente. O token pode ser válido por uma operação ou por um período (ex: 24 horas);
3. O servidor envia mensagens de sucesso para o cliente e envia o token;
4. O cliente invoca serviços no servidor sempre enviando o token de autorização
5. O servidor valida cada requisição checando se foi enviado um token válido. Se um token não foi enviado, já tiver expirado ou for inválido, o servidor retorna mensagem ao cliente exigindo autorização, da mesma forma que com *HTTP Basic* ou *Digest*.

Além de segurança, outros tópicos poderão ser explorados para aperfeiçoar o funcionamento e utilização da API, por exemplo: paginação dos resultados para não gerar volume grande de dados, desenvolver um algoritmo para localizar melhor as rotas – os elementos latitude e longitude do projeto foram pensados para contribuir com isso – bem como disponibilizar outras informações interessantes para passageiros, como preço da passagem e informações de suporte à acessibilidade.

Próximo a finalização do trabalho foi possível encontrar uma ferramenta chamada *GTFIS*, iniciativa da empresa norte-americana Google que pode ser utilizada para compartilhar dados estáticos de transportes públicos no Google Maps.

A Especificação Geral de *Feeds* de Transporte Público (*GTFIS*) define um formato comum para horários de transportes públicos e informações geográficas associadas. Os *feeds* *GTFIS* permitem que as agências de transporte público publiquem suas informações e que os desenvolvedores criem aplicativos que consumam esses dados com interoperabilidade.

Da busca por referências nacionais que utilizam a ferramenta GTFS foi possível encontrar uma página exclusiva¹¹ criada por desenvolvedores da SPTrans para fornecer os dados de transporte público da cidade de São Paulo. Para ter acesso à API foi necessário criar uma conta no site, confirmar o cadastro e fazer o primeiro acesso. Após o login inicial foi cadastrado um aplicativo. Cada aplicativo cadastrado recebe uma chave de acesso que deverá ser utilizada para efetuar a autenticação no serviço.

Analisando a estrutura de informações que são fornecidas pela API SPTrans foi possível compreender que existem semelhanças com a nomenclatura e tipos de dados que são exibidos na resposta da requisição. De fato, chega-se à conclusão que o trabalho deve se encorajado a continuar com melhorias de implementações para futuras integrações.

Tabela 10 Exemplo de requisição da API SPTrans

```
GET /Linha/Buscar?termosBusca=8000
RETURN

[
  {
    "CodigoLinha": 1273,
    "Circular": false,
    "Letreiro": "8000",
    "Sentido": 1,
    "Tipo": 10,
    "DenominacaoTPTS": "PCA.RAMOS DE AZEVEDO",
    "DenominacaoTSTP": "TERMINAL LAPA",
    "Informacoes": null
  },
  {
    "CodigoLinha": 34041,
    "Circular": false,
    "Letreiro": "8000",
    "Sentido": 2,
    "Tipo": 10,
    "DenominacaoTPTS": "PCA.RAMOS DE AZEVEDO",
    "DenominacaoTSTP": "TERMINAL LAPA",
    "Informacoes": null
  }
]
```

¹¹ <http://www.sptrans.com.br/desenvolvedores/APIOlhoVivo/Documentacao.aspx>

7 CONCLUSÃO

O desenvolvimento deste projeto e implementação da aplicação proporcionou aplicar todos os conhecimentos e técnicas aprendidas, além de expandir a visão no desenvolvimento de aplicações web, sendo assim possível aperfeiçoar o código e incrementar funcionalidades. Houve um aprendizado também na utilização de frameworks para web, onde foi necessário realizar uma pesquisa entre diversas opções a fim de encontrar um framework compatível, leve e visualmente amigável ao usuário.

Houveram algumas dificuldades no decorrer da implementação, como a utilização da função de filtrar os horários e itinerários de uma requisição, onde foi necessário aprofundar o estudo das consultas no banco de dados e implementar nas rotas do código-fonte da aplicação. Também foi necessário realizar um levantamento teórico acerca da utilização de web services, alteração de registros e mensagens do servidor, garantindo assim uma padronização de mensagens na solução.

As ferramentas utilizadas para o desenvolvimento da aplicação são todas gratuitas, onde foram adquiridas por meio de download em seus sites oficiais, evitando-se assim um problema de custo na implementação. A ferramenta Express.js já trouxe todo o conjunto de ferramentas necessárias para o desenvolvimento, trazendo consigo o servidor Node.js, o banco de dados MySQL e a linguagem JavaScript, bastando apenas uma breve leitura em seus manuais para a correta configuração.

A aplicação demonstrou um grande potencial para resolver os problemas envolvendo os processos necessários para que fosse possível disponibilizar recursos para exibir informações de transporte público em sites ou aplicativo de smartphones. Desta forma a empresa prestadora de transportes, se interessada, pode utilizar da aplicação para concentrar em serviços para a comunidade.

REFERÊNCIAS

DENHARDT, R. **Teoria Geral da Administração Pública**. São Paulo, SP - Brasil: Thomson/Wadsworthm, 2008.

DAVENPORT, T. **Reengenharia de Processos**. São Paulo, Campus, 1994.

GAMA, Kiev; ALVARO Alexandre; PEIXOTO, Eduardo. **Em Direção a um Modelo de Maturidade Tecnológica para Cidades Inteligentes**. CESAR – Centro de Estudos e Sistemas Avançados do Recife. Volume 1. Número 1, 2012. Disponível em <<http://www.lbd.dcc.ufmg.br/colecoes/sbsi/2012/0018.pdf/>>. Acesso em: 08 dez. 2015.

ROUSE, Margaret. **Web applications (Web app)**. 2011. Disponível em <<http://searchsoftwarequality.techtarget.com/definition/Web-application-Web-app>>. Acesso em: 23 set. 2014.

NATIONS, Daniel. **Web applications**. 2013. Disponível em: <http://webtrends.about.com/od/webapplications/a/web_application.htm>. Acesso em: 23 set. 2014.

ACUNETIX. **Web applications: What are They? What of Them?** 2013. Disponível em: <<http://www.acunetix.com/websitesecurity/web-applications>>. Acesso em: 24 set. 2014.

ANDRETO, Douglas Eduardo; AMARAL, Aline Maria Malachini. **Engenharia de Software para web**. Maringá, PR – Brasil: 2006. p. 157-166. Disponível em <<http://cesumar.br/pesquisa/periodicos/index.php/iccesumar/article/download/268/87>>. Acesso em: 25 set. 2014.

HOWE, Denis. **FOLDOC: Free On-Line Dictionary Of Computing**. Disponível em <<http://foldoc.org/Application%20Program%20Interface>>. Acesso em: 27 jun. 2015.

NEGRINO, Tom; SMITH, Dori. **JavaScript Para World Wide Web, Tradução 3a. Edição, Visual QuickStart Guide**. São Paulo, Campus, 1999.

PEREIRA, Cristian. **NoSQL e o conceito de banco de dados não relacional**. Disponível em <http://www.fxplabs.com.br/blog/nosql-conceito-de-banco-de-dados-nao-relacional/>. Acesso em: 24 ago. 2015.

MILANI, André. **MySQL - Guia do Programador**. São Paulo, Novatec, 2007.

BRASIL. **Lei n. 12.537, de 18 de novembro de 2011. Regula o acesso a informações previsto no inciso XXXIII do art. 5o, no inciso II do § 3o do art. 37 e no § 2o do art. 216 da Constituição Federal; altera a Lei no 8.112, de 11 de dezembro de 1990; revoga a Lei no 11.111, de 5 de maio de 2005, e dispositivos da Lei no 8.159, de 8 de janeiro de 1991; e dá outras providências**. Disponível em: http://www.planalto.gov.br/ccivil_03/_ato2011-2014/2011/lei/l12527.htm. Acesso em: 05 nov. 2014.

IBGE, Diretoria de Pesquisas, Coordenação de População e Indicadores Sociais. **População de Municípios do Paraná**. Disponível em: <http://cidades.ibge.gov.br/xtras/perfil.php?codmun=410690/>. Acesso em: 25 ago. 2015.

CURITIBA, Biocidade. **Sistema de Transporte Coletivo de Curitiba**. Disponível em: <http://www.biocidade.curitiba.pr.gov.br/biocity/33.html/>. Acesso em: 25 ago. 2015.

TEOREY, Toby; LIGHTSTONE, Sam; NADEAU, Tom. **Projeto e Modelagem de Bancos de Dados** 2ª Ed. São Paulo, Campus, 2014.

ABINADER, Jorge Abílio; LINS, Rafael Dueire. **Web Services em Java**. Rio de Janeiro, Brasport, 2006.

FERRARI, Felipe Farias. **Autenticação e criptografia na Web API**. Revista .Net Magazine ed. 107, São Paulo. 2014. Disponível em <http://www.devmedia.com.br/autenticacao-e-criptografia-na-web-api/28709/> Acesso em 29 ago. 2015.

APÊNDICE A - Questionário aplicado a TI da empresa TCGL para Levantamento de Requisitos

Questionário sobre o site da TCGL

Quem fornece os dados sobre as linhas e itinerários da TCGL?

R: TCGL (sistemas ERPs)

Existe um painel administrativo para inserção/atualização das informações?

R: SIM

Existe uma maneira de obter esses dados de forma aberta? (para utilização em outros serviços, usar como uma API, por exemplo)

R: SIM

Quais as tecnologias estão sendo utilizadas no portal?

R: XML, SOAP

O sistema foi desenvolvido em qual(is) linguagem(ns) de programação?

R: ASP.NET

Qual o banco de dados utilizado?

R: SQL

Existem protocolos de comunicação de dados? (SOAP, JSON, XML, etc...) **Quais?**

R: SOAP, XML

Qual é o servidor de aplicações?

R: IIS

Existe alguma ferramenta ou alguém responsável por acompanhamento de tráfego de acessos? Se sim, responder as próximas 2 perguntas.

R: NÃO

Quantidade de acessos por dia / semana / mês.

R:

Existe acessos por celular?

R: SIM (o portal não é responsivo, mesma página para dispositivos móveis e desktop)

Existe alguma maneira de identificar quais os onibus da TCGL operam com suporte à acessibilidade (espaço cadeirante).

R: sim

Se a resposta anterior for não, onde posso obter essa informação?

R:

Quem define/definiu os padrões de interface do portal?

R: Terceiro (empresa criação de sites)

Existem queixas de má interpretação ou algo semelhante no site?

R: Não

Caso queira deixar alguma observação, favor usar o campo abaixo:

Dados de quem respondeu este questionário:

Data: 17/07/2014

Nome: Fabio Alves

Cargo: Analista

Empresa: TCGL

Telefone para contato: 43-33782400

APÊNDICE B - O Transporte Público de Curitiba

De acordo com IBGE (2014), Curitiba possui cerca de 1,8 milhões de habitantes e detém um dos sistemas mais eficientes de planejamento urbano, transporte e mobilidade do Brasil.

Atualmente, o sistema está integrado com 13 municípios da Região Metropolitana. Ao todo, cerca de 2 milhões e 300 mil passageiros utilizam todos os dias os quase 2 mil ônibus que percorrem 480 mil quilômetros a cada 24 horas.

É notável o cuidado com que a Prefeitura zela pela eficiência do sistema, com os olhos voltados para o meio ambiente. A idade média da frota é de pouco mais de 6 anos. Ônibus novos, mais conforto para os passageiros, menos poluição (BIOCIDADE, 2015).

	Tipo de linha	capac./veic
	Circular Centro	30
	Convencional	80
	Alimentador	80
	Interbairros Padron	110
	Interbairros Articulado	160
	Linha Direta	110
	Expresso Padron	110
	Expresso Articulado	160
	Expresso Biarticulado	270

Figura 26 Classificação dos Circulares que operam na região metropolitana de Curitiba - PR

A Rede Integrada de Transporte Coletivo de Curitiba (RIT) permite ao usuário a utilização de mais de uma linha de ônibus com o pagamento de apenas uma tarifa. O processo de integração ocorre a partir de terminais de integração onde o cidadão pode desembarcar de uma linha e embarcar em qualquer outra dentro daquele espaço sem um novo pagamento. Assim, o usuário pode compor o seu próprio trajeto para se deslocar por diversos bairros de Curitiba.

Curitiba mantém a infraestrutura de transporte da RIT à disposição do Sistema de Transporte Coletivo Metropolitano para integrações físicas tarifárias. Através do site¹² da empresa URBS – Urbanização de Curitiba S/A – é possível pesquisar a relação de horários de todas as linhas que circulam na região metropolitana. No canto superior direito, o usuário deve inicialmente realizar 2 instruções: selecionar o bairro e dias úteis para obter a relação dos horários. Por fim ele é direcionado à uma nova página com a relação de todas as tabelas de horários da região selecionada.

Ter, 25 de Agosto de 2015

Acesso Rápido

mobile mapa do site english español

URBS

Itinerários
linhas de ônibus
consulte!

Utilidades Institucional Comunidade Transporte Acessibilidade Notícias Horário de ônibus Fale conosco

home > horário de ônibus > Rede Integrada de Transporte

Horário de ônibus
Rede Integrada de Transporte

DIA ÚTIL Busca QBusca Imprimir

165 - UNIVERSIDADES

Esta linha só aceita o pagamento da tarifa com cartão transporte

Ponto: **BAIRRO RAQUEL PRADO**
Válido a partir de: 27/05/2015
DIA ÚTIL

05:22	<u>05:46</u>	<u>06:10</u>	06:35	<u>07:05</u>	<u>07:35</u>	08:05
<u>08:40</u>	<u>09:15</u>	09:45	<u>10:15</u>	<u>10:45</u>	11:18	<u>11:48</u>
<u>12:15</u>	12:46	<u>13:16</u>	<u>13:46</u>	14:16	<u>14:46</u>	<u>15:16</u>
15:46	<u>16:16</u>	<u>16:46</u>	17:18	<u>17:54</u>	<u>18:30</u>	19:05
<u>19:40</u>	<u>20:14</u>	20:48	<u>21:18</u>	<u>21:48</u>	22:18	<u>22:48</u>

Os horários sublinhados são operados por veículos equipados com elevador.

Horário de ônibus
Linhas urbanas Linhas Rodoviárias

UNIVERSIDADES
DIA ÚTIL
Linhas Metropolitanas Consultar

92,66% da frota do transporte já está equipada com elevador

Figura 27 Horários de uma linha de ônibus no site URBS

¹² <http://www.urbs.curitiba.pr.gov.br/horario-de-onibus/>

Apesar de objetiva, as informações não são recuperáveis; a relação de tabelas disponibilizadas na lista é estática; não é possível consumir nenhum recurso de serviço web. Toda vez que tornar necessário efetuar uma pesquisa, o usuário deve informar o bairro e o período (dia útil, sábado ou domingo).

APÊNDICE C - O Transporte Público de Maringá

Apelidada de “Cidade Canção”, o município de Maringá possui aproximadamente 391 mil habitantes e é localizado a 100 quilômetros de Londrina. O serviço de transporte coletivo é gerenciado pela SETRANS – Secretaria do Trânsito e da Segurança. A SETRANS é responsável por definir o preço das passagens de ônibus e também da distribuição de linhas por horários e itinerários.

Existem dois meios de visualizar a relação de horários de ônibus em Maringá: o primeiro meio é o próprio site¹³ da prefeitura e o segundo é no site da TCCC – Transportes Coletivos Cidade Canção¹⁴.

No site da SETRANS há uma lista de todas as linhas em formato de tabela. Ao acessar um hyperlink de uma linha em específica, o navegador abre uma nova página com um arquivo .pdf com a tabela de horários e itinerário.

DIAS ÚTEIS						SÁBADOS					
AV. GUAIAPO			AV. MANDACARU (HU)			AV. GUAIAPO			AV. MANDACARU (HU)		
06:30	21:30		07:00	22:00		06:30	19:30		07:05	19:05	
07:30	22:30		08:00	23:00		07:30	20:20		07:55	19:55	
08:30			09:00			08:20	21:10		08:45	20:45	
09:30			10:00			09:10	22:00		09:35	21:35	
10:30			11:00			10:00	22:55		10:25	22:25	
11:30			12:00			10:50			11:15	23:20	
12:30			13:00			11:40			12:05		
13:30			14:00			12:30			12:55		
14:30			15:00			13:35			13:10		
15:30			16:00			14:25			14:00		
16:30			17:00			15:15			14:50		
17:30			18:00			16:05			15:40		
18:30			19:00			16:55			16:30		
19:30			20:00			17:45			17:20		
20:30			21:00			18:35			18:10		

DOMINGOS E FERIADOS					
AV. GUAIAPO			AV. MANDACARU (HU)		
06:30	19:30		07:05	19:55	
07:30	20:20		07:55	20:45	
08:20	21:10		08:45	21:35	
09:10	22:00		09:35	22:25	
10:00	22:55		10:25	23:20	
10:50			11:15		
11:40			12:05		
12:30			13:10		
13:35			14:00		
14:25			14:50		
15:15			15:40		
16:05			16:30		
16:55			17:20		
17:45			18:10		
18:35			19:05		

Reclamações e sugestões pelo fone 156

Atualizado em 13/10/05

Horários sujeitos a alterações

Itinerário linha 007					
Sentido: Av. Guaiapó - Av. Mandacaru (HU):					
Av. Guaiapó (Supermercado São José) - Praça Pion. Bento de Freitas da Silva - Av. D. Sophia Rasgulaeff - Praça São Vicente - Av. D. Sophia Rasgulaeff - Praça Vila Rica - Av. Morangureira - Praça Jard. Altino Cardoso - Av. Kakogawa - Praça Emilio Fajardo Espejo - Av. Kakogawa - Praça Megumi Tanaka - Av. das Palmeiras - Praça das Palmeiras - Av. das Palmeiras - Praça Vitor Rodrigues Martins (Fac. Nobel) - Av. Mandacaru - Praça Atleta Reinaldo G. Bittencurt - Av. Mandacaru - Praça Jacinto Ferreira Branco - Av. Mandacaru (Hospital Universitário).					
Sentido: Av. Mandacaru (HU) - Av. Guaiapó:					
Av. Mandacaru (Hospital Universitário) - Praça Atleta Reinaldo G. Bittencurt - Av. Mandacaru - Praça Vitor Rodrigues Martins (Fac. Nobel) - Av. das Palmeiras - Praça das Palmeiras - Av. das Palmeiras - Praça Megumi Tanaka - Av. Kakogawa - Praça Emilio Fajardo Espejo - Av. Kakogawa - Praça Jard. Altino Cardoso - Av. Morangureira - Praça Vila Rica - Av. D. Sophia Rasgulaeff - Praça São Vicente - Av. D. Sophia Rasgulaeff - Praça Pion. Bento de Freitas da Silva - Av. Guaiapó (PI17).					

Figura 28 Tabela de horários em arquivo .pdf no site SETRANS

¹³ http://www.maringa.pr.gov.br/arquivos/horario_onibus/horarios_onibus.htm

¹⁴ <http://www.tccc.com.br>

No site da companhia de Transportes Coletivos Cidade Canção há um formulário de consulta no canto superior esquerdo, onde o usuário deve escolher a linha, origem, destino e dia da semana que deseja receber a relação de horários. Ao clicar no botão Consultar, o usuário recebe a relação de horários e itinerário da linha na mesma tela em que efetuou a pesquisa.

CONSULTA DE HORÁRIOS

UEM-UNIVERSIDADE

Origem: TERMINAL

Destino: CAMPUS DA UEM

Tipo Dia: Util

CONSULTAR

EMPRESAS
Agora Ficou Fácil
Compre seu Crédito

ONLINE

SOLICITAR CARTÃO
passefácil

Linha: UEM-UNIVERSIDADE
Sentido: TERMINAL - CAMPUS DA UEM

Impressão

Dia Útil

PARTIDA	CHEGADA	Obs:	ITINERÁRIO
07:35	07:55		TERMINAL
17:10	17:35		AV. DUQUE DE CAXIAS
17:50	18:05		PANIFICADORA CRISTAL
18:25	18:45		AUTO ESCOLA BRASÍLIA
19:10	19:30		AV. PROF. LAURO E. WERNECK
			CAMPUS DA UEM
			R. PROFESSOR OBERAN
			R. DR. MÁRIO CLAPIER URBINATTI
			R. PARANAGUÁ
			R. MANDAGUARI
			R. MARQUES DE ABRANTES
			R. OSVALDO CRUZ
			R. QUINTINO BOCAIÚVA

Figura 29 Relação de horários no site TCCC

Em ambos os sites foi possível perceber que não existe nenhum serviço web que automatiza a maneira que as informações são solicitadas, forçando o usuário a sempre realizar o mesmo procedimento para obter o resultado desejado.

APÊNDICE D - O Transporte Público de Ponta Grossa

Com o 4º lugar de maior município do estado do Paraná, localizada no centro do estado, Ponta Grossa possui cerca de 334 mil habitantes; É o núcleo de uma das regiões mais populosas do Paraná: Campos Gerais do Paraná que tem uma população de mais de 1 milhão de habitantes (IBGE, 2014) e o maior parque industrial do interior do estado.

De acordo com a Prefeitura de Ponta Grossa (2014), o serviço de transporte coletivo é oferecido pela VCG – Viação dos Campos Gerais. Desde 2003, por um processo licitatório, a concessionária opera com uma frota de 197 ônibus, distribuídos em 89 linhas. Diariamente são transportadas 90 mil pessoas.

É possível obter a relação de horários de cada linha através do site da Prefeitura Municipal de Ponta Grossa e do site da Viação dos Campos Gerais. No primeiro site¹⁵, todas as linhas de ônibus são listadas em uma tabela e contém um hyperlink para abrir o seu conteúdo em documento .pdf; no segundo site¹⁶ é possível realizar a pesquisa de horários da linha escolhendo uma das regiões disponíveis no formulário de pesquisa – do canto superior direito – da página.

¹⁵ <http://esic.pontagrossa.pr.gov.br/smp/sobre-o-transporte-coletivo>

¹⁶ <http://www.vcg.com.br>



[Início](#) >


Sobre o Transporte Coletivo

DECRETO 603/2002 - por webmaster em 14/05/2013 - 16:47:20
PLANILHA DE CUSTOS (JULHO/2013) - por webmaster em 05/07/2013 - 17:47:56
PLANILHA DE CUSTOS (MAIO/2013) - por webmaster em 14/05/2013 - 16:47:20
RELACÃO DE LINHAS - por webmaster em 14/05/2013 - 16:47:20
LEI Nº 7018/2002 - por webmaster em 14/05/2013 - 16:47:20
DECRETO Nº 6281/2012 - por webmaster em 14/05/2013 - 16:47:20
DECRETO Nº 1000/2006 - por webmaster em 14/05/2013 - 16:47:20
ROTA - MADRUGUEIRO 1.jpg - por webmaster em 14/05/2013 - 16:47:20
ROTA - MADRUGUEIRO 2.jpg - por webmaster em 14/05/2013 - 16:47:20
ROTA - MADRUGUEIRO 3.jpg - por webmaster em 14/05/2013 - 16:47:20
ROTA - MADRUGUEIRO 4.jpg - por webmaster em 14/05/2013 - 16:47:20
ROTA - TC - 31 DE MARCO.jpg - por webmaster em 14/05/2013 - 16:48:24
ROTA - TC - BARAUNA.jpg - por webmaster em 14/05/2013 - 16:48:24

Figura 30 Tabela de linhas de ônibus da cidade de Ponta Grossa - PR

Escolha um dia da semana **D. Úteis** Sáb. Dom. e Feriado

Escolha uma Linha

Linha:  Versão para Impressão

Hora	Ponto	Hora	Ponto
06:34	CAMPUS	06:45	CAMPUS
06:54	CAMPUS	07:04	CAMPUS
07:14	CAMPUS	07:24	CAMPUS
07:34	CAMPUS	07:44	CAMPUS
07:54	CAMPUS	08:04	CAMPUS
08:14	CAMPUS	08:24	CAMPUS
08:34	CAMPUS	08:54	CAMPUS
09:14	CAMPUS	09:34	CAMPUS
09:54	CAMPUS	10:14	CAMPUS
10:34	CAMPUS	10:54	CAMPUS
11:14	CAMPUS	11:24	CAMPUS

Figura 31 Tabela de horários no site VCG

Em ambos os sites é passível de problemas com usabilidade; o usuário sempre deve repetir o mesmo trajeto para buscar o horário. Todo o conteúdo é estático e não reaproveitado, tornando repetitivo este processo de interação.

APÊNDICE E - O Transporte Público de Cascavel

O Município de Cascavel possui aproximadamente 310 mil habitantes; É gerenciado e fiscalizado pela CETTRANS - Companhia de Engenharia de Transporte e Trânsito, que possui uma estrutura de frota de 136 ônibus, de duas empresas, que operam 52 linhas, 3 terminais de transbordo e mais de 1.000 pontos de ônibus.

Através do site¹⁷ é possível visualizar uma tabela com a relação de todas as linhas que operam na cidade. Ao clicar no ícone do relógio é possível visualizar os horários de cada linha; ao clicar no ícone do mapa é possível visualizar o itinerário da respectiva linha.

Todo o conteúdo da tabela é estático e disponibilizado ao usuário através de arquivo *.pdf*, que por sua vez não oferece uma experiência positiva para o usuário se ele estiver acessando o conteúdo através de um celular, por exemplo.

¹⁷ <http://www.cettrans.com.br/pagina.php?id=22>

Cettrans
COMPANHIA DE ENGENHARIA DE TRANSPORTE E TRÂNSITO

PRINCIPAL HISTÓRIA LEGISLAÇÃO FALE CONOSCO

Cascavel, 24 de agosto de 2015 Notícias Vídeos Horários de atendimento

A Cettrans

- » Institucional
- » Estatísticas
- » Licitações
- » Projetos
- » Transparência Pública

Transporte

- » Transporte Coletivo
- » Táxi
- » Transporte Escolar
- » Terminal Rodoviário
- » Aeroporto

Trânsito

Municipalização

TABELA DE HORÁRIOS E ITINERÁRIOS

Confira os horários e itinerários dos ônibus do Transporte Coletivo Urbano de Cascavel.

» Para acessar a tabela de horários basta clicar no ícone .
E para visualizar o itinerário da linha clique no ícone .

-   » Aeroporto - Terminal Oeste
-   » Ferropar - Terminal Leste
-   » Frigorífico Globoaves - Terminal Leste
-   » Fundetec - Terminal Leste
-   » PIC - Penitenciária Industrial de Cascavel
-   » Rodovia das Cataratas
-   » Univel

Figura 32 Listagem de linhas de ônibus do site CETTRANS

APÊNDICE F - O Transporte Público de Foz Do Iguaçu

A cidade de Foz do Iguaçu possui um serviço de Transporte Público Coletivo, através de ônibus regulares que atendem todas as regiões da cidade, e que podem ser utilizados para chegar aos principais pontos turísticos, com um valor bastante econômico. Existem também, linhas de transporte público internacionais, que vão até Ciudad del Este (Paraguai), e Puerto Iguazú (Argentina), e um Terminal de Transporte Urbano (TTU), no centro da cidade.

De acordo com a Prefeitura do município, a FOZTRANS – Instituto de Transportes e Trânsito de Foz do Iguaçu é a companhia responsável por realizar os serviços de ônibus coletivo. No site da companhia é possível visualizar uma tabela com a relação de todas as linhas que fazem itinerários na cidade.

🏠 ▶ Órgãos do Governo ▶ FOZTRANS ▶ Transporte Coletivo ▶ Tabelas de Horários

Tabelas de Horários e Demais Informações



**INTEGRAÇÃO
TEMPORAL**



**BÔNUS
NA TARIFA**

 Linhas
utilizadas para
alguns dos

Nº LINHA - NOME : ATUALIZAÇÃO - Links HORÁRIOS - ITINERÁRIOS - AVISOS

- 010 - Cidade Nova: 18/06/13 - [Horários](#) - [Itinerário](#)
- 025 - Madrugadão Conj."C" - Porto Meira: 11/10/12 - [Horários](#) - [Itinerário](#)
- 035 - Jardim Aporã: 04/10/13 - [Horários](#) - [Itinerário](#) - [Aviso](#)
- 040 - Conj. "C" Rodoviária: 04/10/13 - [Horários](#) - [Itinerário](#) - [Aviso](#)
- 050 - Vila "A" via PR: 21/08/13 - [Horários](#) - [Itinerário](#) - [Aviso](#) - [Aviso 2](#)
- 055 - Vila "A" via JK: 11/10/13 - [Horários](#) - [Itinerário](#)
- 060 - Jardim Paraná: 04/10/13 - [Horários](#) - [Itinerário](#)
- 065 - Universitária: 18/10/13 - [Horários](#) - [Itinerário](#) - [Aviso](#)
- 070 - Cidade Nova / INSS: 01/11/13 - [Horários](#) - [Itinerário](#) - [Aviso](#)
- 075 - Itaipu Binacional: 11/05/12 - [Horários](#) - [Itinerário](#)
- 100 - Alimentador Remanso G.: 06/02/12 - [Horários](#) - [Itinerário](#)
- 101/102 - Vila "C": 04/10/13 - [Hor.](#) [SEG](#) [Hor.](#) [SAB/DOM](#) - [Itiner.101](#) [Itiner.102](#)
- 103 - Porto Belo - Jd. Flores: 17/10/13 - [Horários](#) - [Itinerário](#) - [Aviso](#)
- 104 - UNILA/ Vila "C": 05/03/13 - [Horários](#) - [Itinerário](#)
- 105 - Av. Morenitas - Carimã: 01/10/13 - [Horários](#) - [Itinerário](#)
- 107 - Jardim Itaipu / TTU: 01/10/12 - [Horários](#) - [Itinerário](#)
- 115 - Jd. das Flores / Rodoviária: 17/10/13 - [Horários](#) - [Itinerário](#) - [Aviso](#)
- 120 - Parque Nacional do Iguaçu: 17/10/13 - [Horários](#) - [Itinerário](#)
- 200 - Gleba Guarani: 07/03/13 - [Horários](#) - [Itinerário](#)
- 205 - Santa Rita: 01/11/13 - [Horários](#) - [Itinerário](#)

Figura 33 Listagem de linhas de ônibus do site FOZTRANS

Ao clicar no botão Horários, o usuário é direcionado para uma página com a tabela de horários separada por dias úteis e fim de semana e as paradas do itinerário. A tabela não possui tamanho favorável de uma boa visualização, causando dificuldades para leitura das informações.

FOZTRANS – Instituto de Transportes e Trânsito de Foz do Iguçu					
120 - PARQUE NACIONAL					
IDA		SEGUNDA A SABADO		DOMINGO	
T.T.U.(ala 02).		CRV>>TTU	TTU>>CRV	CRV>>TTU	TTU>>CRV
R. Tarobá			05:25		05:20
R. Rui Barbosa			05:45		05:45
Av. Juscelino Kubitscheck			06:04 ESCOLA	05:30	06:15
Av. Jorge Schimmelpfeng			06:26 ESCOLA	PNE 06:00	06:45
Av. das Cataratas		05:45	06:48 ESCOLA	PNE 06:30	07:15
		06:04	07:05 EXTRA-ESCOLA	07:00	07:45
BR 469			07:15	PNE 07:30	08:15
Aeroporto		PNE 06:26	07:32	PNE 08:00	08:45
BR 469		PNE 06:48	07:54		09:15
		07:10	08:16	PNE 09:00	09:45
Parque Nacional (até o Portão)		07:32	08:38	PNE 09:30	10:15
		PNE 07:54	09:00		10:45
VOLTA		PNE 08:16	09:22	PNE 10:30	11:15
Parque Nacional		08:38	09:44	PNE 11:00	11:45
BR 469		09:00	10:06		12:15
Aeroporto		PNE 09:22	10:28	PNE 11:30	12:45
BR 469		PNE 09:44	10:50	PNE 12:00	12:45
Av. das Cataratas		10:06	11:12	PNE 12:30	13:15
Av. Jorge Schimmelpfeng		10:28	11:34		13:45
Av. Juscelino Kubitscheck		10:50	11:56	PNE 13:30	14:15
Av. Republica Argentina		PNE 11:12	12:18	PNE 14:00	14:45
R. Taroba		PNE 11:34	12:40		15:15
T.T.U.(ala 01, ala 02)		11:56		PNE 14:30	15:15
				14:45 EXTRA	15:30

Figura 34 Tabela de horários da linha 120 Parque Nacional de Foz do Iguçu - PR

APÊNDICE G - Script SQL para criar o Banco de Dados

```
CREATE SCHEMA IF NOT EXISTS `db_api_transporte_coletivo` DEFAULT CHARACTER  
SET utf8 COLLATE utf8_general_ci ;  
USE `db_api_transporte_coletivo` ;
```

APÊNDICE H - Script SQL para criar a tabela Cidades

```
DROP TABLE IF EXISTS `db_api_transporte_coletivo`.`cidades` ;

CREATE TABLE IF NOT EXISTS `db_api_transporte_coletivo`.`cidades` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(60) NOT NULL,
  `estado` VARCHAR(2) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;
```


APÊNDICE I - Script SQL para criar a tabela Empresas

```
DROP TABLE IF EXISTS `db_api_transporte_coletivo`.`empresas` ;

CREATE TABLE IF NOT EXISTS `db_api_transporte_coletivo`.`empresas` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `id_cidade` INT NOT NULL,
  `nome` VARCHAR(60) NOT NULL,
  `data_criacao` VARCHAR(20) NULL,
  `data_atualizacao` VARCHAR(20) NULL,
  PRIMARY KEY (`id`, `id_cidade`),
  INDEX `fk_empresas_cidades1_idx` (`id_cidade` ASC),
  CONSTRAINT `fk_empresas_cidades1`
    FOREIGN KEY (`id_cidade`)
    REFERENCES `db_api_transporte_coletivo`.`cidades` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

APÊNDICE J - Script SQL para criar a tabela Circulares

```
DROP TABLE IF EXISTS `db_api_transporte_coletivo`.`circulares` ;

CREATE TABLE IF NOT EXISTS `db_api_transporte_coletivo`.`circulares` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `id_empresa` INT NOT NULL,
  `linha` VARCHAR(45) NULL,
  `nome` VARCHAR(45) NOT NULL,
  `data_criacao` VARCHAR(20) NULL,
  `data_atualizacao` VARCHAR(20) NULL,
  PRIMARY KEY (`id`, `id_empresa`),
  INDEX `fk_circulares_empresas1_idx` (`id_empresa` ASC),
  CONSTRAINT `fk_circulares_empresas1`
    FOREIGN KEY (`id_empresa`)
      REFERENCES `db_api_transporte_coletivo`.`empresas` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

APÊNDICE K - Script SQL para criar a tabela Vias

```
DROP TABLE IF EXISTS `db_api_transporte_coletivo`.`vias` ;

CREATE TABLE IF NOT EXISTS `db_api_transporte_coletivo`.`vias` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `id_onibus` INT NOT NULL,
  `nome` VARCHAR(60) NOT NULL,
  PRIMARY KEY (`id`, `id_onibus`),
  INDEX `fk_vias_circulares1_idx` (`id_onibus` ASC),
  CONSTRAINT `fk_vias_circulares1`
    FOREIGN KEY (`id_onibus`)
    REFERENCES `db_api_transporte_coletivo`.`circulares` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

APÊNDICE L - Script SQL para criar a tabela Horários

```
DROP TABLE IF EXISTS `db_api_transporte_coletivo`.`horarios` ;

CREATE TABLE IF NOT EXISTS `db_api_transporte_coletivo`.`horarios` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `id_via` INT NOT NULL,
  `ponto_inicial` VARCHAR(90) NOT NULL,
  `dias_uteis` TINYINT(1) NULL,
  `sabado` TINYINT(1) NULL,
  `domingo` TINYINT(1) NULL,
  `feriado` TINYINT(1) NULL,
  `hora` VARCHAR(5) NULL,
  PRIMARY KEY (`id`, `id_via`),
  INDEX `fk_horarios_vias1_idx` (`id_via` ASC),
  CONSTRAINT `fk_horarios_vias1`
    FOREIGN KEY (`id_via`)
    REFERENCES `db_api_transporte_coletivo`.`vias` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

APÊNDICE M - Script SQL para criar a tabela Itinerários

```
DROP TABLE IF EXISTS `db_api_transporte_coletivo`.`itinerarios` ;

CREATE TABLE IF NOT EXISTS `db_api_transporte_coletivo`.`itinerarios` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `id_via` INT NOT NULL,
  `ordem` INT NOT NULL,
  `nome` VARCHAR(150) NULL,
  `latitude` VARCHAR(10) NULL,
  `longitude` VARCHAR(10) NULL,
  PRIMARY KEY (`id`, `id_via`),
  INDEX `fk_itinerarios_vias1_idx` (`id_via` ASC),
  CONSTRAINT `fk_itinerarios_vias1`
    FOREIGN KEY (`id_via`)
    REFERENCES `db_api_transporte_coletivo`.`vias` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

APÊNDICE N - Código-fonte da API /cidades

```

var express = require('express');
var router = express.Router();

router.route('/')
  // Obter todas as cidades
  .get(function(req, res) {
    req.getConnection(function(err, connection) {
      var query = connection.query('SELECT * FROM cidades', function(err,
rows) {
        if (err) {
          res.status(400).json({
            status: false,
            message: 'Erro desconhecido. Por favor tente novamente.'
          });
        } else {
          if (rows.length > 0) {
            res.json({
              status: true,
              cidades: rows
            });
          } else {
            res.status(200).json({
              status: false,
              message: 'Não existem cidades cadastradas.'
            });
          }
        }
      });
    });
  });

// Adiciona uma nova cidade
.post(function(req, res) {
  var cidade = {};

  if (req.body.nome && req.body.estado) {
    cidade.nome = req.body.nome;
    cidade.estado = req.body.estado;

    req.getConnection(function(err, connection) {
      connection.query('INSERT INTO cidades SET ?', cidade,
function(err, result) {
        if (err) {
          res.status(400).json({
            status: false,
            message: 'Erro desconhecido. Por favor tente
novamente.'
          });
        } else {
          res.status(201).json({
            status: true,
            id: result.insertId,
            message: 'Cidade cadastrada com sucesso.'
          });
        }
      });
    });
  });
});

```

```

    });
  } else {
    res.status(200).json({
      status: false,
      message: 'Os campos "nome", e "estado" são obrigatórios.'
    });
  }
});

router.route('/:id')
  // Obter informações de uma cidade
  .get(function(req, res) {
    var id = req.params.id;
    req.getConnection(function(err, connection) {
      var query = connection.query('SELECT * FROM cidades WHERE id = ?',
id, function(err, rows) {
        if (err) {
          res.status(400).json({
            status: false,
            message: 'Erro desconhecido. Por favor tente novamente.'
          });
        } else {
          if (rows.length > 0) {
            res.json({
              status: true,
              cidade: rows
            });
          } else {
            res.status(200).json({
              status: false,
              message: 'Não existe cidade com esse id.'
            });
          }
        }
      });
    });
  });
});

// Editar um registro de uma cidade
.put(function(req, res) {
  var id = req.params.id;
  var cidade = {};
  if (req.body.nome)
    cidade.nome = req.body.nome;
  if (req.body.estado)
    cidade.estado = req.body.estado;
  req.getConnection(function(err, connection) {
    connection.query('UPDATE cidades SET ? WHERE id = ?', [cidade, id],
function(err, result) {
    if (err) {
      res.status(400).json({
        status: false,
        message: 'Erro desconhecido. Por favor tente novamente.'
      });
    } else {
      if (result.affectedRows > 0) {
        res.status(200).json({
          status: true,
          id: req.params.id,

```

```

        message: 'Cidade editada com sucesso.'
    });
    } else {
        res.status(200).json({
            status: false,
            message: 'Não existe cidade com esse id.'
        });
    }
    });
});
})

// Deletar um registro de uma cidade
.delete(function(req, res) {
    var id = req.params.id;
    req.getConnection(function(err, connection) {
        connection.query('DELETE from cidades WHERE id = ?', id,
function(err, result) {
            if (err) {
                res.status(400).json({
                    status: false,
                    message: 'Erro desconhecido. Por favor tente novamente.'
                });
            } else {
                if (result.affectedRows > 0) {
                    res.status(200).json({
                        status: true,
                        message: 'Cidade removida com sucesso.'
                    });
                } else {
                    res.status(200).json({
                        status: false,
                        message: 'Não existe cidade com esse id.'
                    });
                }
            }
        });
    });
});
});

module.exports = router;

```


APÊNDICE O - Código-fonte da API /empresas

```

var express = require('express');
var moment = require('moment');
var router = express.Router();
var timestamp = moment(Date.now()).format('DD/MM/YYYY HH:mm:ss');

router.route('/')
  // Obter todas as empresas
  .get(function(req, res) {
    req.getConnection(function(err, connection) {
      var query = connection.query('SELECT * FROM empresas', function(err,
rows) {
        if (err) {
          res.status(400).json({
            status: false,
            message: 'Erro desconhecido. Por favor tente novamente.'
          });
        } else {
          if (rows.length > 0) {
            res.json({
              status: true,
              empresas: rows
            });
          } else {
            res.status(404).json({
              status: false,
              message: 'Não existem empresas cadastradas.'
            });
          }
        }
      });
    });
  });
});

// Adiciona uma nova empresa
.post(function(req, res) {

  var empresa = {
    data_criacao: timestamp,
    data_atualizacao: timestamp
  };

  if (req.body.id_cidade && req.body.nome) {
    empresa.id_cidade = req.body.id_cidade;
    empresa.nome = req.body.nome;

    req.getConnection(function(err, connection) {
      connection.query('INSERT INTO empresas SET ?', empresa,
function(err, result) {
        if (err) {
          res.status(200).json({
            status: false,
            message: 'Não existe cidade com esse id.'
          });
        } else {
          res.status(201).json({
            status: true,

```

```

        id: result.insertId,
        message: 'Empresa cadastrada com sucesso.'
    });
    }
  });
});

} else {
  res.status(200).json({
    status: false,
    message: 'Os campos "id_cidade", e "nome" são obrigatórios.'
  });
}
});

router.route('/:id')
  // Obter informações de uma empresa
  .get(function(req, res) {
    var id = req.params.id;
    req.getConnection(function(err, connection) {
      var query = connection.query('SELECT * FROM empresas WHERE id = ?',
id, function(err, rows) {
        if (err) {
          res.status(400).json({
            status: false,
            message: 'Erro desconhecido. Por favor tente novamente.'
          });
        } else {
          if (rows.length > 0) {
            res.json({
              status: true,
              empresa: rows
            });
          } else {
            res.status(200).json({
              status: false,
              message: 'Não existe empresa com esse id.'
            });
          }
        }
      });
    });
  });
});

// Editar um registro de uma empresa
.put(function(req, res) {
  var id = req.params.id;
  var empresa = {
    data_atualizacao: timestamp
  };

  if (req.body.id_cidade && req.body.nome) {
    empresa.id_cidade = req.body.id_cidade;
    empresa.nome = req.body.nome;
  }

  req.getConnection(function(err, connection) {
    connection.query('UPDATE empresas SET ? WHERE id = ?', [empresa,
id], function(err, result) {
      if (err) {

```

```

        res.status(400).json({
            status: false,
            message: 'Erro desconhecido. Por favor tente
novamente.'
        });
    } else {
        if (result.affectedRows > 0) {
            res.status(200).json({
                status: true,
                id: req.params.id,
                message: 'Empresa editada com sucesso.',
                data_atualizacao: timestamp
            });
        } else {
            res.status(200).json({
                status: false,
                id: req.params.id,
                message: 'Não existe empresa com esse id.'
            });
        }
    }
});
});
} else {
    res.status(200).json({
        status: false,
        message: 'Os campos "id_cidade" e "nome" são obrigatórios.'
    });
}
})

// Deletar um registro de uma empresa
.delete(function(req, res) {
    var id = req.params.id;
    req.getConnection(function(err, connection) {
        connection.query('DELETE from empresas WHERE id = ?', id,
function(err, result) {
            if (err) {
                res.status(400).json({
                    status: false,
                    message: 'Erro desconhecido. Por favor tente novamente.'
                });
            } else {
                if (result.affectedRows > 0) {
                    res.status(200).json({
                        status: true,
                        message: 'Empresa removida com sucesso.'
                    });
                } else {
                    res.status(200).json({
                        status: false,
                        message: 'Não existe empresa com esse id.'
                    });
                }
            }
        });
    });
});
});

```

```
module.exports = router;
```

APÊNDICE P - Código-fonte da API /onibus

```

var express = require('express');
var moment = require('moment');
var router = express.Router();
var timestamp = moment(Date.now()).format('DD/MM/YYYY HH:mm:ss');

router.route('/')
  // Obter todos os onibus
  .get(function(req, res) {
    req.getConnection(function(err, connection) {
      var query = connection.query('SELECT * FROM circulares',
function(err, rows) {
      if (err) {
        res.status(400).json({
          status: false,
          message: 'Erro desconhecido. Por favor tente novamente.'
        });
      } else {
        if (rows.length > 0) {
          res.json({
            status: true,
            onibus: rows
          });
        } else {
          res.status(200).json({
            status: false,
            message: 'Não existem ônibus cadastrados.'
          });
        }
      }
    });
  });
});

// Adiciona um novo onibus
.post(function(req, res) {

  var onibus = {
    linha: req.body.linha,
    data_criacao: timestamp,
    data_atualizacao: timestamp
  };

  if (req.body.id_empresa && req.body.nome) {
    onibus.id_empresa = req.body.id_empresa;
    onibus.nome = req.body.nome;
  }

  req.getConnection(function(err, connection) {
    connection.query('INSERT INTO circulares SET ?', onibus,
function(err, result) {
      if (err) {
        res.status(200).json({
          status: false,
          message: 'Não existe empresa com esse id.'
        });
      } else {
        res.status(201).json({

```

```

        status: true,
        id: result.insertId,
        message: 'Ônibus cadastrado com sucesso.'
    });
    }
  });
});

} else {
  res.status(200).json({
    status: false,
    message: 'Os campos "id_empresa" e "nome" são obrigatórios.'
  });
}
});

router.route('/:id')
  // Obter informações de um ônibus
  .get(function(req, res) {
    var id = req.params.id;
    req.getConnection(function(err, connection) {
      var query = connection.query('SELECT * FROM circulares where id = ?',
id, function(err, rows) {
        if (err) {
          res.status(200).json({
            status: false,
            message: 'Erro desconhecido. Por favor tente novamente.'
          });
        } else {
          if (rows.length > 0) {
            res.json({
              status: true,
              onibus: rows
            });
          } else {
            res.status(200).json({
              status: false,
              message: 'Não existe ônibus com esse id.'
            });
          }
        }
      });
    });
  });
});

// Editar um registro de um ônibus
.put(function(req, res) {
  var id = req.params.id;
  var onibus = {
    data_atualizacao: timestamp
  };

  if (req.body.linha) {
    onibus.linha = req.body.linha;
  }

  if (req.body.id_empresa && req.body.nome) {
    onibus.id_empresa = req.body.id_empresa;
    onibus.nome = req.body.nome;
  }
});

```

```

    req.getConnection(function(err, connection) {
        connection.query('UPDATE circulares SET ? WHERE id = ?', [onibus,
id], function(err, result) {
            if (err) {
                res.status(200).json({
                    status: false,
                    message: 'Erro desconhecido. Por favor tente
novamente.'
                });
            } else {
                if (result.affectedRows > 0) {
                    res.status(200).json({
                        status: true,
                        id: req.params.id,
                        message: 'Ônibus editado com sucesso.'
                    });
                } else {
                    res.status(200).json({
                        status: false,
                        message: 'Não existe ônibus com esse id.'
                    });
                }
            }
        });
    });
} else {
    res.status(200).json({
        status: false,
        message: 'Os campos "id_empresa" e "nome" são obrigatórios.'
    });
}
})

// Deletar um registro de um onibus
.delete(function(req, res) {
    var id = req.params.id;
    req.getConnection(function(err, connection) {
        connection.query('DELETE from circulares WHERE id = ?', id,
function(err, result) {
            if (err) {
                res.status(200).json({
                    status: false,
                    message: 'Erro desconhecido. Por favor tente novamente.'
                });
            } else {
                if (result.affectedRows > 0) {
                    res.status(200).json({
                        status: true,
                        message: 'Ônibus removido com sucesso.'
                    });
                } else {
                    res.status(200).json({
                        status: false,
                        message: 'Não existe ônibus com esse id.'
                    });
                }
            }
        });
    });
}
}

```

```
    });  
  });  
});  
  
module.exports = router;
```


APÊNDICE Q - Código-fonte da API /vias

```

var express = require('express');
var router = express.Router();

router.route('/')
  // Obter todas as vias de onibus
  .get(function(req, res) {
    req.getConnection(function(err, connection) {
      var query = connection.query('SELECT * FROM vias', function(err,
rows) {
        if (err) {
          res.status(400).json({
            status: false,
            message: 'Erro desconhecido. Por favor tente novamente.'
          });
        } else {
          if (rows.length > 0) {
            res.json({
              status: true,
              vias: rows
            });
          } else {
            res.status(200).json({
              status: false,
              message: 'Não existem vias cadastradas.'
            });
          }
        }
      });
    });
  });

// Adiciona uma nova via de onibus
.post(function(req, res) {
  var via = {};
  if (req.body.id_onibus && req.body.nome) {
    via.id_onibus = req.body.id_onibus;
    via.nome = req.body.nome;

    req.getConnection(function(err, connection) {
      var query = connection.query('INSERT INTO vias SET ?', via,
function(err, result) {
        if (err) {
          res.status(200).json({
            status: false,
            message: 'Não existe ônibus com esse id.'
          });
        } else {
          res.status(201).json({
            status: true,
            id: result.insertId,
            message: 'Via cadastrada com sucesso.'
          });
        }
      });
    });
  });
});

```

```

    } else {
        res.status(200).json({
            status: false,
            message: 'Os campos "id_onibus" e "nome" são obrigatórios.'
        });
    }
});

router.route('/onibus/:id_onibus')
    // Obter informações de todas vias de ônibus
    .get(function(req, res) {
        var id_onibus = req.params.id_onibus;
        req.getConnection(function(err, connection) {
            var query = connection.query('SELECT * FROM vias WHERE id_onibus =
?', id_onibus, function(err, rows) {
                if (err) {
                    res.status(400).json({
                        status: false,
                        message: 'Erro desconhecido. Por favor tente novamente.'
                    });
                } else {
                    if (rows.length > 0) {
                        res.json({
                            status: true,
                            vias: rows
                        });
                    } else {
                        res.status(200).json({
                            status: false,
                            message: 'Não existe(m) via(s) para o ônibus'
                        });
                    }
                }
            });
        });
    });

router.route('/:id')
    // Obter informações de uma via de onibus
    .get(function(req, res) {
        var id = req.params.id;
        req.getConnection(function(err, connection) {
            var query = connection.query('SELECT * FROM vias WHERE id = ?', id,
function(err, rows) {
                if (err) {
                    res.status(400).json({
                        status: false,
                        message: 'Erro desconhecido. Por favor tente novamente.'
                    });
                } else {
                    if (rows.length > 0) {
                        res.json({
                            status: true,
                            vias: rows
                        });
                    } else {
                        res.status(200).json({
                            status: false,
                            message: 'Não existe via com esse id.'
                        });
                    }
                }
            });
        });
    });

```

```

    });
  });
});
})

// Editar um registro de uma via de onibus
.put(function(req, res) {
  var id = req.params.id;
  var via = {};

  if (req.body.id_onibus && req.body.nome) {
    via.id_onibus = req.body.id_onibus;
    via.nome = req.body.nome;

    req.getConnection(function(err, connection) {
      var query = connection.query('UPDATE vias SET ? WHERE id = ?',
[via, id], function(err, result) {
        if (err) {
          res.status(200).json({
            status: false,
            message: 'Não existe ônibus com este id.'
          });
        } else {
          if (result.affectedRows > 0) {
            res.status(200).json({
              status: true,
              message: 'Via editada com sucesso.'
            });
          }
        }
      });
    });
  } else {
    res.status(200).json({
      status: false,
      message: 'Os campos "id_onibus" e "nome" são obrigatórios.'
    });
  }
})

// Deletar um registro de uma via de onibus
.delete(function(req, res) {
  var id = req.params.id;
  req.getConnection(function(err, connection) {
    connection.query('DELETE from vias WHERE id = ?', id, function(err,
result) {
      if (err) {
        res.status(400).json({
          status: false,
          message: 'Erro desconhecido. Por favor tente novamente.'
        });
      } else {
        if (result.affectedRows > 0) {
          res.status(200).json({
            status: true,
            message: 'Via removida com sucesso.'
          });
        }
      }
    });
  });
});

```

```
    });  
  } else {  
    res.status(200).json({  
      status: false,  
      message: 'Não existe via com esse id.'  
    });  
  }  
});  
});  
});  
module.exports = router;
```

APÊNDICE R - Código-fonte da API /itinerarios

```

var express = require('express');
var router = express.Router();

router.route('/via/:id_via')
  // Obter todas paradas do itinerário de um ônibus
  .get(function(req, res) {
    var id_via = req.params.id_via;
    req.getConnection(function(err, connection) {
      var query = connection.query('SELECT * FROM itinerarios WHERE id_via
= ?', id_via, function(err, rows) {
        if (err) {
          res.status(400).json({
            status: false,
            message: 'Erro desconhecido. Por favor tente novamente.'
          });
        } else {
          if (rows.length > 0) {
            res.json({
              status: true,
              paradas: rows
            });
          } else {
            res.status(200).json({
              status: false,
              message: 'Não existe itinerário para esta via de
ônibus.'
            });
          }
        }
      });
    });
  });

router.route('/')
  // Adiciona uma nova parada do itinerário do ônibus
  .post(function(req, res) {
    var parada = {
      latitude: req.body.latitude,
      longitude: req.body.longitude
    };

    if (req.body.id_via && req.body.ordem && req.body.nome) {
      parada.id_via = req.body.id_via;
      parada.ordem = req.body.ordem;
      parada.nome = req.body.nome;

      req.getConnection(function(err, connection) {
        connection.query('INSERT INTO itinerarios SET ?', parada,
function(err, result) {
          if (err) {
            res.status(200).json({
              status: false,
              message: 'Não existe via com esse id.'
            });
          } else {

```

```

        res.status(201).json({
            status: true,
            id: result.insertId,
            message: 'Itinerário cadastrado com sucesso.'
        });
    });
} else {
    res.status(200).json({
        status: false,
        message: 'Os campos "id_via", "ordem" e "nome" são obrigatórios.'
    });
}
});

router.route('/:id')
// Editar um registro de parada do itinerário do ônibus
.put(function(req, res) {
    var id = req.params.id;
    var parada = {};

    if (req.body.latitude)
        parada.latitude = req.body.latitude;

    if (req.body.longitude)
        parada.longitude = req.body.longitude;

    if (req.body.id_via && req.body.ordem && req.body.nome) {
        parada.id_via = req.body.id_via;
        parada.ordem = req.body.ordem;
        parada.nome = req.body.nome;

        req.getConnection(function(err, connection) {
            connection.query('UPDATE itinerarios SET ? WHERE id = ?',
[parada, id], function(err, result) {
                if (err) {
                    res.status(200).json({
                        status: false,
                        message: 'Não existe via de ônibus com esse id'
                    });
                } else {
                    if (result.affectedRows > 0) {
                        res.status(200).json({
                            status: true,
                            message: 'Parada do itinerário editada com
sucesso.'
                        });
                    } else {
                        res.status(200).json({
                            status: false,
                            message: 'Não existe parada de ônibus com esse
id'
                        });
                    }
                }
            });
        });
    }
});
});

```

```

    } else {
      res.status(200).json({
        status: false,
        message: 'Os campos "id_via", "ordem" e "nome" são obrigatórios.'
      });
    }
  })

  // Deletar um registro de parada do itinerário do ônibus
  .delete(function(req, res) {
    var id = req.params.id;
    req.getConnection(function(err, connection) {
      connection.query('DELETE from itinerarios WHERE id = ?', id,
function(err, result) {
      if (err) {
        res.status(400).json({
          status: false,
          message: 'Erro desconhecido. Por favor tente novamente.'
        });
      } else {
        if (result.affectedRows > 0) {
          res.status(200).json({
            status: true,
            message: 'Parada de itinerário removida com sucesso.'
          });
        } else {
          res.status(200).json({
            status: false,
            message: 'Não existe parada de itinerário com esse
id.'
          });
        }
      }
    });
  });
});

module.exports = router;

```

APÊNDICE S - Código-fonte da API /horarios

```

var express = require('express');
var router = express.Router();

router.route('/:id_via')
  // Obter informações de horários de uma via
  .get(function(req, res) {
    var id_via = req.params.id_via;
    req.getConnection(function(err, connection) {
      var query = connection.query('SELECT * FROM horarios WHERE id_via = ?'
?, id_via, function(err, rows) {
        if (err) {
          res.status(400).json({
            status: false,
            message: 'Erro desconhecido. Por favor tente novamente.'
});
        } else {
          if (rows.length > 0) {
            res.json({
              status: true,
              horarios: rows
            });
          } else {
            res.status(200).json({
              status: false,
              message: 'Não existe(m) horário(s) cadastrado(s) ou
para o id desta via.'
});
          }
        }
      });
    });
  });

router.route('/:id_via/dias-uteis')
  // Obter informações de horários de uma via em dias úteis
  .get(function(req, res) {
    var id_via = req.params.id_via;
    req.getConnection(function(err, connection) {
      var query = connection.query('SELECT * FROM horarios WHERE id_via = ?
AND dias_uteis = 1', id_via, function(err, rows) {
        if (err) {
          res.status(400).json({
            status: false,
            message: 'Erro desconhecido. Por favor tente novamente.'
});
        } else {
          if (rows.length > 0) {
            res.json({
              status: true,
              horarios: rows
            });
          } else {
            res.status(200).json({
              status: false,
              message: 'Não existe(m) horário(s) cadastrado(s) ou

```



```

    para o id desta via.'
    });
  });
});

router.route('/via/:id_via/sabado')
  // Obter informações de horários de uma via aos sabados
  .get(function(req, res) {
    var id_via = req.params.id_via;
    req.getConnection(function(err, connection) {
      var query = connection.query('SELECT * FROM horarios WHERE id_via = ?
AND sabado = 1', id_via, function(err, rows) {
        if (err) {
          res.status(400).json({
            status: false,
            message: 'Erro desconhecido. Por favor tente novamente.'
          });
        } else {
          if (rows.length > 0) {
            res.json({
              status: true,
              horarios: rows
            });
          } else {
            res.status(200).json({
              status: false,
              message: 'Não existe(m) horário(s) para sábado ou
para o id desta via.'
            });
          }
        }
      });
    });
  });

router.route('/via/:id_via/domingo')
  // Obter informações de horários de uma via aos domingos
  .get(function(req, res) {
    var id_via = req.params.id_via;
    req.getConnection(function(err, connection) {
      var query = connection.query('SELECT * FROM horarios WHERE id_via = ?
AND domingo = 1', id_via, function(err, rows) {
        if (err) {
          res.status(400).json({
            status: false,
            message: 'Erro desconhecido. Por favor tente novamente.'
          });
        } else {
          if (rows.length > 0) {
            res.json({
              status: true,
              horarios: rows
            });
          } else {

```

```

        res.status(200).json({
            status: false,
            message: 'Não existe(m) horário(s) para domingo ou
para o id desta via.'
        });
    }
}
});
});
});

router.route('/via/:id_via/feriados')
// Obter informações de horários de uma via aos feriados
.get(function(req, res) {
    var id_via = req.params.id_via;
    req.getConnection(function(err, connection) {
        var query = connection.query('SELECT * FROM horarios WHERE id_via = ?
AND feriado = 1', id_via, function(err, rows) {
            if (err) {
                res.status(400).json({
                    status: false,
                    message: 'Erro desconhecido. Por favor tente novamente.'
                });
            } else {
                if (rows.length > 0) {
                    res.json({
                        status: true,
                        horarios: rows
                    });
                } else {
                    res.status(200).json({
                        status: false,
                        message: 'Não existe(m) horário(s) para feriado ou
para o id desta via.'
                    });
                }
            }
        });
    });
});

router.route('/')
// Adiciona um novo horario de onibus
.post(function(req, res) {
    var horario = {};
    if (req.body.dias_uteis)
        horario.dias_uteis = req.body.dias_uteis;
    if (req.body.sabado)
        horario.sabado = req.body.sabado;
    if (req.body.domingo)
        horario.domingo = req.body.domingo;
    if (req.body.feriado)
        horario.feriado = req.body.feriado;

    if (req.body.id_via && req.body.ponto_inicial && req.body.hora) {
        horario.id_via = req.body.id_via;
    }
});
});
});

```

```

    horario.ponto_inicial = req.body.ponto_inicial;
    horario.hora = req.body.hora;

    req.getConnection(function(err, connection) {
        connection.query('INSERT INTO horarios SET ?', horario,
function(err, result) {
            if (err) {
                res.status(200).json({
                    status: false,
                    message: 'Não existe cidade com esse id.'
                });
            } else {
                res.status(201).json({
                    status: true,
                    id: result.insertId,
                    message: 'Horário de via cadastrado com sucesso.'
                });
            }
        });
    });

} else {
    res.status(200).json({
        status: false,
        message: 'Os campos "id_via", "ponto_inicial" e "hora" são
obrigatórios.'
    });
}
});

router.route('/:id')
    // Editar um registro de horário de ônibus
    .put(function(req, res) {
        var id = req.params.id;
        var horario = {};
        if (req.body.dias_uteis)
            horario.dias_uteis = req.body.dias_uteis;
        if (req.body.sabado)
            horario.sabado = req.body.sabado;
        if (req.body.domingo)
            horario.domingo = req.body.domingo;
        if (req.body.feriado)
            horario.feriado = req.body.feriado;

        if (req.body.id_via && req.body.ponto_inicial && req.body.hora) {

            horario.id_via = req.body.id_via;
            horario.ponto_inicial = req.body.ponto_inicial;
            horario.hora = req.body.hora;

            req.getConnection(function(err, connection) {
                connection.query('UPDATE horarios SET ? WHERE id = ?', [horario,
id], function(err, result) {
                    if (err) {
                        res.status(400).json({
                            status: false,
                            message: 'Erro desconhecido. Por favor tente
novamente.'
                        });
                    }
                });
            });
        }
    });
}

```

```

    });
  } else {
    if (result.affectedRows > 0) {
      res.status(200).json({
        status: true,
        id: req.params.id,
        message: 'Horário de via editado com sucesso.'
      });
    } else {
      res.status(200).json({
        status: false,
        id: req.params.id,
        message: 'Não existe horário de via com esse id.'
      });
    }
  }
});
});
} else {
  res.status(200).json({
    status: false,
    message: 'Os campos "id_via", "ponto_inicial" e "hora" são obrigatórios.'
  });
}
})

// Deletar um registro de horário de ônibus
.delete(function(req, res) {
  var id = req.params.id;
  req.getConnection(function(err, connection) {
    connection.query('DELETE from horarios WHERE id = ?', id,
function(err, result) {
  if (err) {
    res.status(400).json({
      status: false,
      message: 'Erro desconhecido. Por favor tente novamente.'
    });
  } else {
    if (result.affectedRows > 0) {
      res.status(200).json({
        status: true,
        message: 'Horário de via removido com sucesso.'
      });
    } else {
      res.status(200).json({
        status: false,
        message: 'Não existe horário de via com esse id.'
      });
    }
  }
});
});
});
});

module.exports = router;

```

APÊNDICE T - Código-fonte para buscar horários

```

router.get('/buscar-horario', function(req, res) {
  req.getConnection(function(err, connection) {
    var query = connection.query('SELECT circulares.nome AS linha,
circulares.linha AS codigo, vias.nome AS via, vias.id AS id_via FROM circulares
INNER JOIN vias ON vias.id_onibus = circulares.id WHERE ' + (req.query.tipo ==
'nome' ? ('CONCAT(circulares.nome, " ", vias.nome) LIKE "%' + req.query.texto +
'%"') : 'circulares.linha = "' + req.query.texto + '"'), function(err, rows) {
      if (err) {
        res.status(400).json({
          status: false,
          message: 'Erro desconhecido. Por favor
tente novamente.'
        });
      } else {
        res.render('tabela-linhas', {
          status: true,
          linhas: rows
        });
      }
    });
  });
});
});

```

APÊNDICE U - Código-fonte para mostrar os horários de uma linha de ônibus

```

router.get('/mostrar-horarios/:id_via', function(req, res) {
  var id_via = req.params.id_via;
  req.getConnection(function(err, connection) {
    var query = connection.query('SELECT circulares.linha AS linha,
circulares.nome AS nome, vias.nome AS via, horarios.hora as hora,
horarios.ponto_inicial as local FROM circulares INNER JOIN vias ON vias.id_onibus
= circulares.id INNER JOIN horarios ON horarios.id_via = vias.id WHERE vias.id =
? AND dias_uteis = 1 ORDER BY horarios.hora', id_via, function(err, rows) {
      if (err) {
        res.status(400).json({
          status: false,
          message: 'Erro desconhecido. Por favor
tente novamente.'
        });
      } else {
        var title = rows[0] ? (rows[0].linha + ' ' +
rows[0].nome + ' - ' + rows[0].via) : 'Indisponível';
        res.render('modal-horarios', {
          horarios: rows,
          modalTitle: title
        });
      }
    });
  });
});

```

APÊNDICE V - Código-fonte para mostrar o itinerário de uma linha de ônibus

```

router.get('/mostrar-itinerarios/:id_via', function(req, res) {
  var id_via = req.params.id_via;
  req.getConnection(function(err, connection) {
    var query = connection.query('SELECT circulares.linha,
circulares.nome, vias.nome AS via, itinerarios.ordem, itinerarios.nome AS parada,
itinerarios.latitude, itinerarios.longitude FROM circulares INNER JOIN vias ON
circulares.id = vias.id_onibus INNER JOIN itinerarios ON itinerarios.id_via =
vias.id WHERE vias.id = ? ORDER BY itinerarios.ordem', id_via, function(err,
rows) {
      if (err) {
        res.status(400).json({
          status: false,
          message: 'Erro desconhecido. Por favor
tente novamente.'
        });
      } else {
        var title = rows[0] ? (rows[0].linha + ' - ' +
rows[0].nome + ' - ' + rows[0].via) : 'Indisponível';
        res.render('modal-itinerarios', {
          paradas: rows,
          modalTitle: title
        });
      }
    });
  });
});

```