

UNIVERSIDADE REGIONAL INTEGRADA DO ALTO URUGUAI E DAS MISSÕES
CAMPUS DE ERECHIM
DEPARTAMENTO DE ENGENHARIAS E CIÊNCIA DA COMPUTAÇÃO
CURSO DE CIÊNCIA DA COMPUTAÇÃO

GABRIEL PAULO CAPOAN CECHE

GERENCIAMENTO E ACOMPANHAMENTO DE ROTAS DE ENTREGAS

ERECHIM - RS
2019

GABRIEL PAULO CAPOAN CECHE

GERENCIAMENTO E ACOMPANHAMENTO DE ROTAS DE ENTREGAS

**Trabalho de Conclusão de Curso
apresentado como requisito parcial
à obtenção do grau de Bacharel,
Departamento de Engenharias e Ciência
da Computação da Universidade Regional
Integrada do Alto Uruguai e das Missões
Campus de Erechim.**

Orientador: Guilherme Afonso Madalozzo

ERECHIM - RS

2019

AGRADECIMENTOS

Agradeço primeiramente a Deus.

À minha família, por me apoiarem incondicionalmente e estarem sempre ao meu lado, sendo um alicerce nos momentos difíceis.

Aos professores do curso de Ciência da Computação por todo o conhecimento, manifestação do caráter, confiança e ética da educação no processo de formação profissional. Também agradeço aos amigos e companheiros de trabalhos, de forma especial ao Prof. Dr. Guilherme Afonso Madalozzo por todo auxílio durante o desenvolvimento do trabalho.

A todas as pessoas que direta ou indiretamente contribuíram de alguma maneira para que este trabalho pudesse ser realizado.

RESUMO

O século XXI é marcado pela imersão da sociedade no meio tecnológico. Tarefas simples, como diálogos e pesquisas, e complexas, como operações bancárias, passam mais ou menos intensamente pelo uso da internet. Assim, constata-se que esse mecanismo tornou-se um enorme aliado do ser humano. Todavia, infelizmente, a não inserção nessa nova era ainda é uma realidade para muitos aspectos do cotidiano. Nesse contexto, vem à tona a falta de tecnologia empregada no ramo de transportes, situação que compromete profundamente os trabalhadores da área, em especial os motoboys. Esses profissionais, além de lidarem com o estresse do trânsito, gastam grande parte de seu tempo organizando previamente a rota que será utilizada para o lote de entregas, e, ainda, correm o risco de não fazê-la de modo adequado. Como consequência, muitas vezes percorrem inúmeros quilômetros desnecessários e deixam de realizar outras entregas, que potencialmente se converteriam em um retorno financeiro mais satisfatório ao fim do mês. Tendo em vista tal circunstância, e ciente da importância de cada segundo para esses indivíduos, o presente trabalho propõe uma aplicação web de gerenciamento de pedidos e de seus destinos finais, compartilhados por meio de uma API de troca de dados (disponibilizada pelo sistema de vendas já implantado operando plenamente no estabelecimento), pela utilização de ferramentas como: Laravel e Google Maps, a fim de oferecer uma otimização de tempo para os envolvidos nesse processo, evitando congestionamentos e quaisquer outras situações que representem impedimentos espaciais. Todo esse processo fornece ao entregador a melhor trajetória em tempo real. Logo, em questão de segundos, a rota devidamente planejada pela aplicação, com possível intervenção do usuário administrador para melhoria, será apresentada para o motoboy em seu *smartphone*.

Palavras-chave: *Delivery*. Geolocalização. Google Maps. Aplicação Web.

ABSTRACT

The 21st century is marked by the immersion of society in the technological environment. Simple tasks like dialogues and research, and complex tasks like banking, pass more or less intensively through the use of the internet. Thus, it appears that this mechanism has become a huge ally of the human being. However, unfortunately, the non insertion into this new era is still a reality for many aspects of everyday life. In this context, the lack of technology used in the transportation sector comes up, this situation deeply affects the workers of the area, especially the motoboys. These professionals, in addition to dealing with traffic stress, spend a large part of their time by pre-arranging the route that will be followed for the delivery batch, and then they even risk not doing it properly. As a consequence, they often travel a lot of unnecessary miles and no longer make other deliveries, which could potentially turn into a more satisfying financial return by the end of the month. Given this circumstance, and aware of the importance of every second to these individuals, the present work proposes a web application for managing orders and their final destinations, shared through a data exchange API (made available by the sales system already fully operating in the establishment), through the use of tools such as: Laravel and Google Maps, in order to provide time optimization for those involved in this process, avoiding jams and any other situations that represent spatial impediments. This entire process gives the delivery man the best real-time trajectory. Therefore, in seconds, the route properly planned by the application, with possible intervention from admin user for improvement, will be presented to motoboy on his smartphone.

Keywords: Delivery. Geolocation. Google Maps. Web Application.

LISTA DE ILUSTRAÇÕES

Figura 1 – API - Estrutura	2
Figura 2 – Funcionamento do A-GPS	6
Figura 3 – O problema do multipath	7
Figura 4 – Localização GPS e A-GPS	8
Figura 5 – Exemplo de visualização do Google Maps	9
Figura 6 – Estrutura CGI	13
Figura 7 – Tendências de eficiência de desenvolvimento - Laravel e CodeIgniter	16
Figura 8 – Estrutura Laravel	20
Figura 9 – Delivery Routes - Requisição HTTP	22
Figura 10 – Delivery Routes - Fluxo do status do pedido	23
Figura 11 – Delivery Routes - Login	23
Figura 12 – Delivery Routes - Dashboard	24
Figura 13 – Webpack - Module blunder	25
Figura 14 – Delivery Routes - Lista de motoboys	26
Figura 15 – Delivery Routes - Edição do motoboy	26
Figura 16 – Delivery Routes - Lista de pagamentos	27
Figura 17 – Delivery Routes - Edição do pagamento	27
Figura 18 – Delivery Routes - Lista de entregas	28
Figura 19 – Delivery Routes - Lista de pedidos em aberto	28
Figura 20 – Delivery Routes - API - Coleta de dados	30
Figura 21 – Delivery Routes - API - order	31
Figura 22 – Delivery Routes - Comanda do pedido	32
Figura 23 – Delivery Routes - Fluxo do pedido	32
Figura 24 – Delivery Routes - Despacho da entrega	33
Figura 25 – Delivery Routes - Mapa da entrega	34

LISTA DE QUADROS

Quadro 1 – Propriedades dos métodos de geolocalização	5
Quadro 2 – Comparativo SQL vs. Eloquent ORM	17
Quadro 3 – Resultados do Sumário Executivo	18

LISTA DE TABELAS

Tabela 1	–	Resultado do experimento no tempo de execução	16
Tabela 2	–	Delivery Routes - Status do pedido	22
Tabela 3	–	Delivery Routes - Exemplo de router do back-end	38

LISTA DE ABREVIATURAS E SIGLAS

3D	Computação Gráfica Tridimensional
3G	Terceira Geração de Padrões e Tecnologias de Telefonia Móvel
A-GPS	<i>Assisted Global Positioning System</i>
AJAX	<i>Asynchronous JavaScript and XML</i>
API	<i>Application Programming Interface</i>
BSD	<i>Berkeley Software Distribution</i>
CDMA	<i>Code Division Multiple Access</i>
CGI	<i>Common Gateway Interface</i>
CLI	<i>Command Line Interfaces</i>
CPF	Cadastro de Pessoa Física
CRUD	<i>Create-Read-Update-Delete</i>
FI	<i>Form Interpreter</i>
GB	Gigabyte
GIS	<i>Geographic Information System</i>
GNU	<i>General Public License</i>
GPRS	<i>General Packet Radio Service</i>
GPS	<i>Global Position System</i>
GSM	<i>Global System for Mobile Communications</i>
HTML	<i>HyperText Markup Language</i>
HTTP	<i>Hypertext Transfer Protocol</i>
ID	<i>Integrated Development Environment</i>
IMAP	<i>Internet Message Access Protocol</i>
IP	<i>Internet Protocol</i>
JS	<i>JavaScript</i>

JSON	<i>JavaScript Object Notation</i>
MB	Megabyte
MIT	Instituto de Tecnologia de Massachusetts
MPM	<i>Multi-Processing-Modules</i>
MVC	<i>Model-View-Controller</i>
NNTP	<i>Network News Transfer Protocol</i>
ORM	<i>Object Relational Mapper</i>
OSDB	<i>Open Source Database Benchmark</i>
PDV	Ponto De Vendas
PHP	<i>Personal Home Page</i>
POP3	<i>Post Office Protocol</i>
POSIX	<i>Portable Operating System Interface</i>
PSR	<i>PHP Standard Recommendation</i>
QR	<i>Quick Response</i>
RDBMS	<i>Relational Database Management Systems</i>
REST	<i>Representational State Transfer</i>
SGBD	Sistemas de Gestão de Base de Dados
SNMP	<i>Simple Network Management Protocol</i>
SOAP	<i>Simple Object Access Protocol</i>
SQL	<i>Structured Query Language</i>
SSL	<i>Secure Sockets Layer</i>
TTF	Tempo de Localização Inicial
URL	<i>Uniform Resource Locator</i>
URI	<i>Uniform Resource Identifier</i>
W3C	<i>World Wide Web Consortium</i>
XML	<i>Extensible Markup Language</i>

LISTA DE FRAGMENTOS DE CÓDIGO

Fragmento de Código 1 – Comandos Artisan	17
Fragmento de Código 2 – Delivery Routes - Trigger tr_Status_Orders	21
Fragmento de Código 3 – Delivery Routes - Exemplo de migration	21
Fragmento de Código 4 – Delivery Routes - Função integradora de pedidos	29
Fragmento de Código 5 – Delivery Routes - Função de troca de prioridade dos pedidos	30
Fragmento de Código 6 – Delivery Routes - Route order	31
Fragmento de Código 7 – Delivery Routes - Função de despacho da entrega	33
Fragmento de Código 8 – Delivery Routes - Função de localização do usuário	35
Fragmento de Código 9 – Delivery Routes - Route pedidos da entrega	35
Fragmento de Código 10 – Delivery Routes - Preenchimento dos pontos de parada	35
Fragmento de Código 11 – Delivery Routes - Requisição de renderização do mapa	36
Fragmento de Código 12 – Delivery Routes - Função de cálculo do deslocamento	36
Fragmento de Código 13 – Delivery Routes - Exemplo de um Controller: Motoboy	37

SUMÁRIO

1	INTRODUÇÃO	1
2	TECNOLOGIAS	2
2.1	API	2
2.1.1	REST	3
2.1.2	W3C	3
2.2	Geolocalização	3
2.2.1	Fontes de geolocalização	4
2.2.1.1	Métodos: getCurrentPosition e watchPosition	4
2.3	A-GPS	5
2.3.1	Diferenças entre GPS e A-GPS	7
2.4	Google Maps	8
2.4.1	Google Maps Platform	11
2.4.1.1	Maps	11
2.4.1.2	Routes	11
2.4.1.3	Places	12
3	FERRAMENTAS	13
3.1	CGI (Common Gateway Interface)	13
3.2	PHP	14
3.3	Laravel	15
3.4	Eloquent	16
3.5	Artisan	17
3.6	MySQL	17
4	DESENVOLVIMENTO DA APLICAÇÃO	19
4.1	Softwares utilizados	19
4.2	Módulo de gerenciamento	22
4.3	Coleta de dados	30
4.4	Consulta de dados	31
4.5	Módulo de entrega	32
4.5.1	Tracking de entregas	32
4.6	Controller	37
4.7	Integração da ferramenta	39
5	CONCLUSÃO E TRABALHOS FUTUROS	40

REFERÊNCIAS	42
APÊNDICES	44
APÊNDICE A – DIAGRAMA DE CLASSES	45

1 INTRODUÇÃO

O cotidiano do mundo moderno faz com que algumas aplicações virtuais sejam cruciais para facilitar a organização de um indivíduo, tais como: *i)* lista de tarefas para que o usuário lembre de atividades que não podem ser esquecidas; *ii)* GPS inteligente para guiá-lo; e *iii)* pagar uma conta, para que o usuário não seja pego de surpresa. O princípio básico para o desenvolvimento de uma aplicação é afirmar que um problema existe e precisa ser resolvido (DIAS; AMORIM; JUNIOR, 2014).

A locomoção é uma grande preocupação para quem vive em grandes centros urbanos ou, até mesmo, para usuários que visitam cidades com caminhos desconhecidos. Utilizar o carro nem sempre é a melhor decisão ou uma experiência agradável, afinal, congestionamentos já fazem parte do dia a dia do brasileiro, principalmente quando é necessário percorrer um longo caminho na cidade para entregar ou buscar um objeto. É por esse motivo que a profissão chamada motoboy surgiu, provavelmente em São Paulo, e foi disseminada em todo o país a partir de 1998 (RODRIGUES, 2017).

Diante do cenário urbano atual, faz-se necessário um algoritmo para otimizar as rotas a fim de que os profissionais da área de transportes não fiquem suscetíveis a perder seu tempo devido a percursos mal analisados. Considerando essa situação, o desenvolvimento de um aplicativo móvel conectado à *Google Maps Platform*, que proporcione significativa economia de tempo para os usuários, tornar-se-á uma solução para o problema.

A partir da utilização da *Google Maps Platform*, é possível encontrar o melhor trajeto de um ponto inicial até um destino com dados abrangentes e, também, com dados do trânsito em tempo real. Dessa maneira, experiências personalizadas e ágeis são oferecidas, as quais são capazes de transformar o mundo real em virtual, entregando aos usuários mapas estáticos e dinâmicos.

No segundo Capítulo, são detalhadas as tecnologias utilizadas para o desenvolvimento do presente trabalho, tais como: métodos de geolocalização disponíveis atualmente, Sistema de Posicionamento Global Assistido (A-GPS), serviço de pesquisa e visualização de mapas e imagens de satélite do Planeta Terra, gratuito via internet, por meio do *Google Maps*.

O terceiro Capítulo aborda o estudo das técnicas e das ferramentas que foram necessárias para a construção da aplicação, incluindo a linguagem de programação base, *frameworks*, banco de dados e servidor web.

No quarto Capítulo é descrito o desenvolvimento da aplicação, denominada Delivery Routes. Esse também, discorre acerca de todos os passos realizados no decorrer do trabalho, tais como: módulo de gerenciamento para controle das entregas, coleta dos dados e o posterior acompanhamento das rotas de entregas.

Por fim, no quinto Capítulo são apresentadas as conclusões e as propostas para futuros trabalhos.

2 TECNOLOGIAS

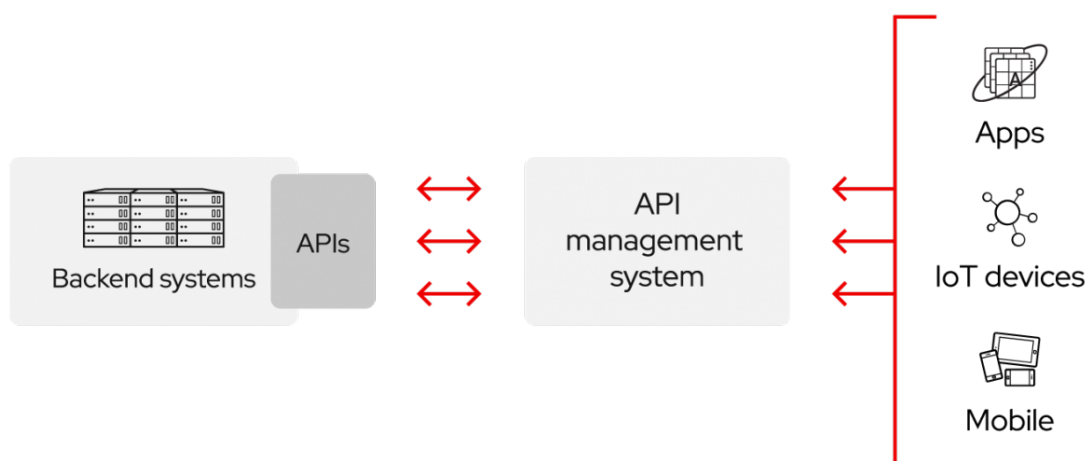
Este Capítulo descreve as tecnologias necessárias para o desenvolvimento deste trabalho.

2.1 API

Destaca-se do pensamento de Pressman (2009) que API é o acrônimo de Interface de Programação de Aplicativos. Na grande maioria das vezes, uma API comunica-se com diversos outros códigos, interligando diversas funções em um aplicativo, como pode ser visto na Figura 1. Essa interface de programação é um conjunto de padrões de programação que permitem a construção de aplicativos e a sua utilização. Portanto, uma API é uma abstração da funcionalidade e da implementação interna de cada componente. Com uma API é possível criar sistemas melhores e minimizar o entendimento de todos os detalhes de um componente. Assim, serve como um “alicerce”, abstraindo a funcionalidade e implementação interna de cada componente.

Paralelamente a isso, segundo Tulach (2008), API é um conjunto de classes, seus métodos e seus campos. Uma boa API é o conjunto de arquivos que o aplicativo lê ou escreve, bem como seu formato. Algumas aplicações sequer fazem uso do disco rígido e, portanto, não precisam se preocupar com essa etapa. Tendo em vista que o primeiro objetivo da engenharia é produzir sistemas funcionais, uma API também precisa ser de fácil utilização, largamente adotada e produtiva.

Figura 1 – API - Estrutura



Fonte: Hat (2019)

2.1.1 REST

Segundo Fielding (2000), REST é uma sigla que significa *Representational State Transfer*, é um modelo arquitetural que define um conjunto de regras simples para implementar serviços web, sem, no entanto, especificar as tecnologias que devem ser utilizadas. Uma forma de implementar REST é utilizar o conceito de *RESTful* que se baseia no protocolo da aplicação HTTP, no padrão de nomenclatura URI e na linguagem XML/JSON. As respostas do REST são cacheáveis, o que possibilita um grande aumento de *performance* para clientes simples. Utilizando esse conceito, os serviços tornam-se disponíveis para diferentes aplicações clientes, ou seja, não há necessidade de mudança de uma aplicação para outra, pois o serviço concentra-se em um servidor próprio.

Depois de seu surgimento, o REST passou a representar uma alternativa mais leve de comunicação entre computadores (cliente e servidor). Para ser considerado REST, o *Web Service* deve ser ou possuir: Interface uniforme, *Stateless*, Cacheável, Cliente-Servidor, Sistema em camadas e Código sob demanda (FIELDING, 2000).

Fielding (2000) destaca os seguintes pontos sobre o REST:

- Fazer uso de diferentes métodos (ou verbos) de comunicação (*GET*, *POST*, *PUT*, *DELETE*, *HEAD*, *OPTIONS*) do HTTP para realizar solicitações para API;
- Utilização de *headers* HTTP (tanto padronizados quanto customizados);
- Definição de arquivos como recursos (cada um com seu próprio endereço)
- Utilização de diferentes tipos de *media type*. Por exemplo, JSON ou XML para representação de dados. Por ser um formato menos verboso e normalmente menor, a transferência de dados com o uso de JSON causa uma carga menor na rede, além de ser facilmente consumido por clientes construídos com qualquer linguagem, em especial HTML5.

2.1.2 W3C

A World Wide Web Consortium é a principal organização de padronização da rede mundial de computadores, desenvolvendo especificações técnicas e orientações através de um processo projetado para maximizar o consenso sobre as recomendações, garantindo qualidades técnicas e editoriais, por meio do desenvolvimento de protocolos comuns e fóruns abertos que promovam a sua evolução e assegurem a sua interoperabilidade, para levar a web ao seu potencial máximo (W3C, 2019).

2.2 Geolocalização

A geolocalização é a capacidade de identificar a localização geográfica, a partir de coordenadas, de determinado objeto ou usuário em monitoramento, com base em dados em tempo real. Esse tipo de sistema tem a capacidade de recuperar as informações de localização por meio do IP, GPS, conexão de *wi-fi*, entre outros. A utilização da geolocalização pode tornar-

se útil para diversos objetivos, a exemplo da obtenção de informações sobre um determinado local ou uma rota de navegação (LABORDE, 2011).

A API de geolocalização consiste em um método para localizar exatamente a posição de um objeto ou usuário, sendo simples de ser implementada e utilizada. Existem duas formas de realizar a captura da geolocalização através da API, compatível com navegadores em *desktop* ou dispositivos móveis. É necessária a autorização do usuário para acessar as suas informações de geolocalização, pois essas podem comprometer a sua privacidade e integridade (LABORDE, 2011).

Compatibilidade com navegadores em *desktop*:

- Firefox 3.5 ou superior;
- Chrome 5.0 ou superior;
- Opera 10,60 ou superior;
- Internet Explorer 9.0 ou superior (LABORDE, 2011).

Compatibilidade com navegadores em dispositivos móveis:

- Android 2.0 ou superior;
- iPhone 3.0 ou superior;
- Opera Mobile 10.1 ou superior;
- Symbian (S60 3rd e 5^a geração);
- Blackberry OS 6;
- Maemo (LABORDE, 2011).

2.2.1 Fontes de geolocalização

Para obter a localização do usuário podem-se utilizar diferentes fontes, as quais têm seu próprio grau de precisão, variação e precisão. Por intermédio do navegador instalado em um *desktop*, o sistema de geolocalização utiliza *wi-fi* (com precisão de 20 metros) ou o IP de conexão, que só se faz necessário ao nível de cidade, uma vez que pode fornecer, algumas vezes, falsas informações. Por sua vez, os dispositivos móveis utilizam a técnica de triangulação do GPS (com precisão de 10m), *wi-fi* e GSM/CDMA (com precisão de 1000 metros) (LABORDE, 2011).

2.2.1.1 Métodos: *getCurrentPosition* e *watchPosition*

Segundo Laborde (2011), *getCurrentPosition* e *watchPosition* são os possíveis métodos para identificar a posição de um objeto ou usuário. Ambos retornam dados de forma imediata e, depois de forma assíncrona, tentam obter a localização atual. Eles carregam o mesmo número de argumentos, dois deles são opcionais: um *callback*, para quando retornar corretamente o dado, outro *callback*, se caso ocorra algum tipo de erro, e um objeto *PositionOptions* que apresenta as seguintes opções:

- *enableHighAccuracy*: atributo de controle que proporciona receber os resultados mais precisos, podem ter respostas mais lentas, ele recebe um valor *boolean*;
- *timeout*: tempo que o dispositivo demora para retornar uma posição. Recebe um valor em milissegundos, com o padrão de 0;
- *maximumAge*: denota o tempo máximo de uma posição em cache que o aplicativo estará disposto a aceitar. Tudo isso sendo realizado em milissegundos com o valor padrão 0, significando que uma tentativa deve ser realizada para obter um novo objeto imediatamente.

Para que seja possível capturar a posição do objeto enquanto ele ou o usuário estiver se movimentando, é necessário utilizar o método *watchPosition*. O Quadro 1 apresenta as propriedades presentes no objeto de retorno da função. Entre esses, apenas *coords.latitude* e *coords.longitude* são garantidos para serem devolvidos (todos os outros podem ser nulos) (LABORDE, 2011).

Quadro 1 – Propriedades dos métodos de geolocalização

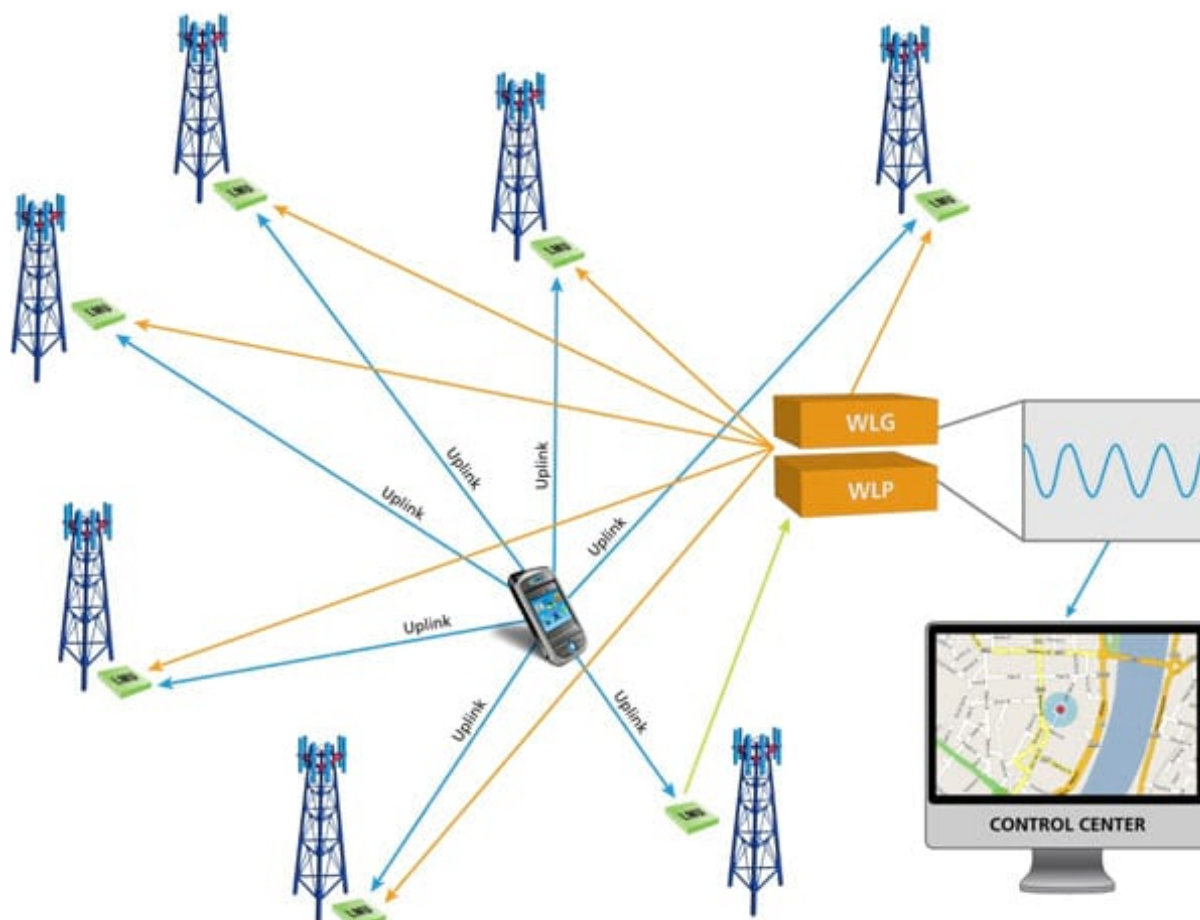
Propriedade	Detalhes
<i>coords.latitude</i>	Número decimal em graus da latitude
<i>coords.longitude</i>	Número decimal em graus da longitude
<i>coords.altitude</i>	Altura em metros da posição abaixo da Elipsoide de referência
<i>coords.accuracy</i>	A precisão em metros do resultado retornado. O valor dessa configuração informa à aplicação o quão útil é o valor retornado para latitude/longitude. Isso pode ajudar a determinar se o resultado retornado é preciso o suficiente para o propósito a que se destina
<i>coords.altitudeAccuracy</i>	A precisão em metros da altitude retornada
<i>coords.heading</i>	Direção que o dispositivo deve tomar, no sentido horário a partir do norte
<i>coords.speed</i>	A velocidade de solo atual do dispositivo em metros por segundo
<i>timestamp</i>	<i>Timestamp</i> de quando a posição foi adquirida

Fonte: Oficina (2018)

2.3 A-GPS

O Sistema de Posicionamento Global Assistido, desenvolvido para localizar satélites com mais rapidez e confiabilidade, dispõe de recursos de rede para localizar e utilizar os satélites em condições de sinal fraco. Logo, é uma versão otimizada do GPS, que recebe informações por meio de uma conexão de dados (GPRS ou 3G), permitindo que o aparelho calcule as coordenadas da sua posição atual ao receber informações satelitais, sob certas condições. Tal aspecto pode ser visto na Figura 2 (OFICINA, 2018).

Figura 2 – Funcionamento do A-GPS

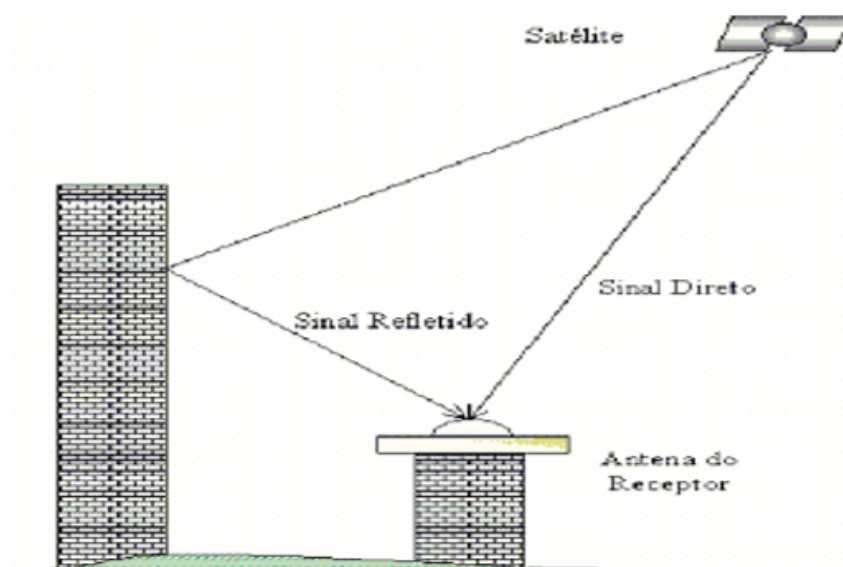


Fonte: Oficina (2018)

O A-GPS torna-se superior à sua antecessora devido à melhora de precisão e disponibilidade das coordenadas do dispositivo em tempo real, apesar dos custos cobrados pela transferência de dados sobre a rede celular pela operadora utilizada. Esse sistema não sofre alterações de propagação *multipath* (caminhos secundários que o sinal percorre até chegar à antena do receptor, como mostrado na Figura 3), por exemplo, de construções, ruas, edifícios, rios, lagos, veículos ou condições atmosféricas (OFICINA, 2018) (POLEZEL ENIUCE MENEZES DE SOUZA, 2004).

A maior utilidade do sistema de A-GPS é vista em áreas urbanas, repletas de desfiladeiros urbanos, melhorando a experiência do usuário para todos os aplicativos que usam o GPS integrado e reduzindo o tempo que um aparelho com GPS leva para localizar a sua posição atual, conhecido como TTFF (Tempo de Localização Inicial) (OFICINA, 2018).

Figura 3 – O problema do multipath



Fonte: Polezel Eniuce Menezes de Souza (2004)

2.3.1 Diferenças entre GPS e A-GPS

Por um lado, o GPS realiza uma busca por satélites disponíveis para auxiliá-lo na navegação. Quando o GPS encontra o primeiro disponível, o aparelho recebe informações sobre a localização dos outros satélites próximos, iniciando a roteirização da localização atual do mesmo. Esse processo é demorado, pelo fato de que a conexão GPS entre os satélites é muito lenta (OFICINA, 2018).

Por outro lado, para dispositivos A-GPS, são utilizados servidores, como, uma torre de rede móvel (antenas de celulares) e bases (servidores) para obter as informações dos satélites constantemente. Como esses servidores estão continuamente enviando e recebendo informações dos satélites, não há atraso para conhecer a órbita e a localização exata deles. O tempo para o recebimento dos dados é muito menor que um GPS normal, já que os dados já estão armazenados nos servidores, que analisam os sinais fragmentados recebidos do receptor GPS e aqueles recebidos diretamente do satélite para corrigir erros, proporcionando uma localização exata do aparelho, uma vez que só precisará realizar alguns cálculos de triangulação com as antenas de redes móveis (OFICINA, 2018).

Ainda conforme Oficina (2018), a conexão inicial é realizada com uma antena de telefonia celular que previamente armazenou a localização desses satélites e a transmite para o aparelho com uma velocidade até 40 vezes maior. Esse recurso só é possível em aparelhos com conexão GPRS, caso esse sinal esteja fraco devido a edifícios muito altos, o A-GPS utiliza o apoio da antena quando o sinal GPS, ajudando a manter o sinal estável. Não existindo cobertura de celular, o aparelho funcionará como um GPS convencional. O uso do A-GPS torna a utilização mais eficaz, evitando o aguardo pela conexão entre satélites.

Na Figura 4 é possível visualizar como é feita a localização GPS e A-GPS.

Figura 4 – Localização GPS e A-GPS



Fonte: Oficina (2018)

2.4 Google Maps

O *Google Maps* é uma aplicação fornecida pela Google amplamente utilizada por aqueles que necessitam de auxílio para chegar a um local desconhecido. Além de oferecer o serviço de mapas, o software inclui outras características muito úteis, como as ferramentas para calcular as direções entre dois pontos. Pode-se obter mais informações sobre qualquer lugar, graças à utilização de fotos, *webcams* e artigos. Seja pelos seus recursos básicos, como encontrar endereços específicos e verificar trajetos e distâncias entre dois ou mais pontos, ou pelos avançados, como a visão de satélite e o *Google Street View*. Em parte, graças à sua enorme abrangência, poucas cidades e ruas ainda não estão mapeadas nesse serviço, permitindo fazer o download de uma área no dispositivo para ser usado sem conexão com a internet. Os mapas salvos dessa forma são renovados mensalmente por meio de redes *wi-fi*, caso o usuário fique mais de 1 mês sem usá-los, o aplicativo sugerirá que ele renove a área ou a apague para economizar espaço. Esse recurso é especialmente útil para viagem a outro país e sem plano de dados móveis (GOOGLE, 2019b). Na Figura 5, pode-se ver um exemplo de visualização de mapa do *Google Maps*.

Figura 5 – Exemplo de visualização do Google Maps



Fonte: Google (2019a)

A principal ferramenta utilizada no *Google Maps* é a busca por endereços. A partir de um computador, independentemente do navegador e do sistema operacional, basta digitar o endereço que deseja encontrar. Além de digitar o local manualmente, é possível indicá-lo diretamente no mapa. Também, é possível descobrir novos locais, como lojas, restaurantes, entre outros pontos de interesse facilmente, através de categorias e subcategorias, como “para ir com crianças” ou “para gastar pouco”. Os locais são escolhidos com base na localização atual e que se adaptam ao gosto de cada usuário. É comum ser disponibilizado mais de um caminho para o destino final, exibindo, por padrão, o mais rápido, levando em consideração a variação de trânsito para o horário do momento. Isso é possível porque o *Google Maps* acumula informações sobre o tráfego da região. Ou seja, ele sabe qual é o fluxo de veículos para um determinado local em um determinado horário. Além de encontrar qualquer localidade, demonstrar informações meteorológicas ou indicações e caminhar ao redor das cidades mais importantes do globo, são seus diferenciais (GOOGLE, 2019b).

Para ajudar quem anda de ônibus, metrô ou trem, pelas cidades do mundo, é possível saber a melhor rota de transporte público, qual o ponto de ônibus mais perto e os locais de parada do coletivo. Dispondo de serviços de altitude para ciclistas, é possível identificar as variações do terreno em metros durante o trajeto, além de levar em conta a estrutura ciclovária da cidade. Às vezes, esse caminho pode não ser o mais direto entre os dois pontos, mas ele pretende ser o mais seguro e agradável para quem está pedalandando (GOOGLE, 2019b).

Com uma enorme base de dados, serviços como Booking, Decolar ou Kayak, podem ser encontrados ao pesquisar por um hotel na plataforma da Google, contando com uma ferramenta interna para ajudar seus usuários, que apresenta diversas opções de quartos vagos na região, e seus preços médios. Caso o local de visita escolhido durante uma viagem seja bem conhecido, é possível fazer um *tour* pelo espaço interno mapeado. No Maracanã, por exemplo, é possível visualizar a divisão das arquibancadas. Muitas das opções de uso citadas fazem parte da *Google Street View*, que mapeia locais possíveis de caminhar virtualmente por várias cidades ao redor do mundo, o que permite encontrar exatamente a fachada do local desejado. Além do *Google Earth* e dos *Indoor Maps*, o *Google Maps* traz também outra função capaz explorar o mundo a partir da tela do computador chamada *Google Treks*, desenvolvido para visitar virtualmente alguns lugares da Terra (GOOGLE, 2019b).

Ao selecionar a cidade de partida, a cidade de chegada e o meio de transporte “Avião”, o *Google Maps* além de exibir informações de voos disponíveis para o trajeto, mostra a duração do voo e quais companhias aéreas operam aquele trajeto (GOOGLE, 2019b).

O recurso mais recente adicionado à plataforma, permite planejar uma rota com *way-points*. Esse mecanismo permite fazer percursos que incluam diversas paradas, indicando o melhor caminho do ponto A até B, e depois de B até C, sem precisar mudar a rota novamente e, portanto, é muito usado para encontrar o melhor caminho para visitar diversos pontos turísticos de uma cidade no mesmo dia (GOOGLE, 2019b).

Várias ferramentas de visualização geoespacial dos principais *players* da indústria da internet estão tomando o mundo digital. Google, Yahoo, Microsoft e Amazon lançaram ferramentas de mapeamento baseadas em geolocalização e, coletivamente, elevaram o nível de mapeamento da internet. Por mais que suas capacidades funcionais não forneçam nada muito diferente do que fornecedores tradicionais da GIS (*Geographic Information System*) já tinham, o surgimento desses serviços foi notório na medida em que eles conseguiram captar um público mais amplo. O Google entrou como líder desses serviços com seu produto *Google Maps*, que fornece uma interface visual atrativa, construídas com tecnologia AJAX, além de dados detalhados de imagens aéreas da superfície terrestre, e uma API aberta que permite a personalização do mapa, incluindo a capacidade de adicionar dados específicos ao mapa e de traçar rotas desejadas (GEOSPATIAL, 2009).

2.4.1 Google Maps Platform

A plataforma para desenvolvedores da Google se chama *Google Maps Platform*, em que 99% de cobertura no mundo, com a abrangência de mais de 200 países e territórios, estão disponíveis. Contemplando informações de local precisas e em tempo real, são mais de 25 milhões de atualizações por dia e 1 bilhão de usuários ativos por mês, tudo isso gerando uma grande base de dados para consulta (GOOGLE, 2019b). Nela, obtêm-se soluções para cada setor:

- Serviço de transporte particular: disponibiliza a integração com qualquer aplicativo de serviço de transporte particular para diminuir os tempos de espera dos clientes e os erros de navegação dos motoristas;
- Jogos: permite criar jogos imersivos e realistas com milhões de estruturas em 3D personalizáveis, dados globais atualizados e integração perfeita com o Unity;
- Rastreamento de recursos: possibilita maior eficiência na localização de veículos e os recursos em tempo real, visualizando o trajeto dos mesmos e orientando os motoristas durante viagens complexas.

2.4.1.1 Maps

O recurso *Maps*, presente na *Google Maps Platform*, leva o mundo real até os usuários por meio de mapas estáticos e dinâmicos, imagens do *Google Street View* e visualizações 360°, visualizando o mundo com mapas eficientes e precisos, podendo ser incorporado em sites ou aplicativos. O *Street View* e as imagens de satélite permitem trabalhar com personalizações e sobreposições capazes de se adequar à necessidade de cada usuário, com uma infraestrutura sempre disponível para ampliação (GOOGLE, 2019b).

2.4.1.2 Routes

Através de dados abrangentes e trânsito em tempo real, encontrar o melhor caminho até o destino final se torna um trabalho rápido com o uso do recurso denominado *Routes*. Com informações de navegação confiáveis e mais de 64 milhões de quilômetros de estradas em todo o mundo, traçar percursos entre cidades ou encontrar hotéis, permite maior eficiência, redução de custos e melhora a experiência dos clientes finais. Esse recurso permite planejar viagens munido de dados atualizados sobre a distância entre pontos, trajetos sugeridos e tempos estimados de deslocamento, com a capacidade de criar trajetos eficientes para até 25 pontos de referência, agilizando os sistemas de entrega, criando itinerários para turistas ou guiando os clientes que alugam carros do seu escritório até os hotéis, realocando o mesmo caso o caminho escolhido possua tráfego (GOOGLE, 2019b). Sua eficiência é justificada pelo agrupamento dos seguinte recursos:

- *Directions*: fornece rotas de transporte público, bicicleta, carro e a pé, calcula os tempos de deslocamento atuais ou futuros com base no trânsito em tempo real;
- *Distance Matrix*: calcula tempos de deslocamento e distâncias para um ou mais locais;
- *Roads*: permite criar itinerários precisos determinando o trajeto percorrido pelo veículo e as vias mais próximas ao longo de cada ponto da viagem do veículo.

2.4.1.3 Places

Contempla informações avançadas de mais de 150 milhões de pontos de interesse, permitindo encontrar locais específicos por meio de números de telefone, endereços e sinais em tempo real. Com usuários atualizando a todo momento, 24 horas por dia, as avaliações medidas pela Google se tornam confiáveis, capazes de planejar uma viagem ou indicar um restaurante por meio dos gostos dos usuários. Com o *Places*, é possível compartilhar detalhes sobre nomes de locais, endereços, classificações, comentários, dados de contato e ambiente. Os Local Guides e os usuários enviam milhões de atualizações todos os dias (GOOGLE, 2019b). Este recurso é o maior agrupador de todos, abrange:

- *Place Details*: fornece nomes, endereços e outros detalhes valiosos, como classificações, avaliações ou dados de contato de pontos de interesse;
- *Current Place*: identifica um local com base nos sinais em tempo real, como hora do dia ou local;
- *Find Place*: localiza usando o número de telefone, endereço ou nome do local;
- *Automatic Filling*: sugere informações de local enquanto os usuários digitam;
- *Geocoding*: converte endereços em coordenadas geográficas e vice-versa;
- *Geolocation*: captura o local preciso de um dispositivo usando o *wi-fi* ou torres de celular como base;
- *Time Zone*: mostra o fuso horário de qualquer local.

3 FERRAMENTAS

Este Capítulo descreve as ferramentas utilizadas para desenvolvimento do presente trabalho.

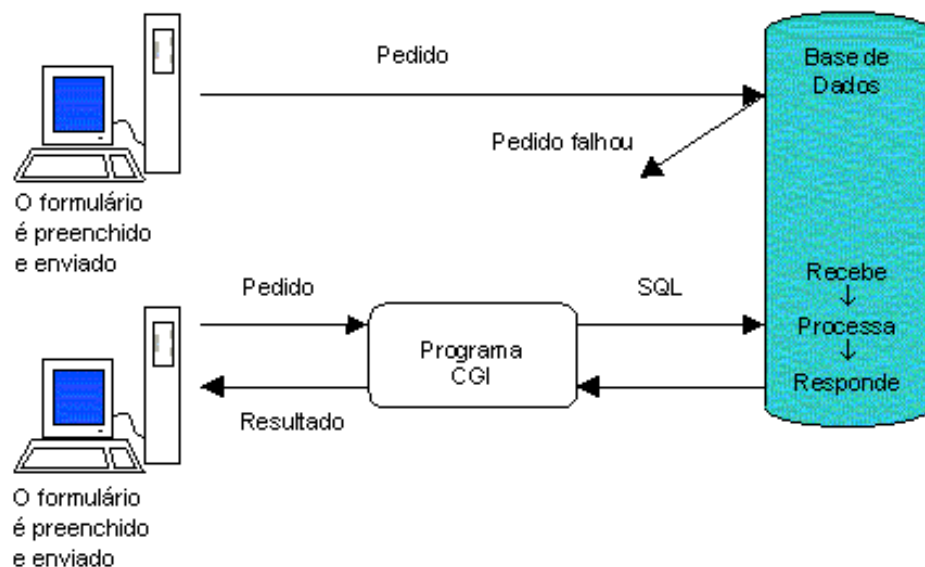
3.1 CGI (Common Gateway Interface)

Côrte (2002) explica que CGI é uma forma de interligação entre aplicativos externos com servidores de informação (servidores HTTP ou servidores web). Uma página HTML comum é composta por informações estáticas. Um programa CGI permite a montagem de uma página HTML em tempo real, com informações mais específicas ao interesse do usuário.

Por exemplo, digamos que se queira disponibilizar informações contidas em uma base de dados para qualquer usuário de um servidor de informações. Para isso, tem-se um formulário inicial, onde o usuário preenche determinadas informações e, ao final, aciona um botão na tela, submetendo o pedido. Essas informações são passadas ao programa CGI, que processa essas informações, consulta uma base de dados e retorna as informações solicitadas através de uma página específica, como pode ser visto na Figura 6 (CÔRTE, 2002).

Um programa CGI pode ser escrito em qualquer linguagem que permita ser executado no sistema. Contudo, esses programas necessitam residir em diretórios especiais (geralmente nomeado *cgi-bin*), por questões de segurança e também para diferenciar de páginas comuns. Assim, o servidor sabe que esses programas devem ser executados e não simplesmente mostrados. Cada vez que o cliente pede o URL correspondente ao programa CGI, este é executado em tempo real, através da criação de um processo específico no servidor (CÔRTE, 2002).

Figura 6 – Estrutura CGI



Fonte: Adaptada de FEUP (2019)

3.2 PHP

PHP é um acrônimo recursivo para PHP: *Hypertext Preprocessor* (Pré-Processador de Hipertexto), que originalmente se chamava *Personal Home Page* (Página Inicial Pessoal). É uma linguagem de *script open source* de uso geral, muito utilizada, especialmente adequada para o desenvolvimento web e que pode ser embutida dentro do HTML, segundo PHP Development (2019).

Bento (2014) explica que o PHP é uma ferramenta que possibilita o pré-processamento de páginas HTML. Dessa forma, o PHP consegue alterar o conteúdo de uma página, antes de enviá-la para o navegador. Além disso, o PHP também permite capturar entradas de dados do usuário, como formulários e outras formas de interação. Trata-se de uma linguagem altamente popular devido à sua natureza de código aberto e suas funcionalidades versáteis, sendo uma linguagem de *script*, escrita de forma procedural, orientada a objetos ou ainda ambas, do tipo *server-side* com diversos propósitos. Porém, ela é principalmente utilizada para gerar conteúdos dinâmicos em um site.

Destaca-se do pensamento de Bento (2014), que existem diversos motivos para escolher esta tecnologia, dentro eles:

- Nasceu para a web e sua integração com qualquer sistema operacional, servidor web e banco de dados, como: dBase, Interbase, MySQL, Oracle, Sybase, PostgreSQL e vários outros, é simples e econômica;
- Curva de aprendizado suave, comparada a outras linguagens;
- Tecnologia livre;
- Suporte para conversar com outros serviços usando protocolos como: IMAP, SNMP, NNTP, POP3 e, logicamente, HTTP. Ainda é possível abrir *sockets* e interagir com outros protocolos.

As origens do PHP datam de 1994, criado com apenas um aglomerado de códigos binários CGI escritos em linguagem C, para fazer a ligação lógica entre dois sistemas ou servidores pela internet. Esse mesmo conjunto de códigos, que nada mais era do que um amontoado de *scripts*, foi inicialmente nomeado como PHP/FI, uma versão prematura do PHP. A ideia inicial era acompanhar o tráfego do site pessoal do criador. Os anos passaram e o criador desenvolvia *scripts*, o que aumentava os recursos que o site dele tinha (PHP DEVELOPMENT, 2019).

Ainda conforme PHP Development (2019), o sucesso dessa linguagem foi tão grande que o criador, Rasmus Lerdorf, transformou o aglomerado de códigos CGI em uma linguagem de programação. Com isso, a grande maioria dos sites e aplicações passaram a utilizar o PHP como linguagem principal.

3.3 Laravel

Laravel é um *framework* livre, de código aberto, voltado para desenvolvimento de sistemas baseado em internet e feito em PHP. Tem como principal característica auxiliar no desenvolvimento de aplicações seguras e performáticas de forma rápida, com código limpo e simples, já que ele incentiva o uso de boas práticas de programação e utiliza o padrão PSR-2 como guia para estilo de escrita do código. Foi criado por Taylor Otwell e tem como principal função desenvolver aplicações web com padrão de arquitetura MVC. Algumas características nativas do Laravel são: sua sintaxe descomplicada e concisa, um sistema modular com gerenciador de dependências dedicado, várias formas de acesso a banco de dados relacionais e vários utilitários indispensáveis no auxílio ao desenvolvimento e manutenção de sistemas. O código fonte do Laravel está hospedado no GitHub e licenciado sob os termos da licença MIT (Instituto de Tecnologia de Massachusetts) (GSTI, 2019).

Segundo Locaweb (2019), Laravel é o *framework* que está no *top trends* dos mais buscados no Google, isso desde agosto de 2014, que coincide com o ano de lançamento da versão 5.0. Sua divisão em *routes*, *controllers* e *models*, sendo compatível com diversos bancos de dados, trabalhando de forma relacional ou orientada a objetos como: MySQL, Postgress, Redis, MongoDB, Cassandra e SQL Server, utilizando *features Scaffold* ou Internacionalização (*i18n*), faz com que o Laravel seja altamente compatível. Sendo assim, basta ajustar as configurações e definir qual o banco de dados desejado e o sistema mantém seu funcionamento caso a estrutura estiver pronta, utilizando uma implementação simples do *ActiveRecord* chamada de Eloquent ORM, que é uma ferramenta que traz várias funcionalidades para facilitar a inserção, atualização, busca e exclusão de registros. Para a criação de interface gráfica, o Laravel utiliza uma *engine* de *template* chamada Blade, que traz uma gama de ferramentas que ajudam a criar interfaces bonitas e funcionais de forma rápida e evitar a duplicação de código.

O Laravel é uma estrutura de aplicativos da web com sintaxe expressiva e elegante. Acreditamos que o desenvolvimento deve ser uma experiência agradável e criativa para ser verdadeiramente gratificante. O Laravel tenta aliviar a dor do desenvolvimento, facilitando tarefas comuns usadas na maioria dos projetos da web (LARAVEL, 2019).

Com sua baixa curva de aprendizagem e elegância, seu diferencial é a grande quantidade de recursos nativos que o mesmo contém para auxiliar o processo de desenvolvimento. Além disso, o *framework* tem como objetivo aumentar a velocidade de codificação, sem esquecer características importantes como a segurança e *performance* da aplicação.

Segundo Das e Saikia (2016), afirmaram que ao comparar PHP procedural com CodeIgniter e Laravel, o Laravel tem bom desempenho e menos tempo de execução para todas as quatro ações. Os resultados em milissegundos de dez entradas como média para as quatro ações são apresentados na Tabela 1.

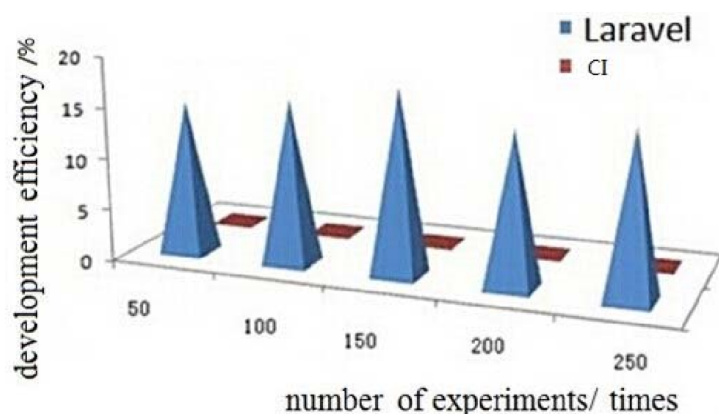
Tabela 1 – Resultado do experimento no tempo de execução

	CREATE	READ	UPDATE	DELETE
PLAIN PHP	0.066	0.063	0.0925	0.065
CODEIGNITER	0.0634	0.0747	0.0825	0.616
LARAVEL	0.201	0.222	0.312	0.301

Fonte: Das e Saikia (2016)

He (2015) também prova a eficácia do método de projeto web com base na estrutura do Laravel, criando um ambiente operacional virtual e comparando com o mesmo ambiente desenvolvido em CodeIgniter. De acordo com a Figura 7, a eficiência do desenvolvimento baseado em Laravel é maior em comparação com o método tradicional de design da web baseado na estrutura do CodeIgniter, justificado pela escalabilidade robusta, de modo a melhorar a eficiência do desenvolvimento.

Figura 7 – Tendências de eficiência de desenvolvimento - Laravel e CodeIgniter



Fonte: He (2015)

3.4 Eloquent

O Eloquent é o *Object Relational Mapper* (ORM) padrão do Laravel. É uma técnica de desenvolvimento de software usada para reproduzir tabelas de banco de dados sem formato de objetos relacionais, deixando a manutenção dos dados mais simples e fácil. Com esta técnica o programador não precisa escrever códigos na linguagem *Structured Query Language* (SQL). Com a interface ORM é possível realizar operações de CRUD, além de criar as tabelas e montar relacionamentos 1:1 (um para um), 1:N (um para muitos) e N:M (muitos para muitos). Utilizando um padrão de registro ativo, o Eloquent transforma a tabela do banco de dados em uma estrutura MVC (*Model-View-Controller*). Por exemplo, para listar todas as entregas da tabela de entregas ou encontrar uma entrega específica basta usar funções simples e intuitivas que equivalem à *query* SQL desejada (Quadro 2).

Quadro 2 – Comparativo SQL vs. Eloquent ORM

SQL	Eloquent ORM
<code>select * from deliveries</code>	<code>Delivery::all()</code>
<code>select * from deliveries where id = 1</code>	<code>Delivery::find(1)</code>

Fonte: Autor

Segundo Stauffer (2017), o Eloquent ORM implementa o *ActiveRecord*, o que significa que uma única classe (*model*) é responsável por interagir com a tabela de forma completa, podendo inserir, atualizar e excluir novos registros.

3.5 Artisan

Artisan é o nome da interface de linha de comando incluída no Laravel, um conjunto de ações internas com a oportunidade de inclusão de outras, executadas pela CLI (*Command Line Interface*). Com o Artisan, o desenvolvedor é capaz de acionar ações, como testes de unidade, tarefas agendadas e execução de migrações do banco de dados (MCCOOL, 2012). Com os comandos do Artisan e seus argumentos opcionais, esboços das classes que compõem a aplicação podem ser gerados, acelerando o processo de desenvolvimento e minimizando a quantidade de erros de programação.

Fragmento de Código 1 – Comandos Artisan

```
1 php artisan make:model Payment -m
2 php artisan make:controller PaymentsController
```

Conforme o Fragmento de Código 1, no primeiro comando a ferramenta já cria diversos arquivos e configurações no projeto para implementar a entidade *Payment*, bastando agora somente preencher seus atributos, relacionamentos e regras de negócios. Do mesmo modo, o segundo comando já monta o arquivo padrão de um controlador.

3.6 MySQL

Segundo Thomson e Welling (2005), o MySQL é o SGBDR (Sistema Gerenciador de Banco de Dados Relacional) mais utilizado atualmente, devido à sua *performance* rápida e consistente, além de estar disponível gratuitamente para fins não lucrativos, distribuído sob a licença GNU. O objetivo de um banco de dados é permitir o armazenamento, a classificação e a recuperação dos dados de uma forma eficiente. Contudo, um sistema de banco de dados tem como vantagens minimizar as incoerências, garantir a integridade dos dados, facilitar a realização de qualquer modificação necessária e possibilitar a criação de restrições de acesso, tudo de maneira muito organizada.

Desenvolvido pela empresa sueca MySQL AB e publicado, originalmente, em maio de 1995. Após, a empresa foi comprada pela Sun Microsystems e, em janeiro de 2010, integrou a compra da Sun pela Oracle Corporation. Atualmente, a Oracle, embora tenha mantido a versão para a comunidade, tornou seu uso mais restrito e os desenvolvedores criaram, então, o projeto MariaDB para continuar desenvolvendo o código da versão 5.1 do MySQL, de forma totalmente aberta e gratuita. O MariaDB pretende manter compatibilidade com as versões lançadas pela Oracle (MARIADB, 2019).

Bento (2014) explica que PHP e MySQL já são velhos amigos. É bem comum encontrar aplicações que fazem uso destas tecnologias em conjunto. Desde pequenos sites pessoais, até grandes sistemas de gestão e lojas virtuais. O que torna esta parceria tão forte e estável é uma combinação de fatores, como o fato de o MySQL ser um banco fácil de aprender, leve e rápido, o PHP ser fácil e flexível e ambos possuírem licenças permissivas para uso pessoal, comercial e em projetos de software livre, através de funções que conectam, executam código SQL e trazem os resultados para a aplicação.

Pires, Nascimento e Salgado (2008) comparam em seu artigo três módulos: carga e estrutura, mono-usuário e multi-usuário nos bancos de dados: MySQL e PostgreSQL, seguindo as métricas do OSDB (*Open Source Database Benchmark*), em plataforma Linux. Para os testes, foram criadas duas bases de dados com tamanhos distintos: 512MB e 1GB. Para esses tamanhos de base a quantidade de linhas por tabela é de 1.280.000 (512MB) e 2.500.000 (1GB), respectivamente.

De acordo com o Quadro 3, o desempenho do PostgreSQL foi superior ao MySQL apenas no módulo de carga e estrutura, especialmente na criação de índices. Nos demais módulos, o MySQL mostrou-se superior, apresentando melhores resultados na maioria dos testes. Diferentemente do PostgreSQL, o otimizador de consulta do MySQL optou por utilizar índice. Em alguns SGBD, a escolha de índices é influenciada pela existência de estatísticas sobre as estruturas de armazenamento. No PostgreSQL, as estatísticas são geradas apenas sob demanda. Como o kit do *benchmark* não contempla a geração de estatísticas em seu código, isso ajuda a explicar o resultado (PIRES; NASCIMENTO; SALGADO, 2008).

Quadro 3 – Resultados do Sumário Executivo

Módulo	512MB		1GB	
	PostgreSQL	MySQL	PostgreSQL	MySQL
Carga e Estrutura	14min	39min	45min	1h23min
Mono-Usuário	27min	8min	57min	19min
Multi-Usuário	1h06min	50min	3h43min	1h34min

Fonte: Pires, Nascimento e Salgado (2008)

4 DESENVOLVIMENTO DA APLICAÇÃO

O presente trabalho apresenta uma API REST, a qual possibilita a coleta das coordenadas geográficas dos pedidos gerados nos estabelecimentos. Além disso, apresenta o desenvolvimento de uma aplicação para gerenciar, por meio de um painel de controle, a logística das entregas de comida, próteses, entre outras mercadorias. Tal aplicação tem por objetivo fornecer ao motoboy um módulo de entregas, possibilitando a ele observar no mapa a rota e os pontos de parada que devem ser percorridos. Todo o processo de comunicação da aplicação ocorre totalmente em tempo real, o que possibilita agilidade e segurança quanto ao destino final.

4.1 Softwares utilizados

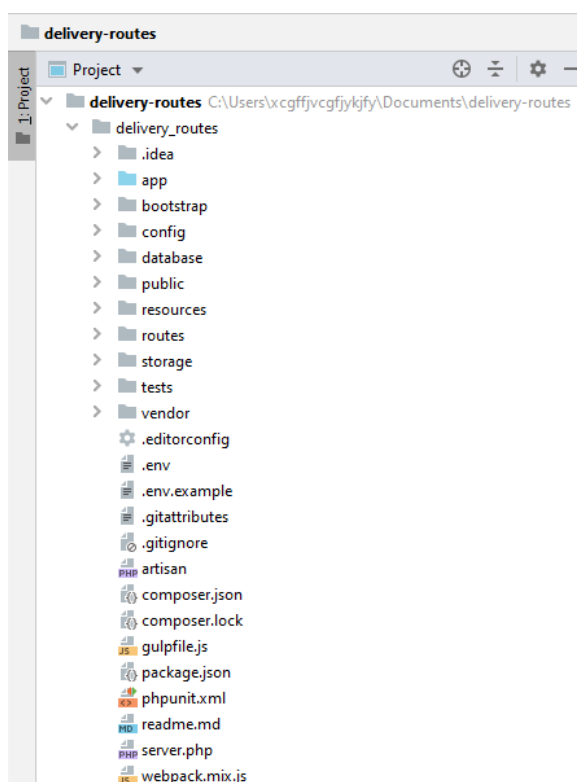
Esta Seção apresenta os softwares utilizados na etapa de codificação da aplicação. Para diminuir o custo de desenvolvimento, optou-se pela utilização de tecnologias gratuitas.

- PhpStorm: IDE completa de desenvolvimento para projetos codificados com a linguagem de programação PHP;
- Composer: gerenciador de pacotes em nível de aplicativo, que fornece um formato padrão para gerenciar dependências de software PHP e bibliotecas necessárias;
- Cmdr: console para execução de linhas de comandos para o sistema operacional Windows. Com essa ferramenta é possível executar comandos Unix diretamente no Windows;
- Laravel: *framework* de programação baseada na linguagem PHP;
- Artisan CLI: interface de linha de comando incluída no Laravel, fornecendo vários comandos úteis durante a criação da aplicação, por exemplo: configuração de ambiente, verificar rotas, interagir com a aplicação e criar diversos tipos de arquivos (*migrations*, *controllers* e *models*);
- Blade: ferramenta para criação de interfaces gráficas, utilizado pelo Laravel como uma ferramenta de *template*, trazendo uma quantidade grande de funcionalidades que ajudam na criação de interfaces interativas;
- Eloquent ORM: ferramenta com funcionalidades que facilitam a inserção, atualização, busca e exclusão de registros diretamente no banco de dados;
- XAMPP: servidor independente de plataforma, que inclui: Apache, MySQL, phpMyAdmin, FileZilla FTP Server, OpenSSL, PHP e Perl;
- Jaspersoft Studio: ferramenta para projetar e executar modelos de relatório com expressões, gráficos, mapas, tabelas e códigos QR, criando documentos de qualquer complexidade a partir de informações presentes no banco de dados;
- GitHub: plataforma de hospedagem de código-fonte para controle de versão;
- Sourcetree: representação visual de repositórios da nuvem.

Para o desenvolvimento deste trabalho, foi escolhido trabalhar com Laravel 5.8.11, um *framework* com um servidor interno já incluso e invocado através do comando *php artisan serve*, em conjunto com o XAMPP, o que facilita e agiliza o desenvolvimento. Como o conteúdo estará armazenado na rede local, o acesso aos arquivos é realizado instantaneamente, disponível no endereço *http://127.0.0.1*.

A execução do comando *composer create-project --prefer-dist laravel/laravel delivery-routes* no Cmdr gera, de forma padronizada, toda a estrutura de um projeto Laravel, apresentada na Figura 8. O uso do Cmdr torna o desenvolvimento, via terminal, do sistema operacional Windows mais amigável, uma vez que esta ferramenta possibilita a execução de comandos Unix em ambiente Microsoft.

Figura 8 – Estrutura Laravel



Fonte: Autor

A montagem do ambiente de desenvolvimento para deixá-lo funcional no *framework* Laravel necessitou de algumas configurações, tais como: primeiramente foi preciso instalar e configurar todas as dependências do PHP, para ter acesso à linguagem de programação através do gerenciados de dependências Composer.

Foi optado, por questões de controle de versionamento do projeto, hospedar o código-fonte do projeto em nuvem, através da plataforma GitHub. A sincronia entre os arquivos locais e o GitHub foi possível com o auxílio do sistema de controle de versões chamado Git, representado visualmente pelo Sourcetree, que deixa o trabalho braçal por linha de comando (*gitbash*) de lado, sendo também necessário sua instalação e configuração na máquina de desenvolvimento.

O banco de dados escolhido para armazenar os registros foi o MySQL, cuja instalação e configuração ocorreu automaticamente pelo XAMPP. Por fim, para ter acesso à manipulação do código, um editor de texto é necessário. No caso descrito, optou-se pelo PHPStorm (IDE paga, mas que, por meio do e-mail acadêmico da URI, foi obtida a licença estudantil), terminando assim a fase configuração de ambiente de desenvolvimento e garantindo aptidão para começar os trabalhos.

O próximo passo foi a modelagem dos dados, onde verificou-se que seria necessário a criação das seguintes tabelas com o recurso *Migration*: *users*, *motoboys*, *deliveries*, *payments* e *orders*. Uma sexta tabela chamada de *deliveries_orders* precisou ser criada para vincular mais de um pedido à cada entrega, onde uma *trigger*, disparada via *after update*, ficou responsável pela atualização do status de cada pedido vinculado à entrega, esta chamada de *tr_Status_Orders*, apresentada no Fragmento de Código 2. Na próxima Seção serão abordados os possíveis status de um pedido.

Fragmento de Código 2 – Delivery Routes - Trigger tr_Status_Orders

```

1 CREATE TRIGGER tr_Status_Orders AFTER UPDATE ON `deliveries`
2 FOR EACH ROW BEGIN
3     UPDATE `orders` o SET `status` = NEW.status WHERE `id` IN
4         (SELECT `order_id` FROM `deliveries_orders`
5          WHERE `delivery_id` = NEW.id);
6 END

```

O recurso do Laravel chamado *Migration*, tem por objetivo gerenciar as mudanças estruturais do banco de dados da aplicação. Para cada tabela, coluna ou índice criados no banco de dados, é possível usar a *migration* para fazer essa operação automática, visando organizar e definir uma sequência de criação das tabelas. Para aplicações com poucas tabelas a utilização de *migrations* para gerenciar a estrutura das tabelas é extremamente útil e válida.

O Fragmento de Código 3 demonstra a criação de uma *migration*, por meio do comando *php artisan make:model Delivery -m*. O parâmetro “-m” indica que além do *model*, uma *migration* deve ser criada. Para atualizar o banco de dados executando todas as *migrations*, é necessário executar o comando *php artisan migrate*.

Fragmento de Código 3 – Delivery Routes - Exemplo de migration

```

1 Schema::create('deliveries', function (Blueprint $table) {
2     $table->increments('id');
3     $table->integer('motoboy_id')->unsigned();
4     $table->foreign('motoboy_id')->references('id')->on('motoboys');
5     $table->string('status',30);
6     $table->timestamps();
7 });

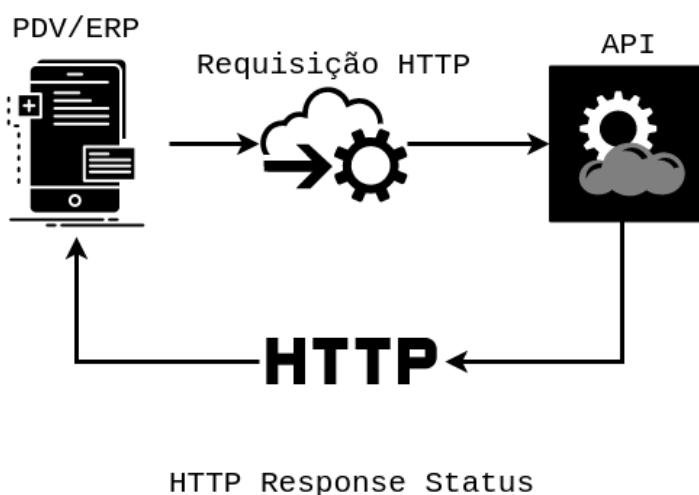
```

4.2 Módulo de gerenciamento

A concepção da ideia do projeto foi desenvolver uma aplicação completa para gestão de vendas, focando na geração do pedido e entrega, sendo essa para qualquer ramo de atividade. Porém, notou-se a falta de exploração de serviços em outro nicho de mercado ainda não explorado e sem concorrentes, com foco somente na otimização da entrega, em que não é preciso disputar clientes com grandes empresas maduras no assunto.

A aplicação desenvolvida foca no gerenciamento das rotas de entregas dos pedidos gerados via sistemas de vendas já operantes, necessitando apenas da disponibilização de uma API no formato de arquivo JSON (Figura 9), essa compondo-se de dados essenciais para realização da integração automática abordada na próxima Seção.

Figura 9 – Delivery Routes - Requisição HTTP



Fonte: iFood (2019)

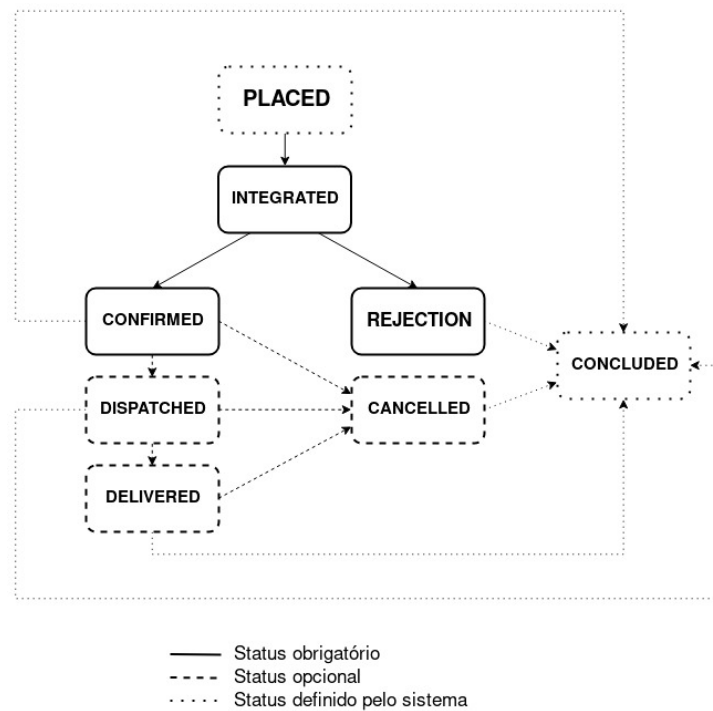
Primeiramente, é preciso entender os possíveis status e o fluxo de um pedido integrado no Delivery Routes, ambos representados na Tabela 2 e na Figura 10.

Tabela 2 – Delivery Routes - Status do pedido

Status	Descrição
PLACED	Indica um pedido disponível para integração no sistema
CONFIRMED	Indica um pedido disponível para entrega
INTEGRATED	Indica um pedido vinculado à uma entrega
CANCELLED	Indica um pedido cancelado
DISPATCHED	Indica um pedido despachado ao cliente
DELIVERED	Indica um pedido entregue pelo motoboy
CONCLUDED	Indica um pedido concluído pelo administrador

Fonte: Adaptada de iFood (2019)

Figura 10 – Delivery Routes - Fluxo do status do pedido



Fonte: iFood (2019)

Na Figura 11 é possível visualizar o acesso ao módulo de gerenciamento, com autenticação apenas para administradores, realizada via e-mail e senha, cujo cadastro é efetuado por meio do link *Register a new membership*. Habilitando a opção *Remember Me*, o controle de única sessão por usuário é ativado, por intermédio de um *hash* gerado no momento do login. É disponibilizado também o link *I forgot my password* para realizar a recuperação da senha, mediante a confirmação de um e-mail existente na base de dados.

Figura 11 – Delivery Routes - Login

Delivery Routes

Sign in to start your session

Email

Password

☐ Remember Me

[I forgot my password](#)

[Register a new membership](#)

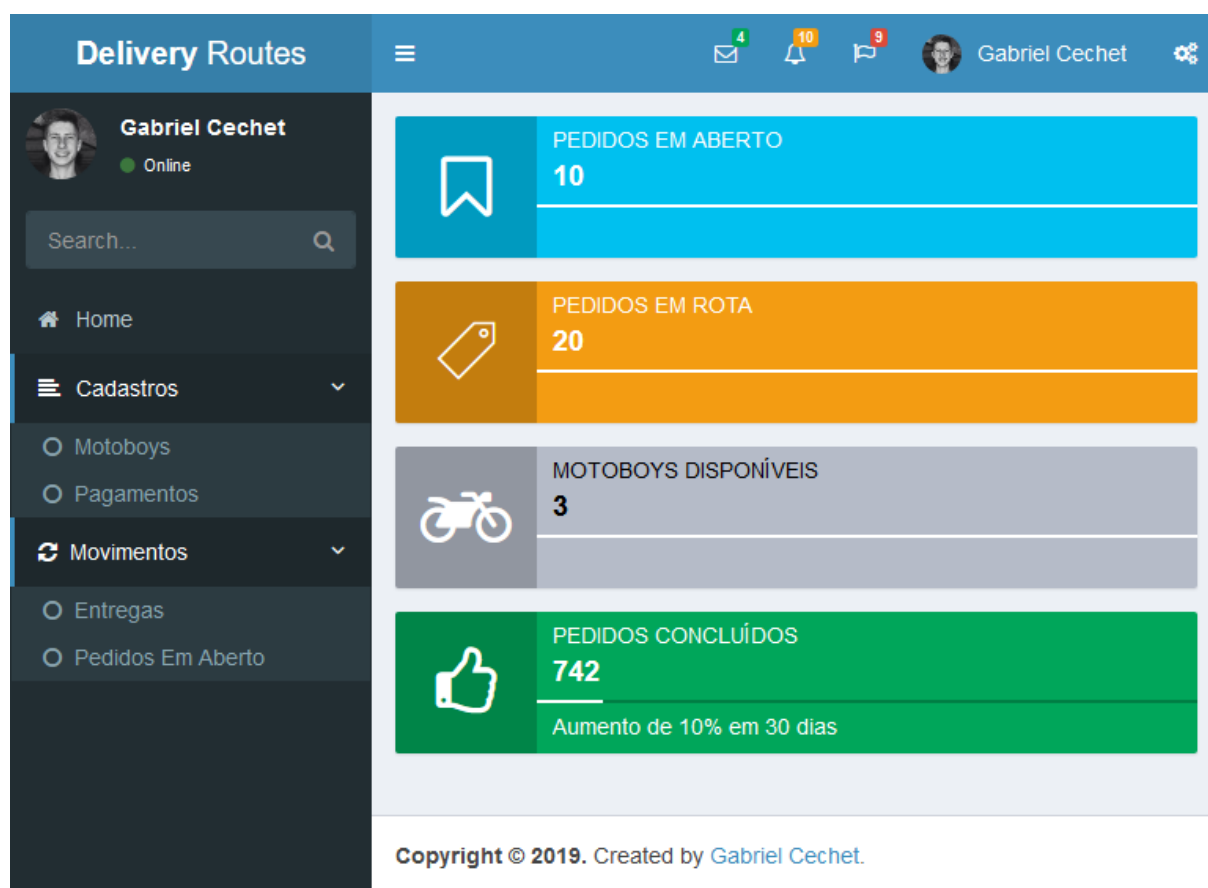
Fonte: Autor

O Laravel, juntamente com o Eloquent ORM, disponibiliza a implementação de autenticação de maneira muito simples, por meio do comando `php artisan make:auth`, onde, com poucas instruções e linhas de código, é possível montar a estrutura de cadastro de usuário, recuperação de senha e memória de sessão de maneira extremamente segura, devido ao algoritmo *Bcrypt*¹, utilizado para realizar a criptografia da senha.

Ao realizar login, o usuário é redirecionado à *dashboard* do sistema, uma interface gráfica que fornece visualização fácil e rápida dos principais indicadores de desempenho, atualizados em tempo real, obtendo assim valores e médias relevantes sobre o processo de negócios.

Destaca-se na Figura 12, além de indicadores de: Pedidos em aberto, Pedidos em rota, Motoboys disponíveis e Pedidos concluídos, uma barra de menus contendo: Cadastros (Motoboys e Pagamentos) e Movimentos (Entregas e Pedidos Em Aberto).

Figura 12 – Delivery Routes - Dashboard



Fonte: Autor

¹Método de criptografia do tipo *hash* adaptativo para senhas que apresenta uma segurança maior em relação à maioria dos outros métodos criptográficos por meio da implementação da variável “custo”, que é proporcional à quantidade de processamento necessária para criptografar a senha (PROVOS; MAZIÈRES, 1999).

Este projeto foi desenvolvido em duas frentes: um desenvolvimento *front-end* e outro *back-end*. O *front-end* foi baseado em um modelo, ou *template*, chamado *Pratt*, uma implementação em Blade do tema AdminLTE 5.4 (desenvolvido com Bootstrap). Já o *back-end* foi feito utilizando o *framework* Laravel, na linguagem PHP, como comentado anteriormente.

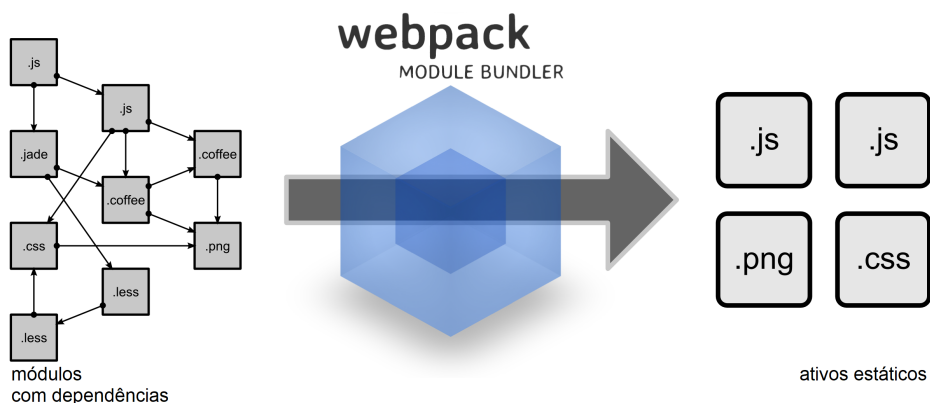
O Bootstrap é a biblioteca de componentes *front-end* mais popular do mundo (BOOTSTRAP, 2018), e é voltado principalmente para o desenvolvimento de projetos mobile. Todos os componentes oferecidos, como alertas, cartões e formulários, são responsivos, ou seja, eles se redimensionam, se escondem, diminuem ou ficam maiores dependendo do tamanho do dispositivo em que a página é renderizada. Atualmente a biblioteca está na versão 4, o que trouxe muitas melhorias em relação a versão 3, como a simplificação dos componentes de navegação. No entanto, a versão usada pelo AdminLTE ainda é a 3.0 e por isso esta é a versão utilizada neste projeto.

O AdminLTE é um *template* para desenvolvimento de *dashboards* ou painéis de controle. Possui código-aberto, construído usando Bootstrap e design responsivo. Por ser de código aberto, dúvidas ou problemas encontrados podem ser postados no repositório do projeto no GitHub ², onde qualquer membro da comunidade pode prestar ajuda e propor soluções.

O *Pratt* é uma implementação baseada no *template* responsivo AdminLTE, que por sua vez utiliza o *framework* Bootstrap. Utilizar projetos como o *Pratt* como base para desenvolver outras aplicações reduz o tempo necessário para iniciar o desenvolvimento e ajuda a evitar possíveis problemas de configurações que um programador pode encontrar.

Dessa forma, o *Pratt* vem com uma configuração pronta do Webpack (Figura 13), um *module blunder*, cujo principal objetivo, além de automatizar processos como a ofuscação, minificação e outras operações de pré-processamento de código, é permitir que o código seja escrito de forma modular, o que faz com que o desenvolvedor possa aproveitar todas as vantagens provindas do uso do *module blunder*, sem ter que se preocupar com sua configuração inicial.

Figura 13 – Webpack - Module blunder

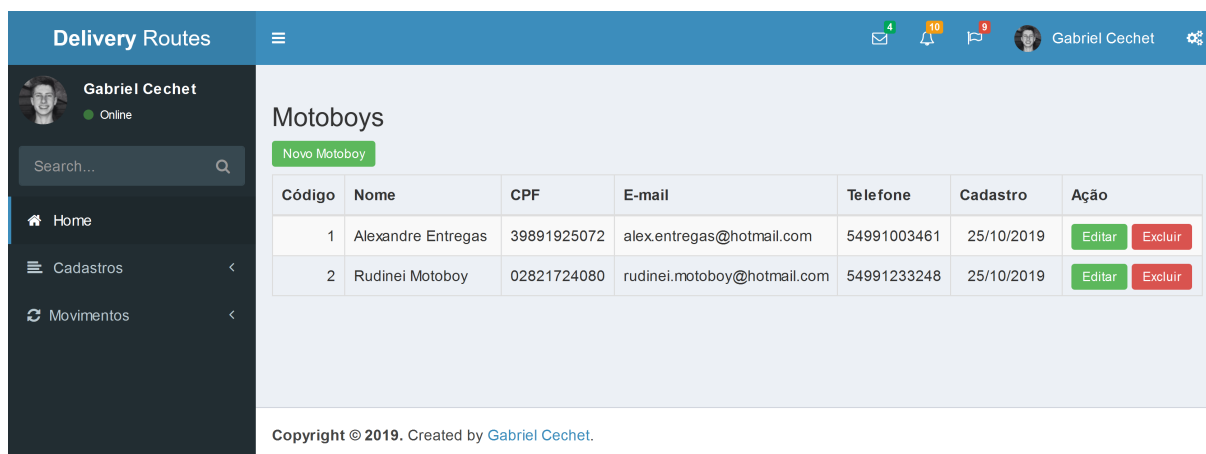


Fonte: Adaptada de Webpack (2019)

²<https://github.com/acacha/adminlte-laravel>

Ao selecionar a opção de menu “Motoboys”, o usuário consegue, então, gerenciá-los. Conforme é mostrado na Figura 14, há uma listagem com o identificador, nome, CPF, e-mail, telefone e data de criação dos registros cadastrados na base de dados. A listagem também conta com botões para editar, excluir ou criar cada registro. A edição do registro é apresentada na Figura 15.

Figura 14 – Delivery Routes - Lista de motoboys

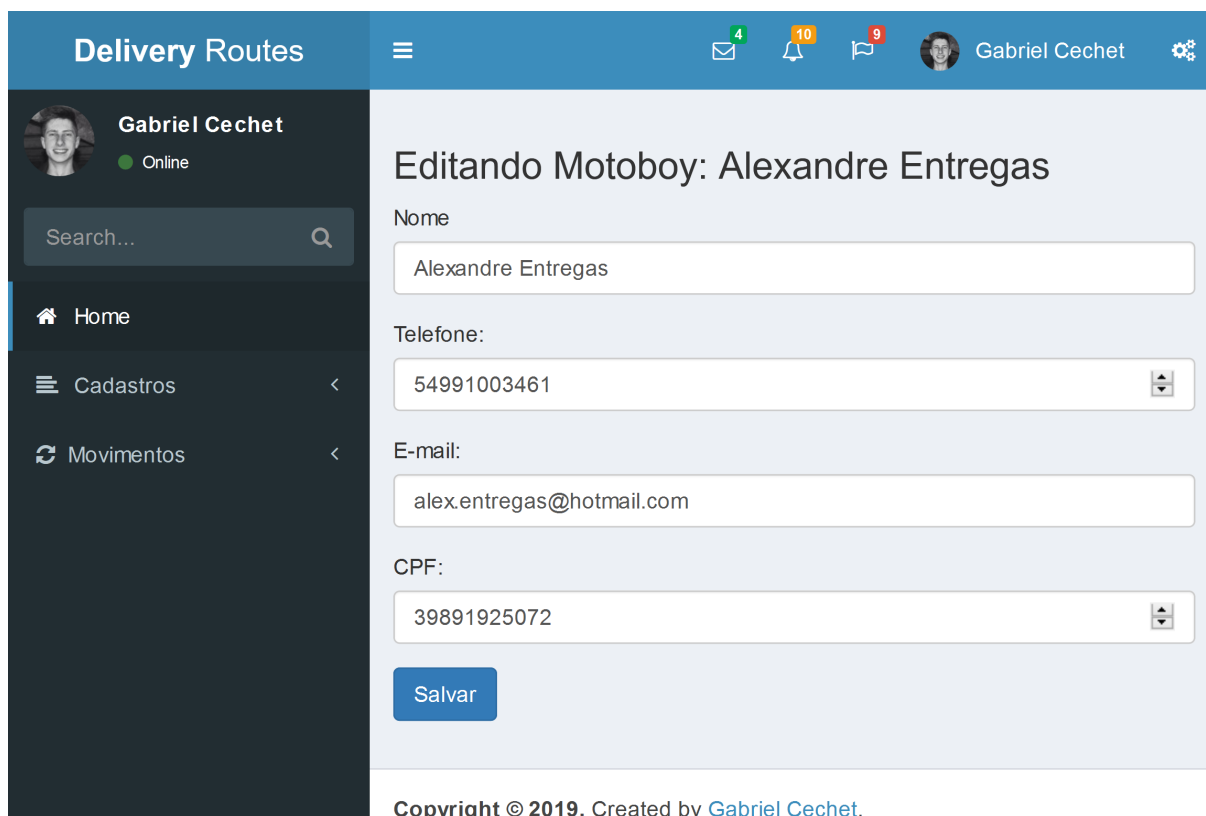


Código	Nome	CPF	E-mail	Telefone	Cadastro	Ação
1	Alexandre Entregas	39891925072	alex.entregas@hotmail.com	54991003461	25/10/2019	Editar Excluir
2	Rudinei Motoboy	02821724080	rudinei.motoboy@hotmail.com	54991233248	25/10/2019	Editar Excluir

Copyright © 2019. Created by [Gabriel Cechet](#).

Fonte: Autor

Figura 15 – Delivery Routes - Edição do motoboy



Editando Motoboy: Alexandre Entregas

Nome

Telefone:

E-mail:

CPF:

[Salvar](#)

Copyright © 2019. Created by [Gabriel Cechet](#).

Fonte: Autor

Igualmente à tela de motoboys, a tela de pagamentos possui uma listagem, dessa vez com o identificador e descrição dos registros cadastrados na base de dados, essa visualizada na Figura 16.

A edição do registro é apresentada na Figura 17.

Figura 16 – Delivery Routes - Lista de pagamentos



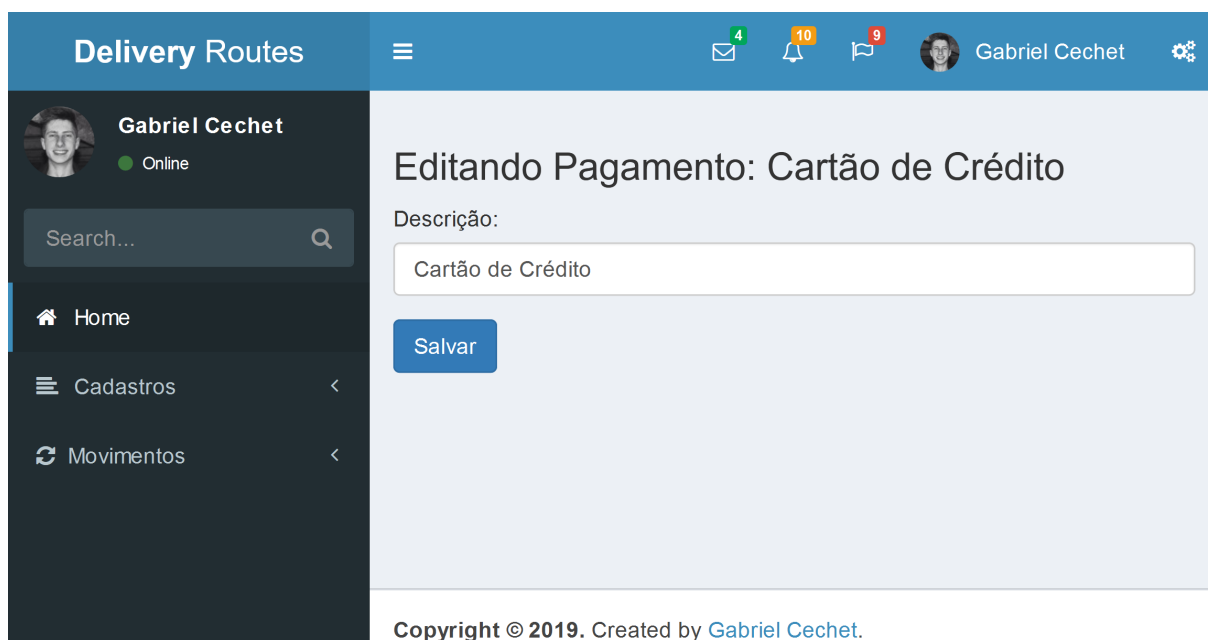
The screenshot displays the 'Pagamentos' (Payments) section of the 'Delivery Routes' application. The interface features a dark sidebar on the left with navigation links: Home, Cadastros, and Movimentos. The main content area has a header 'Pagamentos' and a 'Novo Pagamento' button. Below this is a table with the following data:

Código	Descrição	Ação
1	Cartão de Crédito	Editar Excluir
2	Dinheiro	Editar Excluir

At the bottom of the screen, there is a copyright notice: 'Copyright © 2019. Created by Gabriel Cechet.'

Fonte: Autor

Figura 17 – Delivery Routes - Edição do pagamento



The screenshot displays the 'Editando Pagamento: Cartão de Crédito' (Editing Payment: Credit Card) screen. The main content area has a header 'Editando Pagamento: Cartão de Crédito' and a 'Descrição:' label. Below the label is a text input field containing the value 'Cartão de Crédito'. A blue 'Salvar' (Save) button is positioned below the input field. The sidebar and top bar are consistent with the previous figure. At the bottom, the copyright notice 'Copyright © 2019. Created by Gabriel Cechet.' is visible.

Fonte: Autor

Ao selecionar a opção de menu “Entregas” (Figura 18), é possível visualizar todo o histórico de entregas já realizadas na aplicação. Nela estão presentes informações como o identificador, motoboy, status e os pedidos que compõem cada entrega. Botões para visualização, impressão da comanda, finalização (altera o status da entrega para *CONCLUDED*) e cancelamento (altera o status da entrega para *CANCELLED*) acompanham cada registro.

Figura 18 – Delivery Routes - Lista de entregas

DR

☰

4

✉

10

🔔

9

🚩



Gabriel Cechet











Entregas

Código	Motoboy	Status	Pedidos	Ação																
1	Alexandre Entregas	CONFIRMED	<table><tr><th>Código</th><th>Pagamento</th><th>Total</th><th>Pago</th></tr><tr><td>1</td><td>Cartão de Crédito</td><td>R\$ 22,00</td><td>Não</td></tr><tr><td>2</td><td>Dinheiro</td><td>R\$ 25,50</td><td>Não</td></tr><tr><td>3</td><td>Cartão de Crédito</td><td>R\$ 21,50</td><td>Não</td></tr></table>	Código	Pagamento	Total	Pago	1	Cartão de Crédito	R\$ 22,00	Não	2	Dinheiro	R\$ 25,50	Não	3	Cartão de Crédito	R\$ 21,50	Não	<div>Visualizar</div> <div>Imprimir</div> <div>Finalizar</div> <div>Cancelar</div>
Código	Pagamento	Total	Pago																	
1	Cartão de Crédito	R\$ 22,00	Não																	
2	Dinheiro	R\$ 25,50	Não																	
3	Cartão de Crédito	R\$ 21,50	Não																	

Copyright © 2019. Created by Gabriel Cechet.

Fonte: Autor

Figura 19 – Delivery Routes - Lista de pedidos em aberto



Código	Endereço	Pagamento	Total	Pago	Status	Distância	Prioridade
12	Av. Quinze de Novembro, 30 - Centro, Erechim - RS, 99700-000, Brazil	Cartão de Crédito	R\$ 22,00	Não	CONFIRMED	0.3 km	1
10	Av. Maurício Cardoso, 17 - Centro, Erechim - RS, 99700-426, Brazil	Dinheiro	R\$ 25,50	Não	CONFIRMED	0.4 km	2
11	Av. Presidente Vargas, 129 - Centro, Erechim - RS, 99700-000, Brazil	Cartão de Crédito	R\$ 21,50	Não	CONFIRMED	0.5 km	3

Copyright © 2019. Created by Gabriel Cechet.

Fonte: Autor

Na Figura 19 é apresentada a tela de “Pedidos Em Aberto”, onde os dados são apresentados quando, via requisição *GET* na rota “/orders/opened” na API do software principal do estabelecimento, executada pela rota /orders. Como resposta, o servidor informa um objeto JSON com a lista dos pedidos com status *PLACED*. No Fragmento de Código 4 é apresentada a função que integra os pedidos no Delivery Routes.

Fragmento de Código 4 – Delivery Routes - Função integradora de pedidos

```

1 public function index() {
2     $req = Request();
3     $data = json_decode(file_get_contents('http://127.0.0.1:8000/
        orders/opened'));
4     foreach ($data as $key => $item) {
5         Order::create([
6             'id' => $item['id'],
7             'latitude' => $item['latitude'],
8             'longitude' => $item['longitude'],
9             'payment_id' => $item['payment_id'],
10            'totalPrice' => $item['totalPrice'],
11            'status' => 'PLACED']);
12        $req->session()->flash('mensagem-sucesso', 'Pedidos
            atualizados. ');
13    }
14
15    $this->changePriority();
16    $orders = Order::all()->where('status', '=', 'CONFIRMED')->
        sortBy('priority');
17    return view('orders.index', ['orders'=> $orders]);
18 }

```

Essa função executa, também, um método denominado *changePriority()* que, por meio das APIs *Distance Matrix* e *Geocoder do Google Maps*, preenchem os campos *distance* e *formatted_address* dos pedidos (Fragmento de Código 5). Além disso, é feita a alteração do status dos pedidos integrados para *CONFIRMED*.

A partir do campo *distance*, preenchido anteriormente, é agregada à prioridade sugerida pela aplicação. A regra aplicada nessa etapa consiste em analisar qual pedido se encontra mais próximo à posição do usuário no momento da requisição à *Distance Matrix* API.

Com todos os dados preenchidos (endereço, distância e prioridade), é possível analisar e gerenciar as informações sugeridas para cada pedido.

Fragmento de Código 5 – Delivery Routes - Função de troca de prioridade dos pedidos

```

1  private function changePriority() {
2      $orders = Order::all()->where('status', '=', 'PLACED');
3      foreach ($orders as $item) {
4          $coordinates = GoogleMaps::geocodeAddress($item->latitude,
5              $item->longitude);
6          $distance = GoogleMaps::distanceMatrix($item->latitude,
7              $item->longitude);
8          $item->update(['formatted_address' => $coordinates['
9              formatted_address'], 'status' => 'CONFIRMED', 'distance'
10             => $distance['distance']]);
11      }
12
13     $orders = Order::all()->where('status', '=', 'CONFIRMED')->
14         sortBy('distance');
15     $priority = 1;
16     foreach ($orders as $item) {
17         $item->update(['priority' => $priority]);
18         $priority++;
19     }
20 }

```

4.3 Coleta de dados

O modelo solicitado para integração via requisição *GET* na rota “/orders/opened” na API do software principal do estabelecimento é exemplificado na Figura 20. O retorno da consulta deve ser em formato JSON, disposto no formato “chave”: “valor” e contendo os seguintes atributos obrigatórios: *id*, *latitude*, *longitude*, *payment_id*, *totalPrice*, *prepaid* e *status*.

Figura 20 – Delivery Routes - API - Coleta de dados

JSON	Raw Data	Headers
Save	Copy	Collapse All Expand All Filter JSON
▼ data:		
▼ 0:		
id:	4	
latitude:	"-27.6343546"	
longitude:	"-52.273859"	
payment_id:	2	
totalPrice:	"25.50"	
prepaid:	0	
status:	"PLACED"	

Fonte: Autor

4.4 Consulta de dados

No momento em que a tela retratada na Figura 21 inicia, um serviço de consulta para capturar informações do pedido requerido é executado. Esse serviço realiza uma requisição *GET* na rota */order/{id}*, parametrizada para receber o código do pedido no estabelecimento, integrado pelo sistema anteriormente.

Nesse momento os possíveis status do pedido poderem ser: *INTEGRATED*, *CANCELLED*, *DISPATCHED*, *DELIVERED* ou *CONCLUDED*.

Figura 21 – Delivery Routes - API - order

JSON	Raw Data	Headers
Save	Copy	Collapse All
	Expand All	Filter JSON
▼ data:		
id:	9	
▼ formatted_address:	"Av. Quinze de Novembro, 30 - Centro, Erechim - RS, 99700-000, Brazil"	
distance:	"0.3 km"	
latitude:	"-27.6351821"	
longitude:	"-52.2732907"	
payment_id:	1	
totalPrice:	"22.00"	
prepaid:	0	
priority:	1	
status:	"INTEGRATED"	
created_at:	"2019-10-17 19:36:15"	
updated_at:	"2019-10-17 19:38:04"	

Fonte: Autor

Como resultado dessa consulta (como demonstra o Fragmento de Código 6), o servidor retorna o objeto requerido (em formato JSON) com os respectivos dados solicitados pela rota, por meio da ferramenta *Resources*, exercendo a sua função como um camada de tratamento entre o Eloquent ORM e as respostas JSON que são expostas pela API. Essas classes, criadas ao executar o comando *php artisan make:resource OrdersResource* através do Artisan, permitem transformar facilmente *models* e *collections* em JSON.

Fragmento de Código 6 – Delivery Routes - Route order

```

1 Route::get('/order/{id}', function ($order_id) {
2     return new OrdersResource(Order::find($order_id));
3 });

```

4.5 Módulo de entrega

Neste módulo é apresentada a usabilidade do sistema por parte do motoboy, realizando a visualização da rota de entrega denominada para o momento.

4.5.1 Tracking de entregas

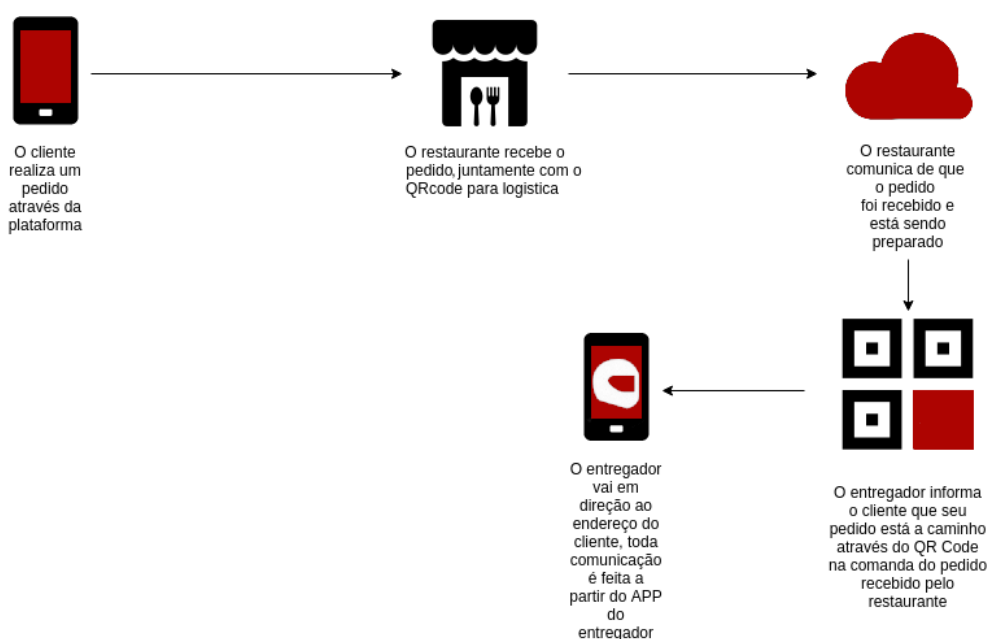
O primeiro status de uma entrega obrigatoriamente será *CONFIRMED*. Isso indica que a mesma foi planejada, visualizada e gerada pelo módulo de gerenciamento. Posteriormente, a comanda (Figura 22) é impressa e então o fluxo de *tracking* do pedido é iniciado (Figura 23).

Figura 22 – Delivery Routes - Comanda do pedido



Fonte: Autor

Figura 23 – Delivery Routes - Fluxo do pedido



Fonte: Adaptada de iFood (2019)

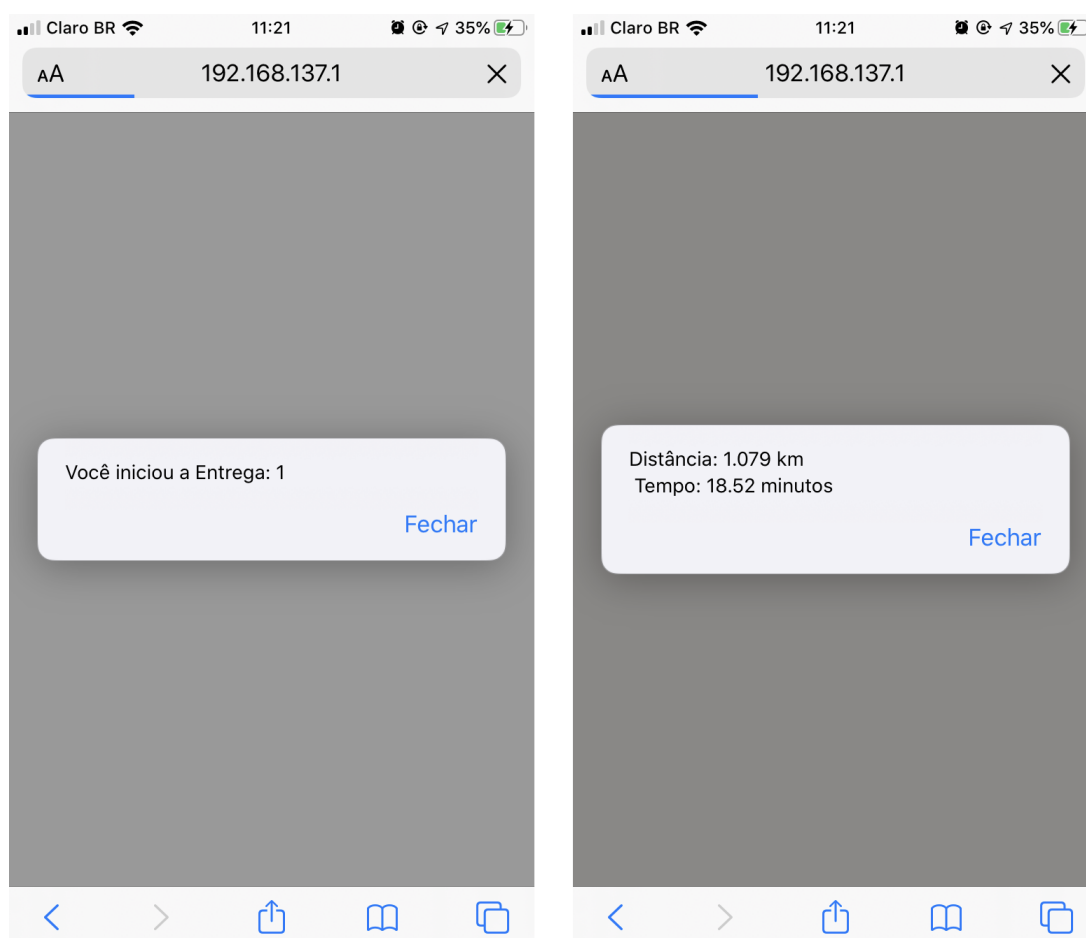
Neste cenário, ao ler o código QR impresso na comanda, o motoboy é direcionado para a rota `/deliveries/id/dispatched` (Fragmento de Código 7), alterando seu status para *DISPATCHED* e executando uma nova rota `/deliveries/id/view`, onde o parâmetro solicitado para ambas é o código da entrega.

Fragmento de Código 7 – Delivery Routes - Função de despacho da entrega

```
1 public function dispatched($id) {
2     $delivery = Delivery::find($id);
3     $delivery->update(['status' => 'DISPATCHED']);
4     return view('deliveries.view', compact('delivery'));
5 }
```

Logo após o despacho da entrega, é apresentada na tela no *smartphone* do motoboy a representação, distância e tempo de deslocamento do percurso que deve ser percorrido na entrega desejada (Figura 24 e Figura 25).

Figura 24 – Delivery Routes - Despacho da entrega



Fonte: Autor

Figura 25 – Delivery Routes - Mapa da entrega



Fonte: Autor

Para obter a rota entre dois pontos e exibí-la no mapa, é necessário utilizar o serviço da *Google Directions API*. Lendo a sua documentação, verificou-se necessário a implementação das classes *google.maps.DirectionsService* e o *google.maps.DirectionsRenderer*.

Começando pelo *DirectionsService*, o seu funcionamento é bem simples. Basta informar um objeto *google.maps.DirectionsRequest*, o qual irá conter o ponto de origem e destino (Fragmento de Código 8), e o meio de transporte (carro, a pé, bicicleta ou transporte público), que ele irá retornar um objeto *google.maps.DirectionsResult*, o qual contém as informações da rota, e o *google.maps.DirectionsStatus*, que por sua vez define o estado final da requisição. Ele pode indicar sucesso (*OK*), sem resultados (*ZERO_RESULTS*), erro (*INVALID_REQUEST* ou *REQUEST_DENIED*), etc.

Fragmento de Código 8 – Delivery Routes - Função de localização do usuário

```
1 navigator.geolocation.getCurrentPosition(position);
2 function success(position) {
3     currentPosition = new google.maps.LatLng(
4         position.coords.latitude, position.coords.longitude);
5 }
```

Para adicionar os pontos de parada durante a rota, adquiridos através da rota */delivering/id* (Fragmento de Código 9), que retorna todos os pedidos da entrega solicitada, é preciso informar uma lista de objetos *google.maps.DirectionsWaypoint* (Fragmento de Código 10), que são alguns pontos pré-definidos no meio do trajeto no objeto *DirectionsRequest* antes de passá-lo para o *directionsService.route*.

Fragmento de Código 9 – Delivery Routes - Route pedidos da entrega

```
1 Route::get('/delivering/{id}', function ($delivery_id) {
2     return new OrdersResource(DB::table('orders')
3         ->leftJoin('deliveries_orders', 'orders.id', '=',
4             'deliveries_orders.order_id')
5         ->where('deliveries_orders.delivery_id', $delivery_id)->get());
6 });
```

Fragmento de Código 10 – Delivery Routes - Preenchimento dos pontos de parada

```
1 $.getJSON(json, function(pontos) {
2     $.each(pontos.data, function(index, ponto) {
3         waypoints[position] = {
4             'location': new google.maps.LatLng(ponto.latitude,
5                 ponto.longitude)
6         };
7     });
8 });
```


Já o *DirectionsRenderer*, basicamente, fica responsável por renderizar o resultado fornecido pelo *DirectionsService* (Fragmento de Código 11).

Fragmento de Código 11 – Delivery Routes - Requisição de renderização do mapa

```

1  var request = {
2      origin: currentPosition,
3      waypoints: waypoints,
4      destination: currentPosition,
5      travelMode: google.maps.TravelMode.DRIVING
6  };
7
8  directionsService.route(request, function(result, status) {
9      if (status == google.maps.DirectionsStatus.OK) {
10         directionsDisplay.setDirections(result);
11     }
12 });

```

O cálculo de distância e tempo (Fragmento de Código 12), exibidos na Figura 24, é a última etapa. Nele é computado o tempo de deslocamento entre os pontos e também o tempo dos trâmites de entrega do produto para o cliente final, estimado em 5 minutos cada.

Fragmento de Código 12 – Delivery Routes - Função de cálculo do deslocamento

```

1  function computeTotalDistance(result) {
2      var totalDist = 0;
3      var totalTime = -300; // retorno
4      var myroute = result.routes[0];
5      for (i = 0; i < myroute.legs.length; i++) {
6          totalDist += myroute.legs[i].distance.value;
7          totalTime += myroute.legs[i].duration.value;
8          totalTime += 300;
9      }
10 }

```

A maioria dos produtos da Google são pagos, porém há um crédito (200 dólares para uso mensal gratuito, que condizem com até 28 mil carregamentos no valor de 7 dólares por milhar) disponível no momento da requisição dos serviços, que vai sendo debitado na medida que é requisitado. Não é cobrado valor algum até que esse crédito seja totalmente utilizado. Ele pode ser usado no *Maps*, no *Routes* ou no *Places*. Para fins de uso e cobrança, é necessário uma chave da API JavaScript do *Google Maps*. A chave da API é um identificador exclusivo usado para autenticar solicitações associadas ao projeto.

4.6 Controller

Esse domínio do MVC é responsável por direcionar os *requests* conforme a regra de negócio. De acordo com a arquitetura do projeto, ele também é responsável por autenticar o usuário e garantir que as rotas do sistema estão sendo acessadas por alguém autenticado.

No caso dos controladores, foi criado um específico para cada classe, onde ele será responsável por direcionar os pedidos conforme a especificidade. Dessa forma, podemos utilizar como exemplo o caso da classe *Motoboy*, com o controlador representado no Fragmento de Código 13.

Fragmento de Código 13 – Delivery Routes - Exemplo de um Controller: Motoboy

```

1  <?php
2  class MotoboysController extends Controller {
3      public function index() {
4          $item = Motoboy::all();
5          return view('motoboys.index', ['motoboys' => $item]);
6      }
7      public function create() {
8          return view('motoboys.create');
9      }
10     public function store(MotoboysRequest $request) {
11         Motoboy::create($request->all());
12         Request()->session()->flash('mensagem-sucesso', 'Registro
13             salvo com sucesso. ');
14         return redirect()->route('motoboys');
15     }
16     public function destroy($id) {
17         Motoboy::find($id)->delete();
18         Request()->session()->flash('mensagem-sucesso', 'Registro
19             excluso com sucesso. ');
20         return redirect()->route('motoboys');
21     }
22     public function edit($id) {
23         $item = Motoboy::find($id);
24         return view('motoboys.edit', compact('item'));
25     }
26     public function update(MotoboysRequest $request, $id) {
27         Motoboy::find($id)->update($request->all());
28         Request()->session()->flash('mensagem-sucesso', 'Registro
29             salvo com sucesso. ');
30         return redirect()->route('motoboys');
31     }
32 }

```

Tabela 3 – Delivery Routes - Exemplo de router do back-end

Método	URI	Ação
GET	/	Exibe a dashboard do aplicativo.
GET	register	Exibe o formulário de inscrição do aplicativo.
POST	register	Manipula uma solicitação de registro para o aplicativo.
GET	login	Exibe o formulário de login do aplicativo.
POST	login	Manipula uma solicitação de login para o aplicativo.
POST	password/email	Envia um link de redefinição para o usuário especificado.
GET	password/reset	Exibe o formulário para solicitar um link de redefinição de senha.
GET	password/reset/{token}	Exibe a visualização de redefinição de senha para o token fornecido.
POST	password/reset	Redefine a senha do usuário fornecido.
POST	logout	Desconecta o usuário do aplicativo.
GET	motoboy	Exibe a listagem de motoboys cadastrados.
GET	motoboy/create	Exibe o formulário para cadastrar um novo motoboy.
POST	motoboy/store	Manipula uma solicitação de cadastro de um motoboy.
GET	motoboy/{id}/edit	Manipula uma solicitação de edição de um motoboy.
PUT	motoboy/{id}/update	Manipula uma solicitação de atualização de um motoboy.
GET	motoboy/{id}/destroy	Manipula uma solicitação de exclusão de um motoboy.
GET	payments	Exibe a listagem de pagamentos cadastrados.
GET	payments/create	Exibe o formulário para cadastrar um novo pagamento.
POST	payments/store	Manipula uma solicitação de cadastro de um pagamento.
GET	payments/{id}/edit	Manipula uma solicitação de edição de um pagamento.
PUT	payments/{id}/update	Manipula uma solicitação de atualização de um pagamento.
GET	payments/{id}/destroy	Manipula uma solicitação de exclusão de um pagamento.
GET	orders	Exibe a listagem de pedidos disponíveis para integração.
GET	order/{id}	Retorna os atributos do pedido desejado em formato JSON.
GET	deliveries	Exibe a listagem de entregas cadastradas.
GET	deliveries/store	Manipula uma solicitação de cadastro de uma entrega.
GET	deliveries/{id}/ticket	Manipula uma solicitação de impressão do ticket de uma entrega.
GET	delivering/{id}	Retorna os atributos dos pedidos da entrega desejada em formato JSON.
GET	deliveries/{id}/view	Manipula uma solicitação de início de uma entrega.
GET	deliveries/{id}/dispatched	Manipula uma solicitação de despacho de uma entrega.
GET	deliveries/{id}/concluded	Manipula uma solicitação de conclusão de uma entrega.
GET	deliveries/{id}/cancelled	Manipula uma solicitação de cancelamento de uma entrega.

Além de implementar os controladores de todas as classes, foi necessário estabelecer quais seriam as rotas e quais controles serão chamados de acordo com a rota. Para isso, o *framework* Laravel disponibiliza um *router*, responsável por organizar as rotas e direcioná-las de acordo com o controlador (Tabela 3). Nesse mesmo *router*, também direciona as rotas de autenticação caso o usuário não esteja logado.

4.7 Integração da ferramenta

As APIs simplificam a forma como os desenvolvedores integram novos componentes de aplicações a uma arquitetura preexistente. Por isso, elas ajudam na colaboração entre as empresas e as equipes de desenvolvimento. Muitas vezes, as necessidades empresariais mudam rapidamente para responder aos mercados digitais em transformação. Nesse ambiente, a ferramenta desenvolvida pode ser integrada em qualquer nicho de mercado que possua a opção de entrega de seus produtos sob gerenciamento de um sistema já existente e aberto à novas integrações.

Em contato com a equipe de desenvolvimento da plataforma *Kitchen* (iFood), para um momento curto, eles não demonstraram interesse em tentar realizar a integração com o Delivery Routes.

5 CONCLUSÃO E TRABALHOS FUTUROS

O desenvolvimento do trabalho possibilitou explorar um grande acervo tecnológico, a partir da interação por meio de rotas com a *Google Maps*, a qual automaticamente fornece sugestões de percursos baseados no fluxo do trânsito do momento. Foi possível perceber também que, por muitas vezes, é vantagem optar por essa tecnologia, pois ela abrange 99% de cobertura no mundo e contempla uma documentação de fácil acesso para realizar a integração de aplicações web com as APIs de geolocalização. Por esse motivo, optou-se pelo desenvolvimento da aplicação em plataforma web, além do fato de existirem *frameworks* como Laravel e Bootstrap que aceleram esse processo.

No transcorrer da elaboração, o tópico mais desafiador foi o cálculo de direções entre dois ou mais locais, por meio do *Directions Service*, um serviço cedido pelo *Google Maps JavaScript API*. Esse mecanismo recebe solicitações de direção e retorna um caminho eficiente. Nesse sentido, por utilizar *waypoints* e *current position*, sua etapa de desenvolvimento foi a mais complexa.

De acordo com a análise realizada durante o estudo dos conceitos utilizados, observou-se que as grandes empresas de comércio online pecam no pós-venda, uma vez que atrasam a entrega de seus produtos e instigam a insatisfação de seus clientes. Partindo dessa questão, foi possível criar um meio de melhorar essa logística e de recomendar aos usuários um planejamento.

Em meio ao cenário de enorme crescimento de pedidos online e *delivery* de produtos, a aplicação Delivery Routes, desenvolvida no presente trabalho, é uma ferramenta com grande possibilidade de aceitação no mercado por diversos fatores, entre os quais destaca-se a oferta de um serviço integrado aos *deliveries* já instalados e operantes nos estabelecimentos, de maneira organizada e documentada, o que possibilita maior praticidade para a *software house* realizar a integração entre os serviços.

Os trabalhos futuros terão como foco o aprimoramento das funcionalidades da aplicação, criando novas telas padrões (fale conosco, ajuda, sobre, configurações, entre outras) que compõem um aplicativo. Além disso, realizar a implementação de um algoritmo, a partir de um grafo, que mostre ao usuário como encontrar o menor caminho entre os pontos de origem e destino, incluindo os pontos de parada, é um dos principais processos planejados para o futuro desta implementação, permitindo ao usuário avaliar o possível percurso e tomar uma decisão.

Para o módulo de entregas, por sua vez, será implementada uma nova *feature*, que permitirá ao motoboy interagir com o método de navegação por voz durante o percurso, com suporte nos principais sistemas operacionais do mercado para não haver limitações. Serão trabalhadas melhorias de fluxos de processos, a fim de evitar possíveis problemas de desempenho. Do mesmo modo, será necessário alocar a aplicação em servidores externos para melhorar a capacidade de processamento com os cálculos de rotas.

Pretende-se, por fim, criar um modelo de autenticação baseado no padrão *OAuth2* com *client_id* e *client_secret*, para identificação na API por meio de credenciais (*token*). Nela, toda comunicação deverá ser realizada sob HTTPS. Um ambiente de homologação será oferecido para auxiliar no desenvolvimento.

Módulos futuros da aplicação contemplarão também a aproximação do cliente final com o estabelecimento. Neste módulo, os clientes terão a facilidade de realizar pagamentos direto pelo aplicativo e informar o recebimento do produto. Com essa funcionalidade presente, adicionalmente será possível calcular precisamente o custo de cada deslocamento.

REFERÊNCIAS

- BENTO, E. **Desenvolvimento Web Com Php E Mysql**. [S.l.]: CASA DO CÓDIGO, 2014. ISBN 9788566250305. Citado 2 vezes nas páginas 14 e 18.
- BOOTSTRAP. **Bootstrap - The most popular HTML, CSS, and JS library in the world**. 2018. Disponível em: <<https://getbootstrap.com/>>. Acesso em: 19 de outubro de 2019. Citado na página 25.
- CÔRTE, L. Método para a avaliação de servidores www no ambiente corporativo. Universidade Federal do Rio Grande do Sul (UFRGS), 2002. Citado na página 13.
- DAS, R.; SAIKIA, D. P. Comparison of procedural php with codeigniter and laravel framework. **International Journal of Current Trends in Engineering Research (IJCTER)**, 2016. Citado 2 vezes nas páginas 15 e 16.
- DIAS, C. H. G.; AMORIM, W. A.; JUNIOR, N. R. de C. Desenvolvimento de aplicações web e dispositivos móveis utilizando frameworks. 2014. Citado na página 1.
- FEUP. **CGI - Common Gateway Interface**. 2019. Disponível em: <<https://web.fe.up.pt/~goii2000/M9/cgi.htm>>. Acesso em: 28 de setembro de 2019. Citado na página 13.
- FIELDING, R. T. Architectural styles and the design of network-based software architectures. UNIVERSITY OF CALIFORNIA, IRVINE, 2000. Citado na página 3.
- GEOSPATIAL. Mashup mania with google maps. 2009. Citado na página 10.
- GOOGLE. **Google Maps**. 2019. Disponível em: <<https://www.google.com.br/maps>>. Acesso em: 06 de agosto de 2019. Citado na página 9.
- GOOGLE. **Google Maps Platform**. 2019. Disponível em: <<https://cloud.google.com/maps-platform/>>. Acesso em: 09 de junho de 2019. Citado 5 vezes nas páginas 8, 9, 10, 11 e 12.
- GSTI, P. **O que é Laravel?** 2019. Disponível em: <<https://www.portalgsti.com.br/laravel/>>. Acesso em: 01 de setembro de 2019. Citado na página 15.
- HAT, R. **O que é uma API?** 2019. Disponível em: <<https://www.redhat.com/pt-br/topics/api/what-are-application-programming-interfaces>>. Acesso em: 07 de setembro de 2019. Citado na página 2.
- HE, R. Y. Design and implementation of web based on laravel framework. In: **2014 International Conference on Computer Science and Electronic Technology (ICCSET 2014)**. [S.l.]: Atlantis Press, 2015. Citado na página 16.
- IFOOD. **Kitchen**. 2019. Disponível em: <<https://developer.ifood.com.br/>>. Acesso em: 18 de outubro de 2019. Citado 3 vezes nas páginas 22, 23 e 32.
- LABORDE, G. **Encontrar a sua posição com Geolocalização**. 2011. Disponível em: <<https://www.oficinadanet.com.br/artigo/desenvolvimento/encontrar-a-sua-posicao-com-geolocalizacao>>. Acesso em: 09 de junho de 2019. Citado 2 vezes nas páginas 4 e 5.

LARAVEL. **Laravel**. 2019. Disponível em: <<https://laravel.com/>>. Acesso em: 01 de setembro de 2019. Citado na página 15.

LOCAWEB. **Frameworks PHP: conheça o Laravel**. 2019. Disponível em: <<http://blog.locaweb.com.br/artigos/frameworks-php-conheca-o-laravel/>>. Acesso em: 01 de setembro de 2019. Citado na página 15.

MARIADB. **About MariaDB**. 2019. Disponível em: <<https://mariadb.org/>>. Acesso em: 02 de outubro de 2019. Citado na página 18.

MCCOOL, S. **Laravel Starter**. [S.l.]: Packt Pub., 2012. Citado na página 17.

OFICINA, R. **O que é A-GPS?** 2018. Disponível em: <https://www.oficinadanet.com.br/artigo/1185/o_que_e_a-gps>. Acesso em: 09 de junho de 2019. Citado 4 vezes nas páginas 5, 6, 7 e 8.

PHP DEVELOPMENT, G. **O que é o PHP?** 2019. Disponível em: <www.php.net>. Acesso em: 27 de setembro de 2019. Citado na página 14.

PIRES, C. E. S.; NASCIMENTO, R. O.; SALGADO, A. C. Comparativo de desempenho entre bancos de dados de código aberto. 2008. Citado na página 18.

POLEZEL ENIUCE MENEZES DE SOUZA, J. F. G. M. W. G. C. Análise dos fatores que influenciam o multicaminho. COBRAC, 2004. Citado 2 vezes nas páginas 6 e 7.

PRESSMAN, R. **Engenharia de Software - 7.ed.** [S.l.]: McGraw Hill Brasil, 2009. Citado na página 2.

PROVOS, N.; MAZIÈRES, D. A future-adaptable password scheme. 1999. Citado na página 24.

RODRIGUES, S. **Motoboy, uma palavra brasileira**. 2017. Disponível em: <<https://veja.abril.com.br/blog/sobre-palavras/motoboy-uma-palavra-brasileira/>>. Acesso em: 01 de maio de 2019. Citado na página 1.

STAUFFER, M. **Desenvolvendo com Laravel: Um framework para a construção de aplicativos PHP modernos**. [S.l.]: NOVATEC, 2017. Citado na página 17.

THOMSON, L.; WELLING, L. **Php E Mysql Desenvolvimento Web**. [S.l.]: Campus, 2005. Citado na página 17.

TULACH, J. **Practical API Design: Confessions of a Java Framework Architect**. [S.l.]: Apress, 2008. ISBN 9781430209744. Citado na página 2.

W3C. **W3C**. 2019. Disponível em: <<http://www.w3c.br/Padroes/>>. Acesso em: 07 de setembro de 2019. Citado na página 3.

WEBPACK. **Webpack**. 2019. Disponível em: <<http://webpack.github.io/>>. Acesso em: 19 de outubro de 2019. Citado na página 25.

Apêndices

APÊNDICE A – Diagrama de Classes

