

**FACULDADE DE TECNOLOGIA DE SÃO JOSÉ DOS CAMPOS
FATEC PROFESSOR JESSEN VIDAL**

JOÃO VITOR FERREIRA GARCIA

SYSRLOG - SISTEMA LOGÍSTICO DE ROTEIRIZAÇÃO

São José dos Campos
2018

JOÃO VITOR FERREIRA GARCIA

SYSRLOG - SISTEMA LOGÍSTICO DE ROTEIRIZAÇÃO

Trabalho de Graduação apresentado à Faculdade de Tecnologia de São José dos Campos, como parte dos requisitos necessários para a obtenção do título de Tecnólogo em Banco de Dados.

Orientador: Me. Lucas Gonçalves Nadalete

São José dos Campos
2018

Dados Internacionais de Catalogação-na-Publicação (CIP)
Divisão de Informação e Documentação

GARCIA, João Vitor Ferreira
SysRLog - Sistema Logístico de Roteirização
São José dos Campos, 2018.
123f.

Trabalho de Graduação – Curso de Tecnologia em Banco de Dados.
FATEC de São José dos Campos: Professor Jessen Vidal, 2018.
Orientador: Prof. Lucas Gonçalves Nadalete.

1. Roteirização. 2.Logística. 3. Software. I. Faculdade de Tecnologia. FATEC de São José dos Campos: Professor Jessen Vidal. Divisão de Informação e Documentação.

REFERÊNCIA BIBLIOGRÁFICA

GARCIA, João Vitor Ferreira. **SysRLog - Sistema Logístico Roteirização.** 2018. 123f.
Trabalho de Graduação - FATEC de São José dos Campos: Professor Jessen Vidal.

CESSÃO DE DIREITOS

NOME(S) DO(S) AUTOR(ES): João Vitor Ferreira Garcia
TÍTULO DO TRABALHO: SysRLog - Sistema Logístico de Roteirização
TIPO DO TRABALHO/ANO: Trabalho de Graduação/2018.

É concedida à FATEC de São José dos Campos: Professor Jessen Vidal permissão para reproduzir cópias deste Trabalho e para emprestar ou vender cópias somente para propósitos acadêmicos e científicos. O autor reserva outros direitos de publicação e nenhuma parte deste Trabalho pode ser reproduzida sem a autorização do autor.

João Vitor Ferreira Garcia
Rua do Porto 718, Caçapava – São Paulo

JOÃO VITOR FERREIRA GARCIA

SISTEMA LOGÍSTICO DE ROTEIRIZAÇÃO

Trabalho de Graduação apresentado à
Faculdade de Tecnologia de São José dos
Campos, como parte dos requisitos
necessários para a obtenção do título de
Tecnólogo em Banco de Dados

Me. Lucas Gonçalves Nadalete - Fatec Prof. Jessen Vidal de São José dos Campos/SP

Me. Emanuel Mineda Carneiro - Fatec Prof. Jessen Vidal de São José dos Campos/SP

**Me. Carlos Augusto Lombardi Garcia - Fatec Prof. Jessen Vidal de São José dos
Campos/SP**

____ / ____ / ____

DATA DA APROVAÇÃO

AGRADECIMENTOS

Agradeço primeiramente a Deus por me manter firme na caminhada da graduação apesar de todas as dificuldades.

Agradeço a todos da minha família que contribuíram direta ou indiretamente nessa jornada. Em especial aos meus pais João José e Carla por toda a motivação, apoio e paciência.

Agradeço minha namorada Ingrid, por toda a motivação e cobrança.

Agradeço a minha segunda Família que são os meus amigos: Rafael, Gean, Victor William, Thales, Jodaci e Paulo. Que sempre estiveram ao meu lado nessa jornada, me ajudando seja com os trabalhos ou seja em conversas. Agradeço em especial ao meu amigo Ticianelli por estar comigo em inúmeras aulas e trabalhos.

Agradeço também ao meu orientador Lucas Nadalete pela proposta de tema e excelente apoio no desenvolvimento do projeto.

Para encerrar deixo meus agradecimentos a todos os professores do curso, que compartilharam seu conhecimento e proporcionaram que eu saísse da Faculdade com um domínio muito maior na área do que eu possuía.

“A primeira regra de qualquer tecnologia utilizada nos negócios é que a automação aplicada a uma operação eficiente aumentará a eficiência. A segunda é que a automação aplicada a uma operação ineficiente aumentará a ineficiência.”

Bill Gates

RESUMO

O presente Projeto, tem como objetivo desenvolver um software para gerar de rotas de entrega. Em pesquisas realizadas, cerca de 54% dos custos logísticos das empresas provém do transporte, além de 92% das empresas darem alta prioridade à redução de custos com transporte. Após a realização de pesquisas exploratórias, os requisitos para o desenvolvimento do Projeto foram definidos no formato de User Stories. O Kanban foi selecionado como metodologia de Desenvolvimento, e o Trello como ferramenta de gestão proporcionando suporte ao KanBan. As tecnologias definidas para a programação do Projeto foram a Linguagem Java, utilizando o Framework Spring e conexão ao Banco de Dados MySQL e o Framework Ionic para o aplicativo, utilizando linguagem TypeScript e Cordova para execução em múltiplas plataformas. O padrão de Projeto foi o MVC(Model, View e Controller) sendo a camada Model composta pelas entidades e Banco de Dados, a Controller pelas Classes com as rotas HTTP e a View representada pelo aplicativo Ionic desenvolvido. O Aplicativo utiliza acesso ao servidor do Via Cep, para consultas de Cep e a API Distance Matrix do Google para o cálculo de distâncias. Com relação a criação de rotas, foi desenvolvida uma implementação do Algoritmo de Dijkstra, que utiliza o servidor Distance Matrix para realizar o cálculo dos pontos e gerar a rota ideal de entrega. Implementação foi baseada nas entidades elencadas nos requisitos. Para segurança da aplicação foi utilizado o Spring Security, com uso de permissões por usuário. Finalizado o Desenvolvimento da aplicação, foram realizados testes de validação para analisar quantitativamente se a rota gerada pelo aplicativo era mais eficiente que a roteirização manual. O percentual de redução, da distância percorrida, utilizando a rota gerada pelo aplicativo frente à roteirização manual foi de 19,48% a 43,40%, e o percentual de tempo foi de 13,33% a 38%. Analisando esses números conclui-se que, o software desenvolvido, beneficia as empresas que trabalham com transporte, por proporcionar uma redução nos custos operacionais e prover uma plataforma para controle das rotas geradas e parametrização de suas regiões de entrega.

Palavras-Chave: Roteirização; Logística; Software; Programação.

ABSTRACT

The purpose of this Project is to develop software to generate delivery routes. In research carried out, about 54% of the companies logistical costs come from transportation, and 92% of the companies give a high priority to reducing transport costs. After conducting exploratory research, the requirements for the development of the Project were defined in User Stories format. KanBan was used as a Development methodology, and Trello were selected as a management tool providing KanBan support. The technologies defined for the Project programming were the Java language, using the Spring Framework and connection to the MySQL Database and the Ionic Framework for the application, using TypeScript language and Cordova for execution in multiple platforms. The Design pattern was MVC (Model, View and Controller) being the Model layer made up of the entities and Database, the Controller for the Classes with the HTTP routes and the View represented by the developed Ionic application. The Application uses Via Cep server access for Cep queries and the Google Distance Matrix API for Distance Calculation. With respect to a route creation, an implementation of the Dijkstra Algorithm was developed, which used the Matrix Distance server to calculate the points and generate an ideal route of delivery. The implementation was based on the entities listed in the requirements. For security of the application was used Spring Security, with use of permissions per user. At the end of the application development, validation tests were performed to quantitatively analyze if the route generated by the application was more efficient than manual routing. The percentage of reduction, of the distance traveled, using the route generated by the application acompared to manual routing was from 19.48% to 43.40% and the percentage of time was from 13.33% to 38%. Analyzing these figures, it is concluded that the software developed benefits the companies that work with transportation, as it provides a reduction in operating costs and provide a platform to control the routes generated and parameterization of their delivery regions.

Keywords: Routing; Logistics; Software; Programming.

LISTA DE FIGURAS

Figura 1 . Era do Transporte Moderno, isolamento das empresas.....	17
Figura 2 . Era da Logística Empresarial.....	17
Figura 3 . Era da Cadeia de Suprimentos.....	18
Figura 4 . Era das redes de Suprimentos.....	18
Figura 5 . Participação do Modal Rodoviário nas Empresas.....	19
Figura 6 . Distribuição dos Custos Logísticos nas Empresas.....	20
Figura 7 . Grau de Priorização das Empresas na Redução de Custos logísticos.....	21
Figura 8 . Fragmento do Quadro e <i>Front-end</i> do Trello.....	34
Figura 9 . Fragmento do Quadro e <i>Back-end</i> do Trello.....	35
Figura 10 . Repositório Github <i>Front-end</i>	44
Figura 11 . Repositório Github <i>Back-end</i>	44
Figura 12 . Arquitetura da Solução - Visão Geral.....	46
Figura 13 . Arquitetura da Solução: <i>Front-end</i>	46
Figura 14 . Arquitetura da Solução: <i>Back-end</i>	48
Figura 15 . Arquitetura da Solução: Implantação de Projeto.....	49
Figura 16 . Diagrama de Componentes.....	50
Figura 17 . Diagrama de Classes: Entidades.....	51
Figura 18 . Diagrama de Classes: Repositórios.....	52
Figura 19 . Diagrama de Classes: Serviços.....	53
Figura 20 . Diagrama de Classes: Controladores e Recursos de Autenticação.....	54
Figura 21 . Definição de Controlador e URL.....	55
Figura 22 . Definição de Acesso ao Método usando <code>@RequestMapping</code>	55
Figura 23 . Definição Injeção de Dependências.....	56
Figura 24 . Definição de Classe de Serviço Utilizando a Anotação <code>@Service</code>	56
Figura 25 . Fragmento da Classe PessoaService.....	56
Figura 26 . Interface PessoaRepository.....	57
Figura 27 . Entidade Pessoa.....	58
Figura 28 . Página de Geração de Rotas.....	59
Figura 29 . Cabeçalho da Página de Geração de Rotas.....	60
Figura 30 . Fragmento da Página de Geração de Rotas.....	60
Figura 31 . Fragmento de Código, Controlador Pagina de Geração de Rotas.....	61
Figura 32 . Fragmento da Classe CepService.....	61
Figura 33 . Tela para Geração e Rotas, com Endereços Inseridos.....	62
Figura 34 . Tela para Geração e Rotas, com Endereços Inseridos.....	62
Figura 35 . Método para Remoção de Endereço da Lista.....	63
Figura 36 . Tela de Geração de Rotas após executada a Geração.....	63
Figura 37 . Tela de Geração de Rotas após executada a Geração.....	64
Figura 38 . Exemplo do Resultado da Aplicação do Algoritmo de Dijkstra.....	65
Figura 39 . Tela de Geração de Rotas após executada a Geração.....	66
Figura 40 . Método geraRota da Classe Gera Rota.....	67
Figura 41 . Busca de Empresa Dentro de GeraRota.....	67
Figura 42 . Definição das Lista a ser Roteirizada e Endereço de Partida.....	68
Figura 43 . Criação de Objetos Auxiliares Necessários à Roteirização.....	68
Figura 44 . <i>Looping</i> de Roteirização.....	69
Figura 45 . Método findMenorDistancia da Classe CalculaDistancia.....	69
Figura 46 . Método calcDistancia da Classe CalculaDistancia.....	70
Figura 47 . Modelo Entidade Relacionamento.....	71
Figura 48 . Propriedade do Liquibase em application.properties.....	78

Figura 49 . Fragmento do Arquivo liquibase-changelog.xml.....	79
Figura 50 . Arquivo 01-create-estado.xml.....	79
Figura 51 . Classe UserPrincipal e Alguns Métodos.....	80
Figura 52 . Método Generate Token de <i>JWTTokenProvider</i>	81
Figura 53 . Classe <i>JWTAuthenticationEntryPoint</i>	81
Figura 54 . Método <i>FilterInternal</i> de <i>JWTAuthenticationFilter</i>	82
Figura 55 . Anotações de parametrização da Classe SecurityConfig.....	82
Figura 56 . Bean para Configuração de Cors.....	83
Figura 57 . Método Configure da Classe <i>SecurityConfig</i>	84
Figura 58 . Diagrama exemplificando Implementação de Segurança.....	84
Figura 59 . Utilização da Anotação <i>@PreAuthorize</i>	85
Figura 60 . Tela de Login.....	86
Figura 61 . Tela de Cadastro.....	86
Figura 62 . Menu Principal.....	87
Figura 63 . Listagem de Rotas Criadas.....	87
Figura 64 . Página de detalhamento da Rota.....	88
Figura 65 . Rota Criada Aberta no Google Maps.....	88
Figura 66 . Página para Gerar a Rota.....	89
Figura 67 . Página Após Geração da Rota.....	89
Figura 68 . Página de Endereço.....	90
Figura 69 . Página Empresa.....	90
Figura 70 . Página de Filiais da Empresa.....	91
Figura 71 . Página de Listagem Funcionários da Empresa.....	91
Figura 72 . Página de Região.....	92
Figura 73 . Página para Alterar a Região.....	93
Figura 74 . Página para Alterar a Região.....	93
Figura 75 . Página de Detalhamento do Usuário.....	94
Figura 76 . Página de Alteração do Usuário.....	95
Figura 77 . Botões para Salvar e Cancelar alteração do Usuário.....	95
Figura 78 . Primeiro Resultado da execução do SonarQube.....	97
Figura 79 . <i>Back-end</i> - Resultado da Execução do SonarQube Após Alterações Efetuadas....	97
Figura 80 . <i>Back-end</i> Outros Resultados.....	98
Figura 81 . <i>Front-end</i> Resultados SonarQube.....	98
Figura 82 . <i>Front-end</i> - Outros Resultados Obtidos com SonarQube.....	99
Figura 83 . Teste Unitários de Serviços e Repositórios de Pessoas.....	100
Figura 84 . Teste Unitários - Utilização da API do ViaCep.....	100
Figura 85 . Teste Unitários - Utilização da API DistanceMatrix.....	101
Figura 86 . Teste Funcionais de API Utilizando PostMan.....	102
Figura 87 . Empresa Cadastrada - Matriz.....	103
Figura 88 . Empresas Cadastradas - Filiais.....	104
Figura 89 . Caso de Testes 1 - Rota Gerada pelo Google Maps.....	105
Figura 90 . Caso de Testes 1 - Rota Gerada pelo SysRLog.....	106
Figura 91 . Caso de Testes 2 - Rota Gerada pelo Google Maps.....	108
Figura 92 . Caso de Testes 2 - Rota Gerada pelo SysRLog.....	108
Figura 93 . Caso de Testes 3 - Rota Gerada pelo Google Maps.....	110
Figura 94 . Caso de Testes 3 - Rota Gerada pelo SysRLog.....	110
Figura 95 . Caso de Testes 4 - Rota Gerada pelo Google Maps.....	112
Figura 96 . Caso de Testes 4 - Rota Gerada pelo SysRLog.....	112
Figura 88 . Caso de Testes 5 - Rota Gerada pelo Google Maps.....	114
Figura 98 . Caso de Testes 5 - Rota Gerada pelo SysRLog.....	114

LISTA DE TABELAS

Tabela 1 . Comparativo de produtividade dos veículos com e sem a roteirização.....	21
Tabela 2 . Atendimentos no Prazo de um Determinado Período Com e Sem a Roteirização..	22
Tabela 3 . Requisitos Funcionais do Projeto.....	26
Tabela 4 . Requisitos Não-Funcionais do Projeto.....	26
Tabela 5 . Lista de Personas com seus comportamentos, necessidades e objetivos.....	27
Tabela 6 . <i>User Story</i> - Otimização de Rota.....	28
Tabela 7 . <i>User Story</i> - Recuperar Rota.....	28
Tabela 8 . <i>User Story</i> - Identificar Entregas Fora da Região de Distribuição da Empresa.....	28
Tabela 9 . <i>User Story</i> - Solicitar Geração de Rotas a partir de Outro Sistema.....	28
Tabela 10 . <i>User Story</i> - Excluir rota gerada.....	29
Tabela 11 . <i>User Story</i> - Cadastrar Usuário.....	29
Tabela 12 . <i>User Story</i> - Alterar Usuário.....	29
Tabela 13 . <i>User Story</i> - Pesquisar Usuários.....	29
Tabela 14 . <i>User Story</i> - Deletar Usuário.....	30
Tabela 15 . <i>User Story</i> - Consultar CEP.....	30
Tabela 16 . <i>User Story</i> - Abrir Rota no Maps.....	30
Tabela 17 . <i>User Story</i> - Consultar CEPs.....	30
Tabela 18 . <i>User Story</i> - Cadastrar Pessoa.....	31
Tabela 19 . <i>User Story</i> - Alterar Pessoa.....	31
Tabela 20 . <i>User Story</i> - Pesquisar Pessoa.....	31
Tabela 21 . <i>User Story</i> - Cadastrar Empresa.....	31
Tabela 22 . <i>User Story</i> - Alterar Empresa.....	32
Tabela 23 . <i>User Story</i> - Pesquisar Empresa.....	32
Tabela 24 . <i>User Story</i> - Cadastrar Funcionário.....	32
Tabela 25 . <i>User Story</i> - Alterar Funcionário.....	32
Tabela 26 . <i>User Story</i> - Deletar Funcionário.....	32
Tabela 27 . <i>User Story</i> - Pesquisar Funcionário.....	33
Tabela 28 . <i>User Story</i> - Pesquisar Região.....	33
Tabela 29 . <i>User Story</i> - Alterar Região.....	33
Tabela 30 . <i>User Story</i> - Deletar Região.....	33
Tabela 31 . <i>User Story</i> - Efetuar Login.....	33
Tabela 32 . <i>User Story</i> - Cadastrar Região.....	34
Tabela 33 . Dicionário de Dados: Tabela Cargo.....	72
Tabela 34 . Dicionário de Dados: Tabela Cep.....	72
Tabela 35 . Dicionário de Dados: Tabela Cidade.....	72
Tabela 36 . Dicionário de Dados: Tabela Empresa.....	73
Tabela 37 . Dicionário de Dados: Tabela Endereço.....	73
Tabela 38 . Dicionário de Dados: Tabela Estado.....	73
Tabela 39 . Dicionário de Dados: Tabela Funcionário.....	74
Tabela 40 . Dicionário de Dados: Tabela Map_config.....	74
Tabela 41 . Dicionário de Dados: Tabela Pessoa.....	74
Tabela 42 . Dicionário de Dados: Tabela Região.....	75
Tabela 43 . Dicionário de Dados: Tabela Roles.....	75
Tabela 44 . Dicionário de Dados: Tabela Telefone.....	75
Tabela 45 . Dicionário de Dados: Tabela Tipo_Empresa.....	76
Tabela 46 . Dicionário de Dados: Tabela Tipo_Pessoa.....	76
Tabela 47 . Dicionário de Dados: Tabela User.....	76
Tabela 48 . Dicionário de Dados: Tabela User_Role.....	77
Tabela 49 . Dicionário de Dados: Tabela Rota.....	77

Tabela 50 . Dicionário de Dados: Tabela Rota_Endereco.....	77
Tabela 51 . Dicionário de Dados: Tabela Responsavel Entrega Cep Rota.....	78
Tabela 52 . Métricas de Qualidade e Resultado Esperado.....	96
Tabela 53 . Tabelas com as Informações das Empresas Cadastradas.....	103
Tabela 54 . Tabelas com a Relação de Ceps Utilizados nos Testes de Roteirização de Cada Empresa.....	104
Tabela 55 . Detalhamento do Caso de Teste 1.....	105
Tabela 56 . Resultados Obtidos no Caso de Teste 1.....	106
Tabela 57 . Detalhamento do Caso de Teste 2.....	107
Tabela 58 . Resultados Obtidos no Caso de Teste 2.....	109
Tabela 59 . Detalhamento do Caso de Teste 3.....	109
Tabela 60 . Resultados Obtidos no Caso de Teste 3.....	111
Tabela 61 . Detalhamento do Caso de Teste 4.....	111
Tabela 62 . Resultados Obtidos no Caso de Teste 4.....	113
Tabela 63 . Detalhamento do Caso de Teste 35.....	113
Tabela 64 . Resultados Obtidos no Caso de Teste 5.....	115
Tabela 65 . Tabela com os Resultados Obtidos no Comparativo de Tempo das Rotas.....	115
Tabela 66 . Tabela com os Resultados Obtidos no Comparativo de Distância das Rotas.....	116

LISTA DE ABREVIATURAS E SIGLAS

CEL	Centro de estudos em Logística
CD	Centro de Distribuição
Copeead	Instituto de Pós-Graduação e Pesquisa em Administração
DTO	<i>Data Transfer Object</i>
FK	<i>Foreign Key</i>
HTML	<i>Hypertext Markup Language</i>
ID	Número identificador
ILOS	Instituto de Logística e <i>Supply Chain</i>
JVM	<i>Java Virtual Machine</i>
JWT	<i>JSONWebToken</i>
MVC	<i>Model View Controller</i>
PK	<i>Primary Key</i>
TI	Tecnologia da Informação

SUMÁRIO

1. INTRODUÇÃO.....	16
1.1. Problema em estudo.....	22
1.2. Relevância do Trabalho.....	22
1.3. Objetivo do Geral.....	22
1.4. Objetivos Específicos.....	23
1.5. Proposta Metodológica.....	23
1.6. Conteúdo do Trabalho.....	24
2. REQUISITOS IDENTIFICADOS E CONTEXTUALIZAÇÃO TECNOLÓGICA....	25
2.1. Especificação de requisitos.....	25
2.2. Requisitos Funcionais.....	25
2.2.1. Requisitos Não-Funcionais:.....	26
2.3. Especificações baseadas em <i>User Stories</i>	26
2.3.1. <i>BackLog</i>	34
2.4. Tecnologias Aplicadas	35
2.4.1. <i>Back-end</i>	36
2.4.1.1. Linguagem Java.....	36
2.4.1.2. Formato para Transmissão de Dados.....	36
2.4.1.3. Maven.....	36
2.4.1.4. Spring.....	37
2.4.1.5. Banco de Dados.....	38
2.4.1.6. Plugins para Base de Dados.....	38
2.4.1.7. Demais Dependências utilizadas.....	38
2.4.1.8. Softwares utilizados.....	39
2.4.1.9. Recursos Externos.....	39
2.4.2. <i>Front-end</i>	39
2.4.2.1. NPM.....	40
2.4.2.2. Ionic.....	40
2.4.2.3. HTML5.....	40
2.4.2.4. CSS.....	41
2.4.2.5. TypeScript.....	41
2.4.2.6. AngularJS.....	41
2.4.2.7. Cordova.....	42
2.4.2.8. Softwares utilizados.....	42
2.4.3. Ferramentas de Teste.....	42
2.4.3.1. JUnit.....	42
2.4.3.2. JaCoCo.....	42
2.4.3.3. SonarQube.....	43
2.4.3.4. PostMan.....	43
2.4.4. Versionamento.....	43
3. DESENVOLVIMENTO.....	45
3.1. Padrão do Projeto.....	45
3.2. Arquitetura da Solução.....	45
3.2.1. Arquitetura da Solução - <i>Front-end</i>	46
3.2.2. Arquitetura da Solução - <i>Back-end</i>	47
3.2.3. Arquitetura da Solução - Implantação.....	48

3.3. Arquitetura do Software.....	49
3.3.1. Diagrama de Componentes.....	49
3.3.2. Diagramas de Classes.....	50
3.3.3. Exemplificação de Funcionamento do <i>Back-end</i>	54
3.3.4. Exemplificação de Funcionamento do <i>Front-end</i>	59
3.4. Algoritmo de Roteirização.....	64
3.5. Modelagem e Gestão dos Dados.....	70
3.5.1. Modelo de Entidade Relacionamento.....	71
3.5.2. Dicionário de Dados.....	71
3.5.3. Liquibase.....	78
3.6. Segurança.....	80
3.6.1. Visão Geral - Segurança.....	84
3.7. Visão geral do Sistema.....	85
4. VALIDAÇÃO E ANÁLISE DOS DOS RESULTADOS OBTIDOS.....	96
4.1. Métricas do Sistema.....	96
4.1.1. Resultados das Métricas <i>Back-end</i>	96
4.1.2. Resultados das Métricas <i>Front-end</i>	98
4.2. Técnicas de Verificação e Validação aplicadas e Resultados.....	99
4.2.1. Testes de Unidade.....	99
4.2.2. Testes de Recursos Externos.....	100
4.2.3. Teste Funcional de API.....	101
4.3. Processo de Validação do Algoritmo de Roteirização e Resultados Obtidos.....	102
4.3.1. Caso de Testes 1 - Cidade de Caçapava.....	104
4.3.2. Caso de Testes 2 - Cidade de São José dos Campos.....	107
4.3.3. Caso de Testes 3 - Cidade de Taubaté.....	109
4.3.4. Caso de Testes 4 - Cidade de Jacareí.....	111
4.3.5. Caso de Testes 5 - Cidade de Caraguatatuba.....	113
4.3.6. Consolidação dos Resultados Obtidos nos casos de Teste.....	115
5. CONCLUSÃO.....	117
5.1. Principais Contribuições.....	117
5.2. Considerações Gerais, Limitações e Dificuldades.....	118
5.3. Sugestão de trabalhos futuros.....	119
REFERÊNCIAS BIBLIOGRÁFICAS.....	121

1. INTRODUÇÃO

No cenário atual do mercado, empresas têm que trabalhar com prazos reduzidos, gerenciar seus estoques, reduzir o valor da produção de seus produtos e economizar com o transporte de suas mercadorias. Devido ao grande número e complexidade de tarefas à serem realizadas pelas empresas, cada vez mais elas buscam terceirizar o transporte de produtos (BRANSKI, 2008), pois com a terceirização surge a oportunidade de redução de custos logísticos (PERÇİN; MIN, 2013).

A logística é uma área vital e de extrema importância para as empresas, pois com o ambiente globalizado em que ocorrem mudanças constantes, as operações logísticas acabam tornando-se cada vez mais complexas, onerosas e importantes sob o ponto de vista estratégico (FLEURY, 1999).

Empresas visam maximizar cada vez mais seu lucro, portanto faz-se necessário atentarem-se as atividades logísticas de planejamento, abastecimento, mão de obra, e entrega final. Pois, planejando adequadamente estes itens, a empresa consegue reduzir custos e consequentemente repassar um menor valor a seus clientes, maximizando seu lucro e tornando-se mais competitiva frente a seus concorrentes (DORNIER et al, 2000).

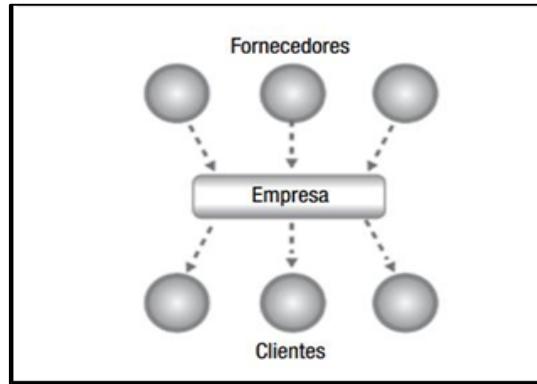
Desde a antiguidade, a logística faz parte das guerras, devido ao deslocamento de tropas, suprimentos e armamentos, por grandes distâncias além do longo período de duração das batalhas. Por conta destes fatores precisava-se de um planejamento, organização e execução de tarefas.

Até os anos 50, os mercados eram muito limitados e funções logísticas estavam dispersas entre os diversos departamentos da organização. A área industrial geria o planejamento e o controle, já a área administrativa comandava o estoque, o departamento de vendas que cuidava dos pedidos e o departamento financeiro que era responsável pelas compras (ROMERO; SOUZA, 2015).

Esse procedimento ocasionava conflitos e descontrole, já que departamentos que não tinham uma base de conhecimento específica para lidar com essas tarefas acabavam tornando-se responsáveis. Dessa forma as corporações acabavam sendo afetadas negativamente, com perda de vantagem competitiva e prejudicando os processos de entrega de valor para o cliente (POZO, 2010 p. 3).

A Figura 1 mostra graficamente o isolamento das empresas na Era do Transporte até 1950.

Figura 1. Era do Transporte Moderno, isolamento das empresas.



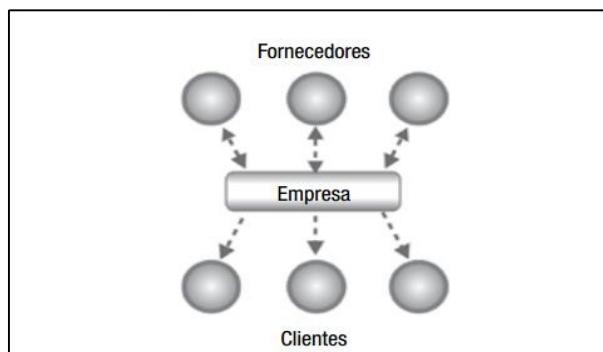
Fonte: Machline (2011)

Entre os anos de 1950 e 1970, a área da logística empresarial, que até esta época era apenas teoria, começou a ser realmente utilizada na prática, com o intuito de melhorar o resultado das empresas como evidenciado por Pozo (2010, p. 5):

“A nova situação econômica do pós-guerra, e principalmente no inicio dos anos 50, era um forte instrumental para fomentar o interesse em Logística. O crescimento econômico substancial que decorreu das novas atitudes e concepções após a Segunda Guerra foi seguido de recessão e um período de prolongada pressão nos ativos das empresas e de seus lucros. Os novos conceitos logísticos, que começavam a aflorar na mente dos administradores, ofereciam a oportunidade de melhorar os resultados das empresas.”

Pozo (2010, p. 7) destaca que com o desenvolvimento da tecnologia, os problemas logísticos tornam-se cada vez mais complexos, exigindo uma visão sistêmica da organização e do mercado devido ao relacionamento bidirecional entre empresa e fornecedores, conforme pode ser visualizada na Figura 2.

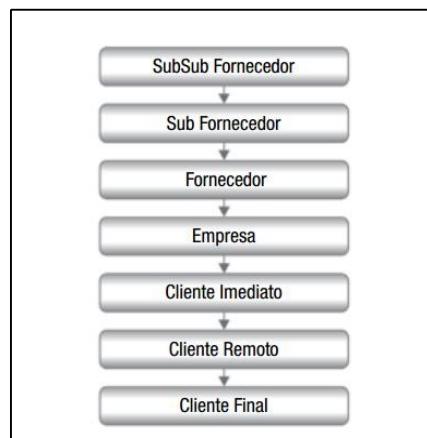
Figura 2. Era da Logística Empresarial.



Fonte: Machline (2011).

Machline (2011) através de uma visão integrada denominou no período entre os anos de 1970 a 2000 como a Era da Cadeia de Suprimentos, conforme é apresentada na Figura 3.

Figura 3. Era da Cadeia de Suprimentos.

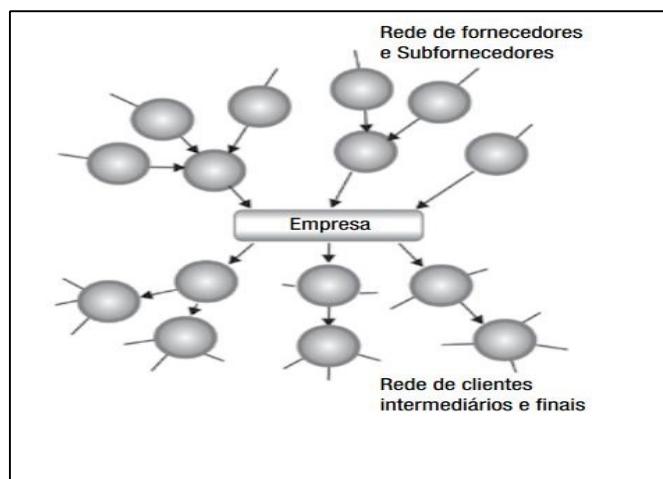


Fonte: Machline (2011).

De modo geral, Francischini e Amaral Gurgel (2002, p. 262) alegam que a cadeia de suprimento é uma “integração dos processos que formam um determinado negócio, desde os fornecedores originais até o usuário final, proporcionando produtos, serviços e informações que agregam valor para o cliente”.

Machline (2011) denominou a Era das Redes de Suprimentos desde o ano 2000 até a atualidade, com uma visão global como monstra na Figura 4.

Figura 4. Era das redes de Suprimentos.



Fonte: Machline (2011).

Chopra e Meindl (2011, p. 154), afirmam que

“devido ao atual cenário globalizado, as cadeias de suprimentos estão sujeitas a mais fatores de risco, do que as cadeias locais do passado. Subestimar os fatores de risco pode acarretar péssimos resultados como aconteceu na crise financeira de 2008, quando devido a grande recessão, o número de exportações e importações caiu drasticamente em diversos países, devido à diminuição do consumo”.

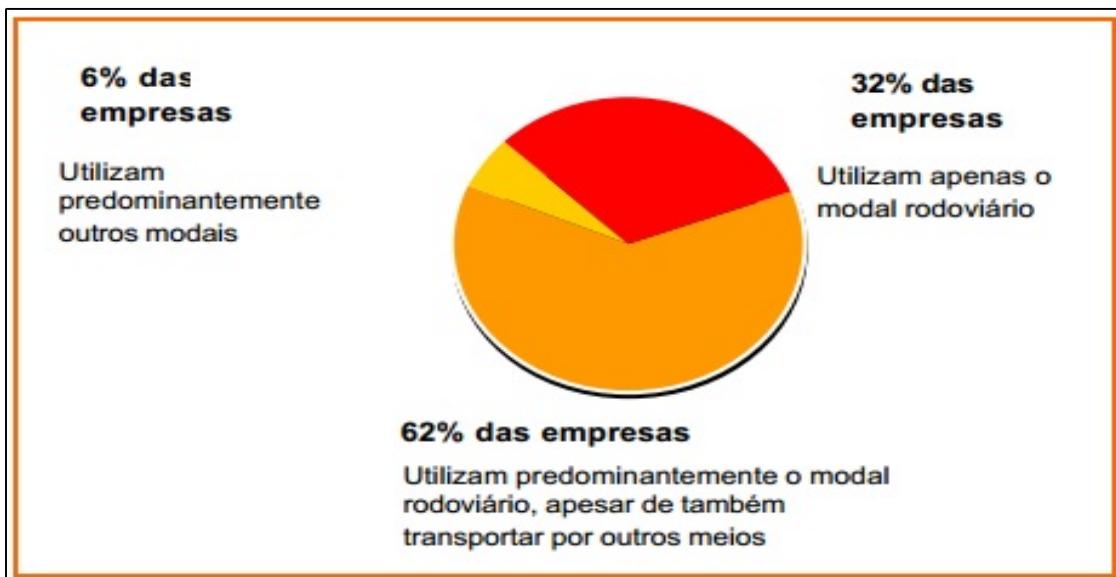
Uma área que conecta os diversos estágios da cadeia de suprimentos, permitindo a coordenação das atividades e contribuindo para a mesma, é a Tecnologia da informação

(TI), pois com softwares para gestão adequados, informações gerenciais e dados estatísticos estão disponíveis em tempo real para serem analisados e compartilhados dentro das empresas, facilitando todo o planejamento logístico (CHOPRA; MEINDL, 2003).

Sem os aplicativos de TI, a troca de informações seria limitada ao papel, gerando um grande descontrole e prejudicando qualquer tipo de operação ou procedimento (NAZÁRIO, 1999). As aplicações de TI na Logística são várias, e englobam tanto os equipamentos como os sistemas de informações. Combinados, o uso destas tecnologias permite o gerenciamento integrado e eficiente dos estoques, armazéns, transportes, processamento de pedidos, compras e manufaturas (FLEURY te al, 2000).

De todos os meios de transporte Logístico, ou modais Logísticos o mais utilizado no Brasil é o Rodoviário (RIBEIRO; FERREIRA, 2002). Em 2006 uma pesquisa do Centro de Estudos em Logística (CEL) / Instituto de Pós-Graduação e Pesquisa em Administração (Copeead) apontou que 88,3% das empresas transportam cargas por rodovia, aproximadamente um terço das empresas entrevistadas usam somente o modal rodoviário e apenas 6% das empresas não utilizam modal rodoviário conforme apresentado na Figura 5:

Figura 5. Participação do Modal Rodoviário nas Empresas.



Fonte: Panorama logístico CEL/COOPEAD – Gestão do transporte rodoviário de carga nas empresas – Práticas e Tendências - (2007).

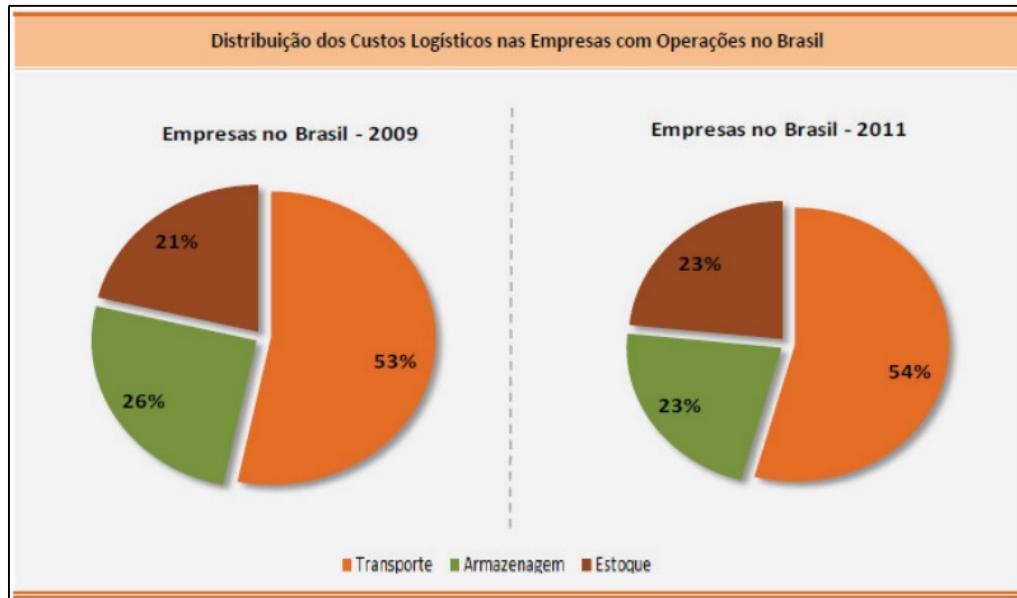
Os pontos fortes do modal rodoviário, tais como disponibilidade para embarques urgentes, rapidez em curtas distâncias, capacidade de atingir grande extensão no território brasileiro, custos fixos baixos, dentre outros, acabam por fazer dele o mais utilizado (DEMARIA; MARJORY, 2004).

Por volta de 2006, iniciou-se no país, um programa de Investimento em Logística que visa criar uma infraestrutura intermodal mais integrada, promover concessões em rodovias a fim de proporcionar uma melhor infraestrutura, para assim aumentar a agilidade e diminuir custos. Cerca de R\$ 198,4 bilhões serão investidos no setor. Segundo Rosa (2007), esses investimentos em infraestrutura se fazem necessários considerando que o mau estado das rodovias provoca uma média de 46% de aumento no custo operacional dos veículos.

Em pesquisa realizada pelo CEL/Coopead cerca de 7,5% da receita líquida das empresas brasileiras é gasta com custos logísticos, englobando gastos com armazenagem, transporte e estoque. Em seu livro, Balou (2006) mostra que o transporte Logístico representa entre um ou dois terços dos custos totais de Logística das empresas.

Em estudo desenvolvido entre 2009 e 2011, pela empresa ILOS (Instituto de Logística e Supply Chain), o percentual de custos com logística derivados do transporte ocorreram na proporção apresentada na Figura 6:

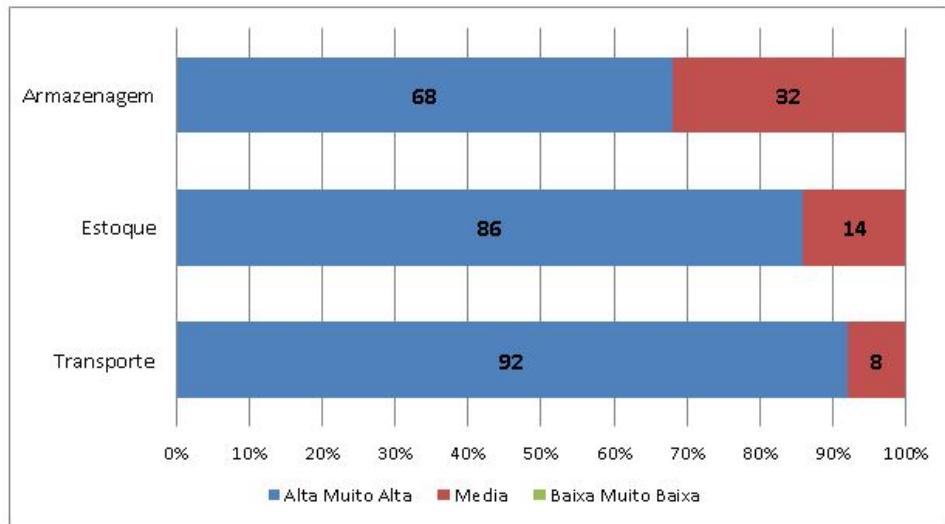
Figura 6. Distribuição dos Custos Logísticos nas Empresas.



Fonte: Panorama Instituto ILOS - Custos Logísticos no Brasil, (2012).

Devido ao grande percentual sobre os custos logísticos, as empresas buscam cada vez mais otimizar suas rotas de transporte. Na pesquisa realizada pelo CEL/Coopead cerca de 92% das empresas responderem que tem alta prioridade em buscar redução de custos com transporte dentro dos custos logísticos, conforme pode ser visualizado na Figura 7:

Figura 7. Grau de Priorização das Empresas na Redução de Custos logísticos.



Fonte: Adaptado de Panorama logístico CEL/COOPED – Gestão do transporte rodoviário de carga nas empresas – Práticas e Tendências, (2007).

Em estudo realizado por Matos Júnior et al. (2013), a roteirização foi capaz de reduzir as devoluções em média 1,57%, o que consequentemente reduz gastos operacionais dos veículos. Aumentou a taxa de ocupação em cerca de 7%, reduzindo o custo por quilo e na entrega conforme apresentado na Tabela 1 .

Tabela 1. Comparativo de produtividade dos veículos com e sem a roteirização.

Id da Rota	Distância	Custo Quilo	Custo por Parada	Custo Total
MYC - 8306	40,8	R\$ 0,14	R\$ 23,88	R\$ 191,00
MYC - 8306	72,7	R\$ 0,16	R\$ 27,38	R\$ 219,00
Redução	43.88%	12,50%	12,78%	12.79%
MYC - 8356	83,4	R\$ 0,15	R\$ 26,63	R\$ 229,00
MYC - 8356	100,3	R\$ 0,16	R\$ 30,50	R\$ 244,00
Redução	16,85%	6,25%	12,69%	6,15%

Fonte: Adaptado Júnior (2013).

No mesmo estudo de Matos Junior et al. (2013), foi evidenciado um aumento médio de 7.32% nas entregas realizadas com sucesso, gerando uma melhoria no desempenho operacional, conforme mostrado na Tabela 2.

Tabela 2. Atendimentos no Prazo de um Determinado Período Com e Sem a Roteirização.

Unidade	Atendimentos no Prazo com roteirização em %				Atendimentos no Prazo sem roteirização em %				Diferença em %			
	JAN	FEV	MAR	ABR	JAN	FEV	MAR	ABR	JAN	FEV	MAR	ABR
EUSÉBIO	92,20%	95,93%	94,36%	98,57%	86,40%	88,60%	87,60%	91,60%	5,80%	7,33%	6,76%	6,97%
BHO	98,65%	98,80%	98,46%	93,83%	91,08%	90,35%	92,41%	85,40%	7,57%	8,45%	6,05%	8,43%
RECIFE	99,21%	96,31%	93,48%	90,84%	92,60%	88,48%	86,60%	83,58%	6,61%	7,83%	6,88%	7,26%
TERESINA	91,47%	95,55%	95,11%	86,67%	78,90%	87,50%	88,64%	79,64%	12,57%	8,05%	6,47%	7,03%
RIO	95,27%	89,35%	86,12%	89,96%	87,25%	83,58%	79,61%	82,50%	8,02%	5,77%	6,51%	7,46%
TIMOM	100,00%	100,00%	100,00%	87,59%	95,00%	93,68%	91,80%	79,50%	5,00%	6,32%	8,40%	8,09%

Fonte: Adaptado de Dados cedidos pela indústria cearense (2013)

Após a apresentação desses dados, percebe-se que uma roteirização bem preparada gera economia para as empresas, melhorando o serviço de transporte entregue aos seus clientes.

1.1. Problema em estudo.

Uma determinada empresa de São José dos Campos (que pediu para não ter seu nome citado neste documento), efetuou a implantação de um Software de gestão logístico para um cliente em Portugal, no entanto essa solução não possuía modulo para criação de rotas. Assim o Cliente português teria que continuar utilizando um dos módulos do antigo Software que era utilizado.

1.2. Relevância do Trabalho.

O presente trabalho será utilizado como prova de conceito para a Empresa de São José dos Campos, visando demonstrar a importância da aplicação de um algoritmo de roteirização para empresas que utilizam-se de transporte logístico. O Projeto também será importante para análise de viabilidade da utilização do Algoritmo de Dijkstra como algoritmo para geração das rotas.

1.3. Objetivo do Geral.

O objetivo geral deste trabalho é desenvolver um Software, dedicado a criação de rotas de entrega utilizando o algoritmo de Dijkstra para otimização dessas rotas.

1.4. Objetivos Específicos.

Este trabalho tem como objetivos específicos:

- Desenvolver um algoritmo para roteirização;
- Desenvolver um software para criação de rotas de entrega baseada em endereços previamente determinados e na área de atuação da empresa;
- Realizar um estudo comparativo entre aplicar uma roteirização manual, por ordem de inserção dos pontos de entrega, e realizar a roteirização utilizando o algoritmo desenvolvido.

1.5. Proposta Metodológica.

Para desenvolver o projeto, primeiramente, foi realizado um trabalho de pesquisa com caráter exploratório, para compreender melhor o setor de logística, seu funcionamento, o quanto ele é vital para as empresas, o custo que ele gera para as empresas, e o interesse que elas têm em economizar com este setor. Realizado essa pesquisa, foi observado que desenvolvimento de um Software para roteirização de entregas poderia auxiliar este setor nas empresas reduzindo seus custos de operação.

Na pesquisa exploratória foi lido diversos artigos sobre roteirização, trechos de livros de autores como: Paulo F. Fleury, Sunil Chopra e Peter Meindl. Autores estes sempre utilizados como referência, em estudos do Setor Logístico. Outro aspecto analisado, foi de artigos e de estudos que comprovam o interesse das empresas em reduzir seus custos com operações logísticas como o Panorama Logístico Desenvolvido pelo CEL/COOPEAD.

No que compreende ao levantamento de requisitos, utilizar-se-á de User Stories, pois seu formato auxilia o codificação, proporcionando de maneira simples e direta a funcionalidade a ser desenvolvida.

Como metodologia de desenvolvimento, utilizar-se-á de Kanban. Essa metodologia pode ser aplicada ao desenvolvimento de Softwares e combina muito bem com as User Stories. O Kanban proporciona uma visão geral do projeto e viabiliza a classificação das tarefas, organizando o andamento do projeto. O Trello foi escolhido como ferramenta de Kanban para o Projeto.

O desenvolvimento realizar-se-á por meio de duas etapas. Na primeira, desenvolver-se-á o back-end do projeto, que compreende toda a camada que disponibiliza os serviços para gestão dos cadastros, criação de rotas e consulta de ceps. A segunda etapa responsabilizar-se-á pelo desenvolvimento do front-end do projeto, que é a Interface Web e também Mobile, na qual o usuário utilizará a aplicação.

Os resultados serão mensurados de maneira quantitativa, analisando o tempo que uma empresa economizará utilizando o aplicativo SysRLog, para definir uma rota de entregas otimizada, em relação a realizar entregas sem nenhum tipo de planejamento ou roteirização prévias.

1.6. Conteúdo do Trabalho

O presente trabalho está estruturado em cinco Capítulos, cujo conteúdo é sucintamente apresentado a seguir:

No Capítulo 2 é apresentada a etapa de engenharia de requisitos, apresentando as User Stories e seu detalhamento. Neste mesmo capítulo, serão apresentadas as tecnologias utilizadas no desenvolvimento do projeto .

O Capítulo 3 comprehende o Desenvolvimento do Trabalho, sendo composto por: Arquitetura da Solução, Modelagem e Gestão dos Dados, Arquitetura do Software, Segurança e para concluir a apresentação de uma Visão Geral do Sistema.

No Capítulo 4 serão apresentadas as experimentações e a análise dos resultados obtidos.

Finalmente, o Capítulo 5 apresenta a conclusão deste trabalho com base nos resultados obtidos com o software desenvolvido.

2. REQUISITOS IDENTIFICADOS E CONTEXTUALIZAÇÃO TECNOLÓGICA

O presente capítulo dedica-se a Engenharia de Requisitos, ramo da Engenharia de Software, que segundo Thayer e Dorfman (2004) é responsável por analisar e documentar requisitos, incluindo análise das necessidades do software e a especificação dos requisitos.

2.1. Especificação de requisitos

Requisitos podem ser definidos como o que o sistema oferece, o que o sistema deve fazer, suas entradas e restrições. Os requisitos são um reflexo da necessidade do cliente para um sistema com função determinada, exemplo: buscar informações, armazenar um dado, controlar um equipamento (SOMMERVILLE, 2011). O processo de descoberta e documentação destes requisitos é denominada Levantamento de Requisitos.

Os requisitos podem ser classificados em dois tipos:

- i. Requisitos Funcionais: Descrevem as funcionalidades do sistema, suas ações, entradas, saídas e exceções;
- ii. Requisitos Não-Funcionais: São requisitos que não estão diretamente ligados com serviços e/ou funcionalidades específicas oferecidas pelo Software. Eles são frequentemente mais críticos como: Definir restrições sobre a implantação do sistema, confiabilidade, tempo de resposta dentro outros aspectos.

2.2. Requisitos Funcionais

Os requisitos funcionais do projetos foram analisados, elencados e categorizados conforme mostrados na Tabela 3.

Tabela 3. Requisitos Funcionais do Projeto.

Requisitos Funcionais		
Requisito	Nível de Priorização	Legenda
Consultar CEP	10	Imprescindível
Gerar Rota a Partir de Lista de CEPs	10	Imprescindível
Apresentar Rota Gerada com o Google Maps	10	Imprescindível
Gerenciar Usuário	10	Imprescindível
Gerenciar Pessoa	8	Importante
Gerenciar Empresa	8	Importante
Gerenciar Região	9	Obrigatório
Gerenciar Filiais	8	Importante
Gerenciar Cargos	4	Desejável
Gerenciar Funcionários	7	Importante
Disponibilizar <i>Web Service</i> de geração de Rotas	8	Importante

Fonte: Autor (2018)

2.2.1. Requisitos Não-Funcionais:

Os requisitos não funcionais elencados serão mostrados na Tabela 4.

Tabela 4. Requisitos Não-Funcionais do Projeto.

Requisitos Não-Funcionais		
Requisito	Nível de Priorização	Legenda
Consultar Cep não Cadastrado externamente	8	Importante
Utilizar autenticação básica para a Aplicação	10	Imprescindível
Desenvolver para Plataforma <i>Web</i> e <i>Android</i>	8	Importante
Desenvolver para plataforma <i>IOS</i>	2	Baixa
Padrão de comunicação <i>back-end - front-end</i> via <i>Http</i>	10	Imprescindível
JSON como formato do arquivo de comunicação	10	Imprescindível
Comunicação constante do Servidor com a Internet	10	Imprescindível

Fonte: Autor (2018)

2.3. Especificações baseadas em *User Stories*

O formato escolhido para a especificação dos requisitos é o de User Stories, por ser uma forma sucinta e direta de apresentar a funcionalidade a ser desenvolvida, os critérios de aceitação e os fluxos de exceções. Esse formato prioriza o processo de desenvolvimento do código e entrega do software.

Para que uma User Story possa ser descrita, primeiramente devem ser elencadas as Personas, ferramenta que utiliza estereótipos de grupos de pessoas fictícias para representar usuários. As Personas envolvidas no projeto estão relacionadas na Tabela 5.

Tabela 5. Lista de Personas com seus comportamentos, necessidades e objetivos.

Personas		
Persona	Comportamentos	Necessidade/Objetivos
Motorista	<ul style="list-style-type: none"> - Não utiliza Sistemas Gerências - Não é familiarizado com computadores - Usa <i>Smartphone</i> 	<ul style="list-style-type: none"> - Que as rotas já estejam otimizadas quando ele sair para realizar as entregas - Ferramenta simples e fácil
Gerente	<ul style="list-style-type: none"> - Usa Sistemas Gerenciais apenas para consulta - Dificilmente realiza algum procedimento operacional - Está interessado em resultados 	<ul style="list-style-type: none"> - Redução de custos com transporte - Aumentar a quantidade de entregas no Prazo - Não precisar investir em um novo Sistema de Gestão
Operador Logístico (Engloba Analistas, técnicos e auxiliares)	<ul style="list-style-type: none"> - Realiza muitos procedimentos operacionais ao longo do dia no Sistema Gerencial. - São familiarizados com computadores e Sistemas - Tem que checar manualmente as entregas para verificar quais não são entregues pela sua respectiva empresa 	<ul style="list-style-type: none"> - Sistema simples e Objetivo para realizar cadastros e alterações - Interface minimalista, com fácil navegação - Identificar automaticamente as entregas à serem transferidas a outros centros de distribuição - Otimizar a rota de entregas automaticamente - Não necessitar refazer os lançamentos que estão no Sistema de gestão
TI	<ul style="list-style-type: none"> - Alto conhecimento sobre sistemas e tecnologias - Preza pela facilidade, sendo contra o retrabalho - É o responsável pelo cadastro de usuários 	<ul style="list-style-type: none"> - Integração entre o sistema de otimização de rotas e o sistema de gestão - Gestão adequada dos usuários

Fonte: Autor (2018)

Após elencadas as Personas envolvidas com o Projeto, as User Stories puderam ser criadas. As Tabelas de 6 a 32 apresentam as User Stories definidas para o desenvolvimento do Projeto.

Tabela 6. User Story - Otimização de Rota.

User Story	Otimização de Rota
Quem ?	Operador Logístico
O que?	Gostaria que a melhor rota de entrega fosse gerada Automaticamente
Por que?	Para reduzir custos operacionais e realizar as entregas mais rapidamente
Critérios de aceitação	Gerar rotas lançando endereços manualmente Gerar rotas a partir de uma lista de endereços já pronta
Fluxo Exceção	Indisponibilidade de calcular distâncias entre os pontos de entrega

Fonte: Autor (2018)

Tabela 7. User Story - Recuperar Rota.

User Story	Recuperar Rota
Quem ?	Motorista
O que?	Quero conseguir encontrar uma rota gerada pelos operadores
Por que?	Para poder visualizar a rota otimizada e realizar as entregas
Critérios de aceitação	Quero visualizar as rotas que foram geradas

Fonte: Autor (2018)

Tabela 8. User Story - Identificar Entregas Fora da Região de Distribuição da Empresa.

User Story	Identificar entregas fora da região de distribuição
Quem ?	Operadores logísticos
O que?	Preciso que seja mostrada as entregas que não correspondem a área de atuação da empresa
Por que?	Para que as entregas sejam devidamente encaminhadas aos seus centros de distribuição respectivos.
Critérios de aceitação	A partir de uma Lista de Entregas identificar as entregas que não são realizadas pela empresa Apresentar a empresa que atende o endereço
Fluxo Exceção	Não haver empresas que atendam esse Endereço de entrega

Fonte: Autor (2018)

Tabela 9. User Story - Solicitar Geração de Rotas a partir de Outro Sistema.

User Story	Solicitar geração de rotas a partir de outro sistema
Quem ?	Operadores logísticos
O que?	Gostaríamos de Gerar a rota a partir do Sistema e Gestão Logística Utilizado
Por que?	Para que não precisemos ficar utilizando outro sistema além do nosso sistema de gestão
Critérios de aceitação	A partir de um outro Sistema que não a Interface do Projeto, solicitar a criação da rota
Fluxo Exceção	Formato de entrada inválido

Fonte: Autor (2018)

Tabela 10. User Story - Excluir rota gerada.

User Story	Excluir rota gerada
Quem ?	Operadores logísticos
O que?	Gostaríamos excluir rotas que foram geradas
Por que?	Para que rotas geradas por engano, ou com algum endereço errado não sejam mais visualizadas
Critérios de aceitação	A partir da lista de rotas geradas, poder selecionar uma e exclui-la desde que já não esteja sendo utilizada
Fluxo Exceção	Rota já foi acessada

Fonte: Autor (2018)

Tabela 11. User Story - Cadastrar Usuário.

User Story	Cadastrar Usuário
Quem ?	TI
O que?	Precisamos cadastrar usuários
Por que?	Para que mais pessoas possam utilizar a aplicação
Critérios de aceitação	Passando um email e senha, seja cadastrado um usuário para utilizar a aplicação
Fluxo Exceção	Email já foi cadastrado

Fonte: Autor (2018)

Tabela 12. User Story - Alterar Usuário.

User Story	Alterar Usuário
Quem ?	TI
O que?	Precisamos Alterar usuários
Por que?	Para ajustar suas permissões quando necessário
Critérios de aceitação	Ao pesquisar um usuário e seleciona-lo, o Sistema permita que eu altere suas permissões

Fonte: Autor (2018)

Tabela 13. User Story - Pesquisar Usuários.

User Story	Pesquisar Usuário
Quem ?	TI
O que?	Precisamos Pesquisar usuários
Por que?	Para poder visualizar os usuários e saber quem está usando o Sistema
Critérios de aceitação	Ao clicar em pesquisar, trazer todos os usuários Ao clicar no email, retornar o usuário

Fonte: Autor (2018)

Tabela 14. User Story - Deletar Usuário.

User Story	Excluir Usuário
Quem ?	TI
O que?	Precisamos Excluir usuários
Por que?	Para que usuários que não estejam mais na empresa não utilizem o sistema
Critérios de aceitação	Ao pesquisar um usuário e seleciona-lo, o Sistema permita que eu o exclua

Fonte: Autor (2018)

Tabela 15. User Story - Consultar CEP.

User Story	Consultar Cep
Quem ?	Motorista
O que?	Precisamos consultar cep e que mostrando a localização dele
Por que?	Para que possamos saber onde se localiza o endereço com esse CEP
Critérios de aceitação	Ao inserir um cep o sistema traga um mapa com a localização dele
Fluxo Exceção	Cep inválido

Fonte: Autor (2018)

Tabela 16. User Story - Abrir Rota no Maps.

User Story	Abrir Rota no Google Maps
Quem ?	Motorista
O que?	Preciso abrir a rota no gerada no google maps
Por que?	Para poder utilizar a rota com o GPS
Critérios de aceitação	Selecionar uma rota gerada,abrir o google maps com o itinerário.

Fonte: Autor (2018)

Tabela 17. User Story - Consultar CEPs.

User Story	Consultar Cep
Quem ?	Operadores de Logística
O que?	Precisamos consultar ceps
Por que?	Para podermos saber a lista de ceps com base na busca para poder criar regiões e analisar demanda
Critérios de aceitação	Ao clicar em buscar cep ele traga todos os ceps cadastrados Ao inserir o nome de uma rua, o sistema deve trazer todas as ocorrências de ceps com o nome de rua inserido Ao inserir o nome de uma cidade e a sigla do estado, traga todos os ceps cadastrados para a cidade Ao inserir o nome de uma cidade e um bairro traga todos os Ceps cadastrados para a cidade

Fonte: Autor (2018)

Tabela 18. User Story - Cadastrar Pessoa.

User Story	Cadastrar Pessoa
Quem ?	Operador Logístico
O que?	Precisamos cadastrar uma usuários
Por que?	Para ter a informação da pessoa que utiliza o sistema e não apenas o email
Critérios de aceitação	Passando as informações de Pessoa Física ou Pessoa Jurídica, o sistema deve cadastrá-las e vincula-las ao usuário
Fluxo Exceção	CPF já cadastrado

Fonte: Autor (2018)

Tabela 19. User Story - Alterar Pessoa.

User Story	Alterar Pessoa
Quem ?	Operador Logístico
O que?	Precisamos Alterar cadastro de pessoa
Por que?	Para corrigir possíveis erros de cadastro
Critérios de aceitação	Após pesquisar uma pessoa, o sistema deve permitir que eu altere seu cadastro

Fonte: Autor (2018)

Tabela 20. User Story - Pesquisar Pessoa.

User Story	Pesquisar Pessoa
Quem ?	Operador Logístico
O que?	Precisamos Pesquisar Pessoas cadastradas
Por que?	Para Verificar as informações pessoas que possuem cadastro
Critérios de aceitação	Ao clicar em buscar, buscar todos Ao inserir um tipo, buscar todas as pessoas do Tipo Ao inserir um CPF/CNPJ retornar a pessoas respectiva Ao inserir razão social, buscar uma lista de pessoas com a razão social buscada

Fonte: Autor (2018)

Tabela 21. User Story - Cadastrar Empresa

User Story	Cadastrar Empresa
Quem ?	Gerente
O que?	Preciso cadastrar minha empresa
Por que?	Para que meus funcionários possam utilizar o Sistema e estarem vinculados a minha empresa
Critérios de aceitação	Após inserida as informações da empresa, o Sistema salva o cadastro
Fluxo Exceção	CNPJ já cadastrado

Fonte: Autor (2018)

Tabela 22. User Story - Alterar Empresa.

User Story	Alterar Empresa
Quem ?	Gerente
O que?	Precisamos Alterar cadastro de Empresa
Por que?	Caso alguma informação da empresa mude
Critérios de aceitação	Após pesquisar retornar a empresa, abrir a tela para poder altera-la.

Fonte: Autor (2018)

Tabela 23. User Story - Pesquisar Empresa.

User Story	Pesquisar Empresa
Quem ?	Gerente
O que?	Precisamos Pesquisar Empresas
Por que?	Verificar as filiais e os dados cadastrados
Critérios de aceitação	Após clicar em buscar, trazer todas as empresas filiais relacionadas

Fonte: Autor (2018)

Tabela 24. User Story - Cadastrar Funcionário.

User Story	Pesquisar Funcionários
Quem ?	Operador Logístico
O que?	Precisamos Cadastrar Funcionários
Por que?	Para que elas estejam vinculados a empresa e consigam utilizar o Sistemas
Critérios de aceitação	Após Inserir as informações do funcionário e clicar em salvar, o sistema deve cadastrar o funcionário

Fonte: Autor (2018)

Tabela 25. User Story - Alterar Funcionário.

User Story	Pesquisar Funcionários
Quem ?	Operador Logístico
O que?	Precisamos Alterar o Cadastros dos Funcionários
Por que?	Por que podem haver mudanças, como cargo, ou mesmo de filial da empresa
Critérios de aceitação	Após pesquisar um funcionário, abrir uma tela para alterar as informações, realizando a alteração dos dados o sistema deve salvar essa alteração.

Fonte: Autor (2018)

Tabela 26. User Story - Deletar Funcionário.

User Story	Pesquisar Funcionários
Quem ?	Operador Logístico
O que?	Precisamos Deletar um Funcionário
Por que?	Por que o funcionário pode sair da empresa
Critérios de aceitação	Após pesquisar um funcionário, seleciona-lo e exclui-lo

Fonte: Autor (2018)

Tabela 27. User Story - Pesquisar Funcionário.

User Story	Pesquisar Funcionários
Quem ?	Operador Logístico
O que?	Precisamos Pesquisar um Funcionário
Por que?	Para poder checar o quadro de funcionários cadastrados no Sistema
Critérios de aceitação	Ao inserir uma empresa, buscar os funcionários cadastrados na nela

Fonte: Autor (2018)

Tabela 28. User Story - Pesquisar Região.

User Story	Pesquisar Região
Quem ?	Operador Logístico
O que?	Precisamos Pesquisar região cadastrada
Por que?	Para poder ver os ceps que compõem a região
Critérios de aceitação	Ao inserir a empresa, buscar as regiões dela Ao inserir uma empresa matriz, trazer todas as regiões da matriz e de suas filiais.

Fonte: Autor (2018)

Tabela 29. User Story - Alterar Região.

User Story	Alterar Região
Quem ?	Operador Logístico
O que?	Precisamos Alterar uma região cadastrada
Por que?	Pois os ceps que a compõem podem mudar.
Critérios de aceitação	Ao buscar a região de uma empresa e selecionar para alterar, abrir a tela de alteração e salvar as alterações

Fonte: Autor (2018)

Tabela 30. User Story - Deletar Região.

User Story	Deletar Região
Quem ?	Operador Logístico
O que?	Precisamos Deletar uma região cadastrada
Por que?	Região pode entrar em desuso
Critérios de aceitação	Ao buscar a região de uma empresa e selecionar para deletar, Sistema deve deletar essa região

Fonte: Autor (2018)

Tabela 31. User Story - Efetuar Login.

User Story	Efetuar Login
Quem ?	Todos
O que?	Precisamos acessar o sistema por meio de autenticação
Por que?	Necessário para que ninguém utilize o sistema sem estar identificado
Critérios de aceitação	Entrar na tela principal inserir email e senha para conectar ao sistema.

Fonte: Autor (2018)

Tabela 32. User Story - Cadastrar Região.

User Story	Cadastrar Região
Quem ?	Operador Logístico
O que?	Precisamos cadastrar uma região
Por que?	Necessário para poder parametrizar as indicações de área de atuação
Critérios de aceitação	Ao inserir Bairro e Cidade, cadastrar todos os ceps do bairro e cidade passados Ao inserir uma Cidade e um Estado, cadastrar todos os ceps da cidade e estado respectivos Ao passar uma lista de ceps, efetuar o cadastro de uma região com essa lista

Fonte: Autor (2018)

2.3.1. BackLog

Todas as tarefas elencadas foram organizadas com o Trello, ferramenta para Kanban escolhido para o Task Control do projeto. As Figuras 8 e 9 apresentam fragmentos dos quadros do Trello de front-end e back-end respectivamente.

Figura 8. Fragmento do Quadro e Front-end do Trello.

Refatoração	Páginas	Forms
Concluído Configuração Complemento BugFix Código 015 - Retirar conexão ao BD para extrair o segredo e criptografia	Concluído HTML/SCSS Código 004 - Criar Página de Login Concluído HTML/SCSS Código 013 - Criar Página de Profile HTML/SCSS Código 014 - Criar Logout Concluído HTML/SCSS Código 019 - Criar Página Roteirização	Concluído HTML/SCSS Código 005 - Criar Formulário de Login Concluído HTML/SCSS Código 029 - Criar Formulário de Login Concluído Complemento HTML/SCSS Código 030 - Inserir Validação no Formulário de login Concluído Complemento HTML/SCSS Código 039 - Criar Formulário para Atualização de Pessoa Concluído HTML/SCSS Código 044 - Desenvolver Form para Alteração de Endereço

Fonte: Autor (2018)

Figura 9. Fragmento do Quadro e Back-end do Trello.

Fonte: Autor (2018)

Segue os links para acesso aos quadros no Trello:

Front-end: <https://trello.com/b/AMaFiirR/front-end-sysrlog-trabalho-de-gradua%C3%A7%C3%A3o>

Back-end: <https://trello.com/b/Ot6gDOSy/back-end-sysrlog-trabalho-de-gradua%C3%A7%C3%A3o>

2.4. Tecnologias Aplicadas

Para atender as User Stories elencadas anteriormente, faz-se necessário a seleção de tecnologias a serem utilizadas para o desenvolvimento do Software. As tecnologias definidas foram divididas em duas categorias: back-end e front-end.

O *back-end* é a camada do servidor que recebe as requisições, faz acesso ao banco de dados e envia as respostas. Também é a responsável pelas regras de negócio e por prover segurança a aplicação (MARQUES, 2017).

O *front-end* é camada correspondente a interação com o usuário, é a camada que irá receber a entrada de dados, enviar ao back-end, receber a resposta e apresentar ao usuário (MARQUES, 2017).

2.4.1. Back-end

A seguir serão apresentadas, as tecnologias aplicadas no desenvolvimento do back-end do software.

2.4.1.1. Linguagem Java

A linguagem de programação escolhida foi a Java. Ela foi desenvolvida na década de 90 pela Sun MicroSystems. No ano de 2008, a empresa foi adquirida pela Oracle. O diferencial da Linguagem Java frente as linguagens convencionais, é que ao invés de um programa ser compilado para o código nativo da plataforma, na qual o programa foi desenvolvido, ele é compilado para um bytecode(código da linguagem Java) que será interpretada pela JVM Java Virtual Machine ou Máquina virtual do Java (CAELUM, 2018).

A JVM é a responsável por fazer com que um código Java possa ser executado em diferentes plataformas que utilizam diferentes Sistemas Operacionais.

Java foi escolhida por conta de sua portabilidade e Frameworks para desenvolvimento, além de sua grande utilização. Segundo dados disponíveis no site da linguagem, 97% dos Desktops corporativos executam Java e cerca 89% dos computadores residências possuem Java instalado. Segundo Udacity (2018) Java aparece sempre nas primeiras posições, como linguagem mais utilizada no mercado.

2.4.1.2. Formato para Transmissão de Dados

Utilizar-se-á o formato Java Script Object Notation (JSON) para a transmissão de dados no Projeto. No site oficial JSON é descrito como um formato leve para troca de dados, fácil para ser escrito e lido por humanos e fácil para serem convertidos e gerados por máquinas (JSON, 2018) .

O formato foi escolhido por conta de além de ser o formato padrão utilizado pelo Spring Framework, pela API de distâncias do Google e pela API de consultas de ceps do viaCep, é também o formato utilizado pelos componentes que fazem parte do *front-end*.

2.4.1.3. Maven

Maven é uma ferramenta desenvolvida para facilitar o processo de desenvolvimento, Build (construir um programa executável, a partir de código fonte) e gerenciar qualquer projeto baseado em Java (APACHE, 2018).

Esta ferramenta foi originalmente iniciado para facilitar o processo de Build , de um Projeto chamado Jakarta Turbine. Este projeto é composto por diversos subprojetos, cada um com suas próprias Builds e JARS (Arquivos Java).

A ferramenta baseou-se na utilização de Project Object Model (POM), um arquivo formato XML que contém todas as informações necessárias para a Build do projeto, configurações, formatos de compilação e a gestão das dependências.

O Maven irá realizar a gestão de todas as dependências da parte de BackLog do projeto que serão listadas a seguir.

2.4.1.4. Spring

O principal Framework e base para o *back-end* é o Spring. Esta ferramenta foi lançada no Ano de 2002. Hoje é mantida pela Pivotal Software. Suas principais características são a Inversão de Controle e a Injeção de dependências.

A Inversão de Controle é quando as chamadas da aplicação não podem ser controladas manualmente ou não possuem ordem definida para execução. A Injeção de Dependências é um padrão de desenvolvimento que busca manter um nível de acoplamento baixo entre os módulos das aplicações e que nesse caso, o Container ou Framework que disponibiliza ou ‘injeta’ os componentes entre os módulos sempre que necessário (FOWLER, 2004).

O Spring é composto também de outros módulos, denominados no site como *Projects* (PIVOTAL, 2018A). Para o desenvolvimento utilizar-se-á dos seguintes *Projects*:

- I. Spring ou Spring MVC: é um dos projetos iniciais do Spring com foco na Inversão de Controle e Injeção de dependências. Esse projeto provê as dependências necessárias para a criação de *Services*, *Views* e *Controllers*;
- II. Spring Boot DevTools: Tem a função de prover toda a configuração necessária para executar uma aplicação baseada Spring de Maneira automática, economizando tempo e trabalho;
- III. Spring Data JPA: Torna a comunicação entre a aplicação e o banco de dados mais simples, a partir de parametrizações é possível definir Classes que são a representação de uma determinada tabela do banco de dados, e definir classes que executam funções de consulta, alteração, criação e exclusão a essas tabelas denominados repositórios. Outra grande vantagem dos repositórios é que eles evitam a criação de código repetitivo;

- IV. Spring Security: Toda a parte de autenticação e segurança é de responsabilidade deste Projeto. Esta ferramenta fornece configurações, classes e métodos para serem utilizados de forma a controlar a segurança da aplicação, porem ele oferece autenticação básica, para complementar utilizar-se-á de JSONWebToken o JWT que será enviado em todas as requisições realizadas;
- V. Spring Test: É responsável pela execução de testes Unitários utilizando o JUnit;

Informações detalhadas ou de outros Projects do Spring podem ser encontrados no seu próprio site, que também apresenta toda a documentação de cada uma dos Projects.

2.4.1.5. Banco de Dados

O Banco de Dados selecionado para o Projeto é o MySQL. É um banco de dados gratuito distribuído pela Oracle utilizado por diversas empresas ao redor do mundo. Algumas características que definiram o MySQL como Banco de Dados do projeto são o excelente desempenho e escalabilidade, a compatibilidade com a linguagem Java e o baixo consumo de recursos do *Host* (MYSQL, 2018).

2.4.1.6. Plugins para Base de Dados

- a. H2Database: Banco de dados em memória, utilizar-se-á o H2 para Testes Unitários, verificação dos Scripts do Liquibase e para testes simples dos Serviços e Endpoints;
- b. MySQL Connector: Plugin necessário para a conexão do Software em Java com o banco de Dados MySQL;
- c. Liquibase: Biblioteca Java de código fonte aberto, para controle e gestão do Banco de Dados de uma aplicação, a partir de Scripts Sql ou arquivos XML. O liquibase pode executar tanto comandos DDL Data Definition Language como DML Data Manipulation Language (DATICAL, 2018) .

2.4.1.7. Demais Dependências utilizadas

- a. Project Lombok: Biblioteca Java que gera métodos essências de forma automática, como métodos get, set, toString, equals, construtores sem atributos dentre outros;
- b. Javax.Json : Biblioteca utilizada para leitura, conversão e criação de arquivos JSON;

- c. HttpClient : Cliente para realizar requisições HTTP para WebServices, faz-se necessária a utilização dessa biblioteca para realizar requisições à API de Distâncias do Google e ao Servidor do ViaCep.

2.4.1.8. Softwares utilizados

Spring Tool Suite conhecido como STS, é uma IDE customizada, que tem o Eclipse como base. É distribuída pela Pivotal Software, desenvolvedora do Projeto Spring. O STS disponibiliza um ambiente completo para criação e execução de uma aplicação baseada no Framework Spring. Contém inclusive um servidor Web embutido, que executa a aplicação desenvolvida automaticamente, sem necessidade de realizar nenhum tipo de configuração (PIVOTAL, 2018B).

XAMPP é um Software Livre Promovido pela Iniciativa Apache Friends. É composto por Servidos Apache, PHP, Perl e atualmente a distribuição Baseada em MySQL MariaDB. Esse software foi elencado para ser utilizado, pelo baixo consumo de recursos de Hardware, fácil instalação, e simples utilização de suas ferramentas (FRIENDS, 2018). Para o projeto a ferramenta da plataforma a ser utilizada é o MariaDB.

2.4.1.9. Recursos Externos

Para satisfazer as necessidades dos usuários, faz-se necessária a utilização de dois recursos externos:

Distance Matrix API é fornecida pela Google. A API disponibiliza o cálculo de distâncias e tempo de percurso, entre ponto de origem e ponto de destino. É utilizada por meio de requisições HTTP, tendo que enviar a URL utilizando o padrão apresentado na documentação fornecida pela Google. Retorna um JSON com as informações de distância e tempo;

ViaCep é um WebService gratuito para consulta de ceps. As requisições são feitas pelo Protocolo HTTP. Ao efetuar uma requisição o WebService retorna um JSON com as informações do CEP, caso ele seja válido.

2.4.2. *Front-end*

As tecnologias apresentadas, a seguir, fazem parte da camada de *front-end* do projeto.

2.4.2.1. NPM

Npm é um Software para gerenciamento de Pacotes/Dependências para linguagem de programação JavaScript/TypeScript. Foi lançado inicialmente em 2010.

O NPM consiste em uma base de dados online com diversas dependências que podem ser baixadas pelo cliente NPM, utilizando linha de comando (NPM, 2018).

2.4.2.2. Ionic

Ionic é um SDK Software Development Kit que foi lançado em 2013, sendo desenvolvido por Max Lynch, Ben Sperry e Adam Bradley. O Ionic possibilita o desenvolvimento de aplicativos híbridos ou multiplataforma, podendo ser executados em IOS, Android, Windows e Browser. O Ionic é executado em uma camada acima do Cordova e provê a Interface ao usuário, enquanto o Cordova age transcrevendo as ações do *front-end* em comandos para a plataforma no qual aplicação Ionic é executada (IONIC, 2018).

Para prover o desenvolvimento multiplataforma o Ionic utiliza os recursos primordiais do desenvolvimento de *front-end* no formato para Web. Para criação de aplicativos são utilizados HTML5, CSS e JavaScript. Esses itens já são o suficiente para o desenvolvimento de uma Página Web, mas o Ionic vai mais além, permitindo implementar também TypeScript e Angular.

Além das tecnologias já citadas, o Ionic ainda permite a utilização de recursos nativos da plataforma que o está executando, como: câmera, GPS, acesso a arquivos locais dentre outros, esses recursos são possíveis de serem executados por conta do Cordova. Outro recurso disponível consiste em componentes já desenvolvidos do Ionic, que podem ser utilizados para enriquecer o *front-end* e deixar o visual da aplicação mais elegante.

O Ionic foi selecionado para implementação do *front-end* por conta da capacidade de ser executado em qualquer plataforma, podendo acessar os recursos nativos do dispositivo no qual é executado, sem a necessidade de fazer nenhum tipo de alteração do código fonte. Além de possuir componentes já preparados para serem utilizados.

2.4.2.3. HTML5

Hypertext Markup Language (HTML) é uma linguagem para estruturação e apresentação de conteúdo. É basicamente a Tecnologia chave da Internet na maneira que conhecemos. Todos os sites utilizam o HTML.

HTML foi originalmente desenvolvido para descrever semanticamente, documentos científicos, mas devido ao seu design, ele pode ser adaptado para tornar-se a estrutura básica de um Site. Sua nova Versão é a 5, foi lançada na íntegra no ano de 2014. Trazendo diversos novos recursos como: funcionamento *offline*, armazenamento local, viabilização de multimídia, novos efeitos, melhoria de desempenho e capacidade de acessar dispositivos (W3C, 2018).

2.4.2.4. CSS

CSS é a abreviação de *Cascade Style Sheets* que traduzindo significa, Folha e Estilos em Cascata. Esta ferramenta é utilizada junto a com linguagens de marcação como HTML (BOS, 2018). Sua principal função é determinar o visual de uma Página *web*, tornando-a apresentável aos olhos de quem a utiliza. Ele é camada responsável por coordenar todos os estilos aplicados a Página Web e ‘dar vida’ a Página estática baseada em HTML.

2.4.2.5. TypeScript

Lançada em 2012 pela Microsoft, o TypeScript é um superconjunto do JavaScript. Sua principal diferença com relação ao JavaScript é suportar o uso de Programação Orientada a Objeto. Com o TypeScript é possível escrever classes, interfaces, indicar o formato de retorno de métodos e indicar o tipo das variáveis (TYPESCRIPT, 2018).

O TypeScript consegue executar JavaScript assim, possibilitando a utilização de ferramentas já desenvolvidas. Na questão de execução, como apresentado no site da ferramenta, todo o código escrito em TypeScript é compilado para JavaScript podendo assim ser interpretado e executado no *Browser*.

2.4.2.6. AngularJS

Angular foi lançado em 2016 e é mantido pela Google. Trata-se de um *Framework* de código aberto, baseado em JavaScript. Tem a função de construir *interfaces* para uma aplicação web a partir da utilização de HTML, CSS e JavaScript.

O Angular funciona a partir da leitura de páginas HTML que contenham atributos adicionais em suas *Tags* e interpreta esses atributos como diretivas, para unir partes de entrada e saída da página, com uma variável. Facilitando a comunicação entre a página e o controlador da página (ANGULARJS, 2018).

2.4.2.7. Cordova

Cordova foi lançado em 2017. É um *framework* para desenvolvimento de aplicações *mobile* baseado em HTML, CSS e JavaScript. Ele também é capaz de acessar recursos nativos do ambiente no qual é executado (CORDOVA, 2018).

2.4.2.8. Softwares utilizados

VSCode, lançado em 2015 e desenvolvido pela Microsoft, o VSCode é um editor de código fonte com distribuições para Windows, Linux e MacOS. Suas principais características são: Complemento inteligente de código (incluindo TypeScript, HTML, CSS e os componentes Ionic), Refatoração de código, suporte a Depuração, *Snippets* e controle de Git Incorporado (MICROSOFT, 2018).

Ionic Cli: *Ionic Command Line Interface* conhecido como Ionic CLI é uma ferramenta criada para o desenvolvimento de aplicações Ionic. Com este *software* é possível criar uma aplicação Ionic e gerar automaticamente alguns componentes. O CLI também facilita os processos de compilação e execução além de prover processos de *Build* e *Live-Reload*.

2.4.3. Ferramentas de Teste

Na sequência serão apresentadas as Ferramentas de Teste utilizadas na Etapa de Verificação e Validação do *software* desenvolvido.

2.4.3.1. JUnit

É um Framework para o desenvolvimento de Testes de Unitários. O conceito de Testes Unitários é definido pela IEEE como: “Atividade capaz de testar unidades de hardware ou software ou grupo de unidades relacionadas”. Foi Desenvolvido originalmente por Kent Beck e Erich Gamma. O Projeto desenvolvido utilizar-se-a do Junit para o desenvolvimento de testes de alguns métodos no *back-end*, nas camadas de Serviço e Repositórios.

2.4.3.2. JaCoCo

JaCoCo é uma biblioteca Java, gratuita, para a análise de cobertura do código. Dentro do Projeto, utilizar-se-a dessa ferramenta no formato de *Plugin* do Maven. Esse *Plugin* permite a execução de um agente de análise para os testes, em tempo de execução. E ao final da execução é gerado um relatório de cobertura dos testes (ECLEMMMA, 2018).

O relatório gerado pelo JaCoCo será utilizado no SonarQube, próxima ferramenta a ser apresentada, para apresentar os dados de cobertura de testes do projeto.

2.4.3.3. SonarQube

SonarQube é um Plataforma *Open-Source* (Código Aberto) desenvolvida pela empresa SonarSource. O SonarQube é definido como: Plataforma para inspeção contínua na qualidade do código de um Projeto. Atualmente suporta diversas linguagens, dentre elas as duas utilizadas no Projeto, Java no *back-end* TypeScript no *front-end* (SONAROURCE, 2018).

Para o projeto, utilizar-se-a das seguintes funcionalidades da plataforma: detecção de *bugs*, detecção de vulnerabilidades, análise da qualidade do código, análise da complexidade do código, análise da cobertura de Testes(utilizando o JaCoCo), análise do atendimento das boas práticas de programação e revisão dos denominados *CodeSmells*, que pode ser entendido como códigos mau cheirosos, quanto mais mau cheiroso um código é, pior será mantê-lo e alterá-lo.

2.4.3.4. PostMan

PostMan é uma API de desenvolvimento, que proporciona mecanismos completos para testes de Requisições HTTP (POSTDOT, 2018). É de grande importância ao Projeto, pois é utilizando-o que as Rotas Desenvolvidas do *back-end* Serão testadas e validadas.

2.4.4. Versionamento

Com relação ao versionamento do projeto, foram criado dois repositórios no GitHub, um para o *front-end* e um para o *back-end*.

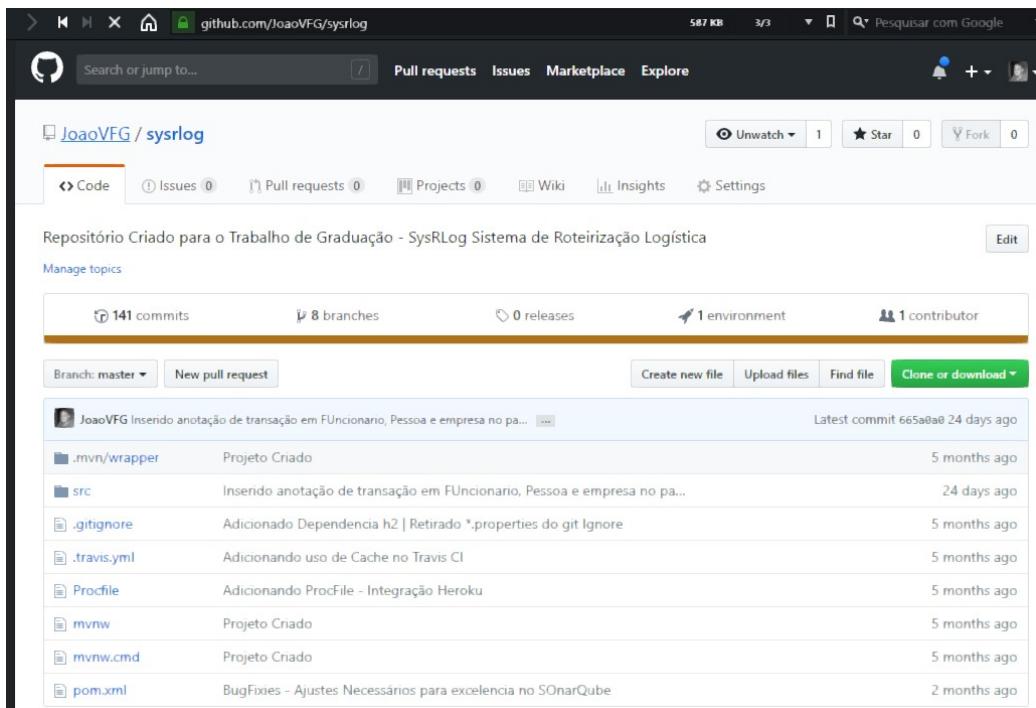
A seguir seguem os links de acesso aos repositórios:

Front-end: <https://github.com/JoaoVFG/sysrlogapp>

Back-end: <https://github.com/JoaoVFG/sysrlog>

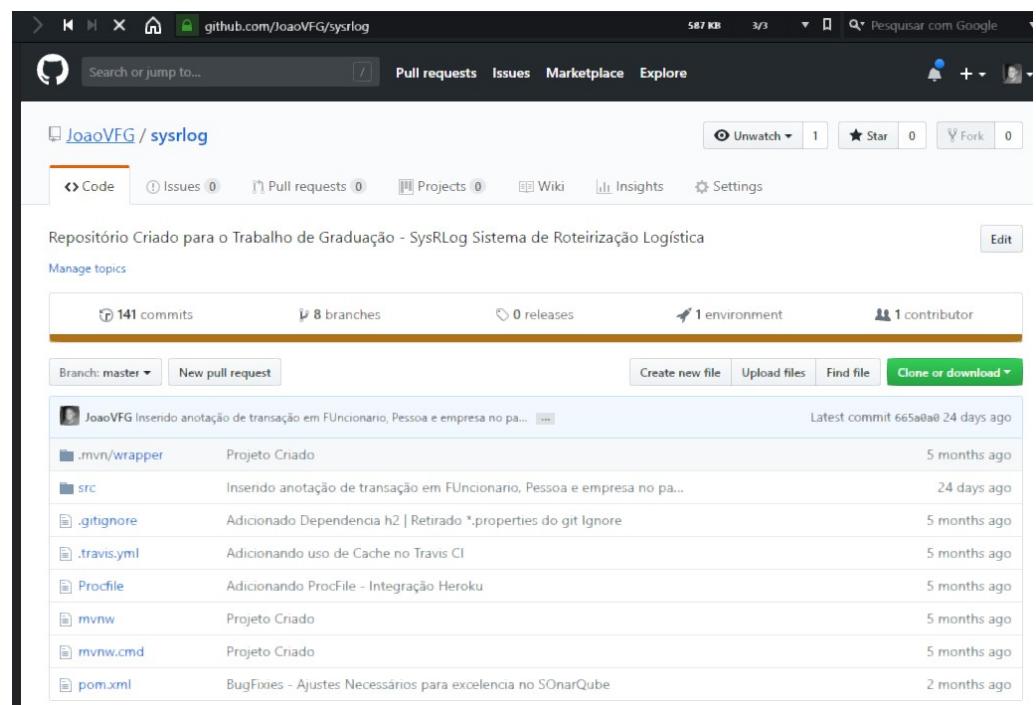
As Figuras 10 e 11 apresentaram os repositórios de *front-end* e *back-end* do Projeto, respectivamente.

Figura 10. Repositório Github *Front-end*.



Fonte: Autor (2018)

Figura 11. Repositório Github *Back-end*.



Fonte: Autor (2018)

3. DESENVOLVIMENTO

Neste capítulo é apresentada a fase de Desenvolvimento do Projeto e será composto por: Arquitetura da Solução, Modelagem e Gestão dos Dados, Arquitetura do Software, Segurança e para concluir a apresentação de uma Visão Geral do Sistema.

3.1. Padrão do Projeto

O padrão de escolhido para o desenvolvimento do projeto é o MVC: Model, View e Controller, traduzido ao português para Modelo, Visão e Controlador. O padrão foi escolhido devido a organização que ele trás ao projeto além de separar o código de maneira lógica facilitando o desenvolvimento (BARROS; SILVA; ESPÍNOLA, 2007).

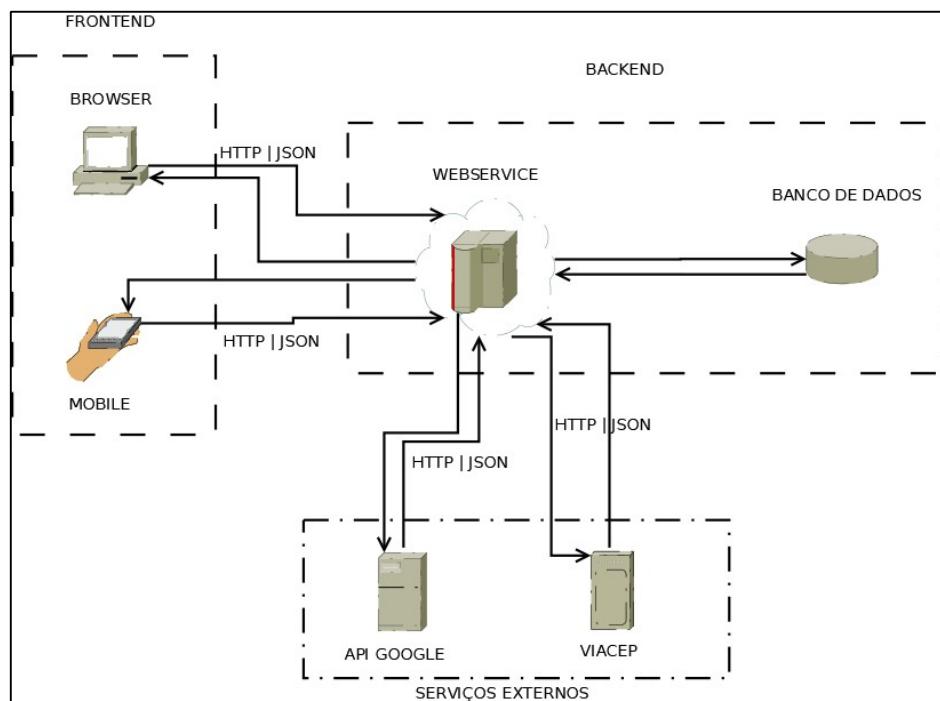
O usuário através do Navegador ou do Celular, fará uma requisição pela tela (View) que será enviada ao Controller. O Controller irá tratar as requisições vindas da View e, se necessário, irá acionar a camada Model, caso não seja, retornará a resposta para View. Se a camada Model tiver sido acionada, ela irá se conectar via um Driver ao Banco de dados e efetuará as devidas operações.

3.2. Arquitetura da Solução

A Arquitetura da Solução foi planejada para que o aplicativo (front-end) seja utilizado tanto em dispositivos móveis quanto em navegadores. O aplicativo irá se comunicar com o Software (back-end) sendo executada em um servidor, por meio de requisições HTTP e arquivos no formato JSON para transferência dos dados.

O software será responsável por receber as requisições, e executar os processos necessários para atender essa requisição, seja acessar a base de dados ou executar consultas a serviços externos. Quando os processos originados pela requisição forem resolvidos o software devolverá a resposta para o aplicativo, seja ela o resultado de uma busca, uma mensagem de confirmação ou uma advertência indicando algum problema com a requisição. A Figura 12 apresenta esses processos a partir de uma visão macro de toda a aplicação:

Figura 12. Arquitetura da Solução - Visão Geral.

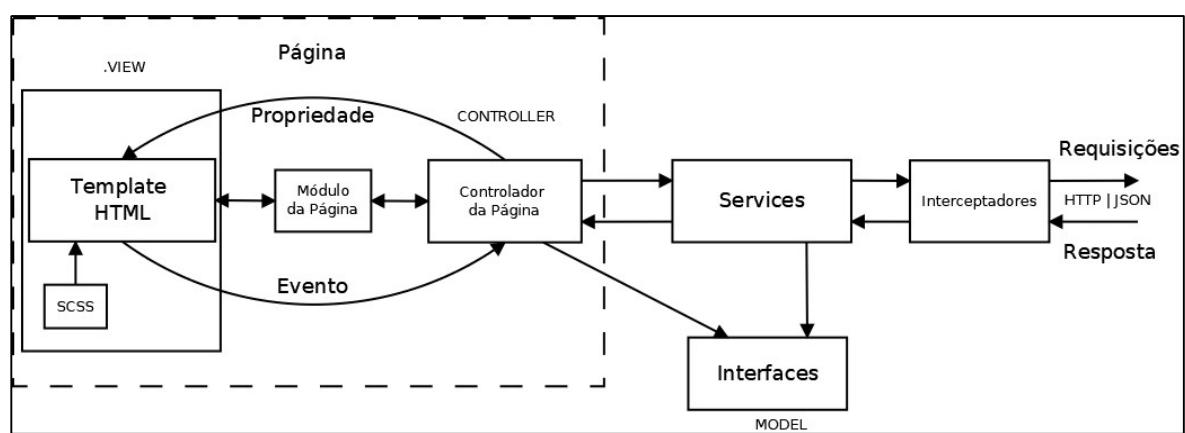


Fonte: Autor (2018)

3.2.1. Arquitetura da Solução - *Front-end*

O front-end do projeto corresponde a toda uma estrutura necessária para que a View envie requisições ao back-end, além de receber e tratar as respostas recebidas. Ao implementar o desenvolvimento utilizando Ionic, Cordova e TypeScrip, foi necessário adotar o padrão MVC também no front-end. A Figura 13 ilustra o funcionamento do front-end bem como onde estão localizados os componentes do padrão MVC

Figura 13. Arquitetura da Solução: *Front-end*.



Fonte: Autor (2018)

A estrutura do front-end é composta por:

- a. Páginas: Formada pelos *Templates* HTML e Arquivos SCSS (a combinação destes dois itens constitui a View do projeto), o Controlador da Página e pelo Módulo da Página (responsável por indicar que determinada página é controlada por determinado Controlador). Essa composição cada uma das Páginas é o Padrão utilizado pelo Ionic;
- b. Serviços: Responsáveis por realizar as Requisições Solicitadas pelas Páginas ao *back-end*, receber as respostas e devolve-las às Páginas;
- c. Interfaces: Define os Objetos que serão enviados e Recebidos. Só é possível utilizar esse recurso na camada de *front-end* devida a utilização da Linguagem TypeScript;
- d. Interceptadores: São componentes criados para realizarem duas funções no front-end: adicionar o *Token* de autenticação a todas as requisições que serão realizadas e ao receber um erro do back-end, mostrá-lo no formato de alerta na tela.

O *front-end* funciona a partir de um evento que ocorre nos *templates* HTML, o *click* em um botão por exemplo. O *click* nesse botão aciona um método dentro do controlador da página.

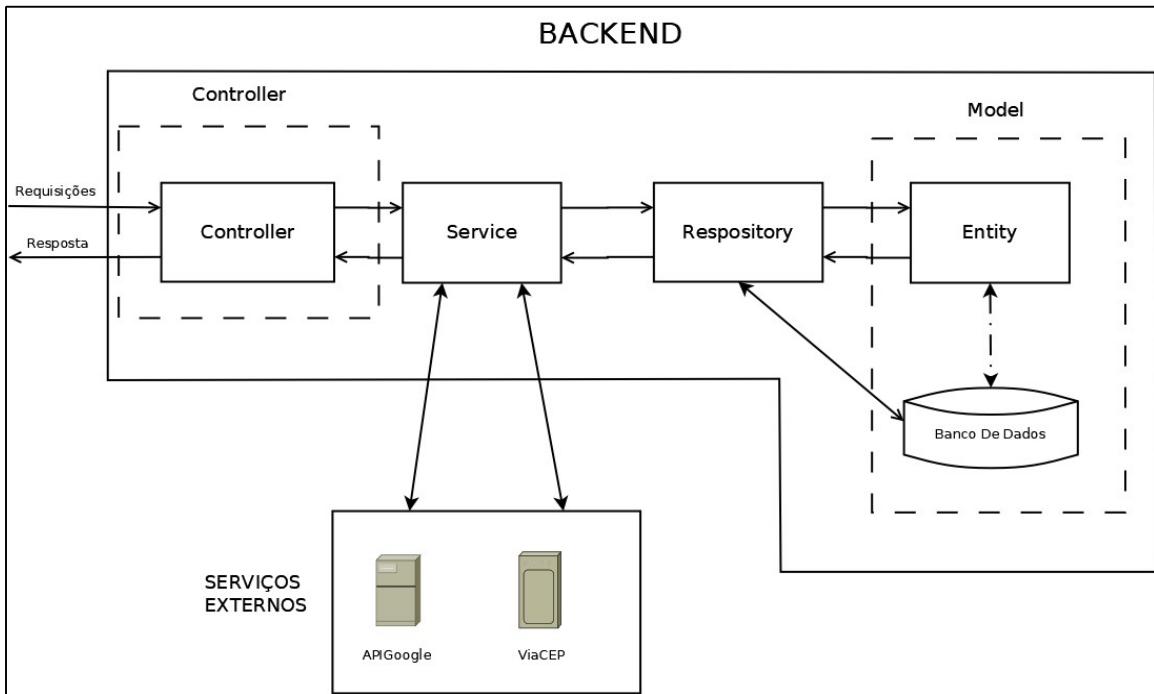
Esse Controlador irá realizar uma requisição acionando determinado método de um dos serviços, que realizará uma requisição HTTP ao *back-end*, passando se necessário, alguma informação pela URL, ou algum objeto definido pelas Models, no corpo da requisição. Quando a requisição for efetuada, o Interceptador irá inserir o *Token* no cabeçalho da requisição.

Ao receber a resposta da requisição o *Interceptador* verificará se a resposta foi um erro, caso não seja, irá deixar ela prosseguir a origem. O controlador, quando receber a resposta, irá transferir as informações aos objetos que estão ligados ao *front-end*. Quando os dados desses objetos são alterados, automaticamente eles são apresentados no *template* ao usuário isso por meio das Diretivas do Angular.

3.2.2. Arquitetura da Solução - *Back-end*

O back-end do Projeto corresponde a Camada de Model e Controller. A Figura 14 apresenta um maior detalhamento dessas camadas.

Figura 14. Arquitetura da Solução: *Back-end*.



Fonte: Autor (2018)

Todas as requisições são recebidas pelos *Controllers*. Eles validam a requisição e repassam à camada de serviço. Por sua vez, a camada de Serviço irá fazer as validações necessárias com os parâmetros passados pelo controlador, e realizar as chamadas necessárias, sejam repositórios ou outro serviço. Caso ao validar a requisição seja identificado algum erro, a camada de serviço irá retornar a exception para o controlador disparar e apresentar a View.

A camada de serviço é a responsável por realizar acesso ao banco de dados para realizar as operações básicas, utilizando as Classes de Entidades como referência.

Após os repositórios executarem seus métodos com sucesso, os serviços irão receber o resultado e retorná-lo aos controladores. O Controlador irá devolver a requisição HTTP feita pela view, com o Status (seja de sucesso ou de falha), o respectivo corpo apresentando o resultado da requisição.

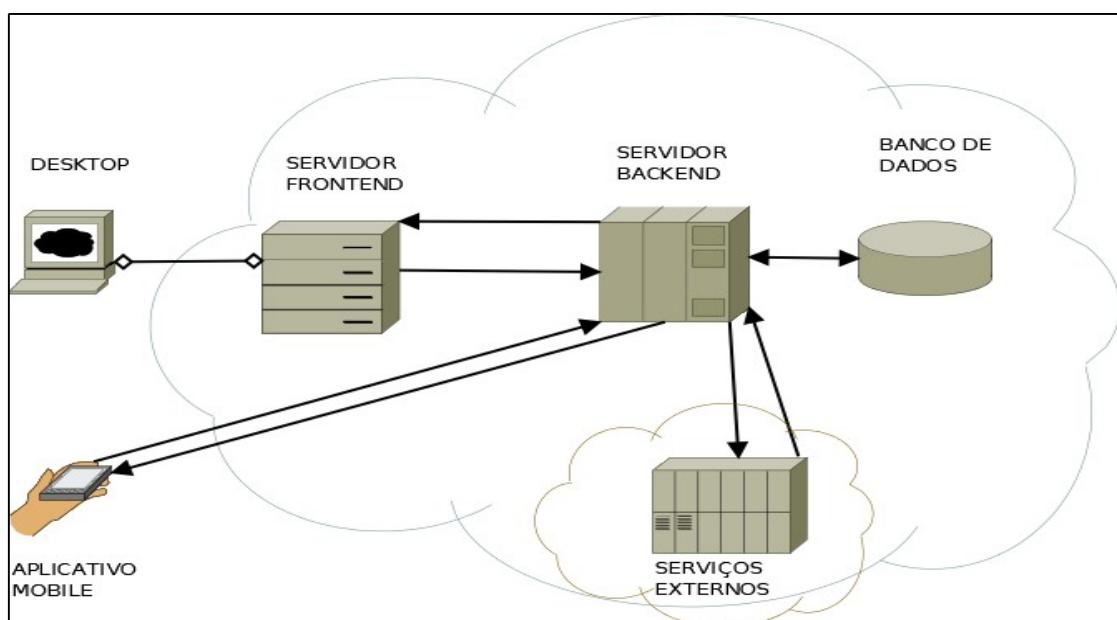
3.2.3. Arquitetura da Solução - Implantação

Para a implantação do Software é necessária a alocação do back-end em um Servidor de Aplicações Java como o TomCat, ou sua implantação em um serviço de hospedagem de aplicações em Cloud como o Heroku. Seja onde o back-end for implementado é obrigatório que ele possua conexão constante com a Internet, para que possa receber requisições e acessar os Serviços Externos.

Além de um servidor de aplicações, é necessário ter um serviço de banco de dados MySQL disponível para a armazenagem dos dados do Software. Seja qual for o serviço definido para implantação, basta configurar o acesso dentro dos arquivos de configuração do Spring.

O front-end pode ser implantado em qualquer servidor com Suporte a Linguagem TypeScript e componentes do Ionic e Cordova. Heroku e Firebase alguns são exemplos. Já para dispositivos móveis, basta que o Projeto Ionic seja compilado para o Sistema Operacional respectivo, tendo o back-end e o Banco de Dados implantados em algum servidor acessível, ele poderá ser executado. A Figura 15 apresenta a arquitetura de Implantação do Projeto.

Figura 15. Arquitetura da Solução: Implantação de Projeto.



Fonte: Autor (2018)

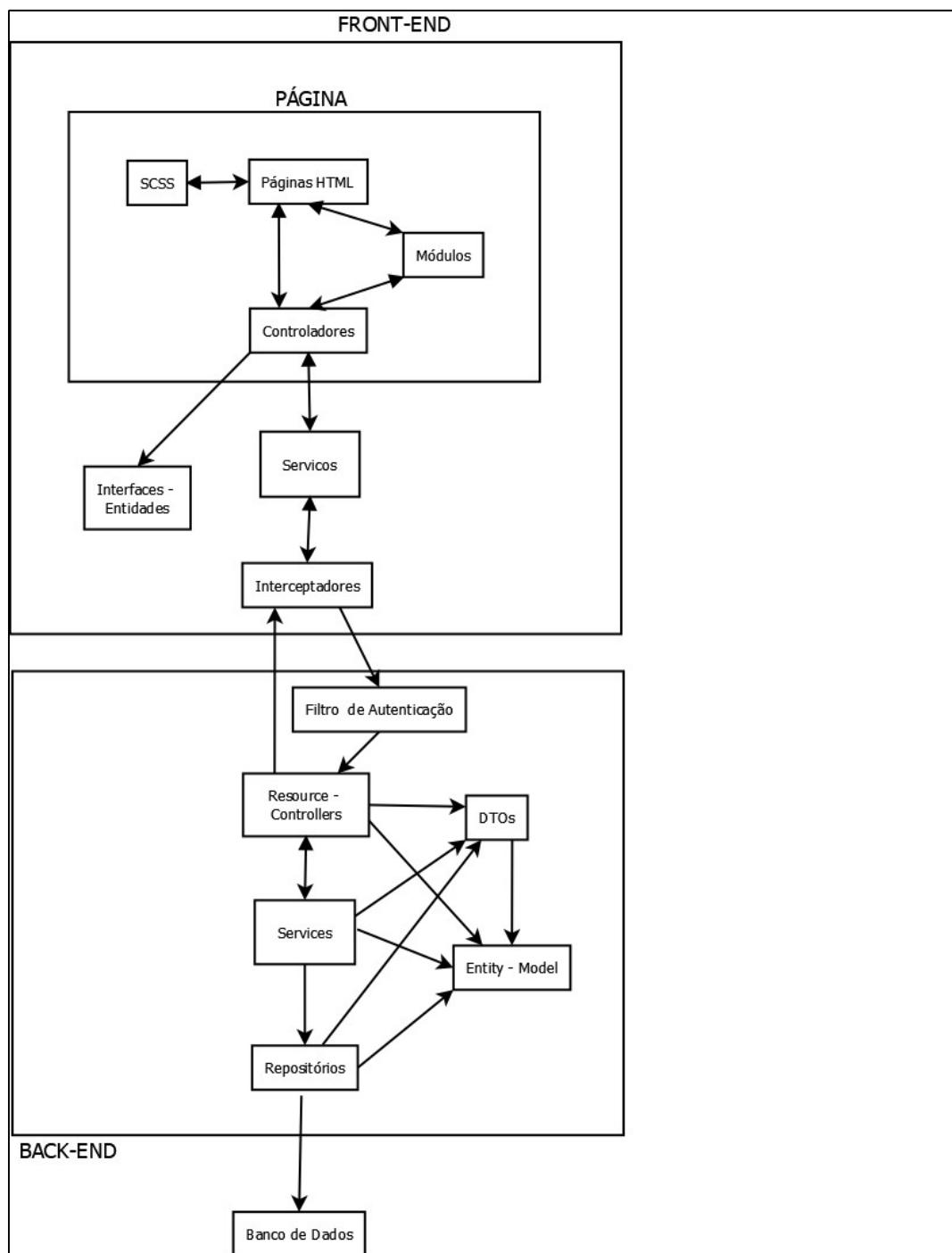
3.3. Arquitetura do Software

O presente subcapítulo apresenta a estruturação do código fonte do Projeto, tanto para o back-end quanto para o front-end, além de explicar seu funcionamento com alguns fragmentos de Código

3.3.1. Diagrama de Componentes

O Diagrama de Componentes apresentado na Figura 12 ilustra os componentes do back-end relacionados as funcionalidades de cadastros, e sua relação entre as camadas e empacotamento.

Figura 16. Diagrama de Componentes.

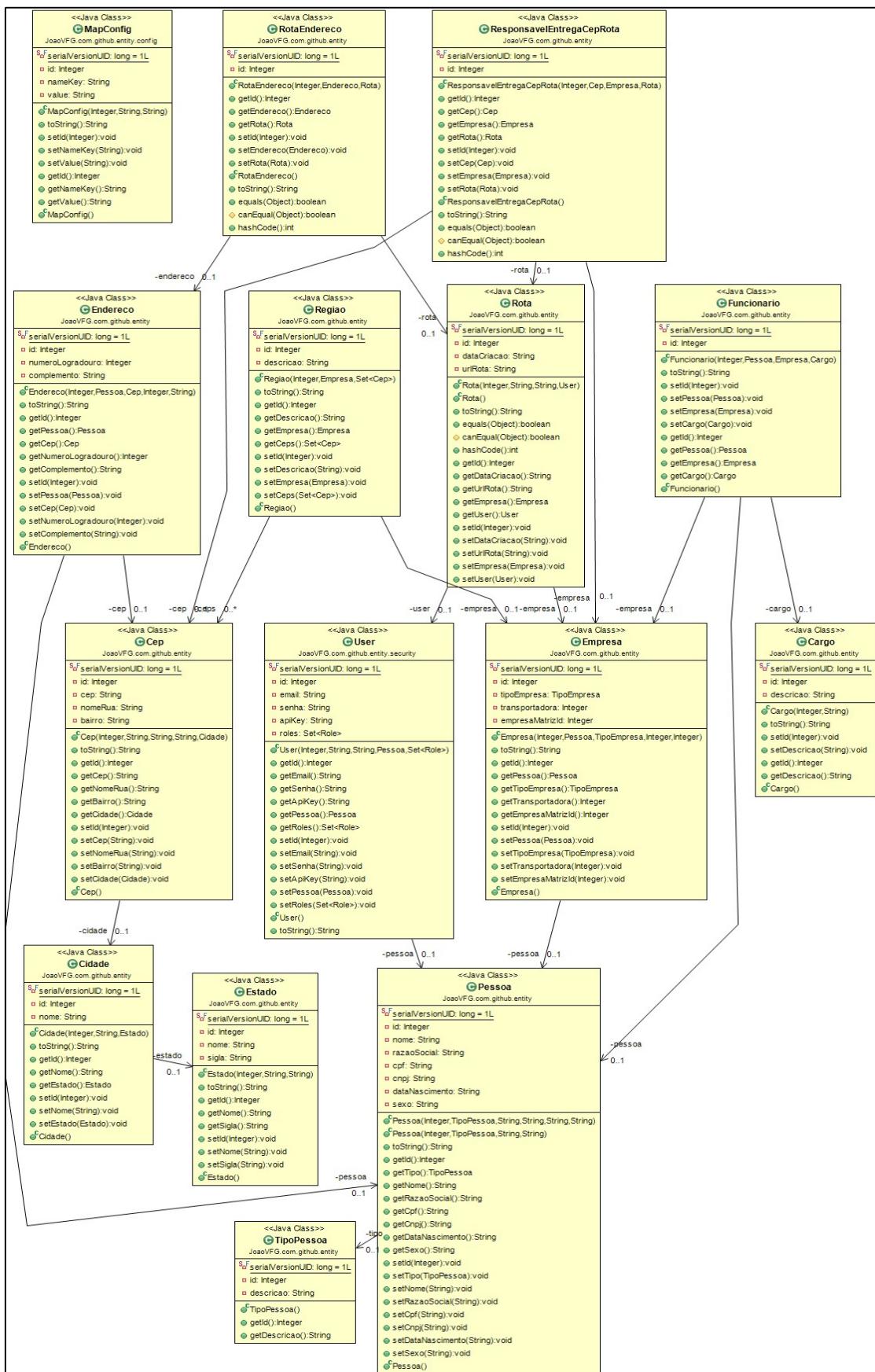


Fonte: Autor (2018)

3.3.2. Diagramas de Classes

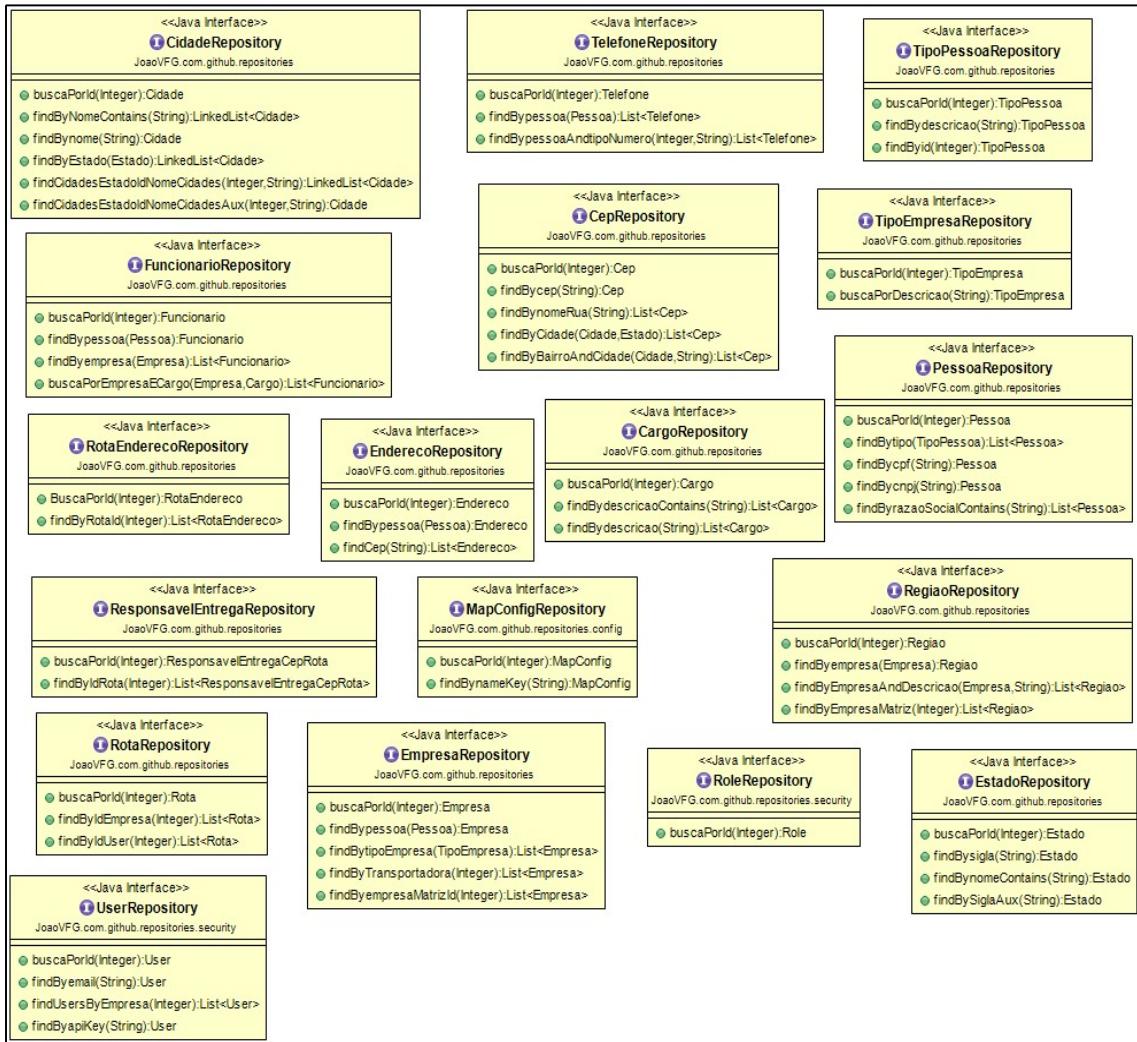
A seguir serão mostrados os Diagramas de Classe do Projeto, eles foram categorizados pelo seu empacotamento dentro do Projeto. As Figuras de 17 à 20 apresentam os Diagramas de Classe de Entidades, Repositórios, Serviço e Controladores respectivamente.

Figura 17. Diagrama de Classes: Entidades.



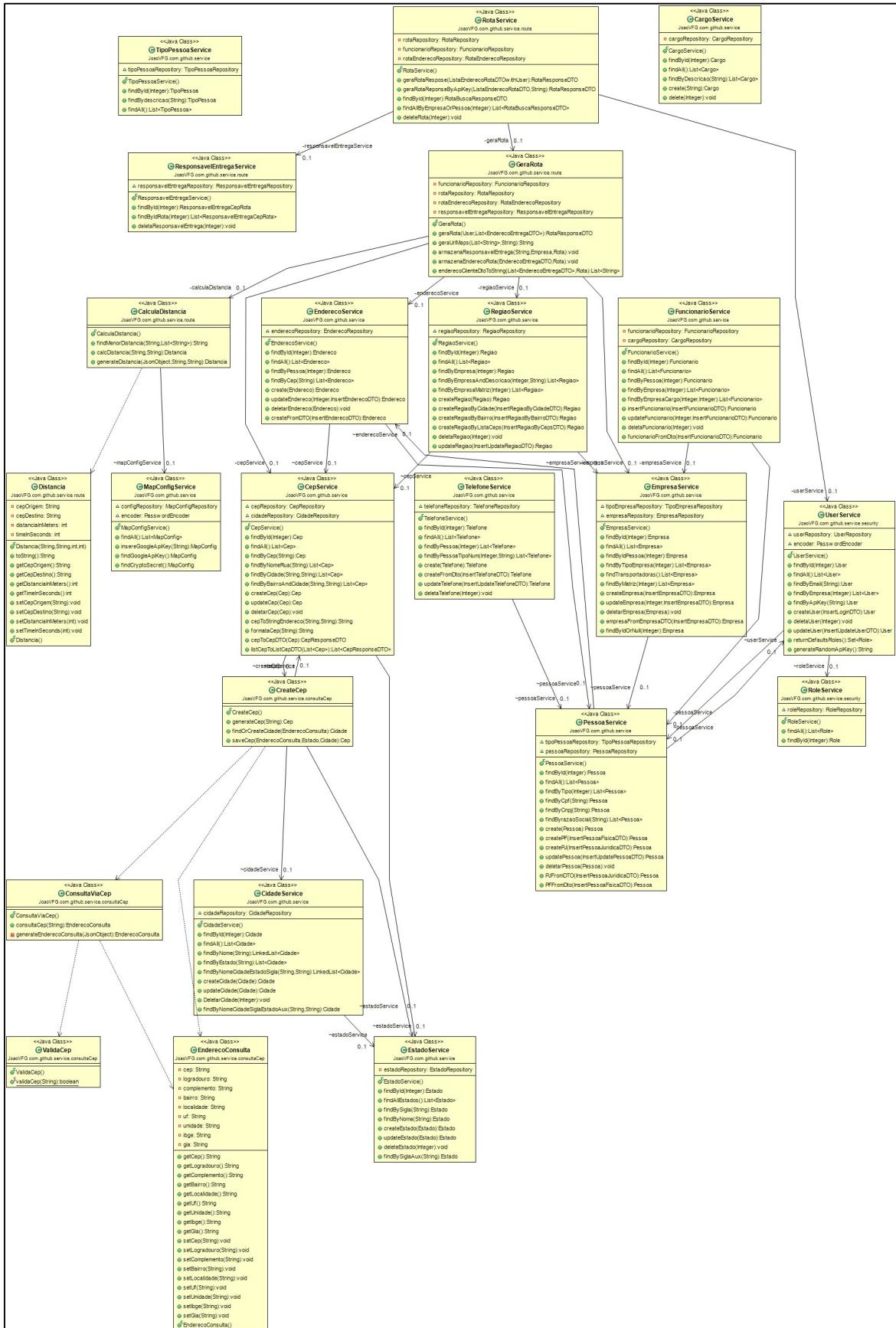
Fonte: Autor (2018)

Figura 18. Diagrama de Classes: Repositórios.



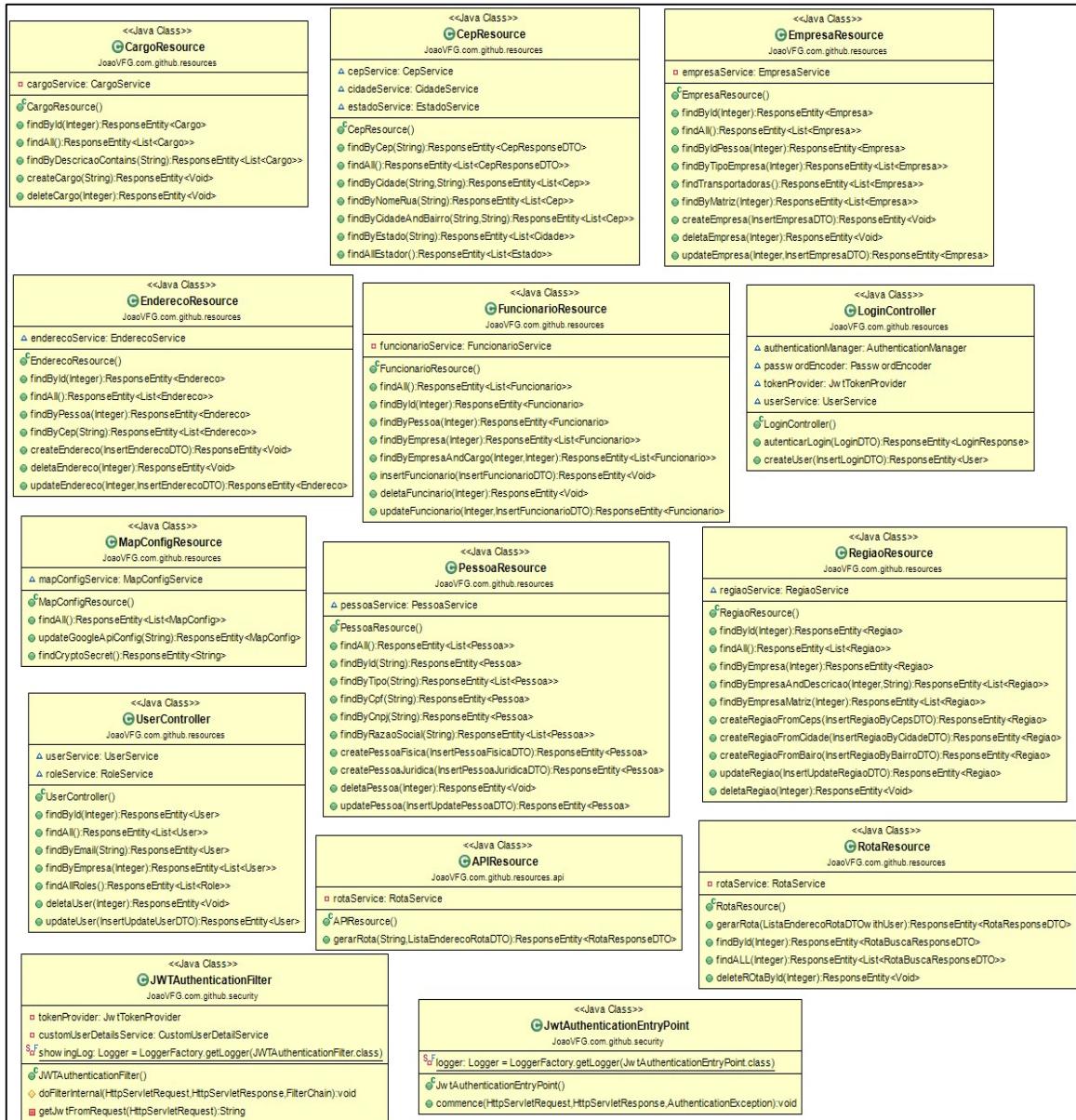
Fonte: Autor (2018)

Figura 19. Diagrama de Classes: Serviços.



Fonte: Autor (2018)

Figura 20. Diagrama de Classes: Controladores e Recursos de Autenticação.



Fonte: Autor (2018)

3.3.3. Exemplificação de Funcionamento do Back-end

Toda requisição realizada ao back-end é direcionada aos Controladores, que são os responsáveis por receber as requisições e repassá-las a camada de serviço, ou caso a requisição seja inválida, o próprio Controller envia uma mensagem de erro para a View. Pelo projeto ser baseado em Spring, deve ser informado as classes que são controladores e suas respectivas URLs de acesso. A Figura 21 exemplifica a forma na qual é realizada essa parametrização.

Figura 21. Definição de Controlador e URL.

```
@RestController
@RequestMapping(value="/pessoa")
public class PessoaResource {
```

Fonte: Autor (2018)

A anotação `@RestController` tem a função de identificar que a Classe é um Controlador e a anotação `@RequestMapping()` informa o Spring que é necessária uma URL para acessar a Classe. A URL é informada dentro do atributo `value`. Devida a utilização do Spring Boot no projeto, não é necessário nenhuma outra configuração ou parametrização, o Spring Boot mapeia todos os arquivos do projeto e com base nessas anotações já atribui as funções destas Classes.

Anotar as Classes apenas define o caminho para a acessá-las, mas para utilizar suas funcionalidades os métodos também devem ser anotados com `@RequestMapping`, como mostrado na Figura 22.

Figura 22. Definição de Acesso ao Método usando `@RequestMapping`.

```
@RequestMapping(value="/buscapessoa/id/{id}", method = RequestMethod.GET)
public ResponseEntity<Pessoa> findById(@PathVariable String id){
    Pessoa pessoa = pessoaService.findById(Integer.parseInt(id));
    return ResponseEntity.ok().body(pessoa);
}
```

Fonte: Autor (2018)

A anotação de um método com `@RequestMapping` necessita de outros atributos além do valor da URL. É necessário informar o tipo da requisição, identificado pelo atributo `method`, no exemplo é informado ao Spring que para acessar o método `findById` é necessário utilizar a URL ‘pessoa/’ definida para classe, combinada com ‘/buscapessoa/id/{id}’, utilizando o método de requisição do tipo `get` traduzido como pegar ou obter. Existem ainda outros tipos de métodos de Requisições como `post` (enviar ou publicar), `delete` (deletar) e `put` (colocar).

Na assinatura do método `findById` existe a anotação `@PathVariable` utilizada para indicar que deve ser passado ao método, uma variável de nome `id`, a partir da URL da requisição. Na URL essa variável é identificada dentro de colchetes e deve ter o mesmo nome da variável de assinatura do método.

Dentro do método `findById` é criado um objeto do tipo `Pessoa` (definido pela camada Model). Esse objeto receberá o retorno da chamada do Método `findById` da Classe de serviço ‘`PessoaService`’. O objeto `pessoaService` que faz a chamada ao método `findById` da Classe de serviço foi instanciado usando Injeção de Dependências pelo Spring. A Figura 23 ilustra como utilizar essa funcionalidade.

Figura 23. Definição Injeção de Dependências.

```
public class PessoaResource {
    @Autowired
    PessoaService pessoaService;
```

Fonte: Autor (2018)

A anotação `@Autowired` que realiza a injeção de dependência. Nesse caso o objeto de pessoa *Service* não precisa ser Criado e inicializado, ele apenas é chamado, liberando acesso aos seus métodos quando necessário.

As classes de serviço são as responsáveis por realizar a chamadas as Classes da camada de Repositórios (*Repositories*), realizar validações e principalmente é a responsável pelas Regras de Negócio do Software. Para identificar uma Classe de serviço é necessária anotá-la com `@Service` como mostrado na Figura 24.

Figura 24. Definição de Classe de Serviço Utilizando a Anotação `@Service`.

```
@Service
public class PessoaService {
```

Fonte: Autor (2018)

O Controlador de Pessoa, ‘PessoaResource’ em seu método `findById`, realizou chamada ao método `findById`, da Classe *PessoaService*, como foi mostrado na Figura 22.

Dentro da Classe *PessoaService* o método `findById` acessa o método `findById` da Interface *PessoaRepository* da camada de Repositório, conforme apresentado na Figura 25. Novamente é realizada Injeção de Dependência, neste caso gerando um objeto *pessoaRepository* do tipo *PessoaRepository*.

Figura 25. Fragmento da Classe *PessoaService*.

```
@Autowired
PessoaRepository pessoaRepository;

public Pessoa findById(Integer id) {
    Optional<Pessoa> pessoa = Optional.ofNullable(pessoaRepository.buscaPorId(id));

    return pessoa.orElseThrow(() -> new ObjectNotFoundException(
        "Pessoa não encontrado! Id: " + id + ". Tipo: " + Pessoa.class.getName()));
}
```

Fonte: Autor (2018)

As Interfaces de Repositório são as responsáveis por realizar acesso ao Banco de Dados, também conhecida como camada de Persistência. Essa camada tem a função de efetuar as operações de CRUD: *create* (criar), *read* (ler ou busca), *update* (atualizar) e *delete* (deletar). A Figura 26 apresenta a Interface *PessoaRepository*:

Figura 26. Interface PessoaRepository.

```

@Repository
public interface PessoaRepository extends JpaRepository<Pessoa, Integer> {

    @Transactional(readOnly = true)
    @Query("SELECT pessoa FROM Pessoa pessoa WHERE pessoa.id = :id")
    public Pessoa buscaPorId(@Param("id") Integer id);

    @Transactional(readOnly = true)
    public List<Pessoa> findBytipo(TipoPessoa tipoPessoa);

    @Transactional(readOnly = true)
    public Pessoa findBycpf(String cpf);

    @Transactional(readOnly = true)
    public Pessoa findBycnpj(String cnpj);

    @Transactional(readOnly = true)
    public List<Pessoa> findByrazaoSocialContains(String razaoSocial);

}

```

Fonte: Autor (2018)

Os repositórios são definidos pela anotação `@Repository` que são as Interfaces que estendem `JpaRepository`. Para estender essa classe é necessário que indicar a qual entidade é referenciada pela Interface, no exemplo a classe referenciada é a Pessoa, e o seu Identificador é do Tipo Integer, conforme na Figura 26, o fragmento ‘extends `JpaRepository<Pessoa, Integer>`’.

Mesmo sendo Interface, quando utilizada a anotação de `@Repository` junto com estender `JpaRepository` a Classe `PessoaRepository` tem suas funções implementadas em tempo de execução pelo próprio Spring, por isso que é possível utilizar seus métodos, sem nenhum tipo de implementação desenvolvida. `JpaRepository` já oferece por padrão, alguns métodos implementados para operações básicas.

Caso exista necessidade de implementar outros métodos é necessário que sejam implementados nos Re却itórios podendo ser criados com base em consultas JPQL, como por exemplo, o método ‘`findById`’ apresentado na Figura 26 ou por assinatura de método como no método ‘`findBytipo`’ também apresentado na Figura 26.

As entidades são definidas pela anotação `@Entity`, são as classes responsáveis por representarem as tabelas do Banco de Dados em objetos. Elas são a denominadas camada *Model* em português Modelo por conta dessa representação. A Figura 27 mostra um fragmento da Classe Pessoa.

Figura 27. Entidade Pessoa.

```

@getter
@Setter
@NoArgsConstructor
@Entity
public class Pessoa implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @ManyToOne
    @JoinColumn(name = "TIPO_ID")
    private TipoPessoa tipo;

    private String nome;

    private String razaoSocial;
}

```

Fonte: Autor (2018)

A anotação `@Entity` indica ao Spring que a Classe é uma Entidade. As anotações `@Getter` (gera os métodos `get` para os atributos) `@Setter` (gera os métodos `set` para os atributos) e `@NoArgsConstructor` (gera um construtor sem argumentos) são anotações do *Framework* Lombok.

Outra anotação que pode ser observada é a `@ManyToOne` (um para muitos) que indica o relacionamento da tabela ‘Pessoa’ para a tabela ‘TipoPessoa’. Ainda há as anotações `@OneToOne` (muitos para um) e `@ManyToMany` (muitos para muitos).

Retornando ao tratamento da requisição, após o repositório receber o resultado da consulta, ele envia o retorno ao método da Classe de Serviço. Na classe de Serviço o retorno é inserido dentro de um Objeto *Optional* traduzido para opcional. Objeto que resumidamente, pode se tornar um tipo de Objeto definido, ou receber *Null*, traduzido para nulo. E quando esse objeto é retornado por uma função, caso ele possua um valor definido caso contrário ele pode automaticamente lançar uma exceção, conforme visto na Figura 25.

Na Classe PessoaResource, caso o esse resultado retornado pela Camada de Serviço seja um objeto, a Camada Controller gera um Objeto do tipo *ResponseEntity*, traduzido para entidade de resposta, cujo corpo será composto pelo objeto retornado pela Camada de Serviço. O Objeto *ResponseEntity*, retorna um JSON em seu corpo. O resultado será enviado para o aplicativo, que faz a representa a camada View. Na camada View, o JSON retornado será interpretado e apresentado ao usuário.

3.3.4. Exemplificação de Funcionamento do *Front-end*

O front-end do Projeto funciona baseado na interação com o usuário. Quando ele clica em um botão, ou clica em um item para detalhamento.

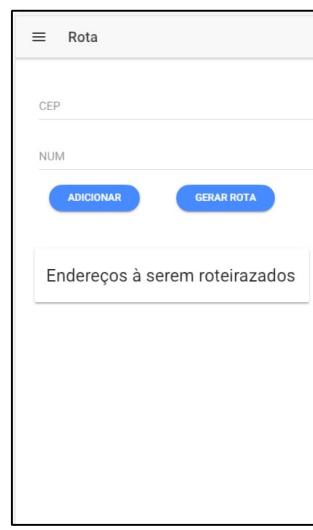
Para uma breve explicação do funcionamento do front-end, utilizaremos como exemplo a função de criação de rotas. Para poder utilizar essa função deve-se acessar a Página de criação de Rotas. No Ionic Página é composta por 4 arquivos:

- a. HTML: é a tela, na qual o usuário vê e interage;
- b. SCSS: arquivo que contém as propriedades gráficas da tela, como definição de posicionamento dos objetos, cor de Fonts;
- c. Módulo : arquivo onde são injetadas as dependências da Página;
- d. Controlador da Página: responsável por captar os eventos, executar ações e retornar dados ao usuário.

As Páginas do Ionic podem ser geradas automaticamente utilizando o Ionic Cli, tendo ele instalado basta executar a seguinte instrução utilizando um prompt de comando: `ionic g page ‘nome da página’`.

Dada a explicação do conceito de Página dentro do Ionic , pode ser analisada a estrutura HTML da Página de geração de rotas. Antes de ser apresentado o código HTML, a Figura 28 mostra a Página, sem ter sofrido interações com o usuário.

Figura 28. Página de Geração de Rotas.



Fonte: Autor (2018)

É possível observar alguns itens que são carregados quando a Página é acessada. A Figura 25 apresenta o código cabeçalho desta Página. Nesse cabeçalho é apresentado o nome da Página e inserido o botão para acessar o menu lateral.

Figura 29. Cabeçalho da Página de Geração de Rotas.



```
<ion-header>
  <ion-navbar>
    <button ion-button menuToggle>
      <ion-icon name="menu"></ion-icon>
    </button>
    <ion-title>Rota</ion-title>
  </ion-navbar>
</ion-header>
```

Fonte: Autor (2018)

É possível também ser observado na Figura 28, alguns outros itens. Dois campos para inserir informações, no caso o cep para entrega e o número do cep, e também dois botões, um para adicionar o endereço digitado a lista de endereços a serem roteirizados e outro para solicitar a criação da rota. A Figura 30 apresenta o código HTML referente a essas funcionalidades.

Figura 30. Fragmento da Página de Geração de Rotas.



```
<ion-item *ngIf="!rotaResponse?.rota">
  <ion-label floating>CEP</ion-label>
  <ion-input [(ngModel)]="enderecoEntrega.cep" name="enderecoCep" type="text"></ion-input>
</ion-item>
<ion-item *ngIf="!rotaResponse?.rota">
  <ion-label floating>NUM</ion-label>
  <ion-input [(ngModel)]="enderecoEntrega.numeroLogradouro" name="enderecoNum" type="text"></ion-input>
</ion-item>

<button *ngIf="!rotaResponse?.rota" name="btnAdicionar"
  ion-button round medium (click)="validaCep()">Adicionar</button>
<button *ngIf="!rotaResponse?.rota" name="btnGerar"
  ion-button round medium (click)="gerarRota()">Gerar Rota</button>
```

Fonte: Autor (2018)

As duas primeiras *Tags* nomeadas ‘ion-item’ é a união do nome do campo (nomeado como ‘ion-lebel’) com o campo de inserção de texto (nomeado ion-input). Dentro de ion-input, é utilizada a diretiva ngModel que tem a função de vincular o campo de input com uma variável declarada no Controlador da Página, os campos ion-input devem receber um nome e também o tipo de dado. Outro ponto a ser notado é a utilização da diretiva *ngIf que na posição que está no código tem a função de mostrar ou não o objeto ion-item. Nesse caso, quando uma rota é gerada, esses dois campos de inserção são ocultados, já que não tem mais utilidade no momento.

Abaixo dos campos de inserção há dois botões, um para inserir o endereço digitado na lista de endereços a ser roteirizada, e o outro serve para enviar a lista de endereços para o back-end e esperar a resposta com a rota gerada. Para que essas ações sejam executadas, é utilizada a propriedade (click) indicando um método do controlador da Página. Outro

aspecto a ser notado é a utilização da diretiva *ngIf, para esconder os botões após a rota ser gerada.

O Botão que adiciona o endereço efetua chamada ao método ‘validaCep’ que efetua uma requisição ao back-end solicitando o CEP que foi digitado, caso essa resposta seja positiva, esse cep é adicionado à lista de endereços para roteirização. Esse processo é mostrado com o fragmento do código do controlador da Página de Geração de Rotas apresentada na Figura 31.

Figura 31. Fragmento de Código, Controlador Pagina de Geração de Rotas.

```

validaCep() {
  this.cepService.findByCep(this.enderecoEntrega.cep)
    .subscribe((response) => {
      this.addToList();
    }, error => {
      console.log(error);
    })
}

addToList() {
  this.listaEndereco.waypoints.push({
    cep: this.enderecoEntrega.cep,
    numeroLogradouro: this.enderecoEntrega.numeroLogradouro
  });
  this.enderecoEntrega.cep = '';
  this.enderecoEntrega.numeroLogradouro = '';
}

```

Fonte: Autor (2018)

O método Valida Cep, executa o método ‘findByCep’ de cepService Figura 32, esse método irá realizar uma requisição ao back-end e receber a resposta. Caso seja um erro, um alerta será mostrado na Tela. Se a requisição obtiver uma resposta, o método addToList será acionado, onde irá inserir esse endereço a Lista a ser roteirizada, e também irá limpar o valor dos campos de cep e número do endereço.

Figura 32. Fragmento da Classe CepService.

```

@Injectable()
export class CepService{

  constructor(public http : HttpClient){

  }

  findByCep(cep : String) : Observable<cep>{
    return this.http.get<cep>(`${API_CONFIG.baseUrl}/ceps/buscacep/` + cep);
  }
}

```

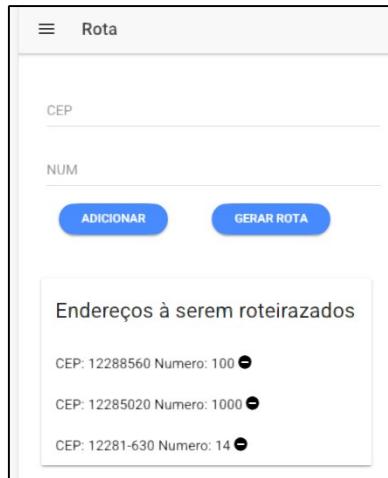
Fonte: Autor (2018)

Devem ser destacados alguns Pontos da Classe de serviço CepService. Toda Classe que será utilizada em algum outro ponto da aplicação deve usar a Anotação `@Injectable`, para que ela possa ser injetada e usada em outras Classes. Todos os recursos que serão utilizados por uma classe, devem ser injetados no construtor da classe, em CepService, por

exemplo, existe a injeção de um objeto HttpClient, o cliente padrão para requisições HTTP. Os métodos das classes de serviço devem retornar a execução de uma requisição HTTP de qualquer tipo, passando como parâmetro obrigatório a URL e se necessário objetos no corpo da requisição.

Realizada a adição de alguns endereços para entrega o Card com os pontos de entrega a serem roteirizados será preenchido como apresentado na Figura 33.

Figura 33. Tela para Geração e Rotas, com Endereços Inseridos.



Fonte: Autor (2018)

Para essa listagem de endereços ser apresentada é utilizada a diretiva *ngFor do Angular, responsável por gerar dinamicamente uma lista de itens que podem ter as suas informações acessadas no HTML da Página mostrado na Figura 34, com base em um vetor declarado no Controlador da Pagina.

Figura 34. Tela para Geração e Rotas, com Endereços Inseridos.

```
<ion-card *ngIf="!rotaResponse?.rota">
  <ion-item>
    <ion-card-title>Endereços à serem roteirizados</ion-card-title>
  </ion-item>
  <ion-list>
    <ion-item *ngFor="let end of listaEndereco.waypoints">
      CEP: {{end.cep}} Numero: {{end.numeroLogradouro}}
      <ion-icon name="remove-circle"
        (click)="removeOfList(end.cep, end.numeroLogradouro)"></ion-icon>
    </ion-item>
  </ion-list>
</ion-card>
```

Fonte: Autor (2018)

Os campos do objeto que está sendo percorrido é acessado através do denominado ‘binding de atributo’ que pode ser entendido como vínculo do elemento dentro da página com uma variável, nesse caso é a variável gerada dinamicamente pelo *ngFor.

Outro elemento a ser destacado é o ícone de remoção que tem função de remover o endereço selecionado da Lista de Endereços, esse ícone quando clicado, aciona o método removeOfList. Esse método busca a posição no vetor que o endereço encontra-se, e irá removê-lo do vetor. O fragmento de código que realiza essa função está apresentado na Figura 35.

Figura 35. Método para Remoção de Endereço da Lista.

```
removeOfList(cepBusca: string, numLogradouro: string) {
  let waypoints: enderecoEntregaDTO = {
    cep: cepBusca,
    numeroLogradouro: numLogradouro
  }

  let index: number = this.listaEndereco.waypoints.findIndex
    (e => e.cep === cepBusca && e.numeroLogradouro === numLogradouro);
  this.listaEndereco.waypoints.splice(index, 1);
}
```

Fonte: Autor (2018)

Ao final da inserção dos endereços de entrega, deverá ser clicado no botão de geração de rotas. A tela será alterada para apresentar os dados da Rota gerada conforme são mostrados na Figura 36.

Figura 36. Tela de Geração de Rotas após executada a Geração.



Fonte: Autor (2018)

Os campos para inserção de endereços são ocultados, assim como a lista de endereços para ser roteirizada. É apresentado um Card, com o botão para abrir a rota no Maps, a lista de endereços onde deverão ser realizadas as entregas, e por fim, caso a

empresa use a parametrização por regiões, será mostrado a relação de entregas que deverão ser realizadas por outras filiais. O método responsável por realizar a solicitação para a criação de rotas é mostrado na Figura 37.

Figura 37. Tela de Geração de Rotas após executada a Geração.

```
gerarRota() {
    let loading = this.loadingService.presentLoading();
    this.rotaService.geraRotaJson(this.listaEndereco)
        .subscribe(response => {
            this.rotaResponse = JSON.parse(response.body);
            loading.dismiss();
        },
        error => {
            loading.dismiss();
            console.log(error);
        })
}
```

Fonte: Autor (2018)

Nesse fragmento de código, é criado um objeto de *Loading* que apresentará uma animação de carregamento, até que seja retornada alguma resposta. Se a resposta for de sucesso, o objeto rotaResponse, irá receber o corpo da resposta, que contém os dados da rota, a URL, os endereços onde a entrega será efetuada, e os endereços atendidos por outras filiais. E esse objeto rotaResponse que é apresentado na tela ao usuário.

3.4. Algoritmo de Roteirização

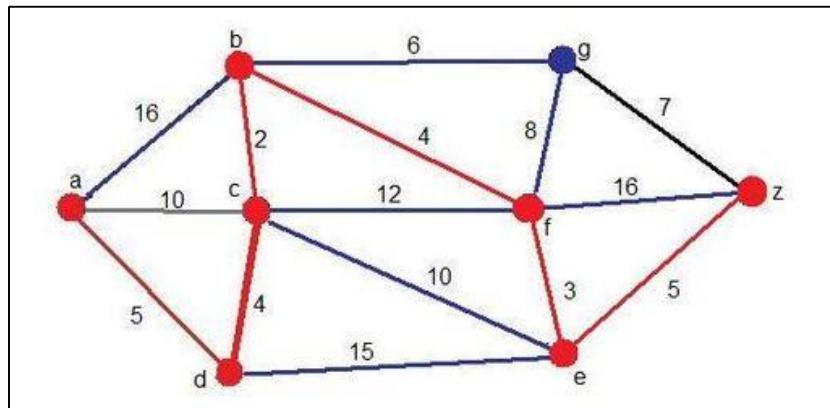
O algoritmo de roteirização desenvolvido no Projeto é uma implementação do algoritmo de Dijkstra com algumas alterações.

O algoritmo de Dijkstra é aplicado para encontrar o caminho mais curto em um Grafo dirigido (PERES, 2016).

Esse algoritmo funciona basicamente da seguinte maneira: Em um determinado nó/vértice de um grafo, o algoritmo deverá localizar o caminho de menor custo (o caminho com a menor distância) entre esse vértice e todos os outros vértices do Grafo, utilizando o valor de peso/custo das arestas. No entanto, também pode ser utilizado para encontrar o caminho mais curto de um determinado vértice para um vértice destino, encerrando a execução do algoritmo quando o caminho mais curto entre os vértices tiver sido encontrado.

A Figura 38 apresenta um exemplo de menor caminho do vértice A até o vértice Z, em um determinado grafo, utilizando o algoritmo de Dijkstra:

Figura 38. Exemplo do Resultado da Aplicação do Algoritmo de Dijkstra.



Fonte: Autor (2018)

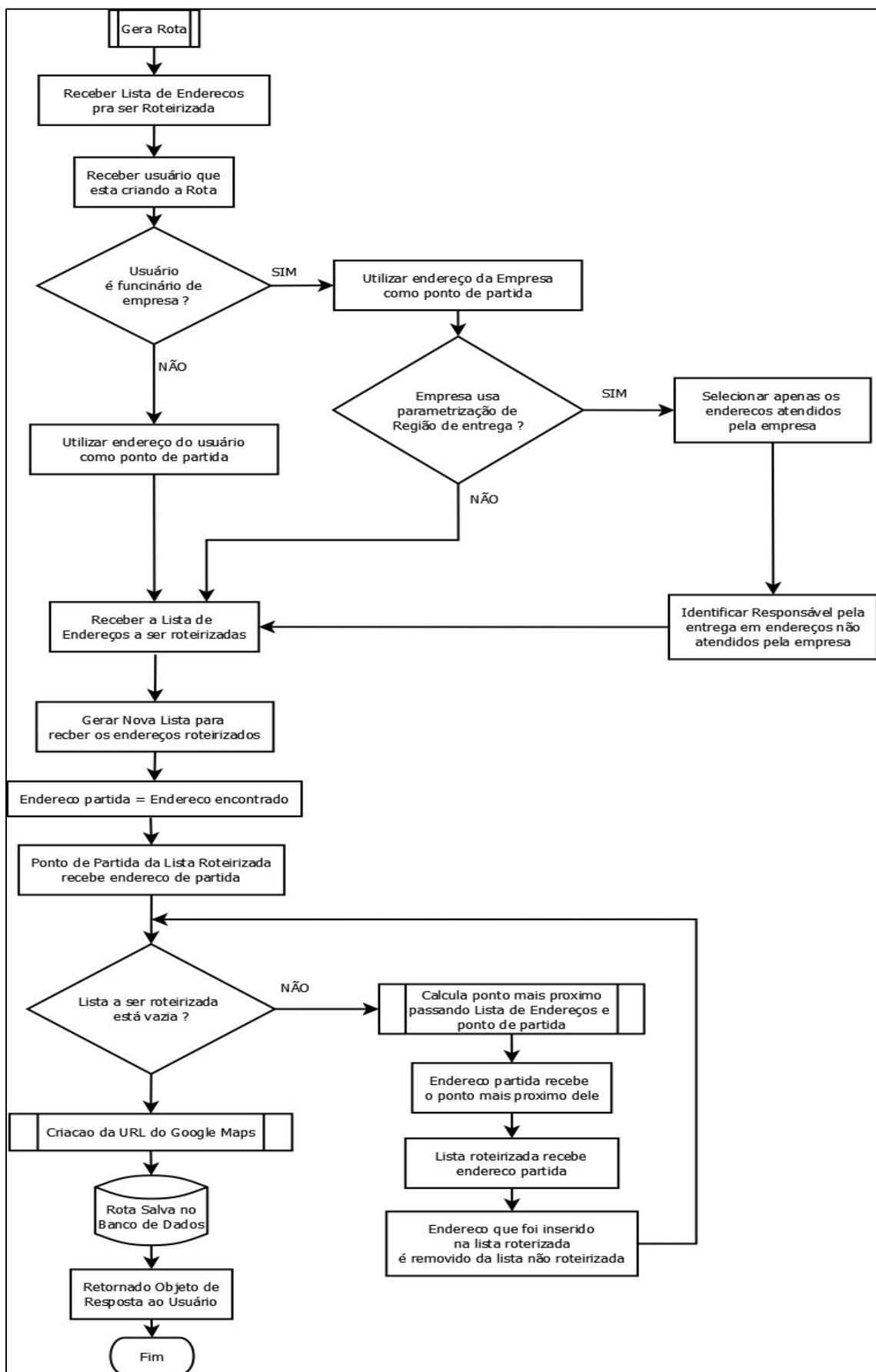
No exemplo o menor caminho entre o vértice A e Z é a sequencia A-D-C-B-F-E-Z.

Esse caminho foi determinado, baseando-se nos valores das arestas. A partir do ponto A o mais próximo dele é o ponto D. A partir do ponto D o mais próximo dele é o ponto C. A partir do ponto C o mais próximo dele é o ponto B. E assim sucessivamente até formar-se o caminho mais rápido deste grafo, entre o ponto A e o ponto Z.

Para o Projeto, será utilizada a ideia principal envolvida no algoritmo de Dijkstra, que é identificar o ponto mais próximo ao ponto atual. A Figura 39, apresenta o Fluxograma do algoritmo utilizado no Projeto. Este fluxograma apresenta os passos básicos para a geração da Rota, ele não representa todas as linhas de código que foram escritas no Projeto.

Devido a grande quantidade de linhas de código envolvida no Projeto, um fluxograma ou pseudocódigo completo com todas as iterações e linhas ficaria muito extenso. Portanto foi desenvolvido essa versão que retrata o conceito geral dos passos procedimentos envolvidos no código fonte.

Figura 39. Tela de Geração de Rotas após executada a Geração.



Fonte: Autor (2018)

O Fluxograma retrata do funcionamento do método geraRota da Classe GeraRota, Figura 40. Esse método recebe como parâmetros o Usuário que está gerando a Rota e um DTO (*Data transfer Object*) com lista de endereços á serem roteirizados (a lista é composta por objetos cujos atributos são uma String para o CEP e uma String para o Número). Esse método está anotado com a `@Transactional` pois as informações são armazenadas as poucos no Banco de Dados, no entanto as Entidades do Banco de Dados que estão envolvidas não ficam em Estado de *Locked*, traduzido como travada.

Figura 40. Método geraRota da Classe Gera Rota.

```
@Transactional()
public RotaResponseDTO geraRota(User user, List<EnderecoEntregaDTO> enderecoEntregaDTOS) {
```

Fonte: Autor (2018)

Utilizar transação foi escolhido para que não tivessem que ser gerados inúmeros objetos em tempo de execução para armazenar os dados que depois iriam ser inseridos no Banco de Dados. Os Dados como os endereços e responsáveis por entregar em endereços que não fazem parte da região de entrega da empresa, são inseridas em tempo de execução. E usando a transação, caso ocorra algum problema no processo de gerar a rota, as alterações são revertidas.

Antes da rota começar a ser gerada, uma rota é salva no Banco de Dados para poder serem armazenadas as suas respectivas informações. Em seguida, são realizadas uma série de verificações. Primeiro Identificar se o usuário é Pessoa Física ou Jurídica. Caso seja jurídica é buscado a empresa dessa pessoa. Caso seja Física, é buscado se esse usuário é funcionário de alguma empresa. Caso ele seja funcionário, a empresa que ele possui vínculo será retornada. Essas verificações podem ser vistas na Figura 41.

Figura 41. Busca de Empresa Dentro de GeraRota.

```
Empresa empresa = new Empresa();

/**
 * objeto de Endereco - INICIALIZAÇÃO Referente ao endereco inicial -> se for
 * pessoa usara seu endereco proprio -> se for funcionario usará endereco da
 * empresa
 */
Endereco endereco = new Endereco();
/**
 * Verifica se a pessoa é fisica ou juridica
 */
if (user.getPessoa().getTipo().getId() == 1) {
    /**
     * Verifica se a pessoa é funcionario de alguma empresa Se for, ira trazer de
     * qual empresa a pessoa é funcionaria
     */
    if (!funcionario == null) {
        empresa = empresaService.findById(funcionario.getEmpresa().getId());
        rota.setEmpresa(empresa);
    }
    /**
     * Se for Juridica atribui ao objeto empresa a empresa vinculada a pessoa do
     * usuário
     */
} else {
    empresa = empresaService.findByIdPessoa(user.getPessoa().getId());
}
```

Fonte: Autor (2018)

Realizar essa busca é necessária para verificar, caso o usuário esteja vinculado a uma Empresa, se essa empresa utiliza a parametrização de regiões de entrega.

Caso a empresa possua parametrização de regiões, será chamado o método buscaResponsavelEntrega que tem por objetivo identificar se todos os endereços à serem roteirizados estão dentro da região de entrega. Os que não estiverem, tem sua filial responsável retornada, e é armazenado o endereço e seu respectivo responsável. Por final a Lista com os Endereços que deverão ser roteirizados é retornada, eliminando os pontos que não são atendidos pela empresa.

Em seguida é definido o endereço de partida. Caso o usuário não tenha vínculo com nenhum empresa, o ponto de partida será seu próprio endereço cadastrado.

Essas validações e chamadas podem ser observadas da Figura 42.

Figura 42. Definição das Lista a ser Roteirizada e Endereço de Partida.

```
if (!(empresa.getId() == null)) {
    // Regiao de atuação de empresa
    Regiao regiao = regiaoService.findByEmpresa(empresa.getId());
    // verifica se a empresa tem região de atuação
    if (!(regiao == null)) {
        listasEnderecos = buscaResponsavelEntrega(empresa, enderecoEntregaDTOs, regiao, rota);
        rotaResponseDTO.setResponsavel(listasEnderecos.getResponsavelRegiaoDTO());
        enderecoEntregaDTOs = listasEnderecos.getEnderecoEntregaDTOs();
    }
    endereco = enderecoService.findByPessoa(empresa.getPessoa().getId());
}
// Atribui o endereço da pessoa do usuário ao endereço de partida
else {
    endereco = enderecoService.findByPessoa(user.getPessoa().getId());
}
```

Fonte: Autor (2018)

Após a lista de endereços que serão roteirizados ter sido definida, junto com o ponto de partida, o é gerado alguns objetos necessários para a roteirização, Figura 43.

Figura 43. Criação de Objetos Auxiliares Necessários à Roteirização.

```
EnderecoEntregaDTO enderecoPartida = new EnderecoEntregaDTO(endereco.getCep().getCep(),
    endereco.getNumeroLogradouro().toString());

EnderecoEntregaDTO enderecoSalvo = enderecoPartida;
enderecoSalvo.setEnderecoString(
    cepService.cepToStringEndereco(enderecoSalvo.getCep(), enderecoSalvo.getNumeroLogradouro()));

rotaResponseDTO.setEnderecoOrigem(
    cepService.cepToStringEndereco(enderecoPartida.getCep(), enderecoPartida.getNumeroLogradouro()));

List<String> listaRoteirizada = new ArrayList<String>();
enderecoEntregaDTOs = enderecoClienteDtoToString(enderecoEntregaDTOs);
```

Fonte: Autor (2018)

O objeto enderecoPartida irá armazenar o ponto de partida da rota, que foi buscado anteriormente, e enderecoSalvo irá armazenar esse mesmo ponto. Dentro do objeto de

resposta do método é inserido o ponto de partida. Por fim, são gerados uma de *String* para os endereços roteirizados e também gerado os endereços em formato de *String*.

Após esses procedimento, é iniciado o processo de roteirização, Figura 44.

Figura 44. Looping de Roteirização.

```

while (!enderecoEntregaDTOs.isEmpty()) {
    // recebe o ponto mais proximo de enderecoString
    enderecoPartida = calculaDistancia.findMenorDistancia(
        cepService.cepToStringEndereco(enderecoPartida.getCep(), enderecoPartida.getNumeroLogradouro()),
        enderecoEntregaDTOs);
    listaRoteirizada.add(enderecoPartida.getEnderecoString());
    armazenaEnderecoRota(enderecoPartida, rota);
    enderecoEntregaDTOs.remove(enderecoPartida);
}

rotaResponseDTO.setWaypoints(listaRoteirizada);

```

Fonte: Autor (2018)

Enquanto o objeto enderecoEntregaDto não estiver vazio, o *looping* será realizado. O objeto enderecoPartida receberá o endereço de retorno do método findMenorDistancia da Classe CalculaDistancia. Esse ponto retornado, é o mais próximo do ponto de partida. Em seguida, o valor atual de enderecoPartida, que é o ponto mais próximo encontrado do valor antigo da mesma variável enderecoPartida, será inserida na listaRoteirizada. Esse ponto encontrado será armazenado no banco de dados por meio do método armazenaEnderecoRota. E por fim, o enderecoPartida é removido da lista que está sendo roteirizada enderecoEntregaDTO. Assim ao final desse looping, todos os endereços foram roteirizados no objeto listaRoteirizada.

Na Figura 45, é apresentado o método findMenorDistancia da classe CalculaDistancia.

Figura 45. Método findMenorDistancia da Classe CalculaDistancia.

```

public EnderecoEntregaDTO findMenorDistancia(String cepOrigem, List<EnderecoEntregaDTO> enderecosCliente) {

    List<Distancia> distanciasDoInicio = new ArrayList<Distancia>();

    for (EnderecoEntregaDTO e: enderecosCliente) {
        Distancia d = calcDistancia(cepOrigem, e);
        distanciasDoInicio.add(d);
    }
    distanciasDoInicio.sort(Comparator.comparing(Distancia::getDistanciaInMeters));

    return distanciasDoInicio.get(0).getEnderecoEntregaDTO();
}

```

Fonte: Autor (2018)

Esse método é responsável por efetuar o calculo do endereço Inicial, o cepOrigem aos endereços listados. Primeiro é gerado uma *Array* de distâncias. Dentro do *Looping for* é realizado a criação de objetos de distância usando o método calcDistancia, responsável

por efetuar as requisições ao GoogleMaps e identificar as distâncias entre dois pontos. Efetuado o cálculo de todas as distâncias, é retornado o ponto com a menor distância do ponto de origem, utilizando o método *comparing* da classe *Comparator*, esse método serve para ordenar um *Array* baseado em um parâmetro passado. No caso foi o método *getDistanciaInMeter*, que retorna a distância em metros de um objeto *distancia*. No final, a primeira posição do *Array* é retornado a classe *GeraRota*.

A seguir, Figura 46, é apresentado o método *findMenorDistancia*. O método responsável por realizar as requisições a API DistanceMatrix do Google.

Figura 46. Método calcDistancia da Classe CalculaDistancia.

```
public Distancia calcDistancia(String cepOrigem, EnderecoEntregaDTO enderecoEntregaDTO) {
    JsonObject objectResponse = null;

    try {
        CloseableHttpClient httpClient = HttpClientBuilder.create().build();
        String url = "https://maps.googleapis.com/maps/api/distancematrix/json?" + "origins="
            + URLEncoder.encode(cepOrigem, "UTF-8") + "&destinations="
            + URLEncoder.encode(enderecoEntregaDTO.getEnderecoString(), "UTF-8") + "&travelmode=driving"
            + "&key=" + mapConfigService.findGoogleApiKey().getValue();
        HttpGet httpGet = new HttpGet(url);
        HttpResponse httpResponse = httpClient.execute(httpGet);

        HttpEntity entity = httpResponse.getEntity();
        objectResponse = Json.createReader(entity.getContent()).readObject();
    } catch (Exception e) {
        throw new RuntimeException(e);
    }

    return generateDistancia(objectResponse, cepOrigem, enderecoEntregaDTO);
}
```

Fonte: Autor (2018)

Essa classe utiliza o objeto *httpClinte* do tipo *CloseableHttpClient* para executar a requisição a uma determinada URL passada ,que foi definida na String *url* e inserida como parâmetro no objeto *httpGet*. O objeto *httpResponse* recebe a execução do método *execute* de *httpClient*, passando o objeto *httpGet* como parâmetro. Por fim o retorno é lido dentro no objeto *objectResponse* e o retorno é a geração de um Objeto *Distância* pelo método *generateDistancia*.

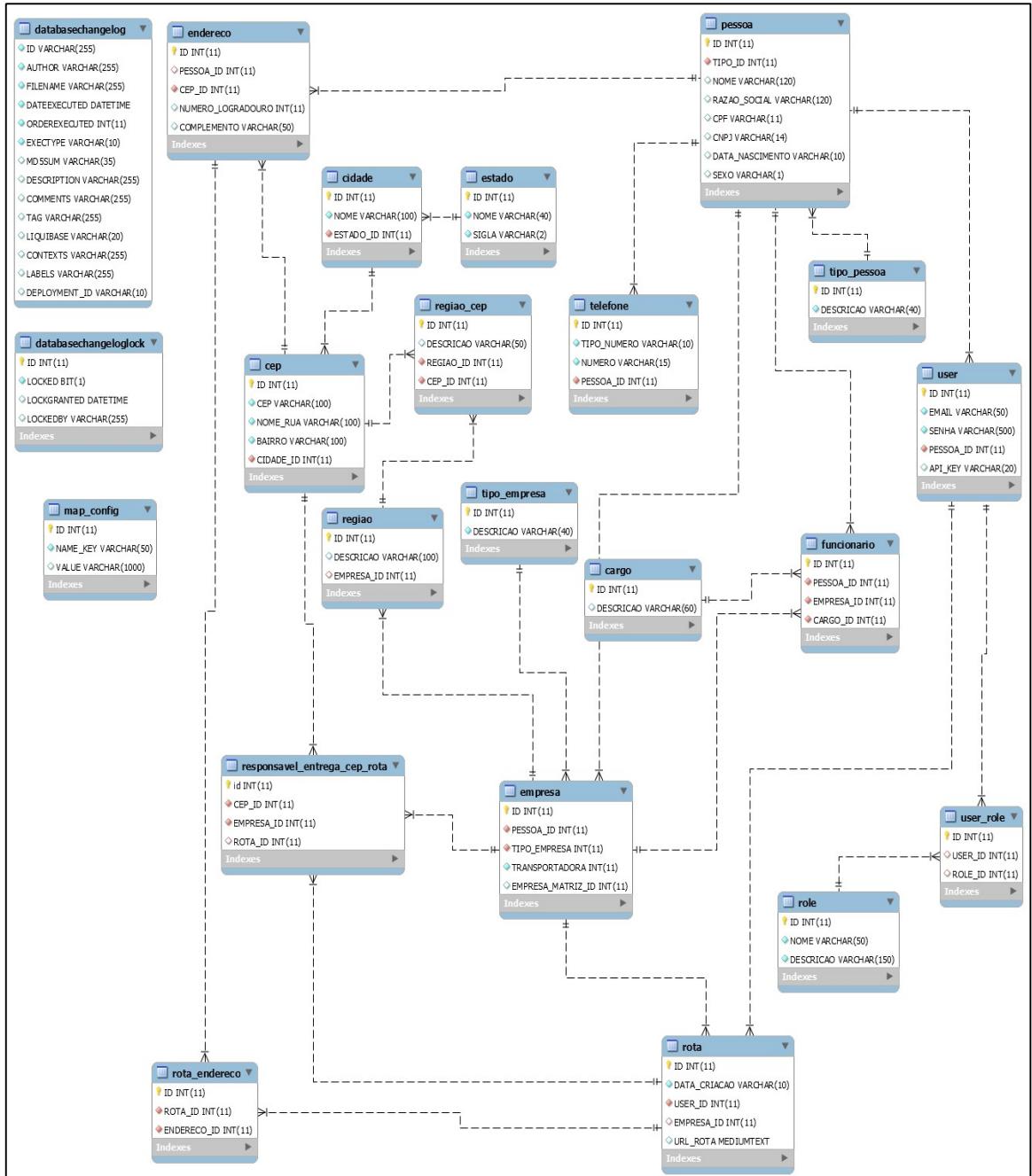
3.5. Modelagem e Gestão dos Dados

No presente Subcapítulo será apresentado a modelagem do Banco de Dados, contemplando o Modelo Entidade Relacionamento, Dicionário de Dados detalhando as tabelas e campos do Banco de dados, bem como a utilização do Liquibase no Projeto.

3.5.1. Modelo de Entidade Relacionamento

O Modelo Entidade Relacionamento representado na Figura 47 mostra as Tabelas do Banco De Dados, seus atributos e seus Relacionamentos.

Figura 47. Modelo Entidade Relacionamento.



Fonte: Autor (2018)

3.5.2. Dicionário de Dados

O Dicionário de Dados tem a função de descrever de forma Objetiva as Tabelas, os seus Atributos, seus Tipos e a função de cada campo. As siglas FK vêm de Foreign Key que significa Chave Estrangeira e PK vem de Primary Key que significa Chave Primária.

As Tabelas de 33 à 51 apresentam o Dicionário de Dados do Projeto. Cada uma das Tabelas representada uma Entidade do Banco de Dados.

Tabela 33. Dicionário de Dados: Tabela Cargo.

Tabela:cargo				
Descrição Da Tabela		Tabela para cadastro dos cargos dos funcionários		
Campos				
Nome Do Campo	Tipo do Campo	PK	FK	Comentário
ID	INT (11)	Sim	Não	Campo identificador para Cargo
DESCRICAQO	VARCHAR (60)	Não	Não	Campo para descrever o Cargo

Fonte: Autor (2018)

Tabela 34. Dicionário de Dados: Tabela Cep.

Tabela:cep				
Descrição Da Tabela		Tabela que contem os ceps e a cidade respectiva		
Campos				
Nome Do Campo	Tipo do Campo	PK	FK	Comentário
ID	INT (11)	Sim	Não	Campo identificador para CEP
CEP	VARCHAR (100)	Não	Não	Campo do CEP
NOME_RUA	VARCHAR (100)	Não	Não	Campo do Nome da Rua
BAIRRO	VARCHAR (100)	Não	Não	Campo para o nome do Bairro
CIDADE_ID	INT (11)	Não	Sim	Campo para FK com Cidade(Atributo Id)

Fonte: Autor (2018)

Tabela 35. Dicionário de Dados: Tabela Cidade.

Tabela:cidade				
Descrição Da Tabela		Tabela que contem as cidades e o estado respectivo		
Campos				
Nome Do Campo	Tipo do Campo	PK	FK	Comentário
ID	INT (11)	Sim	Não	Campo identificador para Estado
NOME	VARCHAR (100)	Não	Não	Campo para o Nome do Estado
ESTADO_ID	INT (11)	Não	Sim	Campo para Sigla do Estado

Fonte: Autor (2018)

Tabela 36. Dicionário de Dados: Tabela Empresa.

Tabela:empresa					
Descrição Da Tabela		Tabela de cadastro das empresas			
Campos					
Nome Do Campo	Tipo do Campo	PK	FK	Comentário	
ID	INT (11)	Sim	Não	Campo identificador para Empresa	
PESSOA_ID	INT (11)	Não	Sim	Campo para FK com Pessoa(Atributo Id)	
TIPO_EMPRESA	INT (11)	Não	Sim	Campo para FK com Tipo_empresa(Atributo Id)	
TRANSPORTADORA	INT (11)	Não	Não	Campo para definir Se é transportadora (1 - Sim ; 0 Não)	
EMPRESA_MATRIZ_ID	INT (11)	Não	Não	Campo para referenciar a Empresa Matriz(Ligado a Pessoa)	

Fonte: Autor (2018)

Tabela 37. Dicionário de Dados: Tabela Endereço.

Tabela:endereco					
Descrição Da Tabela		Tabela que contem os endereços das pessoas			
Campos					
Nome Do Campo	Tipo do Campo	PK	FK	Comentário	
ID	INT (11)	Sim	Não	Campo identificador para Endereço	
PESSOA_ID	INT (11)	Não	Sim	Campo para FK com Pessoa(Atributo Id)	
CEP_ID	INT (11)	Não	Sim	Campo para FK com Cep(Atributo Id)	
NUMERO_LOGRADOURO	INT (11)	Não	Não	Campo para armazenar o Número do Endereço	
COMPLEMENTO	VARCHAR (50)	Não	Não	Campo para armazenar o Complemento do Endereço	

Fonte: Autor (2018)

Tabela 38. Dicionário de Dados: Tabela Estado.

Tabela:estado					
Descrição Da Tabela		Tabela que contem estados e Siglas			
Campos					
Nome Do Campo	Tipo do Campo	PK	FK	Comentário	
ID	INT (11)	Sim	Não	Campo identificador para Estado	
NOME	VARCHAR (40)	Não	Não	Campo para armazenar Nome do Estado	
SIGLA	VARCHAR (2)	Não	Não	Campo para armazenar a Sigla do Estado	

Fonte: Autor (2018)

Tabela 39. Dicionário de Dados: Tabela Funcionário.

Tabela:Funcionário				
Descrição Da Tabela	Tabela para cadastro dos funcionários			
Campos				
Nome Do Campo	Tipo do Campo	PK	FK	Comentário
ID	INT (11)	Sim	Não	Campo identificador para Funcionário
PESSOA_ID	INT (11)	Não	Sim	Campo para FK com Pessoa(Atributo Id)
EMPRESA_ID	INT (11)	Não	Sim	Campo para FK com Empresa(Atributo Id)
CARGO_ID	INT (11)	Não	Sim	Campo para FK com Cargo(Atributo Id)

Fonte: Autor (2018)

Tabela 40. Dicionário de Dados: Tabela Map_config.

Tabela:map_config				
Descrição Da Tabela	Tabela para salvar Chaves de configuração			
Campos				
Nome Do Campo	Tipo do Campo	PK	FK	Comentário
ID	INT (11)	Sim	Não	Campo identificador para MAP_CONFIG
NAME_KEY	VARCHAR (50)	Não	Não	Nome da chave de configuração
VALUE	VARCHAR (1000)	Não	Não	Nome do valor da Chave de configuração

Fonte: Autor (2018)

Tabela 41. Dicionário de Dados: Tabela Pessoa.

Tabela:pessoa				
Descrição Da Tabela	Tabela que cadastro das pessoas que usam o sistema			
Campos				
Nome Do Campo	Tipo do Campo	PK	FK	Comentário
ID	INT(11)	Sim	Não	Campo identificador para Pessoa
TIPO_ID	INT(11)	Não	Sim	Campo para FK com Tipo_Pessoa(Atributo Id)
NOME	VARCHAR (120)	Não	Não	Campo para armazenar Nome da Pessoa
RAZAO_SOCIAL	VARCHAR (120)	Não	Não	Campo para armazenar Razão Social da Pessoa
CPF	VARCHAR (11)	Não	Não	Campo para armazenar CPF da Pessoa
CNPJ	VARCHAR (14)	Não	Não	Campo para armazenar CNPJ da Pessoa
DATA_NASCIMENTO	VARCHAR (10)	Não	Não	Campo Data de nascimento da pessoa
SEXO	VARCHAR (1)	Não	Não	Campo para armazenar sexo da Pessoa

Fonte: Autor (2018)

Tabela 42. Dicionário de Dados: Tabela Região.

Tabela:Região					
Descrição Da Tabela		Tabela para cadastro das regiões de entrega das empresas			
Campos					
Nome Do Campo	Tipo do Campo	PK	FK	Comentário	
ID	INT (11)	Sim	Não	Campo identificador para Região	
DESCRICAO	VARCHAR (100)	Não	Não	Descrição da Região(facilitar Identificação)	
EMPRESA_ID	INT (11)	Não	Sim	Campo para FK com Empresa(Atributo Id)	

Fonte: Autor (2018)

Tabela 43. Dicionário de Dados: Tabela Roles.

Tabela:role					
Descrição Da Tabela		TABELA PARA CONTROLE DAS AUTORIZACOES			
Campos					
Nome Do Campo	Tipo do Campo	PK	FK	Comentário	
ID	INT(11)	Sim	Não	Campo identificador para Role	
NOME	VARCHAR(50)	Não	Não	Campo para armazenar Nome da Role	
DESCRICAO	VARCHAR(150)	Não	Não	Campo para armazenar a descrição da Role	

Fonte: Autor (2018)

Tabela 44. Dicionário de Dados: Tabela Telefone.

Tabela:telefone					
Descrição Da Tabela		Tabela que contem os celulares das pessoas			
Campos					
Nome Do Campo	Tipo do Campo	PK	FK	Comentário	
ID	INT (11)	Sim	Não	Campo identificador para	
TIPO_NUMERO	VARCHAR (10)	Não	Não	Campo para identificar se é celular ou fixo	
NUMERO	VARCHAR (15)	Não	Não	Campo para armazenar o número do Celular	
PESSOA_ID	INT (11)	Não	Sim	Campo para FK com Pessoa(Atributo Id)	

Fonte: Autor (2018)

Tabela 45. Dicionário de Dados: Tabela Tipo_Empresa.

Tabela:tipo_empresa				
Descrição Da Tabela		Tabela que para diferenciar matriz de filiais		
Campos				
Nome Do Campo	Tipo do Campo	PK	FK	Comentário
ID	INT (11)	Sim	Não	Campo identificador para
DESCRICAO	VARCHAR (40)	Não	Não	Descrição do Tipo da Empresa

Fonte: Autor (2018)

Tabela 46. Dicionário de Dados: Tabela Tipo_Pessoa.

Tabela:tipo_pessoa				
Descrição Da Tabela		Tabela que contem os tipos das pessoas		
Campos				
Nome Do Campo	Tipo do Campo	PK	FK	Comentário
ID	INT (11)	Sim	Não	Campo identificador para
DESCRICAO	VARCHAR (40)	Não	Não	Descrição do Tipo da Pessoa

Fonte: Autor (2018)

Tabela 47. Dicionário de Dados: Tabela User.

Tabela:user				
Descrição Da Tabela		Tabela para cadastro dos usuários		
Campos				
Nome Do Campo	Tipo do Campo	PK	FK	Comentário
ID	INT(11)	Sim	Não	Campo identificador paraUser
EMAIL	VARCHAR (50)	Não	Não	Campo para armazenar email do usuário
SENHA	VARCHAR (500)	Não	Não	Campo para armazenar senha do usuário
PESSOA_ID	INT(11)	Não	Sim	Campo para FK com Pessoa(Atributo Id)

Fonte: Autor (2018)

Tabela 48. Dicionário de Dados: Tabela User_Role.

Tabela:user_role				
Descrição Da Tabela	Tabela N pra N de User para Role			
Campos				
Nome Do Campo	Tipo do Campo	PK	FK	Comentário
ID	INT (11)	Sim	Não	Campo identificador para
USER_ID	INT (11)	Não	Sim	Campo para FK com User(Atributo Id)
ROLE_ID	INT (11)	Não	Sim	Campo para FK com Role(Atributo Id)

Fonte: Autor (2018)

Tabela 49. Dicionário de Dados: Tabela Rota.

Tabela:Rota				
Descrição Da Tabela	Tabela para armazenamento de Rotas			
Campos				
Nome Do Campo	Tipo do Campo	PK	FK	Comentário
ID	INT (11)	Sim	Não	Campo identificador para
DATA_CRIACAO	VARCHAR (10)	Não	Não	Campo que armazena data de criação da rota
USER_ID	INT (11)	Não	Sim	Campo para FK com user(Atributo Id)
EMPRESA_ID	INT (11)	Não	Sim	Campo para FK com empresa(Atributo Id)
URL_ROTA	MEDIUMTEXT	Não	Não	Campo para armazenar a url do Maps

Fonte: Autor (2018)

Tabela 50. Dicionário de Dados: Tabela Rota_Endereco.

Tabela:rota_endereco				
Descrição Da Tabela	Tabela N pra N de Rota para Endereço			
Campos				
Nome Do Campo	Tipo do Campo	PK	FK	Comentário
ID	INT (11)	Sim	Não	Campo identificador para
ROTA_ID	INT (11)	Não	Sim	Campo para FK com Rota(Atributo Id)
ENDERECO_ID	INT (11)	Não	Sim	Campo para FK com Endereço(Atributo Id)

Fonte: Autor (2018)

Tabela 51. Dicionário de Dados: Tabela Responsavel Entrega Cep Rota.

Tabela:responsvel_entrega_cep_rota				
Descrição Da Tabela	Tabela para armazenar os responsáveis por entregas em endereços que a empresa não entrega			
Campos				
Nome Do Campo	Tipo do Campo	PK	FK	Comentário
ID	INT(11)	Sim	Não	Campo identificador para
CEP_ID	INT(11)	Não	Sim	Campo para FK com CEP(Atributo Id)
EMPRESA_ID	INT(11)	Não	Sim	Campo para FK com empresa(Atributo Id)
ROTA_ID	INT(11)	Não	Sim	Campo para FK com rota(Atributo Id)

Fonte: Autor (2018)

3.5.3. Liquibase

A Biblioteca Liquibase é capaz de executar tanto as instruções DDL como DML. A criação do Banco de Dados do Projeto foi toda realizada através do Liquibase utilizando arquivos XML e SQL.

Para utilizar a Biblioteca, é necessário primeiramente adicionar sua dependência ao arquivo POM.xml. Realizada a adição da dependência é necessário especificar ao SpringBoot a localização do arquivo principal do Liquibase, dentro do arquivo application.properties utilizando a propriedade ‘spring.liquibase.change-log’ conforme apresentado na Figura 48.

Figura 48. Propriedade do Liquibase em application.properties.

```
server.port=${port:8000}
spring.profiles.active=dev
spring.liquibase.change-log=classpath:db/liquibase-changelog.xml
app.jwtExpirationInMs = 604800000
```

Fonte: Autor (2018)

O arquivo principal é denominado dentro da documentação do liquibase como ChangeLogMaster que pode ser entendido como o arquivo responsável por controlar a execução de outros arquivos. Dentro do arquivo liquibase-changelog.xml foi inserido a utilização de diversos arquivos conforme fragmento mostrado na Figura 49.

Figura 49. Fragmento do Arquivo liquibase-changelog.xml.

```
<?xml version="1.0" encoding="UTF-8"?>

<databaseChangeLog
    xmlns="http://www.liquibase.org/xml/ns/dbchangelog"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog
        http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-3.1.xsd">

    <include file="changelog/01-create-estado.xml" relativeToChangelogFile="true"/>
    <include file="changelog/02-insert-estados.xml" relativeToChangelogFile="true"/>
    <include file="changelog/03-create-cidade.xml" relativeToChangelogFile="true"/>
    <include file="changelog/04-create-cep.xml" relativeToChangelogFile="true"/>
    <include file="changelog/05-add-autincrement.xml" relativeToChangelogFile="true"/>
    <include file="changelog/06-add-foreignkey-cidade.xml" relativeToChangelogFile="true"/>
    <include file="changelog/07-add-foreignkey-cep.xml" relativeToChangelogFile="true"/>
    <include file="changelog/08-create-tipopessoa.xml" relativeToChangelogFile="true"/>
    <include file="changelog/09-create-pessoa.xml" relativeToChangelogFile="true"/>
    <include file="changelog/10-add-foreignkey-pessoa.xml" relativeToChangelogFile="true"/>
```

Fonte: Autor (2018)

A Figura 50 apresenta o arquivo ‘01-create-estado.xml’ onde é realizada a criação da Tabela Estado utilizando o formato XML.

Figura 50. Arquivo 01-create-estado.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<databaseChangeLog
    xmlns="http://www.liquibase.org/xml/ns/dbchangelog"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog
        http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-3.1.xsd">

    <changeSet id="01" author="JoaoVFG">

        <createTable tableName="ESTADO"
            remarks="Tabela que contem estados e Siglas">

            <column name="ID" type="int">
                <constraints primaryKey="true" unique="true"/>
            </column>

            <column name="NOME" type="varchar(40)">
                <constraints unique="true" nullable="false" />
            </column>

            <column name="SIGLA" type="varchar(2)">
                <constraints unique="true" nullable="false" />
            </column>

        </createTable>

    </changeSet>

</databaseChangeLog>
```

Fonte: Autor (2018)

Para todo arquivo a ser executado pelo Liquibase deve ser informado o Id do ChangeSet e o autor. Todas as tags à serem utilizadas podem ser encontradas no Site Oficial do Liquibase.

A utilização do Liquibase é de grande importância pois centraliza o controle do Banco de Dados junto ao projeto, facilitando a gestão de mudanças. No desenvolvimento

caso seja realizada alguma alteração nas classes de Entidade, é necessário apenas inserir a instrução que reflita essa alteração no Banco de Dados, dentro de um arquivo XML ou SQL e inseri-lo dentro do Liquibase-changelog.xml essa facilidade faz com que o desenvolvedor não necessite abrir um SGBD para realizar as alterações, economizando tempo.

3.6. Segurança

A Segurança do Projeto ficará sob responsabilidade do módulo de segurança do Spring, unido à utilização de Token JWT.

Para a utilização do SpringSecurity devem ser desenvolvidas algumas Classes. A Primeira é a Classe UserPrincipal que implementa UserDetails e seus métodos, sendo obrigatória para o funcionamento o JWT. Suas funcionalidades são todas focadas em prover as informações de Usuários e suas respectivas Permissões com base no formato de Usuário utilizado na implementação do Projeto. Na Figura 51 é mostrada a definição da Classe e implementação de alguns métodos de UserDetails.

Figura 51. Classe UserPrincipal e Alguns Métodos.

```
public class UserPrincipal implements UserDetails {
    private static final long serialVersionUID = 1L;

    public static UserPrincipal create(User user) {
        List<GrantedAuthority> authorities = user.getRoles().stream().map(role ->
            new SimpleGrantedAuthority(role.getName())).collect(Collectors.toList());
        String name = null;
        if(user.getPessoa().getTipo().getId() == 1) {
            name = user.getPessoa().getRazaoSocial();
        } else {
            name = user.getPessoa().getNome();
        }
        return new UserPrincipal(user.getId(), name, user.getEmail(), user.getSenha(), authorities);
    }

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        return authorities;
    }

    @Override
    public String getPassword() {
        return password;
    }

    @Override
    public String getUsername() {
        return email;
    }
}
```

Fonte: Autor (2018)

A Classe é a JJWTTokenProvider, que tem a função de gerar um Token com base na autenticação informada, além de também validar Tokens recebidos. A seguir é apresentada

na Figura 52 um trecho da Classe `JWTTokenProvider`, com o método responsável por gerar o Token de autenticação.

Figura 52. Método Generate Token de *JWTTokenProvider*.

```
public String generateToken(Authentication authentication) {
    UserPrincipal userPrincipal = (UserPrincipal) authentication.getPrincipal();

    Date now = new Date();
    Date expiryDate = new Date(now.getTime() + jwtExpirationInMs);

    return Jwts.builder()
        .setSubject(Integer.toString(userPrincipal.getId()))
        .setIssuedAt(new Date()).setExpiration(expiryDate)
        .signWith(SignatureAlgorithm.HS512, getMapConfigJWT().getValue())
        .claim("email", userPrincipal.getEmail())
        .claim("idUser", userService.findByEmail(userPrincipal.getEmail()).getId())
        .compact();
}
```

Fonte: O autor

A partir de um objeto do tipo `authentication` será extraído o usuário para manipulação. O método irá retornar um Token com as seguintes informações, Id (Número Identificador) do Usuário, data de expiração do Token, o segredo de criptografia (no projeto o segredo é gerado automaticamente toda vez que o software é inicializado, e salvo no banco de dados, o objeto `getValue()` de `getMapConfigJWT` faz acesso ao banco de dados para retornar esse segredo), por fim são inseridos dois outros parâmetros, email do usuário e Id do usuário.

Prosseguindo com as Classes essenciais para o Spring Security, é necessário o desenvolvimento de uma Classe que implemente a Interface `AuthenticationEntryPoint`. No Projeto essa Classe é a `JWTAuthenticationEntryPoint`, cuja função é retornar erros caso os clientes tentem acessar um recurso sem a devida autorização. A Figura 53 apresenta a implementação dessa Classe.

Figura 53. Classe *JWTAuthenticationEntryPoint*.

```
@Component
public class JwtAuthenticationEntryPoint implements AuthenticationEntryPoint {

    private static final Logger logger = LoggerFactory.getLogger(JwtAuthenticationEntryPoint.class);

    @Override
    public void commence(HttpServletRequest request, HttpServletResponse response,
        AuthenticationException authException) throws IOException, ServletException {

        logger.error("Respondendo com erro de Usuário em autorização. Mensagem - {}" + authException.getMessage());
        response.sendError(HttpServletResponse.SC_UNAUTHORIZED,
            "Desculpe mas você não tem autorização para acessar esse recurso");
    }
}
```

Fonte: Autor (2018)

A última Classe que serve como Base para o Spring Security é JWTAuthenticationFilter que tem as seguintes funções:

- a. Ler o Token do Cabeçalho Authorization das Requisições;
- b. Validar o Token;
- c. Carregar os detalhes do Usuário associados ao Token;
- d. Inserir os Detalhes do Usuário dentro do Contexto de Segurança do Spring, para que o Spring possa fazer as checagens de segurança.

A seguir, na Figura 54 é apresentado o método principal da Classe de JWTAuthenticationFilter. Método esse responsável por executar as funções mencionadas anteriormente.

Figura 54. Método *FilterInternal* de *JWTAuthenticationFilter*.

```
@Override
protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain)
    throws ServletException, IOException {

    try {
        String jwt = getJwtFromRequest(request);

        if (StringUtils.hasText(jwt) && tokenProvider.validateToken(jwt)) {
            Integer userId = tokenProvider.getUserIdFromJWT(jwt);
            UserDetails userDetails = customUserDetailsService.loadByUserId(userId);
            UsernamePasswordAuthenticationToken authentication = new UsernamePasswordAuthenticationToken(
                userDetails, null, userDetails.getAuthorities());
            authentication.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));

            SecurityContextHolder.getContext().setAuthentication(authentication);
        }
    } catch (Exception e) {
        logger.error("Could not set user authentication in security context", e);
    }

    filterChain.doFilter(request, response);
}
```

Fonte: Autor (2018)

Além das Classes mencionada anteriormente. Existem a necessidade do desenvolvimento de uma Classe com as configurações de Segurança. Essa Classe deverá estender a Classe ‘WebSecurityConfigurerAdapter’ como o mostrado na Figura 55.

Figura 55. Anotações de parametrização da Classe SecurityConfig.

```
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(securedEnabled = true, jsr250Enabled = true, prePostEnabled = true)
public class SecurityConfig extends WebSecurityConfigurerAdapter {
```

Fonte: Autor (2018)

Também foi necessária a utilização de algumas anotações nessa Classe: `@Configuration` indicando que a Classe é uma Classe de configuração, `@EnableWebSecurity` anotação tem a função de desativar a utilização de autenticação e das configurações de segurança Padrão do Spring e utilizar uma própria definição de segurança e a última anotação `@EnableGlobalMethodSecurity` com parâmetros necessários para habilitar módulo de segurança nos métodos do Projeto, e sinalizando que podemos utilizar anotações para verificar as autorizações na definição dos métodos.

É nesta mesma classe onde são efetuadas as parametrizações para liberação de CORS (*Cross Origin Requests*) em português, requisições vindas de origens adversas. É um recurso que barra as requisições de origens que não sejam o próprio ambiente de execução, deixando o CORS habilitado, não é possível fazer com que os Serviços do front-end se comuniquem com o back-end. A Figura 56 mostra a configuração utilizada no Projeto.

Figura 56. Bean para Configuração de Cors.

```

@Bean
CorsConfigurationSource corsConfigurationSource() {

    CorsConfiguration corsConfiguration = new CorsConfiguration().applyPermitDefaultValues();

    corsConfiguration.setAllowedMethods(Arrays.asList("POST", "GET", "PUT", "DELETE", "OPTIONS"));

    final UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();

    source.registerCorsConfiguration("/**", corsConfiguration);

    return source;
}

```

Fonte: Autor (2018)

O método principal da Classe SecurityConfig é o método `configure` mostrado na Figura 57. Além desse método indicar que haverá parametrização de CORS no Software, ele também é responsável por realizar a liberação de rotas Http que não precisam de proteção por autenticação, por exemplo a rota para cadastro de novo usuário.

Figura 57. Método Configure da Classe *SecurityConfig*.

```

@Override
protected void configure(HttpSecurity http) throws Exception {
    http.cors().and().csrf().disable().exceptionHandling().authenticationEntryPoint(unauthorizedHandler).and()
        .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS).and().headers()
        .frameOptions().sameOrigin().and().authorizeRequests()
        .antMatchers("/", "/favicon.ico", "/**/*.png", "/**/*.gif", "/**/*.svg", "/**/*.jpg", "/**/*.html",
                    "/**/*.css", "/**/*.js")
        .permitAll().antMatchers("/login/**", "/h2-console/**").permitAll()
        .antMatchers(HttpMethod.GET, "/ceps/buscaccep/**", "/configs/buscacrypto", "user/buscauser/**",
                    "pessoa/buscapessoa/id/**")
        .permitAll().antMatchers(HttpMethod.POST, "/pessoa/inserepf", "/pessoa/inserepj", "/api/**").permitAll()
        .anyRequest().authenticated();

    // Add our custom JWT security filter
    http.addFilterBefore(jwtAuthenticationFilter(), UsernamePasswordAuthenticationFilter.class);
}

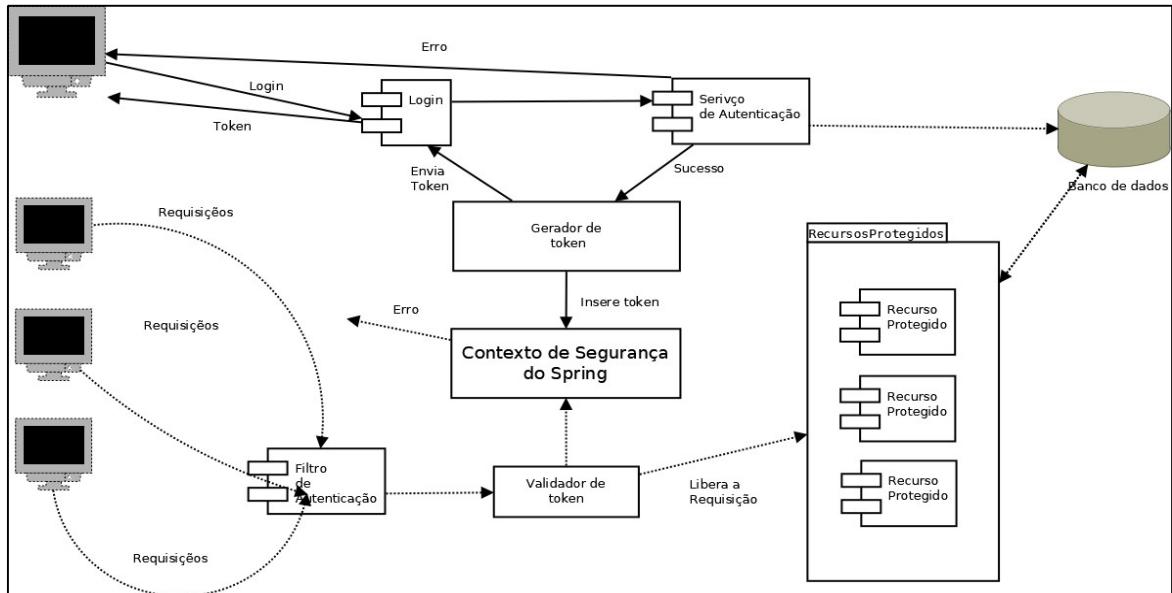
```

Fonte: Autor (2018)

3.6.1. Visão Geral - Segurança

A Figura 58 ilustra de maneira básica, uma visão geral sobre o funcionamento do modulo de segurança dentro da aplicação.

Figura 58. Diagrama exemplificando Implementação de Segurança.



Fonte: Autor (2018)

Todas as requisições realizadas, que estejam parametrizadas com liberação na classe SecurityConfig, são recebidas diretamente pelos Controladores e repassadas as Classes de serviço. Um exemplo é o método de Login, ele precisa ser liberado, pois um usuário que ainda não efetuou Login, ainda não possui o Token de Autenticação e ainda também não se encontra no contexto de Segurança.

Efetuado o método de Login, caso as credenciais estejam corretas, será retornado o token de autenticação do usuário, e o Usuário será inserido no Contexto de Segurança. Essa inserção acontece para que o mesmo Token possa ser utilizado em várias requisições por um período de tempo determinado nos parâmetros de configuração do Projeto. Caso ocorra algum problema no Login, o JWTAuthenticationEntryPoint irá lançar um erro ao usuário.

Nos demais casos de métodos que possuem rotas protegidas todas as requisições irão sofrer o mesmo processo.

Antes das requisições acessarem os método das Classes Controladoras, eles serão filtradas pela Classe JWTAuthenticationFilter , essa classe lê o cabeçalho *Authorization* dos Tokens. Extraí suas informações e solicita validação do Token para a Classe JWTTokenProvider. Caso exista algum problema com o Token, automaticamente é retornado um erro ao usuário. Se o Token for válido, a requisição é liberada.

Feito esse processo, a requisição chegará dentro das Classes Controladoras, onde antes do método ser executado, é realizada a última validação. É validada a se o usuário possui a permissão para acessar o recurso, utilizando a anotação @PreAuthorize conforme mostrado na Figura 59. Na Figura, é apresentado o método para alteração do cadastro de uma pessoa, mas para acessar essa rota precisa ter a autorização ‘ROLE_UPDATE_PESSOA’.

Figura 59. Utilização da Anotação @PreAuthorize.

```
@PreAuthorize("hasRole('ROLE_UPDATE_PESSOA') or hasRole('ROLE_ADMIN')")
@RequestMapping(value = "/update", method = RequestMethod.PUT)
public ResponseEntity<Pessoa> updatePessoa(@RequestBody Pessoa updatePessoa){
    Pessoa pessoa = pessoaService.updatePessoa(updatePessoa);
    return ResponseEntity.ok(pessoa);
}
```

Fonte: Autor (2018)

3.7. Visão geral do Sistema

No presente subcapítulo serão apresentadas algumas funcionalidades do Software desenvolvido.

A primeira funcionalidade a ser apresentada é a tela de Login, Figura 60, na tela é mostrado o Logo do Software, e os campos de login e senha para acesso. Um dos botões tem a funcionalidade de transmitir os dados para a tentativa de login, e o outro direciona o usuário à pagina de cadastro.

Figura 60. Tela de Login.

Fonte: Autor (2018)

Ao clicar no botão de cadastro, será apresentada a página para inserção das informações básicas de cadastro, Figura 61, dependendo do tipo de pessoa escolhido (Pessoa Física ou Pessoa Jurídica). Caso seja escolhido Pessoa Física, terá que ser informado CPF, nome, Data de Nascimento e Sexo, caso a Pessoa informada seja Jurídica terá que ser informado CNPJ e razão Social. Cards de endereço e Usuário tem que ser cadastrados obrigatoriamente.

Figura 61. Tela de Cadastro.

Fonte: Autor (2018)

No menu Principal, localizado à esquerda, são apresentadas todas as Páginas principais, conforme mostrado na Figura 62.

Figura 62. Menu Principal.



Fonte: Autor (2018)

A opção Rotas Geradas, apresenta a lista de todas as rotas já geradas pela empresa, caso o usuário esteja vinculado a uma empresa. Ou todas as rotas que o usuário gerou, caso ele seja uma Pessoa Física sem vínculo com nenhuma empresa, Figura 63. Caso o usuário clique em cima de uma das rotas geradas, ele será direcionado á pagina com o detalhamento do Rota, Figura 64, que mostra o botão para abrir a rota no Google Maps, os pontos de entrega, e os pontos não atendidos pela empresa. Esse último campo é alimentado apenas caso no ato de geração de rotas, se o usuário possuir vínculo com alguma empresa e a empresa possuir parametrização por regiões, não atendendo algum cep em específico dos ceps que foram inseridos.

Figura 63. Listagem de Rotas Criadas.

Rotas Criadas		
Rotas Geradas		
Id da Rota: 4		
Data de Criação da Rota:	15/09/2018	
Usuário Criador:	adm@adm.com.br	
Id da Rota: 5		
Data de Criação da Rota:	15/09/2018	
Usuário Criador:	adm@adm.com.br	
Id da Rota: 8		
Data de Criação da Rota:	17/09/2018	
Usuário Criador:	adm@adm.com.br	
Id da Rota: 9		
Data de Criação da Rota:	17/09/2018	
Usuário Criador:	adm@adm.com.br	
Id da Rota: 10		
Data de Criação da Rota:	17/09/2018	
Usuário Criador:	adm@adm.com.br	

Fonte: Autor (2018)

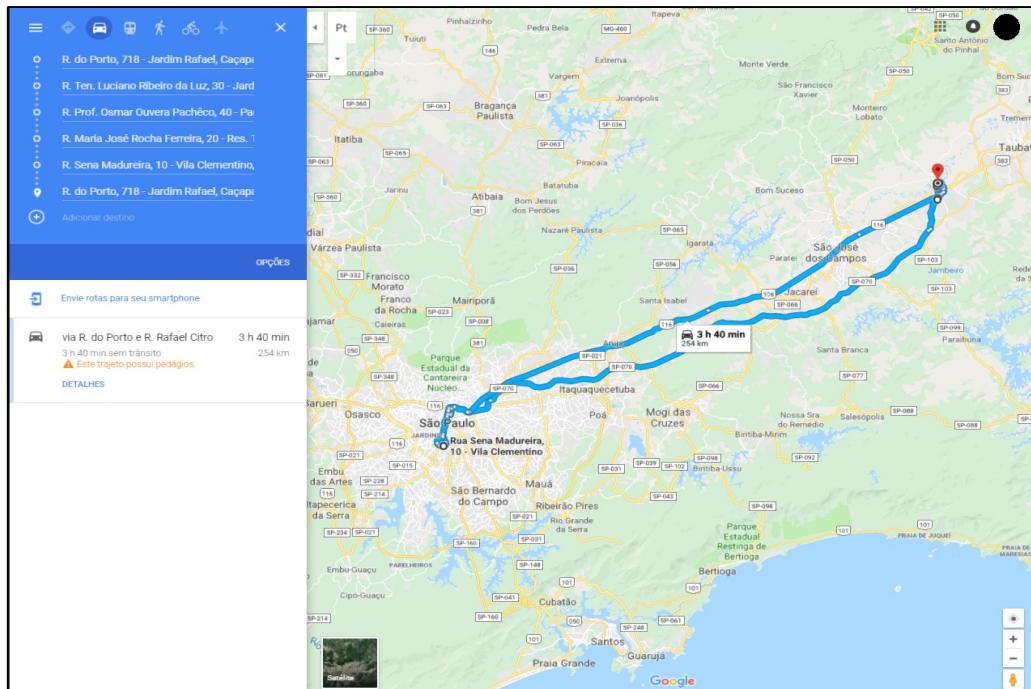
Figura 64. Página de detalhamento da Rota.



Fonte: Autor (2018)

Quando clicar em abrir rota no Google Maps ele será direcionado ao Google Maps com a Rota já sendo carregada automaticamente conforme Figura 65.

Figura 65. Rota Criada Aberta no Google Maps.



Fonte: Autor (2018)

Na Página de Geração de Rotas, a medida que os ceps vão sendo inseridos, eles são listados, podendo ser removidos caso tenham sido inseridos erroneamente, Figura 66. Todos os ceps inseridos são validados antes de aparecerem na listagem. Ao final da

Inserção dos ceps o usuário pode clicar no botão Gerar Rota para que sua rota seja calculada e o botão para abrir no Google Maps apareça, Figura 67.

Figura 66. Página para Gerar a Rota.

CEP
NUM
ADICIONAR GERAR ROTA

Endereços à serem roteirizados

- CEP: 12288560 Numero: 321
- CEP: 12285020 Numero: 1000
- CEP: 12281-630 Numero: 10
- CEP: 12281-900 Numero: 125

Fonte: Autor (2018)

Figura 67. Página Após Geração da Rota.

URL MAPS
ROTA

Endereços da Rota

- Rua do Porto 321, Jardim Rafael, Caçapava, São Paulo, 122885
- Praça da Bandeira 10, Centro, Caçapava, São Paulo, 12281630
- Praça da Bandeira 125, Centro, Caçapava, São Paulo, 12281900
- Rodovia João Amaral Gurgel 1000, Parque Residencial Maria El

Lista de Responsáveis por endereços

Fonte: Autor (2018)

Neste exemplo o Campo Lista de Responsáveis por endereços ficou em branco pois, o usuário é Pessoa Física e não possui nenhum tipo de parametrização de Regiões.

Dentro do menu principal existe a opção de acessar o endereço cadastrado do usuário Figura 68, tendo a opção de realizar alteração, caso necessário.

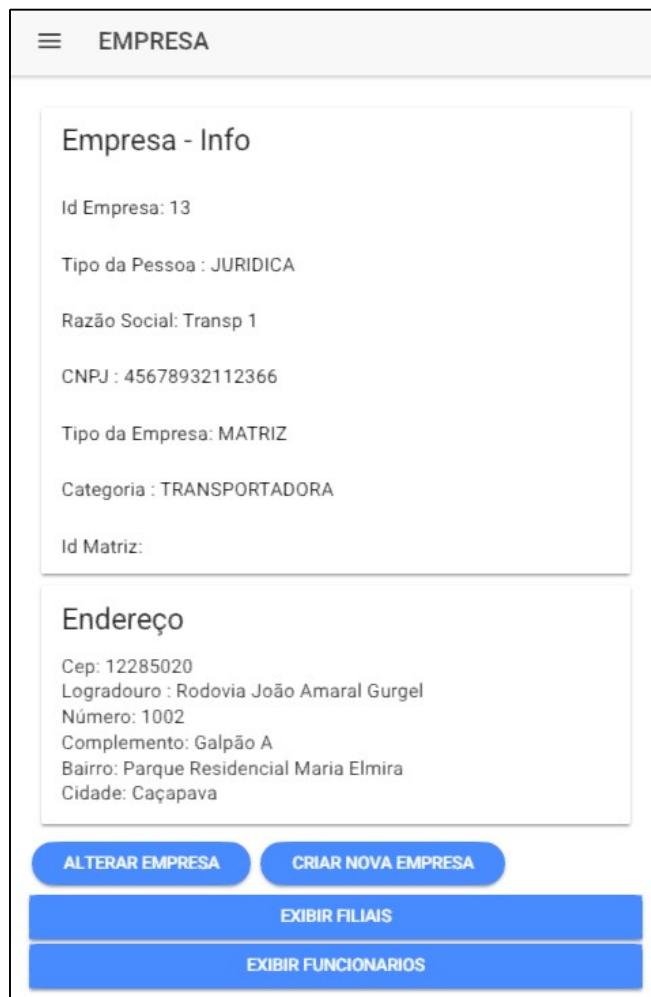
Figura 68. Página de Endereço.



Fonte: Autor (2018)

A Página de Empresas Figura 69, exibe as informações da Empresa e permite que seja acessado a Funcionalidade de Alteração da Empresa e Criar uma nova Empresa, nesta Opção, automaticamente a empresa Criada se torna uma Filial, da empresa do Usuário que está efetuando o cadastro. Outras opções apresentadas nesta Página é a exibição das Filiais da Empresa Figura 70, e também exibição dos seus Funcionários Figura 71.

Figura 69. Página Empresa.



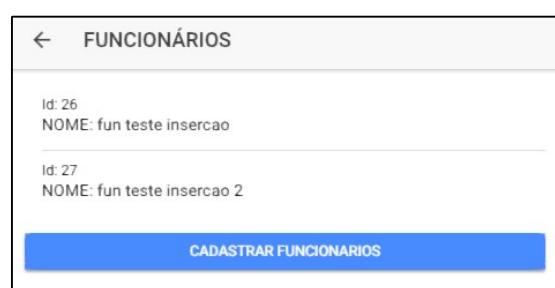
Fonte: Autor (2018)

Figura 70. Página de Filiais da Empresa.



Fonte: Autor (2018)

Figura 71. Página de Listagem Funcionários da Empresa.



Fonte: Autor (2018)

Tanto na Listagem de Funcionários, quanto na Listagem das Filiais, quando é clicado em cima de um item da lista o usuário é redirecionado a pagina de detalhamento do mesmo, onde há todas as informações do item selecionado, seja empresa ou funcionário

A Página de Regiões apresentada na, Figura 72 traz as informações da região cadastrada para a empresa. Se o usuário necessitar, também poderá exibir todos os Ceps dessa região.

Figura 72. Página de Região.

REGIÃO

INFORMAÇÕES DA REGIÃO CADASTRADA

Identificação: 2
Cadastrado para : Transp 1
Descrição da Rota : TESTE INSERÇÃO

Cep: 12209060
Logradouro: Rua Tenente Manuel Pedro de Carvalho
Bairro: Vila Santa Helena
Cidade : São José dos Campos. Estado : São Paulo

Cep: 12209310
Logradouro: Rua Felisbino Pinto da Cunha
Bairro: Vila Nova São José
Cidade : São José dos Campos. Estado : São Paulo

Cep: 04021001
Logradouro: Rua Sena Madureira
Bairro: Vila Clementino
Cidade : São Paulo. Estado : São Paulo

Cep: 12216300
Logradouro: Rua Opala
Bairro: Jardim São José Centro
Cidade : São José dos Campos. Estado : São Paulo

Cep: 12209010
Logradouro: Avenida São José
Bairro: Jardim Bela Vista
Cidade : São José dos Campos. Estado : São Paulo

ESCONDER CEPS

ALTERAR REGIÃO

Fonte: Autor (2018)

Quando o usuário Clicar em Alterar Região ele será levado para a Página de alteração de Região (Figura 73). Essa Página contém diversas opções, podendo remover todos os Ceps da região, ou remover alguns Ceps desejados. Outra opção é adicionar mais ceps a Região, os ceps para essa adição podem buscados a partir de uma cidade, ou de um bairro de uma cidade. O usuário pode adicionar todos os Ceps dessa busca, ou apenas os que ele desejar. Realizando todas as alterações desejadas, o usuário pode clicar em salvar para efetuar a alteração na Região.

Figura 73. Página para Alterar a Região.

ALTERAÇÃO DE REGIÃO

TESTE INSERÇÃO

CEPS DA REGIÃO

- CEP : 12209060
Nome rua : Rua Tenente Manuel Pedro de Carvalho
- CEP : 12209310
Nome rua : Rua Felisbino Pinto da Cunha
- CEP : 04021001
Nome rua : Rua Sena Madureira
- CEP : 12216300
Nome rua : Rua Opala
- CEP : 12209010
Nome rua : Avenida São José

ADICIONAR CEP

Selecionar modo de Busca Busca por Estado e Cid...

Estado : São Paulo

Cidade : Caraguatatuba

CEP : 11660070
Nome do logradouro : Rua Guarulhos

ADICIONAR TODOS

NOVA BUSCA **SALVAR ALTERAÇÕES**

Fonte: Autor (2018)

A última funcionalidade a ser apresentada é a de Gestão dos Usuários. Ao clicar em usuários no Menu Principal, será listado todos os usuários da empresa, Figura 74.

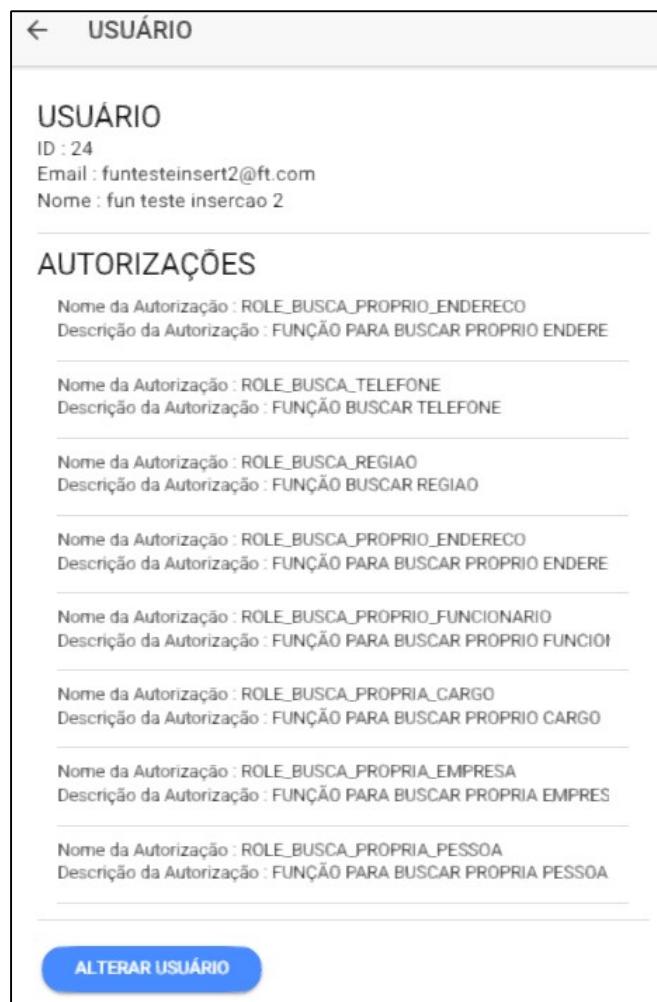
Figura 74. Página para Alterar a Região.

≡ USUÁRIO
Usuários da empresa
Nome: fun teste insercao Email : funtesteinsert@ft.com.br
Nome: fun teste insercao 2 Email : funtesteinsert2@ft.com

Fonte: Autor (2018)

Ao clicar no Funcionário desejado o usuário é direcionado a Pagina de detalhamento do usuário, onde é exibido todas as permissões que o usuário possui, Figura 75, e também é apresentada a opção de alteração do usuário.

Figura 75. Página de Detalhamento do Usuário.



Fonte: Autor (2018)

Caso o usuário clique em alterar será aberta uma pagina que listará todas as permissões do usuário, que podem ser tiradas clicando no ícone de remover, e todas as permissões disponíveis para serem aplicadas ao usuário, que podem ser aplicadas clicando no ícone de adição, Figura 76. Ao realizar as alterações desejadas o usuário poderá clicar em Salvar alteração, Figura 77.

Figura 76. Página de Alteração do Usuário.

The screenshot shows a user modification form. At the top, there are three entries for user roles:

- Nome da Autorização : ROLE_BUSCA_PROPRIA_CARGO
Descrição da Autorização : FUNÇÃO PARA BUSCAR PRÓPRIO CARGO
- Nome da Autorização : ROLE_BUSCA_PROPRIA_EMPRESA
Descrição da Autorização : FUNÇÃO PARA BUSCAR PRÓPRIA EMPRESA
- Nome da Autorização : ROLE_BUSCA_PROPRIA_PESSOA
Descrição da Autorização : FUNÇÃO PARA BUSCAR PRÓPRIA PESSOA

A horizontal line separates these from the next section. Below it is a bold header:

AUTORIZAÇÕES DO USUÁRIO

Below the header, there are four more entries:

- Nome da Autorização : ROLE_ADMIN
Descrição da Autorização : FUNÇÃO DE ADMINISTRADOR DO SISTEMA
- Nome da Autorização : ROLE_CREATE_CARGO
Descrição da Autorização : FUNÇÃO DE CRIAR CARGOS
- Nome da Autorização : ROLE_DELETE_CARGO
Descrição da Autorização : FUNÇÃO DE DELETAR CARGOS
- Nome da Autorização : ROLE_BUSCA_CARGO
Descrição da Autorização : FUNÇÃO DE BUSCAR CARGOS

Fonte: Autor (2018)

Figura 77. Botões para Salvar e Cancelar alteração do Usuário.

The screenshot shows a user modification form. At the top, there are two entries for user roles:

- Nome da Autorização : ROLE_BUSCA_USERS
Descrição da Autorização : FUNÇÃO PARA BUSCAR USUÁRIOS
- Nome da Autorização : ROLE_BUSCA_ROLES
Descrição da Autorização : FUNÇÃO PARA BUSCAR AUTORIZAÇÕES

A horizontal line separates these from the bottom section. At the bottom are two large blue buttons:

- SALVAR ALTERAÇÃO**
- CANCELAR ALTERAÇÃO**

Fonte: Autor (2018)

Essas foram as principais funcionalidades desenvolvidas no Software.

4. VALIDAÇÃO E ANÁLISE DOS DOS RESULTADOS OBTIDOS

Neste Capítulo serão apresentadas as Métricas do Sistema desenvolvido, as Técnicas e Ferramentas utilizadas para Verificação e Validação do Software e os resultados obtidos. O último subcapítulo será responsável por descrever o procedimento de validação do algoritmo de roteirização e os resultados obtidos nesta etapa.

4.1. Métricas do Sistema

As métricas do sistema foram definidas baseando-se nas métricas padrão da ferramenta SonarQube. Na Tabela 52 são apresentadas as métricas definidas e o resultado esperado. Os resultados esperados e apresentados na tabela, são baseados nos critérios de avaliação do SonarQube.

Tabela 52. Métricas de Qualidade e Resultado Esperado.

Métricas	Resultado Esperado
Bugs	Rank A
Vulnerabilidades	Rank A
Code Smells	Rank A
Cobertura de testes	Acima de 10 %
Segurança	Rank A
Manutenção	Rank A
Complexidade	Menos de 50 pontos por Classe
Problemas(críticos)	Menos de 30

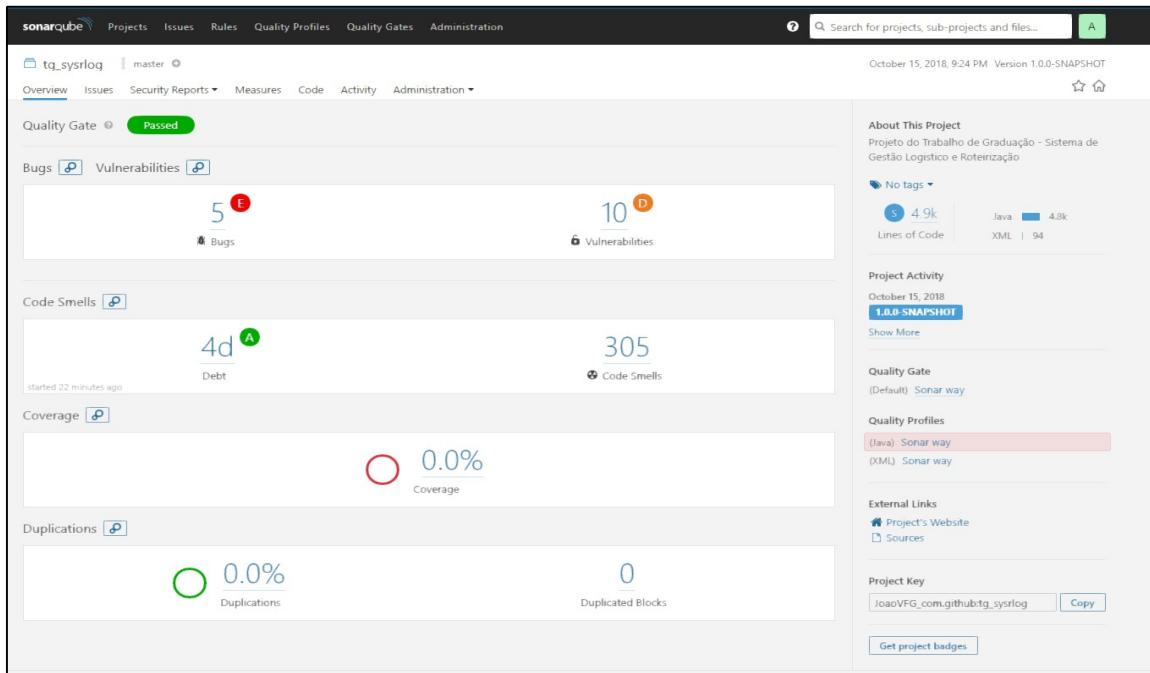
Fonte: Autor (2018)

Para a excelência do Projeto, foram definidos ‘Rank A’ nos Critérios de Bugs, Vulnerabilidades, CodeSmells, Segurança e Manutenção. Para a cobertura de testes, foi definida uma porcentagem baixa, pois o teste de Fato foi realizado utilizando o Postman, baseando-se os testes a partir das rotas HTTP. O quesito complexidade é baseado em uma fórmula no SonarQube, quanto mais baixo é a pontuação, menor é a complexidade do código.

4.1.1. Resultados das Métricas *Back-end*

Ao executar o SonarQube pela primeira vez, foi identificado alguns problemas no código, Figura 78, que fez com que os resultados não fossem satisfatórios e não atingissem as métricas definidas. Além do resultado não apresentar os dados de Cobertura, por conta da ausência de uma configuração dentro do pom.XML para utilização do JaCoCo.

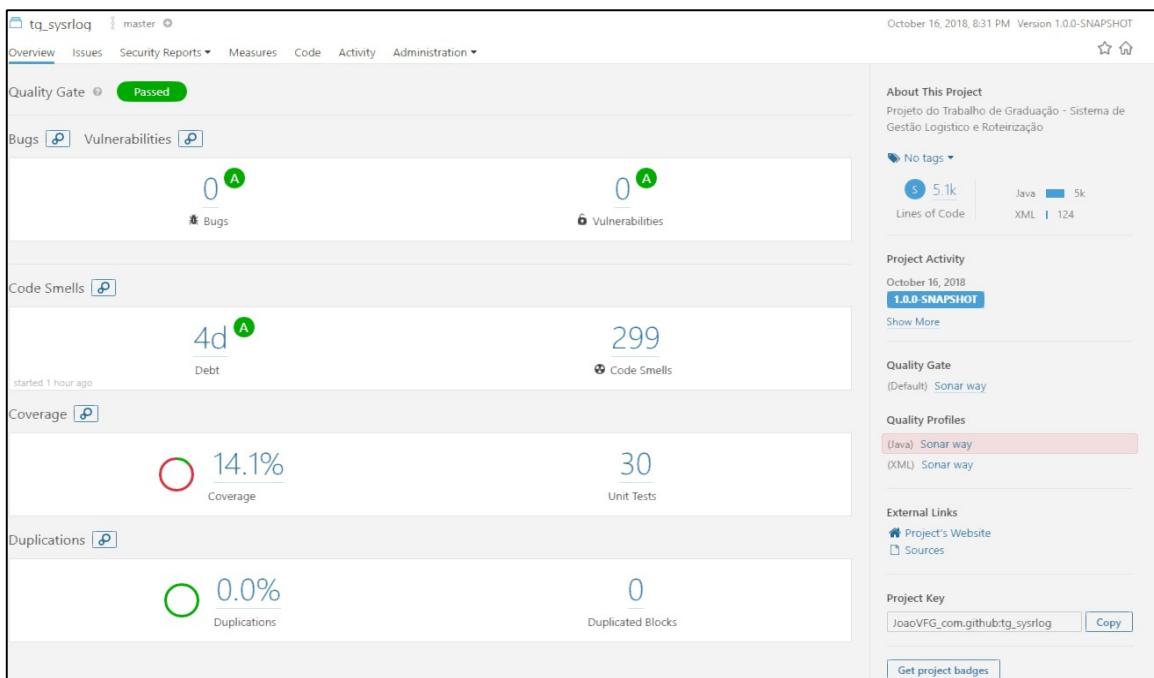
Figura 78. Primeiro Resultado da execução do SonarQube.



Fonte: Autor (2018)

Após o resultado, foi realizado os ajustes necessários no código para atingir os resultados esperados. A Figura 79, mostra o resultado de outra execução da Análise do código após realizada as alterações.

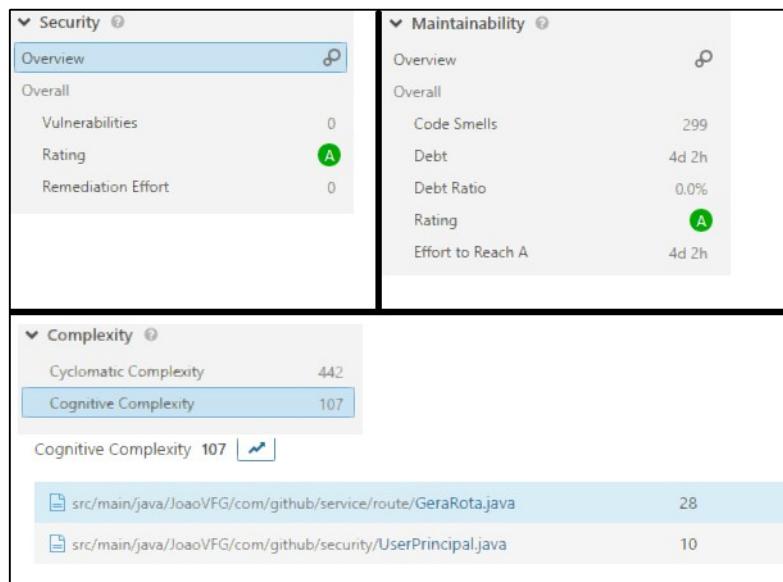
Figura 79. Back-end - Resultado da Execução do SonarQube Após Alterações Efetuadas.



Fonte: Autor (2018)

Conforme apresentado na Figura 79, o código atingiu os Ranks desejados em todos os aspectos analisados. A Figura 80, apresenta os resultados restantes.

Figura 80. Back-end Outros Resultados.

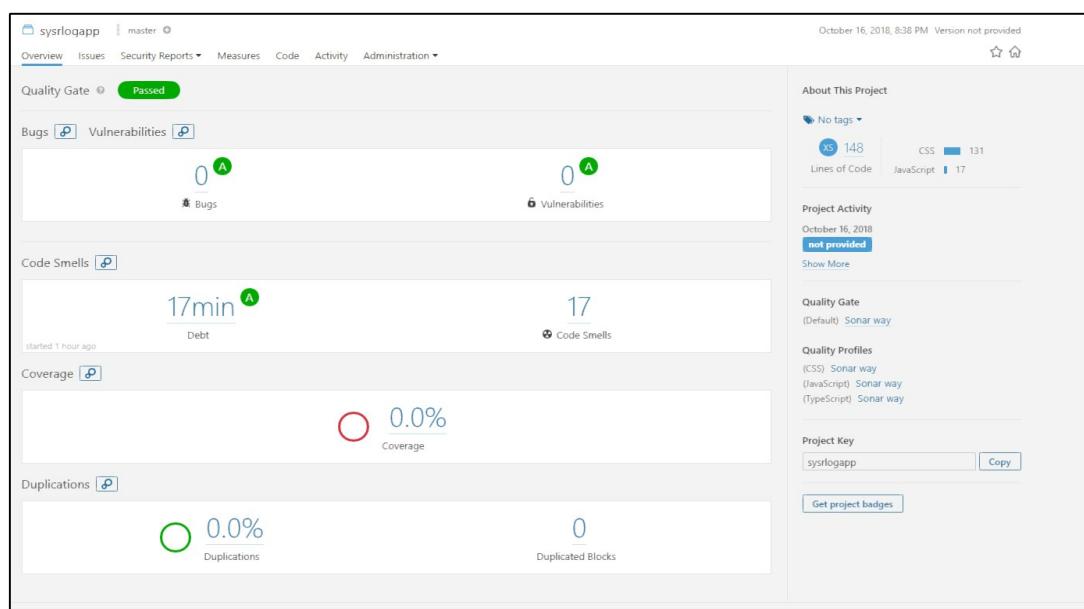


Fonte: Autor (2018)

4.1.2. Resultados das Métricas *Front-end*

A execução do SonarQube no código fonte do *front-end* resultou em uma pontuação melhor do que o do back-end. Logo na primeira execução já foi obtido um resultado que satisfaz as métricas definidas. A Figura 81 apresenta os Resultados Obtidos com a Execução.

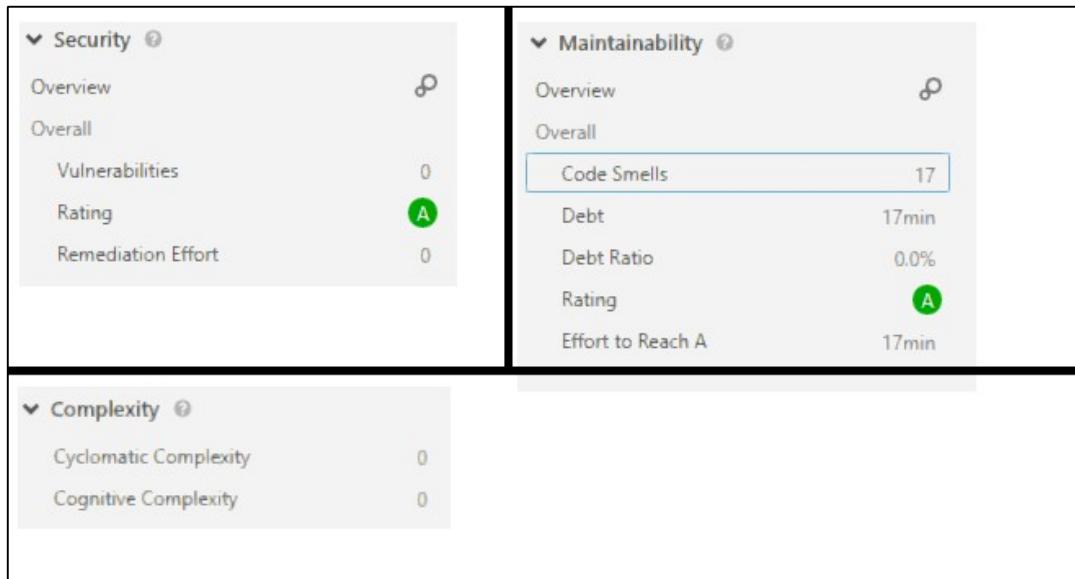
Figura 81. Front-end Resultados SonarQube.



Fonte: Autor (2018)

Conforme apresentado na Figura 81, as métricas foram todas satisfeitas, não sendo necessário nenhum tipo de alteração no código. Na Figura 82 estão apresentados os demais resultados da execução do SonarQube.

Figura 82. Front-end - Outros Resultados Obtidos com SonarQube.



Fonte: Autor (2018)

4.2. Técnicas de Verificação e Validação aplicadas e Resultados

Para validação das funcionalidades do Projeto foram desenvolvidos alguns testes, sendo:

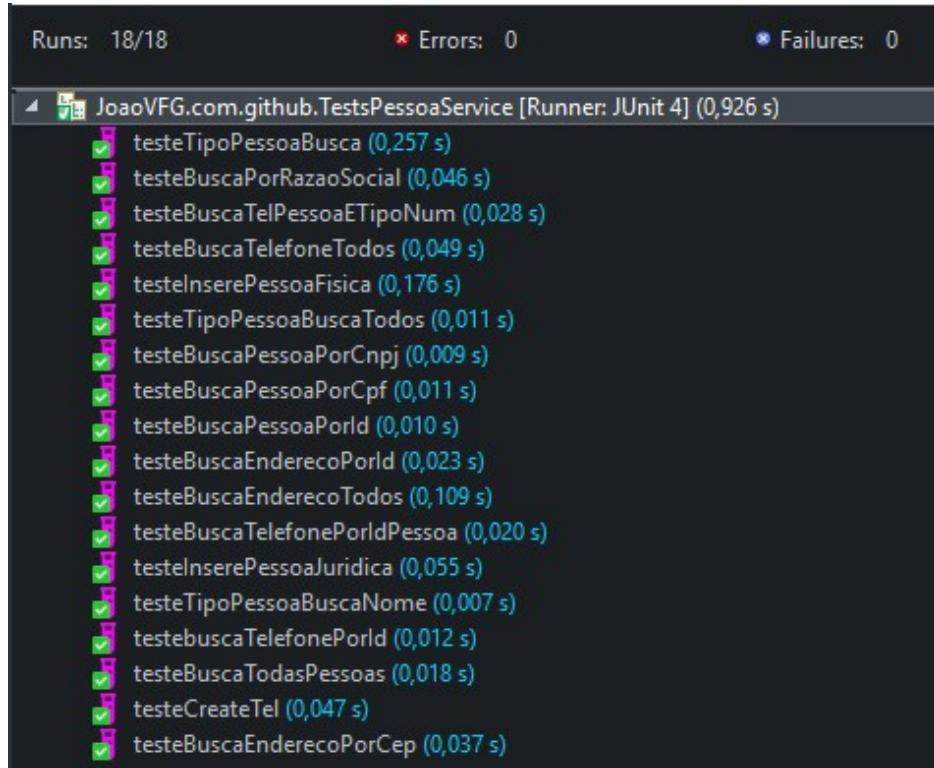
- Testes de Unidade: Testes para analisar o funcionamento de determinados métodos do projeto.
- Testes de Recursos Externos: Para validar o comportamento do software quando executado métodos que consomem recursos externos
- Testes Funcionais de API: Os testes de funcionais foram executados utilizando o Software Postman, tendo como objetivo validar o funcionamento das rotas, e seu comportamento acionando as camadas de Serviço, Repositório e Entidade.
- Testes Funcionais: Foram testes com alguns cenários específicos para validar o funcionamento do front-end.

4.2.1. Testes de Unidade

Os Testes de Unidade foram utilizados na fase inicial do desenvolvimento do Projeto, apenas para testar o funcionamento de determinados Repositories e Services. Esses testes não foram amplamente utilizados, por conta do Plano de Desenvolvimento, onde foi especificado que o projeto seria desenvolvido por módulos, assim os testes foram executados à nível de API.

A Figura 83 apresenta a Classe de Testes relacionados as funcionalidades de cadastro e Gestão de Pessoas, para estes Testes foi utilizando o Framework JUnit.

Figura 83. Teste Unitários de Serviços e Repositórios de Pessoas.



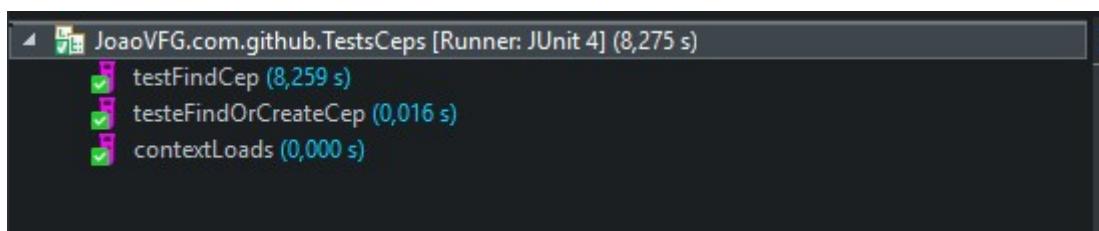
Fonte: Autor (2018)

4.2.2. Testes de Recursos Externos

O projeto utiliza dois Recursos Externos: O Servidor de ceps do ViaCep, e a API Distance Matrix do Google para Cálculo de Distâncias entre endereços. Para validar a utilização destes recursos externos foram também desenvolvidos testes de unidade.

O primeiro teste serviu para validar a utilização do Servidor do ViaCeps. Foram inseridos 7 CEPS(12020220, 12061001, 12061600, 12062490, 12071110, 12423060, 12050410) que não estavam armazenados no Banco de Dados do Projeto. O Software deve acessar o Servidor do ViaCep retornar os dados do cep e armazená-lo no Banco de Dados. Figura 84 apresenta o resultado da execução deste teste.

Figura 84. Teste Unitários - Utilização da API do ViaCep.



Fonte: Autor (2018)

O segundo teste teve como objetivo testar o acesso a API de distâncias. Nesse caso o software deverá acessar a API, e retornar uma distância, também foi testado o método de roteirização que realiza múltiplos acessos a API. A Figura 85 apresenta o resultado da execução destes testes.

Figura 85. Teste Unitários - Utilização da API DistanceMatrix.



Fonte: Autor (2018)

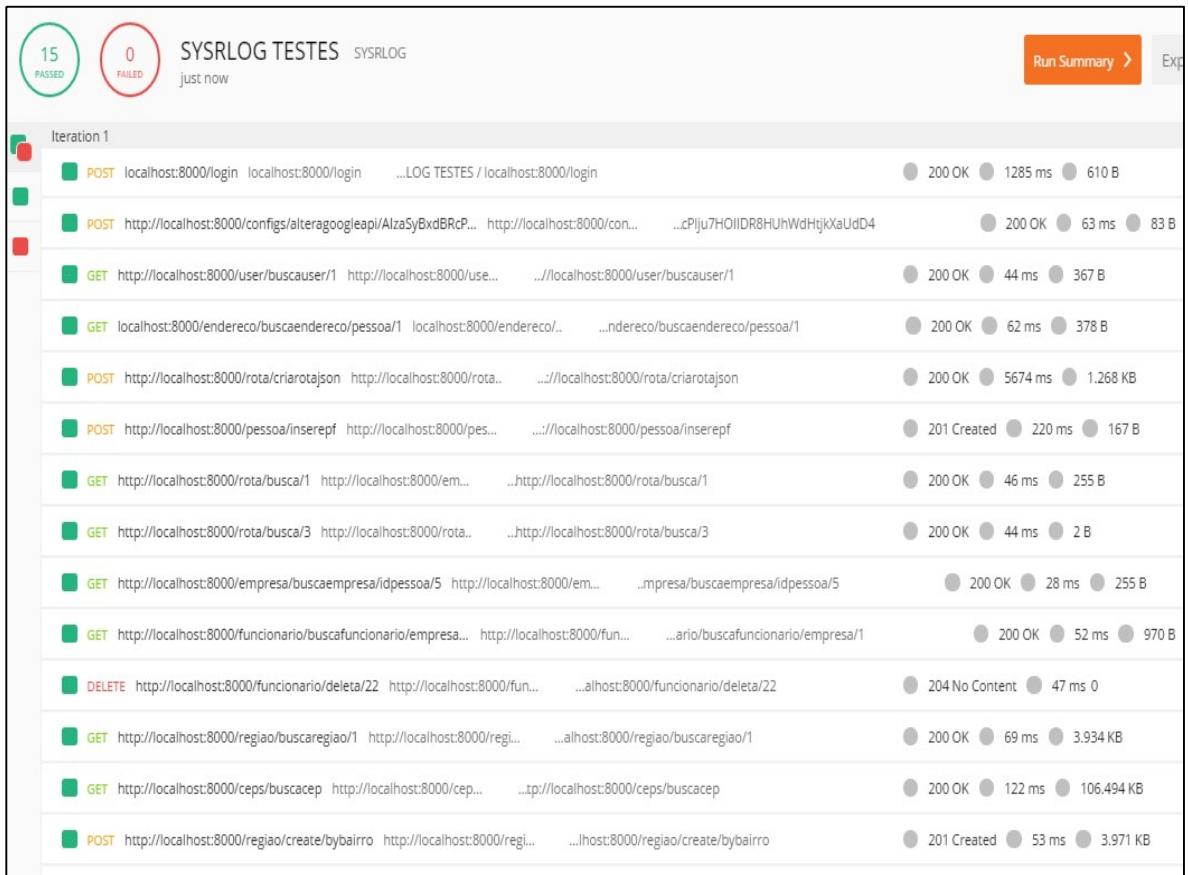
4.2.3. Teste Funcional de API

Os Testes funcionais de API, foram realizados utilizando o Software Postman. Esses testes foram desenvolvidos para validar o funcionamento de todas as rotas do desenvolvidas para o Projeto. Testando as rotas, consequentemente já realizava a validação das camadas de Serviço, Repositório e Entidade, devida a arquitetura do Software.

A Figura 86, apresenta uma Suite de Testes com algumas das rotas mais utilizadas dentro do Projeto. Nesta Suite de testes, foram inseridas algumas rotas, e cada um dos testes teve o código do retorno esperado parametrizado. Assim, ao final da execução da Coleção de testes, foi possível ter a contagem de testes que tiveram sucesso, e a quantidade de falhas.

Dentre os testes que falharam, foi realizada uma análise para identificar o problema e corrigi-lo. Ao final da correção do problema, o teste era executado isoladamente para validar a solução, e caso o retorno fosse o esperado, era novamente realizado uma execução de todos os Testes da Suite, avaliando se a correção não acarretou em problemas com outras funcionalidades.

Figura 86. Teste Funcionais de API Utilizando PostMan.



Fonte: Autor (2018)

4.3. Processo de Validação do Algoritmo de Roteirização e Resultados Obtidos

Os Testes de Roteirização foram os mais importantes implementados, devido á serem os responsáveis por validar o funcionamento e a eficiência do Algoritmo de Roteirização desenvolvido no Projeto. Estes testes foram realizados da seguinte forma:

1. Primeiramente foi realizado o cadastro de empresas fictícias. Os dados das empresas criadas são apresentados na Tabela 53. As Figuras 87 e 88 mostram o as empresas já cadastradas dentro do Software;
2. Foram montadas 5 listas, mostradas na Tabela 54, com endereços de supermercados reais referentes as áreas de atendimento das empresas cadastradas . Uma lista para cada empresa;
3. A lista com os endereços foi inserida dentro do Google Maps, e a Rota Final documentada;
4. As mesmas listas inseridas no Google Maps foram inseridas no Software do Projeto e foi executada a roteirização;

5. Os resultados da rota do Google Maps e do SysRLog foram comparadas para avaliação dos Resultados.

Tabela 53. Tabelas com as Informações das Empresas Cadastradas.

Empresa - 1			
Razão Social	Transportadora CPV	CNPJ	11111111111111
Endereço	Rua Vereador Geraldo Nogueira da Silva, 1000	Cidade	Caçapava
Email	transportadoracpv@gmail.com		
Empresa - 2			
Razão Social	Transportadora SJC	CNPJ	22222222222222
Endereço	Estr. Mun. Martins Guimarães, 1050	Cidade	São José Dos Campos
Email	transportadorasjc@gmail.com		
Empresa - 3			
Razão Social	Transportadora TBT	CNPJ	33333333333333
Endereço	Estrada Municipal João Gadioli, 1330	Cidade	Taubaté
Email	transportadoratbt@gmail.com		
Empresa - 4			
Razão Social	Transportadora JAC	CNPJ	44444444444444
Endereço	Av. Getúlio Dorneles Vargas, 1390	Cidade	Jacareí
Email	transportadorajac@gmail.com		
Empresa - 5			
Razão Social	Transportadora CAR	CNPJ	55555555555555
Endereço	Rua Hermes da Fonseca, 217	Cidade	Caraguatatuba
Email	transportadoracar@gmail.com		

Fonte: Autor (2018)

Figura 87. Empresa Cadastrada - Matriz.



Fonte: Autor (2018)

Figura 88. Empresas Cadastradas - Filiais.

FILIAIS	
Filiais	
Id da Empresa: 17	Nome da Empresa : Transportadora SJC
Id da Empresa: 18	Nome da Empresa : Transportadora TBT
Id da Empresa: 19	Nome da Empresa : Transportadora JAC
Id da Empresa: 20	Nome da Empresa : Transportadora CAR

Fonte: Autor (2018).

Tabela 54. Tabelas com a Relação de Ceps Utilizados nos Testes de Roteirização de Cada Empresa.

Empresa 1 - Caçapava		Empresa 2 - São José dos Campos		Empresa 3 - Taubaté		Empresa 4 - Jacareí		Empresa 5 - Caraguatatuba	
Cep	NUM	Cep	NUM	Cep	NUM	Cep	NUM	Cep	NUM
12285-020	980	12242-000	2200	12040-900	1700	12307-200	88	11673-110	141
12284-070	35	12236-660	3359	12091-000	7181	12305-000	1591	11666-525	4651
12281-370	101	12230-011	591	12030-040	1160	12320-670	291	11675-000	297
12287-020	401	12220-000	6005	12040-670	1780	12324-350	42	11674-750	96
12281-020	625	12210-010	29	12043-490	111	12322-000	109	11665-251	838
12280-034	148	12230-000	227	12040-000	420	12301-330	s/n	11660-971	840
12289-010	700	12240-540	1501	12061-100	384	12312-000	2010		
		12215-656	4496	12070-100	808				

Fonte: Autor (2018)

Nos próximos sub-capítulos serão apresentados os resultados obtidos em cada uma dos Testes.

4.3.1. Caso de Testes 1 - Cidade de Caçapava

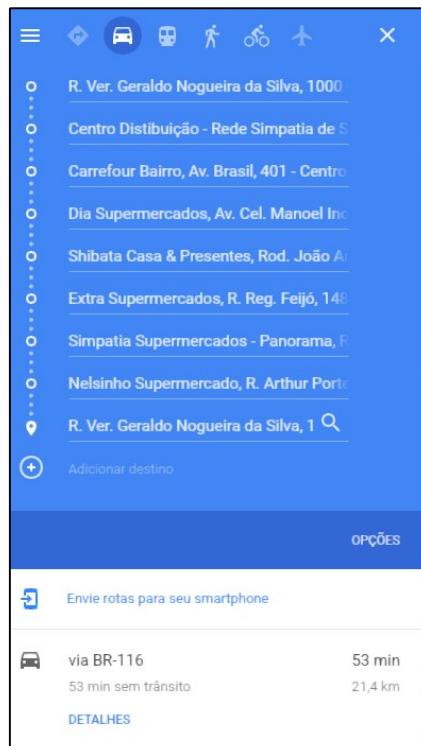
O primeiro Caso de testes está relacionada a primeira empresa fictícia cadastrada, que possui a cidade de Caçapava como região de atuação. A Tabela 55 apresenta o detalhamento dos Caso de Teste, relacionando os endereços envolvidos e o resultado esperado.

Tabela 55. Detalhamento do Caso de Teste 1.

ID	Descrição do Cenário de Teste	Nome dos Pontos Envoltos (Sequência Inserida No Maps)	Resultado esperado
CT1	Cenário de Teste 1 :Esta validação tem por objetivo a definição de 7 pontos atendidos pela Empresa 1 na Cidade de Caçapava- São Paulo. Sendo esses pontos compostos pela Origem e Destino que é o endereço da empresa e 7 pontos Intermediários	P0 - R. Ver. Geraldo Nogueira da Silva, 1000 - Res. Terras do Vale P1 - R. Dr. Agenor Genésio do Nascimento, 35 - Jardim Maria Cândida P2 - Av. Brasil, 401 - Centro P3 - Av. Cel. Manoel Inocêncio, 625 - Vila Resende P4 - Rod. João Amaral Gurgel, 980 - Res. Terras do Vale P5 - R. Antônio Guedes Tavares, 700 - Jardim Panorama P6 - R. Reg. Feijó, 148 - Vila Santos P7 - R. Arthur Portes, 101 - Vila São João P0 - R. Ver. Geraldo Nogueira da Silva, 1000 - Res. Terras do Vale	P0 -> P4 -> P1 -> P3 -> P7 -> P2 -> P6 -> P5 -> P0

Fonte: Autor (2018)

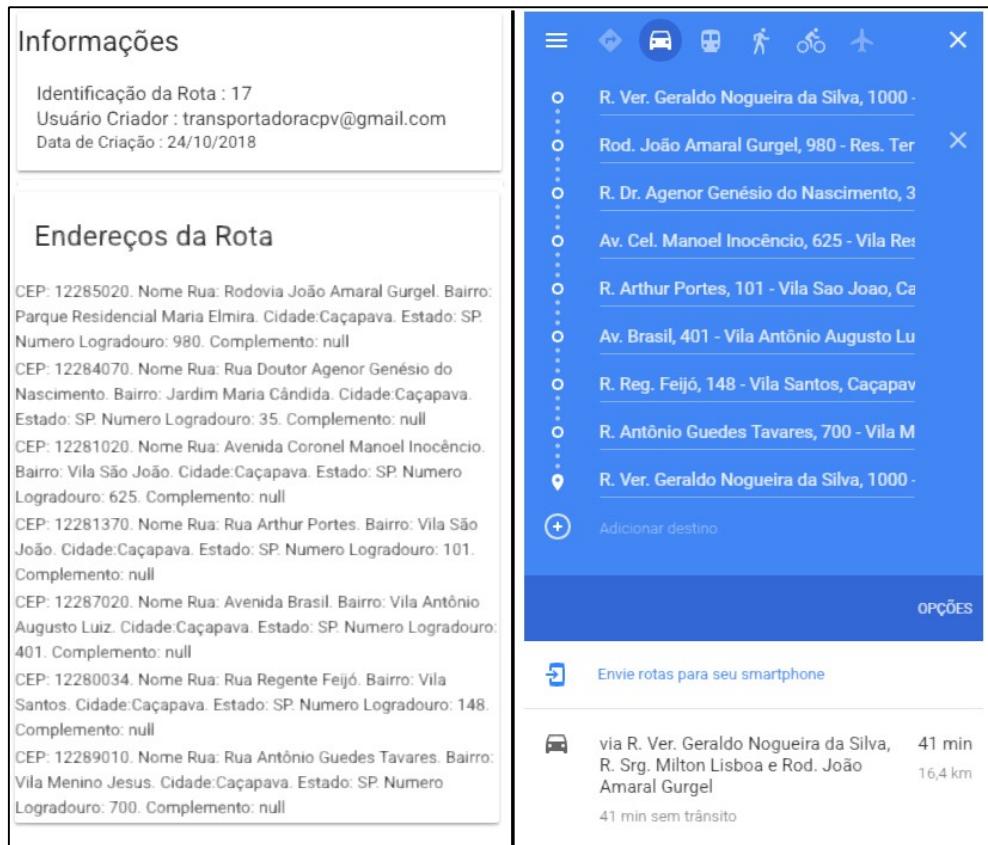
A Figura 89 apresenta a Rota gerada pelo Google Maps após a inserção dos pontos conforme a ordem apresentada na Tabela 55.

Figura 89. Caso de Testes 1 - Rota Gerada pelo Google Maps

Fonte: Autor (2018)

A Figura 90 apresenta a Rota que foi gerada com a Otimização do algoritmo Desenvolvido no Projeto. No lado esquerdo da imagem, é a tela de consulta a rota gerada do Software e do lado direito é ela aberta no Google Maps.

Figura 90. Caso de Testes 1 - Rota Gerada pelo SysRLog.



Fonte: Autor (2018)

A Rota gerada pelo SysRLog apresentou o resultado desejado, definido na Tabela 55. A otimização conseguiu obter resultados satisfatórios, tendo uma menor distância e um menor tempo para ser percorrida que o lançamento desses pontos diretamente no Google Maps. A Tabela 56 apresenta o comparativo entre as duas rotas.

Tabela 56. Resultados Obtidos no Caso de Teste 1.

Resultado Roteirização - Caso de Teste 1 - Cidade de Caçapava				
	Valor obtido GoogleMaps	Valor Obtido SysRLog	Diferença	Percentual de Redução
Tempo(horas)	0:53	0:41	0:12	22,64%
Distância(KMs)	21,4	16,4	5	23,36%

Fonte: Autor (2018)

4.3.2. Caso de Testes 2 - Cidade de São José dos Campos

O segundo Caso de Teste foi realizado tendo como base uma empresa fictícia com atuação na Cidade de São José dos Campos. A Tabela 57 apresenta o detalhamento dos Caso de Teste, relacionando os endereços envolvidos e o resultado esperado.

Tabela 57. Detalhamento do Caso de Teste 2.

ID	Descrição do Cenário de Teste	Nome dos Pontos Envolvidos (Sequência Inserida No Maps)	Resultado esperado
CT2	Cenário de Teste 2 :Esta validação tem por objetivo a definição de 9 pontos atendidos pela Empresa 2 na Cidade de São José dos Campos- São Paulo. Sendo esses pontos compostos pela Origem e Destino que é o endereço da empresa e 9 pontos Intermediários	P0 - Estr. Mun. Martins Guimarães, 1050 - Vila Tesouro P1 - Av. São João, 2200 - Jardim das Colinas P2 - Av. Dr. João Batista de Souza Soares, 3359 - Jardim Morumbi P3 - Av. Cassiopeia, 591 - Jardim Satélite P4 - Av. Pres. Juscelino Kubitschek, 6005 P5 - R. Claudino Pinto, 29 - Centro P6 - Av. Andrômeda, 227 - Jardim Satélite P7 - Av. Cassiano Ricardo, 1501 - Jardim Alvorada P8 - Av. Nair Toledo de Mira, 4496 - Jardim Paulista P0 - Estr. Mun. Martins Guimarães, 1050 - Vila Tesouro	P0 -> P4 -> P8-> P5 -> P1 ->P7 -> P6 -> P3 -> P2 -> P0

Fonte: Autor (2018)

A Figura 91 apresenta a Rota gerada pelo Google Maps após a inserção dos pontos conforme a ordem apresentada na Tabela 57.

Figura 91. Caso de Testes 2 - Rota Gerada pelo Google Maps.



Fonte: Autor (2018)

A Figura 92 apresenta a Rota que foi gerada com a Otimização do algoritmo Desenvolvido no Projeto. No lado esquerdo da imagem, é a tela de consulta a rota Gerada do Software e do lado direito é ela aberta no Google Maps.

Figura 92. Caso de Testes 2 - Rota Gerada pelo SysRLog.

Informações

Identificação da Rota : 24
Usuário Criador : transportadorasjc@gmail.com
Data de Criação : 26/10/2018

Endereços da Rota

CEP: 12242000. Nome Rua: Avenida São João. Bairro: Jardim das Colinas. Cidade:São José dos Campos. Estado: SP. Numero Logradouro: 2200. Complemento: null
CEP: 12236660. Nome Rua: Avenida João Batista de Souza Soares. Bairro: Cidade Morumbi. Cidade:São José dos Campos. Estado: SP. Numero Logradouro: 3359. Complemento: null
CEP: 12230011. Nome Rua: Avenida Cassiopéia. Bairro: Jardim Satélite. Cidade:São José dos Campos. Estado: SP. Numero Logradouro: 591. Complemento: null
CEP: 12220000. Nome Rua: Avenida Presidente Juscelino Kubitschek. Bairro: Vila Industrial. Cidade:São José dos Campos. Estado: SP. Numero Logradouro: 6005. Complemento: null
CEP: 12210010. Nome Rua: Rue Cláudio Pinto. Bairro: Centro. Cidade:São José dos Campos. Estado: SP. Numero Logradouro: 29. Complemento: null
CEP: 12230000. Nome Rua: Avenida Andromeda. Bairro: Jardim Satélite. Cidade:São José dos Campos. Estado: SP. Numero Logradouro: 227. Complemento: null
CEP: 12240540. Nome Rua: Avenida Cassiano Ricardo. Bairro: Jardim Alvorada. Cidade:São José dos Campos. Estado: SP. Numero Logradouro: 1501. Complemento: null
CEP: 12215656. Nome Rua: Avenida Nair Toledo de Mira. Bairro: Jardim Paulista. Cidade:São José dos Campos. Estado: SP. Numero Logradouro: 4496. Complemento: null

Estr. Mun. Martins Guimarães, 1050 - Vila Industrial
Av. Pres. Juscelino Kubitschek, 6005 - Vila Industrial
Av. Nair Toledo de Mira, São José dos Campos, SP
R. Cláudio Pinto, 29 - Centro, São José dos Campos, SP
Av. Andromeda, 227 - Jardim Satélite, São José dos Campos, SP
Av. Cassiano Ricardo, 1501 - Jardim Alvorada, São José dos Campos, SP
Av. Nair Toledo de Mira, São José dos Campos, SP
Estr. Mun. Martins Guimarães, 1050 - Vila Industrial

Fonte: Autor (2018)

A Rota gerada pelo SysRLog apresentou o resultado desejado, definido na Tabela 57. A otimização conseguiu obter resultados satisfatórios, tendo uma menor distância e um menor tempo para ser percorrida que o lançamento desses pontos diretamente no Google Maps. A Tabela 58 apresenta o comparativo entre as duas rotas.

Tabela 58. Resultados Obtidos no Caso de Teste 2.

Resultado Roteirização - Caso de Teste 2 - Cidade de São José dos Campos				
	Valor obtido GoogleMaps	Valor Obtido SysRLog	Diferença	Percentual de Redução
Tempo(horas)	1:48	1:28	0:20	18,52%
Distância(KMs)	68,1	46,9	21,2	31,13%

Fonte: Autor (2018)

4.3.3. Caso de Testes 3 - Cidade de Taubaté

O Terceiro Caso de Teste realizado teve como base uma empresa fictícia com atuação na Cidade de Taubaté. A Tabela 59 apresenta o detalhamento dos Caso de Teste, Relacionando os endereços envolvidos e o resultado esperado:

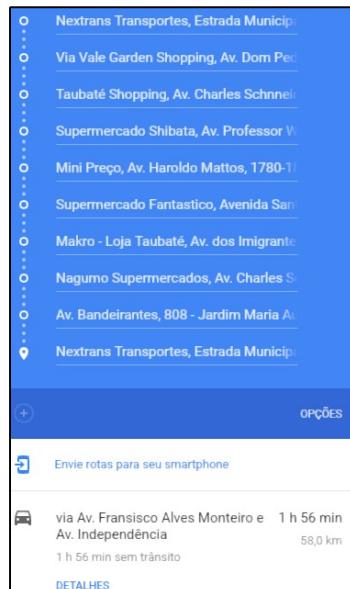
Tabela 59. Detalhamento do Caso de Teste 3.

ID	Descrição do Cenário de Teste	Nome dos Pontos Envolvidos (Sequência Inserida No Maps)	Resultado esperado
CT3	Cenário de Teste 3 :Esta validação tem por objetivo a definição de 9 pontos atendidos pela Empresa 3 na Cidade de Taubaté- São Paulo. Sendo esses pontos compostos pela Origem e Destino que é o endereço da empresa e 9 pontos Intermediários	P0 - Estrada Municipal João Gadioli, 1330 - Quiririm P1 - Av. Dom Pedro I, 7181 - Res. Estoril P2 - Av. Charles Schnneider, 1700 - Vila Costa P3 - Av. Professor Walter Taumaturgo, 1160 - Centro P4 - Av. Haroldo Mattos, 1780 - Parque Sr. do Bonfim, P5 - Avenida Santa Cruz, 384 - Areão P6 - Av. dos Imigrantes, 111 - Conj. Res. Quiririm, P7 - Av. Charles Schnneider, 420 - Parque Sr. do Bonfim P8 - Av. Bandeirantes, 808 - Jardim Maria Augusta P0 - Estrada Municipal João Gadioli, 1330 - Quiririm	P0 -> P6 -> P4 -> P2 -> P7 -> P3 -> P8 -> P5 -> P1 -> P0

Fonte: Autor (2018)

A Figura 93 apresenta a Rota gerada pelo Google Maps após a inserção dos pontos conforme a ordem apresentada na Tabela 59.

Figura 93. Caso de Testes 3 - Rota Gerada pelo Google Maps.



Fonte: Autor (2018)

A Figura 94 apresenta a Rota que foi gerada com a Otimização do algoritmo Desenvolvido no Projeto. No lado esquerdo da imagem, é a tela de consulta a rota Gerada do Software e do lado direito é ela aberta no Google Maps.

Figura 94. Caso de Testes 3 - Rota Gerada pelo SysRLog.

<p>Informações</p> <p>Identificação da Rota : 25 Usuário Criador : transportadoratbt@gmail.com Data de Criação : 27/10/2018</p> <p>Endereços da Rota</p> <p>CEP: 12040900. Nome Rua: Avenida Charles Schneider. Bairro: Parque Senhor do Bonfim. Cidade:Taubaté. Estado: SP. Numero Logradouro: 1700. Complemento: null CEP: 12091000. Nome Rua: Avenida Dom Pedro I. Bairro: Jardim Baronesa. Cidade:Taubaté. Estado: SP. Numero Logradouro: 7181. Complemento: null CEP: 12030040. Nome Rua: Avenida Professor Walter Thaumaturgo. Bairro: Jardim das Nações. Cidade:Taubaté. Estado: SP. Numero Logradouro: 1160. Complemento: null CEP: 12040670. Nome Rua: Avenida Haroldo Mattos. Bairro: Esplanada Independência. Cidade:Taubaté. Estado: SP. Numero Logradouro: 1780. Complemento: null CEP: 12043490. Nome Rua: Avenida dos Imigrantes. Bairro: Quiririm. Cidade:Taubaté. Estado: SP. Numero Logradouro: 111. Complemento: null CEP: 12040000. Nome Rua: Avenida Charles Schneider. Bairro: Parque Senhor do Bonfim. Cidade:Taubaté. Estado: SP. Numero Logradouro: 420. Complemento: null CEP: 12061100. Nome Rua: Avenida Santa Cruz do Areao. Bairro: Vila Areao. Cidade:Taubaté. Estado: SP. Numero Logradouro: 384. Complemento: null CEP: 12070100. Nome Rua: Avenida Bandeirantes. Bairro: Jardim Maria Augusta. Cidade:Taubaté. Estado: SP. Numero Logradouro: 808. Complemento: null</p>	<p>The screenshot shows a list of waypoints in Google Maps:</p> <ul style="list-style-type: none"> Nextrans Transportes, Estrada Municipal Av. dos Imigrantes, 111 - Centro, Taubaté Av. Haroldo Mattos, 1780 - Parque Sr. do Rio Av. Charles Schneider, 1700 - Parque São Paulo Av. Charles Schneider, 420 - Parque São Paulo Av. Professor Walter Thaumaturgo, 1160 Av. Bandeirantes, 808 - Jardim Maria Augusta Av. Santa Cruz do Areão, 384 - Vila Areão Av. Dom Pedro I, 7181 - Granjas Reunidas Nextrans Transportes, Estrada Municipal <p>Below the list are "OPÇÕES" and "Envie rotas para seu smartphone". Under "DETALHES", it shows the route via Av. Carlos Pedroso da Silveira, 1 h 13 min, 36,0 km.</p>
--	---

Fonte: Autor (2018)

A Rota gerada pelo SysRLog apresentou o resultado desejado, definido na Tabela 59. A otimização conseguiu obter resultados satisfatórios, tendo uma menor distância e um menor tempo para ser percorrida que o lançamento desses pontos diretamente no Google Maps. A Tabela 60 apresenta o comparativo entre as duas rotas.

Tabela 60. Resultados Obtidos no Caso de Teste 3.

Resultado Roteirização - Caso de Teste 3 - Cidade de Taubaté				
	Valor obtido GoogleMaps	Valor Obtido SysRLog	Diferença	Percentual de Redução
Tempo(horas)	58	36	22	37,93%
Distância(KMs)	1:56	1:13	0:43	37,07%

Fonte: Autor (2018)

4.3.4. Caso de Testes 4 - Cidade de Jacareí

O Quarto Caso de Teste realizado teve como base uma empresa fictícia com atuação na Cidade de Jacareí. A Tabela 61 apresenta o detalhamento dos Caso de Teste, relacionando os endereços envolvidos e o resultado esperado:

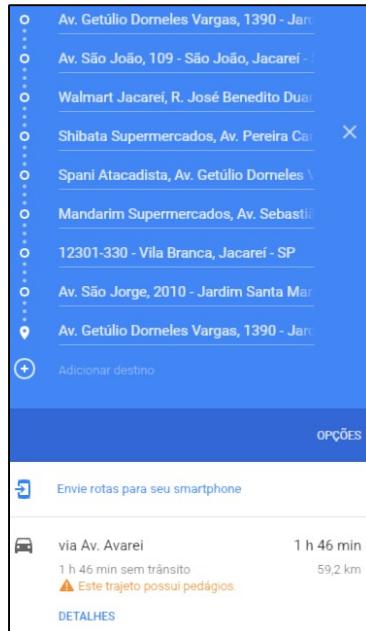
Tabela 61. Detalhamento do Caso de Teste 4.

ID	Descrição do Cenário de Teste	Nome dos Pontos Envolvidos (Sequência Inserida No Maps)	Resultado esperado
CT4	Cenário de Teste 4 :Esta validação tem por objetivo a definição de 7 pontos atendidos pela Empresa 4 na Cidade de Jacareí- São Paulo. Sendo esses pontos compostos pela Origem e Destino que é o endereço da empresa e 7 pontos Intermediários	P0 -Av. Getúlio Dorneles Vargas, 1390 - Jardim California P1 - Av. São João, 109 - São João P2 -R. José Benedito Duarte, 88 - Parque Itamarati P3 - Av. Pereira Campos, 291 - Jardim Didinha, P4 - Av. Getúlio Dorneles Vargas, 1591 - Jardim California, P5 - Av. Sebastião Lopes, 42 - Jardim Nova Esperança P6 -Avenida das Letras, Loteamento Villa Branca P7 - Av. São Jorge, 2010 - Jardim Santa Marina P0 -Av. Getúlio Dorneles Vargas, 1390 - Jardim California	P0 -> P4 -> P2 -> P1 -> P3 -> P5 -> P6 -> P7 -> P0

Fonte: Autor (2018)

A Figura 95 apresenta a Rota gerada pelo Google Maps após a inserção dos pontos conforme a ordem apresentada na Tabela 61.

Figura 95. Caso de Testes 4 - Rota Gerada pelo Google Maps.



Fonte: Autor (2018)

A Figura 96 apresenta a Rota que foi gerada com a Otimização do algoritmo Desenvolvido no Projeto. No lado esquerdo da imagem, é a tela de consulta a rota Gerada do Software e do lado direito é ela aberta no Google Maps.

Figura 96. Caso de Testes 4 - Rota Gerada pelo SysRLog.

Informações

Identificação da Rota : 21
Usuário Criador : transportadorajac@gmail.com
Data de Criação : 24/10/2018

Endereços da Rota

CEP: 12307200. Nome Rua: Rua José Benedito Duarte. Bairro: Parque Itamarati. Cidade:Jacareí. Estado: SP Numero Logradouro: 88. Complemento: null
CEP: 12305000. Nome Rua: Avenida Getúlio Vargas. Bairro: Jardim Califórnia. Cidade:Jacareí. Estado: SP Numero Logradouro: 1591. Complemento: null
CEP: 12320670. Nome Rua: Avenida Pereira Campos. Bairro: Jardim Didinha. Cidade:Jacareí. Estado: SP Numero Logradouro: 291. Complemento: null
CEP: 12324350. Nome Rua: Avenida Sebastião Lopes. Bairro: Jardim Nova Esperança. Cidade:Jacareí. Estado: SP Numero Logradouro: 42. Complemento: null
CEP: 12322000. Nome Rua: Avenida São João. Bairro: São João. Cidade:Jacareí. Estado: SP Numero Logradouro: 109. Complemento: null
CEP: 12312000. Nome Rua: Avenida São Jorge. Bairro: Cidade Salvador. Cidade:Jacareí. Estado: SP Numero Logradouro: 2010. Complemento: null
CEP: 12301330. Nome Rua: Avenida das Letras. Bairro: Loteamento Villa Branca. Cidade:Jacareí. Estado: SP Numero Logradouro: null. Complemento: null

The sidebar shows the same list of waypoints as Figure 95. Below the waypoints are options to "Enviar rotas para seu smartphone" and "DETALHES". A summary at the bottom indicates a route via Av. Getúlio Dorneles Vargas, 1 h 2 min, 35,1 km, with a note: "Este trajeto possui pedágios."

Fonte: Autor (2018)

A Rota gerada pelo SysRLog apresentou o resultado desejado, definido na Tabela 61. A otimização conseguiu obter resultados satisfatórios, tendo uma menor distância e um menor tempo para ser percorrida que o lançamento desses pontos diretamente no Google Maps. A Tabela 62 apresenta o comparativo entre as duas rotas.

Tabela 62. Resultados Obtidos no Caso de Teste 4.

Resultado Roteirização - Caso de Teste 4 - Cidade de Jacareí				
	Valor obtido GoogleMaps	Valor Obtido SysRLog	Diferença	Percentual de Redução
Tempo(horas)	51,2	35,1	16,1	31,45%
Distância(KMs)	1:40	1:02	0:38	38,00%

Fonte: Autor (2018)

4.3.5. Caso de Testes 5 - Cidade de Caraguatatuba

O último Caso de Teste realizado teve como base uma empresa fictícia com atuação na Cidade de Caraguatatuba. A Tabela 63 apresenta o detalhamento dos Caso de Teste, relacionando os endereços envolvidos e o Resultado esperado:

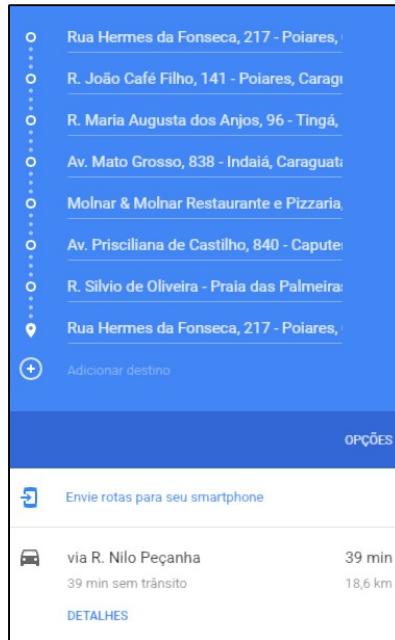
Tabela 63. Detalhamento do Caso de Teste 35.

ID	Descrição do Cenário de Teste	Nome dos Pontos Envolvidos (Sequência Inserida No Maps)	Resultado esperado
CT5	Cenário de Teste 5 :Esta validação tem por objetivo a definição de de 6 pontos atendidos pela Empresa 4 na Cidade de Caraguatatuba- São Paulo. Sendo esses pontos compostos pela Origem e Destino que é o endereço da empresa e 6 pontos Intermediários	P0 - Rua Hermes da Fonseca, 217 - Poiares P1 - Av. Prisciliana de Castilho, 840 - Caputera P2 - R. Silvio de Oliveira - Praia das Palmeiras P3 - Av. Pres. Campos Sales, 297 - Jaraguazinho, P4 - R. João Café Filho, 141 - Poiares P5 - R. Maria Augusta dos Anjos, 96 - Tingá P6 - Av. Mato Grosso, 838 - Indaiá P0 - Rua Hermes da Fonseca, 217 - Poiares	P0 -> P4 -> P5 -> P6 -> P3 -> P1 -> P2 -> P0

Fonte: Autor (2018)

A Figura 88 apresenta a Rota gerada pelo Google Maps após a inserção dos pontos conforme a ordem apresentada na Tabela 63.

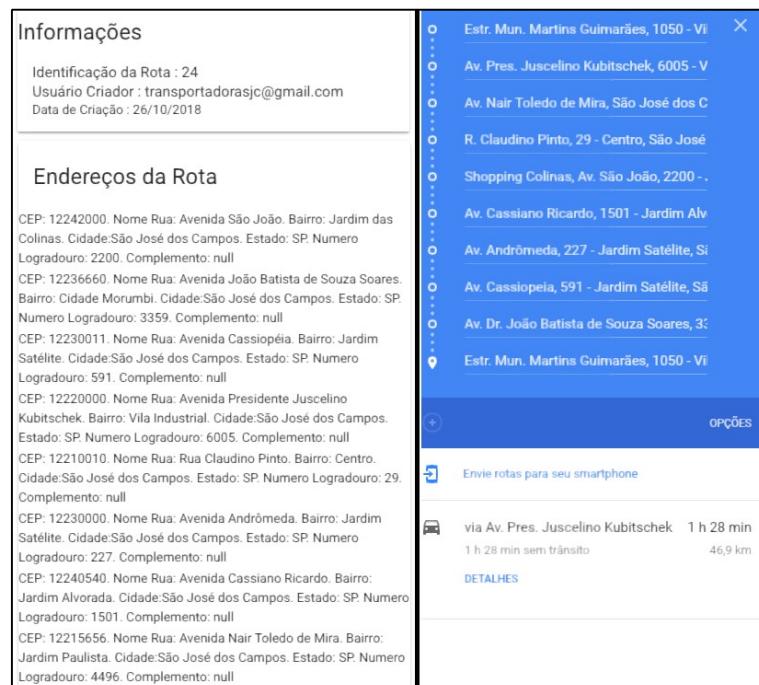
Figura 88. Caso de Testes 5 - Rota Gerada pelo Google Maps.



Fonte: Autor (2018)

A Figura 98 apresenta a Rota que foi gerada com a Otimização do algoritmo Desenvolvido no Projeto. No lado esquerdo da imagem, é a tela de consulta a rota Gerada do Software e do lado direito é ela aberta no Google Maps.

Figura 98. Caso de Testes 5 - Rota Gerada pelo SysRLog.



Fonte: Autor (2018)

A Rota gerada pelo SysRLog apresentou o resultado desejado, definido na Tabela 63. A otimização conseguiu obter resultados satisfatórios, tendo uma menor distância e um menor tempo para ser percorrida que o lançamento desses pontos diretamente no Google Maps. A Tabela 64 apresenta o comparativo entre as duas rotas.

Tabela 64. Resultados Obtidos no Caso de Teste 5.

Resultado Roteirização - Caso de Teste 5 - Cidade de Caraguatatuba				
	Valor obtido GoogleMaps	Valor Obtido SysRLog	Diferença	Percentual de Redução
Tempo(horas)	0:45	0:39	0:06	13,33%
Distância(KMs)	23,1	18,6	4,5	19,48%

Fonte: Autor (2018)

4.3.6. Consolidação dos Resultados Obtidos nos casos de Teste

A seguir serão apresentadas as Tabelas 52 e 53, consolidando os resultados obtidos com o teste do algoritmo de roteirização, frente a rota gerada pelo Google Maps.

A Tabela 65 apresenta os dados referentes ao tempo da Rota Gerada no Google Maps e o tempo da Rota gerada pelo Software. O percentual de Redução variou de 13,33% na menor redução de tempo á 38% na rota que teve maior percentual de redução. Em todas as rotas geradas testadas, o Software e Roteirização gerou uma rota mais rápida de ser percorrida.

Tabela 65. Tabela com os Resultados Obtidos no Comparativo de Tempo das Rotas.

Resultado Roteirização - Tempo em Horas e Minutos				
Testes	Tempo Maps	Tempo SysRLog	Diferença	Percentual de Redução
Teste 1 - Caçapava	0:53	0:41	0:12	22,64%
Teste 2 - São José dos Campos	1:48	1:28	0:20	18,52%
Teste 3 - Tabaté	1:54	1:17	0:37	32,46%
Teste 4 - Jacareí	1:46	1:02	0:38	38,00%
Teste 5 - Caraguatatuba	0:45	0:39	0:06	13,33%

Fonte: Autor (2018)

Analizando a Tabela 66 pode ser observado que o menor percentual de redução na distância percorrida, foi de 19,48% e o melhor percentual atingiu 43,40%. Assim como na Análise de tempo, no parâmetro de distância a roteirização pelo Software desenvolvido, também foi mais eficiente entregando uma distância menor a ser percorrida, e consequentemente reduzindo os custos com combustível e desgaste do veículo.

Tabela 66. Tabela com os Resultados Obtidos no Comparativo de Distância das Rotas

Resultado Roteirização - Distância				
Testes	Distância Maps	Distância SysRLog	Diferença	Percentual de Redução
Teste 1 - Caçapava	21,4	16,4	5	23,36%
Teste 2 - São José dos Campos	65,9	37,3	28,6	43,40%
Teste 3 - Tabaté	66,3	37,8	28,5	42,99%
Teste 4 - Jacareí	59,2	35,1	16,1	31,45%
Teste 5 - Caraguatatuba	23,1	18,6	4,5	19,48%

Fonte: Autor (2018)

5. CONCLUSÃO

O presente Capítulo tem como objetivo apresentar a conclusão do Projeto, citando as Principais Contribuições que o Projeto desenvolvido possa vir a gerar direta ou indiretamente. Também será apresentada as Considerações Finais a cerca do trabalho, as limitações e dificuldades ocorridas durante o desenvolvimento do Projeto. E encerrando será apresentada Sugestões para trabalhos futuros

5.1. Principais Contribuições

A seguir serão apresentadas as principais contribuições proporcionadas pelo Projeto desenvolvido, tanto do ponto de vista comercial quanto do ponto de vista acadêmico:

- Reduzir custos operacionais com transporte em empresas baseado em rotas mais eficientes do que uma roteirização realizada manualmente.
- Reduzir tempo desperdiçado, indicando as entregas que não fazem parte da região de atuação da empresa no momento da geração da rota
- Proporcionar melhoria nos serviços de entrega da empresa, tornando-os mais eficientes e proporcionando mais entregas em menos tempo com menores custos operacionais
- Aumentar a rentabilidade da empresa por reduzir seus custos de operação
- Prover agilidade no trabalho das empresas, por poderem utilizar a aplicação no formato de API, não necessitando acessar a utilizar a aplicação, caso o Software da empresa desenvolva a conexão com a API.
- Prover a capacidade da empresa tornar-se mais competitiva no mercado
- Auxiliar alunos que ainda tenham dúvidas de quais tecnologias utilizar no desenvolvimento do seu Trabalho de Graduação.
- Guiar alunos quanto ao funcionamento de uma aplicação com base em Spring Framework e Arquitetura de camadas
- Apresentar aos alunos um framework para desenvolvimento de aplicativos híbridos, Cordova e Ionic e provendo uma explicação básica sobre seu funcionamento.

5.2. Considerações Gerais, Limitações e Dificuldades

Este Trabalho teve como objetivo o desenvolvimento de um Software para roteirização. Na pesquisa preliminar, identificou-se uma grande preocupação por parte das empresas logísticas em reduzir o custo com os transportes, além de comprovar que uma roteirização básica é capaz de aumentar a produtividade de uma empresa. Essa pesquisa foi utilizada como justificativa e motivação para o desenvolvimento do projeto.

O desenvolvimento da aplicação foi de grande enriquecimento acadêmico e profissional, devida a aplicação prática de todas as etapas de desenvolvimento de um Projeto. Desde a fase de escolha da metodologia de trabalho, levantamento de requisitos, passando pela seleção das tecnologias e desenvolvimento do Software, e utilização das tecnologias escolhidas para prover uma aplicação que atendesse os requisitos elencados. Encerrando com a fase de testes, coleta de resultados e validação.

Ao realizar cada uma das etapas, torna mais nítida a importância de uma boa execução de cada uma delas. Não há como desenvolver um Software sem uma metodologia de desenvolvimento bem aplicada e sem gestão do desenvolvimento. O Levantamento de requisitos tem que ser devidamente especificado para que não ocorram problemas no Desenvolvimento. As tecnologias devem ser bem selecionadas e mensuradas para que consigam atender a demanda e suportar a aplicação. E os testes devem ser executados à fim de validar devidamente a aplicação. Mas para conseguir alcançar esse entendimento é necessário executar essas etapas, por isso que o Desenvolvimento desse Projeto foi enriquecedor

Em todas as etapas houveram desafios e dificuldades a serem superados. Existem diversas Metodologias de Desenvolvimento para serem escolhidas e cada uma delas com particularidades, selecionar uma é um trabalho cuidadoso. A fase de levantamento de requisitos necessita de muita atenção e cuidado para que a especificação seja correta e forneça todas as informações relevantes ao desenvolvedor.

A fase de desenvolvimento é uma fase difícil, por mais que os requisitos tenham sido devidamente elencados, a partir do momento em que há o inicio da escrita do código irá ocorrer erro em algum ponto do projeto ou irão ocorrer dúvidas de como aplicar os *frameworks* na lógica do Projeto. Investigar o erro requer tempo e esforço, mas a comunidade de desenvolvimento nos fóruns na Internet é incrível e sempre a disposta ajudar.

Com os erros foi identificada a importância dos testes. Devido ao tempo, não foi possível aplicar testes de unidade minuciosamente. Os testes foram aplicados na última camada, a dos controladores assim consequentemente as camadas de serviço e repositório.

Já o desenvolvimento do front-end foi a etapa mais difícil do Projeto, devido não haver familiaridade com as tecnologias á serem utilizadas. Foi necessária muita pesquisa para o entendimento de como a programação assíncrona funciona, que os recursos solicitados poderão não estar disponíveis em certos pontos do código. Mas novamente, a comunidade de desenvolvimento tem muita informação para ser estudada e as pesquisas realizadas conseguiram prover conhecimento necessário para finalizar o desenvolvimento.

Durante os testes do front-end foi identificada uma limitação que não é devidamente divulgada pelo Google Maps. A restrição da URL aceitar no máximo dez pontos para roteirização, contanto o destino final. Devido a essa limitação que o Google Maps oferece, no front-end, a quantidade de pontos ser inserida teve que ser limitada a nove pontos.

Com a aplicação finalizada, foi realizado o teste de confrontar as rotas geradas aleatoriamente no Google Maps, com as rotas geradas dentro do Software desenvolvido. Os resultados obtidos comprovaram a eficácia do algoritmo implementado no back-end fornecendo um percentual de redução de no tempo de 13,33% à 38% e uma redução em distâncias à serem percorridas de 19,48% à 43,40%. Esses resultados também validam o cumprimento do objetivo do projeto, a entrega de um software que execute uma otimização em rotas.

5.3. Sugestão de trabalhos futuros

Neste subcapítulo, serão apresentados as sugestões de trabalhos futuros, que foram identificados durante o desenvolvimento do Projeto, e transcendem os objetivos iniciais.

1. Transferir a utilização do mapa do Google Maps para o Bing Maps.

Devido ao Google Maps limitar os pontos de rota com no máximo 10 pontos, contando o destino final, uma opção é ao invés de usar o Google Maps como mapa para a rota gerada pelo back-end, utilizar o Bing Maps da Microsoft. A aplicação Bing Maps suporta até 25 pontos para montagem de uma rota.

Atualmente os ceps que são recebidos para roteirização são, após serem selecionados se fazem parte ou não da região de atuação da empresa (caso a empresa tenha essa parametrização), transferidos para o método de cálculo das distâncias, utilizando a API DistanceMatrix. Após gerada a rota mais otimizada, essa lista ordenada das entregas é

transformada de cep e número, para um endereço com número. E por final é gerada a URL que abre no Google Maps.

Para utilizar a BingMaps, deverá ser realizada uma alteração nesse Fluxo. O Bing não trabalha com endereço, apenas com geolocalização. Nesse caso ao invés do conjunto cep e número ser transformado em endereço, ele deverá consultar uma API do Bing, para retornar a Latitude Longitude do endereço. Ai a partir dessa lista é possível montar uma URL válida para utilização do Bing Maps.

2. Desenvolver exclusão lógica dos objetos:

Devido a arquitetura do banco de dados e objetos que foi desenvolvido para o Projeto, há muitas situações de ligação entre as tabelas, o que impossibilita a exclusão física de registros, pois ocasionaria perda de integridade referencial além de perda de histórico.

Para solucionar essa questão poderia ser desenvolvidas colunas dentro do banco de dados para informar se determinado registro foi excluído ou não. Sendo necessária validação e filtragem nas Classes de repositórios e de Serviço.

3. Migração do mecanismo de Banco de Dados:

Utilizar Banco de Dados Oracle ou SQLServer para armazenagem dos dados, por serem Banco de Dados com melhor performance.

4. Tratamento dos Erros 404:

Foi utilizado um tratamento básico para os erros 404, poderá ser realizado um tratamento mais específico para cada exceção em cada Classe, além de deixar a apresentação com um visual mais agradável.

5. Aprimorar a Interface:

No Projeto, a paleta de cores da aplicação, animações e formatações dos componentes são em sua maioria as básicas do Ionic. Mesmo o Design estando fluído ele poderá ser aprimorado para prover uma interface mais agradável ao usuário

REFERÊNCIAS BIBLIOGRÁFICAS

- A MELHOR de cada Segmento.** Revista As Melhores do Transporte. **Editora OTM**, ano 14, no 14, novembro 2001.
- ANGULARJS. Add Some Control.** Disponível em: <https://angularjs.org>. Acesso em: 10/08/2018.
- APACHE. What is Maven?.** Disponível em: <https://maven.apache.org/what-is-maven.html>. Acesso em: 10/08/2018.
- BALLOU, R. H. Gerenciamento da cadeia de suprimentos: Logística empresarial.** 5 ed. Porto Alegre, Bookman, 2006.
- BARROS, T.; SILVA, M.; ESPÍNOLA, E. State MVC: Estendendo o padrão MVC para uso no desenvolvimento de aplicações para dispositivos móveis.** Em: Sexta Conferência Latino-Americana em Linguagens de Padrões para Programação. 2007.
- BOS, Bert. WHAT IS CSS?** Disponível em: <https://www.w3.org/Style/CSS/>. Acesso em: 10/08/2018.
- BRANSKI, R. M. O papel da tecnologia da informação no processo logístico:** estudo de caso com operadores logísticos. 2008. 252 f. Tese (Doutorado em Engenharia) – Escola Politécnica, Universidade de São Paulo, São Paulo.
- CAELUM. Java e Orientação a Objetos. Curso FJ11.** Disponível em : <https://www.caelum.com.br/download-apostilas>. Acesso em: 10/08/2018.
- CENTRO DE ESTUDOS EM LOGÍSTICA –CEL/COPPEAD. Panorama Logístico – Gestão do Transporte Rodoviário de Cargas nas Empresas - Práticas e Tendências,** 2007.
- CHOPRA, S.; MEINDL P. Gestão da Cadeia de suprimentos: Estratégias, Planejamentos e Operações.** 4^a Ed. São Paulo: Pearson, 2011.
- CHOPRA, S. MEINDL, P. Gerenciamento da cadeia de suprimento: Estratégia, planejamento e operação.** São Paulo: Prentice Hall, 2003.
- CORDOVA. Overview.** Disponível em: <https://cordova.apache.org/docs/en/latest/guide/overview/index.html>. Acesso em: 10/08/2018.
- Demaria, M.. "O operador de transporte multimodal com fator de otimização da logística."** (2004).
- DATICAL. Source Control for your Database.** Disponível em: <https://www.liquibase.org/index.html>. Acesso em: 10/08/2018.
- DORNIER, P. ERNST, R. FENDER, Michel. KOUVELIS, Panos. Logística e operações globais. Textos e casos.** São Paulo: Atlas, 2000.
- ECLEMMMA, JaCoCo Java Code Coverage Library.** Disponível em: <https://www.jacoco.org/jacoco/>. Acesso em: 10/08/2018.
- FLEURY, P. F. Vantagens competitivas e estratégicas no uso de operadores logísticos.** Revista TecnoLogística, São Paulo, ano V, n. 46, set. 1999.

FLEURY, P. F. **Vantagens Competitivas e Estratégicas no Uso de Operadores Logísticos. Logística Empresarial: a perspectiva brasileira.** Ed. Atlas S.A., São Paulo, 2000.

FRANCISCHINI, P.G.; AMARAL GURGEL, F. **Administração de materiais e do patrimônio.** São Paulo: Pioneira Thomson, 2002.

FRIENDS, Apache. **Sobre.** Disponível em: https://www.apachefriends.org/pt_br/about.html. Acessado em: 10/08/2018.

FOWLER, M.. **Inversion of Control Containers and the Dependency Injection pattern** - Disponível em: <https://martinfowler.com/articles/injection.html> . Acesso em: 05/09/2018.

ILOS. **Panorama “Custos Logísticos na Economia e nas Empresas no Brasil”.** Rio de Janeiro. 2012.

IONIC. **The dev-friendly app platform for building cross-platform apps with one codebase, for any device, with the web.** Disponível em: <https://ionicframework.com/what-is-ionic>. Acessado em: 10/08/2018.

JSON. **Introducing JSON.** Disponível em:<https://www.json.org>. Acesso em: 10/08/2018.

MACHLINE, C. **Cinco décadas de logística empresarial e administração da cadeia de suprimentos no Brasil.** Rev. adm. empres. vol.51 no.3 São Paulo May/June 2011. Disponível em: http://www.scielo.br/scielo.php?pid=S0034-75902011000300003&script=sci_arttext. Acesso em: 29 de mar 2017

MATOS JUNIOR, C. A.; NUNES, R. V.; ASSIS, C. W. C.; FONSECA, R. C.; ADRIANO; N. A.; SANTOS, G. P. **O papel da roteirização na redução de custos logísticos e melhoria do nível de serviço em uma empresa do segmento alimentício no Ceará.** In: Anais do Congresso Brasileiro de Custos-ABC. 2013.

MARQUES, K. de L.. **Back-end vs Front-end vs Full-Stack: qual é a melhor escolha?** Disponível em: <https://becode.com.br/back-end-front-end-full-stack/> . Acesso em: 06/09/2018

MICROSOFT. **Getting Started.** Disponível em: <https://code.visualstudio.com/docs>. Acesso em: 10/08/2018.

MySQL. **About MySQL.** Disponível em: <https://www.mysql.com/about/> .Acesso em: 10/08/2018.

NAZÁRIO, P. **A importância de sistemas de informação para a competitividade logística.** Rio de Janeiro: Centro de Estudos em Logística, Coppead, 1999.

PIVOTAL, **Main Projects.** Disponível em: <https://spring.io/projects>. Acesso em: 10/08/2018A.

PIVOTAL, **Spring Tools 4.** Disponível em: <https://spring.io/tools>. Acesso em: 10/08/2018B.

PERES, R. **ALGORITMO DE DIJKSTRA.** Disponível em: <https://www.revista-programar.info/artigos/algoritmo-de-dijkstra/>. Acesso em: 17/12/2018.

PERÇİN,S.; MIN, H. A hybrid quality fonction deployment and fuzzy decision-making methodology for the optimal selection of third-party logistics service providers.

International Journal of Logistics: Research and Applications, [S1], v. 16, n. 5, p.380-397 - 2013.

POSTDOT. **Postman's Tools Support Every Stage of the API Lifecycle.** Disponível em: <https://www.getpostman.com>. Acesso em: 10/08/2018.

POZO, H. **Administração de recursos materiais e patrimoniais: uma abordagem logística.** 6^a Ed. São Paulo: Atlas, 2010.

RIBEIRO, P.; Ferreira, ARAÚJO, K. **Logística e transportes: uma discussão sobre os modais de transporte e o panorama brasileiro.** XXII Encontro Nacional de Engenharia de Produção (2002).

ROMERO M, SOUZA D. **Gerenciamento da cadeia de suprimentos.** Revista Científica Emersão v.1, nº 1 – maio/2015 – p. 146-155 Porto Belo/ SC.

ROSA, A. **Gestão do transporte na logística de distribuição física: uma análise da minimização do custo operacional.** 2007. Tese de Doutorado. Dissertação (Mestrado). Departamento de Economia, Contabilidade e Administração, Universidade de Taubaté, SP, Brasil.

THAYER, R; DORFMAN, M. **System and Software Requirements Engineering - Second Edition.** Los Alamitos: IEEE Computer Society Press Tutorial, 2000. 528p.

TYPESCRIPT. TypeScript in 5 minutes. Disponível em: <https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html>. Acesso em: 10/08/2018.

SOMMERVILLE, I. **Engenharia de software.** Tradução: Ivan Bosnic e Kalinka G. O. Gonçalves; Revisão técnica: Kechi Hirama. 9 ed. São Paulo: Pearson Prentice Hall, 2011.

SONARCOURSE. Roadmap. Disponível em: <https://www.sonarqube.org/roadmap/>. Acesso em: 10/10/2018.

UDACITY. **Conheça as linguagens de programação mais utilizadas no Brasil e no Mundo** - Disponível em: <https://br.udacity.com/blog/post/linguagens-de-programacao-mais-usadas-no-brasil-e-no-mundo> Acesso em: 05/09/2018.

W3C. **HTML 5.3 Editor's Draft.** Disponível em: <http://w3c.github.io/html/introduction.html#history-1>. Acesso em: 10/08/2018.