Universidade Candido Mendes - UCAM

Desenvolvimento Orientado a Objetos com Java

Sergio Santrovitsch Dorneles

DESENVOLVIMENTO DE SISTEMA DE CONTROLE DE FROTAS COM TECNOLOGIAS JAVA

Prof. Dr. Carlos Frederico Fronza

RIO DE JANEIRO

2017

Universidade Candido Mendes - UCAM

Desenvolvimento Orientado a Objetos com Java

Sergio Santrovitsch Dorneles

DESENVOLVIMENTO DE SISTEMA DE CONTROLE DE FROTAS COM TECNOLOGIAS JAVA

Prof. Dr. Carlos Frederico Fronza

Trabalho de Conclusão apresentado ao curso de Pós-Graduação Latu Senso em Desenvolvimento Orientado a Objetos com Java, oferecido pela Universidade Candido Mendes - UCAM, como requisito parcial para obtenção do grau de especialista, sob a orientação do Prof. Dr. Carlos Frederico Fronza.

RIO DE JANEIRO

2017

Termo de Aprovação

Aluno: Sergio Santrovitsch Dorneles.

Título: Desenvolvimento de sistema de controle de frotas com tecnologias Java.

Trabalho de Conclusão apresentado ao curso de Pós-Graduação Latu Senso em Desenvolvimento Orientado a Objetos com Java, oferecido pela Universidade Candido Mendes - UCAM, como requisito parcial para obtenção do grau de especialista, sob a orientação do Prof. Dr. Carlos Frederico Fronza.

RIO DE JANEIRO, 18 de fevereiro de 2017.

**Banca examinadora:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Professor (a) orientador (a)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Professor (a) convidado (a)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Professor (a) convidado (a)

DESENVOLVIMENTO DE SISTEMA DE CONTROLE DE FROTAS COM TECNOLOGIAS JAVA

**Sergio Santrovitsch Dorneles[[1]](#footnote-1)**

Acadêmico do Programa de Pós-Graduação lato sensu “Desenvolvimento Orientado a Objetos com Java” da Universidade Candido Mendes – UCAM.

Resumo

O presente trabalho foi realizado com o intuito de elaborar uma solução computacional para o controle de veículos de uma empresa, utilizando três vertentes da linguagem Java - Web, Desktop e Mobile - e aplicando as boas práticas e conceitos estabelecidos durante este curso de pós-graduação. Inicialmente foi realizado o estudo para identificação das funcionalidades que o sistema deveria possuir, para que este atenda as necessidades de uma empresa de pequeno porte, sendo então realizada a Análise e Modelagem dos processos, bem como seus relacionamentos e comportamentos, utilizando-se da Linguagem Unificada de Modelagem (UML) para a representação dos mesmos em forma de diagramas. Após foi realizada a codificação utilizando a metodologia de desenvolvimento de software orientada a objetos, a qual engloba os processos de Análise, Projeto e Implementação, resultando em um sistema com 3 módulos: uma interface Web voltada para a parte gerencial; uma interface Desktop para o setor de oficina; e uma interface móvel a ser utilizada pelos operadores de portaria. Para o armazenamento dos dados foi optado por utilizar o banco de dados PostgreSQL em conjunto com o framework Hibernate, possibilitando o uso da linguagem nativa de programação Java para a persistência dos objetos da aplicação sem necessidade de conversão de dados ou o uso de comandos SQL.

**Palavras-chave:** Sistema de Informação. Java. Persistência Orientada a Objeto.

Abstract

This program was designated to address the development of a computational solution for vehicle control in a small business, using three different Java platforms - Web, Desktop and Mobile - and applying the good practices and concepts established during this specialization course. The idea was to identify the functionalities that the system should have, so that it meets the needs of a small company, then go through the Analysis and Modeling phase, establishing entities relationships and behaviors, using the Unified Language Modeling (UML) to represent them in diagrams. After that, the solution was coded using object-oriented software development methodology, which includes Analysis, Design and Implementation processes, resulting in a system with 3 modules: a Web user interface focused on the management side of the business; a desktop interface for garage; And a mobile application to be used in exit/entrance control. For data storage, PostgreSQL relational database was used along with Hibernate framework, allowing the use of native Java programming language for data persistence, making it unnecessary to handle data conversion or SQL commands.

**Keywords:** Information System. Java. Object-Oriented Persistence.

Lista de Figuras

Xxxxxxxxx

Lista de Abreviaturas e Siglas

xxxx

**Sumário**

[1. Introdução 10](#_Toc473926970)

[1.1. Considerações iniciais 10](#_Toc473926971)

[1.2. Objetivos 11](#_Toc473926972)

[1.2.1. Objetivo geral 11](#_Toc473926973)

[1.2.2. Objetivos específicos 11](#_Toc473926974)

[1.3. Justificativa 12](#_Toc473926975)

[1.4. Estrutura do trabalho 12](#_Toc473926976)

[2. Referencial Bibliográfico 12](#_Toc473926977)

[2.1. Java 13](#_Toc473926978)

[2.2. UML 13](#_Toc473926979)

[2.2.1. Diagrama de caso de uso 13](#_Toc473926980)

[2.2.2. Diagrama de classe 14](#_Toc473926981)

[2.2.3. Diagrama de sequência 14](#_Toc473926982)

[2.3. Padrão MVC 14](#_Toc473926983)

[2.4. Servlets 15](#_Toc473926984)

[2.5. JavaServer Pages 15](#_Toc473926985)

[2.6. JSTL 16](#_Toc473926986)

[2.7. PostgreSQL 16](#_Toc473926987)

[2.8. Hibernate 17](#_Toc473926988)

[2.9. Netbeans 17](#_Toc473926989)

[2.10. Servidor de aplicação GlassFish 17](#_Toc473926990)

[3. Desenvolvimento 19](#_Toc473926991)

[3.1. Modelagem comum aos 3 módulos 19](#_Toc473926992)

[3.2. Modelo relacional 19](#_Toc473926993)

[3.3. Módulo Web 19](#_Toc473926994)

[3.3.1. Modelagem do módulo web 19](#_Toc473926995)

[3.3.2. Diagrama de caso de uso 19](#_Toc473926996)

[3.3.3. Diagrama de classes 19](#_Toc473926997)

[3.3.4. Diagrama de sequência 19](#_Toc473926998)

[3.3.5. Páginas do módulo 19](#_Toc473926999)

[3.4. Módulo desktop 19](#_Toc473927000)

[3.4.1. Modelagem do módulo desktop 19](#_Toc473927001)

[3.4.2. Diagrama de caso de uso 19](#_Toc473927002)

[3.4.3. Diagrama de classes 19](#_Toc473927003)

[3.4.4. Diagrama de sequência 19](#_Toc473927004)

[3.4.5. Telas do módulo 19](#_Toc473927005)

[3.5. Módulo mobile 19](#_Toc473927006)

[3.5.1. Modelagem do módulo mobile 19](#_Toc473927007)

[3.5.2. Diagrama de caso de uso 20](#_Toc473927008)

[3.5.3. Diagrama de classes 20](#_Toc473927009)

[3.5.4. Diagrama de sequência 20](#_Toc473927010)

[3.5.5. Telas do módulo 20](#_Toc473927011)

[4. Conclusão 20](#_Toc473927012)

[Referências 21](#_Toc473927013)

[Apêndice A – Script de criação do banco de dados 23](#_Toc473927014)

[Apêndice B – Classes Java – Pacote Model 28](#_Toc473927015)

[Apêndice C – Classes Java – Pacote Controller 28](#_Toc473927016)

[Apêndice D – Classes Java – Pacote View 28](#_Toc473927017)

[Apêndice E – Classes Java – Pacote DAO 28](#_Toc473927018)

[Apêndice F – Classes Java – Pacote Servlet 28](#_Toc473927019)

[Apêndice G – Classes Java – Pacote Filter 28](#_Toc473927020)

[Apêndice H – Páginas JSP 28](#_Toc473927021)

# Introdução

Este capítulo apresenta as considerações iniciais, os objetivos e a justificativa da realização deste trabalho. No final do capítulo é apresentada a organização do texto por meio de uma breve descrição dos seus capítulos

## Considerações iniciais

A utilização dos sistemas de informação está cada dia mais presente nos diversos tipos de organizações, trazendo melhorias em seus processos produtivos e gerenciais. No cenário da tecnologia atual, o acesso à informação de maneira rápida e precisa é fundamental para o bom funcionamento de uma empresa, seja ela pública ou privada. As informações têm importância cada vez mais elevada e modificam o modo de trabalho e as ações cotidianas. O desenvolvimento desses sistemas de informação de maneira eficiente, permitindo ao usuário ter acesso às informações que ele precisa da melhor maneira possível, portanto, é uma das atividades fundamentais do profissional de tecnologia da informação.

Utilizando a metodologia de desenvolvimento com base na orientação a objetos, foram identificados os principais processos internos de controle e operação do setor de veículos de uma organização, passando por sua administração, oficina e controle de entrada e saída.

O presente artigo tem por finalidade desenvolver um sistema de informação para o controle e gerenciamento de frotas veiculares para pequenas empresas, com funcionalidades de cadastro de usuário, cadastro de veículo, cadastro de revisão, cadastro de saída e retorno, realização de revisão, além de fornecer relatórios gerenciais.

Para aproveitar suas características de funcionamento em diversos sistemas operacionais e em dispositivos diferentes, a linguagem de programação Java foi amplamente utilizada no desenvolvimento da solução, culminando em um módulo Web para a administração, um módulo desktop para a oficina e um módulo mobile para a portaria, permitindo passar pelas principais plataformas de uso da tecnologia. Aliadas ao Java, foram empregadas diversas outras ferramentas e conceitos bem estabelecidos para o desenvolvimento do sistema multiplataforma: Netbeans IDE, padrão MVC, Servlets, JSP, JSTL, Hibernate, banco de dados, UML, servidor de aplicação, entre outros.

## Objetivos

O objetivo geral está relacionado ao resultado principal que é esperado da realização deste trabalho. Já os objetivos específicos complementam o objetivo geral em termos de funcionalidades do sistema.

### Objetivo geral

Desenvolver um sistema computacional para gerenciar pequenas frotas, permitindo um melhor controle gerencial e operacional da manutenção veicular.

### Objetivos específicos

O sistema deverá prover as seguintes funcionalidades:

* Cadastro de veículos de propriedade da organização, através de uma aplicação Web;
* Cadastro de funcionários, de acordo com sua função na organização, através de uma aplicação Web;
* Atribuição de revisão veicular a um funcionário pela administração, através de uma aplicação Web;
* Realização de revisão veicular por um funcionário, através de uma aplicação desktop;
* Controle de entrada e saída de veículos, permitindo histórico de motorista, quilometragem, itinerário, data e hora em uma aplicação mobile;
* Emissão de relatórios gerenciais para a administração, através de uma aplicação Web.

## Justificativa

A falta de praticidade no uso de fichas impressas ou planilhas eletrônicas para o controle da manutenção veicular é a justificativa principal para a realização deste trabalho. O benefício que o sistema é capaz de prover através da análise e processamento da informação estruturada de maneira automatizada é outro diferencial que se torna possível com o auxílio da tecnologia.

Para o administrador, o sistema torna mais fácil ter uma visão geral sobre seus veículos e funcionários, auxiliando a tomada de decisão.

Para os funcionários da oficina, é possível manter um histórico de revisões de cada veículo e organizar de maneira mais eficiente seu processo de trabalho.

Por fim, para os funcionários da portaria, o sistema permite uma maneira mais segura e eficaz de verificar se um veículo tem condições e autorização de sair pelos portões da organização.

## Estrutura do trabalho

O presente artigo está organizado em XX capítulos além desta introdução. O capítulo 2 apresenta os conceitos e tecnologias empregados na construção da solução. O capítulo 3 aborda o desenvolvimento do sistema, incluindo a metodologia aplicada e a implementação de cada uma das etapas do projeto. Encerrando o texto, o capítulo 4 discorre sobre as conclusões obtidas, com ênfase nos objetivos alcançados.

# Referencial Bibliográfico

Nesta seção serão abordadas as principais tecnologias e conceitos empregados para alcançar os objetivos propostos no artigo.

## Java

O desenvolvimento do sistema será feito utilizando a linguagem de programação Java, através da plataforma Java Enterprise Edition (Java EE). A linguagem permite a utilização dos conceitos de orientação a objetos, como herança, polimorfismo, encapsulamento e abstração, facilitando o processo de desenvolvimento. Outro fator favorável ao Java é a quantidade de frameworks disponíveis para a linguagem, tornando o desenvolvimento de qualquer aplicação, seja ela Web, desktop ou mobile mais rápido e eficiente.

Segundo Gonçalves (2007), a linguagem Java é utilizada por empresas que desejam trafegar uma grande quantidade de dados e necessitam de estabilidade e portabilidade.

Já para Deitel (2005), o Java tornou-se a linguagem preferida para implementar aplicativos baseados na internet e software para dispositivos que se comunicam em uma rede. Ele não é mais utilizado simplesmente para tornar as páginas da World Wide Web mais dinâmicas. Tornou-se a linguagem preferida para atender às necessidades de programação corporativas de muitas empresas.

## UML

A Unified Modeling Language (UML) é uma linguagem de modelagem livre que permite o desenvolvedor visualizar o sistema em diagramas padronizados. Cabe ressaltar que a UML não é uma metodologia de desenvolvimento, ela não diz o que fazer ou como projetar um sistema, mas auxilia na visualização do desenho do software e na comunicação entre objetos.

Segundo a OMG (2016), a UML é uma linguagem para especificação, construção, visualização e documentação de artefatos de um sistema de software.

### Diagrama de caso de uso

Segundo a OMG (2016), o diagrama de caso de uso é um modelo de interação que descreve as funcionalidades do sistema, demonstrando as interações externas. É a especificação de um conjunto de ações executado por um sistema que entrega um resultado observável para um usuário.

### Diagrama de classe

O diagrama de classes é um diagrama de categoria estrutural. É um dos diagramas mais conhecidos e importantes da UML, permitindo a compreensão da estrutura e funcionalidades do sistema. Ele apresenta as classes do sistema, bem como seus atributos e métodos (OMG, 2016).

### Diagrama de sequência

O diagrama de sequência é um diagrama que apresenta, numa ordem lógica, os fluxos de mensagens entre objetos pertinentes ao respectivo caso de uso num determinado período de tempo.

Assim como o diagrama de caso de uso, também é considerado um modelo de interação. Enquanto este apresenta a troca de mensagens entre objetos internos do sistema, aquele apresenta as interações externas, entre o usuário e o sistema (OMG, 2016).

## Padrão MVC

O padrão MVC (Model-View-Controller) é uma arquitetura que sugere a separação da aplicação em três camadas de desenvolvimento (Modelo, Visão e Controle). Essa separação facilita o desenvolvimento, a manutenção e o reaproveitamento de código.

Segundo Luckow e Melo (2010), o padrão MVC não define como separar em camadas, mas como as camadas devem interagir. Esse conceito permite a modularização do sistema, onde a troca de uma camada não tem grandes impactos em outra. Por exemplo, a troca de uma interface de usuário não altera a lógica das camadas modelo e controle.

A separação lógica da aplicação nessas três partes assegura que a camada Modelo não sabe nada sobre o que é exibido. Igualmente, a camada de Visão só está relacionada a exibir os dados e não com implementar a lógica de negócios que é controlada pela camada Modelo. A camada de Controle, como um gerenciador de tráfego, dirige as apresentações a serem exibidas e com as devidas mudanças de dados e recuperações vindas da camada Modelo. (Gonçalves, 2007)

## Servlets

Servlets são classes Java que possuem a funcionalidade de receber as solicitações através do protocolo HTTP, pelos métodos GET e POST.

Segundo Gonçalves (2007), Servlets são classes Java desenvolvidas de acordo com uma estrutura bem definida, e que, quando instaladas junto a um servidor que implemente um Servlet Container (um servidor que permita a execução de Servlets, muitas vezes chamado de Servidor de Aplicações Java), podem tratar requisições recebidas de clientes.

## JavaServer Pages

A tecnologia JavaServer Pages (JSP) é uma extensão da tecnologia de servlets que permite servir conteúdo dinâmico em uma aplicação Web. Uma JSP é uma mistura de tags HTML estáticas, tags de JSP e código Java, que pode ser incluído diretamente na página ou através de um servlet associado (Todd e Szolkowski, 2003).

Quando a JSP é acessada pela primeira vez, ela deve ser traduzida para um servlet Java. O servlet é utilizado para enviar a resposta de volta para o cliente. Portanto, o conteúdo real da JSP no servidor não é o que o cliente vê em seu navegador Web. Isto é, o cliente vê o resultado do servidor processando a JSP (Todd e Szolkowski, 2003).

A comunicação entre Servlets e JSP, é iniciada no cliente através de uma página JSP no browser do usuário, passando para um dos métodos do servlet, por exemplo Get ou Post e, somente após isso, é iniciada a regra de negócio nas demais classes Java processando a requisição do usuário (Basham, Sierra e Bates, 2008).

## JSTL

A API JavaServer Pages Standard Tag Library (JSTL) é um conjunto de marcações (tags) que encapsulam funcionalidades comuns a muitas aplicações JSP, de acordo com sua aplicabilidade. Fazendo uso extensivamente de Linguagem de Expressão (operadores e objetos implícitos), ela simplifica o processo de criação de páginas dinâmicas, eliminando boa parte do código escrito na própria página através de scriptlets (Oracle, 2010).

Segundo Delisle (2006), outro fator importante sobre a JSTL é a padronização do trabalho de programação em JSP, permitindo o reuso de código através de suas tags simples, como as de controle de laços, comandos de decisão, entre outras.

As principais categorias de bibliotecas de tags da API são: Core (manipulação de dados), SQL (manipulação de banco de dados), Format (formatação de dados) e XML (configuração de arquivos XML).

## PostgreSQL

O PostgreSQL é um sistema gerenciador de banco de dados objeto-relacional de código aberto. Tem mais de 15 anos de desenvolvimento ativo e uma arquitetura com forte reputação de confiabilidade, integridade de dados e conformidade a padrões (PostgreSQL, 2016).

Funciona nos principais sistemas operacionais e está em conformidade com o padrão ACID (Atomicidade, Consistência, Isolamento e Durabilidade), com suporte a chaves estrangeiras, joins, views, triggers e stored procedures.

## Hibernate

O processo de mapeamento de objetos em Java para as tabelas do banco de dados é chamado mapeamento objeto-relacional (ORM). Para realizar o mapeamento foi utilizado o framework Hibernate, que permite ao desenvolvedor armazenar, atualizar e recuperar dados do banco de dados relacional para objetos Java com transparência, trabalhando diretamente com objetos e não instruções SQL.

O Hibernate é um framework open-source de mapeamento objeto-relacional. É bastante poderoso e dá suporte ao mapeamento de associações entre objetos, herança, polimorfismo, composição e coleções (King et al, 2012).

## Netbeans

O Netbeans IDE (Integrated Development Environment) é um ambiente de desenvolvimento integrado gratuito e de código aberto para desenvolvedores de software nas linguagens Java, C, C++, entre outras.

É mantido pela Oracle, a mesma empresa responsável pelo Java atualmente, e está disponível para diversos sistemas operacionais. Possui ferramentas e frameworks já integrados, como o JUnit para testes, o Hibernate para mapeamento objeto-relacional e JDBC para conexão com banco de dados, permitindo o desenvolvimento de soluções robustas para Web, desktop e mobile.

## Servidor de aplicação GlassFish

Um servidor de aplicação é um software que fornece um ambiente para a execução das aplicações Web. Ele fornece, entre outras coisas: uma API padronizada para acesso a recursos Java EE; uma implementação para o modelo de arquitetura Java EE para computação distribuída incluindo acesso e comunicação entre componentes locais e remotos; uma ou mais interfaces de gerenciamento que permitam o controle integrado dos recursos disponibilizados a aplicações.

O GlassFish é um servidor de aplicação open-source para a plataforma Java EE apoiado pela Oracle. Acompanha a maioria das IDEs e é amplamente utilizado no desenvolvimento de aplicações e serviços Web (Oracle, 2013).

# Desenvolvimento

## Modelagem comum aos 3 módulos

## Modelo relacional

## Módulo Web

## Modelagem do módulo web

## Diagrama de caso de uso

## Diagrama de classes

## Diagrama de sequência

## Páginas do módulo

## Módulo desktop

## Modelagem do módulo desktop

## Diagrama de caso de uso

## Diagrama de classes

## Diagrama de sequência

## Telas do módulo

## Módulo mobile

## Modelagem do módulo mobile

## Diagrama de caso de uso

## Diagrama de classes

## Diagrama de sequência

## Telas do módulo

# Conclusão

# Referências

BASHAM, B.; SIERRA, K.; BATES, B. Use a Cabeça Servlets & JSP. Alta Book Editora - Starlin Alta Con Com Ltda, 2008.

DEITEL, Harvey M.; DEITEL, Paul. J., Java: Como programar. 6 ed. São Paulo: Editora Prentice-Hall. 2005.

DELISLE, P. JavaServer Pages Standard Tag Library: (Version 1.2). Sun Microsystem. 2006.

GONÇALVEZ, Edson. Desenvolvendo Aplicações Web com JSP, Servlets, JavaServer Faces, Hibernate, EJB 3 Persistence e Ajax. 1 ed. São Paulo: Editora Ciência Moderna, 2007.

KING, Gavin et al. Hibernate: persistência relacional para Java idiomático. Documentação de referência Hibernate. 2012. Disponível em https://docs.jboss.org/hibernate/orm/3.6/reference/pt-BR/html\_single, acesso em 22/01/2017.

LUCKOW, Décio; MELO, Alexandre. Programação Java para a Web. São Paulo: Editora Novatec, 2010.

OMG. Introduction to OMG’s Unified Modeling Language (UML). 2016. Disponível em http://www.omg.org/gettingstarted/what\_is\_uml.htm, acesso em 03/01/2017.

ORACLE. GlassFish Server Open Source Edition Reference Manual. 2013. Disponível em https://glassfish.java.net/docs/4.0/reference-manual.pdf, acesso em 18/01/2017.

ORACLE. JSP Standard Tag Library. 2010. Disponível em https://jstl.java.net, acesso em 10/01/2017.

POSTGRESQL. PostgreSQL 9.4.10 Documentation. 2016. Disponível em https://www.postgresql.org/docs/9.4/static/index.html, acesso em 10/01/2017.

TODD, Nick; SZOLKOWSKI, Mark. JavaServer Pages: O Guia do Desenvolvedor. 3 ed. Rio de Janeiro: Editora Elsevier, 2003.

# Apêndice A – Script de criação do banco de dados

DROP DATABASE controlefrota;

CREATE DATABASE;

ALTER DATABASE controlefrota OWNER TO postgres;

CREATE SCHEMA controlefrota;

ALTER SCHEMA controlefrota OWNER TO postgres;

CREATE EXTENSION IF NOT EXISTS plpgsql WITH SCHEMA pg\_catalog;

COMMENT ON EXTENSION plpgsql IS 'PL/pgSQL procedural language';

SET search\_path = controlefrota, pg\_catalog;

CREATE SEQUENCE seq\_combustivel

START WITH 0

INCREMENT BY 1

MINVALUE 0

NO MAXVALUE

CACHE 1;

ALTER TABLE seq\_combustivel OWNER TO postgres;

CREATE SEQUENCE seq\_funcao

START WITH 0

INCREMENT BY 1

MINVALUE 0

NO MAXVALUE

CACHE 1;

ALTER TABLE seq\_funcao OWNER TO postgres;

CREATE SEQUENCE seq\_revisao

START WITH 0

INCREMENT BY 1

MINVALUE 0

NO MAXVALUE

CACHE 1;

ALTER TABLE seq\_revisao OWNER TO postgres;

CREATE SEQUENCE seq\_saida

START WITH 0

INCREMENT BY 1

MINVALUE 0

NO MAXVALUE

CACHE 1;

ALTER TABLE seq\_saida OWNER TO postgres;

CREATE SEQUENCE seq\_tipo\_veiculo

START WITH 0

INCREMENT BY 1

MINVALUE 0

NO MAXVALUE

CACHE 1;

ALTER TABLE seq\_tipo\_veiculo OWNER TO postgres;

CREATE SEQUENCE seq\_usuario

START WITH 0

INCREMENT BY 1

MINVALUE 0

NO MAXVALUE

CACHE 1;

ALTER TABLE seq\_usuario OWNER TO postgres;

CREATE SEQUENCE seq\_veiculo

START WITH 0

INCREMENT BY 1

MINVALUE 0

NO MAXVALUE

CACHE 1;

ALTER TABLE seq\_veiculo OWNER TO postgres;

CREATE TABLE tb\_combustivel (

id\_combustivel integer DEFAULT nextval('seq\_combustivel'::regclass) NOT NULL,

txt\_combustivel character varying

);

ALTER TABLE tb\_combustivel OWNER TO postgres;

CREATE TABLE tb\_funcao (

id\_funcao integer DEFAULT nextval('seq\_funcao'::regclass) NOT NULL,

txt\_funcao character varying

);

ALTER TABLE tb\_funcao OWNER TO postgres;

CREATE TABLE tb\_revisao (

id\_revisao integer DEFAULT nextval('seq\_revisao'::regclass) NOT NULL,

id\_veiculo integer NOT NULL,

itens\_a\_verificar character varying,

itens\_verificados character varying,

trocas\_efetuadas character varying,

problemas\_remanescentes character varying,

id\_usuario integer,

data\_realizacao timestamp without time zone,

odometro double precision,

observacao character varying,

concluida character(1)

);

ALTER TABLE tb\_revisao OWNER TO postgres;

CREATE TABLE tb\_saida (

id\_saida integer DEFAULT nextval('seq\_saida'::regclass) NOT NULL,

data\_saida timestamp without time zone NOT NULL,

id\_veiculo integer NOT NULL,

id\_usuario integer NOT NULL,

odometro\_saida double precision NOT NULL,

motorista character varying NOT NULL,

destino character varying NOT NULL,

data\_retorno timestamp without time zone,

odometro\_retorno double precision,

autorizacao character(1)

);

ALTER TABLE tb\_saida OWNER TO postgres;

CREATE TABLE tb\_tipo\_veiculo (

id\_tipo\_veiculo integer DEFAULT nextval('seq\_tipo\_veiculo'::regclass) NOT NULL,

finalidade character varying,

txt\_tipo character varying

);

ALTER TABLE tb\_tipo\_veiculo OWNER TO postgres;

CREATE TABLE tb\_usuario (

id\_usuario integer DEFAULT nextval('seq\_usuario'::regclass) NOT NULL,

nome character varying,

sobrenome character varying,

id\_funcao integer,

email character varying,

usuario character varying NOT NULL,

senha character varying,

data\_nascimento timestamp without time zone,

data\_admisssao timestamp without time zone,

data\_demissao timestamp without time zone,

observacao character varying,

telefone character varying

);

ALTER TABLE tb\_usuario OWNER TO postgres;

CREATE TABLE tb\_veiculo (

id\_veiculo integer DEFAULT nextval('seq\_veiculo'::regclass) NOT NULL,

id\_tipo\_veiculo integer NOT NULL,

placa character varying NOT NULL,

chassi character varying,

ano\_fabricacao integer,

ano\_modelo integer,

data\_aquisicao timestamp without time zone,

data\_exclusao timestamp without time zone,

capacidade\_pessoas integer,

cilindradas double precision,

capacidade\_carga double precision,

fabricante character varying,

modelo character varying,

cor character varying,

renavam character varying,

id\_combustivel integer,

portas integer,

odometro\_atual double precision,

bloqueado "char",

observacao character varying

);

ALTER TABLE tb\_veiculo OWNER TO postgres;

SELECT pg\_catalog.setval('seq\_combustivel', 6, true);

SELECT pg\_catalog.setval('seq\_funcao', 3, true);

SELECT pg\_catalog.setval('seq\_revisao', 0, false);

SELECT pg\_catalog.setval('seq\_saida', 0, false);

SELECT pg\_catalog.setval('seq\_tipo\_veiculo', 6, true);

SELECT pg\_catalog.setval('seq\_usuario', 0, true);

SELECT pg\_catalog.setval('seq\_veiculo', 0, false);

INSERT INTO tb\_combustivel VALUES (0, 'Gasolina');

INSERT INTO tb\_combustivel VALUES (1, 'Etanol');

INSERT INTO tb\_combustivel VALUES (2, 'Flex');

INSERT INTO tb\_combustivel VALUES (3, 'Diesel');

INSERT INTO tb\_combustivel VALUES (4, 'Biodiesel');

INSERT INTO tb\_combustivel VALUES (5, 'GNV');

INSERT INTO tb\_combustivel VALUES (6, 'Outros');

INSERT INTO tb\_funcao VALUES (0, 'Gerente');

INSERT INTO tb\_funcao VALUES (1, 'Mecânico');

INSERT INTO tb\_funcao VALUES (2, 'Porteiro');

INSERT INTO tb\_funcao VALUES (3, 'Motorista');

INSERT INTO tb\_tipo\_veiculo VALUES (0, 'Passeio', 'Automóvel');

INSERT INTO tb\_tipo\_veiculo VALUES (1, 'Carga', 'Caminhão');

INSERT INTO tb\_tipo\_veiculo VALUES (2, 'Carga', 'Caminhão Trator');

INSERT INTO tb\_tipo\_veiculo VALUES (3, 'Passeio/Carga', 'Caminhonete');

INSERT INTO tb\_tipo\_veiculo VALUES (4, 'Passeio', 'Utilitário');

INSERT INTO tb\_tipo\_veiculo VALUES (5, 'Passeio', 'Ônibus');

INSERT INTO tb\_tipo\_veiculo VALUES (6, 'Passeio/Carga', 'Motocicleta');

INSERT INTO tb\_usuario VALUES (0, 'admin', 'admin', 0, 'admin@admin.com', 'admin', 'admin', '2017-01-01 00:00:00', '2017-01-01 00:00:00', NULL, 'administrador do sistema', '21 99999-9999');

ALTER TABLE ONLY tb\_combustivel

ADD CONSTRAINT tb\_combustivel\_pkey PRIMARY KEY (id\_combustivel);

ALTER TABLE ONLY tb\_funcao

ADD CONSTRAINT tb\_funcao\_pkey PRIMARY KEY (id\_funcao);

ALTER TABLE ONLY tb\_revisao

ADD CONSTRAINT tb\_revisao\_pkey PRIMARY KEY (id\_revisao);

ALTER TABLE ONLY tb\_saida

ADD CONSTRAINT tb\_saida\_pkey PRIMARY KEY (id\_saida);

ALTER TABLE ONLY tb\_tipo\_veiculo

ADD CONSTRAINT tb\_tipo\_veiculo\_pkey PRIMARY KEY (id\_tipo\_veiculo);

ALTER TABLE ONLY tb\_usuario

ADD CONSTRAINT tb\_usuario\_pkey PRIMARY KEY (id\_usuario);

ALTER TABLE ONLY tb\_usuario

ADD CONSTRAINT tb\_usuario\_usuario\_key UNIQUE (usuario);

ALTER TABLE ONLY tb\_veiculo

ADD CONSTRAINT tb\_veiculo\_chassi\_key UNIQUE (chassi);

ALTER TABLE ONLY tb\_veiculo

ADD CONSTRAINT tb\_veiculo\_pkey PRIMARY KEY (id\_veiculo);

ALTER TABLE ONLY tb\_veiculo

ADD CONSTRAINT tb\_veiculo\_placa\_key UNIQUE (placa);

ALTER TABLE ONLY tb\_revisao

ADD CONSTRAINT tb\_revisao\_id\_usuario\_fkey FOREIGN KEY (id\_usuario) REFERENCES tb\_usuario(id\_usuario);

ALTER TABLE ONLY tb\_revisao

ADD CONSTRAINT tb\_revisao\_id\_veiculo\_fkey FOREIGN KEY (id\_veiculo) REFERENCES tb\_veiculo(id\_veiculo);

ALTER TABLE ONLY tb\_saida

ADD CONSTRAINT tb\_saida\_id\_usuario\_fkey FOREIGN KEY (id\_usuario) REFERENCES tb\_usuario(id\_usuario);

ALTER TABLE ONLY tb\_saida

ADD CONSTRAINT tb\_saida\_id\_veiculo\_fkey FOREIGN KEY (id\_veiculo) REFERENCES tb\_veiculo(id\_veiculo);

ALTER TABLE ONLY tb\_veiculo

ADD CONSTRAINT tb\_veiculo\_id\_combustivel\_fkey FOREIGN KEY (id\_combustivel) REFERENCES tb\_combustivel(id\_combustivel);

ALTER TABLE ONLY tb\_veiculo

ADD CONSTRAINT tb\_veiculo\_id\_tipo\_veiculo\_fkey FOREIGN KEY (id\_tipo\_veiculo) REFERENCES tb\_tipo\_veiculo(id\_tipo\_veiculo);

ALTER TABLE ONLY tb\_usuario

ADD CONSTRAINT tb\_veiculo\_id\_funcao\_fkey FOREIGN KEY (id\_funcao) REFERENCES tb\_funcao(id\_funcao);

# Apêndice B – Classes Java – Pacote Model

# Apêndice C – Classes Java – Pacote Controller

# Apêndice D – Classes Java – Pacote View

# Apêndice E – Classes Java – Pacote DAO

# Apêndice F – Classes Java – Pacote Servlet

# Apêndice G – Classes Java – Pacote Filter

# Apêndice H – Páginas JSP

1. ssdorneles@gmail.com [↑](#footnote-ref-1)