Documentação do Projeto: DriveControl

1. Descrição Geral do Projeto

O **DriveControl** é um sistema corporativo desenvolvido em Java para o gerenciamento inteligente de frotas de veículos. O objetivo do software é permitir que uma empresa controle seus ativos (veículos), seus colaboradores (motoristas e administradores) e os eventos relacionados ao uso da frota (viagens e manutenções).

O projeto foi estruturado aplicando os quatro pilares fundamentais da Orientação a Objetos: Encapsulamento, Herança, Abstração e Polimorfismo, além de utilizar persistência de dados em banco relacional (SQLite).

Funcionalidades do Sistema

O sistema é dividido em dois perfis de acesso, cada um com funcionalidades específicas:

Perfil Administrador (Gestor de Frota):

- **Gestão de Motoristas:** Permite cadastrar, listar, editar e remover motoristas, validando dados como CNH.
- **Gestão de Veículos:** Cadastro completo da frota, permitindo editar características e alterar o status (Disponível, Em Uso, Manutenção).
- **Controle de Manutenção:** Agendamento e finalização de manutenções, com registro de custos previstos e reais, além do histórico da oficina.
- **Auditoria e Relatórios:** Visualização do histórico completo de uso dos veículos e listagem unificada de todos os usuários do sistema (demonstrando polimorfismo).

Perfil Motorista:

- **Autenticação:** Acesso seguro ao sistema via login e senha.
- **Consulta:** Visualização em tempo real dos veículos disponíveis para uso.
- **Registro de Operação:** Realização da retirada e devolução do veículo, com atualização automática da quilometragem rodada.

2. Tabela de Entidades e Atributos

Abaixo estão listadas as principais entidades do modelo de domínio e seus respectivos atributos encapsulados.

Entidade	Atributo	Tipo de Dado	Descrição
Usuario	id	int	Identificador único do usuário no banco.
	nome	String	Nome completo do usuário.
	username	String	Nome de usuário para login (único).
	senha	String	Senha para autenticação.
	ativo	boolean	Define se o usuário tem acesso ao sistema.
Administrador	*Herda de Usuario*	-	(Herda todos os atributos acima).
	cargo	String	Cargo administrativo (ex: Gerente de Frota).
Motorista	*Herda de Usuario*	-	(Herda todos os atributos de Usuario).
	cnh	String	Carteira Nacional de Habilitação (única).

	setor	String	Setor da empresa ao qual pertence (ex: Logística).
Veiculo	id	int	Identificador único do veículo.
	placa	String	Placa de identificação (única).
	modelo	String	Modelo do veículo (ex: Uno, Gol).
	marca	String	Fabricante do veículo.
	ano	int	Ano de fabricação.
	cor	String	Cor do veículo.
	status	StatusVeiculo	Enum (DISPONIVEL, EM_USO, MANUTENCAO...).
	quilometragemAtual	double	Quilometragem total do veículo.
	ultimaDataDeRevisao	Date	Data da última manutenção finalizada.

RegistroUso	id	int	Identificador da viagem.
	veiculo	Veiculo	Objeto do veículo utilizado.
	motorista	Motorista	Objeto do motorista responsável.
	dataHoraSaida	Date	Momento da retirada do veículo.
	dataHoraRetorno	Date	Momento da devolução (pode ser nulo se ativo).
	kmSaida	double	Km do veículo na saída.
	kmRetorno	double	Km do veículo no retorno.
Manutencao	id	int	Identificador do serviço.
	veiculo	Veiculo	Veículo que está sofrendo manutenção.
	descricaoServico	String	Detalhes do serviço a ser feito.

	nomeOficina	String	Local onde o serviço será realizado.
	dataEntrada	Date	Data de início da manutenção.
	dataSaidaPrevista	Date	Data estimada para término.
	custoPrevisto	double	Valor orçado do serviço.
	custoReal	double	Valor final pago pelo serviço.

3. Descrição das Entidades e Objetivos

Esta seção descreve o papel arquitetural de cada classe e interface presente no código fonte.

Pacote: Model (Entidades)

- **Usuario (Classe Abstrata):** Representa a generalização de qualquer pessoa que acessa o sistema. É abstrata para impedir a criação de um usuário sem perfil definido e para forçar a implementação do método ***exibirMenuPrincipal( )*** nas subclasses (Polimorfismo).
- **Administrador (Classe Concreta):** Especialização de ***Usuario***. Representa quem tem poder de gestão sobre o sistema. Implementa o menu administrativo.
- **Motorista (Classe Concreta):** Especialização de ***Usuario***. Representa o funcionário operacional. Possui atributos específicos como CNH e implementa o menu de uso de veículos.
- **Veiculo (Classe Concreta):** Representa o bem físico da empresa. Contém toda a lógica de estado (Status) e validação de dados do automóvel (ex: ano não pode ser inválido).
- **StatusVeiculo (Enum):** Define os estados imutáveis possíveis de um veículo, garantindo consistência no código (evita o uso de Strings soltas como "quebrado" ou "em uso").

- **RegistroUso (Classe Concreta):** Representa a associação entre um Motorista e um Veículo durante um período de tempo (uma viagem).
- **Manutencao (Classe Concreta):** Representa o evento de reparo de um veículo, controlando datas e custos financeiros.

Pacote: Repository (Persistência)

- **UsuarioRepository, VeiculoRepository, etc.:** Classes responsáveis exclusivamente pela comunicação com o Banco de Dados (SQL). Elas isolam a lógica de ***INSERT, SELECT, UPDATE*** e ***DELETE***, permitindo que o restante do sistema não precise lidar com queries SQL diretamente.

Pacote: Service (Regras de Negócio)

- **UsuarioService, VeiculoService, etc.:** Classes que contém a inteligência do sistema. Elas validam as regras antes de chamar os repositórios (ex: "Não permitir excluir um motorista que tem viagens pendentes").

Pacote: Database (Infraestrutura)

- **DatabaseConnection:** Classe que implementa o padrão de projeto **Singleton**. Seu objetivo é garantir que exista apenas uma única conexão com o banco de dados ativa durante toda a execução do programa, otimizando recursos.

Pacote: App (Interface)

- **Main:** Classe principal responsável pela inicialização do sistema e pela interação com o usuário via terminal (menus, leitura de dados e exibição de mensagens).