

CONTROLE DE ROTAS E ABASTECIMENTO

APRESENTAÇÃO DO SISTEMA DE LOGISTICA



DESENVOLVEDORES

APRESENTAÇÃO DA EQUIPE DE DESENVOLVIMENTO DO SISTEMA
DE LOGISTICA - VERSÃO 2.0 COM BANCO DE DADOS



MARCKLEN GUIMARÃES



THASSIO VAGNER



LUCAS ALVES

CONFRONTANDO REQUISITOS MINIMOS

Utilização do Padrão MVC como solicitado , também foi desenvolvido 2 interfaces EmViagem e Perfil e adicionado mais uma classe de exceção que usamos para que não fossem violados as regras de negócio

+EXCEPTION

tratamento de erros de forma personalizada

+MODEL

entidades do banco de dados

+REPOSITORY

conexão e persistência com banco de dados

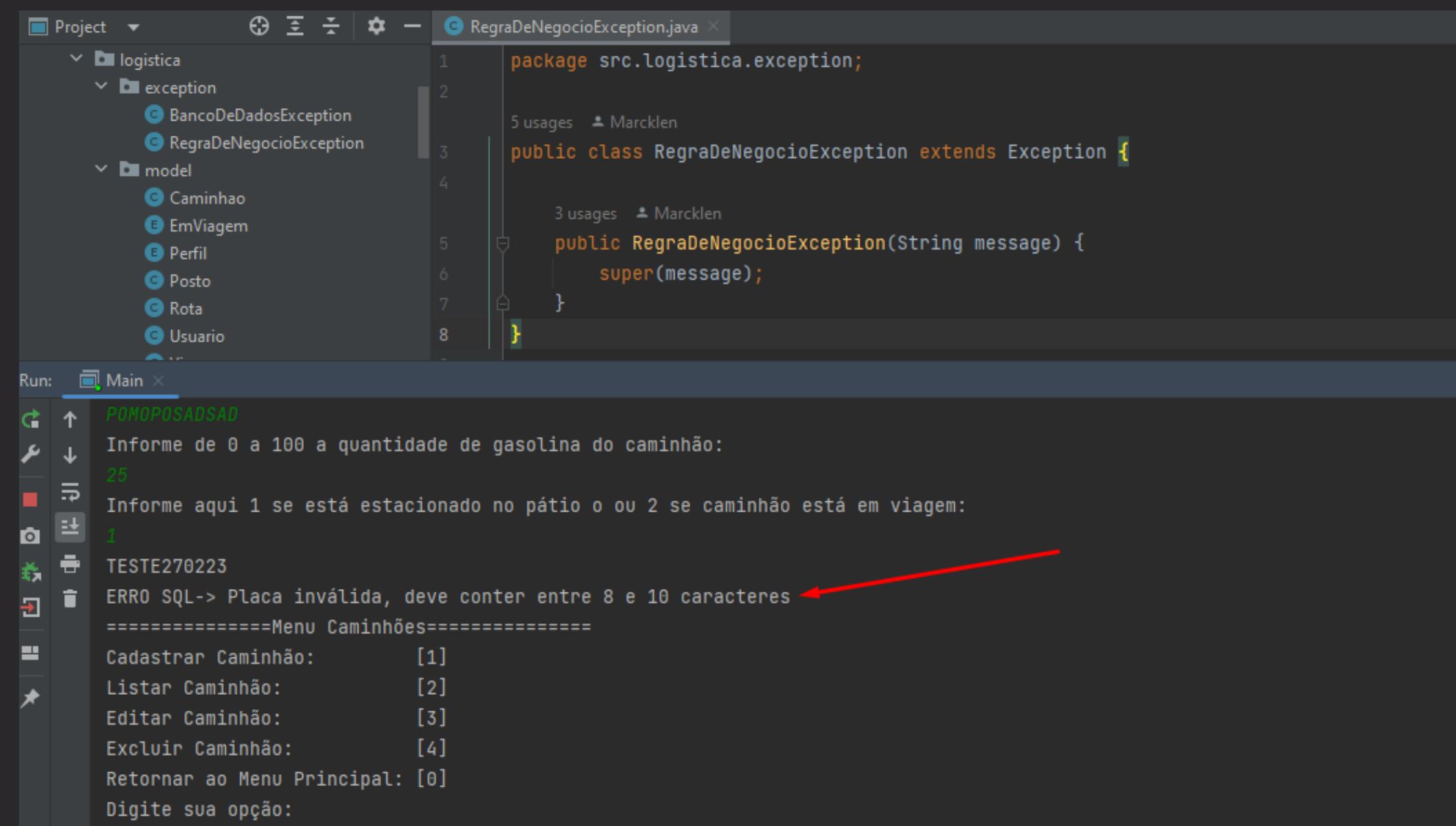
+SERVICE

regras de negócio

+VIEW

menus da aplicação

REGRA DE NEGOCIO EXCEPTION



The screenshot shows an IDE interface with a code editor and a terminal window.

Code Editor: The file `RegraDeNegocioException.java` is open, showing the following Java code:

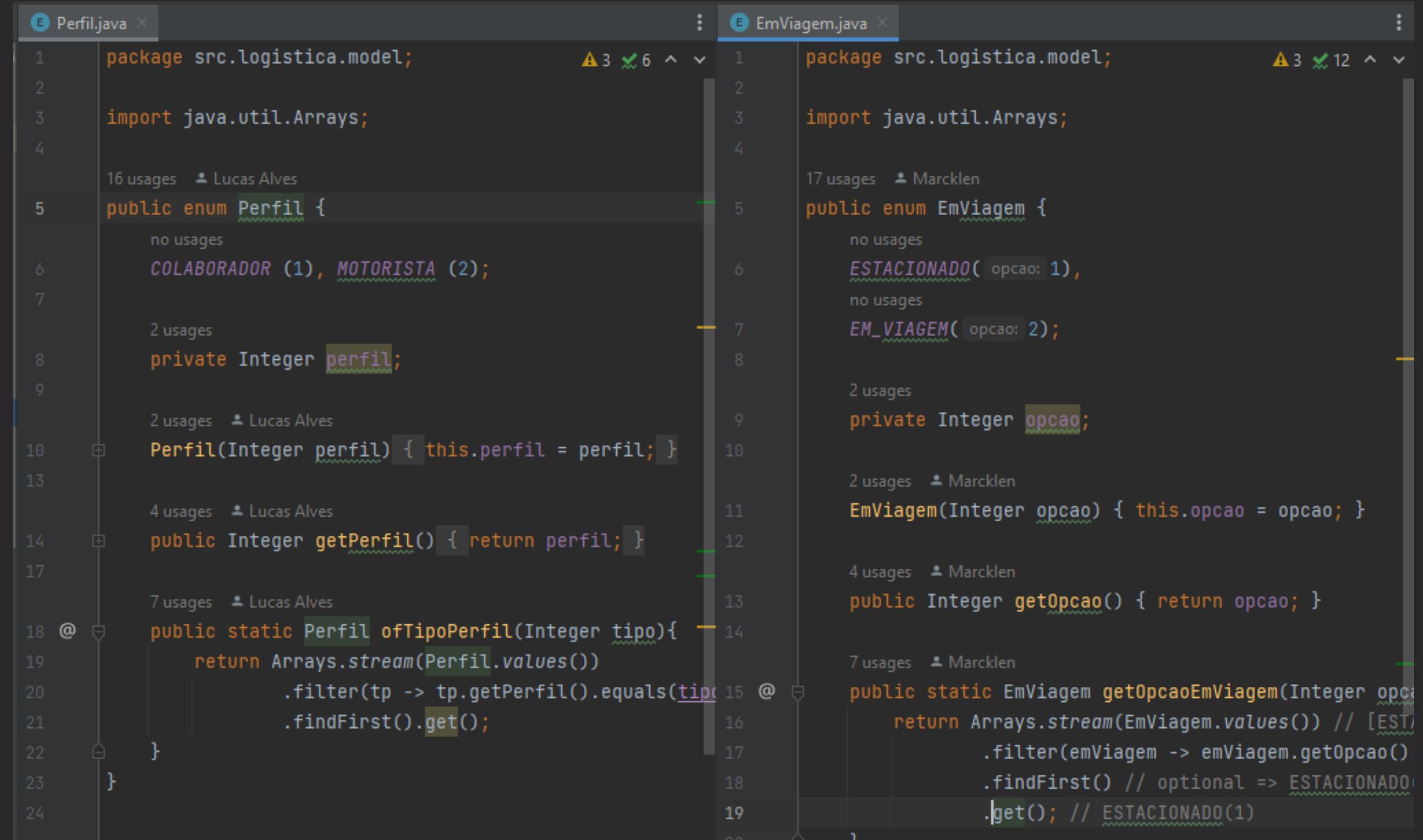
```
package src.logistica.exception;
public class RegraDeNegocioException extends Exception {
    public RegraDeNegocioException(String message) {
        super(message);
    }
}
```

Terminal Window: The terminal window shows the output of a program run named `Main`. The output is:

```
POMOPOSADSAD
Informe de 0 a 100 a quantidade de gasolina do caminhão:
25
Informe aqui 1 se está estacionado no pátio o ou 2 se caminhão está em viagem:
1
TESTE270223
ERRO SQL-> Placa inválida, deve conter entre 8 e 10 caracteres
=====Menu Caminhões=====
Cadastrar Caminhão: [1]
Listar Caminhão: [2]
Editar Caminhão: [3]
Excluir Caminhão: [4]
Retornar ao Menu Principal: [0]
Digite sua opção:
```

A red arrow points from the text "ERRO SQL-> Placa inválida, deve conter entre 8 e 10 caracteres" in the terminal output to the word "ERRO" in the code editor's run log.

ENUMS TYPES



```

Perfil.java
package src.logistica.model;
import java.util.Arrays;
16 usages ▾ Lucas Alves
public enum Perfil {
    no usages
    COLABORADOR (1), MOTORISTA (2);

    2 usages
    private Integer perfil;

    2 usages ▾ Lucas Alves
    Perfil(Integer perfil) { this.perfil = perfil; }

    4 usages ▾ Lucas Alves
    public Integer getPerfil() { return perfil; }

    7 usages ▾ Lucas Alves
    @
    public static Perfil ofTipoPerfil(Integer tipo){
        return Arrays.stream(Perfil.values())
            .filter(tp -> tp.getPerfil().equals(tipo))
            .findFirst().get();
    }
}

EmViagem.java
package src.logistica.model;
17 usages ▾ Marcklen
public enum EmViagem {
    no usages
    ESTACIONADO( opcao: 1),
    no usages
    EM_VIAGEM( opcao: 2);

    2 usages
    private Integer opcao;

    2 usages ▾ Marcklen
    EmViagem(Integer opcao) { this.opcao = opcao; }

    4 usages ▾ Marcklen
    public Integer getOpcao() { return opcao; }

    7 usages ▾ Marcklen
    public static EmViagem getOpcaoEmViagem(Integer opcao) {
        return Arrays.stream(EmViagem.values()) // [ESTACIONADO, EM_VIAGEM]
            .filter(emViagem -> emViagem.getOpcao()
            .findFirst() // optional => ESTACIONADO
            .get(); // ESTACIONADO(1)
    }
}

```

CRUD OPERATIONS

```

@Override
public Usuario adicionar(Usuario usuario) throws BancoDeDadosException {
    Connection con = null;
    try {
        con = ConexaoBancoDeDados.getConnection();
        Integer proximoId = this.getProximoId(con);
        usuario.setId(proximoId);

        String sql = "INSERT INTO LOGISTICA.USUARIO\n" +
                    "(ID_USUARIO, NOME, USUARIO, SENHA, PERFIL, CPF, CNH)\n" +
                    "VALUES(?, ?, ?, ?, ?, ?, ?)\n";
        PreparedStatement stmt = con.prepareStatement(sql);

        stmt.setInt( parameterIndex: 1, usuario.getId());
        stmt.setString( parameterIndex: 2, usuario.getNome());
        stmt.setString( parameterIndex: 3, usuario.getUsuario());
        stmt.setString( parameterIndex: 4, usuario.getSenha());
        stmt.setInt( parameterIndex: 5, usuario.getPerfil().getPerfil());
        stmt.setString( parameterIndex: 6, usuario.getCpf());
        stmt.setString( parameterIndex: 7, usuario.getCnh());
    }
}

```

```

@Override
public boolean editar(Integer id, Usuario usuario) throws BancoDeDadosException {
    Connection con = null;
    try {
        con = ConexaoBancoDeDados.getConnection();

        StringBuilder sql = new StringBuilder();
        sql.append("UPDATE LOGISTICA.USUARIO SET ");
        sql.append(" NOME = ?, ");
        sql.append(" USUARIO = ?, ");
        sql.append(" SENHA = ?, ");
        sql.append(" PERFIL = ?, ");
        sql.append(" CPF = ?, ");
        sql.append(" CNH = ? ");
        sql.append(" WHERE ID_USUARIO = ? ");

        PreparedStatement stmt = con.prepareStatement(sql.toString());

        stmt.setString( parameterIndex: 1, usuario.getNome());
        stmt.setString( parameterIndex: 2, usuario.getUsuario());
        stmt.setString( parameterIndex: 3, usuario.getSenha());
        stmt.setInt( parameterIndex: 4, usuario.getPerfil().getPerfil());
        stmt.setString( parameterIndex: 5, usuario.getCpf());
        stmt.setString( parameterIndex: 6, usuario.getCnh());
    }
}

```

```

@Override
public boolean remover(Integer id) throws BancoDeDadosException {
    Connection con = null;
    try {
        con = ConexaoBancoDeDados.getConnection();

        String sql = "DELETE FROM LOGISTICA.USUARIO WHERE ID_USUARIO = ?";

        PreparedStatement stmt = con.prepareStatement(sql);
        stmt.setInt( parameterIndex: 1, id);
        int res = stmt.executeUpdate();
        if (res == 0) {
            throw new BancoDeDadosException("Erro ao remover usuário");
        } else {
            System.out.println("Usuário removido com sucesso!" +
                "\nremoverUsuario.res=" + res);
            System.out.println("Usuário removido com sucesso!");
            return res > 0;
        }
    } catch (SQLException e) {
        throw new BancoDeDadosException("Erro ao remover usuário" + e);
    } finally {
        try {

```

• Marcklen

```

@Override
public List<Usuario> listar() throws BancoDeDadosException {
    List<Usuario> usuarios = new ArrayList<>();
    Connection con = null;
    try {
        con = ConexaoBancoDeDados.getConnection();
        Statement stmt = con.createStatement();

        String sql = "SELECT * FROM LOGISTICA.USUARIO"; // Consulta SQL no banco

        ResultSet rs = stmt.executeQuery(sql); // Executa-se a consulta

        while (rs.next()) {
            Usuario usuario = new Usuario();
            usuario.setId(rs.getInt( columnLabel: "ID_USUARIO"));
            usuario.setNome(rs.getString( columnLabel: "NOME"));
            usuario.setUsuario(rs.getString( columnLabel: "USUARIO"));
            usuario.setSenha(rs.getString( columnLabel: "SENHA"));
            usuario.setPerfil(Perfil.ofTipoPerfil(rs.getInt( columnLabel: "PERFIL")));
            usuario.setCpf(rs.getString( columnLabel: "CPF"));
            usuario.setCnh(rs.getString( columnLabel: "CNH"));
            usuarios.add(usuario);
        }
    }
}

```

**UM MUITO
OBRIGADO!**
**EM SEGUIDA A EXPLICAÇÃO
DO DIAGRAMA**



WEBSITE

www.logistica.com.brdbc

FALE CONOSCO

+85-98123.4567

ENVIE-NOS UM EMAIL

logistica@dbcccompany.com.br