



Universidade do Estado do Rio de
Janeiro

Remodelagem do JDBC -Uma ponte Java-SQL-

Grupo JavaXT:

Elen Cezar
Fabiano Prazeres
Jeane Cesário
Luciana Fontelles
Roni Rosa



Trabalho Entregue em 16/12/2007 ao
Professor *Júlio Leite*
Disponível em: <http://www.javaxt.blogspot.com/>

Introdução

Nosso desafio neste trabalho consiste em aplicar a aplicação JDBC – Uma ponte entre Java-SQL, o padrão de arquitetura Façade.

Toda a descrição dos acontecimentos durante o desenvolvimento do trabalho está descrito no site <http://www.javaxt.blogspot.com/>.

A seguir faremos uma breve descrição da aplicação e depois mostraremos a arquitetura e o código final.

Sobre a aplicação

Originalmente o sistema tem o propósito de, através da API JDBC oferecida pela plataforma Java, acessar e gerenciar qualquer banco de dados SQL, em um programa não-gráfico ou um applet através de um driver JDBC que funciona como um tradutor entre a tecnologia JDBC e os diferentes tipos de SGBD's. A aplicação desenvolvida interage com o banco de dados de uma concessionária de veículos, hospedado em um servidor de banco de dados, permitindo que um usuário consulte e atualize remotamente as tabelas do banco de dados referentes aos estoques de automóveis e caminhões.

O sistema JDBC – Uma ponte entre Java-SQL é uma aplicação Java que interage com o SGBD PostgreSQL utilizando a API JDBC, em um ambiente Unix-like.

Para a aplicação do design pattern Façade, algumas alterações foram feitas, como a construção de uma interface gráfica e a separação do sistema baseado no Padrão de Modelo MVC (Model View Controller), separando a aplicação em três camadas.

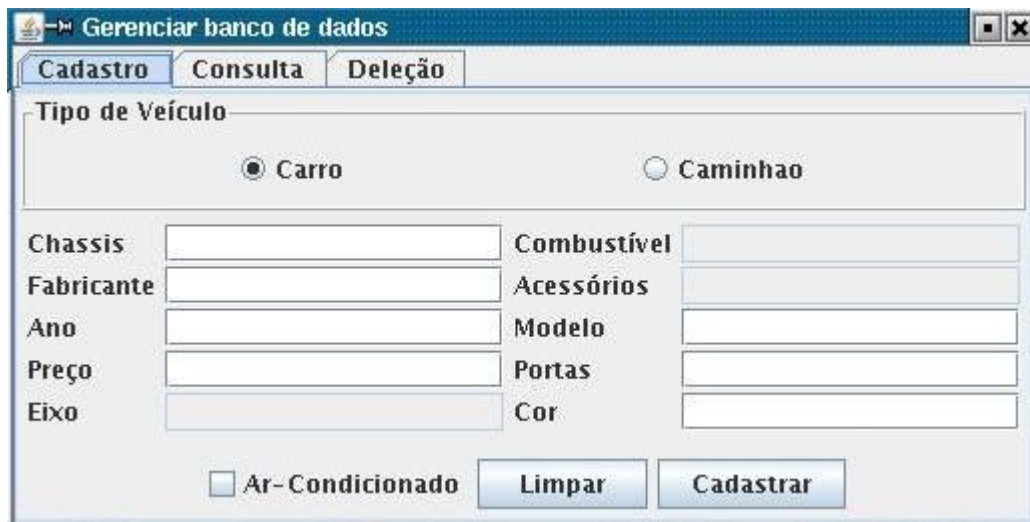
A seguir descrevemos brevemente cada uma das telas:

1. A figura 1 é a tela de Login que permite o usuário efetuar o login no sistema para realizar as ações desejadas.



Figura 1: Tela de Login

2. A figura 2 consiste no cadastro de um veículo, onde algumas propriedades do cadastro são requeridas.



Gerenciar banco de dados

Cadastro Consulta Deleção

Tipo de Veículo

Carro Caminhao

Chassis Combustível

Fabricante Acessórios

Ano Modelo

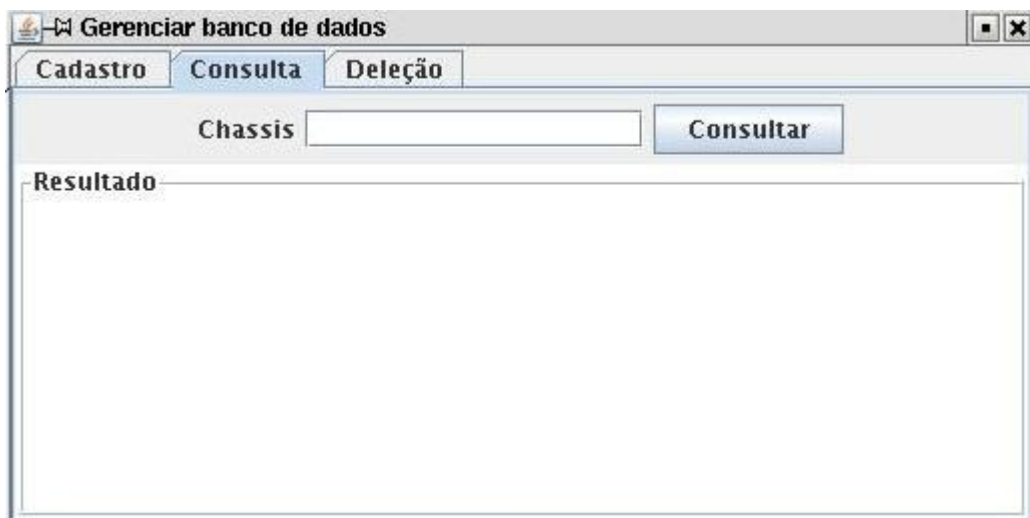
Preço Portas

Eixo Cor

Ar-Condicionado

Figura 2: Tela de Login

3. A consulta de um veículo é feita por chassi e é mostrada em Resultados.



Gerenciar banco de dados

Cadastro Consulta Deleção

Chassis

Resultado

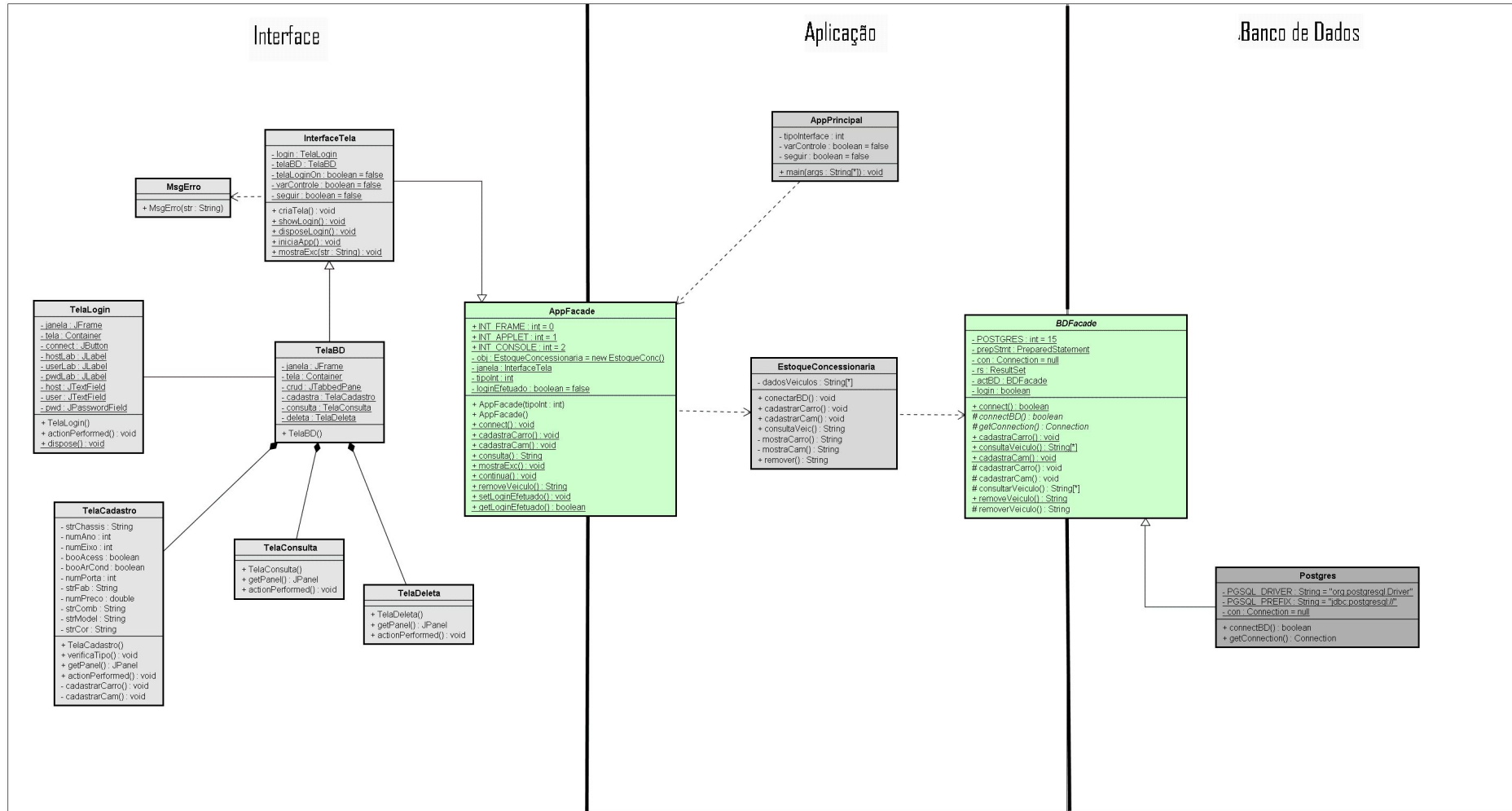
Figura 3: Tela de Consulta

4. O usuário pode também, através da tela de Deleção, excluir um veículo cadastrado na base de dados.



Tela 4: Tela de Deleção

Arquitetura



Código



AppPrincipal.java

```
/* @author JavaXT
 * Essa classe cria um AppFacade para interface gráfica (frame). Pode
 * criar outras
 * interfaces de acordo com o parâmetro passado pelo usuário.
 */
package projeto.aplicacao;
import projeto.facade.AppFacade;
import projeto.interfaces.gui.*;
import java.sql.SQLException;

public class AppPrincipal
{
    private int tipoInterface;
    private boolean varControle = false;
    private boolean seguir = false;

    public static void main (String[] args)
    {
        AppFacade start = new AppFacade(AppFacade.INT_FRAME);
    }
}
```

EstoqueConcessionaria.java

```
/*Classe principal da camada de aplicação. Permite ao usuário
gerenciar o banco de dados que representa o
 *estoque de uma concessionária de carros e caminhões
 */
package projeto.aplicacao;

import projeto.facade.AppFacade;
import projeto.facade.BDFacade;
import java.sql.SQLException;

public class EstoqueConcessionaria
{
    private String[] dadosVeiculos;

    /*Esse método invoca o BDFacade para efetuar o login de um
usuário*/
    public void conectarBD(String url, String user, String pwd, int
sgbd) throws ClassNotFoundException, SQLException
```

```

    {
        boolean login = BDFacade.connect(url, user, pwd, sgbd);

        /*Esse if verifica se o login foi efetuado e chama o método
do AppFacade responsável
        *por tratar adequadamente cada situação
        */
        if(login == true)
        {
            AppFacade.continua();
        }
        else
        {
            AppFacade.setLoginEfetuado(true);
        }
    }

    /*Invoca o BDFacade para cadastrar um carro*/
    public void cadastrarCarro(String fab, int ano, String chassis,
double preco, String model, int numPortas, String cor, boolean ar)
throws SQLException
    {
        BDFacade.cadastraCarro(fab, ano, chassis, preco, model,
numPortas, cor, ar);
    }

    /*Invoca o BDFacade para cadastrar um caminhao*/
    public void cadastrarCam(String fab, int ano, String chassis,
double preco, int numEixos, String combustivel, boolean acessorios)
throws SQLException
    {
        BDFacade.cadastraCam(fab, ano, chassis, preco,
numEixos, combustivel, acessorios);
    }

    /* Esse método recebe do BDFacade um array de String, e de
acordo com o tamanho
    * desse array, devolve para a interface os dados de um carro,
de um caminhão ou
    * a mensagem de veículo não encontrado
    */
    public String consultaVeic(String chassis) throws SQLException
    {
        String retorno="";

        dadosVeiculos = BDFacade.consultaVeiculo(chassis);

        switch(dadosVeiculos.length)
        {
            case 0: retorno = "Veiculo não encontrado\n";
                    break;

            case 8: retorno = "\nCarro Encontrado\n" +
mostraCarro();
                    break;

            case 7: retorno = "\nCaminhao Encontrado\n"
+mostraCam();
        }
    }

```

```

        return retorno;
    }

    /*Gera a string com os dados do carro*/
    private String mostraCarro()
    {
        String retorno = "\nChassis: " +dadosVeiculos[2]+
"\nFabricante: " +dadosVeiculos[0]+ "\nModelo: " +dadosVeiculos[4]+
"\nAno: " +dadosVeiculos[1];
        retorno += "Preço: " +dadosVeiculos[3]+ "Portas: "
+dadosVeiculos[5]+ "\nCor: " +dadosVeiculos[6]+ "\nAr Condicionado: "
+dadosVeiculos[7];

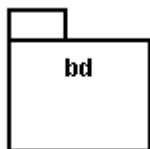
        return retorno;
    }

    /*Gera a string com os dados do caminhão*/
    private String mostraCam()
    {
        String retorno = "\nChassis: " +dadosVeiculos[2]+
"\nFabricante: " +dadosVeiculos[0]+ "\nModelo: " +dadosVeiculos[4]+
"\nAno: " +dadosVeiculos[1];
        retorno += "\nPreço: " +dadosVeiculos[3]+ "\nEixos: "
+dadosVeiculos[4]+ "\nCombustível: " +dadosVeiculos[5]+ "\nAcessórios:
" +dadosVeiculos[6];

        return retorno;
    }

    /*Invoca o BDFacade para remover um veículo*/
    public String remover(String chassis) throws SQLException
    {
        return BDFacade.removeVeiculo(chassis);
    }
}

```



InterfaceTela.java

```

/*@author JavaXT
 *Classe que implementa o método abstrato connect de BDFacade, para
que a conexão seja efetuada em um
 *banco de dados PostgreSQL. No caso da adição de outros sgbd's ao
sistema, devem ser criadas classes
 *semelhantes à essa, mas utilizando o driver JDBC adequado ao sgbd
que se pretende utilizar
 */

package projeto.bd;

import projeto.facade.BDFacade;

import java.sql.Connection;
import java.sql.DriverManager;

```

```

import java.sql.SQLException;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class Postgres extends BDFacade
{
    //O driver e o prefixo são específicos de cada SGBD
    private final static String PGSQL_DRIVER =
"org.postgresql.Driver";
    private final static String PGSQL_PREFIX = "jdbc:postgresql://";

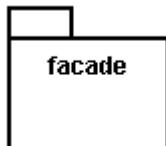
    private static Connection con = null;

    public boolean connectBD(String url, String user, String pwd)
throws SQLException, ClassNotFoundException
    {
        //Carrega o driver para iniciar a conexão
        Class.forName(PGSQL_DRIVER);
        con = DriverManager.getConnection(PGSQL_PREFIX+url, user,
pwd);

        if(con.isClosed() == true) //Se a conexão está fechada
            return false;
        else
            return true;
    }

    public Connection getConnection()
    {
        return con;
    }
}

```



AppFacade.java

```

/* @author JavaXT
 * Facade que conecta as interfaces às classes de aplicação do
programa.
 */
package projeto.facade;

import projeto.aplicacao.EstoqueConcessionaria;
import projeto.interfaces.gui.TelaLogin;
import projeto.interfaces.gui.TelaBD;
import projeto.interfaces.gui.TelaCadastro;
import projeto.interfaces.gui.MsgErro;
import projeto.interfaces.gui.InterfaceTela;
import java.sql.SQLException;
import javax.swing.JOptionPane;

public class AppFacade
{

```

```

//Constantes para definir o tipo de interface
public static final int INT_FRAME = 0; //Janela
public static final int INT_APPLET = 1; //Applet
public static final int INT_CONSOLE = 2; //Aplicação em linha de
comando

//Variáveis
private static EstoqueConcessionaria obj = new
EstoqueConcessionaria();
private static InterfaceTela janela;
private static int tipoInt;
private static boolean loginEfetuado = false;

/*De acordo com o parâmetro recebido, cria um objeto que exibirá
*a interface adequada
*/
public AppFacade(int tipoInt)
{
    this.tipoInt = tipoInt;
    //Caso sejam criados outros tipos de interface, basta
acrescentá-los aos cases
    switch(tipoInt)
    {
        case INT_FRAME: janela = new InterfaceTela();
                        janela.criaTela();
                        break;
    }
}

/*Esse construtor é para viabilizar a herança*/
public AppFacade()
{
}

/*Esse método chama o método da EstoqueConcessionaria que vai
ser responsável pelo login*/
public static void connect(String url, String user, String pwd,
int sgbd) throws ClassNotFoundException, SQLException
{
    obj.conectarBD(url, user, pwd, sgbd);
}

/*Método que permite cadastrar um carro. Ele invoca o método da
classe de aplicação com os parâmetros recebidos*/
public static void cadastraCarro(String fab, int ano, String
chassis, double preco, String model, int numPortas, String cor,
boolean ar)
{
    try
    {
        obj.cadastrarCarro(fab, ano, chassis, preco, model,
numPortas, cor, ar);

        //Exibe a mensagem de acordo com o tipo de interface
ativo
        switch(tipoInt)
        {
            case INT_FRAME:
JOptionPane.showMessageDialog(null, "Carro Cadastrado", "Mensagem",
JOptionPane.INFORMATION_MESSAGE);
                                break;

```

```

        }
    }
    catch(SQLException e)
    {
        mostraExc(e.toString());
    }
}

/*Método que permite cadastrar um caminhão. Ele invoca o método
da classe de aplicação com os parâmetros recebidos*/
public static void cadastraCam(String fab, int ano, String
chassis, double preco, int numEixos, String combustivel, boolean
acessorios) throws SQLException
{
    try
    {
        obj.cadastrarCam(fab, ano, chassis, preco, numEixos,
combustivel, false);

        switch(tipoInt)
        {
            case INT_FRAME:
JOptionPane.showMessageDialog(null, "Caminhão Cadastrado", "Mensagem",
JOptionPane.INFORMATION_MESSAGE);
                                break;
        }
    }
    catch(SQLException e)
    {
        mostraExc(e.toString());
    }
}

/*Método que permite consultar um veículo. Ele invoca o método
da classe de aplicação com o parâmetro recebido*/
public static String consulta(String chassis)
{
    String retorno=null;
    try
    {
        retorno = obj.consultaVeic(chassis);
    }
    catch(SQLException e)
    {
        mostraExc(e.toString());
    }

    return retorno;
}

/*Esse método exibe uma mensagem de erro*/
public static void mostraExc(String erro)
{
    //Mostra a exceção de acordo com o tipo de interface
    if(tipoInt == INT_FRAME)
    {
        janela.mostraExc(erro);
    }
}

```

```

        /*Esse método exibe a tela de gerenciamento do BD, quando a
        classe EstoqueConcessionaria determina
        *que o login foi efetuado*/
        public static void continua()
        {
            if(tipoInt == INT_FRAME)
            {
                setLoginEfetuado(true);
                janela.disposeLogin();
                janela.iniciaApp(); //Invoca a segunda interface
            }
        }

        /*Método de remoção de veículos*/
        public static String removeVeiculo(String chassis) throws
        SQLException
        {
            return obj.remover(chassis);
        }

        public static void setLoginEfetuado(boolean status)
        {
            loginEfetuado = status;
        }

        public static boolean getLoginEfetuado()
        {
            return loginEfetuado;
        }
    }

```

BDFacade.java

```

/*@author JavaXT
 *Façade que conecta a classe de aplicação ao banco de dados. O JDBC
permite que os métodos declarados aqui
 *sejam utilizados por qualquer SGBD, exceto o método connect que por
isso mesmo é abstrato.
 */

package projeto.facade;

import projeto.bd.Postgres;
import java.sql.SQLException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.PreparedStatement;

public abstract class BDFacade
{
    /*Constantes da classe - Tipo de SGBD. Para permitir a adição de
    outros bancos de dados, basta
    *declarar uma constante numérica para ele, e adicioná-lo aos
    switches*/
    private final static int POSTGRES = 15;

    private static PreparedStatement prepStmt;
    private static Connection con = null;
    private static ResultSet rs;

```

```

private static BDFacade actBD;
private static boolean login;

public static boolean connect(String url, String user, String
pwd, int sgbd) throws ClassNotFoundException, SQLException
{
    //No caso aqui declarado, o sgbd utilizado é o PostgreSQL
    switch(sgbd)
    {
        case POSTGRES: actBD = new Postgres();
        login =
actBD.connectBD(url, user, pwd);
        con =
actBD.getConnection();
        break;
    }
    return login;
}

protected abstract boolean connectBD(String url, String user,
String pwd) throws SQLException, ClassNotFoundException;

protected abstract Connection getConnection();

public static void cadastraCarro(String fab, int ano, String
chassis, double preco, String model, int numPortas, String cor,
boolean ar) throws SQLException
{
    actBD.cadastrarCarro(fab, ano, chassis, preco, model,
numPortas, cor, ar);
}

public static String[] consultaVeiculo(String chassis) throws
SQLException
{
    return actBD.consultarVeiculo(chassis);
}

public static void cadastraCam(String fab, int ano, String
chassis, double preco, int numEixos, String combustivel, boolean
acessorios) throws SQLException
{
    actBD.cadastrarCam(fab, ano, chassis, preco,
numEixos, combustivel, acessorios);
}

/*Esse método é o mesmo para qualquer tipo de SGBD utilizado.
Ele cria uma chamada SQL com os parâmetros inseridos pelo usuário e
faz a inserção no banco de dados
*/
protected void cadastrarCarro(String fab, int ano, String
chassis, double preco, String model, int numPortas, String cor,
boolean ar) throws SQLException
{
    prepStmt = con.prepareStatement("INSERT INTO Automovel
VALUES (?, ?, ?, ?, ?, ?, ?, ?)");
    prepStmt.setString(1, fab); //(posicao, parametro)
    prepStmt.setObject(2, Integer.valueOf(ano));
    prepStmt.setString(3, chassis);
}

```

```

        prepStmt.setDouble(4, preco);
        prepStmt.setString(5, model);
        prepStmt.setInt(6, numPortas);
        prepStmt.setString(7, cor);
        prepStmt.setBoolean(8, ar);

        prepStmt.executeUpdate();
    }

    /*Age da mesma maneira que o cadastrarCarro, mas para
    caminhões*/
    protected void cadastrarCam(String fab, int ano, String chassis,
    double preco, int numEixos, String combustivel, boolean acessorios)
    throws SQLException
    {
        prepStmt = con.prepareStatement("INSERT INTO Caminhao
VALUES (?, ?, ?, ?, ?, ?, ?, ?)");

        prepStmt.setString(1, fab); //(posicao, parametro)
        prepStmt.setInt(2, ano);
        prepStmt.setString(3, chassis);
        prepStmt.setDouble(4, preco);
        prepStmt.setInt(5, numEixos);
        prepStmt.setString(6, combustivel );
        prepStmt.setBoolean(7, acessorios);

        prepStmt.executeUpdate();
    }

    /**Método que busca um veículo nas tabelas de carros e
    caminhões. A chave primária
    * é o número do chassis.
    */
    protected String[] consultarVeiculo(String chassis) throws
    SQLException
    {
        String[] ret=null;

        prepStmt = con.prepareStatement("SELECT Fabricante,
AnoFabric, NoChassis, Preco, Modelo, NoPortas, Cor, ArCondicionado
FROM Automovel WHERE NoChassis = '"+chassis+"'");
        boolean carro = false;

        rs = prepStmt.executeQuery();

        while (rs.next())
        {
            String[] retorno = {rs.getString("Fabricante"),
""+rs.getInt("AnoFabric"), rs.getString("NoChassis"),
""+rs.getFloat("Preco"),
                                rs.getString("Modelo"),
""+rs.getInt("NoPortas"), rs.getString("Cor"),
""+rs.getBoolean("ArCondicionado")};
            ret = retorno;
            carro = true;
        }

        if(carro == false)
        {

```

```

        prepStmt = con.prepareStatement("SELECT Fabricante,
AnoFabric, NoChassis, Preco, NumEixos, Combustivel, Acessorios FROM
Caminhao WHERE NoChassis = '"+chassis+"'");
        rs = prepStmt.executeQuery();
        while (rs.next())
        {
            String[] retorno = {rs.getString("Fabricante"),
""+rs.getInt("AnoFabric"), rs.getString("NoChassis"),
""+rs.getFloat("Preco"),

""+rs.getInt("NumEixos"), ""+rs.getString("Combustivel"),
""+rs.getBoolean("Acessorios")};
            ret = retorno;
            carro = true;
        }
    }

    /* Se o veículo foi encontrado, carro tem valor true, caso
contrário,
    * carro com valor false indica que não há caminhão nem
veículo com esse chassis
    */
    if(carro)
    {
        return ret;
    }
    else
    {
        ret = new String[0];
        return ret;
    }
}

public static String removeVeiculo(String chassis) throws
SQLException
{
    return actBD.removerVeiculo(chassis);
}

/*Esse método remove um veículo - carro ou caminhão - ou exibe a
mensagem de veículo não encontrado*/
protected String removerVeiculo(String chassis) throws
SQLException
{
    String[] existe;
    String retorno = null;

    existe = consultarVeiculo(chassis);
    if(existe.length != 0)
    {
        prepStmt = con.prepareStatement("DELETE FROM
Automovel WHERE NoChassis = '"+chassis+"'");
        prepStmt.executeUpdate();

        prepStmt = con.prepareStatement("DELETE FROM Caminhao
WHERE NoChassis = '"+chassis+"'");
        prepStmt.executeUpdate();

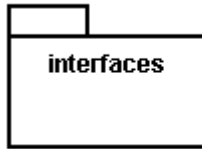
        retorno = "Veículo removido com sucesso";
    }
    else

```

```

        retorno = "Veiculo não encontrado";
    }
    return retorno;
}

```



InterfaceTela. java

```

/*@author JavaXT
 *Essa classe estende a AppFacade e é herdada pelas telas. Em vez de invocar
 diretamente a AppFacade,
 *as outras classes do pacote utilizam essa classe InterfaceTela. Todas as outras classes
 do pacote herdam *de InterfaceTela.
 */
package projeto.interfaces.gui;

import projeto.facade.*;

public class InterfaceTela extends AppFacade
{
    private static TelaLogin login; //Tela de Login
    private static TelaBD telaBD; //Tela de gerenciamento do BD
    private static boolean telaLoginOn = false; //Indica se já há uma tela de login
ativa ou não
    private static boolean varControle = false;
    private static boolean seguir = false;

    public void criaTela()
    {
        showLogin(); //Invoca a tela de login
    }

    /*Esse método exibe a tela de login, se ela ainda não está ativa e retorna
    *para a classe Principal o status do usuário: logado (true) ou não-logado(false)
    */
    public static void showLogin()
    {
        //Se não há uma tela de login ativa, ela é exibida
        if (telaLoginOn == false)
        {
            login = new TelaLogin();
            telaLoginOn = true;
        }
    }
}

```

```

    }
}

/*Desfaz a tela de login, liberando a memória ocupada por ela*/
public static void disposeLogin()
{
    if(telaLoginOn == true)
    {
        login.dispose();
        telaLoginOn = false;
    }
}

public static void iniciaApp()
{
    new TelaBD();
}

/*Quando ocorre alguma exceção/erro, as interfaces invocam esse método
*para exibir uma caixinha de mensagem*/
public static void mostraExc(String str)
{
    new MsgErro(str);
}
}

```

MsgErro.java

```

/*@author JavaXT
*Classe que cria uma caixa de diálogo que exibe uma mensagem de erro
*/
package projeto.interfaces.gui;

import javax.swing.JOptionPane;

public class MsgErro
{
    public MsgErro(String str)
    {
        JOptionPane.showMessageDialog(null, str, "Atenção",
        JOptionPane.WARNING_MESSAGE);
    }
}

```

TelaBD.java

```

/*@author JavaXT
*Tela do gerenciamento do banco de dados. Junta as telas de cadastro,
consulta e remoção através de abas
*/
package projeto.interfaces.gui;

```

```

import javax.swing.JFrame;
import java.awt.Container;
import javax.swing.JPanel;
import javax.swing.ButtonGroup;
import javax.swing.JTabbedPane;
import javax.swing.SwingUtilities;

public class TelaBD extends InterfaceTela
{
    private static JFrame janela;
    private static Container tela;
    private static JTabbedPane crud;

    //Cada um dos JPanel contidos pelas abas é um objeto
    private static TelaCadastro cadastra;
    private static TelaConsulta consulta;
    private static TelaDeleta delete;

    public TelaBD()
    {
        janela = new JFrame("Gerenciar banco de dados");
        janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        tela = janela.getContentPane();

        crud = new JTabbedPane();
        cadastra = new TelaCadastro();
        consulta = new TelaConsulta();
        delete = new TelaDeleta();
        crud.add("Cadastro", cadastra.getPanel());
        crud.add("Consulta", consulta.getPanel());
        crud.add("Deleção", delete.getPanel());
        tela.add(crud);

        janela.pack();
        janela.setResizable(false);
        janela.setVisible(true);
    }
}

```

TelaCadastro.java

```

/**@author JavaXT
 *Tela de Cadastro de veículos
 */
package projeto.interfaces.gui;

import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.ButtonGroup;
import javax.swing.JRadioButton;
import javax.swing.JCheckBox;
import javax.swing.JPanel;
import javax.swing.border.TitledBorder;
import javax.swing.border.EtchedBorder;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.BorderLayout;

```

```

import java.awt.GridLayout;

public class TelaCadastro extends InterfaceTela implements
ActionListener
{
    private JPanel telaC;
    private JPanel radioB;
    private JPanel label1;
    private JPanel label2;
    private JPanel texto1;
    private JPanel texto2;
    private JPanel miolo;
    private JPanel base;
    private JLabel chassisLab;
    private JLabel anoLab;
    private JLabel eixoLab;
    private JLabel acessLab;
    private JLabel portaLab;
    private JLabel fabLab;
    private JLabel precoLab;
    private JLabel combLab;
    private JLabel modelLab;
    private JLabel corLab;
    private JTextField chassis;
    private JTextField ano;
    private JTextField eixo;
    private JTextField acess;
    private JTextField porta;
    private JTextField fab;
    private JTextField preco;
    private JTextField comb;
    private JTextField model;
    private JTextField cor;
    private ButtonGroup grupo;
    private JRadioButton carro;
    private JRadioButton caminhao;
    private JCheckBox arCond;
    private JButton cadastro;
    private JButton limpar;

    public TelaCadastro()
    {
        //Painel maior
        telaC = new JPanel(new BorderLayout());

        //Painel dos radioButtons
        radioB = new JPanel();
        grupo = new ButtonGroup();
        carro = new JRadioButton("Carro", true);
        caminhao = new JRadioButton("Caminhao");
        JLabel vaziao1 = new JLabel("

");

        carro.addActionListener(this);
        caminhao.addActionListener(this);
        grupo.add(carro);
        grupo.add(caminhao);
        radioB.setBorder(new TitledBorder(new
EtchedBorder(EtchedBorder.RAISED), "Tipo de Veículo"));
        radioB.add(carro);
        radioB.add(vaziao1);
        radioB.add(caminhao);

```

```

telaC.add(radioB, BorderLayout.NORTH);

//Painel dos labes e tFields
label1 = new JPanel(new GridLayout(5, 1, 0, 6));
label2 = new JPanel(new GridLayout(5, 1, 0, 6));
texto1 = new JPanel(new GridLayout(5, 1, 0, 2));
texto2 = new JPanel(new GridLayout(5, 1, 0, 2));
miolo = new JPanel();
chassisLab = new JLabel("Chassis");
chassis = new JTextField(15);
fabLab = new JLabel("Fabricante");
fab = new JTextField(15);
anoLab = new JLabel("Ano");
ano = new JTextField(15);
precoLab = new JLabel("Preço");
preco = new JTextField(15);
eixoLab = new JLabel("Eixo");
eixo = new JTextField(15);
combLab = new JLabel("Combustível");
comb = new JTextField(15);
acessLab = new JLabel("Acessórios");
acess = new JTextField(15);
modelLab = new JLabel("Modelo");
model = new JTextField(15);
portaLab = new JLabel("Portas");
porta = new JTextField(5);
corLab = new JLabel("Cor");
cor = new JTextField(10);
label1.add(chassisLab);
texto1.add(chassis);
label1.add(fabLab);
texto1.add(fab);
label1.add(anoLab);
texto1.add(ano);
label1.add(precoLab);
texto1.add(preco);
label1.add(eixoLab);
texto1.add(eixo);
label2.add(combLab);
texto2.add(comb);
label2.add(acessLab);
texto2.add(acess);
label2.add(modelLab);
texto2.add(model);
label2.add(portaLab);
texto2.add(porta);
label2.add(corLab);
texto2.add(cor);
miolo.add(label1);
miolo.add(texto1);
miolo.add(label2);
miolo.add(texto2);
telaC.add(miolo, BorderLayout.CENTER);

//Painel da base
base = new JPanel();
arCond = new JCheckBox("Ar-Condicionado");
cadastro = new JButton("Cadastrar");
limpar = new JButton(" Limpar ");
arCond.addActionListener(this);

```

```

        cadastro.addActionListener(this);
        limpar.addActionListener(this);
        base.add(arCond);
        base.add(limpar);
        base.add(cadastro);
        telaC.add(base, BorderLayout.SOUTH);

        verificaTipo();
    }

    public void verificaTipo()
    {
        if (carro.isSelected())
        { //Se carro está selecionado, desabilita parâmetros do
caminhao ou vice-versa
            comb.setText("");
            acess.setText("");
            eixo.setText("");
            comb.setEditable(false);
            acess.setEditable(false);
            eixo.setEditable(false);
            model.setEditable(true);
            cor.setEditable(true);
            porta.setEditable(true);
            arCond.setEnabled(true);
        }
        else
        {
            model.setText("");
            cor.setText("");
            porta.setText("");
            comb.setEditable(true);
            acess.setEditable(true);
            eixo.setEditable(true);
            model.setEditable(false);
            cor.setEditable(false);
            porta.setEditable(false);
            arCond.setSelected(false);
            arCond.setEnabled(false);
        }
    }

    public JPanel getPanel()
    {
        return telaC;
    }

    public void actionPerformed(ActionEvent evt)
    {
        if(evt.getSource() == cadastro)
        {
            if(carro.isSelected())
                cadastrarCarro();
            else
                cadastrarCam();
        }

        if (evt.getSource() == carro || evt.getSource() ==
caminhao)
        {
            verificaTipo();
        }
    }

```

```

    }
    else
    {
        if (evt.getSource() == limpar)
        {
            chassis.setText("");
            ano.setText("");
            fab.setText("");
            preco.setText("");
            comb.setText("");
            acess.setText("");
            eixo.setText("");
            model.setText("");
            cor.setText("");
            porta.setText("");
            arCond.setSelected(false);
            carro.setSelected(true);
            verificaTipo();
        }
    }
}

private String strChassis;
private int numAno;
private int numEixo;
private boolean booAcess;
private boolean booArCond;
private int numPorta;
private String strFab;
private double numPreco;
private String strComb;
private String strModel;
private String strCor;

private void cadastrarCarro()
{
    try
    {
        strChassis = chassis.getText().trim();
        numAno = Integer.parseInt(ano.getText().trim());
        numPorta = Integer.parseInt(porta.getText().trim());
        numPreco = Integer.parseInt(preco.getText().trim());
        strFab = fab.getText().trim();
        strComb = comb.getText().trim();
        strModel = model.getText().trim();
        strCor = cor.getText().trim();

        if(arCond.isSelected())
            booArCond = true;
        else
            booArCond = false;

        InterfaceTela.cadastraCarro(strFab, numAno,
strChassis, numPreco, strModel, numPorta, strCor, booArCond);
    }
    catch(Exception e)
    {
        InterfaceTela.mostraExc(e.toString());
    }
}
}

```

```

private void cadastrarCam()
{
    try
    {
        strChassis = chassis.getText().trim();
        numAno = Integer.parseInt(ano.getText().trim());
        numEixo = Integer.parseInt(eixo.getText().trim());
        numPreco = Integer.parseInt(preco.getText().trim());
        strFab = fab.getText().trim();
        strComb = comb.getText().trim();
        booAcess = Boolean.parseBoolean(acess.getText());

        InterfaceTela.cadastraCam(strFab, numAno, strChassis,
numPreco, numEixo, strComb, booAcess);
    }
    catch(Exception e)
    {
        InterfaceTela.mostraExc(e.toString());
    }
}
}

```

TelaConsulta.java

```

/*@author JavaXT
 *Tela de Consulta
 */
package projeto.interfaces.gui;

import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JTextArea;
import javax.swing.JCheckBox;
import javax.swing.JComboBox;
import javax.swing.JPanel;
import javax.swing.border.TitledBorder;
import javax.swing.border.EtchedBorder;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.BoxLayout;
import java.awt.BorderLayout;

public class TelaConsulta implements ActionListener
{
    private JPanel telaR;
    private JPanel topo;
    private JPanel meio;
    private JLabel chassisLab;
    private JTextField chassis;
    private JTextArea result;
    private JButton consult;

    public TelaConsulta()
    {
        //Painéis
        telaR = new JPanel(new BorderLayout());
        topo = new JPanel();
    }
}

```

```

        meio = new JPanel(new BorderLayout());
        //Componentes
        chassisLab = new JLabel("Chassis");
        chassis = new JTextField(15);
        result = new JTextArea(10, 40);
        consult = new JButton("Consultar");
        consult.addActionListener(this);
        result.setBorder(new TitledBorder(new
EtchedBorder(EtchedBorder.RAISED), "Resultado"));
        //Agrupando componentes
        topo.add(chassisLab);
        topo.add(chassis);
        topo.add(consult);
        meio.add(result, BorderLayout.CENTER);
        telaR.add(topo, BorderLayout.NORTH);
        telaR.add(meio, BorderLayout.CENTER);
    }

    public JPanel getPanel()
    {
        return telaR;
    }

    public void actionPerformed(ActionEvent evt)
    {
        String strChassis;

        if(evt.getSource() == consult)
        {
            strChassis = chassis.getText().trim();
            result.setText(InterfaceTela.consulta(strChassis));
        }
    }
}

```

TelaDeleta.java

```

/*@author JavaXT
 *Tela de Remoção de veículos
 */
package projeto.interfaces.gui;

import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JTextArea;
import javax.swing.JPanel;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.BorderLayout;
import javax.swing.border.TitledBorder;
import javax.swing.border.EtchedBorder;

public class TelaDeleta implements ActionListener
{
    private JPanel telaD;
    private JPanel topo;
    private JPanel meio;
    private JButton remove;
    private JLabel chassisLab;

```

```

private JTextField chassis;
private JTextArea result;

public TelaDeleta()
{
    //Painéis
    telaD = new JPanel(new BorderLayout());
    topo = new JPanel();
    meio = new JPanel();
    //Componentes
    chassisLab = new JLabel("Chassis");
    chassis = new JTextField(15);
    remove = new JButton("Remover");
    result = new JTextArea(5, 40);
    result.setBorder(new TitledBorder(new
EtchedBorder(EtchedBorder.RAISED), "Resultado"));
    //Agrupando
    topo.add(chassisLab);
    topo.add(chassis);
    topo.add(remove);
    remove.addActionListener(this);
    meio.add(result);
    telaD.add(topo, BorderLayout.NORTH);
    telaD.add(meio, BorderLayout.CENTER);
}

public JPanel getPanel()
{
    return telaD;
}

public void actionPerformed(ActionEvent evt)
{
    if(evt.getSource() == remove)
    {
        try
        {

result.setText(InterfaceTela.removeVeiculo(chassis.getText()));

        }
        catch (Exception e)
        {
            InterfaceTela.mostraExc(e.toString());
        }
    }
}
}

```

TelaLogin.java

```

/*@author JavaXT
 *Tela de Login
 */
package projeto.interfaces.gui;

import javax.swing.JFrame;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JPasswordField;

```

```

import javax.swing.JOptionPane;
import java.awt.GridLayout;
import java.awt.Container;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.SQLException;

public class TelaLogin extends InterfaceTela implements ActionListener
{
    private static JFrame janela;
    private static Container tela;
    private static JButton connect;
    private static JLabel hostLab;
    private static JLabel userLab;
    private static JLabel pwdLab;
    private static JTextField host;
    private static JTextField user;
    private static JPasswordField pwd;

    public TelaLogin()
    {
        //Definindo a estrutura da janela
        janela = new JFrame("Conectar");
        janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        janela.setSize(300, 100);
        janela.setResizable(false);
        tela = janela.getContentPane();
        tela.setLayout(new GridLayout(4, 2));

        //Labels
        hostLab = new JLabel("Host");
        userLab = new JLabel("Usuário");
        pwdLab = new JLabel("Senha");

        //TextFields
        host = new JTextField(50);
        user = new JTextField(50);
        pwd = new JPasswordField(50);

        //Botao
        connect = new JButton("Conectar");
        connect.addActionListener(this);

        //Coloca todo mundo na tela
        tela.add(hostLab);
        tela.add(host);
        tela.add(userLab);
        tela.add(user);
        tela.add(pwdLab);
        tela.add(pwd);
        tela.add(connect);

        //Exibe a janela
        janela.setVisible(true);
    }

    public void actionPerformed(ActionEvent evt)
    {
        if(evt.getSource() == connect)
        {

```

```

        try
        {
            //15 é o número correspondente ao BD
PostgreSQL. Para acrescentar outros tipos de bd,
            //é preciso criar um novo campo na interface
            connect(host.getText(), user.getText(), new
String(pwd.getPassword()), 15);
        }
        catch(SQLException e)
        {
            mostraExc(e.toString());
        }
        catch(ClassNotFoundException e)
        {
            mostraExc(e.toString());
        }
    }

    public static void dispose()
    {
        janela.dispose();
    }
}

```