RELATÓRIO DE CRIAÇÃO DESENVOLVIMENTO E IMPLEMENTAÇÃO PROJETO GYNLOG

Projeto: Controle de gastos da frota veicular
Versão do Documento: RCDIP01
Data de Início: 28/11/2025
Local / Plataforma: FATESG/SENAI

---

1. Participantes

Alexsander de Almeida Nogueira
Caio Nunes de Abreu
Cassiano Nunes de Abreu
Gabriel Naoki Uto Turigoe
Wyllian Mariano Nogueira

2. Objetivo

Pensar e desenvolver um projeto de gestão de frota, que permita a criação de cadastro de novos automóveis, informações e dados importantes como placa, ano, modelo, cor e Renavam, que são essenciais para a organização financeira e legal da empresa solicitante **GynLog**.

3. Contexto e Justificativa do Projeto

Desenvolver uma solução para o gerenciamento financeiro para a empresa **GynLog**, utilizando um software em Java (Swing), e o sistema deve permitir registro, categorização e emissão de relatórios de despesas, armazenar os dados em arquivos de texto (.txt), e uma infraestrutura de rede.

4. Arquitetura do Sistema

4.1 Paradigma de Desenvolvimento

O sistema foi desenvolvido com o uso de **Paradigma de Programação Orientada a Objetos (POO)**, com aplicação dos seguinte conceitos:

1. **Encapsulamento:** Proteção de dados através de modificadores de acesso
2. **Herança:** Uso de classes base para reutilização de código
3. **Polimorfismo:** Adaptação na implementação de métodos
4. **Abstração:** Classes abstratas para geração de relatórios
5. **Tratamento de Exceções:** Validações e capturas de erros

4.2 Paradigma de Desenvolvimento

- *src/*
  - *model/*
    - *Veiculo.java*
    - *TipoDespesa.java*
    - *Movimento.java*
  - *view/*
    - *TelaInicio.java*
    - *TelaRegistroVeiculo.java*
    - *TelaPesquisaVeiculo.java*
    - *TelaRegistroPrejuizo.java*
    - *telaGerarRelatorios.java*
  - *util/*
    - *ArquivoTXT_Veiculo.java*
    - *ArquivoTXT_Movimento.java*
    - *ArquivoTXT_Despesa.java*
    - *ArquivoExcel_Veiculo.java*
    - *ArquivoExcel_Movimento.java*
    - *GeradorRelatorios.java*
    - *ThemeAdm.java*
    - *relatorioDespesaVeiculo.java*
    - *relatorioDespesaTotalFrota.java*
    - *relatorioGastoMensalCombustivelTotalFrota.java*
    - *relatorioIPVATotalAnualFrota.java*
    - *relatorioTotalMulasVeiculos.java*
    - *relatorioVeicuosInativos.java*

5. Inicializadores e Métodos

5.1 inicializadores

Inicializadores servem para transformar o “conceito” de um botão ou janela em algo real, que conseguimos visualizar, nós utilizamos alguns inicializadores para cada objetivo de visualização, logo abaixo estamos identificando quais utilizamos.

Tela Slash:

```
public TelaSlash() {
        setUndecorated(true);
        initComponents();

        UIManager.put("nimbusOrange", new Color(80, 120, 255));
        SwingUtilities.updateComponentTreeUI(this);

        configurarJanela();
        iniciarCarregamento();
        ThemeAdm.addRememberOnClose(this);
        ThemeAdm.applyTheme(this);
    }
```

Tela Início:

```
public TelaInicio() {
        initComponents();
        ThemeAdm.setTheme(ThemeAdm.loadSavedTheme());
        ThemeAdm.applyThemeAndSetup(this);
        setLocationRelativeTo(null);
        this.setTitle("GynLog");
    }
```

Tela Registro Veículo:

```
public TelaRegistroVeiculo() {
        initComponents();
        inicializarMarcasModelos();
        adicionarPlaceholders();
        configurarAutocomplete();
        adicionarValidacoesFoco();
        configurarComboBoxDisponibilidade();
        configurarCampoIdAutomatico();
        permitirApenasNumeros(jTFAnoFabric);
        setLocationRelativeTo(null);
        ThemeAdm.setTheme(ThemeAdm.loadSavedTheme());
        ThemeAdm.applyThemeAndSetup(this);
        this.setTitle("GynLog");
    }
```

Tela Lista Veículo:

```
public TelaListaVeiculo() {
        initComponents();
        configurarTabela();
        configurarComboFiltro();
        carregarDadosNaTabela();
        configurarSelecaoTabela();
        setLocationRelativeTo(null);
        this.setTitle("GynLog");
        ThemeAdm.setTheme(ThemeAdm.loadSavedTheme());
        ThemeAdm.applyThemeAndSetup(this);
    }
```

Tela Pesquisa Veículo:

```
public TelaPesquisaVeiculo() {
        initComponents();
        inicializarMarcasModelos();
        configurarComponentes();
        setLocationRelativeTo(null);
        this.setTitle("GynLog");
        ThemeAdm.setTheme(ThemeAdm.loadSavedTheme());
        ThemeAdm.applyThemeAndSetup(this);
    }
```

Tela Registro Prejuízo:

```
public TelaRegistroPrejuizo() {
        initComponents();
        adicionarPlaceholders();
        configurarCampoIdAutomatico();
        configurarCalendario();
        configurarComboBoxTipoDespesa();
        permitirApenasNumeros(jTFEncontrarID);
        adicionarValidacoesFoco();
        setLocationRelativeTo(null);
        ThemeAdm.setTheme(ThemeAdm.loadSavedTheme());
        ThemeAdm.applyThemeAndSetup(this);
        this.setTitle("GynLog");
    }
```

Tela Gerar Relatórios:

```
public telaGerarRelatorios() {
        initComponents();
        jTFBuscaRelatorio.setEditable(true);
        jTFBuscaRelatorio.setEnabled(true);
        ThemeAdm.setTheme(ThemeAdm.loadSavedTheme());
        ThemeAdm.applyThemeAndSetup(this);
    }
```

5.2 Métodos

Os métodos são o que chamamos de “ação”, eles servem para mudar os componentes (você manda nele) e também para reagir ao usuários (ele obedece a um evento), e para conseguirmos apresentar o que foi realizado, vou seguir com evidências visuais do que conseguimos utilizando os métodos.

5.2.1. Tela Slash

Nome: configurarJanela()
Descrição: Realiza as configurações visuais e comportamentais, definindo a largura da barra de progresso e inicializando-a com valor zero (0). Além disso, define o texto inicial da legenda de carregamento a partir de uma lista rotativa de frases pré-definidas.

```
private void configurarJanela() {
        setLocationRelativeTo(null);

                // Barra de carregamento (altura 5px) ---------------------------------
        jPBLoading.setPreferredSize(new
java.awt.Dimension(jPBLoading.getPreferredSize().width, 10));

        // Inicia barra de progresso em 0 -------------------------------------
        jPBLoading.setValue(0);
        jPBLoading.setStringPainted(true);

        // Texto carregamento -------------------------------------------------
        jLInitializing.setText("Inicializando sistema...");
    }
```

Nome: iniciarCarregamento()
Descrição: Ao executar o arquivo, este método é acionado, estabelecendo a duração necessária para que a barra de carregamento seja concluída e o tempo de atraso (delay) para a inicialização do sistema. Uma vez que a barra de carregamento atinge o final, o método prossegue ativando o próximo método descrito adiante.

```
private void iniciarCarregamento() {
        SwingWorker<Void, Integer> worker = new SwingWorker<Void, Integer>() {
            @Override
            protected Void doInBackground() throws Exception {
                int progresso = 0;
                int incremento = 5;
                int delayBase = 200;

                while (progresso <= 100) {
                    publish(progresso);

                    // Delay variável para parecer mais natural ----------------
                    int delay = delayBase + random.nextInt(120); // 200-300ms
                    Thread.sleep(delay);
```

```
            progresso += incremento;
        }
        return null;
    }

    @Override
    protected void process(java.util.List<Integer> chunks) {
        int ultimoProgresso = chunks.get(chunks.size() - 1);

        jPBLoading.setValue(ultimoProgresso);

        // Rolagem de frases ----------------------------------------
        if (ultimoProgresso < 100) {
            int indiceFrase = (ultimoProgresso / 5) % FRASES_CARREGAMENTO.length;
            jLInitializing.setText(FRASES_CARREGAMENTO[indiceFrase]);
        } else {
            jLInitializing.setText("Carregamento concluído!");
        }
    }

    @Override
    protected void done() {
        try {
            Thread.sleep(300);
        } catch (InterruptedException e) {
            logger.log(java.util.logging.Level.WARNING, "Interrupção no sleep final", e);
        }

        // Abre tela principal (precisa de uma tela principal) ---------
        abrirTelaPrincipal();
    }
};
worker.execute();
}
```

Nome: abrirTelaPrincipal()
Descrição: Com esse método, executado no método anterior, o sistema abre janela TelaInicio.java e em seguida fecha a janela TelaSlash.java execução deste método, que se segue ao método anterior, resulta na abertura da janela TelaInicio.java e, consequentemente, no fechamento da janela Tela Slash.java.

```
private void abrirTelaPrincipal() {
    java.awt.EventQueue.invokeLater(() -> {
        new TelaInicio().setVisible(true);
        dispose();
    });
}
```

5.2.2. TelaRegistroVeiculo
Método: inicializarMarcasModelos()
Descrição: Esse método cria uma lista de sugestões locais de nomes de veículos e marcas.

```
private void inicializarMarcasModelos(){
    // Volkswagen ----------------------------------------------------
    marcasModelos.put("Volkswagen", Arrays.asList(
        "Delivery 4.150", // caminhão leve
        "Delivery 11.180", // caminhão leve
        "Constellation 24.280", // caminhão pesado
        "Constellation 33.460", // caminhão extrapesado
        "Worker 13.180", // caminhão
        "Worker 17.210", // caminhão
        "Amarok", // caminhonete
        "Amarok V6", // caminhonete
        "Kombi Furgão", // van
        "Volksbus 9-160", // ônibus urbano
        "Volksbus 17-230", // ônibus rodoviário
        "Crafter 2.0", // van
        "Crafter Truck", // caminhão leve
        "Transporter T6", // van
        "Transporter Chassis" // caminhão leve
    ));

    // Mercedes-Benz -------------------------------------------------
    marcasModelos.put("Mercedes-Benz", Arrays.asList(
        "Accelo 815", // caminhão leve
        "Accelo 1016", // caminhão leve
        "Atego 1719", // caminhão médio
        "Atego 2426", // caminhão pesado
        "Actros 2651", // caminhão extrapesado
        "Actros 2553", // caminhão extrapesado
        "Sprinter 314 CDI", // van
        "Sprinter 415 CDI", // van
        "Sprinter 515 CDI", // van
        "Sprinter Street", // van
        "OF-1519", // ônibus urbano
        "OH-1622", // ônibus rodoviário
        "O-500 RSD", // ônibus rodoviário
        "X-Class", // caminhonete
        "G-Class Pickup (custom)" // caminhonete
    ));

    // Ford ----------------------------------------------------------
    marcasModelos.put("Ford", Arrays.asList(
        "F-250", // caminhonete pesada
        "F-350", // caminhonete pesada
```

```
        "F-4000", // caminhão leve
        "F-4000 4x4", // caminhão leve
        "Cargo 1119", // caminhão leve
        "Cargo 1519", // caminhão médio
        "Cargo 2429", // caminhão pesado
        "Cargo 3131", // caminhão pesado
        "Transit Van", // van
        "Transit Furgão", // van
        "Transit Chassi", // caminhão leve
        "Torino Custom" // ônibus urbano (modificado)
));

// Chevrolet ----------------------------------------------------------
marcasModelos.put("Chevrolet", Arrays.asList(
        "S10", // caminhonete
        "S10 High Country", // caminhonete
        "Silverado 1500", // caminhonete
        "Silverado 2500 HD", // caminhonete pesada
        "Silverado 3500 HD", // caminhonete pesada
        "Chevy Van G20", // van
        "Express 2500", // van
        "Express 3500 Cutaway" // caminhão leve
));

// Fiat ---------------------------------------------------------------
marcasModelos.put("Fiat", Arrays.asList(
        "Ducato Furgão", // van
        "Ducato Minibus", // van
        "Ducato Cargo Maxi", // van
        "Scudo", // van
        "Talento", // van
        "Toro", // caminhonete leve
        "Fullback" // caminhonete
));

// Iveco --------------------------------------------------------------
marcasModelos.put("Iveco", Arrays.asList(
        "Daily 35-150", // van
        "Daily 45-170", // caminhão leve
        "Daily Minibus", // van
        "Tector 11-190", // caminhão médio
        "Tector 24-300", // caminhão pesado
        "Hi-Way 480", // caminhão extrapesado
        "Hi-Road 440", // caminhão extrapesado
        "Hi-Way 560", // caminhão extrapesado
        "S-Way 570", // caminhão extrapesado
        "CityClass", // ônibus urbano
        "Popstar" // micro-ônibus
```

```
));

// Volvo ---------------------------------------------------------------
marcasModelos.put("Volvo", Arrays.asList(
        "FH 460", // caminhão extrapesado
        "FH 540", // caminhão extrapesado
        "FH 500", // caminhão extrapesado
        "FM 420", // caminhão pesado
        "FMX 540", // caminhão fora de estrada
        "B270F", // ônibus urbano
        "B310R", // ônibus rodoviário
        "B450R" // ônibus rodoviário
));

// Scania --------------------------------------------------------------
marcasModelos.put("Scania", Arrays.asList(
        "R 450", // caminhão extrapesado
        "R 500", // caminhão extrapesado
        "S 500", // caminhão extrapesado
        "S 620", // caminhão extrapesado
        "P 360", // caminhão médio
        "P 280", // caminhão leve
        "K 310 IB", // ônibus rodoviário
        "F 280", // ônibus urbano
        "K 400 6x2" // ônibus rodoviário
));

// MAN -----------------------------------------------------------------
marcasModelos.put("MAN", Arrays.asList(
        "TGX 28.440", // caminhão extrapesado
        "TGX 29.480", // caminhão extrapesado
        "TGS 26.440", // caminhão pesado
        "TGM 26.290", // caminhão médio
        "TGL 12.250", // caminhão leve
        "Lion's Coach", // ônibus rodoviário
        "Lion's City" // ônibus urbano
));

// Renault -------------------------------------------------------------
marcasModelos.put("Renault", Arrays.asList(
        "Master Furgão", // van
        "Master Minibus", // van
        "Master Chassi", // caminhão leve
        "Trafic", // van
        "Alaskan" // caminhonete
));

// Peugeot -------------------------------------------------------------
```

```
marcasModelos.put("Peugeot", Arrays.asList(
        "Boxer Cargo", // van
        "Boxer Minibus", // van
        "Expert", // van
        "Rifter", // van
        "Landtrek" // caminhonete
));

// Citroën ------------------------------------------------------------
marcasModelos.put("Citroën", Arrays.asList(
        "Jumper Furgão", // van
        "Jumpy", // van
        "Spacetourer", // van
        "Berlingo Van" // van
));

// Toyota -------------------------------------------------------------
marcasModelos.put("Toyota", Arrays.asList(
        "Hilux", // caminhonete
        "Hilux CD SRX", // caminhonete
        "Hilux Chassi", // caminhão leve
        "Coaster", // micro-ônibus
        "HiAce" // van
));

// Nissan -------------------------------------------------------------
marcasModelos.put("Nissan", Arrays.asList(
        "Frontier", // caminhonete
        "Frontier XE", // caminhonete
        "NV350 Urvan", // van
        "Titan XD" // caminhonete pesada
));

// Mitsubishi ---------------------------------------------------------
marcasModelos.put("Mitsubishi", Arrays.asList(
        "L200 Triton", // caminhonete
        "L200 Savana", // caminhonete
        "L300", // van
        "Fuso Canter 6.5", // caminhão leve
        "Fuso Canter 8.5" // caminhão leve
));

// Hyundai ------------------------------------------------------------
marcasModelos.put("Hyundai", Arrays.asList(
        "HR 2.5", // caminhão leve
        "HD 80", // caminhão leve
        "HD 160", // caminhão médio
        "County", // micro-ônibus
```

```
        "Solati" // van
));

// Kia ----------------------------------------------------------------
marcasModelos.put("Kia", Arrays.asList(
        "Bongo K2500", // caminhão leve
        "Bongo K2700", // caminhão leve
        "Grand Carnival", // van
        "Pregio", // van
        "K9 Van" // van
));

// Ram ----------------------------------------------------------------
marcasModelos.put("Ram", Arrays.asList(
        "Ram 1500", // caminhonete
        "Ram 2500", // caminhonete pesada
        "Ram 3500", // caminhonete extrapesada
        "ProMaster Van", // van
        "ProMaster Rapid" // van
));

// Sprinter (custom brand) ---------------------------------------------
marcasModelos.put("Sprinter Custom", Arrays.asList(
        "Sprinter Ambulância", // van
        "Sprinter Escolar", // van
        "Sprinter Executiva", // van
        "Sprinter Fretamento" // van
));

// Agrale -------------------------------------------------------------
marcasModelos.put("Agrale", Arrays.asList(
        "Marruá AM200", // caminhonete militar
        "Marruá AM300", // caminhonete militar
        "Agrale 8700", // ônibus
        "Agrale 10500", // ônibus
        "Agrale 14000 S" // caminhão médio
));

// Marcopolo (carroceria) ----------------------------------------------
marcasModelos.put("Marcopolo", Arrays.asList(
        "Torino", // ônibus urbano
        "Paradiso 1200", // ônibus rodoviário
        "Paradiso 1800 DD", // ônibus rodoviário double deck
        "Viaggio 1050", // ônibus rodoviário
        "Senior Midi" // micro-ônibus
));

// Caio Induscar -------------------------------------------------------
```

```
        marcasModelos.put("Caio", Arrays.asList(
                "Apache Vip", // ônibus urbano
                "Millennium", // ônibus urbano
                "Mondego", // ônibus rodoviário
                "Foz Super", // micro-ônibus
                "Tile Ade" // ônibus rodoviário
        ));
            //marcasModelos.put("Peugeot", Arrays.asList("")); modelo para mais marcas
    }
```

Método: configurarAutocomplete()
Descrição: Esse método realiza a automação de correção, por exemplo, ao escrever algo o usuário vai ter uma ajuda automática sobre o que ele está escrevendo.

```
private void configurarAutocomplete(){
        // Sugestão automatica de modelo ---------------------------------------
        configurarDropdownSugestao(jTFModelo, false);
    }
```

Método: configurarCampoIdAutomatico()
Descrição: Esse método permite que o usuário administrador proíba o usuário a escrever nesse campo, já que o próprio sistema vai definir o ‘auto’ ID

```
private void configurarCampoIdAutomatico() {
        // Torna o campo não editável ----------------------------------------
        jTFVeiculoID.setEditable(false);
    }
```

Método: configurarDropdownSugestao()
Descrição: Esse método permite a configuração da movimentação do usuário, é mais uma correção de um erro sistêmico que não permitia que o usuário corresse de maneira direta.

```
// Configuração de dropdown -------------------------------------------------
    private void configurarDropdownSugestao(javax.swing.JTextField campo, boolean
isMarca) {
        javax.swing.JPopupMenu dropdown = new javax.swing.JPopupMenu();
        dropdown.setFocusable(false);

        final java.util.List<javax.swing.JMenuItem> menuItems = new java.util.ArrayList<>();
        final int[] selectedIndex = {-1};
        final boolean[] enterPressed = {false};

        final Color COR_NORMAL = Color.WHITE;
        final Color COR_DESTAQUE = new Color(230, 240, 255);

        campo.addKeyListener(new java.awt.event.KeyAdapter() {

            @Override
```

```
public void keyPressed(java.awt.event.KeyEvent evt) {
    if (dropdown.isVisible()) {
        switch (evt.getKeyCode()) {
            case java.awt.event.KeyEvent.VK_DOWN:
                evt.consume();
                if (selectedIndex[0] < menuItems.size() - 1) {
                    if (selectedIndex[0] >= 0) {
                        menuItems.get(selectedIndex[0]).setBackground(COR_NORMAL);
                    }
                    selectedIndex[0]++;
                    menuItems.get(selectedIndex[0]).setBackground(COR_DESTAQUE);
                }
                break;
            case java.awt.event.KeyEvent.VK_UP:
                evt.consume();
                if (selectedIndex[0] > 0) {
                    menuItems.get(selectedIndex[0]).setBackground(COR_NORMAL);
                    selectedIndex[0]--;
                    menuItems.get(selectedIndex[0]).setBackground(COR_DESTAQUE);
                } else if (selectedIndex[0] == 0) {
                    menuItems.get(selectedIndex[0]).setBackground(COR_NORMAL);
                    selectedIndex[0] = -1;
                }
                break;
            case java.awt.event.KeyEvent.VK_ENTER:
                evt.consume();
                enterPressed[0] = true;
                if (selectedIndex[0] >= 0 && selectedIndex[0] < menuItems.size()) {
                    menuItems.get(selectedIndex[0]).doClick();
                }
                dropdown.setVisible(false);
                menuItems.clear();
                selectedIndex[0] = -1;
                break;
            case java.awt.event.KeyEvent.VK_ESCAPE:
                evt.consume();
                dropdown.setVisible(false);
                selectedIndex[0] = -1;
                break;
        }
    }
}
@Override
public void keyReleased(java.awt.event.KeyEvent evt) {
    if (enterPressed[0]) {
        enterPressed[0] = false; // Reseta a flag
        return; // Sai sem fazer nada
    }
```

```
String texto = campo.getText().trim().toLowerCase();

if (dropdown.isVisible() && (
        evt.getKeyCode() == java.awt.event.KeyEvent.VK_UP ||
        evt.getKeyCode() == java.awt.event.KeyEvent.VK_DOWN ||
        evt.getKeyCode() == java.awt.event.KeyEvent.VK_ENTER ||
        evt.getKeyCode() == java.awt.event.KeyEvent.VK_ESCAPE)) {
    return;
}

if (texto.isEmpty() || evt.getKeyCode() == java.awt.event.KeyEvent.VK_ESCAPE) {
    dropdown.setVisible(false);
    menuItems.clear();
    selectedIndex[0] = -1;
    return;
}

java.util.List<String> sugestoes = new java.util.ArrayList<>();
java.util.Map<String, String> modeloParaMarca = new java.util.HashMap<>();

for (String marca : marcasModelos.keySet()) {
    for (String modelo : marcasModelos.get(marca)) {
        if (modelo.toLowerCase().startsWith(texto)) {
            sugestoes.add(modelo);
            modeloParaMarca.put(modelo, marca);
        }
    }
}

if (sugestoes.size() >= 10) {
    sugestoes = sugestoes.subList(0, 10);
}

dropdown.removeAll();
menuItems.clear();
selectedIndex[0] = -1;

if (!sugestoes.isEmpty()) {
    for (int i = 0; i < sugestoes.size(); i++) {
        String sugestao = sugestoes.get(i);
        javax.swing.JMenuItem item = new javax.swing.JMenuItem(sugestao);

        item.setFocusable(false);
        item.setBackground(COR_NORMAL);
        item.setOpaque(true);

        final int indice = i;
        item.addMouseListener(new java.awt.event.MouseAdapter() {
```

```
                    @Override
                    public void mouseEntered(java.awt.event.MouseEvent e) {
                        if (selectedIndex[0] >= 0 && selectedIndex[0] < menuItems.size()) {
                            menuItems.get(selectedIndex[0]).setBackground(COR_NORMAL);
                        }
                        selectedIndex[0] = indice;
                        item.setBackground(COR_DESTAQUE);
                    }
                });
                item.addActionListener(e -> {
                    campo.setText(sugestao);
                    campo.setForeground(Color.black);

                    String marcaCorrespondente = modeloParaMarca.get(sugestao);
                    if (marcaCorrespondente != null) {
                        jTFMarca.setText(marcaCorrespondente);
                        jTFMarca.setForeground(Color.black);
                    }

                    dropdown.setVisible(false);
                    menuItems.clear();
                    selectedIndex[0] = -1;
                    campo.requestFocusInWindow();
                });

                dropdown.add(item);
                menuItems.add(item);
            }

            dropdown.show(campo, 0, campo.getHeight());
        } else {
            dropdown.setVisible(false);
        }
    }
});
// Remove dropdown quando campo perde foco ----------------------------------------------------
----------------------
campo.addFocusListener(new java.awt.event.FocusAdapter() {
    @Override
    public void focusLost(java.awt.event.FocusEvent evt) {
        javax.swing.SwingUtilities.invokeLater(() -> {
            dropdown.setVisible(false);
            menuItems.clear();
            selectedIndex[0] = -1;
        });
    }
});
}
```

Método: permitirApenasNumeros()
Descrição: Esse método restringe que o usuário digite apenas números inteiros.

```
// Apenas recebe numeros inteiros no campo jTFAnoFabric ----------------------------------------
--------------------
   private void permitirApenasNumeros(javax.swing.JTextField campo) {
      campo.addKeyListener(new java.awt.event.KeyAdapter() {
         @Override
         public void keyTyped(java.awt.event.KeyEvent evt) {
            char c = evt.getKeyChar();
            if (!Character.isDigit(c) && c != java.awt.event.KeyEvent.VK_BACK_SPACE && c
!= java.awt.event.KeyEvent.VK_DELETE) {
               evt.consume(); // Bloqueia o caractere
            }
         }
      });
   }
```

Método: aplicarPlaceholder()
Descrição: Esse método permite que o usuário consiga uma fácil visualização, já que ele “limpa” a área em que o usuário vai colocar alguma informação, como um ‘mini-tutorial’

```
// Metodos de Placeholder ----------------------------------------------------------------------
--------------------
   private void aplicarPlaceholder(javax.swing.JTextField campo, String placeholder) {
      final Color PLACEHOLDER_COLOR = ThemeAdm.getTheme() ==
ThemeAdm.Theme.LIGHT ? Color.GRAY : new Color(150, 150, 150);

      campo.setText(placeholder);
      campo.setForeground(PLACEHOLDER_COLOR);

      campo.addFocusListener(new java.awt.event.FocusAdapter() {
         @Override
         public void focusGained(java.awt.event.FocusEvent evt) {
            if (campo.getText().equals(placeholder)) {
               campo.setText("");
               campo.setForeground(ThemeAdm.getTextColor());
            }
         }
         @Override
         public void focusLost(java.awt.event.FocusEvent evt) {
            if (campo.getText().isEmpty()) {
               campo.setText(placeholder);
               campo.setForeground(PLACEHOLDER_COLOR);
            }
         }
      });
   }
```

Método: atualizarPlaceholders()
Descrição: Esse método vai definir o que é que vai ser descrito no método anterior.

```
private void atualizarPlaceholders() {
      final Color PLACEHOLDER_COLOR = ThemeAdm.getTheme() ==
ThemeAdm.Theme.LIGHT ? Color.GRAY : new Color(150, 150, 150);

      javax.swing.JTextField[] campos = {jTFVeiculoID, jTFPlaca, jTFMarca, jTFModelo,
jTFAnoFabric};
      String[] placeholders = {
         "ID do veículo",
         "Placa do veículo",
         "Marca do veículo",
         "Modelo do veículo",
         "Ano de fabricação"
      };

      for (int i = 0; i < campos.length; i++) {
         if (campos[i].getText().equals(placeholders[i])) {
            campos[i].setForeground(PLACEHOLDER_COLOR);
         } else if (!campos[i].getText().isEmpty()) {
            campos[i].setForeground(ThemeAdm.getTextColor());
         }
      }
   }
```

Método: adicionarPlaceholders()
Descrição: Esse método acrescenta informação quando o campo estiver vazio.

```
// Atribuição de textos Placeholders em campos vazios ----------------------
   private void adicionarPlaceholders() {
      aplicarPlaceholder(jTFVeiculoID, "ID do veículo");
      aplicarPlaceholder(jTFPlaca, "Placa do veículo");
      aplicarPlaceholder(jTFMarca, "Marca do veículo");
      aplicarPlaceholder(jTFModelo, "Modelo do veículo");
      aplicarPlaceholder(jTFAnoFabric, "Ano de fabricação");
   }
```

Método: isPlacaValidaMercosul()
Descrição: Esse método determina que é necessário que os dados estejam conforme o que é definido pelo Mercosul.

```
// Verificação de padronização legal de placas originais -------------------
   private static boolean isPlacaValidaMercosul(String placa) {
      String regex = "^[A-Z]{3}-?[0-9][A-Z0-9][0-9]{2}$";
      return placa.toUpperCase().matches(regex);
   }
```

Método: configurarComboBoxDisponibilidade()
Descrição: Esse método define se o veículo registrado está “Disponível” ou “Indisponível”.

```
private void configurarComboBoxDisponibilidade() {
    jCBDisponibilidade.setSelectedIndex(0);

    jCBDisponibilidade.addActionListener(e -> {
        int index = jCBDisponibilidade.getSelectedIndex();
        if (index == 0) {
            // "---" selecionado - estado neutro
        } else if (index == 1) {
            // "Disponivel" selecionado
            System.out.println("Status: Disponível");
        } else if (index == 2) {
            // "Indisponível" selecionado
            System.out.println("Status: Indisponível");
        }
    });
}
```

Método: limparCampos()
Descrição: Esse método permite que ao clicar no botão registrar, ele envia todas as informações e limpa os campos, para caso seja necessário a criação de mais cadastros.

```
private void limparCampos() {
    jTFVeiculoID.setText("ID do veículo");
    jTFVeiculoID.setForeground(Color.GRAY);

    jTFPlaca.setText("Placa do veículo");
    jTFPlaca.setForeground(Color.GRAY);

    jTFMarca.setText("Marca do veículo");
    jTFMarca.setForeground(Color.GRAY);

    jTFModelo.setText("Modelo do veículo");
    jTFModelo.setForeground(Color.GRAY);

    jTFAnoFabric.setText("Ano de Fabricação");
    jTFAnoFabric.setForeground(Color.GRAY);

    jCBDisponibilidade.setSelectedIndex(0);

    jTFPlaca.requestFocus();
}
```

Método: obterDisponibilidadeDoComboBox()
Descrição: Esse método impede o prosseguimento do registro caso o usuário não tenha definido a disponibilidade do veículo a ser registrado

```
private Boolean obterDisponibilidadeDoComboBox() {
    int selectedIndex = jCBDisponibilidade.getSelectedIndex();

    if (selectedIndex == 0) {
        JOptionPane.showMessageDialog(this,
                "Por favor, selecione a disponibilidade do veículo.",
                "Campo Obrigatório",
                JOptionPane.WARNING_MESSAGE
        );
        jCBDisponibilidade.requestFocus();
        return null;
    } else if (selectedIndex == 1) {
        return true;
    } else {
        return false;
    }
}
```

Método: adicionarValidacoesFoco()
Descrição: Esse método define o comportamento do sistema com as validações, esse gatilho acontece quando o usuário sai do campo de texto sem cumprir com os requisitos necessários.

```
private void adicionarValidacoesFoco() {
    // Validação do Id ao perder o foco ----------------------------------
    jTFVeiculoID.addFocusListener(new java.awt.event.FocusAdapter() {
        @Override
        public void focusLost(java.awt.event.FocusEvent evt) {
            String raw = jTFVeiculoID.getText().trim();

            if (raw.isEmpty() || raw.equals("ID do veículo")) {
                return;
            }

            if (!raw.matches("\\d+")) {
                JOptionPane.showMessageDialog(TelaRegistroVeiculo.this,
                        "O ID deve conter apenas números",
                        "Erro de Validação",
                        JOptionPane.ERROR_MESSAGE);
                jTFVeiculoID.requestFocus();
                return;
            }

            if (raw.length() != 6) {
```

```
                JOptionPane.showMessageDialog(TelaRegistroVeiculo.this,
                        "O ID deve possuir exatamente 6 dígitos.",
                        "Erro de Validação",
                        JOptionPane.ERROR_MESSAGE);
                jTFVeiculoID.requestFocus();
                return;
            }

            try {
                int id = Integer.parseInt(raw);
                java.util.ArrayList<model.Veiculo> Lista = util.ArquivoTXT_Veiculo.LerArquivo();
                for (model.Veiculo veiculo : Lista) {
                    if (veiculo.getIdVeiculo() == id) {
                        JOptionPane.showMessageDialog(TelaRegistroVeiculo.this,
                                "Já existe um veículo com esse ID.",
                                "ID Duplicado",
                                JOptionPane.ERROR_MESSAGE);
                        jTFVeiculoID.requestFocus();
                        return;
                    }
                }
            } catch (Exception e) {
                System.err.println("Erro ao verificar duplicidade: " + e.getMessage());
            }
        }
    });

    // Validação da Placa ao perder foco ----------------------------------
    jTFPlaca.addFocusListener(new java.awt.event.FocusAdapter() {
        @Override
        public void focusLost(java.awt.event.FocusEvent evt) {
            String placaRaw = jTFPlaca.getText().trim();

            if (placaRaw.isEmpty() || placaRaw.equals("Placa do veículo")) {
                return;
            }

            String placa = placaRaw.replaceAll("[^A-Za-z0-9]", "").toUpperCase();

            if (!isPlacaValidaMercosul(placa)) {
                JOptionPane.showMessageDialog(TelaRegistroVeiculo.this,
                        "Placa inválida. Padrão esperado: Mercosul (ABC1D23) ou Placa Cinza
(ABC1234)",
                        "Erro de Validação",
                        JOptionPane.ERROR_MESSAGE);
                jTFPlaca.requestFocus();
                return;
            }
```

```
            if(util.ArquivoTXT_Veiculo.placaJaExiste(placa)) {
                JOptionPane.showMessageDialog(TelaRegistroVeiculo.this,
                        "Já existe um veículo com a placa: " + placa + "\n" +
                        "Por favor, verifique se não é um registro duplicado.",
                        "Placa Duplicada",
                        JOptionPane.ERROR_MESSAGE
                );
                jTFPlaca.requestFocus();
                return;
            }

            jTFPlaca.setText(placa);

            int proximoId = util.ArquivoTXT_Veiculo.gerarProximoId();
            jTFVeiculoID.setText(String.valueOf(proximoId));
            jTFVeiculoID.setForeground(Color.BLACK);
        }
    });

    // Validação do Modelo ao perder foco ---------------------------------
    jTFModelo.addFocusListener(new java.awt.event.FocusAdapter() {
        @Override
        public void focusLost(java.awt.event.FocusEvent evt) {
            String raw = jTFModelo.getText().trim();

            if (raw.isEmpty() || raw.equals("Modelo do veículo")) {
                return;
            }

            // Padroniza (primeira letra maiúscula) ------------------------
            String modeloPadr = raw.substring(0,1).toUpperCase() + (raw.length() > 1 ?
raw.substring(1) : "");
            jTFModelo.setText(modeloPadr);
        }
    });

    // Validação do Ano ao perder foco ------------------------------------
    jTFAnoFabric.addFocusListener(new java.awt.event.FocusAdapter() {
        @Override
        public void focusLost(java.awt.event.FocusEvent evt) {
            String raw = jTFAnoFabric.getText().trim();

            if (raw.isEmpty() || raw.equals("Ano de fabricação")) {
                return;
            }

            try {
```

```
            int ano = Integer.parseInt(raw);
            int anoAtual = java.time.Year.now().getValue();
            int anoLimite = anoAtual + 1;

            if (ano < 1886 || ano > anoLimite) {
                JOptionPane.showMessageDialog(TelaRegistroVeiculo.this,
                        "Ano inválido. Aceito entre 1886 e " + anoLimite + ".",
                        "Erro de Validação",
                        JOptionPane.ERROR_MESSAGE);
                jTFAnoFabric.requestFocus();
            }
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(TelaRegistroVeiculo.this,
                    "Ano deve ser um número inteiro.",
                    "Erro de Validação",
                    JOptionPane.ERROR_MESSAGE);
            jTFAnoFabric.requestFocus();
        }
    }
});
}
```

Método: adicionarValidacoesFoco()
Descrição: Esse método consiste em validações que são gatilhos quando o usuário sai do campo de texto. O método lê a informação registrada e define se condiz com os requisitos necessários

```
private void adicionarValidacoesFoco() {
    // Validação do Id ao perder o foco ----------------------------------
    jTFVeiculoID.addFocusListener(new java.awt.event.FocusAdapter() {
        @Override
        public void focusLost(java.awt.event.FocusEvent evt) {
            String raw = jTFVeiculoID.getText().trim();

            if (raw.isEmpty() || raw.equals("ID do veículo")) {
                return;
            }

            if (!raw.matches("\\d+")) {
                JOptionPane.showMessageDialog(TelaRegistroVeiculo.this,
                        "O ID deve conter apenas números",
                        "Erro de Validação",
                        JOptionPane.ERROR_MESSAGE);
                jTFVeiculoID.requestFocus();
                return;
            }

            if (raw.length() != 6) {
```

```
                JOptionPane.showMessageDialog(TelaRegistroVeiculo.this,
                        "O ID deve possuir exatamente 6 dígitos.",
                        "Erro de Validação",
                        JOptionPane.ERROR_MESSAGE);
                jTFVeiculoID.requestFocus();
                return;
            }

            try {
                int id = Integer.parseInt(raw);
                java.util.ArrayList<model.Veiculo> Lista = util.ArquivoTXT_Veiculo.LerArquivo();
                for (model.Veiculo veiculo : Lista) {
                    if (veiculo.getIdVeiculo() == id) {
                        JOptionPane.showMessageDialog(TelaRegistroVeiculo.this,
                                "Já existe um veículo com esse ID.",
                                "ID Duplicado",
                                JOptionPane.ERROR_MESSAGE);
                        jTFVeiculoID.requestFocus();
                        return;
                    }
                }
            } catch (Exception e) {
                System.err.println("Erro ao verificar duplicidade: " + e.getMessage());
            }
        }
    });

    // Validação da Placa ao perder foco ----------------------------------
    jTFPlaca.addFocusListener(new java.awt.event.FocusAdapter() {
        @Override
        public void focusLost(java.awt.event.FocusEvent evt) {
            String placaRaw = jTFPlaca.getText().trim();

            if (placaRaw.isEmpty() || placaRaw.equals("Placa do veículo")) {
                return;
            }

            String placa = placaRaw.replaceAll("[^A-Za-z0-9]", "").toUpperCase();

            if (!isPlacaValidaMercosul(placa)) {
                JOptionPane.showMessageDialog(TelaRegistroVeiculo.this,
                        "Placa inválida. Padrão esperado: Mercosul (ABC1D23) ou Placa Cinza
(ABC1234)",
                        "Erro de Validação",
                        JOptionPane.ERROR_MESSAGE);
                jTFPlaca.requestFocus();
                return;
            }
```

```
            if(util.ArquivoTXT_Veiculo.placaJaExiste(placa)) {
                JOptionPane.showMessageDialog(TelaRegistroVeiculo.this,
                        "Já existe um veículo com a placa: " + placa + "\n" +
                        "Por favor, verifique se não é um registro duplicado.",
                        "Placa Duplicada",
                        JOptionPane.ERROR_MESSAGE
                );
                jTFPlaca.requestFocus();
                return;
            }

            jTFPlaca.setText(placa);

            int proximoId = util.ArquivoTXT_Veiculo.gerarProximoId();
            jTFVeiculoID.setText(String.valueOf(proximoId));
            jTFVeiculoID.setForeground(Color.BLACK);
        }
    });

    // Validação do Modelo ao perder foco ---------------------------------
    jTFModelo.addFocusListener(new java.awt.event.FocusAdapter() {
        @Override
        public void focusLost(java.awt.event.FocusEvent evt) {
            String raw = jTFModelo.getText().trim();

            if (raw.isEmpty() || raw.equals("Modelo do veículo")) {
                return;
            }

            // Padroniza (primeira letra maiúscula) ------------------------
            String modeloPadr = raw.substring(0,1).toUpperCase() + (raw.length() > 1 ?
raw.substring(1) : "");
            jTFModelo.setText(modeloPadr);
        }
    });

    // Validação do Ano ao perder foco ------------------------------------
    jTFAnoFabric.addFocusListener(new java.awt.event.FocusAdapter() {
        @Override
        public void focusLost(java.awt.event.FocusEvent evt) {
            String raw = jTFAnoFabric.getText().trim();

            if (raw.isEmpty() || raw.equals("Ano de fabricação")) {
                return;
            }

            try {
```

```
                int ano = Integer.parseInt(raw);
                int anoAtual = java.time.Year.now().getValue();
                int anoLimite = anoAtual + 1;

                if (ano < 1886 || ano > anoLimite) {
                    JOptionPane.showMessageDialog(TelaRegistroVeiculo.this,
                            "Ano inválido. Aceito entre 1886 e " + anoLimite + ".",
                            "Erro de Validação",
                            JOptionPane.ERROR_MESSAGE);
                    jTFAnoFabric.requestFocus();
                }
            } catch (NumberFormatException e) {
                JOptionPane.showMessageDialog(TelaRegistroVeiculo.this,
                        "Ano deve ser um número inteiro.",
                        "Erro de Validação",
                        JOptionPane.ERROR_MESSAGE);
                jTFAnoFabric.requestFocus();
            }
        }
    });
  }
```

5.2.3. TelaRegistroVeiculo
Método: configurarTabela()
Descrição: Esse método permite que o usuário administrador consiga configurar informações da tabela.

```
private void configurarTabela() {
        jTDados.setDefaultEditor(Object.class, null);

        // Seleção de uma linha individual ---------------------------------

jTDados.setSelectionMode(javax.swing.ListSelectionModel.MULTIPLE_INTERVAL_SELEC
TION);
  }
```

Método: carregarDadosNaTabela()
Descrição: Esse método recebe os dados do arquivo (.txt).

```
// Recebe dados do arquivo Veiculo.txt -----------------------------------
   private void carregarDadosNaTabela() {
       String caminho = "Veiculo.txt";

       DefaultTableModel model = (DefaultTableModel) jTDados.getModel();
       model.setRowCount(0);

       try (BufferedReader br = new BufferedReader(new FileReader(caminho))) {
           String linha;
```

```
boolean primeiraLinha = true;

while ((linha = br.readLine()) != null) {
    linha = linha.trim();

    // Pula linhas vazias
    if (linha.isEmpty()) {
        continue;
    }

    // Pula o cabeçalho ----------------------------------------
    if(primeiraLinha) {
        primeiraLinha = false;
        continue;
    }

    String[] dados = linha.split("\\s*\\|\\s*"); // obrigatorio!!!!!!!!

    // Verificação do tamanho da linha
    if (dados.length != 6) {
        throw new IllegalArgumentException(
            "Linha do arquivo com formato incorreto \n" + linha
        );
    }

    // Filtro de disponibilidade ---------------------------------
    String statusTexto = dados[5].trim();
    boolean disp = statusTexto.equalsIgnoreCase("Disponível") ||
                statusTexto.equalsIgnoreCase("Disponivel") ||
                statusTexto.equalsIgnoreCase("True");

    boolean deveExibir = false;

    switch (filtroAtual) {
        case TODOS:
            deveExibir = true;
            break;
        case DISPONIVEIS:
            deveExibir = disp;
            break;
        case INDISPONIVEIS:
            deveExibir = !disp;
            break;
    }

    if (deveExibir) {
        for (int i = 0; i < dados.length; i++) {
            dados[i] = dados[i].trim();
```

```
                }

                dados[5] = disp ? "Disponível" : "Indisponível";
                // Adiciona a linha de dados ao modelo da tabela ----------
                model.addRow(dados);
            }
        }
    } catch (IOException e) {
        JOptionPane.showMessageDialog(this,
                "Erro ao ler o arquivo TXT: " + e.getMessage(),
                "Erro",
                JOptionPane.ERROR_MESSAGE
        );
    }
}
```

Método: alternarDisponibilidadeVeiculo()
Descrição: Esse método permite que ao clicar na linha ou em múltiplas linhas ele vai alterar o status do veículo.

```
// Alteração de disponibilidade ----------------------------------------
private void alternarDisponibilidadeVeiculo() {
    int[] linhasSelecionadas = jTDados.getSelectedRows();

    if(linhasSelecionadas.length == 0) {
        JOptionPane.showMessageDialog(this,
            "Por favor, selecione um ou mais veículos para alterar a disponibilidade.",
            "Nenhum veículo selecionado",
            JOptionPane.WARNING_MESSAGE
        );
        return;
    }

    DefaultTableModel model = (DefaultTableModel) jTDados.getModel();

    // Coleta informações dos veículos selecionados
    ArrayList<String> idsParaAlterar = new ArrayList<>();
    StringBuilder infoVeiculos = new StringBuilder();

    // Verifica se há mistura de disponíveis e indisponíveis
    int qtdDisponiveis = 0;
    int qtdIndisponiveis = 0;

    for (int i = 0; i < linhasSelecionadas.length; i++) {
        int linha = linhasSelecionadas[i];
        String idVeiculo = model.getValueAt(linha, 0).toString().trim();
        String placa = model.getValueAt(linha, 1).toString().trim();
        String marca = model.getValueAt(linha, 2).toString().trim();
```

```
    String modeloVeiculo = model.getValueAt(linha, 3).toString().trim();
    String statusAtual = model.getValueAt(linha, 5).toString().trim();

    idsParaAlterar.add(idVeiculo);

    if (statusAtual.equals("Disponível")) {
        qtdDisponiveis++;
    } else {
        qtdIndisponiveis++;
    }

    infoVeiculos.append(String.format("  • %s %s (Placa: %s) - %s\n",
                                marca, modeloVeiculo, placa, statusAtual));
}

// Define a ação baseada na seleção
String acao;
String novoStatus;
boolean tornarDisponivel;

if (linhasSelecionadas.length == 1) {
    // Comportamento original para 1 veículo
    String statusAtual = model.getValueAt(linhasSelecionadas[0], 5).toString().trim();
    boolean estaDisponivel = statusAtual.equals("Disponível");
    tornarDisponivel = !estaDisponivel;
    novoStatus = estaDisponivel ? "Indisponível" : "Disponível";
    acao = "alternar";
} else {
    // Para múltiplos veículos, pergunta qual ação tomar
    if (qtdDisponiveis > 0 && qtdIndisponiveis > 0) {
        // Há mistura - pergunta o que fazer
        Object[] opcoes = {"Marcar todos como Disponível",
                           "Marcar todos como Indisponível",
                           "Cancelar"};
        int escolha = JOptionPane.showOptionDialog(this,
                String.format("Você selecionou %d veículos:\n" +
                              "- %d disponíveis\n" +
                              "- %d indisponíveis\n\n" +
                              "O que deseja fazer?",
                              linhasSelecionadas.length, qtdDisponiveis, qtdIndisponiveis),
                "Escolha a ação",
                JOptionPane.YES_NO_CANCEL_OPTION,
                JOptionPane.QUESTION_MESSAGE,
                null,
                opcoes,
                opcoes[0]);

        if (escolha == 0) {
```

```
                tornarDisponivel = true;
                novoStatus = "Disponível";
                acao = "marcar como disponível";
            } else if (escolha == 1) {
                tornarDisponivel = false;
                novoStatus = "Indisponível";
                acao = "marcar como indisponível";
            } else {
                return; // Cancelou
            }
        } else if (qtdDisponiveis > 0) {
            // Todos disponíveis - marcar como indisponível
            tornarDisponivel = false;
            novoStatus = "Indisponível";
            acao = "marcar como indisponível";
        } else {
            // Todos indisponíveis - marcar como disponível
            tornarDisponivel = true;
            novoStatus = "Disponível";
            acao = "marcar como disponível";
        }
    }

    // Confirmação com o usuário
    String mensagem;
    if (linhasSelecionadas.length == 1) {
        mensagem = String.format("Deseja alterar a disponibilidade do
veículo?\n\n%s\nNovo status: %s",
                infoVeiculos.toString(), novoStatus);
    } else {
        mensagem = String.format("Deseja %s %d veículos?\n\n%s\nNovo status de todos:
%s",
                acao, linhasSelecionadas.length, infoVeiculos.toString(), novoStatus);
    }

    int confirmacao = JOptionPane.showConfirmDialog(this,
            mensagem,
            "Confirmar Alteração",
            JOptionPane.YES_NO_OPTION,
            JOptionPane.QUESTION_MESSAGE
    );
    if (confirmacao != JOptionPane.YES_OPTION) {
        return;
    }
    try {
        ArrayList<model.Veiculo> listaVeiculos = util.ArquivoTXT_Veiculo.LerArquivo();
        int alterados = 0;
        for (String id : idsParaAlterar) {
```

```
            for (model.Veiculo veiculo : listaVeiculos) {
                if (String.valueOf(veiculo.getIdVeiculo()).equals(id)) {
                    veiculo.setAtivo(tornarDisponivel);
                    alterados++;
                    System.out.println("Status alterado: ID=" + id + " → " + novoStatus);
                    break;
                }
            }
        }

        if (alterados == 0) {
            JOptionPane.showMessageDialog(this,
                    "Nenhum veículo foi encontrado no sistema.",
                    "Erro",
                    JOptionPane.ERROR_MESSAGE
            );
            return;
        }

        util.ArquivoTXT_Veiculo.AtualizarTxtExcel(listaVeiculos);
        util.ArquivoExcel_Veiculo.Transf_Excel(listaVeiculos, "Veiculo.xlsx");

        String mensagemSucesso;
        if (alterados == 1) {
            mensagemSucesso = String.format("Disponibilidade alterada com sucesso!\nNovo
status: %s", novoStatus);
        } else {
            mensagemSucesso = String.format("%d veículos alterados com sucesso!\nNovo
status: %s", alterados, novoStatus);
        }

        JOptionPane.showMessageDialog(this,
                mensagemSucesso,
                "Sucesso",
                JOptionPane.INFORMATION_MESSAGE
        );

        // Salva os IDs antes de recarregar
        ArrayList<String> idsParaReselecionar = new ArrayList<>(idsParaAlterar);

        // Recarrega a tabela
        carregarDadosNaTabela();

        // Tenta reselecionar os veículos
        jTDados.clearSelection();
        for (int i = 0; i < jTDados.getRowCount(); i++) {
            String idLinha = model.getValueAt(i, 0).toString();
            if (idsParaReselecionar.contains(idLinha)) {
```

```
                jTDados.addRowSelectionInterval(i, i);
            }
        }

    } catch (Exception e) {
        JOptionPane.showMessageDialog(this,
                "Erro ao alterar disponibilidade:\n" + e.getMessage(),
                "Erro",
                JOptionPane.ERROR_MESSAGE
        );
        e.printStackTrace();
    }
}
```

Método: configurarSelecaoTabela()
Descrição: Esse método permite que o usuário consiga selecionar múltiplos elementos usando as teclas SHIFT e CTRL permitindo edição de múltiplos veículos cadastrados

```
// Adicione um SelectionListener na tabela para atualizar o botão
private void configurarSelecaoTabela() {
    jTDados.getSelectionModel().addListSelectionListener(e -> {
        if (!e.getValueIsAdjusting()) {
            int[] linhasSelecionadas = jTDados.getSelectedRows();

            if (linhasSelecionadas.length == 0) {
                jBtnDisponib.setText("Alterar Disponibilidade");
                jBtnDisponib.setEnabled(false);

            } else if (linhasSelecionadas.length == 1) {
                // Uma linha selecionada - comportamento original
                DefaultTableModel model = (DefaultTableModel) jTDados.getModel();
                String status = model.getValueAt(linhasSelecionadas[0], 5).toString().trim();

                if (status.equalsIgnoreCase("Disponível")) {
                    jBtnDisponib.setText("Marcar como Indisponível");
                } else {
                    jBtnDisponib.setText("Marcar como Disponível");
                }
                jBtnDisponib.setEnabled(true);
            } else {
                // Múltiplas linhas selecionadas
                jBtnDisponib.setText(String.format("Alterar %d veículos",
linhasSelecionadas.length));
                jBtnDisponib.setEnabled(true);
            }
        }
    });
}
```

Método: deletarVeiculoSelecionado()
Descrição: Esse método permite que o usuário consiga deletar o veículo selecionado.

```
   // Deletando linha selecionada ------------------------------------------
   private void deletarVeiculoSelecionado() {
      int[] linhasSelecionadas = jTDados.getSelectedRows();

      if (linhasSelecionadas.length == 0) {
         JOptionPane.showMessageDialog(this,
               "Por favor, selecione um ou mais veículos para deletar.",
               "Nenhum veículo selecionado",
               JOptionPane.WARNING_MESSAGE
         );
         return;
      }

      DefaultTableModel model = (DefaultTableModel) jTDados.getModel();

      // Coleta informações dos veículos selecionados
      ArrayList<String> idsParaDeletar = new ArrayList<>();
      StringBuilder infoVeiculos = new StringBuilder();

      for (int i = 0; i < linhasSelecionadas.length; i++) {
         int linha = linhasSelecionadas[i];
         String idVeiculo = model.getValueAt(linha, 0).toString().trim();
         String placa = model.getValueAt(linha, 1).toString().trim();
         String modeloVeiculo = model.getValueAt(linha, 3).toString().trim();

         idsParaDeletar.add(idVeiculo);
         infoVeiculos.append(String.format("  • ID: %s | Placa: %s | Modelo: %s\n",
                                 idVeiculo, placa, modeloVeiculo));
      }

      // Confirmação com o usuário
      String mensagem;
      if (linhasSelecionadas.length == 1) {
         mensagem = "Tem certeza que deseja deletar o veículo?\n\n" +
infoVeiculos.toString();
      } else {
         mensagem = String.format("Tem certeza que deseja deletar %d veículos?\n\n%s",
                        linhasSelecionadas.length, infoVeiculos.toString());
      }

      int confirmacao = JOptionPane.showConfirmDialog(this,
            mensagem,
            "Confirmar Exclusão",
            JOptionPane.YES_NO_OPTION,
```

```
            JOptionPane.WARNING_MESSAGE
        );

        if (confirmacao != JOptionPane.YES_OPTION) {
            return;
        }

        // Leitura de todos os veículos
        ArrayList<model.Veiculo> listaVeiculos = util.ArquivoTXT_Veiculo.LerArquivo();

        // Remove os veículos com os IDs correspondentes
        int removidos = 0;
        for (String id : idsParaDeletar) {
            boolean removeu = listaVeiculos.removeIf(v ->
String.valueOf(v.getIdVeiculo()).equals(id));
            if (removeu) removidos++;
        }

        if (removidos == 0) {
            JOptionPane.showMessageDialog(this,
                "Nenhum veículo foi encontrado na lista.",
                "Erro",
                JOptionPane.ERROR_MESSAGE
            );
            return;
        }

        util.ArquivoTXT_Veiculo.AtualizarTxtExcel(listaVeiculos);
        util.ArquivoExcel_Veiculo.Transf_Excel(listaVeiculos, "Veiculo.xlsx");

        String mensagemSucesso;
        if (removidos == 1) {
            mensagemSucesso = "Veículo deletado com sucesso!";
        } else {
            mensagemSucesso = String.format("%d veículos deletados com sucesso!",
removidos);
        }

        JOptionPane.showMessageDialog(this,
            mensagemSucesso,
            "Sucesso",
            JOptionPane.INFORMATION_MESSAGE
        );

        carregarDadosNaTabela();
    }
```

Método: configurarComboFiltro()
Descrição: Esse método usa um combobox para que o usuário selecione filtros de pesquisa.

```
    // Configurações da Combo Box ---------------------------------------------
    private void configurarComboFiltro() {
        jComboBox1.removeAllItems();
        jComboBox1.addItem("Todos");
        jComboBox1.addItem("Disponíveis");
        jComboBox1.addItem("Indisponíveis");

        jComboBox1.addActionListener(e -> {
            String opcao = jComboBox1.getSelectedItem().toString();

            switch (opcao) {
                case "Todos":
                    alternarFiltro(FiltroDisponibilidade.TODOS);
                    break;
                case "Disponíveis":
                    alternarFiltro(FiltroDisponibilidade.DISPONIVEIS);
                    break;
                case "Indisponíveis":
                    alternarFiltro(FiltroDisponibilidade.INDISPONIVEIS);
                    break;
            }
        });
    }
```

Método: alternarFiltro()
Descrição: Esse método permite que o usuário consiga se movimentar nos filtros de disponibilidade.

```
// Altera filtro de disponibilidade ---------------------------------------
    private void alternarFiltro(FiltroDisponibilidade novoFiltro) {
        filtroAtual = novoFiltro;
        carregarDadosNaTabela();
    }
```

5.2.4. TelaPesquisaVeiculo()
Método: inicializarMarcasModelos()
Descrição: Esse método permite a criação de modelos e marcas de veículos.

```
// Lista de veículos
    private void inicializarMarcasModelos(){
        // Volkswagen ----------------------------------------------------
        marcasModelos.put("Volkswagen", Arrays.asList(
                "Delivery 4.150", // caminhão leve
                "Delivery 11.180", // caminhão leve
                "Constellation 24.280", // caminhão pesado
                "Constellation 33.460", // caminhão extrapesado
                "Worker 13.180", // caminhão
                "Worker 17.210", // caminhão
                "Amarok", // caminhonete
                "Amarok V6", // caminhonete
                "Kombi Furgão", // van
                "Volksbus 9-160", // ônibus urbano
                "Volksbus 17-230", // ônibus rodoviário
                "Crafter 2.0", // van
                "Crafter Truck", // caminhão leve
                "Transporter T6", // van
                "Transporter Chassis" // caminhão leve
        ));

        // Mercedes-Benz -------------------------------------------------
        marcasModelos.put("Mercedes-Benz", Arrays.asList(
                "Accelo 815", // caminhão leve
                "Accelo 1016", // caminhão leve
                "Atego 1719", // caminhão médio
                "Atego 2426", // caminhão pesado
                "Actros 2651", // caminhão extrapesado
                "Actros 2553", // caminhão extrapesado
                "Sprinter 314 CDI", // van
                "Sprinter 415 CDI", // van
                "Sprinter 515 CDI", // van
                "Sprinter Street", // van
                "OF-1519", // ônibus urbano
                "OH-1622", // ônibus rodoviário
                "O-500 RSD", // ônibus rodoviário
                "X-Class", // caminhonete
                "G-Class Pickup (custom)" // caminhonete
        ));

        // Ford ----------------------------------------------------------
        marcasModelos.put("Ford", Arrays.asList(
                "F-250", // caminhonete pesada
                "F-350", // caminhonete pesada
```

```
        "F-4000", // caminhão leve
        "F-4000 4x4", // caminhão leve
        "Cargo 1119", // caminhão leve
        "Cargo 1519", // caminhão médio
        "Cargo 2429", // caminhão pesado
        "Cargo 3131", // caminhão pesado
        "Transit Van", // van
        "Transit Furgão", // van
        "Transit Chassi", // caminhão leve
        "Torino Custom" // ônibus urbano (modificado)
));

// Chevrolet ----------------------------------------------------------
marcasModelos.put("Chevrolet", Arrays.asList(
        "S10", // caminhonete
        "S10 High Country", // caminhonete
        "Silverado 1500", // caminhonete
        "Silverado 2500 HD", // caminhonete pesada
        "Silverado 3500 HD", // caminhonete pesada
        "Chevy Van G20", // van
        "Express 2500", // van
        "Express 3500 Cutaway" // caminhão leve
));

// Fiat ---------------------------------------------------------------
marcasModelos.put("Fiat", Arrays.asList(
        "Ducato Furgão", // van
        "Ducato Minibus", // van
        "Ducato Cargo Maxi", // van
        "Scudo", // van
        "Talento", // van
        "Toro", // caminhonete leve
        "Fullback" // caminhonete
));

// Iveco --------------------------------------------------------------
marcasModelos.put("Iveco", Arrays.asList(
        "Daily 35-150", // van
        "Daily 45-170", // caminhão leve
        "Daily Minibus", // van
        "Tector 11-190", // caminhão médio
        "Tector 24-300", // caminhão pesado
        "Hi-Way 480", // caminhão extrapesado
        "Hi-Road 440", // caminhão extrapesado
        "Hi-Way 560", // caminhão extrapesado
        "S-Way 570", // caminhão extrapesado
        "CityClass", // ônibus urbano
        "Popstar" // micro-ônibus
```

```
));

// Volvo ---------------------------------------------------------------
marcasModelos.put("Volvo", Arrays.asList(
        "FH 460", // caminhão extrapesado
        "FH 540", // caminhão extrapesado
        "FH 500", // caminhão extrapesado
        "FM 420", // caminhão pesado
        "FMX 540", // caminhão fora de estrada
        "B270F", // ônibus urbano
        "B310R", // ônibus rodoviário
        "B450R" // ônibus rodoviário
));

// Scania --------------------------------------------------------------
marcasModelos.put("Scania", Arrays.asList(
        "R 450", // caminhão extrapesado
        "R 500", // caminhão extrapesado
        "S 500", // caminhão extrapesado
        "S 620", // caminhão extrapesado
        "P 360", // caminhão médio
        "P 280", // caminhão leve
        "K 310 IB", // ônibus rodoviário
        "F 280", // ônibus urbano
        "K 400 6x2" // ônibus rodoviário
));

// MAN -----------------------------------------------------------------
marcasModelos.put("MAN", Arrays.asList(
        "TGX 28.440", // caminhão extrapesado
        "TGX 29.480", // caminhão extrapesado
        "TGS 26.440", // caminhão pesado
        "TGM 26.290", // caminhão médio
        "TGL 12.250", // caminhão leve
        "Lion's Coach", // ônibus rodoviário
        "Lion's City" // ônibus urbano
));

// Renault -------------------------------------------------------------
marcasModelos.put("Renault", Arrays.asList(
        "Master Furgão", // van
        "Master Minibus", // van
        "Master Chassi", // caminhão leve
        "Trafic", // van
        "Alaskan" // caminhonete
));

// Peugeot -------------------------------------------------------------
```

```
marcasModelos.put("Peugeot", Arrays.asList(
        "Boxer Cargo", // van
        "Boxer Minibus", // van
        "Expert", // van
        "Rifter", // van
        "Landtrek" // caminhonete
));

// Citroën ------------------------------------------------------------
marcasModelos.put("Citroën", Arrays.asList(
        "Jumper Furgão", // van
        "Jumpy", // van
        "Spacetourer", // van
        "Berlingo Van" // van
));

// Toyota -------------------------------------------------------------
marcasModelos.put("Toyota", Arrays.asList(
        "Hilux", // caminhonete
        "Hilux CD SRX", // caminhonete
        "Hilux Chassi", // caminhão leve
        "Coaster", // micro-ônibus
        "HiAce" // van
));

// Nissan -------------------------------------------------------------
marcasModelos.put("Nissan", Arrays.asList(
        "Frontier", // caminhonete
        "Frontier XE", // caminhonete
        "NV350 Urvan", // van
        "Titan XD" // caminhonete pesada
));

// Mitsubishi ---------------------------------------------------------
marcasModelos.put("Mitsubishi", Arrays.asList(
        "L200 Triton", // caminhonete
        "L200 Savana", // caminhonete
        "L300", // van
        "Fuso Canter 6.5", // caminhão leve
        "Fuso Canter 8.5" // caminhão leve
));

// Hyundai ------------------------------------------------------------
marcasModelos.put("Hyundai", Arrays.asList(
        "HR 2.5", // caminhão leve
        "HD 80", // caminhão leve
        "HD 160", // caminhão médio
        "County", // micro-ônibus
```

```
        "Solati" // van
));

// Kia ----------------------------------------------------------------
marcasModelos.put("Kia", Arrays.asList(
        "Bongo K2500", // caminhão leve
        "Bongo K2700", // caminhão leve
        "Grand Carnival", // van
        "Pregio", // van
        "K9 Van" // van
));

// Ram ----------------------------------------------------------------
marcasModelos.put("Ram", Arrays.asList(
        "Ram 1500", // caminhonete
        "Ram 2500", // caminhonete pesada
        "Ram 3500", // caminhonete extrapesada
        "ProMaster Van", // van
        "ProMaster Rapid" // van
));

// Sprinter (custom brand) ---------------------------------------------
marcasModelos.put("Sprinter Custom", Arrays.asList(
        "Sprinter Ambulância", // van
        "Sprinter Escolar", // van
        "Sprinter Executiva", // van
        "Sprinter Fretamento" // van
));

// Agrale -------------------------------------------------------------
marcasModelos.put("Agrale", Arrays.asList(
        "Marruá AM200", // caminhonete militar
        "Marruá AM300", // caminhonete militar
        "Agrale 8700", // ônibus
        "Agrale 10500", // ônibus
        "Agrale 14000 S" // caminhão médio
));

// Marcopolo (carroceria) ----------------------------------------------
marcasModelos.put("Marcopolo", Arrays.asList(
        "Torino", // ônibus urbano
        "Paradiso 1200", // ônibus rodoviário
        "Paradiso 1800 DD", // ônibus rodoviário double deck
        "Viaggio 1050", // ônibus rodoviário
        "Senior Midi" // micro-ônibus
));

// Caio Induscar -------------------------------------------------------
```

```
        marcasModelos.put("Caio", Arrays.asList(
                "Apache Vip", // ônibus urbano
                "Millennium", // ônibus urbano
                "Mondego", // ônibus rodoviário
                "Foz Super", // micro-ônibus
                "Tile Ade" // ônibus rodoviário
        ));
            //marcasModelos.put("Peugeot", Arrays.asList("")); modelo para mais marcas
    }
```

Método: configurarComponentes()
Descrição: Esse método permite que o usuário ao clicar no campo “outro” consiga cadastrar uma nova categoria.

```
// Configuração de componentes ------------------
    private void configurarComponentes() {
        // Configurar tabela como não editável
        jTDadosPesquisa.setDefaultEditor(Object.class, null);

        // Configurar campo de entrada inicialmente invisível
        jTFEntrada.setVisible(false);
        jLTexto.setVisible(false);
        jTFEntrada.setText("");

        // Configurar listener do ComboBox
        jCBFiltros.addActionListener(e -> configurarCampoPesquisa());
    }
```

Método: configurarCampoPesquisa()
Descrição: Esse método permite que o usuário consiga selecionar um filtro x de informação, e nesse filtro ele vai conseguir colocar especificações pré definidas.

```
// COnfiguração do campo de pesquisa usando filtros
    private void configurarCampoPesquisa() {
        String filtroSelecionado = jCBFiltros.getSelectedItem().toString();

        // Limpa a tabela
        limparTabela();

        // Remove listeners anteriores
        removerListeners(jTFEntrada);

        switch (filtroSelecionado) {
            case "---":
                jTFEntrada.setVisible(false);
                jLTexto.setVisible(false);
                limparTabela();
                break;
```

```
case "Id do Veículo":
    jTFEntrada.setVisible(true);
    jLTexto.setVisible(true);
    jTFEntrada.setText("");
    jLTexto.setText("Digite o ID:");
    configurarCampoId();
    break;

case "Placa":
    jTFEntrada.setVisible(true);
    jLTexto.setVisible(true);
    jTFEntrada.setText("");
    jLTexto.setText("Digite a Placa:");
    configurarCampoPlaca();
    break;

case "Modelo":
    jTFEntrada.setVisible(true);
    jLTexto.setVisible(true);
    jTFEntrada.setText("");
    jLTexto.setText("Digite o Modelo:");
    configurarCampoModelo();
    break;

case "Marca":
    jTFEntrada.setVisible(true);
    jLTexto.setVisible(true);
    jTFEntrada.setText("");
    jLTexto.setText("Digite a Marca:");
    configurarCampoMarca();
    break;

case "Ano de Fabricação":
    jTFEntrada.setVisible(true);
    jLTexto.setVisible(true);
    jTFEntrada.setText("");
    jLTexto.setText("Digite o Ano:");
    configurarCampoAno();
    break;

case "Disponíveis":
    jTFEntrada.setVisible(false);
    jLTexto.setVisible(false);
    pesquisarDisponiveis(true);
    break;

case "Indisponíveis":
```

```
            jTFEntrada.setVisible(false);
            jLTexto.setVisible(false);
            pesquisarDisponiveis(false);
            break;
        }

        jTFEntrada.requestFocus();
    }
```

Método: configurarCampoId()
Descrição: Esse método configura uma definição específica do que pode ser acrescentado em um campo específico.

```
// Filtro: ID
    private void configurarCampoId() {
        // Aceita apenas números
        jTFEntrada.addKeyListener(new java.awt.event.KeyAdapter() {
            @Override
            public void keyTyped(java.awt.event.KeyEvent evt) {
                char c = evt.getKeyChar();
                if (!Character.isDigit(c) && c != java.awt.event.KeyEvent.VK_BACK_SPACE) {
                    evt.consume();
                }
            }

            @Override
            public void keyReleased(java.awt.event.KeyEvent evt) {
                String texto = jTFEntrada.getText().trim();
                if (!texto.isEmpty()) {
                    pesquisarPorId(texto);
                } else {
                    limparTabela();
                }
            }
        });
    }
```

Método: pesquisarPorId()
Descrição: Esse método permite o uso de filtros dentro do combobox.

```
    private void pesquisarPorId(String idTexto) {
        try {
            int id = Integer.parseInt(idTexto);
            ArrayList<Veiculo> todosVeiculos = util.ArquivoTXT_Veiculo.LerArquivo();
            ArrayList<Veiculo> resultados = new ArrayList<>();

            for (Veiculo v : todosVeiculos) {
                if (String.valueOf(v.getIdVeiculo()).contains(idTexto)) {
```

```
                resultados.add(v);
            }
        }

        if (resultados.isEmpty() && idTexto.length() == 6) {
            int resposta = JOptionPane.showConfirmDialog(this,
                "ID não encontrado. Deseja registrar um novo veículo?",
                "ID não cadastrado",
                JOptionPane.YES_NO_OPTION,
                JOptionPane.QUESTION_MESSAGE
            );

            if (resposta == JOptionPane.YES_OPTION) {
                new TelaRegistroVeiculo().setVisible(true);
                this.dispose();
            }
        }

        preencherTabela(resultados);

    } catch (NumberFormatException e) {
        limparTabela();
    }
}
```

Método: configurarCampoPlaca()
Descrição: Esse método utiliza filtros que permite movimentação dentro do combobox.

```
// Filtro: Placa --------------------------------
    private void configurarCampoPlaca() {
        jTFEntrada.addKeyListener(new java.awt.event.KeyAdapter() {
            @Override
            public void keyReleased(java.awt.event.KeyEvent evt) {
                String texto = jTFEntrada.getText().trim().toUpperCase();
                if (!texto.isEmpty()) {
                    pesquisarPorPlaca(texto);
                } else {
                    limparTabela();
                }
            }
        });

        jTFEntrada.addFocusListener(new java.awt.event.FocusAdapter() {
            @Override
            public void focusLost(java.awt.event.FocusEvent evt) {
                String placa = jTFEntrada.getText().trim();
                if (!placa.isEmpty() && !isPlacaValidaMercosul(placa)) {
                    JOptionPane.showMessageDialog(TelaPesquisaVeiculo.this,
```

```
                    "Placa inválida. Formato esperado: Mercosul (ABC1D23) ou Placa Cinza
(ABC1234)",
                    "Formato Inválido",
                    JOptionPane.WARNING_MESSAGE
                );
            }
        }
    });
}
```

Método: pesquisarPorPlaca()
Descrição: Esse método permite que o usuário realize pesquisa por placa.

```
private void pesquisarPorPlaca(String placa) {
    ArrayList<Veiculo> todosVeiculos = util.ArquivoTXT_Veiculo.LerArquivo();
    ArrayList<Veiculo> resultados = new ArrayList<>();

    for (Veiculo v : todosVeiculos) {
        if (v.getPlaca().toUpperCase().contains(placa)) {
            resultados.add(v);
        }
    }

    if (resultados.isEmpty() && isPlacaValidaMercosul(placa)) {
        int resposta = JOptionPane.showConfirmDialog(this,
            "Placa não encontrada. Deseja registrar um novo veículo?",
            "Placa não cadastrada",
            JOptionPane.YES_NO_OPTION,
            JOptionPane.QUESTION_MESSAGE
        );

        if (resposta == JOptionPane.YES_OPTION) {
            new TelaRegistroVeiculo().setVisible(true);
            this.dispose();
        }
    }

    preencherTabela(resultados);
}
```

Método: isPlacaValidaMercosul()
Descrição: Esse método restringe que a informação colocada seja conforme o que é indicado pelo Mercosul.

```
private boolean isPlacaValidaMercosul(String placa) {
    String regex = "^[A-Z]{3}-?[0-9][A-Z0-9][0-9]{2}$";
    return placa.toUpperCase().matches(regex);
}
```

Método: configurarCampoModelo()
Descrição: Esse método usa métodos de configuração das sugestões de marcas e modelos já registrados para sugerir o conteúdo baseado no preenchimento manual do usuário no campo de texto.

```
// Filtro: Modelo
    private void configurarCampoModelo() {
        configurarDropdownSugestao(jTFEntrada, false);

        jTFEntrada.addKeyListener(new java.awt.event.KeyAdapter() {
            @Override
            public void keyReleased(java.awt.event.KeyEvent evt) {
                String texto = jTFEntrada.getText().trim();
                if (!texto.isEmpty()) {
                    pesquisarPorModelo(texto);
                } else {
                    limparTabela();
                }
            }
        });
    }
```

Método: pesquisarPorModelo()
Descrição: Esse método permite ao usuário a pesquisa por modelo de veículos.

```
    private void pesquisarPorModelo(String modelo) {
        ArrayList<Veiculo> todosVeiculos = util.ArquivoTXT_Veiculo.LerArquivo();
        ArrayList<Veiculo> resultados = new ArrayList<>();

        for (Veiculo v : todosVeiculos) {
            if (v.getModelo().toLowerCase().contains(modelo.toLowerCase())) {
                resultados.add(v);
            }
        }

        if (resultados.isEmpty()) {
            // Verifica se o modelo existe na lista de modelos cadastrados
            boolean modeloExiste = false;
            for (List<String> modelos : marcasModelos.values()) {
                for (String m : modelos) {
                    if (m.equalsIgnoreCase(modelo)) {
                        modeloExiste = true;
                        break;
                    }
                }
            }

            if (modeloExiste) {
```

```
            JOptionPane.showMessageDialog(this,
                "Modelo '" + modelo + "' existe na lista, mas não há veículos cadastrados.",
                "Sem resultados",
                JOptionPane.INFORMATION_MESSAGE
            );
        } else {
            int resposta = JOptionPane.showConfirmDialog(this,
                "Modelo não encontrado. Pode estar digitado incorretamente.\n" +
                "Deseja registrar um novo veículo?",
                "Modelo não cadastrado",
                JOptionPane.YES_NO_OPTION,
                JOptionPane.QUESTION_MESSAGE
            );

            if (resposta == JOptionPane.YES_OPTION) {
                new TelaRegistroVeiculo().setVisible(true);
                this.dispose();
            }
        }
    }

    preencherTabela(resultados);
}
```

Método: configurarCampoMarca()
Descrição: Esse método permite o filtro no campo de marca do veículo.

```
// Filtro: Marca --------------------------------
    private void configurarCampoMarca() {
        configurarDropdownMarca(jTFEntrada);

        jTFEntrada.addKeyListener(new java.awt.event.KeyAdapter() {
            @Override
            public void keyReleased(java.awt.event.KeyEvent evt) {
                String texto = jTFEntrada.getText().trim();
                if (!texto.isEmpty()) {
                    pesquisarPorMarca(texto);
                } else {
                    limparTabela();
                }
            }
        });
    }
```

Método: pesquisarPorMarca()
Descrição: Esse método permite que o usuário consiga realizar pesquisa por marca.

```
private void pesquisarPorMarca(String marca) {
    ArrayList<Veiculo> todosVeiculos = util.ArquivoTXT_Veiculo.LerArquivo();
    ArrayList<Veiculo> resultados = new ArrayList<>();

    for (Veiculo v : todosVeiculos) {
        if (v.getMarca().toLowerCase().contains(marca.toLowerCase())) {
            resultados.add(v);
        }
    }

    if (resultados.isEmpty()) {
        boolean marcaExiste = marcasModelos.containsKey(marca);

        if (marcaExiste) {
            JOptionPane.showMessageDialog(this,
                "Marca '" + marca + "' existe na lista, mas não há veículos cadastrados.",
                "Sem resultados",
                JOptionPane.INFORMATION_MESSAGE
            );
        } else {
            int resposta = JOptionPane.showConfirmDialog(this,
                "Marca não encontrada. Pode estar digitada incorretamente.\n" +
                "Deseja registrar um novo veículo?",
                "Marca não cadastrada",
                JOptionPane.YES_NO_OPTION,
                JOptionPane.QUESTION_MESSAGE
            );

            if (resposta == JOptionPane.YES_OPTION) {
                new TelaRegistroVeiculo().setVisible(true);
                this.dispose();
            }
        }
    }

    preencherTabela(resultados);
}
```

Método: configurarCampoAno()
Descrição: Esse método permite o uso de filtro no campo do ano do veículo.

```
// Filtro: Ano de Fabricação ----------------------------
private void configurarCampoAno() {
    jTFEntrada.addKeyListener(new java.awt.event.KeyAdapter() {
        @Override
        public void keyTyped(java.awt.event.KeyEvent evt) {
            char c = evt.getKeyChar();
            if (!Character.isDigit(c) && c != java.awt.event.KeyEvent.VK_BACK_SPACE) {
                evt.consume();
            }
        }

        @Override
        public void keyReleased(java.awt.event.KeyEvent evt) {
            String texto = jTFEntrada.getText().trim();
            if (!texto.isEmpty()) {
                pesquisarPorAno(texto);
            } else {
                limparTabela();
            }
        }
    });
}
```

Método: pesquisarPorAno()
Descrição: Esse método permite ao usuário pesquisa por ano.

```
private void pesquisarPorAno(String anoTexto) {
    try {
        int ano = Integer.parseInt(anoTexto);
        ArrayList<Veiculo> todosVeiculos = util.ArquivoTXT_Veiculo.LerArquivo();
        ArrayList<Veiculo> resultados = new ArrayList<>();

        for (Veiculo v : todosVeiculos) {
            if (String.valueOf(v.getanoFabricacao()).contains(anoTexto)) {
                resultados.add(v);
            }
        }

        preencherTabela(resultados);

    } catch (NumberFormatException e) {
        limparTabela();
    }
}
```

Método: pesquisarDisponiveis()
Descrição: Esse método permite que o usuário realize pesquisa sobre veículos disponíveis.

```
   // Filtro: Disponibilidade ---------------------------
   private void pesquisarDisponiveis(boolean disponivel) {
      ArrayList<Veiculo> todosVeiculos = util.ArquivoTXT_Veiculo.LerArquivo();
      ArrayList<Veiculo> resultados = new ArrayList<>();

      for (Veiculo v : todosVeiculos) {
         if (v.isStatus() == disponivel) {
            resultados.add(v);
         }
      }

      preencherTabela(resultados);

      if (resultados.isEmpty()) {
         JOptionPane.showMessageDialog(this,
            "Nenhum veículo " + (disponivel ? "disponível" : "indisponível") + " encontrado.",
            "Sem resultados",
            JOptionPane.INFORMATION_MESSAGE
         );
      }
   }
```

Método: configurarDropdownSugestao()
Descrição: Esse método permite que o usuário consiga navegar no DropDown usando as teclas de direção e a tecla de ENTER como confirmação.

```
   // Dropdown (Modelo e Marca)
   private void configurarDropdownSugestao(javax.swing.JTextField campo, boolean
isMarca) {
      javax.swing.JPopupMenu dropdown = new javax.swing.JPopupMenu();
      dropdown.setFocusable(false);

      final java.util.List<javax.swing.JMenuItem> menuItems = new java.util.ArrayList<>();
      final int[] selectedIndex = {-1};
      final boolean[] enterPressed = {false};

      final Color COR_NORMAL = Color.WHITE;
      final Color COR_DESTAQUE = new Color(230, 240, 255);

      campo.addKeyListener(new java.awt.event.KeyAdapter() {
         @Override
         public void keyPressed(java.awt.event.KeyEvent evt) {
            if (dropdown.isVisible()) {
               switch (evt.getKeyCode()) {
```

```
            case java.awt.event.KeyEvent.VK_DOWN:
                evt.consume();
                if (selectedIndex[0] < menuItems.size() - 1) {
                    if (selectedIndex[0] >= 0) {
                        menuItems.get(selectedIndex[0]).setBackground(COR_NORMAL);
                    }
                    selectedIndex[0]++;
                    menuItems.get(selectedIndex[0]).setBackground(COR_DESTAQUE);
                } else if (selectedIndex[0] == 0) {
                    menuItems.get(selectedIndex[0]).setBackground(COR_NORMAL);
                    selectedIndex[0] = -1;
                }
                break;
            case java.awt.event.KeyEvent.VK_ENTER:
                evt.consume();
                enterPressed[0] = true;
                if (selectedIndex[0] >= 0 && selectedIndex[0] < menuItems.size()) {
                    menuItems.get(selectedIndex[0]).doClick();
                }
                dropdown.setVisible(false);
                menuItems.clear();
                selectedIndex[0] = -1;
                break;
            case java.awt.event.KeyEvent.VK_ESCAPE:
                evt.consume();
                dropdown.setVisible(false);
                selectedIndex[0] = -1;
                break;
        }
    }
}

@Override
public void keyReleased(java.awt.event.KeyEvent evt) {
    if (enterPressed[0]) {
        enterPressed[0] = false;
        return;
    }

    String texto = campo.getText().trim().toLowerCase();

    if (dropdown.isVisible() && (
            evt.getKeyCode() == java.awt.event.KeyEvent.VK_UP ||
            evt.getKeyCode() == java.awt.event.KeyEvent.VK_DOWN ||
            evt.getKeyCode() == java.awt.event.KeyEvent.VK_ENTER ||
            evt.getKeyCode() == java.awt.event.KeyEvent.VK_ESCAPE)) {
        return;
    }
```

```
if (texto.isEmpty() || evt.getKeyCode() == java.awt.event.KeyEvent.VK_ESCAPE) {
    dropdown.setVisible(false);
    menuItems.clear();
    selectedIndex[0] = -1;
    return;
}

java.util.List<String> sugestoes = new java.util.ArrayList<>();

// Busca apenas marcas
for (String marca : marcasModelos.keySet()) {
    if (marca.toLowerCase().startsWith(texto)) {
        sugestoes.add(marca);
    }
}

dropdown.removeAll();
menuItems.clear();
selectedIndex[0] = -1;

if (!sugestoes.isEmpty()) {
    for (int i = 0; i < sugestoes.size(); i++) {
        String sugestao = sugestoes.get(i);
        javax.swing.JMenuItem item = new javax.swing.JMenuItem(sugestao);

        item.setFocusable(false);
        item.setBackground(COR_NORMAL);
        item.setOpaque(true);

        final int indice = i;
        item.addMouseListener(new java.awt.event.MouseAdapter() {
            @Override
            public void mouseEntered(java.awt.event.MouseEvent e) {
                if (selectedIndex[0] >= 0 && selectedIndex[0] < menuItems.size()) {
                    menuItems.get(selectedIndex[0]).setBackground(COR_NORMAL);
                }
                selectedIndex[0] = indice;
                item.setBackground(COR_DESTAQUE);
            }
        });

        item.addActionListener(e -> {
            campo.setText(sugestao);
            campo.setForeground(Color.black);
            dropdown.setVisible(false);
            menuItems.clear();
            selectedIndex[0] = -1;
```

```
                campo.requestFocusInWindow();
            });

            dropdown.add(item);
            menuItems.add(item);
        }

        dropdown.show(campo, 0, campo.getHeight());
    } else {
        dropdown.setVisible(false);
    }
  }
});

campo.addFocusListener(new java.awt.event.FocusAdapter() {
    @Override
    public void focusLost(java.awt.event.FocusEvent evt) {
        javax.swing.SwingUtilities.invokeLater(() -> {
            dropdown.setVisible(false);
            menuItems.clear();
            selectedIndex[0] = -1;
        });
    }
});
}
```

Método: removerListeners()
Descrição: Esse método permite que o usuário consiga sair ao utilizar o botão tab, ele foi mais um método de ‘correção’ já que apresentava um bug nesse campo.

```
// Utilitario ------------------
private void removerListeners(javax.swing.JTextField campo) {
    for (java.awt.event.KeyListener kl : campo.getKeyListeners()) {
        campo.removeKeyListener(kl);
    }
    for (java.awt.event.FocusListener fl : campo.getFocusListeners()) {
        campo.removeFocusListener(fl);
    }
}
```

Método: limparTabela()
Descrição: Esse método permite que o usuário limpe a tabela.

```
private void limparTabela() {
    DefaultTableModel model = (DefaultTableModel) jTDadosPesquisa.getModel();
    model.setRowCount(0);
}
```

Método: preencherTabela()
Descrição: Esse método preenche a tabela tendo como referência o arquivo txt de veículos registrados

```
private void preencherTabela(ArrayList<Veiculo> veiculos) {
    DefaultTableModel model = (DefaultTableModel) jTDadosPesquisa.getModel();
    model.setRowCount(0);

    for (Veiculo v : veiculos) {
        Object[] linha = {
            v.getIdVeiculo(),
            v.getPlaca(),
            v.getMarca(),
            v.getModelo(),
            v.getanoFabricacao(),
            v.getStatusTextual()
        };
        model.addRow(linha);
    }
}
```

Método: configurarDropdownMarca()
Descrição: Esse método define as configurações visuais do dropdown, definindo cores e comportamentos de navegação

```
private void configurarDropdownMarca(javax.swing.JTextField campo) {
    javax.swing.JPopupMenu dropdown = new javax.swing.JPopupMenu();
    dropdown.setFocusable(false);

    final java.util.List<javax.swing.JMenuItem> menuItems = new java.util.ArrayList<>();
    final int[] selectedIndex = {-1};
    final boolean[] enterPressed = {false};

    final Color COR_NORMAL = Color.WHITE;
    final Color COR_DESTAQUE = new Color(230, 240, 255);

    campo.addKeyListener(new java.awt.event.KeyAdapter() {
        @Override
        public void keyPressed(java.awt.event.KeyEvent evt) {
            if (dropdown.isVisible()) {
                switch (evt.getKeyCode()) {
                    case java.awt.event.KeyEvent.VK_DOWN:
                        evt.consume();
                        if (selectedIndex[0] < menuItems.size() - 1) {
                            if (selectedIndex[0] >= 0) {
                                menuItems.get(selectedIndex[0]).setBackground(COR_NORMAL);
                            }
```

```
                    selectedIndex[0]++;
                    menuItems.get(selectedIndex[0]).setBackground(COR_DESTAQUE);
                }
                break;

                // PAREI AQUI MANO
//              case java.awt.event.KeyEvent.VK_UP:
//                  evt.consume();
//                  if (selectedIndex[0] > 0) {
//                      menuItems.get(selectedIndex[0]).setBackground(COR_NORMAL);
//                      selectedIndex[0]--;
//                      menuItems.get(selectedIndex[0]).setBackground(COR_DESTAQUE);
//
//
//                      }
//                  break;
            case java.awt.event.KeyEvent.VK_UP:
                evt.consume();
                if (selectedIndex[0] > 0) {
                    menuItems.get(selectedIndex[0]).setBackground(COR_NORMAL);
                    selectedIndex[0]--;
                    menuItems.get(selectedIndex[0]).setBackground(COR_DESTAQUE);
                } else if (selectedIndex[0] == 0) {
                    menuItems.get(selectedIndex[0]).setBackground(COR_NORMAL);
                    selectedIndex[0] = -1;
                }
                break;
            case java.awt.event.KeyEvent.VK_ENTER:
                evt.consume();
                enterPressed[0] = true;
                if (selectedIndex[0] >= 0 && selectedIndex[0] < menuItems.size()) {
                    menuItems.get(selectedIndex[0]).doClick();
                }
                dropdown.setVisible(false);
                menuItems.clear();
                selectedIndex[0] = -1;
                break;
            case java.awt.event.KeyEvent.VK_ESCAPE:
                evt.consume();
                dropdown.setVisible(false);
                selectedIndex[0] = -1;
                break;
        }
    }
}

@Override
public void keyReleased(java.awt.event.KeyEvent evt) {
```

```
if (enterPressed[0]) {
    enterPressed[0] = false;
    return;
}

String texto = campo.getText().trim().toLowerCase();

if (dropdown.isVisible() && (
        evt.getKeyCode() == java.awt.event.KeyEvent.VK_UP ||
        evt.getKeyCode() == java.awt.event.KeyEvent.VK_DOWN ||
        evt.getKeyCode() == java.awt.event.KeyEvent.VK_ENTER ||
        evt.getKeyCode() == java.awt.event.KeyEvent.VK_ESCAPE)) {
    return;
}

if (texto.isEmpty() || evt.getKeyCode() == java.awt.event.KeyEvent.VK_ESCAPE) {
    dropdown.setVisible(false);
    menuItems.clear();
    selectedIndex[0] = -1;
    return;
}

java.util.List<String> sugestoes = new java.util.ArrayList<>();

for (String marca : marcasModelos.keySet()) {
    for (String modelo : marcasModelos.get(marca)) {
        if (modelo.toLowerCase().startsWith(texto)) {
            sugestoes.add(modelo);
        }
    }
}

if (sugestoes.size() >= 10) {
    sugestoes = sugestoes.subList(0, 10);
}

dropdown.removeAll();
menuItems.clear();
selectedIndex[0] = -1;

if (!sugestoes.isEmpty()) {
    for (int i = 0; i < sugestoes.size(); i++) {
        String sugestao = sugestoes.get(i);
        javax.swing.JMenuItem item = new javax.swing.JMenuItem(sugestao);

        item.setFocusable(false);
        item.setBackground(COR_NORMAL);
        item.setOpaque(true);
```

```
                final int indice = i;
                item.addMouseListener(new java.awt.event.MouseAdapter() {
                    @Override
                    public void mouseEntered(java.awt.event.MouseEvent e) {
                        if (selectedIndex[0] >= 0 && selectedIndex[0] < menuItems.size()) {
                            menuItems.get(selectedIndex[0]).setBackground(COR_NORMAL);
                        }
                        selectedIndex[0] = indice;
                        item.setBackground(COR_DESTAQUE);
                    }
                });

                item.addActionListener(e -> {
                    campo.setText(sugestao);
                    campo.setForeground(Color.black);
                    dropdown.setVisible(false);
                    menuItems.clear();
                    selectedIndex[0] = -1;
                    campo.requestFocusInWindow();
                });

                dropdown.add(item);
                menuItems.add(item);
            }

            dropdown.show(campo, 0, campo.getHeight());
        } else {
            dropdown.setVisible(false);
        }
    }
});

campo.addFocusListener(new java.awt.event.FocusAdapter() {
    @Override
    public void focusLost(java.awt.event.FocusEvent evt) {
        javax.swing.SwingUtilities.invokeLater(() -> {
            dropdown.setVisible(false);
            menuItems.clear();
            selectedIndex[0] = -1;
        });
    }
});
}
```

5.2.5. TelaRegistroPrejuizo.java
Método: configurarCampoIdAutomatico()
Descrição: Esse método permite que o ID seja inserido de maneira automática.

```
private void configurarCampoIdAutomatico() {
        jTFIdMovimentacao.setEditable(false);
    }
```

Método: permitirApenasNumeros()
Descrição: Esse método impede que o usuário escreva informações que não sejam números.

```
// Permite apenas números no campo ID do veículo
    private void permitirApenasNumeros(javax.swing.JTextField campo) {
        campo.addKeyListener(new java.awt.event.KeyAdapter() {
            @Override
            public void keyTyped(java.awt.event.KeyEvent evt) {
                char c = evt.getKeyChar();
                if (!Character.isDigit(c) && c != java.awt.event.KeyEvent.VK_BACK_SPACE && c
!= java.awt.event.KeyEvent.VK_DELETE) {
                    evt.consume();
                }
            }
        });
    }
```

Método: configurarComboBoxTipoDespesa()
Descrição: Esse método cria um combobox que filtra tipos de despesas.

```
    private void configurarComboBoxTipoDespesa() {
        // Esconde o campo "Outros" inicialmente
        jTFOutros.setVisible(false);

        // Adiciona listener para detectar mudanças na seleção
        jCBidTipoDespesa.addActionListener(e -> {
            int selectedIndex = jCBidTipoDespesa.getSelectedIndex();

            // Se selecionou "Outros" (índice 14), mostra o campo
            if (selectedIndex == 14) {
                jTFOutros.setVisible(true);
                jTFOutros.setText(""); // Limpa o campo
                jTFOutros.setForeground(Color.BLACK);
                jTFOutros.requestFocus(); // Foca no campo para o usuário digitar
            } else {
                // Se selecionou qualquer outra opção, esconde o campo
                jTFOutros.setVisible(false);
                jTFOutros.setText("Outro");
```

```
            jTFOutros.setForeground(Color.GRAY);
        }
    });
}
```

Método: configurarCalendario()
Descrição: Esse método cria um calendário e puxa a data atual do sistema como hoje, e define a apresentação como DD/MM/AAAA.

```
// Configura o calendário com a data atual
private void configurarCalendario() {
    // Define a data atual no JDateChooser
    jDateChooser.setDate(new java.util.Date());

    // Define formato de exibição da data
    jDateChooser.setDateFormatString("dd/MM/yyyy");
}
```

Método: obterDataSelecionada()
Descrição: Esse método permite a seleção de data, e se não tiver nenhuma ele define como a atual.

```
// Obtém a data selecionada no calendário formatada
private String obterDataSelecionada() {
    java.util.Date dataSelecionada = jDateChooser.getDate();
    if (dataSelecionada == null) {
        // Se nenhuma data foi selecionada, usa a data atual
        dataSelecionada = new java.util.Date();
    }

    // Formata a data para dd/MM/yyyy
    java.text.SimpleDateFormat formatter = new
java.text.SimpleDateFormat("dd/MM/yyyy");
    return formatter.format(dataSelecionada);
}
```

Método: adicionarValidacoesFoco()
Descrição: Esse método permite que o usuário ao usar o comando Tab, ele puxa o foco de volta no caso em que o usuário não tenha conseguido cumprir os requisitos necessários do campo de texto.

```
// Adicionando validações quando o campo perde o foco
private void adicionarValidacoesFoco(){
    // Validação do ID do veículo
    jTFEncontrarID.addFocusListener(new java.awt.event.FocusAdapter() {
```

```
        @Override
        public void focusLost(java.awt.event.FocusEvent evt) {
            String raw = jTFEncontrarID.getText().trim();

            if (raw.isEmpty() || raw.equals("Digite o ID do veículo")) {
                return;
            }

            if (!raw.matches("\\d+")) {
                JOptionPane.showMessageDialog(TelaRegistroPrejuizo.this,
                        "O ID deve conter apenas números",
                        "Erro de Validação",
                        JOptionPane.ERROR_MESSAGE);
                jTFEncontrarID.requestFocus();
                return;
            }

            try {
                int idVeiculo = Integer.parseInt(raw);
                java.util.ArrayList<Veiculo> listaVeiculos =
util.ArquivoTXT_Veiculo.LerArquivo();
                boolean encontrado = false;

                for (Veiculo v : listaVeiculos) {
                    if (v.getIdVeiculo() == idVeiculo) {
                        encontrado = true;
                        break;
                    }
                }

                if (!encontrado) {
                    JOptionPane.showMessageDialog(TelaRegistroPrejuizo.this,
                            "Veículo com ID " + idVeiculo + " não encontrado!\n" +
                            "Por favor, verifique o ID e tente novamente.",
                            "Veículo Não Encontrado",
                            JOptionPane.WARNING_MESSAGE);
                    jTFEncontrarID.requestFocus();
                    return;
                }

                int proximoId = gerarProximoIdMovimento();
                jTFIdMovimentacao.setText(String.valueOf(proximoId));
                jTFIdMovimentacao.setForeground(Color.BLACK);

            } catch (Exception e) {
                System.err.println("Erro ao verificar veículo: " + e.getMessage());
            }
        }
```

```
    });



    // Validação do valor
    jTFRRebeValor.addFocusListener(new java.awt.event.FocusAdapter() {
        @Override
        public void focusLost(java.awt.event.FocusEvent evt) {
            String raw = jTFRRebeValor.getText().trim();

            if (raw.isEmpty() || raw.equals("R$: ")) {
                return;
            }

            // Remove caracteres não numéricos exceto vírgula e ponto
            String valorLimpo = raw.replace(",", ".");

            try {
                double valor = Double.parseDouble(valorLimpo);
                if (valor < 0) {
                    JOptionPane.showMessageDialog(TelaRegistroPrejuizo.this,
                            "O valor não pode ser negativo.",
                            "Erro de Validação",
                            JOptionPane.ERROR_MESSAGE);
                    jTFRRebeValor.requestFocus();
                    return;
                }
                // Formata o valor para exibição
                jTFRRebeValor.setText(String.format("%.2f", valor));
                jTFRRebeValor.setForeground(Color.BLACK);
            } catch (NumberFormatException e) {
                JOptionPane.showMessageDialog(TelaRegistroPrejuizo.this,
                        "Valor inválido. Use apenas números.",
                        "Erro de Validação",
                        JOptionPane.ERROR_MESSAGE);
                jTFRRebeValor.requestFocus();
            }
        }
    });
}
```

Método: gerarProximoIdMovimento()
Descrição: Esse método realiza a leitura do último ID registrado e adicionar mais um em sequência.

```
// Gera o próximo ID de movimentação
private int gerarProximoIdMovimento() {
    java.util.ArrayList<Movimento> lista = util.ArquivoTXT_Movimento.lerArquivo();

    // Se a lista está vazia, começa com 10000
    if (lista.isEmpty()) {
        return 10000;
    }

    int maiorId = 10000; // ID mínimo
    for (Movimento m : lista) {
        if (m.getIdMovimento() > maiorId) {
            maiorId = m.getIdMovimento();
        }
    }
    return maiorId + 1;
}
```

Método: limparCampos()
Descrição: Esse método autoriza a ação de limpar campos após realizar o registro.

```
// Limpa todos os campos do formulário
private void limparCampos() {
    jTFEncontrarID.setText("Digite o ID do veículo");
    jTFEncontrarID.setForeground(Color.GRAY);

    jTFIdMovimentacao.setText("Gerado automaticamente");
    jTFIdMovimentacao.setForeground(Color.GRAY);

    jDateChooser.setDate(new java.util.Date());

    jTFRRebeValor.setText("R$: ");
    jTFRRebeValor.setForeground(Color.GRAY);

    jCBidTipoDespesa.setSelectedIndex(0);

    jTFOutros.setText("Outro");
    jTFOutros.setForeground(Color.GRAY);
    jTFOutros.setVisible(false);

    jTADescricao.setText("");

    jTFEncontrarID.requestFocus();
```

```
}
```

Método: obterIdTipoDespesa()
Descrição: Esse método define a categoria do tipo de despesa se baseando na escolha que o usuário fez no ComboBox.

```
// Converte o item selecionado para ID de tipo de despesa
private int obterIdTipoDespesa() {
    int selectedIndex = jCBidTipoDespesa.getSelectedIndex();

    if (selectedIndex == 0) {
        return -1; // Inválido
    }

    // Converte índice para ID: índice 1 = ID 101, índice 2 = ID 102, etc.
    return 100 + selectedIndex;
}
```

Método: obterTipoDespesaSelecionado()
Descrição: Esse método define o valor numérico baseado na seleção do combobox realizada pelo usuário.

```
// Obtém o tipo de despesa selecionado
private int obterTipoDespesaSelecionado() {
    int selectedIndex = jCBidTipoDespesa.getSelectedIndex();

    if (selectedIndex == 0) {
        JOptionPane.showMessageDialog(this,
                "Por favor, selecione o tipo de despesa.",
                "Campo Obrigatório",
                JOptionPane.WARNING_MESSAGE);
        jCBidTipoDespesa.requestFocus();
        return -1;
    }

    return selectedIndex; // Retorna o índice que representa o tipo
}
```

Método: obterTipoDespesaTexto()
Descrição: Esse método recebe a versão textual da categoria de despesa, considerando o valor numérico encontrado nos métodos anteriores

```
// Obtém o tipo de despesa ou o texto customizado
private String obterTipoDespesaTexto() {
    int selectedIndex = jCBidTipoDespesa.getSelectedIndex();

    if (selectedIndex == 0) {
        JOptionPane.showMessageDialog(this,
                "Por favor, selecione o tipo de despesa.",
                "Campo Obrigatório",
                JOptionPane.WARNING_MESSAGE);
        jCBidTipoDespesa.requestFocus();
        return null;
    }

    // Se selecionou "Outros", retorna o texto digitado pelo usuário
    if (selectedIndex == 14) {
        String outrosText = jTFOutros.getText().trim();
        if (outrosText.isEmpty() || outrosText.equals("Outro")) {
            JOptionPane.showMessageDialog(this,
                    "Por favor, especifique o tipo de despesa em 'Outros'.",
                    "Campo Obrigatório",
                    JOptionPane.WARNING_MESSAGE);
            jTFOutros.requestFocus();
            return null;
        }
        return outrosText;
    }

    // Caso contrário retorna a opção selecionada no CB
    return (String) jCBidTipoDespesa.getSelectedItem();
}
```

Método: registrarMovimento()
Descrição: Esse método valida as informações obtidas pelo usuário, denunciando erros ou problemas caso necessário ou avisando de sucesso quando o registro for realizado nos documentos locais

```
// Método principal de registro
private void registrarMovimento() {
    try {
        // Validações básicas
        String idVeiculoText = jTFEncontrarID.getText().trim();
        String idMovimentoText = jTFIdMovimentacao.getText().trim();
        String dataText = obterDataSelecionada();
```

```
        String valorText = jTFRRebeValor.getText().trim();
        String descricaoText = jTADescricao.getText().trim();

        // Verifica ID do veículo
        if (idVeiculoText.equals("Digite o ID do veículo") || idVeiculoText.isEmpty()) {
            JOptionPane.showMessageDialog(this,
                    "Por favor, preencha o ID do veículo.",
                    "Campo Obrigatório",
                    JOptionPane.WARNING_MESSAGE);
            jTFEncontrarID.requestFocus();
            return;
        }
        // Verifica se o ID da movimentação foi gerado
        if (idMovimentoText.equals("Gerado automaticamente") ||
idMovimentoText.isEmpty()) {
            JOptionPane.showMessageDialog(this,
                    "Por favor, preencha um ID de veículo válido primeiro.\n" +
                    "O ID da movimentação será gerado automaticamente.",
                    "ID Não Gerado",
                    JOptionPane.WARNING_MESSAGE);
            jTFEncontrarID.requestFocus();
            return;
        }
                // Verifica tipo de despesa
        int selectedIndex = jCBidTipoDespesa.getSelectedIndex();
        if (selectedIndex == 0) {
            JOptionPane.showMessageDialog(this,
                    "Por favor, selecione o tipo de despesa.",
                    "Campo Obrigatório",
                    JOptionPane.WARNING_MESSAGE);
            jCBidTipoDespesa.requestFocus();
        return;
    }

        int idTipoDespesa = 100 + selectedIndex;

        // Recebe o texto da despesa
        String nomeTipoDespesa = obterTipoDespesaTexto();
        if (nomeTipoDespesa == null) {
            return;
        }

        // Verifica valor
        if (valorText.equals("R$: ") || valorText.isEmpty()) {
            JOptionPane.showMessageDialog(this,
                    "Por favor, preencha o valor da despesa.",
                    "Campo Obrigatório",
                    JOptionPane.WARNING_MESSAGE);
```

```
            jTFRRebeValor.requestFocus();
            return;
        }
        // Verifica se o usuário digitou uma descrição
        if (descricaoText.isEmpty()) {
            JOptionPane.showMessageDialog(this,
                    "Por favor, descreva o motivo/necessidade desta movimentação.",
                    "Campo Obrigatório",
                    JOptionPane.WARNING_MESSAGE);
            jTADescricao.requestFocus();
            return;
        }
        // Converte os valores
        int idVeiculo = Integer.parseInt(idVeiculoText);
        int idMovimento = Integer.parseInt(idMovimentoText);
        double valor = Double.parseDouble(valorText.replace(",", "."));

        Movimento novoMovimento = new Movimento(
            idMovimento,
            idVeiculo,
            idTipoDespesa,
            valor,
            descricaoText,
            dataText
        );

        // TXTs
        util.ArquivoTXT_Movimento.salvarLinha(novoMovimento);
        util.ArquivoTXT_Despesa.sincronizarComMovimento();

        // XLSXs
        java.util.ArrayList<Movimento> listaMovimentos =
util.ArquivoTXT_Movimento.lerArquivo();
        util.ArquivoExcel_Movimento.Transf_Excel(listaMovimentos, "Movimento.xlsx");
                // Sync
        util.ArquivoExcel_Despesa.Transf_Excel(null, "Despesas.xlsx");

        JOptionPane.showMessageDialog(this,
                "Movimentação registrada com sucesso!\n" +
                "ID da movimentação: " + idMovimento + "\n" +
                "ID do veículo: " + idVeiculo + "\n" +
                "Tipo de despesa: " + nomeTipoDespesa + "\n" +
                "Data: " + dataText,
                "Sucesso",
                JOptionPane.INFORMATION_MESSAGE
        )
        limparCampos();
```

```
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(this,
                "Erro: Verifique se os campos numéricos contêm valores válidos.\n\n" +
                "Detalhes técnicos: " + e.getMessage(),
                "Formato Inválido",
                JOptionPane.ERROR_MESSAGE);
    } catch (Exception e) {
        JOptionPane.showMessageDialog(this,
                "Erro ao registrar movimentação!\n\n" +
                "Detalhes: " + e.getMessage(),
                "Erro",
                JOptionPane.ERROR_MESSAGE);
        e.printStackTrace();
    }
}
```

Método: aplicarPlaceholder()
Descrição: Esse método aplica placeholders no caso dos campos de textos estiverem vazios ou de selecionados

```
// Métodos de Placeholder ----------------------------------------------------------
private void aplicarPlaceholder(javax.swing.JTextField campo, String placeholder) {
    final Color PLACEHOLDER_COLOR = ThemeAdm.getTheme() ==
ThemeAdm.Theme.LIGHT ? Color.GRAY : new Color(150, 150, 150);

    campo.setText(placeholder);
    campo.setForeground(PLACEHOLDER_COLOR);

    campo.addFocusListener(new java.awt.event.FocusAdapter() {
        @Override
        public void focusGained(java.awt.event.FocusEvent evt) {
            if (campo.getText().equals(placeholder)) {
                campo.setText("");
                campo.setForeground(ThemeAdm.getTextColor());
            }
        }

        @Override
        public void focusLost(java.awt.event.FocusEvent evt) {
            if (campo.getText().isEmpty()) {
                campo.setText(placeholder);
                campo.setForeground(PLACEHOLDER_COLOR);
            }
        }
    });
}
```

Método: adicionarPlaceholders()

Descrição: Esse método adiciona Placeholders pré definidos no campos de textos correspondentes.

```
private void adicionarPlaceholders() {
    aplicarPlaceholder(jTFEncontrarID, "Digite o ID do veículo");
    aplicarPlaceholder(jTFIdMovimentacao, "Gerado automaticamente");
    aplicarPlaceholder(jTFRRebeValor, "R$: ");
    aplicarPlaceholder(jTFOutros, "Outro");
}
```

5.3. Tela de Início.

A classe telaInicio é a janela de boas-vindas e o menu principal do sistema GynLog, construída em Java Swing , onde ela inicializa a interface, aplica o tema visual definido pelo usuário, e serve como ponto de navegação central para as principais funcionalidades, como registrar veículos, registrar movimentações e acessar os relatórios, com cada botão redirecionando o usuário para a tela correspondente e gerenciando o encerramento seguro da aplicação.

Método: direcionarTelaRegistroMovimentacoes()
Descrição: botão que direciona o usuário para a tela de registro de movimentações.

```
private void jButtonRegistrarMovimentaçõesActionPerformed(java.awt.event.ActionEvent
evt) {
    logger.info("Redirecionando para tela de Registro de Movimentações...");
    new TelaRegistroPrejuizo().setVisible(true);
    this.dispose();
}
```

Método: direcionarTelaVerRelatorios()
Descrição: botão que direciona o usuário para a tela de geração de relatórios.

```
private void jButtonVerRelatoriosActionPerformed(java.awt.event.ActionEvent evt) {
    logger.info("Redirecionando para tela de Relatórios...");
    new telaGerarRelatorios().setVisible(true);
    this.dispose();
}
```

Método: direcionarTelaRegistrarVeiculo()
Descrição: botão que direciona o usuário para a tela de registro de veículos na frota.

```
private void jButtonRegistrarVeiculoActionPerformed(java.awt.event.ActionEvent evt) {
    logger.info("Redirecionando para tela de Registro de Veículos...");
    new TelaRegistroVeiculo().setVisible(true);
    this.dispose();
}
```

Método: fecharSistema()
Descrição: botão que exibe uma janela perguntando se o usuário realmente deseja fechar o sistema.

```
   private void jButtonFehcarSistemaActionPerformed(java.awt.event.ActionEvent evt) {
      int confirmacao =  javax.swing.JOptionPane.showConfirmDialog(this,
            "Tem certeza que deseja sair do sistema?" ,
            "Confirmação de Saída" ,
            javax.swing.JOptionPane.YES_NO_OPTION,
            javax.swing.JOptionPane.QUESTION_MESSAGE
      );

      if(confirmacao == javax.swing.JOptionPane.YES_OPTION){
         System.exit(0);
      }
   }
```

5.4. Tela Ver Relatórios

Método: gerarRelatorioDespesasVeiculo()
Descrição: botão que aciona uma janela perguntando se o usuário deseja um arquivo txt ou Excel do relatório de despesas de um veículo específico identificado por ID.

```
   private void jButtonRelatorioVeiculoActionPerformed(java.awt.event.ActionEvent evt) {

      String filtro = jTFBuscaRelatorio.getText().trim();

      if (filtro.isEmpty() || !filtro.matches("^\\d{6}$")) {
         JOptionPane.showMessageDialog(this,
               "Digite apenas o ID do Veículo (ex: 100001) com 6 dígitos para gerar o relatório geral de movimentos.",
               "Filtro Necessário",
               JOptionPane.WARNING_MESSAGE
         );
         jTFBuscaRelatorio.requestFocus();
         return;
      }

      new util.relatorioDespesasVeiculo().gerarRelatorio(
            "Movimentações Gerais do Veículo ID " + filtro,
            new String[]{"Data", "Descrição", "Tipo Despesa", "Valor (R$)"},
            filtro
      );

   }
```

Método: filtroBusca()
Descrição: campo que o usuário digita o IDVeículo, mês ou ano para filtrar e gerar o arquivo específico desejado.

```
    private void jTFBuscaRelatorioActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }
```

Método: gerarRelatorioDespesasMensalFrota()
Descrição: botão que aciona uma janela perguntando se o usuário deseja um arquivo txt ou Excel do relatório de despesas gerais da frota total específico identificado por mês e ano..

```
    private void jButtonDespesasMensaisActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:

        String filtro = jTFBuscaRelatorio.getText().trim();

        if (filtro.isEmpty() || !filtro.matches("\\d{1,2}/\\d{4}")) {
            JOptionPane.showMessageDialog(this, "Digite o Mês/Ano (ex: 11/2025) para gerar o
Relatório de Despesas Mensais da Frota.", "Filtro Necessário",
JOptionPane.WARNING_MESSAGE);
            jTFBuscaRelatorio.requestFocus();
            return;
        }

        try {
            String[] partes = filtro.split("/");
            int mes = Integer.parseInt(partes[0]);
            int ano = Integer.parseInt(partes[1]);

            if (mes < 1 || mes > 12) {
                JOptionPane.showMessageDialog(this, "O Mês deve ser um valor entre 01 e 12.",
"Erro de Data", JOptionPane.ERROR_MESSAGE);
                jTFBuscaRelatorio.requestFocus();
                return;
            }

            if (ano > 2025 || (ano == 2025 && mes > 11)) {
                JOptionPane.showMessageDialog(this,
                        "A data máxima permitida é Novembro de 2025 (11/2025).",
                        "Erro de Limite",
                        JOptionPane.ERROR_MESSAGE
                );
                jTFBuscaRelatorio.requestFocus();
                return;
            }
```

```
        } catch (NumberFormatException e) {
            return;
        }

        new util.relatorioDespesaTotalFrota().gerarRelatorio(
                "Despesa Mensal da Frota (" + filtro + ")",
                new String[]{"ID Veículo", "Placa", "Total Custo (R$)"},
                filtro
        );

    }
```

Método: gerarRelatorioGastoCombustivel()
Descrição: botão que aciona uma janela perguntando se o usuário deseja um arquivo txt ou Excel do relatório de gasto total de combustível especificado por mês.

```
    private void jButtonGastoCombustivelActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:

        String filtro = jTFBuscaRelatorio.getText().trim();

        // 1. Validação de Formato (Mês/Ano: Ex. 12/2024)
        if (filtro.isEmpty() || !filtro.matches("\\d{1,2}/\\d{4}")) {
            JOptionPane.showMessageDialog(this,
                    "Digite o Mês/Ano (ex: 11/2025) para filtrar o custo de combustível.",
                    "Filtro Necessário",
                    JOptionPane.WARNING_MESSAGE
            );
            jTFBuscaRelatorio.requestFocus();
            return;
        }

        // 2. Validação de Lógica (Mês <= 12 e Limite 11/2025)
        try {
            String[] partes = filtro.split("/");
            int mes = Integer.parseInt(partes[0]);
            int ano = Integer.parseInt(partes[1]);

            if (mes < 1 || mes > 12) {
                JOptionPane.showMessageDialog(this, "O Mês deve ser um valor entre 01 e 12.",
"Erro de Data", JOptionPane.ERROR_MESSAGE);
                jTFBuscaRelatorio.requestFocus();
                return;
            }

            // Regra de Limite: Não pode ser ano > 2025 OU (ano == 2025 E mês > 11)
            if (ano > 2025 || (ano == 2025 && mes > 11)) {
                JOptionPane.showMessageDialog(this,
```

```
                    "A data máxima permitida para este relatório é Novembro de 2025
(11/2025).",
                    "Erro de Limite",
                    JOptionPane.ERROR_MESSAGE
                );
                jTFBuscaRelatorio.requestFocus();
                return;
            }

        } catch (NumberFormatException e) {
            return;
        }

        // Chamada do Relatório com 'util.'
        new util.relatorioGastoMensalCombustivelTotalFrota().gerarRelatorio(
                "Custo Mensal de Combustível da Frota (" + filtro + ")",
                new String[]{"ID Veículo", "Data", "Custo (R$)"},
                filtro
        );

    }
```

Método: gerarVoltarTelaInicial()
Descrição: botão que retorna à tela inicial e fecha a tela Ver Relatórios.

```
    private void jButtonVoltarRelatorioActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        new TelaInicio().setVisible(true);
        dispose();
    }
```

Método: gerarRelatorioMultasVeiculo()
Descrição: botão que aciona uma janela perguntando se o usuário deseja um arquivo txt ou Excel do relatório de multas anuais de um veículo específico identificado por ID e ano..

```
    private void jButtonMultasAnuaisVeiculoActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
        String filtro = jTFBuscaRelatorio.getText().trim();

        if (filtro.isEmpty() || !filtro.matches("^\\d{6}/\\d{4}$")) {
            JOptionPane.showMessageDialog(this,
                    "Digite o ID do Veículo (6 dígitos) e o Ano (ex: 100001/2024) no campo de
busca.",
                    "Filtro Necessário",
                    JOptionPane.WARNING_MESSAGE
            );
            jTFBuscaRelatorio.requestFocus();
```

```
            return;
        }

        String[] partesFiltro = filtro.split("/");
        String idVeiculoStr = partesFiltro[0].trim();
        String anoFiltro = partesFiltro[1].trim();

        new util.relatorioTotalMultasVeiculo().gerarRelatorio(
                "Multas Anuais do Veículo " + idVeiculoStr + " (" + anoFiltro + ")",
                new String[]{"Data", "Valor (R$)", "Descrição"},
                filtro
        );

    }
```

Método: gerarRelatorioIPVATotalFrota()
Descrição: botão que aciona uma janela perguntando se o usuário deseja um arquivo txt ou Excel do relatório do total de IPVA pago por toda a frota especificado por um ano.

```
    private void jButtonTotalIPVAanualActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:

        String filtro = jTFBuscaRelatorio.getText().trim();

        if (filtro.isEmpty() || !filtro.matches("\\d{4}")) {
            JOptionPane.showMessageDialog(this,
                    "Digite o Ano (ex: 2024) para filtrar o custo de IPVA.",
                    "Filtro Necessário",
                    JOptionPane.WARNING_MESSAGE
            );
            jTFBuscaRelatorio.requestFocus();
            return;
        }

        try {
            int ano = Integer.parseInt(filtro);

            if (ano > 2024) {
                JOptionPane.showMessageDialog(this,
                        "O ano máximo permitido para relatórios de IPVA é 2024.",
                        "Erro de Limite",
                        JOptionPane.ERROR_MESSAGE
                );
                jTFBuscaRelatorio.requestFocus();
                return;
            }
        } catch (NumberFormatException e) {
            return;
```

```
    }

    new util.relatorioIPVATotalAnualFrota().gerarRelatorio(
            "IPVA Detalhado da Frota no Ano (" + filtro + ")",
            new String[]{"ID Veículo", "Valor (R$)"},
            filtro
    );

}
```

Método: gerarRelatorioVeiculosInativos()
Descrição: botão que aciona uma janela perguntando se o usuário deseja um arquivo txt ou Excel do relatório de veículos inativos da frota.

```
private void jButtonVeiculosInativosActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    jTFBuscaRelatorio.setText("");

    new relatorioVeiculosInativos().gerarRelatorio(
            "Lista de Veículos Inativos da Frota",
            new String[]{"ID Veículo", "Placa", "Marca", "Modelo", "Ano", "Status"},
            ""
    );
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    new TelaListaVeiculo().setVisible(true);
    this.dispose();
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    new TelaRegistroVeiculo().setVisible(true);
    this.dispose();
}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    new TelaRegistroPrejuizo().setVisible(true);
    this.dispose();
}

private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {
    int confirmacao =  javax.swing.JOptionPane.showConfirmDialog(this,
        "Tem certeza que deseja sair do sistema?" ,
        "Confirmação de Saída" ,
        javax.swing.JOptionPane.YES_NO_OPTION,
        javax.swing.JOptionPane.QUESTION_MESSAGE
    );
```

```
    if(confirmacao == javax.swing.JOptionPane.YES_OPTION){
        System.exit(0);
    }
}

private void jBtnTemaActionPerformed(java.awt.event.ActionEvent evt) {
    if (ThemeAdm.getTheme() == ThemeAdm.Theme.LIGHT) {
            ThemeAdm.setTheme(ThemeAdm.Theme.DARK);
        } else {
            ThemeAdm.setTheme(ThemeAdm.Theme.LIGHT);
        }
    ThemeAdm.applyTheme(this);
}

private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
    new telaGerarRelatorios().setVisible(true);
    this.dispose();
}

private void jBtnListaMovimento1ActionPerformed(java.awt.event.ActionEvent evt) {
    new TelaPesquisaVeiculo().setVisible(true);
    this.dispose();
}
```

7. Interface Gráfica

7.1 Tecnologia Utilizada

- **Framework:** Java Swing
- **Builder:** NetBeans Apache 28 IDE 28
- **Responsividade:** Ajuste automático de componentes
- **Feedback Visual:** Mensagens de sucessos, erros e avisos

9. Assinaturas

Gerente do Projeto: Prof. Gylles de Assis Furtado.
Cliente / Solicitante: GynLog (proff. Gylles de Assis Furtado, proff, Rêmulo Pereira Borges, proff. Silvio Vidal, proff. Ivone Borges Souza).
Demais Participantes: Alexsander de Almeida Nogueira, Caio Nunes de Abreu, Cassiano Nunes de Abreu, Gabriel Naoki Uto Turigoe, Wyllian Mariano Nogueira.