

# **Projeto: Sistema de Informações para Concessionária de Automóveis desenvolvido em Java.**

---

**UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE – UFRN**

**INSTITUTO METRÓPOLE DIGITAL – IMD**

**DIM0116 - LINGUAGEM DE PROGRAMAÇÃO II**

**Docente: Francimar Carlos de Macêdo**

**Alunos: André Augusto Fernandes & Edvaldo Dantas de Medeiros Júnior**

---

## **1. Introdução**

Este projeto foi desenvolvido para a disciplina de Linguagem de Programação II, como meio de avaliar os conhecimentos adquiridos pelo aluno ao longo da disciplina.

Esta etapa consiste de entrega final do projeto, na qual foi aperfeiçoado o programa, implementados todos os pré-requisitos da avaliação e desenvolvida uma interface gráfica para o mesmo.

## **2. Objetivo**

Os requisitos da avaliação (3ª Unidade) eram desenvolver uma aplicação em JAVA na qual fosse possível aplicar os conceitos abaixo:

1. Classes e Objetos;
2. Herança;
3. Classes Abstratas;
4. Interfaces;
5. Composição;
6. Polimorfismo;
7. Tratamento de Exceção;
8. Coleções;

## 9. Interface Gráfica

### 3. Proposta

Para nosso projeto, criamos um sistema de informações para concessionárias de automóveis, no qual será possível administrar os departamentos de:

- Recursos Humanos;
- Clientes;
- Estoques;
- Abastecimento e Vendas.

Tais departamentos serão dispostos em forma de menus para o usuário do sistema, em que cada menu possuirá sub-menus para realizar operações específicas para cada área, como cadastrar clientes, cadastrar funcionários, abastecer estoque, realizar vendas, entre outras.

### 4. Instruções de Compilação e Execução/Uso

#### 4.1. Compilação

Para compilar o programa Java na linha de comando, siga as etapas abaixo:

1. Certifique-se de ter o Java Development Kit (JDK) instalado no seu sistema. Você pode verificar isso digitando `java -version` no prompt de comando. Se o JDK estiver instalado corretamente, você verá a versão do Java sendo exibida. Certifique-se também que o make está instalado no seu sistema. Você pode verificar isso digitando `make -v` no prompt de comando. Se o make estiver instalado corretamente, você verá a versão do make sendo exibida. Caso o JDK ou o make não estejam instalados, você pode instalá-los através dos links abaixo:
  - [JDK](#)
  - [make](#)
2. Baixe os arquivos deste repositório para uma pasta local de sua preferência.
3. Abra um prompt de comando ou terminal no seu sistema operacional.

4. Navegue até o diretório raiz do projeto, onde se encontra o arquivo Makefile, e execute o seguinte comando: `make all`. O comando acima irá compilar os arquivos .java e gerar os arquivos .class. OBS: Caso o comando acima não funcione, tente executar o seguinte comando: `javac -d bin -cp "lib/*" src/*.java`. Após isto, se tudo correr bem, seu programa estará compilado e pronto para execução.

Os arquivos serão compilados na estrutura padrão de diretórios de um projeto Java:

```
.
└─ LP2-Trabalho-U3_V2-GUI/
    ├── bin
    ├── lib
    └─ src
```

## 4.2. Execução

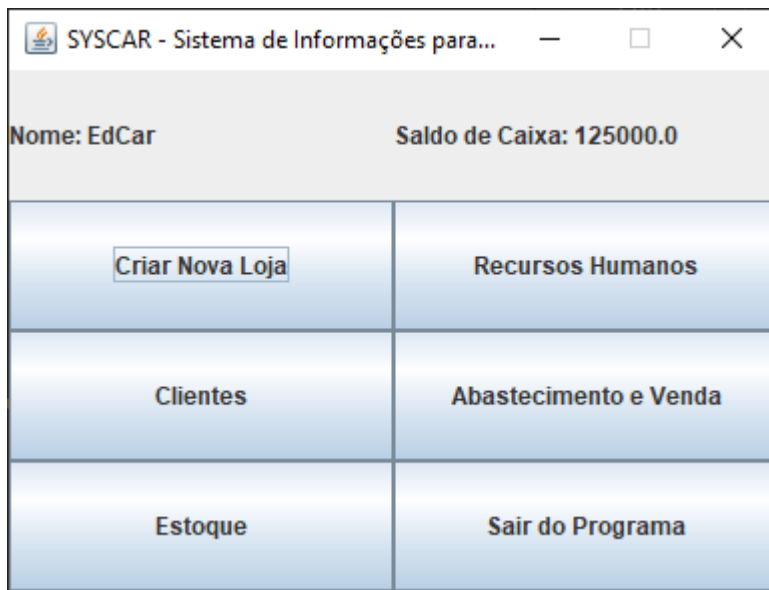
Para executar o programa, a partir do diretório raiz do projeto `LP2-Trabalho-U3_V2-GUI/` digite no terminal:

- do WINDOWS: `make run`
- do LINUX: `make runlinux`

**OBS:** Caso o comando acima não funcione, tente executar o seguinte comando:

- no Windows: `java -cp "bin;lib/*" AppGUI`
- no Linux: `java -cp "bin:lib/*" AppGUI`

O seguinte menu será exibido:



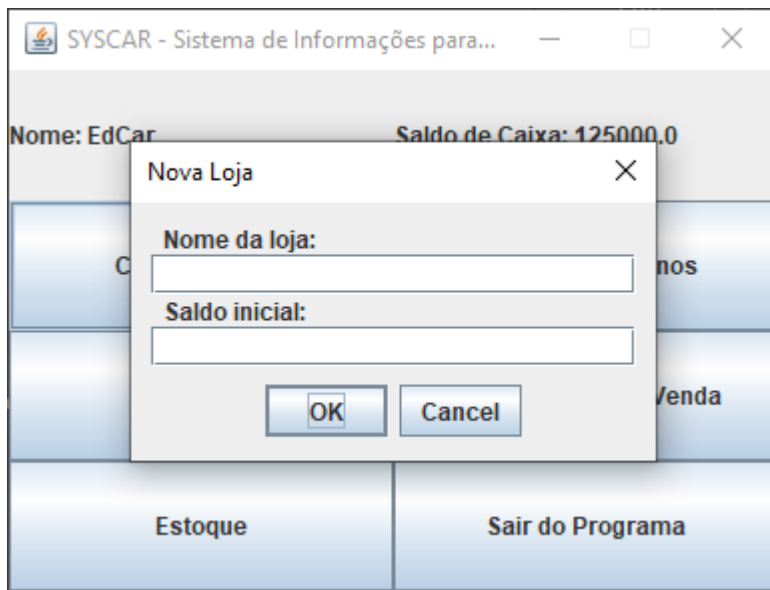
A partir do menu acima, o usuário iniciará sua jornada pelo SYSCAR - Sistema de Informações para Concessionárias de Automóveis. Aqui temos informações relativas à Loja e os botões dos menus que o usuário poderá acessar.

Ao inicializar o programa, um arquivo JSON com os dados da loja, preenchidos em um acesso anterior, será carregado. Os dados nele contidos serão utilizados para instanciar um objeto Loja, que será utilizado para gerenciar os departamentos da loja. Caso o arquivo JSON não exista, um novo objeto Loja será instanciado de forma genérica e o programa continuará normalmente.

Ao sair do sistema, o objeto Loja será convertido para formato JSON e salvo em um arquivo, para que os dados sejam preservados para acessos futuros.

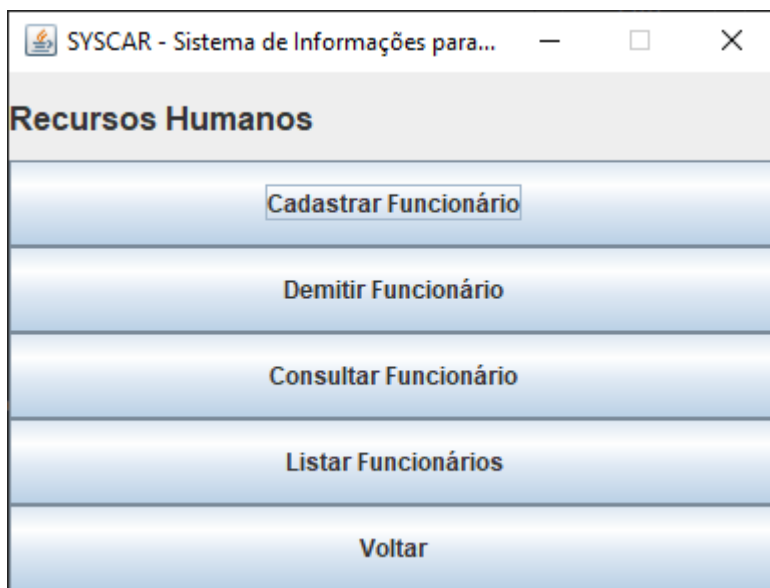
#### 4.2.0. Criar Nova Loja

Ao clicar no botão Nova Loja, o usuário poderá "setar" uma nova instância de Loja, apagando todos os dados correntes e inicializando um novo projeto.



#### 4.2.1. Recursos Humanos

Ao entrar em recursos humanos, o usuário terá acesso às seguintes operações:



Aqui o usuário poderá:

- Cadastrar Funcionário: As informações do funcionário serão passadas pelo usuário ao programa e, em seguida, serão colocadas em um objeto funcionário, para então serem inseridas em uma lista no objeto loja.
- Demitir Funcionário: A instância de funcionário cuja matrícula for fornecida pelo usuário será excluído da lista funcionários em loja.

- Consultar Funcionário: A instância de funcionário cuja matrícula for fornecida pelo usuário será exibida na tela. Caso não exista um funcionário com a matrícula fornecida, o programa exibirá uma mensagem de erro.
- Listar Funcionários: Todos os funcionários cadastrados serão exibidos na tela. Caso não exista nenhum funcionário cadastrado, o programa exibirá uma mensagem de erro.

#### 4.2.2. 2 - Clientes

Ao entrar em clientes, o usuário terá acesso às seguintes operações:

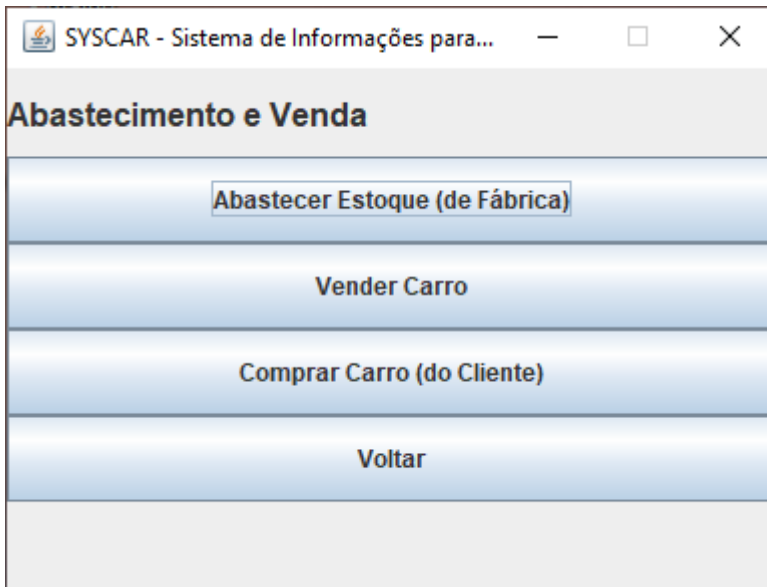


- Cadastrar Cliente: Aqui, o sistema irá solicitar ao usuário a matrícula do funcionário que está cadastrando o cliente, para que seja possível realizar a operação. O programa irá verificar a lista de funcionários da loja e, caso a matrícula fornecida não exista, o programa exibirá uma mensagem de erro. Caso a matrícula exista, o programa irá solicitar as informações do cliente e, em seguida, irá colocá-las em um objeto cliente, para então serem inseridas em uma lista no objeto loja.
- Consultar Cliente: A instância de cliente cujo cadastro for fornecido pelo usuário será exibida na tela. Caso não exista um cliente com o CPF fornecido, o programa exibirá uma mensagem de erro.
- Listar Clientes: Todos os clientes cadastrados serão exibidos na tela. Caso não exista nenhum cliente cadastrado, o programa exibirá uma mensagem de erro.

- Listar Clientes VIP: Todos os clientes cadastrados que possuírem 1 ou mais automóveis serão exibidos na tela. Caso não exista nenhum cliente cadastrado, o programa exibirá uma mensagem de alerta.

#### 4.2.3. 3 - Abastecimento e Vendas

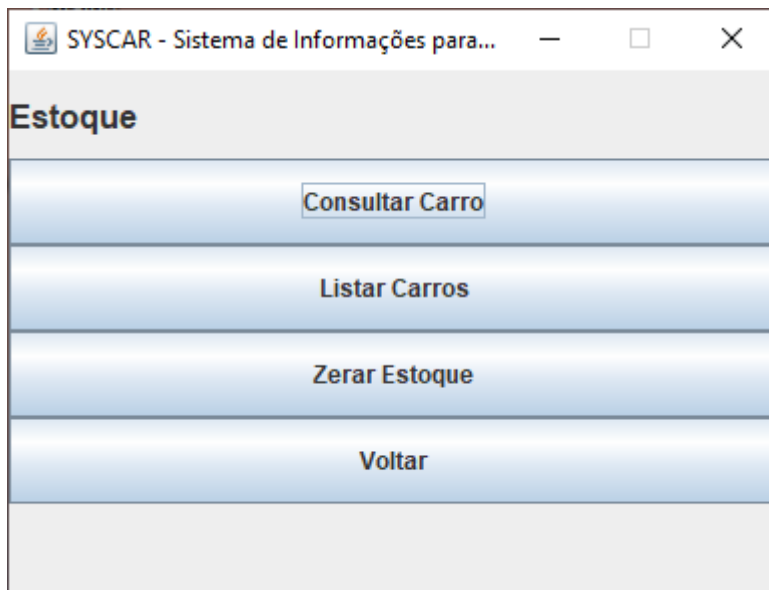
Ao entrar em abastecimento e vendas, o usuário terá acesso às seguintes operações:



- Abastecer Estoque: O usuário poderá abastecer o estoque da loja com novos automóveis. As informações do automóvel serão passadas pelo usuário ao programa e, em seguida, serão colocadas em um objeto carro, para então serem inseridas em uma lista no objeto loja.
- Vender Automóvel: O usuário poderá vender um automóvel para um cliente. O programa irá solicitar a matrícula do cliente e o chassi do automóvel. Caso o cliente não exista, o programa exibirá uma mensagem de erro. Caso o automóvel não exista, o programa exibirá uma mensagem de erro. Por fim, o programa irá exibir uma mensagem de sucesso e o automóvel será removido da lista de automóveis da loja e adicionado à lista de automóveis do cliente.
- Comprar Carro (do Cliente): O usuário poderá comprar um automóvel de um cliente. O programa irá solicitar a matrícula do cliente e o chassi do automóvel. Caso o cliente não exista, o programa exibirá uma mensagem de alerta. Caso o automóvel não exista, o programa exibirá uma mensagem de alerta. Por fim, o programa irá exibir uma mensagem de sucesso e o automóvel será removido da lista de automóveis do cliente e adicionado à lista de automóveis da loja.

#### 4.2.4. 4 - Estoque

Ao entrar em estoque, o usuário terá acesso às seguintes operações:



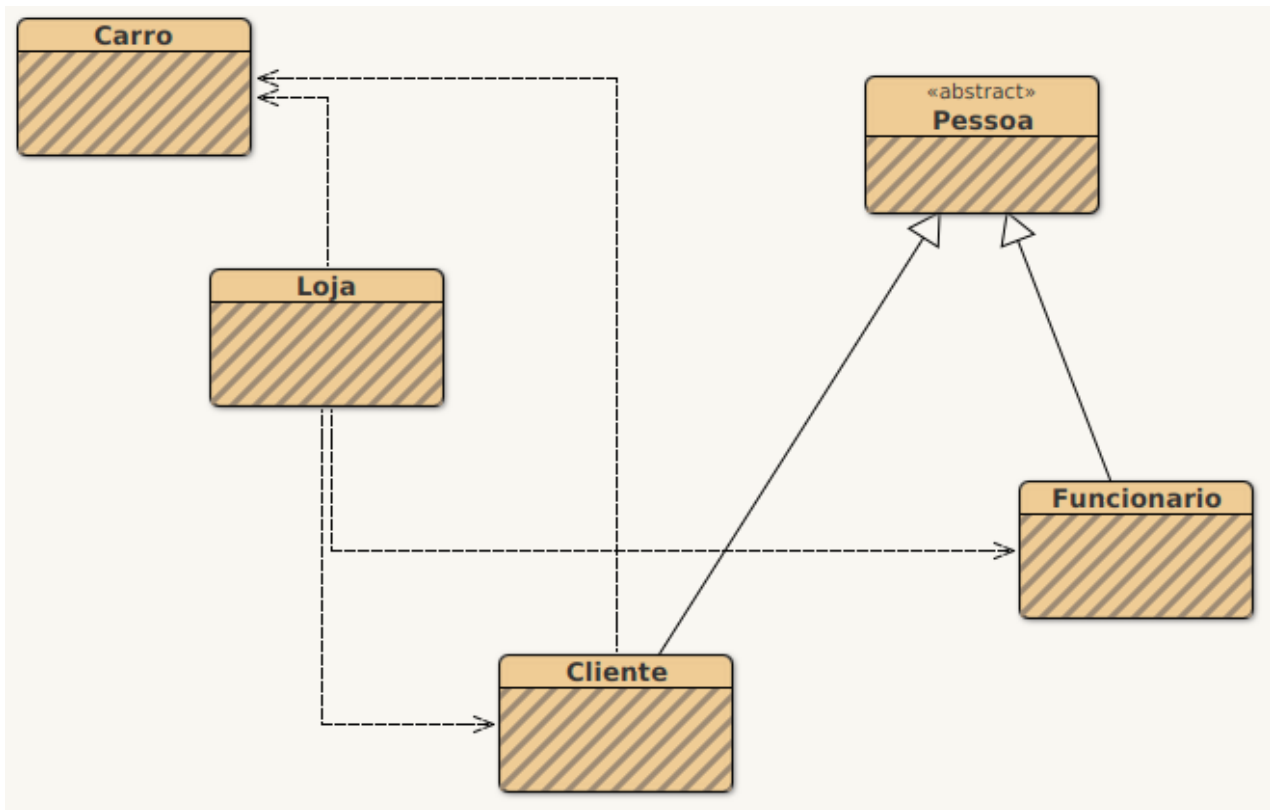
- Consultar Automóvel: O usuário poderá consultar um automóvel pelo seu modelo. Caso o automóvel não exista, o programa exibirá uma mensagem de erro. Caso o automóvel exista, o programa exibirá as informações do automóvel.
- Listar Automóveis: Todos os automóveis cadastrados serão exibidos na tela. Caso não exista nenhum automóvel cadastrado, o programa exibirá uma mensagem de erro.
- Zerar Estoque: Todos os automóveis cadastrados serão removidos da lista de automóveis da loja. Caso não exista nenhum automóvel cadastrado, o programa exibirá uma mensagem de alerta.

## 5. Implementação

### 5.1. Classes

O projeto possui a seguinte estrutura de classes:






- Pessoa: Pessoa é uma classe abstrata com atributos e métodos genéricos de uma pessoa física;
- Cliente, Funcionários: são classes herdeiras de Pessoa, com especializações próprias. Cliente possui uma coleção do tipo List<> para armazenar seus automóveis. Já funcionário, além de seus atributos especializados, possui também métodos próprios para fazer transações ordinárias da loja;
- Carro: classe que será utilizada para instanciar os automóveis da loja e dos clientes.;
- Loja: classe que possui os atributos e métodos necessários para gerenciar os departamentos da loja. Aqui se encontram as coleções de funcionários, clientes e automóveis da loja;
- AppGUI: Classe que contém o main, no qual será executado o programa.

## 5.2. Maps

Maps foi implementado na criação da coleção ClientesVIP, que contém todos os clientes da loja que possuam pelo menos um carro comprado. A indexação dos clientes foi feita o número de CPF, que é único para cada pessoa. Veja trechos deste código abaixo:



```

1  import java.util.*;
2  import java.io.*;
3  import com.google.gson.Gson;
4  import com.google.gson.GsonBuilder;
5  import javax.swing.*;
6
7  public class Loja {
8      private String nome;
9      private Double caixa;
10     private ArrayList<Carro> carros;
11     private ArrayList<Cliente> clientes;
12     private ArrayList<Funcionario> funcionarios;
13     private Map<String, Cliente> clientesVIP = new HashMap<String, Cliente>();
14
15     private transient Scanner input = new Scanner(System.in);
16
17     private transient JPanel Panel;
18     private transient JPanel mensagens;
19
20     public Loja() {
21         this.nome = "Template";
22         this.caixa = 0.0;
23         this.carros = new ArrayList<Carro>();
24         this.clientes = new ArrayList<Cliente>();
25         this.funcionarios = new ArrayList<Funcionario>();
26         this.clientesVIP = new HashMap<String, Cliente>();
27     }
28
29     public Loja(String nome, Double caixa) {
30         this.nome = nome;
31         this.caixa = caixa;
32         this.carros = new ArrayList<Carro>();
33         this.clientes = new ArrayList<Cliente>();
34         this.funcionarios = new ArrayList<Funcionario>();
35         this.clientesVIP = new HashMap<String, Cliente>();
36     }

```

### 5.3. Interface Gráfica

A implementação da interface gráfica foi feita utilizando a biblioteca Swing. Veja trechos do código abaixo:



```
1  import java.awt.*;
2  import javax.swing.*;
3  import java.util.*;
4
5  public class AppGUI extends JFrame {
6      private Loja loja;
7
8      private JPanel initialPanel;
9      private JPanel resourcesPanel;
10     private JPanel clientsPanel;
11     private JPanel inventoryPanel;
12     private JPanel salesPanel;
13
14     //private JPanel novaLojaPanel;
15
16     public AppGUI() {
17         super("SYSCAR - Sistema de Informações para Concessionárias de Automóveis");
18
19         loja = new Loja();
20
21         // Carregar objeto loja de arquivo JSON
22         try {
23             loja = loja.carregarLoja();
24         } catch (Exception e) {
25
26             System.out.println("Erro ao carregar arquivo loja.json!");
27         }
28
29         createInitialPanel();
30         createResourcesPanel();
31         createClientsPanel();
32         createInventoryPanel();
33         createSalesPanel();
34         //createNovaLojaPanel();
35
36         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
37         setSize(600, 400);
38         setLocationRelativeTo(null);
39         setResizable(false);
40         setVisible(true);
41     }
```



```
1 private void createInitialPanel() {
2     initialPanel = new JPanel();
3     initialPanel.setLayout(new GridLayout(4, 2));
4
5     JLabel nameLabel = new JLabel("Nome: " + loja.getNome());
6     JLabel caixaLabel = new JLabel("Saldo de Caixa: " + loja.getCaixa());
7
8     JButton resourcesButton = new JButton("Recursos Humanos");
9     resourcesButton.addActionListener(e -> showResourcesPanel());
10
11     JButton clientsButton = new JButton("Clientes");
12     clientsButton.addActionListener(e -> showClientsPanel());
13
14     JButton inventoryButton = new JButton("Abastecimento e Venda");
15     inventoryButton.addActionListener(e -> showInventoryPanel());
16
17     JButton salesButton = new JButton("Estoque");
18     salesButton.addActionListener(e -> showSalesPanel());
19
20     JButton exitButton = new JButton("Sair do Programa");
21     exitButton.addActionListener(e -> exitProgram());
22
23     JButton novaLojaButton = new JButton("Criar Nova Loja");
24     //novaLojaButton.addActionListener(e -> showNovaLojaPanel());
25     novaLojaButton.addActionListener(e -> {
26         loja.novaLojaGUI();
27         createInitialPanel();
28         showInitialPanel();
29     });
30
31     initialPanel.add(nameLabel);
32     initialPanel.add(caixaLabel);
33     initialPanel.add(novaLojaButton);
34     initialPanel.add(resourcesButton);
35     initialPanel.add(clientsButton);
36     initialPanel.add(inventoryButton);
37     initialPanel.add(salesButton);
38     initialPanel.add(exitButton);
39
40     add(initialPanel);
41 }
```



```
1  private void showResourcesPanel() {  
2      setContentPane(resourcesPanel);  
3      revalidate();  
4      repaint();  
5  }
```



```
1  public static void main(String[] args) {  
2      SwingUtilities.invokeLater(() -> {  
3          new AppGUI();  
4      });  
5  }
```

## 6. Tecnologias utilizadas

Para desenvolver o sistema foi utilizada a linguagem de programação Java e o editor de código VS Code.

Para compilar o programa, foi utilizado o compilador javac e o make.

Para realizar as conversões de arquivo JSON-Objeto e Objeto-JSON, foi utilizada a biblioteca GSON.

Para desenvolver a interface gráfica, foi utilizada a biblioteca Swing.

## 7. Conclusão

O projeto foi desenvolvido com sucesso, atendendo aos requisitos da avaliação. O sistema possui uma interface gráfica simples e intuitiva, que permite ao usuário navegar facilmente entre os menus e realizar as operações desejadas.