# Análise de Código Java: Sistema de Cadastro de Clientes

Análise gerada por IA (Gemini)

12 de junho de 2025

## 1 Visão Geral do Projeto

O projeto consiste em uma aplicação de desktop para o cadastro de clientes, desenvolvida em Java com a biblioteca Swing. A aplicação permite ao usuário adicionar, excluir, buscar, e salvar/carregar dados de clientes em um arquivo. A arquitetura segue uma abordagem semelhante ao padrão **Model-View-Controller (MVC)**.

- **Model**: As classes `Cliente.java` e `ClienteTableModel.java` representam o modelo de dados e a lógica de negócio.
- **View**: A classe `TelaCadastro.java` é responsável por toda a interface gráfica.
- **Controller**: A lógica de controle é implementada através de `ActionListeners` aninhados na classe `TelaCadastro.java`, que respondem às interações do usuário.

## 2 Análise dos Arquivos

### 2.1 Cliente.java (Modelo de Dados)

Esta é uma classe POJO (Plain Old Java Object) que encapsula os dados de um cliente. Sua única função é servir como um contêiner de dados.

```java
package model;

public class Cliente {

  private String nome;
  private String email;
  private String sexo;
  private String telefone;

  public Cliente(String nome, String telefone, String email, String sexo) {
    super();
    this.nome = nome;
    this.email = email;
    this.sexo = sexo;
    this.telefone = telefone;
  }

    // Getters e Setters...
}
```

Listing 1: Cliente.java

### 2.2 ClienteTableModel.java (Modelo da Tabela)

Esta classe adapta a lista de clientes (`ArrayList<Cliente>`) para ser exibida e gerenciada por um componente `JTable`. Ela herda de `AbstractTableModel` e implementa os métodos necessários para que a tabela funcione corretamente.

- `getColumnCount()`: Retorna o número de colunas.
- `getRowCount()`: Retorna o número de clientes na lista.

- `getValueAt(row, col)`: Retorna o dado específico de um cliente para preencher uma célula da tabela.
- `addCliente(Cliente c)`: Adiciona um novo cliente e notifica a tabela para se redesenhar.
- `removerCliente(int i)`: Remove um cliente e notifica a tabela.

```
package model;
// ... imports
public class ClienteTableModel extends AbstractTableModel{

  private String [] colunas = {"Nome", "Telefone", "Email", "Sexo"};
  private ArrayList<Cliente> clientes = new ArrayList<>();

  // Construtor e m todos...

  @Override
  public Object getValueAt(int rowIndex, int rowColumn) {
    Cliente cliente = clientes.get(rowIndex);
    switch(rowColumn) {
      case 0: return cliente.getNome();
      case 1: return cliente.getTelefone();
      case 2: return cliente.getEmail();
      case 3: return cliente.getSexo();
      default: return null;
    }
  }
    // ... outros m todos
}
```

Listing 2: ClienteTableModel.java

## 2.3 TelaCadastro.java (Interface e Controle)

É a classe principal que cria a janela e todos os seus componentes visuais, além de conter a lógica para tratar os eventos do usuário.

**Funcionalidades Principais:**

- **Inicialização da GUI**: Monta a janela com painéis, rótulos, campos de texto, botões e a tabela.
- **Tratamento de Eventos**: Utiliza `ActionListeners` para os botões "Salvar", "Excluir"e "Buscar".
- **Salvar Cliente**: Lê os dados dos campos de texto, cria um objeto `Cliente` e o adiciona ao modelo da tabela.
- **Excluir Cliente**: Obtém a linha selecionada na tabela e solicita a remoção ao modelo.
- **Persistência de Dados**: Oferece, através do menu "Arquivo", a opção de salvar os dados da tabela em um arquivo texto (CSV) ou carregar dados a partir de um.

### Bug Crítico Identificado

No `ActionListener` do botão `btnNewButtonSalvar`, a ordem dos argumentos passados para o construtor de `Cliente` está incorreta, o que resulta na inserção de dados nos campos errados.

```
// ... dentro do ActionListener do bot o Salvar
String nome = textFieldNome.getText().toString();
String telefone = textFieldTelefone.getText().toString();
String email = textFieldEmail.getText().toString();
String sexo = rdbtnFeminino.isSelected() ? "Feminino" : "Masculino";

// A ordem correta do construtor    (nome, telefone, email, sexo)
// A chamada abaixo est  incorreta:
Cliente cliente = new Cliente(nome, email, sexo, telefone);
modelo.addCliente(cliente);
```

Listing 3: Trecho com bug em TelaCadastro.java

**Correção Sugerida:**

```
Cliente cliente = new Cliente(nome, telefone, email, sexo);
```

## 2.4 AcaoBotao.java (Componente Inutilizado)

Esta classe implementa a interface `ActionListener`, mas seu método `actionPerformed` está vazio. Ela não é instanciada ou utilizada em nenhuma parte do projeto, sendo um código inativo.

```
package view;
// ... imports
public class AcaoBotao implements ActionListener{

  private JTextField texto;

  public AcaoBotao(JTextField text) {
    this.texto = text;
  }

  @Override
  public void actionPerformed(ActionEvent arg0) {
    // Nenhuma acao implementada.
  }
}
```

Listing 4: AcaoBotao.java