- Sistema de Validação de Ligações Suspeitas

# 1. Funcionalidade Básica

O sistema visa combater fraudes telefônicas, especialmente aquelas originadas de falsas centrais de atendimento bancário. O usuário pode consultar, denunciar e simular ligações suspeitas. O administrador possui controle total do sistema, podendo aprovar, editar ou remover denúncias, além de exportar logs e relatórios.

# 2. Qualidade do Código

## 2.1 Classes e Objetos

O sistema está dividido em camadas bem definidas:  
- model/: representa entidades como Usuario, Pessoa, Telefone  
- dao/: acesso ao banco com classes como UsuarioDAO, TelefoneDAO  
- view/: interface gráfica com Swing (TelaLogin, TelaCadastro, etc.)  
- util/: componentes auxiliares como ConexaoMySQL, Sessao, EstiloComponentes

## 2.2 Encapsulamento

Todos os atributos das classes são privados, com acesso controlado por métodos getters e setters. Isso protege os dados e mantém a integridade das operações.

## 2.3 Relacionamentos

As entidades do sistema estão relacionadas de forma coerente. Por exemplo:  
- Usuario possui um Perfil (enum: ADMIN ou USUARIO)  
- Telefone está ligado a um status (enum TipoTelefone) e uma descrição (motivo da denúncia)

## 2.4 Herança

A classe Usuario herda da classe Pessoa. Isso permite que atributos como nome e telefone fiquem centralizados, evitando repetição e facilitando manutenções futuras.

## 2.5 Polimorfismo

O sistema utiliza:  
- Sobrecarga de métodos/constructores (Telefone possui dois construtores com parâmetros diferentes)  
- Polimorfismo de execução: o menu exibido depende do tipo de usuário logado (ADMIN ou USUARIO)

## 2.6 Enum

Os enums organizam valores fixos do sistema:  
- Perfil: ADMIN, USUARIO  
- TipoTelefone: AUTORIZADO, SUSPEITO, DESCONHECIDO

# 3. Documentação README.md

## 3.1 Objetivo

Oferecer uma ferramenta antifraude que permita denúncias, consultas, simulações e relatórios sobre ligações suspeitas.

## 3.2 Instruções de Execução

1. Execute o script SQL para criar o banco de dados.  
2. Configure a classe ConexaoMySQL.java com seu usuário/senha do MySQL.  
3. Execute o projeto no NetBeans, iniciando por TelaLogin.java.  
4. Adicione o conector JDBC do MySQL ao projeto.

## 3.3 Exemplo de Uso

1. O usuário se cadastra.  
2. Consulta ou denuncia um número.  
3. O admin verifica e aprova/recusa a denúncia.  
4. O usuário simula uma ligação e recebe um alerta se o número for suspeito.  
5. O administrador exporta relatórios e logs.

## 3.4 Formatação

Cores consistentes: verde (cadastro), azul (ações), vermelho (denúncia).  
Botões e campos padronizados (300x40).  
Telas com centralização, layout limpo e mensagens padronizadas.  
  
4. Estrutura do Projeto

A estrutura segue o padrão MVC (Model-View-Controller) e está dividida em pacotes organizados:

src/

├── 📁 model/ # Classes de domínio

│ ├── Pessoa.java

│ ├── Usuario.java

│ ├── Telefone.java

│ ├── TipoTelefone.java (enum)

│ └── Perfil.java (enum)

├── 📁 dao/ # Acesso ao banco de dados

│ ├── UsuarioDAO.java

│ ├── TelefoneDAO.java

│ └── LogDAO.java

├── 📁 util/ # Utilitários auxiliares

│ ├── ConexaoMySQL.java

│ ├── Sessao.java

│ └── EstiloComponentes.java

├── 📁 view/ # Interface gráfica (Swing)

│ ├── TelaLogin.java

│ ├── TelaCadastroUsuario.java

│ ├── TelaMenuUsuario.java

│ ├── TelaPrincipal.java

│ ├── TelaConsulta.java

│ ├── TelaDenuncia.java

│ ├── TelaSimulacaoLigacao.java

│ ├── TelaContatosOficiais.java

│ ├── TelaOrientacao.java

│ ├── TelaGerenciarNumeros.java

│ ├── TelaGerenciarBloqueios.java

│ ├── TelaRelatorios.java

│ ├── TelaExportarDados.java

│ └── TelaUsuariosCadastrados.java

# 5. Comentários no Código

Os métodos importantes são documentados com comentários claros. Por exemplo:  
- inserir(): adiciona um novo número  
- consultar(): verifica o status de um número  
- registrar(): grava logs de ações

## 6. Justificativa das Decisões do Projeto

### 6.1 Padrão MVC

Separar responsabilidades em camadas (Model, View, Controller) melhora:

* Manutenção do sistema
* Legibilidade do código
* Reaproveitamento de componentes

### 6.2 Java + Swing

* Java: robusto, multiplataforma, ideal para POO
* Swing: facilita a criação de GUIs desktop com baixo custo de recursos

### 6.3 MySQL + JDBC

* MySQL é gratuito, estável e usado amplamente
* JDBC permite comunicação direta entre Java e banco de dados com segurança

### 6.4 DAO e Utilitários

* DAO separa regras de banco da lógica da aplicação
* Utilitários como ConexaoMySQL, Sessao, e EstiloComponentes reduzem duplicação de código

### 6.5 Sessão e Controle de Acesso

* A classe Sessao armazena o usuário logado
* Permite exibir menus e funcionalidades diferentes para ADMIN e USUARIO
* Garante segurança e personalização da experiência

# 7. Conclusão

Este sistema demonstra domínio de Programação Orientada a Objetos, integração com banco de dados, boas práticas de projeto (MVC, DAO), e responsabilidade na segurança da informação. É funcional, escalável e pronto para aplicação prática.