## **Sistema de Gerenciamento de Contatos**

### **Objetivo**

Desenvolver um aplicativo desktop em Java que permita aos usuários gerenciar uma lista de contatos. O sistema deverá oferecer funcionalidades para adicionar, editar, excluir, visualizar e pesquisar contatos, utilizando os diversos componentes Swing.

### **Descrição Geral**

O **Sistema de Gerenciamento de Contatos** permitirá que os usuários mantenham uma lista organizada de contatos pessoais ou profissionais. Cada contato deve conter informações como Nome, Telefone, E-mail, Endereço e Categoria.

**Requisitos Funcionais**

1. **Interface Principal:**
   * **JFrame** principal contendo:
     + **JToolBar** com botões para **Adicionar**, **Editar**, **Excluir**, **Salvar** e **Carregar** contatos.
     + **JMenuBar** com menus como **Arquivo**, **Editar**, **Ajuda**, contendo opções correspondentes às funcionalidades da toolbar e outras adicionais.
     + **JTable** para exibir a lista de contatos com colunas: Nome, Telefone, E-mail, Endereço, Categoria.
     + **JPopupMenu** acionado pelo clique direito na tabela com opções **Editar** e **Excluir**.
     + **JPanel** inferior ou lateral com **JList** ou **JComboBox** para filtrar contatos por Categoria.
2. **Funcionalidades:**
   * **Adicionar Contato:**
     + Ao clicar no botão **Adicionar** ou selecionar a opção no menu, abrir uma **JDialog** com campos para inserir Nome, Telefone, E-mail, Endereço e Categoria.
     + Utilizar **JTextField**, **JFormattedTextField** (com **MaskFormatter** para telefone), **JComboBox** para selecionar a Categoria.
     + Botões **Salvar** e **Cancelar**.
   * **Editar Contato:**
     + Permitir a edição das informações de um contato selecionado na tabela através do botão **Editar**, menu ou **JPopupMenu**.
     + Abrir um **JDialog** pré-preenchido com os dados do contato, permitindo modificações.
   * **Excluir Contato:**
     + Permitir a exclusão de um contato selecionado após confirmação através de **JOptionPane.showConfirmDialog**.
   * **Salvar e Carregar Contatos:**
     + Implementar funcionalidades para **salvar** a lista de contatos em um arquivo (pode ser texto ou JSON (gostaria muito que implementassem esse) e **carregar** contatos de um arquivo existente.
     + Utilizar **JFileChooser** para seleção de arquivos. (Funcionalidade extra)
   * **Pesquisar Contatos:**
     + Implementar uma barra de pesquisa utilizando **JTextField** e **JButton** para buscar contatos pelo Nome ou E-mail, exibindo os resultados na **JTable**.
     + Utilizar **JOptionPane.showInputDialog** para buscas rápidas, se desejado.
   * **Filtrar por Categoria:**
     + Utilizar **JComboBox** ou **JList** para permitir que o usuário filtre os contatos exibidos na tabela com base na Categoria selecionada.
   * **Validação de Dados:**
     + Implementar validações nos campos de entrada, como formatos de telefone e e-mail válidos.
     + Utilizar **JOptionPane.showMessageDialog** para exibir mensagens de erro ou confirmação.
3. **Componentes Adicionais:**
   * **JList** para exibir categorias existentes e permitir a adição de novas categorias.
   * **JLabel** e **JTextArea** para exibir detalhes adicionais de um contato selecionado.
   * **JButton** para ações secundárias, como limpar filtros ou recarregar a lista de contatos.

### **Requisitos Não-Funcionais**

* **Usabilidade:**
  + Interface intuitiva e amigável.
  + Disposição organizada dos componentes.
* **Manutenibilidade:**
  + Código bem estruturado e comentado.
  + Uso de padrões de projeto quando aplicável.
* **Persistência de Dados:**
  + Implementar a funcionalidade de salvar e carregar contatos para garantir que os dados não sejam perdidos ao fechar o aplicativo.

### **Critérios de Avaliação**

**Funcionalidade (50%)**

* + Todas as funcionalidades descritas estão implementadas e funcionando corretamente.

**Interface do Usuário (20%)**

* + Design limpo e organizado.
  + Uso apropriado dos componentes Swing.

**Qualidade do Código (20%)**

* + Código bem-organizado, com classes e métodos adequadamente separados.
  + Uso correto de nomenclaturas e convenções de código.

**Documentação e Comentários (10%)**

* + Código comentado onde necessário.

### **Entrega**

* **Data de Entrega:** **[09/10]**
* **Como Entregar:** Apresentação para a turma do funcionamento e do código implementado.

### **Dicas para Implementação**

* + Esboce a interface do usuário antes de começar a codificar.
  + Defina as classes necessárias.
  + Utilize bibliotecas como **Gson** para facilitar a serialização e desserialização, se optar por JSON.
  + Assegure-se de que os dados inseridos pelos usuários são válidos.
  + Trate exceções adequadamente para evitar que o aplicativo para de funcionar.
  + Teste todas as funcionalidades para garantir que não há bugs.
  + Verifique o comportamento do aplicativo ao salvar e carregar dados.

### **Exemplo de Fluxo de Uso**

**Iniciar o Aplicativo:**

* + A tela principal exibe a **JTable** vazia e a **JToolBar** com botões funcionais.

**Adicionar um Novo Contato:**

* + O usuário clica no botão **Adicionar**, abre-se um **JDialog** com campos para preenchimento.
  + Após preencher e salvar, o contato aparece na **JTable**.

**Editar um Contato:**

* + O usuário seleciona um contato na tabela e clica em **Editar**.
  + O **JDialog** abre com os dados do contato, permitindo modificações.

**Excluir um Contato:**

* + O usuário seleciona um contato e clica em **Excluir**, confirma a ação através de um **showConfirmDialog**.

**Salvar e Carregar Contatos:**

* + O usuário pode salvar a lista atual de contatos em um arquivo ou carregar contatos de um arquivo existente.

**Pesquisar e Filtrar:**

* + O usuário pode pesquisar contatos por Nome ou E-mail usando a barra de pesquisa.
  + Filtrar contatos por Categoria utilizando **JComboBox** ou **JList**.