

Telas da Interface e Navegação

Use uma única janela principal (`JFrame`) com um *menu bar* para navegar entre as funcionalidades ¹ . Por exemplo, crie um `JMenuBar` acoplado ao frame (`frame.setJMenuBar(menuBar)`) contendo menus como **Clientes**, **Pacotes**, **Serviços** e **Relatórios**. Em cada menu, itens de menu (`JMenuItem`) abrem telas distintas em `JPanels` (usando `CardLayout` ou trocando o conteúdo do frame). Por exemplo: no menu **Clientes**, itens “Cadastrar Cliente Nacional”, “Cadastrar Cliente Estrangeiro” e “Listar Clientes”; em **Pacotes** itens “Cadastrar Pacote”, “Listar Pacotes”; em **Serviços** itens “Cadastrar Serviço”, “Listar Serviços”; em **Relatórios** itens “Pacotes por Cliente” e “Clientes por Pacote”. Ao clicar no item de menu, carregue o painel correspondente no frame.

Uma tela de listagem pode usar um `JTable` para exibir vários registros (clientes, pacotes, etc.) em colunas e linhas ² . Como mostra a figura, cada coluna tem um título (“First Name”, “Last Name” etc.) e cada linha mostra um registro distinto. O `JTable` deve ficar dentro de um `JScrollPane` para permitir rolagem ³ . Em geral, cada tela de listagem terá:

- **Barra de busca:** um `JTextField` e um botão “Pesquisar”. Ao digitar um critério (nome, código, etc.) e pressionar Buscar, o programa consulta o serviço correspondente e atualiza o conteúdo da tabela.
- **Tabela de resultados:** use `new JTable(modelo)` onde o modelo é montado a partir dos objetos retornados do serviço. Exemplo Oracle:

```
JScrollPane scrollPane = new JScrollPane(table);
table.setFillsViewportHeight(true);
```

Isso insere a tabela no painel com cabeçalho fixo e barras de rolagem automáticas ³ .

- **Botões de ação:** abaixo ou ao lado da tabela, coloque botões como “Novo”, “Editar”, “Excluir”. Por exemplo, ao selecionar uma linha de cliente, o botão “Excluir” chama `ClienteService.delete(...)`. Use `JOptionPane.showConfirmDialog` para confirmar ações de exclusão ⁴ . Em diálogo de confirmação, pode-se usar algo como:

```
int choice = JOptionPane.showConfirmDialog(null,
    "Confirma exclusão do pacote selecionado?",
    "Confirmar Exclusão", JOptionPane.YES_NO_OPTION);
```

Se o usuário escolher Yes, executa a exclusão; senão, aborta ⁴ .

Já em telas de **cadastro/edição** (formulários), use um `JPanel` com `JLabels` e campos de entrada: `JTextField` para texto (nome, preço, e-mail etc.), `JFormattedTextField` para dados com máscara (como CPF) ou `JComboBox` / `JRadioButton` para opções. Por exemplo, na tela “Cadastrar Cliente Nacional” use:

- **CPF:** `JFormattedTextField` com máscara “###.###.###-##” para facilitar a entrada, ou um `JTextField` comum com validação manual depois.
- **Nome, Endereço, Email:** `JTextField`. Para Email, valide formato com regex ou com `InternetAddress` ⁵ .
- **Tipo de Cliente:** se for comum ao mesmo formulário, pode usar `JRadioButton` ou `JComboBox` para escolher “Nacional” ou “Estrangeiro” e habilitar campo CPF ou passaporte correspondente.

(Opcionalmente, prefira telas separadas para cada tipo para simplificar).

- **Lista de Pacotes Contratados:** opcionalmente, na tela do cliente pode-se incluir um campo de seleção múltipla (por ex. `JList` ou outro `JTable` com checkboxes) para associar pacotes ao cadastrar/editar o cliente.

Na tela “Cadastrar Pacote”:

- **Nome do Pacote, Descrição, Preço:** `TextField` para nome, `TextArea` para descrição, `FormattedTextField` ou `Spinner` para preço (somente números).

- **Serviços Incluídos:** use um componente de seleção múltipla – por exemplo, uma lista (`JList` com `setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SELECTION)`) ou uma tabela com uma coluna de checkboxes (pode ser um `JTable` com modelo que inclua uma coluna booleana). Esses componentes devem listar todos os serviços disponíveis (obtidos via `ServicoService`). Assim, ao salvar o pacote, armazena-se também a associação aos serviços selecionados.

Na tela “Cadastrar Serviço”: campos simples (nome do serviço, preço).

Em telas de **relacionamentos/consultas**: - “Pacotes por Cliente”: mostre um formulário para buscar/selecionar um cliente (ex: combo ou campo de busca) e, ao confirmar, exiba em um `JTable` todos os pacotes que ele contratou (obtidos do serviço).

- “Clientes por Pacote”: similar, selecione um pacote e mostre clientes associados.

Use `JComboBox` para campos de seleção fixa. Por exemplo, ao cadastrar um estrangeiro, pode haver um `JComboBox` para o país de origem. Como observa GeeksforGeeks, “`JComboBox` mostra um menu pop-up com uma lista e o usuário pode selecionar uma opção dessa lista”⁶. É adequado para escolhas como tipo de cliente, seleção de serviço, estado ou país, etc.

Integração com o Backend

Em cada ação da interface, invoque os serviços Java existentes para manipular os dados. Por exemplo: ao clicar em “Salvar Cliente”, colete os valores dos campos, valide-os e chame `ClienteService.inserir(cliente)`; ao clicar em “Pesquisar Cliente”, chame `ClienteService.buscaPorNome(nome)` e exiba os resultados na tabela. Para excluir, chame `ClienteService.excluir(cliente.getId())`. O mesmo vale para pacotes e serviços, usando `PacoteService` e `ServicoService`. Se o backend tiver métodos como `getAll()` ou `listarTodos()`, use-os para preencher inicialmente as tabelas. Caso contrário, adapte as chamadas (por ex. usando `findAll()` ou consultas SQL diretas).

Nas telas de associação, use métodos de serviço apropriados. Por exemplo, ao adicionar um pacote a um cliente, chame algo como `ClienteService.adicionarPacote(cliente, pacote)`; e ao incluir serviços num pacote, `PacoteService.adicionarServico(pacote, servico)`. Se tais métodos não existirem, é preciso criá-los (ver seção a seguir). Para as consultas especializadas: “pacotes contratados por um cliente” ou “clientes de um pacote”, crie métodos no serviço que retornem as listas desejadas (por ex. `List<Pacote> PacoteService.pacotesDeCliente(clienteId)`). Em suma, a lógica de negócio do backend (classes de serviço) fica oculta atrás de chamadas nos *listeners* dos botões ou seleção de menu, mantendo assim a separação MVC recomendada.

Ajustes Necessários no Backend

É provável que faltem métodos para gerenciar associações e consultas específicas. Verifique se as entidades possuem relacionamentos mapeados (por exemplo, um atributo `List<Pacote> pacotes` em `Cliente` e `List<Cliente> clientes` em `Pacote`; e semelhante para `Servico` em `Pacote`). Se o backend atual não trata isso, será preciso:

- **Modelagem de Relacionamentos:** adicione, no código Java ou no banco, as tabelas de junção (por exemplo, uma tabela `cliente_pacote` e outra `pacote_servico`) e atributos nas entidades.
- **Serviços de Associação:** implemente métodos como `void contratarPacote(int clienteId, int pacoteId)` e `void incluirServico(int pacoteId, int servicoId)` nos serviços. Eles devem atualizar as tabelas de junção.
- **Consultas Personalizadas:** crie métodos como `List<Pacote> buscarPacotesPorCliente(int clienteId)` e `List<Cliente> buscarClientesPorPacote(int pacoteId)`. No nível SQL, isso pode ser uma `JOIN` entre tabela de clientes-pacotes. No código, reflita isso nos serviços e repositórios correspondentes.
- **Restrições de Integridade:** certifique-se de que o banco não permita excluir pacotes com clientes associados (por exemplo, restrição `FOREIGN KEY ON DELETE RESTRICT`). Se o backend não lançar exceção apropriada, implemente uma verificação manual: antes de `service.excluir(pacote)`, faça uma consulta do tipo `SELECT count(*) FROM cliente_pacote WHERE pacote_id = ?`; se `>0`, informe que não pode excluir.

Em resumo, identifique no código atual *onde* faltam essas operações (possivelmente nas classes de serviço ou DAO) e crie os métodos correspondentes, seguindo o mesmo padrão dos existentes para CRUD básico.

Validações e Tratamento de Erros

Na interface, implemente validações dos campos antes de chamar o serviço:

- **CPF/Passaporte:** para cliente nacional, exija CPF no formato correto. Pode-se usar `JFormattedTextField` com `MaskFormatter("###.###.###-##")`. Além disso, aplique o algoritmo dos dígitos verificadores para garantir CPF válido. Como exemplo, o classificador `CpfCnpjValidator` citado realiza essa validação do CPF retornando `true` se válido ⁷. Para cliente estrangeiro, valide o passaporte (por exemplo, usando regex alfanumérico de 3 a 20 caracteres e não apenas zeros ⁸). Uma expressão regular sugerida é `^(?!^0+$)[a-zA-Z0-9]{3,20}$` ⁸, que impede entradas formadas só por zeros e permite dígitos/letras.
- **Email:** valide o formato do e-mail com regex ou use a classe `InternetAddress` do `JavaMail`, que “é muito mais confiável para analisar um endereço de e-mail do que uma expressão regular” ⁵. Ou seja, tente criar um `new InternetAddress(email)` e verifique se `validate()` passa.
- **Campos obrigatórios:** antes de salvar, verifique `if (campo.getText().trim().isEmpty())` e informe via `JOptionPane.showMessageDialog` que “Este campo é obrigatório”.
- **Confirmação de ações:** antes de excluir um registro, mostre `JOptionPane.showConfirmDialog` para evitar exclusões acidentais ⁴.
- **Exclusão Protegida:** não permita excluir um pacote que tenha clientes associados. Se o usuário tentar, cancele a ação e exiba uma mensagem de erro (por exemplo, `JOptionPane.showMessageDialog(null, "Não é possível excluir pacote com clientes associados.", "Erro", JOptionPane.ERROR_MESSAGE)`). Essa lógica pode ser feita via consulta prévia ou capturando exceção de integridade referencial do banco.

Em todos os casos de erro (dados inválidos, exceções do banco, etc.), use caixas de diálogo (`JOptionPane.showMessageDialog`) para notificar o usuário de forma clara e amigável.

Plano de Desenvolvimento

1. **Configurar o JFrame principal e JMenuBar:** criar a janela principal e adicionar a barra de menus (Clientes, Pacotes, Serviços, Relatórios) ¹. Implementar `ActionListener` para itens do menu que troquem o painel exibido.
2. **Tela de Cadastro de Clientes:** projetar um `JPanel` com campos (`TextField` para nome, e-mail, `FormattedTextField` para CPF, etc.) e botões Salvar/Cancelar. Diferenciar nacional/estrangeiro via seleção (radio ou aba separada) para mostrar CPF ou passaporte. No evento Salvar, executar validações (CPF/Passaporte, e-mail, campos não vazios) e então `ClienteService.inserir(cliente)`. Se falhar, mostrar mensagem de erro.
3. **Tela de Listagem/Busca de Clientes:** criar painel com `TextField` de pesquisa e um `JTable` para exibir todos os clientes (use modelo apropriado). Coloque botões "Pesquisar" (chama `ClienteService.buscaPorNome` ou similar) e "Excluir" (pega linha selecionada e chama `excluir`). Use `JOptionPane` para confirmar antes de excluir e para avisos de erro.
4. **Cadastro e Listagem de Pacotes:** similar ao cliente. Na tela de cadastro, incluir componente para associar serviços (como um `JList` de múltipla seleção com todos os serviços carregados de `ServicoService.listarTodos()`). Ao salvar, além de inserir o pacote, salve também as ligações selecionadas aos serviços (através de `PacoteService.adicionarServico`). Na listagem, crie tabela de pacotes com botões para editar/excluir; ao excluir, verifique dependências (clientes associados).
5. **Cadastro e Listagem de Serviços:** tela simples com nome e preço; tabela de serviços para exclusão/busca.
6. **Relacionamentos Cliente-Pacote e Pacote-Serviço:** se não integrados nas telas de cadastro, criar rotinas auxiliares. Por exemplo, ao editar um cliente, botão "Adicionar Pacote" abre diálogo com `JList` ou `JTable` listando pacotes disponíveis; ao confirmar, chama serviço de associação. Repetir analogamente para serviços em pacotes.
7. **Consultas Especiais:** implementar telas ou diálogos em **Relatórios**: "Pacotes por Cliente" e "Clientes por Pacote". Em cada um, permita selecionar o cliente (ou pacote) de uma lista (via `JComboBox` ou pesquisa) e exiba numa `JTable` as entidades relacionadas, obtidas por serviço (ex.: `pacotes = PacoteService.pacotesDeCliente(idCliente)`).
8. **Validar e Tratar Exceções:** após as principais telas funcionando, revisar todos os pontos de entrada de dados e operações críticas para adicionar validações pendentes (CPF, e-mail, obrigatoriedade) e tratamento de erros (ex.: falha de conexão, chave duplicada). Use mensagens claras com `JOptionPane`. Testar cenários de erro (excluir pacote associado, inserir CPF inválido, etc.) para garantir que o sistema não trave e informe corretamente o usuário.
9. **Teste de Integração:** após implementar cada módulo (Clientes, Pacotes, Serviços), testar a integração com o banco MySQL. Verificar se as operações de CRUD funcionam via GUI como esperado e se as consultas retornam dados corretos.
10. **Refinamentos Finais:** ajustar layouts para melhor usabilidade (usar `BorderLayout`, `GridLayout` ou bibliotecas de layout como MigLayout para alinhar campos), estilizar títulos de seção (`JLabel` em negrito), e adicionar ícones se desejar. Garantir que a interface seja intuitiva e consistente.

Em todo o desenvolvimento, siga esta ordem lógica: primeiro telas básicas de cadastro e listagem (clientes, pacotes, serviços), depois funcionalidades de relacionamento, por fim consultas e refinamentos. Assim, cada camada de funcionalidade ("módulo cliente", "módulo pacote", etc.) fica terminada antes de passar à próxima, facilitando testes e integração.

Referências: A API Swing suporta `JMenuBar` / `JMenu` para navegação ¹, `JTable` para tabelas de dados ², e `JComboBox` para listas suspensas ⁶. Validações podem usar regex ou classes de utilitários como `InternetAddress` para e-mail ⁵ e implementações padrão de CPF ⁷. Diálogos de confirmação/erro usam `JOptionPane` (ex.: `showConfirmDialog` e `showMessageDialog` ⁴). Esses recursos devem ser combinados conforme descrito acima para atender a todos os requisitos.

¹ How to Use Menus (The Java™ Tutorials > Creating a GUI With Swing > Using Swing Components)

<https://docs.oracle.com/javase/tutorial/uiswing/components/menu.html>

² ³ How to Use Tables (The Java™ Tutorials > Creating a GUI With Swing > Using Swing Components)

<https://docs.oracle.com/javase/tutorial/uiswing/components/table.html>

⁴ Java JOptionPane - GeeksforGeeks

<https://www.geeksforgeeks.org/java/java-joptionpane/>

⁵ java - How to validate a JTextField of email ID with a regex in swing code? - Stack Overflow

<https://stackoverflow.com/questions/15269507/how-to-validate-a-jtextfield-of-email-id-with-a-regex-in-swing-code>

⁶ Java Swing | JComboBox with examples - GeeksforGeeks

<https://www.geeksforgeeks.org/java/java-swing-jcombobox-examples/>

⁷ CpfCnpjValidator.java · GitHub

<https://gist.github.com/clairtonluz/0e82a03e8b6c148608f1>

⁸ c# - Regex for Passport Number - Stack Overflow

<https://stackoverflow.com/questions/40647728/regex-for-passport-number>