

PESSOA 2 - MÓDULOS PRINCIPAIS (CRUD)

Estrutura de Módulos

```
src/app/components/  
├── categoria/  
│   ├── categoria-list/  
│   ├── categoria-form/  
│   ├── categoria.module.ts  
│   └── categoria-routing.module.ts  
├── cliente/  
│   ├── cliente-list/  
│   ├── cliente-form/  
│   ├── cliente.module.ts  
│   └── cliente-routing.module.ts  
└── servico/  
    ├── servico-list/  
    ├── servico-form/  
    ├── servico.module.ts  
    └── servico-routing.module.ts  
  
src/app/models/  
├── categoria.model.ts  
├── cliente.model.ts  
└── servico.model.ts  
  
src/app/services/  
├── categoria.service.ts  
├── cliente.service.ts  
└── servico.service.ts
```

MÓDULO CATEGORIA

Models (/models/categoria.model.ts)

typescript

```
export interface Categoria {  
  id?: number;  
  nome: string;  
  descricao: string;  
  beneficios?: string;  
  ativo: boolean;  
  createdAt?: Date;  
  updatedAt?: Date;  
}
```

Service (/services/categoria.service.ts)

typescript

```
@Injectable()  
export class CategoriaService {  
  private baseUrl = 'http://localhost:8080/categorias';  
  
  // CRUD básico  
  listarTodas(): Observable<Categoria[]>  
  buscarPorId(id: number): Observable<Categoria>  
  criar(categoria: Categoria): Observable<Categoria>  
  atualizar(id: number, categoria: Categoria): Observable<Categoria>  
  deletar(id: number): Observable<void>  
  
  // Filtros personalizados  
  buscarAtivas(): Observable<Categoria[]>  
  buscarPorNome(nome: string): Observable<Categoria[]>  
}
```

Componentes:

1. CategoriaListComponent

- Tabela com todas as categorias
- Filtros (nome, ativo/inativo)
- Botões de ação (editar, excluir)
- Paginação

2. CategoriaFormComponent

- Formulário reativo
- Validações obrigatórias
- Modal ou rota (sua escolha)

Models (/models/cliente.model.ts)

typescript

```
export interface Cliente {  
  id?: number;  
  nome: string;  
  cpf: string;  
  email?: string;  
  telefone?: string;  
  dataNascimento: Date;  
  ativo: boolean;  
  statusCadastro?: string; // COMPLETO, INCOMPLETO  
  categoria?: Categoria;  
  enderecos?: Endereco[];  
  contratos?: Contrato[];  
}
```

Service (/services/cliente.service.ts)

typescript

```
@Injectable()  
export class ClienteService {  
  private baseUrl = 'http://localhost:8080/clientes';  
  
  // CRUD básico  
  listarTodos(): Observable<Cliente[]>  
  buscarPorId(id: number): Observable<Cliente>  
  criar(cliente: Cliente): Observable<Cliente>  
  atualizar(id: number, cliente: Cliente): Observable<Cliente>  
  deletar(id: number): Observable<void>  
  
  // Filtros personalizados  
  buscarPorNome(nome: string): Observable<Cliente[]>  
  buscarPorCategoria(categoriaId: number): Observable<Cliente[]>  
  buscarPorCpf(cpf: string): Observable<Cliente>  
}
```

Componentes:

1. ClienteListComponent

- Tabela responsiva
- Filtros: nome, categoria, CPF
- Status do cadastro (badge colorido)

- Ações: visualizar, editar, excluir

2. ClienteFormComponent

- Formulário complexo com validações
- Campo categoria (select com categorias ativas)
- Máscara para CPF e telefone
- Validação de CPF único

MÓDULO SERVIÇO

Models (`/models/servico.model.ts`)

typescript

```
export interface Servico {  
  id?: number;  
  nome: string;  
  descricao: string;  
  valor: number;  
  categoria: string; // RECARGA, FINANCEIRO, DIGITAL  
  ativo: boolean;  
  createdAt?: Date;  
  updatedAt?: Date;  
}
```

Service (`/services/servico.service.ts`)

typescript

```
@Injectable()  
export class ServicoService {  
  private baseUrl = 'http://localhost:8080/servicos';  
  
  // CRUD básico  
  listarTodos(): Observable<Servico[]>  
  buscarPorId(id: number): Observable<Servico>  
  criar(servico: Servico): Observable<Servico>  
  atualizar(id: number, servico: Servico): Observable<Servico>  
  deletar(id: number): Observable<void>  
  
  // Filtros personalizados  
  buscarAtivos(): Observable<Servico[]>  
  buscarPorNome(nome: string): Observable<Servico[]>  
  buscarPorCategoria(categoria: string): Observable<Servico[]>  
}
```

Componentes:

1. **ServicoListComponent**

- Grid de cards ou tabela
- Filtros: nome, categoria, ativo
- Exibir valor formatado (currency pipe)

2. **ServicoFormComponent**

- Categoria como select (RECARGA, FINANCEIRO, DIGITAL)
- Validação de valor positivo
- Toggle ativo/inativo

Padrões de Interface

Lista (Template padrão):

html

```

<div class="container-fluid">
  <div class="row">
    <div class="col-12">
      <div class="card">
        <div class="card-header d-flex justify-content-between">
          <h5>{{ titulo }}</h5>
          <button class="btn btn-primary" (click)="novo()">
            <i class="fas fa-plus"></i> Novo
          </button>
        </div>
        <div class="card-body">
          <!-- Filtros -->
          <div class="row mb-3">
            <div class="col-md-6">
              <input [(ngModel)]="filtroNome"
                placeholder="Buscar por nome..."
                class="form-control">
            </div>
            <div class="col-md-3">
              <button (click)="filtrar()" class="btn btn-outline-primary">
                Filtrar
              </button>
            </div>
          </div>
          </div>

          <!-- Tabela -->
          <div class="table-responsive">
            <table class="table table-hover">
              <thead>
                <tr>
                  <th>ID</th>
                  <th>Nome</th>
                  <th>Status</th>
                  <th>Ações</th>
                </tr>
              </thead>
              <tbody>
                @for (item of lista; track item.id) {
                  <tr>
                    <td>{{ item.id }}</td>
                    <td>{{ item.nome }}</td>
                    <td>
                      <span class="badge"
                        [class]="item.ativo ? 'bg-success' : 'bg-danger'">
                        {{ item.ativo ? 'Ativo' : 'Inativo' }}
                      </span>
                    </td>
                  </tr>
                }
              </tbody>
            </table>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>

```

```

        </td>
        <td>
            <button (click)="editar(item)" class="btn btn-sm btn-warning me-2">
                <i class="fas fa-edit"></i>
            </button>
            <button (click)="excluir(item.id)" class="btn btn-sm btn-danger">
                <i class="fas fa-trash"></i>
            </button>
        </td>
    </tr>
}
</tbody>
</table>
</div>
</div>
</div>
</div>
</div>
```

Formulário (Template padrão):

html

```

<div class="container">
  <div class="row justify-content-center">
    <div class="col-md-8">
      <div class="card">
        <div class="card-header">
          <h5>{{ titulo }}</h5>
        </div>
        <div class="card-body">
          <form [formGroup]="formulario" (ngSubmit)="salvar()">
            <!-- Campos do formulário -->
            <div class="row">
              <div class="col-md-6 mb-3">
                <label class="form-label">Nome *</label>
                <input formControlName="nome"
                  class="form-control"
                  [class.is-invalid]="isInvalid('nome')">
                <div class="invalid-feedback">
                  {{ getErrorMessage('nome') }}
                </div>
              </div>
            </div>
            <div class="d-flex justify-content-end">
              <button type="button" (click)="voltar()"
                class="btn btn-secondary me-2">
                Cancelar
              </button>
              <button type="submit"
                [disabled]="formulario.invalid"
                class="btn btn-primary">
                Salvar
              </button>
            </div>
          </form>
        </div>
      </div>
    </div>
  </div>
</div>

```

✓ Checklist de Entregáveis:

- ☐ Módulo Categoria completo (CRUD + filtros)
- ☐ Módulo Cliente completo (CRUD + filtros + relacionamento)
- ☐ Módulo Serviço completo (CRUD + filtros)

- ☐ Validações reativas em todos os formulários
- ☐ Tratamento de erros com SweetAlert
- ☐ Interface responsiva seguindo tema RV Digital
- ☐ Integração 100% com backend (todos os endpoints)

Pontos de Atenção:

- Usar `@for` e `@if` (Angular 19, não ngFor/ngIf)
- Implementar subscribe com next/error em todos os services
- Relacionamento Cliente-Categoria via select
- Validação de CPF único no front
- Formatação de valores monetários
- Status visual com badges coloridos