

Plano de Adaptação: CRM-ORG → ERP Igrejas

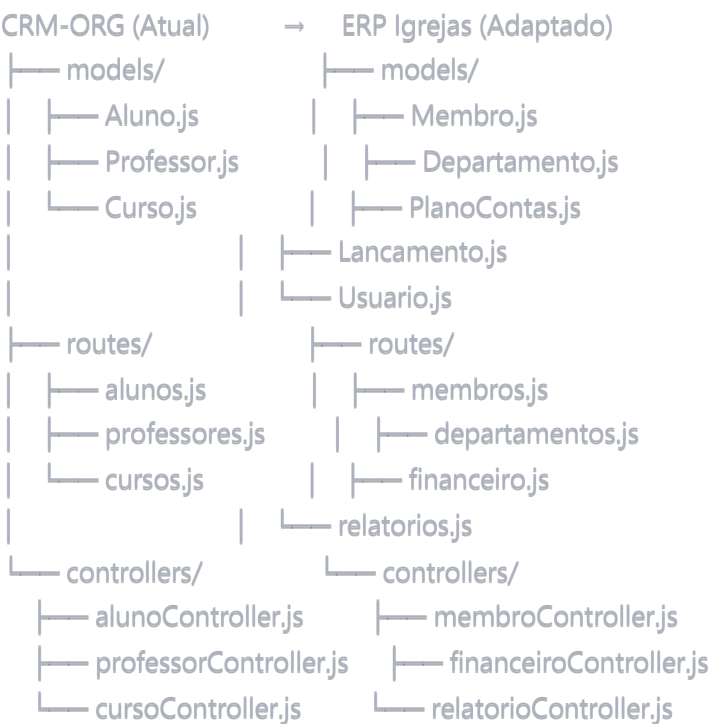
🔄 MAPEAMENTO DE ESTRUTURAS

Mantém da Base CRM-ORG

- ✅ **Arquitetura completa:** Node.js + Express + JWT + MySQL
- ✅ **Frontend:** React.js + Bootstrap (UI/UX já testada)
- ✅ **API RESTful:** Axios + JSON responses
- ✅ **Deploy:** Vercel (frontend) + Backend separado
- ✅ **Autenticação:** JWT (sistema de login/perfis)
- ✅ **Estrutura de pastas:** Organização do projeto

Adaptações Necessárias

BACKEND (Node.js + Express)



FRONTEND (React.js)



ESTRUTURA DE DADOS ADAPTADA

Modelos MySQL

1. Plano de Contas (Novo)

sql

```
CREATE TABLE plano_contas (  
  id INT AUTO_INCREMENT PRIMARY KEY,  
  codigo VARCHAR(20) UNIQUE NOT NULL,  
  nome_conta VARCHAR(255) NOT NULL,  
  tipo ENUM('D', 'C') NOT NULL,  
  centro_custo VARCHAR(50),  
  ativo BOOLEAN DEFAULT TRUE,  
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP  
);
```

2. Lançamentos (Novo)

sql

```
CREATE TABLE lancamentos (  
  id INT AUTO_INCREMENT PRIMARY KEY,  
  data_lancamento DATE NOT NULL,  
  conta_codigo VARCHAR(20) NOT NULL,  
  descricao TEXT NOT NULL,  
  valor DECIMAL(10,2) NOT NULL,  
  tipo ENUM('D', 'C') NOT NULL,  
  departamento_id INT,  
  usuario_id INT NOT NULL,  
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  FOREIGN KEY (conta_codigo) REFERENCES plano_contas(codigo),  
  FOREIGN KEY (departamento_id) REFERENCES departamentos(id),  
  FOREIGN KEY (usuario_id) REFERENCES usuarios(id)  
);
```

3. Departamentos (Adaptado de "Cursos")

sql

```
CREATE TABLE departamentos (  
  id INT AUTO_INCREMENT PRIMARY KEY,  
  nome VARCHAR(255) NOT NULL,  
  responsavel VARCHAR(255),  
  centro_custo VARCHAR(50),  
  ativo BOOLEAN DEFAULT TRUE,  
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP  
);
```

4. Membros (Adaptado de "Alunos")

sql

```
CREATE TABLE membros (  
  id INT AUTO_INCREMENT PRIMARY KEY,  
  nome VARCHAR(255) NOT NULL,  
  email VARCHAR(255),  
  telefone VARCHAR(20),  
  endereco TEXT,  
  data_nascimento DATE,  
  departamento_id INT,  
  ativo BOOLEAN DEFAULT TRUE,  
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
  FOREIGN KEY (departamento_id) REFERENCES departamentos(id)  
);
```

5. Usuários (Mantém estrutura)

sql

```
CREATE TABLE usuarios (  
  id INT AUTO_INCREMENT PRIMARY KEY,  
  nome VARCHAR(255) NOT NULL,  
  email VARCHAR(255) UNIQUE NOT NULL,  
  senha VARCHAR(255) NOT NULL,  
  perfil ENUM('admin', 'financeiro', 'departamento') DEFAULT 'departamento',  
  departamento_id INT,  
  ativo BOOLEAN DEFAULT TRUE,  
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  FOREIGN KEY (departamento_id) REFERENCES departamentos(id)  
);
```

Semana 1: Preparação

- ☐ Fork/Clone do CRM-ORG
- ☐ Análise detalhada da estrutura atual
- ☐ Planejamento das modificações
- ☐ Setup do ambiente de desenvolvimento

Semana 2: Backend - Modelos

- ☐ Adaptar models existentes
- ☐ Criar novos models (PlanoContas, Lancamentos)
- ☐ Migração/Adaptação do banco MySQL
- ☐ Testes unitários dos models

Semana 3: Backend - APIs

- ☐ Adaptar routes existentes
- ☐ Criar novas routes (financeiro, relatórios)
- ☐ Implementar controllers
- ☐ Validações e middleware

Semana 4: Frontend - Componentes

- ☐ Adaptar componentes existentes
- ☐ Criar novos componentes (Financeiro, Relatórios)
- ☐ Manter o design/UI Bootstrap
- ☐ Testes de interface

Semana 5: Integração

- ☐ Conectar frontend adaptado com backend
- ☐ Testes de integração
- ☐ Ajustes de bugs
- ☐ Documentação das mudanças

Semana 6: Finalização

- ☐ Deploy e testes finais
- ☐ Documentação completa
- ☐ Treinamento/Manual
- ☐ Validação com usuário final



Acelera o Desenvolvimento

- **80% do código reutilizado:** Estrutura, autenticação, UI
- **Reduz tempo:** 6 semanas vs 8 semanas do zero
- **Menor risco:** Base já testada e funcional

Mantém Qualidade

- **UI/UX conhecida:** Bootstrap já aplicado
- **Padrões estabelecidos:** Estrutura de pastas, naming
- **Deploy testado:** Vercel + Backend separado

Facilita Manutenção

- **Código familiar:** Você já conhece a estrutura
 - **Padrões consistentes:** Mesmo estilo de código
 - **Documentação:** Base para futuras expansões
-



FUNCIONALIDADES RESULTANTES

Módulo Financeiro

- ☒ Plano de Contas (baseado no CRUD de Cursos)
- ☒ Lançamentos (receitas/despesas)
- ☒ Contas a pagar/receber
- ☒ Dashboard financeiro

Módulo Relatórios




- ☒ Relatórios sintéticos/analíticos
- ☒ Gráficos (adaptação do dashboard atual)
- ☒ Relatórios por departamento
- ☒ Exportação (PDF/Excel)

Módulo Departamentos

- ☒ Gestão de departamentos (adaptação de Cursos)
- ☒ Controle de acesso
- ☒ Centro de custos
- ☒ Responsáveis

Módulo Membros

- ☒ Cadastro completo (adaptação de Alunos)

-  Vinculação com departamentos
 -  Histórico de participação
 -  Relatórios de membros
-

ESTIMATIVA REVISADA

Cronograma

- **Desenvolvimento:** 6 semanas (vs 8 do zero)
- **Redução:** 25% do tempo
- **Risco:** Baixo (base testada)

Custos

- **Desenvolvimento:** R\$ 18.000 (vs R\$ 25.000)
- **Economia:** R\$ 7.000
- **ROI:** Maior retorno mais rápido

Qualidade

- **UI/UX:** Mantém padrão já aprovado
 - **Performance:** Base otimizada
 - **Manutenção:** Estrutura conhecida
-

PRÓXIMOS PASSOS

1. **Análise detalhada** dos repositórios CRM-ORG
2. **Mapeamento específico** de cada arquivo/função
3. **Criação do projeto** ERP Igrejas baseado no CRM
4. **Adaptação gradual** seguindo o cronograma
5. **Testes** e validação contínua

Resultado: ERP para Igrejas robusto, rápido e econômico, aproveitando toda a base sólida do seu CRM-ORG!