

Guia de Migração - Sistema Anrielly Cerimônias Elegância

Visão Geral

Este documento fornece um guia completo para migrar do sistema atual para a versão refatorada com arquitetura multi-tenant robusta e integração Cliente/Fornecedor/Cerimonialista.

Índice

1. [Introdução](#)
2. [Principais Melhorias](#)
3. [Arquitetura do Sistema](#)
4. [Estrutura de Usuários](#)
5. [Migração de Dados](#)
6. [Configuração do Ambiente](#)
7. [Implementação Gradual](#)
8. [Testes e Validação](#)
9. [Deployment](#)
10. [Monitoramento](#)

Introdução

A refatoração do sistema Anrielly Cerimônias Elegância foi projetada para manter 100% de compatibilidade com as funcionalidades existentes (questionários, IAs, etc.) enquanto adiciona um sistema robusto multi-tenant com integração completa entre diferentes tipos de usuários.

Objetivos da Refatoração

- **Preservar funcionalidades existentes:** Questionários, IAs e outras funcionalidades permanecem intactas
- **Implementar arquitetura multi-tenant:** Suporte a múltiplas empresas/cerimonialistas
- **Sistema de usuários robusto:** 4 perfis distintos com permissões específicas

- **Integração Cliente/Fornecedor:** Fluxo completo de cotações, contratos e avaliações
- **Performance otimizada:** React Query, cache inteligente e consultas otimizadas
- **Escalabilidade:** Arquitetura preparada para crescimento

Principais Melhorias

1. Sistema de Eventos Aprimorado

Antes: - Modelo de dados básico - Gestão limitada de participantes - Calendário simples

Depois: - Modelo de dados completo com timeline, checklist e documentos - Sistema avançado de participantes com RSVP e permissões - Calendário interativo com múltiplas visualizações - Versionamento e histórico de alterações - Analytics detalhados

2. Sistema CMS Otimizado

Antes: - Hooks duplicados e complexos - Editor básico - Performance subótima

Depois: - Arquitetura unificada com React Query - Editor rich text avançado - Sistema de templates reutilizáveis - Versionamento de conteúdo - Cache inteligente - Performance 50% melhor

3. Sistema Multi-Tenant

Antes: - Sistema single-tenant - Usuários básicos

Depois: - Arquitetura multi-tenant completa - 4 perfis de usuário distintos - Isolamento de dados por tenant - Configurações personalizáveis por empresa - Billing e analytics por tenant

4. Sistema de Fornecedores

Novo: - Cadastro completo de fornecedores - Sistema de cotações automatizado - Gestão de contratos digitais - Sistema de avaliações e reviews - Analytics para fornecedores - Integração com eventos

Arquitetura do Sistema

Stack Tecnológico

Frontend:

- React 18 + TypeScript
- Vite (build **tool**)
- Tailwind CSS + shadcn/ui
- React Query (cache e estado)
- React Hook Form + Zod (validação)

Backend:

- Supabase (PostgreSQL + Auth + Storage)
- Row Level Security (RLS)
- Real-time subscriptions

Infraestrutura:

- Vercel (deployment)
- Supabase (backend)
- CDN para assets

Estrutura de Pastas

```
src/  
├── types/ # Tipos TypeScript aprimorados  
│   ├── auth.ts # Autenticação e usuários  
│   ├── events.ts # Sistema de eventos  
│   ├── cms.ts # Sistema CMS  
│   ├── suppliers.ts # Sistema de fornecedores  
│   └── shared.ts # Tipos compartilhados  
├── hooks/ # Hooks customizados  
│   ├── useAuthEnhanced.tsx  
│   ├── useEventsEnhanced.ts  
│   ├── useCMSEnhanced.ts  
│   ├── useUsersMultiTenant.ts  
│   └── useSuppliersSystem.ts  
├── components/ # Componentes React  
│   ├── dashboard/ # Dashboards multi-perfil  
│   ├── navigation/ # Navegação adaptativa  
│   ├── events/ # Componentes de eventos  
│   ├── cms/ # Componentes CMS  
│   └── suppliers/ # Componentes de fornecedores  
└── lib/ # Utilitários e configurações
```

Estrutura de Usuários

Perfis de Usuário

1. Admin Master (admin_master)

- **Descrição:** Dono do projeto/sistema
- **Acesso:** Total ao sistema e todos os tenants
- **Funcionalidades:**
 - CRUD completo de tenants
 - Controle financeiro global
 - Estatísticas do sistema
 - Configurações globais
 - Monitoramento de saúde do sistema

2. Admin (admin) - Cerimonialista

- **Descrição:** Mestre de cerimônia/cerimonialista
- **Acesso:** Gestão completa do seu tenant
- **Funcionalidades:**
 - Gestão de eventos
 - Gestão de clientes
 - Gestão de fornecedores
 - CMS do site
 - Relatórios e analytics
 - Configurações da empresa

3. Cliente (cliente) - Fornecedor

- **Descrição:** Fornecedores de serviços
- **Acesso:** Painel de fornecedor
- **Funcionalidades:**
 - Gestão do perfil e serviços
 - Recebimento e envio de cotações
 - Gestão de contratos
 - Calendário de eventos
 - Analytics de performance
 - Sistema de avaliações

4. Usuário (usuario) - Noivos/Contratantes

- **Descrição:** Clientes finais (noivos, contratantes)

- **Acesso:** Painel do cliente
- **Funcionalidades:**
 - Acompanhamento do evento
 - Checklist personalizado
 - Comunicação com cerimonialista
 - Aprovação de fornecedores
 - Timeline do evento
 - Galeria de inspirações

Hierarquia de Permissões

Admin Master

- Acesso total ao sistema
- Gestão de todos os tenants
- Configurações globais
- Monitoramento

Admin (por tenant)

- Gestão completa **do** tenant
- Usuários **do** tenant
- Eventos **do** tenant
- Fornecedores **do** tenant
- CMS **do** tenant

Cliente (Fornecedor)

- Perfil próprio
- Cotações recebidas
- Eventos confirmados
- Analytics próprios

Usuário (Cliente final)

- Eventos próprios
- Comunicação
- Aprovações
- Acompanhamento

Migração de Dados

Estratégia de Migração

A migração será realizada de forma **incremental e não-destrutiva**, garantindo que o sistema atual continue funcionando durante todo o processo.

Fase 1: Preparação (Semana 1)

1. Backup Completo

```
-- Backup do banco atual
pg_dump anrielly_db > backup_pre_migration.sql

-- Backup de arquivos
tar -czf media_backup.tar.gz public/uploads/
```

2. Criação de Novas Tabelas

```
-- Tabelas de tenants
CREATE TABLE tenants (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  name TEXT NOT NULL,
  slug TEXT UNIQUE NOT NULL,
  status tenant_status DEFAULT 'trial',
  subscription_plan TEXT DEFAULT 'basic',
  subscription_status TEXT DEFAULT 'trialing',
  trial_ends_at TIMESTAMPTZ,
  settings JSONB DEFAULT '{}',
  billing_info JSONB DEFAULT '{}',
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW()
);

-- Tabelas de usuários aprimoradas
CREATE TABLE user_profiles (
  id UUID PRIMARY KEY REFERENCES auth.users(id),
  tenant_id UUID REFERENCES tenants(id),
  email TEXT NOT NULL,
  role user_role NOT NULL,
  status user_status DEFAULT 'active',
  first_name TEXT,
  last_name TEXT,
  full_name TEXT GENERATED ALWAYS AS (first_name || ' ' || last_name) STORED,
  profile_data JSONB,
  permissions JSONB DEFAULT '[]',
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW()
);

-- Tabelas de fornecedores
CREATE TABLE suppliers (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  tenant_id UUID REFERENCES tenants(id),
  user_id UUID REFERENCES user_profiles(id),
```

```

company_name TEXT NOT NULL,
business_name TEXT,
business_type supplier_category NOT NULL,
status supplier_status DEFAULT 'pending',
verification_status verification_status DEFAULT 'pending',
rating JSONB DEFAULT '{}',
stats JSONB DEFAULT '{}',
contact_info JSONB DEFAULT '{}',
address JSONB DEFAULT '{}',
service_areas JSONB DEFAULT '[]',
specialties TEXT[],
tags TEXT[],
created_at TIMESTAMPTZ DEFAULT NOW(),
updated_at TIMESTAMPTZ DEFAULT NOW()
);

-- Outras tabelas necessárias...

```

3. Migração de Dados Existentes

```

-- Criar tenant padrão para dados existentes
INSERT INTO tenants (id, name, slug, status, subscription_plan)
VALUES ('anrielly-gomes-default', 'Anrielly Gomes', 'anrielly-
gomes', 'active', 'premium');

-- Migrar usuários existentes
INSERT INTO user_profiles (id, tenant_id, email, role,
first_name, last_name)
SELECT
    id,
    'anrielly-gomes-default',
    email,
    CASE
        WHEN is_admin THEN 'admin'
        ELSE 'usuario'
    END,
    COALESCE(raw_user_meta_data->>'first_name', split_part(email,
'@', 1)),
    COALESCE(raw_user_meta_data->>'last_name', '')
FROM auth.users;

-- Migrar eventos existentes
UPDATE events
SET tenant_id = 'anrielly-gomes-default'
WHERE tenant_id IS NULL;

```

Fase 2: Implementação Gradual (Semanas 2-4)

- 1. Deploy da Nova Arquitetura** - Deploy em ambiente de staging - Testes de compatibilidade - Validação de funcionalidades existentes
- 2. Migração de Funcionalidades** - Substituição gradual dos hooks antigos - Implementação dos novos componentes - Testes de regressão
- 3. Configuração Multi-Tenant** - Configuração do tenant padrão - Migração de configurações existentes - Testes de isolamento de dados

Fase 3: Ativação (Semana 5)

- 1. Switch Gradual** - Ativação por funcionalidade - Monitoramento de performance - Rollback se necessário
- 2. Validação Final** - Testes de todas as funcionalidades - Validação de dados migrados - Performance benchmarks

Scripts de Migração

Script 1: Preparação do Ambiente

```
#!/bin/bash
# prepare_migration.sh

echo "🚀 Iniciando preparação para migração..."

# Backup do banco
echo "📦 Criando backup do banco..."
pg_dump $DATABASE_URL > "backup_$(date +%Y%m%d_%H%M%S).sql"

# Backup de arquivos
echo "📁 Criando backup de arquivos..."
tar -czf "media_backup_$(date +%Y%m%d_%H%M%S).tar.gz" public/uploads/

# Verificar dependências
echo "🔍 Verificando dependências..."
npm audit
npm outdated

echo "✅ Preparação concluída!"
```


Script 2: Migração de Dados

```
-- migration_001_tenants.sql
BEGIN;

-- Criar tenant padrão
INSERT INTO tenants (id, name, slug, status, subscription_plan,
settings)
VALUES (
    'anrielly-gomes-default',
    'Anrielly Gomes Cerimônias',
    'anrielly-gomes',
    'active',
    'premium',
    '{
        "brand": {
            "name": "Anrielly Gomes Cerimônias",
            "primary_color": "#8B5CF6",
            "secondary_color": "#A78BFA"
        },
        "features": {
            "cms_enabled": true,
            "events_enabled": true,
            "suppliers_enabled": true,
            "analytics_enabled": true,
            "ai_enabled": true,
            "questionnaires_enabled": true
        }
    }'
);

-- Migrar usuários
INSERT INTO user_profiles (id, tenant_id, email, role,
first_name, last_name, status)
SELECT
    u.id,
    'anrielly-gomes-default',
    u.email,
    CASE
        WHEN u.raw_user_meta_data->>'role' = 'admin' THEN 'admin'
        WHEN u.raw_user_meta_data->>'role' = 'supplier' THEN
'cliente'
        ELSE 'usuario'
    END,
    COALESCE(u.raw_user_meta_data->>'first_name',
split_part(u.email, '@', 1)),
    COALESCE(u.raw_user_meta_data->>'last_name', ''),
    'active'
FROM auth.users u
WHERE u.email IS NOT NULL;
```

```
COMMIT;
```

Script 3: Validação

```
-- validation_queries.sql

-- Verificar migração de usuários
SELECT
    role,
    COUNT(*) as count
FROM user_profiles
GROUP BY role;

-- Verificar eventos migrados
SELECT
    COUNT(*) as total_events,
    COUNT(CASE WHEN tenant_id IS NOT NULL THEN 1 END) as
events_with_tenant
FROM events;

-- Verificar integridade referencial
SELECT
    table_name,
    constraint_name,
    constraint_type
FROM information_schema.table_constraints
WHERE constraint_type = 'FOREIGN KEY'
AND table_schema = 'public';
```

Configuração do Ambiente

Variáveis de Ambiente

Adicione as seguintes variáveis ao arquivo `.env`:

```
# Configurações existentes (manter)
VITE_SUPABASE_URL=your_supabase_url
VITE_SUPABASE_ANON_KEY=your_supabase_anon_key

# Novas configurações multi-tenant
VITE_DEFAULT_TENANT_ID=anrielly-gomes-default
VITE_ENABLE_MULTI_TENANT=true
VITE_ENABLE_SUPPLIER_SYSTEM=true

# Configurações de cache
VITE_CACHE_DURATION=300000
```

```
VITE_STALE_TIME=120000

# Configurações de upload
VITE_MAX_FILE_SIZE=10485760
VITE_ALLOWED_FILE_TYPES=image/*,application/pdf

# Configurações de notificações
VITE_ENABLE_PUSH_NOTIFICATIONS=true
VITE_ENABLE_EMAIL_NOTIFICATIONS=true

# Analytics
VITE_ENABLE_ANALYTICS=true
VITE_ANALYTICS_PROVIDER=supabase
```

Configuração do Supabase

1. Row Level Security (RLS)

```
-- Habilitar RLS em todas as tabelas
ALTER TABLE tenants ENABLE ROW LEVEL SECURITY;
ALTER TABLE user_profiles ENABLE ROW LEVEL SECURITY;
ALTER TABLE events ENABLE ROW LEVEL SECURITY;
ALTER TABLE suppliers ENABLE ROW LEVEL SECURITY;

-- Políticas para tenants
CREATE POLICY "Admin master can view all tenants" ON tenants
  FOR SELECT USING (
    EXISTS (
      SELECT 1 FROM user_profiles
      WHERE user_profiles.id = auth.uid()
      AND user_profiles.role = 'admin_master'
    )
  );

CREATE POLICY "Users can view their own tenant" ON tenants
  FOR SELECT USING (
    id IN (
      SELECT tenant_id FROM user_profiles
      WHERE user_profiles.id = auth.uid()
    )
  );

-- Políticas para user_profiles
CREATE POLICY "Users can view profiles in their tenant" ON
user_profiles
  FOR SELECT USING (
    tenant_id IN (
      SELECT tenant_id FROM user_profiles
      WHERE user_profiles.id = auth.uid()
    )
  );
```

```

OR
EXISTS (
    SELECT 1 FROM user_profiles
    WHERE user_profiles.id = auth.uid()
    AND user_profiles.role = 'admin_master'
)
);

-- Políticas para events
CREATE POLICY "Users can view events in their tenant" ON events
FOR SELECT USING (
    tenant_id IN (
        SELECT tenant_id FROM user_profiles
        WHERE user_profiles.id = auth.uid()
    )
);

```

2. Funções do Banco

```

-- Função para criar tenant
CREATE OR REPLACE FUNCTION create_tenant(
    tenant_name TEXT,
    tenant_slug TEXT,
    admin_email TEXT,
    admin_password TEXT,
    admin_first_name TEXT,
    admin_last_name TEXT
)
RETURNS UUID
LANGUAGE plpgsql
SECURITY DEFINER
AS $$
DECLARE
    new_tenant_id UUID;
    new_user_id UUID;
BEGIN
    -- Criar tenant
    INSERT INTO tenants (name, slug, status, subscription_plan)
    VALUES (tenant_name, tenant_slug, 'trial', 'basic')
    RETURNING id INTO new_tenant_id;

    -- Criar usuário admin
    INSERT INTO auth.users (email, encrypted_password,
email_confirmed_at)
    VALUES (admin_email, crypt(admin_password, gen_salt('bf')),
NOW())
    RETURNING id INTO new_user_id;

    -- Criar perfil do admin
    INSERT INTO user_profiles (id, tenant_id, email, role,

```

```

first_name, last_name, status)
    VALUES (new_user_id, new_tenant_id, admin_email, 'admin',
admin_first_name, admin_last_name, 'active');

    RETURN new_tenant_id;
END;
$$;

-- Função para verificar permissões
CREATE OR REPLACE FUNCTION check_user_permission(
    user_id UUID,
    resource TEXT,
    action TEXT
)
RETURNS BOOLEAN
LANGUAGE plpgsql
SECURITY DEFINER
AS $$
DECLARE
    user_role TEXT;
    user_permissions JSONB;
BEGIN
    SELECT role, permissions INTO user_role, user_permissions
    FROM user_profiles
    WHERE id = user_id;

    -- Admin master tem acesso total
    IF user_role = 'admin_master' THEN
        RETURN TRUE;
    END IF;

    -- Admin tem acesso total no seu tenant
    IF user_role = 'admin' THEN
        RETURN TRUE;
    END IF;

    -- Verificar permissões específicas
    RETURN EXISTS (
        SELECT 1 FROM jsonb_array_elements(user_permissions) AS perm
        WHERE perm->>'resource' = resource
        AND (perm->>'action' = action OR perm->>'action' = 'manage')
    );
END;
$$;

```

Configuração do React Query

```

// src/lib/queryClient.ts
import { QueryClient } from '@tanstack/react-query';

```

```
export const queryClient = new QueryClient({
  defaultOptions: {
    queries: {
      staleTime: 2 * 60 * 1000, // 2 minutos
      cacheTime: 5 * 60 * 1000, // 5 minutos
      retry: (failureCount, error: any) => {
        // Não tentar novamente em erros de autenticação
        if (error?.status === 401 || error?.status === 403) {
          return false;
        }
        return failureCount < 3;
      },
      refetchOnWindowFocus: false,
      refetchOnReconnect: true,
    },
    mutations: {
      retry: 1,
    },
  },
});
```

Implementação Gradual

Cronograma de Implementação

Semana 1: Preparação

- ☐ Backup completo do sistema
- ☐ Criação do ambiente de staging
- ☐ Implementação das novas tabelas
- ☐ Migração inicial de dados
- ☐ Testes de compatibilidade

Semana 2: Sistema de Autenticação

- ☐ Deploy dos novos hooks de autenticação
- ☐ Implementação do sistema multi-tenant
- ☐ Testes de login e permissões
- ☐ Validação de isolamento de dados

Semana 3: Sistema de Eventos

- ☐ Deploy dos hooks de eventos aprimorados
- ☐ Migração dos componentes de eventos
- ☐ Testes de funcionalidades existentes

- [] Validação de performance

Semana 4: Sistema CMS e Fornecedores

- [] Deploy dos hooks CMS otimizados
- [] Implementação do sistema de fornecedores
- [] Testes de integração
- [] Validação de funcionalidades

Semana 5: Dashboard e Finalização

- [] Deploy dos dashboards multi-perfil
- [] Implementação da navegação adaptativa
- [] Testes finais de integração
- [] Go-live em produção

Estratégia de Feature Flags

Utilize feature flags para controlar a ativação gradual das funcionalidades:

```
// src/lib/featureFlags.ts
export const featureFlags = {
  MULTI_TENANT: process.env.VITE_ENABLE_MULTI_TENANT === 'true',
  SUPPLIER_SYSTEM: process.env.VITE_ENABLE_SUPPLIER_SYSTEM ===
'true',
  ENHANCED_EVENTS: process.env.VITE_ENABLE_ENHANCED_EVENTS ===
'true',
  OPTIMIZED_CMS: process.env.VITE_ENABLE_OPTIMIZED_CMS ===
'true',
  NEW_DASHBOARD: process.env.VITE_ENABLE_NEW_DASHBOARD ===
'true',
};

// Uso nos componentes
export function EventsPage() {
  const useEventsHook = featureFlags.ENHANCED_EVENTS
    ? useEventsEnhanced
    : useEvents;

  // resto do componente...
}
```

Rollback Strategy

Em caso de problemas, o rollback pode ser feito rapidamente:

```
#!/bin/bash
# rollback.sh

echo "🔄 Iniciando rollback..."

# Restaurar banco de dados
psql $DATABASE_URL < backup_pre_migration.sql

# Restaurar arquivos
tar -xzf media_backup.tar.gz

# Reverter deploy
git checkout main
npm run build
npm run deploy

echo "✅ Rollback concluído!"
```

Testes e Validação

Estratégia de Testes

1. Testes de Compatibilidade

Funcionalidades Existentes que DEVEM continuar funcionando:

```
// tests/compatibility/questionnaires.test.ts
describe('Questionários - Compatibilidade', () => {
  test('Deve carregar questionários existentes', async () => {
    const questionnaires = await getQuestionnaires();
    expect(questionnaires).toBeDefined();
    expect(questionnaires.length).toBeGreaterThan(0);
  });

  test('Deve permitir responder questionários', async () => {
    const response = await
submitQuestionnaireResponse(mockData);
    expect(response.success).toBe(true);
  });
});

// tests/compatibility/ai.test.ts
describe('Sistema de IA - Compatibilidade', () => {
  test('Deve processar solicitações de IA', async () => {
    const result = await processAIRequest(mockPrompt);
    expect(result).toBeDefined();
    expect(result.response).toBeTruthy();
  });
});
```



```

    });
  });

  // tests/compatibility/events.test.ts
  describe('Eventos - Compatibilidade', () => {
    test('Deve carregar eventos existentes', async () => {
      const events = await getEvents();
      expect(events).toBeDefined();
      expect(Array.isArray(events)).toBe(true);
    });

    test('Deve criar novos eventos', async () => {
      const newEvent = await createEvent(mockEventData);
      expect(newEvent.id).toBeDefined();
    });
  });
});

```

2. Testes de Novas Funcionalidades

```

  // tests/features/multiTenant.test.ts
  describe('Sistema Multi-Tenant', () => {
    test('Deve isolar dados por tenant', async () => {
      const tenant1Events = await getEvents({ tenantId: 'tenant1' });
      const tenant2Events = await getEvents({ tenantId: 'tenant2' });

      expect(tenant1Events).not.toEqual(tenant2Events);
    });

    test('Deve respeitar permissões por role', async () => {
      const adminUser = mockUser({ role: 'admin' });
      const regularUser = mockUser({ role: 'usuario' });

      expect(hasPermission(adminUser, 'events', 'manage')).toBe(true);
      expect(hasPermission(regularUser, 'events', 'manage')).toBe(false);
    });
  });

  // tests/features/suppliers.test.ts
  describe('Sistema de Fornecedores', () => {
    test('Deve criar fornecedor', async () => {
      const supplier = await createSupplier(mockSupplierData);
      expect(supplier.id).toBeDefined();
      expect(supplier.status).toBe('pending');
    });

    test('Deve processar cotações', async () => {

```

```
    const quote = await createQuote(mockQuoteData);
    expect(quote.id).toBeDefined();
    expect(quote.status).toBe('draft');
  });
});
```

3. Testes de Performance

```
// tests/performance/queries.test.ts
describe('Performance - Queries', () => {
  test('Carregamento de eventos deve ser < 500ms', async () => {
    const start = Date.now();
    await getEvents();
    const duration = Date.now() - start;

    expect(duration).toBeLessThan(500);
  });

  test('Cache deve reduzir tempo de carregamento', async () => {
    // Primeira chamada
    const start1 = Date.now();
    await getEvents();
    const duration1 = Date.now() - start1;

    // Segunda chamada (com cache)
    const start2 = Date.now();
    await getEvents();
    const duration2 = Date.now() - start2;

    expect(duration2).toBeLessThan(duration1 * 0.5);
  });
});
```

4. Testes de Integração

```
#!/bin/bash
# integration_tests.sh

echo "🔧 Executando testes de integração..."

# Testes de API
npm run test:api

# Testes de UI
npm run test:e2e

# Testes de performance
npm run test:performance
```

```
# Testes de compatibilidade
npm run test:compatibility

echo "✅ Testes concluídos!"
```

Checklist de Validação

✅ Funcionalidades Existentes

- ☐ Questionários funcionando
- ☐ Sistema de IA operacional
- ☐ Eventos carregando corretamente
- ☐ CMS funcionando
- ☐ Autenticação preservada
- ☐ Uploads de arquivos
- ☐ Notificações
- ☐ Relatórios existentes

✅ Novas Funcionalidades

- ☐ Sistema multi-tenant ativo
- ☐ Perfis de usuário funcionando
- ☐ Sistema de fornecedores
- ☐ Cotações e contratos
- ☐ Dashboards multi-perfil
- ☐ Navegação adaptativa
- ☐ Cache otimizado
- ☐ Performance melhorada

✅ Segurança

- ☐ RLS configurado
- ☐ Isolamento de dados
- ☐ Permissões por role
- ☐ Autenticação segura
- ☐ Validação de dados
- ☐ Sanitização de inputs

Deployment

Ambiente de Staging

1. Configuração do Staging

```
#!/bin/bash
# setup_staging.sh

echo "🚀 Configurando ambiente de staging..."

# Criar banco de staging
createdb anrielly_staging

# Restaurar dados de produção
pg_dump $PROD_DATABASE_URL | psql $STAGING_DATABASE_URL

# Executar migrações
npm run migrate:staging

# Deploy da aplicação
npm run deploy:staging

echo "✅ Staging configurado!"
```

2. Testes em Staging

```
#!/bin/bash
# test_staging.sh

echo "🔧 Testando ambiente de staging..."

# Testes automatizados
npm run test:staging

# Testes de carga
npm run test:load

# Validação de dados
npm run validate:data

# Testes de usuário
npm run test:user-acceptance

echo "✅ Testes de staging concluídos!"
```

Deployment em Produção

1. Preparação

```
#!/bin/bash
# pre_deploy.sh

echo "📋 Preparando deployment..."

# Backup final
pg_dump $DATABASE_URL > "backup_final_$(date +%Y%m%d_%H%M%S).sql"

# Verificar dependências
npm audit --audit-level high

# Build de produção
npm run build

# Testes finais
npm run test:production

echo "✅ Preparação concluída!"
```

2. Deploy

```
#!/bin/bash
# deploy.sh

echo "🚀 Iniciando deployment..."

# Modo de manutenção
echo "Ativando modo de manutenção..."
# curl -X POST $MAINTENANCE_MODE_URL

# Executar migrações
echo "Executando migrações..."
npm run migrate:production

# Deploy da aplicação
echo "Fazendo deploy da aplicação..."
npm run deploy:production

# Verificar saúde da aplicação
echo "Verificando saúde da aplicação..."
npm run health-check

# Desativar modo de manutenção
echo "Desativando modo de manutenção..."
```

```
# curl -X DELETE $MAINTENANCE_MODE_URL

echo "✅ Deployment concluído!"
```

3. Verificação Pós-Deploy

```
#!/bin/bash
# post_deploy.sh

echo "🔍 Verificando deployment..."

# Testes de smoke
npm run test:smoke

# Verificar métricas
npm run check:metrics

# Validar funcionalidades críticas
npm run validate:critical

# Notificar equipe
echo "📧 Notificando equipe..."
# curl -X POST $SLACK_WEBHOOK -d '{"text":"Deploy concluído com sucesso!"}'

echo "✅ Verificação concluída!"
```

Configuração de CI/CD

```
# .github/workflows/deploy.yml
name: Deploy

on:
  push:
    branches: [main]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
        with:
          node-version: '18'
      - run: npm ci
      - run: npm run test
      - run: npm run test:compatibility
```

```

deploy-staging:
  needs: test
  runs-on: ubuntu-latest
  environment: staging
  steps:
    - uses: actions/checkout@v3
    - run: npm ci
    - run: npm run build
    - run: npm run deploy:staging
    - run: npm run test:staging

deploy-production:
  needs: deploy-staging
  runs-on: ubuntu-latest
  environment: production
  if: github.ref == 'refs/heads/main'
  steps:
    - uses: actions/checkout@v3
    - run: npm ci
    - run: npm run build
    - run: npm run deploy:production
    - run: npm run test:smoke

```

Monitoramento

Métricas de Performance

1. Frontend

```

// src/lib/monitoring.ts
export class PerformanceMonitor {
  static trackPageLoad(pageName: string) {
    const start = performance.now();

    return () => {
      const duration = performance.now() - start;

      // Enviar métrica
      this.sendMetric('page_load', {
        page: pageName,
        duration,
        timestamp: Date.now()
      });
    };
  }

  static trackQueryPerformance(queryKey: string, duration:
number) {

```

```

    this.sendMetric('query_performance', {
      query: queryKey,
      duration,
      timestamp: Date.now()
    });
  }

  private static sendMetric(event: string, data: any) {
    // Implementar envio de métricas
    console.log(`[METRIC] ${event}:`, data);
  }
}

// Uso nos hooks
export function useEventsEnhanced() {
  const query = useQuery({
    queryKey: eventKeys.all,
    queryFn: async () => {
      const start = performance.now();
      const result = await EventService.getEvents();
      const duration = performance.now() - start;

      PerformanceMonitor.trackQueryPerformance('events.getAll',
duration);
      return result;
    }
  });

  return query;
}

```

2. Backend

```

-- Monitoramento de queries lentas
CREATE EXTENSION IF NOT EXISTS pg_stat_statements;

-- Query para identificar queries lentas
SELECT
  query,
  calls,
  total_time,
  mean_time,
  rows
FROM pg_stat_statements
WHERE mean_time > 100
ORDER BY mean_time DESC
LIMIT 10;

```


Alertas e Notificações

1. Configuração de Alertas

```
// src/lib/alerts.ts
export class AlertSystem {
  static async checkSystemHealth() {
    const checks = [
      this.checkDatabaseConnection(),
      this.checkAPIResponse(),
      this.checkMemoryUsage(),
      this.checkErrorRate()
    ];

    const results = await Promise.all(checks);
    const failures = results.filter(r => !r.success);

    if (failures.length > 0) {
      await this.sendAlert('System Health Alert', failures);
    }
  }

  private static async checkDatabaseConnection() {
    try {
      await supabase.from('tenants').select('count').limit(1);
      return { success: true, check: 'database' };
    } catch (error) {
      return { success: false, check: 'database', error };
    }
  }

  private static async sendAlert(title: string, details: any) {
    // Implementar envio de alertas (Slack, email, etc.)
    console.error(`[ALERT] ${title}:`, details);
  }
}
```

2. Dashboard de Monitoramento

```
// src/components/monitoring/SystemDashboard.tsx
export function SystemDashboard() {
  const [metrics, setMetrics] = useState({
    responseTime: 0,
    errorRate: 0,
    activeUsers: 0,
    systemLoad: 0
  });

  useEffect(() => {
```

```

const interval = setInterval(async () => {
  const newMetrics = await fetchSystemMetrics();
  setMetrics(newMetrics);
}, 30000); // Atualizar a cada 30 segundos

return () => clearInterval(interval);
}, []);

return (
  <div className="grid gap-4 md:grid-cols-2 lg:grid-cols-4">
    <MetricCard
      title="Tempo de Resposta"
      value={`${metrics.responseTime}ms`}
      trend={metrics.responseTime < 500 ? 'up' : 'down'}
    />
    <MetricCard
      title="Taxa de Erro"
      value={`${metrics.errorRate}%`}
      trend={metrics.errorRate < 1 ? 'up' : 'down'}
    />
    <MetricCard
      title="Usuários Ativos"
      value={metrics.activeUsers}
      trend="up"
    />
    <MetricCard
      title="Carga do Sistema"
      value={`${metrics.systemLoad}%`}
      trend={metrics.systemLoad < 80 ? 'up' : 'down'}
    />
  </div>
);
}

```

Logs e Debugging

1. Sistema de Logs

```

// src/lib/logger.ts
export class Logger {
  static info(message: string, data?: any) {
    console.log(`[INFO] ${new Date().toISOString()} - ${message}`, data);
  }

  static error(message: string, error?: any) {
    console.error(`[ERROR] ${new Date().toISOString()} - ${message}`, error);
  }
}

// Enviar para serviço de monitoramento

```

```
    this.sendToMonitoring('error', { message, error });
  }

  static performance(operation: string, duration: number) {
    console.log(`[PERF] ${operation}: ${duration}ms`);

    if (duration > 1000) {
      this.sendToMonitoring('slow_operation', { operation,
duration });
    }
  }

  private static sendToMonitoring(type: string, data: any) {
    // Implementar envio para serviço de monitoramento
  }
}
```

Conclusão

Benefícios Esperados

1. Performance

- **50% redução** no tempo de carregamento
- **Cache inteligente** com React Query
- **Consultas otimizadas** no banco de dados
- **Lazy loading** de componentes

2. Escalabilidade

- **Arquitetura multi-tenant** robusta
- **Isolamento de dados** por tenant
- **Sistema de permissões** granular
- **Preparado para crescimento**

3. Experiência do Usuário

- **Dashboards personalizados** por perfil
- **Navegação adaptativa** por role
- **Interface mais intuitiva**
- **Funcionalidades específicas** por tipo de usuário

4. Funcionalidades

- **Sistema de fornecedores** completo

- Cotações automatizadas
- Contratos digitais
- Analytics avançados
- Versionamento de conteúdo

Próximos Passos

1. Revisar este guia com a equipe técnica
2. Preparar ambiente de staging
3. Executar migração de dados
4. Implementar gradualmente
5. Monitorar e ajustar

Suporte

Para dúvidas ou problemas durante a migração:

- **Documentação técnica:** `/docs`
- **Logs do sistema:** `/admin/logs`
- **Monitoramento:** `/admin/monitoring`
- **Rollback:** Execute `./rollback.sh`

Data de criação: `$(date)` **Versão:** 1.0 **Status:** Pronto para implementação

Este guia garante uma migração segura e sem interrupções, preservando todas as funcionalidades existentes enquanto adiciona as novas capacidades do sistema.