📋 Guia de Execução das Migrations SQL

📁 Localização dos Arquivos

Os arquivos SQL estão em:

```
/home/ubuntu/project/supabase/migrations/
├── 001_multiempresa.sql      (já executado)
├── 002_rls_policies.sql      (já executado)
├── 003_v_user_context.sql    ← EXECUTAR
├── 004_processes.sql         ← EXECUTAR
├── 005_rls_processes.sql     ← EXECUTAR
```

🔢 Ordem de Execução

Execute na seguinte ordem no **Supabase SQL Editor**:

1. **003_v_user_context.sql** - Cria a view de contexto do usuário
2. **004_processes.sql** - Cria as tabelas de processos
3. **005_rls_processes.sql** - Aplica as políticas de segurança RLS

📝 SQL para Copiar e Colar

1️⃣ Migration 003: v_user_context View

```sql
-- ============================================
-- SGI FV - Migration 003: v_user_context View
-- ============================================
-- Data: 2026-02-22
-- Descrição: View para contexto do usuário autenticado
-- ============================================

-- Criar view v_user_context para consulta fácil do contexto do usuário
CREATE OR REPLACE VIEW public.v_user_context AS
SELECT
  p.id as user_id,
  p.email,
  p.nome_completo,
  p.org_id,
  om.role as org_role,
  o.slug as org_slug,
  o.name as org_name
FROM profiles p
LEFT JOIN org_members om ON om.user_id = p.id AND om.org_id = p.org_id
LEFT JOIN organizations o ON o.id = p.org_id;

-- Grant access to authenticated users
GRANT SELECT ON public.v_user_context TO authenticated;

COMMENT ON VIEW public.v_user_context IS 'Contexto completo do usuário para uso no
frontend';
```

2️⃣ Migration 004: Processes Tables

```sql
-- ============================================
-- SGI FV - Migration 004: Processes Tables
-- ============================================
-- Data: 2026-02-22
-- Descrição: Tabelas de processos e eventos
-- ============================================

-- ============================================
-- 1. TABELA: processes
-- ============================================
CREATE TABLE IF NOT EXISTS public.processes (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  org_id uuid NOT NULL REFERENCES organizations(id) ON DELETE CASCADE,
  titulo text NOT NULL,
  protocolo text UNIQUE,
  status text NOT NULL DEFAULT 'cadastro' CHECK (status IN ('cadastro','triagem','an-
alise','concluido')),
  cliente_nome text,
  cliente_documento text,
  cliente_contato text,
  responsavel_user_id uuid REFERENCES auth.users(id),
  created_at timestamptz DEFAULT now(),
  updated_at timestamptz DEFAULT now()
);

COMMENT ON TABLE public.processes IS 'Processos administrativos da organização';
COMMENT ON COLUMN processes.status IS 'cadastro → triagem → analise → concluido';

-- ============================================
-- 2. TABELA: process_events
-- ============================================
CREATE TABLE IF NOT EXISTS public.process_events (
  id uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  org_id uuid NOT NULL REFERENCES organizations(id) ON DELETE CASCADE,
  process_id uuid NOT NULL REFERENCES processes(id) ON DELETE CASCADE,
  tipo text NOT NULL CHECK (tipo IN ('registro','status_change','observacao','docu-
mento','atribuicao')),
  mensagem text NOT NULL,
  created_by uuid REFERENCES auth.users(id),
  created_at timestamptz DEFAULT now()
);

COMMENT ON TABLE public.process_events IS 'Timeline de eventos do processo';
COMMENT ON COLUMN process_events.tipo IS 'registro=criação, status_change=mudança de
status, observacao=nota, documento=upload, atribuicao=mudança de responsável';

-- ============================================
-- 3. ÍNDICES
-- ============================================
CREATE INDEX IF NOT EXISTS idx_processes_org_id ON processes(org_id);
CREATE INDEX IF NOT EXISTS idx_processes_status ON processes(status);
CREATE INDEX IF NOT EXISTS idx_processes_created_at ON processes(created_at DESC);
CREATE INDEX IF NOT EXISTS idx_process_events_process_id ON pro-
cess_events(process_id);
CREATE INDEX IF NOT EXISTS idx_process_events_org_id ON process_events(org_id);
CREATE INDEX IF NOT EXISTS idx_process_events_created_at ON process_events(created_at
DESC);

-- ============================================
-- 4. TRIGGER: updated_at automático
-- ============================================
CREATE OR REPLACE FUNCTION update_updated_at_column()
```

```
RETURNS TRIGGER AS $$
BEGIN
  NEW.updated_at = now();
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS update_processes_updated_at ON processes;
CREATE TRIGGER update_processes_updated_at
  BEFORE UPDATE ON processes
  FOR EACH ROW
  EXECUTE FUNCTION update_updated_at_column();

-- ============================================
-- 5. FUNÇÃO: gerar protocolo automático
-- ============================================
CREATE OR REPLACE FUNCTION generate_protocol()
RETURNS TRIGGER AS $$
DECLARE
  year_str text;
  seq_num int;
  new_protocol text;
BEGIN
  IF NEW.protocolo IS NULL THEN
    year_str := to_char(now(), 'YYYY');

    SELECT COALESCE(MAX(
      CAST(SUBSTRING(protocolo FROM 'SGI-' || year_str || '-([0-9]+)') AS int)
    ), 0) + 1
    INTO seq_num
    FROM processes
    WHERE protocolo LIKE 'SGI-' || year_str || '-%';

    NEW.protocolo := 'SGI-' || year_str || '-' || LPAD(seq_num::text, 3, '0');
  END IF;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS trigger_generate_protocol ON processes;
CREATE TRIGGER trigger_generate_protocol
  BEFORE INSERT ON processes
  FOR EACH ROW
  EXECUTE FUNCTION generate_protocol();
```

3️⃣ Migration 005: RLS for Processes

```sql
-- ============================================
-- SGI FV - Migration 005: RLS for Processes
-- ============================================
-- Data: 2026-02-22
-- Descrição: Políticas de segurança para processos
-- ============================================

-- ============================================
-- 1. HABILITAR RLS
-- ============================================
ALTER TABLE processes ENABLE ROW LEVEL SECURITY;
ALTER TABLE process_events ENABLE ROW LEVEL SECURITY;

-- ============================================
-- 2. POLICIES: processes
-- ============================================

-- Membros podem ver processos da sua organização
DROP POLICY IF EXISTS "Members can view org processes" ON processes;
CREATE POLICY "Members can view org processes"
  ON processes FOR SELECT
  USING (is_org_member(org_id));

-- Admins/Owners podem inserir processos
DROP POLICY IF EXISTS "Admins can insert processes" ON processes;
CREATE POLICY "Admins can insert processes"
  ON processes FOR INSERT
  WITH CHECK (is_org_admin(org_id));

-- Admins/Owners podem atualizar processos
DROP POLICY IF EXISTS "Admins can update processes" ON processes;
CREATE POLICY "Admins can update processes"
  ON processes FOR UPDATE
  USING (is_org_admin(org_id));

-- Admins/Owners podem deletar processos
DROP POLICY IF EXISTS "Admins can delete processes" ON processes;
CREATE POLICY "Admins can delete processes"
  ON processes FOR DELETE
  USING (is_org_admin(org_id));

-- ============================================
-- 3. POLICIES: process_events
-- ============================================

-- Membros podem ver eventos da sua organização
DROP POLICY IF EXISTS "Members can view org process events" ON process_events;
CREATE POLICY "Members can view org process events"
  ON process_events FOR SELECT
  USING (is_org_member(org_id));

-- Admins/Owners podem inserir eventos
DROP POLICY IF EXISTS "Admins can insert process events" ON process_events;
CREATE POLICY "Admins can insert process events"
  ON process_events FOR INSERT
  WITH CHECK (is_org_admin(org_id));

-- Admins/Owners podem atualizar eventos
DROP POLICY IF EXISTS "Admins can update process events" ON process_events;
CREATE POLICY "Admins can update process events"
  ON process_events FOR UPDATE
  USING (is_org_admin(org_id));
```

🚀 Como Executar no Supabase

Passo a Passo:

1. **Acesse seu projeto Supabase**: https://supabase.com/dashboard
2. **Navegue até**: SQL Editor (ícone no menu lateral esquerdo)
3. **Clique em**: “New query”
4. **Copie e cole** o SQL da migration 003 primeiro
5. **Clique em**: “RUN” (ou Ctrl+Enter)
6. **Verifique**: Se não houve erros (Success: ...)
7. **Repita** para migrations 004 e 005

⚠️ Importante:

- Execute **na ordem** (003 → 004 → 005)
- A migration 005 depende das funções `is_org_member()` e `is_org_admin()` que devem existir da migration 002
- Após executar, **recarregue** a aplicação para testar

✅ Verificação

Após executar todas as migrations, verifique se foram criadas:

```
-- Verificar tabelas
SELECT table_name FROM information_schema.tables
WHERE table_schema = 'public'
AND table_name IN ('processes', 'process_events', 'organizations', 'org_members', 'pro
files');

-- Verificar view
SELECT * FROM v_user_context LIMIT 1;

-- Verificar políticas RLS
SELECT tablename, policyname FROM pg_policies
WHERE tablename IN ('processes', 'process_events');
```

🔗 Pré-requisitos

As migrations 003-005 dependem de:
- ✅ `organizations` (criada em 001)
- ✅ `org_members` (criada em 001)
- ✅ `profiles` (criada em 001)
- ✅ `is_org_member()` (criada em 002)
- ✅ `is_org_admin()` (criada em 002)

Se algum desses não existir, execute as migrations 001 e 002 primeiro.