Repositorio con Proyecto

https://github.com/JoseIgnacioSalinasLopez/2er-Departamental

Organizacion Scripts SQL

Estructura de tablas

```sql
-- 1. TABLA DE PERFILES
CREATE TABLE public.perfiles (
    id UUID PRIMARY KEY REFERENCES auth.users (id) ON DELETE CASCADE,
    nombre_completo TEXT NOT NULL,
    rol TEXT NOT NULL DEFAULT 'user',
    creado_en TIMESTAMPTZ DEFAULT NOW()
);

ALTER TABLE public.perfiles ENABLE ROW LEVEL SECURITY;

-- 2. TABLA DE DIRECCIONES DE ENVÍO
CREATE TABLE public.direcciones_envio (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    usuario_id UUID NOT NULL REFERENCES public.perfiles (id) ON DELETE CASCADE,
    linea_direccion_1 TEXT NOT NULL,
    linea_direccion_2 TEXT,
    ciudad TEXT NOT NULL,
    estado TEXT,
    codigo_postal TEXT NOT NULL,
    pais TEXT NOT NULL,
    es_predeterminada BOOLEAN DEFAULT FALSE
);

ALTER TABLE public.direcciones_envio ENABLE ROW LEVEL SECURITY;

-- 3. TABLA DE CATEGORÍAS
CREATE TABLE public.categorias (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    nombre TEXT NOT NULL UNIQUE,
    descripcion TEXT,
    esta_activa BOOLEAN NOT NULL DEFAULT TRUE
);

ALTER TABLE public.categorias ENABLE ROW LEVEL SECURITY;

-- 4. TABLA DE PRODUCTOS
CREATE TABLE public.productos (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    categoria_id UUID NOT NULL REFERENCES public.categorias (id) ON DELETE RESTRICT,
    nombre TEXT NOT NULL,
    descripcion TEXT,
    precio NUMERIC(10, 2) NOT NULL CHECK (precio >= 0),
    stock INTEGER NOT NULL CHECK (stock >= 0),
    url_imagen TEXT,
    esta_activo BOOLEAN NOT NULL DEFAULT TRUE
);
```

```sql
ALTER TABLE public.productos ENABLE ROW LEVEL SECURITY;

-- 5. TABLA DE CARRITOS
CREATE TABLE public.carritos (
    usuario_id UUID PRIMARY KEY REFERENCES public.perfiles (id) ON DELETE CASCADE,
    creado_en TIMESTAMPTZ DEFAULT NOW(),
    actualizado_en TIMESTAMPTZ DEFAULT NOW()
);

ALTER TABLE public.carritos ENABLE ROW LEVEL SECURITY;

-- 6. TABLA DE ITEMS DEL CARRITO
CREATE TABLE public.items_carrito (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    carrito_id UUID NOT NULL REFERENCES public.carritos (usuario_id) ON DELETE CASCADE,
    producto_id UUID NOT NULL REFERENCES public.productos (id) ON DELETE RESTRICT,
    cantidad INTEGER NOT NULL CHECK (cantidad > 0),
    UNIQUE (carrito_id, producto_id)
);

ALTER TABLE public.items_carrito ENABLE ROW LEVEL SECURITY;

-- 7. TABLA DE PAGOS
CREATE TABLE public.pagos (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    usuario_id UUID NOT NULL REFERENCES auth.users(id) ON DELETE CASCADE,
    total DECIMAL(10, 2) NOT NULL,
    moneda TEXT DEFAULT 'MXN',
    tipo_tarjeta TEXT CHECK (tipo_tarjeta IN ('Crédito', 'Débito')),
    ultimos_4_digitos TEXT,
    nombre_titular TEXT,
    estado TEXT DEFAULT 'pendiente' CHECK (estado IN ('pendiente', 'procesando', 'completado',
'fallido', 'reembolsado')),
    fecha_pago TIMESTAMPTZ DEFAULT NOW(),
    created_at TIMESTAMPTZ DEFAULT NOW()
);

ALTER TABLE public.pagos ENABLE ROW LEVEL SECURITY;

-- 8. TABLA DE PEDIDOS
CREATE TABLE public.pedidos (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    pago_id UUID NOT NULL REFERENCES public.pagos(id) ON DELETE CASCADE,
    usuario_id UUID NOT NULL REFERENCES public.perfiles(id) ON DELETE RESTRICT,
    direccion_envio_id UUID REFERENCES public.direcciones_envio(id) ON DELETE RESTRICT,
    estado TEXT DEFAULT 'pendiente' CHECK (estado IN ('pendiente', 'confirmado', 'enviado',
'entregado', 'cancelado')),
    numero_rastreo TEXT,
    numero_pedido TEXT UNIQUE,
    fecha_pedido TIMESTAMPTZ DEFAULT NOW(),
    created_at TIMESTAMPTZ DEFAULT NOW()
);

ALTER TABLE public.pedidos ENABLE ROW LEVEL SECURITY;

-- 9. TABLA DE ITEMS DEL PEDIDO
CREATE TABLE public.items_pedido (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    pedido_id UUID NOT NULL REFERENCES public.pedidos(id) ON DELETE CASCADE,
    producto_id UUID NOT NULL REFERENCES public.productos(id) ON DELETE RESTRICT,
    cantidad INTEGER NOT NULL CHECK (cantidad > 0),
    precio_unitario DECIMAL(10, 2) NOT NULL,
    subtotal DECIMAL(10, 2) NOT NULL,
    nombre_producto TEXT NOT NULL,
```

```
    descripcion_producto TEXT,
    created_at TIMESTAMPTZ DEFAULT NOW()
);

ALTER TABLE public.items_pedido ENABLE ROW LEVEL SECURITY;

-- ============================================
-- ÍNDICES PARA OPTIMIZACIÓN
-- ============================================

CREATE INDEX idx_productos_categoria ON public.productos(categoria_id);
CREATE INDEX idx_productos_activo ON public.productos(esta_activo);
CREATE INDEX idx_items_carrito_carrito ON public.items_carrito(carrito_id);
CREATE INDEX idx_items_carrito_producto ON public.items_carrito(producto_id);
CREATE INDEX idx_pedidos_usuario ON public.pedidos(usuario_id);
CREATE INDEX idx_pedidos_pago ON public.pedidos(pago_id);
CREATE INDEX idx_pagos_usuario ON public.pagos(usuario_id);

-- ============================================
-- TRIGGER: Crear perfil automáticamente
-- ============================================

CREATE OR REPLACE FUNCTION public.handle_new_user()
RETURNS trigger AS $$
BEGIN
  INSERT INTO public.perfiles (id, nombre_completo, rol)
  VALUES (new.id, new.raw_user_meta_data->>'nombre_completo', 'user');
  RETURN new;
END;
$$ LANGUAGE plpgsql SECURITY DEFINER;

CREATE TRIGGER on_auth_user_created
AFTER INSERT ON auth.users
FOR EACH ROW
EXECUTE FUNCTION public.handle_new_user();

-- ============================================
-- FUNCIÓN: Generar número de pedido único
-- ============================================

CREATE OR REPLACE FUNCTION generar_numero_pedido()
RETURNS TEXT AS $$
DECLARE
    nuevo_numero TEXT;
    existe BOOLEAN;
BEGIN
    LOOP
        nuevo_numero := 'ORD' || TO_CHAR(NOW(), 'YYYYMMDD') || LPAD(FLOOR(RANDOM() * 10000)::TEXT, 4,
'0');
        SELECT EXISTS(SELECT 1 FROM public.pedidos WHERE numero_pedido = nuevo_numero) INTO existe;
        IF NOT existe THEN
            EXIT;
        END IF;
    END LOOP;
    RETURN nuevo_numero;
END;
$$ LANGUAGE plpgsql SECURITY DEFINER;
```

Políticas de seguridad

```
-- ============================================
-- POLÍTICAS PARA PERFILES
-- ============================================
```

```sql
CREATE POLICY "Usuarios ven su perfil"
ON public.perfiles FOR SELECT
USING (auth.uid() = id);

CREATE POLICY "Usuarios actualizan su perfil"
ON public.perfiles FOR UPDATE
USING (auth.uid() = id)
WITH CHECK (auth.uid() = id);

CREATE POLICY "Admin acceso total a perfiles"
ON public.perfiles FOR ALL
USING (
  EXISTS (
    SELECT 1 FROM public.perfiles
    WHERE id = auth.uid() AND rol = 'admin'
  )
);

-- ============================================
-- POLÍTICAS PARA CATEGORÍAS
-- ============================================

CREATE POLICY "Público ve categorías activas"
ON public.categorias FOR SELECT
USING (esta_activa = TRUE);

CREATE POLICY "Admins gestionan categorías"
ON public.categorias FOR ALL
USING (
  EXISTS (
    SELECT 1 FROM public.perfiles
    WHERE id = auth.uid() AND rol = 'admin'
  )
);

-- ============================================
-- POLÍTICAS PARA PRODUCTOS
-- ============================================

CREATE POLICY "Público ve productos activos"
ON public.productos FOR SELECT
USING (
  esta_activo = TRUE
  OR
  EXISTS (
    SELECT 1 FROM public.perfiles
    WHERE id = auth.uid() AND rol = 'admin'
  )
);

CREATE POLICY "Admins gestionan productos"
ON public.productos FOR ALL
USING (
  EXISTS (
    SELECT 1 FROM public.perfiles
    WHERE id = auth.uid() AND rol = 'admin'
  )
);

-- ============================================
-- POLÍTICAS PARA CARRITOS
-- ============================================
```

```sql
CREATE POLICY "Usuario gestiona su carrito"
ON public.carritos FOR ALL
USING (auth.uid() = usuario_id)
WITH CHECK (auth.uid() = usuario_id);

-- ============================================
-- POLÍTICAS PARA ITEMS DEL CARRITO
-- ============================================

CREATE POLICY "Usuario gestiona sus items del carrito"
ON public.items_carrito FOR ALL
USING (
  EXISTS (
    SELECT 1 FROM public.carritos c
    WHERE c.usuario_id = auth.uid() AND c.usuario_id = carrito_id
  )
)
WITH CHECK (
  EXISTS (
    SELECT 1 FROM public.carritos c
    WHERE c.usuario_id = auth.uid() AND c.usuario_id = carrito_id
  )
);

-- ============================================
-- POLÍTICAS PARA DIRECCIONES DE ENVÍO
-- ============================================

CREATE POLICY "Usuario gestiona sus direcciones"
ON public.direcciones_envio FOR ALL
USING (auth.uid() = usuario_id)
WITH CHECK (auth.uid() = usuario_id);

-- ============================================
-- POLÍTICAS PARA PAGOS
-- ============================================

CREATE POLICY "Usuarios ven sus pagos"
ON public.pagos FOR SELECT
USING (auth.uid() = usuario_id);

CREATE POLICY "Usuarios crean pagos"
ON public.pagos FOR INSERT
WITH CHECK (auth.uid() = usuario_id);

CREATE POLICY "Admins actualizan pagos"
ON public.pagos FOR UPDATE
USING (
  EXISTS (
    SELECT 1 FROM public.perfiles
    WHERE id = auth.uid() AND rol = 'admin'
  )
);

-- ============================================
-- POLÍTICAS PARA PEDIDOS
-- ============================================

CREATE POLICY "Usuarios ven sus pedidos"
ON public.pedidos FOR SELECT
USING (auth.uid() = usuario_id);

CREATE POLICY "Usuarios crean pedidos"
ON public.pedidos FOR INSERT
```

```sql
WITH CHECK (auth.uid() = usuario_id);

CREATE POLICY "Admins ven todos los pedidos"
ON public.pedidos FOR SELECT
USING (
  EXISTS (
    SELECT 1 FROM public.perfiles
    WHERE id = auth.uid() AND rol = 'admin'
  )
);

CREATE POLICY "Admins actualizan pedidos"
ON public.pedidos FOR UPDATE
USING (
  EXISTS (
    SELECT 1 FROM public.perfiles
    WHERE id = auth.uid() AND rol = 'admin'
  )
);

-- ============================================
-- POLÍTICAS PARA ITEMS DEL PEDIDO
-- ============================================

CREATE POLICY "Usuarios ven items de sus pedidos"
ON public.items_pedido FOR SELECT
USING (
  EXISTS (
    SELECT 1 FROM public.pedidos
    WHERE id = items_pedido.pedido_id AND usuario_id = auth.uid()
  )
);

CREATE POLICY "Usuarios crean items de pedido"
ON public.items_pedido FOR INSERT
WITH CHECK (
  EXISTS (
    SELECT 1 FROM public.pedidos
    WHERE id = items_pedido.pedido_id AND usuario_id = auth.uid()
  )
);
```

Datos de prueba

```sql
-- CATEGORÍAS
INSERT INTO public.categorias (nombre, descripcion, esta_activa) VALUES
  ('Trabajo', 'Motocicletas diseñadas para trabajo y transporte urbano', TRUE),
  ('Deportivas', 'Motos de alto rendimiento ideales para velocidad y pista', TRUE),
  ('Doble propósito', 'Motos versátiles para ciudad y caminos difíciles', TRUE);

-- Categoría: Trabajo
INSERT INTO public.productos (categoria_id, nombre, descripcion, precio, stock, url_imagen,
esta_activo) VALUES
  ('0afe8ab2-241d-4809-a25c-3eed89da6b4a', 'Honda Cargo 150', 'Moto robusta ideal para reparto y
trabajo diario.', 52000, 5,
'https://aznibghvqqbkrpgoutrw.supabase.co/storage/v1/object/public/Imagenes_Motos/Honda%20Cargo%20150.p
ng', TRUE),
  ('0afe8ab2-241d-4809-a25c-3eed89da6b4a', 'Italika FT150 TS', 'Económica, resistente y de bajo
mantenimiento para el trabajo urbano.', 28500, 8,
'https://aznibghvqqbkrpgoutrw.supabase.co/storage/v1/object/public/Imagenes_Motos/Italika%20FT150%20TS.
png', TRUE),
  ('0afe8ab2-241d-4809-a25c-3eed89da6b4a', 'Suzuki AX100', 'Clásica moto de trabajo con excelente
rendimiento y durabilidad.', 31000, 6,
```

```
'https://aznibghvqqbkrpgoutrw.supabase.co/storage/v1/object/public/Imagenes_Motos/Suzuki%20AX100.png',
TRUE),
  ('0afe8ab2-241d-4809-a25c-3eed89da6b4a', 'Yamaha Crypton 110', 'Motocicleta ligera y confiable,
perfecta para mensajería.', 34000, 7,
'https://aznibghvqqbkrpgoutrw.supabase.co/storage/v1/object/public/Imagenes_Motos/Yamaha%20Crypton%2011
0.png', TRUE),
  -- Categoría: Deportivas
  ('2e531f5f-271d-4cf2-92cb-765d8c71a43f', 'Yamaha R6', 'Moto deportiva de alto rendimiento, ideal para
velocidad en pista.', 260000, 3,
'https://aznibghvqqbkrpgoutrw.supabase.co/storage/v1/object/public/Imagenes_Motos/Yamaha%20R6.png',
TRUE),
  ('2e531f5f-271d-4cf2-92cb-765d8c71a43f', 'Kawasaki Ninja ZX-6R', 'Potente motor y diseño aerodinámico
para máxima adrenalina.', 270000, 2,
'https://aznibghvqqbkrpgoutrw.supabase.co/storage/v1/object/public/Imagenes_Motos/Kawasaki%20Ninja%20ZX
-6R.png', TRUE),
  ('2e531f5f-271d-4cf2-92cb-765d8c71a43f', 'Honda CBR650R', 'Estilo agresivo con excelente control y
confort.', 245000, 4,
'https://aznibghvqqbkrpgoutrw.supabase.co/storage/v1/object/public/Imagenes_Motos/Honda%20CBR650R.png',
TRUE),
  ('2e531f5f-271d-4cf2-92cb-765d8c71a43f', 'KTM Duke 390', 'Deportiva ligera con gran potencia para
ciudad o pista.', 130000, 5,
'https://aznibghvqqbkrpgoutrw.supabase.co/storage/v1/object/public/Imagenes_Motos/KTM%20Duke%20390.png'
, TRUE),
  -- Categoría: Doble propósito
  ('389aa4d9-f6e6-4d3c-9007-bafa7d45aba7', 'BMW G 310 GS', 'Moto doble propósito ideal para aventuras
dentro y fuera de la ciudad.', 170000, 3,
'https://aznibghvqqbkrpgoutrw.supabase.co/storage/v1/object/public/Imagenes_Motos/BMW%20G%20310%20GS.pn
g', TRUE),
  ('389aa4d9-f6e6-4d3c-9007-bafa7d45aba7', 'Honda XRE 300', 'Perfecta para caminos difíciles y
recorridos largos.', 155000, 4,
'https://aznibghvqqbkrpgoutrw.supabase.co/storage/v1/object/public/Imagenes_Motos/Honda%20XRE%20300.png
', TRUE);
```