Contents

GUÍA TÉCNICA DE CONFIGURACIÓN Y DEPLOYMENT

CERTUS Drive - Sistema de Gestión de Recursos Administrativos

---

ÍNDICE

REQUISITOS PREVIOS

Cuentas Necesarias

Servicio	Propósito	Plan	Costo
GitHub	Control de versiones	Free	Gratis
Supabase	Base de datos + Auth	Free	Gratis
Netlify	Hosting + CDN	Free	Gratis
Google Drive	Almacenamiento archivos	Institucional	Incluido

Herramientas de Desarrollo

```
# Git
git --version  # 2.30 o superior

# Editor de código (cualquiera de estos)
# - VS Code (recomendado)
# - Sublime Text
# - Atom

# Navegador moderno
# - Chrome (recomendado para dev tools)
```

```
# - Firefox
# - Edge
```

Conocimientos Técnicos Básicos

- HTML/CSS/JavaScript
- Git y GitHub
- SQL básico (PostgreSQL)
- Conceptos de REST APIs
- Línea de comandos básica

CONFIGURACIÓN DE SUPABASE

Paso 1: Crear Proyecto en Supabase

1.1 Registro

1. Ir a https://supabase.com
2. Hacer clic en **"Start your project"**
3. Registrarse con GitHub (recomendado) o Email
4. Confirmar email si es necesario

1.2 Crear Nuevo Proyecto

1. En el dashboard, hacer clic en **"New Project"**
2. Completar el formulario:

```
Project name: certus-drive
Database Password: [generar password seguro - GUARDAR]
Region: South America (São Paulo) - o más cercano
Pricing Plan: Free
```

3. Hacer clic en **"Create new project"**
4. Esperar 2-3 minutos mientras se provisiona

1.3 Obtener Credenciales Una vez creado el proyecto:

1. Ir a **Settings** → **API**
2. Copiar y guardar:
   - **Project URL**: `https://[tu-proyecto].supabase.co`
   - **anon/public key**: Clave larga que empieza con `eyJ...`

IMPORTANTE: Guardar estas credenciales en un lugar seguro.

Paso 2: Configurar Base de Datos

2.1 Crear Tabla `areas`

1. En Supabase Dashboard, ir a **SQL Editor**
2. Hacer clic en **"New query"**

3. Copiar y pegar este script:

```sql
-- ============================================
-- CREAR TABLA AREAS
-- ============================================

CREATE TABLE IF NOT EXISTS public.areas (
    id UUID DEFAULT gen_random_uuid() PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL UNIQUE,
    correo VARCHAR(255),
    descripcion TEXT,
    icono TEXT,
    orden INTEGER NOT NULL,
    activo BOOLEAN DEFAULT true,
    created_at TIMESTAMPTZ DEFAULT now(),
    updated_at TIMESTAMPTZ DEFAULT now()
);

-- Índices para performance
CREATE UNIQUE INDEX idx_areas_nombre ON public.areas(nombre);
CREATE INDEX idx_areas_orden ON public.areas(orden);
CREATE INDEX idx_areas_activo ON public.areas(activo);

-- Comentarios
COMMENT ON TABLE public.areas IS 'Áreas operativas de CERTUS';
COMMENT ON COLUMN public.areas.nombre IS 'Nombre único del área';
COMMENT ON COLUMN public.areas.correo IS 'Email de contacto del área';
COMMENT ON COLUMN public.areas.icono IS 'Emoji o ícono representativo';
COMMENT ON COLUMN public.areas.orden IS 'Orden de visualización en el carrusel';
```

4. Hacer clic en **"Run"** (o F5)
5. Verificar mensaje de éxito

```sql
-- ============================================
-- DATOS INICIALES: 8 ÁREAS DE CERTUS
-- ============================================

INSERT INTO public.areas (nombre, correo, descripcion, icono, orden, activo)
VALUES
('ATENCIÓN - FRONT', 'atencion.front@certus.edu.pe', 'Atención presencial al personal', ' ', 1,
('ATENCIÓN - CANALES', 'atencion.canales@certus.edu.pe', 'Atención por canales digitales', ' ',
('CRÉDITO Y COBRANZAS', 'credito.cobranzas@certus.edu.pe', 'Gestión de créditos institucionales
('FACTURACIÓN', 'facturacion@certus.edu.pe', 'Facturación y comprobantes', ' ', 4, true),
('RR.AA', 'recursos.academicos@certus.edu.pe', 'Recursos Administrativos', ' ', 5, true),
('PEC', 'pec@certus.edu.pe', 'Programa de Educación Continua', ' ', 6, true),
('REINGRESO', 'reingreso@certus.edu.pe', 'Gestión de reingresos', ' ', 7, true),
('OPERACIONES', 'operaciones@certus.edu.pe', 'Operaciones generales', ' ', 8, true)
```

```
ON CONFLICT (nombre) DO NOTHING;
```

2.2 Insertar Datos Iniciales de Áreas Ejecutar query y verificar que se insertaron 8 filas.

```
-- ============================================
-- CREAR TABLA RECURSOS
-- ============================================

CREATE TABLE IF NOT EXISTS public.recursos (
    id UUID DEFAULT gen_random_uuid() PRIMARY KEY,
    area_id UUID REFERENCES public.areas(id) ON DELETE SET NULL ON UPDATE CASCADE,
    titulo VARCHAR(255) NOT NULL,
    descripcion TEXT,
    enlace TEXT NOT NULL,
    palabras_clave TEXT[],
    is_active BOOLEAN DEFAULT true,
    created_by UUID REFERENCES auth.users(id) ON DELETE SET NULL,
    created_at TIMESTAMPTZ DEFAULT now(),
    updated_at TIMESTAMPTZ DEFAULT now()
);

-- Índices
CREATE INDEX idx_recursos_area_id ON public.recursos(area_id);
CREATE INDEX idx_recursos_is_active ON public.recursos(is_active);
CREATE INDEX idx_recursos_created_at ON public.recursos(created_at DESC);
CREATE INDEX idx_recursos_created_by ON public.recursos(created_by);

-- Trigger para updated_at
CREATE OR REPLACE FUNCTION update_updated_at_column()
RETURNS TRIGGER AS $$
BEGIN
    NEW.updated_at = now();
    RETURN NEW;
END;
$$ language 'plpgsql';

CREATE TRIGGER update_recursos_updated_at BEFORE UPDATE
    ON public.recursos FOR EACH ROW
    EXECUTE PROCEDURE update_updated_at_column();

CREATE TRIGGER update_areas_updated_at BEFORE UPDATE
    ON public.areas FOR EACH ROW
    EXECUTE PROCEDURE update_updated_at_column();

-- Comentarios
COMMENT ON TABLE public.recursos IS 'Recursos institucionales internos de Google Drive';
```

```sql
COMMENT ON COLUMN public.recursos.area_id IS 'FK a tabla areas';
COMMENT ON COLUMN public.recursos.titulo IS 'Título del recurso';
COMMENT ON COLUMN public.recursos.enlace IS 'URL de Google Drive';
COMMENT ON COLUMN public.recursos.palabras_clave IS 'Array de tags para búsqueda';
```

2.3 Crear Tabla `recursos` Ejecutar y verificar.

```sql
-- ============================================
-- CREAR TABLA PROFILES
-- ============================================

CREATE TABLE IF NOT EXISTS public.profiles (
    id UUID REFERENCES auth.users(id) ON DELETE CASCADE PRIMARY KEY,
    email VARCHAR(255) NOT NULL,
    full_name VARCHAR(255),
    role VARCHAR(20) DEFAULT 'user' CHECK (role IN ('admin', 'user')),
    created_at TIMESTAMPTZ DEFAULT now()
);

-- Índices
CREATE INDEX idx_profiles_role ON public.profiles(role);
CREATE INDEX idx_profiles_email ON public.profiles(email);

-- Trigger para crear profile automáticamente al registrar usuario
CREATE OR REPLACE FUNCTION public.handle_new_user()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO public.profiles (id, email, full_name)
    VALUES (NEW.id, NEW.email, NEW.raw_user_meta_data->>'full_name');
    RETURN NEW;
END;
$$ LANGUAGE plpgsql SECURITY DEFINER;

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

-- Comentarios
COMMENT ON TABLE public.profiles IS 'Perfiles extendidos de usuarios';
COMMENT ON COLUMN public.profiles.role IS 'Rol del usuario: admin o user';
```

2.4 Crear Tabla `profiles`

Paso 3: Configurar Row Level Security (RLS)

```sql
-- ============================================
-- HABILITAR RLS EN TODAS LAS TABLAS
-- ============================================

ALTER TABLE public.recursos ENABLE ROW LEVEL SECURITY;
ALTER TABLE public.areas ENABLE ROW LEVEL SECURITY;
ALTER TABLE public.profiles ENABLE ROW LEVEL SECURITY;
```

3.1 Habilitar RLS

```sql
-- ============================================
-- POLÍTICAS RLS: RECURSOS
-- ============================================

-- Política 1: Lectura pública de recursos activos
CREATE POLICY "Public read active recursos"
ON public.recursos FOR SELECT
USING (is_active = true);

-- Política 2: Admin tiene acceso total
CREATE POLICY "Admin full access recursos"
ON public.recursos FOR ALL
USING (
    EXISTS (
        SELECT 1 FROM public.profiles
        WHERE profiles.id = auth.uid()
        AND profiles.role = 'admin'
    )
);

-- Política 3: Usuarios autenticados pueden insertar
-- (Opcional - comentar si solo admin debe crear)
CREATE POLICY "Authenticated users can insert recursos"
ON public.recursos FOR INSERT
WITH CHECK (auth.role() = 'authenticated');
```

3.2 Políticas para Tabla recursos

```sql
-- ============================================
-- POLÍTICAS RLS: AREAS
-- ============================================

-- Política 1: Lectura pública de áreas activas
CREATE POLICY "Public read active areas"
ON public.areas FOR SELECT
USING (activo = true);
```

```sql
-- Política 2: Admin tiene acceso total
CREATE POLICY "Admin full access areas"
ON public.areas FOR ALL
USING (
    EXISTS (
        SELECT 1 FROM public.profiles
        WHERE profiles.id = auth.uid()
        AND profiles.role = 'admin'
    )
);
```

3.3 Políticas para Tabla areas

```sql
-- ============================================
-- POLÍTICAS RLS: PROFILES
-- ============================================

-- Política 1: Usuarios ven su propio perfil
CREATE POLICY "Users can view own profile"
ON public.profiles FOR SELECT
USING (auth.uid() = id);

-- Política 2: Admin ve todos los perfiles
CREATE POLICY "Admin can view all profiles"
ON public.profiles FOR SELECT
USING (
    EXISTS (
        SELECT 1 FROM public.profiles
        WHERE profiles.id = auth.uid()
        AND profiles.role = 'admin'
    )
);

-- Política 3: Usuarios pueden actualizar su perfil
CREATE POLICY "Users can update own profile"
ON public.profiles FOR UPDATE
USING (auth.uid() = id);
```

3.4 Políticas para Tabla profiles

Paso 4: Crear Usuario Administrador

4.1 Registrar Usuario por Email

1. En Supabase Dashboard, ir a **Authentication** → **Users**
2. Hacer clic en **"Add user"** → **"Create new user"**

3. Completar:

   ```
   Email: admin@certus.edu.pe
   Password: [contraseña segura]
   Auto Confirm User:  (marcar)
   ```

4. Hacer clic en **"Create user"**

4.2 Asignar Rol Admin

1. Ir a **SQL Editor**
2. Ejecutar:

```sql
-- Actualizar rol del usuario a admin
UPDATE public.profiles
SET role = 'admin'
WHERE email = 'admin@certus.edu.pe';

-- Verificar
SELECT id, email, role FROM public.profiles WHERE role = 'admin';
```

Paso 5: Verificar Configuración

```sql
-- Debe retornar las 8 áreas
SELECT * FROM public.areas WHERE activo = true;

-- No debe retornar nada si no hay recursos todavía
SELECT * FROM public.recursos WHERE is_active = true;
```

5.1 Test de Lectura Pública

5.2 Test de Permisos Ir a **Settings** → **API** → **API URL** y hacer una prueba con curl:

```bash
# Test: Leer áreas (debe funcionar)
curl 'https://[tu-proyecto].supabase.co/rest/v1/areas?select=*' \
  -H "apikey: [tu-anon-key]" \
  -H "Authorization: Bearer [tu-anon-key]"

# Debe retornar JSON con las 8 áreas
```

CONFIGURACIÓN DEL PROYECTO

Paso 1: Clonar o Descargar el Repositorio

```bash
# Abrir terminal y ejecutar
git clone https://github.com/certusdrives-gif/CERTUSDRIVES.git
```

```
# Entrar al directorio
cd CERTUSDRIVES/temp-deploy
```

Opción A: Clonar desde GitHub

Opción B: Descargar ZIP

1. Ir al repositorio en GitHub
2. Hacer clic en **Code** → **Download ZIP**
3. Extraer el ZIP
4. Abrir la carpeta `temp-deploy`

Paso 2: Configurar Variables de Entorno

2.1 Crear archivo de configuración **Archivo**: `js/config.js`

```
// ============================================
// CONFIGURACIÓN DE SUPABASE
// ============================================

const supabaseUrl = 'https://[TU-PROYECTO].supabase.co'
const supabaseKey = '[TU-ANON-KEY-AQUI]'

// Inicializar cliente de Supabase
const supabase = window.supabase.createClient(supabaseUrl, supabaseKey)

// Exportar para uso global
window.supabase = supabase
```

IMPORTANTE: - Reemplazar `[TU-PROYECTO]` con el nombre de tu proyecto Supabase - Reemplazar `[TU-ANON-KEY-AQUI]` con la clave anon obtenida anteriormente

2.2 Archivo `.env.example` Ya existe en el proyecto como plantilla:

```
# .env.example
SUPABASE_URL=tu_supabase_url_aqui
SUPABASE_ANON_KEY=tu_supabase_anon_key_aqui
```

Nota: Este archivo es solo referencia. La configuración real está en `js/config.js`.

Paso 3: Verificar Estructura de Archivos

```
temp-deploy/
  index.html          Debe existir
  admin.html          Debe existir
  login.html          Debe existir
  netlify.toml        Debe existir
  _redirects          Debe existir
  css/
```

```
    styles.css
    admin.css
  js/
    config.js        Configurar
    auth.js
    login.js
    script.js
    admin.js
    areas.js
  assets/
     logo.svg
     favicon.ico
```

Paso 4: Prueba Local (Opcional)

```
# Python 3
python -m http.server 8000

# Abrir navegador en: http://localhost:8000
```

4.1 Usando Python

```
# Instalar http-server globalmente
npm install -g http-server

# Ejecutar
http-server -p 8000

# Abrir navegador en: http://localhost:8000
```

4.2 Usando Node.js

4.3 Usando VS Code

1. Instalar extensión **"Live Server"**
2. Clic derecho en `index.html`
3. Seleccionar **"Open with Live Server"**

DEPLOYMENT EN NETLIFY

Método 1: Deploy desde GitHub (Recomendado)

```
# Si aún no tienes repositorio
git init
git add .
```

```
git commit -m "Initial commit: CERTUS Drive"

# Crear repositorio en GitHub.com
# Luego conectar y push
git remote add origin https://github.com/[tu-usuario]/certus-drive.git
git branch -M main
git push -u origin main
```

Paso 1: Subir Código a GitHub

Paso 2: Conectar Netlify con GitHub

1. Ir a https://app.netlify.com
2. Registrarse o iniciar sesión (usar cuenta de GitHub)
3. Hacer clic en **"Add new site"** → **"Import an existing project"**
4. Seleccionar **"GitHub"**
5. Autorizar Netlify a acceder a GitHub
6. Buscar y seleccionar tu repositorio `certus-drive`

Paso 3: Configurar Build Settings

```
Branch to deploy: main
Base directory: temp-deploy
Build command: [dejar vacío]
Publish directory: . (punto)
```

Hacer clic en **"Deploy site"**

Paso 4: Esperar Deploy

- El deploy toma 30-60 segundos
- Verás el progreso en tiempo real
- Al finalizar, obtendrás una URL como: `https://random-name-123.netlify.app`

Paso 5: Configurar Dominio Custom (Opcional)

1. En Netlify, ir a **Site settings** → **Domain management**
2. Hacer clic en **"Add custom domain"**
3. Ingresar dominio (ej: `drive.certus.edu.pe`)
4. Seguir instrucciones para configurar DNS
5. Netlify provee SSL automático

Método 2: Deploy Manual (Drag & Drop)

Paso 1: Preparar Archivos

1. Asegurarse de que `js/config.js` esté configurado
2. Verificar que todos los archivos estén en la carpeta `temp-deploy`

Paso 2: Deploy

1. Ir a https://app.netlify.com
2. Arrastrar la carpeta `temp-deploy` a la zona de **"Drag and drop"**
3. Esperar a que suba
4. Netlify asigna URL automáticamente

Limitaciones del método manual: - No hay deploy automático - Cada cambio requiere subir de nuevo - No hay historial de versiones

Método 3: Deploy por CLI

```
npm install -g netlify-cli
```

Paso 1: Instalar Netlify CLI

```
netlify login
```

Paso 2: Login

```
# Entrar a la carpeta del proyecto
cd temp-deploy

# Deploy en borrador (testing)
netlify deploy

# Deploy en producción
netlify deploy --prod
```

Paso 3: Deploy

Verificar Deployment

Checklist Post-Deploy **Página principal carga**: Ir a URL de Netlify
Áreas se muestran: Verificar carrusel con 8 áreas
Login funciona: Ir a `/login.html` y autenticarse
Panel admin accesible: Ir a `/admin.html`
HTTPS activo: Verificar candado en navegador
No hay errores en consola: F12 → Console

Troubleshooting Deploy **Problema: 404 en rutas** - Verificar que existe `netlify.toml` y `_redirects` - Configurar redirects en Netlify UI

Problema: Variables no definidas - Verificar `js/config.js` con credenciales correctas - No usar variables de entorno del sistema (Netlify no las lee para frontend)

Problema: CSS/JS no cargan - Verificar rutas relativas en HTML - Deben ser `/css/...` no `./css/...`

CONFIGURACIÓN DE GOOGLE DRIVE

Paso 1: Organizar Estructura de Carpetas

1.1 Crear Carpeta Principal

```
CERTUS Drive/
```

1.2 Crear Subcarpetas por Área

```
CERTUS Drive/
   ATENCIÓN - FRONT/
   ATENCIÓN - CANALES/
   CRÉDITO Y COBRANZAS/
   FACTURACIÓN/
   RR.AA/
   PEC/
   REINGRESO/
   OPERACIONES/
```

Paso 2: Configurar Permisos

2.1 Permisos Recomendados **Para carpetas compartidas:** - **Propietario**: Cuenta institucional CERTUS - **Editores**: Coordinadores de área - **Lectores**: Todo el personal

Para archivos específicos: - Decidir caso por caso - Usar enlace "Cualquiera con el enlace puede ver"

2.2 Obtener Enlaces Compartidos

1. Abrir archivo/carpeta en Drive
2. Clic derecho → **"Obtener enlace"**
3. Cambiar a **"Cualquiera con el enlace puede ver"**
4. Copiar enlace
5. Pegar en el sistema al crear recurso

Formato de enlaces:

```
Archivo: https://drive.google.com/file/d/1ABC123xyz/view
Carpeta: https://drive.google.com/drive/folders/1ABC123xyz
```

Paso 3: Mejores Prácticas

Nomenclatura consistente: `[Área] - [Tipo] - [Nombre] - [Año]`
Ejemplo: `FACTURACION - Manual - Emisión Recibos - 2025.pdf`

No cambiar enlaces: Una vez compartido, no mover el archivo

Versioning: Usar "v1", "v2" en nombre si hay actualizaciones

Revisar permisos: Auditar cada 6 meses

VARIABLES DE ENTORNO

Frontend (Público)

Archivo: `js/config.js`

```
const supabaseUrl = 'https://[proyecto].supabase.co'
const supabaseKey = 'eyJ[...]' // anon/public key
```

Seguridad: - La `anon key` es segura para exponer en frontend - RLS protege los datos en el backend - NUNCA exponer `service_role key`

Variables en Netlify (Opcional)

Si quieres usar environment variables:

1. Ir a Netlify Dashboard
2. **Site settings → Build & deploy → Environment**
3. Agregar variables:

   ```
   SUPABASE_URL=https://[proyecto].supabase.co
   SUPABASE_ANON_KEY=eyJ[...]
   ```

Nota: Para frontend estático, es más simple usar `config.js` directamente.

TESTING Y VALIDACIÓN

Test 1: Conexión a Supabase

Consola del navegador (F12):

```
// Test conexión
console.log('Supabase URL:', supabase.supabaseUrl)

// Test lectura de áreas
const { data, error } = await supabase
  .from('areas')
  .select('*')

console.log('Áreas:', data)
console.log('Error:', error)
```

Resultado esperado: Array con 8 áreas, error = null

Test 2: Autenticación

1. Ir a `/login.html`
2. Ingresar credenciales de admin

3. Verificar redirección a `/admin.html`
4. Abrir consola:

```
const { data: { session } } = await supabase.auth.getSession()
console.log('Usuario:', session.user.email)
console.log('Rol:', session.user.role)
```

Test 3: RLS Policies

```
// Debe funcionar - lectura de áreas activas
const { data } = await supabase.from('areas').select('*')
console.log('Áreas públicas:', data) //  8 áreas

// Debe fallar - insertar área sin permisos
const { error } = await supabase
  .from('areas')
  .insert([{ nombre: 'TEST', orden: 99 }])
console.log('Error esperado:', error) //  Permission denied
```

Test como usuario público (sin login)

```
// Login primero
await supabase.auth.signInWithPassword({
  email: 'admin@certus.edu.pe',
  password: 'tu-password'
})

// Debe funcionar - admin puede insertar
const { data, error } = await supabase
  .from('areas')
  .insert([{ nombre: 'AREA TEST', orden: 99, activo: false }])
  .select()
console.log('Área creada:', data) //  Nuevo registro

// Limpiar test
await supabase.from('areas').delete().eq('nombre', 'AREA TEST')
```

Test como admin

Test 4: CRUD Completo

Script de prueba (`test-crud.js`):

```
async function testCRUD() {
  console.log('=== Test CRUD Recursos ===')

  // CREATE
  const { data: created, error: createError } = await supabase
```

```
    .from('recursos')
    .insert([{
      titulo: 'Test Recurso',
      descripcion: 'Descripción de prueba',
      enlace: 'https://drive.google.com/file/d/test/view',
      palabras_clave: ['test', 'prueba'],
      is_active: false // inactivo para no aparecer en producción
    }])
    .select()

  console.log(' CREATE:', created)
  const recursoId = created[0].id

  // READ
  const { data: read } = await supabase
    .from('recursos')
    .select('*')
    .eq('id', recursoId)
    .single()

  console.log(' READ:', read)

  // UPDATE
  const { data: updated } = await supabase
    .from('recursos')
    .update({ titulo: 'Test Actualizado' })
    .eq('id', recursoId)
    .select()

  console.log(' UPDATE:', updated)

  // DELETE
  const { error: deleteError } = await supabase
    .from('recursos')
    .delete()
    .eq('id', recursoId)

  console.log(' DELETE:', deleteError ? 'Error' : 'Success')
}

// Ejecutar test (solo como admin)
testCRUD()
```

Test 5: Performance

Herramientas: - Chrome DevTools → **Lighthouse** - GTmetrix - PageSpeed Insights

Métricas objetivo: - Performance: > 90 - Accessibility: > 95 - Best Practices: > 90 - SEO: > 85

MANTENIMIENTO

Backups de Base de Datos

Opción 1: Manual desde Supabase

1. Ir a Supabase Dashboard
2. **Database → Backups**
3. Hacer clic en **"Download backup"**
4. Se descarga un archivo SQL

Frecuencia recomendada: Semanal

```
# Conectarse a la base de datos
pg_dump "postgresql://postgres:[password]@[host]:5432/postgres" > backup_$(date +%Y%m%d).sql
```

Opción 2: Automatizado con pg_dump

```
-- Export tabla recursos
COPY (SELECT * FROM recursos) TO '/tmp/recursos_backup.csv' CSV HEADER;

-- Export tabla areas
COPY (SELECT * FROM areas) TO '/tmp/areas_backup.csv' CSV HEADER;
```

Opción 3: Export de Tablas

Actualización de Dependencias

Supabase Client Verificar versión actual en HTML:

```
<script src="https://cdn.jsdelivr.net/npm/@supabase/supabase-js@2"></script>
```

Para actualizar a nueva versión:

```
<script src="https://cdn.jsdelivr.net/npm/@supabase/supabase-js@2.x.x"></script>
```

Verificar changelog en GitHub de Supabase

Monitoreo

Netlify Analytics (Opcional - Paid)

- Tráfico en tiempo real
- Geolocalización de usuarios
- Páginas más visitadas

Supabase Logs

1. En Supabase Dashboard: **Logs & Analytics**

2. Ver:
    - Queries ejecutadas
    - Errores de autenticación
    - Uso de API

Google Analytics (Opcional) Agregar en `<head>` de todos los HTML:

```
<!-- Google Analytics -->
<script async src="https://www.googletagmanager.com/gtag/js?id=G-XXXXXXXXXX"></script>
<script>
  window.dataLayer = window.dataLayer || [];
  function gtag(){dataLayer.push(arguments);}
  gtag('js', new Date());
  gtag('config', 'G-XXXXXXXXXX');
</script>
```

Limpieza de Datos

Script de mantenimiento mensual:

```
-- Ver recursos inactivos hace más de 6 meses
SELECT id, titulo, updated_at
FROM recursos
WHERE is_active = false
  AND updated_at < NOW() - INTERVAL '6 months';

-- Opcionalmente eliminarlos
-- DELETE FROM recursos WHERE id IN (...);

-- Ver áreas sin recursos
SELECT a.id, a.nombre, COUNT(r.id) as recursos_count
FROM areas a
LEFT JOIN recursos r ON a.id = r.area_id
GROUP BY a.id, a.nombre
HAVING COUNT(r.id) = 0;
```

TROUBLESHOOTING

Error: "Failed to fetch"

Síntoma: No cargan datos de Supabase

Causas posibles: 1. URL de Supabase incorrecta 2. API key incorrecta 3. CORS bloqueado 4. Red/Firewall

Solución:

```
// Verificar configuración
console.log('URL:', supabase.supabaseUrl)
```

```
console.log('Key:', supabaseKey.substring(0, 20) + '...')

// Test directo
fetch('https://[tu-proyecto].supabase.co/rest/v1/')
  .then(r => r.json())
  .then(console.log)
```

Error: RLS Policies

Síntoma: "new row violates row-level security policy"

Solución: 1. Verificar que el usuario tenga rol correcto 2. Revisar políticas RLS 3. Test con `service_role` key (solo backend/testing)

```
-- Ver políticas activas
SELECT * FROM pg_policies WHERE tablename = 'recursos';
```

Error: "Cannot read property of undefined"

Síntoma: Error al renderizar datos

Causa: JOIN no devuelve datos esperados

Solución:

```
// Usar optional chaining
const nombreArea = recurso.areas?.nombre || 'Sin área'

// Verificar datos en consola
console.log('Recurso completo:', recurso)
```

ESCALABILIDAD

Límites del Plan Free

Supabase Free: - 500 MB database - 1 GB file storage - 2 GB bandwidth/month - 50,000 monthly active users

Netlify Free: - 100 GB bandwidth/month - 300 build minutes/month - Unlimited sites

Cuándo Escalar

Indicadores: - $>$ 10,000 recursos en BD - $>$ 1,000 usuarios activos/mes - $>$ 50 GB bandwidth/mes - Necesidad de real-time

Plan de Escalamiento

Fase 1: Optimización (Free) - Implementar caching - Lazy loading de imágenes - Paginación eficiente - Índices en BD

Fase 2: Upgrade a Pro ($25-50/mes) - Supabase Pro ($25/mes) - Netlify Pro ($19/mes) - Más recursos y soporte

Fase 3: Infraestructura Custom - VPS dedicado - PostgreSQL separado - CDN Premium (Cloudflare Pro) - Load balancer

---

Última actualización: Noviembre 2025
Versión del documento: 1.0
Sistema: CERTUS Drive v1.0