**RESUMEN EJECUTIVO FINAL DISEÑADOR DE CLASES**

**1) Resumen ejecutivo**

* **Objetivo**: Plataforma web para diseñar, publicar y revisar clases, organizada por roles: **Administrador**, **Coordinador** y **Estudiante**.
* **Frontend**: HTML + CSS + JS nativo (sin bundler). Módulos ES6 por rol.
* **Backend**: **Supabase** (PostgreSQL, Row Level Security, RPC).
* **Hosting**: **GitHub Pages** (branch main, carpeta / (root)).
* **Autenticación** por **código** (campo codigo en tabla usuarios), no por sesión tradicional. Se guarda una “*session*” mínima en localStorage.
* **Diseño**: Escala completa de ancho, tipografía del sistema, color primario **azul intenso** y acento **naranja**, componentes reutilizables (tarjetas, botones, badges).
* **Separación fuerte por rol**: cada rol usa sus **propios módulos** (supabaseClient + auth) para evitar colisiones y facilitar cambios independientes.

**2) Estructura de carpetas (root del repo)**

SDCS2026/

├─ administrador/

│ ├─ administrador.html # Acceso por código (admin)

│ ├─ dashboard.html # Dashboard del administrador

│ ├─ adminAuth.js # Auth/Admin (módulo propio)

│ ├─ adminSupabaseClient.js # Supabase client (admin)

│ └─ ... # JS específicos de creación/edición/publicación

│

├─ coordinador/

│ ├─ coordinador.html # Acceso por código (coordinador)

│ ├─ dashboard-coordinador.html # Dashboard de Coordinación

│ ├─ acceso-coordinador.js # Lógica de formulario de acceso (ESM)

│ ├─ coordAuth.js # Auth/Coordinador (ESM)

│ ├─ coordSupabaseClient.js # Supabase client (coordinador, ESM)

│ ├─ coordinador.js # Lógica del dashboard (carga, filtros, modal)

│ ├─ coordinador.css # Estilos específicos del rol

│ └─ ... # Utilidades locales del rol

│

├─ estudiante/

│ ├─ dashboard-estudiante.html

│ ├─ estudiante.css

│ └─ ... # Lógica de vista/recursos del estudiante

│

├─ base/

│ ├─ styles.css # Estilos globales (tokens, layout, botones, etc.)

│ ├─ recursos-viewer.css # Estilos del visor de recursos embebidos

│ └─ ... # Fonts, helpers de estilo, etc.

│

├─ js/ # Utilidades transversales (si aplica)

│ ├─ auth.js # (histórico; NO usado por coord/admin)

│ └─ acceso.js # (histórico estudiantes)

│

├─ assets/ # Íconos, imágenes, logos

├─ DOCS/ # Documentación y checkpoints de fases

└─ index.html (si existe) # (opcional) Home/landing

**Rutas**: Todo el Frontend usa **rutas relativas** (./ o ../) para que GitHub Pages resuelva bien (nunca /ruta/absoluta).

**3) Variables y entornos**

**Supabase** necesita:

* SUPABASE\_URL
* SUPABASE\_ANON\_KEY (rol anon)

Se inyectan en el **cliente por rol** (p. ej., coordSupabaseClient.js), **no** en variables de entorno del build (no hay bundler).  
Ejemplo mínimo:

// coordinador/coordSupabaseClient.js

import { createClient } from 'https://cdn.jsdelivr.net/npm/@supabase/supabase-js@2/+esm';

const SUPABASE\_URL = 'https://<tu-proyecto>.supabase.co';

const SUPABASE\_ANON\_KEY = '<anon-key>';

export const supabase = createClient(SUPABASE\_URL, SUPABASE\_ANON\_KEY, {

auth: { persistSession: false }, // no usamos auth tradicional

});

**Seguridad**: Casi todas las operaciones del rol Coordinador/Estudiante son **lecturas** con RLS; escritura/publicación es responsabilidad del **Administrador**.

**4) Backend (Supabase / PostgreSQL)**

**4.1 Tablas mínimas**

* public.usuarios
  + id (uuid o serial)
  + codigo (text o bigint) ← autenticación por código
  + nombre (text)
  + perfil (text) ∈ {ADMINISTRADOR, COORDINADOR, ESTUDIANTE}
  + Índices:
    - idx\_usuarios\_codigo (btree) sobre codigo
* public.qva (Qué Vamos a Aprender)
  + id, grado, trimestre, titulo, descripcion, publicada (bool), fecha\_publicacion (date)
  + RLS: lectura abierta a anon si publicada = TRUE (o por rol)
* public.proyecto
  + Campos similares (grado, trimestre, publicada, …)
* public.clases
  + id, grado, trimestre, titulo, descripcion, publicada, fecha\_publicacion
  + contenido (cuerpo enriquecido)
  + recursos (jsonb) o tablas auxiliares:
    - recursos\_clase (id, clase\_id, tipo, etiqueta, url, mimetype, orden)

**Nota**: Durante el desarrollo hubo referencias a recursos\_qa/recursos\_proyecto. Si existen, mantener políticas análogas.

**4.2 RLS (Row Level Security)**

* **Usuarios**: RLS activo, sin select directo desde anon (evitar filtrar todo el padrón).
* **Clases/QVA/Proyecto**: permitir select a anon **solo si** publicada = TRUE.

Ejemplo de política:

-- Solo lectura de contenidos publicados:

create policy "anon select clases publicadas"

on public.clases

for select to anon

using (publicada = true);

**4.3 RPC para validación por código**

Para evitar exponer usuarios con RLS la **validación** se hace vía **función SQL** (security definer) y **RPC**:

-- función: valida\_coordinador(text)

create or replace function public.valida\_coordinador(\_codigo text)

returns table (codigo text, nombre text, perfil text)

language sql

security definer

as $$

select u.codigo::text, u.nombre, u.perfil

from public.usuarios u

where u.codigo::text = \_codigo

limit 1;

$$;

-- Permisos de ejecución desde rol anon:

revoke all on function public.valida\_coordinador(text) from public;

grant execute on function public.valida\_coordinador(text) to anon;

**Casting**: el parámetro de la RPC debe **match** el tipo en la tabla (text vs bigint). En el frontend convertimos a Number o String según corresponda.

**5) Flujos por rol**

**5.1 Coordinador**

**Acceso** (coordinador/coordinador.html + acceso-coordinador.js):

* Input 6 dígitos
* coordAuth.js llama a supabase.rpc('valida\_coordinador', { \_codigo })
* Verifica perfil === 'COORDINADOR'
* Guarda **mini-sesión** en localStorage
* Redirige a dashboard-coordinador.html

**Dashboard** (coordinador/dashboard-coordinador.html + coordinador.js):

* Filtros: **Grado**, **Trimestre**, **Estado** (Publicado / Todos), **Búsqueda por texto**
* Tarjetas-resumen (conteos de QVA, Proyectos y Clases)
* Listado de **Clases** con:
  + **Previsualizar** (modal que muestra título, contenido y **recursos embebidos** PDF/IMG/VIDEO sin salir de la plataforma)
  + **Sin publicar**: **NO** se muestra botón publicar (capacidad de publicar es solo del **Administrador**)
* Estilos: base/styles.css + coordinador/coordinador.css (azul intenso y naranja, botones legibles)

**5.2 Administrador**

**Acceso** (administrador/administrador.html + adminAuth.js):

* Similar al coordinador, pero perfiles ADMINISTRADOR

**Dashboard** (administrador/dashboard.html):

* Crear/editar clase: título, contenido enriquecido, recursos (URL o archivo), tags
* Guardar **borrador**
* **Publicar / Despublicar**
* Eliminar clase
* Subida de recursos: si se usan buckets de Supabase Storage, endpoints desde JS y guardar URL en recursos\_clase

**5.3 Estudiante**

**Dashboard** (estudiante/dashboard-estudiante.html):

* Ver clases publicadas por grado y trimestre
* Visualizar contenido
* Abrir recursos en visor embebido

**6) Módulos críticos (resumen)**

**6.1 Acceso coordinador (fragmento)**

// coordinador/acceso-coordinador.js

import { validarAccesoPorCodigoCoord, redirigirCoordinador, setCoordSession } from './coordAuth.js';

const $ = (id) => document.getElementById(id);

const input = $('codigo');

const btn = $('btn');

const msg = $('msg');

function showMsg(text, kind = 'err') {

msg.style.display = text ? 'block' : 'none';

msg.className = 'msg ' + (kind || '');

msg.textContent = text;

}

btn.addEventListener('click', async () => {

showMsg('', '');

btn.disabled = true;

const codigo = (input.value || '').trim();

if (!/^\d{6}$/.test(codigo)) {

showMsg('El código debe tener 6 dígitos.');

return (btn.disabled = false);

}

const { data, error } = await validarAccesoPorCodigoCoord(codigo);

if (error) { showMsg('Error: ' + (error.message || 'No se pudo validar.')); return (btn.disabled = false); }

if (!data) { showMsg('Código no válido.'); return (btn.disabled = false); }

if (String(data.perfil).toUpperCase() !== 'COORDINADOR') {

showMsg(`Acceso denegado: tu perfil es ${data.perfil}.`); return (btn.disabled = false);

}

setCoordSession({ codigo: data.codigo, nombre: data.nombre, perfil: data.perfil });

try { redirigirCoordinador(data.perfil); } catch (e) { showMsg(e.message || 'No autorizado.'); }

finally { btn.disabled = false; }

});

input.addEventListener('keydown', e => (e.key === 'Enter') && btn.click());

**6.2 coordAuth.js (fragmento)**

// coordinador/coordAuth.js

import { supabase } from './coordSupabaseClient.js';

const KEY = 'coord\_session';

export function setCoordSession(session){ localStorage.setItem(KEY, JSON.stringify(session)); }

export function getCoordSession(){ try { return JSON.parse(localStorage.getItem(KEY) || 'null'); } catch { return null; } }

export function clearCoordSession(){ try { localStorage.removeItem(KEY); } catch {} }

export async function validarAccesoPorCodigoCoord(codigo) {

const { data, error } = await supabase.rpc('valida\_coordinador', { \_codigo: String(codigo) });

return { data: data && data[0] ? data[0] : null, error };

}

export function redirigirCoordinador(perfil) {

if (String(perfil).toUpperCase() !== 'COORDINADOR') throw new Error('Perfil no permitido');

location.href = './dashboard-coordinador.html';

}

**Administrador** tiene sus módulos **propios**: adminAuth.js y adminSupabaseClient.js con lógica análoga.

**7) UI/UX y estilos**

* **Grid fluido** con contenedor a 1200–1280px máx, paddings amplios.
* **Tokens de color** (en base/styles.css):
  + --color-primary: #1f4bf2 (azul intenso)
  + --color-accent: #ff7a00 (naranja)
  + --color-bg, --color-surface, --color-muted, etc.
* **Componentes**: .card, .btn, .badge, .input, .select, .modal.
* **Botones**:
  + .btn (base)
  + .btn.primary (fondo azul, texto blanco)
  + .btn.outline (borde azul, texto azul, fondo blanco)
* **Cache busting**: **siempre** añadir ?v=<n> cuando cambias CSS/JS.

**8) Despliegue (GitHub Pages)**

* **Settings → Pages**:
  + **Source**: Deploy from a branch
  + **Branch**: main
  + **Folder**: / (root)
* **Rutas** **SIEMPRE relativas**: href="../base/styles.css" / src="./coordinador.js".
* **No** usar rutas root /algo ni prefijos /SDCS2026/... en HTML.

**Pipeline Git recomendado:**

git fetch origin main

git rebase origin/main # resolver conflictos si aparecen

# (editar archivos con marcas <<<<<<< ======= >>>>>>> si aplica)

git add .

git rebase --continue # si estaba en rebase

git push origin main

**Verificación post-deploy:**

* Abrir: https://artifextsp.github.io/SDCS2026/coordinador/coordinador.html
* Abrir DevTools > Network: **sin 404** en styles.css, coordinador.css, acceso-coordinador.js, coordAuth.js, coordSupabaseClient.js.

**9) Pruebas funcionales (checklist rápido)**

**Acceso**

* Código inválido → mensaje de error
* Código válido con perfil incorrecto → “Acceso denegado…”
* Código válido perfil Coordinador → redirección al dashboard

**Dashboard Coordinador**

* Filtros (grado, trimestre, estado)
* Conteos correctos (QVA, Proyecto, Clases)
* Listado de clases con **Título**, **Descripción**
* **Previsualizar** abre modal con contenido y **recursos embebidos**
* No hay botón de **Publicar** (solo admin)

**Admin**

* Crear/Guardar borrador de clase
* Adjuntar recursos (url y/o archivo)
* **Publicar/Despublicar**
* **Eliminar** (confirmación + borrado de recursos asociados si aplica)

**10) Errores comunes y solución**

* **401 en Supabase**: falta permiso (RLS/policy o GRANT EXECUTE en la función RPC).
* **404 en CSS/JS**: ruta incorrecta o mayúsculas diferentes a las del archivo → usar **rutas relativas** y chequear nombres exactos.
* **Error casting código**: define valida\_coordinador(\_codigo text) y **castea** u.codigo::text en SQL; en frontend envía String(codigo).
* **Página de GitHub muestra UI vieja**: **cambia ?v=** en <link>/<script> y recarga.
* **Merge/push rechazado**: hacer git fetch + git rebase origin/main, resolver conflictos, git push.

**11) Extensiones / Futuras mejoras**

* **Autenticación con OTP** vía email/supabase auth si se quiere endurecer el acceso.
* **Roles finos** por permisos (policy por grado/trimestre).
* **Storage**: mover archivos a buckets de Supabase y firmar URLs temporales.
* **Editor enriquecido** para Admin (markdown/toolbar propia).
* **Analytics** básicos (vistas de clase, uso de recursos).

**12) Referencias rápidas (paths y enlaces clave)**

* **Acceso Coordinador**: coordinador/coordinador.html
* **Dashboard Coordinador**: coordinador/dashboard-coordinador.html
* **Acceso Admin**: administrador/administrador.html
* **Dashboard Admin**: administrador/dashboard.html
* **Estilos globales**: base/styles.css
* **Visor recursos**: base/recursos-viewer.css
* **Supabase (coord)**: coordinador/coordSupabaseClient.js
* **Auth (coord)**: coordinador/coordAuth.js

**13) Plantillas de mensajes/UI (consistencia)**

* Errores de acceso: El código debe tener 6 dígitos., Código no válido., Acceso denegado: tu perfil es XXX.
* Estados: PUBLICADA, BORRADOR
* Botones: Entrar, Ver detalle, Previsualizar, Guardar, Publicar, Despublicar, Eliminar