**FASE 2 PREPARADOR DE CLASE**

**Objetivo**

Diseñar e implementar el **dashboard de administrador** desde el cual se van a preparar las clases de los grados 1° - 11° de robótica + steam, el poryecto de trimestre y el qué vamos a aprender, clases , proyecto y qué vamos a aprender que los estudiantes van a poder visualizar desde cada dashboard de grado 1° - 11°, y que la coordinadora va a poder revisar desde su dash también.

**WorfkFlow**

1. Al acceder a través de la interfaz y código de administrador la primera interfaz que recibe al administrador está compuesta de 3 botones con sus respectivos iconos y con el estilo creado para el sistema estos botones son 1. QUÉ VAMOS A APRENDER 2. PROYECTO 3. CLASES.

Nota: La idea es diseñar una herramienta que permita escribir y adjuntar pdf, link de video, video y audio que son recursos que usamos normalmente en las clases.

1. Al dar clic en el botón CLASES:
   1. Se debe poder seleccionar el grado para el cual se va a crear, editar o eliminar la clase (1° - 11°)
   2. Se debe poder seleccionar el trimestre para el cual se va a crear, editar o eliminar la clase (son tres trimestres)
   3. Ahora deben aparecer 3 botones 1. EDITAR CLASE CREADA 2. DISEÑAR CLASE 3. ELIMINAR CLASE. (Si es la primera clase a diseñar entonces los botones de editar y eliminar clase deben aparecer inactivos).
   4. Al dar Clic en el botón DISEÑAR CLASE deben aparecer los controles:
      1. Me aparece la fecha y el numero de la clase que estoy diseñando, el sistema debe llevar el consecutivo de las clases creadas con los nombres clase 1, clase 2, clase 3 etc, esto es solo información.
      2. Me debe permitir colocar nombre a la clase y una breve descripción de la misma, también la fecha de ejecución de la clase (un control tipo calendario para escoger la fecha de cuando se dictará la clase)
      3. Debe aparecer un campo amplio para escribir el cuerpo o contenido de la clase que es diferente de la descripción, porque la descripción es corta.
         1. NEGRILLA
         2. CURSIVA
         3. SUBRAYADO
         4. TAMAÑO DE LA LETRA
         5. COLOR DE LA LETRA
         6. LINK (PERMITE CONVERTIR EL TEXTO EN UN HIPERVINCULO A UN SITIO WEB)
      4. En este campo de texto escribiré las instrucciones el plan de la clase etc
      5. Debe contar con una sección llamada recursos con botones los cuales me permitan adjuntar los siguientes recursos:
         1. Documentos en formato pdf.
         2. Imágenes
         3. Audios
         4. Videos de youtube por ejemplo
      6. Estos recursos el sistema debe permitir al usuario adjuntarlos de su disco Duro, es decir abrir la interfaz de adjuntar que lo busque en sus carpetas y lo adjunte el sistema debe subir este recurso a su respectivo bucket y registrarlo en la tabla que designes para ello.
      7. Debe permitirme colocarle una etiqueta o nombre a cada recurso adjunto, para que los estudiantes y yo con ese nombre sepamos a que recurso nos referimos.
      8. Los recursos que se vayan adjuntando se van listando en esta interfaz de diseño en la parte de abajo, PERO EN EL DASH DEL ESTUDIANTE LOS RECURSOS APARECERAN COMO PEQUEÑAS TARJETAS EN EN LA PARTE BAJA DEL TEXTO ETIQUETADOS CON LOS NOMBRES QUE LES ASIGNAMOS AQUÍ.
      9. Debe contar con un botón vista de estudiante para que desde esta ventana de diseño se pueda ir visualizando como esta quedando el diseño de la clase.
      10. Debe contar finalmente con un botón publicar clase, lo que significa que se publicará en el respectivo dashboard del grado correspondiente (1° - 11°), al dar clic en publicar debe aparecer una ventana flotante indicando que se ha creado la clase numero xxxxx titulo de la clase xxxxxx descripción de la clase xxxxx grado xxxxxx y fecha de ejecución xxxx
   5. Al dar clic en el botón EDITAR CLASE debe
      1. Aparecer las clases que han sido creadas, clase 1, clase 2, etc cuando pasemos el mouse sobre la clase debe aparecer la información flotante de titulo, descripción y fecha de ejecución.
      2. Al dar clic sobre la clase que se desea modificar se despliegan los mismos controles de la interfaz DISEÑAR CLASE con los datos cargados de la clase que vamos a modificar, una vez realizadas las modificaciones de nuevo debemos dar clic en el botón publicar clase y de nuevo una ventana flotante con los datos correspondientes me debe confirmar que la clase ha sido modificada con éxito, y la clase ahora debe tener una etiqueta adicional que diga CLASE MODIFICADA.
   6. Al dar clic en el botón ELIMINAR CLASE debe
      1. Aparecer las clases que han sido creadas, clase 1, clase 2, etc cuando pasemos el mouse sobre la clase debe aparecer la información flotante de titulo, descripción y fecha de ejecución.
      2. Al dar clic cobre la clase que se desea eliminar , el sistema debe pedir confirmación de eliminación de la clase, una vez confirmada la orden el sistema debe eliminar esta clase del sistema.
2. Al dar clic en el botón QUÉ VAMOS A APRENDER o PROYECTOS, la interfaz debe ser la misma que usamos para DISEÑAR CLASE, con la diferencia que estos contenidos diseñados estarán en el dash del estudiante en QUÉ VAMOS A APRENDER o en PROYECTOS, pero para mi como administrador el proceso de creación, modificació o eliminación del QUÉ VAMOS A PRENDER o PROYECTOS será exactamente igual que los procesos con las clases.

ESTRUCTURAS DEL BACKEND

|  |
| --- |
| table\_name,column\_name,data\_type,character\_maximum\_length,is\_nullable |
| clases,id,integer,null,NO |
| clases,grado,integer,null,NO |
| clases,trimestre,integer,null,YES |
| clases,numero\_clase,integer,null,NO |
| clases,nombre,text,null,NO |
| clases,descripcion,text,null,YES |
| clases,fecha\_ejecucion,date,null,YES |
| clases,contenido,text,null,YES |
| clases,publicado,boolean,null,YES |
| clases,fecha\_publicacion,timestamp without time zone,null,YES |
| proyectos,id,integer,null,NO |
| proyectos,grado,integer,null,NO |
| proyectos,trimestre,integer,null,YES |
| proyectos,nombre,text,null,NO |
| proyectos,descripcion,text,null,YES |
| proyectos,fecha\_inicio,date,null,YES |
| proyectos,fecha\_final,date,null,YES |
| proyectos,publicado,boolean,null,YES |
| proyectos,fecha\_publicacion,timestamp without time zone,null,YES |
| que\_aprender,id,integer,null,NO |
| que\_aprender,grado,integer,null,NO |
| que\_aprender,trimestre,integer,null,YES |
| que\_aprender,contenido,text,null,NO |
| que\_aprender,publicado,boolean,null,YES |
| que\_aprender,fecha\_publicacion,timestamp without time zone,null,YES |
| recursos,id,integer,null,NO |
| recursos,clase\_id,integer,null,YES |
| recursos,tipo\_recurso,text,null,NO |
| recursos,nombre,text,null,NO |
| recursos,url,text,null,NO |
| recursos,fecha\_subida,timestamp without time zone,null,YES |
| usuarios,codigo,bigint,null,NO |
| usuarios,nombre,text,null,NO |
| usuarios,perfil,USER-DEFINED,null,NO |
| usuarios,created\_at,timestamp with time zone,null,NO |

**Checkpoint técnico – Fase 2 (corte al día de hoy)**

**1) Alcance entregado**

* Se mantiene intacto el **módulo de autenticación (Fase 1)** y se corrigen llamadas a la función fn\_validar\_codigo.
* Se implementa el **Dashboard del Administrador** con:
  + Diseño de clases (crear/editar/publicar).
  + Descripción corta de la clase (nuevo campo en UI).
  + Editor enriquecido (negrita/itálica/subrayado, tamaño, color, enlaces) con **barra de formato en línea**.
  + Gestión de **recursos** para Clases, Proyecto y Qué vamos a aprender:
    - Carga de archivos a Supabase Storage (por grado: buckets 1..11, subcarpetas img/pdf/audio/video).
    - Admisión de **URL externas** (p. ej., YouTube).
  + Botonera agrupada en **fieldset “Controles”** para una UI más profesional.
* Se establece **línea gráfica**: escala de grises, títulos/subtítulos azul claro; tarjetas centradas y densidad compacta.

**2) Estructura de archivos (front)**

SDCS2026/

└─ administrador/

├─ administrador.html (login administrador – Fase 1)

├─ administrador.js (lógica de acceso – Fase 1)

├─ administrador.css (estilos globales admin – actualizado)

└─ dashboard.html (Dashboard Administrador – Fase 2)

└─ js/

└─ supabaseClient.js (createClient con URL/ANON KEY)

└─ base/

└─ styles.css (tu hoja de estilos base, si aplica)

**Dependencias:**  
CDN Supabase v2 (incluida en dashboard.html):

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

**3) Cambios/ajustes en backend (Supabase)**

**3.1 RLS – Storage (leer/subir/borrar objetos)**

Los buckets 1..11 ya existen y están públicos; no se crean.  
**Nota:** En políticas de INSERT usar solo WITH CHECK.

-- Lectura pública de objetos por buckets de grado

drop policy if exists "public read grade buckets" on storage.objects;

create policy "public read grade buckets"

on storage.objects

for select

to anon

using (bucket\_id in ('1','2','3','4','5','6','7','8','9','10','11'));

-- Subida desde el front (IMPORTANTE: WITH CHECK)

drop policy if exists "anon upload grade buckets" on storage.objects;

create policy "anon upload grade buckets"

on storage.objects

for insert

to anon

with check (bucket\_id in ('1','2','3','4','5','6','7','8','9','10','11'));

-- (Opcional) Borrado desde el front

drop policy if exists "anon delete grade buckets" on storage.objects;

create policy "anon delete grade buckets"

on storage.objects

for delete

to anon

using (bucket\_id in ('1','2','3','4','5','6','7','8','9','10','11'));

**3.2 RLS – Tabla public.recursos**

alter table public.recursos enable row level security;

drop policy if exists "anon insert recursos" on public.recursos;

create policy "anon insert recursos"

on public.recursos

for insert

to anon

with check (true);

drop policy if exists "anon select recursos" on public.recursos;

create policy "anon select recursos"

on public.recursos

for select

to anon

using (true);

Si existen tablas adicionales para Proyecto/QA (p. ej. recursos\_proyecto, recursos\_qa), replicar políticas de insert/select.

**3.3 Fix de autenticación (recordatorio)**

* La llamada a la RPC debe castear el código a **integer/bigint** y llamar una sola firma:

const clean = String(codigoStr).trim();

const codigoNum = parseInt(clean, 10);

const { data, error } = await supabase.rpc('fn\_validar\_codigo', { codigo\_in: codigoNum });

* **Evitar** firmas duplicadas de fn\_validar\_codigo (bigint/int) para no provocar “could not choose best candidate”.

**4) Implementación de recursos (front)**

* Subida de archivo:
  + Bucket por grado: String(grado) → “1”, “2”, …
  + Carpeta por tipo: img, pdf, audio, video.
  + **Ruta final (fix aplicado):**
  + const path = `${folder}/${Date.now()}\_${file.name}`;
  + publicUrl obtenido vía getPublicUrl(path).
* Alternativa por **URL externa** (si no se adjunta archivo).
* Persistencia en BD:
  + Clases: inserta en public.recursos (clase\_id, tipo\_recurso, nombre, url).
  + Proyecto/QA: el código intenta insertar en recursos\_proyecto / recursos\_qa si existen (no bloqueante si no están).

**5) Flujos implementados**

**Clases**

* **Listar** por (grado, trimestre).
* **Diseñar** (crea “Clase N” auto-siguiente).
* **Editar** (título, descripción corta, fecha, contenido).
* **Recursos**: PDF/Imagen/Audio/Video + URL/archivo.
* **Guardar** (borrador) / **Publicar** (marca publicado y fecha\_publicacion).
* **Vista previa** básica (modal).

**Proyecto / Qué vamos a aprender**

* Formulario con **recursos** (mismo patrón de Clases).
* Guardar / Publicar / Listar.

**6) UX/UI acordada**

* Paleta en escala de grises; **títulos/subtítulos en azul claro**.
* Tarjetas centradas; **densidad compacta**; campos en **dos columnas** cuando aplica.
* **Barra de formato en línea**.
* Acciones agrupadas en **fieldset “Controles”**.
* Botones con **estado deshabilitado** cuando no corresponda.

**7) Pruebas mínimas (QA rápido)**

1. Login admin (Fase 1) → redirección a dashboard.html.
2. Clases:
   * Seleccionar grado y trimestre → **Diseñar clase** → completar **título/descr. corta/fecha** → escribir contenido.
   * Agregar **recurso** por archivo y por URL → verificar lista de recursos.
   * **Guardar** y luego **Publicar** → verificar flags en BD.
   * Revisar objeto en **Storage** (bucket del grado, carpeta por tipo).
3. Proyecto / QA:
   * Repetir flujo de recursos + guardar/publicar.
4. Ver que no se emitan errores RLS (consola limpia) ni 400/401.

**8) Pendientes / Próximo sprint**

* **Eliminar** recursos individuales y clases (UI + RLS si se habilita delete en tablas).
* **Editar**/reordenar recursos (arrastrar/orden).
* **Vista Estudiante** dedicada (lectura limpia + recursos embebidos).
* Validaciones de campos (longitud, tipos de archivo, tamaño).
* **Logs/auditoría** (opcional: tabla de eventos).
* Tests de regresión sobre autenticación.

**9) Riesgos/Notas**

* Mantener una sola firma de fn\_validar\_codigo para evitar ambigüedad de tipos.
* Políticas RLS: recordar que INSERT exige **WITH CHECK** (no USING).
* Si se agregan tablas de recursos para Proyecto/QA, replicar políticas y claves foráneas.

**10) Versionado sugerido (git)**

git add -A

git commit -m "Checkpoint Fase 2: dashboard admin + recursos (Clases/Proyecto/QA), estilos y RLS"

git push

**Estado:** Fase 2 en marcha, **dashboard funcional** con diseño/recursos y publicación. Autenticación estable (Fase 1).