1. **Cómo se arma todo**

* El **motor de base de datos** se configura leyendo DATABASE\_URL del .env. Si falta, el programa se detiene con un error claro. Además, el engine usa pool\_pre\_ping (evita conexiones muertas) y pool\_recycle=3600 (renueva cada hora). Bien por robustez en conexiones largas.

db

* La **app FastAPI** define rutas para salud, debug de conectores, reconciliación en tres sabores, sugerencias GBIF tipo autocomplete, procesamiento en lote (JSON/CSV/XLSX) y carga de archivo; aplica CORS abierto y una **API key opcional** (si no configuras API\_KEY, no exige llave). También levanta las tablas al inicio.

main

1. **Modelo de datos (lo que guardas)**

* Tienes dos tablas: **Taxon** (la especie/taxón con jerarquía, estado, claves GBIF, categoría IUCN, fuentes, provenance JSON) y **Synonym** (sinónimos con autoría/estado/fuente/clave externa). Pusiste varios **índices** útiles (por gbif\_key, accepted\_gbif\_key, status, rank, family+genus, epiteto\_especifico, etc.) y codificación **utf8mb4** correcta. Bien pensado para consultas rápidas y nombres con caracteres raros.

models

1. **Endpoints clave y flujo de reconciliación**

* /reconcile (compacto en inglés) y /reconciliar (compacto en español) devuelven campos básicos; /reconciliar/detalle suelta todo (incluye sinónimos y provenance). Internamente usan reconcile\_name, normalización y derivan el **epíteto específico** al vuelo (útil para reglas). Buen diseño “thin controller”: la lógica pesada está en servicios/modelo.

main

* /reconciliar/bulk y /reconciliar/archivo te permiten procesar **varios nombres** devolviendo JSON/CSV/XLSX, y si quieres, agregas una hoja de **sinónimos**. Si pides “modo=db”, la salida se alinea con las columnas reales de Taxon, lo cual facilita ETL.

main

* La UI /ui/buscador trae **autocomplete GBIF** y una cuadrícula para resultados/descargas (CSV/XLSX). Para ambientes con seguridad, soporta guardar la **API key** en localStorage.

main

1. **Diagnóstico de conectores**

* Hay un módulo de **diagnósticos** que intenta primero los **clientes internos** (gbif/col/worms/itis/iucn/sib) y, si fallan, hace **ping HTTP directo** a sus endpoints. Maneja cabeceras seguras y sanitiza valores. Además contempla token de **IUCN** y header para **SIB Colombia**. Esto te da un “panel de salud” rápido para saber si la red o el token están bien.

diagnostics

* En la API, /debug/conectores corre todos esos checks en paralelo con asyncio.gather y te devuelve un resumen. También hay endpoints de **depuración cruda** para ITIS/CoL que ayudan muchísimo a investigar casos raros.

main

1. **Esquemas de salida (Pydantic)**

* Definiste modelos compactos y de detalle (incluye lista de sinónimos y provenance) con from\_attributes=True para mapear desde ORM con mínima fricción. Esto mantiene estable la interfaz del API aunque cambie el modelo interno.

schemas

**Hallazgos + mejoras accionables (concretas)**

**A. Seguridad / Operación**

* **API key opcional:** Actualmente, si no hay API\_KEY el sistema queda abierto (útil en dev, no en prod). Recomendación: forzar API key en producción o al menos cerrar las rutas de **/bulk** y **/archivo**, que pueden consumir bastante CPU/red.

main

* **CORS ‘\*’**: práctico para pruebas; en despliegue fija tu origen (dominio) para evitar uso cruzado indeseado.

main

**B. Rendimiento / Escalabilidad**

* **Lotes grandes**: /reconciliar/bulk procesa nombres **secuencialmente**. Si metes cientos/miles, tardará. Propongo:
  1. Resolver nombres con **paralelismo controlado** (e.g., asyncio.Semaphore(5–8)) y **backoff exponencial** por fuente.
  2. Agregar **caché LRU** en memoria (y/o tabla match\_cache) para repetidos.
  3. Parametrizar **limit\_per\_source** o timeouts por conector para evitar cuellos.

main

* **Sinónimos en bulk**: ya consolidas synonyms\_csv por taxon con un groupby—👍. Solo notar que si la tabla crece, conviene **precalcular** o paginar sinónimos (si la UI no necesita todos).

**C. Calidad de datos**

* **epiteto\_especifico indexado** pero no veo persistencia explícita en el flujo (se calcula para la respuesta). Asegura que reconcile\_name **actualice** la columna para que el índice realmente rinda (o elimina el índice si no lo usarás en consultas).
* **sources\_csv**: gran idea para trazabilidad, pero considera migrar a una tabla taxon\_source (1:N) si planeas filtros del tipo “dame taxones con evidencia en SIB+GBIF pero no en IUCN”, etc.
* **provenance\_json**: perfecto para auditoría. Sugiero estandarizar un **esquema mínimo** (campos obligatorios por fuente: id, timestamp, query, decisión de aceptación) y validar antes de guardar.

**D. UX / DX**

* **Detección de columna en /reconciliar/archivo**: ya contemplas varias (scientific\_name, nombre\_cientifico, etc.). Añadir alias locales comunes: especie, taxon, taxón, podría reducir rechazos en campo.
* **UI**: muy útil el autocomplete. Para listas largas, añade **paginación**/scroll virtual en la tabla y un indicador de **progreso** cuando corras bulk desde la UI.

**E. Conectividad con Colombia (tu prioridad)**

* Ya tienes chequeo para **SIB Colombia** con token opcional; excelente base. Añadir un **endpoint** que liste el **estado de los tokens** (presentes/expiran) y un **rate-limit** por conector te evitará bloqueos del servicio en campañas grandes.

**F. Operación y soporte**

* **/debug/config** enmascara credenciales (bien). Suma un /health/db que haga un **SELECT 1** y reporte latencia real a DB para monitoreo.
* **Índices**: además de los existentes, si vas a buscar mucho por accepted\_gbif\_key (taxa válidos) y family+genus, ya estás cubierto. Considera un índice para **canonical\_name** si lo usas para búsquedas exactas.

**“Pruebas rápidas” que te recomiendo correr**

* **Salud de conectores**: GET /debug/conectores y revisar que todos estén “OK” (o “NO\_TOKEN” en IUCN si aplica).
* **Caso canónico**: GET /reconciliar/detalle?q=Ateles belzebuth y confirmar jerarquía + sinónimos + provenance.
* **Bulk JSON**: POST /reconciliar/bulk con {"names":["Bos taurus","Ateles belzebuth"], "modo":"resumen","format":"json"} verificando synonyms\_csv.
* **Archivo**: subir un .xlsx con columna scientific\_name y validar que descargue **dos hojas**: taxones y sinonimos.

**Respuesta formal (breve)**

**Arquitectura y DB.**  
El proyecto configura el motor SQLAlchemy por .env con saneamiento de pool (pre\_ping, recycle) y expone una API FastAPI que cubre salud, diagnóstico de conectores (GBIF/CoL/WoRMS/ITIS/IUCN/SIB), reconciliación (básica y detallada), sugerencias GBIF, procesamiento en lote y carga de archivos. El modelo relacional (Taxon, Synonym) está bien indexado y preparado para Unicode.

**Fortalezas.**  
(1) Diagnósticos asíncronos con fallback HTTP, (2) exportaciones JSON/CSV/XLSX con consolidación de sinónimos, (3) UI ligera con autocomplete y manejo de API key local.

**Riesgos/mejoras.**  
(1) Endpoints abiertos si no se define API\_KEY y CORS “\*” (cerrar en prod), (2) bulk secuencial: agregar concurrencia limitada, caché y backoff, (3) garantizar persistencia/uso del índice epiteto\_especifico, (4) considerar normalizar sources\_csv y estandarizar provenance\_json, (5) ampliar alias de columna en carga y añadir paginación/progreso en UI.

Si quieres, en la siguiente iteración te paso **parches puntuales** (semáforo de concurrencia para bulk, cache LRU y un middleware de rate-limit por IP/endpoint).

Diseño **pantalla principal** (home) para que, desde ahí, elijas qué hacer. Te dejo primero “plastilina” y abajo “respuesta formal” breve.

**Plastilina (explicación didáctica)**

**Objetivo:** una **pantalla de inicio** simple, limpia y estable que sea el “hub” del sistema. Desde ahí eliges: reconciliar 1 nombre, procesar un archivo, abrir el buscador web, correr diagnósticos, ver configuración, etc. (aunque el *reconciliador* “core” no esté en línea todavía).

**1) Qué se ve (layout propuesto)**

* **Encabezado**: título (“Biotico – Centro de herramientas”) + versión corta (vX.Y).
* **Tarjetas / botones grandes** (grid 2×3 en desktop, 1×N en móvil):
  1. **Reconciliar un nombre**  
     Abre un formulario mínimo: caja de texto + botón “Reconciliar”.  
     (Muestra resultado compacto y link a “detalle” si existe).
  2. **Reconciliar por archivo (lote)**  
     Sube CSV/XLSX/JSON. Opciones: “modo=Resumen/Detalle”, “incluir sinónimos”.
  3. **Buscador web (externo)**  
     Botón que **abre en nueva pestaña** biotico-app-web.onrender.com (por ahora sin integrar).
  4. **Diagnóstico de conectores**  
     Ejecuta los checks y muestra semáforo (GBIF, CoL, WoRMS, ITIS, IUCN, SIB).
  5. **Configuración**  
     API keys (IUCN, SIB), URL de DB, timeouts, idioma interfaz.
  6. **Ayuda / Acerca de**  
     Guía corta de uso + enlaces a documentación.
* **Pie de página**: estado rápido (DB: OK / Latencia; Último ping a conectores).

Ventaja: aunque el “motor” de reconciliación todavía no esté expuesto en línea, el **home** ya te sirve como puerta de entrada organizada. Si algo aún no está listo, el botón muestra un aviso “próximamente” o abre un modal de configuración.

**2) Cómo activarlo “sin código” (o con mínimo cambio)**

Tienes tres vías, empezando por la **más simple**:

* **A) Solo un “home” estático** (0 backend):  
  Coloca una **página HTML estática** como portada (index) con botones que:
  + Abren **en nueva pestaña** el buscador web externo.
  + Para funciones que aún no están, muestran un **mensaje** (“próximamente”).
  + Cuando el reconciliador esté disponible, esos botones apuntan a sus rutas.

Esto se puede servir con cualquier hosting estático o como “/” en tu servidor actual, sin tocar el API.

* **B) “Home” como portada del servidor actual** (cambio mínimo):  
  Agrega una **plantilla index** (o un HTML simple) como ruta raíz /.
  + Los botones linkean a:
    - /ui/reconciliar (form 1 nombre)
    - /ui/bulk (subida de archivo)
    - /ui/buscador (si ya tienes una UI propia) o **link externo** al buscador.
    - /debug/conectores (diagnóstico)
    - /config (si existe, o un modal declarativo)
  + Si alguna ruta no existe aún, deja el botón **deshabilitado** con tooltip.
* **C) Reverse proxy con “landing”** (sin tocar tu app actual):  
  Un Nginx/Cloudflare sirve una **landing** bonita y enruta:
  + / → la landing
  + /consulta/\* → el buscador externo (proxy-pass)
  + /api/\* → tu backend cuando esté listo  
    Útil si quieres “un solo dominio” aunque tengas varios servicios por detrás.

**3) Flujo de usuario (simple y claro)**

1. Entra al **Home**.
2. Elige una **tarjeta**:
   * Si elije **Reconciliar un nombre**: abre formulario → ver resultado → botón “Ver detalle” (si aplica).
   * Si elije **Archivo**: sube archivo → muestra barra de progreso → descarga CSV/XLSX de salida.
   * Si elije **Buscador web**: abre en **otra pestaña** (no rompe su flujo en el Home).
   * Si elije **Diagnóstico**: corre checks y muestra semáforo con tiempos.
3. Desde cualquier vista, un botón “Inicio” te **regresa al Home**.

**4) Estados vacíos y mensajes (UX que no estorba)**

* Si el reconciliador **no está disponible**, los botones relacionados muestran:
  + “Servicio no disponible (aún)” + tip: “revisa Configuración” o “intenta Diagnóstico”.
* Si no hay API keys, muestra un **banner**: “Configura IUCN/SIB para mejores resultados”.

**5) Cuando me cuentes entorno y OS**

* Si es **web (FastAPI/Flask)**: te doy un **index.html** listo con estilos básicos (o Tailwind CDN) y enlaces, sin tocar Python.
* Si es **escritorio (Tkinter)**: te paso una **ventana principal** con botones y frame contenedor (sin romper tu lógica).
* Si es **Android (Kivy/KivyMD)**: te preparo una **Screen** “Home” con MDGridLayout y navegación (respetando tu estructura actual).

**Respuesta formal (breve)**

Propongo una **pantalla principal** (home) con 6 tarjetas: (1) Reconciliar un nombre, (2) Reconciliar por archivo, (3) Buscador web (abre en nueva pestaña), (4) Diagnóstico de conectores, (5) Configuración y (6) Ayuda/Acerca de. Se puede implementar **sin tocar el backend** añadiendo una **página estática** como portada con enlaces (opción más rápida), o bien con un **index** servido por tu aplicación actual, o mediante un **reverse proxy** que sirva la landing y enrute a servicios externos/internos. Esta estructura funciona aunque el reconciliador aún no esté en línea y permite habilitar cada módulo en cuanto esté listo, con estados vacíos y mensajes claros. Cuando me digas entorno y SO, te entrego la versión concreta (HTML estático, Tkinter o Kivy) sin romper lo existente.

Propuesta en código inicial

import React from "react";

import { motion } from "framer-motion";

import {

Search,

Upload,

Globe,

Activity,

Settings,

HelpCircle,

ChevronRight,

} from "lucide-react";

/\*\*

\* Biotico – Pantalla Principal (Web Mockup)

\* -------------------------------------------------------

\* - Single-file React component you can drop into any app.

\* - TailwindCSS for styling (no custom CSS required).

\* - Uses lucide-react for icons and framer-motion for subtle animations.

\* - Buttons that don’t exist yet are marked as “Próximamente”.

\* - The “Buscador web (externo)” abre en nueva pestaña la URL solicitada.

\*/

const Card = ({ icon: Icon, title, desc, cta, href, onClick, disabled, external }) => (

<motion.div

initial={{ opacity: 0, y: 12 }}

animate={{ opacity: 1, y: 0 }}

transition={{ duration: 0.35 }}

className={`relative rounded-2xl border border-zinc-200/70 dark:border-zinc-700/60 bg-white/80 dark:bg-zinc-900/60 backdrop-blur p-5 shadow-sm hover:shadow-md transition-shadow ${

disabled ? "opacity-70 pointer-events-none" : ""

}`}

>

<div className="flex items-start gap-4">

<div className="shrink-0 rounded-xl p-3 border border-zinc-200/60 dark:border-zinc-700/60">

<Icon className="h-6 w-6" aria-hidden />

</div>

<div className="min-w-0">

<div className="flex items-center gap-2">

<h3 className="text-base font-semibold leading-tight truncate">{title}</h3>

{disabled && (

<span className="text-[10px] rounded-full px-2 py-0.5 bg-amber-100 text-amber-700 dark:bg-amber-900/40 dark:text-amber-300">Próximamente</span>

)}

</div>

<p className="mt-1 text-sm text-zinc-600 dark:text-zinc-300 line-clamp-3">{desc}</p>

{href ? (

<a

href={href}

target={external ? "\_blank" : undefined}

rel={external ? "noopener noreferrer" : undefined}

className="mt-4 inline-flex items-center gap-2 text-sm font-medium rounded-xl border border-zinc-300/70 dark:border-zinc-700 px-3 py-2 hover:bg-zinc-50 dark:hover:bg-zinc-800"

>

{cta} {external && <ChevronRight className="h-4 w-4" />}

</a>

) : (

<button

onClick={onClick}

className="mt-4 inline-flex items-center gap-2 text-sm font-medium rounded-xl border border-zinc-300/70 dark:border-zinc-700 px-3 py-2 hover:bg-zinc-50 dark:hover:bg-zinc-800"

>

{cta}

</button>

)}

</div>

</div>

</motion.div>

);

export default function BioticoHome() {

return (

<div className="min-h-dvh bg-gradient-to-b from-zinc-50 to-white dark:from-zinc-950 dark:to-zinc-900 text-zinc-900 dark:text-zinc-50">

{/\* Header \*/}

<header className="max-w-6xl mx-auto px-4 pt-10 pb-6">

<div className="flex items-start justify-between gap-4">

<div>

<h1 className="text-2xl sm:text-3xl font-bold tracking-tight">Biotico — Centro de herramientas</h1>

<p className="mt-2 text-sm text-zinc-600 dark:text-zinc-300 max-w-2xl">

Elige lo que quieres hacer. Esta es la pantalla principal. Algunas funciones pueden estar deshabilitadas mientras el reconciliador no esté en línea.

</p>

<div className="mt-3 flex items-center gap-2 text-xs">

<span className="inline-flex items-center rounded-full border border-zinc-300/70 dark:border-zinc-700 px-2 py-0.5">v0.1</span>

<span className="inline-flex items-center rounded-full border border-emerald-300/60 text-emerald-700 dark:text-emerald-300 dark:border-emerald-800 px-2 py-0.5">DB: pendiente</span>

<span className="inline-flex items-center rounded-full border border-zinc-300/70 dark:border-zinc-700 px-2 py-0.5">CSP/iframe: por confirmar</span>

</div>

</div>

<div className="hidden sm:flex flex-col items-end gap-2">

<span className="text-xs text-zinc-500">Estado general</span>

<div className="flex gap-2">

<span className="h-2.5 w-2.5 rounded-full bg-amber-400" title="Conectores: revisar" />

<span className="h-2.5 w-2.5 rounded-full bg-zinc-300" title="DB: sin probar" />

<span className="h-2.5 w-2.5 rounded-full bg-zinc-300" title="API: sin probar" />

</div>

</div>

</div>

</header>

{/\* Grid of actions \*/}

<main className="max-w-6xl mx-auto px-4 pb-16">

<div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-5">

{/\* Reconciliar un nombre \*/}

<Card

icon={Search}

title="Reconciliar un nombre"

desc="Ingresa un nombre científico y obtén un resultado compacto."

cta="Abrir formulario"

onClick={() => alert("Formulario local: próximamente")}

disabled={true}

/>

{/\* Reconciliar por archivo \*/}

<Card

icon={Upload}

title="Reconciliar por archivo (lote)"

desc="Sube CSV/XLSX/JSON. Puedes pedir sinónimos y exportar resultados."

cta="Subir archivo"

onClick={() => alert("Carga de archivo: próximamente")}

disabled={true}

/>

{/\* Buscador web (externo) \*/}

<Card

icon={Globe}

title="Buscador web (externo)"

desc="Abre el buscador web en una nueva pestaña mientras integramos el resto."

cta="Abrir biotico-app-web.onrender.com"

href="https://biotico-app-web.onrender.com/"

external

/>

{/\* Diagnóstico de conectores \*/}

<Card

icon={Activity}

title="Diagnóstico de conectores"

desc="Ejecuta pruebas de GBIF, CoL, WoRMS, ITIS, IUCN y SIB."

cta="Ejecutar diagnóstico"

onClick={() => alert("Diagnóstico: próximamente (requiere backend)")}

disabled={true}

/>

{/\* Configuración \*/}

<Card

icon={Settings}

title="Configuración"

desc="API keys, URL de base de datos, timeouts y preferencia de idioma."

cta="Abrir configuración"

onClick={() => alert("Configuración: próximamente")}

disabled={true}

/>

{/\* Ayuda / Acerca de \*/}

<Card

icon={HelpCircle}

title="Ayuda / Acerca de"

desc="Guía rápida, preguntas frecuentes y notas de versión."

cta="Ver ayuda"

onClick={() => alert("Ayuda: próximamente")}

disabled={false}

/>

</div>

{/\* Tip bar \*/}

<div className="mt-8 rounded-2xl border border-zinc-200/70 dark:border-zinc-700/60 p-4 text-sm bg-zinc-50/60 dark:bg-zinc-900/50">

<p>

Consejo: si el host externo bloquea iframes, mantenlo como enlace externo. Cuando el backend esté listo, cambia estos botones para navegar a las rutas internas.

</p>

</div>

</main>

{/\* Footer \*/}

<footer className="max-w-6xl mx-auto px-4 pb-8 text-xs text-zinc-500">

<div className="flex items-center justify-between">

<span>© {new Date().getFullYear()} Biotico</span>

<div className="flex items-center gap-2">

<a href="#" className="hover:underline">Política de privacidad</a>

<span>•</span>

<a href="#" className="hover:underline">Términos</a>

</div>

</div>

</footer>

</div>

);

}

Imagen inicial



Arquitectura de la pagina

[ Usuario ]

│

├── Navega a: buscador.biotico.app → (Render - Web buscador, desplegado desde GitHub)

│ │

│ └── Conecta por TCP/SSL a → [ Railway MySQL ] (tu DB actual)

│

└── Navega a: api.biotico.app (cuando publiques el reconciliador)

│

└── Conecta por TCP/SSL a → [ Railway MySQL ] (misma DB)

**Puntos clave para “seguir igual, pero en producción”:**

1. **Una sola DB (Railway) para todo**
   * Mantén tu DATABASE\_URL de Railway como **variable de entorno** tanto en el buscador (Render) como en el reconciliador (cuando lo montes).
   * Si Railway ofrece **SSL**, actívalo desde el lado del cliente (Render) para evitar bloqueos por redes.
2. **Despliegue del buscador (ya lo tienes):**
   * **GitHub → Render (Static/Web Service)**.
   * Variables en Render: DATABASE\_URL y cualquier API\_KEY que uses (IUCN/SIB).
   * **CORS**: si el buscador llama a APIs externas propias, fija ALLOWED\_ORIGINS al dominio público (ver punto 4).
3. **Publicar el reconciliador (cuando decidas):**
   * Súbelo también a **Render** (Web Service FastAPI/Flask).
   * Mismas **env vars**: DATABASE\_URL, API\_KEY, LANG=es, SIB\_TOKEN, IUCN\_TOKEN (si aplica).
   * Tu código ya crea tablas al iniciar, así que con la misma DB de Railway **no crea duplicados** (solo asegura mismo esquema).
4. **Dominios bonitos (opcional pero recomendado):**
   * Usa subdominios:
     + buscador.tudominio.com → servicio de Render del buscador.
     + api.tudominio.com → servicio del reconciliador.
   * En **CORS** del API pon https://buscador.tudominio.com (y si necesitas, también https://biotico-app-web.onrender.com mientras migras).
5. **“Pestaña” desde el Home**
   * En tu **pantalla principal** (la que ya maquetamos), el botón “Buscador web” **abre la URL de Render** en nueva pestaña.
   * Si en el futuro quieres **incrustar** el buscador dentro de la misma página, sólo si el servidor del buscador **permite iframe**; si no, mantenlo como link o usa **reverse proxy**.
6. **Credenciales y seguridad**
   * Nada de credenciales en el repo de GitHub. Solo **env vars** en Render.
   * Si tu API queda abierta, exige API\_KEY (tu código ya lo soporta) al menos en rutas pesadas como “bulk”.
   * Limita CORS a tus dominios reales (quita \* en producción).
7. **Conexiones y límites**
   * Render tiene **concurrencia** y **cold start** en planes gratuitos. Para cargas grandes, usa plan con **autoscale** o workers.
   * Tu SQLAlchemy ya usa pool\_pre\_ping y pool\_recycle=3600 (bien). Mantén POOL\_SIZE moderado (p.ej., 5–10) para no saturar Railway.
8. **Monitoreo rápido**
   * Expón /health (API) con SELECT 1 a DB y /debug/conectores para ver estado GBIF/CoL/ITIS/IUCN/SIB.
   * En el **home**, un semáforo que lea esos endpoints.

**Checklist de “arranque en línea” (sin tocar código o con mínimos cambios):**

* Crear/confirmar servicio del **buscador** en Render (ya).
* Poner **DATABASE\_URL** de Railway en Render (buscador) y **probar conexión**.
* Definir **dominio** (subdominio) para el buscador y ajustar **CORS** si aplica.
* Mantener botón en el **Home** que abre el buscador de Render.
* (Cuando publiques el reconciliador) crear servicio en Render, mismas env vars, **CORS** hacia el buscador, y probar /reconciliar y /debug/conectores.

**Respuesta formal (breve)**

Sí: mantén **Railway** como única DB; el **buscador** seguirá en **GitHub → Render** y, cuando liberes el **reconciliador**, también en Render, ambos apuntando a la **misma DATABASE\_URL**. Usa **env vars** (no credenciales en código), **CORS** restringido a tus dominios, y (opcional) subdominios tipo buscador.tu-dominio.com y api.tu-dominio.com. Desde la **pantalla principal** dejamos un botón que abre el buscador en Render; más adelante se puede integrar por proxy o iframe si el servidor lo permite.