無料サービスだけで作る
掲示板付きの教科書サポートページ

2026 年 4 月 27 日

概要

出版した教科書の読者から質問や誤植情報を受け付けるための「掲示板付きサポートページ」を、無料のサービスのみで構築する方法をまとめる。訪問者はアカウント登録不要で匿名投稿でき、LaTeX 記法による数式表示、スパム対策、新規投稿時の管理者へのメール通知にも対応する。月間 10 万リクエスト程度であれば全て無料枠で運用できる。

なお、筆者自身はパソコンに詳しいわけではない。本ノートに記した実装は、希望する仕様を Claude Code（Anthropic 社の対話型 AI コーディングアシスタント）に伝え、その指示通りに作業するだけで、サイトの構築から公開・運用までを完結することができた。

目次

1 はじめに

教科書を出版すると、読者から質問や誤植のご指摘をいただくことがある。これらに対して一つ一つ真摯に対応することは、著者として果たすべき責任だと考えている。掲示板形式で質問や誤植情報を募集するという案自体は、琉球大学の前野昌弘先生のウェブサイト（https://irobutsu.a.la9.jp/mybook/index.html）を参考にしている。

筆者はこれまで Google Sites と Google Forms を組み合わせた「フォーム送信式の掲示板」を利用してきたが、この構成は連携が壊れやすく、運用負担が大きかった。そこで本ノートでは、Google サービスを使わず、より安定した自前運用が可能な構成を紹介する。著者自身が運用する掲示板付きの教科書サポートページという取り組みが、他の著者の間にも広がり、教科書を介した著者と読者のやりとりが一層活発になることを願っている。

この実装には次の特徴がある。

- 訪問者はアカウント登録不要で匿名投稿できる（ハンドルネームと本文のみ）
- LaTeX 記法による数式表示に対応（`$x^2$` のような書式が綺麗に組版される）
- 三重のスパム対策：CAPTCHA、IP ベースのレート制限、ハニーポット
- 新規投稿があった際の管理者へのメール通知
- 不適切投稿を削除するためのパスワード認証付き管理画面
- 月間 10 万リクエストまで完全無料、それ以上の運用も極めて低コスト

なお本ノートは、教科書サポートのような小規模 Q&A 用途を想定している。大規模掲示板やリアルタイム性を要する用途は別途検討してほしい。

2 システム構成

全体は大きく分けて 3 つのコンポーネントで構成される。

1. **静的サイト（フロントエンド）** ― HTML / CSS / JavaScript で書かれたページ群を、無料の静的ホスティングサービスで配信する。

2. **API バックエンド** ー 投稿の受付・取得・削除を担うサーバーレス関数。
3. **データベース** ー 投稿を保存する SQLite データベース。

訪問者がブラウザでサイトを開くと、フロントエンドの JavaScript が API バックエンドに対して投稿一覧を取得しに行き、ページに表示する。新規投稿時は同じく API バックエンドにデータを送り、データベースに保存される（図 1 のような流れ）。

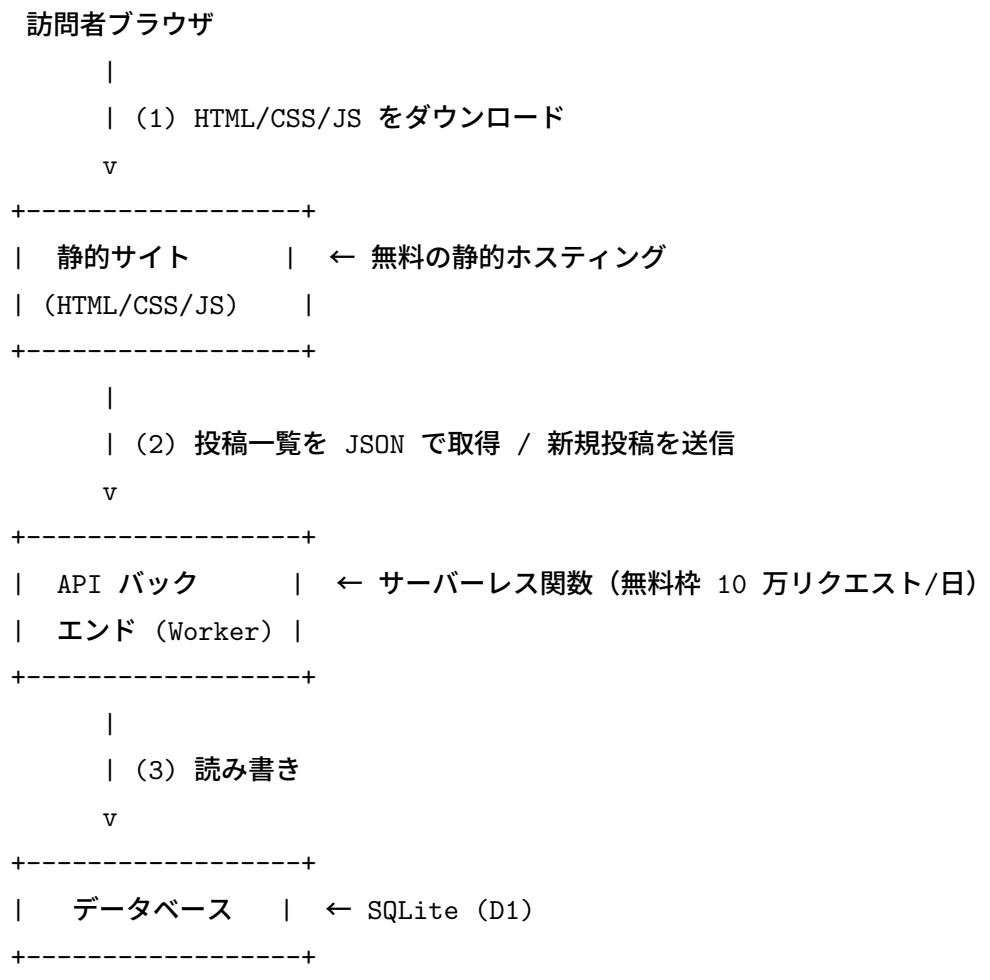

図 1　システム全体の流れ

3　使用サービスと必要なアカウント

本構成で利用するサービスは次の通りで、すべて無料プランの範囲内で運用できる。

GitHub　静的サイトのソースコード管理と公開（GitHub Pages）。教科書著者個人のアカウントを 1 つ作成。

Cloudflare　サーバーレス関数（Workers）、データベース（D1）、CAPTCHA（Turnstile）。1 つのアカウントで全機能を利用できる。

Resend　メール通知用の送信 API。月 3,000 通／日 100 通まで無料。

クレジットカードの登録は不要で、メールアドレスとパスワードだけで全アカウントを作成できる。

4　構築の流れ

実際の構築ステップを順を追って示す。

4.1　静的サイトの作成

普通の HTML / CSS / JavaScript でサイトを作る。教科書情報や著者プロフィールなどの「読み物」部分は静的 HTML として書く。掲示板部分は、ページ内の特定の箇所に投稿フォームと投稿表示エリアを置き、JavaScript で API を呼び出して動的に投稿を表示する。

最低限のフォーム構造を以下に示す。

```
<section class="board" data-board-id="ca">
  <form class="board-form">
    <label>ハンドルネーム
      <input type="text" name="handle" required maxlength="40" />
    </label>
    <label>質問内容
      <textarea name="body" required maxlength="4000"></textarea>
    </label>
    <button type="submit">投稿する</button>
  </form>
  <div class="posts"><!-- 投稿一覧を動的に挿入 --></div>
</section>
```

Listing 1　掲示板フォームの最小構造

4.2　GitHub Pages での公開

1. GitHub アカウントを取得する。
2. 公開リポジトリを 1 つ作成する。
3. ローカルでサイトのファイルを Git 管理し、リポジトリへ `push` する。
4. リポジトリの設定画面で **Pages** を有効化する（ブランチ `main`、ディレクトリ `/`）。
5. 数分後、`https://<ユーザ名>.github.io/<リポジトリ名>/` で公開される。

GitHub Pages は無料・無制限・SSL 自動・広告なしで、静的サイトのホスティングとして十分に高品質である。

4.3 Cloudflare Worker と D1 データベース

掲示板の投稿の受付・取得・削除を担うバックエンドを Cloudflare Workers で構築する。Cloudflare は近年、静的・動的問わずあらゆるホスティングを統合した「フルスタック」プラットフォームに進化しており、本用途に特に向いている。

4.3.1 Wrangler のインストール

Cloudflare Workers の CLI ツール `wrangler` を使う。Node.js（v18 以上）が必要。

```
mkdir worker && cd worker
npm init -y
npm install --save-dev wrangler
npx wrangler login
```

`wrangler login` でブラウザが開き、Cloudflare アカウントへの認可を求められるので承認する。

4.3.2 D1 データベース作成

```
npx wrangler d1 create my-board
```

返却される `database_id` を後述の `wrangler.toml` に記入する。

スキーマは次のような単純なテーブル 1 つで十分である。

```
CREATE TABLE IF NOT EXISTS posts (
  id              INTEGER PRIMARY KEY AUTOINCREMENT,
  board           TEXT    NOT NULL,
  handle          TEXT    NOT NULL,
  body            TEXT    NOT NULL,
  ip_hash         TEXT    NOT NULL,
  created_at      TEXT    NOT NULL,
  created_at_unix INTEGER NOT NULL,
  deleted_at      TEXT
);
CREATE INDEX IF NOT EXISTS idx_posts_board_id   ON posts (board, id DESC);
CREATE INDEX IF NOT EXISTS idx_posts_iphash_ts  ON posts (ip_hash,
    created_at_unix);
```

Listing 2　`schema.sql`

スキーマの適用：

```
npx wrangler d1 execute my-board --remote --file=./schema.sql
```

4.3.3 Worker の構成ファイル

`wrangler.toml` の例：

```
name = "my-board"
main = "src/index.js"
compatibility_date = "2025-01-01"

[vars]
ALLOWED_ORIGINS = "https://<your-username>.github.io"
NOTIFY_EMAIL    = "owner@example.com"
SITE_BASE_URL   = "https://<your-username>.github.io/<repo>"

[[d1_databases]]
binding = "DB"
database_name = "my-board"
database_id   = ("wrangler d1 create が返した) ID"
```

Listing 3　wrangler.toml

4.3.4　Worker 本体

JavaScript で次のようなエンドポイントを実装する。要点だけ抜粋する（完全版は付録を参照）。

```
export default {
  async fetch(request, env, ctx) {
    const url = new URL(request.url);
    if (request.method === "OPTIONS") return cors(env, new Response(null, {
        status: 204}));

    if (url.pathname === "/posts" && request.method === "GET")  return cors(env
        , await listPosts(env, url));
    if (url.pathname === "/posts" && request.method === "POST") return cors(env
        , await createPost(request, env, ctx));
    // /admin/posts ... 認証付きで一覧と削除()
    return cors(env, json({error: "not_found"}, 404));
  },
};
```

Listing 4　src/index.js（要点）

4.3.5　シークレットの登録とデプロイ

管理画面用パスワードや IP ハッシュ用のソルトをシークレットとして登録する。

```
npx wrangler secret put ADMIN_PASSWORD
npx wrangler secret put IP_HASH_SALT      # openssl rand -hex 32 などで生成
npx wrangler deploy
```

deploy 後に表示される URL（https://<worker 名>.<アカウント名>.workers.dev）が API のエンドポイントになる。

4.4　フロントエンドからの接続

フロントエンドの JavaScript から、上で得た Worker の URL を呼び出す。設定は別ファイル（例：`config.js`）に切り出しておくと扱いやすい。

```
window.BOARD_API_URL = "https://my-board.<your-cf>.workers.dev";
window.TURNSTILE_SITE_KEY = "（後述の Turnstile で取得）";
```

クライアント側の処理は概ね次の流れで実装する。

- ページ読み込み時に `GET /posts?board=...` で投稿一覧を取得し DOM に挿入。
- 投稿フォームの `submit` 時に `POST /posts` で投稿を送信、成功したら一覧を再取得。

5　追加機能の実装

5.1　LaTeX 記法による数式表示

KaTeX という JavaScript ライブラリを使うと、ブラウザ上で LaTeX 数式が美しく組版される。HTML の `<head>` に CDN から CSS と JS を読み込み、auto-render 拡張で投稿本文の数式を一括処理する。

```
<link rel="stylesheet"
  href="https://cdn.jsdelivr.net/npm/katex@0.16.11/dist/katex.min.css">
<script defer
  src="https://cdn.jsdelivr.net/npm/katex@0.16.11/dist/katex.min.js"></script>
<script defer
  src="https://cdn.jsdelivr.net/npm/katex@0.16.11/dist/contrib/auto-render.min.
      js">
</script>
```

投稿本文を DOM に挿入した後、次の関数を呼ぶ。

```
renderMathInElement(container, {
  delimiters: [
    {left: "$$", right: "$$", display: true},
    {left: "$",  right: "$",  display: false},
  ],
  throwOnError: false,
});
```

これで `$f(z) = \sum a_n z^n$` のような書式がブラウザで自然に数式として表示される。サーバー側に LaTeX 環境は不要で、すべてクライアント側で完結する。

5.2　スパム対策

匿名投稿を許す以上、スパムは必ず来る。次の三段で対処する。

5.2.1 Cloudflare Turnstile（CAPTCHA）

「私はロボットではない」を確認する CAPTCHA。Cloudflare の Turnstile は完全無料・無制限で、しかも訪問者のほとんどに対しては不可視で動作する。導入手順：

1. Cloudflare ダッシュボードで Turnstile セクションを開き、サイトを追加。
2. ホスト名（GitHub Pages のドメイン）を登録、Widget Mode は **Managed** を選択。
3. 発行された **Site Key** と **Secret Key** を取得。
4. Site Key はフロントの `config.js` に、Secret Key は `wrangler secret put TURNSTILE_SECRET_KEY` で Worker に登録。
5. フォーム内に `<div class="cf-turnstile" data-sitekey="..."></div>` を置く。
6. Worker 側で受け取ったトークンを Cloudflare の検証 API（`siteverify`）に POST して検証。

5.2.2 ハニーポット

ボットだけが入力してしまうダミーの入力欄を、CSS で画面外に隠して配置する。

```
<div style="position:absolute;left:-9999px" aria-hidden="true">
  <label>Website (do not fill in)
    <input type="text" name="website" tabindex="-1" autocomplete="off" />
  </label>
</div>
```

サーバー側で `website` 欄が空でない投稿を即座に拒否する。

5.2.3 IP ベースのレート制限

各投稿について、訪問者の IP アドレスをハッシュ化（プライバシー配慮）してデータベースに保存しておき、直近の一定時間内の投稿数をカウントする。閾値を超えた場合は HTTP 429 を返す。

```
const since = nowSec() - 60;  // 60 秒以内
const recent = await env.DB.prepare(
  "SELECT COUNT(*) AS c FROM posts WHERE ip_hash = ? AND created_at_unix >= ?"
).bind(ipHash, since).first();
if (recent.c >= 3) return json({error: "rate_limited"}, 429);
```

IP のハッシュ化は、原文を保存せず短い salted SHA-256 を保存することで、識別性を保ちつつプライバシーを守る。

5.3 メール通知

新規投稿があった際に、管理者にメールで通知する仕組みを Resend 経由で実装する。Resend はモダンなメール送信 API で、開発者向けに使いやすい。

5.3.1 Resend のセットアップ

1. Resend アカウントを取得（メール + パスワードで登録）。
2. ダッシュボードで API キーを発行（送信権限のみ）。
3. Worker のシークレットに登録：`npx wrangler secret put RESEND_API_KEY`。
4. 送信元として、初期は Resend のテスト用ドメイン（`onboarding@resend.dev`）を使う。後でドメイン認証すれば任意の送信元に変更可能。

5.3.2 Worker からの送信

Worker の投稿処理の最後に、`ctx.waitUntil` を使って非同期にメールを送る。これによりレスポンスを遅延させずに通知できる。

```
async function sendNotification(env, board, handle, body, postId) {
  await fetch("https://api.resend.com/emails", {
    method: "POST",
    headers: {
      "Authorization": "Bearer " + env.RESEND_API_KEY,
      "Content-Type": "application/json",
    },
    body: JSON.stringify({
      from: "Board Notifier <onboarding@resend.dev>",
      to:   [env.NOTIFY_EMAIL],
      subject: `掲示板[] ${board} に新規投稿 (${handle}) `,
      text: `投稿者: ${handle}\n\n${body}\n\n---\nID: ${postId}`,
    }),
  });
}
```

メールが迷惑メールフォルダに振り分けられる場合は、メールクライアントの「迷惑メールではない」設定や Safe senders への追加で改善できる。長期的には独自ドメインを Resend で認証すると到達率が大きく向上する。

5.4 管理画面

不適切な投稿を削除するための簡易管理画面を別ページに用意する。仕組みは単純で：

- ブラウザでパスワード入力 → `sessionStorage` に保存
- API へのリクエスト時に `X-Admin-Password` ヘッダで送信

- Worker 側でシークレット ADMIN_PASSWORD と定数時間比較で検証
- 検証 OK なら一覧取得・削除を許可

削除はソフト削除（deleted_at を埋めるだけ）にしておくと、誤削除時の復元やログ確認が容易になる。

6 運用と保守

6.1 コードの更新

サイトのコンテンツを更新したい時は、ローカルで HTML を書き換えて git push すれば、GitHub Pages は数分で再ビルドされ反映される。Worker 側のコードを変更した場合は npx wrangler deploy を実行する。

6.2 過去データの取り込み

旧来の Google Forms 等で蓄積された過去の投稿は、CSV / xlsx をエクスポートし、整形した INSERT 文を D1 に流し込むことで一括投入できる。

```
npx wrangler d1 execute my-board --remote --file=./seed.sql
```

6.3 無料枠の確認

GitHub Pages　リポジトリ 1 GB、月 100 GB 帯域、ソフトな制限のみ。
Cloudflare Workers　1 日 10 万リクエスト、CPU 時間 10 ms / リクエスト。
Cloudflare D1　ストレージ 5 GB、月 500 万行読み取り、月 1 億行書き込み。
Cloudflare Turnstile　完全無制限。
Resend　月 3,000 通、1 日 100 通。

教科書サポート規模であれば、これらの上限を超えることはまずない。仮に超えた場合も、Cloudflare・Resend ともに月 5〜20 USD 程度の安価な有料プランで大幅に増額できる。

6.4 セキュリティ・プライバシーの注意

- ADMIN_PASSWORD、RESEND_API_KEY、TURNSTILE_SECRET_KEY、IP_HASH_SALT は必ず Worker のシークレットとして登録し、Git リポジトリには絶対にコミットしない。
- GitHub のリポジトリは public でも問題ないが、.gitignore で機密ファイル・元データ（個人情報を含み得る xlsx 等）を除外する。
- 訪問者の IP は原文ではなく salted SHA-256 ハッシュで保存することで、個人識別性を最低限に抑えつつレート制限を機能させる。

7　まとめ

教科書サポートページに掲示板を付けるための、無料サービスを使った構成を紹介した。要点を再掲する。

- フロントは GitHub Pages で配信、バックエンドは Cloudflare Workers + D1。
- LaTeX 数式は KaTeX、CAPTCHA は Cloudflare Turnstile、メール通知は Resend。
- スパム対策は CAPTCHA・レート制限・ハニーポットの三重で実用上は十分。
- すべて無料枠で運用可能。

旧来の Google Sites / Forms と比較した利点は、構成がシンプルでサービス間連携の破綻が起きにくく、長期運用に耐える点である。教科書サポートページのみならず、研究室の Q&A 用掲示板や、学会プレゼンテーション後の質問受付など、似た用途にも応用できる。

付録 A　参考リンク

GitHub Pages　https://docs.github.com/pages
Cloudflare Workers　https://developers.cloudflare.com/workers/
Cloudflare D1　https://developers.cloudflare.com/d1/
Cloudflare Turnstile　https://developers.cloudflare.com/turnstile/
KaTeX　https://katex.org/
Resend　https://resend.com/