Supabaseの紹介

Press Space for next page →

自己紹介

- 📝 飯野陽平（wheatandcat）
- 🏢 法人設立（合同会社UNICORN 代表社員）
- 💻 Work: シェアフル株式会社CTO
- 📚 Blog: https://www.wheatandcat.me/
- 🛠️ 今までに作ったもの
  - memoir
  - ペペロミア
  - MarkyLinky

Supabaseとは？

- Firebase代替として注目を集めているサービス
- 現在は以下の機能を提供している
  - Database
  - Authentication
  - Storage
  - Edge Functions

Firebaseとの違い

- 完全オープンソース
- データベースはPostgreSQLを使用
- コンソール画面が非常にモダン
- 機能数はFirebaseの方が多い

Authentication

- 認証周りは以下のようなサービスと連携して認証が可能
  - ※ダッシューボードを見せながら説明
  - リンク
- 設定はシンプルで使いたいサービスを選択して有効にするだけでOK 👌

コード

以下のようなコードで認証は実装可能[1]

```
const handleOAuthLogin = async (provider: Provider, scopes = "email") => {
  await supabase.auth.signInWithOAuth({
    provider,
    options: {
      scopes,
      redirectTo: location.href
    }
  })
}

略)

// ログイン後は以下から認証情報を取得できる
const { data, error } = await supabase.auth.getSession()
```

---

1. Learn More

コード

以下のようなコードで認証は実装可能[1]

```
const handleOAuthLogin = async (provider: Provider, scopes = "email") => {
  await supabase.auth.signInWithOAuth({
    provider,
    options: {
      scopes,
      redirectTo: location.href
    }
  })
}

略)

// ログイン後は以下から認証情報を取得できる
const { data, error } = await supabase.auth.getSession()
```

---

1. Learn More

コード

以下のようなコードで認証は実装可能[1]

```
const handleOAuthLogin = async (provider: Provider, scopes = "email") => {
  await supabase.auth.signInWithOAuth({
    provider,
    options: {
      scopes,
      redirectTo: location.href
    }
  })
}

略)

// ログイン後は以下から認証情報を取得できる
const { data, error } = await supabase.auth.getSession()
```

---

1. Learn More

コード

以下のようなコードで認証は実装可能[1]

```
const handleOAuthLogin = async (provider: Provider, scopes = "email") => {
  await supabase.auth.signInWithOAuth({
    provider,
    options: {
      scopes,
      redirectTo: location.href
    }
  })
}

略)

// ログイン後は以下から認証情報を取得できる
const { data, error } = await supabase.auth.getSession()
```

---

1. Learn More

Database

- PostgreSQLを使用できる
- Webコンソールの作りが良いのでSQLクライアントは不要
- クライアントから直接アクセス可能 & セキュリティ面も安全に実装可能
    - Supabase自体が認証の情報を持っているので次のような設定が可能

Databaseのセキュリティポリシーの設定①

- 以下のSQLを実行して、tableを作成
- ```
  create table
    public.items (
      id integer not null default nextval('items_id_seq'::regclass),
      user_id uuid not null,
      title text not null,
      url text not null,
      favIconUrl text null,
      created timestamp with time zone not null default current_timestamp,
      constraint items_pkey primary key (id)
    ) tablespace pg_default;
  ```
  ```

Databaseのセキュリティポリシーの設定②

- 以下のSQLを実行して、tableにセキュリティポリシーを設定
- ```
  CREATE policy "All own items"
  ON items
  FOR ALL
  USING ( auth.uid() = user_id );
  ```
- 上記を設定することで、items.user_id == **認証済みのuid**のレコードのみアクセス可能な設定になる
  ```

Databaseのセキュリティポリシーの設定②

- 以下のSQLを実行して、tableにセキュリティポリシーを設定
- ```
  CREATE policy "All own items"
  ON items
  FOR ALL
  USING ( auth.uid() = user_id );
  ```
- 上記を設定することで、items.user_id == **認証済みのuid**のレコードのみアクセス可能な設定になる

- SupabaseのDatabaseクライントを使用してSQLを実行すると以下の動作になる
  ```

Databaseのセキュリティポリシーの設定②

- 以下のSQLを実行して、tableにセキュリティポリシーを設定
- ```
  CREATE policy "All own items"
  ON items
  FOR ALL
  USING ( auth.uid() = user_id );
  ```
- 上記を設定することで、items.user_id == **認証済みのuid**のレコードのみアクセス可能な設定になる

- SupabaseのDatabaseクライントを使用してSQLを実行すると以下の動作になる
  - ⬜ コード
    - ```
      const { data, error } = await supabase.from('items').select('*')
      ```

Databaseのセキュリティポリシーの設定②

- 以下のSQLを実行して、tableにセキュリティポリシーを設定
- ```
  CREATE policy "All own items"
  ON items
  FOR ALL
  USING ( auth.uid() = user_id );
  ```
- 上記を設定することで、items.user_id == **認証済みのuid**のレコードのみアクセス可能な設定になる

- SupabaseのDatabaseクライントを使用してSQLを実行すると以下の動作になる
  - ⬜ コード
    - ```
      const { data, error } = await supabase.from('items').select('*')
      ```
  - ⬜ 実際に実行されるSQL
    - ```
      SELECT * FROM items WHERE auth.uid() = items.user_id
      ```
      ```

Databaseの型安全①

- Supabaseは公式でDatabaseの設計からTypeScriptの型変換をサポートしている

Databaseの型安全①

- Supabaseは公式でDatabaseの設計からTypeScriptの型変換をサポートしている
- 以下のコマンドでログイン
  - ```
    $ npx supabase login
    ```
    ```

Databaseの型安全①

- Supabaseは公式でDatabaseの設計からTypeScriptの型変換をサポートしている
- 以下のコマンドでログイン
  - ```
    $ npx supabase login
    ```
- 以下のコマンドで初期設定
  - ```
    $ npx supabase init
    $ npx supabase link --project-ref <プロジェクトのID>
    ```

Databaseの型安全①

- Supabaseは公式でDatabaseの設計からTypeScriptの型変換をサポートしている
- 以下のコマンドでログイン
  - ```
    $ npx supabase login
    ```
- 以下のコマンドで初期設定
  - ```
    $ npx supabase init
    $ npx supabase link --project-ref <プロジェクトのID>
    ```
- 以下のコマンドでTypeScriptの型情報を生成
  - ```
    $ npx supabase gen types typescript --linked > schema.ts
    ```
  - 生成されたファイル → schema.ts
    ```

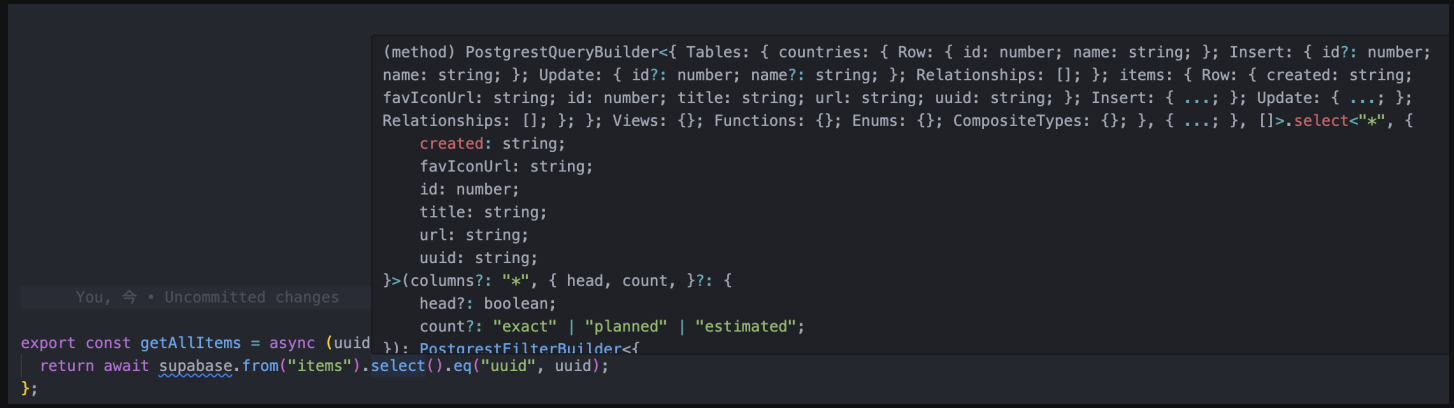
Databaseの型安全②

- 先程作成したファイルをimportして、型情報を設定する

```ts
import { createClient } from "@supabase/supabase-js";
import type { Database } from "../schema";

export const supabase = createClient<Database>(
  process.env.PLASMO_PUBLIC_SUPABASE_URL,
  process.env.PLASMO_PUBLIC_SUPABASE_KEY,
);
```

- これで補完が効くようになる



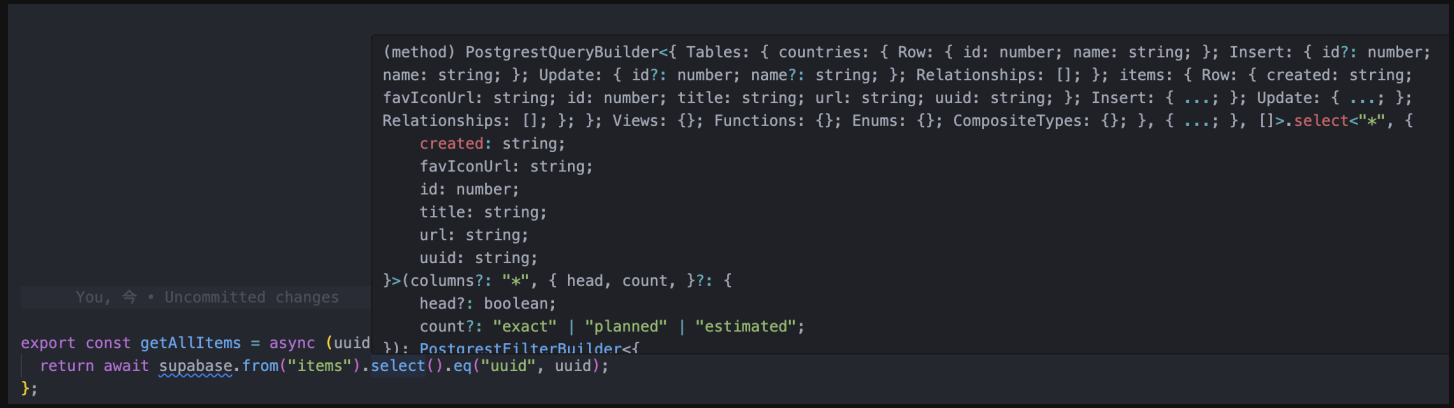
Databaseの型安全②

- 先程作成したファイルをimportして、型情報を設定する

```ts
import { createClient } from "@supabase/supabase-js";
import type { Database } from "../schema";

export const supabase = createClient<Database>(
  process.env.PLASMO_PUBLIC_SUPABASE_URL,
  process.env.PLASMO_PUBLIC_SUPABASE_KEY,
);
```

- これで補完が効くようになる



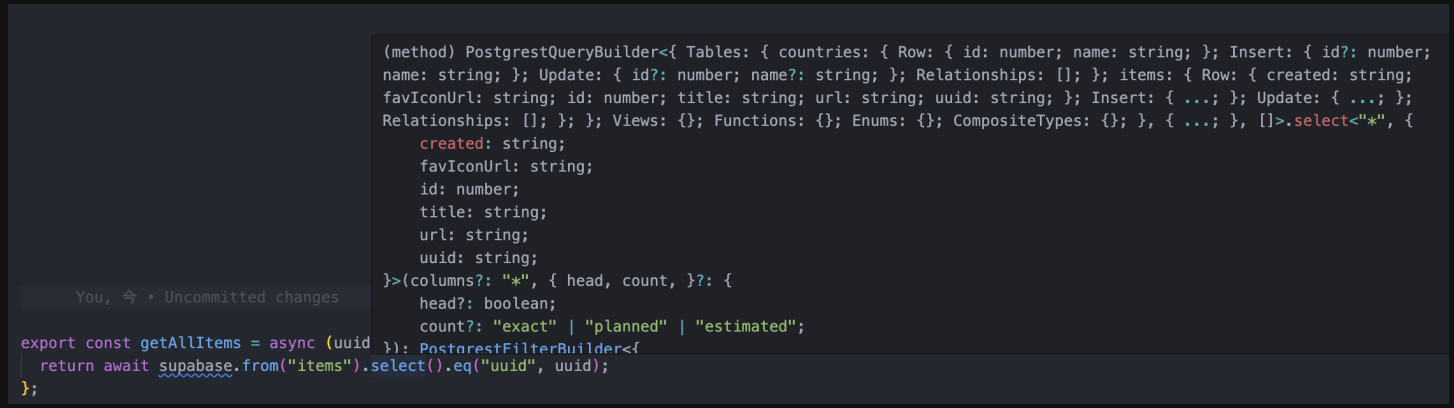
Databaseの型安全②

- 先程作成したファイルをimportして、型情報を設定する

```ts
import { createClient } from "@supabase/supabase-js";
import type { Database } from "../schema";

export const supabase = createClient<Database>(
  process.env.PLASMO_PUBLIC_SUPABASE_URL,
  process.env.PLASMO_PUBLIC_SUPABASE_KEY,
);
```

- これで補完が効くようになる



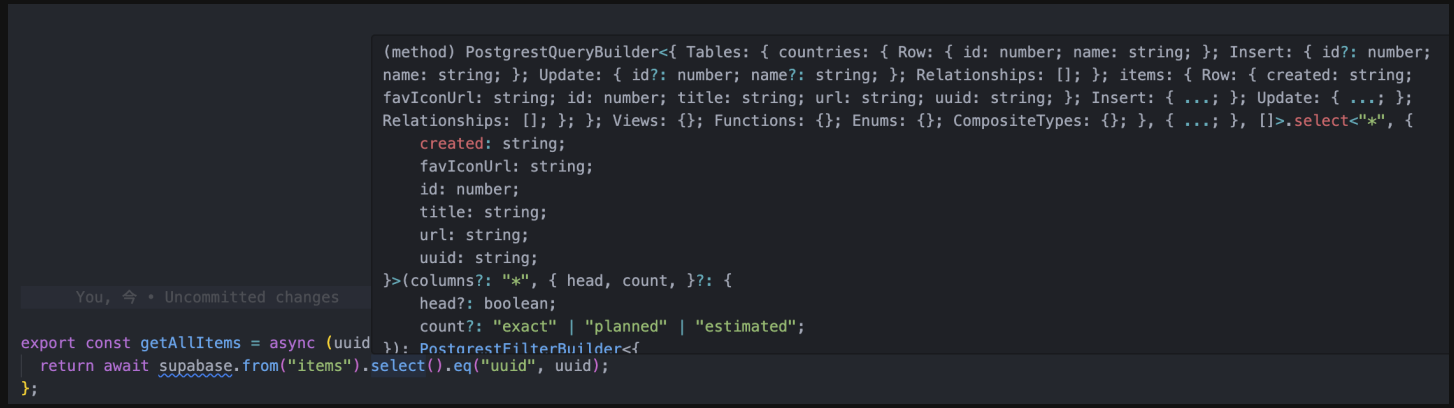
Databaseの型安全②

- 先程作成したファイルをimportして、型情報を設定する

```ts
import { createClient } from "@supabase/supabase-js";
import type { Database } from "../schema";

export const supabase = createClient<Database>(
  process.env.PLASMO_PUBLIC_SUPABASE_URL,
  process.env.PLASMO_PUBLIC_SUPABASE_KEY,
);
```

- これで補完が効くようになる



その他の特徴

- 簡単にローカル環境が構築できる
  - 参考: Supabaseのローカル開発環境構築
- DB操作の基本APIを自動生成してくれる
  - 参考 Serverless APIs
- リアルタイムリスナーをサポート
  - 参考: Supabase JSのリアルタイムリスナーを使ってみる
- 有料版なら自動でバックアップを取ってくれる
  - 参考: Pricing & fees | Supabase

その他の機能紹介

Storage

- 画像、ビデオ、テキスト等のファイルが保存できる
- セキュリティポリシーを設定することで、アクセス制限が可能
- コンソール画面からバケットに対してアップロードできるファイルサイズや、ファイル形式を制限できる

Edge Functions

- FaaS
- TypeScript、Deno、Wasmをサポート

実装

- 以下のPRで実装している
  - supabaseでログイン + データ同期機能を追加 by wheatandcat
- デモで以下を紹介
  - ログイン画面
  - データ同期

まとめ

- Supabaseは個人開発で使うには、かなり良いサービス
  - コンソールが使いやすい
  - セキュリティ周りも、わかりやすい
  - 従量制課金では無いので、コストも安心
- プロダクトで使う場合は制限周りを確認する必要がある
  - Pricing & fees | Supabase

ご清聴ありがとうございました🎉