Efficient App Development with Supabase Functions

supabase

サンプルプロジェクト

@vicktorManuel

http://francodev.live

What is Supabase?
何はsupabase?

Supabase simplifies the development process by offering ready-to-use backend features. It uses PostgreSQL, a robust SQL database, allowing developers to efficiently and safely manage data using SQL techniques

Supabase は、すぐに使用できるバックエンド機能を提供することで、開発プロセスを簡素化します。

強力な SQL データベースである PostgreSQL を使用しており、開発者が SQL 技術を使用して効率的かつ安全にデータを管理できるようにしています

supabase

Services

AUTHORIZATION SERVICE

REALTIME DATABASE

STORAGE

Understanding Functions in Supabase

Edge Functions

Written in **TypeScript** or **JavaScript** using **Deno**.

Example: Handling webhooks, performing real-time operations.

TypeScript または JavaScript で Deno を使用して記述。

例: Webhooks の処理、リアルタイム操作の実行。

SQL FUNCTIONS

Written in **PL/pgSQL** (SQL with procedural extensions).

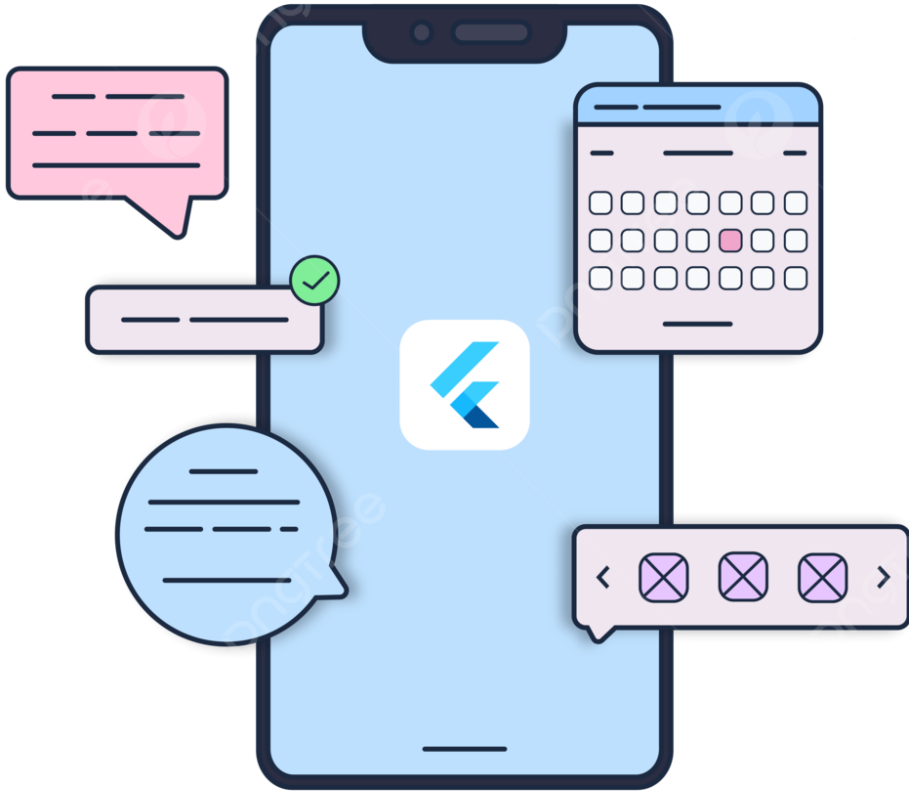
Example: Updating user scores, processing transactions.

PL/pgSQL（手続き型拡張を持つSQL）で記述。

例: ユーザーのスコアを更新、トランザクションの処理。

Why Use Supabase Functions?

- **Simplify Complex Logic:** Move complex business logic from client to server.
- **Security:** Better security control with server-side validation.
- **Performance:** Reduce network requests by combining multiple operations in a single function.
- 複雑なロジックを簡素化: クライアントからサーバーに複雑なビジネスロジックを移動させる。
- セキュリティ: サーバー側での検証によるより良いセキュリティ管理。
- パフォーマンス: 複数の操作を1つの関数に組み合わせることで、ネットワークリクエストを削減する。



Language plpgsql

```
create or replace function INSERT_USERS(count integer)
returns text as $$
declare
   i integer;
 begin
 for i in 1..count LOOP
 insert into users (name,points) values ('user' || i, 1);
 END LOOP;
 END;
 $$ language plpgsql;
```

これから面白くなるぞ！

Functions sql vs. App-Side Logic

users	
id	integer
name	text
points	integer
create_at	timestamp

```
  Future<void> insert100Users() async {
    for (var i = 0; i < 100; i++) {
      await Supabase.instance.client
          .from('users')
          .insert({'name': 'name $i', 'points': 1}).then((value)
{
        print("insert!");
      }).catchError((error) {
        print(error);
      });
    }
  }
```

```
  Future<void> insert100UsersFunctionSQL() async {
    await Supabase.instance.client
        .rpc('create_users', params: {'count': 100}).then((value)
{
      print("insert ok!");
    }).catchError((error) {
      print(error);
    });
  }
```

CAR SHOP EXAMPLE

Task

- Checks Stock Availability
- Calculates Total Price
- Updates Product Quantity
- Records the Purchase
- Updates User Points
- 在庫の確認
- 合計金額の計算
- 商品の数量を更新
- 購入を記録
- ユーザーポイントを更新