🚀 Supabase Integration - Setup Guide

Überblick

Diese Anleitung zeigt dir, wie du die Pokémon Card Scanner App mit Supabase verbindest, damit User ihre Karten-Sammlungen speichern können.

📋 Voraussetzungen

- ✅ Flutter Projekt ist bereits erstellt
- ✅ Supabase Account (kostenlos bei supabase.com)

1️⃣ Supabase Projekt erstellen

Schritt 1: Neues Projekt anlegen

1. Gehe zu supabase.com
2. Klicke auf **"New Project"**
3. Wähle:
   - **Organization**: Deine Organisation (oder erstelle eine neue)
   - **Name**: `pokemon-card-scanner`
   - **Database Password**: Wähle ein sicheres Passwort
   - **Region**: `Europe (Frankfurt)` (für deutschen Markt optimal)
4. Klicke auf **"Create new project"**
5. ⏰ Warte ~2 Minuten bis Projekt bereit ist

2️⃣ Datenbank-Schema einrichten

Schritt 1: SQL Editor öffnen

1. Gehe in deinem Supabase-Projekt zu **"SQL Editor"** (linkes Menü)
2. Klicke auf **"New Query"**

Schritt 2: Schema ausführen

1. Kopiere den **kompletten SQL-Code** aus dem Artifact `Supabase SQL Schema`
2. Füge ihn in den SQL Editor ein

3. Klicke auf **"Run"** (oder `Cmd/Ctrl + Enter`)
4. ✅ Du solltest sehen: `Success. No rows returned`

Was wurde erstellt?

- **Tabelle** `profiles`: User-Profile (Username, Avatar)
- **Tabelle** `user_cards`: Karten-Sammlung mit allen Details
- **Row Level Security**: Automatischer Datenschutz (User A sieht nur seine Karten)
- **Indexes**: Für schnelle Abfragen
- **Trigger**: Automatische Timestamps & Profil-Erstellung

3️⃣ API Keys holen

Schritt 1: Project Settings öffnen

1. Klicke auf **Settings** (⚙️ Icon unten links)
2. Gehe zu **"API"**

Schritt 2: Keys kopieren

Du brauchst zwei Werte:

A) Project URL

```
https://deinproject.supabase.co
```

B) Anon/Public Key (langer String mit `eyJ...`)

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

⚠️ **WICHTIG**: Nutze den `anon public` Key, NICHT den `service_role` Key!

4️⃣ Flutter App konfigurieren

Schritt 1: main.dart öffnen

Öffne `lib/main.dart` und ersetze die Platzhalter:

```dart
```

```
// ⚠️ WICHTIG: Ersetze diese Werte mit deinen echten Supabase Credentials
const String SUPABASE_URL = 'https://DEIN-PROJECT.supabase.co';
const String SUPABASE_ANON_KEY = 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...';
```

Schritt 2: Dependencies installieren

```
bash

flutter pub get
```

Schritt 3: App starten

```
bash

flutter run
```

5️⃣ Testen

Test 1: Registrierung

1. Öffne die App
2. Tippe auf das **Login-Icon** (oben rechts)
3. Registriere einen Test-Account:
   - **E-Mail**: `test@example.com`
   - **Passwort**: `test1234`
4. ✅ Du solltest sehen: "Account erstellt! Bitte E-Mail bestätigen."

Test 2: E-Mail bestätigen

1. Gehe zu Supabase Dashboard
2. Öffne **"Authentication"** > **"Users"**
3. Klicke auf deinen Test-User
4. Klicke **"Confirm User"** (oben rechts)

Test 3: Login

1. Zurück in der App
2. Logge dich ein mit `test@example.com` / `test1234`
3. ✅ Du solltest sehen: "Erfolgreich eingeloggt!"

Test 4: Karte scannen & speichern

1. Tippe auf **"Karte scannen"**
2. Halte eine Pokémon-Karte vor die Kamera
3. Wenn erkannt: Tippe auf **"Zur Sammlung"**
4. Gehe zurück zum Homescreen
5. Tippe auf das **Profil-Icon**
6. ✅ Du solltest deine Karte in der Sammlung sehen!

6️⃣ Row Level Security (RLS) Überprüfen

Was ist RLS?

Row Level Security sorgt dafür, dass:

- **User A** nur seine eigenen Karten sehen kann
- **User B** NICHT die Karten von User A sehen/löschen kann
- Automatischer Datenschutz auf Datenbankebene

RLS testen

1. Erstelle 2 Test-Accounts: `user1@test.com` und `user2@test.com`
2. Scanne mit `user1` eine Karte und speichere sie
3. Logge dich aus und logge dich mit `user2` ein
4. ✅ User 2 sollte KEINE Karten von User 1 sehen

RLS in Supabase prüfen

1. Gehe zu **"Database"** > **"Tables"** > **"user_cards"**
2. Klicke auf **"Policies"**
3. ✅ Du solltest 4 Policies sehen:
   - `Users can view own cards`
   - `Users can insert own cards`
   - `Users can update own cards`
   - `Users can delete own cards`

---

7 Datenbank-Struktur verstehen

Tabelle `user_cards`

Spalte	Typ	Beschreibung
`id`	bigint	Primärschlüssel (auto-increment)
`user_id`	uuid	Foreign Key zu `auth.users`
`api_card_id`	text	Eindeutige ID von Pokémon TCG API
`name`	text	Kartenname (z.B. "Pikachu")
`set_name`	text	Set-Name (z.B. "Base Set")
`card_number`	text	Kartennummer (z.B. "025/165")
`image_url`	text	URL zum Kartenbild
`rarity`	text	Seltenheit (Common, Rare, etc.)
`condition`	text	Zustand (Near Mint, Played, etc.)
`purchase_price_eur`	numeric	Kaufpreis in Euro
`current_market_price_eur`	numeric	Aktueller Marktwert
`quantity`	integer	Anzahl (Standard: 1)
`is_foil`	boolean	Ist die Karte Holo/Foil?
`notes`	text	User-Notizen
`created_at`	timestamp	Erstellungsdatum
`updated_at`	timestamp	Letzte Änderung

Tabelle `profiles`

Spalte	Typ	Beschreibung
`id`	uuid	Primärschlüssel (= `user_id`)
`username`	text	Benutzername (unique)
`avatar_url`	text	Profilbild-URL
`created_at`	timestamp	Registrierungsdatum
`updated_at`	timestamp	Letzte Änderung

---

8 Erweiterte Features (Optional)

Feature 1: Marktpreise aktualisieren

Du kannst später eine Edge Function erstellen, die automatisch die Marktpreise aktualisiert:

```sql
```

```sql
-- Beispiel: Alle Preise auf NULL setzen (zum Testen)
update public.user_cards
set current_market_price_eur = null;
```

Feature 2: Sammlungs-Statistiken

Nutze die vorbereitete View:

```sql
-- Statistiken für aktuellen User
select * from public.user_collection_stats
where user_id = auth.uid();
```

In Flutter:

```dart
final stats = await supabase
    .from('user_collection_stats')
    .select()
    .eq('user_id', supabase.auth.currentUser!.id)
    .single();

print('Total Cards: ${stats['total_cards']}');
print('Collection Value: ${stats['total_collection_value']}€');
```

Feature 3: Top-Wert-Karten

Nutze die Funktion:

```dart
final topCards = await supabase
    .rpc('get_top_value_cards', params: {
      'p_user_id': supabase.auth.currentUser!.id,
      'p_limit': 10,
    });

print('Wertvollste Karten: $topCards');
```

9️⃣ Troubleshooting

Problem: "Row Level Security violated"

Lösung: Du versuchst auf Daten zuzugreifen, die nicht dir gehören.

- Prüfe ob User eingeloggt ist: `supabase.auth.currentUser != null`
- Prüfe RLS Policies in Supabase Dashboard

Problem: "JWT expired" oder "Invalid token"

Lösung: Session ist abgelaufen.

- Logge User erneut ein
- Supabase macht automatisches Token-Refresh

Problem: Karten werden nicht gespeichert

Lösung:

1. Prüfe ob User eingeloggt ist
2. Schaue in Supabase Dashboard > **Table Editor** > **user_cards**
3. Prüfe die Logs in der Flutter Console

Problem: "Insert violates Row Level Security"

Lösung: Die `user_id` stimmt nicht überein.

- Stelle sicher, dass `user_id` = `auth.uid()` in deinem Insert

🔒 Sicherheits-Best Practices

✅ DO's

- ✅ Nutze IMMER Row Level Security (RLS)
- ✅ Speichere Supabase Keys in `.env` Files (nicht in Git)
- ✅ Nutze den `anon` Key in der App, NIEMALS `service_role`
- ✅ Validiere User-Input im Flutter Code
- ✅ Nutze HTTPS für alle API-Calls

❌ DON'Ts

- ❌ Committe NIEMALS deine Keys in Git
- ❌ Deaktiviere NIEMALS Row Level Security
- ❌ Gebe NIEMALS deinen `service_role` Key weiter
- ❌ Speichere KEINE sensiblen Daten in `localStorage`

📊 Performance-Tipps

1. Indexes nutzen

Die Schema-Datei hat bereits optimale Indexes:

- `user_cards_user_id_idx` - Schnelle User-Lookups
- `user_cards_api_card_id_idx` - Duplikat-Checks
- `user_cards_created_at_idx` - Sortierung nach Datum

2. Batch-Operationen

Statt einzelnen Inserts:

```dart
// ❌ Langsam
for (var card in cards) {
  await supabase.from('user_cards').insert(card);
}

// ✅ Schnell
await supabase.from('user_cards').insert(cards);
```

3. Select nur benötigte Spalten

```dart
// ❌ Lädt alle Spalten
final cards = await supabase.from('user_cards').select();

// ✅ Lädt nur Name und Bild
final cards = await supabase
    .from('user_cards')
    .select('name, image_url');
```

🎉 Fertig!

Du hast jetzt eine vollständig funktionierende Supabase-Integration!

Next Steps:

1. ✅ Teste ausgiebig mit mehreren Accounts
2. ✅ Implementiere weitere Features (Preis-Updates, Social Features)
3. ✅ Deploye deine App auf iOS/Android

📞 Support

Dokumentation: supabase.com/docs
Community: discord.gg/supabase

Happy Coding! 🚀