Tutorial Konfigurasi Supabase untuk Aplikasi LaporKerja

Pendahuluan

Panduan ini berisi langkah-langkah lengkap untuk menyiapkan backend Supabase bagi aplikasi **LaporKerja**, sebuah aplikasi manajemen pekerjaan khusus freelancer. Aplikasi ini menggunakan arsitektur *offline-first* dengan Flutter sebagai frontend dan Drift sebagai database lokal.

Arsitektur dan Tech Stack

LaporKerja adalah aplikasi Android yang dirancang untuk membantu freelancer mengelola:

- Proyek dan tugas dari berbagai klien
- Pelacakan waktu kerja (time tracking)
- Manajemen pendapatan dan status pembayaran
- Penyimpanan data klien

Tech Stack yang Digunakan:

- **Frontend:** Flutter (Dart)
- **Backend:** Supabase (PostgreSQL)
- **Database Lokal:** Drift (SQLite)
- **State Management:** Riverpod + Freezed
- **Arsitektur:** Offline-first dengan sinkronisasi cloud

Langkah 1: Inisialisasi Proyek Supabase

1.1 Membuat Proyek Baru

1. Buka supabase.com dan masuk ke akun Anda
2. Klik tombol **"New Project"**
3. Pilih **Organization** yang sesuai
4. Isi detail proyek:
   - **Project Name:** LaporKerja
   - **Database Password:** Buat password yang kuat dan simpan dengan aman
   - **Region:** Pilih Asia Pacific (Singapore) untuk performa terbaik di Indonesia
5. Klik **"Create new project"** dan tunggu hingga proses selesai (biasanya 1-2 menit)

Langkah 2: Konfigurasi Autentikasi

2.1 Mengaktifkan Provider Email

1. Dari dashboard proyek, buka menu **Authentication** di sidebar kiri
2. Navigasi ke **Configuration → Providers**
3. Temukan provider **Email** dan klik toggle untuk mengaktifkannya
4. Biarkan pengaturan lainnya sebagai default

2.2 Pengaturan untuk Development (Opsional)

Untuk mempercepat proses testing selama development:

1. Buka **Configuration → Email Templates**
2. **Matikan** toggle **"Confirm email"**
3. **PENTING:** Ingat untuk mengaktifkan kembali sebelum rilis ke publik!

Langkah 3: Membuat Storage Buckets

3.1 Bucket untuk Avatar (Public)

1. Buka menu **Storage** dari sidebar
2. Klik **"New Bucket"**
3. Isi detail bucket:
   - **Bucket name:** `avatars`
   - **Public bucket:** ✅ Centang (untuk akses publik foto profil)
4. Klik **"Create bucket"**

3.2 Bucket untuk File Proyek (Private)

1. Klik **"New Bucket"** lagi
2. Isi detail bucket:
   - **Bucket name:** `project_files`
   - **Public bucket:** ❌ Jangan centang (untuk keamanan file pribadi)
3. Klik **"Create bucket"**

Langkah 4: Menjalankan Skrip SQL Lengkap

4.1 Membuka SQL Editor

1. Buka menu **SQL Editor** dari sidebar
2. Klik **"+ New query"** untuk membuat query baru

4.2 Skrip SQL Lengkap

Salin dan tempel seluruh skrip SQL berikut ke editor, lalu klik **"RUN"**:

```
-- ================================================================
-- SKRIP SQL LENGKAP UNTUK APLIKASI LAPORKERJA (OFFLINE-FIRST)
-- ================================================================

-- BAGIAN 1: FUNGSI HELPER
-- Fungsi untuk memperbarui kolom 'updated_at' secara otomatis
CREATE OR REPLACE FUNCTION update_updated_at_column()
RETURNS TRIGGER AS $$
BEGIN
   NEW.updated_at = now();
   RETURN NEW;
END;
$$ language 'plpgsql';

-- BAGIAN 2: PEMBUATAN TABEL
-- Tabel 'profiles' untuk data publik user
CREATE TABLE profiles (
  id UUID PRIMARY KEY REFERENCES auth.users ON DELETE CASCADE,
  username TEXT UNIQUE,
  avatar_url TEXT,
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT now()
);

-- Tabel 'clients' untuk data klien
CREATE TABLE clients (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES auth.users(id) ON DELETE CASCADE,
  local_id TEXT UNIQUE,
  name VARCHAR(255) NOT NULL,
  contact_info VARCHAR(255),
  created_at TIMESTAMP WITH TIME ZONE DEFAULT now(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT now(),
  last_modified_locally TIMESTAMP WITH TIME ZONE,
  is_deleted BOOLEAN DEFAULT FALSE
);

-- Tabel 'projects' untuk proyek freelancer
CREATE TABLE projects (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES auth.users(id) ON DELETE CASCADE,
  local_id TEXT UNIQUE,
  client_id UUID REFERENCES clients(id) ON DELETE SET NULL,
  project_name VARCHAR(255) NOT NULL,
  description TEXT,
  start_date DATE,
  deadline DATE,
```

```
  status VARCHAR(50) DEFAULT 'ongoing',
  created_at TIMESTAMP WITH TIME ZONE DEFAULT now(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT now(),
  last_modified_locally TIMESTAMP WITH TIME ZONE,
  is_deleted BOOLEAN DEFAULT FALSE
);

-- Tabel 'tasks' untuk tugas-tugas proyek
CREATE TABLE tasks (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES auth.users(id) ON DELETE CASCADE,
  local_id TEXT UNIQUE,
  project_id UUID NOT NULL REFERENCES projects(id) ON DELETE CASCADE,
  task_name VARCHAR(255) NOT NULL,
  description TEXT,
  status VARCHAR(50) DEFAULT 'todo',
  deadline DATE,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT now(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT now(),
  last_modified_locally TIMESTAMP WITH TIME ZONE,
  is_deleted BOOLEAN DEFAULT FALSE
);

-- Tabel 'time_entries' untuk pelacakan waktu
CREATE TABLE time_entries (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES auth.users(id) ON DELETE CASCADE,
  local_id TEXT UNIQUE,
  task_id UUID NOT NULL REFERENCES tasks(id) ON DELETE CASCADE,
  start_time TIMESTAMP WITH TIME ZONE NOT NULL,
  end_time TIMESTAMP WITH TIME ZONE,
  duration INTERVAL,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT now(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT now(),
  last_modified_locally TIMESTAMP WITH TIME ZONE,
  is_deleted BOOLEAN DEFAULT FALSE
);

-- Tabel 'incomes' untuk pendapatan proyek
CREATE TABLE incomes (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES auth.users(id) ON DELETE CASCADE,
  local_id TEXT UNIQUE,
  project_id UUID NOT NULL REFERENCES projects(id) ON DELETE CASCADE,
  amount NUMERIC(12, 2) NOT NULL,
  payment_status VARCHAR(50) DEFAULT 'unpaid',
  payment_date DATE,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT now(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT now(),
  last_modified_locally TIMESTAMP WITH TIME ZONE,
  is_deleted BOOLEAN DEFAULT FALSE
);

-- BAGIAN 3: PEMASANGAN TRIGGER
-- Menerapkan auto-update timestamp ke semua tabel
CREATE TRIGGER update_clients_updated_at
```

```
  BEFORE UPDATE ON clients
  FOR EACH ROW EXECUTE PROCEDURE update_updated_at_column();

CREATE TRIGGER update_projects_updated_at
  BEFORE UPDATE ON projects
  FOR EACH ROW EXECUTE PROCEDURE update_updated_at_column();

CREATE TRIGGER update_tasks_updated_at
  BEFORE UPDATE ON tasks
  FOR EACH ROW EXECUTE PROCEDURE update_updated_at_column();

CREATE TRIGGER update_time_entries_updated_at
  BEFORE UPDATE ON time_entries
  FOR EACH ROW EXECUTE PROCEDURE update_updated_at_column();

CREATE TRIGGER update_incomes_updated_at
  BEFORE UPDATE ON incomes
  FOR EACH ROW EXECUTE PROCEDURE update_updated_at_column();

-- BAGIAN 4: KONFIGURASI KEAMANAN (ROW-LEVEL SECURITY)
-- Mengaktifkan RLS untuk semua tabel
ALTER TABLE profiles ENABLE ROW LEVEL SECURITY;
ALTER TABLE clients ENABLE ROW LEVEL SECURITY;
ALTER TABLE projects ENABLE ROW LEVEL SECURITY;
ALTER TABLE tasks ENABLE ROW LEVEL SECURITY;
ALTER TABLE time_entries ENABLE ROW LEVEL SECURITY;
ALTER TABLE incomes ENABLE ROW LEVEL SECURITY;

-- Policies untuk tabel 'profiles'
CREATE POLICY "Public profiles are viewable by everyone."
  ON profiles FOR SELECT USING (true);
CREATE POLICY "Users can insert their own profile."
  ON profiles FOR INSERT WITH CHECK (auth.uid() = id);
CREATE POLICY "Users can update their own profile."
  ON profiles FOR UPDATE USING (auth.uid() = id);

-- Policies untuk tabel data pribadi
CREATE POLICY "Enable all access for own clients"
  ON clients FOR ALL USING (auth.uid() = user_id)
  WITH CHECK (auth.uid() = user_id);

CREATE POLICY "Enable all access for own projects"
  ON projects FOR ALL USING (auth.uid() = user_id)
  WITH CHECK (auth.uid() = user_id);

CREATE POLICY "Enable all access for own tasks"
  ON tasks FOR ALL USING (auth.uid() = user_id)
  WITH CHECK (auth.uid() = user_id);

CREATE POLICY "Enable all access for own time entries"
  ON time_entries FOR ALL USING (auth.uid() = user_id)
  WITH CHECK (auth.uid() = user_id);

CREATE POLICY "Enable all access for own incomes"
  ON incomes FOR ALL USING (auth.uid() = user_id)
  WITH CHECK (auth.uid() = user_id);
```

```
-- BAGIAN 5: KEAMANAN STORAGE
-- Policies untuk bucket 'avatars' (publik)
CREATE POLICY "Avatar images are publicly accessible."
  ON storage.objects FOR SELECT USING (bucket_id = 'avatars');
CREATE POLICY "Anyone can upload an avatar."
  ON storage.objects FOR INSERT WITH CHECK (bucket_id = 'avatars');
CREATE POLICY "Anyone can update their own avatar."
  ON storage.objects FOR UPDATE USING (auth.uid() = owner)
  WITH CHECK (bucket_id = 'avatars');

-- Policies untuk bucket 'project_files' (privat)
CREATE POLICY "Allow authenticated uploads to project files"
  ON storage.objects FOR INSERT TO authenticated
  WITH CHECK (bucket_id = 'project_files');
CREATE POLICY "Allow individual read access to project files"
  ON storage.objects FOR SELECT TO authenticated
  USING (auth.uid() = owner);
CREATE POLICY "Allow individual update access to project files"
  ON storage.objects FOR UPDATE TO authenticated
  USING (auth.uid() = owner);
CREATE POLICY "Allow individual delete access to project files"
  ON storage.objects FOR DELETE TO authenticated
  USING (auth.uid() = owner);

-- BAGIAN 6: OTOMATISASI PEMBUATAN PROFIL
-- Fungsi untuk membuat profil otomatis saat user baru mendaftar
CREATE OR REPLACE FUNCTION public.handle_new_user()
RETURNS TRIGGER AS $$
BEGIN
  INSERT INTO public.profiles (id)
  VALUES (new.id);
  RETURN new;
END;
$$ LANGUAGE plpgsql SECURITY DEFINER;

-- Trigger untuk memanggil fungsi di atas
CREATE OR REPLACE TRIGGER on_auth_user_created
  AFTER INSERT ON auth.users
  FOR EACH ROW EXECUTE PROCEDURE public.handle_new_user();
```

4.3 Verifikasi Hasil

Setelah menjalankan skrip, verifikasi bahwa:

1. Semua tabel telah dibuat dengan benar
2. RLS aktif pada semua tabel (indikator hijau di dashboard)
3. Policies telah terpasang
4. Tidak ada error yang muncul

Langkah 5: Mengambil Kredensial API

5.1 Akses Pengaturan API

1. Buka **Project Settings** (ikon gerigi di bagian bawah sidebar)
2. Pilih tab **API**

5.2 Salin Kredensial Penting

Dari halaman API, salin dan simpan dengan aman:

Project URL:

```
https://your-project-ref.supabase.co
```

Anon (public) Key:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

5.3 Keamanan API Keys

⚠ **PENTING:**

- **Gunakan HANYA** `anon (public) key` di aplikasi Flutter
- **JANGAN PERNAH** menggunakan `service_role` key di sisi klien
- Simpan kredensial dalam file `.env` atau `dotenv` untuk keamanan

Fitur-Fitur Utama yang Telah Disiapkan

Database dengan Arsitektur Offline-First

- **Kolom** `local_id`**:** Untuk memetakan data lokal ke cloud
- **Kolom** `last_modified_locally`**:** Untuk resolusi konflik data
- **Kolom** `is_deleted`**:** Untuk implementasi soft delete
- **UUID sebagai Primary Key:** Mencegah konflik ID antar perangkat

Sistem Keamanan Berlapis

- **Row-Level Security (RLS):** Isolasi data antar pengguna
- **Storage Policies:** Kontrol akses file berdasarkan kepemilikan
- **Autentikasi Email:** Sistem login yang aman

Storage Terorganisir

- **Bucket `avatars`:** Foto profil dengan akses publik
- **Bucket `project_files`:** File proyek dengan akses privat
- **Image Transformations:** Optimasi ukuran gambar otomatis

Otomatisasi Backend

- **Auto-update Timestamps:** Kolom `updated_at` dikelola otomatis
- **Auto-create Profiles:** Profil pengguna dibuat otomatis saat registrasi
- **Cascading Deletes:** Relasi data terjaga saat penghapusan

Pemeriksaan Akhir Sebelum Development

Checklist Keamanan

- [ ] RLS aktif di semua tabel data
- [ ] Policies terpasang dengan benar
- [ ] Storage policies dikonfigurasi sesuai kebutuhan
- [ ] API keys tersimpan dengan aman

Checklist Fungsionalitas

- [ ] Semua tabel database telah dibuat
- [ ] Trigger dan fungsi berjalan normal
- [ ] Storage buckets telah dibuat
- [ ] Autentikasi email aktif

Persiapan Production (untuk masa depan)

- [ ] Aktifkan kembali "Confirm email" sebelum rilis
- [ ] Pertimbangkan Custom SMTP untuk email profesional
- [ ] Aktifkan MFA pada akun Supabase
- [ ] Siapkan environment terpisah untuk development

Kesimpulan

Backend Supabase untuk aplikasi **LaporKerja** telah dikonfigurasi dengan lengkap dan siap untuk mendukung pengembangan aplikasi Flutter dengan arsitektur *offline-first*. Semua aspek penting telah ditangani:

1. **Struktur Database** yang mendukung sinkronisasi offline
2. **Keamanan Data** dengan RLS dan policies yang ketat
3. **Manajemen File** dengan storage yang terorganisir

4. **Otomatisasi** untuk mengurangi kompleksitas di sisi klien

Langkah selanjutnya adalah memulai pengembangan aplikasi Flutter dengan integrasi Drift untuk database lokal dan implementasi logika sinkronisasi data.

Panduan ini dibuat berdasarkan diskusi dan perencanaan lengkap untuk aplikasi LaporKerja. Untuk pertanyaan lebih lanjut atau pembaruan konfigurasi, selalu merujuk ke dokumentasi resmi Supabase di docs.supabase.com.