Nama : Isep Lutpi Nur
NPM : 2113191079
Kelas : Informatika / A2
Mata Kuliah : Mobile Programming Lanjut
Semester : 7 (Genap)
Tugas : Minggu 7

---

1. Sesuai dengan tugas 1 yang ditugaskan untuk mencari 2 judul aplikasi dengan Flutter dan PhoneGap, Cari dan anlisislah jenis database yang digunakan dalam aplikasi Flutter tersebut jika ada. Jika tidak ada, silahkan terangkan cara kerja penyimpanan data tersebut seperti apa dan bagaimana mekanismesnya.
2. Jangan lupa file tugasnya berbentuk pdf dan ukurannya di bawah 1 MB.

Source Code: https://kodingajar.com/mengelola-database-pada-aplikasi-flutter-dengan-studi-kasus-membuat-aplikasi-buku-tamu/

1. Database yang digunakan

   Dalam Projek ini menggunakan paket sqflite.

2. Tabel skema

   Dalam projek ini terdapat 2 tabel yaitu:

   A. Tabel Event untuk menampung data event/acara.

```
CREATE TABLE tblEvent (
    eventId INTEGER PRIMARY KEY AUTOINCREMENT,
    createDate INTEGER,
    eventDate INTEGER,
    eventTime INTEGER,
    eventName VARCHAR(50),
    eventAddress VARCHAR(500),
    isCompleted INTEGER
);
CREATE INDEX idx_tblEvent_row_event ON tblEvent (eventId);
```

   B. Tabel Guest untuk menampung data pengunjung acara, tabel ini mempunyai relasi ke tabel event banyak ke satu. Satu event bisa mempunyai banyak guest.

```
CREATE TABLE tblGuest (
    guestId INTEGER PRIMARY KEY AUTOINCREMENT,
```

```
    eventId INTEGER,
    guestFullName VARCHAR(50),
    guestNoPhone VARCHAR(20),
    guestEmail VARCHAR(50),
    guestAddress VARCHAR(500),
    guestNote TEXT,
    guestVisitTime INTEGER
);
CREATE INDEX idx_tblGuest_row_guest ON tblGuest (guestId,eventId);
```

3. Proses CRUD

   A. Tabel event (tblEvent)

      1. Create

```
Future<EventItem> insertEvent(EventItem event) async {
  event.eventId = await database.insert(
    tableEvent,
    event.toMapInsert(),
  );
  return event;
}
```

      2. Read

```
/* Fungsi untuk mengambil baris di tabel tblEvent */
Future<EventItem> getEvent(int id) async {
  List<Map> maps = await database.query(tableEvent,
      columns: getColumnsName(eventColumns),
      where: eventColumns[0].name + ' = ?',
      whereArgs: [id]);
  if (maps.isNotEmpty) {
    return EventItem.fromMap(maps.first);
  }
  return null;
}

/* Fungsi untuk mengambil daftar baris di tabel tblEvent */
Future<List<EventItem>> getListEvent() async {
  List<Map> maps = await database.query(
    tableEvent,
    columns: getColumnsName(eventColumns),
  );
  List<EventItem> listEvent = [];
  if (maps.isNotEmpty) {
```

```
    for (Map map in maps) {
      EventItem event = EventItem.fromMap(map);
      listEvent.add(event);
    }
    return listEvent;
  }
  return listEvent;
}
```

3. Update

```
Future<int> updateEvent(EventItem event) async {
  return await database.update(
    tableEvent,
    event.toMap(),
    where: eventColumns[0].name + ' = ?',
    whereArgs: [event.eventId],
  );
}
```

4. Delete

```
Future<int> deleteEvent(int id) async {
  return await database.delete(
    tableEvent,
    where: eventColumns[0].name + ' = ?',
    whereArgs: [id],
  );
}
```

B. Tabel event (tblEvent)

1. Create

```
Future<GuestItem> insertGuest(GuestItem guest) async {
  guest.guestId = await database.insert(
    tableGuest,
    guest.toMapInsert(),
  );
  return guest;
}
```

2. Read

```
/* Fungsi untuk mengambil baris di tabel tblGuest */
Future<GuestItem> getGuest(int id) async {
  List<Map> maps = await database.query(tableGuest,
      columns: getColumnsName(guestColumns),
      where: guestColumns[0].name + ' = ?',
      whereArgs: [id]);
  if (maps.isNotEmpty) {
    return GuestItem.fromMap(maps.first);
  }
  return null;
}

/* Fungsi untuk mengambil daftar baris di tabel tblGuest berdasarkan
eventId */
Future<List<GuestItem>> getListGuest(int eventId) async {
  List<Map> maps = await database.query(tableGuest,
      columns: getColumnsName(guestColumns),
      where: guestColumns[1].name + " = ? ",
      whereArgs: [eventId]);
  List<GuestItem> listGuest = [];
  if (maps.isNotEmpty) {
    for (Map map in maps) {
      GuestItem guest = GuestItem.fromMap(map);
      listGuest.add(guest);
    }
    return listGuest;
  }
  return listGuest;
}
```

3. Update

```
Future<int> updateGuest(GuestItem guest) async {
  return await database.update(
    tableGuest,
    guest.toMap(),
    where: guestColumns[0].name + ' = ?',
    whereArgs: [guest.guestId],
  );
}
```

4. Delete

```
Future<int> deleteGuest(int id) async {
  return await database.delete(
    tableGuest,
    where: guestColumns[0].name + ' = ?',
    whereArgs: [id],
  );
}
```