LAPORAN BASIS DATA MEMBUAT APLIKASI FLEXOFAST, LAYANAN PENYIMPANAN GUDANG

Dosen Pengampu:

Paramitha Nerisafitra, S.ST.,M.kom

Disusun Oleh:

Kelompok 7

Rafly Syauqi Abrori	(23051204370)
M.Farhan Nabil	(23051204372)
Alfian Indra Pratama	(23051204383)
Rayyan	(23051204385)

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS NEGERI SURABAYA

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam dunia bisnis modern, pengelolaan informasi yang terintegrasi menjadi kunci utama dalam mendukung operasi yang efisien. Perusahaan sering menghadapi tantangan dalam mengelola data yang tersebar, seperti informasi klien, distribusi barang, pengelolaan gudang, dan catatan keuangan. Database "Flexofast" dirancang untuk menyediakan solusi terpadu bagi perusahaan dalam mengelola berbagai aspek operasional, mulai dari pengelolaan klien hingga pengawasan transaksi dan persetujuan. Dengan sistem ini, perusahaan dapat meningkatkan efisiensi, mengurangi risiko kesalahan manusia, dan memaksimalkan penggunaan sumber daya.

1.2 Tujuan Proyek

Proyek ini bertujuan untuk:

1. Mengintegrasikan data operasional seperti klien, distributor, gudang, pabrik, dan barang dalam satu sistem database.
2. Mempermudah pelacakan transaksi, sewa, dan pembayaran melalui pengelolaan data yang sistematis.
3. Meningkatkan akurasi dan kecepatan dalam pengambilan keputusan melalui penyajian data yang terstruktur dan mudah diakses.
4. Memberikan alat yang memungkinkan perusahaan untuk memantau aktivitas persetujuan dan penagihan secara real-time.

1.3 Lingkup Proyek

Lingkup proyek meliputi:

1. Manajemen Klien: Mengelola data klien termasuk identitas, kontak, dan alamat mereka.
2. Pengelolaan Distributor dan Pabrik: Mencatat informasi mengenai distributor dan pabrik yang terkait dengan perusahaan.
3. Manajemen Gudang: Memastikan pengelolaan tipe gudang, volume, dan harga sewa yang terintegrasi.
4. Pengelolaan Barang: Melacak jenis barang yang tersedia dan digunakan dalam transaksi.
5. Manajemen Transaksi: Mencakup pencatatan transaksi barang dan volume pengiriman yang dilakukan.

6. Sewa Gudang: Merekam informasi mengenai penyewaan gudang, termasuk periode sewa dan klien terkait.
7. Persetujuan dan Tagihan: Memonitor persetujuan oleh pegawai terkait dan pengelolaan tagihan untuk pembayaran.
8. Pelaporan Keuangan: Memberikan rekam jejak pembayaran yang dilakukan dan memantau batas waktu tagihan.

Proyek ini diharapkan dapat memberikan manfaat jangka panjang dengan menyediakan sistem pengelolaan data yang efektif, efisien, dan dapat diandalkan untuk berbagai operasi perusahaan.

BAB II

ANALISIS PERMASALAHAN

2.1 Identifikasi Permasalahan

Dalam pengelolaan bisnis yang kompleks, perusahaan menghadapi berbagai tantangan yang dapat menghambat efisiensi operasional dan pengambilan keputusan. Berikut adalah beberapa permasalahan utama yang diidentifikasi:

1. Data yang Terfragmentasi: Informasi yang tersebar di berbagai departemen, seperti klien, transaksi, gudang, dan keuangan, sulit untuk diintegrasikan, sehingga memperlambat alur kerja.
2. Kekurangan Sistem Pemantauan Real-Time: Tidak adanya mekanisme untuk memantau transaksi, persetujuan, dan tagihan secara langsung mengakibatkan potensi keterlambatan dalam penyelesaian tugas penting.
3. Risiko Kesalahan Manual: Pengelolaan data secara manual meningkatkan risiko kesalahan pencatatan yang dapat memengaruhi keakuratan laporan dan pengambilan keputusan.
4. Kesulitan dalam Pelacakan Transaksi: Sistem yang ada tidak mendukung pelacakan transaksi secara rinci, sehingga menyulitkan identifikasi volume barang, lokasi pengiriman, dan status persetujuan.
5. Ketidakoptimalan Penggunaan Gudang: Kurangnya pengelolaan yang terstruktur pada kapasitas gudang dan harga sewa dapat menyebabkan inefisiensi dalam penggunaan sumber daya.

2.2 Dampak Permasalahan

Permasalahan ini dapat membawa dampak serius, seperti:

1. Penurunan produktivitas karena waktu yang terbuang untuk mencari dan mengolah data.
2. Ketidakakuratan laporan yang menghambat pengambilan keputusan strategis.
3. Potensi kerugian finansial akibat keterlambatan pembayaran atau salah hitung pada transaksi.
4. Hubungan dengan klien dan mitra bisnis terganggu karena ketidaksesuaian informasi.

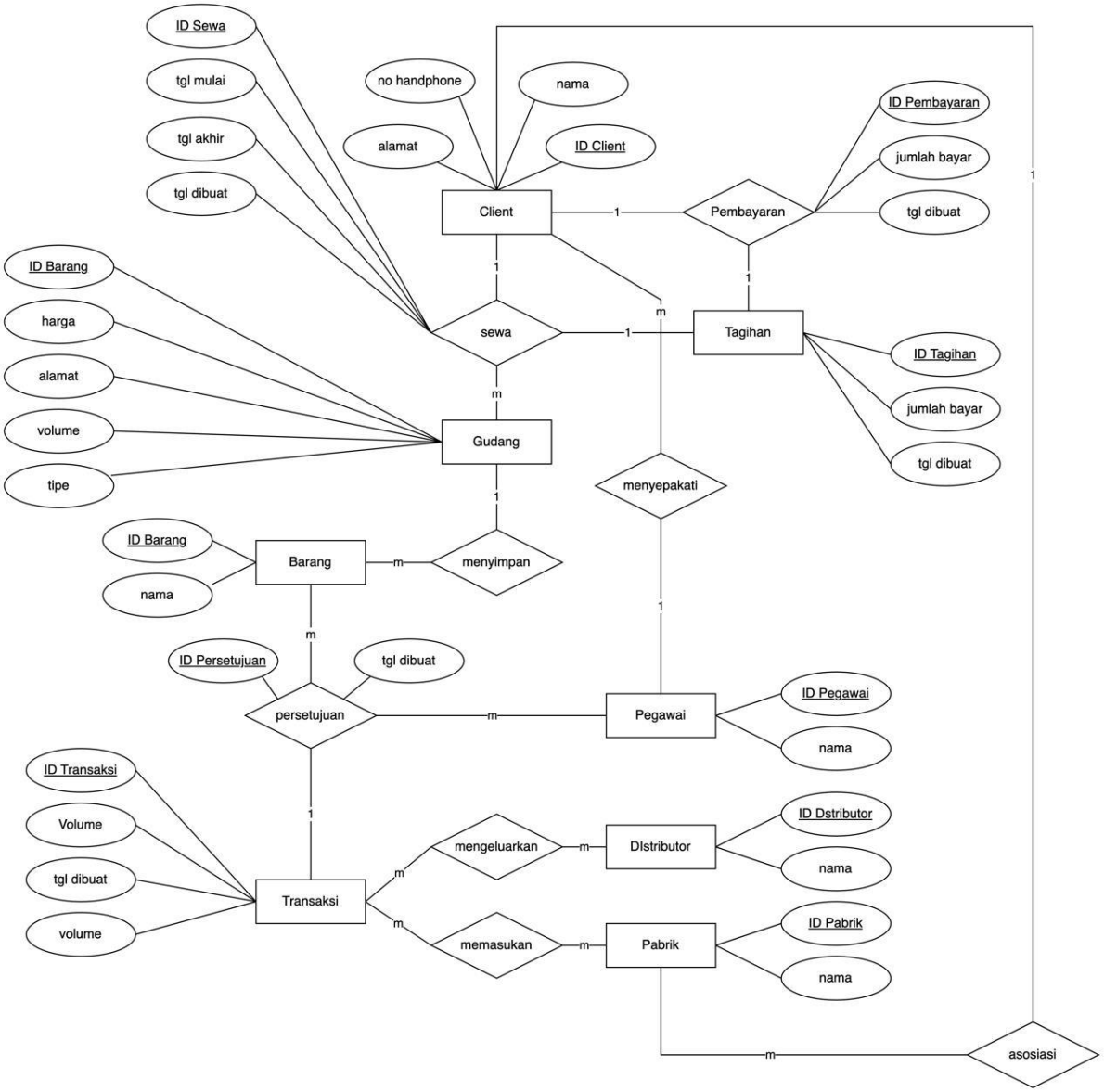
2.3 Penyebab Permasalahan

1. Kurangnya Integrasi Sistem: Sistem yang ada belum dirancang untuk menghubungkan berbagai elemen operasional dalam satu platform.
2. Keterbatasan Teknologi: Sistem manual atau teknologi lama tidak mendukung otomatisasi proses.
3. Minimnya Standar Operasional: Tidak adanya prosedur standar untuk mengelola data menyebabkan ketidakkonsistenan dalam pencatatan.

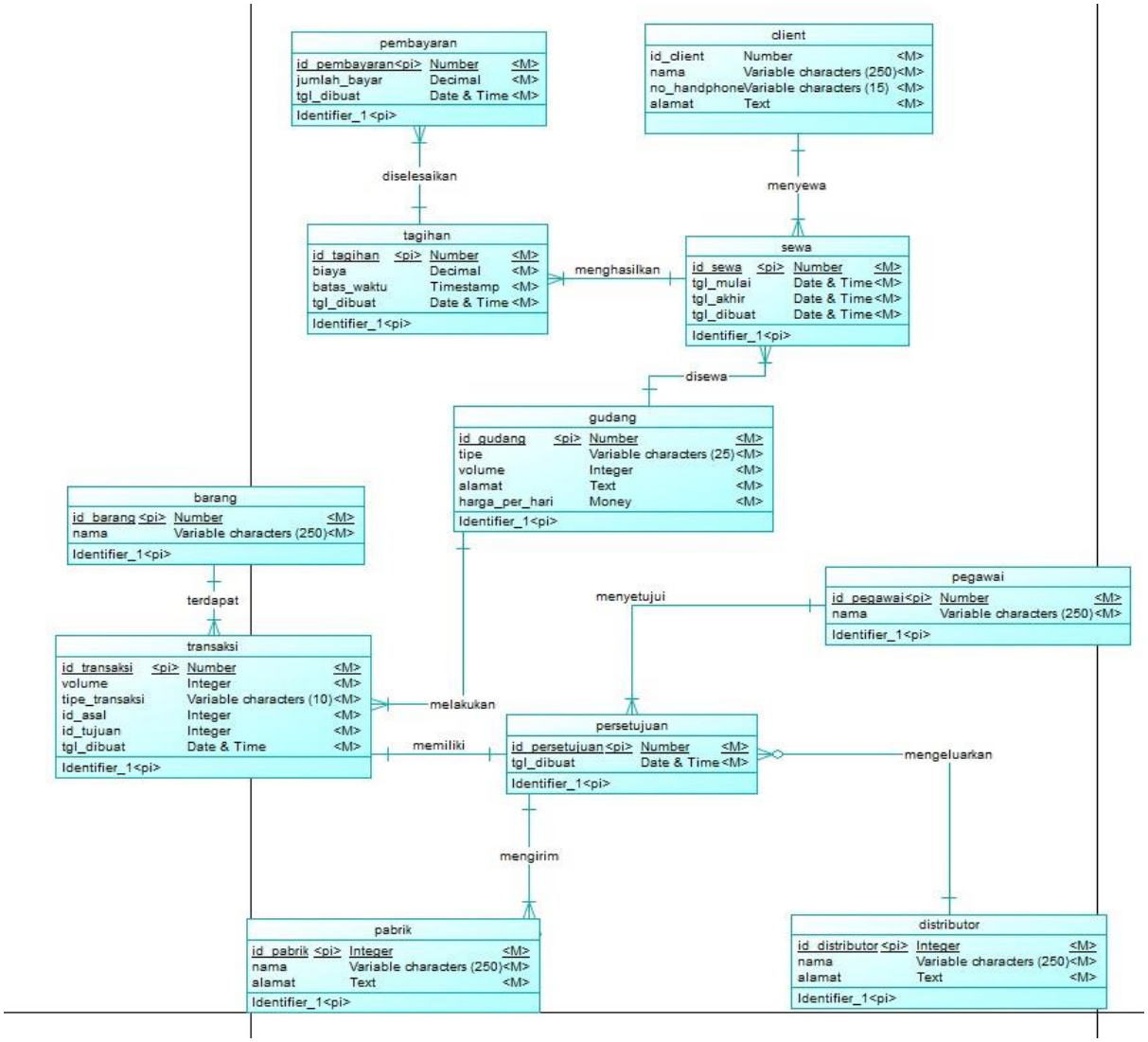
BAB III

PERANCANGAN BASIS DATA

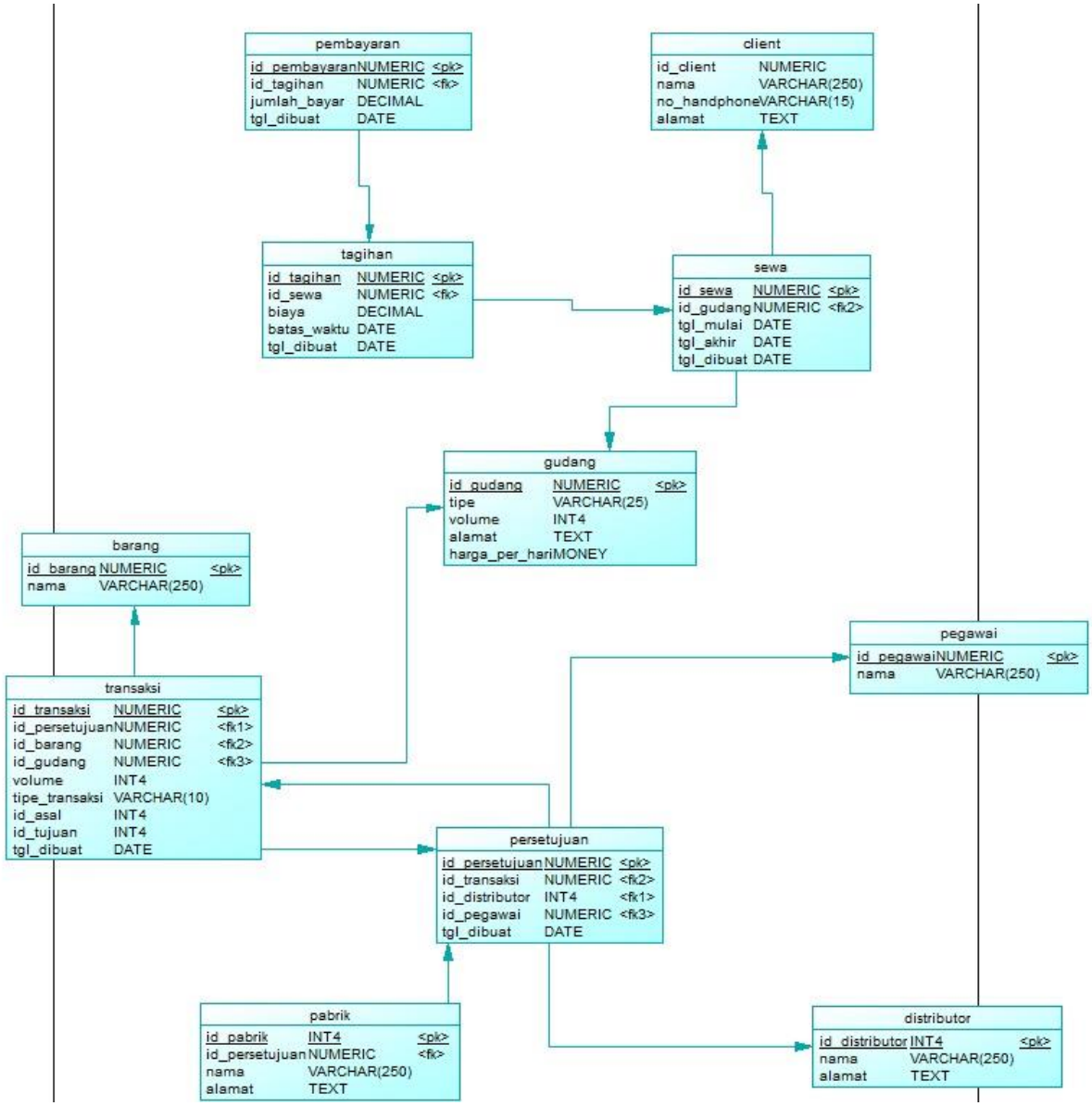
3.1 Desain ERD



3.2 Desain CDM



3.3 Desain PDM



BAB IV

IMPLEMENTASI

4.1 Penggunaan ORM Drift (SQLite)

Pada proyek ini, pengembangan database dibantu menggunakan ORM (Object Relational Mapping) Drift. Drift adalah package Flutter yang mempermudah pengelolaan database SQLite dengan cara yang efisien, aman, dan fleksibel. Drift memberikan antarmuka yang modern sehingga pengelolaan data menjadi lebih intuitif dibandingkan penggunaan SQLite secara langsung.

- **Keunggulan Drift**
    - **Tipe Data yang Aman:**

Drift secara otomatis memetakan tipe data di SQLite ke tipe data di Dart, sehingga mengurangi risiko kesalahan tipe data.

    - **Query Builder:**

Drift menyediakan cara penulisan query SQL yang lebih terstruktur, baik dalam bentuk generated Drift SQL maupun raw SQL yang bisa digunakan secara langsung.

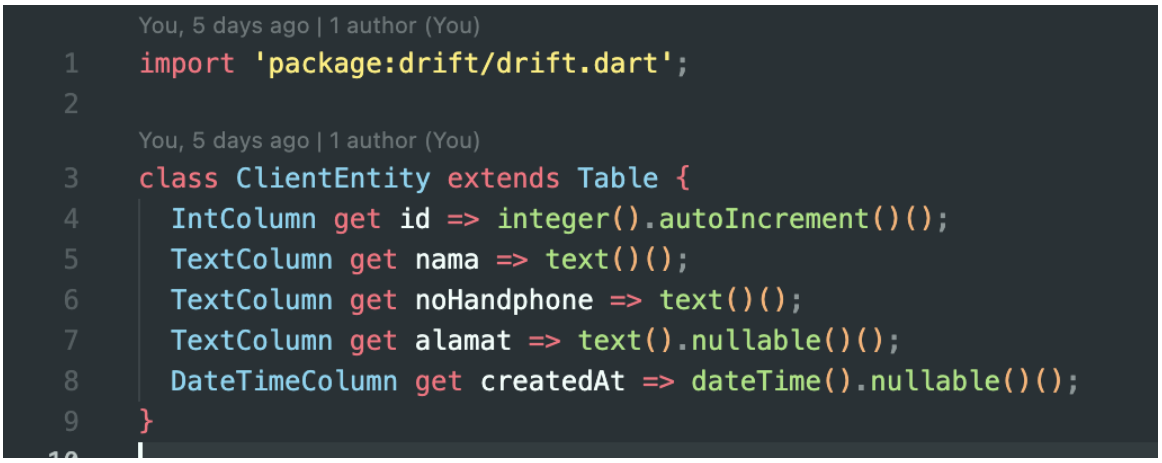
    - **Relasi Antar Tabel:**

Drift mendukung relasi antar tabel, sehingga memudahkan untuk mengatur data yang memiliki hubungan seperti tabel transaksi yang terhubung ke tabel pelanggan dan tabel produk.

- **Implementasi Pada Proyek**

Dalam proyek ini, Drift digunakan untuk:

- **Membuat Databse dan Tabel:**

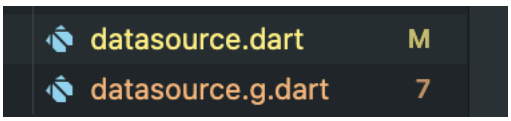


Drift menggunakan pendekatan berbasis model untuk mendefinisikan tabel. Setiap tabel didefinisikan sebagai class yang mewarisi Table, di mana Anda mendeklarasikan kolom, tipe datanya, serta constraint seperti primary key, foreign key, atau panjang teks.

Drift akan digunakan untuk membuat tabel seperti Barang,Gudang, transaksi, Client dan lain lainya dengan definisi yang jelas dan simpel, Drift mempermudah pembuatan tabel dengan menyediakan beberapa tool seperti Struktur tabel yang dapat didefinisikan dengan deklarasi yang singkat namun tetap eksplisit, memastikan tiap kolom memiliki tipe data dan aturan validasi yang sesuai sehingga menugrangi resiko kesalahan saat pengembangan.

```dart
@DriftDatabase(
  tables: [
    ClientEntity,
    DistributorEntity,
    GudangEntity,
    PabrikEntity,
    PegawaiEntity,
    PembayaranEntity,
    PersetujuanEntity,
    SewaEntity,
    TagihanEntity,
    TransaksiEntity,
    BarangEntity
  ],
  views: [
    DetailSewaView,
  ],
)
You, 3 hours ago | 1 author (You)
class Datasource extends _$Datasource {
  Datasource() : super(_openConnectionStatic());

  static Datasource instance = Datasource();

  @override
  int get schemaVersion => 1;

  static LazyDatabase _openConnectionStatic() {
    return LazyDatabase(() async {
      final dbFolder = await getApplicationDocumentsDirectory();
      final file = File(p.join(dbFolder.path, 'flexofast.db'));

      if (Platform.isAndroid) {
        await applyWorkaroundToOpenSqlite3OnOldAndroidVersions();
      }
      final cachebase = (await getTemporaryDirectory()).path;
      sqlite3.tempDirectory = cachebase;
      return NativeDatabase.createInBackground(file);      You, 3 hours ago • add ba
    }); // LazyDatabase
  }
}
```

Anotasi @DriftDatabase digunakan untuk menggabungkan tabel-tabel yang sudah didefinisikan ke dalam sebuah class database. Class ini bertindak sebagai interface utama untuk berinteraksi dengan database, seperti melakukan operasi CRUD, query kustom, atau migrasi.



Setelah mendefinisikan tabel dan database, Drift menggunakan build runner untuk menghasilkan file otomatis (biasanya dengan ekstensi .g.dart). File ini memuat kode API yang mempermudah akses ke tabel dan operasi database.

Untuk menghasilkan file otomatis, gunakan perintah berikut:

```
flutter pub run build_runner build
```

Build runner membaca definisi tabel dan database, lalu menghasilkan kode yang diperlukan untuk operasi database.

- **Penggunaan DAO(Data Access Object) dan Operasi CRUD:**

```dart
import 'package:drift/drift.dart';
import 'package:flexofast_basis_data_dashboard/database/datasource.dart';
import 'package:flexofast_basis_data_dashboard/entity/barang_entity.dart';
part 'barang_dao.g.dart';

@DriftAccessor(tables: [BarangEntity])
You, 4 days ago | 1 author (You)
class BarangDao extends DatabaseAccessor<Datasource> with _$BarangDaoMixin {
  BarangDao(super.db);

  static final BarangDao instance = BarangDao(Datasource.instance);

  Future<List<BarangEntityData>> getAllBarang() async {
    final result = await select(db.barangEntity).get();
    return result;
  }

  Future<BarangEntityData> getBarangById(int id) async {
    final result = await (select(db.barangEntity)
          ..where((tbl) => tbl.id.equals(id)))
        .getSingle();
    return result;
  }

  Future<int> insertBarang(BarangEntityCompanion barang) async {
    final result = await into(db.barangEntity).insert(barang);
    return result;
  }

  Future updateBarang(BarangEntityData barang) async {
    await update(db.barangEntity).replace(barang);
  }

  Future deleteBarang(int id) async {
    await (delete(db.barangEntity)..where((tbl) => tbl.id.equals(id))).go();
  }
}
```

DAO adalah pola desain yang digunakan untuk memisahkan logika akses data dari logika bisnis dalam aplikasi. Dalam konteks Drift, DAO berfungsi untuk mengelola operasi database, seperti **insert**, **update**, dan **delete**, pada tabel tertentu.

Sebagai contoh, pada **class BarangDao**, anotasi @DriftAccessor menentukan tabel apa yang akan dioperasikan. Class ini juga mengambil **instance** langsung dari class database utama, sehingga dapat berinteraksi langsung dengan file database yang telah terbuka.

Untuk operasi-operasi yang dapat dilakukan, kita dapat menjalankan berbagai operasi **CRUD** (Create, Read, Update, Delete) ataupun query lainnya pada tabel yang dipilih. Drift sudah menyediakan alat bantu untuk operasi ini, seperti:

- select untuk membaca data,
- insert untuk menambahkan data baru,
- update untuk memperbarui data yang ada,
- delete untuk menghapus data tertentu.
- where untuk filterisasi data yang diambil
- Joins untuk menggabungkan beberapa tabel dalam query
- transaction untuk melakukan transaksi penulisan data antar beberapa tabel
- Dll.

Dengan pendekatan ini, penggunaan DAO dalam Drift mempermudah pengelolaan database sekaligus menjaga modularitas dan kejelasan struktur kode dalam aplikasi.

- **Relasi Antar Tabel:**

```
class PersetujuanEntity extends Table {
  IntColumn get id => integer().autoIncrement()();
  IntColumn get idPegawai =>
      integer().nullable().references(PegawaiEntity, #id)();
  IntColumn get idTransaksi => integer().references(TransaksiEntity, #id)();
  DateTimeColumn get createdAt => dateTime().nullable()();
}
```

Drift mendukung relasi antar tabel melalui penggunaan foreign key, memungkinkan tabel saling terhubung untuk menjaga integritas data dan mempermudah pengambilan data terstruktur. Relasi ini dibuat dengan mendefinisikan kolom dalam tabel yang merujuk pada primar key pada tabel lain menggunakan metode .references().

- **Tabel Utama (Parent Table)**

Tabel utama adalah tabel yang akan dirujuk oleh tabel lain dan harus memiliki primary key unik. Drift secara otomatis menghasilkan primary key jika kolom didefinisikan dengan .autoIncrement().

- **Tabel Relasi (Child Table)**

Tabel relasi adalah tabel yang memiliki foreign key yang didefinisikan menggunakan .references(TabelUtama, #kolomUtama). Kolom ini menyimpan nilai yang menunjuk ke primary key di tabel utama.

Dengan adanya relasi ini, query dapat dilakukan menggunakan **join** untuk menggabungkan data dari beberapa tabel sekaligus. Relasi ini membantu mengorganisir data ke dalam tabel-tabel dengan fungsi spesifik, sehingga lebih mudah dikelola. Drift mempermudah pengelolaan relasi antar tabel dengan menyediakan fitur bawaan untuk **foreign key**, mendukung integritas data tanpa perlu menulis logika relasi secara manual. Hasilnya, struktur data dalam database menjadi lebih terorganisir, efisien, dan mudah dikelola.

4.2 Desain UI Project atau Form Pengisian Data

Proyek ini bertujuan untuk menghasilkan sebuah aplikasi yang dapat membantu pengguna dalam penginputan data ke dalam database. Database tersebut akan digunakan sebagai datasource lokal yang mendukung aplikasi untuk menyimpan dan mengelola data secara efisien. Selain berfungsi sebagai form input data, aplikasi ini juga dirancang untuk mempermudah pengguna dalam memproses dan mengakses data yang tersimpan.

- ## Teknologi yang Digunakan

1. Flutter Flamework:

Aplikasi akan menggunakan Flutter Framework sebagai widget builder utama untuk membangun antarmuka pengguna yang interaktif, responsif, dan modern.

2. **State Management**:

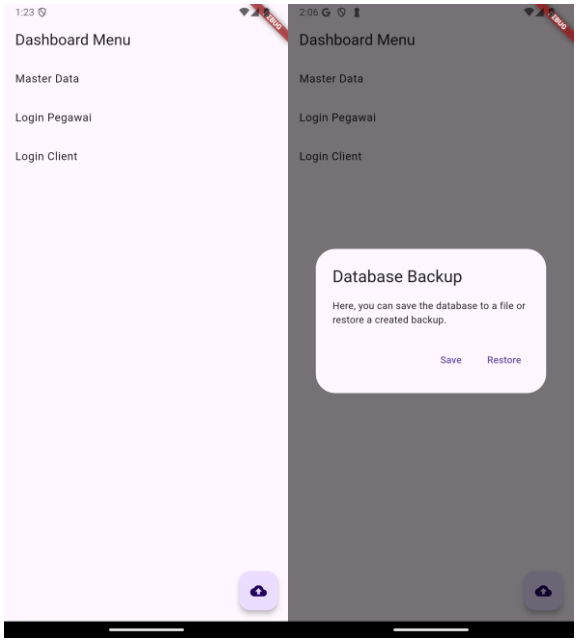
Manajemen state akan menggunakan Bloc (Business Logic Component) untuk mengatur alur data secara terorganisir dan mendukung pemisahan logika bisnis dari antarmuka pengguna.

3. **Backend Lokal**:

Database lokal akan dikelola menggunakan Drift untuk menyediakan datasource lokal yang mendukung relasi antar tabel, menjaga integritas data, dan mempermudah pengambilan data secara terstruktur.

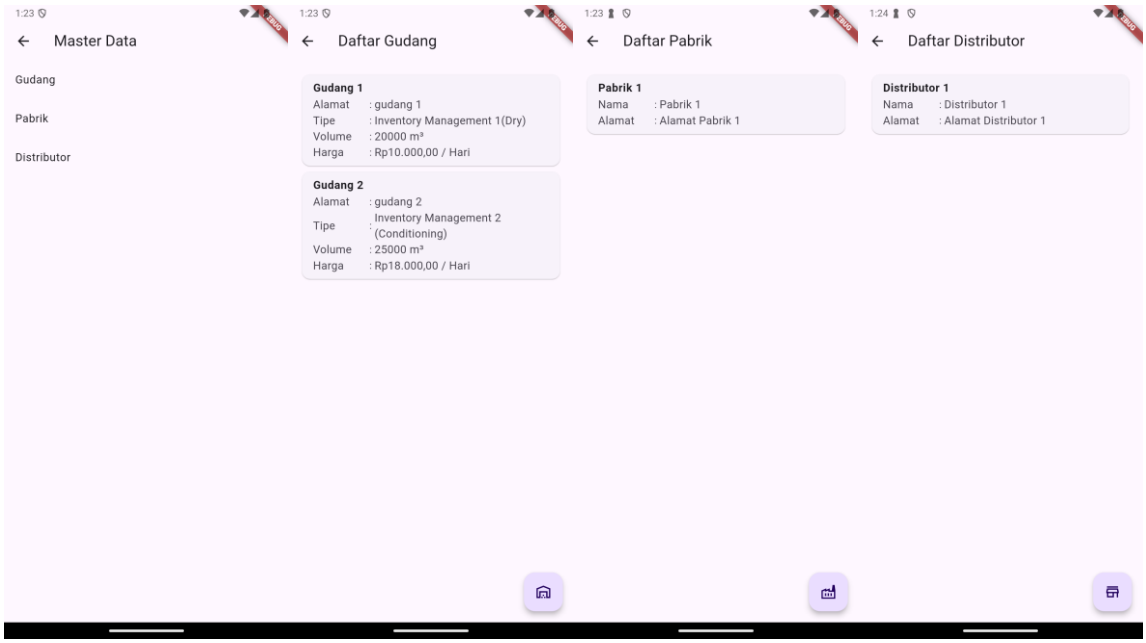
- ## Tampilan Utama Dan Alur Aplikasi

1.Dashboard Menu (Utama)

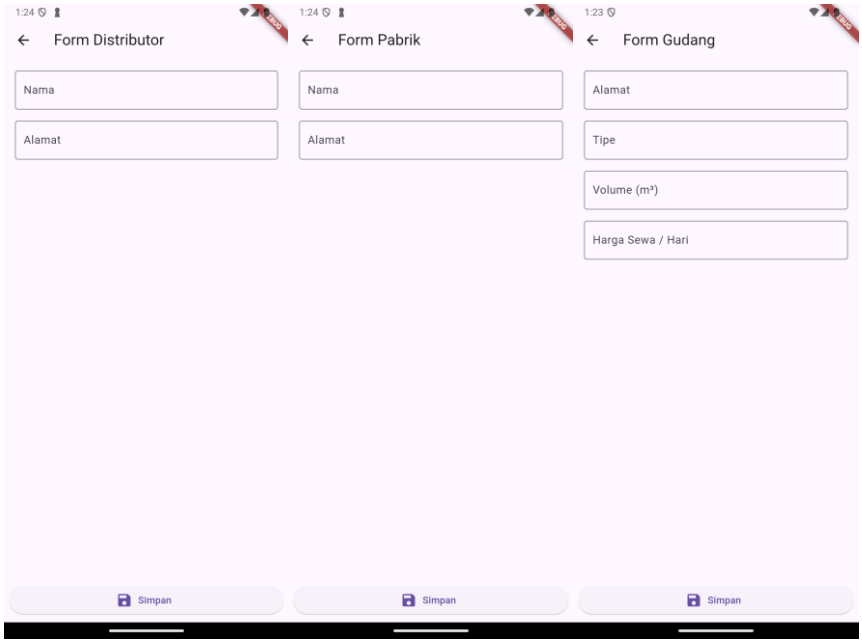


Pada user masuk Pertama kali user akan masuk ke halaman Dashboard dimana user akan melihat beberapa opsi seperti Master Data, Login Pegawai , dan Login Client , serta terdapat action button untuk memunculkan opsi Export databse atau tidak.

2.Master Data

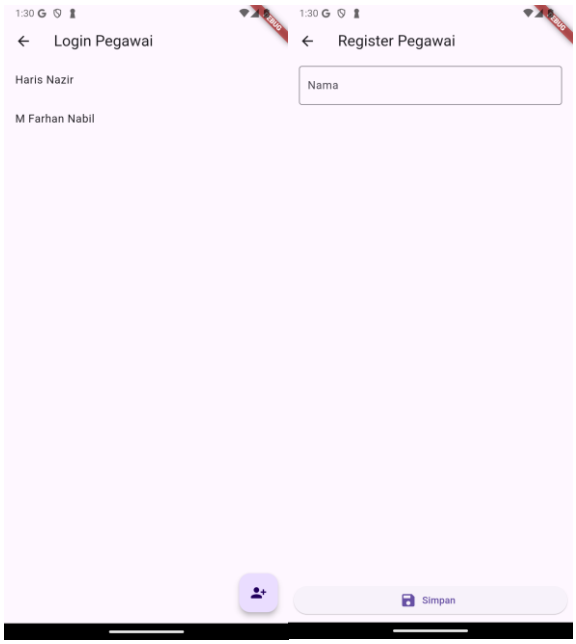


Berisi beberapa opsi form pengisian data untuk pengisian tabel gudang distributor dan pabrik,

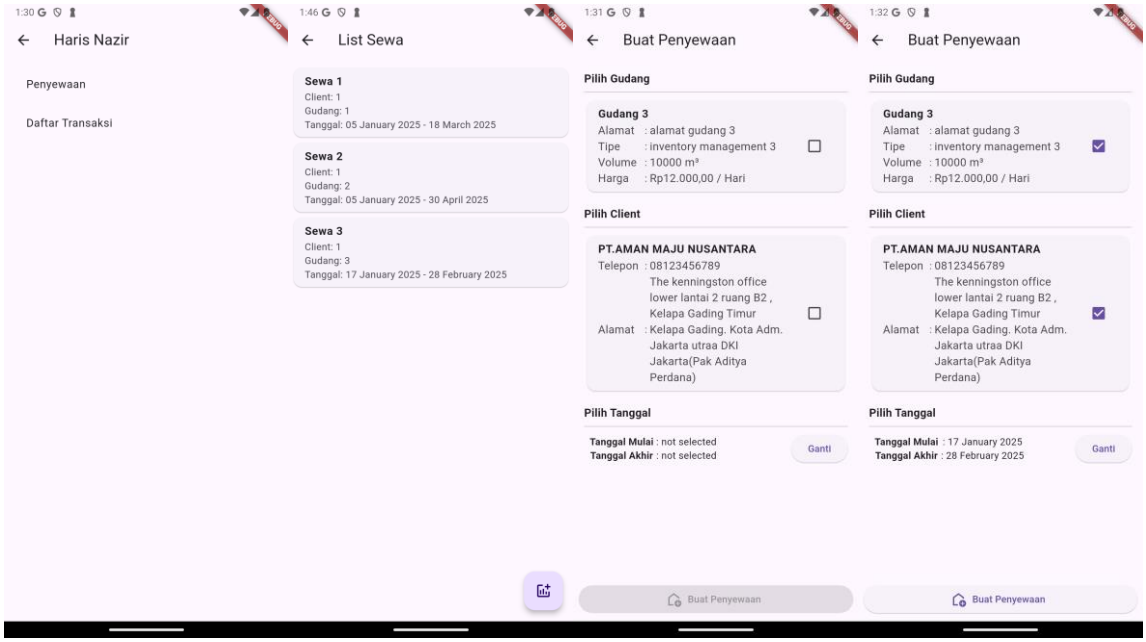


Untuk Mengedit salah satu item user dapat menekan salah satu item untuk membuka form yang sama untuk mengedit item tersebut.

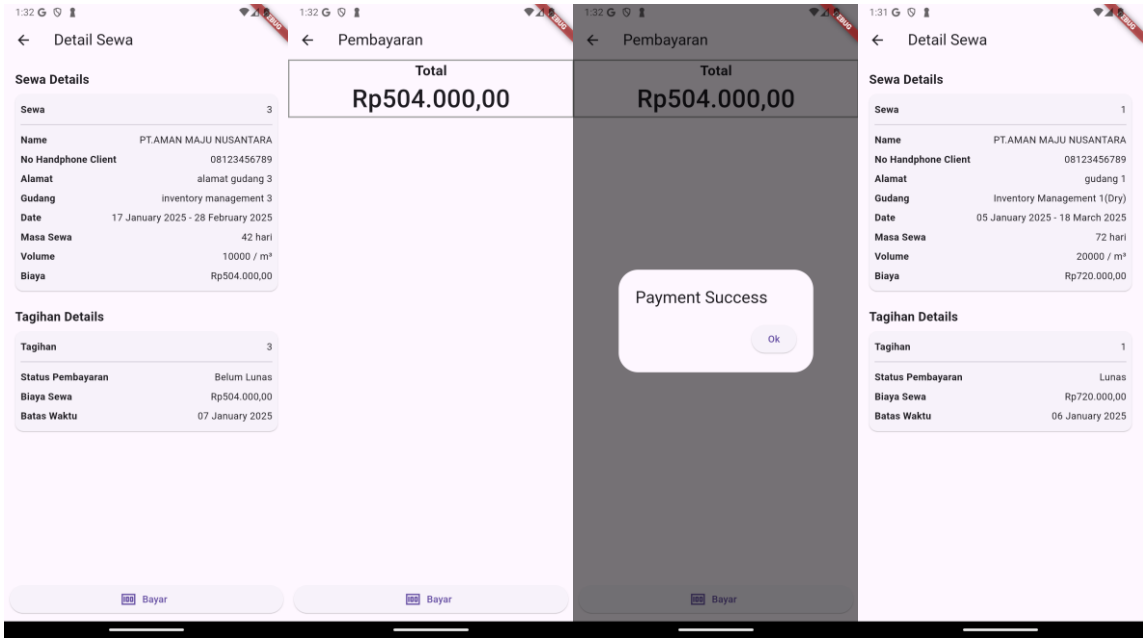
3.Login Pegawai



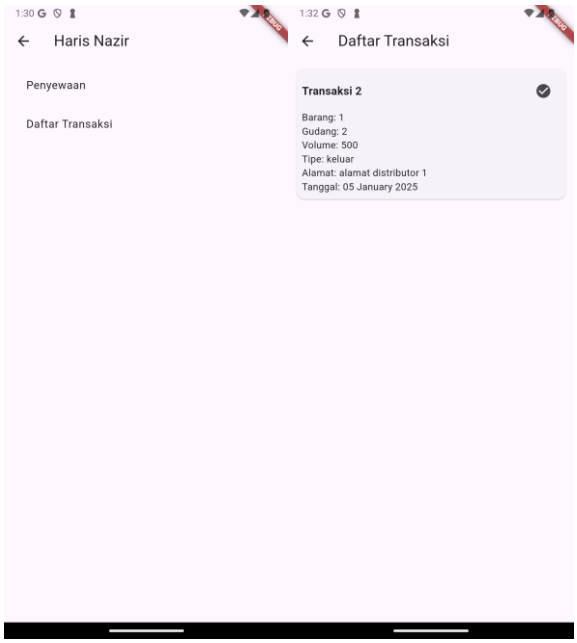
Pada laman login Pegawai user dapat melihat list pegawai , dan terdapat action button di bagian bawah yang dapat menambah kan Pegawai melalui form Register Pegawai , jika user menekan salah satu item dari list pegawai maka akan masuk ke halaman dashboard pegawai tersebut.



Pada laman dashboard pegawai user memiliki beberapa fungsi seperti melakukan penyewaan dimana user dapat memilih gudang yang sudah di daftarkan pada master data . Kemudian dapat memilih Client yang ingin menyewa gudang , serta tanggal awal dan akhir yang dapat di ganti melalui tombol ganti . List sewa akan bertambah jika form terisi dengan benar.

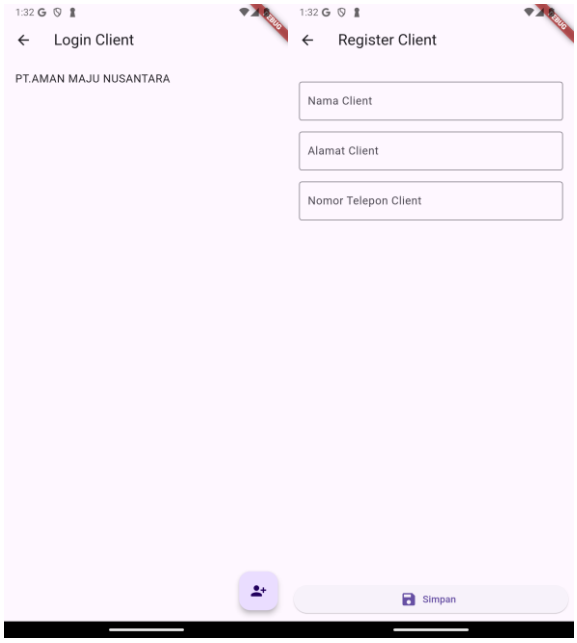


Pada Laman sebelumnya user dapat menekan salah satu item sewa dan akan masuk ke detail sewa di mana dapat dilakukan pembayaran terhadap sewa tersebut ,status sewa akan lunas jika telah terjadi pembayaran.

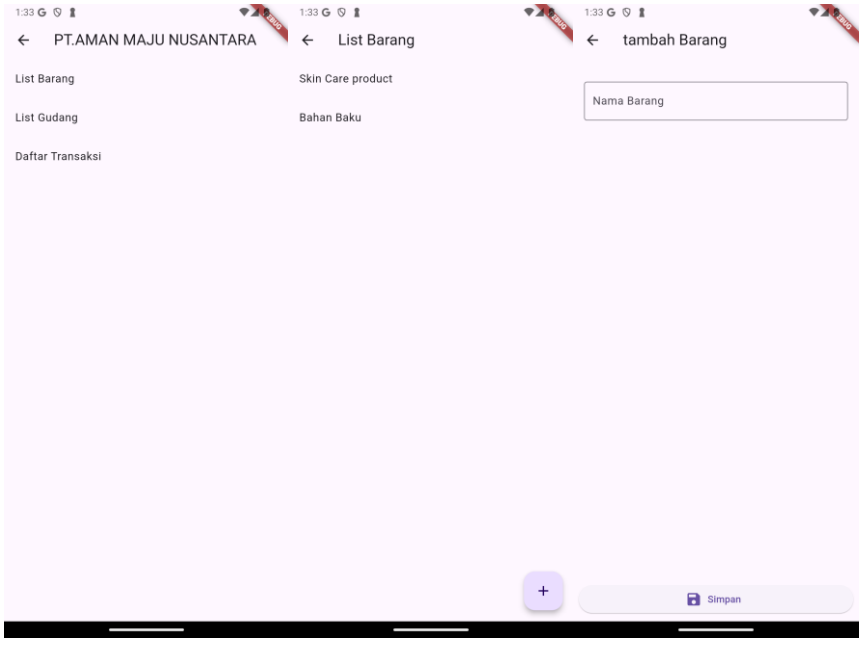


Pada Halaman ini user dapat melihat beberapa transaksi yang akRan terjadi pada gudang yang dilakukan oleh client. Terdapat list transaksi gudang yang di tuju , volume barang , tipe transaksi antara masuk atau keluar gudang , Dll. Pegawai dapat menerima transaksi tersebut dengan tombol di pojok kanan atas item.

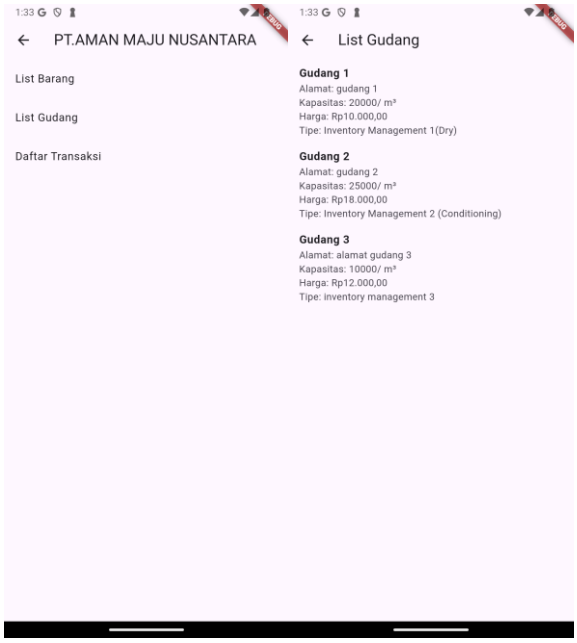
4.Login Client



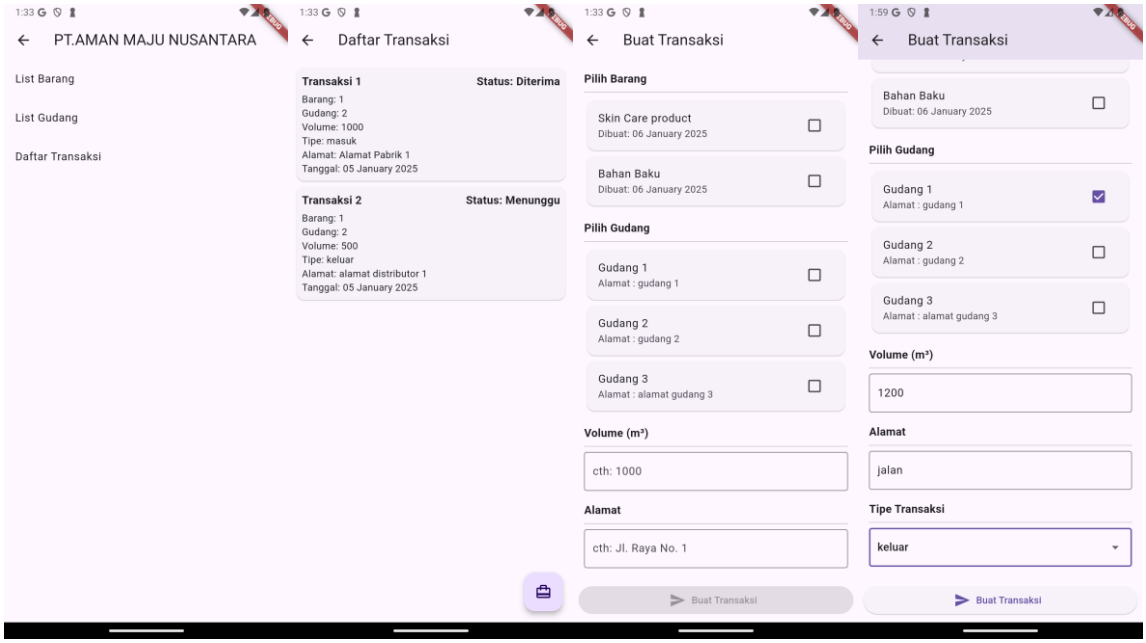
Pada laman login Client user dapat melihat list User , dan terdapat action button di bagian bawah yang dapat menambah kan Pegawai melalui form Register Client , jika user menekan salah satu item dari list pegawai maka akan masuk ke halaman dashboard pegawai tersebut.



Pada Laman Dashboard Client user dapat melihat beberapa fungsi yang dapat dilakukan client .pada List Barang client dapat melihat Semua Barang dan user dapat menambahkan Barang melalui tombol action di pojok kiri bawah untuk masuk ke form tambah Barang.



Pada laman list gudang user dapat melihat gudang yang sudah di sewa.



Pada laman daftar Transaksi user dapat melihat semua transaksi yang pernah dilakukan , berserta status dari transaksi tersebut , user juga dapat menambahkan transaksi barang baru pada tombol action di bawah pojok kiri, dan user akan masuk ke dalam form buat transaksi dan transaksi akan bertambah ke dalam list dengan status menunggu persetujuan pegawai.

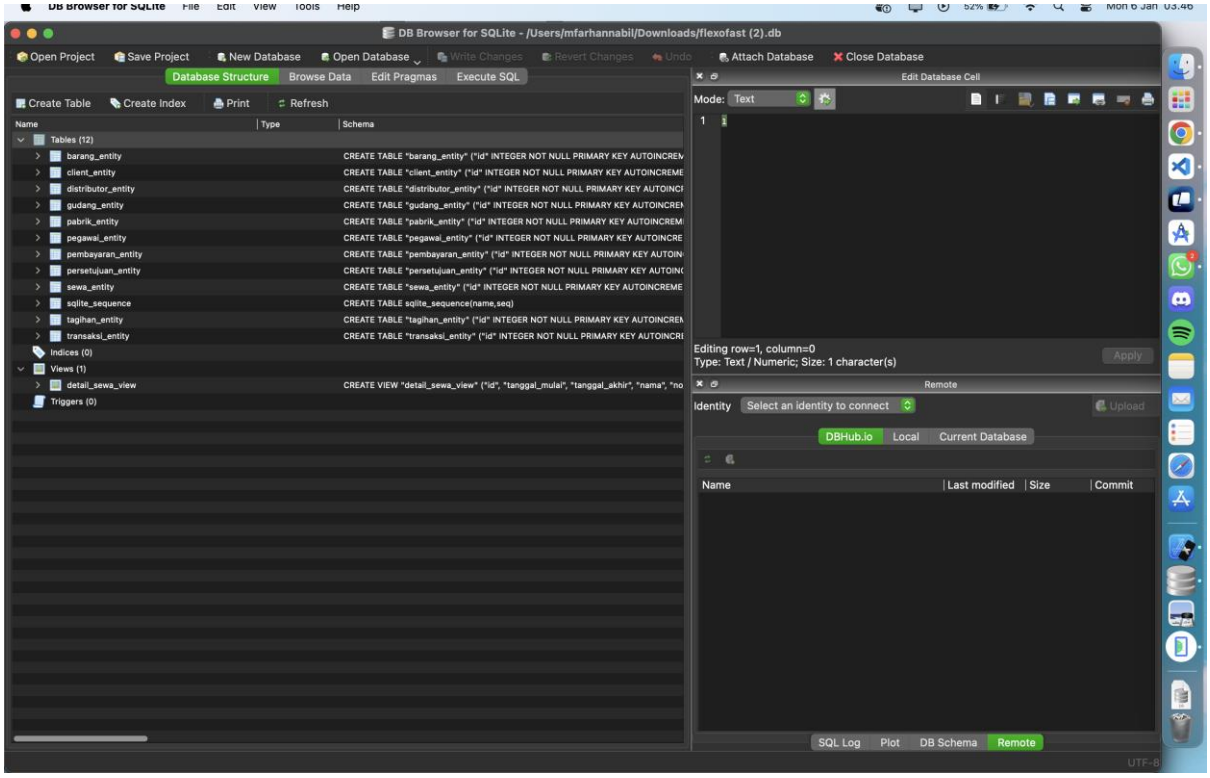
BAB V

HASIL

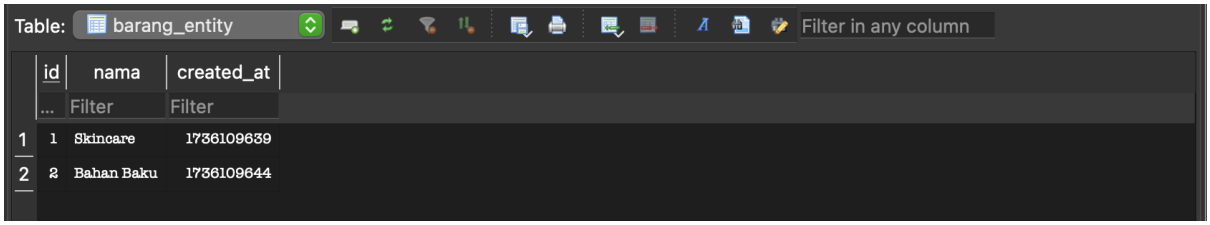
5.1 Tampilan Tabel

Untuk melihat tabel dari file .db yang sudah di ekspor dapat di buka melalui aplikasi DB browser for SQLite (Windows ,MacOS ,dan Linux) dan SQLite Viewer(PlayStore).

Database Structure:



Tabel barang_entity :



Tabel client_entity :

client_entity

Table: client_entity — Filter in any column

	id	nama	no_handphone	alamat	created_at
	...	Filter	Filter	Filter	Filter
1	1	PT AMAN MAJU NUSANTARA	08123456789	The Kensington office tower lantai 2 range B2 , Kelapa gading Timur , Kelapa Gading. ...	1736109496

Tabel distributor_entity :

distributor_entity

Table: distributor_entity — Filter in any column

	id	nama	alamat	created_at
	...	Filter	Filter	Filter
1	1	Distributor 2	Alamat Distributor 2	1736109355
2	2	Distributor 1	Alamat Distributor 1	1736109382

Tabel gudang_entity :

gudang_entity

Table: gudang_entity — Filter in any column

	id	tipe	volume	alamat	harga	created_at
	...	Filter	Filter	Filter	Filter	Filter
1	1	Inventory Management 1 (Dry)	20000	alamat gudang 1	10000	1736109226
2	2	Inventory Management 2 (Conditioning)	25000	Alamat gudang 2	18000	1736109267
3	3	Inventory Management 3 (normal)	30000	alamat gudang 3	15000	1736109316

Tabel pabrik_entity :

pabrik_entity

Table: pabrik_entity — Filter in any column

	id	nama	alamat	created_at
	...	Filter	Filter	Filter
1	1	pabrik 1	alamat pabrik 1	1736109327
2	2	pabrik 2	alamat pabrik 2	1736109336

Tabel pegawai_entity

pegawai_entity

Table: pegawai_entity — Filter in any column

	id	nama	created_at
	...	Filter	Filter
1	1	Haris Nazir	1736109533
2	2	M Farhan Nabil	1736109544

Tabel pembayaran_entity :

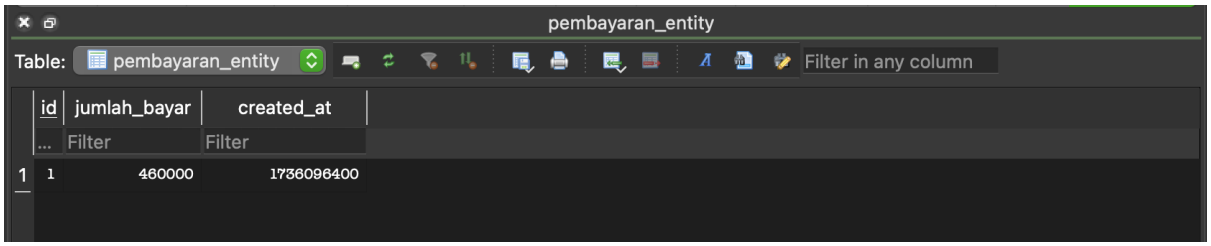

pembayaran_entity

Table: pembayaran_entity

Filter in any column

	id	jumlah_bayar	created_at
	...	Filter	Filter
1	1	460000	1736096400

Tabel persetujuan_entity :

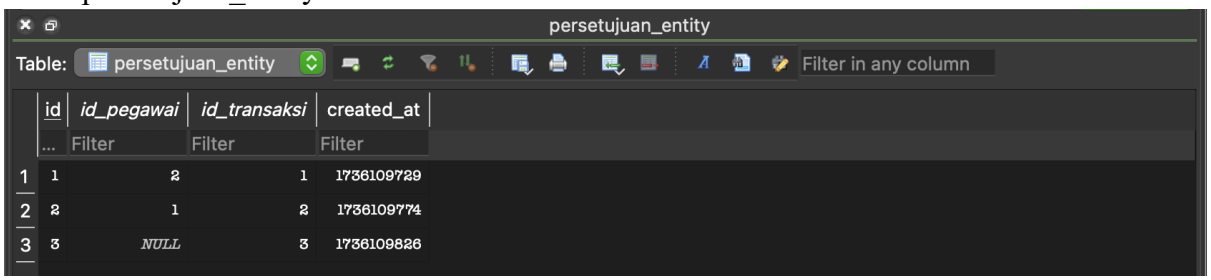

persetujuan_entity

Table: persetujuan_entity

Filter in any column

	id	id_pegawai	id_transaksi	created_at
	...	Filter	Filter	Filter
1	1	2	1	1736109729
2	2	1	2	1736109774
3	3	NULL	3	1736109826

Tabel sewa_entity :

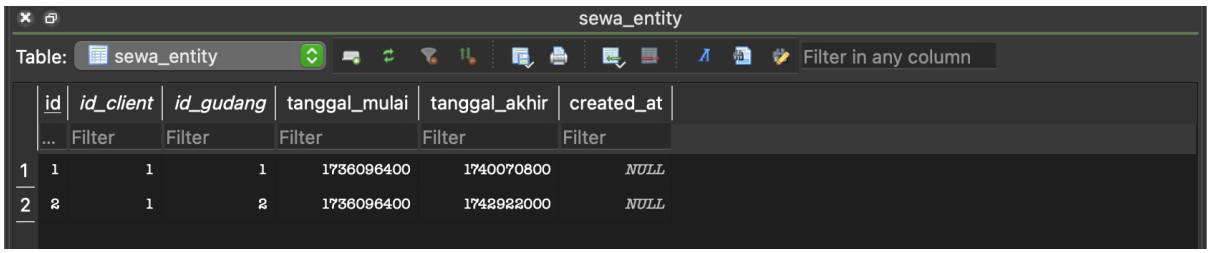

sewa_entity

Table: sewa_entity

Filter in any column

	id	id_client	id_gudang	tanggal_mulai	tanggal_akhir	created_at
	...	Filter	Filter	Filter	Filter	Filter
1	1	1	1	1736096400	1740070800	NULL
2	2	1	2	1736096400	1742922000	NULL

Tabel tagihan_entity :

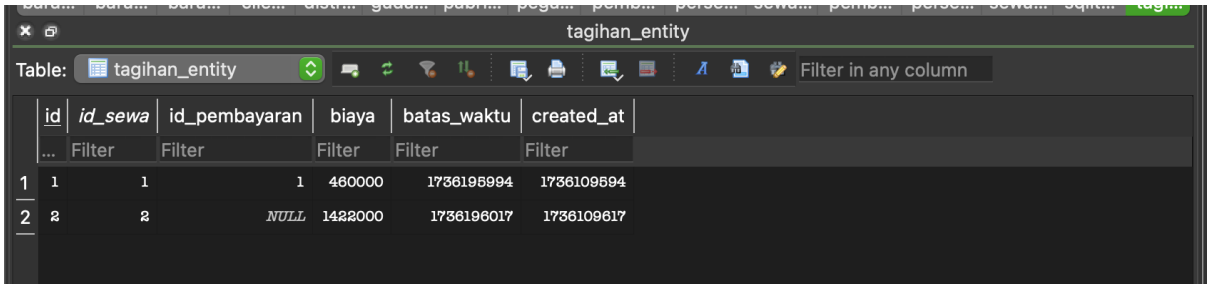

tagihan_entity

Table: tagihan_entity

Filter in any column

	id	id_sewa	id_pembayaran	biaya	batas_waktu	created_at
	...	Filter	Filter	Filter	Filter	Filter
1	1	1	1	460000	1736195994	1736109594
2	2	2	NULL	1422000	1736196017	1736109617

Tabel transaksi_entity :

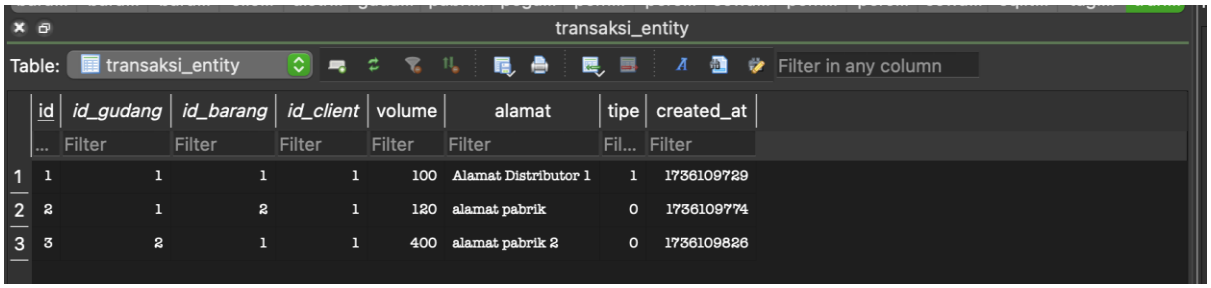
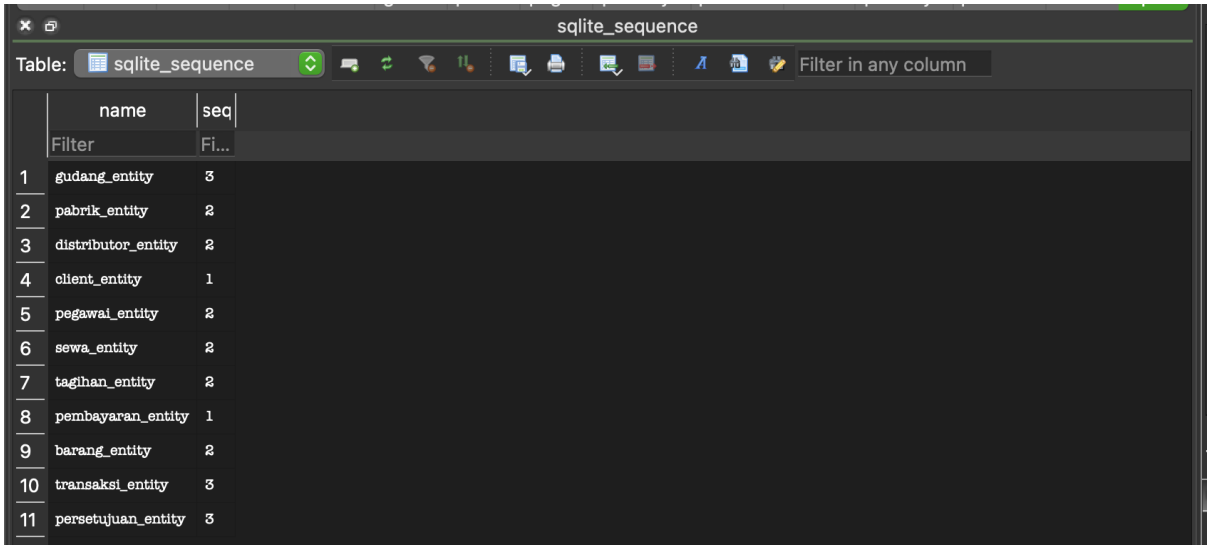

transaksi_entity

Table: transaksi_entity

Filter in any column

	id	id_gudang	id_barang	id_client	volume	alamat	tipe	created_at
	...	Filter	Filter	Filter	Filter	Filter	Fil...	Filter
1	1	1	1	1	100	Alamat Distributor 1	1	1736109729
2	2	1	2	1	120	alamat pabrik	0	1736109774
3	3	2	1	1	400	alamat pabrik 2	0	1736109826

Tabel sql_sequence :



5.2 Repository Projek

Adapun Untuk versi lengkap project tersedia pada Link Repository Berikut :
https://github.com/nabilban/flexofast_basis_data_dashboard.git

BAB VI

KESIMPULAN DAN SARAN

6.1 Kesimpulan

Sistem Flexofast yang dikembangkan sebagai solusi pengelolaan data berbasis database modern telah berhasil menciptakan integrasi operasional yang sangat dibutuhkan oleh perusahaan dalam menghadapi tantangan bisnis yang kompleks. Dengan menggunakan teknologi ORM Drift berbasis SQLite, sistem ini memberikan kemudahan dalam pembuatan, pengelolaan, dan pengoperasian database, termasuk fitur-fitur seperti pembuatan tabel yang

terstruktur, relasi antar tabel yang fleksibel, serta operasi CRUD yang efisien dan aman. Sistem ini mampu mengatasi berbagai permasalahan utama seperti data yang terfragmentasi, ketidakmampuan memantau aktivitas secara real-time, serta risiko kesalahan manual dalam pengelolaan data, yang sebelumnya menjadi hambatan signifikan dalam operasional perusahaan. Selain itu, keunggulan sistem ini terletak pada kemampuannya untuk meningkatkan efisiensi pelacakan transaksi, optimalisasi sumber daya gudang, dan penyajian data yang terstruktur untuk mendukung pengambilan keputusan yang lebih cepat dan akurat. Dengan penerapan sistem yang terintegrasi ini, perusahaan dapat mengoptimalkan proses bisnisnya, meminimalkan potensi kerugian akibat keterlambatan atau kesalahan data, serta menciptakan alur kerja yang lebih efektif untuk memenuhi kebutuhan operasional dan strategis jangka panjang.

6.2 Saran

1. Pengembangan Lebih Lanjut
   Disarankan untuk menambahkan fitur pelaporan yang lebih komprehensif dan berbasis analitik untuk mendukung pengambilan keputusan strategis perusahaan.
2. Pengujian Skalabilitas
   Lakukan pengujian lebih lanjut terhadap skalabilitas sistem untuk memastikan sistem dapat mendukung pertumbuhan volume data seiring perkembangan perusahaan.
3. Integrasi dengan Sistem Eksternal
   Perlu dikaji kemungkinan integrasi dengan sistem lain seperti ERP (Enterprise Resource Planning) atau CRM (Customer Relationship Management) untuk meningkatkan sinergi antar fungsi perusahaan.
4. Pelatihan Pengguna
   Untuk memastikan sistem digunakan secara maksimal, perusahaan harus memberikan pelatihan kepada pengguna, khususnya dalam hal pengoperasian fitur-fitur baru.
5. Pengelolaan Data Berkelanjutan
   Disarankan untuk menetapkan kebijakan pengelolaan data yang berkelanjutan, termasuk backup rutin, audit data, dan pembaruan sistem secara berkala.