**2. Pemilihan Arsitektur Aplikasi**

**2.1. Tujuan Pemilihan Arsitektur**

* Menyediakan fondasi yang stabil untuk pengembangan, pemeliharaan, dan skalabilitas aplikasi Platform Toko Sembako.
* Memastikan kebutuhan fungsional (pengelolaan stok, pemesanan, dan pembayaran) dan non-fungsional (keamanan, performa, dan ketersediaan) dapat terpenuhi dengan baik.

**2.2. Pilihan Arsitektur**

**2.2.1. Model-View-Controller (MVC)**

* **Deskripsi**: Arsitektur MVC membagi aplikasi menjadi tiga komponen utama: Model: Mengelola data dan logika bisnis seperti manajemen produk, pengguna, dan pesanan, View: Menampilkan data dalam bentuk antarmuka yang mudah digunakan oleh pelanggan dan admin toko, Controller: Mengelola input pengguna dan mengarahkan ke model atau view yang sesuai.
* **Keuntungan**: Pemisahan yang jelas antara logika bisnis, antarmuka pengguna, dan kontrol aplikasi. Memudahkan pengembangan fitur baru seperti diskon atau promosi tanpa mengganggu bagian lain.
* **Kekurangan**: Kompleksitas dapat meningkat jika tidak diatur dengan baik, terutama untuk aplikasi yang terus berkembang.

**2.2.2. Microservices**

* **Deskripsi**: Aplikasi dibagi menjadi layanan kecil dan independen yang mengelola fungsi spesifik seperti manajemen pengguna, katalog produk, dan sistem pembayaran. Komunikasi antar layanan dilakukan melalui API.
* **Keuntungan**: Memungkinkan pengembangan dan penyebaran setiap layanan secara mandiri. Skalabilitas tinggi untuk menangani beban tertentu, misalnya, layanan pembayaran pada waktu promo besar-besaran.
* **Kekurangan**: Kompleksitas manajemen layanan, termasuk orkestrasi dan pemantauan antar layanan. Memerlukan infrastruktur yang lebih canggih untuk mengelola layanan terdistribusi.

**2.2.3. Single Page Application (SPA)**

* **Deskripsi**: Aplikasi web ini memuat satu halaman HTML dengan konten yang diperbarui secara dinamis melalui JavaScript untuk pengalaman pengguna yang lebih interaktif.
* **Keuntungan**: Antarmuka pengguna yang cepat dan interaktif, cocok untuk belanja online dengan katalog produk yang terus berubah. Mengurangi waktu muat halaman dan memberikan pengalaman belanja yang mulus.
* **Kekurangan**: SEO menjadi tantangan karena SPA bergantung pada JavaScript untuk memuat konten. Lebih kompleks dalam pengelolaan status pengguna dan rute aplikasi.

**2.3. Rekomendasi Arsitektur**

* **Rekomendasi**: Berdasarkan analisis kebutuhan, proyek platform toko sembako direkomendasikan untuk menggunakan Model-View-Controller (MVC) sebagai arsitektur utama.Alasan: Pemisahan logika bisnis, kontrol, dan tampilan akan memudahkan pengelolaan aplikasi. Cocok untuk aplikasi toko sembako yang bersifat monolitik dengan kebutuhan pengembangan berkelanjutan.

**2. Desain Struktur Database Awal**

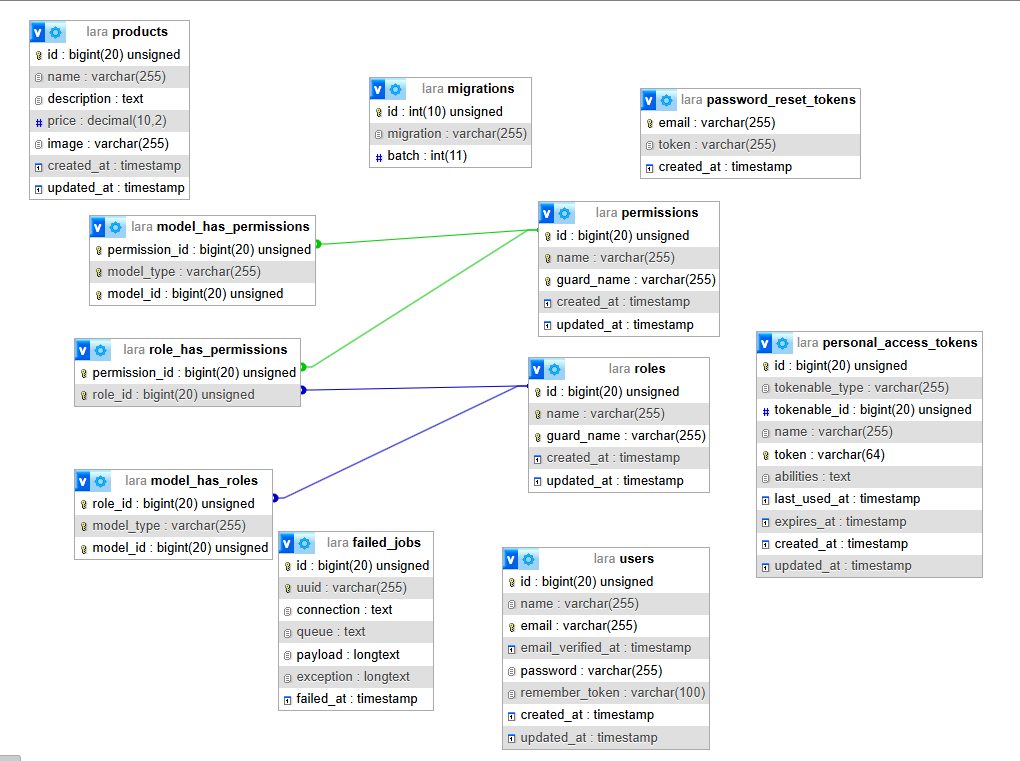
**2.1. Tujuan Desain Struktur Database**

* Memastikan bahwa komponen frontend, backend, dan database dapat berkomunikasi dengan lancar.
* Mendukung pengalaman pengguna yang mulus dengan respons yang cepat dari server ke antarmuka pengguna.

**2.2. Desain Struktur Database**

**2.2.1. Diagram Entity-Relationship (ER)**

* **Deskripsi**: Diagram ER menggambarkan entitas dalam sistem, atribut mereka, dan hubungan antar entitas.
* **Diagram**:



**2.2.2. Tabel-Tabel Utama**

* **Tabel Pengguna**
  + **Kolom**: UserID (PK), Username, PasswordHash, Email, CreatedAt, UpdatedAt
* **Tabel Produk**
  + **Kolom**: ProductID (PK), ProductName, Description, Price, CreatedAt, UpdatedAt
* **Tabel Pesanan**
  + **Kolom**: OrderID (PK), UserID (FK), OrderDate, TotalAmount, Status
* **Tabel DetailPesanan**
  + **Kolom**: OrderDetailID (PK), OrderID (FK), ProductID (FK), Quantity, Price

**2.3. Rekomendasi Struktur Database**

* **Rekomendasi**: Struktur database yang diusulkan ini dirancang untuk memastikan integritas data dan efisiensi operasi. Tabel-tabel utama telah diidentifikasi dan hubungan antar tabel diatur untuk mendukung fungsionalitas aplikasi yang diinginkan.

**3. Pemilihan Teknologi Frontend dan Backend**

**3.1. Tujuan Pemilihan Teknologi**

* Memastikan teknologi yang dipilih dapat memenuhi kebutuhan fungsional dan teknis aplikasi.
* Mempertimbangkan aspek performa, skalabilitas, dan kemudahan pemeliharaan.

**3.2. Pemilihan Teknologi Frontend**

**3.2.1. Framework/Library**

* **Pilihan**: React, Angular, Vue.js
* **Deskripsi**: Framework/library untuk membangun antarmuka pengguna yang interaktif.
* **Kriteria Pemilihan**: [misalnya, performa, komunitas, dokumentasi, kemudahan integrasi dengan backend]

**3.2.2. Teknologi Tambahan**

* **CSS Framework**: Bootstrap, Tailwind CSS
* **Tooling**: Webpack, Babel

**3.3. Pemilihan Teknologi Backend**

**3.3.1. Bahasa Pemrograman dan Framework**

* **Pilihan**: Node.js (Express), Python (Django/Flask), Ruby (Rails), Java (Spring Boot)
* **Deskripsi**: Bahasa dan framework untuk pengembangan logika aplikasi dan layanan backend.
* **Kriteria Pemilihan**: [misalnya, performa, kemudahan pengembangan, dukungan komunitas, integrasi dengan database]

**3.3.2. Basis Data**

* **Pilihan**: MySQL, PostgreSQL, MongoDB
* **Deskripsi**: Sistem manajemen basis data yang digunakan untuk penyimpanan data.
* **Kriteria Pemilihan**: [misalnya, model data, skalabilitas, fitur, dukungan transaksi]

**3.4. Rekomendasi Teknologi**

* **Frontend**: **[pilihan teknologi frontend yang dipilih]** karena [alasan spesifik yang mendukung pilihan ini].
* **Backend**: **[pilihan teknologi backend yang dipilih]** karena [alasan spesifik yang mendukung pilihan ini].