**2. Pemilihan Arsitektur Aplikasi**

**2.1. Tujuan Pemilihan Arsitektur**

* Menyediakan dasar yang solid untuk pengembangan, pemeliharaan, dan skalabilitas aplikasi.
* Memastikan bahwa arsitektur yang dipilih mampu mendukung aksesibilitas terbatas (hanya Super Admin dan Admin Toko Kewirausahaan) serta dapat menangani pemrosesan transaksi menggunakan kode QR dari siswa.

**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), View (menyajikan data kepada pengguna), dan Controller (menangani input dan memperbarui model).
* **Keuntungan** : Struktur yang terorganisir dan modular sehingga memudahkan pengembangan fitur baru atau pemeliharaan. Cocok untuk aplikasi dengan alur transaksi sederhana seperti pemindaian QR untuk verifikasi.
* **Kekurangan** : Bisa menjadi kompleks jika tidak dikelola dengan baik, terutama dalam aplikasi yang sangat besar.

**2.2.2. Microservices**

* **Deskripsi** : Pendekatan arsitektur di mana aplikasi dibagi menjadi serangkaian layanan kecil dan independen yang dapat dikembangkan dan disebarkan secara mandiri.
* **Keuntungan** : Setiap layanan bisa ditangani secara independen seperti layanan pemrosesan QR, pemantauan transaksi, dan manajemen pengguna.
* **Kekurangan** : Memerlukan manajemen yang baik dan komunikasi antar-layanan yang efisien.

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

* **Deskripsi** : Aplikasi web yang memuat satu halaman HTML dan mengupdate konten secara dinamis dengan JavaScript.
* **Keuntungan** : Pengalaman pengguna yang lebih mulus dan interaktif. Cocok untuk aplikasi dengan sedikit pengguna (Super Admin dan Admin Toko) yang membutuhkan respon cepat saat transaksi.
* **Kekurangan** : SEO mungkin tidak terlalu relevan karena sistem ini hanya diakses oleh Admin. Pengelolaan state dan rute lebih kompleks pada skala besar.

**2.3. Rekomendasi Arsitektur**

* **Rekomendasi**: Berdasarkan analisis kebutuhan, proyek ini direkomendasikan untuk menggunakan **Arsitektur Model-View-Controller (MVC)**. Hal ini Hal ini karena MVC memberikan struktur yang lebih mudah dikelola, dengan pemisahan logika bisnis, tampilan, dan kontrol. Untuk aplikasi pemantauan transaksi yang cukup sederhana dengan pemrosesan kode QR dan manajemen pengguna yang terbatas, MVC cukup efisien tanpa menambah kompleksitas.

**2. Desain Struktur Database Awal**

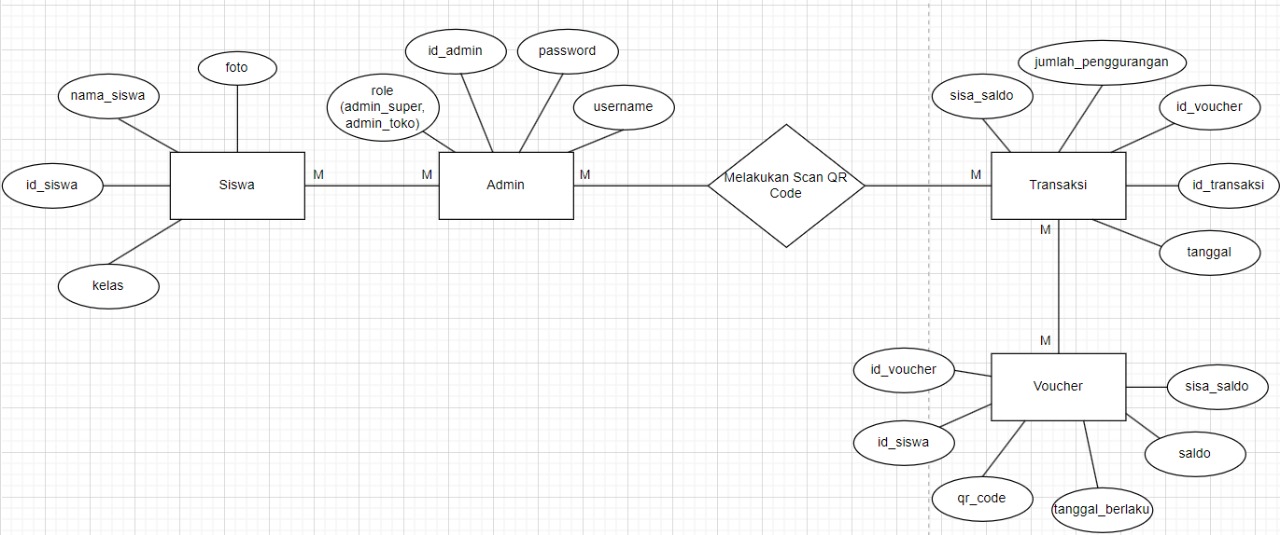
**2.1. Tujuan Desain Struktur Database**

* Memastikan penyimpanan data yang efisien dan terstruktur dengan baik untuk mempermudah pengelolaan transaksi serta informasi pengguna (Super Admin dan Admin Toko).
* Memastikan semua transaksi dicatat dengan baik.

**2.2. Desain Struktur Database**

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

* **Deskripsi**: Diagram ER menggambarkan entitas utama dalam sistem seperti pengguna (Super Admin dan Admin), entitas siswa, serta transaksi yang melibatkan pemindaian kode QR siswa.
* **Diagram**:



**2.2.2. Tabel-Tabel Utama**

* **Tabel Admin**
  + **Kolom :** id\_admin (PK), username, password, role.
* **Tabel Siswa**
  + **Kolom** : id\_siswa(PK), nama\_siswa, foto, kelas.
* **Tabel Voucheer**
  + **Kolom :** id\_voucheer(PK), id-siswa(FK), tanggal\_berlaku, qr\_code, saldo, sisa\_saldo.
* **Tabel Transaksi**
  + **Kolom** : id\_transaksi(PK), id\_voucheer(FK), jumlah\_pengggurangan, sisa\_saldo, taggal.

**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, termasuk pengelolaan data siswa, dan administrasi toko.

**3. Pemilihan Teknologi Frontend dan Backend**

**3.1. Tujuan Pemilihan Teknologi**

* Memilih teknologi yang stabil dan dapat diandalkan untuk mendukung performa sistem serta memenuhi kebutuhan transaksi dan pemantauan.

**3.2. Pemilihan Teknologi Frontend**

**3.2.1. Framework/Library**

* **Deskripsi** : laravel
* **Kriteria Pemilihan** : performa, komunitas, dokumentasi, kemudahan integrasi dengan backend.

**3.3. Pemilihan Teknologi Backend**

**3.3.1. Bahasa Pemrograman dan Framework**

* **Pilihan** : PHP dengan Laravel.
* **Deskripsi** : Framework PHP untuk pengembangan logika aplikasi dan layanan backend.
* **Kriteria Pemilihan**: kemudahan dalam melakukan transaksi

**3.3.2. Basis Data**

* **Pilihan** : MySQL
* **Deskripsi** : Sistem manajemen basis data relasional yang digunakan untuk penyimpanan data.
* **Kriteria Pemilihan** :model data, skalabilitas, fitur, dukungan transaksi.

**3.4. Rekomendasi Teknologi**

* **Backend Dan Frontend** : PHP dengan Laravel karena kemudahan pengembangan, dukungan komunitas yang kuat, dan kemampuan untuk menangani logika bisnis yang kompleks.