**2. Pemilihan Arsitektur Aplikasi**

**2.1. Tujuan Pemilihan Arsitektur**

* Menyediakan fondasi yang stabil untuk pengembangan, pemeliharaan, dan skalabilitas website makanan kebab delicios.
* 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, 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.
* **Kekurangan**: Kompleksitas dapat meningkat jika tidak diatur dengan baik, terutama untuk aplikasi yang terus berkembang.

**2.2.2. Microservices**

* **Deskripsi**: website ini 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 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 yangberubah Ketika stok habis dan ada produk baru. 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 website Kebab Delicious direkomendasikan untuk menggunakan arsitektur **Model-View-Controller (MVC)**.
* **Alasan:** Pemisahan logika bisnis, kontrol, dan tampilan mempermudah pengelolaan aplikasi. Pendekatan ini cocok untuk web penjualan kebab yang bersifat monolitik, dengan kebutuhan pengembangan yang berkelanjutan.

**2. Desain Struktur Database Awal**

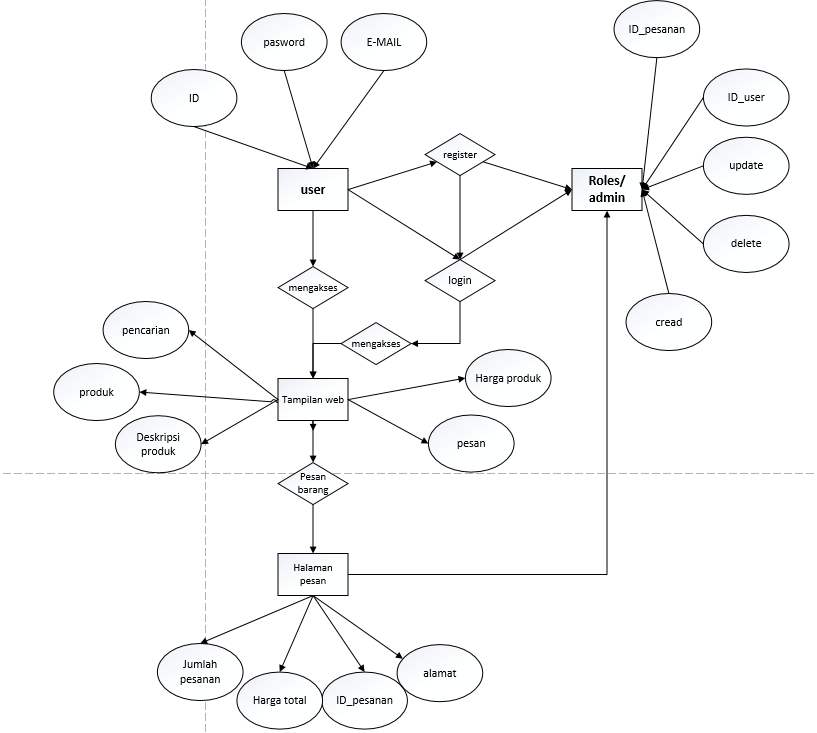
**2.1. Tujuan Desain Struktur Database**

* 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**: Laravel Blade
* **Deskripsi**: Laravel Blade adalah template engine bawaan Laravel yang digunakan untuk membangun antarmuka pengguna yang dinamis dan terintegrasi langsung dengan data dari aplikasi. Blade memungkinkan pengembang menggunakan sintaks yang sederhana dan bersih untuk mengelola elemen HTML serta logika presentasi.
* **Kriteria Pemilihan**:
  + Blade dirancang khusus untuk bekerja mulus dengan framework Laravel, mempermudah pengelolaan data dan tampilan dalam proyek monolitik.
  + Membantu pengembang mengurangi kompleksitas dalam menyusun tampilan tanpa kehilangan fleksibilitas.
  + Memungkinkan pembuatan komponen UI yang dapat digunakan kembali, sehingga meningkatkan efisiensi pengembangan.
  + Blade mendukung mekanisme cache untuk meningkatkan performa rendering tampilan.

**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**: PHP
* **Deskripsi**: PHP adalah bahasa pemrograman server-side yang populer untuk pengembangan aplikasi web. Laravel adalah framework PHP modern yang menyediakan fitur lengkap untuk pengembangan aplikasi berbasis web, seperti routing, middleware, autentikasi, dan ORM (Eloquent). Laravel dirancang untuk meningkatkan produktivitas pengembang dengan sintaks yang elegan dan alat yang intuitif.
* **Kriteria Pemilihan**:
  + Laravel menyediakan banyak fitur bawaan, seperti migrasi database, sistem autentikasi, dan manajemen cache, yang mempercepat proses pengembangan.
  + Laravel memiliki komunitas yang besar dan aktif, sehingga mudah menemukan dokumentasi, tutorial, dan bantuan dari pengembang lain.
  + Laravel dapat menangani aplikasi dengan jumlah pengguna yang besar, terutama dengan optimasi seperti caching dan queue handling.

**3.3.2. Basis Data**

* **Pilihan**: MySQL
* **Deskripsi**: MySQL adalah sistem manajemen basis data relasional (RDBMS) yang bersifat open-source dan digunakan secara luas untuk aplikasi web. MySQL mendukung struktur data berbasis tabel, memungkinkan pengelolaan data secara efisien, serta mendukung berbagai operasi seperti query, manipulasi data, dan manajemen skema. MySQL juga kompatibel dengan berbagai bahasa pemrograman, termasuk PHP, sehingga menjadi pilihan populer untuk aplikasi berbasis web.
* **Kriteria Pemilihan**:
  + MySQL menggunakan model data berbasis tabel, yang ideal untuk pengelolaan data terstruktur dengan hubungan antar entitas.
  + MySQL dapat menangani skala aplikasi mulai dari kecil hingga besar, dengan dukungan untuk replikasi dan sharding untuk meningkatkan performa.
  + Mendukung fitur seperti transaksi ACID, indexing, dan query lanjutan menggunakan SQL, yang cocok untuk aplikasi dengan kebutuhan kompleks.

**3.4. Rekomendasi Teknologi**

* **Frontend**:

Laravel Blade karena ini adalah template engine bawaan Laravel yang memungkinkan pengembangan antarmuka pengguna dengan sintaks yang sederhana dan mudah dipahami. Blade juga mendukung logika presentasi yang bersih, kemampuan caching, dan integrasi sempurna dengan data dari backend Laravel.

* **Backend**:

Laravel (PHP) karena framework ini menyediakan banyak fitur bawaan untuk pengembangan aplikasi web yang efisien, seperti routing, middleware, autentikasi, dan ORM (Eloquent). Laravel juga memiliki dokumentasi lengkap, komunitas yang aktif, dan mendukung praktik pengembangan yang modern dan terstruktur.

* **Database**:

MySQL karena merupakan sistem manajemen basis data relasional (RDBMS) yang handal, skalabel, dan memiliki kompatibilitas tinggi dengan Laravel. MySQL mendukung fitur seperti transaksi ACID, indexing, dan kemampuan untuk menangani data dalam jumlah besar dengan performa optimal.