**2. Pemilihan Arsitektur Aplikasi**

**2.1. Tujuan Pemilihan Arsitektur**

* Memastikan bahwa aplikasi terus beroperasi dengan kinerja optimal melalui pemantauan, pengujian, dan penyesuaian konfigurasi sesuai kebutuhan
* Mendukung kemudahan pemeliharaan dan pengembangan lebih lanjut, termasuk penambahan fitur baru, perbaikan bug, dan peningkatan performa tanpa mengganggu layanan yang sudah berjalan
* Menjaga efisiensi biaya dalam operasional dan pemeliharaan aplikasi dengan memanfaatkan sumber daya secara optimal dan mempertimbangkan opsi-opsi untuk mengurangi biaya tanpa mengorbankan kualitas dan kinerja

**2.2. Pilihan Arsitektur**

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

* **Deskripsi**: Model-View-Controller (MVC) memisahkan aplikasi menjadi tiga komponen utama: Model, View, dan Controller. **Model** mengelola data dan logika bisnis, **View** menangani presentasi data kepada pengguna, dan **Controller** berfungsi sebagai penghubung antara Model dan View, mengatur aliran data dan mengontrol interaksi pengguna dengan sistem.
* **Keuntungan**: Memisahkan logika bisnis, antarmuka pengguna, dan kontrol aliran data membuat pemeliharaan dan pengembangan lebih mudah.
* **Kekurangan**: Meskipun bermanfaat untuk aplikasi besar, MVC dapat membuat aplikasi kecil menjadi terlalu rumit dan mengganggu kinerja.

**2.2.2. Microservices**

* **Deskripsi**: aplikasi dibangun sebagai kumpulan layanan kecil yang independen. Setiap layanan berfokus pada fungsi tertentu dan dapat dikembangkan, diterapkan, dan diskalakan secara terpisah.
* **Keuntungan**: Setiap layanan dapat dibangun dengan teknologi yang paling sesuai untuk fungsinya.
* **Kekurangan**: Mengelola komunikasi antar layanan dapat menjadi kompleks dan memerlukan tambahan layer seperti API gateway.

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

* **Deskripsi**: aplikasi web yang memuat satu halaman HTML dan memperbarui konten secara dinamis tanpa memuat ulang seluruh halaman dan memberikan pengalaman pengguna yang lebih responsif dan interaktif.
* **Keuntungan**: memberikan pengalaman pengguna yang lebih responsif dan cepat karena tidak perlu memuat ulang halaman secara keseluruhan.
* **Kekurangan**: Memuat semua konten di awal bisa menyebabkan waktu muat awal yang lebih lama.

**2.3. Rekomendasi Arsitektur**

* **Rekomendasi**: Berdasarkan analisis kebutuhan, pengembangan proyek Cathering online “Dapur Fauzan” direkomendasikan untuk menggunakan arsitektur **Model-View-Controller (MVC)**.

**Alasan :** Hal ini dikarenakan MVC menawarkan pemisahan yang jelas antara logika bisnis, antarmuka pengguna, dan kontrol aliran data, sehingga memudahkan pemeliharaan dan pengembangan lebih lanjut. Selain itu, Laravel, yang merupakan framework PHP berorientasi MVC, memiliki dokumentasi yang baik dan komunitas yang besar, yang akan sangat membantu dalam proses pengembangan dan pemecahan masalah. Dengan struktur yang modular, MVC juga memungkinkan skalabilitas yang baik untuk mengakomodasi pertumbuhan platform kedepannya.

**2. Desain Struktur Database Awal**

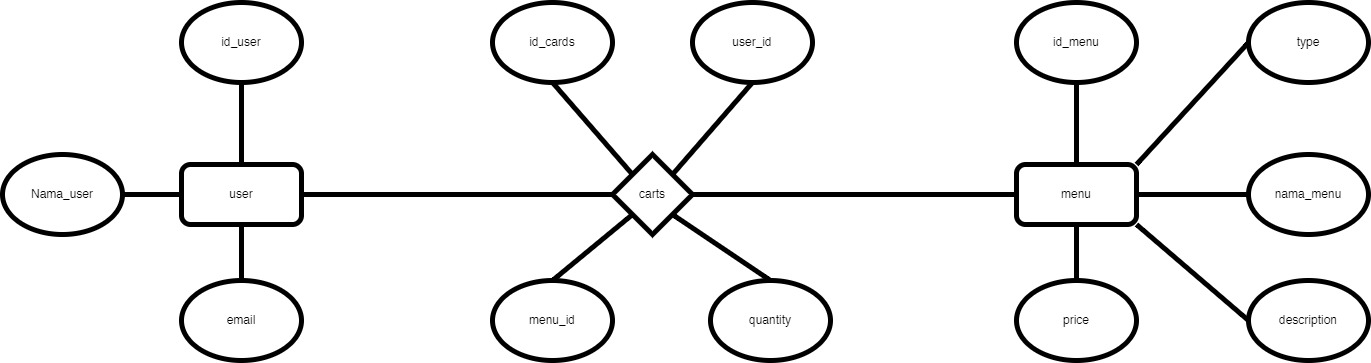
**2.1. Tujuan Desain Struktur Database**

* Menyediakan data yang relevan dan dibutuhkan oleh pengguna dan aplikasi. Dengan struktur yang baik, pengguna dapat dengan mudah mengakses informasi yang diperlukan untuk menjalankan proses bisnis secara efisien
* Mendukung kebutuhan pemrosesan transaksi bisnis. Dengan demikian, sistem dapat mengelola berbagai operasi, seperti pemesanan makanan, pengolahan pembayaran, dan manajemen inventaris secara efektif

**2.2. Desain Struktur Database**

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

* **Deskripsi**: ERD menunjukkan hubungan antar entitas dalam sistem serta atribut yang dimiliki oleh masing-masing entitas. Dengan menggunakan ERD, pengembang dapat memvisualisasikan bagaimana data saling terhubung, sehingga mempermudah dalam perancangan dan pengelolaan database
* **Diagram**:



**2.2.2. Tabel-Tabel Utama**

* **Tabel Pengguna**
* **Kolom:**
* id (PK): ID unik untuk setiap pengguna.
* name: Nama pengguna.
* email: Alamat email pengguna.
* email\_verified\_at: Waktu email diverifikasi (NULL jika belum diverifikasi).
* password: Hash dari password pengguna untuk autentikasi.
* remember\_token: Token autentikasi untuk sesi diperpanjang (opsional).
* created\_at: Tanggal dan waktu pembuatan data pengguna.
* updated\_at: Tanggal dan waktu terakhir pembaruan data pengguna.
* UserID (PK), Username, PasswordHash, Email, CreatedAt, UpdatedAt
* **Tabel Produk**
* **Kolom:**
  + id (PK): ID unik untuk setiap produk.
  + type: Kategori atau jenis produk (misalnya, "Makanan", "Snack").
  + name: Nama produk.
  + description: Deskripsi singkat tentang produk.
  + image: Path atau URL lokasi gambar produk.
  + price: Harga produk.
  + created\_at: Tanggal dan waktu pembuatan data produk.
  + updated\_at: Tanggal dan waktu terakhir pembaruan data produk.
* **Tabel Pesanan**
* **Kolom:**
* id (PK): ID unik untuk setiap detail pesanan.
* user\_id (FK): Merujuk ke tabel pengguna yang melakukan pesanan.
* menu\_id (FK): Merujuk ke tabel produk (menu) yang dipesan.
* quantity: Jumlah produk yang dipesan.
* created\_at: Tanggal dan waktu pembuatan detail pesanan.
* updated\_at: Tanggal dan waktu terakhir pembaruan detail pesanan.

**2.3. Rekomendasi Struktur Database**

**Rekomendasi**: Struktur database yang diusulkan ini dirancang untuk memastikan integritas data dan efisiensi operasi dalam mendukung fungsionalitas aplikasi. Tabel-tabel utama telah diidentifikasi dan dirancang dengan relasi yang jelas, mencakup:

* **Tabel Pengguna** untuk mengelola data pengguna, termasuk autentikasi dan otorisasi.
* **Tabel Produk** untuk menyimpan informasi menu/produk yang ditawarkan.
* **Tabel Pesanan** untuk mencatat informasi pesanan utama yang dilakukan oleh pengguna.
* **Tabel DetailPesanan** untuk menyimpan rincian setiap produk dalam pesanan, termasuk jumlah dan harga.

Hubungan antar tabel dirancang dengan menerapkan konsep **normalisasi database** untuk mengurangi redundansi data dan meningkatkan efisiensi. Struktur ini juga memungkinkan skalabilitas untuk mendukung pertumbuhan data di masa mendatang. Dengan rancangan ini, aplikasi dapat memproses transaksi bisnis secara efisien, seperti pemesanan makanan, pengelolaan inventaris, dan pengolahan pembayaran.

**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 (Template Engine bawaan Laravel)
* **Deskripsi**: Laravel Blade adalah sistem template bawaan Laravel yang memudahkan pengembangan antarmuka pengguna dengan sintaks sederhana dan fleksibel.
* **Kriteria Pemilihan**:
* Integrasi penuh dengan Laravel Framework.
* Sintaks yang sederhana dan mudah dipahami.
  + Mendukung penggunaan komponen UI yang dapat digunakan kembali.

**3.2.2. Teknologi Tambahan**

* **CSS Framework:** Tailwind CSS
  + **Deskripsi:** Framework CSS modern yang memungkinkan pengembangan desain antarmuka dengan cepat dan konsisten.
* **Tooling:** Desain antarmuka dikembangkan menggunakan **Figma** untuk memastikan desain yang terorganisir dan mudah diimplementasikan.**3.3. Pemilihan Teknologi Backend**

**3.3.1. Bahasa Pemrograman dan Framework**

* **Pilihan:** PHP (Laravel 10)
* **Deskripsi:** Laravel adalah framework PHP modern yang berorientasi pada arsitektur MVC (Model-View-Controller), mendukung pengembangan aplikasi web dengan efisiensi tinggi.
* **Kriteria Pemilihan:**
  + Framework yang mendukung berbagai fitur bawaan seperti routing, middleware, dan ORM (Eloquent).
  + Dokumentasi yang lengkap dan komunitas besar yang memudahkan pemecahan masalah.
  + Memiliki keamanan bawaan seperti validasi input dan proteksi CSRF.

**3.3.2. Basis Data**

* **Pilihan:** MySQL (Melalui XAMPP)
* **Deskripsi:** MySQL adalah sistem manajemen basis data relasional (RDBMS) yang andal, cepat, dan mendukung transaksi. Dijalankan menggunakan server lokal melalui **XAMPP** untuk pengembangan.
* **Kriteria Pemilihan:**
* Mendukung model data relasional yang sesuai dengan kebutuhan aplikasi.
* Kompatibel dengan Laravel Eloquent ORM.
* Dokumentasi luas dan dukungan komunitas yang besar.

**3.4. Rekomendasi Teknologi**

* **Frontend:** Laravel Blade dan Tailwind CSS dipilih karena memungkinkan pengembangan antarmuka yang responsif, mudah dikelola, dan memiliki integrasi yang baik dengan framework Laravel.
* **Backend:** Laravel 10 dipilih karena efisiensinya dalam pengembangan aplikasi berbasis MVC, fitur bawaan yang lengkap, serta komunitas yang besar yang dapat mendukung pengembangan dan pemeliharaan aplikasi.
*  **Database:** MySQL dipilih karena kesesuaian dengan Laravel dan keandalan dalam mengelola data aplikasi secara relasional, serta XAMPP digunakan sebagai lingkungan pengembangan lokal yang mudah diatur.