**Bước 3 (Y3): Thiết kế Kiến trúc Tổng quan - ByteBite**

Tài liệu này trình bày mô hình kiến trúc, sơ đồ tổng quan, và các lựa chọn công nghệ cho hệ thống ByteBite, tập trung vào ngăn xếp công nghệ C# (.NET) và SQL Server.

**1. Lựa chọn Mô hình Kiến trúc**

**Mô hình được chọn: Kiến trúc Monolith Module hóa (Modular Monolith)**

Đây là một biến thể của kiến trúc 3 lớp (3-Layer) truyền thống, nhưng được cấu trúc hóa một cách rõ ràng. Thay vì một "khối" (monolith) duy nhất, chúng ta chia ứng dụng Backend thành các "Module" logic riêng biệt (ví dụ: UserModule, OrderModule, RestaurantModule).

**Lý do lựa chọn:**

1. **Phù hợp với .NET:** Ngăn xếp C# (ASP.NET Core) rất mạnh mẽ trong việc xây dựng các ứng dụng Monolith. Cấu trúc "Solution" của Visual Studio cho phép chúng ta tạo nhiều "Project" (dự án) riêng biệt cho từng Module, giúp giữ mã nguồn sạch sẽ và tách bạch.
2. **Phù hợp Phạm vi:** Với các giới hạn (chỉ COD, không có thuật toán phức tạp), một hệ thống Microservices đầy đủ là quá phức tạp (overkill). Monolith Module hóa cung cấp sự cân bằng hoàn hảo: đủ đơn giản để triển khai, đủ cấu trúc để bảo trì.
3. **Quản lý Dữ liệu:** Tất cả các module có thể cùng truy cập một cơ sở dữ liệu (Database) SQL Server duy nhất, giúp đơn giản hóa các giao dịch (transactions). Ví dụ, khi đặt hàng (Order), ta có thể dễ dàng kiểm tra món ăn (Restaurant) và cập nhật thông tin người dùng (User) trong cùng một giao dịch.
4. **Hiệu suất:** Giao tiếp giữa các module diễn ra trong bộ nhớ (in-memory function calls) thay vì qua mạng (network calls như Microservices), giúp giảm độ trễ đáng kể.
5. **Dễ dàng Phát triển & Kiểm thử:** Toàn bộ hệ thống có thể được chạy và gỡ lỗi (debug) trên một máy phát triển duy nhất.

**2. Sơ đồ Kiến trúc Tổng quan**

Kiến trúc tổng thể của ByteBite sẽ bao gồm các thành phần chính sau:

a. Clients (Giao diện Người dùng):

Đây là lớp giao diện (Frontend) mà các tác nhân (actors) tương tác. Do bạn chọn html/css/javascript, chúng ta sẽ xây dựng các Ứng dụng Web Đáp ứng (Responsive Web Apps) hoặc Progressive Web Apps (PWA).

* **User Web App:** Giao diện cho Người dùng đặt món, theo dõi đơn (F2, F3, F4).
* **Restaurant Web App:** Giao diện cho Nhà hàng quản lý menu, xác nhận đơn (F5).
* **Shipper Web App:** Giao diện cho Tài xế nhận và giao đơn (F6).
* **Admin Portal:** Trang quản trị (nội bộ) để quản lý toàn hệ thống.

b. API Gateway:

Là một lớp trung gian (sử dụng C# với Ocelot hoặc YARP) đứng giữa Clients và Backend.

* **Chức năng:** Điểm vào duy nhất cho mọi request.
* **Nhiệm vụ:** Xác thực (Authentication) token, điều phối (Routing) request đến đúng Module, và bảo vệ (Rate Limiting).

c. Application Layer (Backend - .NET Modular Monolith):

Đây là "trái tim" của hệ thống, được xây dựng bằng ASP.NET Core Web API (C#). Nó được chia thành các Module logic:

* **Identity Module (Module Định danh):** Xử lý F1 (Đăng ký, Đăng nhập, Quản lý Hồ sơ).
* **Restaurant Module (Module Nhà hàng):** Xử lý F2, F5 (Quản lý Menu, Quản lý Nhà hàng, Tìm kiếm).
* **Order Module (Module Đơn hàng):** Xử lý F3, F7.3 (Quản lý Giỏ hàng, Đặt hàng, Lịch sử).
* **Delivery Module (Module Giao vận):** Xử lý F4, F6 (Quản lý Tài xế, Cập nhật trạng thái giao).
* **Notification Module (Module Thông báo):** Xử lý F7.1, F7.2 (Gửi thông báo, Quản lý Đánh giá).

**d. Data Storage (Lưu trữ):**

* **Cơ sở dữ liệu chính (SQL Server):** Lưu trữ toàn bộ dữ liệu nghiệp vụ (người dùng, nhà hàng, món ăn, đơn hàng). Các module sẽ chia sẻ chung Database này nhưng có thể sử dụng các Schema riêng biệt để phân tách.
* **Dịch vụ Cache (Gợi ý: Redis):** Dùng để lưu trữ các dữ liệu truy cập thường xuyên (session đăng nhập, menu nhà hàng) để tăng tốc độ phản hồi.

e. External Services (Dịch vụ bên ngoài):

Các dịch vụ của bên thứ ba mà hệ thống tích hợp:

* **Dịch vụ Bản đồ (Google Maps API):** Tính toán khoảng cách và (có thể) hiển thị vị trí.
* **Dịch vụ SMS (Vd: Twilio, VNPT SMS...):** Gửi mã OTP (F1.1).

**3. Đề xuất Công nghệ chi tiết**

Dựa trên yêu cầu của bạn, đây là ngăn xếp công nghệ cụ thể:

* **Frontend (Clients):** HTML, CSS, JavaScript.
* *Gợi ý Framework:* **React.js** hoặc **Vue.js** (để xây dựng các Web App hiện đại, dễ bảo trì) và **Bootstrap/TailwindCSS** (để có giao diện đẹp, đáp ứng nhanh).
* **Backend (Application Layer & API Gateway):**
* **C# 12** với **.NET 8 (ASP.NET Core)**.
* **Entity Framework Core (EF Core):** Để làm việc với SQL Server.
* **Database:**
* **Microsoft SQL Server**.
* **Công nghệ Real-time (Rất quan trọng):**
* **SignalR (tích hợp sẵn trong .NET):** Bắt buộc phải có. Dùng để đẩy thông báo thời gian thực từ Backend đến Clients (ví dụ: Nhà hàng nhận đơn mới, User thấy tài xế đang di chuyển).