### **PRD: Nền tảng Giao đồ ăn FoodFast Delivery**

#### **Problem Alignment (Phân tích Vấn đề)**

* **Đối với Nhà bán hàng (Nhà hàng/Quán ăn):** Các nhà hàng vừa và nhỏ thiếu nguồn lực công nghệ và kinh phí để xây dựng hệ thống đặt hàng và giao vận riêng, làm hạn chế khả năng tiếp cận khách hàng trực tuyến.
* **Đối với Người mua (Khách hàng):** Khách hàng mong muốn có một nền tảng duy nhất cung cấp đa dạng lựa chọn từ nhiều nhà hàng khác nhau, với quy trình đặt hàng, thanh toán và theo dõi giao hàng liền mạch, đáng tin cậy.
* **Đối với Nền tảng:** Việc xây dựng một hệ thống có khả năng mở rộng để xử lý đồng thời hàng nghìn đơn hàng, quản lý nhiều đối tác, và đảm bảo tính nhất quán dữ liệu là một thách thức kỹ thuật lớn.

#### **High Level Approach (Hướng tiếp cận Tổng thể)**

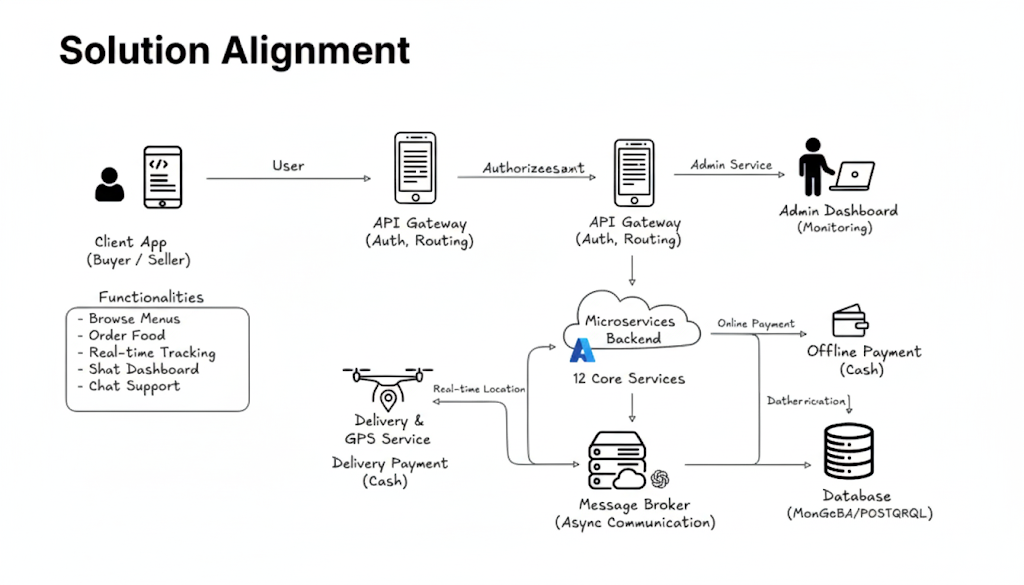
* Xây dựng một nền tảng Sàn giao dịch đa đối tác (Multi-Vendor Marketplace) dựa trên kiến trúc Microservices.
* Phân tách rõ ràng các nghiệp vụ cốt lõi vào 12 service độc lập để đảm bảo khả năng phát triển, bảo trì và mở rộng linh hoạt.
* Sử dụng phương thức giao tiếp bất đồng bộ (Message Broker) cho các luồng nghiệp vụ quan trọng (đặt hàng, thanh toán, giao vận) để tăng cường sự ổn định và nhất quán dữ liệu.
* Tích hợp hệ thống theo dõi và giám sát (Monitoring & Logging) toàn diện để đảm bảo khả năng quan sát (Observability) và chủ động phát hiện sự cố.

#### **Narrative (Kịch bản Người dùng)**

* **Chị An, chủ một quán ăn nhỏ,** muốn mở rộng kinh doanh. Chị đăng ký tài khoản Nhà bán hàng trên FoodFast, hoàn tất xác thực (KYC), và dễ dàng đăng tải thực đơn, hình ảnh món ăn cũng như cập nhật số lượng suất ăn còn lại trong ngày.
* **Anh Bình, một nhân viên văn phòng,** cảm thấy đói và mở ứng dụng FoodFast. Anh thấy quán của chị An và nhiều quán khác. Anh quyết định đặt một phần "Cơm tấm sườn bì chả" từ quán chị An và một ly "Trà sữa trân châu" từ một cửa hàng khác trong cùng một đơn hàng.
* **Hệ thống FoodFast** tự động xử lý đơn hàng của anh Bình:
  1. Tách đơn hàng thành hai phần cho hai quán khác nhau.
  2. Kiểm tra và tạm giữ tồn kho tại cả hai nơi.
  3. Xử lý thanh toán an toàn qua ví điện tử.
  4. Sau khi thanh toán thành công, hệ thống điều phối một drone (hoặc shipper) đến lấy hàng từ hai địa điểm và giao đến cho anh Bình.
  5. Anh Bình có thể theo dõi vị trí của drone và thời gian dự kiến nhận hàng (ETA) ngay trên ứng dụng.

#### **Goals (Mục tiêu)**

* Cung cấp một nền tảng ổn định, hiệu suất cao cho phép hàng nghìn Nhà bán hàng và hàng triệu Người mua tương tác một cách liền mạch.
* Đảm bảo tính nhất quán và toàn vẹn dữ liệu cho các giao dịch phức tạp như đặt hàng, thanh toán và quản lý tồn kho thông qua việc áp dụng mô hình Saga.
* Tối ưu hóa quy trình giao hàng bằng cách tự động hóa việc điều phối và cho phép theo dõi đơn hàng theo thời gian thực.
* Xây dựng một hệ thống có khả năng mở rộng linh hoạt (scalable) và dễ dàng bảo trì, nâng cấp từng thành phần mà không ảnh hưởng đến toàn bộ hệ thống.



### **Giải thích Chi tiết Sơ đồ Kiến trúc FoodFast Delivery**

Sơ đồ này mô tả một cách trực quan luồng di chuyển của dữ liệu và sự tương tác giữa các thành phần chính trong hệ thống, từ lúc người dùng thao tác trên ứng dụng cho đến khi đơn hàng được xử lý và giao đến tay họ.

### **Các Thành phần Chính (Key Components)**

#### **Client App (Ứng dụng Người dùng)**

* Đây là giao diện mà **Người mua (Buyer)** và **Người bán (Seller)** tương tác trực tiếp, thường là một ứng dụng di động.
* **Chức năng cho Người mua:**
  + Browse Menus: Xem thực đơn từ nhiều nhà hàng.
  + Order Food: Đặt món ăn.
  + Real-time Tracking: Theo dõi vị trí của shipper/drone theo thời gian thực.
  + Chat Support: Trò chuyện, hỗ trợ.
* **Chức năng cho Người bán (Seller Center):** Quản lý cửa hàng, đăng sản phẩm, cập nhật tồn kho, xem báo cáo doanh thu.

#### **Admin Dashboard (Trang Quản trị)**

* Đây là giao diện web dành riêng cho **Quản trị viên (Admin)** của hệ thống.
* **Vai trò:** Cho phép Admin theo dõi (Monitoring) sức khỏe toàn bộ hệ thống, quản lý người dùng/nhà hàng, xem các báo cáo kinh doanh tổng hợp, và xử lý các vấn đề vận hành.

#### **API Gateway (Cổng Giao tiếp)**

* Đây là **cửa ngõ duy nhất** cho mọi yêu cầu (request) từ Client App và Admin Dashboard gửi vào hệ thống backend.
* **Vai trò chính:**
  + **Auth (Authentication - Xác thực):** Kiểm tra xem người dùng là ai, đã đăng nhập hay chưa.
  + **Authorization (Ủy quyền):** Sau khi xác thực, nó kiểm tra xem người dùng này có quyền thực hiện hành động họ đang yêu cầu hay không (ví dụ: Seller không thể đặt hàng, Buyer không thể sửa menu).
  + **Routing (Định tuyến):** Hướng yêu cầu đến đúng microservice tương ứng ở phía sau để xử lý. Ví dụ, yêu cầu đăng nhập sẽ được chuyển đến User Service, yêu cầu đặt hàng sẽ được chuyển đến Order Service.

#### **Microservices Backend (Khối Backend)**

* Đây là "bộ não" của toàn bộ hệ thống, bao gồm **12 services cốt lõi** đã được định nghĩa trong tài liệu PRD.
* Mỗi service là một chương trình độc lập, chịu trách nhiệm cho một nghiệp vụ cụ thể (ví dụ: User Service quản lý người dùng, Product Service quản lý sản phẩm, Order Service quản lý đơn hàng).
* **Ưu điểm:** Việc chia nhỏ này giúp hệ thống dễ dàng nâng cấp, sửa lỗi và mở rộng từng phần mà không ảnh hưởng đến các phần khác.

#### **Message Broker (Hệ thống Hàng đợi Tin nhắn)**

* Đây là trung tâm giao tiếp **bất đồng bộ (Asynchronous Communication)** giữa các microservice.
* **Cách hoạt động (ví dụ):** Khi một đơn hàng được tạo, Order Service không cần phải "gọi điện" trực tiếp cho Payment Service và Delivery Service. Thay vào đó, nó chỉ cần "gửi một thông báo" (OrderCreated) vào Message Broker. Các service khác có nhiệm vụ lắng nghe và sẽ tự động nhận thông báo này để thực hiện công việc của mình.
* **Lợi ích:** Giúp hệ thống hoạt động ổn định và đáng tin cậy. Nếu Delivery Service bị lỗi tạm thời, thông báo vẫn nằm trong hàng đợi và sẽ được xử lý ngay khi service hoạt động trở lại, đảm bảo không bị mất dữ liệu. Đây là thành phần cốt lõi để triển khai mô hình Saga.

#### **Delivery & GPS Service**

* Đây là một service chuyên biệt, chịu trách nhiệm cho toàn bộ quy trình giao hàng.
* Nó nhận lệnh điều phối từ khối backend, tìm shipper/drone, và quản lý lộ trình.
* Điểm đặc biệt là nó gửi trực tiếp thông tin **Vị trí thời gian thực (Real-time Location)** về cho Client App để người dùng theo dõi. Luồng này thường đi riêng để giảm độ trễ.
* Nó cũng xử lý luồng **Thanh toán khi giao hàng (Delivery Payment - Cash)**, xác nhận shipper đã thu tiền và cập nhật lại trạng thái cho hệ thống.

#### **Database (Cơ sở dữ liệu)**

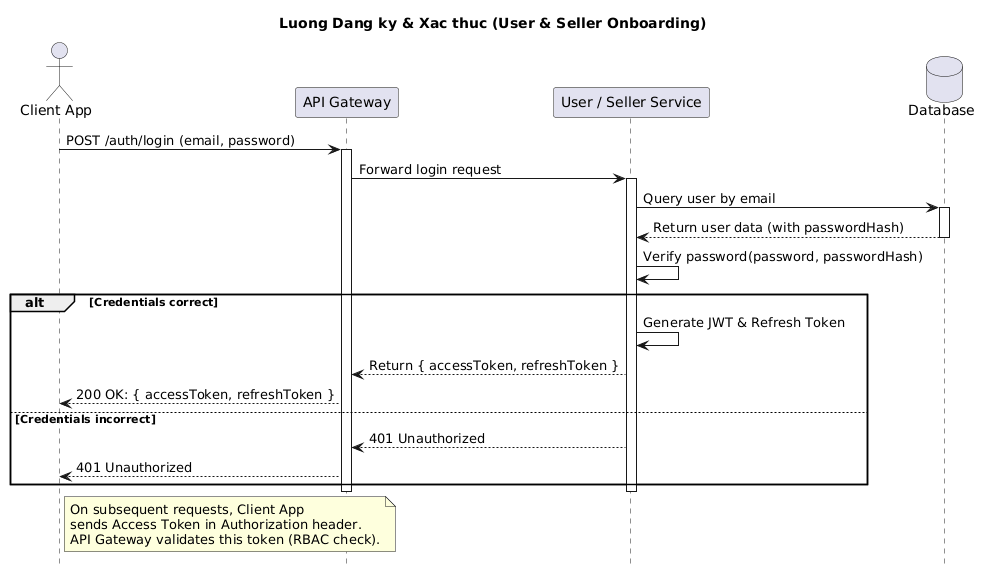
* Đây là nơi lưu trữ toàn bộ dữ liệu của hệ thống: thông tin người dùng, sản phẩm, đơn hàng, giao dịch,...

#### **Key Features (Tính năng Cốt lõi)**

1. **Hệ thống Quản lý Đối tác (Seller Center):** Cho phép Nhà bán hàng tự quản lý thông tin cửa hàng, thực đơn, sản phẩm, và số lượng tồn kho (Seller Service, Product Service, Inventory Service).
2. **Hệ thống Quản lý Người dùng & Phân quyền (User Management):** Quản lý tài khoản Người mua và Admin, đảm bảo phân quyền truy cập chặt chẽ qua vai trò (RBAC) (User Service).
3. **Quy trình Đặt hàng Đa nhà cung cấp (Multi-Vendor Ordering):** Cho phép người dùng đặt sản phẩm từ nhiều nhà hàng khác nhau trong cùng một giỏ hàng và hệ thống tự động xử lý tách đơn (Cart Service, Order Service).
4. **Cổng Thanh toán Tích hợp (Payment Gateway):** Xử lý giao dịch thanh toán an toàn, hỗ trợ nhiều phương thức và đảm bảo tính nhất quán với đơn hàng (Payment Service).
5. **Hệ thống Điều phối & Giao hàng Thông minh (Delivery Dispatch & Tracking):** Tự động tìm kiếm và phân công drone/shipper, lập lộ trình tối ưu từ vị trí của shop và cung cấp khả năng theo dõi GPS thời gian thực (Delivery Dispatcher Service, Delivery & GPS Service).
6. **Cổng API Tập trung (API Gateway):** Là điểm truy cập duy nhất cho mọi request, chịu trách nhiệm xác thực, giới hạn truy cập (rate limiting) và định tuyến thông minh.
7. **Hệ thống Giao tiếp Bất đồng bộ (Message Broker):** Đóng vai trò xương sống cho các giao dịch phức tạp, đảm bảo các service giao tiếp ổn định và đáng tin cậy.
8. **Hệ thống Giám sát & Ghi log (Monitoring & Logging):** Cung cấp khả năng quan sát toàn bộ hệ thống, theo dõi hiệu suất, và cảnh báo sự cố sớm thông qua Correlation ID.

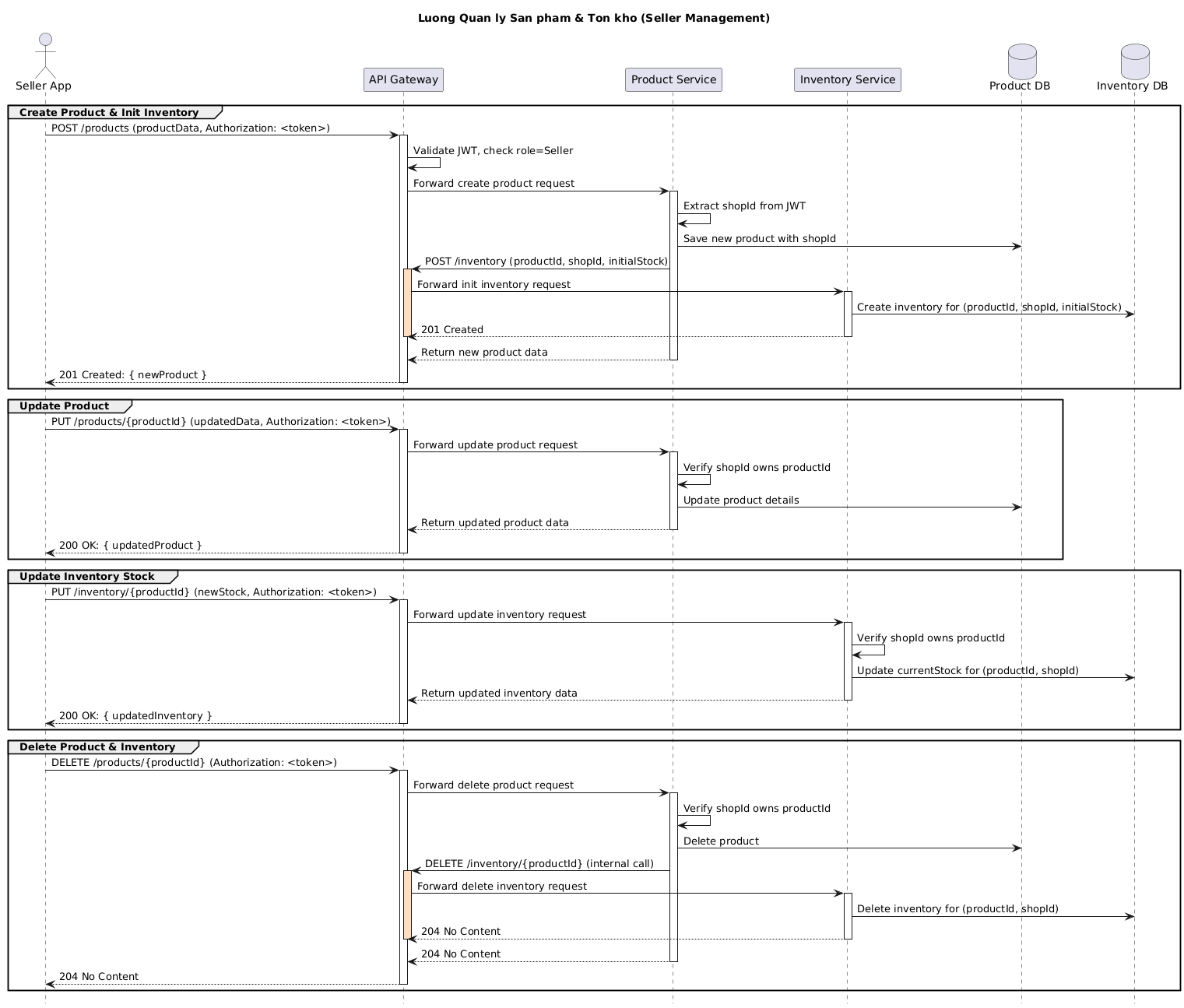
#### 

#### **Key Flows (Luồng Nghiệp vụ Cốt lõi)**



#### **1. Luồng Đăng ký & Xác thực (User & Seller Onboarding) - (Khối màu xanh lá cây)**

* **Client App**: Người dùng (Buyer hoặc Seller) bắt đầu luồng bằng cách gửi yêu cầu đăng ký/đăng nhập từ ứng dụng.
* **API Gateway**: Đây là điểm tiếp nhận yêu cầu. Nó thực hiện:
  + **Auth/RBAC**: Xác thực JWT/Refresh Token (nếu là đăng nhập) hoặc xử lý yêu cầu đăng ký. Nó cũng kiểm tra phân quyền (Role-Based Access Control - RBAC) để đảm bảo người dùng có quyền truy cập.
* **User Service / Seller Service**:
  + API Gateway sẽ định tuyến yêu cầu đến User Service (cho người mua/admin) hoặc Seller Service (cho người bán) tương ứng.
  + Các service này xử lý logic nghiệp vụ đăng ký/đăng nhập, mã hóa mật khẩu, và sau đó cấp JWT/Refresh Token cho Client App.
  + Cả hai service đều tương tác với Database riêng của mình để lưu trữ thông tin người dùng/người bán.



### **2. Luồng Quản lý Sản phẩm & Tồn kho (Seller Management)**

* **Client App (Seller App):** Seller gửi các yêu cầu như POST /products (để đăng món mới), PUT /products/{productId} (để cập nhật thông tin món ăn), PUT /inventory/{productId} (để cập nhật số lượng tồn kho), hoặc DELETE /products/{productId} (để xóa món ăn) từ ứng dụng Seller Center.
* **API Gateway:**
  + Nhận yêu cầu, thực hiện xác thực Token (JWT) và kiểm tra người dùng có vai trò là **Seller**.
  + Định tuyến yêu cầu đến service phù hợp (Product Service cho các thao tác về sản phẩm, Inventory Service cho các thao tác về tồn kho).
* **Product Service:**
  + Nhận các yêu cầu liên quan đến sản phẩm (tạo, cập nhật, xóa).
  + Thực hiện trích xuất shopId từ Token (ShopId Extraction) để đảm bảo Seller chỉ thao tác trên sản phẩm của cửa hàng mình.
  + Xử lý logic nghiệp vụ và tương tác với cơ sở dữ liệu sản phẩm (Product DB).
  + Khi tạo hoặc xóa sản phẩm, nó sẽ gửi một yêu cầu nội bộ (qua Gateway) đến Inventory Service để tạo hoặc xóa tồn kho tương ứng, đảm bảo dữ liệu nhất quán.
* **Inventory Service:**
  + Nhận các yêu cầu liên quan đến tồn kho (khởi tạo, cập nhật, xóa).
  + Xác minh rằng shopId của người yêu cầu sở hữu productId tương ứng.
  + Xử lý logic nghiệp vụ và tương tác với cơ sở dữ liệu tồn kho (Inventory DB), cập nhật các trường như currentStock.

#### 

#### **3. Luồng Đặt hàng & Thanh toán (Order & Payment Saga) - (Khối màu cam)**

* **Client App (Buyer App)**: Người mua xác nhận giỏ hàng.
* **Order Service**:
  + Nhận yêu cầu đặt hàng từ Client App (có thể qua Gateway). Xử lý logic tách đơn hàng (nếu có món từ nhiều shop).
  + Gửi yêu cầu Reserve Stock (tạm giữ tồn kho) đến Inventory Service.
  + Nếu tồn kho đủ, Order Service tạo đơn hàng với trạng thái PENDING và xuất bản OrderCreated Event lên Message Broker.
* **Message Broker**: Là trung tâm giao tiếp bất đồng bộ, truyền tải OrderCreated Event.
* **Payment Service**:
  + Lắng nghe và nhận OrderCreated Event từ Message Broker.
  + Xử lý giao dịch thanh toán (gọi đến cổng thanh toán bên ngoài).
  + Sau khi xử lý, xuất bản PaymentSucceeded hoặc PaymentFailed Event lên Message Broker.
  + Lưu trữ chi tiết giao dịch vào Database riêng.
* **Order Service (tiếp tục)**:
  + Lắng nghe và nhận PaymentSucceeded/Failed Event từ Message Broker.
  + **Nếu PaymentSucceeded**: Cập nhật trạng thái đơn hàng thành CONFIRMED. Gửi yêu cầu Deduct Stock (trừ kho vĩnh viễn) đến Inventory Service.
  + **Nếu PaymentFailed**: Cập nhật trạng thái đơn hàng thành FAILED. Gửi yêu cầu Revert Stock (hoàn trả tồn kho đã giữ) đến Inventory Service.
  + Cập nhật trạng thái đơn hàng vào Database riêng.
* **Inventory Service**: Nhận yêu cầu Deduct Stock hoặc Revert Stock từ Order Service và cập nhật currentStock, reservedStock trong Database.

#### 

#### **4. Luồng Vận hành & Giao hàng (Delivery & Monitoring) - (Khối màu tím)**

* **Order Service**: Sau khi đơn hàng được CONFIRMED, nó xuất bản OrderConfirmed Event (kèm shopId, deliveryAddress) lên Message Broker.
* **Delivery Dispatcher Service**:
  + Lắng nghe và nhận OrderConfirmed Event từ Message Broker.
  + Gọi Seller Service để Get Shop Location (lấy vị trí của cửa hàng).
  + Xử lý logic điều phối: tìm kiếm drone/shipper phù hợp.
  + Gửi yêu cầu giao hàng đến Delivery & GPS Service.
  + Lưu trữ thông tin điều phối vào Database riêng.
* **Delivery & GPS Service**:
  + Nhận yêu cầu giao hàng.
  + Lập lộ trình, quản lý quá trình giao hàng.
  + Thực hiện Real-time GPS tracking (theo dõi GPS thời gian thực) và **đẩy trực tiếp (push)** thông tin vị trí & ETA (Estimated Time of Arrival) đến Client App của người dùng qua WebSocket/SSE.
  + Lưu trữ dữ liệu chuyến đi drone vào Database riêng.

#### **Các Thành phần Chung**

* **Monitoring & Logging Service**:
  + Tất cả các Microservices (User Service, Seller Service, Product Service, v.v.) đều được cấu hình để Send Logs/Metrics kèm theo Correlation ID (một ID duy nhất cho mỗi yêu cầu người dùng đi qua hệ thống).
  + Service này thu thập, phân tích các log và metrics.
* **Admin Dashboard**: Truy vấn dữ liệu từ Monitoring & Logging Service để hiển thị Alerts & Reports về hiệu suất, lỗi, và các chỉ số kinh doanh.

#### **Future considerations (Định hướng Tương lai):**

1. **Promotion Service:** Xây dựng một microservice riêng để quản lý các chương trình khuyến mãi, mã giảm giá, và voucher.
2. **Rating & Review Service:** Cho phép người dùng đánh giá nhà hàng và món ăn, cung cấp dữ liệu đầu vào cho hệ thống gợi ý.
3. **Recommendation System:** Áp dụng Machine Learning để cá nhân hóa và gợi ý món ăn/nhà hàng phù hợp với sở thích của người dùng.
4. **Data Analytics Dashboard:** Cung cấp cho Nhà bán hàng các báo cáo phân tích về doanh thu, sản phẩm bán chạy, và hành vi khách hàng.

#### **Launch Plan (Kế hoạch Ra mắt)**

#### **Open Questions (Các câu hỏi mở)**

* Chiến lược chiêu mộ và quản lý đội ngũ tài xế/drone sẽ được thực hiện như thế nào?
* Mô hình kinh doanh chi tiết là gì? (Tỷ lệ hoa hồng trên mỗi đơn hàng, phí nền tảng, v.v.)
* Quy trình xử lý tranh chấp/khiếu nại giữa Người mua, Nhà bán hàng và Nền tảng sẽ ra sao?
* Hệ quản trị cơ sở dữ liệu cụ thể nào sẽ được lựa chọn (MongoDB hay PostgreSQL) và lý do chi tiết cho từng service?

#### **FAQs (Câu hỏi thường gặp)**

* **Tại sao lại chọn kiến trúc Microservices?**
  + Để tăng khả năng mở rộng độc lập cho từng nghiệp vụ, giảm thiểu ảnh hưởng khi một service gặp sự cố, và cho phép các đội ngũ phát triển song song.
* **Làm thế nào để đảm bảo dữ liệu nhất quán giữa các service?**
  + Bằng cách sử dụng mô hình giao dịch Saga và hệ thống Message Broker, đảm bảo rằng một chuỗi các giao dịch hoặc sẽ thành công toàn bộ, hoặc sẽ được hoàn tác (compensate) hoàn toàn nếu có lỗi xảy ra.
* **Hệ thống sẽ hỗ trợ những cổng thanh toán nào?**
  + Hệ thống được thiết kế để dễ dàng tích hợp với nhiều nhà cung cấp. Giai đoạn đầu sẽ tập trung vào các ví điện tử phổ biến như Momo, ZaloPay và thanh toán qua thẻ ngân hàng.