Chào bạn, ý tưởng dự án **Smart Restaurant Ordering System** tích hợp IoT của bạn rất hay và đầy đủ tính thử thách cho một Backend Developer. Việc sử dụng kiến trúc **Microservices, Kafka và IoT** cho bài toán này là một cách tuyệt vời để "flex" kỹ năng trong CV (dù với quy mô quán nhỏ thì Monolith là đủ, nhưng để học/showcase thì Microservices là lựa chọn chuẩn).

Dưới đây là bản thiết kế chi tiết cho hệ thống của bạn:

---

## 1. Kiến trúc Tổng quan (High-Level Architecture)

Hệ thống sẽ được chia thành các Services nhỏ, giao tiếp với nhau chủ yếu qua **Kafka** (bất đồng bộ) và **gRPC/REST** (đồng bộ).

- **Frontend (Next.js):**
  - **Client App:** Giao diện cho khách/nhân viên order (Mobile-first).
  - **Admin Dashboard:** Giao diện quản lý, thống kê (Desktop view).
- **Backend (NestJS Microservices):**
  1. **API Gateway:** Cổng vào duy nhất, điều hướng request.
  2. **Auth Service:** Quản lý User, Staff, Admin, JWT.
  3. **Product Service:** Quản lý Menu, Danh mục, Topping.
  4. **Order Service:** Xử lý đặt món, tính tổng tiền, trạng thái đơn.
  5. **Payment Service:** Xử lý thanh toán (Momo/VNPay giả lập), lưu lịch sử giao dịch.
  6. **Reporting Service:** Lắng nghe sự kiện đơn hàng để tổng hợp báo cáo doanh thu.
  7. **IoT/Notification Service:** Service quan trọng nhất kết nối với ESP32.
- **Message Broker (Kafka):** Trái tim của hệ thống, xử lý luồng sự kiện (Event-driven).
- **IoT Device (ESP32):** Nhận lệnh từ IoT Service -> Phát loa.

---

## 2. Danh sách Chức năng (Functional Requirements)

### A. Phân hệ Khách hàng / Nhân viên (Order App)

1. **Quét QR Code:** Xác định số bàn (Table ID) thông qua URL.
2. **Xem Menu:** Hiển thị danh sách món ăn, hình ảnh, giá, tình trạng (còn/hết).
3. **Đặt món (Order):** Chọn món, thêm ghi chú (ít đường, không hành), chọn số lượng.
4. **Giỏ hàng & Xác nhận:** Xem lại các món đã chọn, tổng tiền tạm tính.
5. **Gửi đơn xuống bếp:** Tạo Order.
6. **Theo dõi trạng thái:** Chờ xác nhận -> Đang nấu -> Đã ra món.
7. **Thanh toán:** Gọi thanh toán hoặc thanh toán online (tích hợp Payment Service).

### B. Phân hệ Quản trị (Admin Dashboard)

1. **Quản lý Menu:** CRUD món ăn, danh mục, giá.
2. **Quản lý Bàn:** Setup sơ đồ bàn, tạo mã QR.
3. **Quản lý Đơn hàng (Kitchen View):** Màn hình cho bếp xem danh sách món cần làm.
4. **Thống kê (Reporting):**

- Doanh thu theo ngày/tháng.
- Món bán chạy nhất (Best seller).
- Khung giờ cao điểm.

### C. Phân hệ IoT (Kitchen Alert)

1. **Thông báo đơn mới:** Khi có Order mới -> Loa phát: *"Bàn số 5 có đơn mới"*.
2. **Thông báo thanh toán:** Khi khách bấm gọi thanh toán -> Loa phát: *"Bàn số 5 yêu cầu thanh toán"*.

---

## 3. Use Case & Luồng xử lý với Kafka

Dưới đây là luồng đi quan trọng nhất: **"Khách đặt món -> Bếp nhận thông báo"**.

1. **User** bấm "Đặt món" trên Next.js.
2. **API Gateway** chuyển request tới **Order Service**.
3. **Order Service**:
   - Lưu đơn hàng vào DB (Status: PENDING).
   - Tính tổng tiền.
   - **Produce (Gửi)** một message vào Kafka topic: order.created.
   - Trả về response "Đặt hàng thành công" cho User ngay lập tức (Non-blocking).
4. **IoT Service** (Consumer):
   - Lắng nghe topic order.created.
   - Nhận payload (Ví dụ: { table: 5, items: ["Cà phê", "Trà sữa"], total: 50k }).
   - Gọi Google TTS API (Text-to-Speech) để chuyển text *"Bàn 5 có đơn mới"* thành file MP3 (hoặc URL).
   - Gửi lệnh xuống **ESP32** qua **MQTT** hoặc **WebSocket**.
5. **ESP32**:
   - Nhận URL file âm thanh hoặc tín hiệu.
   - Stream và phát ra loa.

---

## 4. Thiết kế Cơ sở dữ liệu (Database Schema - PostgreSQL)

Bạn có thể dùng chung 1 DB (chia schema) hoặc mỗi service 1 DB riêng (chuẩn Microservices). Dưới đây là thiết kế Logic (ERD):

### Bảng users (Auth Service)

- id (PK), username, password_hash, role (ADMIN, STAFF, CUSTOMER), created_at.

### Bảng categories (Product Service)

- id (PK), name, image_url.

### Bảng products (Product Service)

- id (PK), category_id (FK), name, description, price, image_url, is_available.

### Bảng tables (Order Service)

- id (PK), name (Bàn 1), qr_code, status (Trống/Có khách).

### Bảng orders (Order Service)

- id (PK), user_id (FK - nullable nếu khách vãng lai), table_id (FK), total_amount, status (PENDING, CONFIRMED, COMPLETED, CANCELLED), created_at.

### Bảng order_items (Order Service)

- id (PK), order_id (FK), product_id, product_name (Lưu cứng tên đề phòng đổi tên), quantity, price (Lưu cứng giá tại thời điểm mua), note.

### Bảng payments (Payment Service)

- id (PK), order_id (FK), amount, method (CASH, MOMO), status, transaction_id.

---

## 5. Chi tiết kỹ thuật IoT (ESP32 + Loa)

Để ESP32 phát ra tiếng nói (Text-to-Speech) từ thông báo Backend, bạn cần làm theo hướng sau:

**Phần cứng:**

- ESP32 (WROOM-32).
- Module DAC âm thanh: **MAX98357A** (I2S Amplifier) - Loại này rẻ và chất lượng âm thanh tốt hơn DAC nội của ESP32.
- Loa 3W 4Ohm.

**Luồng dữ liệu (Data Flow):**

1. **NestJS (IoT Service):**
    - Nhận Kafka Event.
    - Tạo câu thông báo: "Có đơn mới tại bàn 10".
    - Dùng thư viện google-tts-api (Node.js) để lấy URL file MP3 của câu nói đó (Google Server trả về).
    - Publish một message tới MQTT Topic kitchen/alert với payload: { "url": "https://translate.google.com/..." }.
2. **ESP32 (Arduino C++):**
    - Kết nối WiFi & MQTT Broker.
    - Subscribe topic kitchen/alert.
    - Khi nhận được URL: Sử dụng thư viện **ESP8266Audio** (Hỗ trợ cả ESP32) để stream

file MP3 từ URL đó và phát ra loa qua giao tiếp I2S.

## 6. Cấu trúc Project (Monorepo recommend)

Sử dụng **Nx** hoặc **TurboRepo** để quản lý:

Plaintext

```
my-restaurant-app/
├── apps/
│   ├── web-client/       (Next.js - Khách order)
│   ├── web-admin/        (Next.js - Quản lý)
│   ├── api-gateway/      (NestJS)
│   ├── auth-service/     (NestJS)
│   ├── order-service/    (NestJS)
│   ├── product-service/  (NestJS)
│   ├── iot-service/      (NestJS - Xử lý Kafka & MQTT)
├── libs/
│   ├── shared-dto/       (Type chia sẻ giữa các service)
│   ├── database/         (Cấu hình TypeORM/Prisma chung)
├── docker-compose.yml    (Chạy Postgres, Kafka, Zookeeper, Redis)
```

## 7. Lời khuyên triển khai

- **Bước 1:** Dựng Docker Compose chạy Kafka và Postgres trước.
- **Bước 2:** Viết **Product Service** và **Order Service** đơn giản (CRUD).
- **Bước 3:** Tích hợp Kafka: Khi tạo Order -> Bắn event.
- **Bước 4:** Viết **IoT Service**: Consum event -> Log ra console trước.
- **Bước 5:** Code ESP32 kết nối MQTT. Test chức năng phát nhạc từ URL.
- **Bước 6:** Nối IoT Service với ESP32.