Análisis y Propuesta de Arquitectura para SmartKet ERP (Versión 2)

Hola de nuevo. Tu aclaración sobre las diferencias entre pollería, farmacia y otros negocios es fundamental. Confirma que la arquitectura modular es la única vía sostenible para tu proyecto.

Tu visión de negocio es correcta:

1. **ERP Base:** Funcionalidad común (Ventas, Inventario, Caja).
2. **Módulos Especializados (Verticales):** Funcionalidad específica por rubro (ej: farmacia_lotes).
3. **Módulos Adicionales (Add-ons):** Funcionalidad opcional con costo extra (ej: app_meseros, app_cocina).

La buena noticia es que la arquitectura que discutimos (Landing, API, App) no solo soporta esto, sino que es la forma ideal de implementarlo.

1. La Base de Datos (Multi-Tenancy Aislada)

Este pilar se mantiene sin cambios:

- **smartket_admin_db (Base Pública):** Guarda usuarios, *tenants* (empresas) y, lo más importante, **qué módulos ha contratado cada *tenant***.
- **db_cliente_xyz (Base Privada):** Guarda los datos *operativos* de ese cliente (sus productos, sus ventas, etc.).

2. La Estructura del Proyecto (Modular, no Separada)

Aquí está la clave para manejar tus "pollerías" vs "farmacias" sin volverte loco. Mantienes los 3 proyectos únicos:

- **smartket-landing**: El portal de marketing y registro.
- **smartket-api (El Cerebro - Laravel)**
- **smartket-app (El ERP - React/Vue/Angular)**

Cómo manejar los Módulos de Pollería vs. Farmacia

NO creas Frontend-Polleria y Frontend-Farmacia.

SÍ creas **MÓDULOS** dentro de tu smartket-app y smartket-api.

Tu arquitectura de software (código) se vería así:

smartket-api/
├── app/

```
│   ├── Http/
│   │   ├── Controllers/
│   │   │   ├── Core/
│   │   │   │   ├── ProductoController.php
│   │   │   │   ├── VentaController.php
│   │   │   ├── Modules/
│   │   │   │   ├── Farmacia/
│   │   │   │   │   └── LoteController.php
│   │   │   │   ├── Polleria/
│   │   │   │   │   ├── PedidoCocinaController.php
│   │   │   │   │   └── MesaController.php
```

```
smartket-app/
├── src/
│   ├── components/
│   │   ├── Core/
│   │   │   ├── TablaProductos.vue
│   │   │   ├── FormularioVenta.vue
│   │   ├── Modules/
│   │   │   ├── Farmacia/
│   │   │   │   └── GestionLotes.vue
│   │   │   ├── Polleria/
│   │   │   │   ├── VistaMesas.vue
│   │   │   │   ├── VistaCocina.vue
```

Ventaja: Si encuentras un bug en VentaController.php, lo arreglas *una vez* y beneficia a todos (pollerías, farmacias, minimarkets).

3. El Flujo de Trabajo: Tu Visión de Negocio Implementada

Este es el flujo exacto que te permite vender módulos adicionales:

Paso 1: Registro (en smartket-landing)

- El usuario "Juan" se registra.
- Elige su rubro: "Pollería".
- Tu API (smartket-api) es llamada.
- La API crea la BBDD db_juan_polleria_123.
- En la smartket_admin_db, guarda:
  - tenant_id: 123
  - usuario: 'juan'

    - rubro: 'polleria'
    - modulos_activos: ['CORE_ERP', 'MODULO_POLLERIA_BASICO']
    - plan: 'GRATUITO'

Paso 2: Uso del ERP (en smartket-app)

- Juan se loguea en smartket-app.
- La app pide a la API el perfil de Juan.
- La API responde con { ..., modulos_activos: ['CORE_ERP', 'MODULO_POLLERIA_BASICO'] }.
- El frontend (smartket-app) lee esto y **habilita los botones y vistas** de "Ventas", "Inventario" y "Gestión de Mesas (Básico)".
- El botón para "App de Meseros" o "Vista de Cocina" aparece, pero **deshabilitado o con un ícono de "Premium"**.

Paso 3: Venta del Módulo Adicional (¡Tu Costo Adicional!)

- Juan hace clic en el botón "Habilitar App de Meseros".
- smartket-app le muestra una ventana de pago (o un "contactar a ventas").
- Juan paga.
- Tu API de pagos (Stripe, etc.) llama a un endpoint en tu smartket-api.
- Tu API **actualiza la smartket_admin_db**:
    - tenant_id: 123
    - modulos_activos: ['CORE_ERP', 'MODULO_POLLERIA_BASICO', 'ADDON_APP_MESEROS', 'ADDON_VISTA_COCINA']
    - plan: 'POLLERIA_PRO'

Paso 4: Activación Instantánea

- Juan refresca la página (o la app lo detecta en tiempo real).
- El frontend vuelve a pedir el perfil.
- La API ahora responde: { ..., modulos_activos: [..., 'ADDON_APP_MESEROS', ...] }.
- El frontend lee la nueva lista y **automáticamente habilita el botón** de "App de Meseros".
- Ahora Juan puede usar esa sección.

¿Cómo funcionaría la "App de Mesero" o "Vista de Cocina"?

No son proyectos separados. Son simplemente **VISTAS DIFERENTES** de tu única smartket-app.

- **App Mesero:** El mesero entraría a app.smartket.com/mesero en su celular. Esta URL cargaría una vista de la smartket-app súper simplificada, que solo muestra las mesas y la toma de pedidos, llamando a los mismos *endpoints* de la API (MesaController.php, etc.).
- **Vista Cocina:** En la cocina tendrían una tablet en app.smartket.com/cocina. Esta vista solo mostraría los PedidoCocinaController.php y se refrescaría automáticamente.

Conclusión:
Tu visión es perfecta. La forma de implementarla profesionalmente es con una Arquitectura

Multi-Tenant Aislada (BBDD por cliente) y una Aplicación Monolítica Modular (un solo frontend, un solo backend) que activa o desactiva funcionalidades basándose en la suscripción del cliente, la cual está guardada en tu base de datos admin.
Esto te da la facilidad de mantenimiento de un solo código base, y la flexibilidad de negocio para vender paquetes y módulos especializados.