Plan de Acción: Implementando SmartKet

Este documento es tu guía paso a paso para construir la arquitectura que diseñamos. Pasaremos de la teoría a la práctica.

Tu sistema se compone de 3 proyectos, pero deben construirse en un orden lógico.

1. `smartket-api` (El Cerebro en Laravel)
2. `smartket-app` (El ERP en React/Vue)
3. `smartket-landing` (El Marketing)

Fase 0: El Núcleo Multi-Tenant (Tu Pregunta de la BBDD)

No puedes construir *nada* (ni productos, ni ventas) hasta que tu sistema sepa cómo manejar múltiples bases de datos. Esto es lo primero.

Objetivo: Hacer que Laravel pueda conectarse a la BBDD de un cliente dinámicamente.

Pasos a Seguir (en `smartket-api`):

1. **Crea la Base de Datos "Admin":** Crea tu base de datos principal `smartket_admin_db`. Aquí vivirán las tablas que *gestionan* a tus clientes, no sus datos de negocio.
   - `tenants`: (id, nombre_negocio, rubro, plan, **nombre_db_cliente**, **usuario_db_cliente**, **pass_db_cliente**)
   - `users`: (id, email, password) - *El dueño global, Juan.*
   - `tenant_user`: (user_id, tenant_id) - *Tabla pivote para saber que Juan es dueño de la Pollería X.*
2. **Instala un Paquete de Multi-Tenancy:** No reinventes la rueda. Usa un paquete profesional para esto. El más recomendado para Laravel es:
   - `spatie/laravel-multitenancy`
   - Este paquete es mágico. Detectará automáticamente qué `tenant` está haciendo la petición (basado en el usuario logueado, por ejemplo) y **cambiará la conexión de la base de datos por ti** en tiempo de ejecución.
3. **Define tu BBDD "Plantilla" de Cliente:** Crea un segundo *schema* de base de datos (ej. `smartket_tenant_template_db`) que contenga todas las tablas que un cliente *nuevo* necesita:
   - `products`
   - `categories`
   - `sales`
   - `sale_details`
   - `cash_registers`
   - `staff` (para los sub-usuarios como meseros, cajeros)
   - `roles` y `permissions` (para esos staff)

4. **Crea el "Script de Provisión":** Este es el corazón de tu "registro fácil". Crea un Comando de Laravel (`php artisan make:command CreateTenant`) que haga lo siguiente:
    1. Reciba el email/pass/nombre de negocio del nuevo cliente.
    2. Cree un nuevo usuario en la tabla `users` de `smartket_admin_db` .
    3. Cree un nuevo registro en la tabla `tenants` de `smartket_admin_db` .
    4. **Ejecute los comandos SQL:**
        - `CREATE DATABASE db_cliente_123;`
        - `CREATE USER user_cliente_123 IDENTIFIED BY '...';`
        - `GRANT ALL PRIVILEGES ON db_cliente_123.* TO user_cliente_123;`
    5. **Clone la plantilla:** Use `mysqldump` para copiar todas las tablas de `smartket_tenant_template_db` a la nueva `db_cliente_123` .
    6. Guarde `db_cliente_123` y `user_cliente_123` en el registro del `tenant` (paso 3).

Resultado de la Fase 0: Tienes un sistema que puede crear un cliente nuevo y aislado en 10 segundos con un solo comando.

Fase 1: El MVP de Pollería (El Producto Real)

Ahora que tienes el "contenedor" (Fase 0), construye el "contenido".

Objetivo: Que tu primer cliente (la pollería) pueda hacer una venta.

Pasos a Seguir (en `smartket-api` y `smartket-app`):

1. **Crea los CRUDs Básicos:** Olvídate de módulos PRO. Enfócate en el flujo principal.
    - API: `ProductoController.php` (CRUD de productos).
    - API: `VentaController.php` (Crear Venta, Anular Venta).
    - API: `CajaController.php` (Abrir Caja, Cerrar Caja).
    - APP: Crea las vistas (en React/Vue) para `Gestión de Productos` y la `Interfaz de Punto de Venta (POS)` .
2. **Prueba Manual:** "Loguéate" como tu cliente de pollería. ¿Puedes crear un "1/4 de Pollo"? ¿Puedes venderlo? ¿Se registra en la tabla `sales` de `db_cliente_123` ?
3. **Impresión de Tickets:** Resuelve el problema de la impresora térmica. Investiga cómo enviar comandos de impresión (ej. usando `javascript-print-raw` o similar) desde tu `smartket-app` a la impresora local del cliente.

Resultado de la Fase 1: Tu producto funciona. Tienes algo que vender y que tu primer cliente puede *usar*.

Fase 2: El Onboarding (El Flujo de "Free Trial")

Ahora une el `smartket-landing` con tu API.

1. **Crea el `smartket-landing` :** Construye la página de marketing.
2. **Crea el Formulario de Registro:** El formulario de "Email, Pass, Nombre de Negocio".

3. **Conecta el Formulario:** Haz que ese formulario llame a un *endpoint* de tu API (`POST /api/register`) que ejecute el **Comando de Provisión** de la Fase 0.
4. **Crea el Setup Wizard:** En `smartket-app` , crea un *middleware* o *guard*. Si detecta que el `tenant` es nuevo (ej. `setup_complete = false`), lo redirige forzosamente al wizard de 4 pasos (Datos Fiscales, Sucursal, Caja, Productos).

Resultado de la Fase 2: Has automatizado tu visión. Un extraño puede llegar a tu web, registrarse y empezar a usar el sistema sin hablar contigo.

Fase 3: Módulos PRO (Roles y Add-ons)

Ahora que el MVP funciona, construye los add-ons de pago.

1. **Roles y Permisos:**
   - Instala `spatie/laravel-permission` en tu plantilla de tenant (Fase 0).
   - Crea la interfaz en `smartket-app` para que el "Dueño" (Usuario Principal) pueda crear los sub-usuarios (Mesero, Cajero, Cocina) y asignarles roles.
2. **Vistas por Rol:**
   - Crea las rutas y vistas especiales en `smartket-app` : `/mesero` , `/cocina` .
   - Añade lógica: "Si el usuario logueado tiene el rol `MESERO` , redirigir a `/mesero` ".
3. **Bloqueo de Módulos:**
   - En tu `smartket_admin_db` (Fase 0), añade una columna a la tabla `tenants` : `modulos_activos` (un campo JSON).
   - Tu API ahora debe revisar: "El cliente 123 está pidiendo la data de `/cocina` . ¿Su plan incluye el módulo `ADDON_COCINA` ? Si no, devolver error 402 (Payment Required)".