📘 FastOps V5 - Documentación Maestra de API & Integración Frontend

Versión del Documento: 1.0 (Final) **Estado del Backend:** Producción / Staging **Arquitectura:** RESTful API + WebSockets (Async) **Autenticación:** JWT (JSON Web Tokens) con soporte Multi-Tenant

1. Configuración del Cliente HTTP (Axios)

1.1 Entornos

- **Desarrollo Local:** `http://localhost:8000/api/v1`
- **WebSocket Local:** `ws://localhost:8000/ws/v1`
- **Producción:** `https://api.tudominio.com/api/v1`

1.2 Cabeceras Obligatorias (Headers)

Todas las peticiones (excepto Login y Registro Público) requieren:

```
{
  "Content-Type": "application/json",
  "Authorization": "Bearer <tu_token_jwt_aqui>"
}
```

> **Nota de Arquitectura:** El `company_id` se extrae automáticamente del Token en el backend. No es necesario enviarlo en el cuerpo de la petición, pero sí es vital que el usuario tenga un `active_branch_id` (sucursal activa) seleccionada en el Frontend si la operación es específica de una sucursal.

2. Diccionario de Datos (Enums & Tipos)

Estos valores deben estar "harcodeados" o definidos como constantes en el Frontend para coincidir con la validación del Backend.

2.1 Estados de Pedido (`OrderStatus`)

Valor	Descripción	Color UI Sugerido
`PENDING`	Recibido, esperando confirmación	Gris

`CONFIRMED`	Confirmado, en cola de cocina	Azul
`PREPARING`	En preparación (Cocinando)	Naranja
`READY`	Listo para entrega/despacho	Verde Claro
`DELIVERED`	Entregado al cliente (Final)	Verde Oscuro
`CANCELLED`	Anulado	Rojo

2.2 Tipos de Pedido (`OrderType`)

Valor	Descripción
`DINE_IN`	Comer en mesa
`TAKE_AWAY`	Para llevar (Recoger)
`DELIVERY`	Domicilio (Requiere dirección)

2.3 Métodos de Pago (`PaymentMethod`)

Valor	Descripción
`CASH`	Efectivo
`CARD`	Tarjeta Débito/Crédito
`TRANSFER`	Transferencia (Nequi/Daviplata/Zelle)

3. Catálogo de Endpoints (Rutas)

👤 Módulo: Autenticación (`/auth`)

1. Iniciar Sesión (Staff/Admin)

- **POST** `/auth/login/access-token`
- **Content-Type:** `application/x-www-form-urlencoded` (Ojo: Form Data, no JSON)
- **Body:**
  - `username` : Email del usuario

- `password` : Contraseña

- **Respuesta:**

```
{
  "access_token": "ey...",
  "token_type": "bearer",
  "user": { "id": 1, "email": "...", "role": "admin", "company_id": 1 }
}
```

2. Perfil de Usuario (Check Auth)

- **GET** `/auth/me`
- **Descripción:** Usar al recargar la página para verificar si el token sigue vivo y traer datos del usuario.

🍔 Módulo: Productos y Menú (`/products` , `/categories`)

1. Listar Categorías

- **GET** `/categories/`
- **Query Params:** `?skip=0&limit=100`
- **Respuesta:** Array de objetos `{ id, name, is_active }` .

2. Listar Productos (Para el POS)

- **GET** `/products/`
- **Query Params:** `?category_id=5` (Opcional), `?search=hamburguesa`
- **Respuesta:**

```
[
  {
    "id": 10,
    "name": "Hamburguesa Doble",
    "price": 15000,
    "image_url": "...",
    "tax_rate": 0.0,
    "is_active": true
  }
]
```

3. Crear Producto (Admin)

- **POST** `/products/`
- **Body:**

```
{
  "name": "Perro Caliente Especial",
  "description": "Con todo",
  "price": 12000,
  "category_id": 2,
  "cost": 5000,
  "stock_control_enabled": true
}
```

⚡ Módulo: POS y Pedidos (`/orders`) - CRÍTICO

1. Crear Pedido (La venta)

- **POST** `/orders/`
- **Body Completo:**

```
{
  "customer_id": "uuid-o-null-si-es-anonimo",
  "branch_id": 1,
  "order_type": "DINE_IN",
  "table_number": 5,  // Opcional
  "items": [
    {
      "product_id": 10,
      "quantity": 2,
      "unit_price": 15000, // Frontend envía precio, Backend valida
      "notes": "Sin cebolla",
      "modifiers": [] // Futuro: Adiciones
    }
  ],
  "discount_amount": 0,
  "tax_amount": 0,
  "total_amount": 30000
}
```

2. Tablero de Pedidos Activos (Cocina/Monitor)

- **GET** `/orders/active`
- **Descripción:** Retorna solo pedidos que NO están `DELIVERED` o `CANCELLED` .

3. Actualizar Estado (KDS - Cocina)

- **PUT** `/orders/{order_id}/status`
- **Body:**

```
{
  "status": "READY"
}
```

💰 Módulo: Caja y Turnos (`/cash`)

1. Verificar Estado de Caja

- **GET** `/cash/status`
- **Respuesta:**

```
{
  "is_open": true,
  "current_shift_id": 123,
  "opened_at": "2024-01-20T10:00:00"
}
```

> *Lógica Frontend:* Si `is_open` es `false` , bloquear la pantalla de POS y mostrar modal "Abrir Caja".

2. Abrir Caja

- **POST** `/cash/open`
- **Body:** `{ "initial_amount": 200000 }` (Base)

3. Cerrar Caja

- **POST** `/cash/close`
- **Body:**

```
{
  "final_amount_counted": 1500000, // Lo que contó el cajero
  "notes": "Todo cuadrado"
}
```

💳 Módulo: Pagos (`/payments`)

1. Registrar Pago (Cobrar Pedido)

- **POST** `/payments/`
- **Body:**

```
{
  "order_id": 55,
  "amount": 30000,
  "payment_method": "CASH",
  "reference": "" // Opcional para vouchers de tarjeta
}
```

> **Nota:** Al recibir `200 OK`, el Frontend debe marcar el pedido como pagado y ofrecer imprimir ticket.

📦 Módulo: Inventario y Recetas (`/inventory`, `/recipes`)

1. Listar Insumos

- **GET** `/inventory/items`
- **Respuesta:** Lista de ingredientes (Pan, Queso, Gaseosa).

2. Movimiento Manual (Ajuste/Compra)

- **POST** `/inventory/transactions`
- **Body:**

```
{
  "inventory_item_id": 5,
  "type": "IN", // IN (Compra) o OUT (Merma)
  "quantity": 50,
  "reason": "Compra semanal"
}
```

3. Crear Receta (Vincular Producto -> Insumos)

- **POST** `/recipes/`
- **Body:**

```
{
  "product_id": 10, // Hamburguesa
  "ingredients": [
    { "inventory_item_id": 1, "quantity": 1 }, // 1 Pan
```

```
        { "inventory_item_id": 2, "quantity": 0.150 } // 150g Carne
      ]
    }
```

🛵 Módulo: Domicilios (`/delivery`)

1. Pedidos Pendientes de Asignar

- **GET** `/delivery/pending`
- **Descripción:** Pedidos `READY` de tipo `DELIVERY` sin conductor.

2. Lista de Domiciliarios

- **GET** `/delivery/drivers`
- **Respuesta:** Lista de usuarios con rol `DRIVER` y su estado (`AVAILABLE` , `BUSY`).

3. Asignar Domicilio

- **POST** `/delivery/assign`
- **Body:** `{ "order_id": 55, "driver_id": 3 }`

4. WebSockets (Tiempo Real)

URL de Conexión: `ws://<host>/ws/v1/connect?token=<jwt>`

Flujo de Conexión Frontend

1. Frontend conecta al socket al iniciar la App.
2. Frontend se une a la sala de su sucursal enviando un evento (o el backend lo hace automático según el token).

Eventos a Escuchar (`socket.on(...)`)

Evento	Payload	Acción en Frontend
`new_order`	Objeto `Order` completo	**KDS/POS:** Reproducir sonido "Ding", agregar tarjeta al tablero Kanban en columna "Pendiente".
`order_updated`	`{ order_id, status }`	**Monitor:** Mover la tarjeta de columna (ej. de Cocina a Listo).
`stock_alert`	`{ item_name, current_stock }`	**Admin/POS:** Mostrar Toast de alerta "Queso bajo en stock".

`print_ticket`	`{ order_id, content }`	**POS:** Disparar orden a la impresora térmica local.

5. Reportes (`/reports`)

1. Ventas Diarias

- **GET** `/reports/sales/daily`
- **Respuesta:** Totales vendidos hoy, desglose por método de pago.

2. Top Productos

- **GET** `/reports/products/top`
- **Respuesta:** Lista de productos más vendidos para gráficos de torta.

6. Manejo de Errores

El Backend sigue el estándar HTTP. El Frontend debe interceptar estos códigos:

- `400 Bad Request` : Error de validación (ej. falta stock). Mostrar mensaje `detail` .
- `401 Unauthorized` : Token vencido o inválido. **Acción:** Redirigir a Login inmediatamente.
- `403 Forbidden` : Usuario logueado pero sin permisos (ej. Cajero intentando crear usuario). Mostrar "Acceso Denegado".
- `422 Unprocessable Entity` : Error de tipos de datos (ej. enviar texto en campo precio). Revisar consola.
- `500 Internal Server Error` : Error grave del servidor. Mostrar "Error de sistema, contacte soporte".

📜 Resumen de Integración para el Desarrollador

1. **Crea el archivo** `api.ts` **:** Configura Axios con la URL base.
2. **Crea** `types.ts` **:** Copia las interfaces de los JSONs de arriba.
3. **Implementa Auth:** Login guarda JWT en LocalStorage.
4. **Implementa Interceptor:** Axios inyecta el token en cada request.