📋 Documentação Completa - APIs iFood para Dashboard Analytics

📖 Índice

🎯 Visão Geral do Sistema

O dashboard **Plano Certo Hub Insights** integra com **6 APIs principais** do iFood para fornecer análises completas para restaurantes:

APIs Integradas:

- 🔐 **Authentication** - Gerenciamento de tokens OAuth2
- 🏪 **Merchant** - Informações dos restaurantes
- 📖 **Catalog** - Produtos e cardápio
- 💰 **Financial** - Dados financeiros e conciliação
- 🎁 **Promotion** - Campanhas promocionais
- 🔔 **Webhook** - Notificações em tempo real

🔐 1. AUTHENTICATION API

🎯 Propósito no Dashboard:

Gerenciar autenticação OAuth2 para conectar com segurança aos dados do iFood, permitindo acesso autorizado aos recursos dos restaurantes.

🔧 Endpoints Principais:

```
POST /oauth/token      # Obter access token
POST /oauth/refresh    # Renovar access token
```

📊 Dados Coletados:

Campo	Tipo	Descrição	Uso no Dashboard
`access_token`	string	Token de acesso (até 8000 chars)	Autenticação de todas as requisições
`refresh_token`	string	Token para renovação	Manter sessão ativa
`expires_in`	number	Tempo de expiração (segundos)	Controle de renovação automática
`token_type`	string	Tipo do token ("Bearer")	Header de autorização

💻 Implementação no Dashboard:

```
// hooks/useIfoodConfig.ts
interface IfoodAuthConfig {
  clientId: string;
  clientSecret: string;
  accessToken: string;
  refreshToken: string;
  expiresAt: Date;
  isActive: boolean;
}

// Tabela: ifood_tokens
const tokenSchema = {
  access_token: 'string',
  client_id: 'string',
  client_secret: 'string',
  expires_at: 'number',
  user_id: 'string'
}
```

🔄 Fluxos Implementados:

1. **Client Credentials Flow** - Aplicações centralizadas
2. **Authorization Code Flow** - Aplicações web públicas
3. **Auto-refresh** - Renovação automática antes da expiração

---

🏪 2. MERCHANT API

🎯 Propósito no Dashboard:

Gerenciar informações dos restaurantes, controlar status operacional, horários e interrupções. Base para sistema multi-tenant.

🔧 Endpoints Principais:

```
GET /merchants                          # Listar restaurantes
GET /merchants/{merchantId}             # Detalhes do restaurante
GET /merchants/{merchantId}/status      # Status operacional
```

```
GET /merchants/{merchantId}/opening-hours # Horários de funcionamento
PUT /merchants/{merchantId}/opening-hours # Atualizar horários
GET /merchants/{merchantId}/interruptions # Listar interrupções
POST /merchants/{merchantId}/interruptions # Criar interrupção
```

📊 Dados Coletados:

Campo	Tipo	Descrição	Uso no Dashboard
`merchant_id`	string	ID único do restaurante	Identificação principal
`name`	string	Nome fantasia	Exibição nos filtros
`corporate_name`	string	Razão social	Relatórios oficiais
`status`	boolean	Status operacional	Monitoramento em tempo real
`address_*`	string	Endereço completo	Localização e relatórios
`operating_hours`	object	Horários por dia da semana	Análise de disponibilidade
`delivery_methods`	array	Métodos de entrega	Configuração operacional
`payment_methods`	array	Formas de pagamento	Análise de preferências
`minimum_order_value`	number	Valor mínimo do pedido	KPIs operacionais
`delivery_fee`	number	Taxa de entrega	Análise de custos

💻 Implementação no Dashboard:

```
// hooks/useIfoodMerchants.ts
interface IfoodMerchant {
  merchant_id: string;
  name: string;
  corporate_name: string | null;
  status: boolean | null;
  address_city: string | null;
  address_state: string | null;
  phone: string | null;
  operating_hours: OperatingHours | null;
  delivery_methods: string[] | null;
  payment_methods: string[] | null;
  user_id: string;
  client_id: string | null;
  last_sync_at: string;
}

// Tabela: ifood_merchants
// Deduplicação automática por merchant_id
// Polling: atualização a cada 30 segundos
// Relacionamento: user_id -> clients.id
```

🎨 Módulos que Utilizam:

- **StoreMonitoring** - Status e operação em tempo real
- **IntegrationStatusCard** - Verificação de conectividade
- **ClientManagement** - Associação restaurante-cliente
- **Header** - Seletor de restaurantes (filtros)

📖 3. CATALOG API V2

🎯 Propósito no Dashboard:

Gerenciar cardápio completo com produtos, categorias e complementos. Permite análise de performance de produtos e otimização de menu.

🔧 Endpoints Principais:

```
GET /catalogs                           # Listar catálogos
GET /catalogs/{catalogId}               # Detalhes do catálogo
GET /catalogs/{catalogId}/categories    # Categorias do catálogo
GET /catalogs/{catalogId}/items         # Items do catálogo
POST /catalogs/{catalogId}/items        # Criar item completo
PUT /catalogs/{catalogId}/items/{itemId} # Atualizar item
GET /products                           # Listar produtos
POST /products                          # Criar produto
```

📊 Dados Coletados:

Campo	Tipo	Descrição	Uso no Dashboard
`product_id`	string	ID único do produto	Identificação
`name`	string	Nome do produto	Exibição e busca
`description`	string	Descrição detalhada	Análise de conteúdo
`price`	number	Preço atual	Análise de precificação
`category`	string	Categoria do produto	Organização e filtros
`image_path`	string	URL da imagem	Exibição visual
`external_code`	string	Código PDV	Integração com sistemas
`is_active`	boolean	Status de disponibilidade	Controle de ativação
`dietary_restrictions`	array	Restrições dietárias	Classificação nutricional
`availability_hours`	object	Horários de disponibilidade	Análise temporal
`ean`	string	Código de barras	Controle de estoque

💻 Implementação no Dashboard:

```
// hooks/useProducts.ts
interface ProductData {
  id: string;
  name: string;
  description: string | null;
  price: number | null;
  client_id: string;
  ifood_product_id: string | null;
  is_active: string | null;
  merchant_id: string | null;
  item_id: string | null;
  imagePath: string | null;
  created_at: string;
  updated_at: string;
}

// Tabela: products
// Relacionamento: client_id -> clients.id
// Índices: name, category, merchant_id, is_active
```

🔄 Funcionalidades Avançadas:

- **Multi-Setup** - Diferentes preços por catálogo
- **Pizza Handling** - Produtos com características especiais
- **Complement Groups** - Grupos de complementos
- **External Code Reuse** - Reutilização de produtos existentes

🎨 Módulos que Utilizam:

- **MenuOptimization** - Análise de performance de produtos
- **MenuManagement** - Gestão completa do cardápio
- **ProductAnalysis** - Análise detalhada por produto
- **Dashboard** - KPIs de produtos mais vendidos

💰 4. FINANCIAL API V2

🎯 Propósito no Dashboard:

Fornecer dados financeiros detalhados para conciliação, análise de receita e repasses. Núcleo do dashboard de analytics financeiro.

🔧 Endpoints Principais:

```
GET /financial/v2.1/merchants/{merchantId}/sales           # Vendas detalhadas
GET /financial/v2/merchants/{merchantId}/salesAdjustments  # Ajustes de vendas
GET /financial/v2/merchants/{merchantId}/occurrences       # Ocorrências
financeiras
GET /financial/v2/merchants/{merchantId}/chargeCancellations # Cancelamentos
```

```
GET /financial/v2/merchants/{merchantId}/maintenanceFees   # Taxas de manutenção
GET /financial/v2/merchants/{merchantId}/salesBenefits     # Benefícios de vendas
GET /financial/v2/merchants/{merchantId}/adjustmentsBenefits # Benefícios de
ajustes
GET /financial/v2/merchants/{merchantId}/incomeTaxes       # Impostos de renda
```

📊 Dados Coletados:

Campo	Tipo	Descrição	Uso no Dashboard
order_number	string	Número único do pedido	Rastreabilidade
date	string	Data da transação	Análises temporais
gross_revenue	number	Receita bruta	KPI principal
net_value	number	Valor líquido a receber	Fluxo de caixa
items_value	number	Valor dos itens	Análise de produtos
delivery_fee	number	Taxa de entrega	Custos operacionais
ifood_commission_value	number	Comissão do iFood	Análise de custos
service_fee	number	Taxa de serviço	Custos adicionais
ifood_promotions	number	Promoções iFood	Desconto externo
store_promotions	number	Promoções da loja	Desconto interno
weekly_plan_fee	number	Taxa do plano semanal	Custo fixo
payment_method	string	Método de pagamento	Análise de preferências
billing_type	string	Tipo de faturamento	Classificação contábil

💻 Implementação no Dashboard:

```
// hooks/useFinancialMetrics.ts
interface FinancialMetric {
  id: string;
  client_id: string;
  date: string;
  revenue: number | null;
  net_revenue: number | null;
  commission: number | null;
  delivery_fee: number | null;
  orders_count: number | null;
  average_ticket: number | null;
  source: string | null; // 'ifood', 'manual', etc.
}

// Tabelas principais:
// - financial_metrics (métricas agregadas diárias)
```

```
// - ifood_detailed_analytics (dados detalhados por pedido)

// Processamento:
// - ifoodProcessor.ts: Processa dados específicos do iFood
// - genericProcessor.ts: Processa outros formatos
// - aggregationUtils.ts: Agrega dados por período
```

🔄 Atualizações de Dados:

- **Frequência**: Diária às 18h
- **Períodos**: Fechados toda quarta-feira
- **Processamento**: Upsert por (client_id, date)
- **Validação**: Mapeamento de colunas dinâmico

🎨 Módulos que Utilizam:

- **Dashboard** - KPIs principais e gráficos
- **IfoodAnalytics** - Análises específicas do iFood
- **IfoodAdvancedAnalytics** - Análises detalhadas por pedido
- **ReportsModule** - Relatórios personalizados
- **RevenueChart** - Gráfico de receita temporal

🎁 5. PROMOTION API

🎯 Propósito no Dashboard:

Gerenciar campanhas promocionais e ofertas especiais. Permite análise de performance de promoções e otimização de conversões.

🔧 Endpoints Principais:

```
POST /merchants/{merchantId}/promotions                    # Criar promoções
POST /merchants/{merchantId}/promotions?reset=true        # Resetar todas
promoções
GET /merchants/{merchantId}/promotions/{aggregationId}/items # Status das
promoções
```

📊 Dados Coletados:

Campo	Tipo	Descrição	Uso no Dashboard
`aggregation_id`	string	ID de rastreamento da promoção	Identificação única
`promotion_type`	string	Tipo (PERCENTAGE, FIXED, FIXED_PRICE)	Classificação
`discount_value`	number	Valor do desconto	Cálculo de impacto
`status`	string	Status (ACTIVE, INACTIVE, PENDING)	Monitoramento

Campo	Tipo	Descrição	Uso no Dashboard
`items`	array	Produtos incluídos na promoção	Análise por produto
`valid_from`	string	Data de início	Período de validade
`valid_until`	string	Data de término	Período de validade
`minimum_quantity`	number	Quantidade mínima	Condições de ativação
`minimum_value`	number	Valor mínimo	Condições de ativação

💻 Implementação no Dashboard:

```
// Possível hook usePromotions.ts (a ser implementado)
interface PromotionData {
  aggregation_id: string;
  promotion_type: 'PERCENTAGE' | 'FIXED' | 'FIXED_PRICE' |
'PERCENTAGE_PER_X_UNITS';
  discount_value: number;
  status: 'ACTIVE' | 'INACTIVE' | 'PENDING';
  items: PromotionItem[];
  created_at: string;
  performance_metrics: {
    impressions: number;
    clicks: number;
    conversions: number;
    revenue_impact: number;
  }
}

// Tabela a ser criada: promotions
// Relacionamento: client_id -> clients.id
// Processamento: Assíncrono (pode levar minutos)
```

⚠️ Características Especiais:

- **Processamento Assíncrono** - Criação pode levar minutos
- **Status 202** - Não garante criação bem-sucedida
- **Validação Final** - Verificar status via GET
- **Exclusivo Marketplace** - Apenas para parceiros específicos

🎨 Módulos que Utilizam:

- **MenuOptimization** - Sugestões de promoções
- **SalesFunnelAnalysis** - Impacto nas conversões
- **Dashboard** - KPIs de promoções ativas

🔔 6. WEBHOOK/NOTIFICATIONS API

🎯 Propósito no Dashboard:

Receber notificações em tempo real sobre eventos do iFood, eliminando necessidade de polling constante e permitindo reações imediatas.

🔧 Configuração:

```
# Configuração no painel do desenvolvedor
Webhook URL: https://seu-app.com/webhooks/ifood
Método: POST
Content-Type: application/json
Timeout: 5 segundos
Response esperada: 202 Accepted
```

📊 Eventos Recebidos:

Evento	Descrição	Payload	Uso no Dashboard
MERCHANT_STATUS_CHANGED	Status da loja mudou	Novo status	Monitoramento de disponibilidade
CATALOG_UPDATED	Cardápio atualizado	Mudanças no catálogo	Sincronização de produtos
PROMOTION_ACTIVATED	Promoção ativada	Dados da campanha	Análise promocional
FINANCIAL_SETTLEMENT	Liquidação financeira	Dados de repasse	Conciliação financeira

💻 Implementação no Dashboard:

```
// api/webhooks/ifood.ts
interface WebhookEvent {
  event_type: string;
  merchant_id: string;
  order_id?: string;
  timestamp: string;
  data: any;
  signature: string; // X-IFood-Signature header
}

// Processamento:
// 1. Validar signature
// 2. Processar evento
// 3. Atualizar banco de dados
// 4. Notificar dashboard via WebSocket
// 5. Retornar 202 Accepted
```

🔒 Segurança:

- **Signature Verification** - Header `X-IFood-Signature`
- **HTTPS Obrigatório** - TLS 1.2+
- **Timeout 5s** - Resposta rápida obrigatória
- **Retry Logic** - Reenvio até 15 minutos

🎨 Módulos que Utilizam:

- **StoreMonitoring** - Atualizações em tempo real
- **Dashboard** - Notificações push
- **OrdersChart** - Atualizações dinâmicas
- **IntegrationStatusCard** - Status de conectividade

🔗 INTEGRAÇÃO NO DASHBOARD

📊 Fluxo de Dados Completo:

```
graph TD
    A[Authentication API] --> B[Token Válido]
    B --> C[Merchant API]
    B --> D[Financial API]
    B --> E[Catalog API]
    B --> F[Promotion API]
    B --> G[Webhook API]

    C --> H[Store Monitoring]
    D --> I[Financial Analytics]
    E --> J[Menu Optimization]
    F --> K[Promotion Analysis]
    G --> L[Real-time Updates]
```

🗄️ Estrutura do Banco de Dados:

```
-- Tabelas principais já implementadas
clients                ☑ (gestão de restaurantes)
financial_metrics      ☑ (métricas financeiras agregadas)
ifood_detailed_analytics ☑ (dados financeiros detalhados)
ifood_merchants        ☑ (dados dos merchants)
ifood_tokens           ☑ (tokens de autenticação)
products               ☑ (catálogo básico)

-- Tabelas a implementar
promotions             ❌ (campanhas promocionais)
webhook_events         ❌ (log de eventos recebidos)
```

🔄 Hooks Implementados vs Necessários:

```
// ☑ Já implementados
useIfoodConfig()        // Authentication
useIfoodMerchants()     // Merchant
useFinancialMetrics()   // Financial
useProducts()           // Catalog (básico)

// ❌ A implementar
usePromotions()         // Promotion API
useWebhookEvents()      // Webhook processing
```

🎯 IMPLEMENTAÇÃO POR MÓDULO DO CLIENTE

📋 MÓDULO 1: Integração iFood + Coleta de Dados

APIs Utilizadas:

- ☑ **Authentication** - Tokens e renovação
- ☑ **Merchant** - Lista de restaurantes
- ❌ **Financial** - Dados financeiros (a implementar)
- ❌ **Webhook** - Notificações (a implementar)

Implementação Atual:

```
// Componentes funcionando
<IfoodApiConfig />          // Configuração de tokens
<IntegrationStatusCard />   // Status da integração
<Dashboard />               // Visão geral (dados fictícios)
<ReportsModule />           // Relatórios PDF (dados fictícios)

// Processamento de dados
ifoodMerchantsService.ts    // Gerencia merchants
// ifoodProcessor.ts        // A implementar para dados financeiros reais
```

Gaps a Implementar:

- 🔘 **Financial API** - Integração completa com dados reais
- Export CSV nos relatórios
- Sistema de alertas configuráveis
- Webhook para atualizações em tempo real

📋 MÓDULO 2: Diagnóstico com IA

APIs Utilizadas:

- ❌ **Financial** - Dados para análise (a implementar)

- ☑ **Merchant** - Contexto operacional
- ❌ **Sistema de IA** - OpenAI/Claude (a implementar)

Dados Necessários para IA:

```
interface DiagnosticData {
  // Financial API (a implementar - contém dados de pedidos também)
  revenue_trends: FinancialMetric[];     // A implementar
  cost_analysis: CostBreakdown;          // A implementar
  order_patterns: OrderPattern[];        // A implementar via Financial API
  peak_hours: HourlyAnalysis[];          // A implementar via Financial API

  // Merchant API
  operational_status: OperationalMetrics;

  // Benchmarks (base de conhecimento)
  sector_benchmarks: BenchmarkData;
}
```

Implementação Necessária:

- Sistema de IA para análise de dados
- Base de conhecimento com metodologia Plano Certo
- Algoritmos de benchmark setorial
- Geração automática de planos de ação

📋 MÓDULO 3: Otimização de Cardápio com IA

APIs Utilizadas:

- ❌ **Catalog** - Produtos e descrições (implementar completamente)
- ❌ **Promotion** - Campanhas promocionais (a implementar)
- ❌ **Financial** - Performance de produtos (a implementar)
- ❌ **Sistema de IA** - Geração de conteúdo (a implementar)

Implementação Necessária:

```
// hooks/useCatalogAnalysis.ts
interface ProductPerformance {
  product_id: string;
  sales_count: number;
  revenue: number;
  profit_margin: number;
  click_through_rate: number;
  conversion_rate: number;
  seasonal_trends: SeasonalData[];
}
```

```
// services/aiContentGenerator.ts
interface AIServices {
  generateDescription: (product: Product) => Promise<string>;
  optimizeImage: (image: File) => Promise<File>;
  suggestPricing: (product: Product, market: MarketData) => number;
  recommendPromotions: (products: Product[]) => PromotionSuggestion[];
}
```

📋 MÓDULO 4: Automação de Cobrança e Relatórios

APIs Utilizadas:

- ❌ **Financial** - Dados para cobrança (a implementar)
- ❌ **WhatsApp Business API** - Envio de cobranças (externa)
- ❌ **Email API** - Relatórios automáticos (externa)
- ❌ **Payment Gateway** - Controle de pagamentos (externa)

Implementação Necessária:

```
// services/billingAutomation.ts
interface BillingService {
  generateInvoice: (client: Client, period: DateRange) => Invoice;
  sendWhatsAppBilling: (invoice: Invoice) => Promise<boolean>;
  sendEmailReport: (report: Report) => Promise<boolean>;
  trackPaymentStatus: (invoice: Invoice) => PaymentStatus;
}

// Integrações externas
WhatsAppBusinessAPI      // Meta ou Twilio
EmailService            // Resend ou SendGrid
PaymentGateway          // Stripe, PagSeguro, MercadoPago
```

🚀 ROADMAP DE IMPLEMENTAÇÃO

FASE 1: Implementar Financial API e Completar Módulo 1 (6-8 semanas)

1. 🔘 **Implementar Financial API completa**
2. ☑ Configurar Webhook para tempo real
3. ☑ Sistema de alertas configuráveis
4. ☑ Export CSV completo
5. ☑ Testes e homologação

FASE 2: Módulo 3 - Cardápio IA (10-12 semanas)

1. ☑ Catalog API V2 completa

2. ☑ Promotion API completa
3. ☑ Sistema de IA para geração de conteúdo
4. ☑ Análise de performance de produtos (dep. Financial API)
5. ☑ Recomendações automatizadas

FASE 3: Módulo 4 - Automação (8-10 semanas)

1. ☑ WhatsApp Business API
2. ☑ Sistema de cobrança automatizada (dep. Financial API)
3. ☑ Relatórios automáticos por email
4. ☑ Integração com gateway de pagamento
5. ☑ Dashboard de cobrança e inadimplência

FASE 4: Módulo 2 - IA Diagnóstico (12-14 semanas)

1. ☑ Treinar IA com metodologia Plano Certo
2. ☑ Sistema de análise automatizada (dep. Financial API)
3. ☑ Base de benchmarks setoriais
4. ☑ Geração de planos de ação personalizados
5. ☑ Interface de recomendações

📈 MÉTRICAS DE SUCESSO

KPIs por API:

- **Authentication**: Uptime > 99.5%, renovação automática de tokens
- **Merchant**: Sincronização < 30s, cobertura de 100% dos merchants
- **Financial**: Conciliação > 99.9%, dados atualizados em 24h
- **Catalog**: Sincronização completa de produtos em < 5min
- **Promotion**: Criação de promoções em < 2min
- **Order**: Latência < 1s, captura de 100% dos eventos
- **Webhook**: Disponibilidade > 99.9%, processamento < 5s

Performance do Dashboard:

- Tempo de carregamento < 3s
- Atualizações em tempo real < 1s
- Exportação de relatórios < 10s
- Disponibilidade geral > 99.5%

🔧 CONFIGURAÇÕES TÉCNICAS

Rate Limits:

- Financial API: 10 req/s
- Catalog API: 15 req/s
- Promotion API: 5 req/s
- Merchant API: 10 req/s

Timeouts:

- API Requests: 30s
- Webhook Response: 5s
- Token Refresh: 10s

Segurança:

- HTTPS/TLS 1.2+ obrigatório
- OAuth2 Bearer tokens
- Webhook signature verification
- Rate limiting por IP/client

Documento criado em: Janeiro 2025
Última atualização: Janeiro 2025
Mantido por: Equipe de Desenvolvimento Plano Certo Hub Insights