Documentação de Integração

iFood Logistics API + App de Entregas

1. Visão Geral

Esta documentação descreve como integrar a API de Logistics do iFood com seu aplicativo de entregas. O objetivo é criar automaticamente uma entrega no seu app quando um pedido no iFood mudar para os status **READY_TO_PICKUP** (pronto para coleta) ou **DISPATCHED** (saiu para entrega).

1.1 Fluxo da Integração

1. Realizar polling a cada 30 segundos no endpoint de eventos do iFood
2. Filtrar eventos por status: READY_TO_PICKUP ou DISPATCHED
3. Buscar detalhes completos do pedido via GET /orders/{orderId}
4. Extrair dados do cliente (nome, telefone, endereço)
5. Criar a entrega no seu aplicativo
6. Enviar acknowledgment dos eventos processados

2. Autenticação

Para acessar a API do iFood, é necessário obter um token de acesso OAuth2.

2.1 Endpoint de Autenticação

```
POST https://merchant-api.ifood.com.br/authentication/v1.0/oauth/token
```

2.2 Parâmetros do Request

Parâmetro	Valor
`grantType`	client_credentials
`clientId`	Seu Client ID (obtido no portal developer)
`clientSecret`	Seu Client Secret (obtido no portal developer)

Importante: Renove o token apenas quando estiver prestes a expirar.

3. Recebendo Eventos de Pedidos

O iFood utiliza uma arquitetura orientada a eventos. Você pode receber eventos via **Polling** ou **Webhook**.

3.1 Polling de Eventos

```
GET https://merchant-api.ifood.com.br/events/v1.0/events:polling
```

Headers obrigatórios:

- `Authorization:` Bearer {access_token}
- `x-polling-merchants:` {merchantId}

Parâmetros opcionais de query:

- `types=RTP,DSP` - Filtra apenas eventos de status específicos
- `groups=ORDER_STATUS` - Filtra por grupo de eventos
- `excludeHeartbeat=true` - Obrigatório para integradoras logísticas

3.2 Eventos de Status Relevantes

Os eventos que você deve monitorar para criar entregas:

Código	fullCode	Descrição
RTP	READY_TO_PICKUP	Pedido pronto para coleta. Ideal para alocar entregador.
DSP	DISPATCHED	Pedido saiu para entrega. Entregador já está a caminho.

3.3 Estrutura do Evento

```
{   "id": "b03392c5-61dd-47c4-a503-bce3109c96c8",   "code": "RTP",   "fullCode":
"READY_TO_PICKUP",   "orderId": "93ba4bf4-f4ae-4de8-8017-35d7c7de9bf1",
"merchantId": "820af392-002c-47b1-bfae-d7ef31743c99",   "createdAt": "2021-02-
17T19:36:55.295Z",   "salesChannel": "IFOOD" }
```

3.4 Acknowledgment dos Eventos

Importante: Após processar os eventos, você DEVE enviar o acknowledgment para evitar recebê-los novamente.

```
POST https://merchant-api.ifood.com.br/events/v1.0/events/acknowledgment
```

```
[   { "id": "b03392c5-61dd-47c4-a503-bce3109c96c8" },   { "id": "outro-event-id-aqui"
} ]
```

4. Obtendo Detalhes do Pedido

Após receber um evento, busque os detalhes completos do pedido para extrair as informações do cliente.

4.1 Endpoint de Detalhes

```
GET https://merchant-api.ifood.com.br/order/v1.0/orders/{orderId}
```

4.2 Dados do Cliente

Os dados do cliente estão no objeto **`customer`**:

```
"customer": {   "id": "22587f70-60b4-423c-8cd2-27d288f47f99",   "name": "Nome do
Cliente",   "documentNumber": "12345678900",   "ordersCountOnMerchant": 8,   "phone":
{     "number": "0800 XXX XXXX",     "localizer": "27534642",
"localizerExpiration": "2024-01-09T00:44:42.547Z"   } }
```

Campo	Descrição
`customer.name`	Nome completo do cliente
`customer.phone.number`	Telefone mascarado do cliente (0800)
`customer.phone.localizer`	Código localizador para identificar a ligação

Nota: O telefone é mascarado por privacidade. O campo `localizer` expira 3 horas após a entrega.

4.3 Endereço de Entrega

O endereço está no objeto **`delivery.deliveryAddress`**:

```
"deliveryAddress": {   "streetName": "Rua Exemplo",   "streetNumber": "1234",
"formattedAddress": "Rua Exemplo, 1234, Apto 101",   "neighborhood": "Centro",
"complement": "Apto 101",   "reference": "Próximo à praça",   "postalCode":
"12345678",   "city": "São Paulo",   "state": "SP",   "country": "BR",
"coordinates": {     "latitude": -23.550520,     "longitude": -46.633308   } }
```

Campo	Descrição
`streetName`	Nome da rua/avenida
`streetNumber`	Número do endereço
`formattedAddress`	Endereço completo formatado
`neighborhood`	Bairro
`complement`	Complemento (apto, bloco, etc)
`reference`	Ponto de referência
`postalCode`	CEP (sem formatação)

`city / state`	Cidade e Estado
`coordinates`	Latitude e longitude para GPS

5. Exemplo de Implementação (Node.js)

5.1 Classe de Integração

```
class IFoodIntegration {   constructor(clientId, clientSecret) {     this.clientId = clientId;
this.clientSecret = clientSecret;     this.baseUrl = 'https://merchant-api.ifood.com.br';
this.accessToken = null;   }    async authenticate() {     const response = await fetch(
`${this.baseUrl}/authentication/v1.0/oauth/token`, {       method: 'POST',       headers: {
'Content-Type': 'application/x-www-form-urlencoded' },       body: new URLSearchParams({
grantType: 'client_credentials',         clientId: this.clientId,         clientSecret:
this.clientSecret       })     });     const data = await response.json();     this.accessToken =
data.accessToken;     return data;   }    async pollEvents(merchantId) {     const url =
`${this.baseUrl}/events/v1.0/events:polling`;     const response = await fetch(url, {
headers: {         'Authorization': `Bearer ${this.accessToken}`,         'x-polling-merchants':
merchantId       }     });     if (response.status === 204) return []; // Sem eventos     return
response.json();   }    async getOrderDetails(orderId) {     const response = await fetch(
`${this.baseUrl}/order/v1.0/orders/${orderId}`, {       headers: { 'Authorization': `Bearer
${this.accessToken}` }     });     return response.json();   }    async
acknowledgeEvents(eventIds) {     const body = eventIds.map(id => ({ id }));     await
fetch(`${this.baseUrl}/events/v1.0/events/acknowledgment`, {       method: 'POST',       headers:
{         'Authorization': `Bearer ${this.accessToken}`,         'Content-Type':
'application/json'       },       body: JSON.stringify(body)     });   } }
```

5.2 Processamento de Eventos

```
async function processIfoodEvents(ifood, merchantId) {   const events = await
ifood.pollEvents(merchantId);   const eventsToAck = [];      for (const event of events) {
eventsToAck.push(event.id);          // Filtrar apenas eventos de interesse     if
(event.fullCode === 'READY TO PICKUP' ||          event.fullCode === 'DISPATCHED') {
const order = await ifood.getOrderDetails(event.orderId);              // Extrair dados para
criar entrega       const deliveryData = {          ifoodOrderId: order.id,          displayId:
order.displayId,          customer: {           name: order.customer.name,           phone:
order.customer.phone.number,           phoneLocalizer: order.customer.phone.localizer         },
address: {           street: order.delivery.deliveryAddress.streetName,           number:
order.delivery.deliveryAddress.streetNumber,           complement:
order.delivery.deliveryAddress.complement,           neighborhood:
order.delivery.deliveryAddress.neighborhood,           city: order.delivery.deliveryAddress.city,
state: order.delivery.deliveryAddress.state,           postalCode:
order.delivery.deliveryAddress.postalCode,           reference:
order.delivery.deliveryAddress.reference,           latitude:
order.delivery.deliveryAddress.coordinates.latitude,           longitude:
order.delivery.deliveryAddress.coordinates.longitude,           formattedAddress:
order.delivery.deliveryAddress.formattedAddress         },         status: event.fullCode,
createdAt: event.createdAt       };              // Criar entrega no seu app       await
createDeliveryInYourApp(deliveryData);     }   }      // Confirmar todos os eventos processados
if (eventsToAck.length > 0) {     await ifood.acknowledgeEvents(eventsToAck);   } }
```

5.3 Loop Principal de Polling

```
// Executar polling a cada 30 segundos async function startPolling() {   const ifood = new
IFoodIntegration(     process.env.IFOOD_CLIENT_ID,     process.env.IFOOD_CLIENT_SECRET   );
await ifood.authenticate();      setInterval(async () => {     try {       await
processIfoodEvents(ifood, process.env.MERCHANT ID);     } catch (error) {
console.error('Erro no polling:', error);       // Reautenticar se token expirou       if
(error.status === 401) {         await ifood.authenticate();       }     }   }, 30000); // 30
segundos }  startPolling();
```

6. Boas Práticas e Regras

- **Polling a cada 30 segundos:** Manter essa frequência para não perder pedidos e manter o merchant online
- **Sempre enviar acknowledgment:** Confirme todos os eventos, mesmo os que você não utiliza
- **Tratar eventos duplicados:** A API pode entregar o mesmo evento mais de uma vez
- **Ordenar por createdAt:** Eventos podem chegar fora de ordem
- **Rate Limit:** 20 requisições por segundo por endpoint
- **Privacidade:** Não imprima CPF e endereço em documentos destinados a entregadores
- **Token de acesso:** Renove apenas quando próximo de expirar

7. Resumo dos Endpoints

Método	Finalidade	Endpoint
POST	Autenticação	`/authentication/v1.0/oauth/token`
GET	Polling eventos	`/events/v1.0/events:polling`
POST	Acknowledgment	`/events/v1.0/events/acknowledgment`
GET	Detalhes pedido	`/order/v1.0/orders/{orderId}`

Base URL: `https://merchant-api.ifood.com.br`

Documentação oficial: https://developer.ifood.com.br