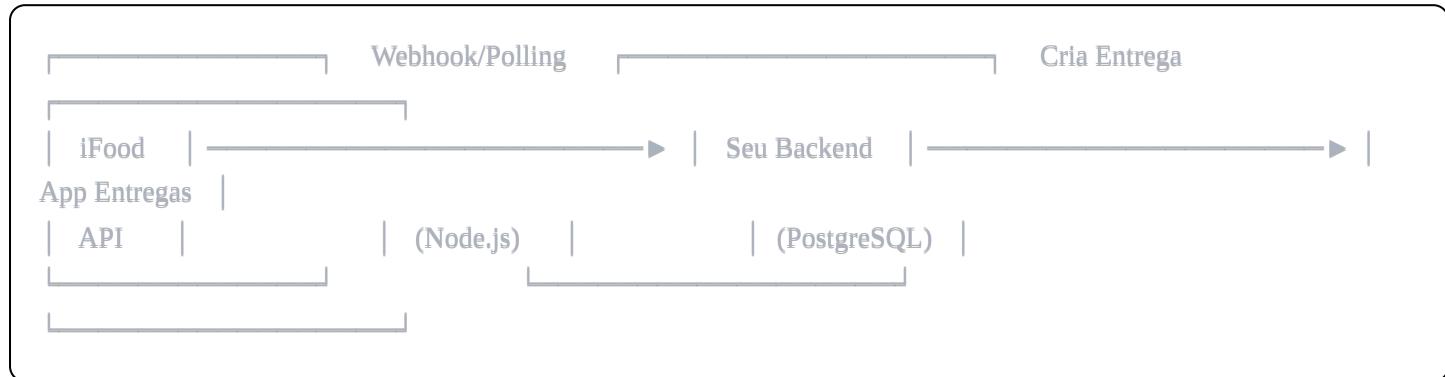


# Integração iFood → App de Entregas

## Visão Geral

Quando um pedido no iFood mudar para status **READY\_TO\_PICKUP** (pronto) ou **DISPATCHED** (saiu para entrega), uma nova entrega será criada automaticamente no seu app.

## Arquitetura da Solução



## Fluxo de Status dos Pedidos iFood

Status	Código	Descrição	Ação no seu App
PLACED	PLC	Pedido criado	-
CONFIRMED	CFM	Pedido confirmado	-
READY_TO_PICKUP	RTP	Pedido pronto para coleta	<b>Criar entrega</b>
DISPATCHED	DSP	Pedido saiu para entrega	<b>Criar entrega</b> (alternativa)
CONCLUDED	CON	Pedido concluído	-
CANCELLED	CAN	Pedido cancelado	Cancelar entrega se existir

## 1. Configuração Inicial no iFood

### 1.1 Cadastro no Portal Developer

1. Acesse: <https://developer.ifood.com.br>
2. Crie conta com **Perfil Profissional** (CNPJ obrigatório)
3. O portal cria automaticamente um **app de teste** com credenciais

## 1.2 Credenciais Necessárias

```
env
```

```
IFOOD_CLIENT_ID=seu_client_id  
IFOOD_CLIENT_SECRET=seu_client_secret  
IFOOD_MERCHANT_ID=id_da_loja
```

---

## 2. Autenticação

### 2.1 Obter Access Token

```
javascript
```

```

// src/services/ifood/auth.js
const axios = require('axios');

const IFOOD_BASE_URL = 'https://merchant-api.ifood.com.br';

let tokenCache = {
  accessToken: null,
  expiresAt: null
};

async function getAccessToken() {
  // Verifica se token ainda é válido
  if (tokenCache.accessToken && tokenCache.expiresAt > Date.now()) {
    return tokenCache.accessToken;
  }

  const response = await axios.post(
    `${IFOOD_BASE_URL}/authentication/v1.0/oauth/token`,
    new URLSearchParams({
      grantType: 'client_credentials',
      clientId: process.env.IFOOD_CLIENT_ID,
      clientSecret: process.env.IFOOD_CLIENT_SECRET
    }),
    {
      headers: {
        'Content-Type': 'application/x-www-form-urlencoded'
      }
    }
  );

  // Guarda token em cache
  tokenCache = {
    accessToken: response.data.accessToken,
    expiresAt: Date.now() + (response.data.expiresIn * 1000) - 60000 // 1min antes
  };

  return tokenCache.accessToken;
}

module.exports = { getAccessToken };

```

### 3. Recebendo Eventos (2 Opções)

#### 3.1 Opção A: Webhook (Recomendado)

O iFood envia eventos automaticamente para sua URL.

### **Configuração no Portal Developer:**

1. Vá em Meus Apps → Seu App → Webhook
2. Configure URL: <https://seu-dominio.com/webhooks/ifood>
3. Ative os eventos desejados

### **Endpoint no seu backend:**

javascript

```
// src/routes/webhooks/ifood.js
const express = require('express');
const crypto = require('crypto');
const router = express.Router();
const { processOrderEvent } = require('../services/ifood/orderProcessor');

// Middleware para validar assinatura do iFood
function validateIFoodSignature(req, res, next) {
  const signature = req.headers['x-ifood-signature'];
  const body = JSON.stringify(req.body);

  const expectedSignature = crypto
    .createHmac('sha256', process.env.IFOOD_CLIENT_SECRET)
    .update(body)
    .digest('hex');

  if (signature !== expectedSignature) {
    return res.status(401).json({ error: 'Invalid signature' });
  }

  next();
}

router.post('/webhooks/ifood',
  express.json({ verify: (req, res, buf) => { req.rawBody = buf; } }),
  validateIFoodSignature,
  async (req, res) => {
    try {
      const event = req.body;

      console.log(`Evento iFood recebido: ${event.fullCode}`, {
        orderId: event.orderId,
        merchantId: event.merchantId
      });

      // Processa o evento de forma assíncrona
      processOrderEvent(event).catch(err => {
        console.error('Erro ao processar evento:', err);
      });
    }
  }
);

// Responde imediatamente com 202 (obrigatório em até 5 segundos)
res.status(202).send();

} catch (error) {
  console.error('Erro no webhook:', error);
  res.status(500).json({ error: error.message });
}
```

```
    }
}
);

module.exports = router;
```

### 3.2 Opção B: Polling (Fallback)

Consulta periódica à API do iFood.

javascript

```
// src/services/ifood/polling.js
const axios = require('axios');
const { getAccessToken } = require('./auth');
const { processOrderEvent } = require('./orderProcessor');

const IFOOD_BASE_URL = 'https://merchant-api.ifood.com.br';

async function pollEvents() {
  try {
    const token = await getAccessToken();

    // Busca eventos pendentes
    const response = await axios.get(
      `${IFOOD_BASE_URL}/events/v1.0/events:polling`,
      {
        headers: { Authorization: `Bearer ${token}` },
        params: {
          excludeHeartbeat: true // Importante: evita manter loja aberta indevidamente
        }
      }
    );

    const events = response.data || [];
    const eventIds = [];

    for (const event of events) {
      console.log(`Evento recebido: ${event.fullCode}`, event.orderId);

      await processOrderEvent(event);
      eventIds.push(event.id);
    }

    // Confirma recebimento dos eventos (acknowledgment)
    if (eventIds.length > 0) {
      await axios.post(
        `${IFOOD_BASE_URL}/events/v1.0/events/acknowledgment`,
        eventIds,
        {
          headers: { Authorization: `Bearer ${token}` }
        }
      );
    }

  } catch (error) {
    console.error('Erro no polling:', error.message);
  }
}
```

```
}
```

```
// Executa a cada 30 segundos (obrigatório pelo iFood)
function startPolling() {
  setInterval(pollEvents, 30000);
  pollEvents(); // Executa imediatamente
}

module.exports = { startPolling, pollEvents };
```

---

## 4. Processador de Eventos

```
javascript
```

```

// src/services/ifood/orderProcessor.js
const { getOrderDetails } = require('./orderService');
const { createDelivery } = require('../delivery/deliveryService');

// Eventos que disparam criação de entrega
const TRIGGER_EVENTS = ['READY_TO_PICKUP', 'DISPATCHED'];

async function processOrderEvent(event) {
  const { fullCode, orderId, merchantId } = event;

  // Verifica se é um evento que dispara criação de entrega
  if (!TRIGGER_EVENTS.includes(fullCode)) {
    console.log(`Evento ${fullCode} ignorado - não é gatilho de entrega`);
    return;
  }

  // Verifica se já existe entrega para este pedido (evita duplicação)
  const existingDelivery = await findDeliveryByExternalId(orderId);
  if (existingDelivery) {
    console.log(`Entrega já existe para pedido ${orderId}`);
    return;
  }

  // Busca detalhes completos do pedido
  const orderDetails = await getOrderDetails(orderId);

  // Cria a entrega no seu app
  await createDeliveryFromIfFoodOrder(orderDetails, merchantId);
}

async function findDeliveryByExternalId(externalId) {
  const { pool } = require('../database');
  const result = await pool.query(
    'SELECT id FROM deliveries WHERE external_id = $1 AND external_source = $2',
    [externalId, 'ifood']
  );
  return result.rows[0];
}

module.exports = { processOrderEvent };

```

## 5. Buscar Detalhes do Pedido

javascript

```
// src/services/ifood/orderService.js
const axios = require('axios');
const { getAccessToken } = require('./auth');

const IFOOD_BASE_URL = 'https://merchant-api.ifood.com.br';

async function getOrderDetails(orderId) {
  const token = await getAccessToken();

  const response = await axios.get(
    `${IFOOD_BASE_URL}/order/v1.0/orders/${orderId}`,
    {
      headers: { Authorization: `Bearer ${token}` }
    }
  );

  return response.data;
}

module.exports = { getOrderDetails };
```

### Estrutura do pedido retornado:

javascript

```
{  
  "id": "63895716-37c3-4372-afd0-3240bfef708d",  
  "displayId": "XPTO",  
  "orderType": "DELIVERY",  
  "orderTiming": "IMMEDIATE", // ou SCHEDULED  
  "salesChannel": "IFOOD",  
  "createdAt": "2021-02-16T18:10:27Z",  
  
  "merchant": {  
    "id": "c54bb20a-bce0-4e38-bd4a-fe5f0a7b6b5a",  
    "name": "Nome do Restaurante"  
  },  
  
  "customer": {  
    "id": "22587f70-60b4-423c-8cd2-27d288f47f99",  
    "name": "Nome do Cliente",  
    "phone": {  
      "number": "0800 XXX XXXX",  
      "localizer": "27534642" // Código para ligar via iFood  
    }  
  },  
  
  "delivery": {  
    "deliveredBy": "IFOOD", // ou MERCHANT (entrega própria)  
    "deliveryDateTime": "2021-02-09T18:10:32Z",  
    "observations": "Deixar na portaria",  
    "deliveryAddress": {  
      "streetName": "Rua Exemplo",  
      "streetNumber": "1234",  
      "formattedAddress": "Rua Exemplo, 1234, Apto 101",  
      "neighborhood": "Centro",  
      "complement": "Apto 101",  
      "reference": "Perto da praça",  
      "postalCode": "12345678",  
      "city": "São Paulo",  
      "state": "SP",  
      "coordinates": {  
        "latitude": -23.550520,  
        "longitude": -46.633308  
      }  
    },  
    "pickupCode": "1234" // Código de retirada  
  },  
  
  "total": {  
    "subTotal": 5000, // Em centavos (R$ 50,00)  
  }  
}
```

```
"deliveryFee": 500,  
"benefits": 0,  
"orderAmount": 5500  
},  
  
"items": [  
{  
"name": "X-Burger",  
"quantity": 2,  
"unitPrice": 2500,  
"totalPrice": 5000  
}  
]  
}
```

## 6. Criar Entrega no seu App

javascript

```
// src/services/delivery/deliveryService.js
const { pool } = require('../database');

async function createDeliveryFromIFoodOrder(order, merchantId) {
  const client = await pool.connect();

  try {
    await client.query('BEGIN');

    // Busca a empresa vinculada ao merchant iFood
    const companyResult = await client.query(
      'SELECT id, name, address, lat, lng FROM companies WHERE ifood_merchant_id = $1',
      [merchantId]
    );

    if (companyResult.rows.length === 0) {
      throw new Error(`Empresa não encontrada para merchant iFood: ${merchantId}`);
    }

    const company = companyResult.rows[0];
    const delivery = order.delivery;
    const address = delivery.deliveryAddress;

    // Calcula valor da entrega (você pode ter sua própria lógica)
    const deliveryFee = await calculateDeliveryFee(
      company.lat,
      company.lng,
      address.coordinates.latitude,
      address.coordinates.longitude
    );

    // Cria a entrega
    const result = await client.query(`
      INSERT INTO deliveries (
        company_id,
        external_id,
        external_source,
        external_display_id,
        status,

        -- Origem (restaurante)
        origin_address,
        origin_lat,
        origin_lng,

        -- Destino (cliente)
        delivery_id,
        delivery_type,
        delivery_status,
        delivery_eta,
        delivery_distance,
        delivery_time
      ) VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11, $12, $13, $14, $15, $16, $17, $18, $19, $20, $21, $22, $23, $24, $25, $26, $27, $28, $29, $30, $31, $32, $33, $34, $35, $36, $37, $38, $39, $40, $41, $42, $43, $44, $45, $46, $47, $48, $49, $50, $51, $52, $53, $54, $55, $56, $57, $58, $59, $60, $61, $62, $63, $64, $65, $66, $67, $68, $69, $70, $71, $72, $73, $74, $75, $76, $77, $78, $79, $80, $81, $82, $83, $84, $85, $86, $87, $88, $89, $90, $91, $92, $93, $94, $95, $96, $97, $98, $99, $100, $101, $102, $103, $104, $105, $106, $107, $108, $109, $110, $111, $112, $113, $114, $115, $116, $117, $118, $119, $120, $121, $122, $123, $124, $125, $126, $127, $128, $129, $130, $131, $132, $133, $134, $135, $136, $137, $138, $139, $140, $141, $142, $143, $144, $145, $146, $147, $148, $149, $150, $151, $152, $153, $154, $155, $156, $157, $158, $159, $160, $161, $162, $163, $164, $165, $166, $167, $168, $169, $170, $171, $172, $173, $174, $175, $176, $177, $178, $179, $180, $181, $182, $183, $184, $185, $186, $187, $188, $189, $190, $191, $192, $193, $194, $195, $196, $197, $198, $199, $200, $201, $202, $203, $204, $205, $206, $207, $208, $209, $210, $211, $212, $213, $214, $215, $216, $217, $218, $219, $220, $221, $222, $223, $224, $225, $226, $227, $228, $229, $230, $231, $232, $233, $234, $235, $236, $237, $238, $239, $240, $241, $242, $243, $244, $245, $246, $247, $248, $249, $250, $251, $252, $253, $254, $255, $256, $257, $258, $259, $260, $261, $262, $263, $264, $265, $266, $267, $268, $269, $270, $271, $272, $273, $274, $275, $276, $277, $278, $279, $280, $281, $282, $283, $284, $285, $286, $287, $288, $289, $290, $291, $292, $293, $294, $295, $296, $297, $298, $299, $300, $301, $302, $303, $304, $305, $306, $307, $308, $309, $310, $311, $312, $313, $314, $315, $316, $317, $318, $319, $320, $321, $322, $323, $324, $325, $326, $327, $328, $329, $330, $331, $332, $333, $334, $335, $336, $337, $338, $339, $340, $341, $342, $343, $344, $345, $346, $347, $348, $349, $350, $351, $352, $353, $354, $355, $356, $357, $358, $359, $360, $361, $362, $363, $364, $365, $366, $367, $368, $369, $370, $371, $372, $373, $374, $375, $376, $377, $378, $379, $380, $381, $382, $383, $384, $385, $386, $387, $388, $389, $390, $391, $392, $393, $394, $395, $396, $397, $398, $399, $400, $401, $402, $403, $404, $405, $406, $407, $408, $409, $410, $411, $412, $413, $414, $415, $416, $417, $418, $419, $420, $421, $422, $423, $424, $425, $426, $427, $428, $429, $430, $431, $432, $433, $434, $435, $436, $437, $438, $439, $440, $441, $442, $443, $444, $445, $446, $447, $448, $449, $450, $451, $452, $453, $454, $455, $456, $457, $458, $459, $460, $461, $462, $463, $464, $465, $466, $467, $468, $469, $470, $471, $472, $473, $474, $475, $476, $477, $478, $479, $480, $481, $482, $483, $484, $485, $486, $487, $488, $489, $490, $491, $492, $493, $494, $495, $496, $497, $498, $499, $500, $501, $502, $503, $504, $505, $506, $507, $508, $509, $510, $511, $512, $513, $514, $515, $516, $517, $518, $519, $520, $521, $522, $523, $524, $525, $526, $527, $528, $529, $530, $531, $532, $533, $534, $535, $536, $537, $538, $539, $540, $541, $542, $543, $544, $545, $546, $547, $548, $549, $550, $551, $552, $553, $554, $555, $556, $557, $558, $559, $550, $551, $552, $553, $554, $555, $556, $557, $558, $559, $560, $561, $562, $563, $564, $565, $566, $567, $568, $569, $570, $571, $572, $573, $574, $575, $576, $577, $578, $579, $580, $581, $582, $583, $584, $585, $586, $587, $588, $589, $580, $581, $582, $583, $584, $585, $586, $587, $588, $589, $590, $591, $592, $593, $594, $595, $596, $597, $598, $599, $590, $591, $592, $593, $594, $595, $596, $597, $598, $599, $600, $601, $602, $603, $604, $605, $606, $607, $608, $609, $600, $601, $602, $603, $604, $605, $606, $607, $608, $609, $610, $611, $612, $613, $614, $615, $616, $617, $618, $619, $610, $611, $612, $613, $614, $615, $616, $617, $618, $619, $620, $621, $622, $623, $624, $625, $626, $627, $628, $629, $620, $621, $622, $623, $624, $625, $626, $627, $628, $629, $630, $631, $632, $633, $634, $635, $636, $637, $638, $639, $630, $631, $632, $633, $634, $635, $636, $637, $638, $639, $640, $641, $642, $643, $644, $645, $646, $647, $648, $649, $640, $641, $642, $643, $644, $645, $646, $647, $648, $649, $650, $651, $652, $653, $654, $655, $656, $657, $658, $659, $650, $651, $652, $653, $654, $655, $656, $657, $658, $659, $660, $661, $662, $663, $664, $665, $666, $667, $668, $669, $660, $661, $662, $663, $664, $665, $666, $667, $668, $669, $670, $671, $672, $673, $674, $675, $676, $677, $678, $679, $670, $671, $672, $673, $674, $675, $676, $677, $678, $679, $680, $681, $682, $683, $684, $685, $686, $687, $688, $689, $680, $681, $682, $683, $684, $685, $686, $687, $688, $689, $690, $691, $692, $693, $694, $695, $696, $697, $698, $699, $690, $691, $692, $693, $694, $695, $696, $697, $698, $699, $700, $701, $702, $703, $704, $705, $706, $707, $708, $709, $700, $701, $702, $703, $704, $705, $706, $707, $708, $709, $710, $711, $712, $713, $714, $715, $716, $717, $718, $719, $710, $711, $712, $713, $714, $715, $716, $717, $718, $719, $720, $721, $722, $723, $724, $725, $726, $727, $728, $729, $720, $721, $722, $723, $724, $725, $726, $727, $728, $729, $730, $731, $732, $733, $734, $735, $736, $737, $738, $739, $730, $731, $732, $733, $734, $735, $736, $737, $738, $739, $740, $741, $742, $743, $744, $745, $746, $747, $748, $749, $740, $741, $742, $743, $744, $745, $746, $747, $748, $749, $750, $751, $752, $753, $754, $755, $756, $757, $758, $759, $750, $751, $752, $753, $754, $755, $756, $757, $758, $759, $760, $761, $762, $763, $764, $765, $766, $767, $768, $769, $760, $761, $762, $763, $764, $765, $766, $767, $768, $769, $770, $771, $772, $773, $774, $775, $776, $777, $778, $779, $770, $771, $772, $773, $774, $775, $776, $777, $778, $779, $780, $781, $782, $783, $784, $785, $786, $787, $788, $789, $780, $781, $782, $783, $784, $785, $786, $787, $788, $789, $790, $791, $792, $793, $794, $795, $796, $797, $798, $799, $790, $791, $792, $793, $794, $795, $796, $797, $798, $799, $800, $801, $802, $803, $804, $805, $806, $807, $808, $809, $800, $801, $802, $803, $804, $805, $806, $807, $808, $809, $810, $811, $812, $813, $814, $815, $816, $817, $818, $819, $810, $811, $812, $813, $814, $815, $816, $817, $818, $819, $820, $821, $822, $823, $824, $825, $826, $827, $828, $829, $820, $821, $822, $823, $824, $825, $826, $827, $828, $829, $830, $831, $832, $833, $834, $835, $836, $837, $838, $839, $830, $831, $832, $833, $834, $835, $836, $837, $838, $839, $840, $841, $842, $843, $844, $845, $846, $847, $848, $849, $840, $841, $842, $843, $844, $845, $846, $847, $848, $849, $850, $851, $852, $853, $854, $855, $856, $857, $858, $859, $850, $851, $852, $853, $854, $855, $856, $857, $858, $859, $860, $861, $862, $863, $864, $865, $866, $867, $868, $869, $860, $861, $862, $863, $864, $865, $866, $867, $868, $869, $870, $871, $872, $873, $874, $875, $876, $877, $878, $879, $870, $871, $872, $873, $874, $875, $876, $877, $878, $879, $880, $881, $882, $883, $884, $885, $886, $887, $888, $889, $880, $881, $882, $883, $884, $885, $886, $887, $888, $889, $890, $891, $892, $893, $894, $895, $896, $897, $898, $899, $890, $891, $892, $893, $894, $895, $896, $897, $898, $899, $900, $901, $902, $903, $904, $905, $906, $907, $908, $909, $900, $901, $902, $903, $904, $905, $906, $907, $908, $909, $910, $911, $912, $913, $914, $915, $916, $917, $918, $919, $910, $911, $912, $913, $914, $915, $916, $917, $918, $919, $920, $921, $922, $923, $924, $925, $926, $927, $928, $929, $920, $921, $922, $923, $924, $925, $926, $927, $928, $929, $930, $931, $932, $933, $934, $935, $936, $937, $938, $939, $930, $931, $932, $933, $934, $935, $936, $937, $938, $939, $940, $941, $942, $943, $944, $945, $946, $947, $948, $949, $940, $941, $942, $943, $944, $945, $946, $947, $948, $949, $950, $951, $952, $953, $954, $955, $956, $957, $958, $959, $950, $951, $952, $953, $954, $955, $956, $957, $958, $959, $960, $961, $962, $963, $964, $965, $966, $967, $968, $969, $960, $961, $962, $963, $964, $965, $966, $967, $968, $969, $970, $971, $972, $973, $974, $975, $976, $977, $978, $979, $970, $971, $972, $973, $974, $975, $976, $977, $978, $979, $980, $981, $982, $983, $984, $985, $986, $987, $988, $989, $980, $981, $982, $983, $984, $985, $986, $987, $988, $989, $990, $991, $992, $993, $994, $995, $996, $997, $998, $999, $990, $991, $992, $993, $994, $995, $996, $997, $998, $999, $1000, $1001, $1002, $1003, $1004, $1005, $1006, $1007, $1008, $1009, $1000, $1001, $1002, $1003, $1004, $1005, $1006, $1007, $1008, $1009, $1010, $1011, $1012, $1013, $1014, $1015, $1016, $1017, $1018, $1019, $1010, $1011, $1012, $1013, $1014, $1015, $1016, $1017, $1018, $1019, $1020, $1021, $1022, $1023, $1024, $1025, $1026, $1027, $1028, $1029, $1020, $1021, $1022, $1023, $1024, $1025, $1026, $1027, $1028, $1029, $1030, $1031, $1032, $1033, $1034, $1035, $1036, $1037, $1038, $1039, $1030, $1031, $1032, $1033, $1034, $1035, $1036, $1037, $1038, $1039, $1040, $1041, $1042, $1043, $1044, $1045, $1046, $1047, $1048, $1049, $1040, $1041, $1042, $1043, $1044, $1045, $1046, $1047, $1048, $1049, $1050, $1051, $1052, $1053, $1054, $1055, $1056, $1057, $1058, $1059, $1050, $1051, $1052, $1053, $1054, $1055, $1056, $1057, $1058, $1059, $1060, $1061, $1062, $1063, $1064, $1065, $1066, $1067, $1068, $1069, $1060, $1061, $1062, $1063, $1064, $1065, $1066, $1067, $1068, $1069, $1070, $1071, $1072, $1073, $1074, $1075, $1076, $1077, $1078, $1079, $1070, $1071, $1072, $1073, $1074, $1075, $1076, $1077, $1078, $1079, $1080, $1081, $1082, $1083, $1084, $1085, $1086, $1087, $1088, $1089, $1080, $1081, $1082, $1083, $1084, $1085, $1086, $1087, $1088, $1089, $1090, $1091, $1092, $1093, $1094, $1095, $1096, $1097, $1098, $1099, $1090, $1091, $1092, $1093, $1094, $1095, $1096, $1097, $1098, $1099, $1100, $1101, $1102, $1103, $1104, $1105, $1106, $1107, $1108, $1109, $1100, $1101, $1102, $1103, $1104, $1105, $1106, $1107, $1108, $1109, $1110, $1111, $1112, $1113, $1114, $1115, $1116, $1117, $1118, $1119, $1110, $1111, $1112, $1113, $1114, $1115, $1116, $1117, $1118, $1119, $1120, $1121, $1122, $1123, $1124, $1125, $1126, $1127, $1128, $1129, $1120, $1121, $1122, $1123, $1124, $1125, $1126, $1127, $1128, $1129, $1130, $1131, $1132, $1133, $1134, $1135, $1136, $1137, $1138, $1139, $1130, $1131, $1132, $1133, $1134, $1135, $1136, $1137, $1138, $1139, $1140, $1141, $1142, $1143, $1144, $1145, $1146, $1147, $1148, $1149, $1140, $1141, $1142, $1143, $1144, $1145, $1146, $1147, $1148, $1149, $1150, $1151, $1152, $1153, $1154, $1155, $1156, $1157, $1158, $1159, $1150, $1151, $1152, $1153, $1154, $1155, $1156, $1157, $1158, $1159, $1160, $1161, $1162, $1163, $1164, $1165, $1166, $1167, $1168, $1169, $1160, $1161, $1162, $1163, $1164, $1165, $1166, $1167, $1168, $1169, $1170, $1171, $1172, $1173, $1174, $1175, $1176, $1177, $1178, $1179, $1170, $1171, $1172, $1173, $1174, $1175, $1176, $1177, $1178, $1179, $1180, $1181, $1182, $1183, $1184, $1185, $1186, $1187, $1188, $1189, $1180, $1181, $1182, $1183, $1184, $1185, $1186, $1187, $1188, $1189, $1190, $1191, $1192, $1193, $1194, $1195, $1196, $1197, $1198, $1199, $1190, $1191, $1192, $1193, $1194, $1195, $1196, $1197, $1198, $1199, $1200, $1201, $1202, $1203, $1204, $1205, $1206, $1207, $1208, $1209, $1200, $1201, $1202, $1203, $1204, $1205, $1206, $1207, $1208, $1209, $1210, $1211, $1212, $1213, $1214, $1215, $1216, $1217, $1218, $1219, $1210, $1211, $1212, $1213, $1214, $1215, $1216, $1217, $1218, $1219, $1220, $1221, $1222, $1223, $1224, $1225, $1226, $1227, $1228, $1229, $1220, $1221, $1222, $1223, $1224, $1225, $1226, $1227, $1228, $1229, $1230, $1231, $1232, $1233, $1234, $1235, $1236, $1237, $1238, $1239, $1230, $1231, $1232, $1233, $1234, $1235, $1236, $1237, $1238, $1239, $1240, $1241, $1242, $1243, $1244, $1245, $1246, $1247, $1248, $1249, $1240, $1241, $1242, $1243, $1244, $1245, $1246, $1247, $1248, $1249, $1250, $1251, $1252, $1253, $1254, $1255, $1256, $1257, $1258, $1259, $1250, $1251, $1252, $1253, $1254, $1255, $1256, $1257, $1258, $1259, $1260, $1261, $1262, $1263, $1264, $1265, $1266, $1267, $1268, $1269, $1260, $1261, $1262, $1263, $1264, $1265, $1266, $1267, $1268, $1269, $1270, $1271, $1272, $1273, $1274, $1275, $1276, $1277, $1278, $1279, $1270, $1271, $1272, $1273, $1274, $1275, $1276, $1277, $1278, $1279, $1280, $1281, $1282, $1283, $1284, $1285, $1286, $1287, $1288, $1289, $1280, $1281, $1282, $1283, $1284, $1285, $1286, $1287, $1288, $1289, $1290, $1291, $1292, $1293, $1294, $1295, $1296, $1297, $1298, $1299, $1290, $1291, $1292, $1293, $1294, $1295, $1296, $1297, $1298, $1299, $1300, $1301, $1302, $1303, $1304, $1305, $1306, $1307, $1308, $1309, $1300, $1301, $1302, $1303, $1304, $1305, $1306, $1307, $1308, $1309, $1310, $1311, $1312, $1313, $1314, $1315, $1316, $1317, $1318, $1319, $1310, $1311, $1312, $1313, $1314, $1315
```

```
destination_address,
destination_complement,
destination_reference,
destination_neighborhood,
destination_city,
destination_state,
destination_zipcode,
destination_lat,
destination_lng,

-- Cliente
customer_name,
customer_phone,

-- Valores
delivery_fee,
order_value,

-- Observações
observations,
pickup_code,

-- Datas
scheduled_for,
created_at
) VALUES (
$1, $2, $3, $4, $5,
$6, $7, $8,
$9, $10, $11, $12, $13, $14, $15, $16, $17,
$18, $19,
$20, $21,
$22, $23,
$24, NOW()
)
RETURNING *
`,[

company.id,
order.id,
'ifood',
order.displayId,
'pending', // Status inicial da entrega

company.address,
company.lat,
company.lng,

address.formattedAddress,
```

```
address.complement,  
address.reference,  
address.neighborhood,  
address.city,  
address.state,  
address.postalCode,  
address.coordinates?.latitude,  
address.coordinates?.longitude,  
  
order.customer.name,  
order.customer.phone?.number,  
  
deliveryFee,  
order.total?.orderAmount,  
  
delivery.observations,  
delivery.pickupCode,  
  
order.orderTiming === 'SCHEDULED' ? delivery.deliveryDateTime : null  
]);  
  
await client.query('COMMIT');  
  
const newDelivery = result.rows[0];  
  
console.log(`✅ Entrega criada: #${newDelivery.id} para pedido iFood ${order.displayId}`);  
  
// Notifica entregadores disponíveis  
await notifyAvailableDrivers(newDelivery);  
  
return newDelivery;  
  
} catch (error) {  
    await client.query('ROLLBACK');  
    console.error('Erro ao criar entrega:', error);  
    throw error;  
} finally {  
    client.release();  
}  
}  
  
async function calculateDeliveryFee(originLat, originLng, destLat, destLng) {  
    // Implementar sua lógica de cálculo de frete  
    // Pode usar distância, zona, etc.  
    return 800; // R$ 8,00 em centavos  
}
```

```
async function notifyAvailableDrivers(delivery) {  
    // Implementar notificação push/websocket para entregadores  
    console.log(`Notificando entregadores sobre entrega #${delivery.id}`);  
}  
  
module.exports = { createDeliveryFromIFoodOrder };
```

## 7. Tabela de Vínculo Empresa ↔ iFood

```
sql  
  
-- Adicionar coluna na tabela de empresas  
ALTER TABLE companies  
ADD COLUMN ifood_merchant_id VARCHAR(100) UNIQUE;  
  
-- Criar índice  
CREATE INDEX idx_companies_ifood_merchant ON companies(ifood_merchant_id);
```

## 8. Tratamento de Cancelamentos

```
javascript
```

```

// Adicionar no orderProcessor.js

async function processOrderEvent(event) {
  const { fullCode, orderId } = event;

  // Criação de entrega
  if (TRIGGER_EVENTS.includes(fullCode)) {
    // ... código anterior
  }

  // Cancelamento do pedido
  if (fullCode === 'CANCELLED') {
    await cancelDeliveryByExternalId(orderId);
  }
}

async function cancelDeliveryByExternalId(externalId) {
  const { pool } = require('../database');

  const result = await pool.query(`

    UPDATE deliveries
    SET status = 'cancelled',
        cancelled_at = NOW(),
        cancellation_reason = 'Pedido cancelado no iFood'
    WHERE external_id = $1
    AND external_source = 'ifood'
    AND status NOT IN ('delivered', 'cancelled')
    RETURNING id
  `, [externalId]);

  if (result.rows.length > 0) {
    console.log(`❌ Entrega #${result.rows[0].id} cancelada (pedido iFood cancelado)`);

    // Se já tinha entregador, notificar
    // await notifyDriverAboutCancellation(result.rows[0].id);
  }
}

```

## 9. Configuração Completa

### 9.1 Arquivo .env

env

```
# iFood
IFOOD_CLIENT_ID=seu_client_id
IFOOD_CLIENT_SECRET=seu_client_secret
IFOOD_BASE_URL=https://merchant-api.ifood.com.br

# Banco de dados
DATABASE_URL=postgresql://user:pass@localhost:5432/deliveries

# Webhook
WEBHOOK_SECRET=seu_secret_para_validacao
```

## 9.2 Estrutura de Pastas

```
src/
  └── routes/
    └── webhooks/
      └── ifood.js
  └── services/
    └── ifood/
      ├── auth.js
      ├── polling.js
      ├── orderService.js
      └── orderProcessor.js
    └── delivery/
      └── deliveryService.js
  └── database/
    └── index.js
```

## 10. Checklist de Homologação iFood

Para ter o app aprovado no iFood, você precisa:

- Fazer polling a cada 30 segundos
- Enviar acknowledgment para todos eventos recebidos
- Receber e processar pedidos DELIVERY imediatos
- Receber e processar pedidos DELIVERY agendados
- Tratar cancelamentos
- Respeitar rate limits da API
- Renovar token somente quando expirar

## 11. Links Úteis

- **Portal Developer:** <https://developer.ifood.com.br>
  - **Documentação API:** <https://developer.ifood.com.br/docs/guides>
  - **API Reference:** <https://developer.ifood.com.br/reference>
  - **Eventos de Pedido:** <https://developer.ifood.com.br/docs/guides/order/events>
- 

## Próximos Passos

1. **Criar conta** no Portal Developer do iFood
2. **Configurar webhook** ou implementar polling
3. **Vincular empresas** ao `ifood_merchant_id`
4. **Testar** com pedidos de teste no ambiente sandbox
5. **Solicitar homologação** após testes completos