APIs — Implementação, Consumo e Integração

1. Implementação (como criar uma API)

Quando falamos em **implementar uma API**, estamos no lado do **servidor**, ou seja, quem oferece os dados ou serviços.
Um jeito simples de entender é imaginar um **restaurante**:

- O **cliente** pede um prato.
- O **garçom** (API) leva o pedido até a **cozinha** (servidor).
- A cozinha prepara e devolve a comida, e o garçom entrega ao cliente.

Na API, os endereços (chamados endpoints) funcionam como "portas" para pedir algo. Por exemplo:

- GET /produtos → lista os produtos disponíveis.
- POST /pedidos → cria um pedido novo.

Ao implementar uma API, é preciso definir **como o servidor vai responder**:

- Escolher um formato padrão (geralmente **JSON**, que organiza os dados em caixinhas).
- Retornar mensagens claras, como **200 OK** quando deu certo ou **404 Not Found** quando não existe.
- Garantir **segurança**, aceitando só conexões seguras (HTTPS) e verificando quem está pedindo (com chaves ou tokens).
- Evitar erros duplicados: se o usuário mandar o mesmo pedido duas vezes, o sistema deve reconhecer e não criar pedidos em dobro (isso é chamado de **idempotência**).

Exemplo simples:
Uma loja online cria uma API para que aplicativos consigam listar produtos, criar pedidos e acompanhar entregas.

IFOOD

Exemplo de Implementação de API — iFood

Imagine o **servidor do iFood**. Ele é quem controla todos os restaurantes, cardápios, pedidos e entregas.
Para que o aplicativo (no celular do cliente) consiga acessar essas informações, o iFood **implementa APIs** no seu sistema.

Essas APIs têm vários **endpoints** (portas de acesso), cada uma com uma função:

- GET /restaurantes → devolve a lista de restaurantes disponíveis perto do usuário.
- GET /restaurantes/{id}/cardapio → devolve o cardápio de um restaurante específico.
- POST /pedidos → cria um pedido novo (com os itens escolhidos pelo cliente).
- GET /pedidos/{id} → mostra o status de um pedido (em preparo, a caminho, entregue).
- PUT /pedidos/{id}/status → o restaurante pode atualizar o status do pedido.

Como funciona na prática:

- **Cliente (dentro do iFood)**: "Quero o cardápio da pizzaria X."
- **API (servidor)**: recebe a requisição no endpoint `/restaurantes/123/cardapio`.
- **Servidor**: busca no banco de dados os itens da pizzaria 123.
- **API**: devolve uma resposta em JSON, algo assim:

```json
{
  "restaurante": "Pizzaria do Bairro",
  "cardapio": [
    {"item": "Pizza de Calabresa", "preco": 45.90},
    {"item": "Refrigerante 2L", "preco": 9.50}
  ]
}
```

- **App do cliente**: traduz esse JSON em telas bonitas para o usuário ver.

O que a implementação precisa cuidar:

- **Segurança**: só usuários logados podem criar pedidos.
- **Idempotência**: se o cliente clicar em "Finalizar pedido" duas vezes, não deve gerar 2 pedidos.

- **Mensagens claras**: se um restaurante estiver fechado, retornar 400 ou 404 com a mensagem “Restaurante indisponível”.
- **Escalabilidade**: como milhões de usuários pedem ao mesmo tempo, a API precisa estar preparada para responder rápido e com estabilidade.

👉 Ou seja, quando dizemos que o iFood “implementa uma API”, significa que ele programou **essas regras, endpoints e respostas** dentro do servidor para que todos os clientes (apps) e parceiros (restaurantes, entregadores, meios de pagamento) consigam se comunicar de forma padronizada.

2. Consumo da API (como usar uma API)

Se a **implementação** é o lado do servidor, o **consumo** é o lado do cliente.
É quando um aplicativo, site ou sistema faz um **pedido** para a API e recebe uma **resposta**.

Voltando ao exemplo do restaurante: agora você está do outro lado, **como cliente**. Você pede a comida (requisição) e recebe o prato pronto (resposta).

No caso do **iFood**, quem consome a API é:

- o **aplicativo no celular do cliente**,
- o **sistema do restaurante**,
- e até **outros serviços integrados** (como meios de pagamento).

Exemplo prático no iFood:

1. O usuário abre o app e procura a “Pizzaria do Bairro”.
2. O app faz uma requisição: GET /restaurantes/123/cardapio.
3. A API respdonde com o cardápio em formato JSON:

```json
{
  "restaurante": "Pizzaria do Bairro",
  "cardapio": [
    {"item": "Pizza Calabresa", "preco": 45.90},
    {"item": "Refrigerante 2L", "preco": 9.50}
  ]
}
```

4. O aplicativo lê essa resposta e transforma em uma **tela visual** com imagens, botões e preços para o usuário.

O que é importante no consumo:

- **Montar a requisição corretamente**: usar o endpoint certo, passar parâmetros (ex.: localização do cliente, filtros de restaurantes).
- **Tratar os erros**: se a internet cair ou o servidor estiver ocupado, o app precisa avisar o usuário ("Não foi possível carregar os restaurantes").
- **Paginação**: se existem centenas de restaurantes na região, a API devolve em páginas, e o app precisa pedir uma de cada vez (?page=1&limit=20).
- **Segurança**: o app precisa mandar junto o **token de autenticação** do usuário para provar que está autorizado a fazer aquele pedido.
- **Desempenho**: algumas informações podem ser guardadas em cache no app (como dados do restaurante) para não precisar pedir de novo a cada clique.

Resumindo com o iFood:

O **consumo da API** é o aplicativo **pedindo dados ao servidor** e transformando as respostas em telas e funcionalidades para o usuário.
É graças a esse consumo que você consegue ver o cardápio, acompanhar a entrega em tempo real e receber notificações sobre o status do pedido.

3. Integração de APIs (como ligar sistemas diferentes)

A integração é quando **diferentes sistemas conversam entre si** usando APIs. O iFood sozinho não consegue entregar toda a experiência: ele precisa se conectar com **restaurantes, entregadores, meios de pagamento e até serviços de mapa**. É nessa hora que as APIs fazem a ponte.

Exemplos de integração no iFood:

- **Pagamento**: quando você paga pelo app, o iFood chama a API de parceiros (ex.: Visa, MasterCard, Mercado Pago). A API confirma se o pagamento foi aprovado e devolve a resposta.
- **Localização**: para mostrar a rota do entregador, o iFood usa a **API do Google Maps** ou de outro serviço de mapas. Assim consegue calcular a distância e estimar o tempo de entrega.
- **Restaurantes**: alguns restaurantes usam sistemas próprios de gestão de pedidos. Esses sistemas recebem os pedidos automaticamente através da integração com a API do iFood, sem precisar digitar nada manualmente.
- **Notificações (webhooks)**: quando o status de um pedido muda (por exemplo, de "em preparo" para "a caminho"), o iFood envia automaticamente uma notificação ao restaurante ou ao cliente usando **webhooks**.

O que é importante na integração:

- **Padronização**: todos os parceiros falam a mesma "língua" (JSON, HTTPS, endpoints bem definidos).
- **Segurança**: cada sistema precisa de suas credenciais de acesso, evitando invasões.
- **Idempotência**: se o sistema de pagamento enviar duas vezes a confirmação, o iFood não pode registrar o pedido como pago em dobro.
- **Escalabilidade**: a integração deve funcionar bem mesmo com milhões de pedidos sendo processados ao mesmo tempo.

Resumindo com o iFood:

A **integração de APIs** é o que permite ao iFood juntar várias peças diferentes — **pagamento, mapa, restaurante e entregador** — para que, no fim, você consiga pedir uma pizza do sofá de casa e acompanhar até ela chegar à sua porta.