

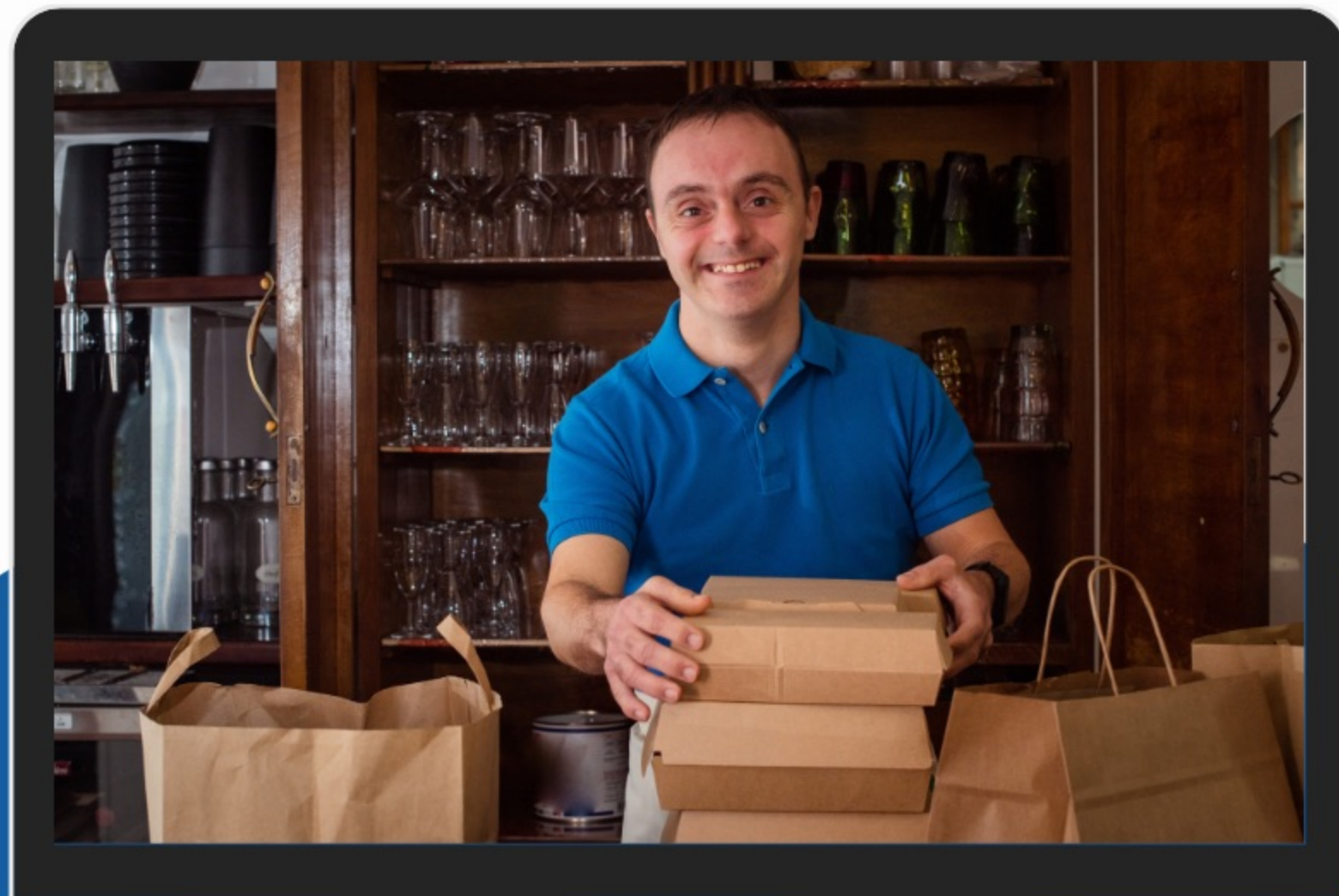
PedidosBot



Assistente de Pedidos

- Sistema de Gerenciamento de Pedidos via Chatbot, uma solução inovadora que facilita o controle de pedidos, otimiza a comunicação e melhora a eficiência operacional do seu negócio.

Maycon Spirlandelli



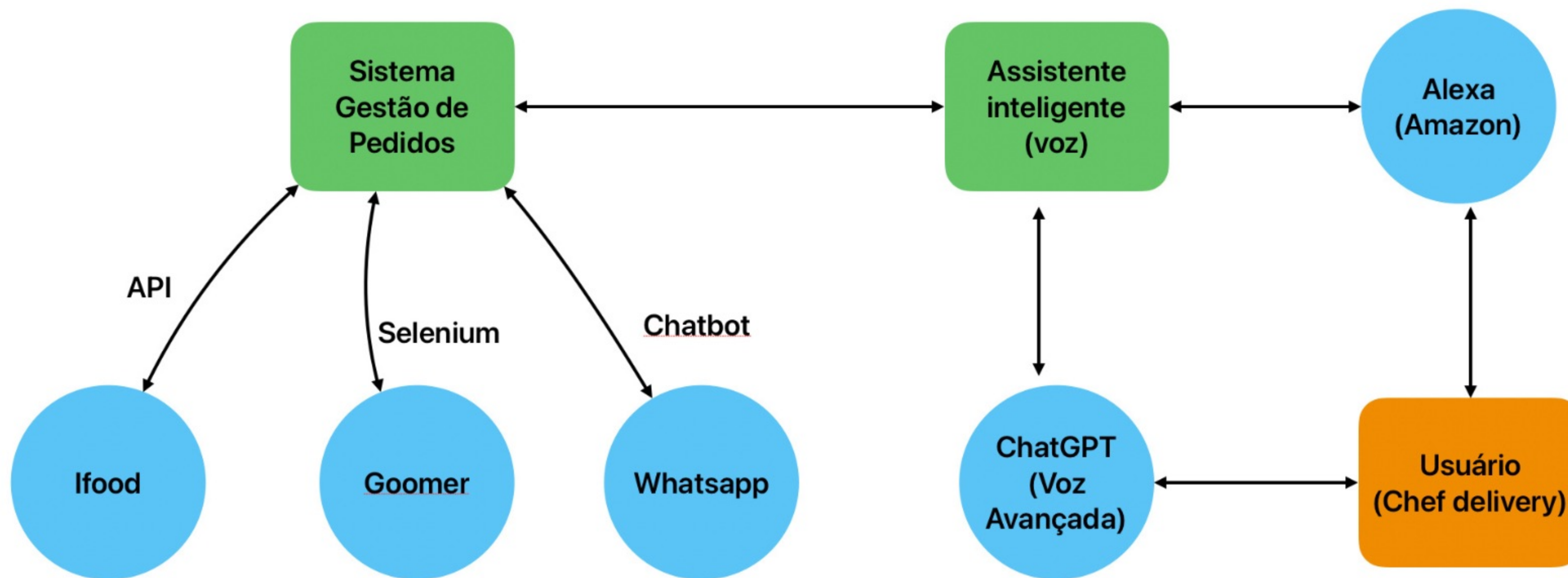
Principais Funcionalidades

- **Recebimento Automático de Pedidos** – O chatbot recebe os pedidos dos clientes, registra os detalhes e os encaminha diretamente para a equipe responsável.
- **Confirmação e Atualizações em Tempo Real** – O cliente recebe notificações automáticas sobre o status do pedido, desde a confirmação até a entrega.
- **Gestão de Cardápio e Estoque** – Atualize produtos, preços e disponibilidade de forma simples e automática.
- **Relatórios e Análises** – Gere relatórios detalhados sobre pedidos, tempo de preparo, vendas e desempenho do delivery.

Problemasenfretados

- Cancelamento de pedidos – atualizar o cardápio
- Demora no atualização do cardápio
- Gestão em planilhas e de forma manual
- Poucos funcionários

Design da Solução



Recebimento de pedidos

Detalhes do pedido

João B. • Pedido 4634

14º pedido na loja • Pelo iFood às 14:07

ChatFazer ligação

Status do pedido: Concluído

Ana Spirandelli Confeitaria

Localizador Nº 55940967

14:36

Entrega confirmada

Itens do pedido

1 Céus de UvasR\$ 15,99

1 Torta Ouro BrancoR\$ 29,99

SubtotalR\$ 45,98

Taxa de entregaR\$ 4,99

TotalR\$ 50,97

#0001Concluído

Kelly Oliveira

Entrega

Rua 7 E, 8, (Q), Ga

[Ir para o Google](#)

Qtd	Itens	Cód.	Preços
1	Pote da Felicidade Bis Max	045	R\$ 13,00
3	Bombom de morango no Pote	0445	R\$ 42,00
1	Bolo no pote Ninho com morango	5654	R\$ 14,00
3	Pote da Felicidade Bis Max	045	R\$ 39,00

Subtotal: R\$ 108,00

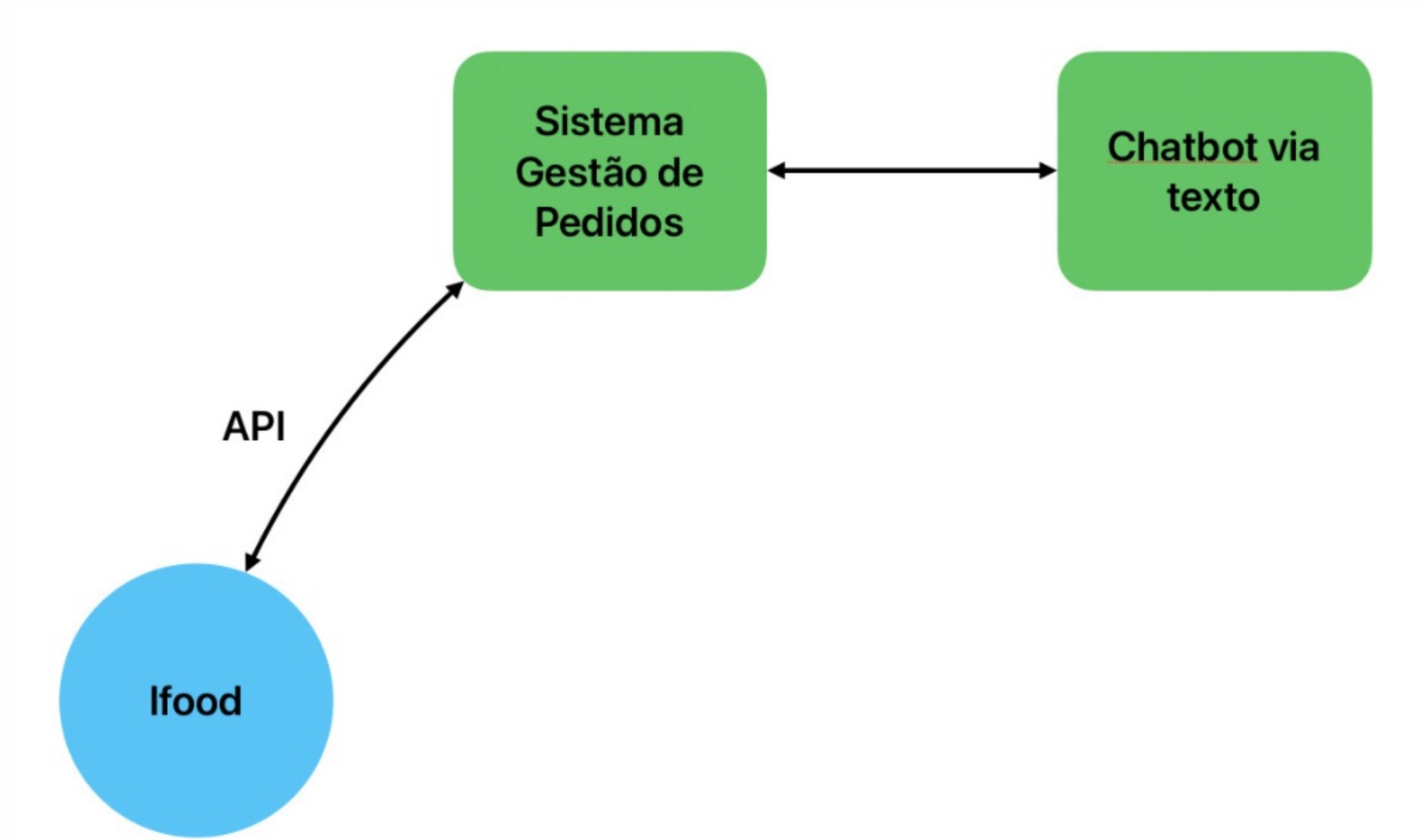
Taxa de entrega: +Grátis

Total: R\$ 108,00

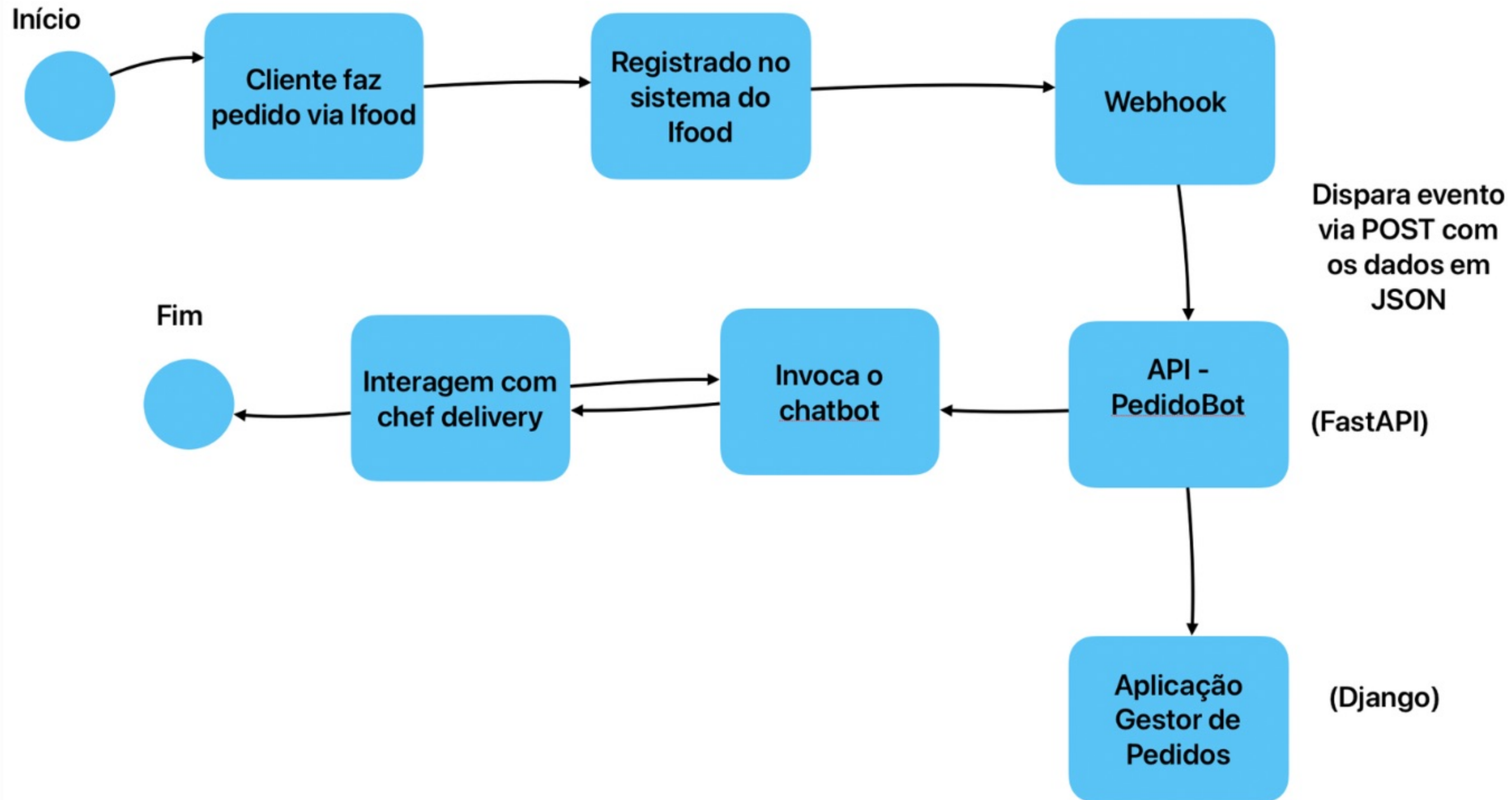
Proposta da Solução

- Atualização rápida do cardápio em todas plataformas
- Acompanhar os pedidos em tempo real – (pedidos são preparados na hora)
- Automação processos
- Aumentar a produtividade do chef de cozinha

Foco da API

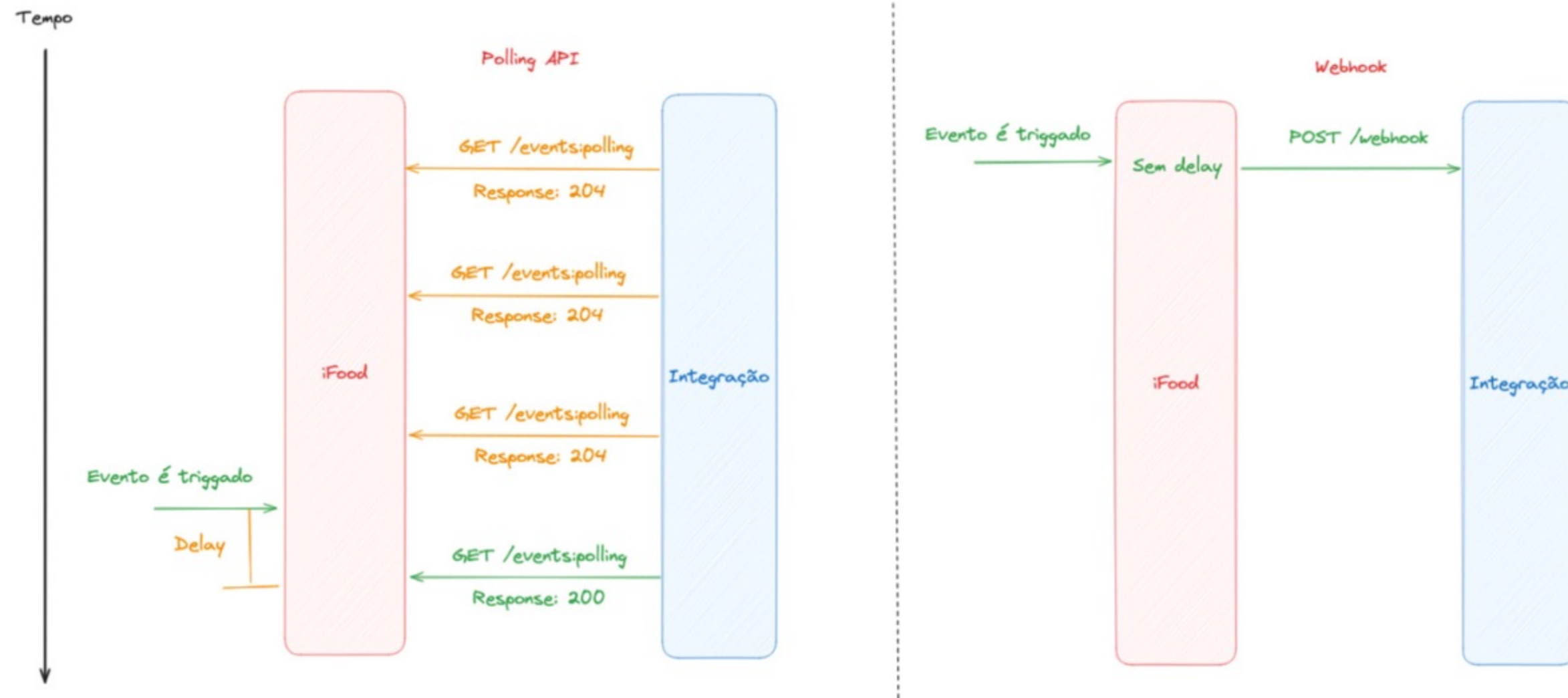


Arquitetura da Solução - pedidos via Ifood



Polling vs Webhook

Webhooks são uma alternativa ao desperdício do polling contínuo. O exemplo a seguir usa os eventos requests/create webhook para ilustrar a diferença:



O aplicativo especifica um endpoint HTTPS hospedado pelo servidor da integradora para receber eventos do webhook.

Fonte: <https://developer.ifood.com.br/pt-BR/docs/guides/order/events/delivery-methods/webhook/overview/>

API

Assistente de Pedidos 1.0 OAS 3.1

[/openapi.json](#)

Agente inteligente responsável por auxiliar o chef de cozinha durante a gestão de pedidos de um delivery

[Terms of service](#)

[Maycon - Website](#)

[Send email to Maycon](#)

[Apache 2.0](#)

Endpoint do sistema

POST

/chatbot/v2 Endpoint para interação com o chatbot.

POST

/relatorio/v3 Endpoint de relatório via chatbot.

Interface com outros sistemas via API

POST

/webhook/ Receber Webhook do Ifood

Schemas

VIDEO

Solução - API

Para invocar API
fastapi dev api.py

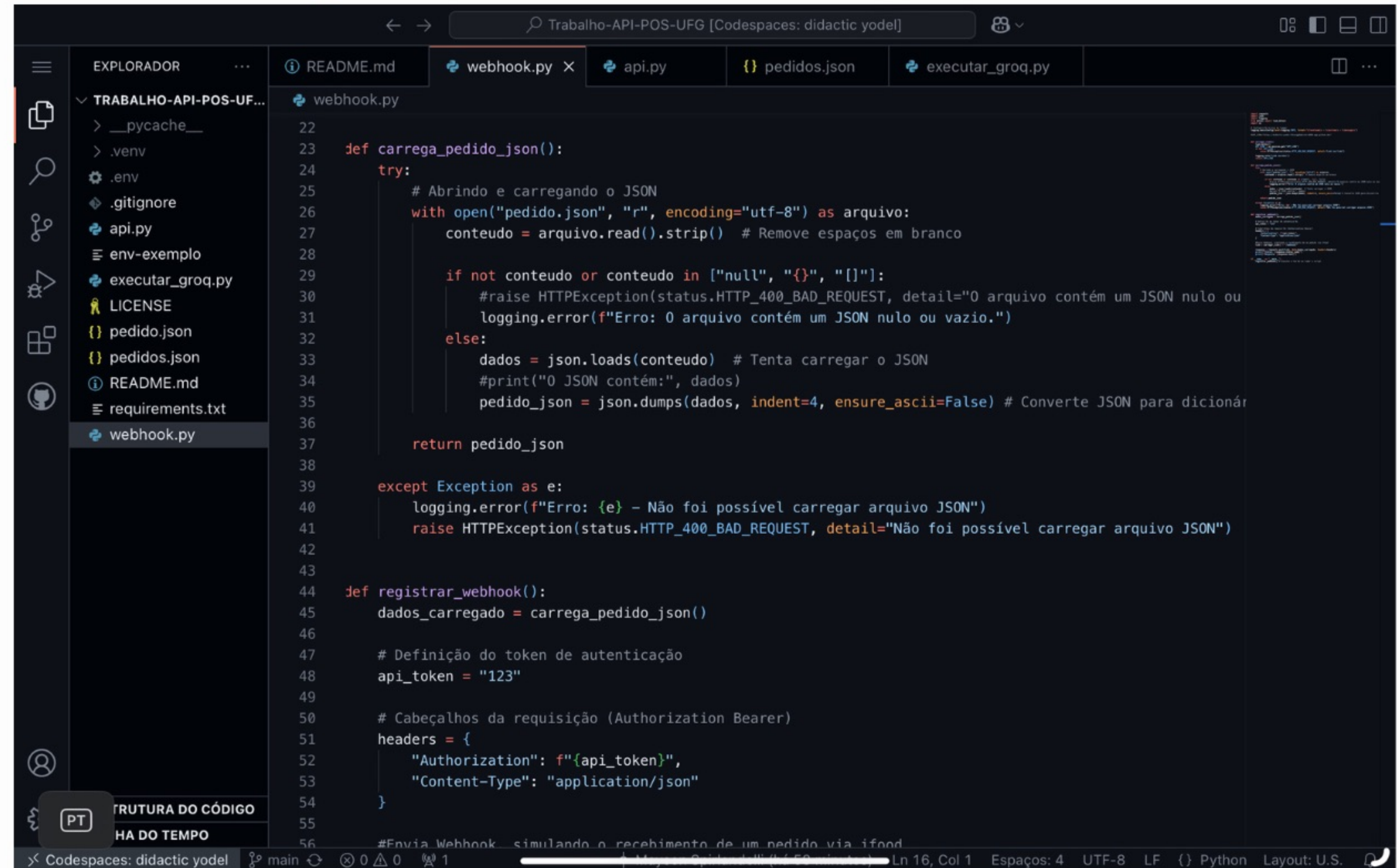
```
Trabalho-API-POS-UF... [Codespaces: didactic yodel]

EXPLORADOR
  > __pycache__
  > .venv
  > .env
  > .gitignore
  > api.py
  > env-exemplo
  > executar_groq.py
  > LICENSE
  > pedido.json
  > pedidos.json
  > README.md
  > requirements.txt
  > webhook.py

api.py
37 def common_api_token(api_token: str):
43     logging.info("Autorizado com sucesso!")
44
45
46 @app.post("/chatbot/v2",
47           tags=[GrupoNome.api],
48           status_code=status.HTTP_200_OK,
49           dependencies=[Depends(common_api_token)],
50           summary="Endpoint para interação com o chatbot.",
51           description=
52             """
53             Este endpoint é responsável por invocar o chatbot
54
55             ### Parâmetros:
56             - prompt (str): Mensagem enviada pelo usuário ao chatbot.
57             - api_token (str): Token obrigatório pra autorização ao sistema.
58
59             Retorno:
60             - mensagem: Resposta gerada pelo chatbot.
61             """
62         )
63 def chatbot(prompt: str):
64     resultado = executar_groq(prompt)
65     print(resultado)
66     return {"mensagem": resultado}
67
68
69 # Webhook para receber pedidos
70 @app.post("/webhook/",
71           tags=[GrupoNome.webhook],
72           status_code=status.HTTP_200_OK,
73           summary="Receber Webhook do Ifood",
74           description=
75             """
76             Este endpoint recebe um webhook do Ifood e invoca o assistente inteligente.
77             """
78         )
```


Solução - webhook

Para invocar webhook
python webhook.py



```
22
23 def carrega_pedido_json():
24     try:
25         # Abrindo e carregando o JSON
26         with open("pedido.json", "r", encoding="utf-8") as arquivo:
27             conteudo = arquivo.read().strip() # Remove espaços em branco
28
29         if not conteudo or conteudo in ["null", "{}", "[]"]:
30             #raise HTTPException(status.HTTP_400_BAD_REQUEST, detail="O arquivo contém um JSON nulo ou
31             logging.error(f"Erro: O arquivo contém um JSON nulo ou vazio.")
32         else:
33             dados = json.loads(conteudo) # Tenta carregar o JSON
34             #print("O JSON contém:", dados)
35             pedido_json = json.dumps(dados, indent=4, ensure_ascii=False) # Converte JSON para dicionário
36
37         return pedido_json
38
39     except Exception as e:
40         logging.error(f"Erro: {e} - Não foi possível carregar arquivo JSON")
41         raise HTTPException(status.HTTP_400_BAD_REQUEST, detail="Não foi possível carregar arquivo JSON")
42
43
44 def registrar_webhook():
45     dados_carregado = carrega_pedido_json()
46
47     # Definição do token de autenticação
48     api_token = "123"
49
50     # Cabeçalhos da requisição (Authorization Bearer)
51     headers = {
52         "Authorization": f"{api_token}",
53         "Content-Type": "application/json"
54     }
55
56     #Envia Webhook simulando o recebimento de um pedido via ifood
```

EXPLORADOR

- TRABALHO-API-POS-UF...
- > __pycache__
- > .venv
- .env
- .gitignore
- api.py
- env-exemplo
- executar_groq.py
- LICENSE
- pedido.json
- pedidos.json
- README.md
- requirements.txt
- webhook.py

Trabalho-API-POS-UFG [Codespaces: didactic yodel]

webhook.py

PT

TRUTURA DO CÓDIGO

HA DO TEMPO

Codespaces: didactic yodel

main

0 0 1

Ln 16, Col 1

Espaços: 4

UTF-8

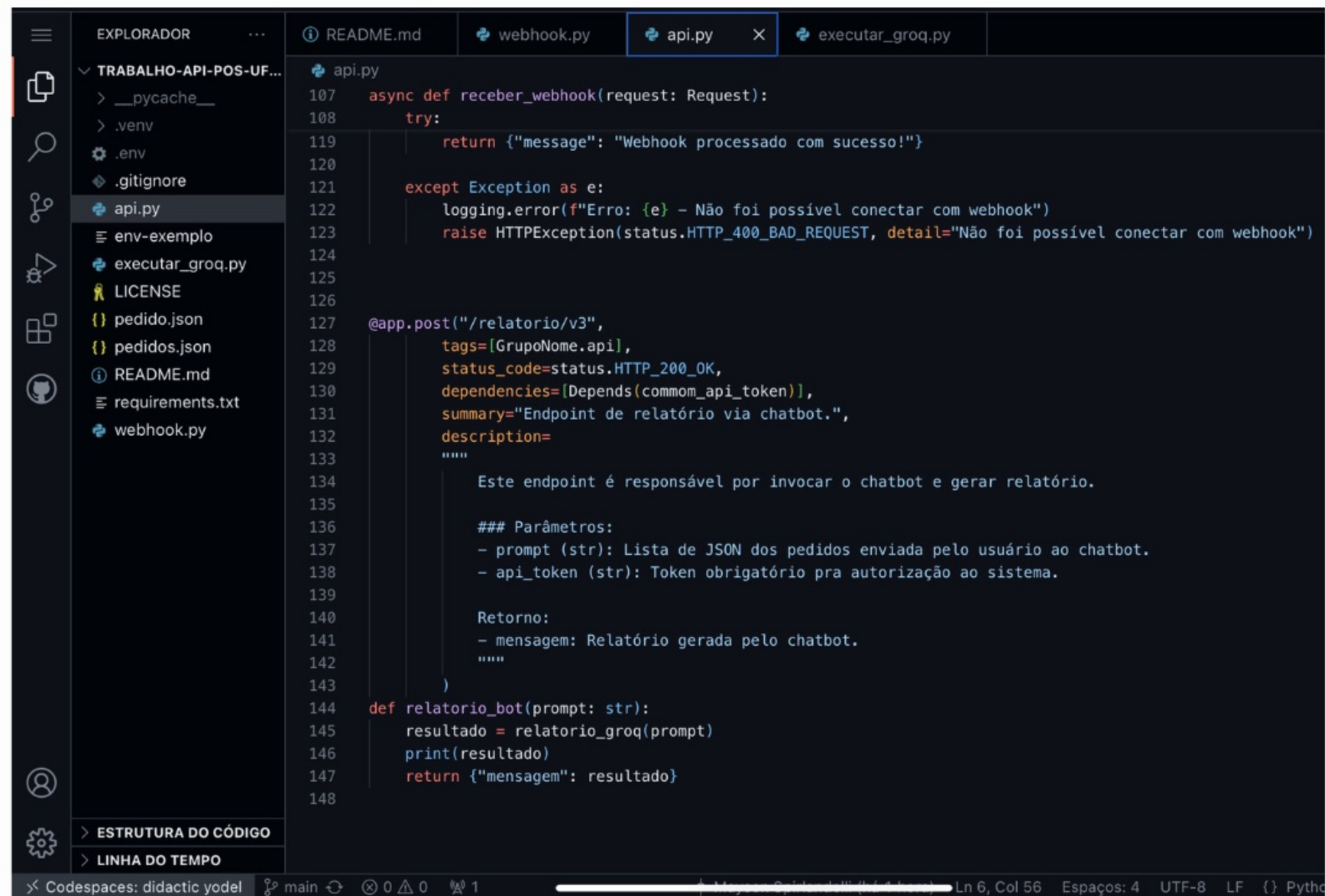
LF

{ } Python

Layout: U.S.

Solução - relatório

API do relatório



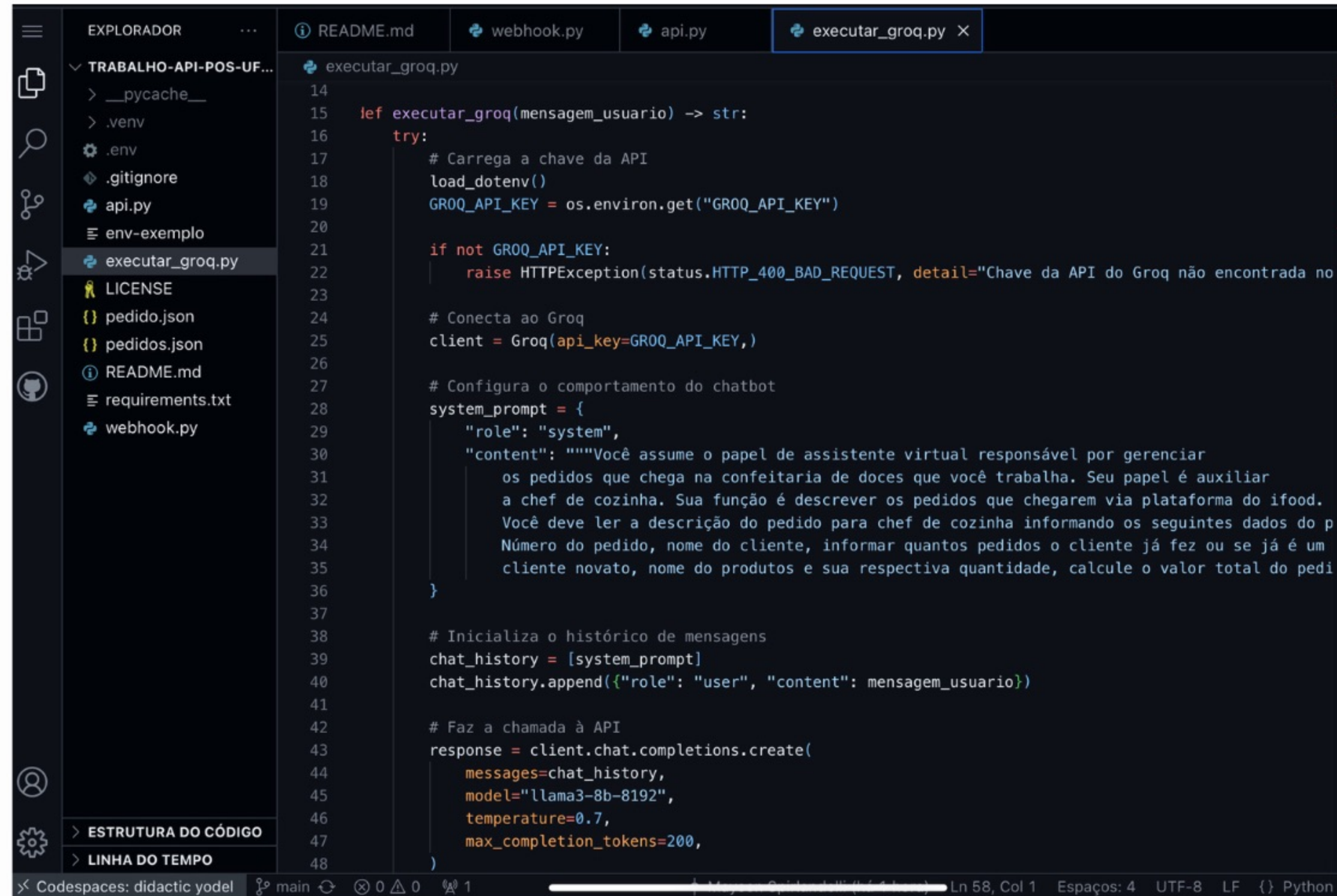
```
EXPLORADOR
  TRABALHO-API-POS-UF...
    __pycache__
    .env
    .gitignore
    api.py
    env-exemplo
    executar_groq.py
    LICENSE
    pedido.json
    pedidos.json
    README.md
    requirements.txt
    webhook.py

api.py
107 async def receber_webhook(request: Request):
108     try:
119         return {"message": "Webhook processado com sucesso!"}
120
121     except Exception as e:
122         logging.error(f"Erro: {e} - Não foi possível conectar com webhook")
123         raise HTTPException(status.HTTP_400_BAD_REQUEST, detail="Não foi possível conectar com webhook")
124
125
126
127 @app.post("/relatorio/v3",
128           tags=[GrupoNome.api],
129           status_code=status.HTTP_200_OK,
130           dependencies=[Depends(commom_api_token)],
131           summary="Endpoint de relatório via chatbot.",
132           description=
133           """
134           Este endpoint é responsável por invocar o chatbot e gerar relatório.
135
136           ### Parâmetros:
137           - prompt (str): Lista de JSON dos pedidos enviada pelo usuário ao chatbot.
138           - api_token (str): Token obrigatório pra autorização ao sistema.
139
140           Retorno:
141           - mensagem: Relatório gerada pelo chatbot.
142           """)
143
144 def relatorio_bot(prompt: str):
145     resultado = relatorio_groq(prompt)
146     print(resultado)
147     return {"mensagem": resultado}
148
```

Codespaces: didactic yodel | main | 0 | 0 | 1 | Ln 6, Col 56 | Espaços: 4 | UTF-8 | LF | {} | Python

Solução - LLM

LLM Groq



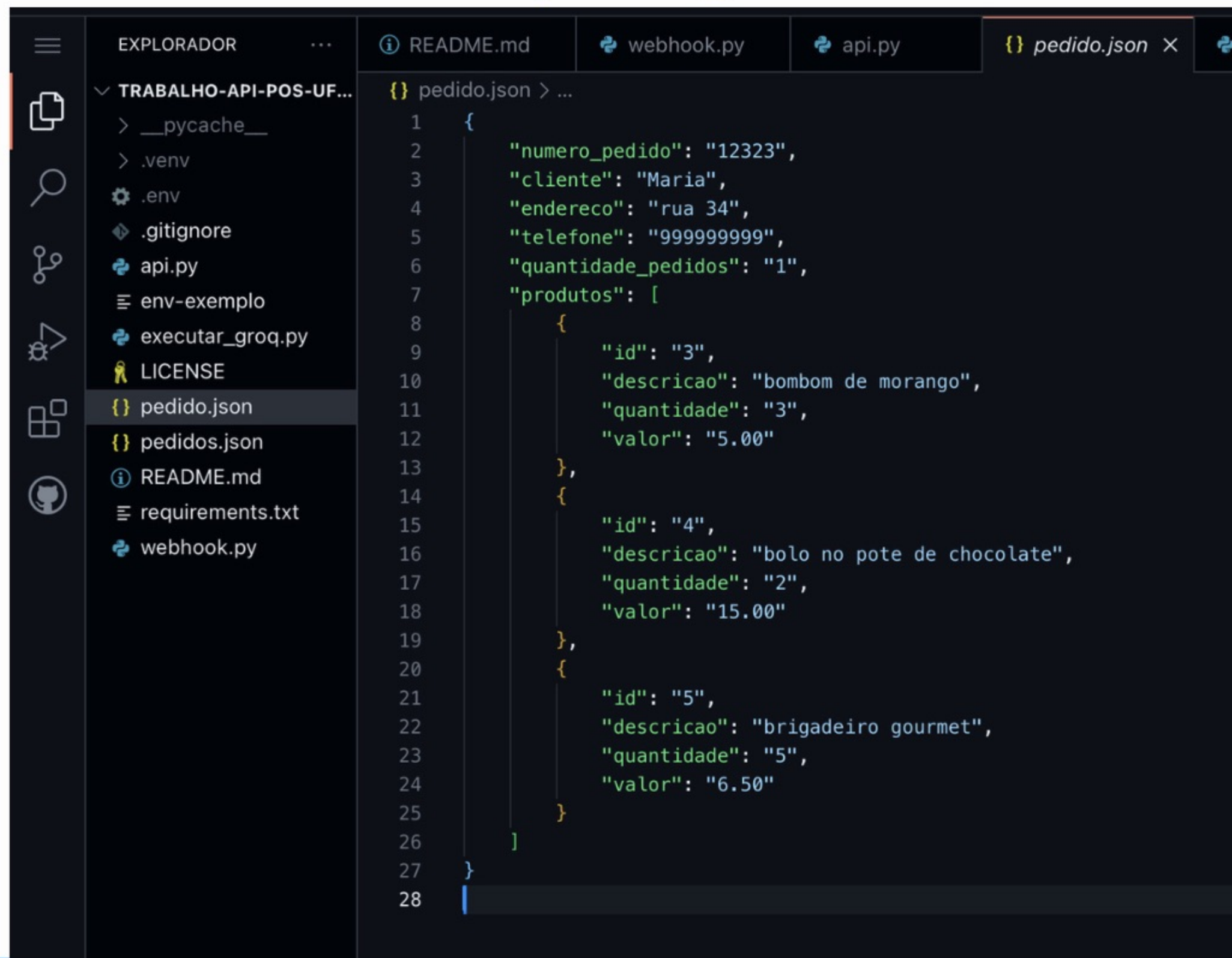
The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'TRABALHO-API-POS-UF...' with files like '__pycache__', '.venv', '.env', '.gitignore', 'api.py', 'env-exemplo', 'executar_groq.py', 'LICENSE', 'pedido.json', 'pedidos.json', 'README.md', 'requirements.txt', and 'webhook.py'. The code editor shows the content of 'executar_groq.py'.

```
14
15 def executar_groq(mensagem_usuario) -> str:
16     try:
17         # Carrega a chave da API
18         load_dotenv()
19         GROQ_API_KEY = os.environ.get("GROQ_API_KEY")
20
21         if not GROQ_API_KEY:
22             raise HTTPException(status.HTTP_400_BAD_REQUEST, detail="Chave da API do Groq não encontrada no
23
24         # Conecta ao Groq
25         client = Groq(api_key=GROQ_API_KEY,)
26
27         # Configura o comportamento do chatbot
28         system_prompt = {
29             "role": "system",
30             "content": "Você assume o papel de assistente virtual responsável por gerenciar
31                 os pedidos que chega na confeitaria de doces que você trabalha. Seu papel é auxiliar
32                 a chef de cozinha. Sua função é descrever os pedidos que chegarem via plataforma do ifood.
33                 Você deve ler a descrição do pedido para chef de cozinha informando os seguintes dados do p
34                 Número do pedido, nome do cliente, informar quantos pedidos o cliente já fez ou se já é um
35                 cliente novato, nome do produtos e sua respectiva quantidade, calcule o valor total do pedi
36         }
37
38         # Inicializa o histórico de mensagens
39         chat_history = [system_prompt]
40         chat_history.append({"role": "user", "content": mensagem_usuario})
41
42         # Faz a chamada à API
43         response = client.chat.completions.create(
44             messages=chat_history,
45             model="llama3-8b-8192",
46             temperature=0.7,
47             max_completion_tokens=200,
48         )
```

Codespaces: didactic yodel main 0 0 1 Ln 58, Col 1 Espaços: 4 UTF-8 LF {} Python

Solução - JSON

JSON de pedido

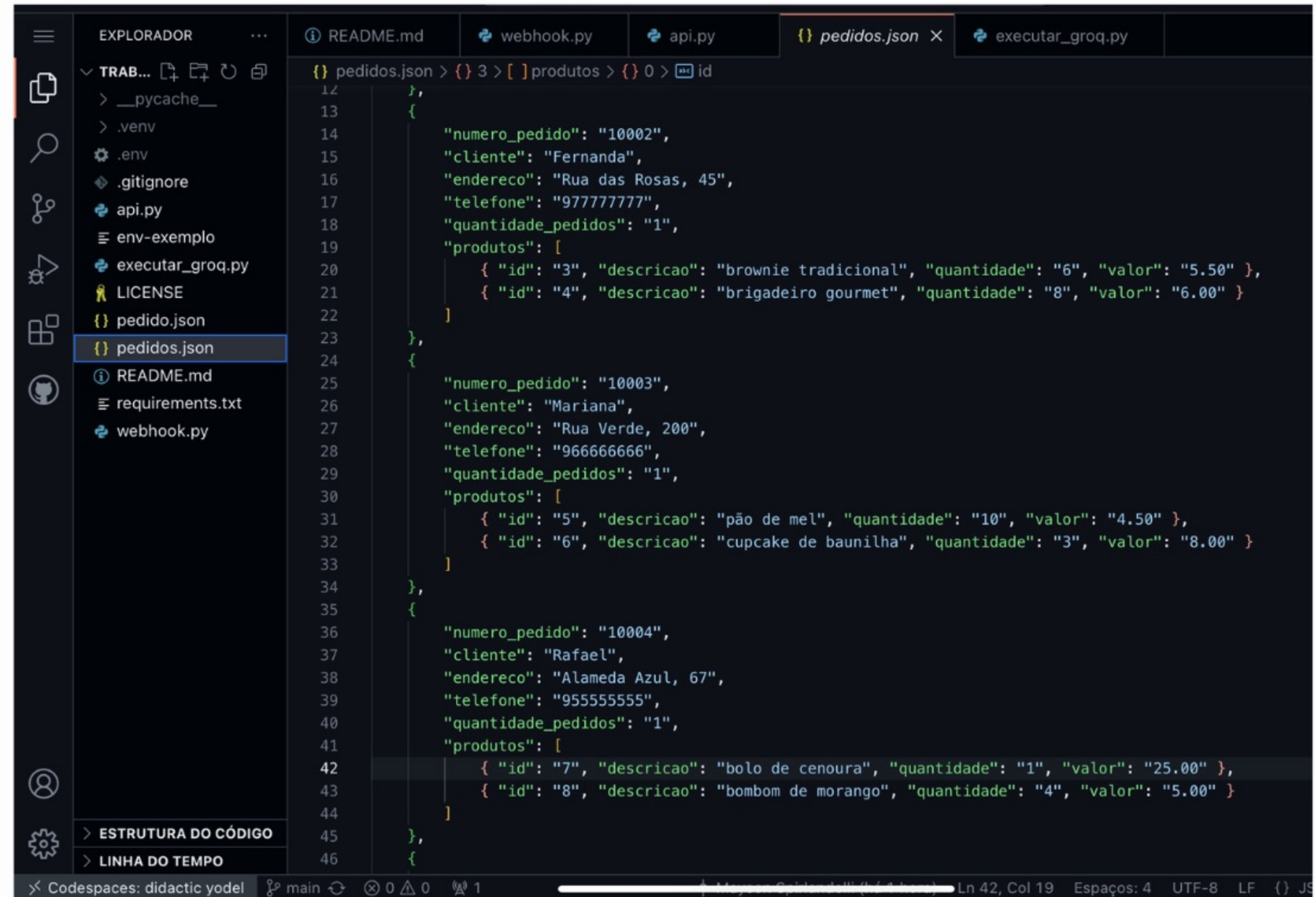


The screenshot shows a code editor with a dark theme. On the left, the 'EXPLORADOR' (Explorer) sidebar shows a project named 'TRABALHO-API-POS-UF...'. The file list includes: '__pycache__', '.venv', '.env', '.gitignore', 'api.py', 'env-exemplo', 'executar_groq.py', 'LICENSE', 'pedido.json' (selected), 'pedidos.json', 'README.md', 'requirements.txt', and 'webhook.py'. The main editor area has tabs for 'README.md', 'webhook.py', 'api.py', and 'pedido.json'. The 'pedido.json' tab is active, displaying the following JSON content:

```
1  {
2      "numero_pedido": "12323",
3      "cliente": "Maria",
4      "endereco": "rua 34",
5      "telefone": "999999999",
6      "quantidade_pedidos": "1",
7      "produtos": [
8          {
9              "id": "3",
10             "descricao": "bombom de morango",
11             "quantidade": "3",
12             "valor": "5.00"
13         },
14         {
15             "id": "4",
16             "descricao": "bolo no pote de chocolate",
17             "quantidade": "2",
18             "valor": "15.00"
19         },
20         {
21             "id": "5",
22             "descricao": "brigadeiro gourmet",
23             "quantidade": "5",
24             "valor": "6.50"
25         }
26     ]
27 }
28
```

Solução - Lista de JSON

JSON com uma lista de pedidos



```
12  },
13  {
14      "numero_pedido": "10002",
15      "cliente": "Fernanda",
16      "endereco": "Rua das Rosas, 45",
17      "telefone": "977777777",
18      "quantidade_pedidos": "1",
19      "produtos": [
20          { "id": "3", "descricao": "brownie tradicional", "quantidade": "6", "valor": "5.50" },
21          { "id": "4", "descricao": "brigadeiro gourmet", "quantidade": "8", "valor": "6.00" }
22      ]
23  },
24  {
25      "numero_pedido": "10003",
26      "cliente": "Mariana",
27      "endereco": "Rua Verde, 200",
28      "telefone": "966666666",
29      "quantidade_pedidos": "1",
30      "produtos": [
31          { "id": "5", "descricao": "pão de mel", "quantidade": "10", "valor": "4.50" },
32          { "id": "6", "descricao": "cupcake de baunilha", "quantidade": "3", "valor": "8.00" }
33      ]
34  },
35  {
36      "numero_pedido": "10004",
37      "cliente": "Rafael",
38      "endereco": "Alameda Azul, 67",
39      "telefone": "955555555",
40      "quantidade_pedidos": "1",
41      "produtos": [
42          { "id": "7", "descricao": "bolo de cenoura", "quantidade": "1", "valor": "25.00" },
43          { "id": "8", "descricao": "bombom de morango", "quantidade": "4", "valor": "5.00" }
44      ]
45  },
46  }
```

Demonstração

OBRIGADO!