**Tech Challenge Fase 01 - Tech Lanches**

**Grupo G12**

Denis Barbosa de Amorim - RM351570

Gabriel Lima Gomes - RM351052

Mateus Aragão Oliveira - RM351130

Nicolas de Oliveira Soares - RM350915

Vitor de Oliveira Lupinetti - RM351670

**Repositório**: <https://github.com/g12-4soat/tech-lanches>

**Wiki**: <https://github.com/g12-4soat/tech-lanches/wiki>

**Miro**: <https://miro.com/app/board/uXjVModCVvo=/?share_link_id=379818088124>

**Swagger**: <http://localhost:5050/swagger/index.html>

**Swagger Json**: <http://localhost:5050/swagger/v1/swagger.json>

**Redoc**: [http://localhost:5050/](http://localhost:5050/swagger/index.html)api-docs/index.html

**Postman Collection**: [https://github.com/g12-4soat/tech-lanches/blob/main/docs/TechLanches.postman\_collection.json](https://github.com/g12-4soat/tech-lanches/blob/main/docs/Tech%20Lanches.postman_collection.json)

**Postman Local Environments**: <https://github.com/g12-4soat/tech-lanches/blob/main/docs/TechLanches-Local.postman_environment.json>

**30 de Outubro de 2023**

**Sumário**

[**1. Diagrama Arquitetural do Projeto 3**](#_gjdgxs)

[**2. Tecnologias 3**](#_f60z1559q8py)

[**3. Como inicializar a aplicação 3**](#_5rugn091poss)

[**3.1. Dependências 3**](#_wv1emx3r9h5g)

[**3.1.1. Dependências opcionais 4**](#_u569m6trv7kg)

[**3.1.2. Dependências obrigatórias 4**](#_dg6n5e6267fx)

[**3.2. Procedimento 4**](#_ifbtm61oxxul)

[**3.3. Setup do Projeto e Seeds 4**](#_s9vuziqji6p7)

[**4. Fluxos da Aplicação 4**](#_k7sktvsxpk8)

[**4.1. Geral 4**](#_k6l048orcf5c)

[**4.2. Compromisso de dados 5**](#_v1cjz52arhzb)

[**4.3. Erros, Validações e Tratativas 5**](#_wqouf2p238f8)

[**5. Contextos 5**](#_975bzza4q7zd)

[**5.1. Contexto: Identificação 5**](#_jag57h4miell)

[Cliente não se identifica informando o CPF 5](#_3l879rmwjq1f)

[Cliente se identifica informando o CPF 5](#_1fbwiz8esiw0)

[Cliente não existente 6](#_lhjqftqdy5yb)

[Cadastro de Cliente 6](#_3br8q67xwoiw)

[**5.2. Contexto: Checkout 6**](#_4g6sev8gx6zr)

[Buscar todos os produtos 7](#_ykitjehbdnut)

[Buscar produtos por categoria 7](#_bw2knhybb1if)

[Pedido com cliente não identificado 8](#_xw798rfmbefe)

[Pedido com cliente identificado 8](#_9zfx4t4tusa)

[**5.3. Contexto: Pagamento 8**](#_cgbvdl17op5w)

[**5.4. Contexto: Preparo 9**](#_uinn139g061m)

[Pedido em preparação 9](#_d2mlowpjle31)

[Pedido pronto 9](#_ms5uov3cegu5)

[**5.5. Contexto: Retirada/Finalização 9**](#_t2go7noduod4)

[Pedidos por Status 10](#_wkogazeyzcbk)

[Pedido Retirado 11](#_slxzf6ppzov9)

**Fluxo de Eventos**

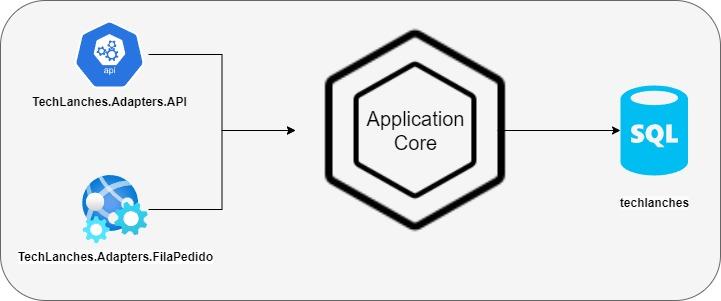
**Lanchonete Tech Lanches**

**(Totem)**

# **Diagrama Arquitetural do Projeto**

O projeto tem como atual estrutura monolítica:

* Uma API que recebe requisições REST para interação com a aplicação
* Um serviço do tipo *background service* que executa e gerencia a fila de pedidos



# Tecnologias

Para a construção e desenvolvimento do projeto, foram escolhidas as seguintes tecnologias:

* C# - Linguagem de Programação
* .NET 6 – Framework de Desenvolvimento Multiplataforma
* MSSQL Server – Banco de Dados Relacional
* Docker - Containers

# Como **inicializar a aplicação**

# **Dependências**

Para se executar a aplicação, é necessário que algumas dependências (algumas delas opcionais) sejam instaladas em sua máquina, sendo elas:

# Dependências **opcionais**

* **Git** – para download do código fonte do projeto via *git clone* ou baixar o .*zip*
* **Visual Studio** ou **Visual Studio Code** – IDE e editor de texto respectivamente para maior facilidade ao analisar o código
* **Postman** – REST Client para interação com o serviço de API (pode-se ser utilizado diretamente o Swagger da aplicação para realizar as requisições)
* **Microsoft SQL Server** – para visualização do banco de dados e suas estruturas. Pode-se utilizar outros tipos de ferramentas como **DBeaver** ou até mesmo diretamente do console usando o **sqlcmd utility**

# Dependências **obrigatórias**

* **Docker**

# **Procedimento**

O procedimento de inicialização do projeto é simples e leva poucos passos:

* Clone ou baixe o **repositório**: <https://github.com/g12-4soat/tech-lanches>
* Abra a pasta via linha de comando. Ex.: ***c:\> cd “c:/projetos/tech-lanches”***
* Da pasta raiz aberta execute “***docker-compose up***”

# **Setup do Projeto e Seeds**

O projeto conta com alguns valores pré-definidos já no seu início. Você pode encontrar alguns clientes, produtos e pedidos previamente cadastrados para melhor interação e entendimento da aplicação e de seus fluxos, mas você terá total liberdade para interagir com os registros da maneira que quiser baseado nas suas preferências.

Todas as interações demonstradas na seção “**Fluxos da Aplicação**” poderão ser executadas via Postman ou Swagger a depender de sua preferência.

Para visualização dos dados salvos em bancos de dados, utilize a ferramenta de sua preferência.

# **Fluxos da Aplicação**

# Geral

Todos os fluxos e interações com a aplicação partem de requisições REST feitas para a API. O serviço que roda em *background* é executado de forma automática com um intervalo mínimo de 5 (cinco) segundos caso não haja pedidos na fila para serem preparados.

# Compromisso de dados

Os dados utilizados nas imagens de demonstração são meramente ilustrativos e não tem fiel relação com dados reais, foram gerados de forma aleatória para atender a operabilidade da aplicação.

# Erros, Validações e Tratativas

Todos os endpoints da API estão utilizando tratativas de erros para as validações dos dados enviados. Requisições que ferem regras de negócio ou tem algum conflito de validação, retornarão um *status code* ***400 – Bad Request.***

# Contextos

# **Contexto: Identificação**

Como primeira interação do totem com os serviços da aplicação, existem alguns cenários onde um cliente pode interagir com o totem para a realização do pedido, sendo eles:

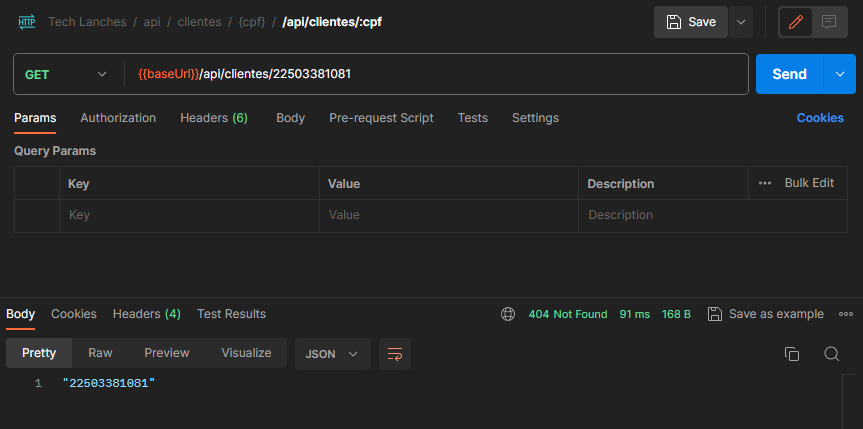
#### **Cliente não se identifica informando o CPF**

Caso o cliente não queira se identificar, vai ser direcionado para o contexto de checkout automaticamente, não é necessária mais nenhuma interação.

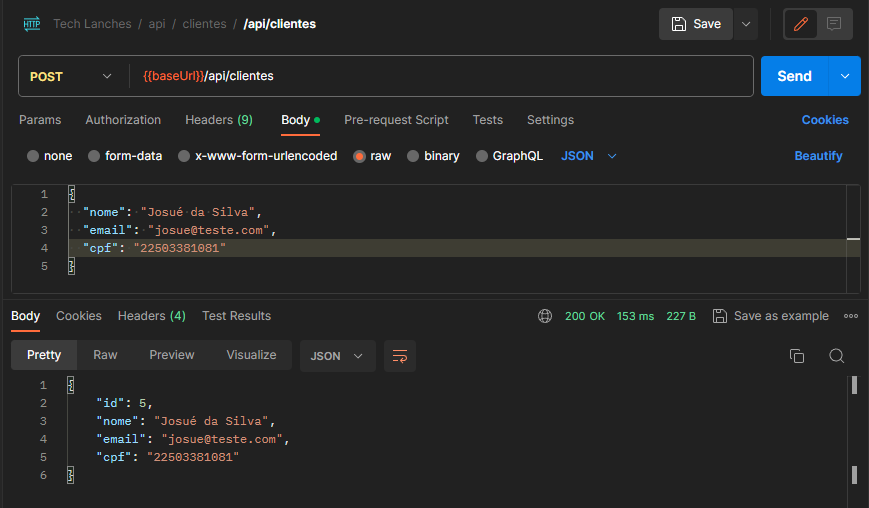
#### **Cliente se identifica informando o CPF**

Nesse cenário, caso o cliente já tenha cadastro na aplicação, segue para o contexto de checkout. Caso o cliente não exista na base de dados e for a primeira vez fazendo um pedido no qual ele se identifica por CPF, ele realizará o cadastro informando Nome, Email e CPF. O totem tentará consultar o CPF e caso não encontre (retorno *404 - NotFound*), será mostrada a tela de cadastro para o cliente informar os seus dados.

#### Cliente não existente



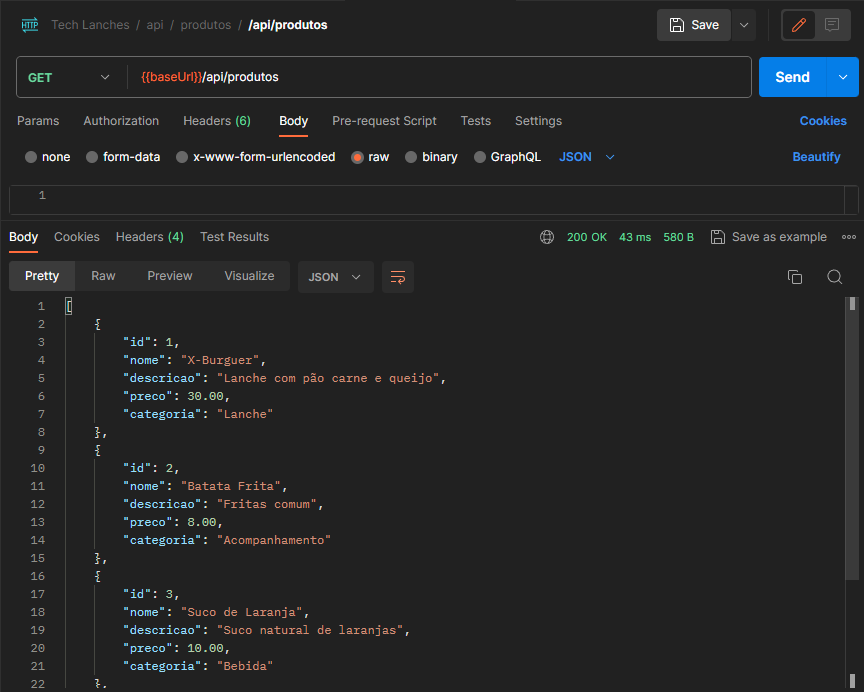
#### Cadastro de Cliente



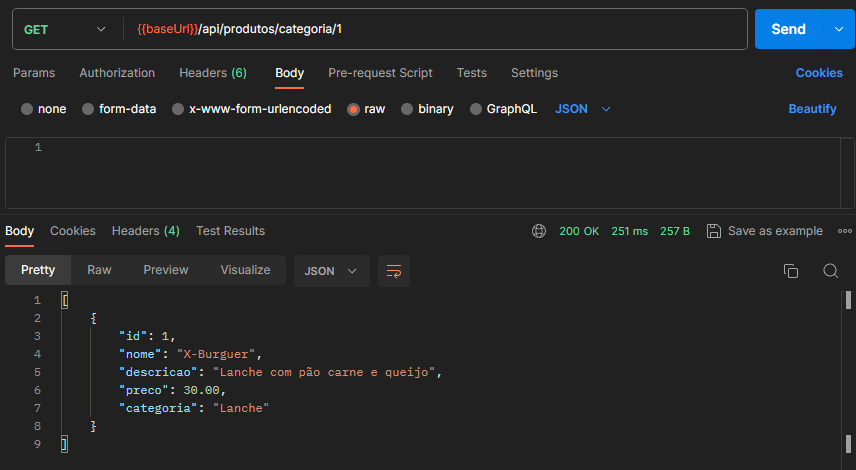
# **Contexto: Checkout**

Nesse contexto, o cliente, identificado ou não, poderá interagir para realizar o seu pedido escolhendo os produtos disponíveis. O totem poderá consumir as informações de pedido via API e o cliente visualiza todos os produtos disponíveis para consumo podendo:

#### Buscar todos os produtos

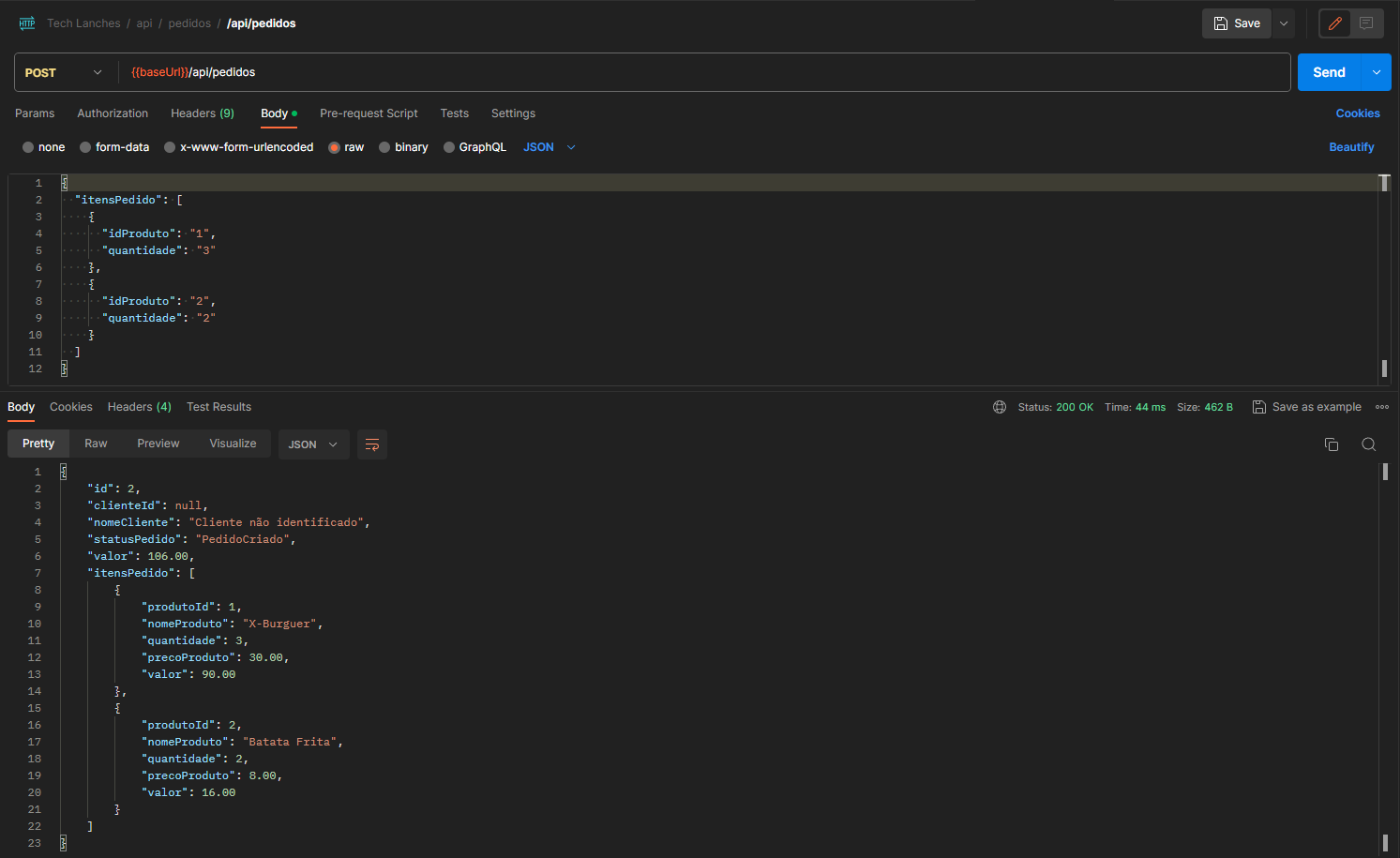


#### Buscar produtos por categoria

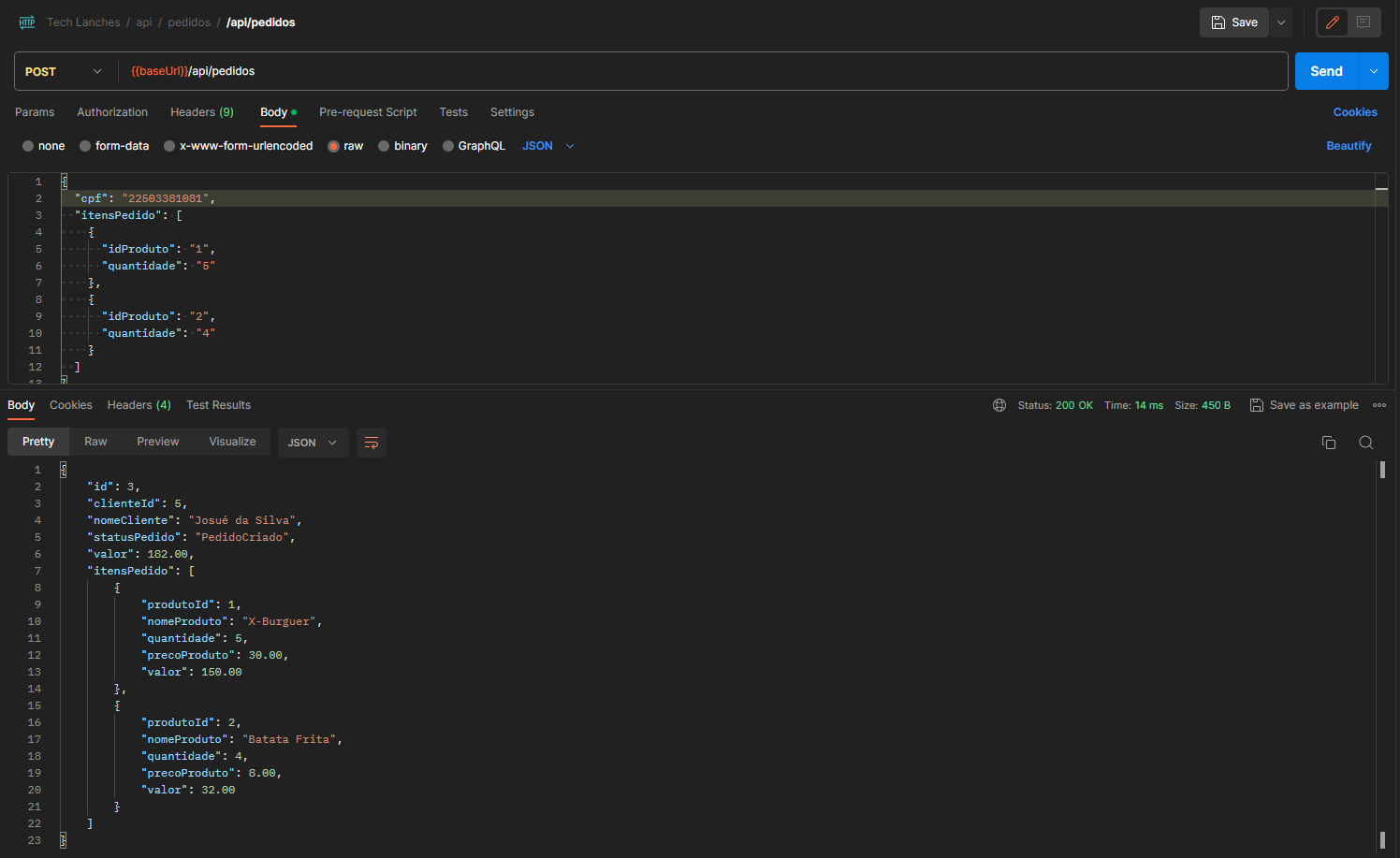


Feito isso, o cliente monta pedido selecionando os produtos e informando a respectiva quantidade, e envia o pedido para checkout.

#### Pedido com cliente não identificado

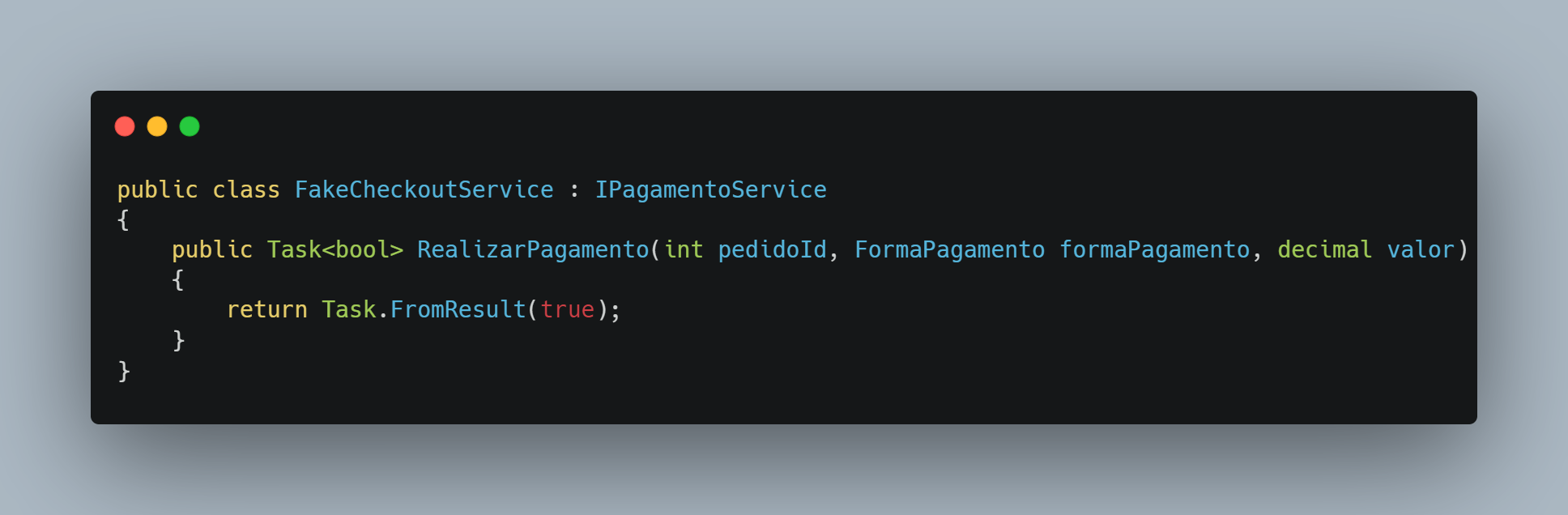


#### Pedido com cliente identificado



# **Contexto: Pagamento**

O cliente realiza o pagamento do valor do pedido via QRCode do Mercado Pago e uma vez aprovado, o pedido é enviado para a cozinha para ser realizado o seu preparo. No presente momento, como solicitado pelos requisitos dos entregáveis, o pagamento está sendo realizado utilizando um *fake checkout,* que retorna o sucesso do pagamento de forma automática sem a necessidade de se conectar a um serviço de pagamento real.



# **Contexto: Preparo**

Com o pagamento realizado e o pedido criado, ele é cadastrado com o status "PedidoCriado", é enviado para o preparo na cozinha, e é automaticamente colocado na fila de pedidos para preparo. Neste momento, o serviço FilaPedido  
busca e ordena todos os pedidos com o status "PedidoCriado" e “PedidoEmPreparacao” de forma crescente pelo menor "Id", de forma que a prioridade sempre será dos pedidos que chegaram primeiro. Os pedidos que atendem esses requisitos se encontram dentro da *view* “*FilaPedidos*” que se encontra na base de dados. O serviço pega o primeiro pedido da lista ordenada e tendo o próximo pedido a ser preparado, altera o status do pedido para "PedidoEmPreparacao".

#### Pedido em preparação

Atualmente, o serviço FilaPedido está configurado para simular um tempo de preparo fictício, sendo ele em torno de 20 (vinte) segundos. Após o término do tempo de preparo, o serviço altera o status do pedido para "PedidoPronto", possibilitando assim, a retirada do pedido que se encontra pronto.

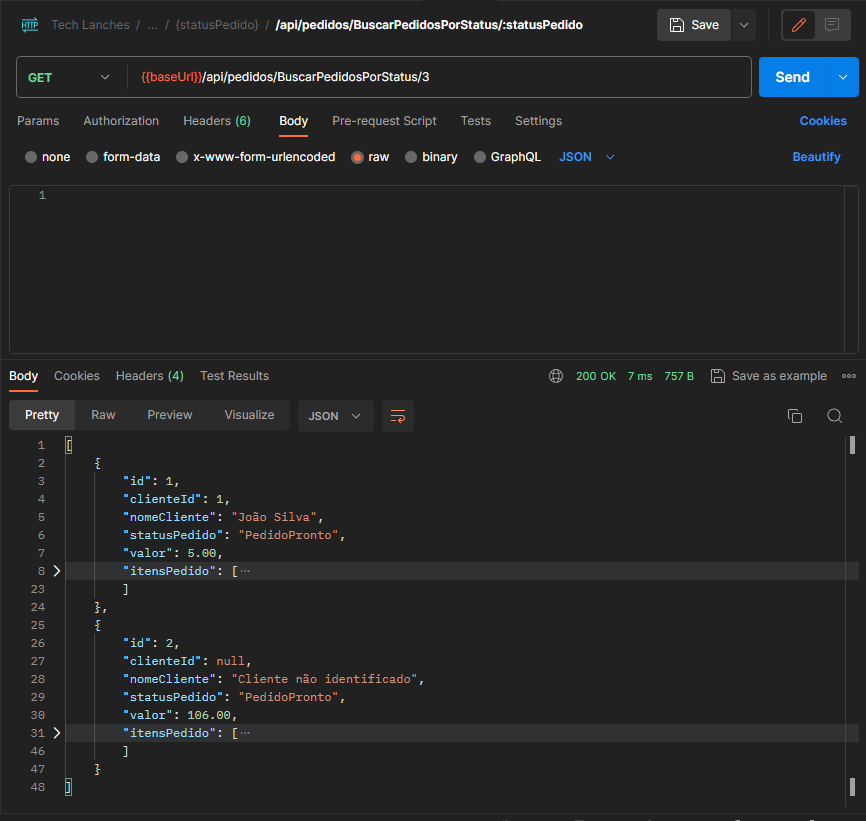
#### Pedido pronto

Terminado o preparo, o pedido pode ser consultado via API. Os pedidos irão aparecer na tela de pedidos prontos da lanchonete e um funcionário deve informar que o pedido está pronto baseado na lista atualizada que é retornada pelo sistema.

# **Contexto: Retirada/Finalização**

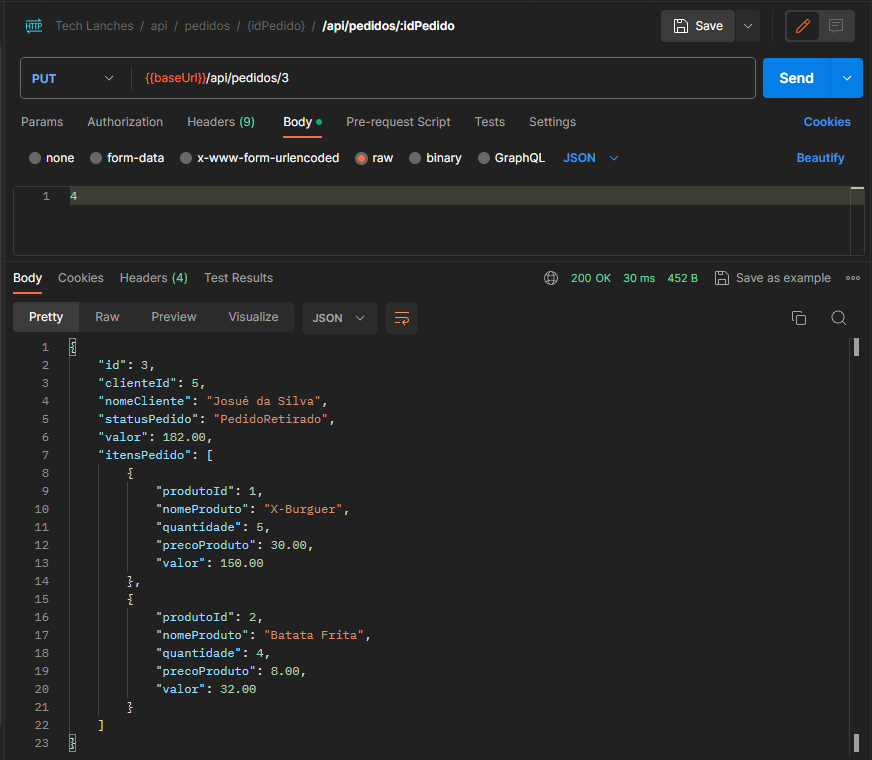
É possível consultar os pedidos prontos para retirada com a seguinte requisição para a API:

#### Pedidos por Status



Estando pronto para retirada, é possível realizar uma requisição para a retirada do pedido no momento que o cliente solicita a retirada do mesmo.

#### **Pedido** Retirado



Após a retirada do pedido pelo cliente, o funcionário atualiza o pedido, alterando o seu status para "PedidoRetirado" e o pedido chega ao fim do seu fluxo.