para testar o programa corretamente existem duas maneiras

Programas necessários:

VsCode

Docker Desktop

Postman

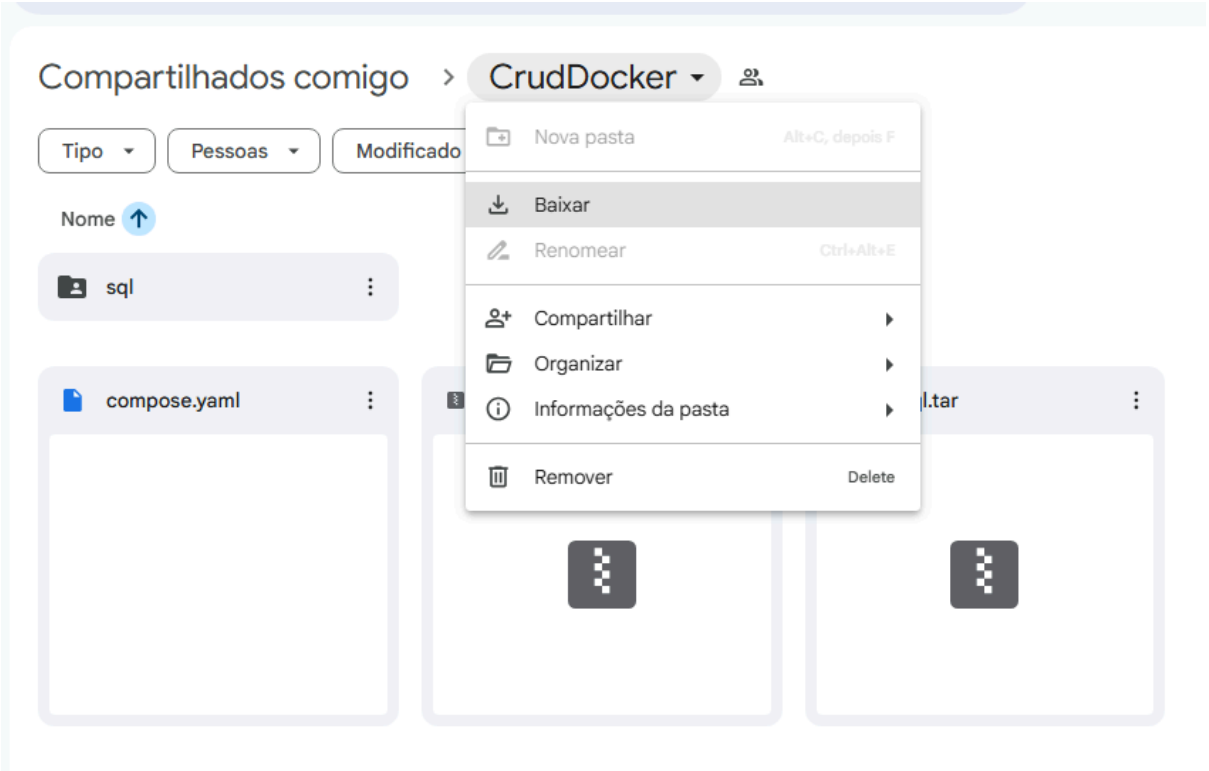
Pelo VSCode:
Baixe o winRAR CrudDocker e extraia a pasta para o local desejado.

Assim que o download concluir clique com o botão direito do mouse e aperte em mostrar na pasta:

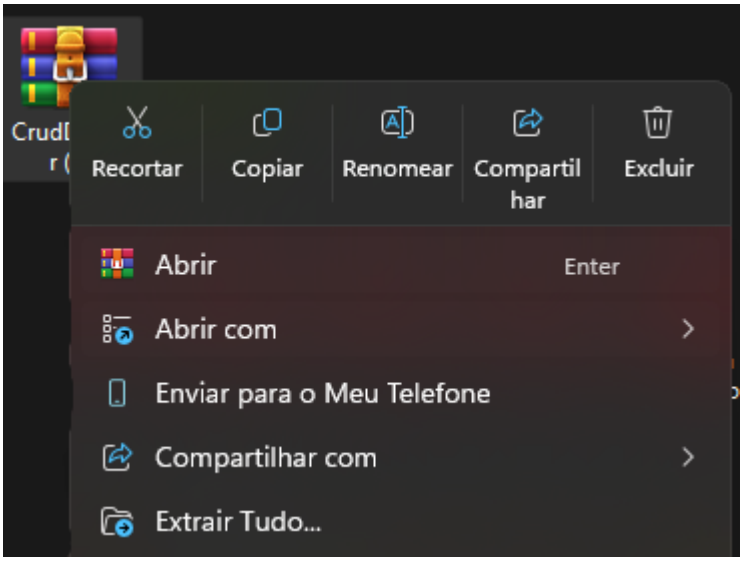
https://drive.google.com/drive/folders/1WKdm_hSwJrQM3on4u--kKQsKBx6AbfON?usp=drive_link

eu tentei compartilhar de diferentes maneiras os arquivos mas o github barrou…

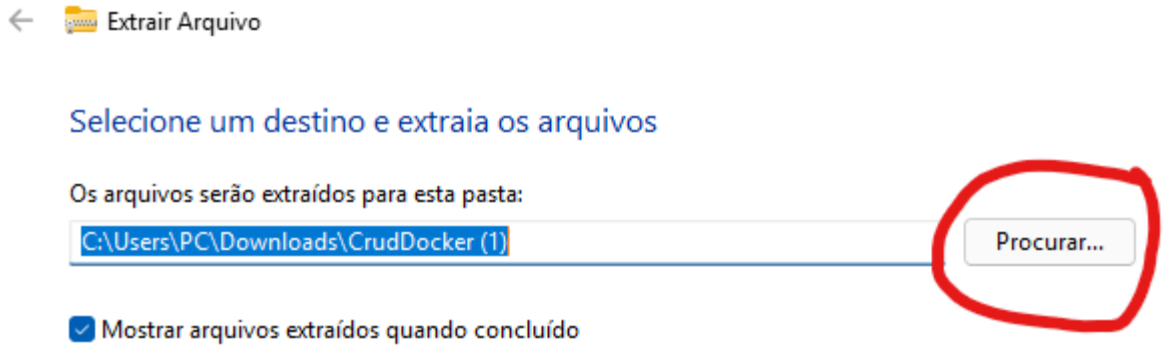
faça o download do arquivo no google drive



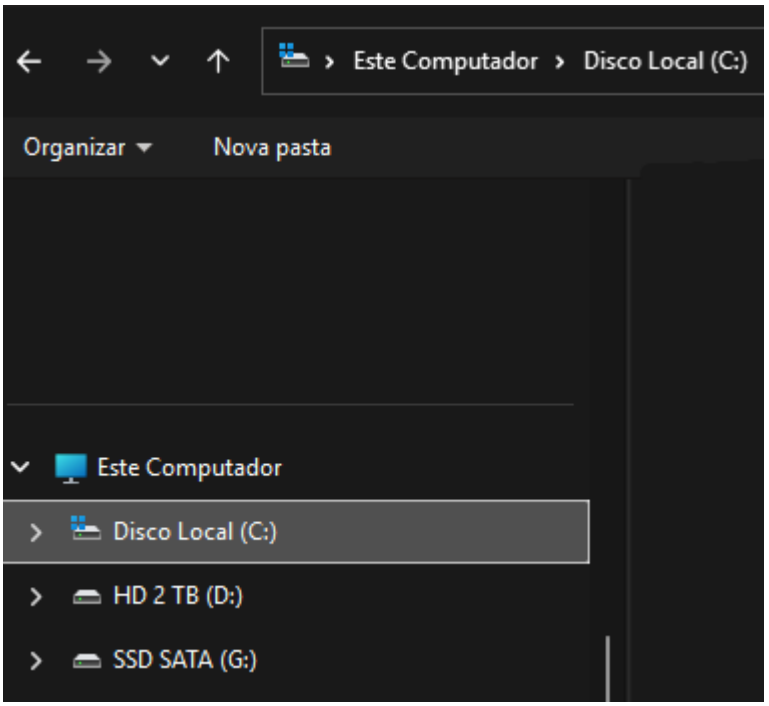
após o download Clique em extrair tudo



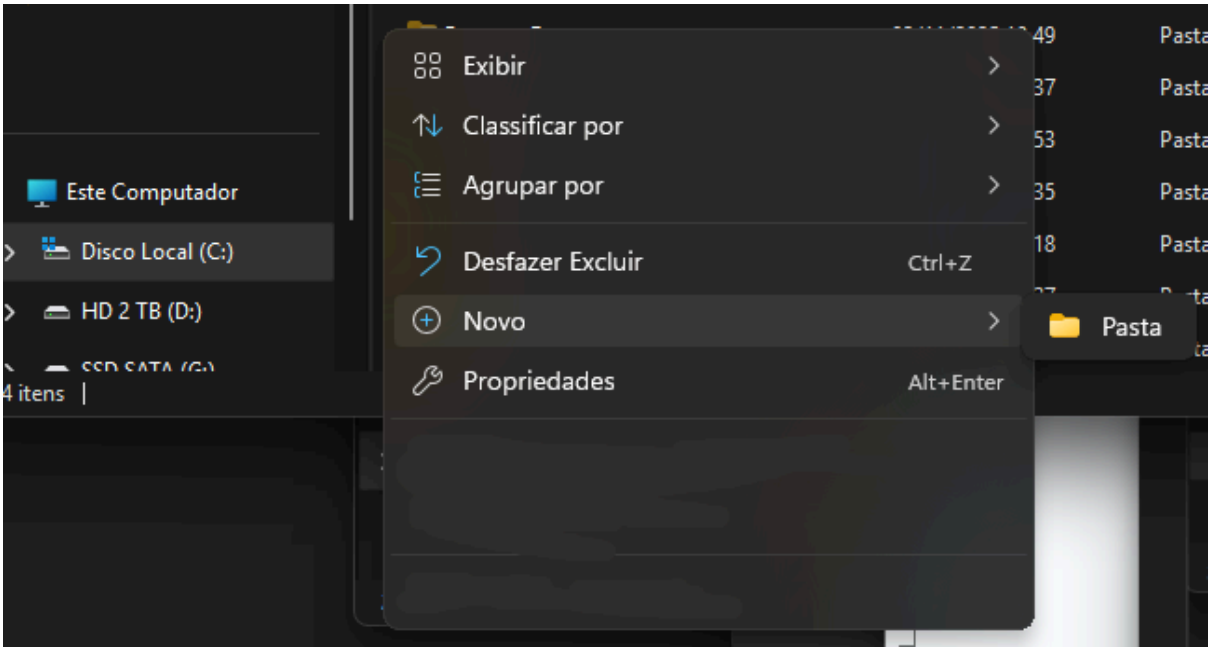
IMPORTANTE!
RECOMENDADO: Extraia na pasta mais próxima do seu disco rígido!
Coloque em C:\



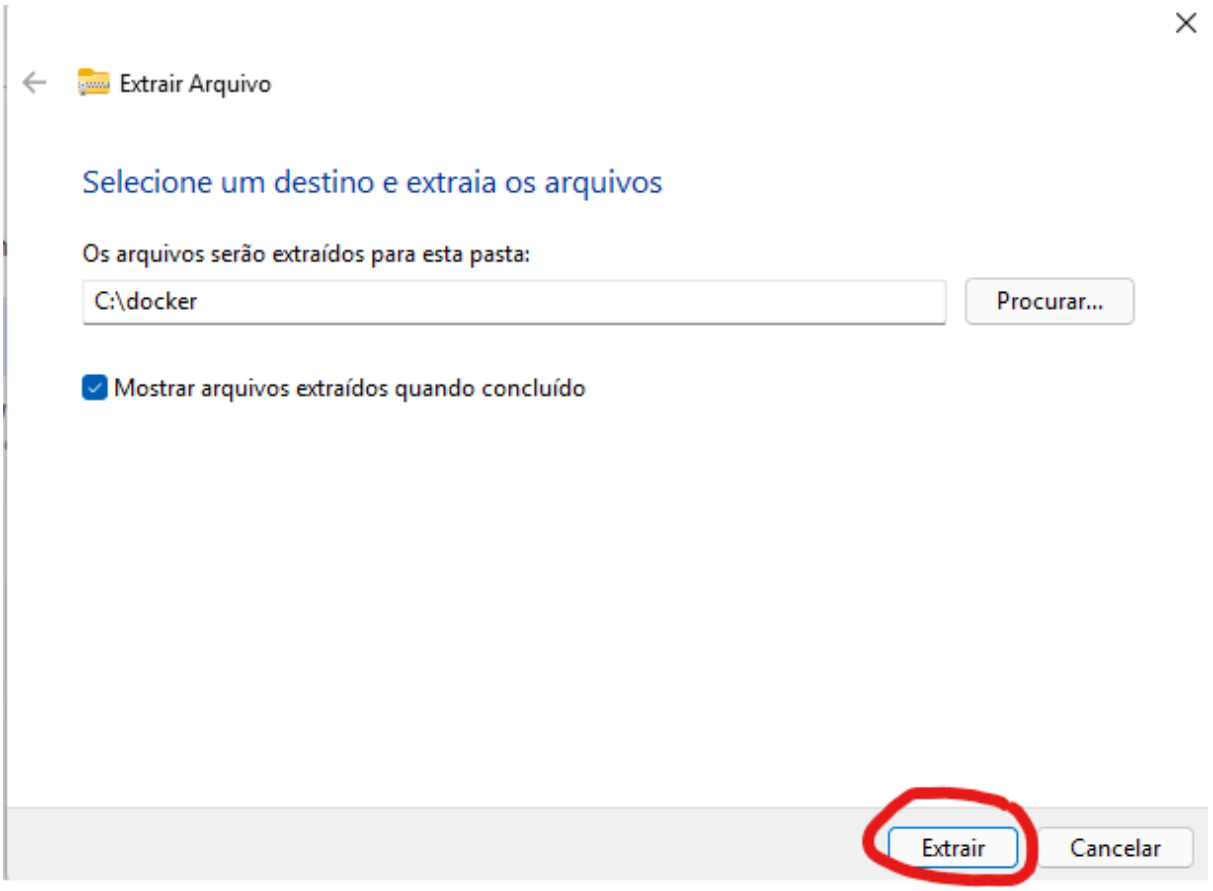
Vá para seu disco Rígido no seu computador



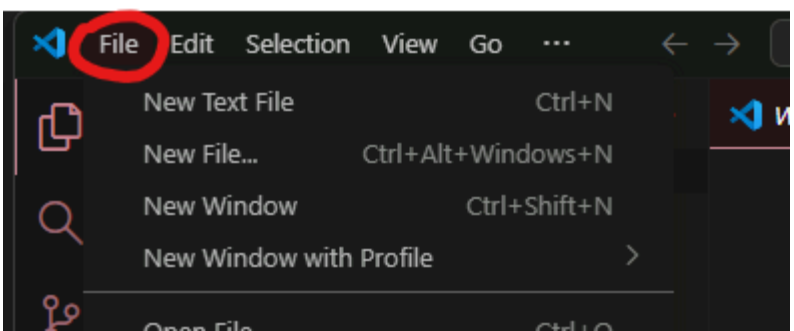
Crie uma pasta nova dentro dele com o nome desejado: (RECOMENDADO: “docker”)



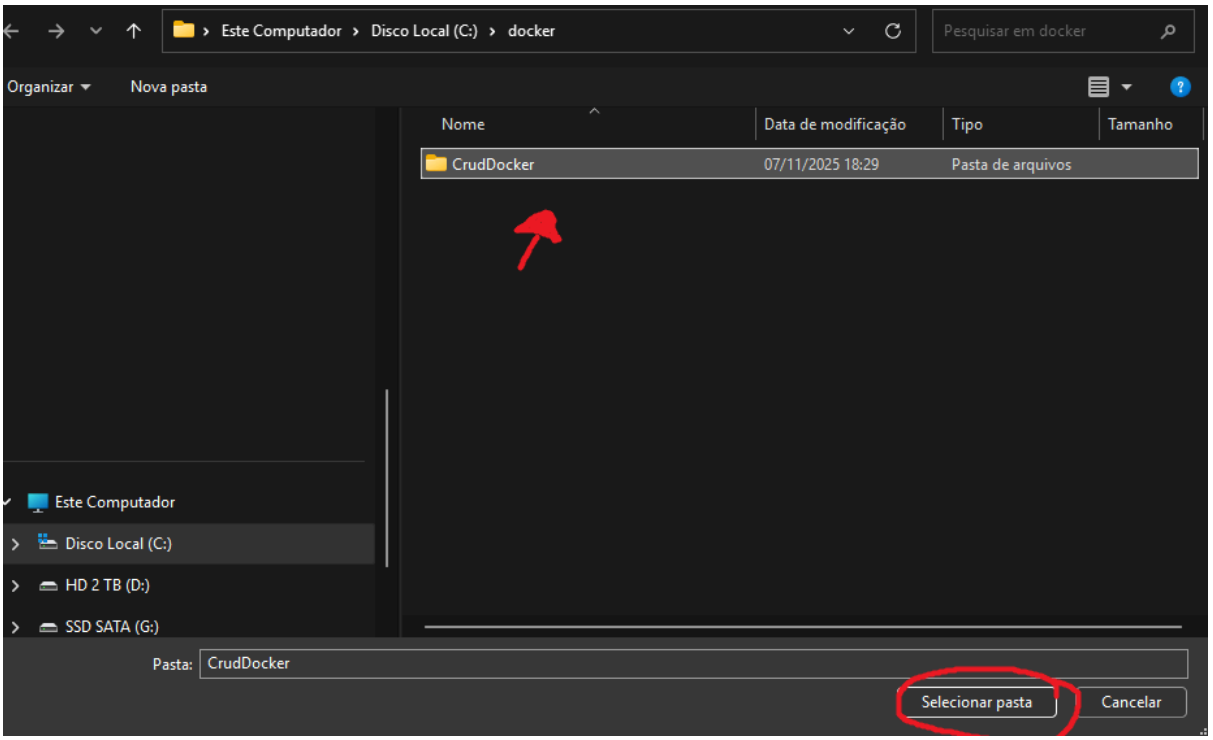
Selecione a pasta criada e termine a extração do arquivo.



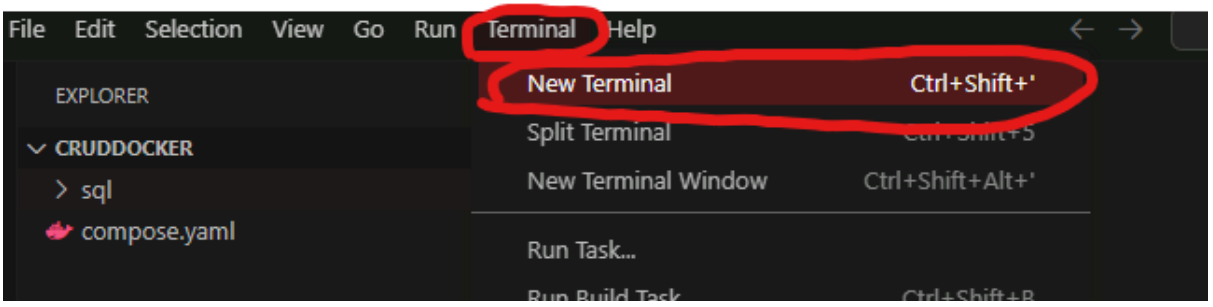
Abra o Visual Studio Code e abra a pasta docker que está com os arquivos extraídos baixados.



Selecione a pasta CrudDocker



Abra um novo terminal:



Essa parte pode demorar um pouco! execute os comandos:
docker load -i crud.tar
docker load -i mysql.tar

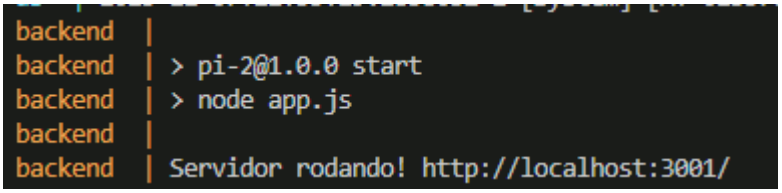
```
PS C:\docker\cruddocker> docker load -i crud.tar
Loaded image: cruddocker-backend:latest
PS C:\docker\cruddocker> docker load -i mysql.tar
Loaded image: mysql:8.0
```

depois que as imagens foram carregadas, execute o comando:

docker compose up --build

e espere até que tudo seja executado

IMPORTANTE!
Procure a parte no terminal que está escrito da seguinte maneira:

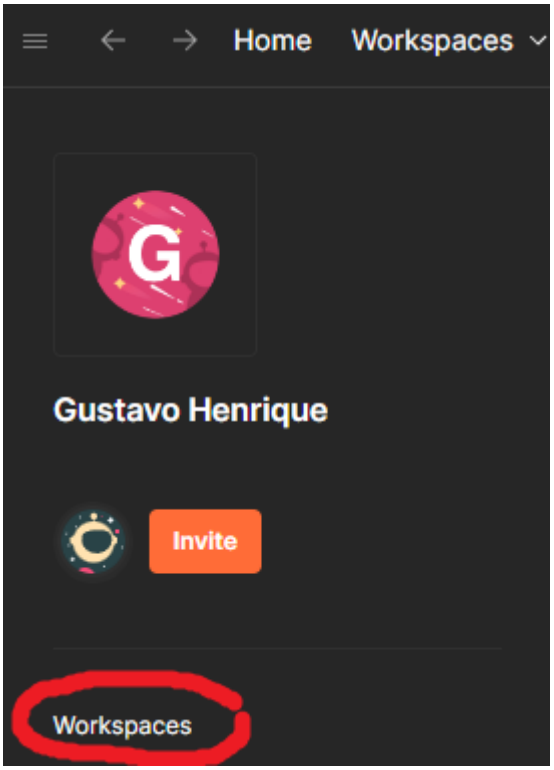


segure ctrl e clique em cima do http://localhost:3001/

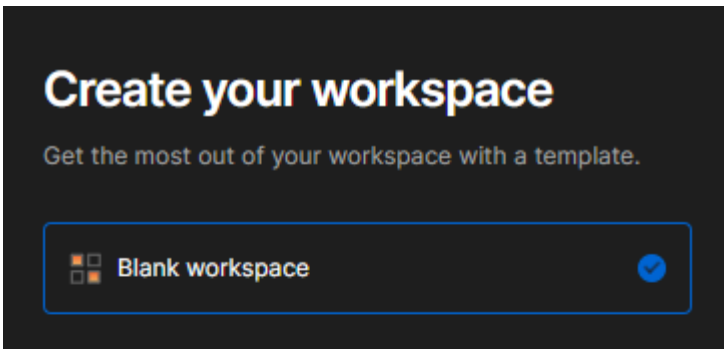
isso abrirá a guia do google com o servidor local e se aparecer a seguinte tela, significa que está funcionando corretamente:

Abra o Postman

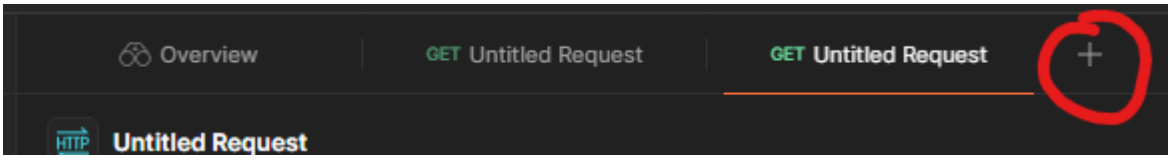
vá em workspace



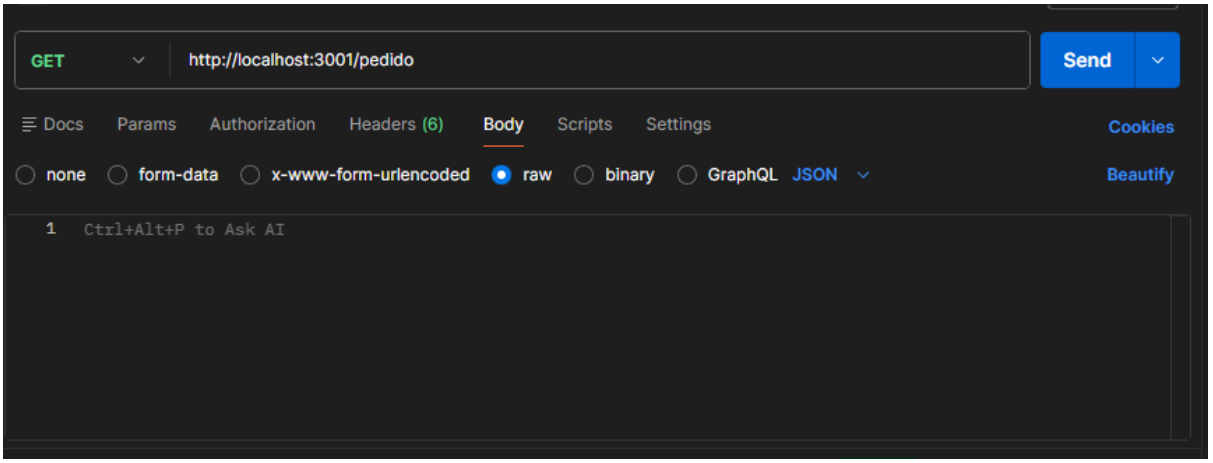
Crie uma workspace



crie uma guia nova de request



coloque os links e os bodys em formato raw para poder fazer os testes:



Pedidos:

Criar pedido: `POST http://localhost:3001/pedido`

Listar pedidos: `GET http://localhost:3001/pedido`

Buscar pedido por ID: `GET http://localhost:3001/pedido/:id`

Atualizar pedido: `PUT http://localhost:3001/pedido/:id`

Deletar pedido: `DELETE http://localhost:3001/pedido/:id`

Corpo criar pedido:

```
{
  "id_consumidor": (id do consumidor existente),
  "status_pedido": "aberto",
  "valor_total": (valor inicial do pedido, geralmente 0)
}
```

Corpo atualizar pedido:

```
{
  "status_pedido": "(aberto ou fechado)",
  "valor_total": (novo valor total, ex: 35.50)
}
```

Itens:

Criar item: `POST http://localhost:3001/item-pedido`

Listar itens: `GET http://localhost:3001/item-pedido`

Buscar item por ID: `GET http://localhost:3001/item-pedido/:id_item`

Atualizar item: `PUT http://localhost:3001/item-pedido/:id_item`

Deletar item: `DELETE http://localhost:3001/item-pedido/:id_item`

Corpo criar item (como está no teu controller atual):

```
{
  "id_pedido": (id do pedido existente),
  "descricao": "(nome do produto ou item pedido)",
  "quantidade": (número inteiro, ex: 2),
  "preco_unitario": (valor decimal, ex: 10.00)
}
```

Corpo criar item (versão correta conforme o banco de dados):

```
{
  "id_pedido": (id do pedido existente),
  "id_item": (id do item do cardápio),
  "quantidade": (número inteiro, ex: 2),
  "preco_unitario": (valor decimal, ex: 10.00)
}
```

Pagamentos:

Criar pagamento: `POST http://localhost:3001/pagamento`

Listar pagamentos: `GET http://localhost:3001/pagamento`

Buscar pagamento por ID: `GET http://localhost:3001/pagamento/:id_pagamento`

Atualizar pagamento: `PUT http://localhost:3001/pagamento/:id_pagamento`

Deletar pagamento: `DELETE http://localhost:3001/pagamento/:id_pagamento`

Corpo criar pagamento:

```
{
  "id_pedido": (id do pedido existente),
  "tipo_pagamento": "(debito | credito | app | pix | dinheiro)",
  "valor_pago": (valor decimal, ex: 25.00)
}
```

Corpo atualizar pagamento:

```
{
  "tipo_pagamento": "(debito | credito | app | pix | dinheiro)",
  "valor_pago": (novo valor decimal, ex: 26.00)
}
```

Consumidor:

Criar consumidor: `POST http://localhost:3001/consumidor`

Listar consumidores: `GET http://localhost:3001/consumidor`

Buscar consumidor por ID: `GET http://localhost:3001/consumidor/:id_consumidor`

Atualizar consumidor: `PUT http://localhost:3001/consumidor/:id_consumidor`

Deletar consumidor: `DELETE http://localhost:3001/consumidor/:id_consumidor`

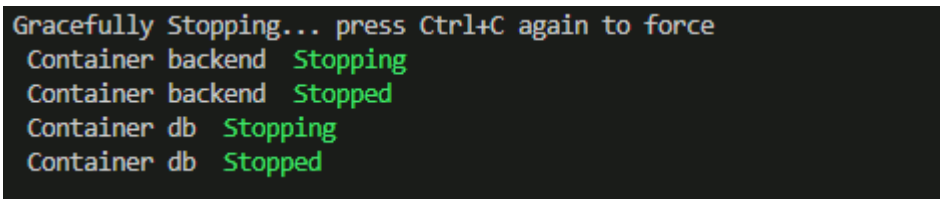
```
{
  "nome_consumidor": "(nome completo do aluno)",
  "senha": "(senha em texto simples)",
  "idade": (idade em número inteiro),
  "tipo_consumidor": "aluno",
  "ra": (número único de registro acadêmico, ex: 12345)
}
```

```
{
  "nome_consumidor": "(novo nome, opcional)",
  "senha": "(nova senha, opcional)",
  "idade": (nova idade, opcional),
  "tipo_consumidor": "(aluno | professor | visitante, opcional)",
  "ra": (novo RA se tipo for aluno)
}
```

Finalização:

assim que finalizar os testes coloque o seguinte comando no terminal:

ctrl + c no terminal 3 vezes para fechar o servidor



use
docker compose down -v
para fechar os contêineres abertos e apagar o volume do bd que ficou gravado.