ChatBot
Aula 11

Prof. Me Daniel Vieira

SENAI

Agenda

1 - Criando API com LLM integrada
2 - FastAPI
3 - Estrutura do projeto
4 - Testes com Insomnia

Estrutura do projeto API com Chatbot

```
chatbot_api/
│
├── main.py
├── serenatto_bot.py
├── documentos/
│   └── serenatto.pdf
├── chroma_db/   ← será criado ao rodar o código
└── requirements.txt
```

Código serenatto_bot.py

```
import os
from typing import List
from llama_index.core import SimpleDirectoryReader, StorageContext,
VectorStoreIndex
from llama_index.core.node_parser import SentenceSplitter
from llama_index.embeddings.huggingface import HuggingFaceEmbedding
from llama_index.vector_stores.chroma import ChromaVectorStore
from llama_index.llms.groq import Groq
from llama_index.core.memory import ChatSummaryMemoryBuffer
import chromadb
from tempfile import TemporaryDirectory
from PyPDF2 import PdfReader
```

Código serenatto_bot.py

```
class ChromaEmbeddingWrapper:
    def __init__(self, model_name: str):
        self.model = HuggingFaceEmbedding(model_name=model_name)

    def __call__(self, input: List[str]) -> List[List[float]]:
        return self.model.embed_documents(input)
```

Código serenatto_bot.py

```
class SerenattoBot:
    def __init__(self):
        self.embed_model =
HuggingFaceEmbedding(model_name='intfloat/multilingual-e5-large')
        self.embed_model_chroma =
ChromaEmbeddingWrapper(model_name='intfloat/multilingual-e5-large')

        chroma_client = chromadb.PersistentClient(path='./chroma_db')
        collection_name = 'documentos_serenatto'
        chroma_collection = chroma_client.get_or_create_collection(
            name=collection_name,
            embedding_function=self.embed_model_chroma
        )
```

Código serenatto_bot.py

```
self.vector_store = ChromaVectorStore(chroma_collection=chroma_collection)
        self.storage_context =
StorageContext.from_defaults(vector_store=self.vector_store)

        self.llms = Groq(model='llama3-70b-8192',
api_key='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')

        self.document_index = None
        self.chat_engine = None

        self.carregar_pdf()
```

Código serenatto_bot.py

```
def carregar_pdf(self):
        with TemporaryDirectory() as tmpdir:
            pdf_path = "documentos/serenatto.pdf"
            text = ""
            reader = PdfReader(pdf_path)
            for page in reader.pages:
                text += page.extract_text() or ""
```

Código serenatto_bot.py

```
 with open(os.path.join(tmpdir, "temp.txt"), "w", encoding="utf-8") as f:
                    f.write(text)

                documentos = SimpleDirectoryReader(input_dir=tmpdir)
                docs = documentos.load_data()

                node_parser = SentenceSplitter(chunk_size=1200)
                nodes = node_parser.get_nodes_from_documents(docs)

                self.document_index = VectorStoreIndex(nodes,
storage_context=self.storage_context, embed_model=self.embed_model)
```

Código serenatto_bot.py

```
memory = ChatSummaryMemoryBuffer(llm=self.llms, token_limit=256)


            self.chat_engine = self.document_index.as_chat_engine(
                chat_mode='context',
                llm=self.llms,
                memory=memory,
                system_prompt='''Você é especialista em cafés da loja Serenatto.
Responda de forma simpática e natural sobre os grãos disponíveis.'''
            )
```

Código serenatto_bot.py

```
def responder(self, mensagem: str) -> str:
        if self.chat_engine is None:
            return "Erro: o bot ainda não está pronto."
        response = self.chat_engine.chat(mensagem)
        return response.response

    def resetar(self):
        if self.chat_engine:
            self.chat_engine.reset()
```

main.py

```
from fastapi import FastAPI, Body
from pydantic import BaseModel
from serenatto_bot import SerenattoBot

app = FastAPI(title="Serenatto Chatbot API")

# Instancia o bot na inicialização
bot = SerenattoBot()
```

main.py

```python
class MensagemRequest(BaseModel):
    mensagem: str

class MensagemResponse(BaseModel):
    resposta: str
@app.post("/conversar", response_model=MensagemResponse)
def conversar(request: MensagemRequest):
    resposta = bot.responder(request.mensagem)
    return MensagemResponse(resposta=resposta)

@app.post("/resetar")
def resetar():
    bot.resetar()
    return {"status": "Chat resetado com sucesso"}
```

main.py

```python
class MensagemRequest(BaseModel):
    mensagem: str

class MensagemResponse(BaseModel):
    resposta: str
@app.post("/conversar", response_model=MensagemResponse)
def conversar(request: MensagemRequest):
    resposta = bot.responder(request.mensagem)
    return MensagemResponse(resposta=resposta)

@app.post("/resetar")
def resetar():
    bot.resetar()
    return {"status": "Chat resetado com sucesso"}
```

Requirements.txt

```
fastapi
uvicorn
llama-index
PyPDF2
chromadb
transformers
huggingface_hub
```

Comando para executar a api

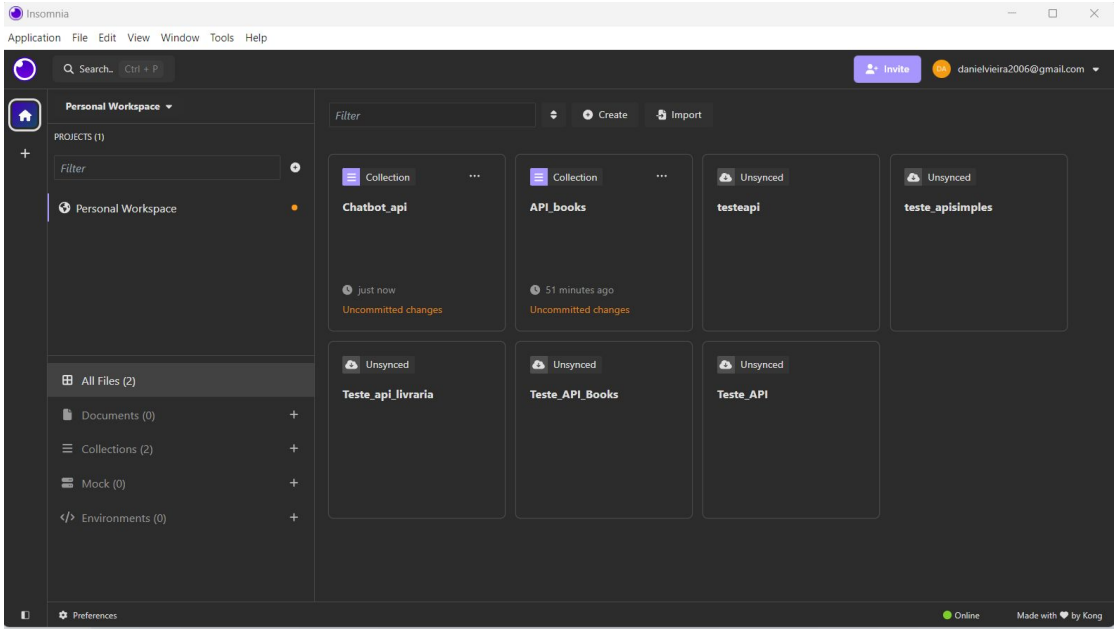
uvicorn main:app –reload

```
PS D:\SENAI\2025-1\ChatBot\chatbot_api> uvicorn main:app --reload
>>
INFO:     Will watch for changes in these directories: ['D:\\SENAI\\2025-1\\ChatBot\\chatbot_api']
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [720] using WatchFiles
INFO:     Started server process [17540]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     127.0.0.1:58451 - "POST /conversar HTTP/1.1" 200 OK
INFO:     127.0.0.1:58457 - "GET /docs HTTP/1.1" 200 OK
INFO:     127.0.0.1:58457 - "GET /openapi.json HTTP/1.1" 200 OK
INFO:     127.0.0.1:58477 - "POST /conversar HTTP/1.1" 200 OK
INFO:     127.0.0.1:58526 - "POST /conversar HTTP/1.1" 200 OK
INFO:     127.0.0.1:58733 - "POST /conversar HTTP/1.1" 200 OK
INFO:     127.0.0.1:58763 - "POST /conversar HTTP/1.1" 200 OK
INFO:     127.0.0.1:58771 - "POST /conversar HTTP/1.1" 200 OK
```

FastApi

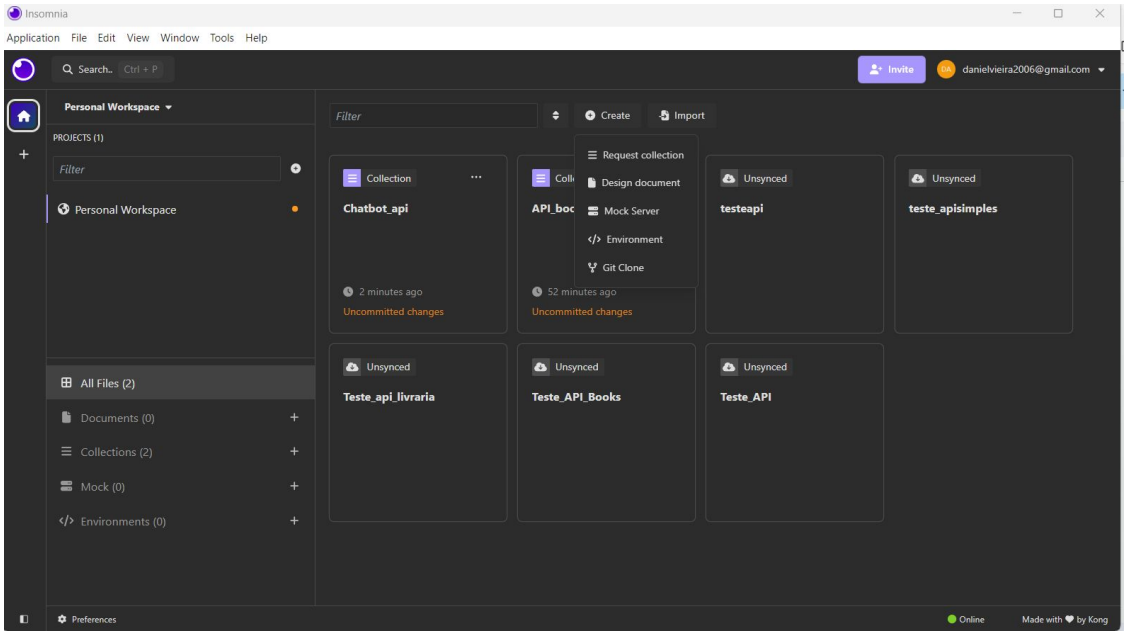

Serenatto Chatbot API
0.1.0
OAS 3.1
/openapi.json
default
POST
/conversar
Conversar
Parameters
Try it out
No parameters
Request body
required
application/json
Example Value
Schema
{
"mensagem": "string"
}
Responses
Code
Description
Links

Para testar a API será utilizado o Insomnia



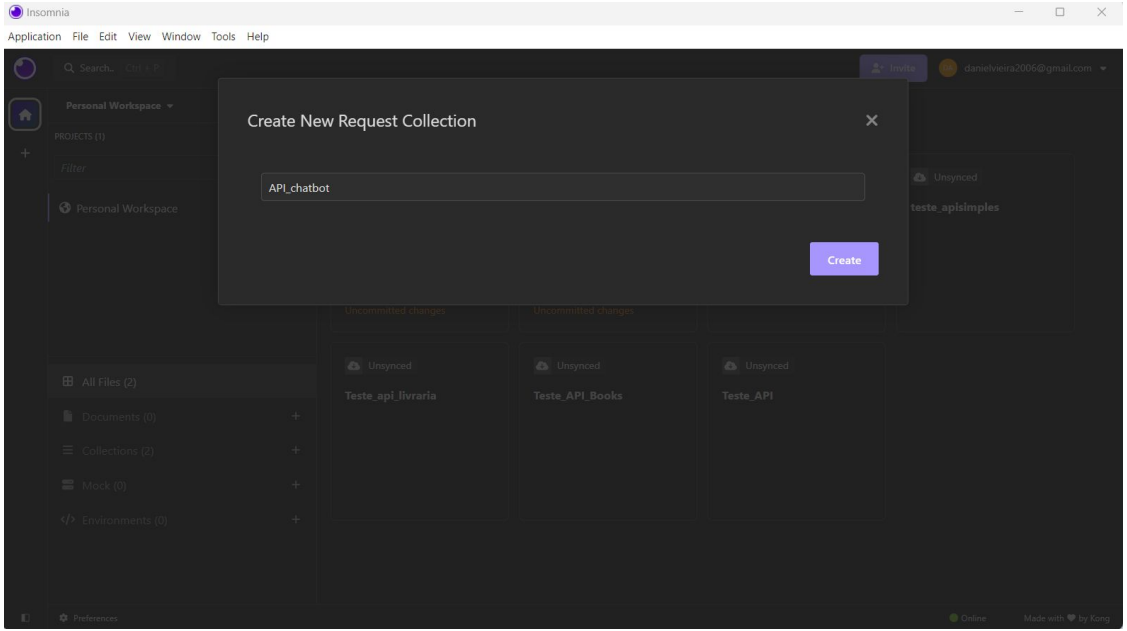
Para testar a API será utilizado o Insomnia

Clicar em create a request collection



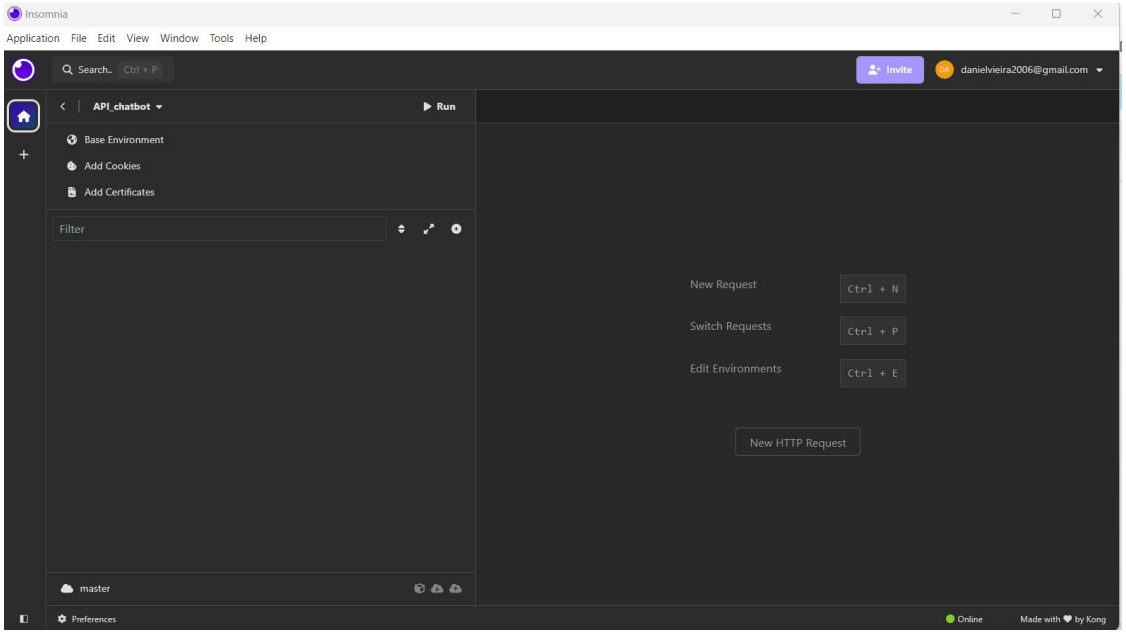
Para testar a API será utilizado o Insomnia

Clicar em create a request collection



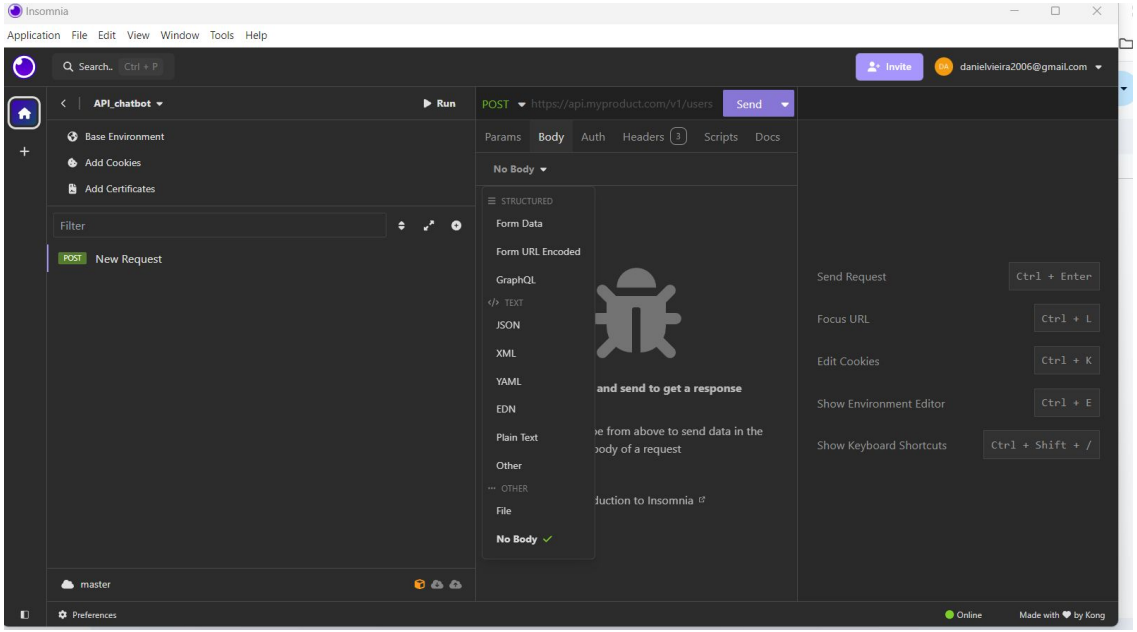
Para testar a API será utilizado o Insomnia

Clicar em New HTTP request



Para testar a API será utilizado o Insomnia

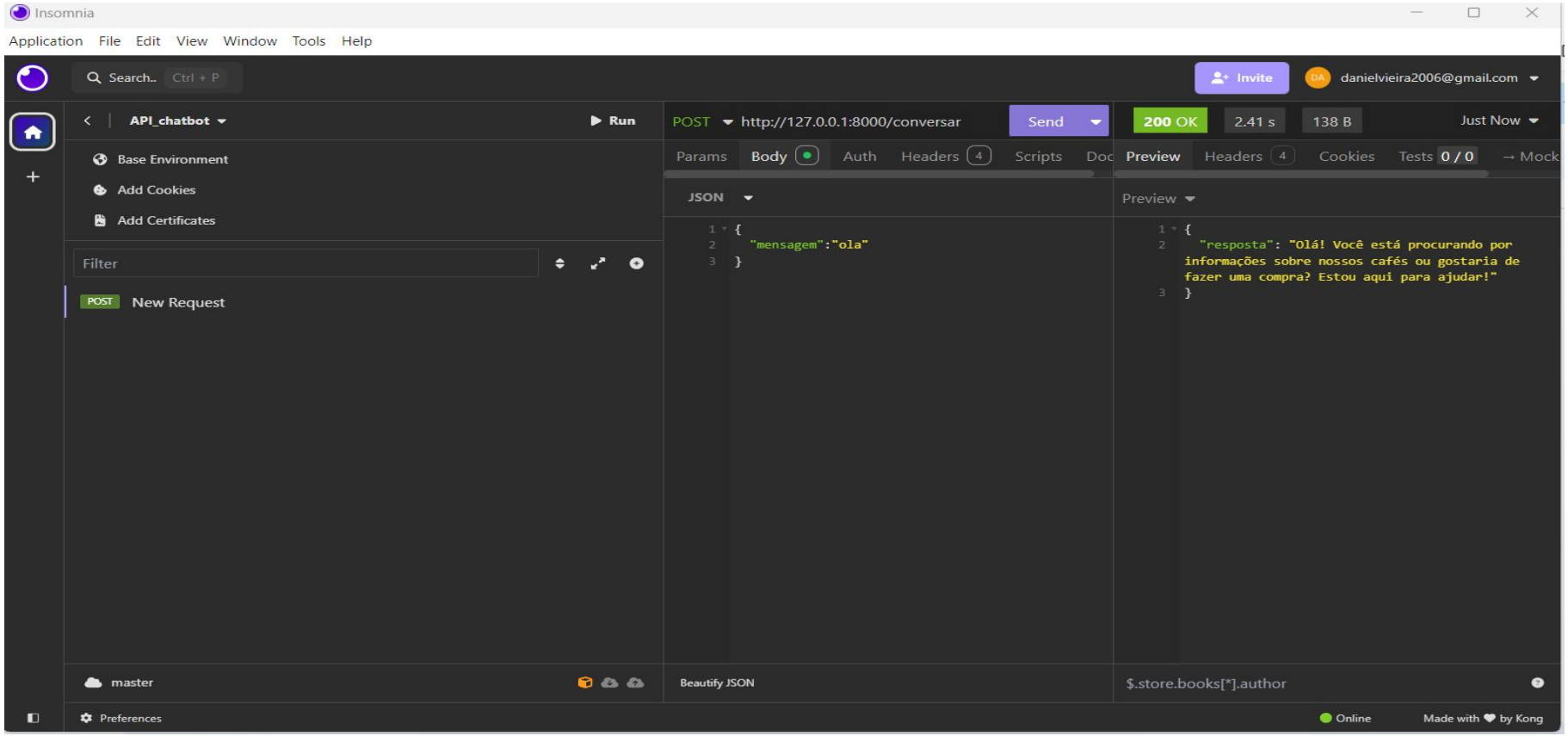
O método http escolher o método post e utilizar o ip http://127.0.0.1:8000/conversar



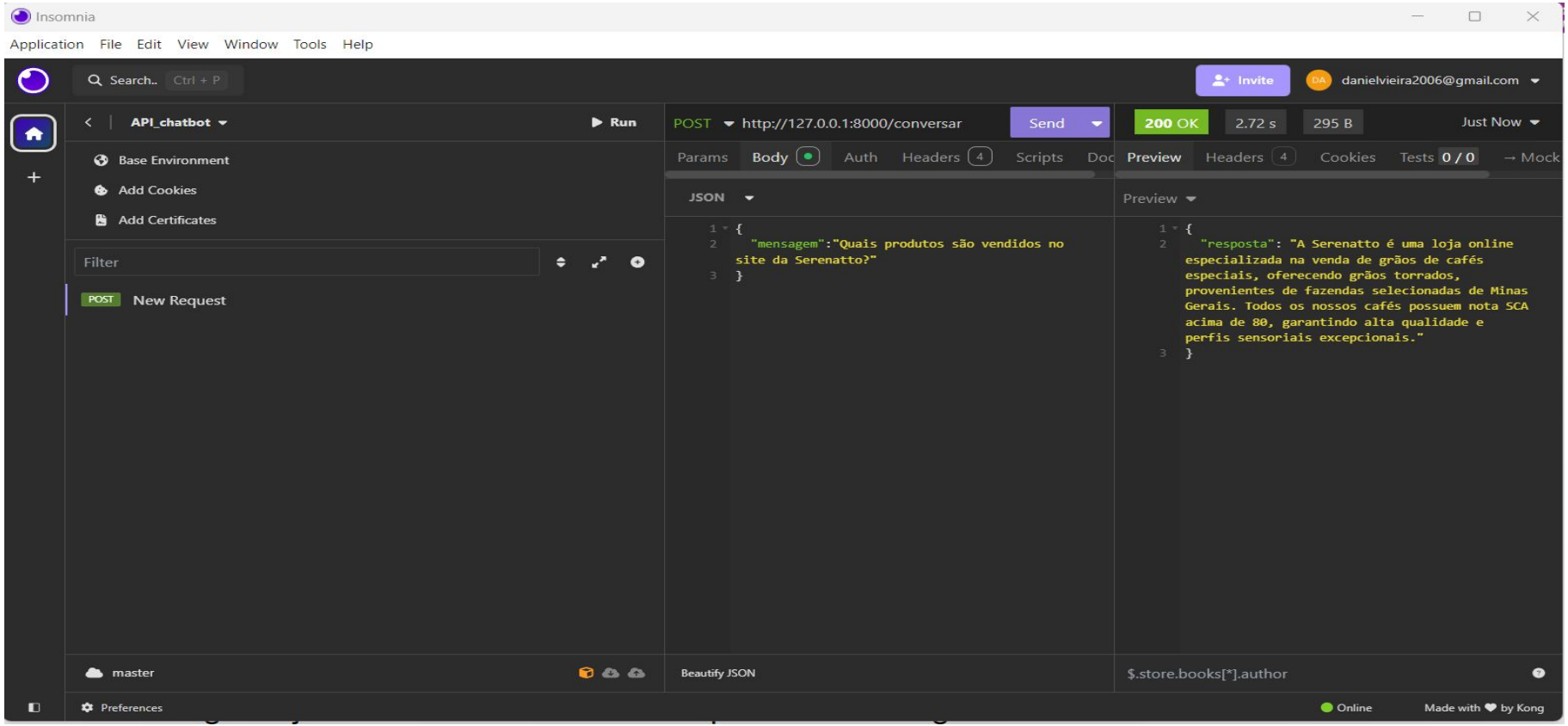
Para testar a API será utilizado o Insomnia

O método http escolher o método post e utilizar o ip http://127.0.0.1:8000/conversar

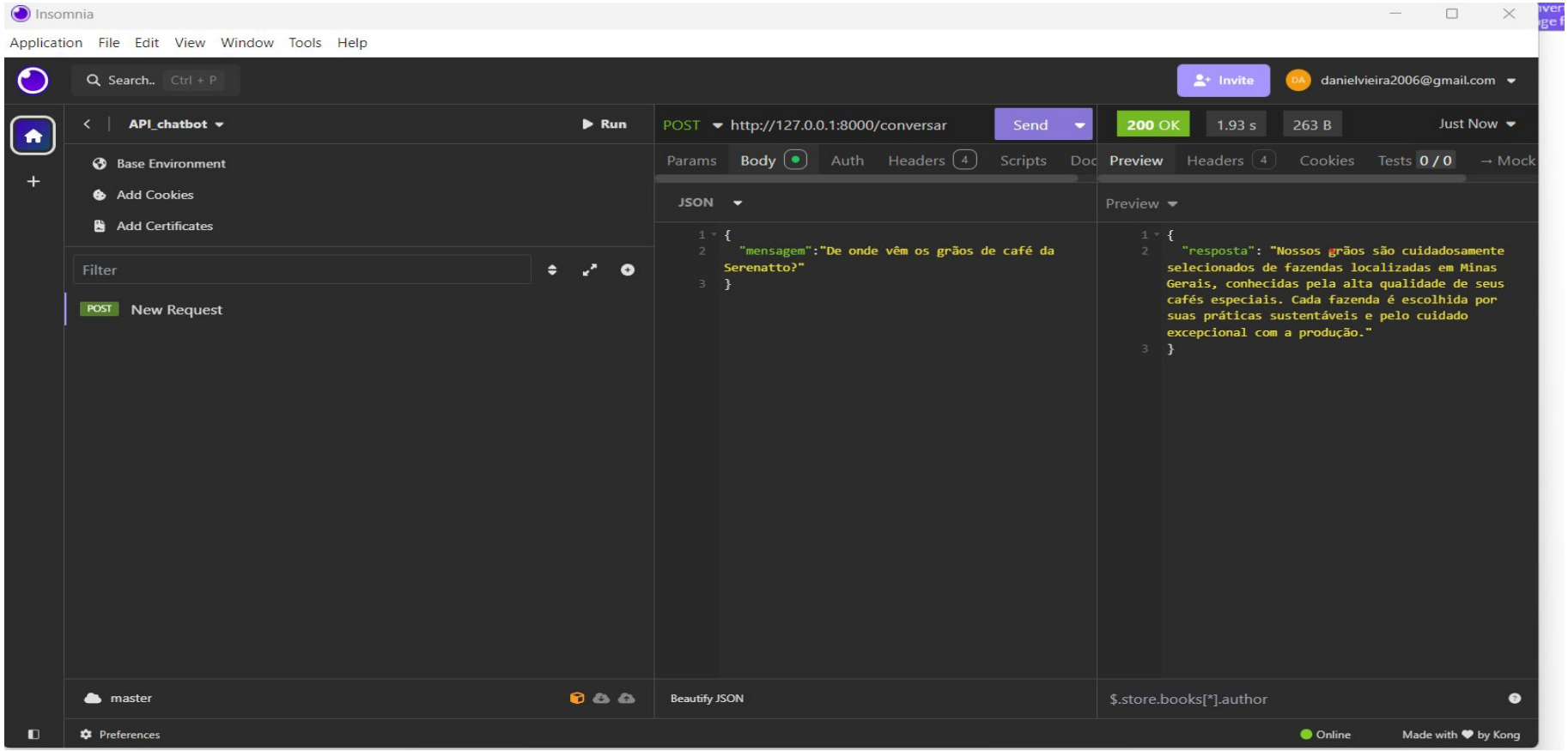
Para testar a API será utilizado o Insomnia



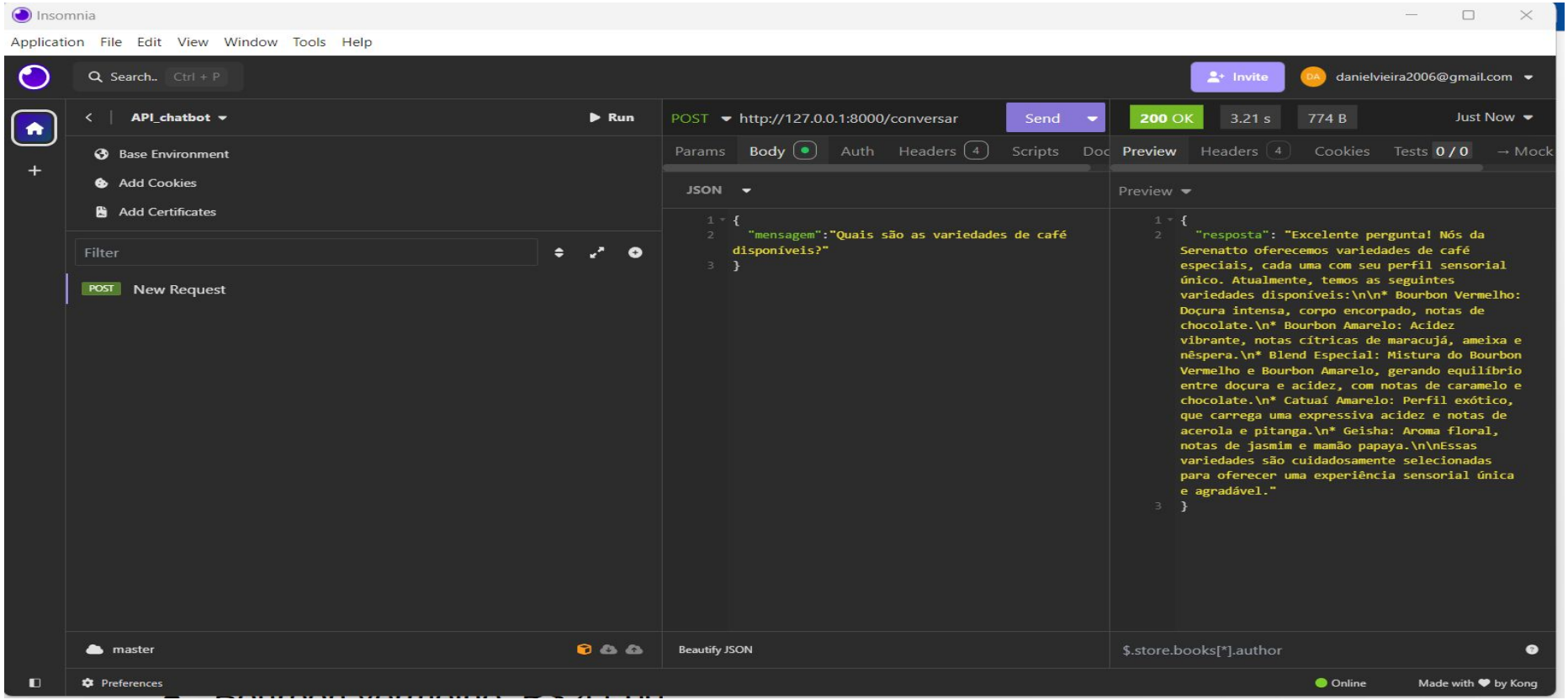
Para testar a API será utilizado o Insomnia



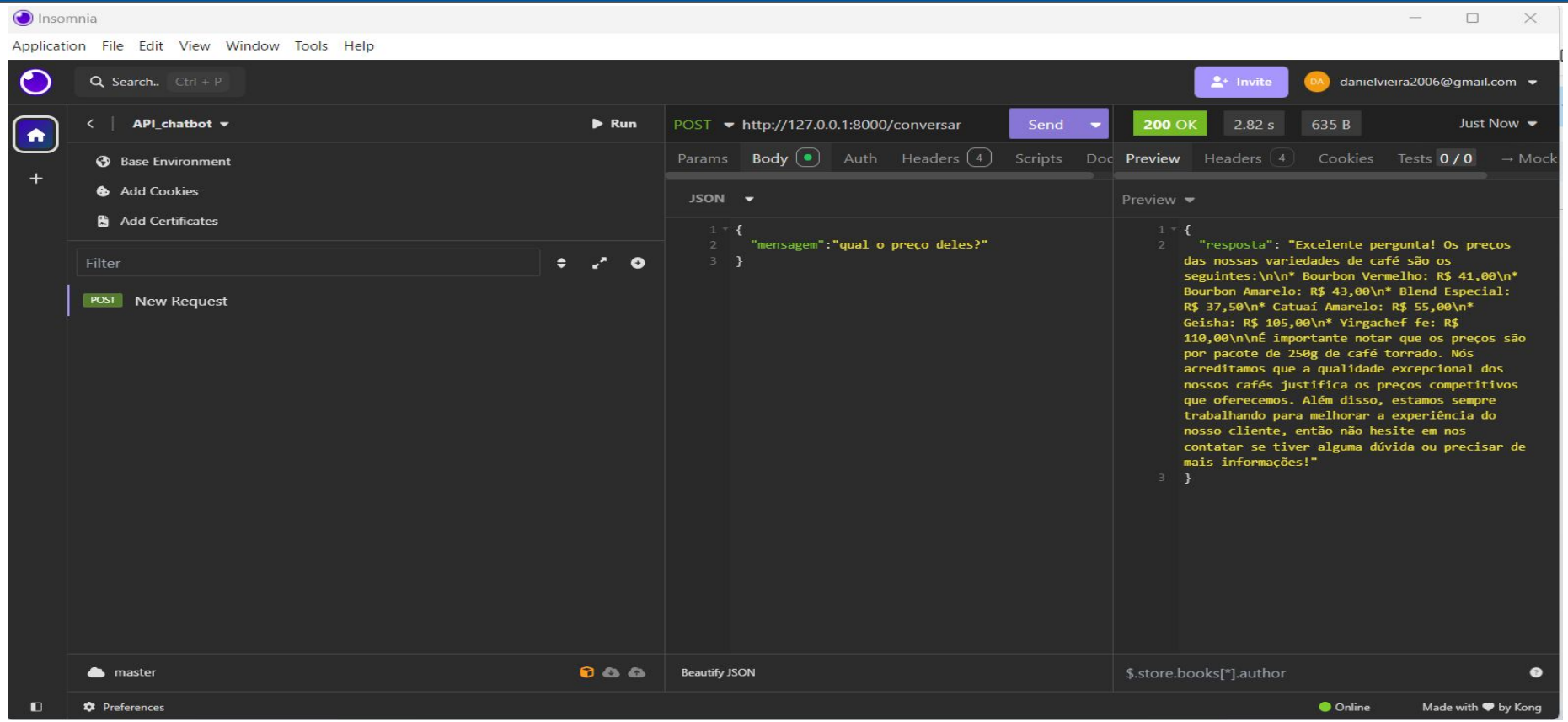
Para testar a API será utilizado o Insomnia



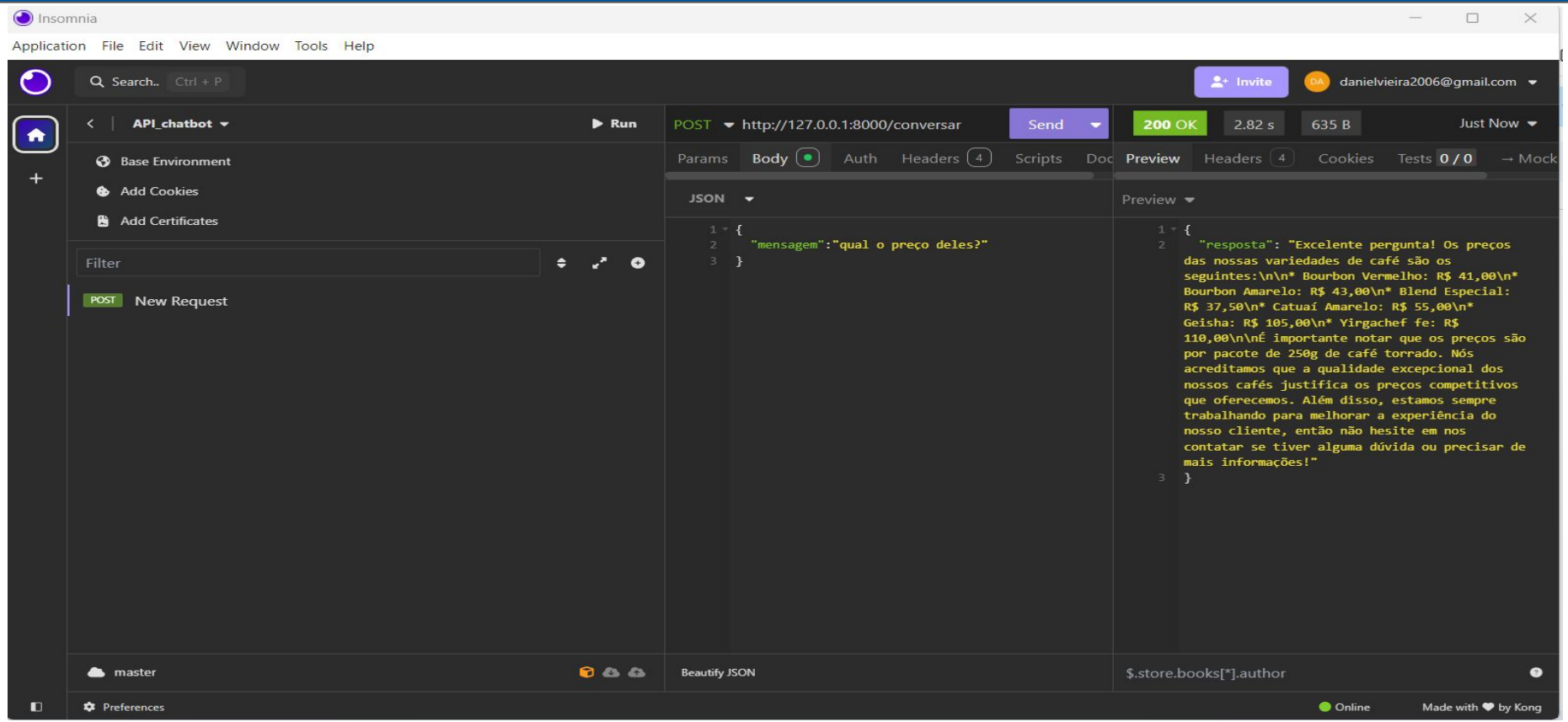
Para testar a API será utilizado o Insomnia



Para testar a API será utilizado o Insomnia



Para testar a API será utilizado o Insomnia



Para testar a API será utilizado o Insomnia

2. Você está testando em um celular ou emulador?

- **Emulador Android:** use `http://10.0.2.2:8000`
- **Celular físico:** use o **IP local da sua máquina**, exemplo: `http://192.168.0.108:8000`
  💡 Você pode descobrir seu IP local com:

```
ipconfig (Windows)
ifconfig (Linux/Mac)
```

3. Você permitiu conexões externas no FastAPI?

Por padrão, o `uvicorn` escuta **apenas `127.0.0.1`**.

👉 Para permitir conexões do celular ou emulador, use:

```
uvicorn main:app --host 0.0.0.0 --port 8000 --reload
```

Obrigado!

Prof. Me Daniel Vieira

Email: danielvieira2006@gmail.com

Linkedin: Daniel Vieira

Instagram: Prof daniel.vieira95