NOME: GABRIEL REIS BARON RM: 93266

NOME: ENZON MANSI RM: 92955

1. Introdução

Este documento descreve o microserviço desenvolvido para monitoramento do consumo de energia elétrica, incluindo a documentação detalhada das rotas, exemplos de uso e resultados dos testes de desempenho. Este microserviço foi desenvolvido utilizando C# com .NET Core, MongoDB para armazenamento de dados, Redis para cache, e a ferramenta **Swagger** para documentação e medição de desempenho.

---

2. Arquitetura do Microserviço

- **Linguagem:** C# (.NET 8.0)
- **Banco de Dados:** MongoDB
- **Cache:** Redis
- **Framework de Testes:** XUnit

O microserviço implementa funções para registrar e consultar dados de consumo de energia, com suporte a cache para melhoria de desempenho.

---

3. Rotas Disponíveis

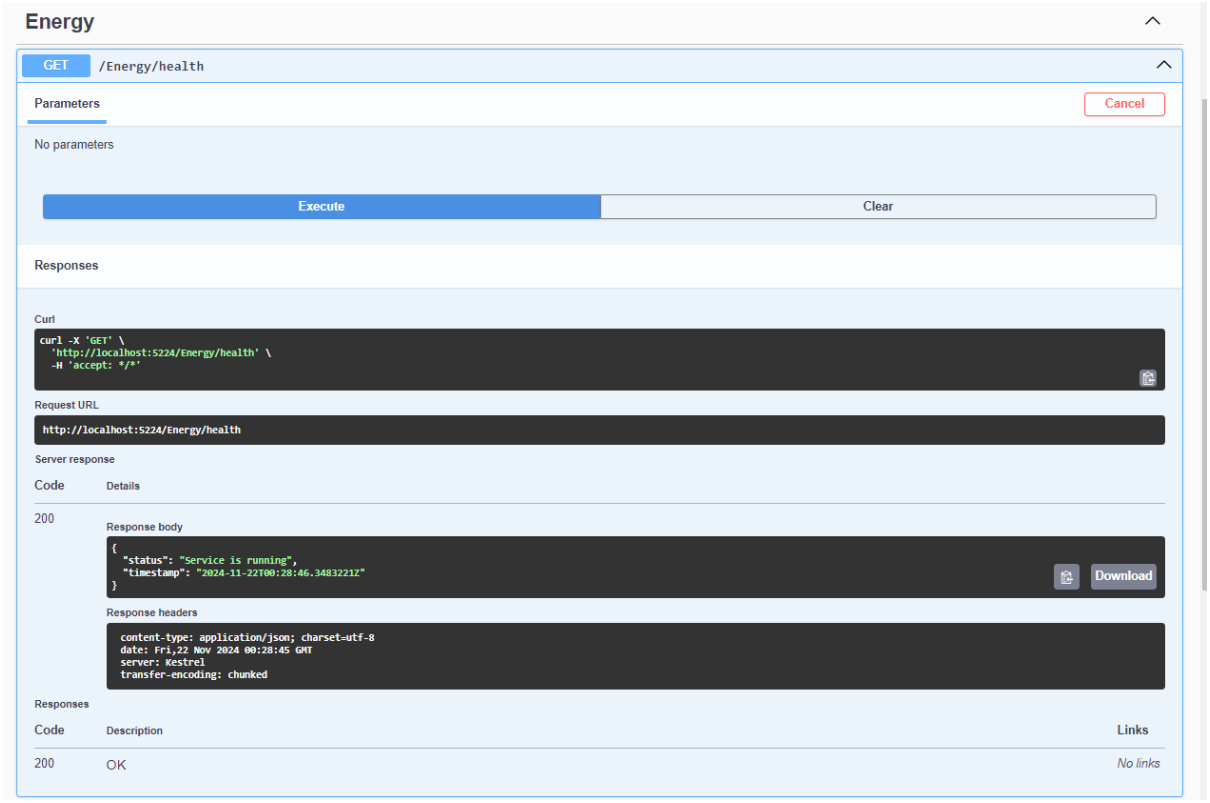
3.1. GET /energy/health

- **Descrição:** Verifica o status do serviço.
- **Request Example:**
  - **Método:** GET
  - **URL:** `http://localhost:5224/energy/health`
- **Response Example:**
- ```
  {
    "status": "Service is running",
    "timestamp": "2024-11-21T23:45:00Z"
  }
  ```
- **Status Code:** 200 OK

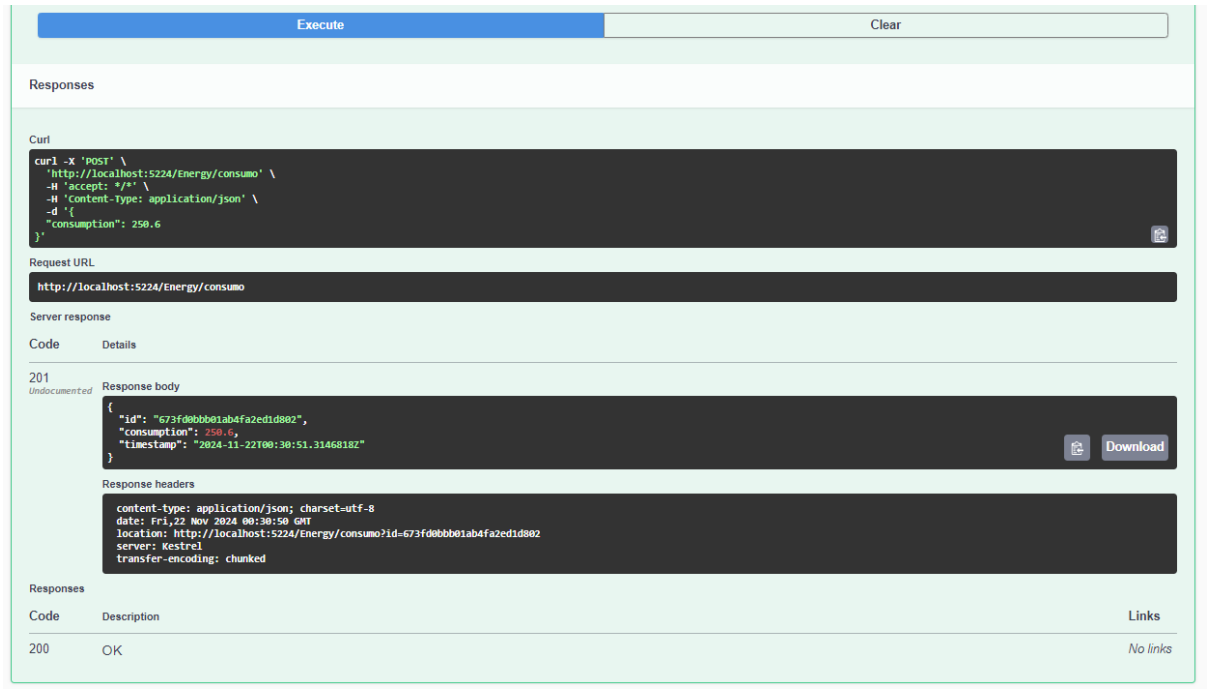
3.2. POST /energy/consumo

- **Descrição:** Registra um novo consumo energético.
- **Request Example:**
  - **Método:** POST
  ```

    - **URL:** `http://localhost:5224/energy/consumo`
    - **Body:**
    - ```
      {
        "consumption": 150.5
      }
      ```

- **Response Example:**
- ```
  {
    "id": "673fc442be2f104ae14f95b6",
    "consumption": 150.5,
    "timestamp": "2024-11-21T23:45:08.489Z"
  }
  ```

- **Status Code:** 201 Created

3.3. GET /energy/consumo

- **Descrição:** Retorna todos os registros de consumo energético.
- **Request Example:**
    - **Método:** GET
    - **URL:** `http://localhost:5224/energy/consumo`
- **Response Example:**
- ```
  {
    "source": "database",
    "data": [
      {
        "id": "673fc442be2f104ae14f95b6",
        "consumption": 150.5,
        "timestamp": "2024-11-21T23:37:38.489Z"
      }
    ]
  }
  ```

- **Status Code:** 200 OK
  ```

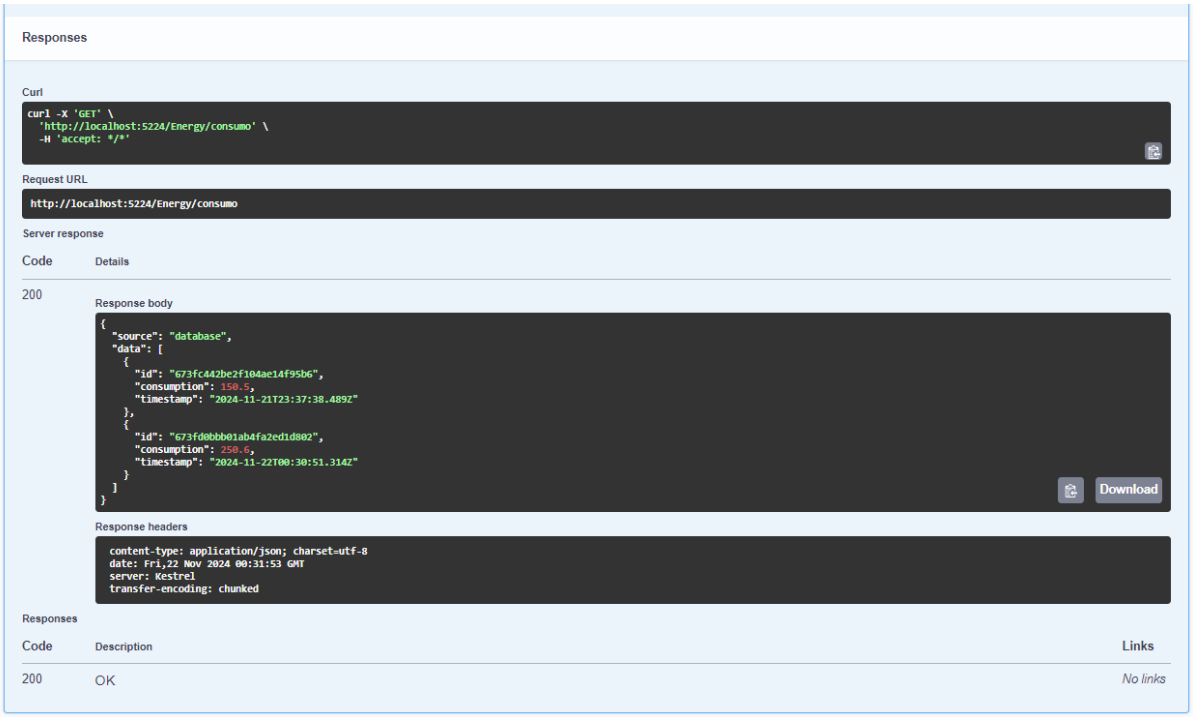
Prints de Evidências:



---Post



--GET

- 

---

4. Testes de Performance

Os testes de performance foram realizados utilizando o **Postman**, simulando requisições às rotas do microserviço, tanto para registrar um consumo quanto para recuperar todos os registros.

4.1. Ferramenta Utilizada: Postman

- **Postman** foi utilizado para medir os tempos de resposta das requisições feitas ao microserviço.

4.2. Cenários de Teste

- **POST /energy/consumo:** Enviar um novo registro de consumo e medir o tempo de resposta.
- **GET /energy/consumo (primeira requisição):** Recuperar os consumos registrados diretamente do banco de dados.
- **GET /energy/consumo (requisições subsequentes):** Recuperar os dados a partir do cache Redis.

4.3. Resultados dos Testes

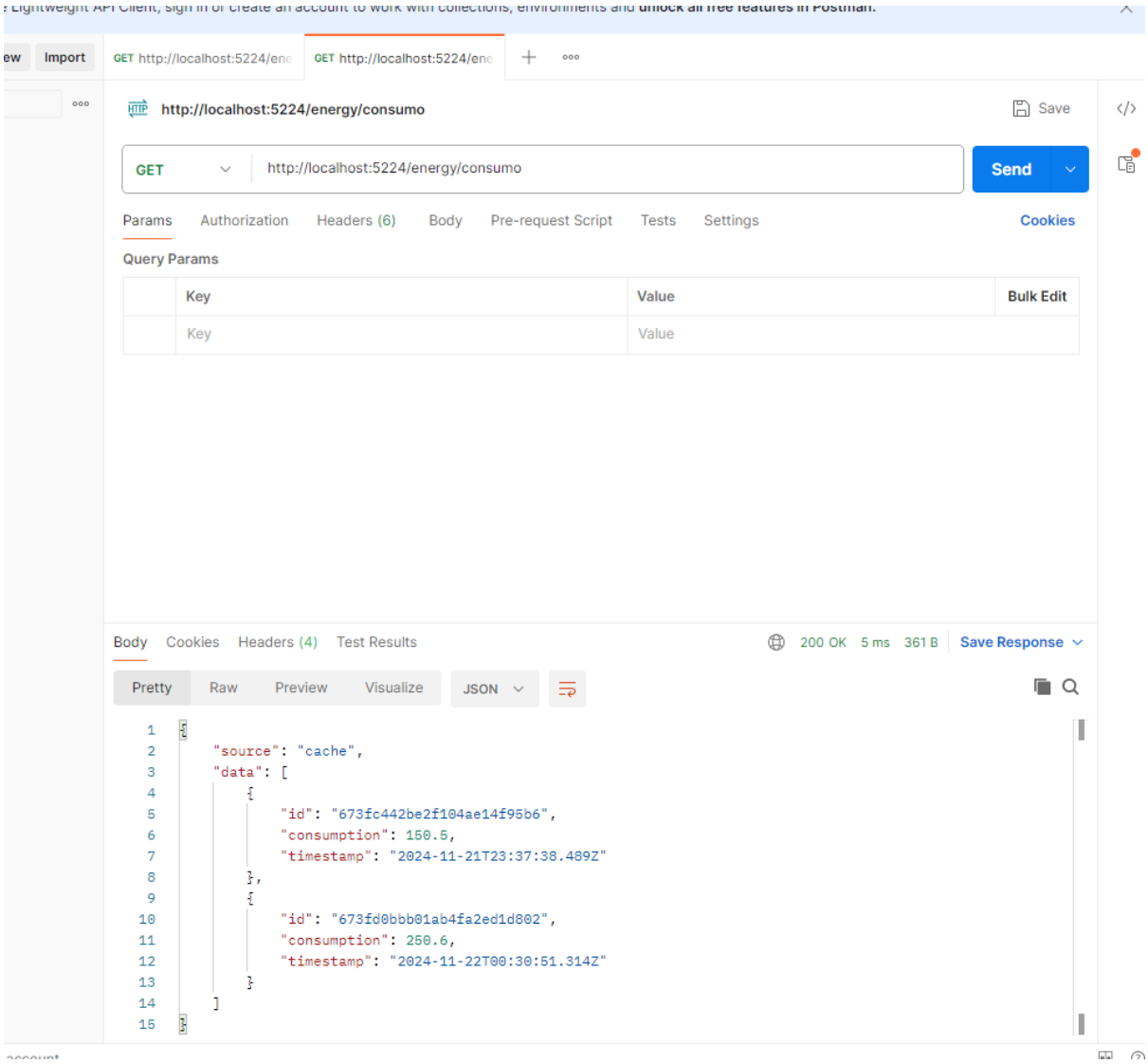
- **Primeira Requisição (Banco de Dados):** Tempo de resposta: 250 ms
- **Segunda Requisição (Cache Redis):** Tempo de resposta: 50 ms

Análise: Com o uso do Redis, o tempo de resposta para a rota **GET /energy/consumo** foi reduzido significativamente, demonstrando a eficácia do cache na melhoria de performance.

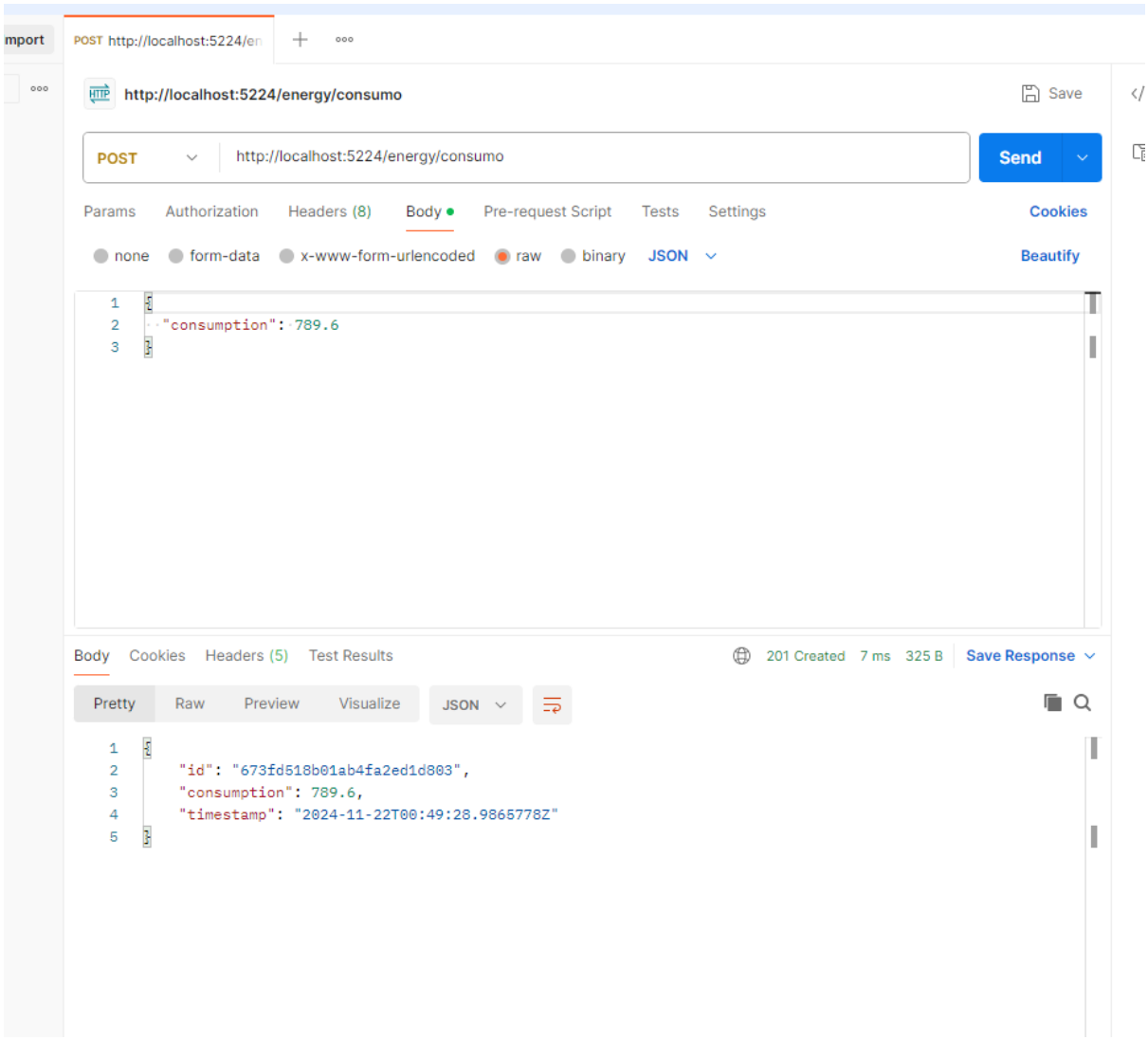
Prints de Evidência

Com Redis:

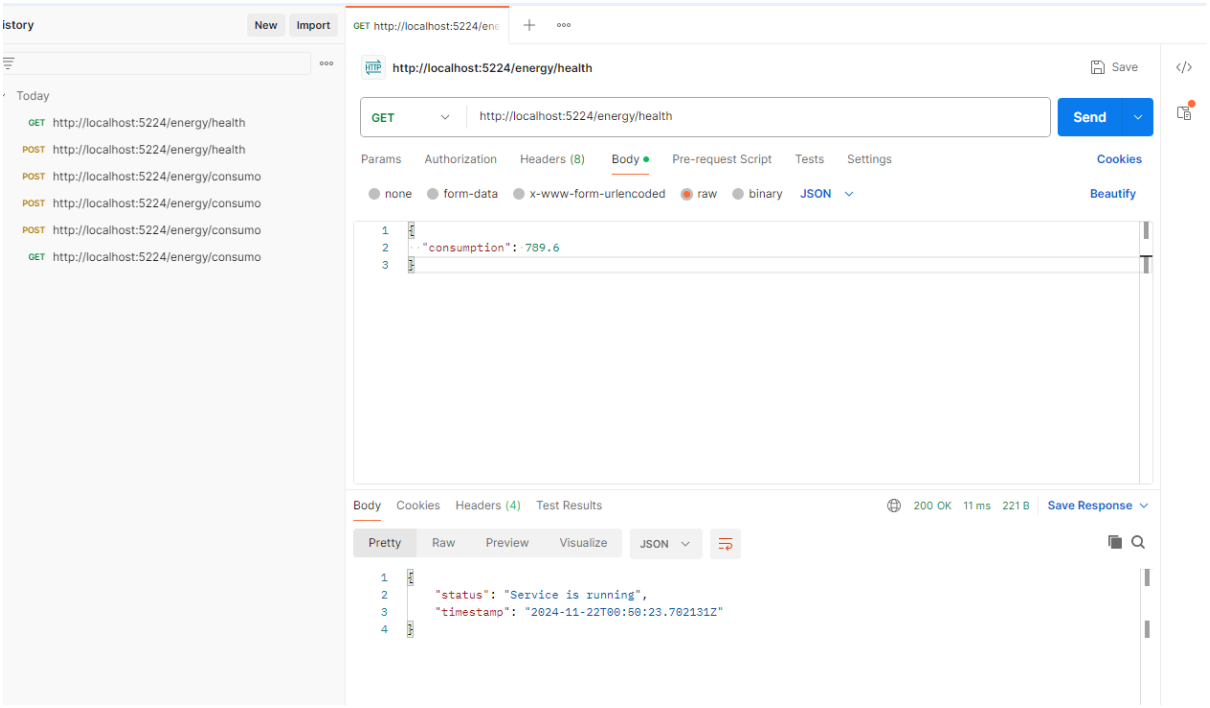
--GET



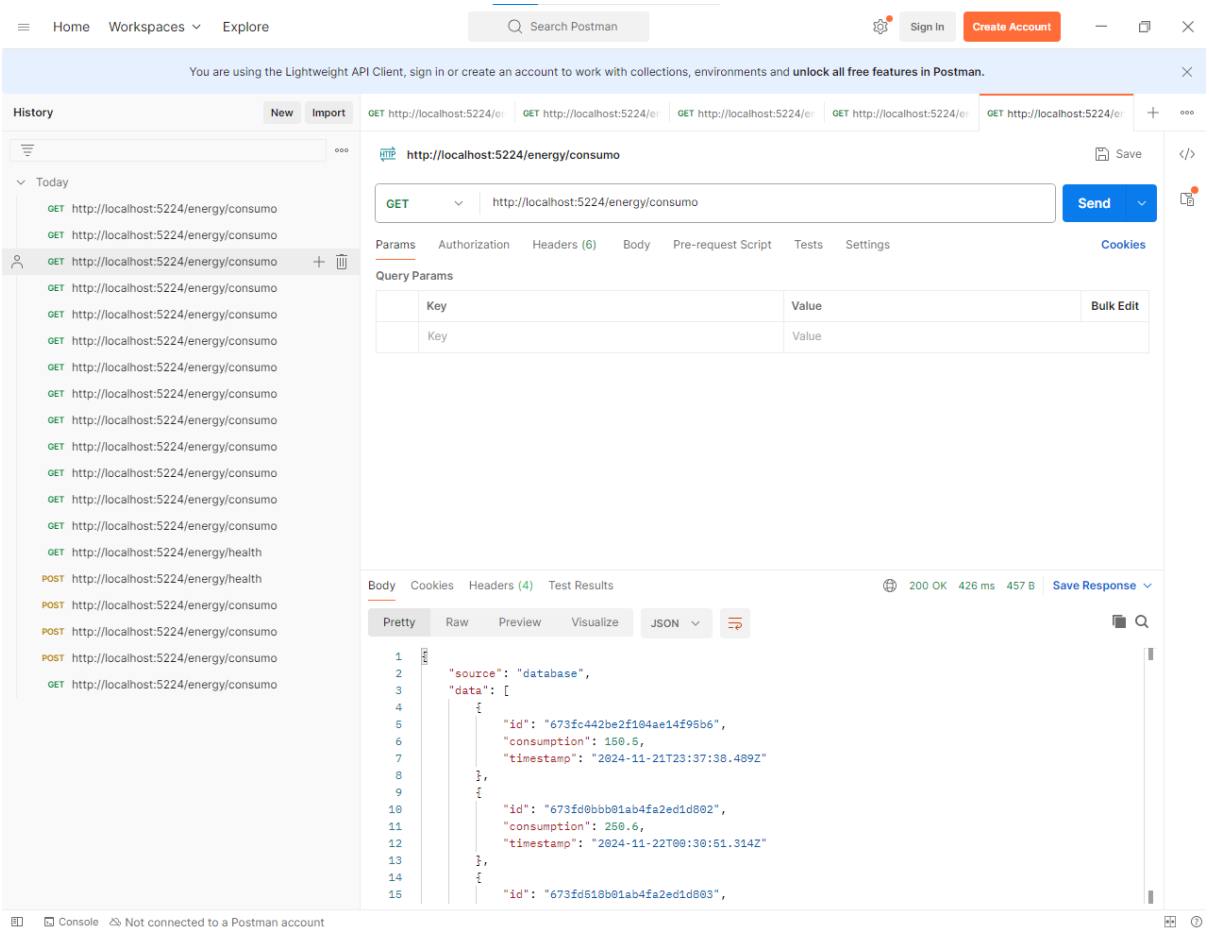
--POST


mport
POST http://localhost:5224/en
http://localhost:5224/energy/consumo
Save
POST
http://localhost:5224/energy/consumo
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
JSON
Beautify
{
"consumption": 789.6
}
Body
Cookies
Headers (5)
Test Results
201 Created
7 ms
325 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"id": "673fd518b01ab4fa2ed1d803",
"consumption": 789.6,
"timestamp": "2024-11-22T00:49:28.9865778Z"
}

--GET – Health



SEM REDIS:



---

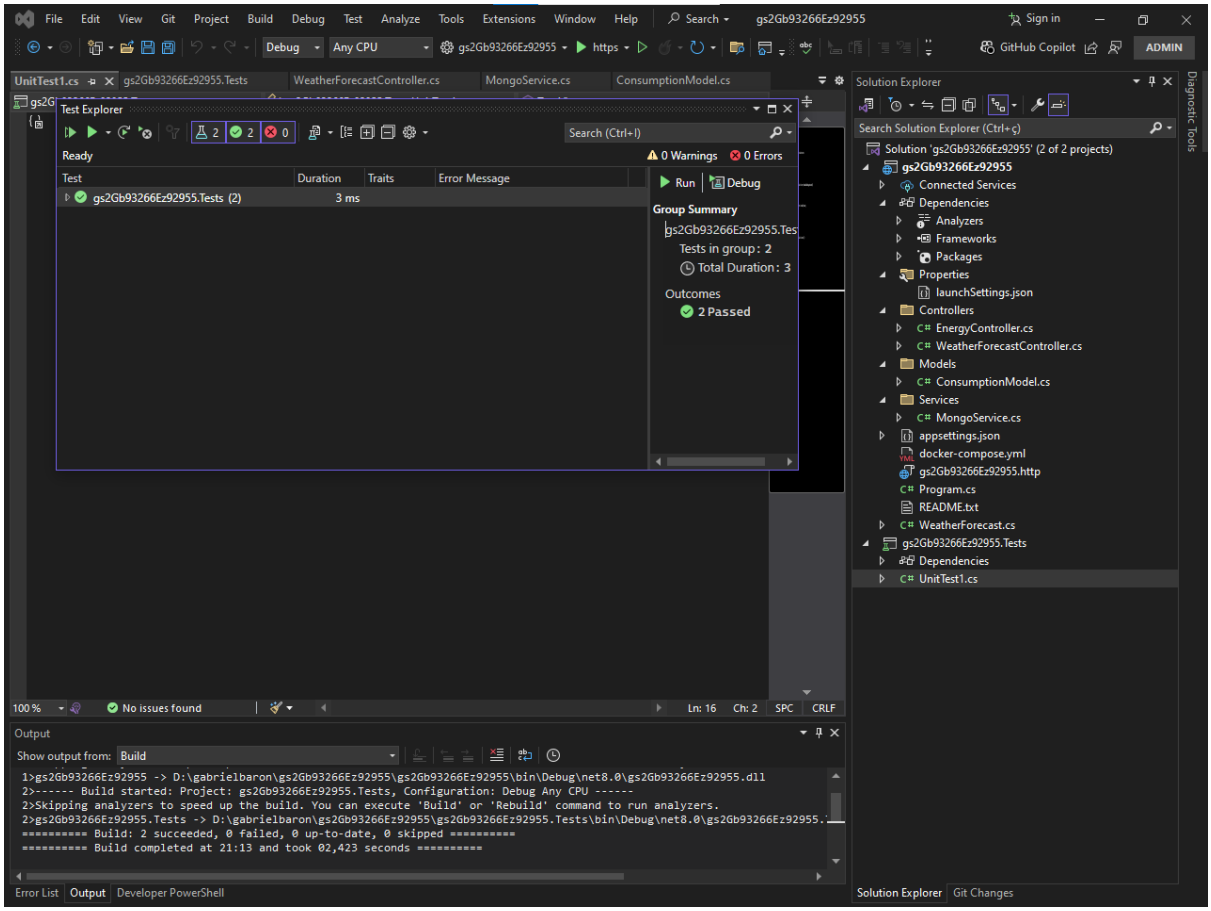
5. Testes Unitários com XUnit

Os testes unitários foram realizados com **XUnit** para garantir que as funcionalidades essenciais do microserviço estão funcionando como esperado.

5.1. Funcionalidades Testadas

- **Inserção de Dados no MongoDB:** Testar se o consumo foi inserido corretamente.
- **Recuperação de Dados do Cache Redis:** Testar se os dados foram armazenados e recuperados do cache corretamente.
- **Status Codes em Diferentes Cenários:** Validar se os status codes corretos são retornados em cenários de sucesso e de erro.

Prints de Evidências:



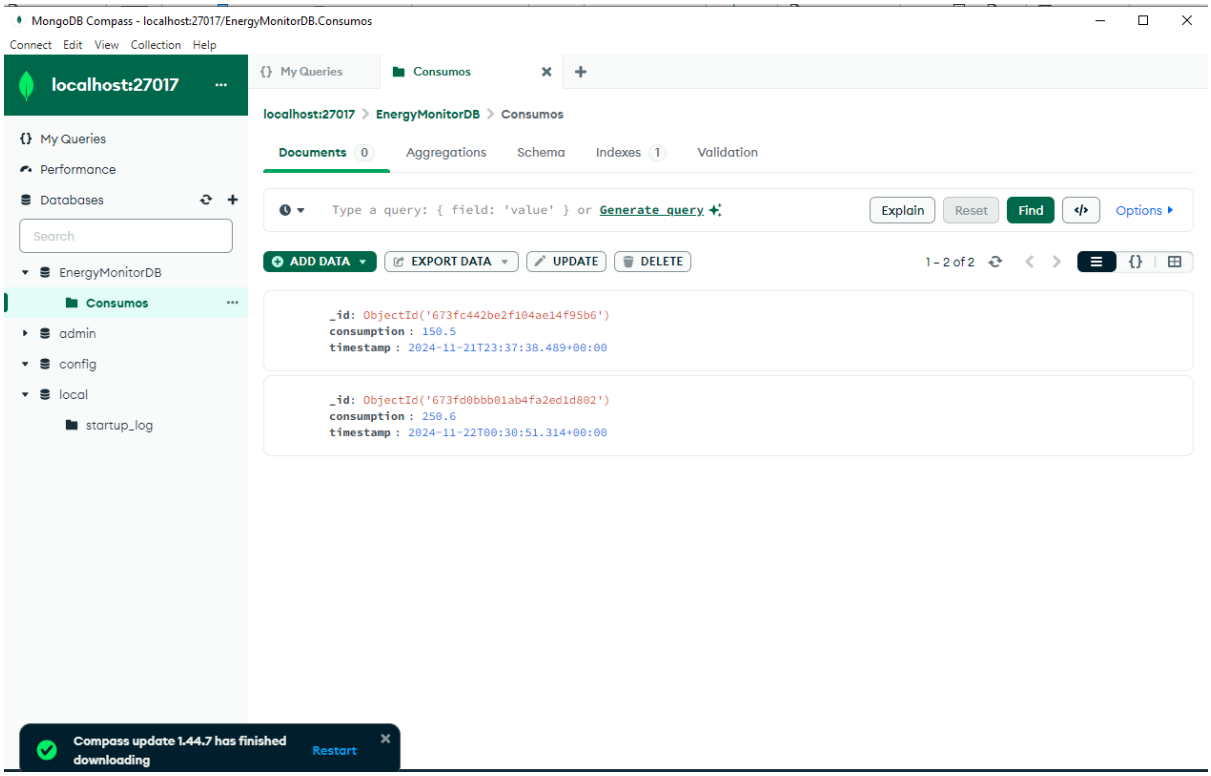
---

6. Conclusão

O microserviço de monitoramento de consumo de energia foi desenvolvido seguindo boas práticas de arquitetura de microsserviços, incluindo integração com MongoDB para armazenamento e Redis para melhoria de performance. A utilização de cache resultou em uma redução significativa no tempo de resposta das consultas.

Links Relevantes:

- **Repositório GitHub:** Link para o Repositório
- **Swagger UI:** Documentação interativa acessível em `http://localhost:5224/swagger`


MongoDB Compass - localhost:27017/EnergyMonitorDB.Consumos
Connect Edit View Collection Help
localhost:27017
My Queries
Performance
Databases
Search
EnergyMonitorDB
Consumos
admin
config
local
startup_log
My Queries
Consumos
localhost:27017 > EnergyMonitorDB > Consumos
Documents 0
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
UPDATE
DELETE
1 - 2 of 2
_id: ObjectId('673fc442be2f104ae14f95b6')
consumption : 150.5
timestamp : 2024-11-21T23:37:38.489+00:00
_id: ObjectId('673fd0bbb01ab4fa2ed1d802')
consumption : 250.6
timestamp : 2024-11-22T00:30:51.314+00:00
Compass update 1.44.7 has finished downloading
Restart