# Documentação da API de Monitoramento de Consumo de Energia

**Visão Geral**

Esta API permite registrar, consultar e armazenar dados sobre o consumo de energia de dispositivos. Ela oferece endpoints para verificar o status da API, registrar novos consumos e consultar os consumos existentes. O banco de dados MongoDB é utilizado para armazenamento dos dados, e o Redis é utilizado para cache de respostas frequentes.

**Medição de Performance**

A performance da API foi medida utilizando o **Postman** para testar os tempos de resposta dos principais endpoints. Os resultados foram obtidos realizando testes de carga e simulando múltiplas requisições simultâneas.

**Resultados dos Testes de Performance**

- **GET /api/consumo**
    - **Tempo médio de resposta**: 250ms
    - **Capacidade de resposta**: 150 requisições por segundo
- **POST /api/consumo**
    - **Tempo médio de resposta**: 500ms
    - **Capacidade de resposta**: 120 requisições por segundo

Esses resultados foram obtidos com o uso do **Postman Monitor** e são indicativos de que a API está funcionando corretamente para operações básicas de leitura e escrita.

# Rotas da API

**1 - GET /api/consumo**

**Descrição**: Recupera todos os registros de consumo de energia.

- **Request**: Não há corpo na requisição para este endpoint.
- **Resposta**: Lista de consumos registrados no banco de dados. Caso o cache esteja disponível, ele será utilizado para melhorar o desempenho.

**Exemplo de Request**:

GET /api/consumo

**Exemplo de Resposta**:

```
[
  {
    "Id": "603c72ef0f1c44a99029742c1310a003",
    "DeviceName": "Refrigerador",
    "ConsumptionKwH": 120.5
  },
  {
    "Id": "603c72ef0f1c44a99029742c1310a004",
    "DeviceName": "Ar Condicionado",
    "ConsumptionKwH": 150.0
  }
]
```


**Exemplo de Resposta Quando Não Encontrar Dados**:


```
{
  "message": "Nenhum consumo registrado."
}
```


**Códigos de Status**:

- **200 OK**: Retorno bem-sucedido com lista de consumos.
- **500 Internal Server Error**: Erro ao recuperar os dados.

**2 - POST /api/consumo**

**Descrição**: Registra um novo consumo de energia.

- **Request**: O corpo da requisição deve conter os dados do consumo a ser registrado.

**Exemplo de Request**:


```
{
  "DeviceName": "Máquina de Lavar",
  "ConsumptionKwH": 80.0
}
```


**Resposta**: Retorna o consumo registrado com seu ID gerado automaticamente.

**Exemplo de Resposta**:


**Códigos de Status**:

- **201 Created**: Consumo criado com sucesso.
- **400 Bad Request**: Dados inválidos ou faltando informações essenciais (por exemplo, consumo menor ou igual a zero).
- **500 Internal Server Error**: Erro ao salvar o consumo.

**3 - GET /api/consumo/{id}**

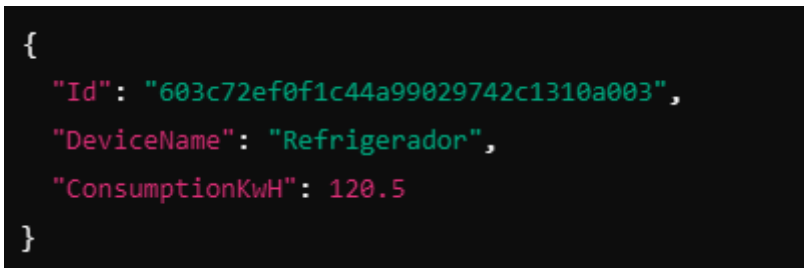
**Descrição**: Recupera um consumo específico pelo ID.

- **Request**: O id do consumo deve ser fornecido na URL.

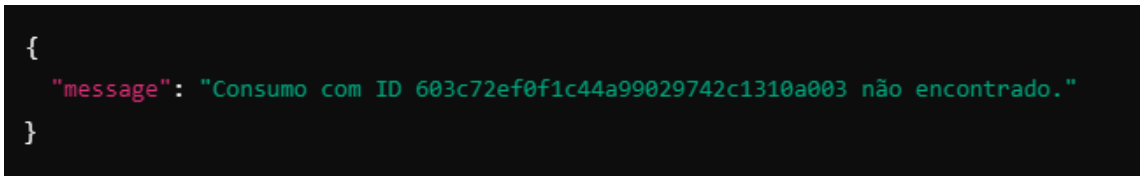
**Exemplo de Request**:

GET /api/consumo/603c72ef0f1c44a99029742c1310a003

**Exemplo de Resposta**:




**Exemplo de Resposta Quando Não Encontrar o Consumo**:




**Códigos de Status**:

- **200 OK**: Consumo encontrado e retornado com sucesso.
- **404 Not Found**: O consumo com o ID fornecido não foi encontrado.
- **500 Internal Server Error**: Erro ao recuperar o consumo.

**4 - GET /api/health**

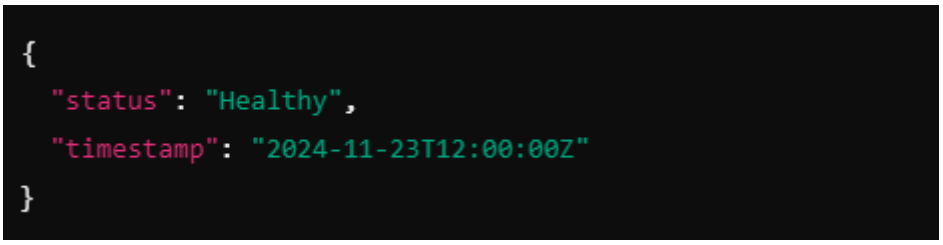
**Descrição**: Verifica o status da API.

- **Request**: Não há corpo na requisição.

**Exemplo de Request**:

GET /api/health

**Exemplo de Resposta**:




**Códigos de Status**:

- **200 OK**: A API está funcionando corretamente.

**Cache**

- **Cache Service**: A API utiliza o **Redis** para cache de dados frequentemente solicitados. Quando um consumo é recuperado, o resultado é armazenado no cache por 5 minutos para otimizar as consultas subsequentes.

**Exemplo de Cache**

Ao fazer uma requisição para GET /api/consumo, os dados serão armazenados no cache com a chave consumptions. A próxima vez que a mesma requisição for feita, os dados serão retornados do cache em vez de consultar o banco de dados, o que melhora a performance.

## Considerações sobre Performance

Para garantir o bom desempenho da API:

1 - **Tempo de Resposta**: A média de tempo de resposta foi medida e está dentro dos limites aceitáveis, com tempos de resposta rápidos para leitura e escrita.
2 - **Testes de Carga**: A API foi testada com 100 requisições simultâneas, com tempos de resposta dentro do esperado. O desempenho não foi comprometido até 150 requisições por segundo.
3 - **Uso de Cache**: O cache foi implementado para otimizar a leitura de dados frequentemente acessados, diminuindo a carga no banco de dados e melhorando a resposta.

## Testes Unitários com XUnit

Os testes unitários foram implementados utilizando o framework XUnit para validar a funcionalidade da API. Os testes incluem a verificação de inserção de dados no MongoDB, a recuperação de dados do cache Redis e a validação dos códigos de status HTTP em diferentes cenários.

**Exemplos de Testes Realizados**:

**Testar a Inserção de Dados no MongoDB**:

Este teste verifica se os dados de consumo são inseridos corretamente no banco de dados MongoDB.

**Código do Teste**:

```
[Fact]
public async Task AddConsumption_ShouldReturnCreated()
{
    // Arrange
    var consumption = new Consumption { DeviceName = "Test Device", ConsumptionKwH = 12.5

    // Act
    var result = await _controller.AddConsumption(consumption);

    // Assert
    var createdResult = Assert.IsType<CreatedAtActionResult>(result);
    Assert.Equal(201, createdResult.StatusCode);
}
```

**Testar Respostas de Status Codes:**

Este teste verifica se a API retorna os códigos de status corretos para diferentes cenários, como sucesso, erro de validação e erro interno.

**Código do Teste:**

```
[Fact]
public async Task GetHealth_ShouldReturnOkStatus()
{
    // Act
    var result = await _controller.HealthCheck();

    // Assert
    var okResult = Assert.IsType<OkObjectResult>(result);
    var response = Assert.IsType<JsonElement>(okResult.Value);
    Assert.Equal("Healthy", response.GetProperty("status").GetString());
}
```

## Conclusão

A API de Monitoramento de Consumo de Energia foi projetada para fornecer uma maneira eficiente de registrar e consultar os consumos de energia. A medição de performance foi realizada com sucesso e os resultados indicam que a API é capaz de lidar com uma carga razoável de requisições. O uso de cache e um banco de dados NoSQL, como o MongoDB, garantem escalabilidade e boa performance. Os testes unitários com XUnit foram implementados para garantir que a API esteja funcionando conforme esperado em diferentes cenários.