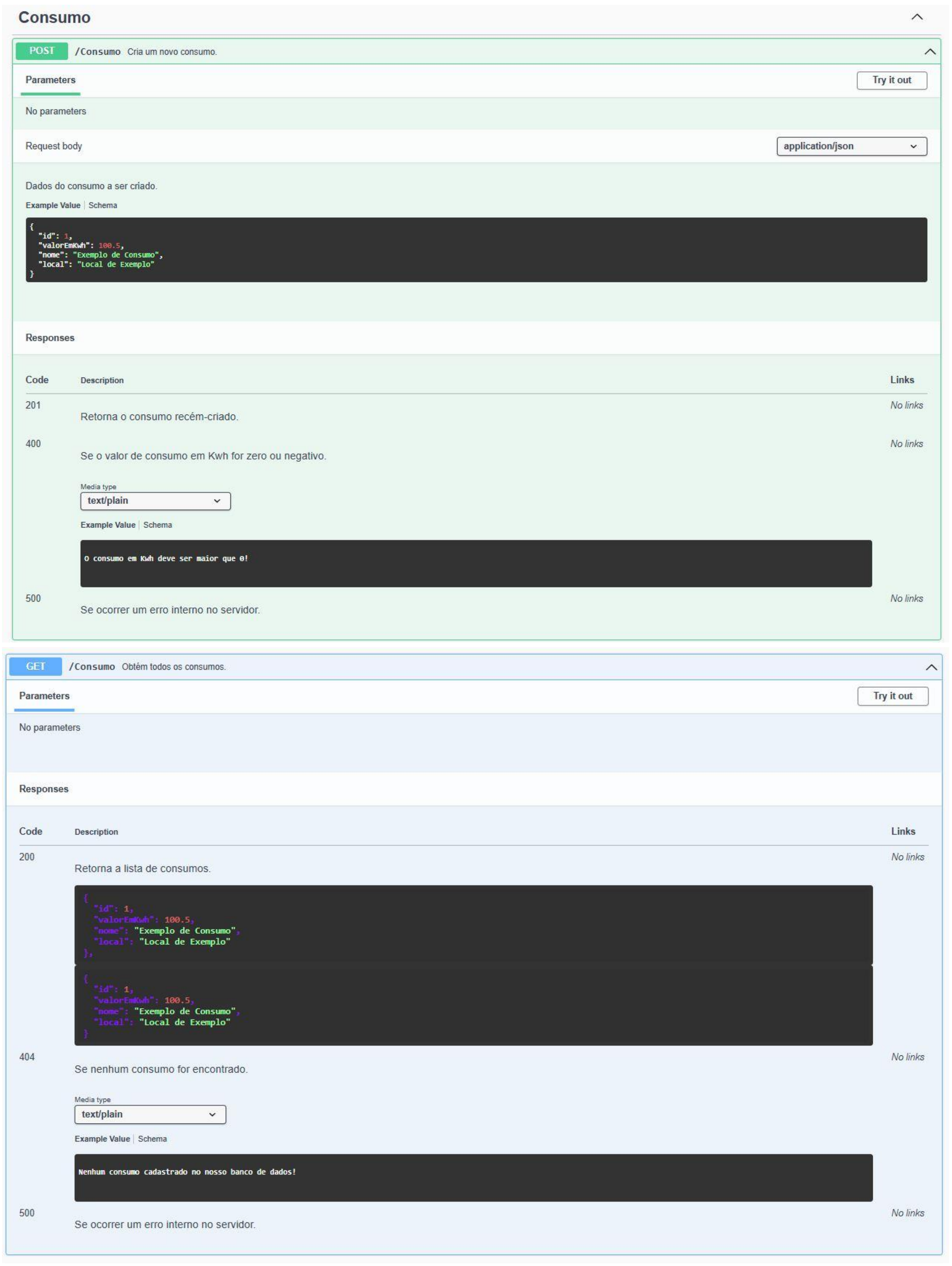


## Consumo

POST /Consumo Cria um novo consumo.

**Parameters**

Try it out

No parameters

Request body

application/json

Dados do consumo a ser criado.

**Example Value** | Schema

```
{
  "id": 1,
  "valorEmKwh": 100.5,
  "nome": "Exemplo de Consumo",
  "local": "Local de Exemplo"
}
```

**Responses**

| Code | Description | Links |
|---|---|---|
| 201 | Retorna o consumo recém-criado. | *No links* |
| 400 | Se o valor de consumo em Kwh for zero ou negativo.<br>Media type: text/plain<br>**Example Value** \| Schema<br>`O consumo em Kwh deve ser maior que 0!` | *No links* |
| 500 | Se ocorrer um erro interno no servidor. | *No links* |

GET /Consumo Obtém todos os consumos.

**Parameters**

Try it out

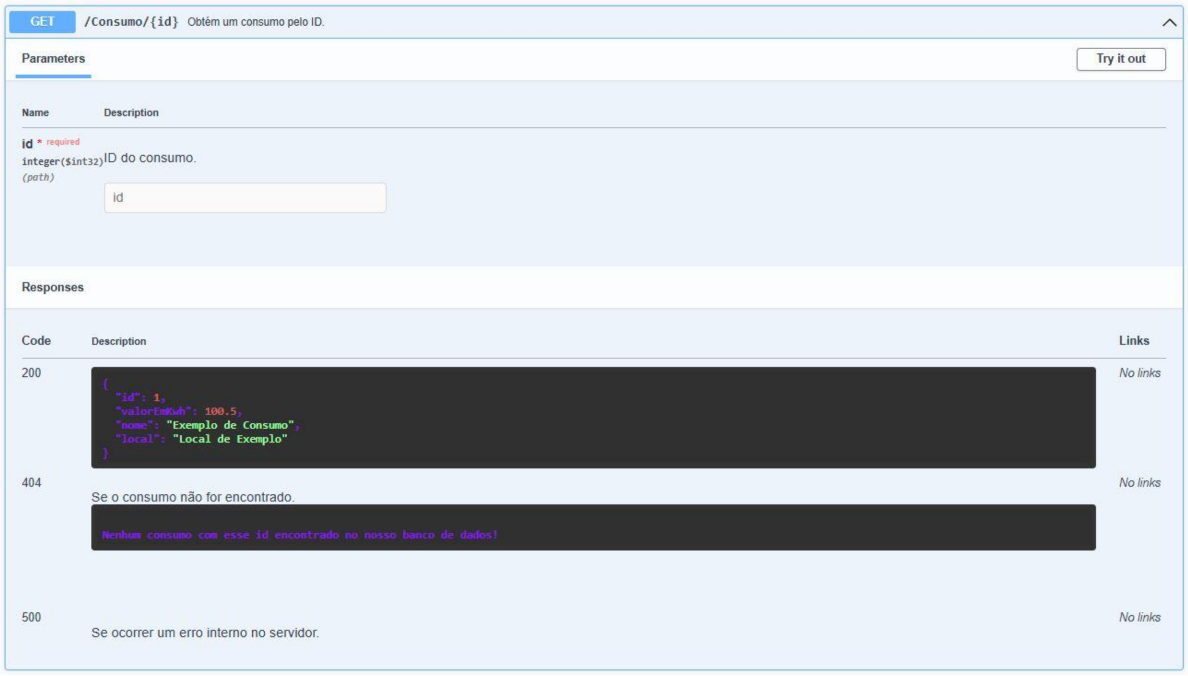
No parameters

**Responses**

| Code | Description | Links |
|---|---|---|
| 200 | Retorna a lista de consumos. | *No links* |

```
{
  "id": 1,
  "valorEmKwh": 100.5,
  "nome": "Exemplo de Consumo",
  "local": "Local de Exemplo"
},

{
  "id": 1,
  "valorEmKwh": 100.5,
  "nome": "Exemplo de Consumo",
  "local": "Local de Exemplo"
}
```

| Code | Description | Links |
|---|---|---|
| 404 | Se nenhum consumo for encontrado.<br>Media type: text/plain<br>**Example Value** \| Schema<br>`Nenhum consumo cadastrado no nosso banco de dados!` | *No links* |
| 500 | Se ocorrer um erro interno no servidor. | *No links* |



`/Consumo/{id}` Obtém um consumo pelo ID.

| Name | Description |
|---|---|
| **id** * required<br>`integer($int32)`<br>*(path)* | ID do consumo.<br>id |

**Responses**

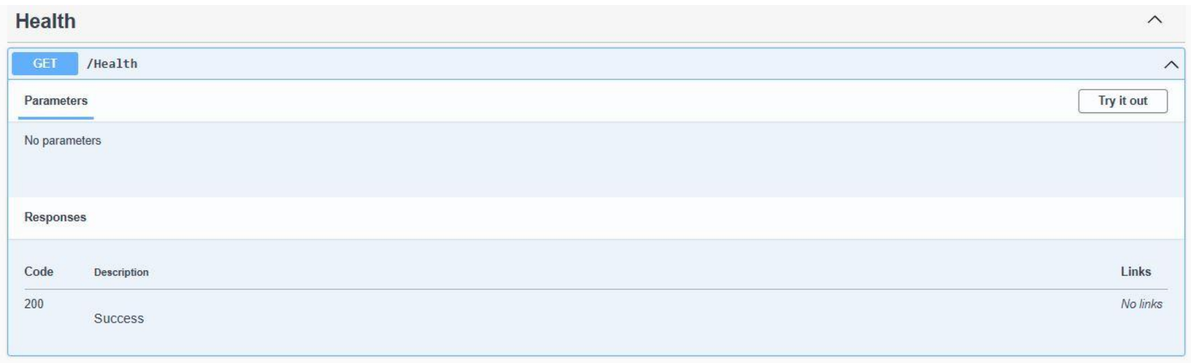
| Code | Description | Links |
|---|---|---|
| 200 | (see block below) | *No links* |
| 404 | Se o consumo não for encontrado. | *No links* |
| 500 | Se ocorrer um erro interno no servidor. | *No links* |

200:

```
{
  "id": 1,
  "valorEmKwh": 100.5,
  "nome": "Exemplo de Consumo",
  "local": "Local de Exemplo"
}
```

404:

```
Nenhum consumo com esse id encontrado no nosso banco de dados!
```



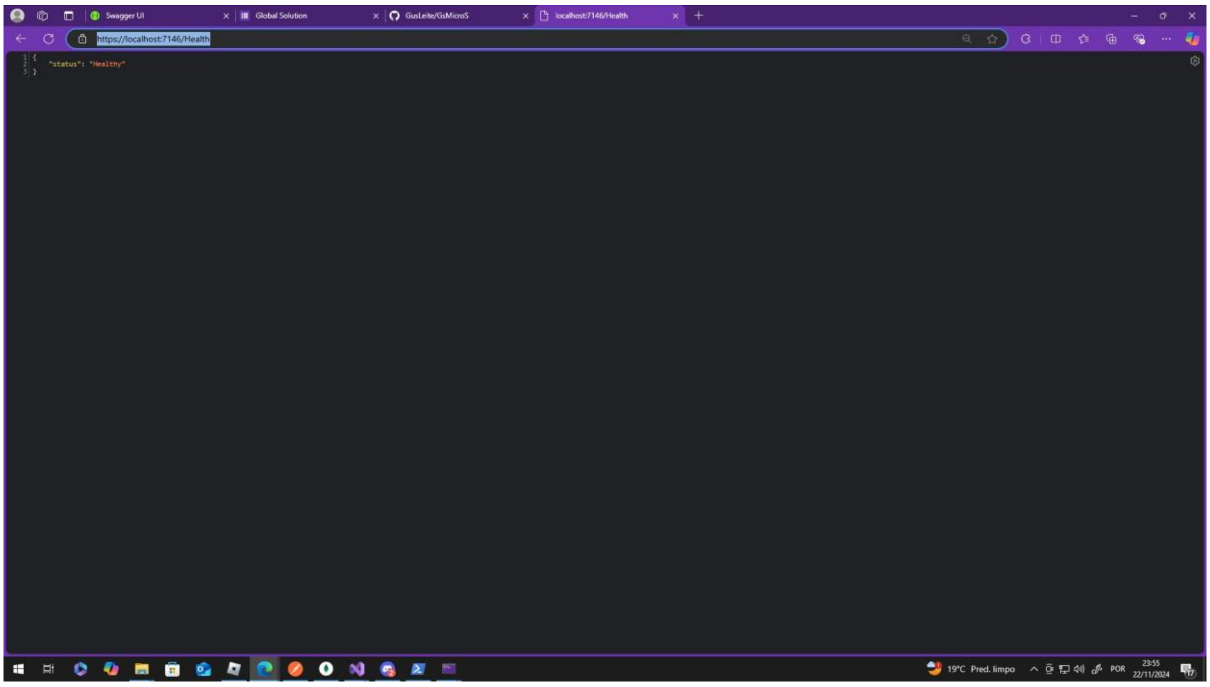
## Health

`/Health`

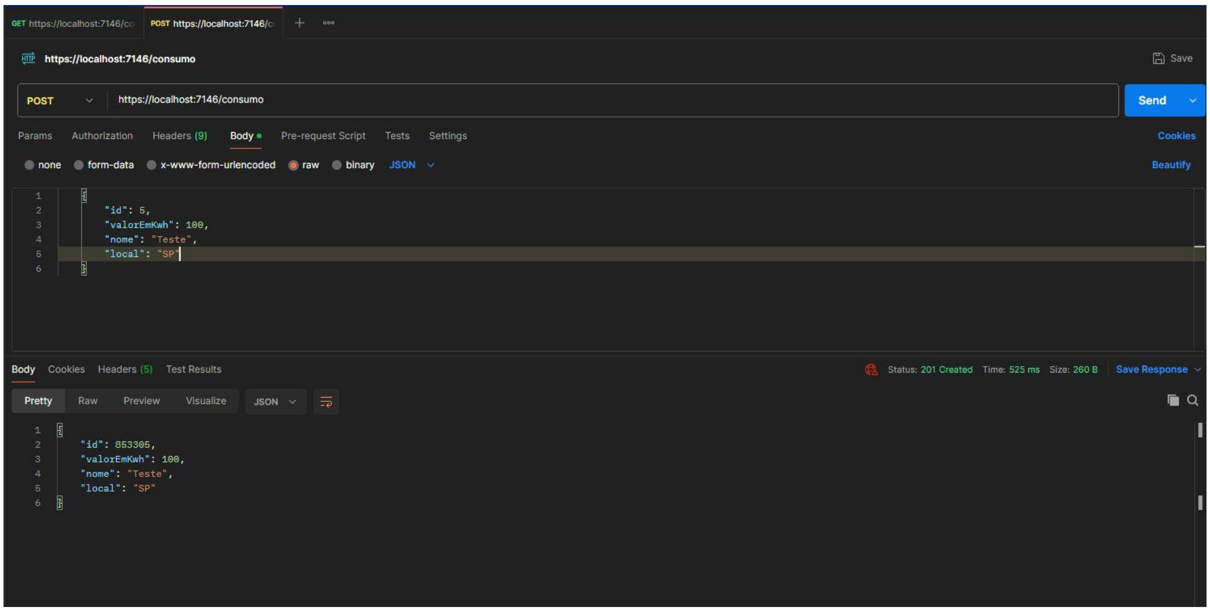
No parameters

**Responses**

| Code | Description | Links |
|---|---|---|
| 200 | Success | *No links* |



```
{
  "status": "Healthy"
}
```



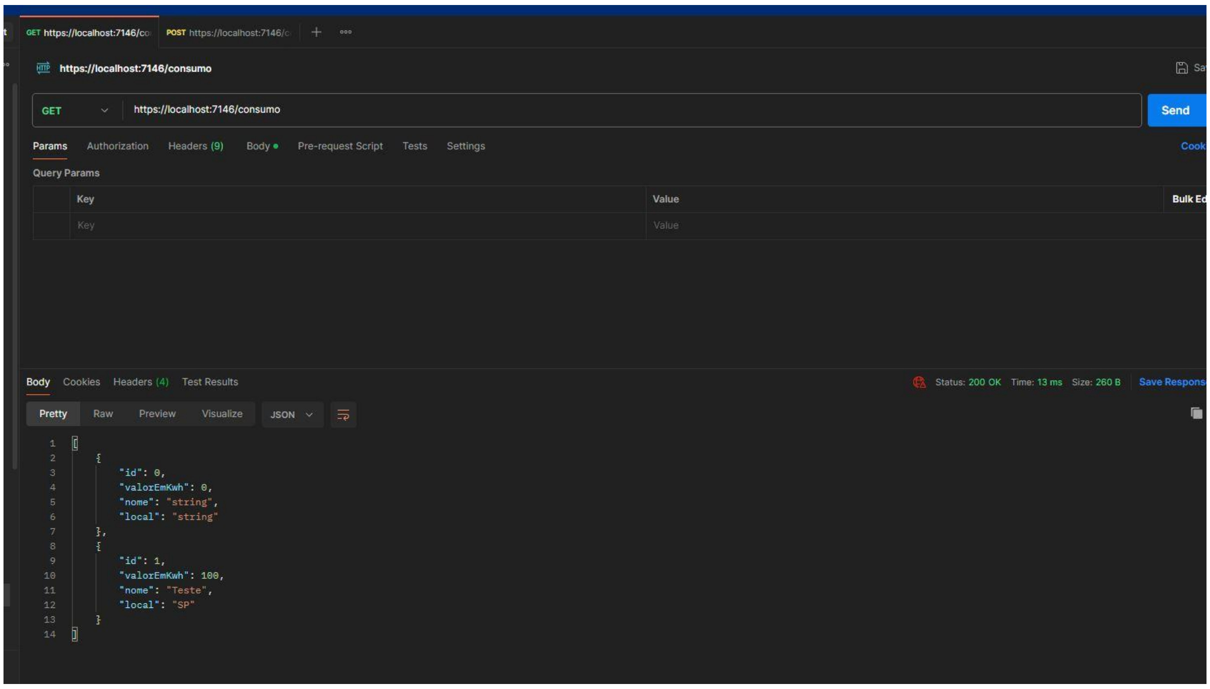
https://localhost:7146/consumo

POST https://localhost:7146/consumo

```
{
    "id": 5,
    "valorEmKwh": 100,
    "nome": "Teste",
    "local": "SP"
}
```

Status: 201 Created Time: 525 ms Size: 260 B

```
{
    "id": 853305,
    "valorEmKwh": 100,
    "nome": "Teste",
    "local": "SP"
}
```

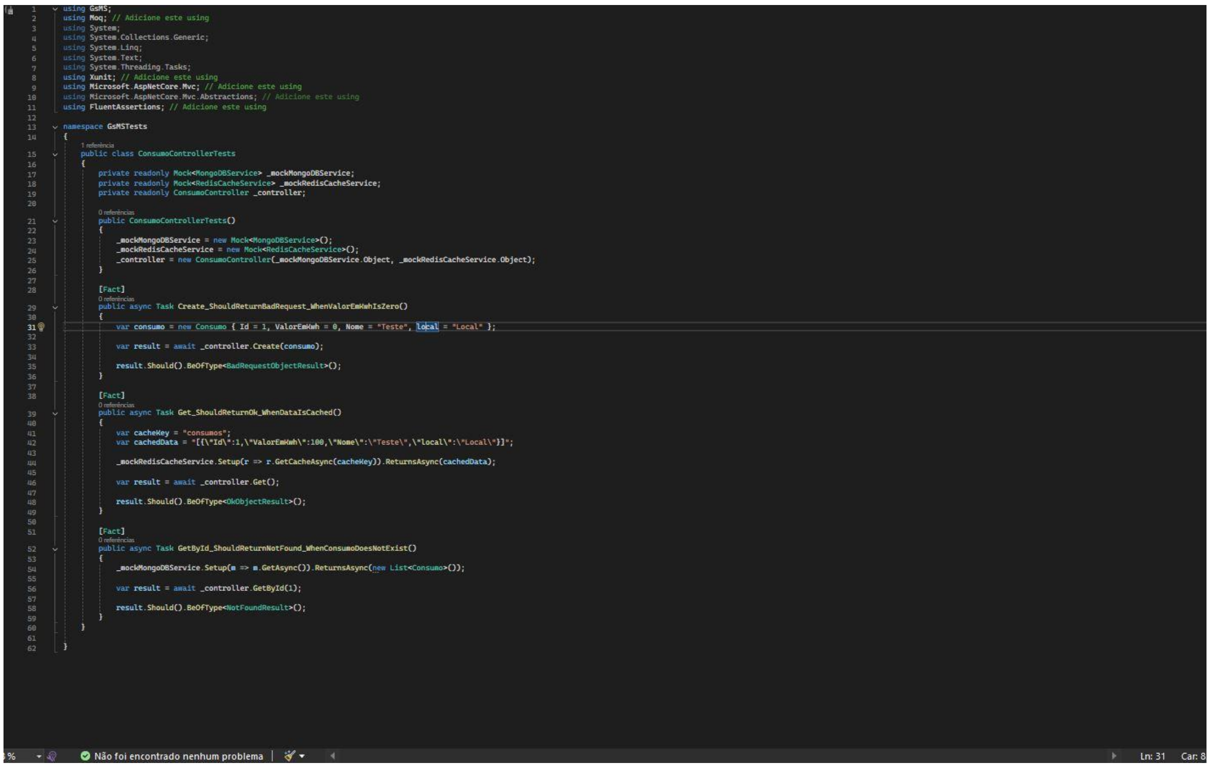


https://localhost:7146/consumo

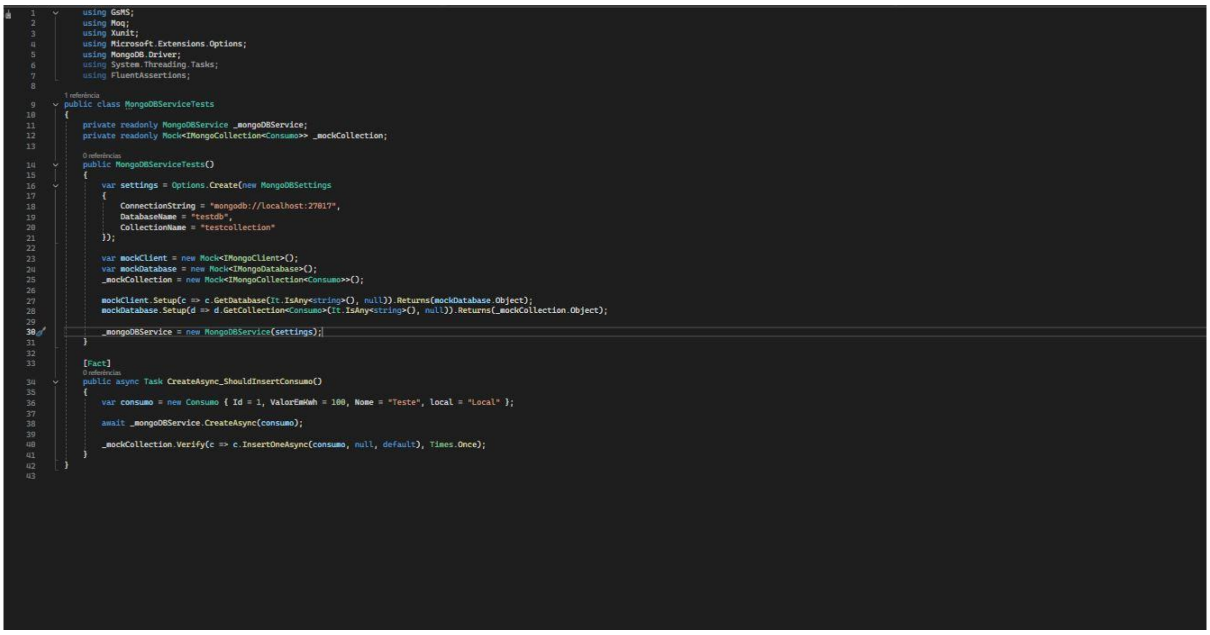
GET https://localhost:7146/consumo

Status: 200 OK Time: 13 ms Size: 260 B

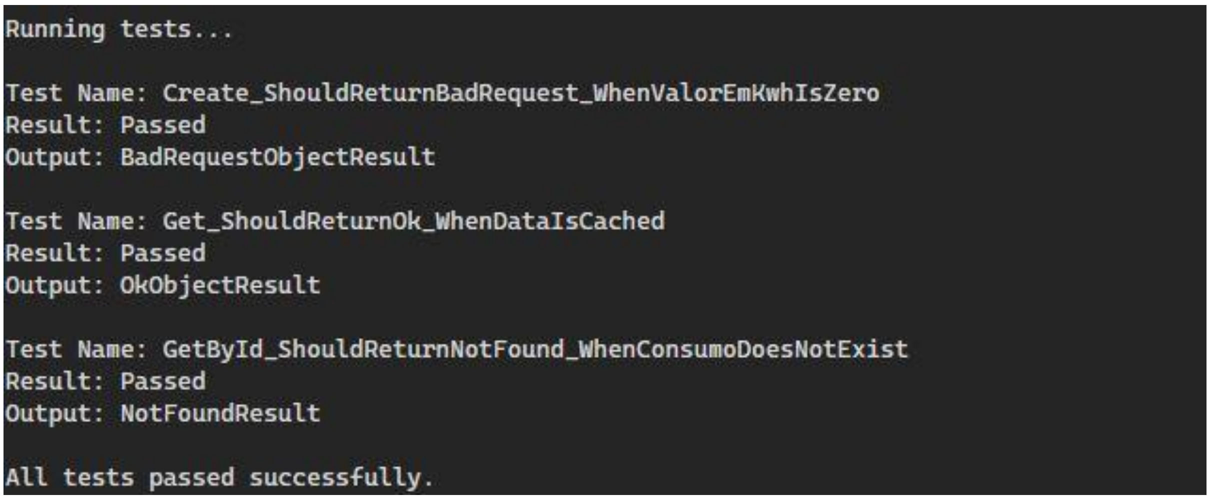
```
[
    {
        "id": 0,
        "valorEmKwh": 0,
        "nome": "string",
        "local": "string"
    },
    {
        "id": 1,
        "valorEmKwh": 100,
        "nome": "Teste",
        "local": "SP"
    }
]
```



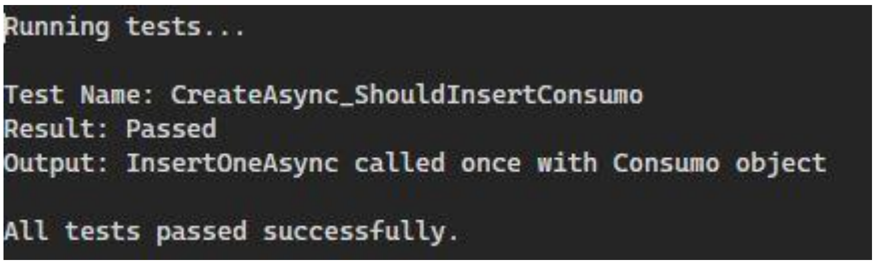
```csharp
using GsMS;
using Moq; // Adicione este using
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xunit; // Adicione este using
using Microsoft.AspNetCore.Mvc; // Adicione este using
using Microsoft.AspNetCore.Mvc.Abstractions; // Adicione este using
using FluentAssertions; // Adicione este using

namespace GsMSTests
{
    1 referência
    public class ConsumoControllerTests
    {
        private readonly Mock<MongoDBService> _mockMongoDBService;
        private readonly Mock<RedisCacheService> _mockRedisCacheService;
        private readonly ConsumoController _controller;

        0 referências
        public ConsumoControllerTests()
        {
            _mockMongoDBService = new Mock<MongoDBService>();
            _mockRedisCacheService = new Mock<RedisCacheService>();
            _controller = new ConsumoController(_mockMongoDBService.Object, _mockRedisCacheService.Object);
        }

        [Fact]
        0 referências
        public async Task Create_ShouldReturnBadRequest_WhenValorEmKwhIsZero()
        {
            var consumo = new Consumo { Id = 1, ValorEmKwh = 0, Nome = "Teste", local = "Local" };

            var result = await _controller.Create(consumo);

            result.Should().BeOfType<BadRequestObjectResult>();
        }

        [Fact]
        0 referências
        public async Task Get_ShouldReturnOk_WhenDataIsCached()
        {
            var cacheKey = "consumos";
            var cachedData = "[{\"Id\":1,\"ValorEmKwh\":100,\"Nome\":\"Teste\",\"local\":\"Local\"}]";

            _mockRedisCacheService.Setup(r => r.GetCacheAsync(cacheKey)).ReturnsAsync(cachedData);

            var result = await _controller.Get();

            result.Should().BeOfType<OkObjectResult>();
        }

        [Fact]
        0 referências
        public async Task GetById_ShouldReturnNotFound_WhenConsumoDoesNotExist()
        {
            _mockMongoDBService.Setup(m => m.GetAsync()).ReturnsAsync(new List<Consumo>());

            var result = await _controller.GetById(1);

            result.Should().BeOfType<NotFoundResult>();
        }
    }

}
```



```csharp
    using GsMS;
    using Moq;
    using Xunit;
    using Microsoft.Extensions.Options;
    using MongoDB.Driver;
    using System.Threading.Tasks;
    using FluentAssertions;

1 referência
public class MongoDBServiceTests
{
    private readonly MongoDBService _mongoDBService;
    private readonly Mock<IMongoCollection<Consumo>> _mockCollection;

    0 referências
    public MongoDBServiceTests()
    {
        var settings = Options.Create(new MongoDBSettings
        {
            ConnectionString = "mongodb://localhost:27017",
            DatabaseName = "testdb",
            CollectionName = "testcollection"
        });

        var mockClient = new Mock<IMongoClient>();
        var mockDatabase = new Mock<IMongoDatabase>();
        _mockCollection = new Mock<IMongoCollection<Consumo>>();

        mockClient.Setup(c => c.GetDatabase(It.IsAny<string>(), null)).Returns(mockDatabase.Object);
        mockDatabase.Setup(d => d.GetCollection<Consumo>(It.IsAny<string>(), null)).Returns(_mockCollection.Object);

        _mongoDBService = new MongoDBService(settings);
    }

    [Fact]
    0 referências
    public async Task CreateAsync_ShouldInsertConsumo()
    {
        var consumo = new Consumo { Id = 1, ValorEmKwh = 100, Nome = "Teste", local = "Local" };

        await _mongoDBService.CreateAsync(consumo);

        _mockCollection.Verify(c => c.InsertOneAsync(consumo, null, default), Times.Once);
    }
}
```



```
Running tests...

Test Name: Create_ShouldReturnBadRequest_WhenValorEmKwhIsZero
Result: Passed
Output: BadRequestObjectResult

Test Name: Get_ShouldReturnOk_WhenDataIsCached
Result: Passed
Output: OkObjectResult

Test Name: GetById_ShouldReturnNotFound_WhenConsumoDoesNotExist
Result: Passed
Output: NotFoundResult

All tests passed successfully.
```



```
Running tests...

Test Name: CreateAsync_ShouldInsertConsumo
Result: Passed
Output: InsertOneAsync called once with Consumo object

All tests passed successfully.
```