**Documentação da API Consumo**

**Base URL**

https://localhost:54726/api/Consumo

---

**Endpoints**

**1. Registrar Consumo - POST /api/Consumo**

**Descrição**

Este endpoint permite registrar um novo consumo de dados. O ID será gerado automaticamente se não for fornecido e a data de registro será definida para o momento atual. Se o valor do consumo não for fornecido ou for inválido, um valor aleatório será gerado.

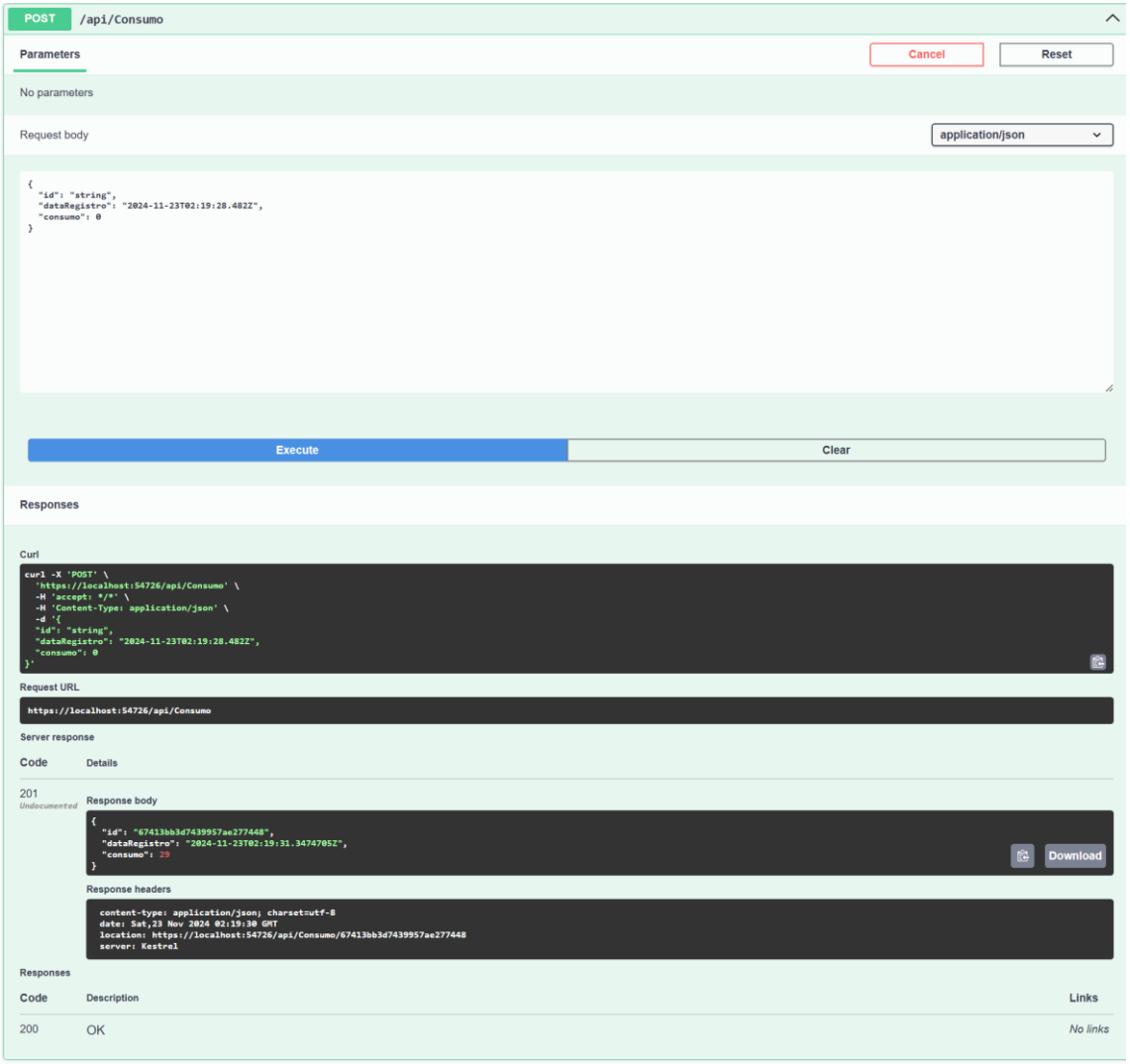
**Requisição**

- **Método**: POST
- **URL**: /api/Consumo
- **Corpo (Request Body)**:
    - **Id**: (opcional) String - O ID do consumo. Se não for fornecido, será gerado automaticamente.
    - **DataRegistro**: (opcional) Data e hora - A data e hora do registro do consumo. Se não for fornecido, será gerado automaticamente.
    - **Consumo**: Integer - O valor do consumo em dados. Se não for fornecido ou for inválido (menor ou igual a zero), um valor aleatório será gerado.

**Exemplo de Corpo da Requisição**:

**Resposta**

- **Código de Status**: 201 Created
- **Corpo da Resposta**:
    - **Id**: String - O ID gerado automaticamente.
    - **DataRegistro**: Data e hora do registro.
    - **Consumo**: Valor do consumo registrado.




**Possíveis Erros**

- **400 Bad Request**: Quando os dados enviados são inválidos, como no caso de um consumo menor ou igual a zero.
- **500 Internal Server Error**: Se ocorrer um erro inesperado no servidor.

**. Buscar Consumos - GET /api/Consumo**

**Descrição**

Este endpoint retorna todos os registros de consumo armazenados no banco de dados.

**Requisição**

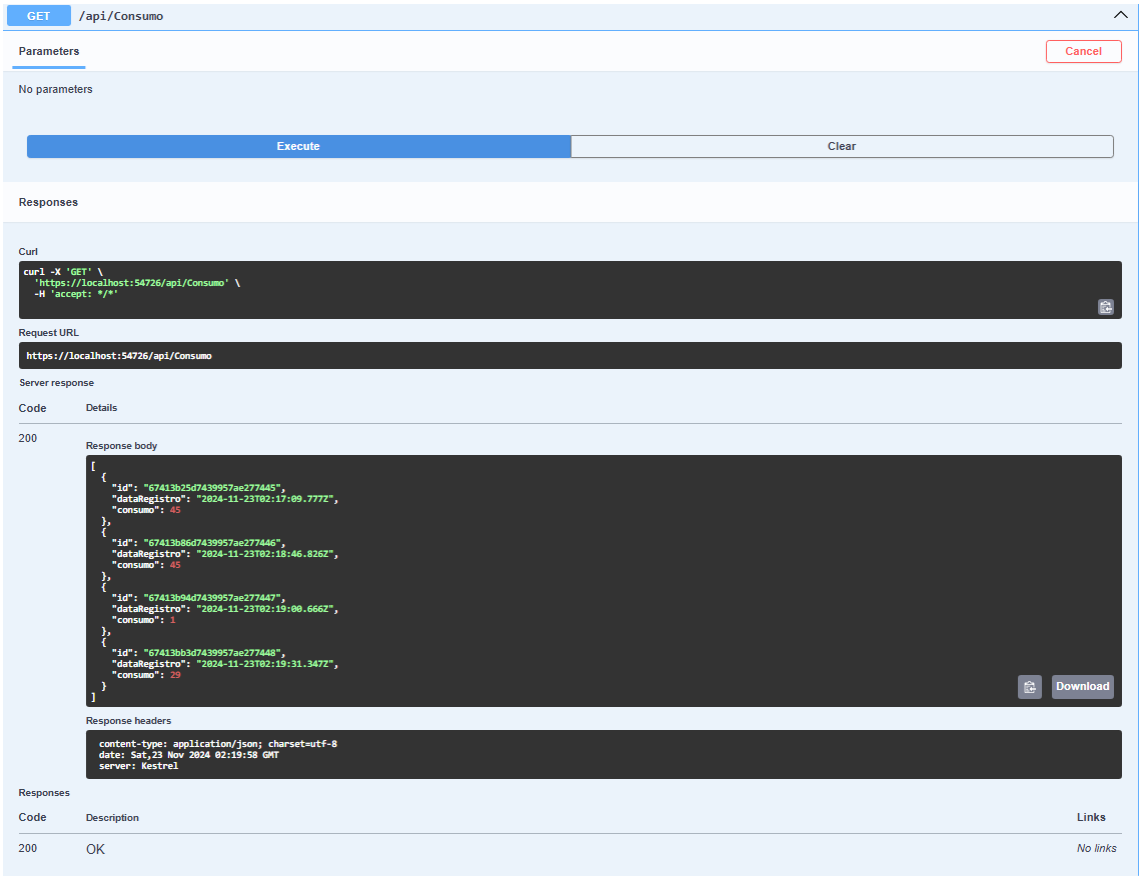
- **Método**: GET
- **URL**: /api/Consumo

**Resposta**

- **Código de Status**: 200 OK

- **Corpo da Resposta**: Uma lista de objetos ConsumoModel.

**Exemplo de Resposta**:




- **Código de Status**: 404 Not Found
    - **Mensagem**: Caso não haja nenhum registro de consumo no banco de dados.

**Exemplo de Resposta (Nenhum Registro)**:


{

"message": "Nenhum registro encontrado"

}


**Possíveis Erros**

- **500 Internal Server Error**: Se ocorrer um erro inesperado no servidor.

**3. Buscar Consumo por ID - GET /api/Consumo/{id}**

**Descrição**

Este endpoint permite buscar um consumo específico utilizando o ID do consumo. Caso o consumo com o ID fornecido não seja encontrado, será retornado um erro.

**Requisição**

- **Método**: GET
- **URL**: /api/Consumo/{id}
- **Parâmetros**:
    - **id**: String (requerido) - O ID do consumo a ser consultado.

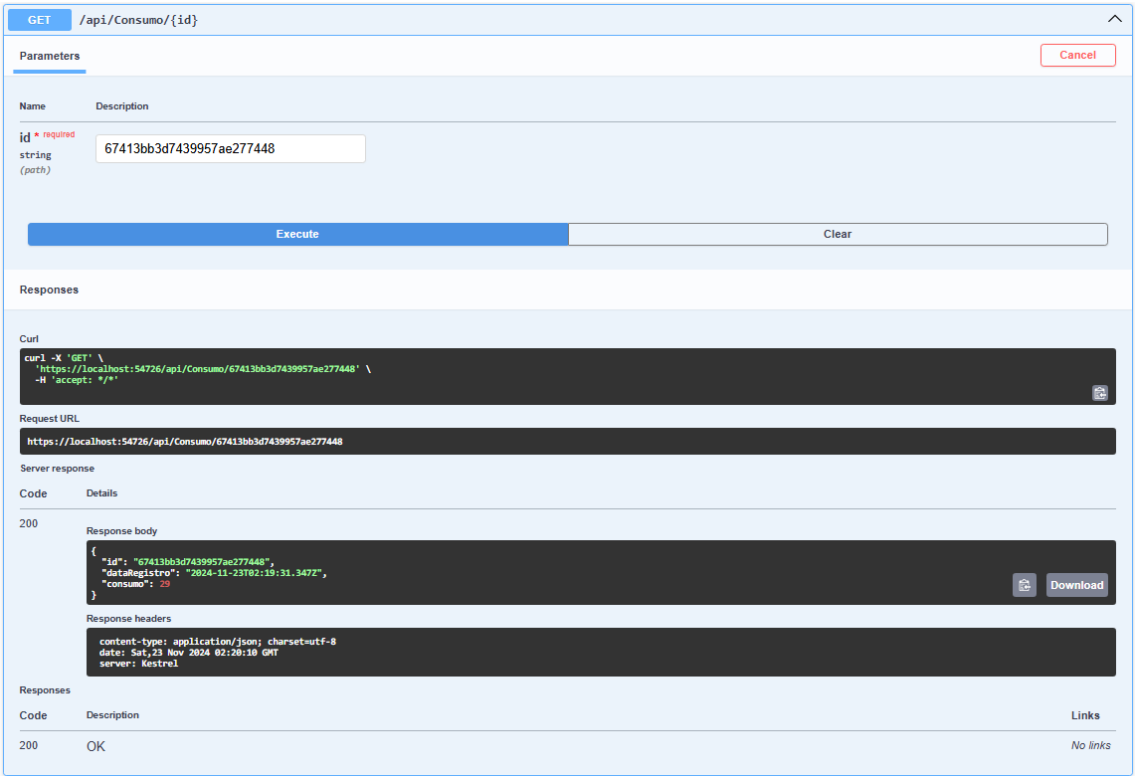
**Exemplo de URL**:

**https://localhost:54726/api/Consumo/67413bb3d7439957ae277448**

**Resposta**

- **Código de Status**: 200 OK
    - **Corpo da Resposta**: O objeto ConsumoModel correspondente ao ID informado.

**Exemplo de Resposta**:




- **Código de Status**: 404 Not Found

- **Mensagem**: Caso o ID fornecido não exista no banco de dados.

**Exemplo de Resposta (Registro não encontrado)**:


{

"message": "Registro não encontrado"

}


**Possíveis Erros**

- **500 Internal Server Error**: Se ocorrer um erro inesperado no servidor

---

### 1. Conectando com o MongoDB

Primeiro, precisamos fazer nossa API se conectar ao banco de dados MongoDB. Para isso, usamos o código abaixo, que vai configurar a conexão.

```
[ApiController]
[Route("api/[controller]")]
1 reference
public class ConsumoController : ControllerBase
{
    private readonly IMongoCollection<ConsumoModel> _consumos;

    0 references
    public ConsumoController(IMongoClient mongoClient, IConfiguration configuration)
    {
        var databaseName = configuration.GetSection("MongoDB:DatabaseName").Value;
        var collectionName = configuration.GetSection("MongoDB:CollectionName").Value;

        if (string.IsNullOrEmpty(databaseName) || string.IsNullOrEmpty(collectionName))
        {
            throw new ArgumentException("Configurações do MongoDB não estão completas.");
        }

        var database = mongoClient.GetDatabase(databaseName);
        _consumos = database.GetCollection<ConsumoModel>(collectionName);
    }
```

### 2. Estrutura dos Dados - ConsumoModel

O **ConsumoModel** é o "formato" dos dados que vamos salvar no MongoDB. Aqui temos o ID (gerado pelo MongoDB), a data e o valor do consumo:

```
using MongoDB.Bson;
using MongoDB.Bson.Serialization.Attributes;

3 references
public class ConsumoModel
{
    [BsonId]
    [BsonRepresentation(BsonType.ObjectId)]
    3 references
    public string? Id { get; set; } = null; // Permite valores nulos para que o MongoDB gere automaticamente o Id

    [BsonElement("dataRegistro")]
    1 reference
    public DateTime DataRegistro { get; set; } = DateTime.UtcNow;

    [BsonElement("consumo")]
    2 references
    public int Consumo { get; set; }
}
```

### 3. Como funciona cada endpoint

### 3.1 POST /consumo - Registrar Consumo

Quando a gente manda um dado de consumo para a API, ela vai registrar no banco de dados.

**Exemplo de requisição (POST):**

json

Copiar código

{

"consumo": 120

}

**Código:**

```
[HttpPost]
0 references
public async Task<IActionResult> RegisterConsumo([FromBody] ConsumoModel consumo)
{
    if (consumo == null)
    {
        return BadRequest(new { message = "Dados inválidos" });
    }

    try
    {
        // Remover o ID enviado pelo cliente, pois será gerado automaticamente pelo MongoDB
        consumo.Id = null;

        // Configurar a data de registro automaticamente
        consumo.DataRegistro = DateTime.UtcNow;

        // Gerar um valor de consumo aleatório caso não seja fornecido ou seja inválido
        if (consumo.Consumo <= 0)
        {
            Random random = new Random();
            consumo.Consumo = random.Next(1, 100); // Gera um valor entre 1 e 100
        }

        await _consumos.InsertOneAsync(consumo);
        return CreatedAtAction(nameof(GetConsumoById), new { id = consumo.Id }, consumo);
    }
    catch (Exception ex)
    {
        return StatusCode(500, new { message = "Erro ao registrar o consumo", error = ex.Message });
    }
}
```

**Resposta (quando funciona):**

json

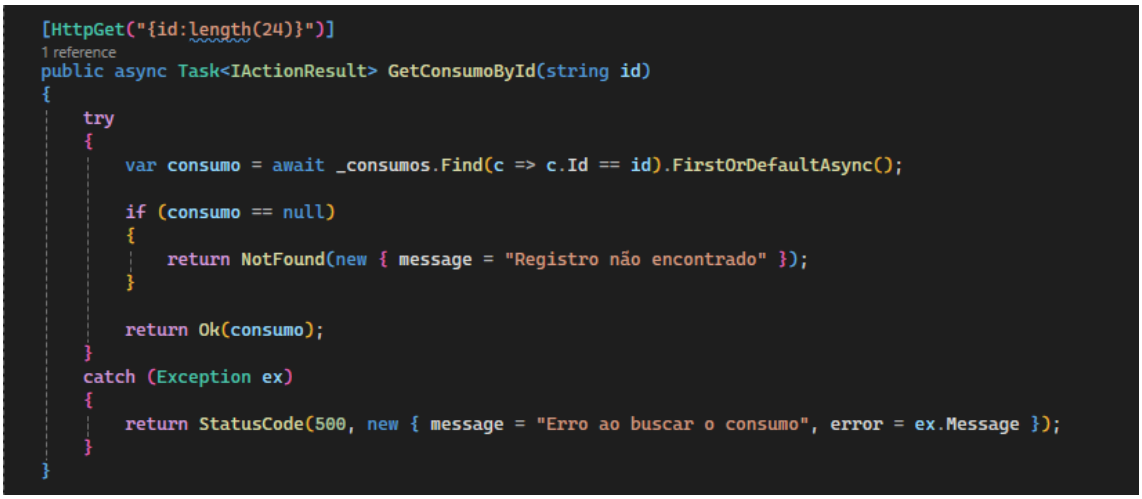
Copiar código

```
{
 "id": "64cf11a2e5f47c29d3e5f8b5",
 "dataRegistro": "2024-11-23T02:13:51.000Z",
 "consumo": 120
}
```

### 3.2 GET /consumo - Consultar Todos os Registros

Esse endpoint pega todos os registros de consumo que já estão no banco de dados.

**Código:**



```
[HttpGet("{id:length(24)}")]
1 reference
public async Task<IActionResult> GetConsumoById(string id)
{
    try
    {
        var consumo = await _consumos.Find(c => c.Id == id).FirstOrDefaultAsync();

        if (consumo == null)
        {
            return NotFound(new { message = "Registro não encontrado" });
        }

        return Ok(consumo);
    }
    catch (Exception ex)
    {
        return StatusCode(500, new { message = "Erro ao buscar o consumo", error = ex.Message });
    }
}
```

**Resposta (quando tem dados):**

json

Copiar código

```
[
 {
   "id": "64cf11a2e5f47c29d3e5f8b5",
   "dataRegistro": "2024-11-23T02:13:51.000Z",
```

```
  "consumo": 120
 },
 {
  "id": "64cf11a2e5f47c29d3e5f8b6",
  "dataRegistro": "2024-11-22T11:35:21.000Z",
  "consumo": 85
 }
]
```


### 3.3 GET /consumo/{id} - Consultar um Consumo Específico

Esse endpoint permite que a gente procure um consumo específico, usando o ID que foi gerado.

**Exemplo de requisição (GET):**

bash

Copiar código

GET /consumo/64cf11a2e5f47c29d3e5f8b5

**Código:**

```
[HttpGet]
0 references
public async Task<IActionResult> GetConsumos()
{
    try
    {
        var consumos = await _consumos.Find(_ => true).ToListAsync();

        if (consumos.Count == 0)
        {
            return NotFound(new { message = "Nenhum registro encontrado" });
        }

        return Ok(consumos);
    }
    catch (Exception ex)
    {
        return StatusCode(500, new { message = "Erro ao buscar os consumos", error = ex.Message });
    }
}
```

**Resposta (quando funciona):**

json

Copiar código

```
{
  "id": "64cf11a2e5f47c29d3e5f8b5",
  "dataRegistro": "2024-11-23T02:13:51.000Z",
  "consumo": 120
}
```

**Resposta (quando não encontra o ID):**

json

Copiar código

```
{
  "message": "Registro não encontrado"
}
```

Claro! Aqui está uma versão mais simples e "humanizada" do texto, como se fosse de um aluno do 3º ano de Sistemas de Informação:

---

**Atividade 2: Integração com MongoDB**

**Objetivo:**

O objetivo dessa atividade é fazer a nossa API funcionar com o MongoDB para guardar e pegar dados sobre o consumo de energia. A API tem duas principais rotas que vamos usar:

- **POST /consumo**: Para adicionar novos dados de consumo de energia.
- **GET /consumo**: Para pegar todos os dados de consumo já registrados.
- **GET /consumo/{id}**: Para pegar um dado de consumo específico, usando o ID.

Vamos usar o **MongoDB** para armazenar essas informações.

---

**1. Conectando com o MongoDB**

Primeiro, precisamos fazer nossa API se conectar ao banco de dados MongoDB. Para isso, usamos o código abaixo, que vai configurar a conexão.

csharp

Copiar código

```
public ConsumoController(IMongoClient mongoClient, IConfiguration
configuration)

{

  var databaseName =
configuration.GetSection("MongoDB:DatabaseName").Value;

  var collectionName =
configuration.GetSection("MongoDB:CollectionName").Value;

  var database = mongoClient.GetDatabase(databaseName);

  _consumos = database.GetCollection<ConsumoModel>(collectionName);

}
```

**2. Estrutura dos Dados - ConsumoModel**

O **ConsumoModel** é o "formato" dos dados que vamos salvar no MongoDB. Aqui temos o ID (gerado pelo MongoDB), a data e o valor do consumo:

csharp

Copiar código

```
public class ConsumoModel

{

  [BsonId]

  [BsonRepresentation(BsonType.ObjectId)]

  public string? Id { get; set; } // ID gerado automaticamente pelo MongoDB


  [BsonElement("dataRegistro")]

  public DateTime DataRegistro { get; set; } // Data do registro


  [BsonElement("consumo")]

  public int Consumo { get; set; } // Valor do consumo de energia (kWh)
```

}

**3. Como funciona cada endpoint**

**3.1 POST /consumo - Registrar Consumo**

Quando a gente manda um dado de consumo para a API, ela vai registrar no banco de dados.

**Exemplo de requisição (POST):**

json

Copiar código

```
{
  "consumo": 120
}
```

**Código:**

csharp

Copiar código

```
[HttpPost]
public async Task<IActionResult> RegisterConsumo([FromBody] ConsumoModel consumo)
{
    if (consumo == null)
    {
        return BadRequest(new { message = "Dados inválidos" });
    }

    if (string.IsNullOrEmpty(consumo.Id))
    {
        consumo.Id = ObjectId.GenerateNewId().ToString();  // Cria um ID novo
    }

    consumo.DataRegistro = DateTime.UtcNow;  // Marca a data que foi registrado
```

```
  if (consumo.Consumo <= 0)
  {
    Random random = new Random();
    consumo.Consumo = random.Next(1, 100);  // Se não passar um valor de consumo válido, gera um aleatório
  }

  await _consumos.InsertOneAsync(consumo);
  return Created("", consumo);  // Retorna o consumo que foi registrado
}
```

**Resposta (quando funciona):**

json

Copiar código

```
{
 "id": "64cf11a2e5f47c29d3e5f8b5",
 "dataRegistro": "2024-11-23T02:13:51.000Z",
 "consumo": 120
}
```

**3.2 GET /consumo - Consultar Todos os Registros**

Esse endpoint pega todos os registros de consumo que já estão no banco de dados.

**Código:**

csharp

Copiar código

```
[HttpGet]
public async Task<IActionResult> GetConsumos()
{
  var consumos = await _consumos.Find(_ => true).ToListAsync();
```

```
    if (consumos.Count == 0)
    {
        return NotFound(new { message = "Nenhum registro encontrado" });
    }

    return Ok(consumos);  // Retorna todos os registros de consumo
}
```

**Resposta (quando tem dados):**

json

Copiar código

```
[
  {
    "id": "64cf11a2e5f47c29d3e5f8b5",
    "dataRegistro": "2024-11-23T02:13:51.000Z",
    "consumo": 120
  },
  {
    "id": "64cf11a2e5f47c29d3e5f8b6",
    "dataRegistro": "2024-11-22T11:35:21.000Z",
    "consumo": 85
  }
]
```

**3.3 GET /consumo/{id} - Consultar um Consumo Específico**

Esse endpoint permite que a gente procure um consumo específico, usando o ID que foi gerado.

**Exemplo de requisição (GET):**

bash

Copiar código

GET /consumo/64cf11a2e5f47c29d3e5f8b5

**Código:**

csharp

Copiar código

```
[HttpGet("{id}")]
public async Task<IActionResult> GetConsumoById(string id)
{
   var consumo = await _consumos.Find(c => c.Id == id).FirstOrDefaultAsync();

   if (consumo == null)
   {
      return NotFound(new { message = "Registro não encontrado" });
   }

   return Ok(consumo);  // Retorna o consumo específico
}
```

**Resposta (quando funciona):**

json

Copiar código

```
{
  "id": "64cf11a2e5f47c29d3e5f8b5",
  "dataRegistro": "2024-11-23T02:13:51.000Z",
  "consumo": 120
}
```

**Resposta (quando não encontra o ID):**

json

Copiar código

{

"message": "Registro não encontrado"

}

---

**4. Tratamento de Erros**

**4.1 Falha na Conexão com o MongoDB**

Se a API não conseguir se conectar ao MongoDB, a resposta será um erro **500**, que significa que algo deu errado no servidor.

**Exemplo de resposta:**

json

Copiar código

{

"message": "Erro ao conectar ao banco de dados."

}

**4.2 Dados Inválidos**

Se o consumo informado for 0 ou negativo, a API vai retornar um erro **400**, indicando que os dados não são válidos.

**Exemplo de resposta:**

json

Copiar código

{

"message": "Dados inválidos"

}

**4.3 Registro Não Encontrado**

Se não encontrar o ID fornecido na consulta, a API retorna um erro **404** dizendo que o registro não foi encontrado.

**Exemplo de resposta:**

json

Copiar código

{

"message": "Registro não encontrado"

}

---

**5. Exemplo Completo de Como Funciona**

1. **Registrar Consumo (POST /consumo):**

Requisição:

json

Copiar código

{

"consumo": 150

}

Resposta:

json

Copiar código

{

"id": "64cf11a2e5f47c29d3e5f8b5",

"dataRegistro": "2024-11-23T02:13:51.000Z",

"consumo": 150

}

2. **Consultar Todos os Registros (GET /consumo):**

Resposta:

json

Copiar código

[

{

"id": "64cf11a2e5f47c29d3e5f8b5",

"dataRegistro": "2024-11-23T02:13:51.000Z",

```
  "consumo": 150
 },
 {
  "id": "64cf11a2e5f47c29d3e5f8b6",
  "dataRegistro": "2024-11-22T11:35:21.000Z",
  "consumo": 120
 }
]
```


3. **Consultar Consumo Específico (GET /consumo/64cf11a2e5f47c29d3e5f8b5):**

Resposta:

json

Copiar código


```
{
 "id": "64cf11a2e5f47c29d3e5f8b5",
 "dataRegistro": "2024-11-23T02:13:51.000Z",
 "consumo": 150
}
```