

|  |  |  |
| --- | --- | --- |
| **Sprint Documentation** | | |
| **Sprint #** | **Start Date: 30/09** | **Final Date: 14/10** |
| **Team Members** | | |
| Franciane Ramos Franco | | |
|  | | |
|  | | |
|  | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Sprint Backlog** | | | |
| Task | Description | Start Date | Final date |
| #36 | Sprint 02 - Desenvolver a API para Manufacturer | 30/09 | 13/10 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Task Description** | | | | | |
| Task | Description | Assigned To | Status | Estimated Hours | Logged Hours |
| #36 | Desenvolver a API para a classe Manufacturer | Franciane | Concluido | 4 | 2 |
|  |  |  |  |  |  |

**Sprint Results**

**1 - Desenvolvimento da API**

O desenvolvimento da API começou com a implementação da classe “Manufacturer” na camada de modelo (model). A classe foi definida com os seguintes atributos privados:

id: int

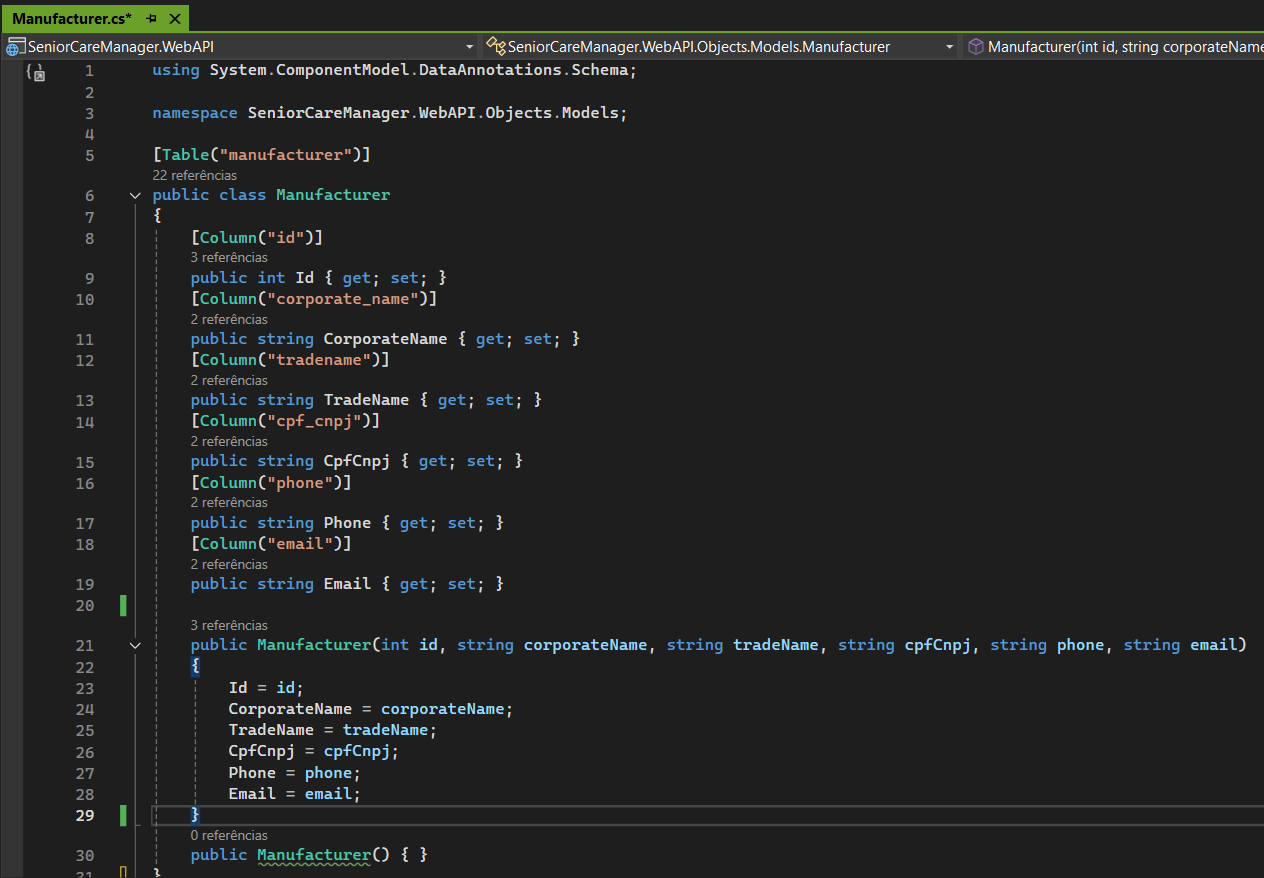
CorporateName: string

Tradename: string

CpfCnpj: string

Phone: string

Email: string



**2 - Modelagem da Classe:**

A classe “Manufacturer” foi criada com todos os atributos necessários, garantindo que cada um tivesse o tipo de dado apropriado.

Foram implementados métodos getters e setters para cada atributo, permitindo o acesso e a modificação dos dados de forma controlada.

**3 - Criação do Controller:**

Um controlador (controller) foi desenvolvido para gerenciar as requisições HTTP relacionadas aos fabricantes.

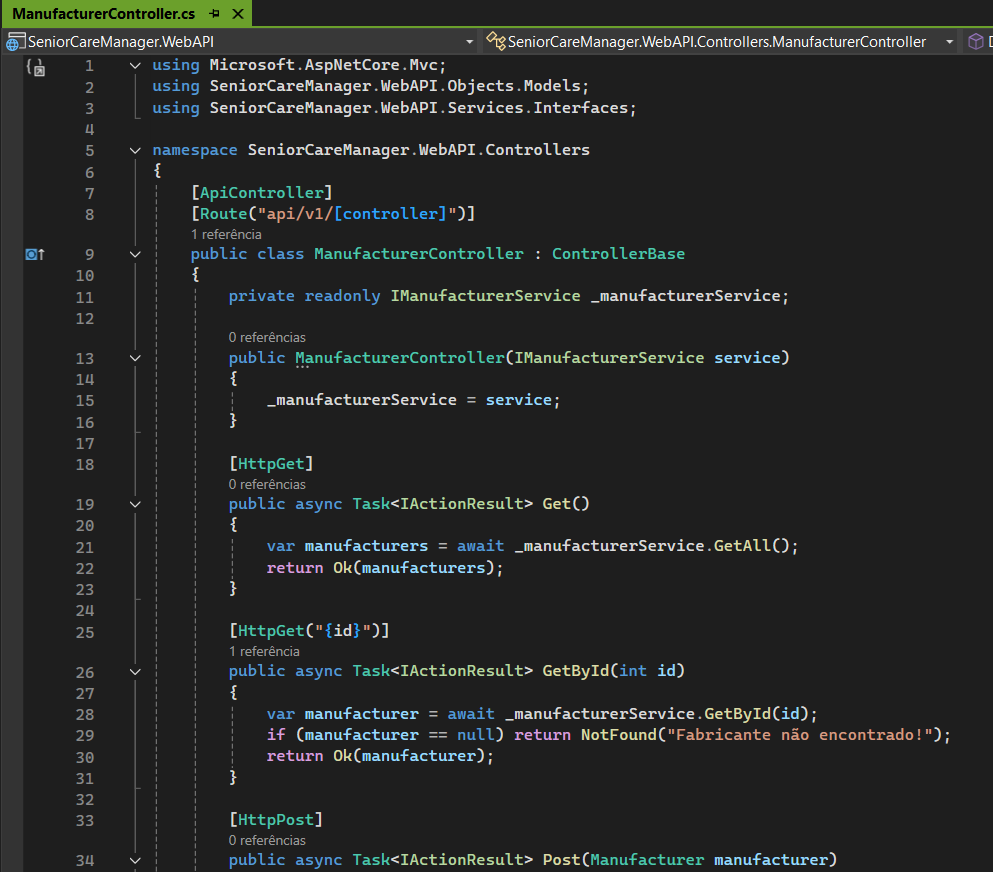
Endpoints foram definidos para operações CRUD (Create, Read, Update, Delete), como:

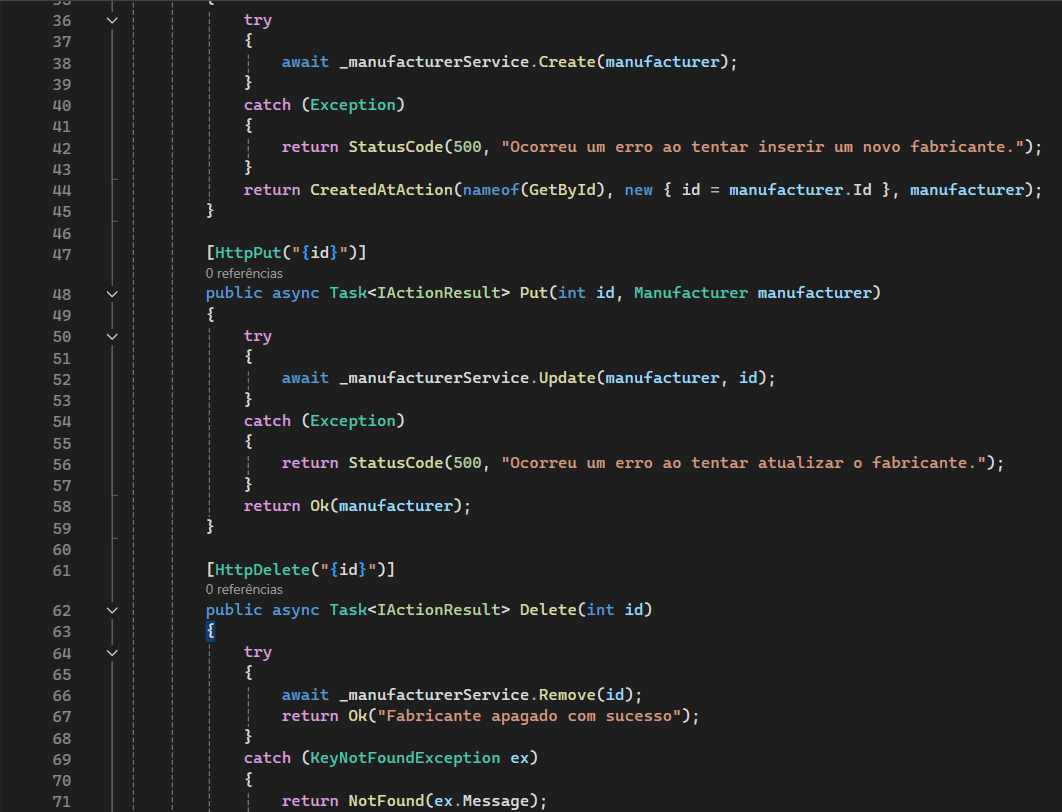
POST /manufacturers: Para criar um novo fabricante.

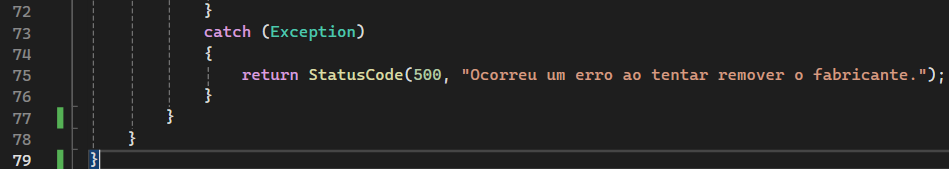
GET /manufacturers/{id}: Para obter os detalhes de um fabricante específico.

PUT /manufacturers/{id}: Para atualizar as informações de um fabricante.

DELETE /manufacturers/{id}: Para deletar um fabricante.

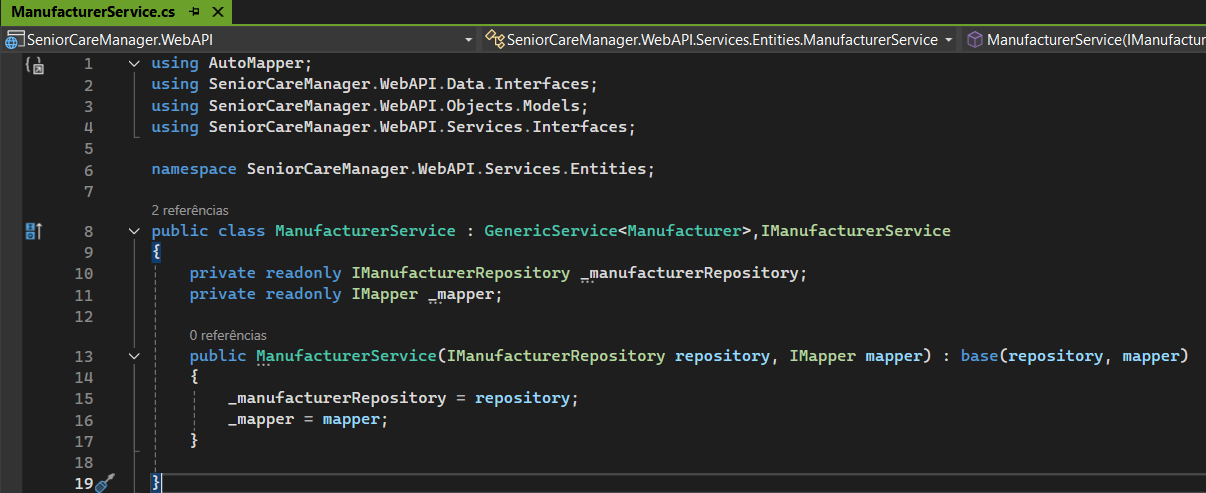


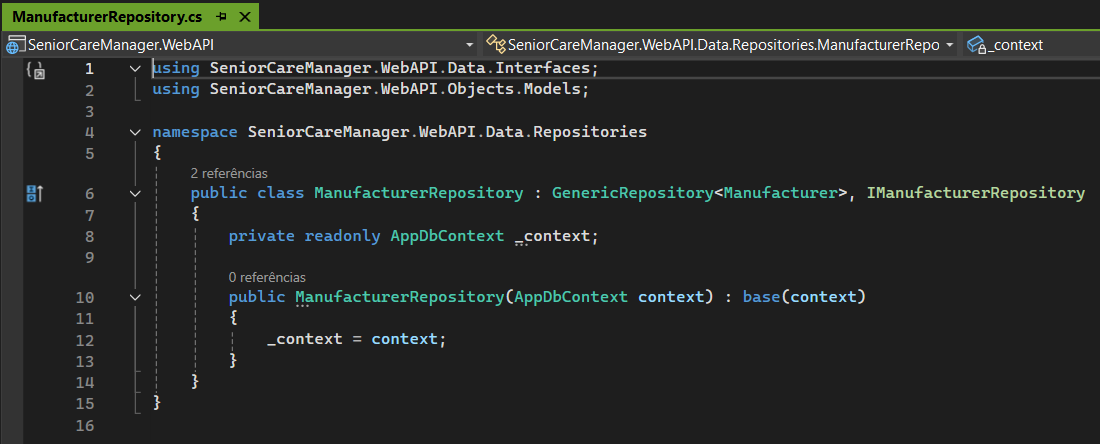




**4 - Implementação das Classes Necessárias:**

Além da classe “Manufacturer”, outras classes auxiliares foram criadas conforme necessário, como classes de serviço (service) para encapsular a lógica de negócios e classes de repositório (repository) para interagir com o banco de dados.

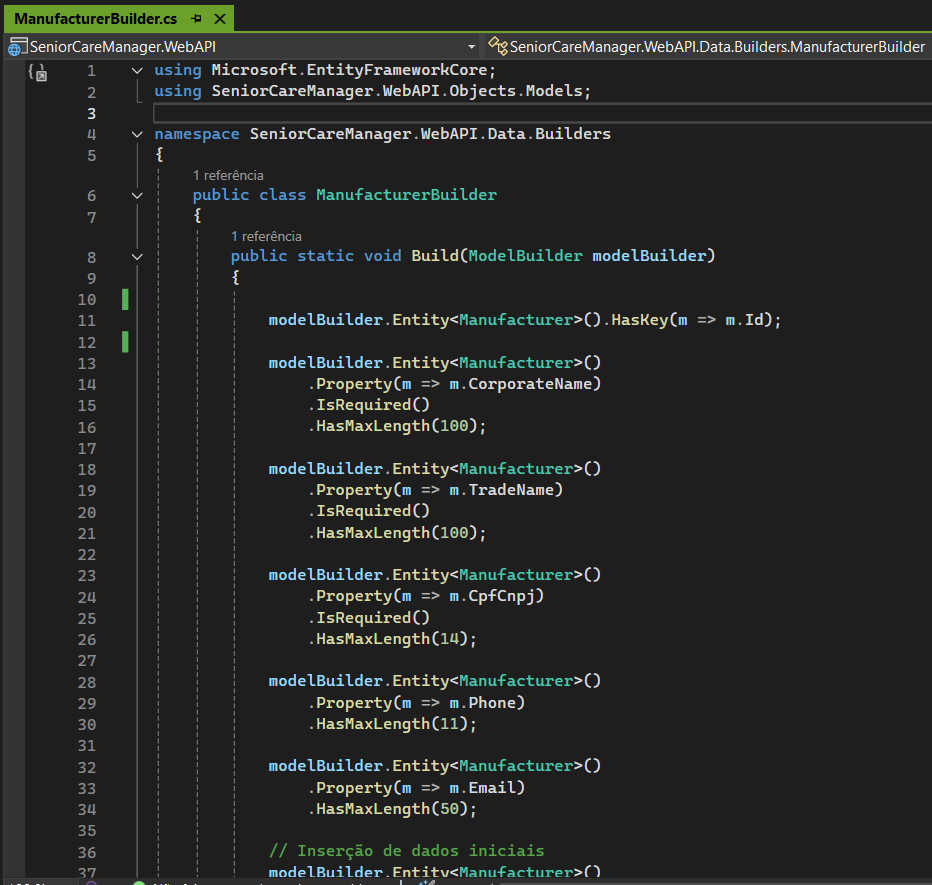


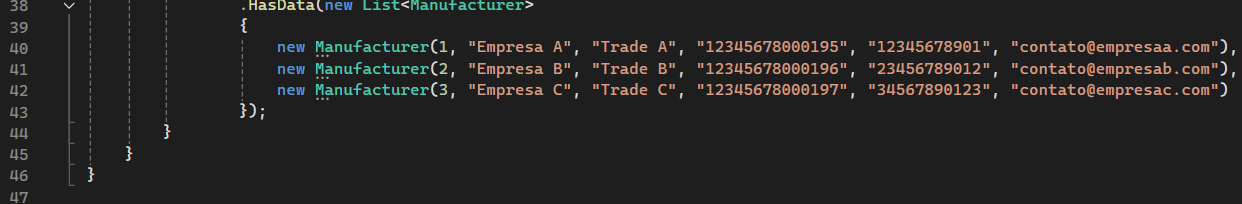


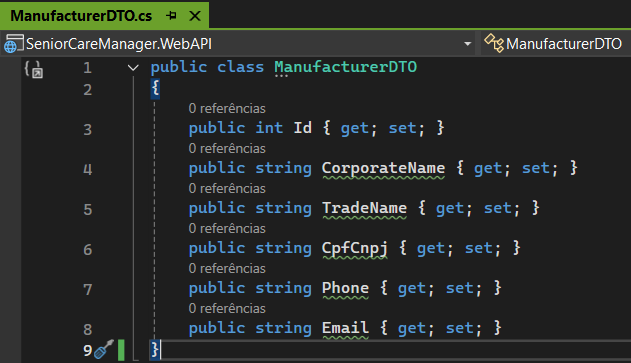
**5 - Integração e Referências:**

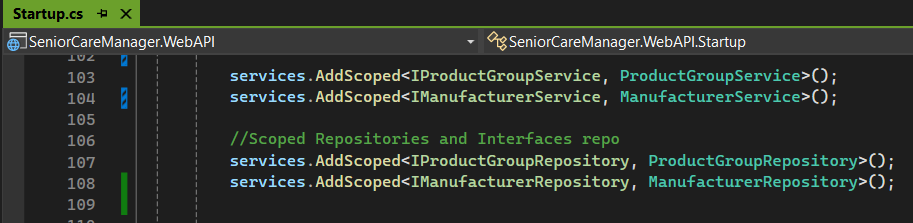
Todas as referências e dependências foram configuradas para garantir a harmonia entre as diferentes partes do sistema.

Foram realizadas integrações com o banco de dados para persistência dos dados dos fabricantes.







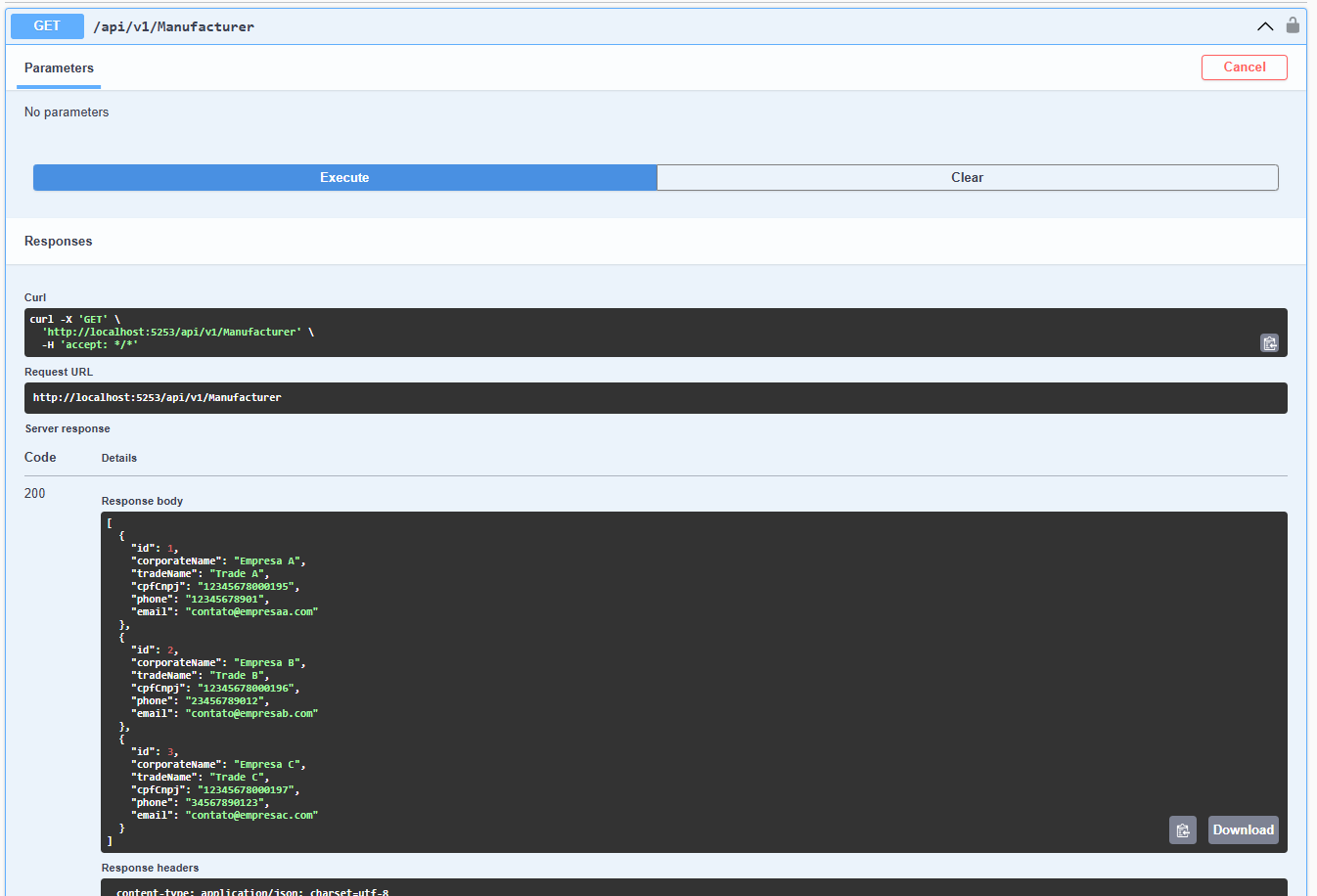


**6- Validação e Tratamento de Erros:**

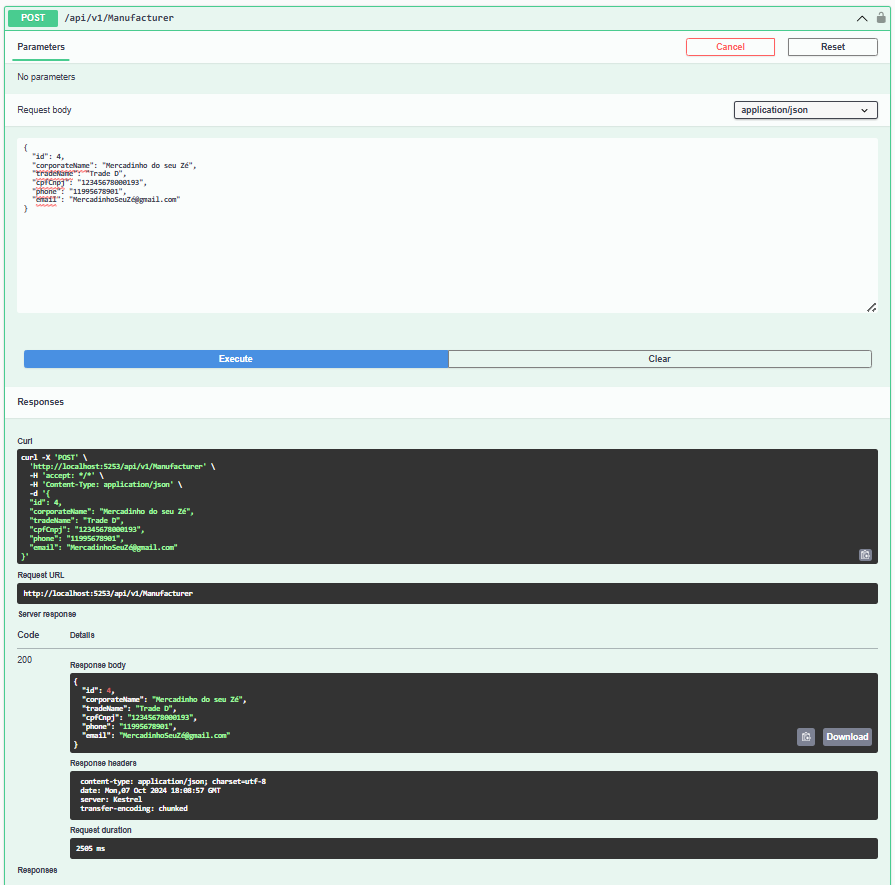
Validações foram implementadas para garantir a integridade dos dados (por exemplo, verificar se o CpfCnpj é único).

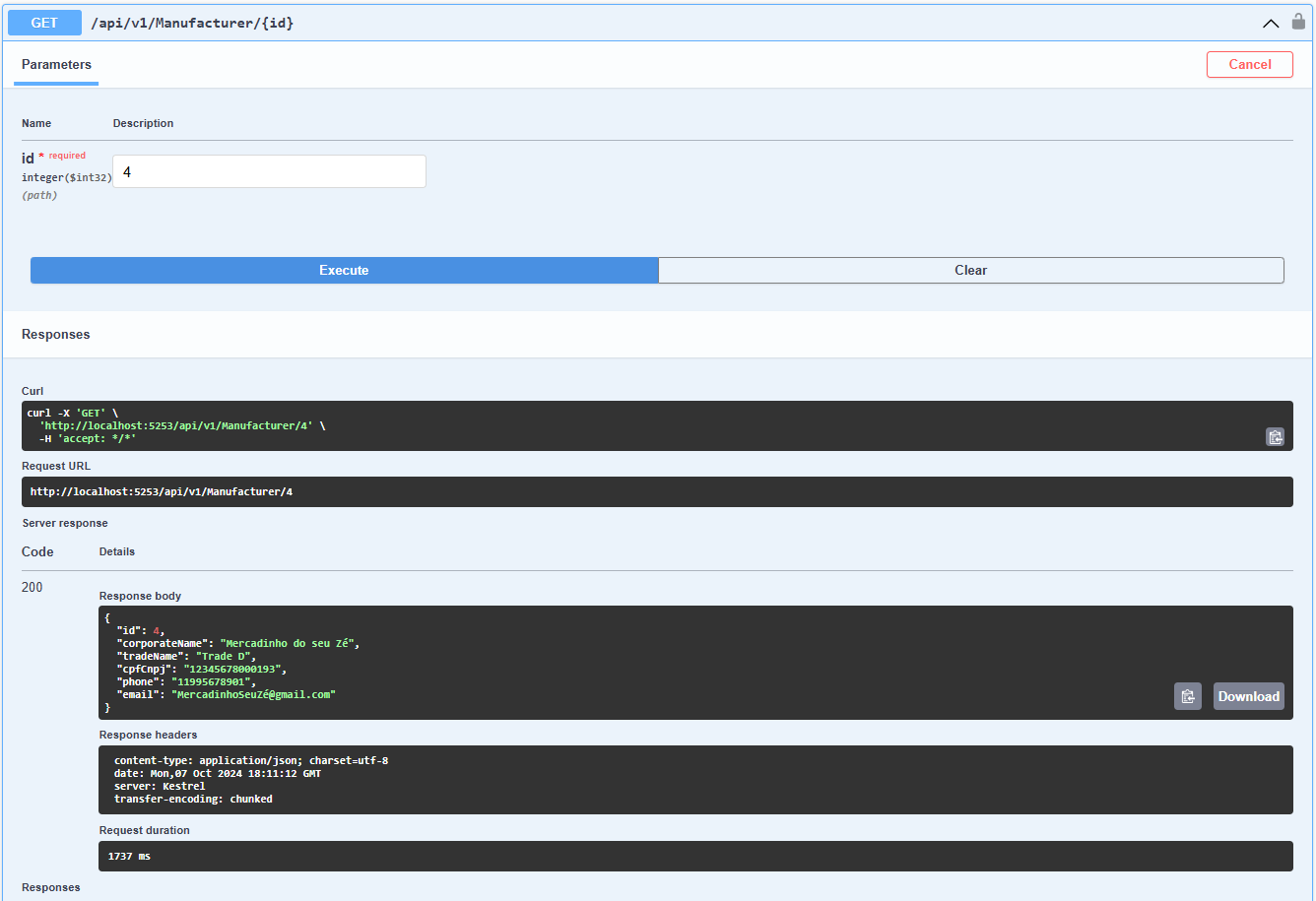
Mecanismos de tratamento de erros foram adicionados para lidar com situações como dados inválidos ou fabricantes não encontrados, retornando respostas apropriadas (por exemplo, erro 404 para fabricante não encontrado).

**7 – Teste Realizados:**

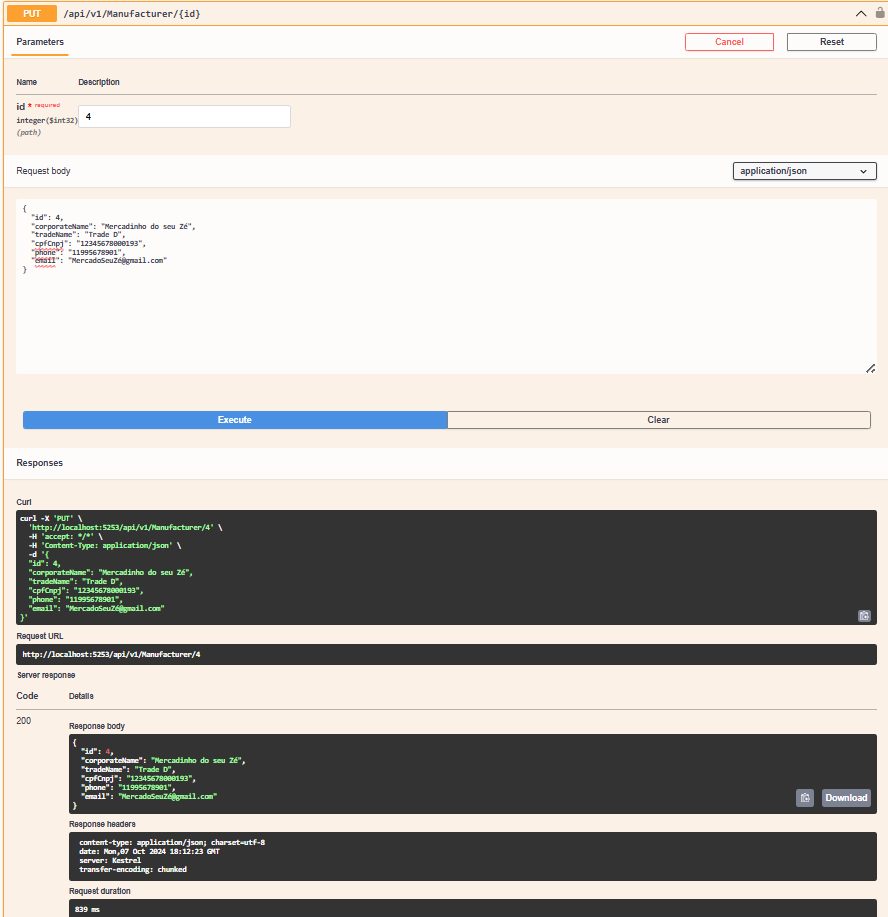
**GET:**

POST:



GET ID:

PUT:



DELETE:

