Bootcamp: Arquitetura de Software XP

Aluno: Denilson Oliveira de Jesus

DOCUMENTAÇÃO TÉCNICA, CRUD DE PRODUTOS DE INFORMÁTICA (FLASK + MVC)

# 1. Introdução

Este documento apresenta a documentação técnica do sistema CRUD de Produtos de Informática, na qual escolhir. Desenvolvido como parte do desafio final – Arquiteto(a) de Software. Nele o objetivo do projeto é implementar uma aplicação completa no padrão MVC utilizando Flask, com API REST, interface web e banco de dados SQLite com migrações automáticas via Alembic.

# 2. Arquitetura do Sistema

A aplicação adota o padrão de arquitetura MVC (Model–View–Controller), com camadas bem definidas para controle, lógica de negócio, acesso a dados e persistência. O projeto utiliza Blueprints do Flask para modularização e separa responsabilidades entre API, interface web e banco de dados.

Estrutura de diretórios:

crud\_produtos/  
|  
|- ApiApplication.py  
 |- controller/  
 |- service/  
 |- repository/  
 |- model/  
 |- templates/  
|- extensions.py

# 3. Fluxo de Dados

Fluxo principal segue a sequência: User -> Controller (Web/API) -> Service -> Repositor -> Model -> Banco de Dados. A camada controller recebe as requisições, a service aplica as regras de negócio, a repository executa as operações no banco via SQLAlchemy e o model representa a estrutura ORM persistida.

**Funcionalidades da API:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tipo de Operação** | **Método HTTP** | **Endpoint** | **Descrição** | **Exemplo de Uso** | **Código de Retorno** |
| **Create** | POST | /produtos | Cria um novo produto com base nos dados JSON fornecidos. | {"nome": "Notebook Lenovo", "preco": 4850.90, "estoque": 12} | 201 Created |
| **Read — Find All** | GET | /produtos | Retorna todos os produtos cadastrados no sistema. | — | 200 OK |
| **Read — Find By ID** | GET | /produtos/<id> | Retorna um produto específico de acordo com o ID informado. | /produtos/1 | 200 OK / 404 Not Found |
| **Read — Find By Name** | GET | /produtos/nome/<nome> | Retorna uma lista de produtos cujo nome contenha o termo pesquisado (busca parcial). | /produtos/nome/notebook | 200 OK |
| **Count** | GET | /produtos/contar | Retorna o número total de produtos cadastrados. | — | 200 OK |
| **Update** | PUT | /produtos/<id> | Atualiza os dados de um produto existente. | {"nome": "Notebook Dell", "preco": 5199.90, "estoque": 8} | 200 OK / 404 Not Found |
| **Delete** | DELETE | /produtos/<id> | Exclui um produto do sistema com base no ID informado. | /produtos/1 | Produto {id} excluído com sucesso |

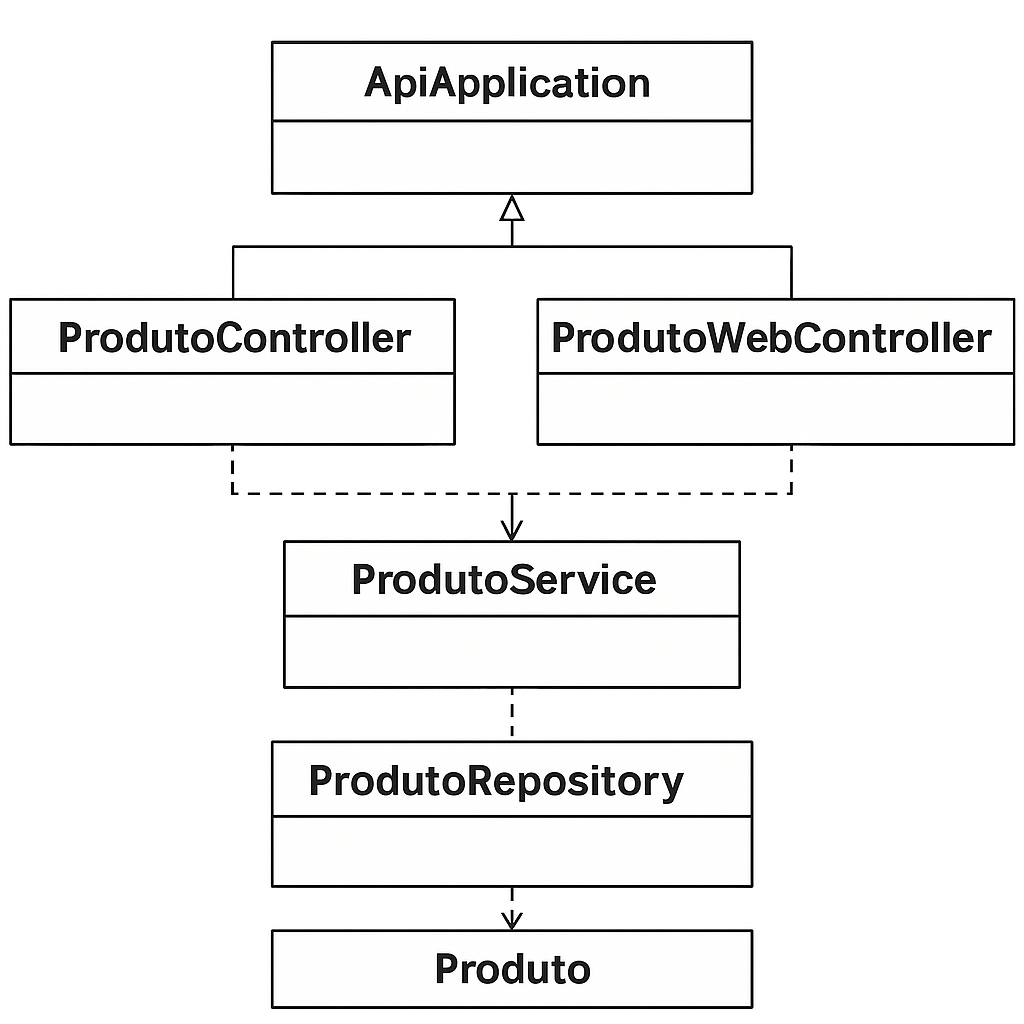
# 4. Banco de Dados

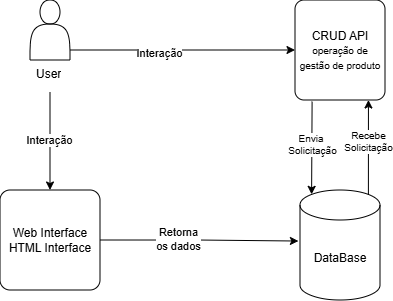
O sistema utiliza SQLite como banco de dados local. A tabela principal é **produtos**, contendo os campos **id, nome, preco e estoque**. As migrações são controladas pelo Alembic (Flask-Migrate).

# 5. Diagrama UML

O diagrama de classes a seguir representa a arquitetura MVC e a relação entre as camadas Controller, Service, Repository e Model, conectadas ao banco de dados SQLite.

[Link de compartilhamento:](https://viewer.diagrams.net/?tags=%7B%7D&lightbox=1&highlight=0000ff&edit=_blank&layers=1&nav=1&title=Diagrama%20C4.drawio&dark=0#R%3Cmxfile%3E%3Cdiagram%20name%3D%22P%C3%A1gina-1%22%20id%3D%22B7blOIqZ6MP5vHv8doVT%22%3E7VnbcuI4EP0aHklZvmEeue3MVJHabDJTmTwKWxjtGMsriwDz9duy5YtsLt4kJMzuVlFgHUuydLp1utv0rMl694njZHXLAhL1TCPY9axpzzRdcwDfEtjngGUOcyDkNMghVAEP9CdRoKHQDQ1IqnUUjEWCJjroszgmvtAwzDnb6t2WLNKfmuCQtIAHH0dt9JEGYpWjXrEtiX8mNFwVT0au2t8aF53VTtIVDti2BlmznjXhjIn8ar2bkEhyV%2FCSj%2FvtyN1yYZzEosuAp%2FiP%2FjA0vj7%2BTGaxM7r9Lpx%2B38xnecbRRm1YLVbsCwZg3Ym8xL5gvGeNtysqyEOCfQluweiArcQ6ghaCSzUh4YLsjq4UlfsHvyFsTQTfQxc1wDQUZcpnHNXcVgawFbSqce8qDCuTh%2BXEFStwoYj5ByRZLZK%2BpYS3iIINw0rGqeDsB5mwSLI1jVkMt8dLGkUNKAUKaRzOyVKu366Qe7UlCTEgchllbrOiQUBiwDgTWOBF9lgD2gmjscj27IzhAyxMjBun58CaJtBGVRs%2BsjsXExbDMjHN7EFwKrYklWtfslioM4jMD7E1Qt2MbV3K2HbL2JP7b%2FBjjO6%2B9Ew3kjQF9BkuQ5FRkEOSOc0d3L82rLjRTzNOR9ABGckuH6buFxOxhHDcm1i90SD7tphUTSIlUNpGhxLOgo1gxcNho%2Fnz9TUBrK204a%2BcbeKABMqWmPuF5Y3zlscRDcEXpz6YmfC3cQXbdc4d%2B%2BEBT0DGpVwBtcWQBBAdVBPO0YqFLMbRrELHFauSxqrPnLFEkfcnEWKvyMbSjBq1ZEfF99r1k5wKzm%2Femu7UzFljXzRi2K8c1DduDLMA8pG2axdANThraaPvCKdAmzRmBuZbl%2Fs9bU6gh224T07wqMwqMA%2BJONHPPewenERY0Gd9HYeMrYbeST2su5WuMKUTFVPk61KjGi5TLuMVXjRoKco9gVAa4wNywlJ5xCFRSDseYmmgOV5A0nX6fMqTSSGtGakba4gmub8SEKej0WR66mirnEsNrjKduo%2BcOFhHhQCc2ERDPSzkrZe6QtGFLZcpuYiVnZaRH8kCgC%2BS%2F6WU0baxP3%2B9nR%2Fq8ovpNho0DpjdFu5SpN9Hud2WNaaQMo1xSo6mt%2F4%2BokArt86TuMgNMF%2BUAPZ%2FhJlZft8ImIYoPFWGcd4oQHYg2jvEM7oUz21pawtUHIxkHSYpjnCaUl8nU4%2BXZSArG7XwdyKEdY9WZ6OQfdgGNY6dAxwX2CuDVWnSQveaRyQPtq1gdTbqtSa6cNTzWp6RyVwzw%2F03RLPB2WBmDJFmjKKivN5wNnzzg62y2iqTlQcb3Rju4HRmC41malqJRKULT5osvFYkzqa0VkcxcT5STKxhI140i%2BbOYtJUJe%2BdU2jjv6Mmw3NqYpiGXif3rz45RuhycgKFgqYnyO1WKcNIu7N62DdW2belRgcU6g2Vxu6oNEeK5%2FdRGttrKI35QqVxrIbSDLopDbgO3te6qZN6CV9uV3qz%2BJnKYh6vZaEQL9Kk9hKuVsk9sIj6VDRV61rq%2FPrLX%2BMi2oZO1zdS3DzPu3o1a78ifq2aVaLk5fpViBJ6yau7Shw9p55tGUeVsdC3y0qZ21HKPrQCs82GlL24AmtU68i5Oilrvya5Jz5ZHHpX9b92HfFKNb1MJyzd3h%2BnY9Cs%2FlfOu1d%2FzluzvwE%3D%3C%2Fdiagram%3E%3C%2Fmxfile%3E)



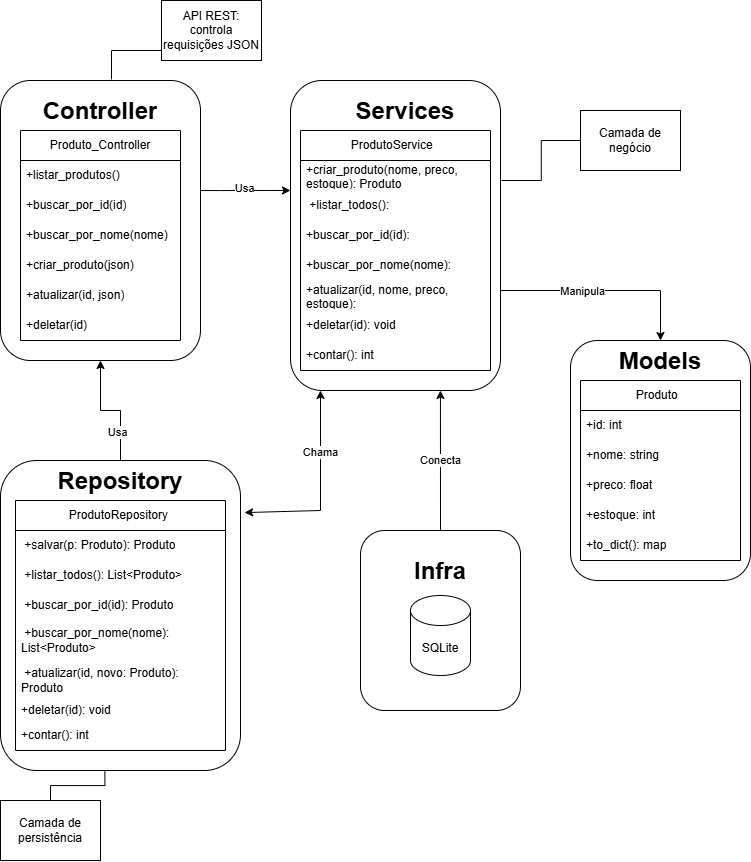


# 6.Diagrama C4 (Instruções para Draw.io)

O modelo C4 fornece uma visão arquitetural em quatro níveis. O diagramas foram criados no Draw.io usando as seguintes orientações:

- Nível 1: Contexto — Mostra o sistema CRUD e suas interações com o usuário e o banco de dados.  
- Nível 2: Containers — Mostra a decomposição em Interface Web, API REST e Banco SQLite.  
- Nível 3: Componentes — Mostra os módulos internos (Controllers, Service, Repository e Model).

[Link de compartilhamento:](https://viewer.diagrams.net/?tags=%7B%7D&lightbox=1&highlight=0000ff&edit=_blank&layers=1&nav=1&dark=auto#R%3Cmxfile%3E%3Cdiagram%20name%3D%22P%C3%A1gina-1%22%20id%3D%226rG6XZuWJmWo0AXHnlHB%22%3E7V3fc6M4Ev5rXJV5yBQ%2FDHYeY2d2bq8yN9nJ3u1mX6YUUGztYOQB2Ynz16%2BEJBBCNiQxmFzxkpiWELj7U3frU4NH7nz19DkB6%2BUXHMJo5Fjh08i9GjnOhefRv0yw4wLPsrlgkaCQixTBLXqGQmgJ6QaFMC11JBhHBK3LwgDHMQxISQaSBD%2BWuz3gqHzVNVjAiuA2AFFV%2BgcKyZJLp86kkP8LosVSXtn2L3jLCsjO4pukSxDiR0Xkfhq58wRjwj%2BtnuYwYrqTeuHn%2FbKnNb%2BxBMakyQl3Wye6vvvf19Vz%2FPOvz7%2F%2FNv3D3Zw7fJQtiDbiC4ubJTupgQRv4hCyQeyRO3tcIgJv1yBgrY%2FU5FS2JKtINFdvStznFiYEPikicZOfIV5BkuxoF9HqCgXuJBCE%2Fh4L9TtStlRU70yFEAiTL%2FKhC63QD0IxL1CSa1CSHxH2ze2SrvyfG2bN2QokCxSfE0y1c0l7WOunTDeynX5asP9zHJMERxFM5ID0%2FtiYvLliCqpBUta3yRyY6vohyqC2RGEIYyorjGjRIxChBZVeBdRK9NpHMdtYM5tdNZs9NZht3JbV7AbYhiGd7OIQJ2SJFzgG0adCqimu6HONmXEzG%2FwNCdkJzwU2BDecEezaBxWbwAgQtC07IpOaxKk3GNFL5AbRJ1LuUuUQKd4kARRnqY5DG8h1agYiFO6QVAa6TBKwU7qtWYf0wA175uvsu6%2Ba%2FvQDv4MCQrlyX48qf1pB1X9TYATWNbinsbAEhsrEY%2FML0WhzKRpWdMJy3MEUPYP7bDyGPKE8Org3G3lXRmAdngb6TM5jprhKKSyZZvi59dFyLbesc370WqjKLvjhIYVkpE%2F5I9hrXDHXTYJDOku%2Fq85Xs176iFYRiJkdHmgv6SGYHYIlisJrsMMbpvSUgOCHPJotcYKeaX8gzU2bEyI8g2uVetyyM8WYma3hjbSkrYm%2BgKdSx2uQEnk39P7BOkXcgnYeemaYELwSndoP2dpEtD2D7%2FdNIVv3JEdz%2Fp7B988ixAzyfc0BkJ592BdgUwqMH3COI0zn6FWMORJQFGkiOZ0j%2BEAOTeaUKh%2FFi%2Bus29W4kHwTuhjvCdvlWT%2Bj2ppbHz02%2F505PbaL48wlrGl8orCmtw9QZkxIofIIGVxoICOAKA7lxag4OL3qoSKw4TaEhtsWMvy9yVyItkXmNbvfpAEDC06%2Bo%2FAMhR%2BUFK3Uc4BQxxDyTw2hycshFNNLn7E%2FA4z6AqOLU8PIvmiIoyBBRdg6%2BzvF8YCivqDIdk4NI3kDtTACZEMN%2FAwSGs2Yvh1rgFKvoNQ0bW4PSnZDKIUwgiQD0gCf3sBnemr4uHbF7B0QbPAJkT%2BVz3dsKGo6fnQl187ZwU4c7F3zck7sUNAW%2FAfnvA4pQ3Rsm%2Bab2BoL5rXDznkanejWsHP%2BxcH%2B7bBzk6oD%2BwJitN5Efabo3H2Ey4spOsbQ2eOS3s%2FFSP2l6GzT%2FopmrM43oTTO27gLZZuca2uUltPaNtQtTLYogOn734RyvTL4T78L5ewloIXWqZ7Bimksvk%2FX3Cja8UBOvw0T%2BoayKc2%2BMGCiRVAY6enyMp%2BRRXyVtk5ggPlHmnDinxv4gc9nAaR9M3TItZsGYOcoNLYJQ%2B0t1Rry2LozkZsgBIfZDggD0rCA6weoTMR2t6B6BbMtNkcGHPUHRyZmu1scVYskXrJDMmCpP1gy8tvdgqnpNkmV367PoQaY9QRmTZPy9shL0zZKiejOcu4tZuAaANI5QEzsdrcAcYyLNrZgTkQibUmOdQBHp%2BBwTDRPt%2)



# 7. Documentação das Camadas

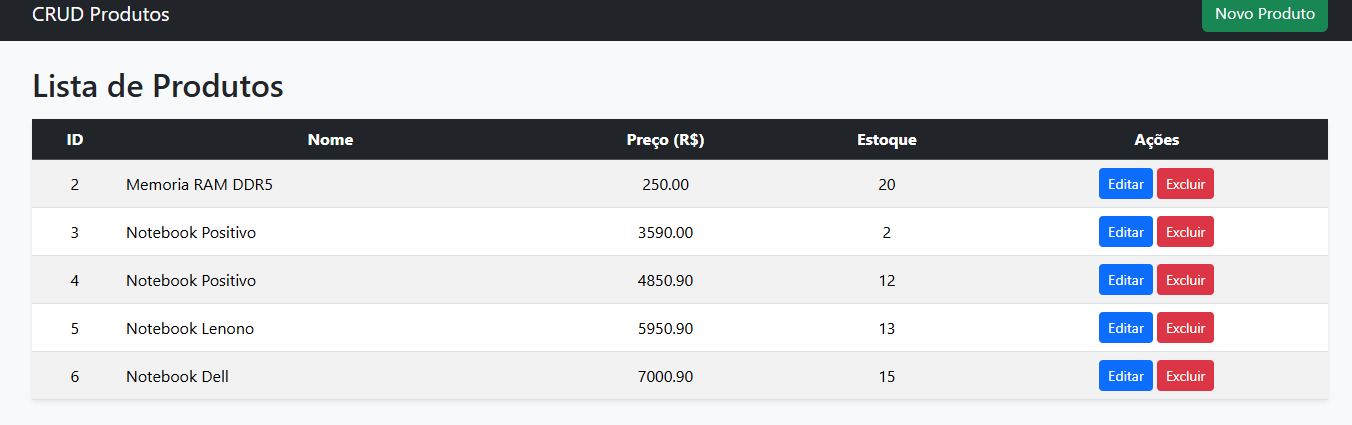
Cada camada da arquitetura possui responsabilidades específicas, conforme a tabela abaixo:

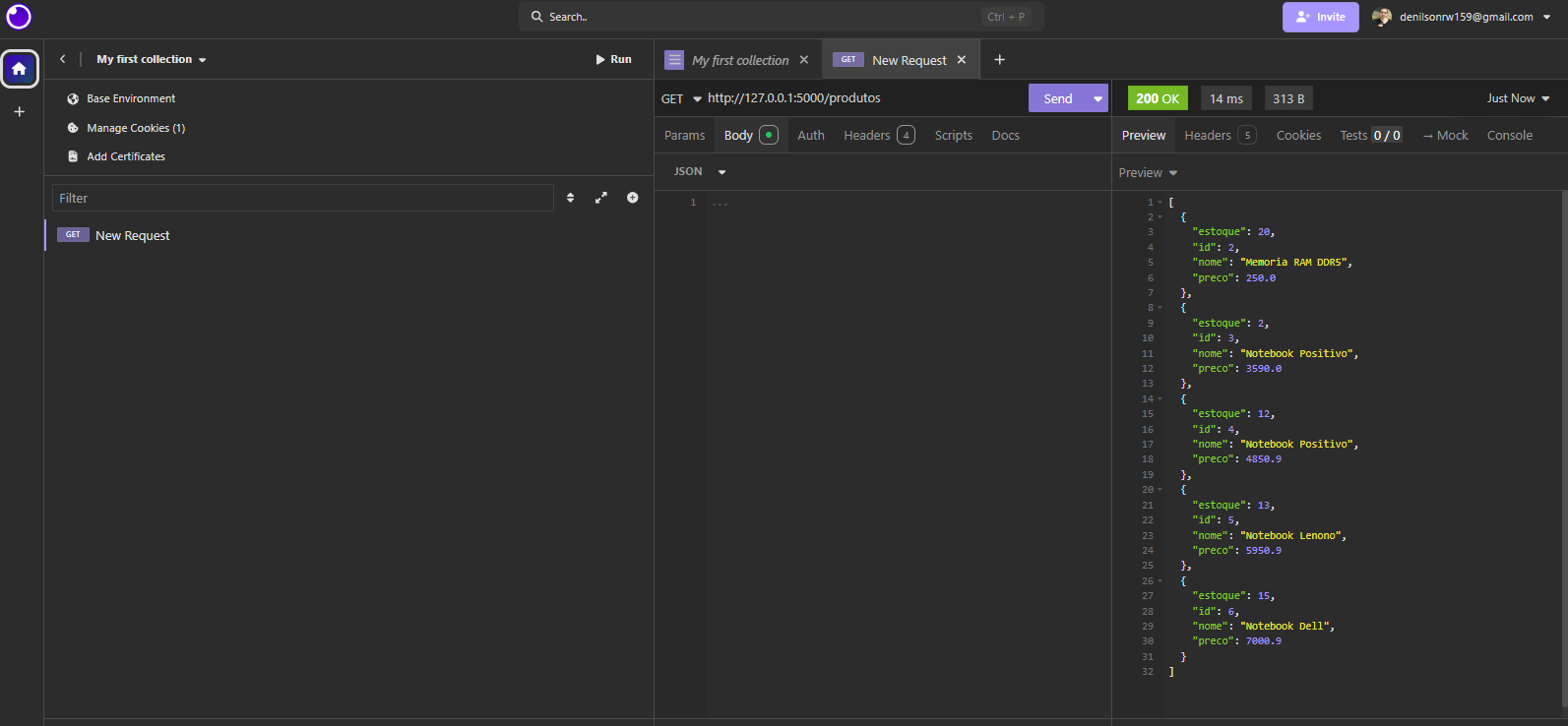
|  |  |  |  |
| --- | --- | --- | --- |
| **Camada** | **Arquivo** | **Descrição** | **Responsabilidade** |
| **Controller** | produto\_controller.py | API REST | Gerencia as requisições JSON da API. |
| **Controller Web** | produto\_web.py | Interface HTML | Renderiza as páginas web e formulários. |
| **Service** | produto\_service.py | Lógica de negócio | Valida dados e orquestra operações CRUD. |
| **Repository** | produto\_repo.py | Acesso a dados | Executa operações SQL via SQLAlchemy. |
| **Model** | produto.py | Estrutura ORM | Define a entidade Produto e seus campos. |
| **Infra** | extensions.py | Configuração | Inicializa SQLAlchemy e Migrate. |

# 8. Testes Realizados

Foram realizados testes completos via interface web e API REST:

- Web: Cadastro, edição, exclusão e listagem de produtos;



  
- API: Operações CRUD via Insomnia;

- Banco: Verificação de persistência no SQLite após reinicialização.

# 9. Conclusão

O sistema CRUD de Produtos de Informática cumpre todos os requisitos propostos no desafio final, adotando uma arquitetura escalável, modular e aderente ao padrão MVC. O projeto está pronto para evoluções futuras, como integração com PostgreSQL, Docker e autenticação de usuários. Desde já agradeço a todos os professos pelo excelente ensino.