**Documentacion de Solucion de Lista**

**Repositorio de Solucion**

<https://github.com/michaelventura01/TASKMANAGER/tree/master>

**Requerimiento**

Desarrollar una pequeña aplicación para gestionar un sistema de gestión de tareas. Esta debe estar compuesta por un backend en .NET Core utilizando Entity Framework Core, aplicando el patrón Onion Architecture y Repository Pattern, y un frontend en Angular.

**Terminologia**

**Onion Architecture** es un patrón de diseño que busca la **separación estricta de responsabilidades** en capas, favoreciendo la **independencia de la lógica de negocio respecto a frameworks, bases de datos y controladores.**

Una **API REST** es una interfaz que permite que clientes (como Angular) interactúen con la aplicación usando HTTP:

**Tecnologias Empleadas**

* Visual Studio 2022 o superior
* EF Core
* Swagger
* Angular 20 (angular cli)

**Arquitectura**

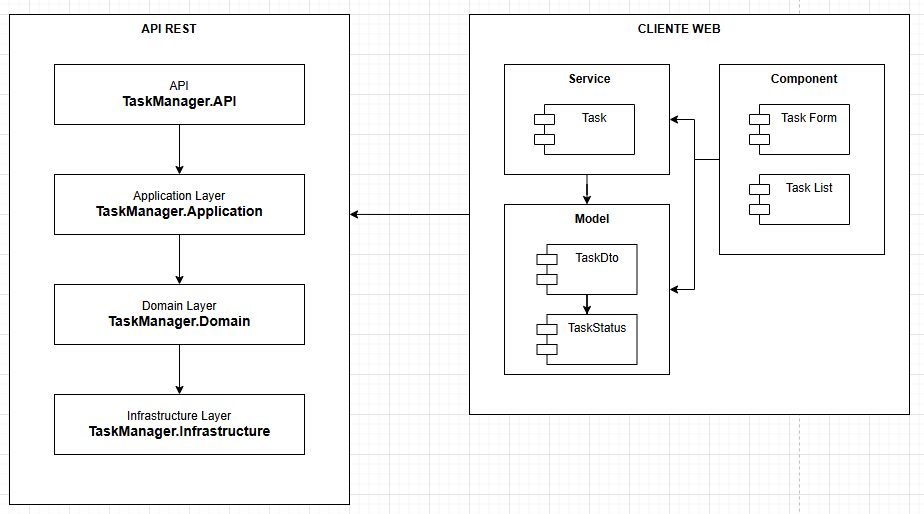
**API**

* TaskManager.API: Capa de presentación (controladores HTTP).
* TaskManager.Application: Lógica de negocio (servicios, DTOs).
* TaskManager.Domain: Entidades y contratos (interfaces, enums).
* TaskManager.Infrastructure: Acceso a datos (repositorios).

**UI**

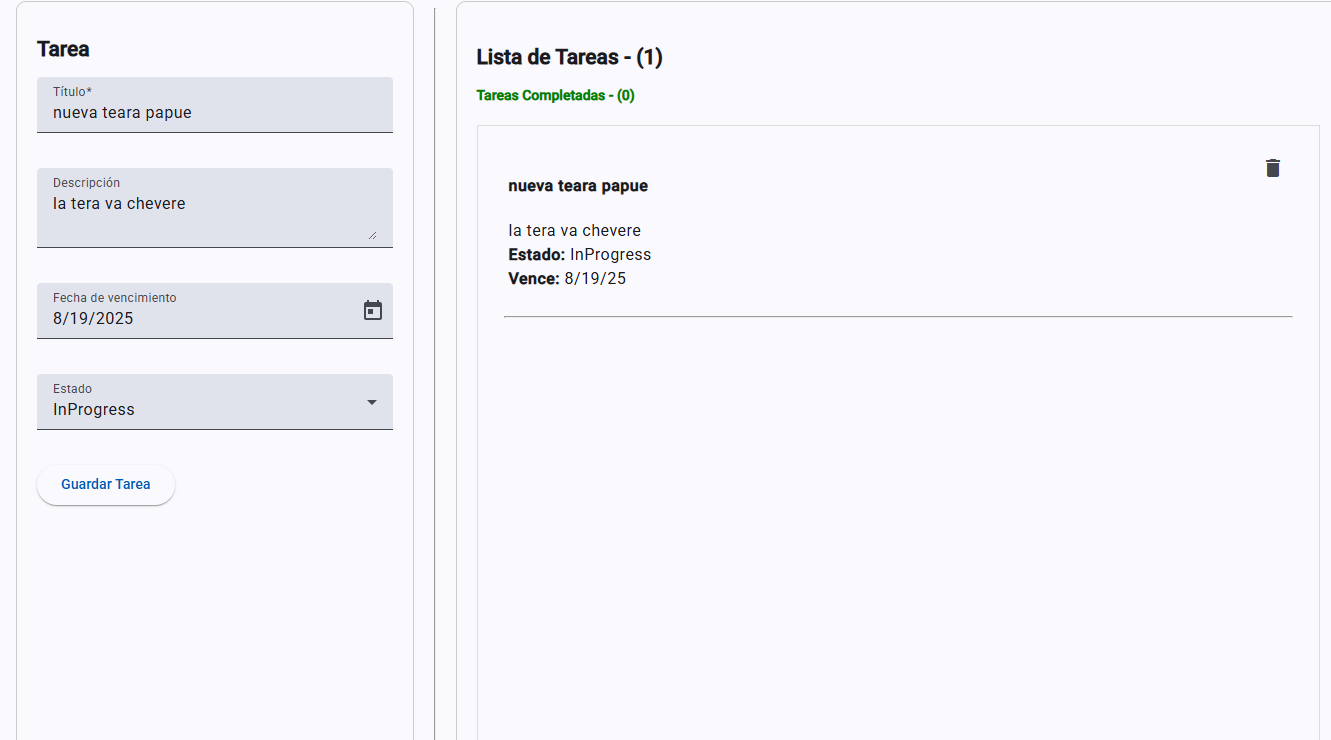
* Models: interfaces y dto.
* ServicesModels: consumo de lógica de negocio
* Task-form: formulario de carga
* Task-list: despliegue de registros almacenados

**Diagramas**

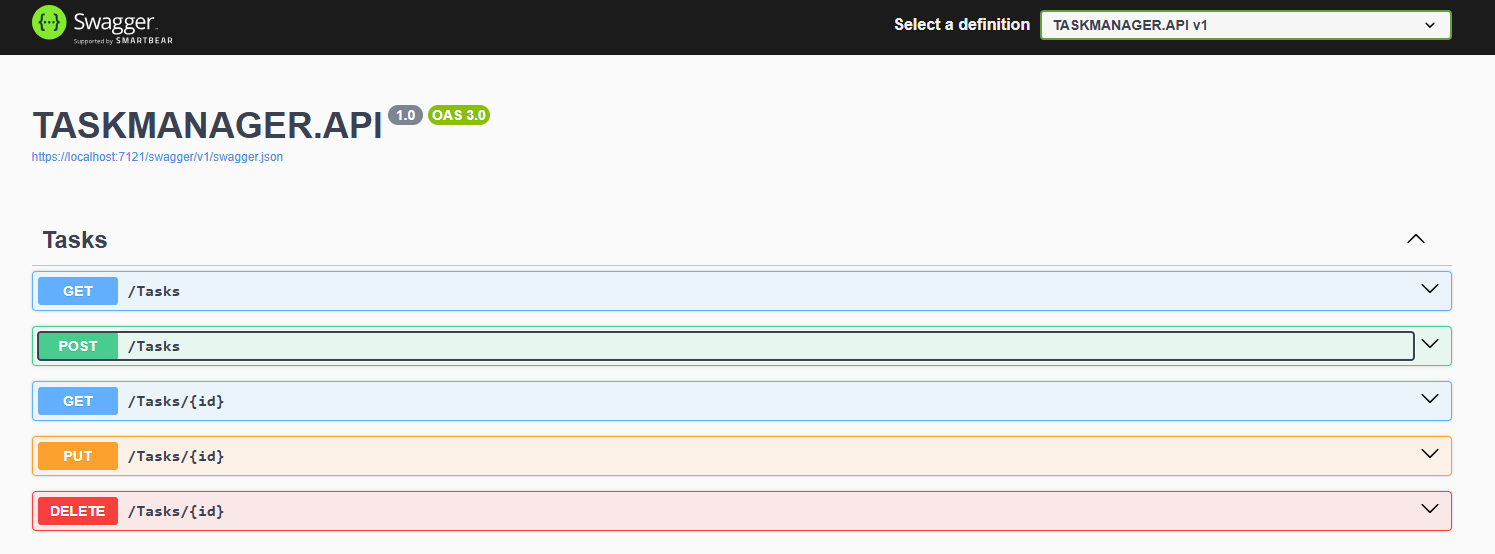


**Evidencias**

**Front End**

****

**API REST**



**Fuentes**

1. **Onion Architecture - Jeffrey Palermo**
   * https://jeffreypalermo.com/2008/07/the-onion-architecture-part-1/
2. **Clean Architecture – Robert C. Martin (Uncle Bob)**
   * https://www.oreilly.com/library/view/clean-architecture/9780134494272/
3. **Microsoft Learn - .NET Architecture Guides**
   * https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures