

Tfs.Portfolio – Arquitectura & API Reference

Documentación Técnica de Producción | API construida con .NET 10, OpenCode, SDD (OpenSpec) y Modelos Zen

.NET 10

Clean Architecture

DDD & CQRS

OpenSpec SDD

Docker / VPS

1. INTRODUCCIÓN Y ALCANCE DEL SISTEMA

Tfs.Portfolio es una API de Gestión de Portafolio construida sobre la plataforma .NET 10. Implementa una solución sólida y desacoplada para la administración centralizada de clientes, sectores, servicios, proyectos e información corporativa.

Repositorio de Código

github.com/ebugedo/tfs-portfolio

Documentación Scalar Producción

api-portfolio.timeforsoftware.com/scalar/v1

Dominio Principal: Clientes (dueños de proyectos), Sectores (categorización industrial), Servicios (ofertas asociadas), Proyectos (vínculos principales con fechas, estado y tecnologías) y Perfil de Empresa (registro único institucional).

2. PILA TECNOLÓGICA COMPLETA

Capa / Componente	Tecnología	Versión	Propósito y Detalle
Tiempo de Ejecución	.NET SDK	10.0	Entorno global de ejecución de la aplicación
Framework Web	ASP.NET Core Web API	.NET 10	Controladores RESTful, middleware y pipeline de solicitudes
Contenedor DI	Autofac	Última	Inyección de dependencias avanzada y módulos
ORM & Persistencia	Entity Framework Core	9.0	Mapeo objeto-relacional y migraciones Code-First
Base de Datos	PostgreSQL / Npgsql	16.0	Motor relacional de producción de alto rendimiento
Validación	FluentValidation	Última	Validación desacoplada de la capa de aplicación
Registro (Logging)	Serilog	Última	Logs estructurados a consola, archivos y servidor Seq
Documentación API	Scalar / OpenAPI	Última	UI interactiva de especificación OpenAPI en /scalar/v1
Serialización	Newtonsoft.Json	Última	Formateo estandarizado de JSON en camelCase
Pruebas Unitarias	xUnit	Última	Ejecución de pruebas automatizadas y aserciones
Pruebas Integración	Testcontainers	Última	Entorno de prueba aislado con PostgreSQL real
Contenerización	Docker / Docker Compose	Última	Empaquetado, aislamiento de infraestructura y ejecución
Proxy Inverso	Nginx	Última	Terminación TLS, balanceo y enrutamiento seguro
CI / CD	GitHub Actions	—	Automatización de compilación, test y despliegue SSH
Registro Imágenes	GHCR	—	GitHub Container Registry para imágenes Docker

3. METODOLOGÍA Y HERRAMIENTAS DE DESARROLLO

El proyecto fue planificado y ejecutado mediante la combinación de **OpenCode** con la metodología **OpenSpec (SDD)** y los modelos IA Zen gratuitos. Esta arquitectura metodológica garantiza trazabilidad completa desde la propuesta hasta el código final.

Flujo de Trabajo OpenSpec

- **proposal.md**: Justificación y alcance del cambio.
- **specs/<capacidad>/spec.md**: Especificaciones delta.
- **design.md**: Estrategia y diseño técnico.
- **tasks.md**: Checklist de implementación.

Principios Arquitectónicos

- **Clean Architecture**: Separación estricta de capas (Presentación, Aplicación, Infraestructura, Dominio).
- **DDD**: Dominio centralizado en Agregados, Entidades y Eventos de Dominio.
- **Diagramación C4**: Diagramas PlantUML para Contexto, Contenedores, Componentes y Código.

4. CATÁLOGO COMPLETO DE ENDPOINTS DE LA API (1/2)

Todos los endpoints están bajo el prefijo `/api/v1/`. Las escrituras exigen la cabecera `X-API-Key`.

Método	Endpoint Path	Descripción de la Operación	Autenticación
GET	<code>/api/v1/clients`</code>	Listar todos los clientes registrados	Público
GET	<code>/api/v1/clients/{id}`</code>	Obtener detalle de cliente por ID identificador	Público
POST	<code>/api/v1/clients`</code>	Crear un nuevo cliente en el sistema	API Key
PUT	<code>/api/v1/clients/{id}`</code>	Actualizar todos los datos de un cliente existente	API Key
DELETE	<code>/api/v1/clients/{id}`</code>	Eliminar un cliente del registro de base de datos	API Key
GET	<code>/api/v1/sectors`</code>	Listar todos los sectores industriales	Público
GET	<code>/api/v1/sectors/{id}`</code>	Obtener información detallada de un sector	Público
POST	<code>/api/v1/sectors`</code>	Dar de alta un nuevo sector comercial	API Key
PUT	<code>/api/v1/sectors/{id}`</code>	Actualizar denominación y atributos del sector	API Key
DELETE	<code>/api/v1/sectors/{id}`</code>	Remover un sector del catálogo	API Key
GET	<code>/api/v1/services`</code>	Listar catálogo completo de servicios	Público
GET	<code>/api/v1/services/{id}`</code>	Obtener servicio por su identificador único	Público
GET	<code>/api/v1/services/category/{category}`</code>	Filtrar servicios ofertados según categoría	Público
POST	<code>/api/v1/services`</code>	Registrar una nueva oferta de servicio	API Key
PUT	<code>/api/v1/services/{id}`</code>	Modificar la definición técnica de un servicio	API Key
DELETE	<code>/api/v1/services/{id}`</code>	Deshabilitar/Eliminar servicio de la oferta	API Key
GET	<code>/api/v1/company-profile`</code>	Obtener perfil institucional de la empresa (Singleton)	Público
PUT	<code>/api/v1/company-profile`</code>	Actualizar información corporativa central	API Key

4. CATÁLOGO COMPLETO DE ENDPOINTS (2/2) – PROYECTOS Y SALUD

Método	Endpoint Path	Descripción de la Operación	Autenticación
GET	`/api/v1/projects`	Listar proyectos (Soporta filtros por cliente, sector o estado)	Público
POST	`/api/v1/projects`	Crear un proyecto asociando cliente, sector y metadatos	API Key
PUT	`/api/v1/projects/{id}`	Actualizar la información general del proyecto	API Key
PATCH	`/api/v1/projects/{id}/status`	Cambiar el estado operativo del proyecto (ej: En Progreso)	API Key
POST	`/api/v1/projects/{id}/services/{serviceId}`	Asociar un servicio específico a un proyecto activo	API Key
DELETE	`/api/v1/projects/{id}/services/{serviceId}`	Remover la asociación de un servicio del proyecto	API Key

MANEJO DE ERRORES – RFC 7807 PROBLEMDetails

```
{
  "type": "https://api.tfs.portfolio/problems/validation-error",
  "title": "Uno o más errores de validación ocurrieron",
  "status": 400,
  "detail": "El campo Nombre es requerido.",
  "instance": "/api/v1/clients"
}
```

Comprobaciones de Salud (Health Checks)

Endpoint	Tipo & Propósito
`GET /health/live`	Prueba de vitalidad del proceso .NET
`GET /health/ready`	Verificación de conectividad a PostgreSQL

Códigos de Estado HTTP

Código	Significado
`400`	Bad Request – Validación fallida
`401`	Unauthorized – API Key inválida/falta
`404`	Not Found – Recurso inexistente
`500`	Internal Server Error – Fallo interno

5. SECRETOS REQUERIDOS EN GITHUB ACTIONS (CI/CD)

Variables y credenciales necesarias para la automatización del despliegue en el VPS de producción:

Nombre del Secreto	Descripción y Formato Esperado
`VPS_HOST`	Dirección IP pública o nombre de dominio del servidor VPS de producción.
`VPS_USERNAME`	Usuario del sistema operativo Linux autorizado para conexión SSH.
`VPS_SSH_KEY`	Clave privada SSH RSA/OpenSSH para autenticación sin contraseña.
`VPS_PORT`	Puerto de acceso del servicio SSH (por defecto: `22`).
`GHC_R_PAT`	Personal Access Token con permisos `read:packages`, `write:packages` y `delete:packages`.
`POSTGRES_PASSWORD`	Contraseña del superusuario de la base de datos PostgreSQL.
`POSTGRES_DB`	Nombre de la base de datos del entorno de producción (`portfolio_db`).
`POSTGRES_USER`	Nombre de usuario administrador de PostgreSQL.
`API_PORT`	Puerto del host del VPS mapeado hacia el contenedor (`8080`).

6. VIRTUDES DEL DESARROLLO – ANÁLISIS SISTEMÁTICO

Grupo 1: OpenCode & OpenSpec

- **Planificación Estructurada:** Flujo proposal → specs → design → tasks.
- **Modelos Zen Gratuitos:** Asistencia IA de alta calidad a costo cero.
- **Flujo Guiado:** Proponer → aplicar → archivar para evitar desvíos.

Grupo 2: Metodología SDD/DDD

- **SDD Executable:** Requisitos verificables mediante escenarios.
- **Clean Architecture:** Desacoplamiento total entre Dominio e Infraestructura.
- **Lenguaje Ubicuidad:** Modelo de dominio alineado con el negocio.

Grupo 3: Pila Tecnológica

- **.NET 10 & EF Core 9:** Máximo rendimiento y LINQ tipado.
- **Scalar / OpenAPI:** Documentación viva e interactiva.
- **Testcontainers:** Pruebas de integración reales con PostgreSQL Docker.

7. PROBLEMAS, DESAFÍOS Y ANÁLISIS DE DESVÍOS

Grupo 1: OpenCode & Zen

- **Disciplina OpenSpec:** Requiere rigor para evitar la desincronización entre especificación y código.
- **Límites IA Zen:** Ventanas de contexto acotadas y cuotas de peticiones.

Grupo 2: SDD, Clean Arch & DDD

- **Posible Sobreingeniería:** Para proyectos pequeños, 4 capas agregan complejidad inicial y mayor código repetitivo (boilerplate).
- **Curva de Aprendizaje:** Exige dominio de agregados, CQRS y eventos de dominio.

Grupo 3: Tecnologías

- **Verbosidad Autofac:** Configuración extensa de contenedores DI.
- **Infraestructura VPS:** Mantenimiento manual de Docker, Nginx TLS y volúmenes de PostgreSQL.

8. CONCLUSIONES Y PASOS SUGERIDOS

Conclusiones del Proyecto

La adopción de Clean Architecture y DDD está plenamente justificada al garantizar la mantenibilidad y probabilidad a largo plazo. Las herramientas OpenSpec y los modelos Zen demostraron que es posible construir y desplegar software profesional a costo cero.

Recomendaciones e Sigüientes Pasos

- Integrar la validación automática de archivos OpenSpec dentro del pipeline de CI/CD.
- Consolidar Registros de Decisiones de Arquitectura (ADRs) en la carpeta `docs/adr/`.
- Configurar alertas automatizadas en Seq para el monitoreo en tiempo real.

9. APÉNDICE: GLOSARIO DE TÉRMINOS TÉCNICOS

Definiciones normativas utilizadas a lo largo de la arquitectura y especificación del sistema Tfs.Portfolio:

Término	Definición Técnica
API Key	Token de autenticación enviado en la cabecera HTTP `X-API-Key` para autorizar operaciones mutantes (POST, PUT, DELETE, PATCH).
CQRS	Command Query Responsibility Segregation — Patrón de diseño que separa las operaciones de lectura (consultas) de las de escritura (comandos).
DDD	Domain-Driven Design — Enfoque de desarrollo cuyo diseño de software se modela directamente según el dominio de negocio.
GHCR	GitHub Container Registry — Registro público/privado de GitHub para alojar y distribuir imágenes de Docker.
OpenCode	Entorno / Asistente de programación asistido por Inteligencia Artificial para acelerar el desarrollo.
OpenSpec	Especificación abierta de trabajo estructurado en fases consecutivas: propuesta, especificación, diseño y tareas.
ProblemDetails	Estándar especificatorio RFC 7807 para representar respuestas de error legibles en APIs HTTP.
SDD	Spec-Driven Development — Metodología donde las especificaciones ejecutables definen los requisitos antes del código.
Scalar	Herramienta web interactiva y moderna de renderizado de documentación OpenAPI / Swagger.
Testcontainers	Biblioteca de pruebas que permite instanciar bases de datos y servicios reales en contenedores Docker durante la ejecución de tests.
VPS	Virtual Private Server — Servidor virtual alojado en la nube donde se despliega la infraestructura productiva.
Modelos Zen	Modelos de Inteligencia Artificial gratuitos proporcionados por el ecosistema OpenCode para asistencia técnica.

Nota Final de Referencia Documental

Esta documentación fue compilada y estructurada a partir del archivo maestro docs/Tfs.Portfolio-Documentation.es.md. Refleja fielmente el estado del arte de la arquitectura desplegada en producción.