**DOCUMENTO TÉCNICO – SISTEMA DE GESTIÓN DOCUMENTAL Y ANALÍTICA**

**Proyecto:** Tostao.DocumentManagement  
**Frontend:** Angular 17  
**Backend:** ASP.NET Core 8 – C#  
**Base de Datos:** SQL Server (GetionDocumentalTostaoBD)  
**Autor:** John Fredy Culma Leal  
**Fecha:** 27 de octubre de 2025

/Tostao.DocumentManagement

│

├── /src

│ ├── /Tostao.Api → Capa de presentación (Web API .NET 8)

│ ├── /Tostao.Application → Lógica de negocio (servicios y contratos)

│ ├── /Tostao.Domain → Entidades del dominio

│ └── /Tostao.Infrastructure → Persistencia con Entity Framework Core

│ └──/Tostao.Test → Test de los datos

├── /TOSTAO\_FRONTEND (Angular)

│ └── /src/app

│ ├── /components → Componentes de UI (documentos-list, navbar)

│ ├── /models → Modelos TypeScript de la app

│ ├── /services → Servicios HTTP que consumen la API REST

│ ├── /environments → Configuración por entorno

│ └── app.module.ts

│

└── /sql

├── database\_setup.sql

└── queries\_optimizadas.sql

**2. Descripción del Sistema**

El sistema gestiona documentos digitales de la empresa Tostao, permitiendo:

* Registrar, consultar, validar y archivar documentos.
* Automatizar archivado de documentos antiguos mediante un proceso interno.
* Exponer una API RESTful consumida por el frontend Angular.

**3. Backend (.NET 8)**

**Tecnologías:**

* ASP.NET Core 8
* Entity Framework Core
* SQL Server
* Moq + MSTest para pruebas unitarias

**Principales entidades:**

* Documento: Entidad principal.
* MovimientoDocumento: Historial de acciones.

**Estados válidos:**  
Registrado, Pendiente, Validado, Archivado

**4. Patrones y Principios SOLID**

| **Principio** | **Implementación** |
| --- | --- |
| **S - Single Responsibility** | DocumentService maneja reglas de negocio; DocumentRepository maneja persistencia. |
| **O - Open/Closed** | Nuevos tipos o estados pueden agregarse sin modificar lógica existente. |
| **L - Liskov Substitution** | Interfaces IDocumentRepository y IDocumentService permiten intercambiar implementaciones. |
| **I - Interface Segregation** | Cada interfaz define solo las operaciones específicas requeridas. |
| **D - Dependency Inversion** | Inyección de dependencias usada en todos los servicios. |

**5. API REST**

**Controlador principal:** DocumentosController  
Ejemplo de endpoints:

| **Método** | **Ruta** | **Descripción** |
| --- | --- | --- |
| GET | /api/documentos | Lista documentos (con paginación opcional) |
| GET | /api/documentos/{id} | Consulta por ID |
| POST | /api/documentos | Crea un nuevo documento |
| PUT | /api/documentos/{id} | Actualiza un documento |
| DELETE | /api/documentos/{id} | Elimina un documento existente |

**Ejemplo JSON (creación):**

{

"titulo": "Acta Comité Directivo",

"autor": "Laura Gómez",

"tipo": "PDF",

"estado": "Registrado"

}

**6. Pruebas Unitarias**

Ubicación: /Tostao.Test/DocumentosControllerTests.cs

**Frameworks utilizados:**

* **MSTest**
* **Moq** para simulación de dependencias

**Resumen de pruebas implementadas:**

| **Test** | **Descripción** |
| --- | --- |
| CreateDocumento\_ReturnsCreatedResult | Verifica creación de documento correctamente. |
| GetDocumento\_ReturnsOkResult | Devuelve documento cuando existe. |
| GetDocumento\_ReturnsNotFound\_WhenDocumentoNotExists | Manejo de documento inexistente. |
| GetDocumentos\_ReturnsOkResult | Lista documentos correctamente. |
| DeleteDocumento\_ReturnsNoContent\_WhenSuccess | Elimina documento exitosamente. |
| DeleteDocumento\_ReturnsNotFound\_WhenDocumentoNotExists |  |

7. Frontend (Angular 17)

**Estructura principal:**

/TOSTAO\_FRONTEND/src/app

│

├── /components

│ ├── /documentos-list

│ │ ├── documentos-list.component.ts

│ │ ├── documentos-list.component.html

│ │ └── documentos-list.component.scss

│ └── /navbar

│ ├── navbar.component.ts

│ ├── navbar.component.html

│ └── navbar.component.scss

│

├── /models

│ └── documento.model.ts

│

├── /services

│ └── documento.service.ts

│

├── /environments

│ └── environment.ts

│

└── app.component.\*, app.module.ts

**Características implementadas:**

* Listado dinámico de documentos (documentos-list.component.ts).
* Navegación con NavbarComponent.
* Consumo de la API mediante HttpClient.
* Manejo de errores con catchError.
* Diseño modular con estilos SCSS personalizados.

**Ejemplo de consumo API (documento.service.ts):**

getDocumentos(): Observable<Documento[]> {

return this.http.get<Documento[]>(`${this.apiUrl}/documentos`);

}

**8. Base de Datos – GetionDocumentalTostaoBD**

**Tablas:**

* Documentos
* LogCambios
* MovimientosDocumento

**Constraint de validación:**

ALTER TABLE Documentos ADD CONSTRAINT CK\_Documentos\_Estado

CHECK (Estado IN ('Registrado','Pendiente','Validado','Archivado'));

**9. Optimización SQL**

**Promedio de días entre registro y validación:**

SELECT

Tipo,

AVG(DATEDIFF(DAY, FechaRegistro, FechaValidacion)) AS PromedioDias

FROM Documentos

WHERE Estado = 'Validado'

GROUP BY Tipo

ORDER BY PromedioDias;

**Índices sugeridos:**

CREATE INDEX IX\_Documentos\_Autor ON Documentos(Autor);

CREATE INDEX IX\_Documentos\_Estado\_Fecha ON Documentos(Estado, FechaRegistro);

**Documento de Estrategia de Optimización**

**Estrategia de Optimización – Sistema de Gestión Documental**

El proceso de optimización se enfocó en mejorar el rendimiento de las consultas analíticas sobre la tabla **Documentos**, especialmente aquellas que calculan el tiempo promedio entre los estados *“Registrado”* y *“Validado”*. Estas consultas implican filtrado por estado, agrupación por tipo de documento y operaciones sobre campos de fecha, por lo que la estrategia se centró en **reducir los tiempos de búsqueda, ordenamiento y agregación** mediante índices adecuados.

Se implementaron índices compuestos y cubiertos que optimizan las rutas de acceso más frecuentes:

* **IX\_Documentos\_Estado\_Fecha**: acelera el filtrado por *Estado* y la ordenación por *FechaRegistro*, útil en reportes temporales.
* **IX\_Documentos\_Estado\_Tipo**: mejora las consultas que agrupan por *Tipo* y filtran por *Estado*.
* **IX\_Documentos\_Reportes** (índice cubriente): incluye columnas adicionales como *Titulo* y *Autor* para evitar operaciones de búsqueda adicionales (*key lookups*), optimizando reportes que requieren información descriptiva.
* **IX\_Documentos\_FechaValidacion**: agiliza las comparaciones y cálculos basados en fechas de validación.

Asimismo, se recomienda mantener las **estadísticas actualizadas** mediante UPDATE STATISTICS dbo.Documentos y revisar periódicamente los **planes de ejecución** para detectar posibles fragmentaciones o ineficiencias.  
Con esta estrategia, se mejora la velocidad de respuesta de las consultas analíticas y se garantiza una mejor escalabilidad en el manejo de grandes volúmenes de documentos.

**10. Ejecución del Proyecto**

**Backend:**

cd src/Tostao.Api

dotnet run

API disponible en: http://localhost:7078.

**Frontend (Angular):**

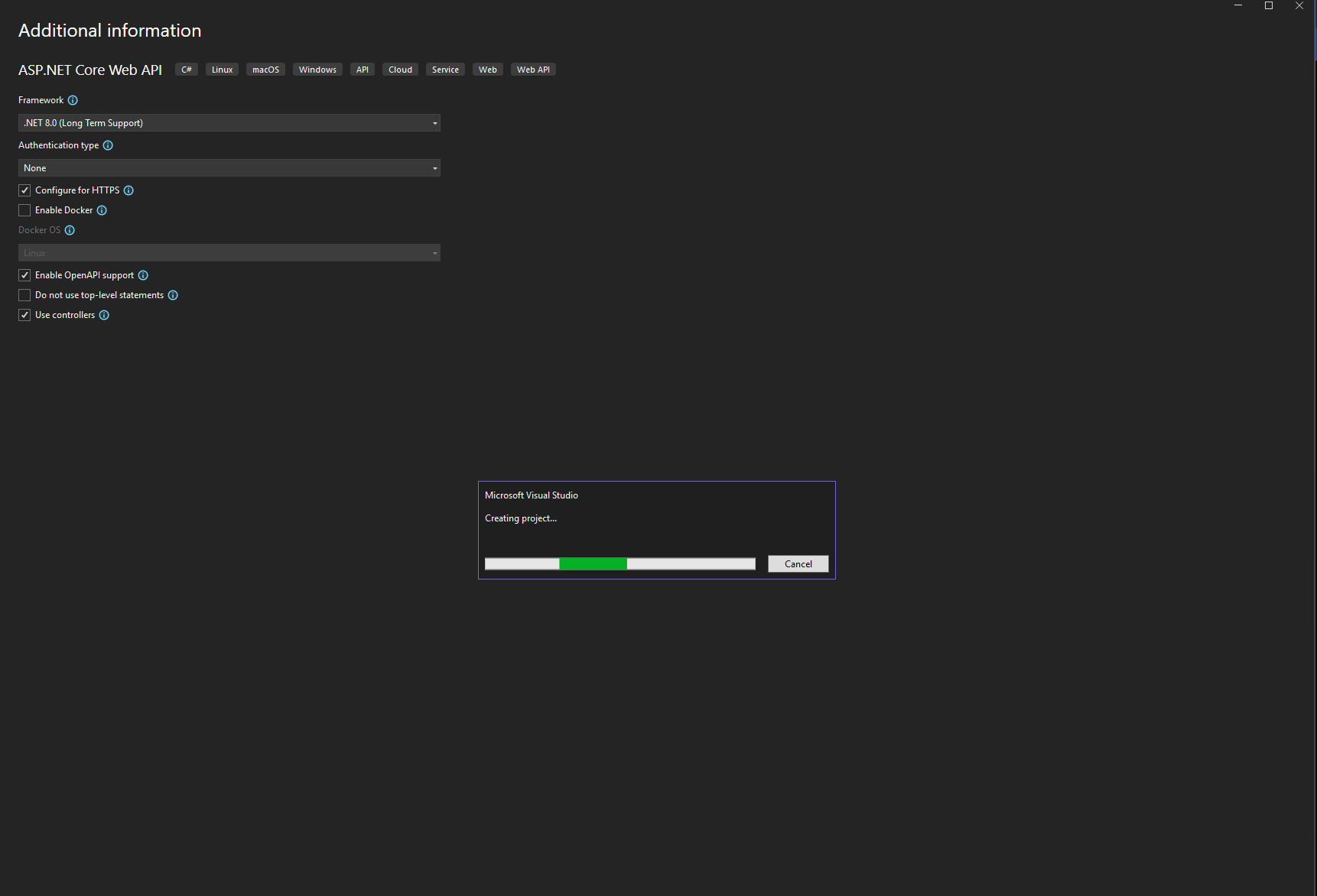
cd TOSTAO\_FRONTEND

npm install

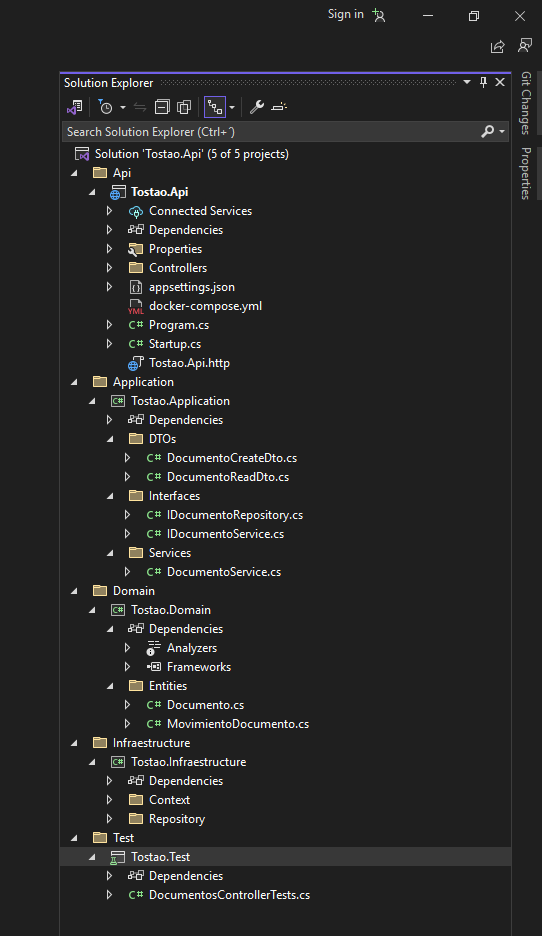
ng serve -o

disponible en: http://localhost:4200/documentos

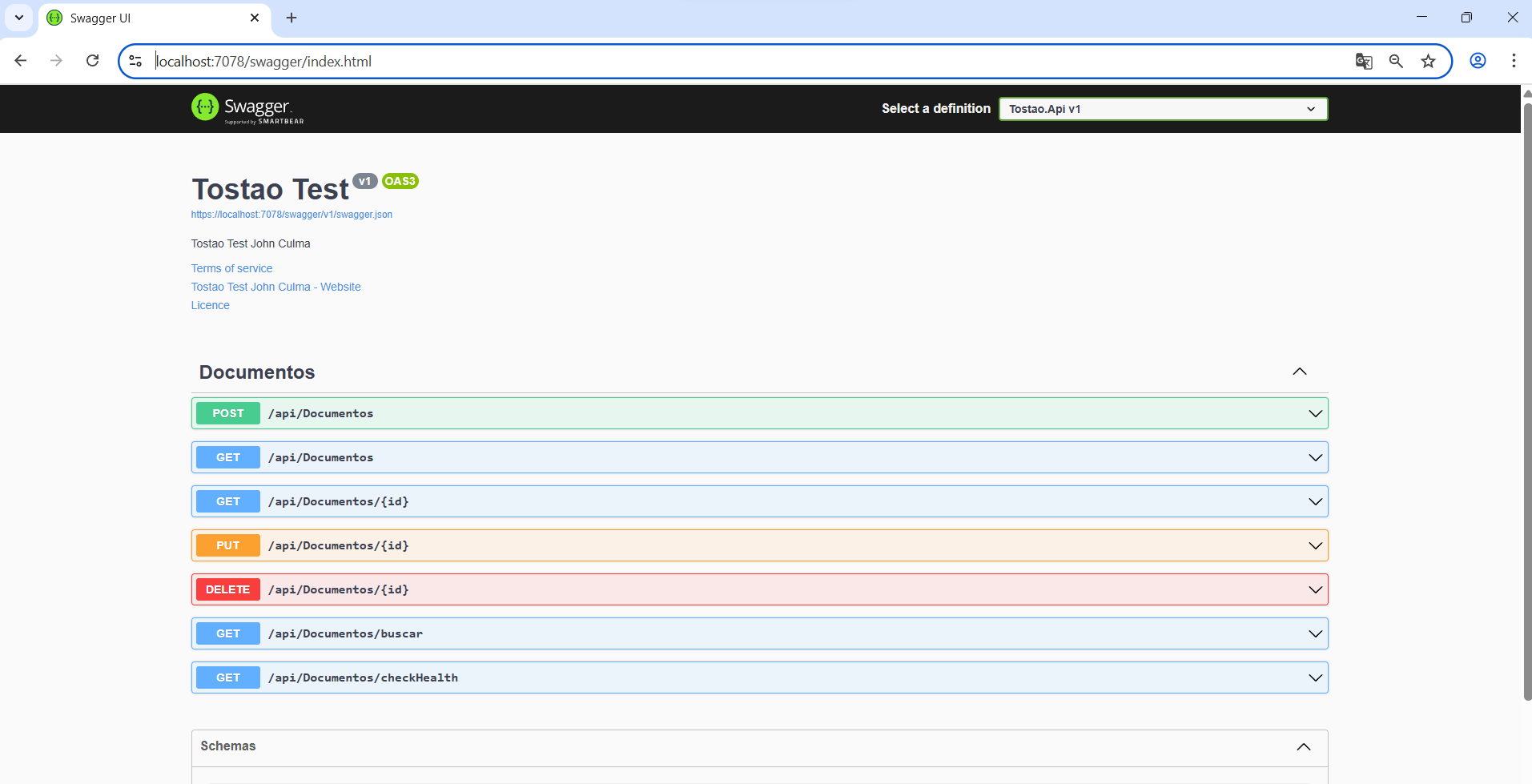
Puebas BackEnd



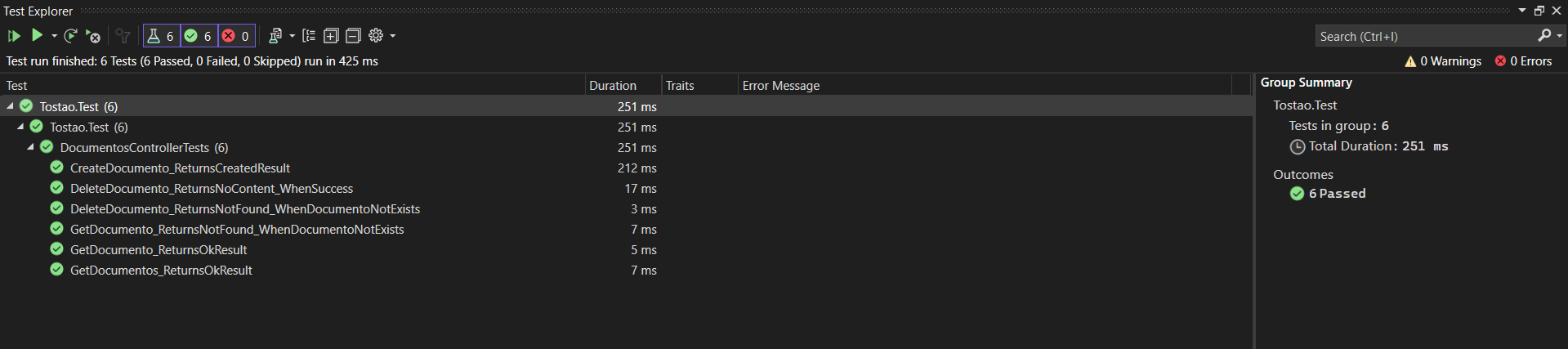
Estructura



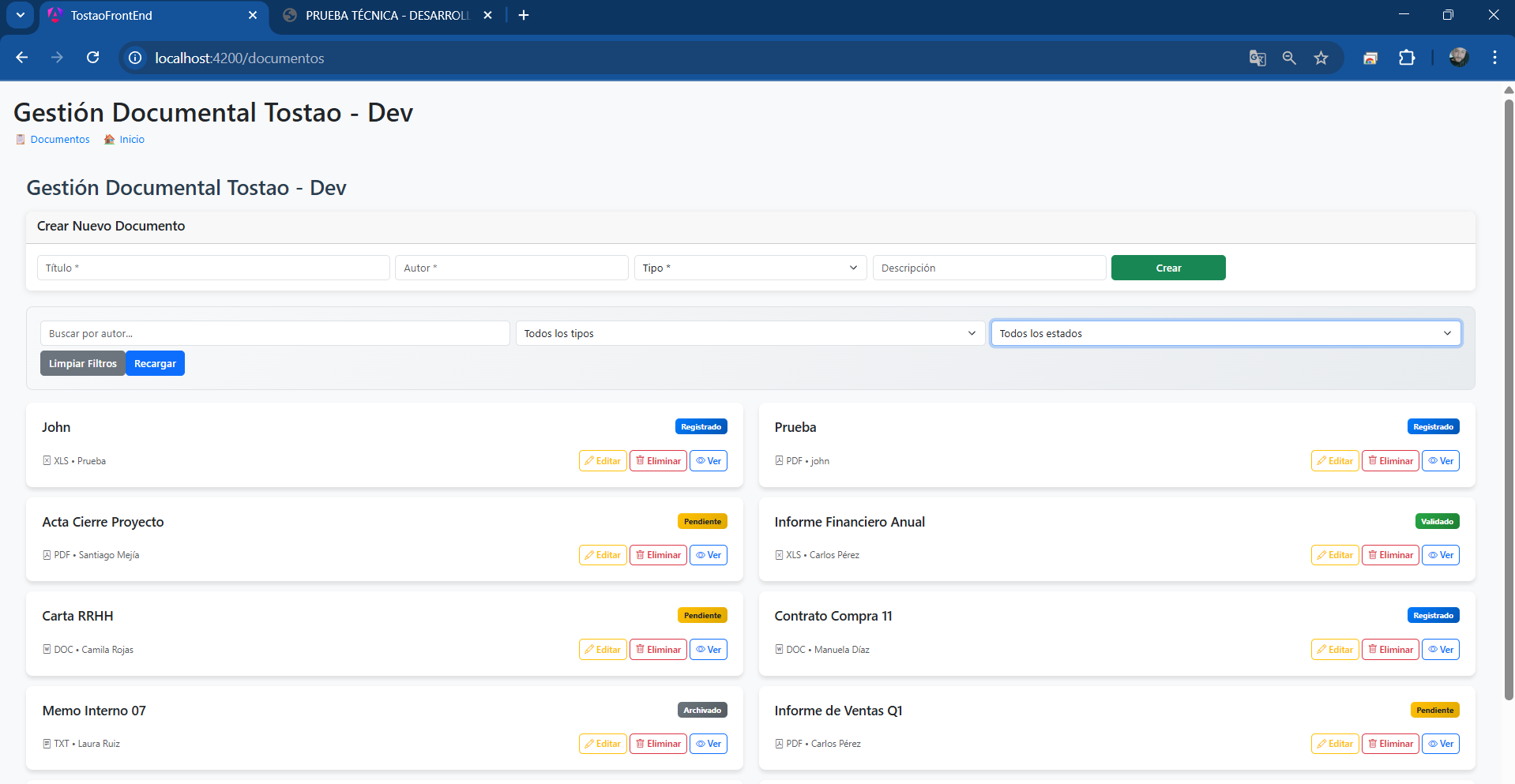
Swagger



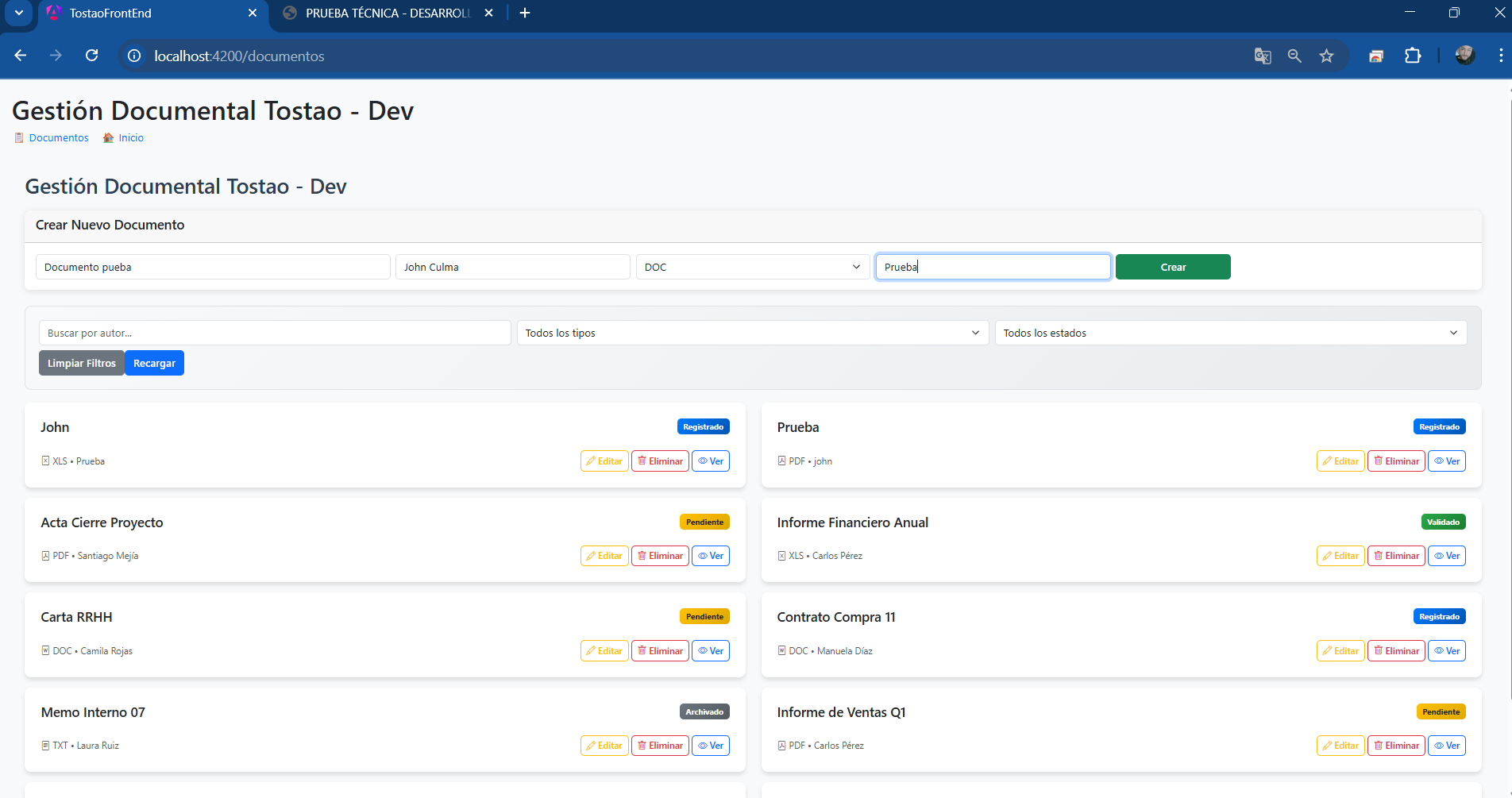
Test de la aplicación

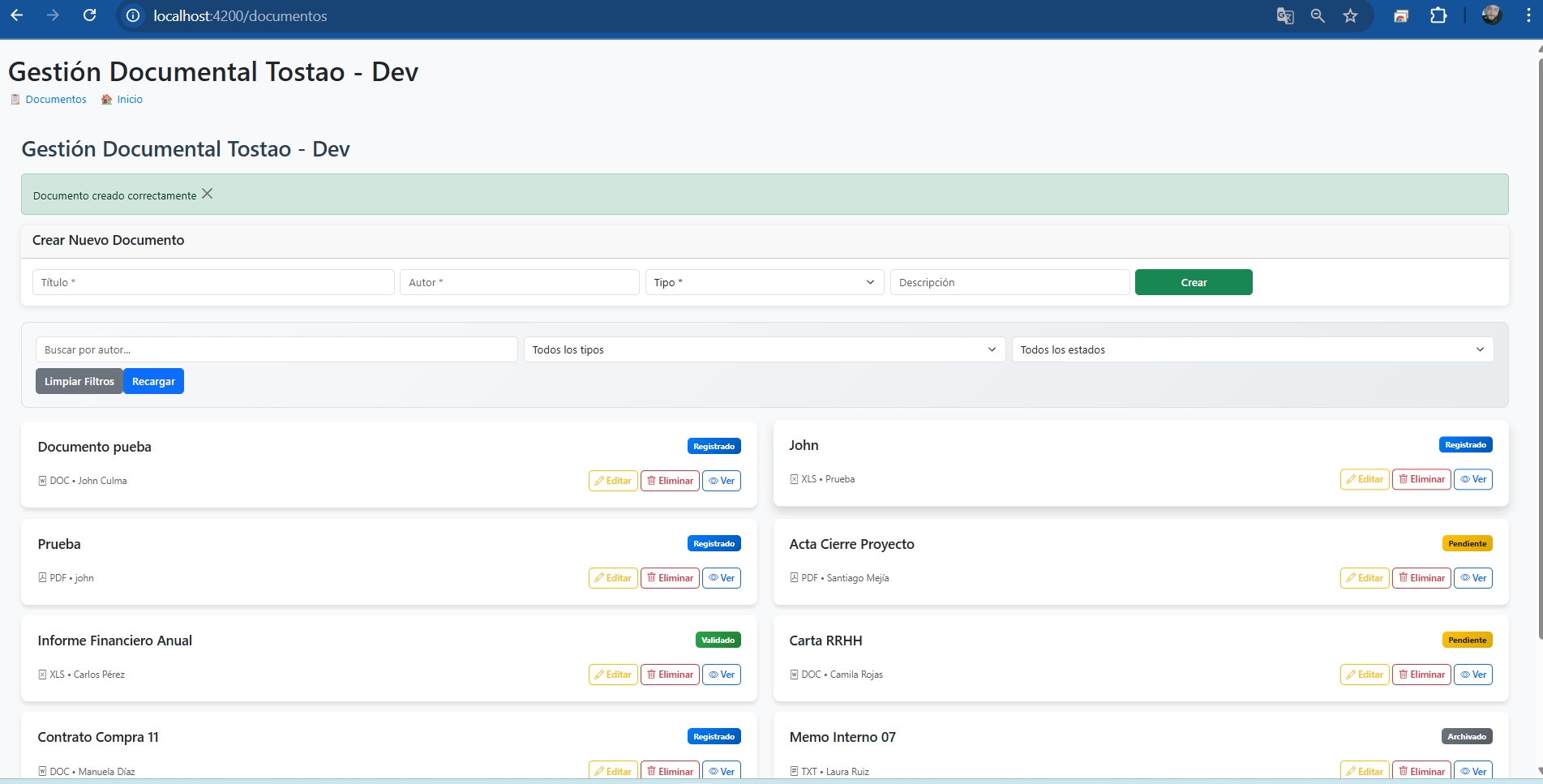


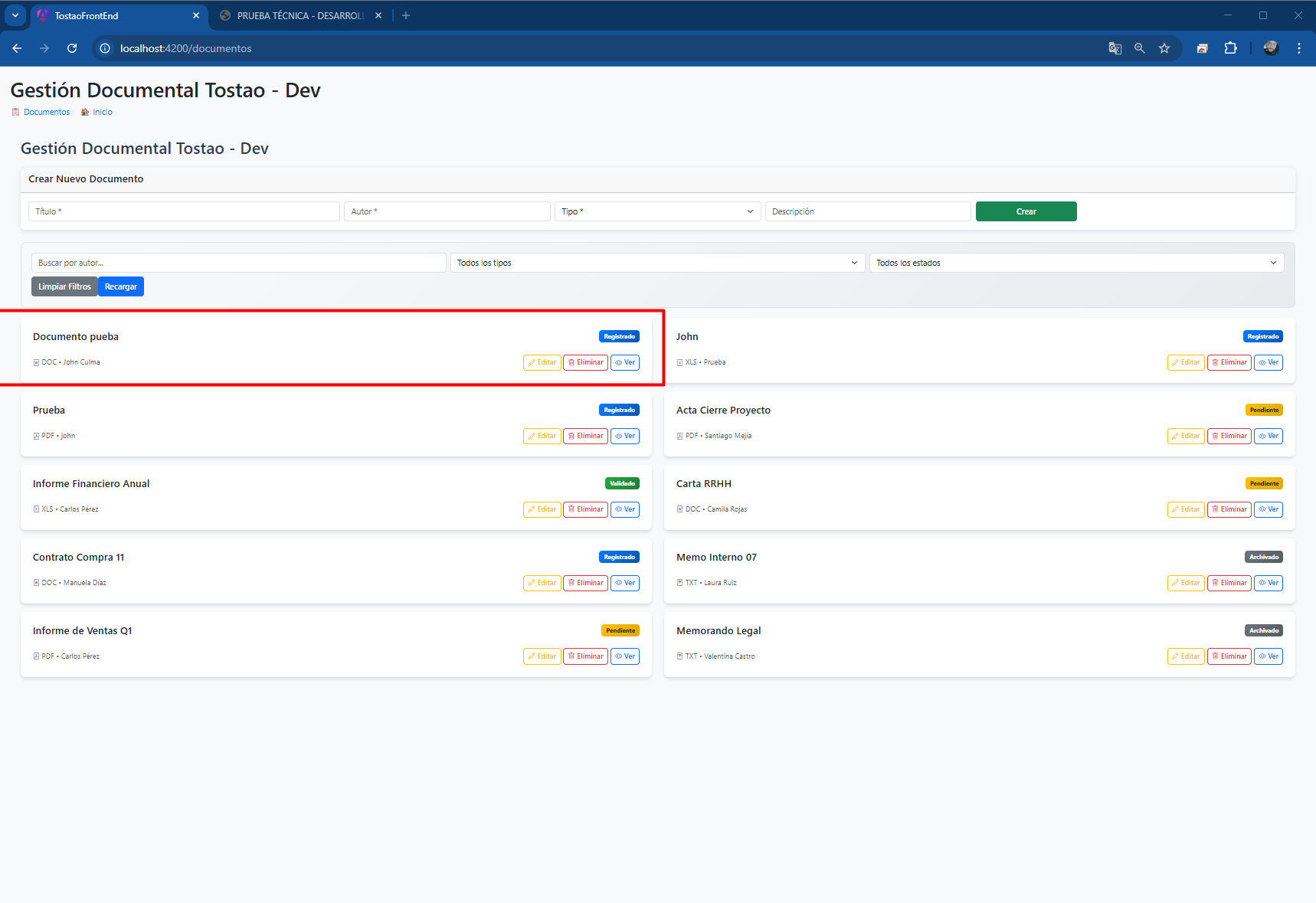
Pruebas FrontEnd



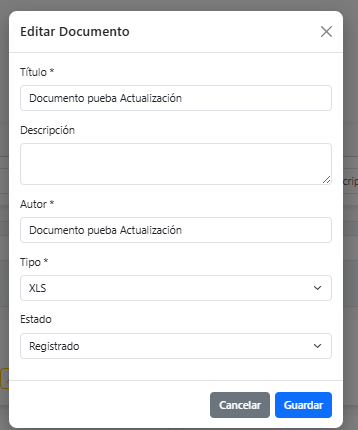
Creación

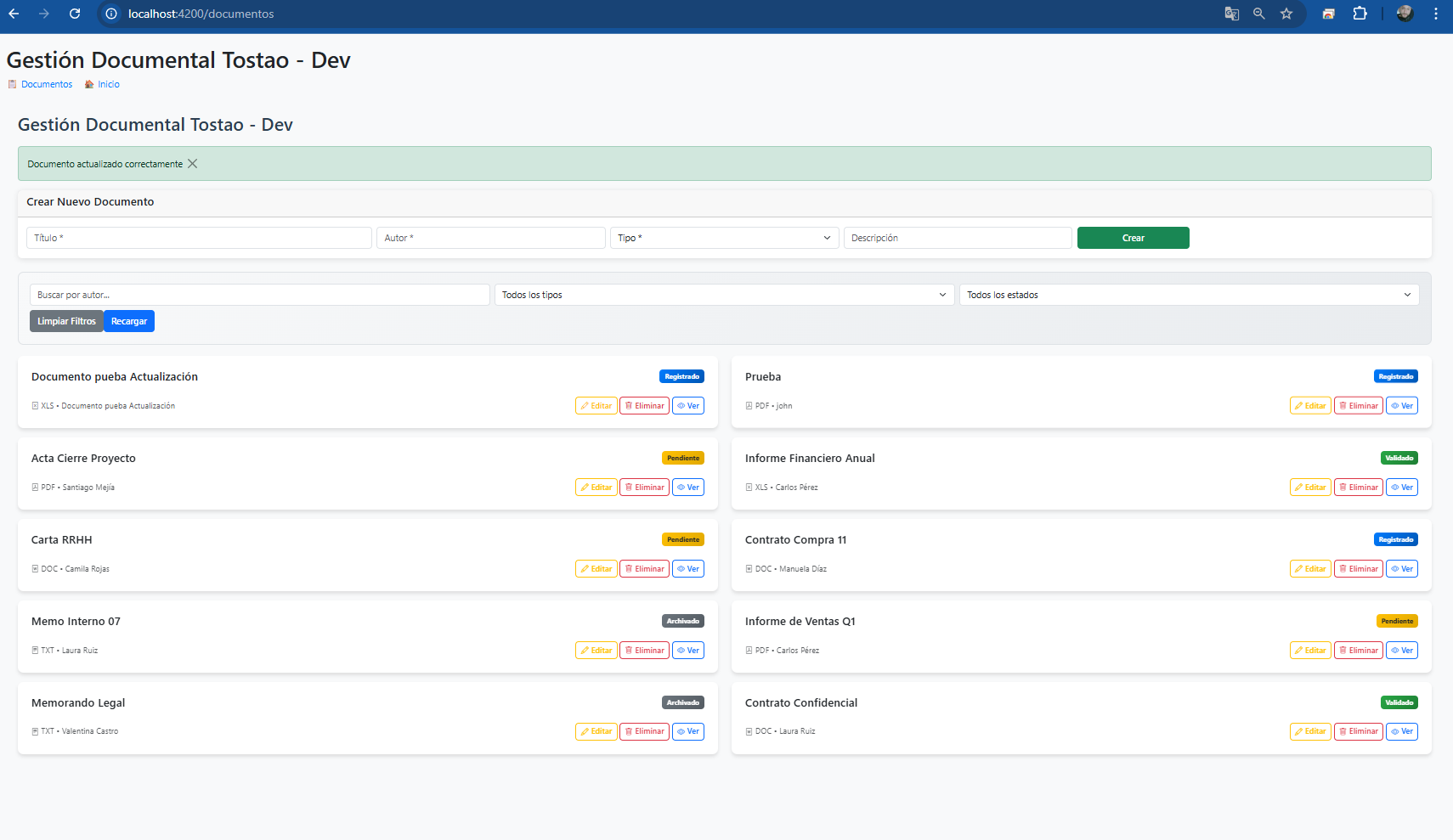




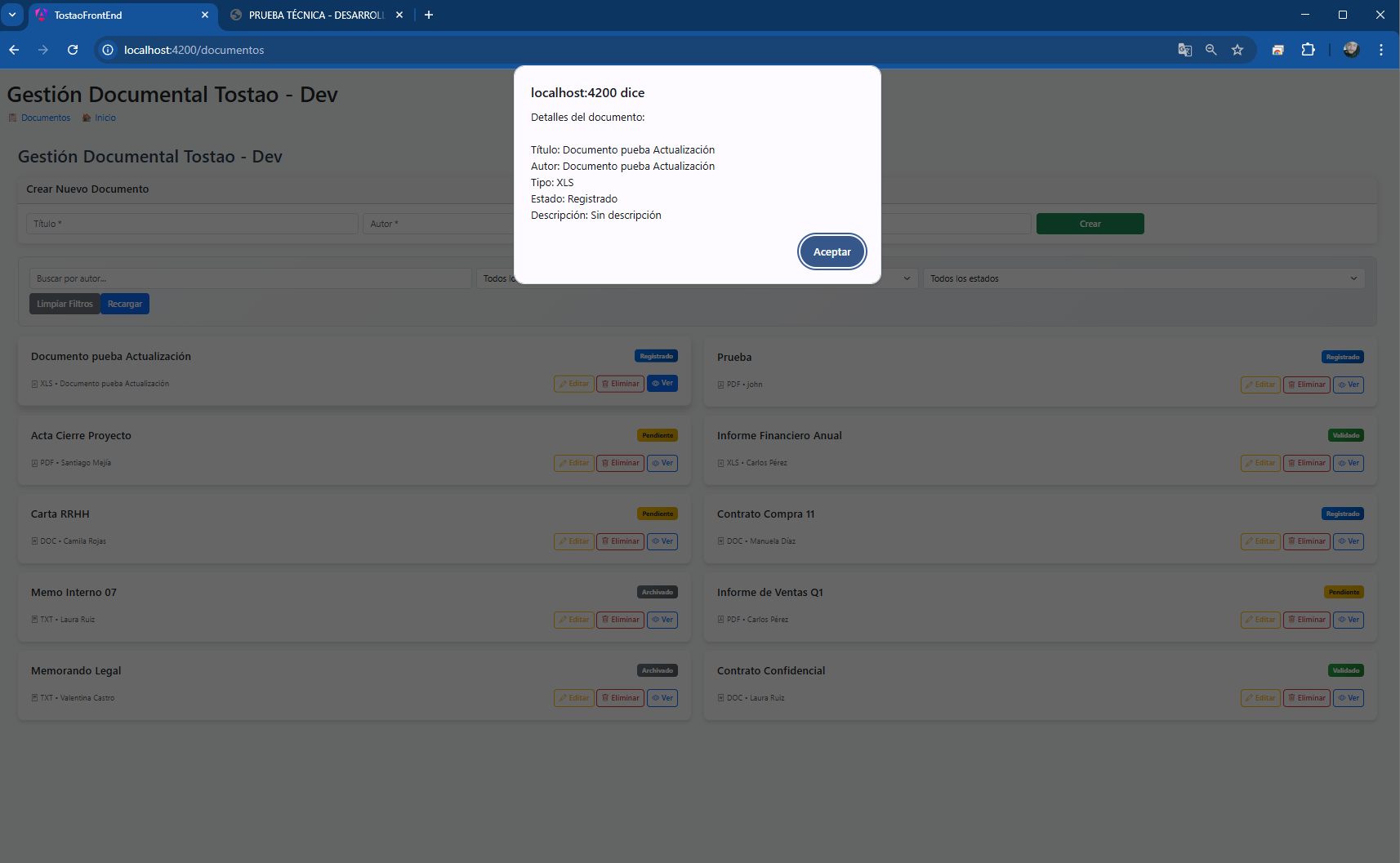


Actualización





Vista



Delete

