

Documento Técnico

Portal Administrativo Library Travel

Contenido

[1. CONTROL DE CAMBIOS 3](#_Toc119429051)

[2. proposito 3](#_Toc119429052)

[3. pre-requisitos 3](#_Toc119429053)

[4. Paquetes nuget 4](#_Toc119429054)

[5. Creacion de la base de datos 8](#_Toc119429055)

[6. modelo de datos (ER) 11](#_Toc119429056)

[7. componentes 11](#_Toc119429057)

[8. capa de presentacion - FRONTEND 12](#_Toc119429058)

[9. utilizacion swagger - backend 15](#_Toc119429059)

# CONTROL DE CAMBIOS

|  |  |  |
| --- | --- | --- |
| **VERSIÓN** | **FECHA** | **DESCRIPCIÓN** |
| 1 | 13-11-2022 | Creación del Documento |

# proposito

El propósito de este documento es detallar la puesta a punto y/o funcionamiento del aplicativo Library TRAVEL, sobre un ambiente de desarrollo.

# pre-requisitos

Para el buen funcionamiento del aplicativo se deberá instalar los siguientes productos

|  |  |
| --- | --- |
| Visual Studio 2019 Comunity Edition | C:\Users\ymercado\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\8913E444.tmp |
| Sql Server 2008 R2 | Exportar uno o más Archivos SQL Server 2008 “varbinary” | Blog ... |
| .Net Core 5 | C:\Users\Administrador\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\24C319A6.tmp |
| Angular 7 | C:\Users\Administrador\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\E86CFCA4.tmp |
| Material Design | C:\Users\Administrador\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\9AB71E52.tmp |
| Node js 14 | C:\Users\Administrador\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\DA367A43.tmp |
| Visual studio Code | How to Set Up Visual Studio Code in 2022 (The Easy Way) |

# Paquetes nuget

Algo para tener en cuenta, es que cada proyecto para su funcionamiento utiliza, algunos paquetes para realizar una(s) funcione(s) especifica(s). a continuación, se detalla por proyecto sus paquetes instalados.

|  |  |
| --- | --- |
| Capa/Proyecto | Nuget |
| LibraryTRAVELApi – API REST |  |
| Aplicacion.Test |  |
| Aplicacion.Contratos |  |
| Aplicacion.Core |  |
| Aplicacion.Implementacion |  |
| Dominio.Contratos |  |
| Dominio.Core |  |
| Datos.Persistencia.Core |  |
| Datos.Persistencia.Implementacion |  |
| Datos.Persistencia.Repositorio |  |
| Datos.Persistencia.Core |  |
| Datos.Persistencia.Implementacion |  |
| Datos.Persistencia.Repositorio |  |
| Utilitarios.IoC |  |

# Creacion de la base de datos

La creación de la base de datos se realiza por medio de Entity frameworks Code First, la librería encargada de realizar la creación de la base de datos se denomina Datos.persistencia.Core

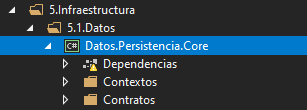


Fig.1. Proyecto datos

Y es aquí donde se deberá de ejecutar el comando de migración.

Antes de ejecutar el comando se debe validar la cadena conexión, validar en el archivo appsettings.json, de la capa de presentación.

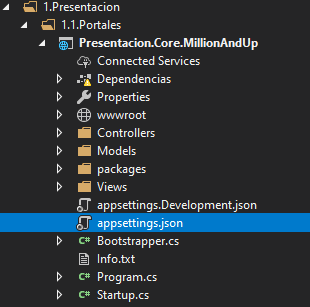


Fig.2. Proyecto presentación

"ConnectionStrings": {

"ConexionTest": "Server=LAPTOP-AKTR691F\\SQLEXPRESS;Initial Catalog=Biblioteca;Integrated Security=true;"

}

Se deberá migrar el modelo de la siguiente manera, compile y desde la consola de paquetes se deberá ejecutar el siguiente comando:

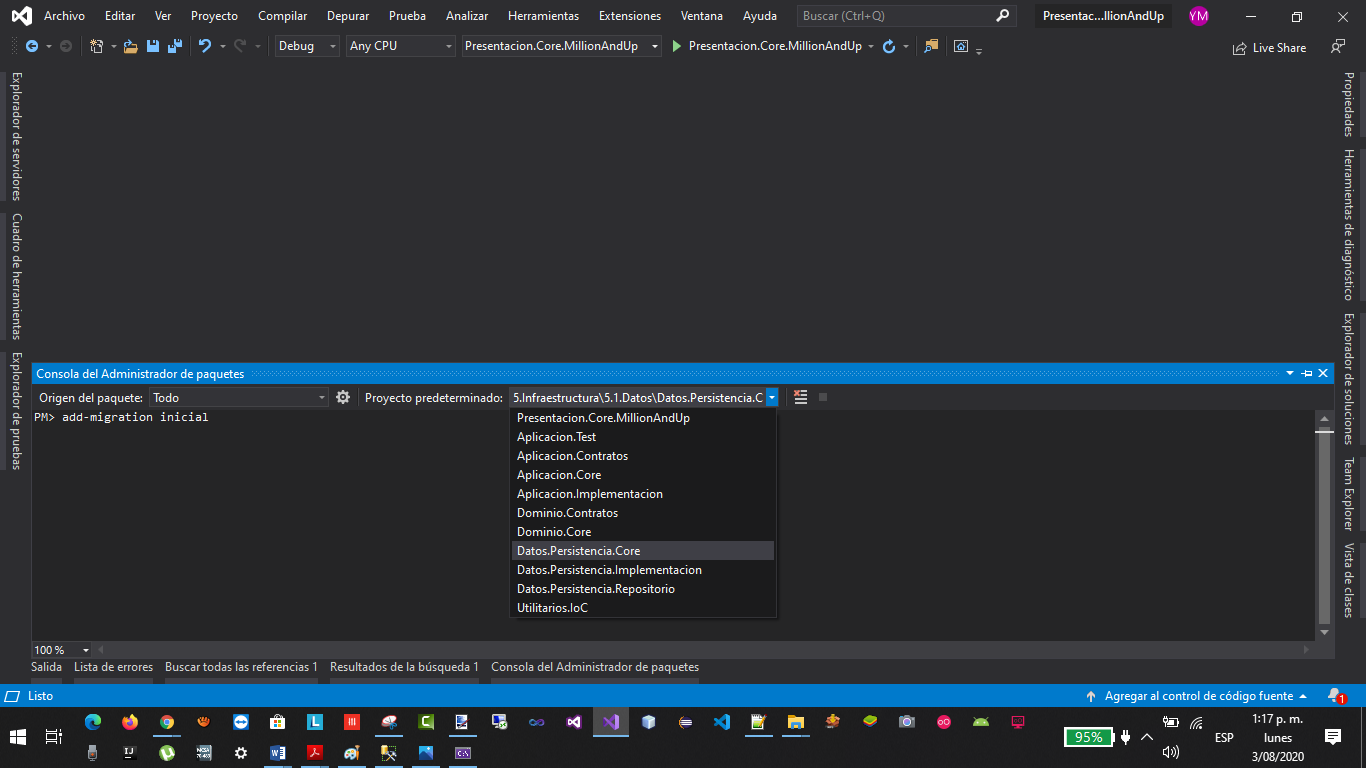


Fig.3. Consola

PM>Enable-Migrations

PM> add-migration inicial (enter)

Build started...

Build succeeded.

To undo this action, use Remove-Migration.

**Y luego**

PM> update-database (enter)

Build started...

Build succeeded.

Done.

Una vez finalizado el proceso, se abra creado la base de datos

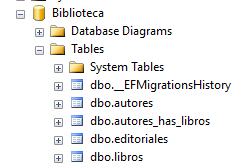


Fig.4. Base de datos Biblioteca

Y habrá creado el archivo de migración en el proyecto donde se ejecutó la migración

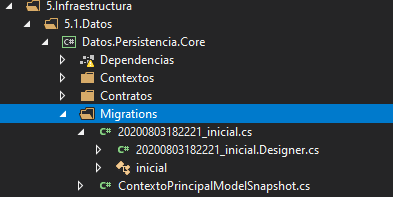


Fig.5. Carpeta de migración generada

Existe otra manera para crear la base de datos, ejecute el archivo denominando, SQL\_Creacion\_Base\_Datos\_Biblioteca.sql que se encuentra en la carpeta Documentos en el proyecto BackEnd (LibraryTRAVELApi) en el Management Studio.

Abra el archivo y ejecútelo.

# modelo de datos (ER)

Se detalla la base de datos con sus respectivas tablas, y la relación que en ella existe.

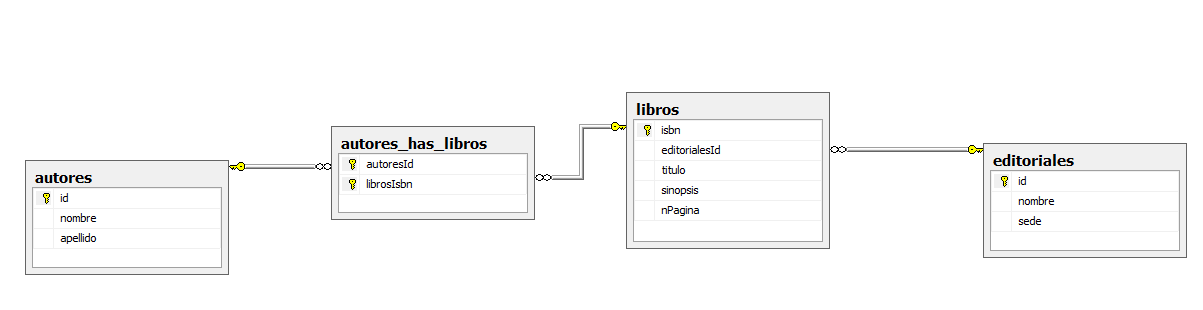


Fig.6.Diagrama ER

# componentes

El software que utilizará Biblioteca TRAVEL, está desarrollado sobre una Arquitectura orientada a dominio (DDD), utiliza inyección de dependencia, unidad de Trabajo y MVC como patrones de diseño. El proyecto API y las librerías, se denomina LibraryTRAVELApi.

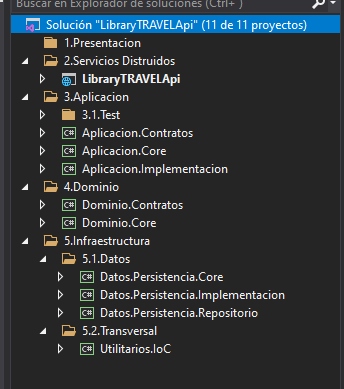


Fig.7. Arquitectura de la aplicación

Para este documento el autor no detallara cada una de las capas.

# capa de presentacion - FRONTEND

A continuación, se mostrarán las pantallas de la apliacion librería TRAVE el FRONT está diseñado con Material Design, sobre Angular. La solución es denominada LibraryTRAVELWeb.



Fig.8. Pantalla inicio

Despliega la información de los libros almacenados en la librería.

Para adicionar un libro, haga clic sobre el icono , la aplicación lo enrutará a una pantalla que permitirá adicionar un Libro.

**Nota**: para crear un libro debe existir datos en las tablas de Autores y Editoriales (para tema de la prueba no se crearon los formularios para llenar las tablas, dicho proceso se puede hacer por Swagger).



Fig.9. Pantalla para crear un libro



Fig.10. Listando la información para almacenar

Para adicionar o editar un libro haga clic sobre el icono 

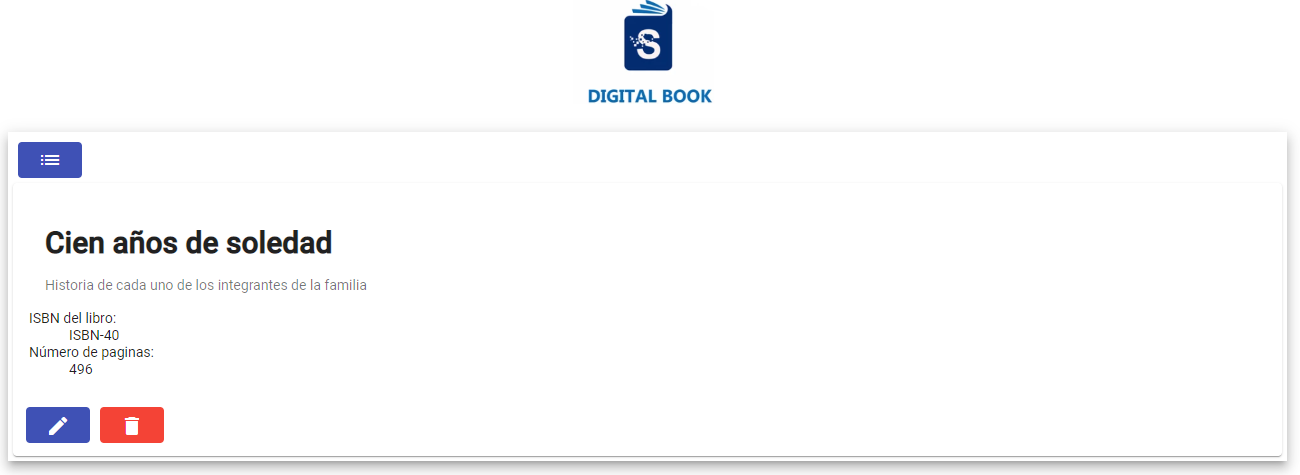


Fig.11. indica que el registro fue almacenado

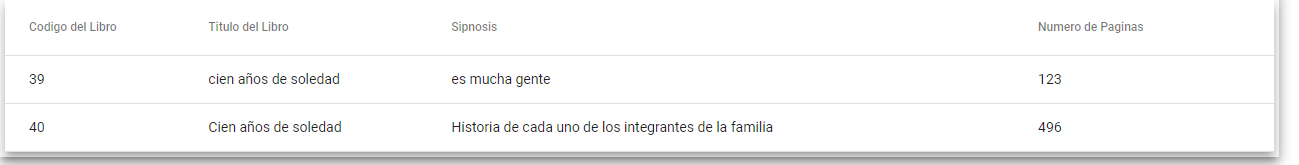
Para editar el registro se puede de dos maneras, la primera es haciendo clic en el icono  o haciendo clic sobre el registro en la tabla de consulta 

Fig. 12. Haz clic en un registro

Para regresar a la pantalla inicial haga clic sobre el icono  y para eliminar el registro haga clic sobre el icono  .

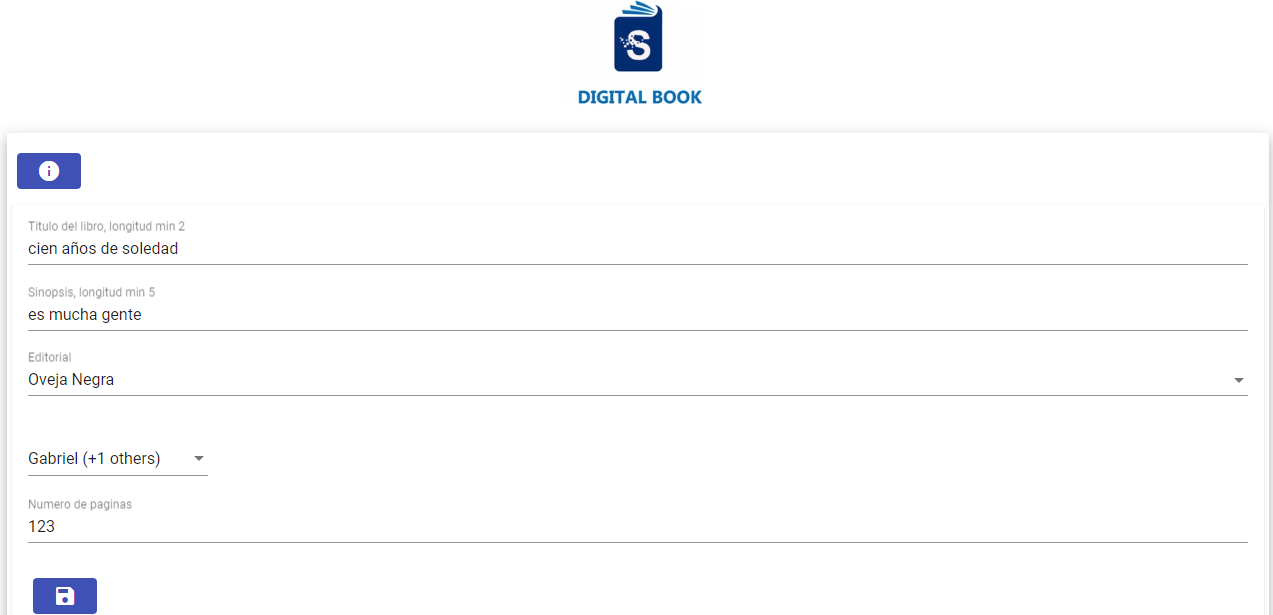


Fig.13. Pantalla edición de libro

# utilizacion swagger - backend

Para realizar la administración de cada una de las tablas que constituye la aplicación de la librería TRAVEL. Se ha creado una API, que es la encargada de comunicarse entre la Capa de presentación y la base de datos.

Por medio de unos métodos Swagger puede comunicarse con el Back, este asu vez se comunica con el core en una arquitectura DDD.

Para iniciar ejecute la aplicación LibraryTRAVELApi, una vez ejcutada cargar Swagger

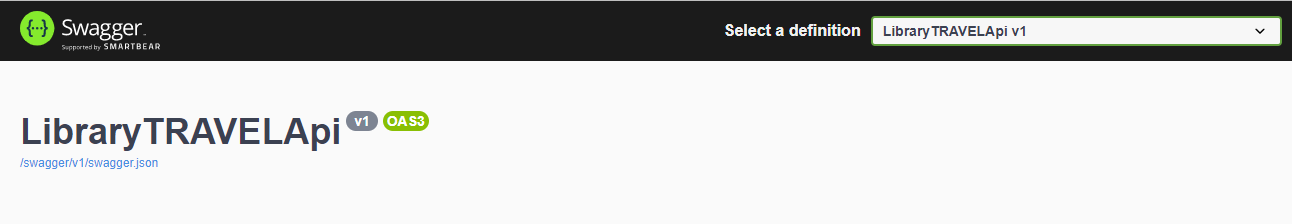


Fig.1. Swagger :: LibraryTRAVELApi v1

Esta herramienta posee 4 métodos para la administración

: Selecciona o genera Consulta sobre la tabla.

: Adiciona un registro sobre la tabla.

: Actualiza un registro sobre la tabla

: Elimina un registro sobre la tabla.

La pagina muestra cada una de las posibles tablas para administrar. Ellas son

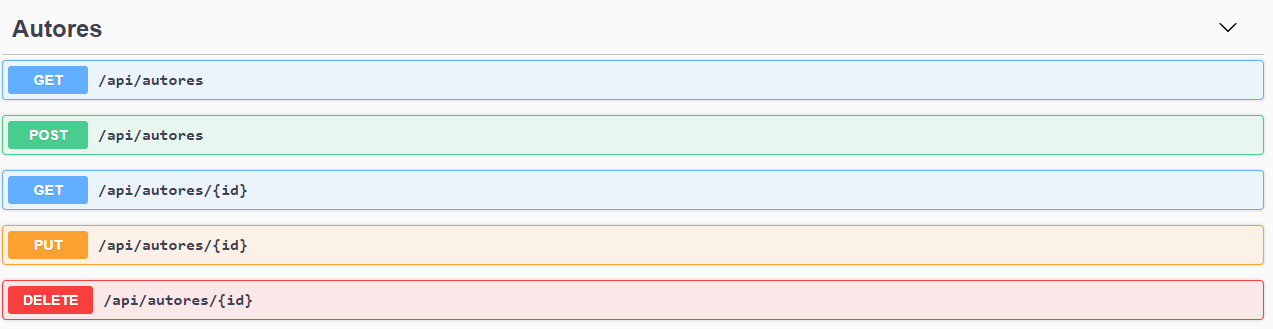


Fig.2. Administración Autores

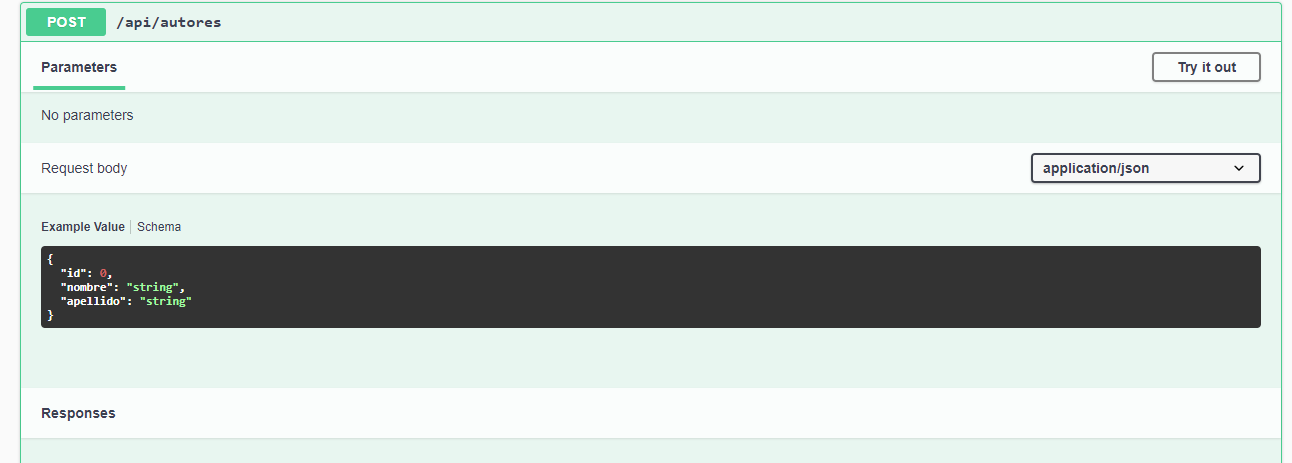


Fig.3. Administración Autores :: POST

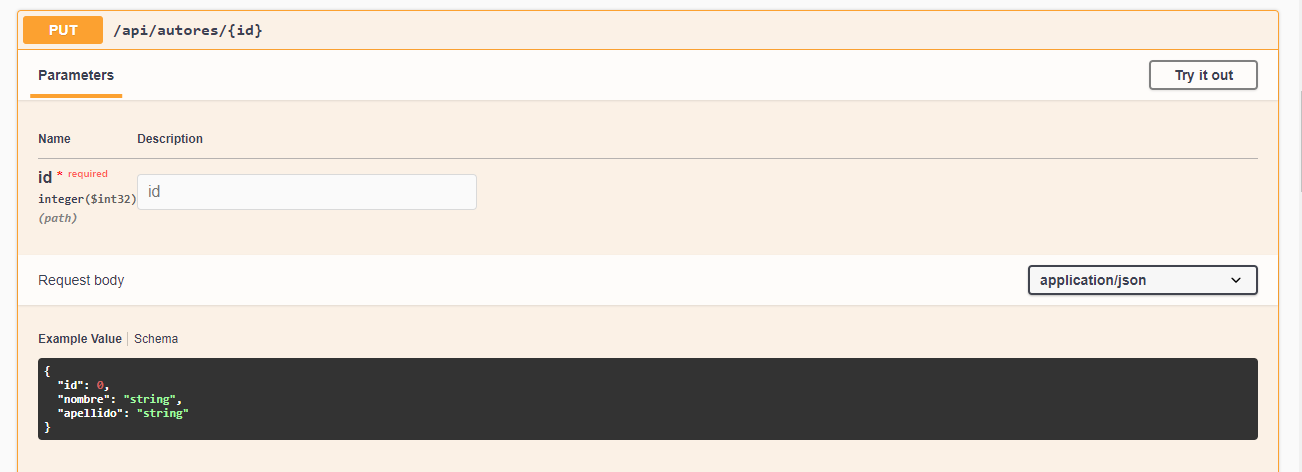


Fig.4. Administración Autores :: PUT

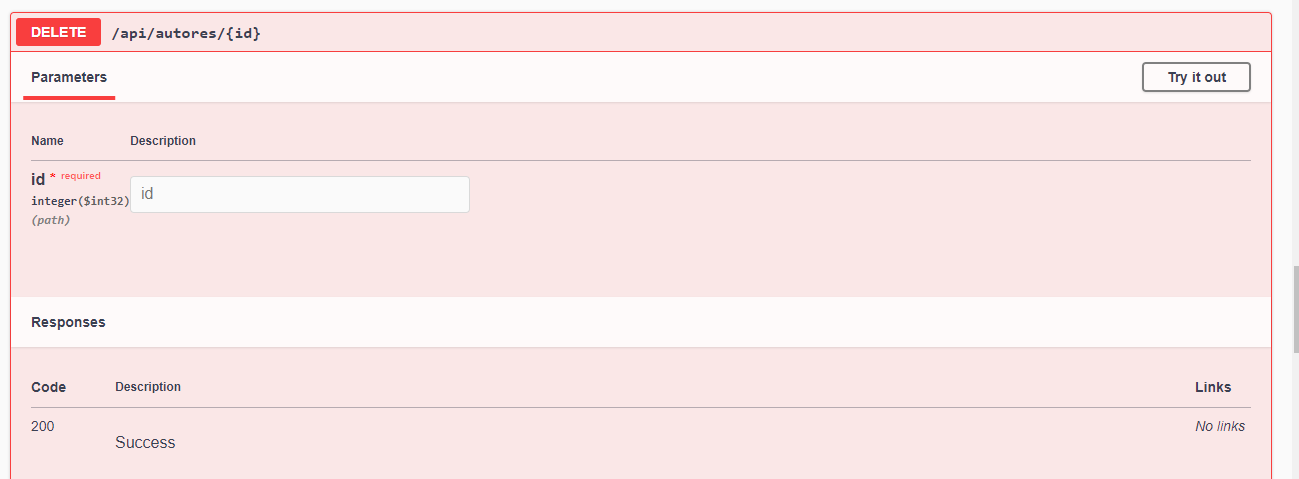


Fig.5. Administración Autores :: DELETE

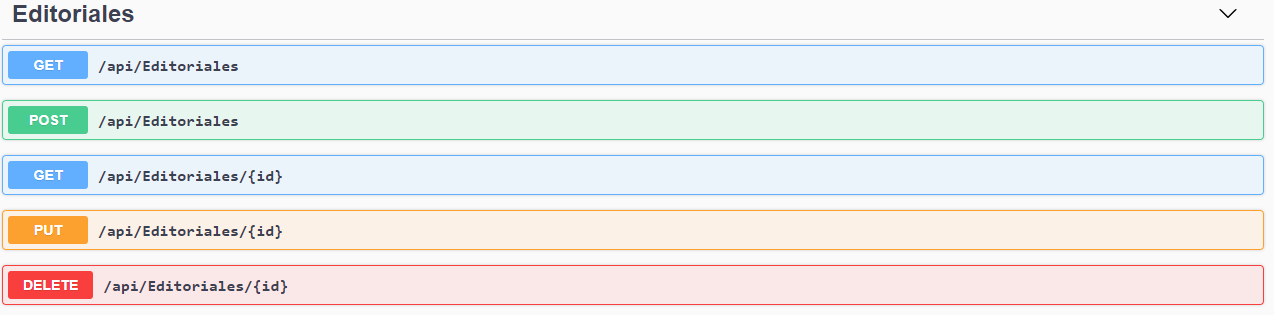


Fig.3. Administración Editoriales

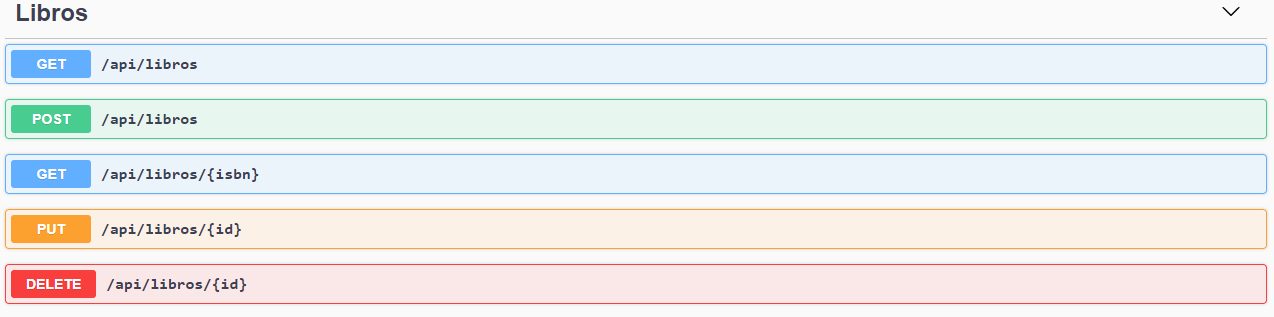


Fig.4.Administracion Libros

En el repositorio GibHub, especifica cual son los paso a seguir para poner a punto la aplicación.