|  |  |
| --- | --- |
| **Documentación Api Personas Supervielle**  [Subtítulo del documento] | Descripción breve  Este documento contiene una breve descripción sobre diferentes puntos del proyecto proporcionando una primera aproximación a la funcionalidad que expone esta API.  ALBARRACIN EDUARDO JAVIER  [Título del curso] |

Contenido

[Tipo de Proyecto – Configuración inicial 2](#_Toc117024243)

[Puesta en marcha 2](#_Toc117024244)

[Arquitectura 2](#_Toc117024245)

[Swagger 3](#_Toc117024246)

[Colección – Recursos 4](#_Toc117024247)

[Logs 5](#_Toc117024248)

[Manejo centralizado de errores - Middleware 5](#_Toc117024249)

# Tipo de Proyecto – Configuración inicial

El proyecto de api denominado “Api Personas”, está desarrollado con el framework de .NET 6, bajo la modalidad code first.

El proyecto cuenta con una serie de recursos expuestos para la gestión de personas, información sobre vínculos y parentescos. Para una mejor evolución con los futuros consumidores de la api, se realizó un versionado de la misma.

# Puesta en marcha

La api se encuentra alojada en el siguiente repositorio (público), para ser descargada/clonada: <https://github.com/ealbarracindev/api-personas-supervielle.git>

1. Clonar del repositorio y extraer la carpeta principal, en el directorio de “src” se encuentra la solución api-personas-supervielle.
2. También en el directorio principal se encuentra otra carpeta denominada “docs” contiene a la carpeta “script”, con el archivo correspondiente para el modelo de datos. Este script contiene las estructuras de datos como datos para las tablas tipo y/o auxiliares.
3. Cambiar la conexión correspondiente en el archivo de configuración Personas.Api/appsettings.dev.json - (WebApi Project)
4. Si se desea ejecutar el proyecto desde la consola de comandos

- dotnet restore

- dotnet run (OR) Run the Solution using Visual Studio 2022

1. El motor de base de datos utilizados fue SQL Server 2019.
2. En el directorio principal “docs”, se encuentra una carpeta “collection” conteniendo un archivo “api personas - supervielle.postman\_collection.json” generado desde la herramienta Postman con los endpoints listos para ser incorporado en cualquier herramienta cliente como la mencionada.

# Arquitectura

El proyecto posee una distribución de carpetas con las cuales se identifican tres partes principales, la parte de los controladores con los logs del proyecto, una parte core y la de infraestructura. Cada una de estas partes fueron distribuidas de forma tal que permitan un crecimiento del proyecto con determinado lineamiento, siendo la parte core como la de infraestructura un posible candidato a ser un proyecto de clase en si mismos.

La sección denominada “core”, tiene la finalidad de contener todas las entidades principales, los dtos, interfaces, etc. Considerando que deberían existir la menor dependencia de terceros.

La finalidad de este proyecto de infraestructura en cambio, resulta contar con dependencias de terceros, configuraciones, gestión de reglas de negocio mediante servicios .

Luego se encuentra la sección de “infraestructure”, donde reside la gestión y enlances entre la api, la sección core, la base de datos y otros posibles servicios externos. La finalidad de esta seccion es que permita no sólo gestionar las reglas de negocio, sino también establecer configuraciones que permitan una api más escalable de manera organizada.

# Swagger

La api se encuentra versionada, en principio sólo se encuentra la versión 1, pero se encuentra lista para generar una próxima versión de parte o la totalidad de sus recursos expuestos.

Se prevee mas iteraciones para aumentar posibles gestiones y mejoras en otros aspectos.

Esta herramienta fue integrada a la api con la finalidad de permitir una mejor comprensión de los recursos expuestos, poder hacer pruebas de los mismos y conocer en una primera aproximación la información gestionada.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

# Colección – Recursos

Los recursos de la api fueron testeados de la herramienta denominada Postman, quedando la colección ordenada como se muestra en la siguiente captura.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Se crearon dos endpoints con la finalidad de exponer la salud de la api como también de sus dependencias, en este caso puntual por el momento con sql server (el motor de base de datos) implementado, pero están tagueados para poder seguir registrando mas posibles despendencias a futuro.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Una prueba de comprobación del estado de la api de forma local podría ser desde un navegador <https://localhost:5001/api/health>, obteniendo el siguiente resultado:

Texto

Descripción generada automáticamente

Otra prueba de comprobación del estado de las dependencias de la api de forma local podría ser desde un navegador <https://localhost:5001/api/health>, obteniendo el siguiente resultado:

Texto

Descripción generada automáticamente

Otra utilidad que podrían permitir ambos recursos es a nivel de infraestructura la posibilidad de una auto gestión de contenedores ante fallos. Como también permitir a otros posibles microservicios informarse de la situación en la cual se encuentra nuestra api para implementar estrategias de reintentos entre otras de resiliencia.

# Logs

La api esta configurada para gestionar logs hacia la base de datos como también hacia un directorio físico en la raíz del proyecto. Esto último para compensar posibles fallos con la base de datos.

Para la gestión de logs se utilizó el paquete “Serilog”, provisto con configuraciones personalizadas mediante el archivo “appsettings.dev.json”.

# Manejo centralizado de errores - Middleware

Se realizó la implementación de un middleware para gestionar de forma global y centralizada las excepciones tanto de negocio como de aplicación, buscando integrar la gestión de logs con errores, respuesta mas acorde al cliente de la api, retornando un formato más amigable.