Rocío Seoane Fernández
2º DAM
2021/2022

ÍNDICE:

1. Descripción del proyecto y ámbito de aplicación

1.1 ¿Qué has desarrollado y por qué?

Para mi proyecto de fin de ciclo he desarrollado una API REST para una casa de apuestas. Me he decidido por este proyecto, ya que, en las prácticas de fin de ciclo nuestro tutor, Alberto Botana, nos hacía formaciones, y en una de ellas nos tocó hacer una aplicación de una casa de apuestas en consola y desarrollar la API pertinente a esta aplicación.

También he desarrollado una página web para probar la API desde ella.

1.2 ¿Quién lo usará?

Esta API, una vez conectada a la aplicación y/o página web, podrá ser utilizada por todo el mundo, mayor de edad, que le guste gastar su tiempo y dinero en hacer apuestas.

Por el momento, podrá ser utilizada por los desarrolladores que tengan las credenciales pertinentes a este proyecto.

1.3 ¿Qué tecnologías has usado?

Para este proyecto he utilizado diferentes tecnologías. Para la base de datos he utilizado *Microsoft SQL Server Management Studio 18,* con el lenguaje estándar de SQL. Para el desarrollo del API he utilizado *Visual Studio 2022,* con lenguaje C# y la plantilla que trae interna Visual Studio para el desarrollo de APIS (ASP .NET Core Web API). Para el desarrollo de la página web he utilizado Notepad++, con el lenguaje HTML. Y, para finalizar, he desarrollado las modificaciones de la página web en Visual Studio Code, con el lenguaje CSS.

1.4 ¿Se prevén cambios en el futuro?

Si, los cambios que se prevén en un futuro son: desarrollar una aplicación, sea en Android, WPF o Web; implementar seguridad, para que no sea posible los hackeos y que no todo el mundo pueda acceder a esta API; y, por último, diseñar el API de forma más agraciada.

2. Recursos hardware y software

2.1 ¿Qué requisitos se necesitan para probar el proyecto (procesador, memoria, sistema operativo…)?

He probado con dos dispositivos:

	HP	Lenovo
Procesador	Intel® Core™ i3-5005U CPU @ 2.00GHz 2.00 GHz	11th Gen Intel® Core™ i5-1135G7 @ 2.40GHz 2.42GHz
RAM Instalada	4,00 GB	16,0 GB (15,7 GB usable)
Tipo de sistema	Sistema operativo de 64 bits, procesador basado en x64	Sistema operativo de 64 bits, procesador basado en x64
Edición	Windows 10 Home	Windows 10 Pro
Versión	21H2	21H2

Bajo mi experiencia, el dispositivo con el que mejor me he manejado y mejor me ha funcionado para este proyecto, fue el Lenovo.

2.2 ¿Has empleado algún IDE o complemento en especial?

Si. Para el desarrollo del código del API he empleado Visual Studio 2022, donde he seleccionado la plantilla de ASP.NET Core Web API.

2.3 ¿Dispones de entorno de desarrollo y entorno de producción diferenciados?

No. Este proyecto está desarrollado y ejecutado en el mismo dispositivo. Se puede acceder a toda parte del proyecto en remoto.

3. Temporalización del proyecto y fases del desarrollo

3.1 ¿Qué planificación has seguido para llevar a cabo el proyecto?

Como se puede ver en el GitHub de este proyecto, he creado distintas *Issues* para tener un desarrollo continuo en el proyecto. Y a parte, he llevado la continuación del desarrollo en una agenda, apuntando lo hecho diariamente, llegando así a poder desarrollar el Gantt y PERT correspondientes.

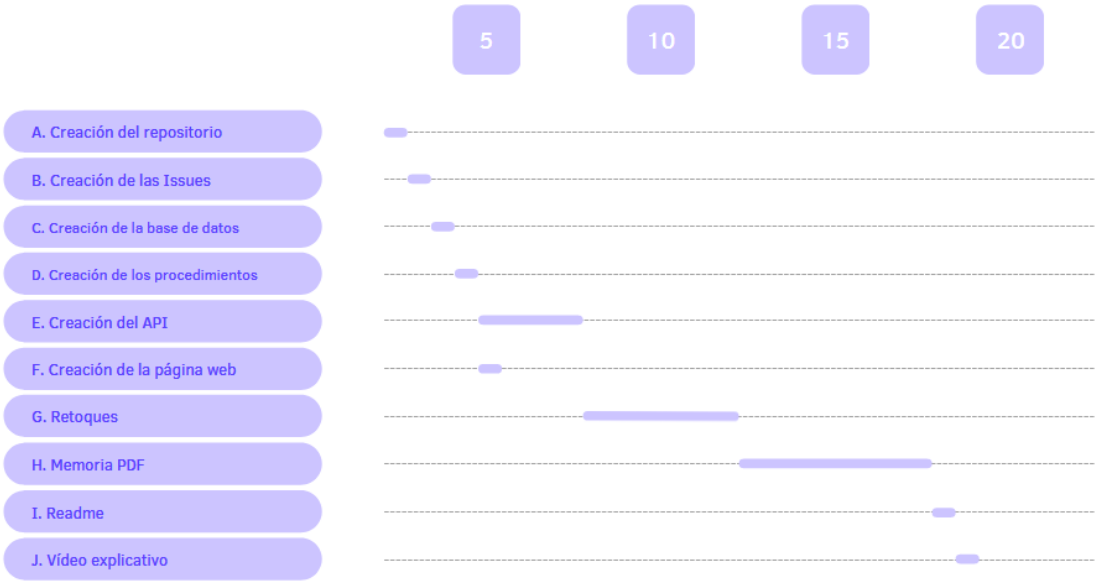
Acción	Duración	Precedente
A. Creación del repositorio	1 día	
B. Creación de las Issues	1 día	A
C. Creación de la base de datos	1 día	B
D. Creación de los procedimientos	1 día	C
E. Creación del API	3 días	D

F. Creación de la página web	1 día	D
G. Retoques	4 días	E, F
H. Memoria PDF	5 días	G
I. Readme	1 día	H
J. Vídeo explicativo	1 día	I

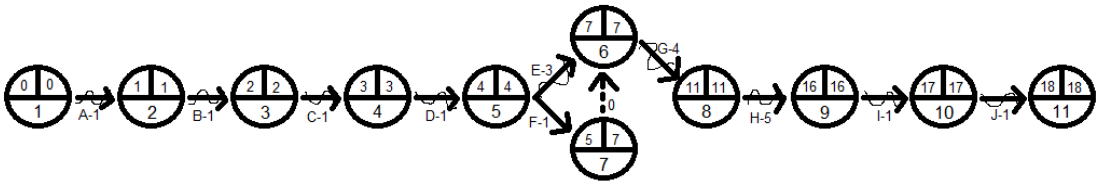
3.2 ¿Puedes plasmarla en diagramas Gantt/PERT?

Si. A continuación, se muestran los diferentes diagramas:

Gantt:



PERT:



3.3 ¿Con qué retos te has enfrentado?

Me he encontrado con varios retos, que también presento en los anteriores diagramas, como, por ejemplo, anteriormente a este proyecto y repositorio, había creado uno, que,

por los errores en la solución de Visual Studio 2022 tuve que crear otro, en el que me daba el mismo error. Por lo que decidí crear un nuevo repositorio y proyecto, en este repositorio, decidí crear la aplicación por consola y el API, la aplicación se ejecutaba con un error, el cual mostraba que un archivo necesario, no estaba en la carpeta pertinente de ese proyecto. Por lo que decidí entregar solo el API.

3.4 ¿Qué has aprendido?

He aprendido a desarrollar una API desde cero, a utilizar un nuevo IDE, Visual Studio Community; y nuevos lenguajes, C# y CSS.

También he podido desarrollar mis conocimientos sobre SQL Server y poner en práctica los conocimientos adquiridos, en el primer año, de HTML.

4. Descripción de datos

4.1 Las partes significativas del proyecto… ¿Cómo funcionan?

Para comenzar, se necesita una base de datos, para almacenar los datos que vamos a utilizar y manejar, para almacenar los procedimientos necesarios para la funcionalidad del API y esta está conectado al código del API mediante una clase en el proyecto de Visual Studio 2022.

Para el API, se necesitan varias clases, como por ejemplo el Controller, en donde definimos el código de este, que es donde se intercambia los datos con la base de datos; en otra clase, que denominamos Conexión, desarrollamos el código pertinente que conecta el API con la base de datos. Y para finalizar en otra clase, DTO, definimos los datos que llegan de la base de datos.

Cuando se ejecuta el programa, se abre un Swagger, en donde se introducen o muestran datos, que son recibidos o introducidos en la base de datos, que llegan a través del Controller del API.

4.2 ¿Puedes elegir ciertos procesos que sean clave y detallarlos?

Para iniciar, tenemos el proceso de registro, para esta funcionalidad tenemos un procedimiento creado en la base de datos en donde verificamos si el usuario insertado ya se encuentra en la base de datos o no. Si no se encuentra se inserta en la tabla usuarios, si se encuentra devuelve un error. En el Controller, tenemos dos métodos definidos para este proceso: uno mediante el Swagger y otro mediante una página web.

Para el Swagger introducimos los datos necesarios al darle a *Try it Out* y los ejecutamos, si no está registrado el usuario insertado, se producirá el código 200 y podremos visualizarlo ejecutando en la base de datos *select * from usuarios.* Si está registrado, se

mostrará el código 400, y, por último, si se produce algún error en el servidor se mostrará el código 500.

Para la página web introducimos los datos en la parte de *Registrarse/web* en donde introducimos los datos pertinentes en su celda y lo ejecutamos al pulsar el botón *Enviar*. En este caso no se puede diferenciar si se ejecuta correctamente, la pantalla se pondrá en negro y debemos ejecutar la consulta *select * from usuarios* para verificar si se ejecutó correctamente o ya estaba registrado el usuario.

A continuación, tenemos el proceso de logearse, para esta funcionalidad tenemos un procedimiento creado en la base de datos en donde verificamos si el usuario está registrado o no. Si no está registrado, se devuelve un error, si está registrado y no se loguea con los datos correctos se muestra otro error, y si está registrado y logueado correctamente se muestra el id del usuario.

En el Swagger introducimos los datos necesarios al pulsar en *Try it Out* y lo ejecutamos, si no está registrado el usuario insertado, se producirá el código 500. Si está registrado y no se introducen los datos correctos se produce el código 400, y si está registrado y se loguea correctamente se produce el código 200 y se muestra el id del usuario.

Seguidamente, tenemos el proceso de mostrar, para la funcionalidad de mostrar tenemos diferentes métodos dependiendo del procedimiento creados en la base de datos.

En el Swagger, pulsamos en *Try it Out*, si necesitamos insertar datos, y lo ejecutamos, si se ejecuta correctamente se produce el código 200 y se muestra un json con los diferentes datos. Si se introducen datos incorrectos, se produce el código 200 pero se no se muestra nada; y si surge algún reto interno se produce el código 500.

Posteriormente, tenemos el proceso de insertar, para la funcionalidad de insertar tenemos diferentes métodos dependiendo del procedimiento creados en la base de datos. Para estos métodos, en el Controller, tenemos dos métodos definidos para este proceso: uno mediante el Swagger y otro mediante una página web.

Para el Swagger introducimos los datos necesarios al darle al *Try it Out* y los ejecutamos, si se ejecutan correctamente, se producirá el código 200 y podremos visualizar los datos insertados ejecutando en la base de datos *select * from "tabla pertinente de la base de datos".* Si se introducen los datos correctos se producirá el código 200, se mostrará el código 400 si los datos introducidos están incorrectos. Por último, si se produce algún error en el servidor se mostrará el código 500.

Para la página web introducimos los datos en la parte de *Insertar"funcionalidad"/web* o *HacerApuesta/web* en donde introducimos los datos pertinentes en su celda y lo ejecutamos al pulsar el botón *Enviar*. En este caso no se puede diferenciar si se ejecuta correctamente, la pantalla se pondrá en negro y debemos ejecutar la consulta *select * from "tabla pertinente de la base de datos"* para verificar si se ejecutó correctamente.

Para finalizar, tenemos el proceso de apuestaGanada, para esta funcionalidad tenemos un procedimiento creado en la base de datos en donde verificamos si la apuesta anteriormente creada está ganada o no. En el Controller tenemos dos métodos definidos para este proceso: uno mediante el Swagger y otro mediante una página web.

Para el Swagger introducimos los datos necesarios al darle a *Try it Out* y los ejecutamos, si se ejecuta correctamente, se producirá el código 200 y podremos visualizarlo ejecutando en la base de datos *select * from apuestas.* Si se introducen datos erroneos, se mostrará el código 400, y, por último, si se produce algún error en el servidor se mostrará el código 500.

Para la página web introducimos los datos en la parte de *HacerApuesta/web* en donde introducimos los datos pertinentes en su celda y lo ejecutamos al pulsar el botón *Enviar*. En este caso no se puede diferenciar si se ejecuta correctamente, la pantalla se pondrá en negro y debemos ejecutar la consulta *select * from apuestas* para verificar si se ejecutó correctamente.

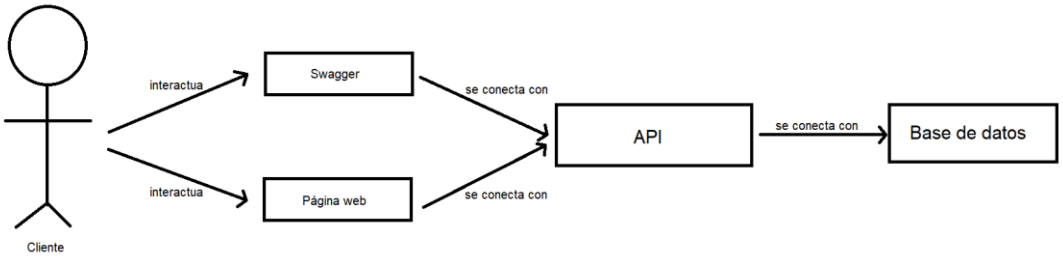
4.3 ¿Qué tipos de datos se manejan (entrada, intermedios, salida)? ¿Cómo?

En este programa, al ser un API y no estar conectada con una aplicación, se manejan datos de entrada y de salida. Por ejemplo, para las pruebas de get, se manejan datos de entrada. Y para las pruebas post, se manejan datos de salida.

En un futuro, al crear la aplicación y conectarla con esta, manejará datos intermedios entre la aplicación y la base de datos.

5. Arquitectura software y de aplicación

5.1 ¿Puedes detallar en un UML los componentes software del proyecto y su interrelación?



5.2 ¿Has aplicado algún patrón de diseño o arquitectura?

No, para esta aplicación no he tenido la necesidad de aplicar ningún patrón de diseño o arquitectura. En un futuro se podrá usar el patrón cliente-servidor, cuando se desarrolle la aplicación pertinente, y/o el patrón de bus de evento, si se desarrolla la aplicación en Android.

5.3 Y dejando a un lado el código… ¿Cómo se explica a alto nivel el funcionamiento de la aplicación?

Para que la aplicación funcione correctamente, al abrir el Management Studio conectamos la base de datos y abrimos el sprint con el código pertinente, el .sql. A continuación, abrimos el proyecto donde se encuentra el código del API, el. snl; y lo ejecutamos.

En el Swagger, la página que se abre al ejecutar el proyecto, podremos insertar datos al pulsar *Try it Out* y *Execute* para que los datos se ejecuten.

Por último, en la página web, el .html, introducimos datos en las celdas pertinentes, al pulsar *Enviar* aparecerá una página en negro. Eso significa que se ejecutó.

5.4 ¿Qué actores (humanos, dispositivos, BBDD, …) hay involucrados y cómo están conectados?

Hay varios actores involucrados en este proyecto. Para comenzar, tenemos la base de datos en donde se encuentran los datos maestros, las tablas y los procedimientos necesarios para el proyecto. También tenemos los humanos, que son los que prueban el proyecto, están conectados a la base de datos mediante el API.

Y para finalizar, el API, en una clase, está desarrollado el código para su funcionamiento, se ejecuta un Swagger o la página web, en donde el actor humano puede jugar con el proyecto. En otra clase se encuentra el desarrollo del código para la conexión con la base de datos.

5.5 ¿Se usan protocolos o estándares específicos en esa comunicación?

Si. Se utilizan protocolos de comunicación y protocolos de red. Para que se produzca la comunicación entre el API y la base de datos se utiliza el protocolo de comunicación, y para que la comunicación en remoto funcione correctamente se usa el protocolo de red.