2023-Noviembre

OBLIGATORIO BASES DE DATOS

Link del proyecto: https://github.com/IgnacioPerez98/Obligatorio-BBDD.git

Lucas Alegre

Nicolas Rodríguez

Ignacio Perez.

Elección de la base de Datos.

Por requerimiento del equipo docente, tuvimos que seleccionar una base datos, de tipo SQL.

En caso de no ser así, también abríamos seleccionado una base SQL , ya que es muy probable que la aplicación sea de uso interno y corra en un servidor interno de la Universidad, por lo que primaria la consistencia y disponibilidad, sobre el particionamiento.

Antes de comenzar el proyecto, definimos que base de datos usar, ya que había varias opciones, entre ellas Microsoft SQL Server, MySQL, PostgreSQL, y SQL Lite.

La primera opción, era usar SQL Server de Microsoft, como el backend está desarrollado en .NET, este producto tiene una gran compatibilidad, ya que están hechos por el mismo fabricante. Lo descartamos, porque es una base de datos propietaria, y si bien tiene una versión gratuita, la misma admite como máximo 1.000.000 de tuplas, además de que el costo va de 290USD a 13.000 USD (Microsoft, 2023)

Otra opción, era SQL Lite. Es una base de datos portátil, que se usa en varios dispositivos principalmente móviles (WhatsApp por ejemplo.). La descartamos porque no es una solución escalable, y tiene problemas con la concurrencia. Su uso es ideal para app pero una web puede tener solicitudes concurrentes y no nos pareció adecuado usar este producto.

La decisión final consto de usar MySQL o PostgreSQL. Al discutirlo entre los miembros del grupo basado en nuestras experiencias personales, decidimos usar MySQL, ya que entendemos, que en caso de aplicaciones pequeñas, como es el caso, tiene una relación rendimiento-complejidad de uso mejor que PostgreSQL, y dado los tiempos para desarrollar e implementar una solución funcional, se opto por MySQL, la cual además, se trabajó en el curso.

Elección de Tecnologías de Frontend-Backend:

Para modularizar el desarrollo, decidimos usar Angular para el frontend, y .NET Core para el backend.

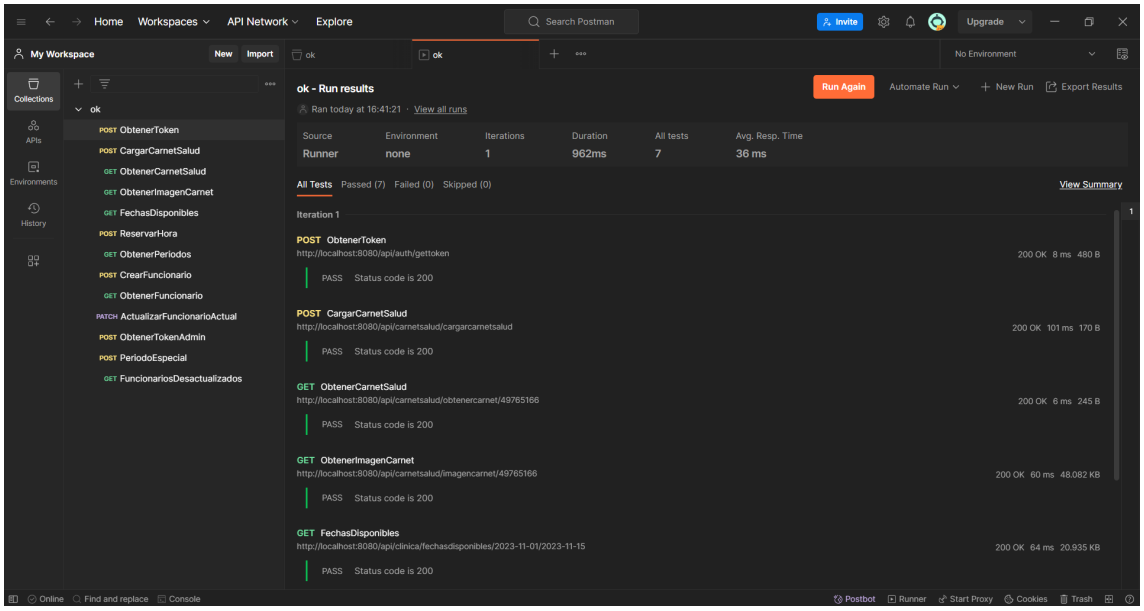
En el caso de backend, c#, nos pareció una mejor opción que usar por ejemplo Sprint, ya que en el curso de Reto, tuvimos que testear el rendimiento de una api con SpringBoot y otra con .NET, y esta ultima presenta una respuesta mas eficiente en lo que respecta a los tiempos, y a la carga recibida.

Por otro lado, el frontend lo realizamos en Angular, ya que es de los frameworks mas robustos que existen hoy en día para el desarrollo de aplicaciones web, y es usado por varias empresas como TATA, GS1 Uruguay o Tienda Inglesa entre otros. Se basa en TypeScript, lo que provee un control de tipo de datos, que nos resulta útil, para realizar validaciones del lado del cliente, y capturar posibles errores, antes de consumir la api. Además es una tecnología que fuimos descubriendo en el curso de desarrollo web por lo que tenemos un conocimiento básico de la misma.

Testing:

Para determinar que la aplicación estaba correcta testeamos el backend y el front de forma independiente.

Para el backend realizamos una colección de Postman, para asegurarnos que los endpoints funcionaran de forma correcta, implementando la función de test.



El frontend, dado que es una aplicación pequeña decidimos testearlo de forma manual, pero si fuera más grande podríamos haber optado por usar Playwright (con Specflow) o Selenium, para automatizar el testing de las diferentes features de la aplicación. Ambas soluciones requieren crear un proyecto, y programar la interacción del browser con la aplicación, por lo que lo descartamos.

Conclusiones:

Cremos que es una buena decisión tener un trabajo de este tipo en la carrera ya que recién finalizando el segundo año realizamos una aplicación que tiene interacción con el usuario final.

El hecho de no poder usar librerías ORM como EntityFramework en nuestro caso, añade una cierta complejidad, ya que las consultas SQL deben realizarse a mano. También debe configurarse aspectos de la conexión como el Pooling, para evitar que la base de datos cree múltiples conexiones. Este no lo sabíamos, pero al testear 300 veces la obtención del token, en la iteración 100, el contenedor de MySQL rechazaba las conexiones y tuvimos que investigar por que se daba.

Respecto a la consultas, usamos una función de la base de datos, que es devolver Objetos y/o Arrays , mapeándolos en la consulta. Esto permite que por ejemplo un listado de horas de Agenda, en lugar de crear un bucle e ir programando los tipos de datos esperados e ir de a uno creando los Objetos, se deserialicen desde un texto con formato JSON, pasando el modelo, lo que hace que esa etapa en particular sea más eficiente.

Para finalizar, seria bueno que este tipo de proyecto lo podamos hacer usando ORM, ya que en la practica es lo que se usa, y tienen otras complejidades. Entender estos frameworks y saber usarlos, creemos que podría haber sido bueno para brindarnos una mejor experiencia del curso.

Referencias:

Microsoft. (27 de 11 de 2023). *Microsoft.com*. Obtenido de Microsoft.com: https://www.microsoft.com/es-mx/sql-server/sql-server-2022-pricing