UTEC
Universidad Tecnológica

UNIVERSIDAD
DE LA REPÚBLICA
URUGUAY

Tecnólogo en Informática
San José de Mayo

UTU - UTEC - UdelaR

Curso: Taller de sistemas de información .NET

Año: 2021

Documento: Lineamientos para el laboratorio final

Docente: Ing. Cristian Bauza

Objetivos

El objetivo de este documento es que los estudiantes del taller de sistemas de información .net tengan un punto de partida claro para comenzar con el laboratorio final y definir lineamientos generales que sirvan de guía para el mismo.

Arquitectura

Para el problema planteado se tendrá como punto de partida las siguientes Tiers:

- Cliente (Navegador WEB)
- Aplicación Web (Front-End, tecnología a elección)
- API Rest (Back-End .NET)
- Base de Datos (SQL Server express o developer de modo local o instancia de azure para prod)

A nivel de Layers, es decir de la división lógica en cada componente, a nivel de front-end es recomendable seguir alguna arquitectura del tipo MVC que tanto ASP.NET MVC como cualquiera de los frameworks SPA modernos siguen ese patrón o muy similar.
A nivel de BackEnd se visualizan las siguientes layers/componentes:

- Service Layer (Web API). Tener en cuenta las siguientes consideraciones:
    - Es recomendable agregar el soporte para Swagger si no está incluido por defecto.
    - Utilizar autenticación del tipo Bearer Token.
- Business Layer. Tener en cuenta las siguientes consideraciones:
    - Tener en cuenta la definición de interfaces e implementaciones.
- Data Access Layer. Tener en cuenta las siguientes consideraciones:
    - Tener en cuenta la definición de interfaces e implementaciones.
- Shared. Es el componente que tendrá todas las entidades de negocio, data types, enumerados, etc. Es importante tener en cuenta que estas NO son las entidades del que utilizará el ORM para la persistencia de datos.
- Unit Test. Es el componente que tendrá el testing unitario.

Multi Tenant

La descripción del problema indica que hay que implementar alguna especie de multi-tenant para cumplir con los requerimientos. Teniendo esto en cuenta pueden haber distintos enfoques.

- Dado que la complejidad de la aplicación en cuanto a el modelo de datos no es demasiada, tal vez puede ser viable la implementación que agrega a todas las entidades un id de institución, y por tanto cuando un usuario se loguea el sistema mostrará siempre lo que corresponda a su institución.
- En cuanto a los enfoques correspondientes a esquemas o bases de datos independientes, también son viables, aquí el escollo está en investigar cómo hacer esto utilizando entity framework. Hay que considerar además de que deberán tener o un esquema maestro o base de datos maestra donde estén definidas las instituciones y el usuario super admin.
- Puede haber un tercer enfoque que sea resolverlo por el lado de la infraestructura. Como se comentó en el teórico, se puede apostar por pensar la aplicación para una única institución y luego replicar la misma a nivel de infraestructura utilizando obviamente una técnica que facilite esto como puede ser docker.

¿Qué Framework Utilizar Para el BackEnd?

En realidad el problema planteado se puede resolver perfectamente con cualquiera de los dos, es decir con .NET Framework o con .NET Core. Actualmente .NET Core ya tiene una madurez y estabilidad que permite utilizarlo casi en cualquier ámbito.

Otras consideraciones

En cuanto al enfoque para el acceso a la base de datos propia, puede utilizarse tanto el enfoque DB First o Code First, utilizando lógicamente Entity Framework.

Próximo Monitoréo (28/09/021)

Para el próximo monitoreo les voy a pedir que me expliquen cómo van a trabajar, es decir, de qué forma van a dividir tareas y cómo van a gestionar el proyecto. Definan etapas y división de tareas.