En este documento, proporcionaremos una guía paso a paso para la creación de un sistema de login utilizando .NET, comenzando desde la creación de un proyecto Web API hasta la configuración de las bibliotecas de clases y las referencias necesarias entre ellas. Este tutorial está diseñado para desarrolladores que buscan implementar una autenticación robusta en sus aplicaciones .NET.

**Objetivos**

1. Crear un proyecto Web API en .NET.
2. Configurar bibliotecas de clases para la separación de responsabilidades.
3. Establecer y configurar las referencias entre las bibliotecas de clases.
4. Implementar un sistema de login básico que autentique a los usuarios.
5. Requisitos Previos
   1. NET SDK instalado en tu máquina.
   2. Un editor de código, como Visual Studio o Visual Studio Code.
   3. Conocimientos básicos de C# y .NET.
   4. Familiaridad con conceptos de Web API y autenticación.

Paso 1: Crear un Proyecto Web API

Primero, crearemos un proyecto Web API en .NET. Abre tu terminal o línea de comandos y navega hasta el directorio donde deseas crear tu proyecto. Luego, ejecuta el siguiente comando:

dotnet new webapi -n LoginAPI

Creamos en la capa principal del proyecto donde tenemos nuestros controladores, creamos la carpeta extensiones, allí creamos el archivo services, ahí estarán los llamados de los diferentes servicios o clases, es decir aquí encontramos el llamado del servicio que debemos crear en nuestra capa de servicios la cual es la capa de servicios externos como lo es JWT y un servicio que nos brinda la facilidad de encriptar la contraseña.

Creamos en la capa de servicios una subcapa llamada seguridad en ella estará los servicios de conexión a JWT y sus diferentes acciones en cuento la duración del token la llave de la firma etc …

Nos dirigimos a crear un DTO el cual nos permite dar uso del encapsulamiento de datos y enviar únicamente la información detallada.

Creamos cada uno de sus subcomponentes necesarios, para el manejo de datos, envio o respuesta de datos.

Creación de los modelos y DTOs correpondiente a los usuarios, en este caso tendremos diferentes variaciones dependiendo del uso que le demos nuestros usuarios en este caso estamos enfocados al uso de entity framework el cual nos facilita el uso de los usuarios, es decir es una plantilla que nos permite la gestión de los usuarios. En nuestra capa principal específicamente en los servicios realizamos la configuración correspondiente a los llamados y uso de servicios por ejemplo JWT el llamado de nuestro llamado a la conexión de la DB.

En nuestra capa de acceso de datos creamos nuestro archivo de conexión y comunicación entre nuestras clases o entidades y la DB, teniendo en cuenta que previamente debemos tener en nuestro archivo appsettings tenemos configurados los datos de dicha DB. Creamos un modelo que nos permitirá crear datos por defecto a las tablas propias que por medio de entityframework creará para la gestión de usuarios.

Nos dirigimos a nuestra capa de negocio donde se hará la lógica de nuestro aplicativo, en este caso para iniciar creamos nuestra lógica para el manejo de logs del aplicativo, para esto debemos tener una arquitectura del manejo de dichos logs, con su respectivo servicio nos permitirá manejar los tipos de logs que se pueden generar y con ellos su respectivo mensaje.

Creamos las diferentes funciones correspondientes a nuestro login, en este caso tendremos 3 funciones primordiales, login, logout, olvido contraseña y verificación de código de doble factor.

En este caso para el uso del código de doble factor usamos una API externa que nos facilita este envío de mensajes de textos con el mensaje deseado, simplemente cumplimos la creación de un llamado HTTP cliente para hacer uso de esta API. Nos dirigimos a crear nuestra interfaz la cual será usado en nuestros controladores, los cuales usaran una arquitectura limpia del manejo de endpoint, empezando por el login, seguido obtenemos el código de doble factor el cual nos cumplirá con la segunda fase del login y asi continuo generaremos nuestro token , ya como podemos ver en el controlado usaremos la funcionalidad propio de JWT para el manejo de autenticación colocando en cada uno de los enpoints su respectivo tipo de petición y el encabezado de JWT ([HttpGet("logout")] [Authorize])