# **Guía para Implementar un Sistema de Login con JWT y EntityFramework en una Arquitectura en N Capas**

## Introducción

En esta guía, proporcionaremos un paso a paso para la creación de un sistema de login utilizando .NET. Empezaremos desde la creación de un proyecto Web API, pasando por la configuración de las bibliotecas de clases y las referencias necesarias entre ellas. Este tutorial está dirigido a 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 A PASO EXPLICACIÓN:

### 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

### Paso 2: Configurar la Arquitectura en N Capas

La arquitectura en N capas nos permite separar claramente las diferentes responsabilidades de nuestra aplicación. Aquí describiremos las capas típicas y cómo configurarlas:

#### 2.1. Capa de Presentación (Web API)

En esta capa, definimos nuestros controladores que manejarán las solicitudes HTTP entrantes. Dentro del proyecto Web API, crea una carpeta llamada Extensions. Dentro de Extensions, crea un archivo llamado ServicesExtensions.cs donde configuraremos los servicios que utilizaremos, como JWT y la encriptación de contraseñas.

#### 2.2. Capa de Servicios

En la capa de servicios, implementamos la lógica de negocio y los servicios externos. Dentro de esta capa, crea una subcarpeta llamada Security para gestionar la conexión con JWT y sus configuraciones, como la duración del token y la clave de firma.

#### 2.3. Capa de Acceso a Datos

Esta capa se encarga de la comunicación con la base de datos. Configura Entity Framework para facilitar la gestión de los usuarios. Dentro de la capa de acceso a datos, crea un archivo de configuración para establecer la conexión con la base de datos. Asegúrate de que el archivo appsettings.json contenga los detalles de la conexión.

#### 2.4. Capa de Modelos y DTOs

En esta capa, definimos los modelos y los objetos de transferencia de datos (DTOs). Estos DTOs encapsulan y transportan los datos necesarios entre las capas de la aplicación. Crea los modelos correspondientes a los usuarios y los DTOs necesarios para las operaciones de login, como las solicitudes y respuestas.

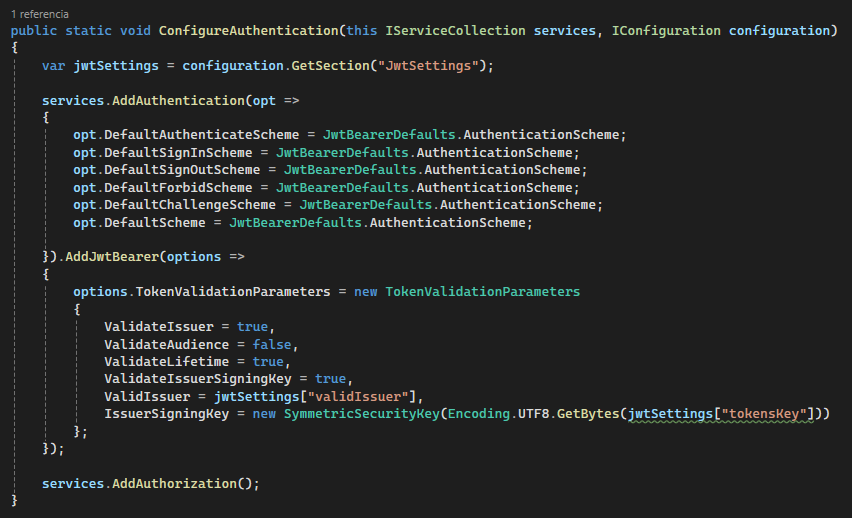
#### 2.5. Capa de Negocio

En la capa de negocio, implementamos la lógica de aplicación. Aquí, gestionamos procesos como el manejo de logs y la autenticación. Crea una lógica para manejar los logs del aplicativo y funciones esenciales para el login, logout, recuperación de contraseña y verificación de doble factor.

### Paso 3: Implementar la Lógica de Autenticación

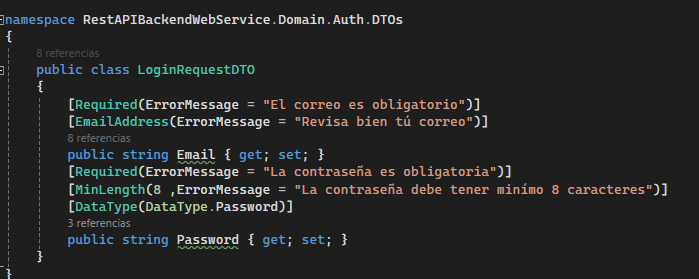
#### 3.1. Configurar JWT

En el archivo ServicesExtensions.cs, configura JWT para gestionar la autenticación:



#### 3.2. Crear DTOs

Define los DTOs necesarios para la autenticación, como LoginRequestDto y LoginResponseDto.

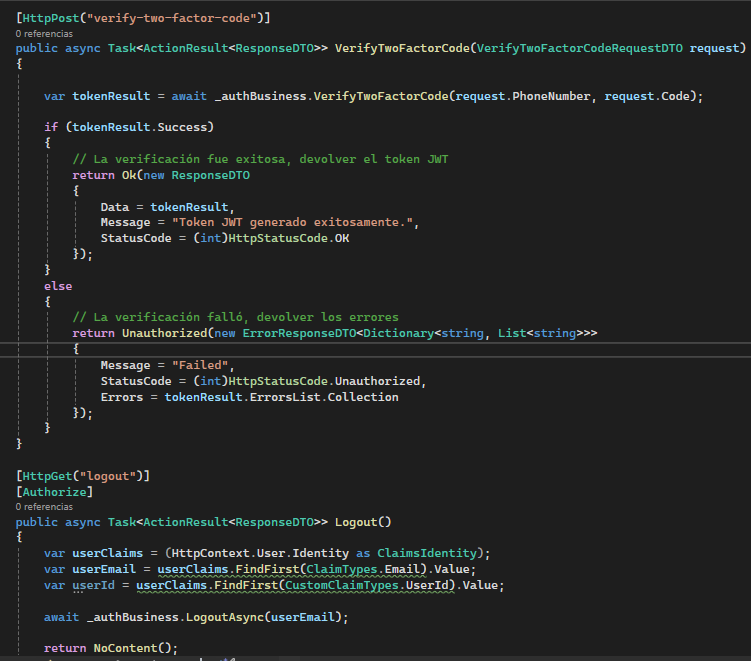


3.3. Implementar Servicios de Autenticación

Implementa los servicios de autenticación en la capa de servicios. Aquí se manejará la lógica para generar y validar tokens JWT, encriptar contraseñas y gestionar usuarios.

#### 3.4. Controladores de la Web API

En los controladores, usa los servicios de autenticación para manejar las solicitudes de login y logout.



### Paso 4: Manejo de la Seguridad

#### 4.1. Encriptación de Contraseñas

Utiliza un servicio para encriptar y comparar contraseñas de manera segura.

#### 4.2. Doble Factor de Autenticación

Implementa una API externa para el envío de códigos de verificación por SMS o correo electrónico. Configura un cliente HTTP para interactuar con esta API.

## Conclusión

Este documento proporciona una guía completa para implementar un sistema de login en una arquitectura en N capas usando .NET. Siguiendo estos pasos, puedes crear una estructura de proyecto robusta y escalable, con una clara separación de responsabilidades y una seguridad mejorada.