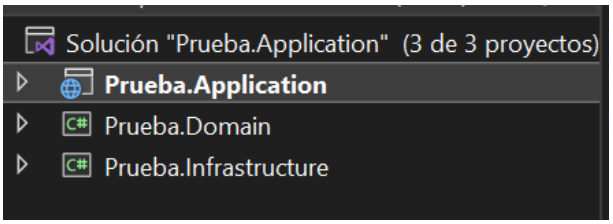
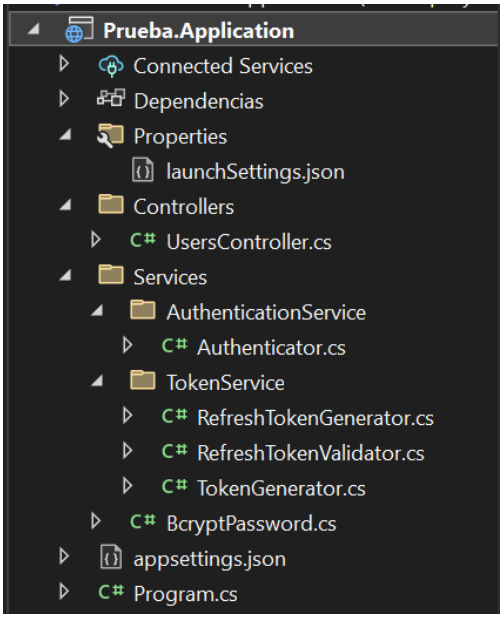
A.- PROYECTO API REST

El proyecto está basado en el patrón arquitectónico DDD y está constituido por 3 ensamblajes.



El ensamblaje o dll Prueba.Application contiene la implementación de la tecnología .Net, sus configuraciones de ejecución y compilación, así como los servicios a utilizar para la aplicación y el controlador que permite acceder a la API REST.

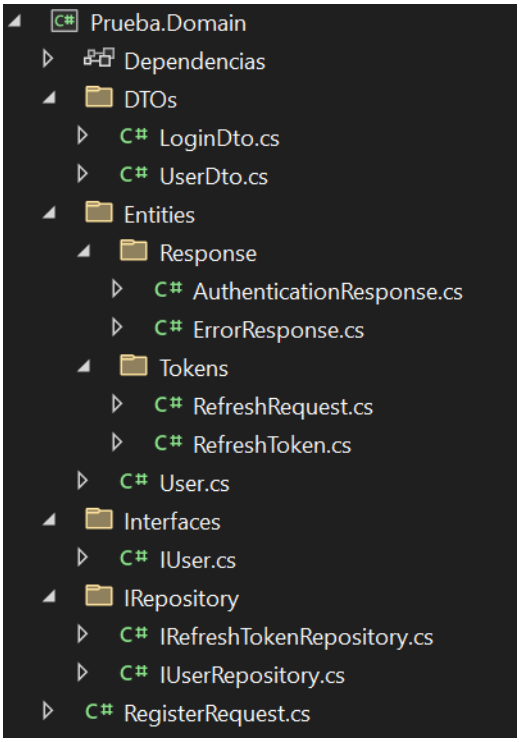


Entrando en detalles de esta dll, en la carpeta Services se tienen 2 subcarpetas en las que se encuentran el servicio de Autentificación y el servicio de Tokens, el servicio de tokens se encarga de la generación del token de usuario según sus datos de inicio de sesión, la creación de los RefreshTokens los cuales permiten mantener una sesión iniciada de forma constante y de validar si el token entregado es válido. Por otro lado el servicio de Autentificación se encarga de llamar a los servicios de tokens para poder ensamblar la respuesta del sistema que permita obtener los tokens del usuario, además se encarga de guardar el refreshToken en la base de datos para poderlo tenerlo de referencia para cuando se requiere renovar la sesión.

El archivo program se encarga de la configuración de todos los servicios, así como de la implementación del contexto de base de datos y de establecer la conexión con la misma, aparte de implementar la autentificación en la aplicación y permitir el acceso a la misma por medio de los CORS.

El controlador es el medio por el que se hacen las llamadas correspondientes para el consumo de servicios, tienen dentro métodos para iniciar sesión, crear un usuario, cerrar sesión, obtener datos del usuario logeado y en el caso de tener permisos de admin, de poder eliminar y ver los demás usuarios creados.

El ensamblaje o dll Prueba.Domain contiene principalmente las entidades que se utilizaran para los métodos de la aplicación y en el caso de que la aplicación requiera implementar una lógica de negocio, esta se implementaría en esta dll.



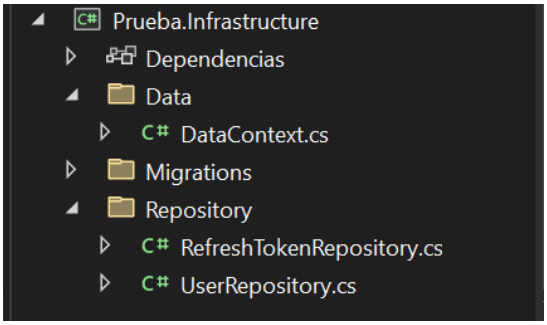
Los DTOs que se encuentran en la carpeta homónima, se utilizan para enviar datos de inicio de sesión(con el objeto LoginDto) y obtener los datos de muestra para consultas de usuario(UserDto).

La carpeta Entities contiene los Response, es decir objetos que tienen una estructura especifica para mostrar un conjunto de datos, entre los que tenemos AuthenticationResponse el cual es el objeto que permite devolver el token y refreshToken del usuario, y ErrorResponse el cual se utiliza para dar un mensaje en caso de error. La carpeta Tokens contiene los modelos para pedir los tokens, y el modelo en que se basa la base de datos para la tabla RefreshTokens.

La carpeta interfaz contiene la interfaz del usuario, en caso de implementar otro tipo de usuario y la carpeta IRepository contiene la interfaz de los repositorios que consumirá la aplicación, en caso de implementar otro servicio de base de datos, la implementación seria mas sencilla.

Por ultimo esta el modelo user en que se basa la base de datos para la tabla Users, y el objeto RegisterRequest se encarga de llevar los datos necesarios para registrar al usuario en la base de datos.

El ensamblaje o dll Prueba.Infrastructure contiene la declaración del contexto de base de datos, de los repositorios y las migraciones que se generen



En la carpeta Data se tiene el contexto de base de datos, el cual utiliza EntityFrameworkCore 6 para virtualizar la base de datos en la aplicación web, lo que permite acceder a los métodos de la base de datos.

La carpeta Migrations contiene el registro de las migraciones que se realicen para actualizar la base de datos en base al código.

La carpeta Repository contiene la declaración de los repositorios para el modelo User y el modelo RefreshToken, así mismo tiene los métodos correspondientes para obtener, subir, editar y borrar registros de la base de datos.

B .- PRUEBA LOGICA. –

Explicación del diagrama de clases:

La clase vehículo tiene una conexión muchos a 1 con la clase Taller, el Taller puede tener varios vehículos pero el vehículo solo puede estar en un taller a la vez, eso se representa con el objeto TallerVehiculo, el cual solo puede albergar 4 vehículos por Taller.

La clase Taller contiene el método RealizarMantenimiento el cual retorna un booleano que indica si se realizó el manteniendo esperado de los vehículos que alberga.

La clase Mantenimiento, contiene el método procesarMantenimiento el cual según la relación entre Talleres y TallerVehiculo hace que se empiece a ejecutar todos los procesos de mantenimiento.

Respecto al código, se creó basado en un caso práctico tomando en cuenta 8 vehículos y 1 taller,

Por lo que el tiempo esperado sería menor a 12 minutos, la ejecución se logra en tiempo descartando los vehículos con fallas fatales, enviándolos directo a un taller externo para su reparación