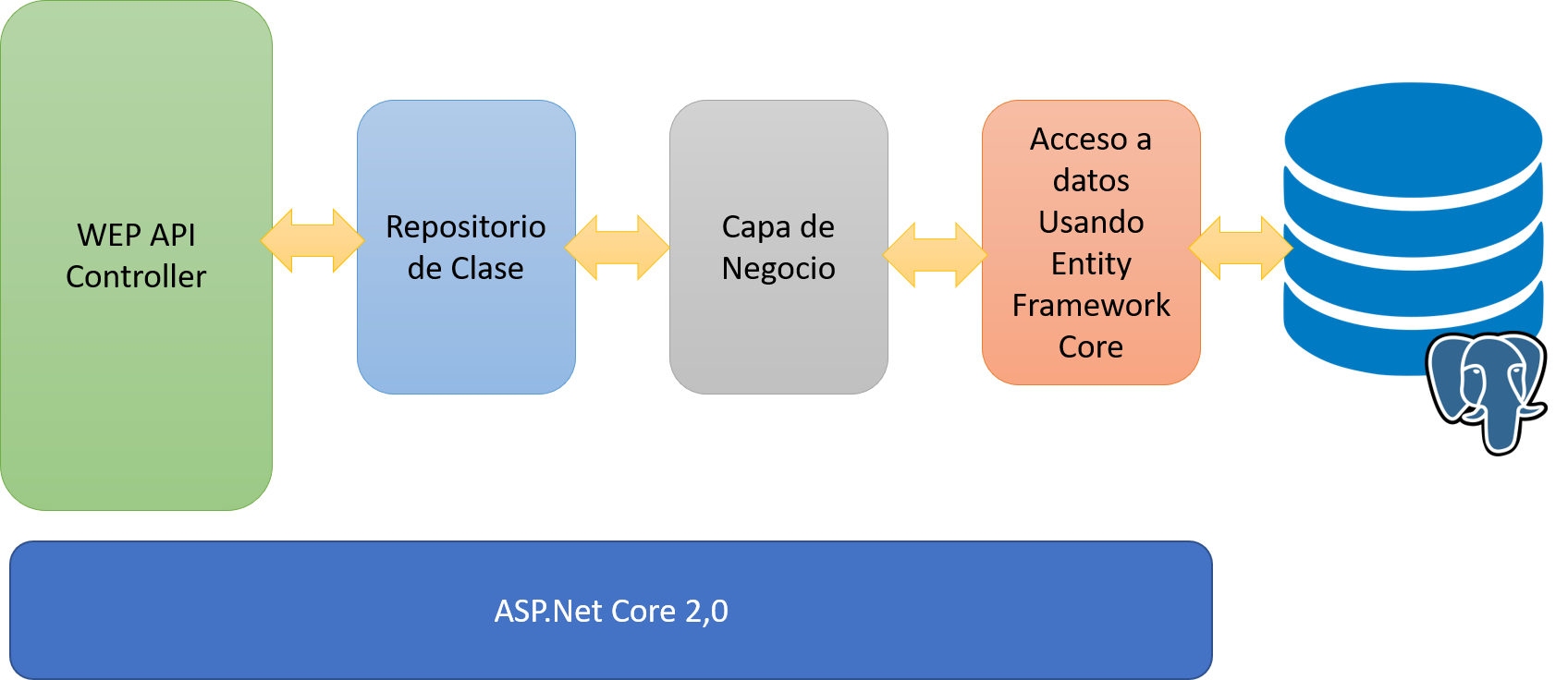
**Prueba Backend**

**Objetivos**

* Evaluar los conocimientos y habilidades del candidato para el desarrollo de aplicaciones web a nivel de Backend.
* Lenguaje Programación: Conocimiento de un lenguaje de programación adecuado para la construcción del Backend de la aplicación web. Opciones: Java, CSharp, PHP
* Base de Datos: Conocimientos del Lenguaje SQL. Conocimiento de un gestor de base de datos y sus particularidades: Lenguaje de programación de Triggers, Procedimientos Almacenados.
* Opciones: PostgreSQL, SQL Server, MySQL Server
* Principios de Programación SOLID: Conocimiento y aplicaciones de los principios de programación SOLID, así como de los principales patrones de programación.

**Arquitectura de solución**



**Figura 1**. WEB API arquitectura de desarrollo.

Se utiliza una estructura de desarrollo WEP API Core para el desarrollo de la prueba backend. Esta arquitectura puede presentar la lógica empresarial en diversas aplicaciones del cliente mediante la API web, puede usar efectivamente los repositorios para desajustar las capas de acceso a datos y negocios de la clase en el controlador WEB Api, además, puede utilizar la inyección de dependencias (DI) incorporada para inyectar las dependencias requeridas en los objetos. Por ejemplo, el acceso a datos se puede registrar en el DI y se puede inyectar en la capa empresarial, además la capa de negocios se puede registrar en DI e inyectarse en el repositorio.

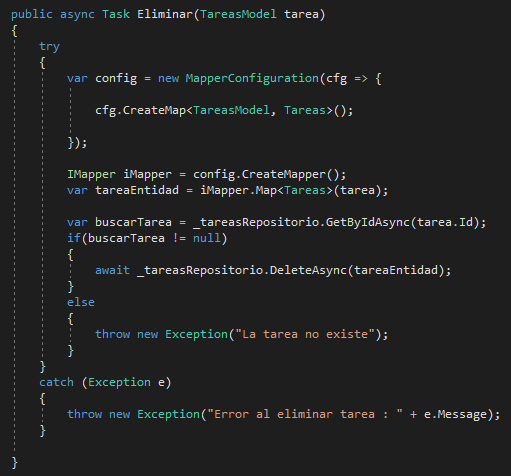
**Estructura de desarrollo**

Se cuenta con una estructura de 5 carpetas denominadas común, negocio, persistencia, servicios y testing.

***La carpeta común***: es la usada para tener las vistas y poder interactuar con los modelos o la capa de negocio. En este caso se usan para interactuar objetos con vistas y pasar la información a la capa de negocio.

***La carpeta de Negocio***: se encuentra la lógica de negocio o el core del negocio. Es la encargada de comunicar con las clases entidad y con las interfaces, además, esta capa contiene todos los algoritmos.

Esta carpeta contiene la lógica del negocio o los algoritmos para hacer funcionar los requisitos solicitados por la prueba. Se tienen dos carpetas Tareas y Usuarios (están se usan para gestionar el sistema), el archivo de tareas cuenta con el CRUD de tareas ver figura 2.



**Figura 2**. Ejemplo del método eliminar tarea.

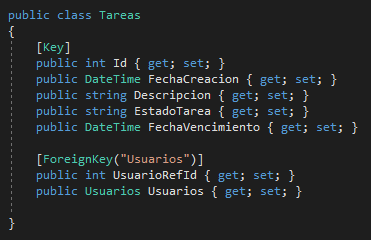
***La carpeta persistencia***: se encuentra todo lo relacionado con las entidades y comunicación con la base de datos, esta carpeta se encarga de transportar los datos con la base de datos y la capa de negocio.

La configuración para la conexión con la base de datos que se realizo fue en **appsettings.json**, en este archivo se configura la ubicación de la base de datos, en este caso se configura se agregar la siguiente línea de comando. “**server=localhost; userid=postgres; database=prueba; password=1234**”.

Seguidamente se ejecuta por la línea de comando cmd **dotnet ef -startup-project ../Servicios/ migrations add initial**

Y **dotnet ef database update**. (Se debe estar ubicado en la capa de persistencia).

Dentro de la estructura de carpetas que se tienen en la capa de persistencia se encuentra una llamada entidades. En esta se crean las entidades de la base de datos ver figura 3.



**Figura 3**. Clase tareas la cual contiene los atributos.

Para poder distinguir los atributos se crean variables para cada tipo, se usan palabras reservadas como Key (para llave primaria), ForeignKey (Llaves foráneas), etc.

Se tiene la carpeta extensiones la cual es usada para inyectar el Dbcontext.

***La carpeta de servicios***: esta es la carpeta encargada de suministrar todos los servicios para ser consumidos.

Entre ellas esta los controladores, archivos de configuración de los servicios, etc.

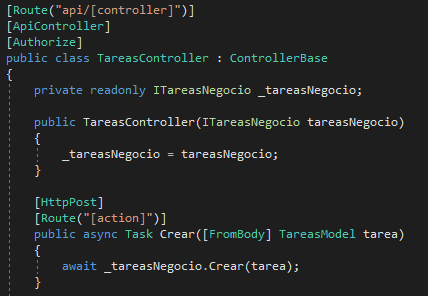
El archivo appsettings.json es usado para la configuración del servicio de base de datos y en este caso para configurar el token para la autenticación.

La ApiAuth es usada para indicar el puerto de configuración el puerto de conexión y la llave secreta para crear la autenticación.

El arhcivo Program.cs es usado para la configuración del puerto de conexión y el llamado a la función startup.cs

El archivo Startup.cs es la que contiene toda la configuración de los servicios para ser consumidos, en este caso se usa la librería Json Wen Tokens, se inyectan las dependencias, se agrega el Dbcontext, la configuración de la Swagger, etc.

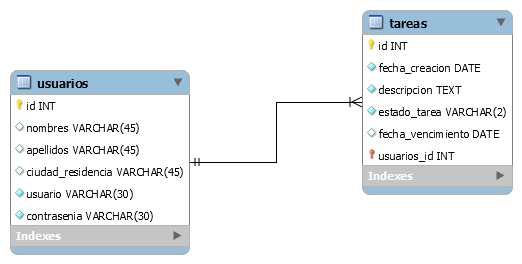
Por otro lado, se encuentra los controladores que serán consumidos por los servicios, este archivo se agrega el tipo de comunicación (Get o Post) ver figura 4, la ruta que tendrá para ser ejecutado, etc.



**Figura 4.** Ejemplo de creación de ruta del servicio para la clase tarea.

***La carpeta de testing***: se realizó para realizar pruebas unitarias de la aplicación.

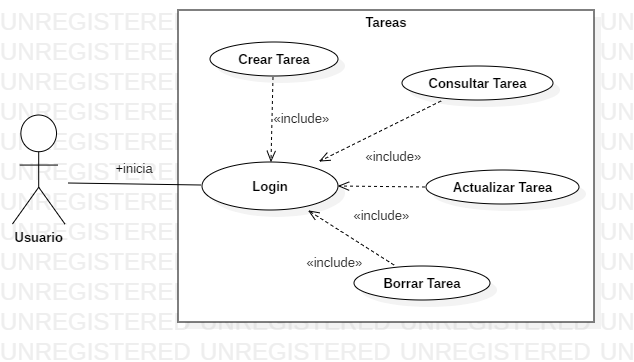
**Base de datos**



**Figura 5.** Modelo entidad relación

El modelo entidad relación desarrollado para el sistema es de dos tablas (usuarios y tareas), el modelo entidad (Ver figura 5) evidencia los atributos de la base datos solicitados por el requerimiento de la prueba.

**Casos de uso**



**Figura 6**. Caso de uso del servicio Web.

El caso de uso como se ve en la figura 6, se inicia al momento de realizar login por el usuario, seguidamente ejecuta cualquiera de las tareas que desee realizar el mismo usuario.

**Consultar Tarea**

Como lo menciona el documento se debe realizar una petición al servidor por método GET, usando la dirección /tareas/consultar y puede realizar cualquiera de las siguientes opciones.

* Consultar todas las tereas
* Consultar solo mis tareas
* Consultar tareas finalizadas
* Ordenar por fecha de vencimiento
* Combinar todas

**Crear Tarea**

Esta realiza una petición al servicio por tipo Post y debe diligenciar todos los campos.

**Actualizar Tarea**

Esta realiza una petición al servicio por tipo Post y debe diligenciar todos los campos.

**Borrar Tarea**

Se realiza por método Post y elimina de la base de datos el registro.

**APIAuth**

Es el servicio de autenticación que se tiene por token por una llave asimétrica.