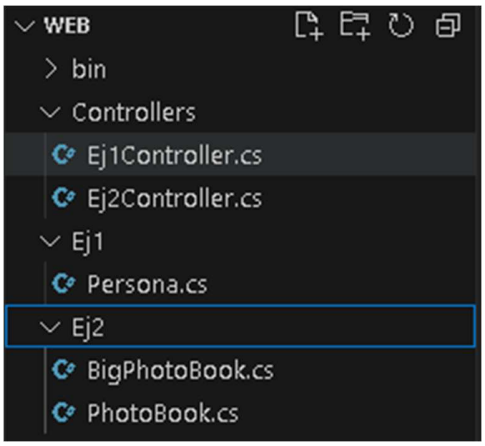
POO

Formato para resolver con un proyecto Web API usando controllers

Por cada ejercicio se debe crear una clase Controller con la siguiente nomenclatura Ej1Controller.cs, Ej2Controller.cs, ... EjnController.cs.

Las clases adicionales que tengan que crear para cada ejercicio, deben colocarlas en una carpeta con la siguiente nomenclatura.
Ej1, Ej2, ... Ejn.

La estructura del proyecto tiene que quedar de la siguiente manera:
Ejemplo para ejercicio 1 y 2



Ej1

Crear un endpoint que reciba tres nombres de personas y retorne una lista con el saludo de cada persona.
Para ello, crear una clase Persona que tenga una propiedad Nombre de tipo string.
Además la clase debe tener un método llamado GetSaludo, que devuelva el siguiente resultado: "Hola! mi nombre es {Nombre}.".

Ej2

Creación de Clases:

Implementar una clase PhotoBook con un atributo protected llamado numPages de tipo int.
Esta clase debe tener un método público GetNumberPages que retorne el número de páginas del álbum.
El constructor por defecto debe inicializar el álbum con 16 páginas.
Incluir un constructor que permita especificar el número de páginas del álbum.

Crear una clase derivada BigPhotoBook cuyo constructor inicialice el álbum con 64 páginas.

API Endpoints:

- Crear Álbum de Fotos Estándar: Un endpoint que permita la creación de un álbum estándar, con la posibilidad de especificar el número de páginas. Si no se especifica, el álbum se creará con 16 páginas por defecto.
- Crear Álbum de Fotos Grande: Un endpoint específico para crear un álbum grande de 64 páginas sin necesidad de especificar el número de páginas.

- Consultar Número de Páginas: Un endpoint para consultar el número de páginas de un álbum específico, identificado por un ID. (agregar el atributo Id a la clase PhotoBook)
- Retornar todos los álbumes.

Para almacenar los álbumes se debe utilizar una lista estática dentro del Controller.
Los endpoints que crean registros, deben utilizar el verbo POST.
Los endpoints que retornan registros y no hacen modificaciones en los datos, debe utilizar el verbo GET.

Ej3

Las clases principales del programa son las siguientes

Persona
Estudiante
Profesor

Las clases Estudiante y Profesor heredan de la clase Persona.

La clase Estudiante incluirá un método público Estudiar() que retornará “Estoy estudiando”.

La clase Persona tendrá dos métodos públicos Saludar() y SetEdad(int edad) que asignará la edad de la persona.

La clase Profesor incluirá un método público Explicar() que retornará “Estoy explicando”.

Crear también un método público MostrarEdad() en la clase Estudiante que escriba Mi edad es: x años en la pantalla.

Al llamar al método Saludar desde Profesor debe mostrar “Hola soy el profesor {Nombre}” y desde Estudiante “Hola soy el estudiante Nombre”.

Crear endpoints para:
- Crear una nueva Persona y hacer que diga hola
- Crear un nuevo Estudiante, establecer una edad, retornar el saludo y su edad.
- Crear un nuevo Profesor, establecer una edad, retornar el saludo y la explicación.

Ej4

Crear una interfaz IVehiculo con dos métodos, uno para Conducir de tipo string y otro para CargarCombustible de tipo bool que tenga un parámetro de tipo entero con la cantidad de combustible a cargar.
Crear una clase Coche con un constructor que reciba un parámetro de tipo entero, con la cantidad de combustible inicial del coche e implemente los métodos de Conducir y CargarCombustible.

El método Conducir retornará el texto "El coche está siendo manejado" si la cantidad de combustible es mayor a 0, de lo contrario retornará el texto "El coche no tiene combustible".
El método CargarCombustible aumentará el combustible del coche y retornará verdadero.

Crear un endpoint que reciba un valor entero de cantidad de combustible, cree un coche con dicha cantidad de combustible, y retorne el mensaje que corresponde a conducir el coche.

Crear otro endpoint que reciba un valor entero de cantidad de combustible, cree un conche sin combustible, le cargue la cantidad indicada en el parámetro, y retorne el mensaje que corresponde a conducir el coche.

Práctica adicional

[Object-Oriented programming (C#) Tutorial]