Taller de Lenguajes II – 2025
Programador Universitario / Licenciatura en informática / Ingeniería en Informática
Trabajo Práctico N° 5

CONTENIDOS

- Implementación de ASP WebApi
- Separación de responsabilidades

1) Desde el repositorio utilizado en el TP 4, cree una nueva rama llamada **ConAccesoADatos** para implementar lo solicitado.
2) Migración del Sistema para Cadeteria:

   Con el objetivo de preparar la integración con una base de datos, se implementará una **nueva capa de acceso a datos**.

   Para ello, se decidió **separar la lógica de acceso a datos en clases específicas para cada modelo**, de manera que cada clase gestione exclusivamente la persistencia de su propio archivo JSON. Con esta actualización deberá generar 3 clases, donde cada una de ellas proporcionará el Acceso a Datos a su propio archivo json

   - AccesoADatosCadeteria → accede a Cadeteria.Json
   - AccesoADatosCadetes → accede a Cadetes.Json
   - AccesoADatosPedidos → accede a Pedidos.Json

   Para tal fin debe implementar:

   ➔ Una clase **AccesoADatosCadeteria** con los siguientes métodos:
   - **Cadeteria** Obtener()

   ➔ Una clase **AccesoADatosCadetes** con los siguientes métodos:
   - **List<Cadetes>** Obtener()

   ➔ Una clase **AccesoADatosPedidos** con los siguientes métodos:
   - **List<Pedidos>** Obtener()
   - **void** Guardar(**List<Pedidos>** Pedidos)

Importante: Como consecuencia de estos cambios se deben eliminar las clases o lista estáticas que se usaron hasta ahora para tener persistencia temporal de datos y solo depender de las clases de acceso a datos que toman los datos de Json.

3) Utilice el método Guardar de la clase AccesoADatosPedidos, cada vez que realice alguna de las siguientes tareas:

- Agregar Pedido
- Cambiar Estado un Pedido
- Asignar un Pedido
- Cambiar de Cadete un Pedido

Nota:

- Para inicializar la cadeteria deberá obtener los datos utilizando las 3 clases de Acceso a datos en el constructor del controlador de Cadeteria

A continuación les dejamos una posible implementación

A continuación les dejamos una posible implementación

```
[ApiController]
[Route("[controller]")]
public class CadeteriaController : ControllerBase
{
    private Cadeteria cadeteria; //Cadeteria ya no es clase estática
    private accesoDatosCadeteria ADCadeteria;
    private accesoDatosCadetes ADCadetes;
    private accesoDatosPedidos ADPedidos;

    public CadeteriaController()
    {
        ADCadeteria= new accesoDatosCadeteria();
        ADCadetes= new accesoDatosCadetes();
        ADPedidos= new accesoDatosPedidos();

        cadeteria= ADCadeteria.Obtener();
        cadeteria.AgregarListaCadetes(ADCadetes.Obtener());
        cadeteria.AgregarListaPedidos(ADPedios.Obtener());

    }
//A partir de aquí van todos los Action Methods (Get, Post,etc.)
}
```

Ejemplo de cómo guardar los pedidos después de haberlo dado de alta

```
  [HttpPost("DarAltaPedido")]
   public ActionResult<string> DarAltaPedido(pedido nuevoPedido)
   {
        cadeteria.DarAltaPedido(nuevoPedido);
        ADPedidos.Guardar(cadeteria.ListaPedidos())
        return Created("Pedido dado de alta exitosamente");
   }
```