Repositorio.

Crear una carpeta nueva en el disco e inicialice el siguiente repositorio en esa ubicación
https://tinyurl.com/tl1-2025-tp10

Ejercicio 1:

Dentro de su repositorio cree una carpeta que se llame “Tareas” y en ella crear una aplicación de consola en C# que interactúe con una API REST para obtener datos, los muestre en pantalla y los guarde en el sistema de archivos local.

- Defina una clase en C# llamada `Tarea` que represente la estructura de los objetos devueltos por la API.
- Utilizando la clase `HttpClient`, realiza una petición `GET` asíncrona al siguiente endpoint: `https://jsonplaceholder.typicode.com/todos/`
- A la respuesta JSON de la API, deserialízala en una lista de objetos de tu clase `Tarea` (`List<Tarea>`).
- Recorra la lista de tareas y muestra en la consola el **título** y el **estado** (completada/pendiente) de cada una de forma clara y legible, Al mostrar los resultados en consola, filtra la lista para mostrar primero todas las tareas pendientes y luego las completadas.
- A la lista completa de tareas y serialízala nuevamente a formato JSON y guarda el resultado en un archivo llamado **`tareas.json`** en el directorio de ejecución de la aplicación.

Ejercicio 2:

Dentro de su repositorio cree una carpeta que se llame “Usuarios” y en ella crear una aplicación de consola en C# que interactúe con una API REST para obtener datos, los muestre en pantalla y los guarde en el sistema de archivos local.

- Implemente una aplicación que consuma el siguiente webservice https://jsonplaceholder.typicode.com/users/
- Muestre en pantalla el **nombre** y **correo electrónico** y **Domicilio** de los primeros cinco usuarios.

Nota de Implementación: **Para llevar a cabo estas tareas, se sugiere investigar y utilizar los siguientes elementos del framework .NET:**

- **Para la Conexión y Petición HTTP:**
  - **Clase: `System.Net.Http.HttpClient`**
    - **Es la clase moderna para enviar peticiones HTTP y recibir respuestas. Se recomienda crear una única instancia estática para la aplicación.**
  - **Métodos Clave:**
    - **`GetAsync()`: Para enviar una petición GET de forma asíncrona.**
    - **`EnsureSuccessStatusCode()`: Método del objeto de respuesta (`HttpResponseMessage`) para verificar si la petición fue exitosa (ej. código 200 OK).**
    - **`ReadAsStringAsync()`: Método del contenido de la respuesta (`HttpContent`) para leer el cuerpo del mensaje como un string.**

- **Para el Manejo de JSON:**
    - **Para trabajar con JSON en tus proyectos, necesitarás agregar el paquete `System.Text.Json`: dotnet add package System.Text.Json**
    - **Namespace: `System.Text.Json`**
    - **Clase: `JsonSerializer`**
        - **Método Clave: `Deserialize<T>(string json)` para convertir el string JSON que recibes de la API en una lista de objetos C# (`List<Usuario>`).**
- **Para convertir un archivo json a una clase c#**
    - **Puede usar la pagina `https://json2csharp.com/`**
- **Para visualizar el contenido de un archivo json**
    - **Puede usar la pagina `https://jsonviewer.stack.hu/`**

Ejercicio 3:

Dentro de su repositorio cree una carpeta que se llame “MiWebApi” y en ella crear una aplicación de consola en C# que interactúe con una API REST para obtener datos, los muestre en pantalla y los guarde en el sistema de archivos local.

Busque un servicio web para consumir una api, consuma los datos que devuelve mostrando por pantalla la información relevante. Por último, guarde los datos obtenidos en un archivo Json.

Utilice el listado de APIs públicas para seleccionar un web services que le interese.

Listado de apis públicas
https://www.apispublicas.com/
https://apipheny.io/free-api/#apis-without-key
https://mixedanalytics.com/blog/list-actually-free-open-no-auth-needed-apis/
https://github.com/public-apis/public-apis