**Consumir API REST desde una aplicación de escritorio WinForms**

Este ejemplo se va a hacer sobre la API <https://rickandmortyapi.com/documentation> que contiene información, lo que va a permitir hacer pruebas, sobre personajes, capítulos… de la serie Rick y Morty. En este ejemplo concreto se utilizará el apartado “Get all characters” a través de su URL <https://rickandmortyapi.com/api/character> para hacer una petición GET, de tal forma que la API nos devuelva todo el JSON con información.

Texto

Descripción generada automáticamente

Con toda esta información es con la que se va a trabajar en este ejemplo, obteniéndola para trabajar con ella desde la aplicación que se desarrolle con WinForms.

Creamos un nuevo proyecto de WinForms y se prepara el espacio de trabajo, creando una serie de carpetas donde guardar las clases necesarias para la gestión de la información procedente de la API.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Una vez definido el espacio de carpetas, se va a desarrollar el modelo que se va a utilizar para mostrar los datos. Se puede crear directamente la clase para gestionar toda esta información o se puede utilizar herramientas que faciliten esta tarea. Estas herramientas son POSTMAN y JSON to C# Converter.

Si vamos a POSTMAN <https://www.postman.com/> y utilizamos el GET de characters de la API que estamos utilizando <https://rickandmortyapi.com/api/character>

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Se obtiene una respuesta en la que se puede ver toda la información que nos ofrece la API:

Texto

Descripción generada automáticamente con confianza media

Con esta información, se puede utilizar el otro recurso JSON to C# Converter <https://www.site24x7.com/tools/json-to-csharp.html> que sirve para convertir el JSON que nosotros tenemos en una clase de C# con todas las características de ese JSON para poder serializar la información.

Deberíamos coger un resultado (result) de la información obtenida en el POSTMAN para que nos devuelva esa clase:

{

            "id": 1,

            "name": "Rick Sanchez",

            "status": "Alive",

            "species": "Human",

            "type": "",

            "gender": "Male",

            "origin": {

                "name": "Earth (C-137)",

                "url": "https://rickandmortyapi.com/api/location/1"

            },

            "location": {

                "name": "Citadel of Ricks",

                "url": "https://rickandmortyapi.com/api/location/3"

            },

            "image": "https://rickandmortyapi.com/api/character/avatar/1.jpeg",

            "episode": [

                "https://rickandmortyapi.com/api/episode/1",

                "https://rickandmortyapi.com/api/episode/2",

                "https://rickandmortyapi.com/api/episode/3",

                "https://rickandmortyapi.com/api/episode/4",

                "https://rickandmortyapi.com/api/episode/5",

                "https://rickandmortyapi.com/api/episode/6",

                "https://rickandmortyapi.com/api/episode/7",

                "https://rickandmortyapi.com/api/episode/8",

                "https://rickandmortyapi.com/api/episode/9",

                "https://rickandmortyapi.com/api/episode/10",

                "https://rickandmortyapi.com/api/episode/11",

                "https://rickandmortyapi.com/api/episode/12",

                "https://rickandmortyapi.com/api/episode/13",

                "https://rickandmortyapi.com/api/episode/14",

                "https://rickandmortyapi.com/api/episode/15",

                "https://rickandmortyapi.com/api/episode/16",

                "https://rickandmortyapi.com/api/episode/17",

                "https://rickandmortyapi.com/api/episode/18",

                "https://rickandmortyapi.com/api/episode/19",

                "https://rickandmortyapi.com/api/episode/20",

                "https://rickandmortyapi.com/api/episode/21",

                "https://rickandmortyapi.com/api/episode/22",

                "https://rickandmortyapi.com/api/episode/23",

                "https://rickandmortyapi.com/api/episode/24",

                "https://rickandmortyapi.com/api/episode/25",

                "https://rickandmortyapi.com/api/episode/26",

                "https://rickandmortyapi.com/api/episode/27",

                "https://rickandmortyapi.com/api/episode/28",

                "https://rickandmortyapi.com/api/episode/29",

                "https://rickandmortyapi.com/api/episode/30",

                "https://rickandmortyapi.com/api/episode/31",

                "https://rickandmortyapi.com/api/episode/32",

                "https://rickandmortyapi.com/api/episode/33",

                "https://rickandmortyapi.com/api/episode/34",

                "https://rickandmortyapi.com/api/episode/35",

                "https://rickandmortyapi.com/api/episode/36",

                "https://rickandmortyapi.com/api/episode/37",

                "https://rickandmortyapi.com/api/episode/38",

                "https://rickandmortyapi.com/api/episode/39",

                "https://rickandmortyapi.com/api/episode/40",

                "https://rickandmortyapi.com/api/episode/41",

                "https://rickandmortyapi.com/api/episode/42",

                "https://rickandmortyapi.com/api/episode/43",

                "https://rickandmortyapi.com/api/episode/44",

                "https://rickandmortyapi.com/api/episode/45",

                "https://rickandmortyapi.com/api/episode/46",

                "https://rickandmortyapi.com/api/episode/47",

                "https://rickandmortyapi.com/api/episode/48",

                "https://rickandmortyapi.com/api/episode/49",

                "https://rickandmortyapi.com/api/episode/50",

                "https://rickandmortyapi.com/api/episode/51"

            ],

            "url": "https://rickandmortyapi.com/api/character/1",

            "created": "2017-11-04T18:48:46.250Z"

        }

La clase que nos genera y que debemos utilizar dentro de nuestro proyecto sería esta:

public class Info

{

public int count { get; set; }

public int pages { get; set; }

public string next { get; set; }

public object prev { get; set; }

}

public class Location

{

public string name { get; set; }

public string url { get; set; }

}

public class Origin

{

public string name { get; set; }

public string url { get; set; }

}

public class Personaje

{

public int id { get; set; }

public string name { get; set; }

public string status { get; set; }

public string species { get; set; }

public string type { get; set; }

public string gender { get; set; }

public Origin origin { get; set; }

public Location location { get; set; }

public string image { get; set; }

public List<string> episode { get; set; }

public string url { get; set; }

public DateTime created { get; set; }

}

Una clase C# con todas las características del objeto JSON que nosotros teníamos.

Una vez definida esta clase denominada “Personaje”, se necesita definir otra clase, que en este caso la vamos a llamar “Personajes”, porque lo que necesitamos es una lista de los personajes

public class Personajes

{

public List<Personaje> results { get; set; }

}

Indico que quiero una lista de Personaje y le ponemos el results para que pueda serializar correctamente la respuesta al objeto

Cuando ya se tiene el modelo listo para utilizarse, se necesita crear el controlador, por lo que creamos una nueva clase llamada “ControlPersonajes” que sea pública.

public class ControlPersonajes

{

//Defino una variable privada de tipo HTTP de nombre client

private HttpClient client;

//Definimos el constructor

public ControlPersonajes()

{

//Creamos una instancia de nuestro cliente

client = new HttpClient();

}

//Creamos nuestra petición,

public async Task<Personajes> GetAllPersonajes()

{

try

{

//Hacemos una instancia de Personajes

Personajes personajes = new Personajes();

//Creamos un objeto de tipo HttpResponseMessage, en el que le pasamos la URL

//que se quiere consultar

HttpResponseMessage response = await client.GetAsync("https://rickandmortyapi.com/api/character");

//Verifica que la respuesta tenga un estado de éxito

//Si no es exitosa, lanza una excepción

response.EnsureSuccessStatusCode();

//String con las respuesta que es el JSON con toda la información

//que habíamos visto previamente

string responseJson = await response.Content.ReadAsStringAsync();

//Enviamos esta respuesta a nuestra modelo, convierte (deserializa)

//el JSON recibido en un objeto de tipo "Personajes" utilizando la

//biblioteca Newtonsoft.Json

personajes = JsonConvert.DeserializeObject<Personajes>(responseJson);

//Devuelve el objeto "personajes" con los datos obtenidos de la API

return personajes;

}

catch (Exception)

{

//Si ocurre algún error (como problemos de conexión o un JSON no válido),

//captura la excepción y devuelve "null" como valor de error

return null;

}

}

}

Es importante, para poder llevar a cabo esta clase, tener instalado el paquete NewtonSoft.Json, que se puede instalar desde Herramientas > Administrador de paquetes Nuget > Administrar paquetes Nuget para la solución y ahí buscamos el paquete mencionado y lo instalamos en el proyecto.

Una vez definido todo el entorno, ya se puede volver al formulario, porque ya se puede trabajar con los datos y definimos una interfaz como la que se muestra a continuación:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

En este caso se utiliza un datagrid con columnas para recoger los datos que se consideran más importantes, como el nombre, género, especie y origen.

Cuando se tiene definida la interfaz, nos vamos a código y definimos lo siguiente:

public partial class Form1 : Form

{

//Declaramos el controlador, un objeto que controla la lógica

//para obtener los personajes de la API

private ControlPersonajes ControlPersonajes;

//Inicializamos el modelo, es un objeto que almacena los datos

//deserializados de la API

private Personajes personajes;

public Form1()

{

InitializeComponent();

//Inicializamos el controlador

ControlPersonajes = new ControlPersonajes();

//Instanciamos el modelo

personajes = new Personajes();

}

//Método asíncrono para obtener los personajes de la API

private async void GetPersonajes()

{

//Llama al método GetAllPersonajes para obtener los personajes de la API

//de manera asíncrona

personajes = await ControlPersonajes.GetAllPersonajes();

//Verifica si el objeto personajes no es nulo, es decir, si la llamada a la

//API fue exitosa

if (personajes != null )

{

//Recorre la lista de resultados (personajes) obtenidos desde la API

foreach (var personaje in personajes?.results)// ? e ! para permitir nulos y evitar errores

{

//Crea una nueva fila en el DataGridView

DataGridViewRow row = new DataGridViewRow();

//Asocia la fila al DataGridView especificado (personajes)

row.CreateCells(dgvPersonajes);

//Asigna los valores de las propiedades del personaje a las celdas de la fila

row.Cells[0].Value = personaje.name;

row.Cells[1].Value = personaje.gender;

row.Cells[2].Value = personaje.species;

row.Cells[3].Value = personaje.origin.name;

//Agrega la fila al DataGridView

dgvPersonajes.Rows.Add(row);

}

}

else

{

//Si personajes es nulo (la llamada falló), muestra un mensaje de error al usuario

MessageBox.Show("No se pudo obtener la petición", "Error",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button1\_Click(object sender, EventArgs e)

{

GetPersonajes();

}

}