Ejercicio A.2: Llamadas a WebServices

Microsoft ha restringido el uso de DotNet en el desarrollo de extensiones con Visual Studio Code.

Uno de los usos más habituales de DotNet es el de hacer llamadas a WebServices externos para podernos comunicar con otros entornos.

Para este caso, Microsoft ha proporcionado una forma alternativa de realizar las llamadas a WebServices a través de nuevos tipos de variables.

Datos de conexión

Para este desarrollo, hay que utilizar las siguientes bases de datos:

- Entorno de desarrollo:
  - Servidor: localhost\NAVDEMO
  - BBDD: BCCU0_AppDevelopment
  - Autenticación: Windows

Indicaciones para el ejercicio

Realiza las siguientes acciones:

1. Crea un procedimiento en una nueva codeunit y realiza la llamada a un WebService
2. Crea una acción en la lista de clientes para realizar las pruebas de la llamada a WebService

Indicaciones paso a paso

Sigue los siguientes pasos en Visual Studio Code:

1. Crea la codeunit 50100 "Call To WebService" y el procedimiento CallToWebService

2. Crea las siguientes variables locales

```
var
    Client: HttpClient;
    ResponseMessage: HttpResponseMessage;
    Url: Text;
    ResponseText: Text;
```

3. Establece la siguiente URL:

```
Url :=
'http://data.fixer.io/api/latest?access_key=be1290e2e192ee708f88c5f8bfea895c&base=EUR&symbols=USD,CAD';
```

   Se trata de un WebService que nos devolverá las últimas tasas de cambio de las divisas que le indiquemos.
   El parámetro access_key se ha obtenido tras registrarse de forma gratuita en fixer.io
   El parámetro base=EUR indica que la divisa base es EUR.
   El parámetro symbols=USD,CAD indica que queremos que nos devuelva la tasa de cambio a USD y CAD.

4. Haz una llamada al método Get de HttpClient, pasándole la Url y la variable de HttpResponseMessage

```
    if not Client.Get(Url, ResponseMessage) then
        Error('La llamada al WS ha fallado');
```

5. Lee y muéstrale al usuario la respuesta del WebService

```
    if not ResponseMessage.IsSuccessStatusCode() then
        error('El WS ha devuelto el siguiente error:\  ' +
                'Status Code: %1\' +
                'Descripción: %2',
                ResponseMessage.HttpStatusCode,
                ResponseMessage.ReasonPhrase);

    ResponseMessage.Content().ReadAs(ResponseText);
    Message(ResponseText);
```

6. Crea una acción en la lista de clientes para poder probar la llamada al WebService

```
pageextension 50101 "Call To WebService" extends "Customer List"
{
    actions
    {
        addafter("&Customer")
        {
            action(CallToWS)
            {
                Promoted = true;
                PromotedIsBig = true;

                trigger OnAction()
                var
                    CallToWS: Codeunit "Call To WebService";
                begin
                    CallToWS.CallToWebService();
                end;
            }
        }
    }
}
```

7. Crea las siguientes variables locales en el procedimiento para poder leer el contenido de la respuesta, que tiene un formato Json

```
    Json: Codeunit "Json Text Reader/Writer";
    JsonBuffer: Record "JSON Buffer" temporary;
    BaseText: Text;
```

8. Utiliza el siguiente código para leer el contenido de la respuesta Json

```
    Json.ReadJSonToJSonBuffer(ResponseText, JsonBuffer);

    JsonBuffer.GetPropertyValue(BaseText, 'base');
    Message(BaseText);
```

El procedimiento final debería ser el siguiente:

```
procedure CallToWebService()
var
    Client: HttpClient;
    ResponseMessage: HttpResponseMessage;
    Url: Text;
    ResponseText: Text;
    Json: Codeunit "Json Text Reader/Writer";
    JsonBuffer: Record "JSON Buffer" temporary;
    BaseText: Text;
begin
    Url :=
'http://data.fixer.io/api/latest?access_key=be1290e2e192ee708f88c5f8bfea895c&base=EUR&s
ymbols=USD,CAD';

    if not Client.Get(Url, ResponseMessage) then
        Error('La llamada al WS ha fallado');

    if not ResponseMessage.IsSuccessStatusCode() then
        error('El WS ha devuelto el siguiente error:\  ' +
                'Status Code: %1\' +
                'Descripción: %2',
                ResponseMessage.HttpStatusCode,
                ResponseMessage.ReasonPhrase);

    ResponseMessage.Content().ReadAs(ResponseText);
    Message(ResponseText);



    Json.ReadJSonToJSonBuffer(ResponseText, JsonBuffer);

    JsonBuffer.GetPropertyValue(BaseText, 'base');
    Message(BaseText);
end;
```