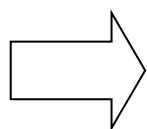
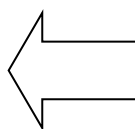


FORMACIÓN PROFESIONAL DUAL



CUADERNO DE INFORMES





DIRECCIÓN ZONAL

ICA/AYACUCHO

FORMACIÓN PROFESIONAL DUAL

CFP/UCP/ESCUELA: SENATI CHINCHA

ESTUDIANTE: LUCAS AFREDO ATUNCAR VALERIO

ID: 139685 BLOQUE: 40PIADS601

CARRERA: ING. DE SOFTWARE CON IA

INSTRUCTOR: JHON EDWARD FRANCIA MINAYA

SEMESTRE: VI DEL: 12/03 AL: 02/04



INSTRUCCIONES PARA EL USO DEL CUADERNO DE INFORMES DE TRABAJO SEMANAL

1. PRESENTACIÓN.

El Cuaderno de Informes de trabajo semanal es un documento de control, en el cual el estudiante, registra diariamente, durante la semana, las tareas, operaciones que ejecuta en su formación práctica en SENATI y en la Empresa.

2. INSTRUCCIONES PARA EL USO DEL CUADERNO DE INFORMES.

- 2.1 En el cuadro de rotaciones, el estudiante, registrará el nombre de las áreas o secciones por las cuales rota durante su formación práctica, precisando la fecha de inicio y término.
- 2.2 Con base al PEA proporcionado por el instructor, el estudiante transcribe el PEA en el cuaderno de informes. El estudiante irá registrando y controlando su avance, marcando en la columna que corresponda.
- 2.3 En la hoja de informe semanal, el estudiante registrará diariamente los trabajos que ejecuta, indicando el tiempo correspondiente. El día de asistencia al centro para las sesiones de tecnología, registrará los contenidos que desarrolla. Al término de la semana totalizará las horas.
De las tareas ejecutadas durante la semana, el estudiante seleccionará la más significativa y hará una descripción del proceso de ejecución con esquemas y dibujos correspondientes que aclaren dicho proceso.
- 2.4 Semanalmente, el estudiante registrará su asistencia, en los casilleros correspondientes.
- 2.5 Semanalmente, el Monitor revisará, anotará las observaciones y recomendaciones que considere; el Instructor revisará y calificará el Cuaderno de Informes haciendo las observaciones y recomendaciones que considere convenientes, en los aspectos relacionados a la elaboración de un Informe Técnico (términos técnicos, dibujo técnico, descripción de la tarea y su procedimiento, normas técnicas, seguridad, etc.)
- 2.6 Si el PEA tiene menos operaciones (151) de las indicadas en el presente formato, puede eliminar alguna página. Asimismo, para el informe de las semanas siguientes, debe agregar las semanas que corresponda.
- 2.7 Escala de calificación:

CUANTITATIVA	CUALITATIVA	CONDICIÓN
16,8 – 20,0	Excelente	Aprobado
13,7 – 16,7	Bueno	
10,5 – 13,6	Aceptable	
00 – 10,4	Deficiente	Desaprobado

PLAN DE ROTACIONES

[illegible]

**PLAN ESPECÍFICO DE APRENDIZAJE (PEA)
SEGUIMIENTO Y EVALUACIÓN**

Nº	OPERACIONES/TAREAS	OPERACIONES EJECUTADAS*				OPERACIONES POR EJECUTAR	OPERACIONES PARA SEMINARIO
		1	2	3	4		
01	Desarrollo de modelos IA						
02	Desarrollo de modelos Machine Learning						
03	Desarrollo de modelos IA con WATSON						
04	Desarrollo de modelos con WATSON Studio						
05	Muestra información usando Master-detail and Drawer Navigation						
06	Personaliza ListView						
07	Utiliza MVVM						
08	Consume servicios REST						
09	Publica aplicaciones						
10	Calcula costes de uso de servicios en la nube						
11	Diseña infraestructura de red en la nube						
12	Asegura la conectividad de red en Microsoft Azure						
13	Diseña Data Warehouse y Arquitectura BI						
14	Diseña Transacciones – Auditorías – Power View						
15	Diseña Dashboard: SQL Server Reporting Services						
16	Aplicar en un caso práctico la configuración y uso de almacenamiento de datos con BI						
17	Implementa paquete Neuralnet						
18	Implementa paquete H2O						
19							
20							
21							
22							
23							
24							
25							
26							
27							
28							
29							
30							
31							
32							
33							
34							

Llenar según avance

INFORME SEMANAL

.....SEMESTRE SEMANA N°..... DEL AL DEL 20.....

DÍA	TRABAJOS EFECTUADOS	HORAS
LUNES		
MARTES	MANEJO DE ARCHIVOS BINARIOS MEDIANTE XAMARIN FORMS Y PHP(APP MOVIL)	5
MIÉRCOLES		
JUEVES		
VIERNES		
SÁBADO		
TOTAL		5

Tarea más significativa:

MANEJO DE ARCHIVOS BINARIOS MEDIANTE XAMARIN FORMS Y PHP(APP MOVIL)

Descripción del proceso:

Habiendo tenido avances en la creación de aplicaciones móviles para Android en Xamarin Forms, como parte de los temas programados, estudiamos el manejo de archivos binarios o siendo más específicos, con imágenes, teniendo nuestra API, la configuramos para poder subir una imagen desde nuestro celular por 2 modalidades:

- Siendo subidas desde la galería de imágenes del celular.
- Tomando fotos desde la cámara del celular.

Esta tarea trajo consigo el conocimiento de conceptos nuevos de la seguridad en un móvil y la forma como se trata dicha información, partiendo de la idea que se debe otorgar permisos a nuestra aplicación para que pueda realizar ciertas tareas que el usuario decidiría entre dar permiso o no, esto complementa lo que en sesiones anteriores se vino enseñando sobre la forma tan distinta que es la programación móvil.

This image shows a single sheet of white paper with horizontal blue or grey ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

HACER ESQUEMA, DIBUJO O DIAGRAMA

La siguiente figura muestra el archivo PHP encargado de la operación de subir una imagen desde el móvil, el trabajo de este archivo es verificar si desde el móvil ha sido enviado una imagen, si es así verifica que solamente sea de la extensión JPG o JPEG

```
if(isset($_POST["submit"])){

    $check = getimagesize($_FILES["file"]["tmp_name"]);

    if($check !== false){

        echo "Archivo es una imagen - " . $check["mime"];
        $uploadOk = 1;

    }else{

        echo "El archivo no es una imagen";
        $uploadOk = 0;

    }

}

//VERIFICAR SI EL ARCHIVO YA EXISTE(OPCIONAL)
if(file_exists($target_file)){

    echo "Lo siento, el archivo ya existe";
    $uploadOk = 0;

}

//PERMITIR UNICAMENTE LAS EXTENSIONES ... (OBLIGATORIO)
if($imageFileType != "jpg" && $imageFileType != "jpeg"){

    echo "Lo siento, solo se permiten JPG o JPEG";
    $uploadOk = 0;

}

//OTRAS VALIDACIONES.. (TAMAÑO)
//FINALMENTE ANALIZAMOS LA VARIABLE BANDER
if($uploadOk == 0){

    echo "Lo siento, el archivo no se puede subir";
}else{

    if(move_uploaded_file($_FILES["file"]["tmp_name"], $target_file)){

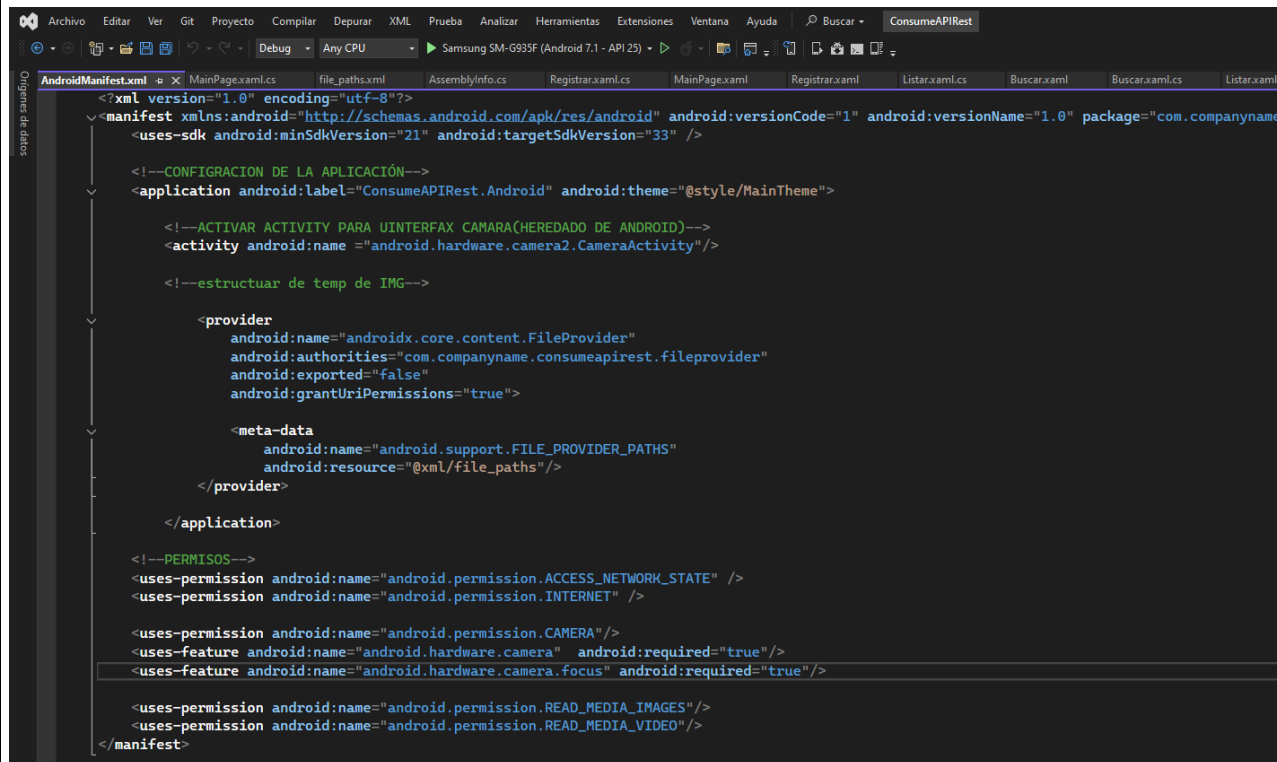
        echo "El archivo " . basename($_FILES["file"]["name"]) . "Has subido una foto";
    }else {

        echo "no hay";

    }

}
```


También incluimos en el Main Page dos botones que no servirá para poder realizara nuestras tareas, previo a esto debimos otortgar permisos en el manifest y crearun xml que nos ayude en lo que es la interacción con la aplicación



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="1" android:versionName="1.0" package="com.companyname">
    <uses-sdk android:minSdkVersion="21" android:targetSdkVersion="33" />

    <!--CONFIGURACION DE LA APLICACIÓN-->
    <application android:label="ConsumeAPIRest.Android" android:theme="@style/MainTheme">

        <!--ACTIVAR ACTIVITY PARA UINTERFAX CAMARA(HEREDADO DE ANDROID)-->
        <activity android:name="android.hardware.camera2.CameraActivity"/>

        <!--estructuar de temp de IMG-->

        <provider
            android:name="androidx.core.content.FileProvider"
            android:authorities="com.companyname.consumeapiREST.fileprovider"
            android:exported="false"
            android:grantUriPermissions="true">

            <meta-data
                android:name="android.support.FILE_PROVIDER_PATHS"
                android:resource="@xml/file_paths"/>
        </provider>

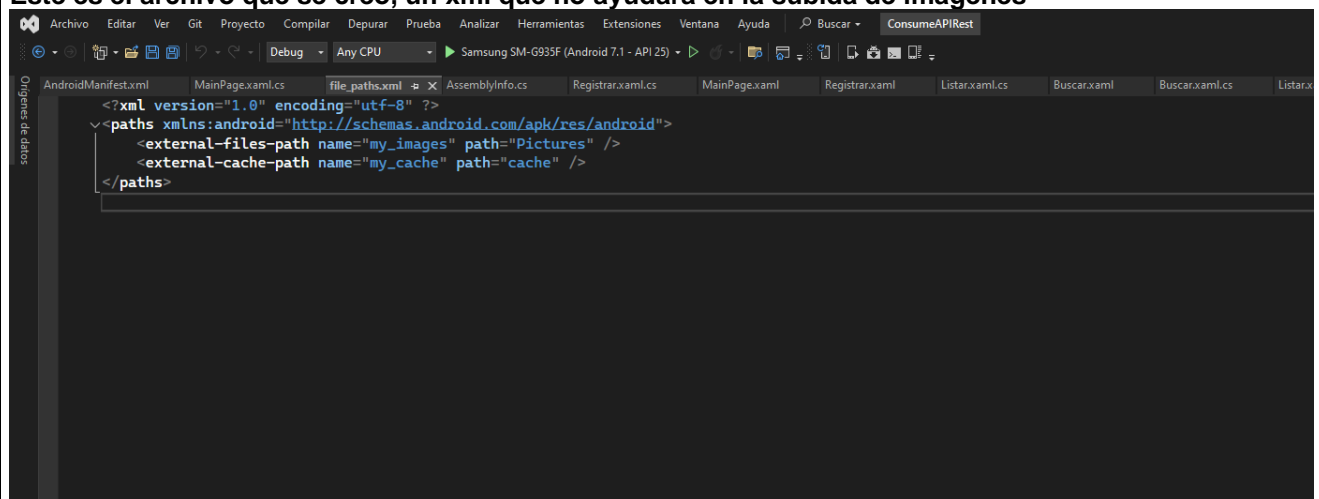
    </application>

    <!--PERMISOS-->
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.INTERNET" />

    <uses-permission android:name="android.permission.CAMERA"/>
    <uses-feature android:name="android.hardware.camera" android:required="true"/>
    <uses-feature android:name="android.hardware.camera.focus" android:required="true"/>

    <uses-permission android:name="android.permission.READ_MEDIA_IMAGES"/>
    <uses-permission android:name="android.permission.READ_MEDIA_VIDEO"/>
</manifest>
```

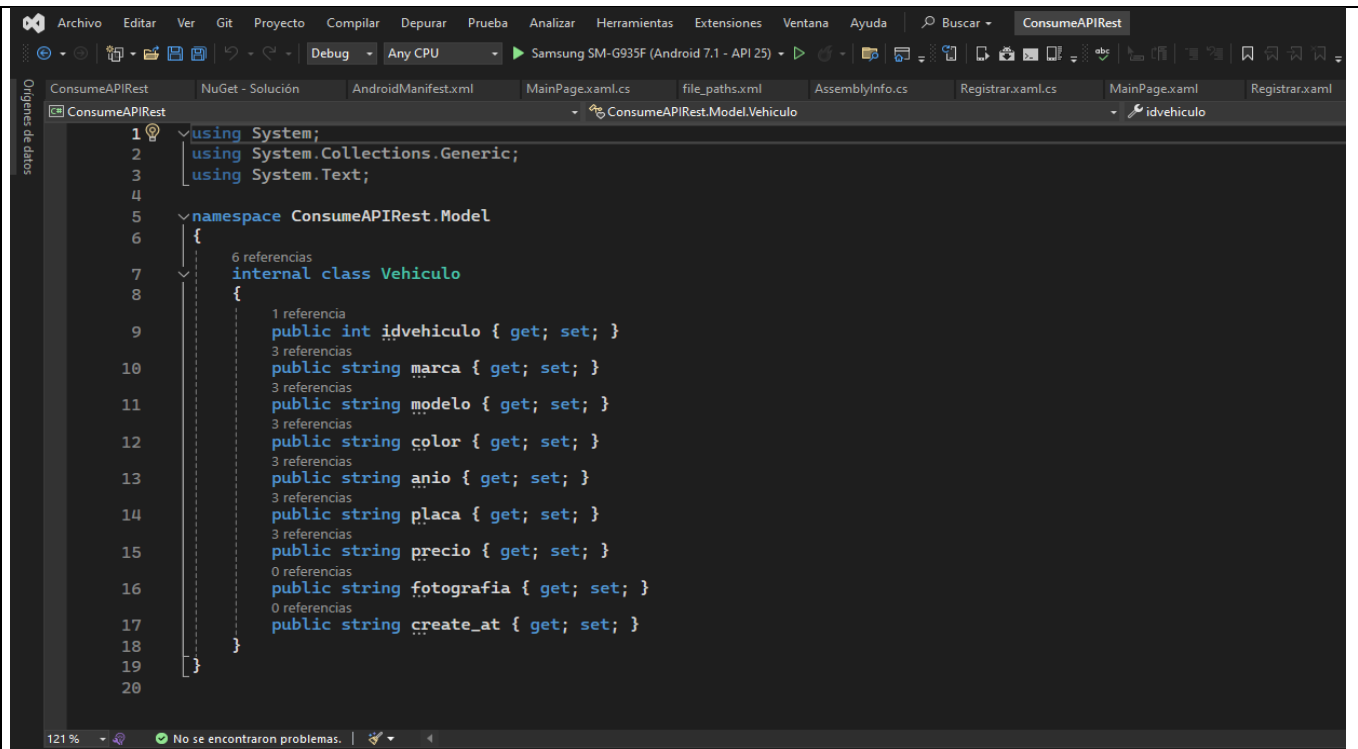
Este es el archivo que se creo, un xml que no ayudara en la subida de imágenes



```
<?xml version="1.0" encoding="utf-8" ?>
<paths xmlns:android="http://schemas.android.com/apk/res/android">
    <external-files-path name="my_images" path="Pictures" />
    <external-cache-path name="my_cache" path="cache" />
</paths>
```

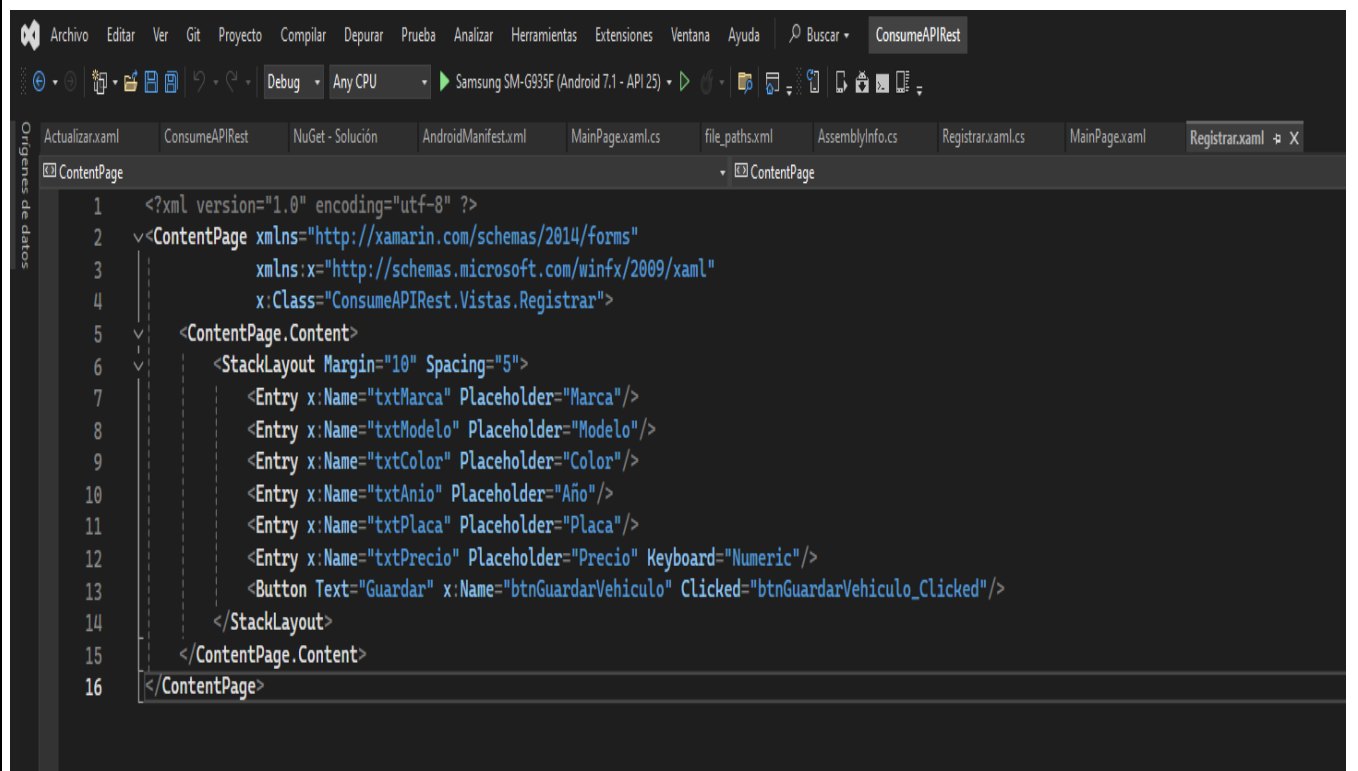
Y todo esto fue como complemento de los conceptos anteriormente vistos CRUDS (Create Read Update Delete Search).

En esta clase se comparten crean los atributos que sirvan para poder manipularlos en la base de datos



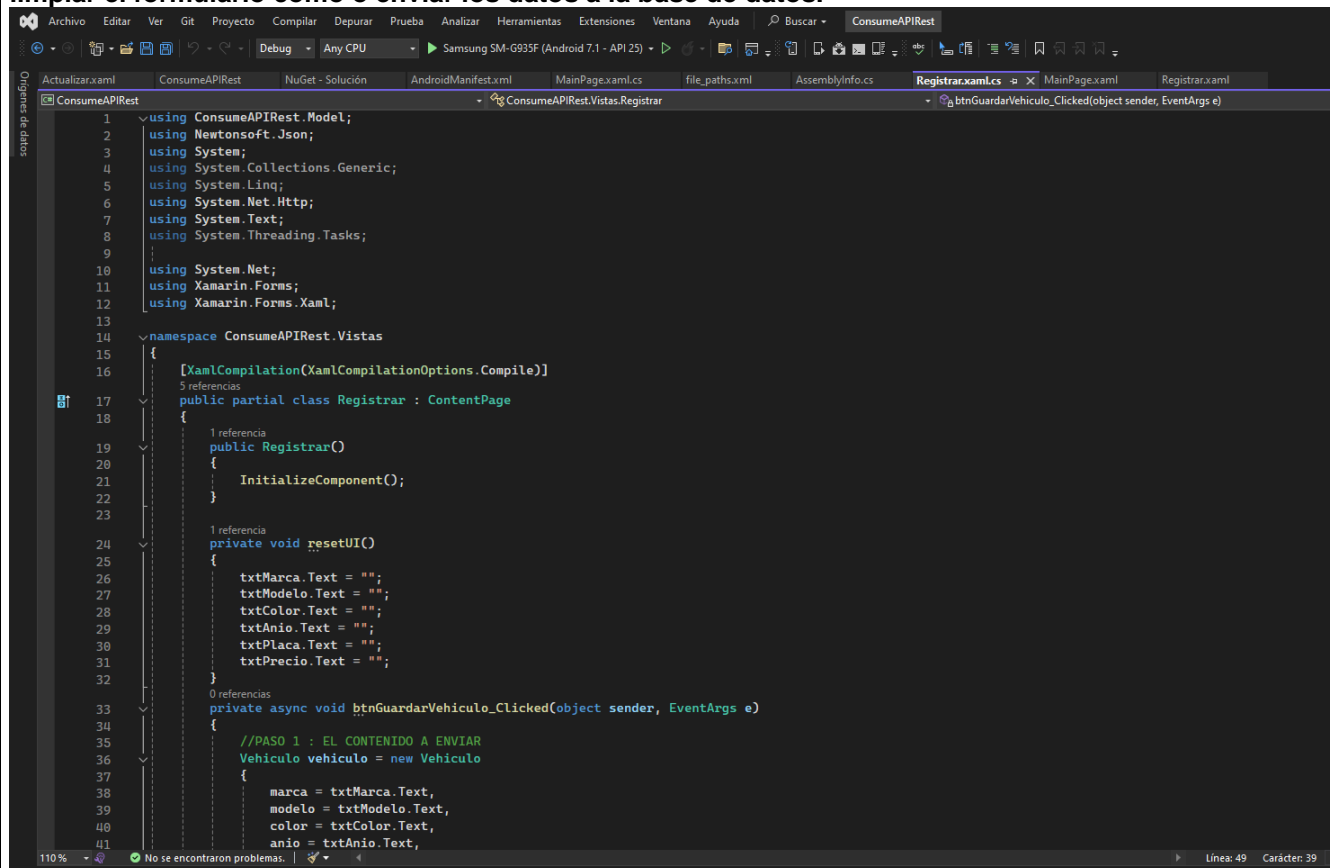
```
1 using System;
2 using System.Collections.Generic;
3 using System.Text;
4
5 namespace ConsumeAPIRest.Model
6 {
7     6 referencias
8     internal class Vehiculo
9     {
10         1 referencia
11         public int idvehiculo { get; set; }
12         3 referencias
13         public string marca { get; set; }
14         3 referencias
15         public string modelo { get; set; }
16         3 referencias
17         public string color { get; set; }
18         3 referencias
19         public string anio { get; set; }
20         3 referencias
21         public string placa { get; set; }
22         3 referencias
23         public string precio { get; set; }
24         0 referencias
25         public string fotografia { get; set; }
26         0 referencias
27         public string create_at { get; set; }
28     }
29 }
```

Creamos sus vistas, cada una con una respectiva tarea como listar, modificar, registrar, eliminar(un crud básico)

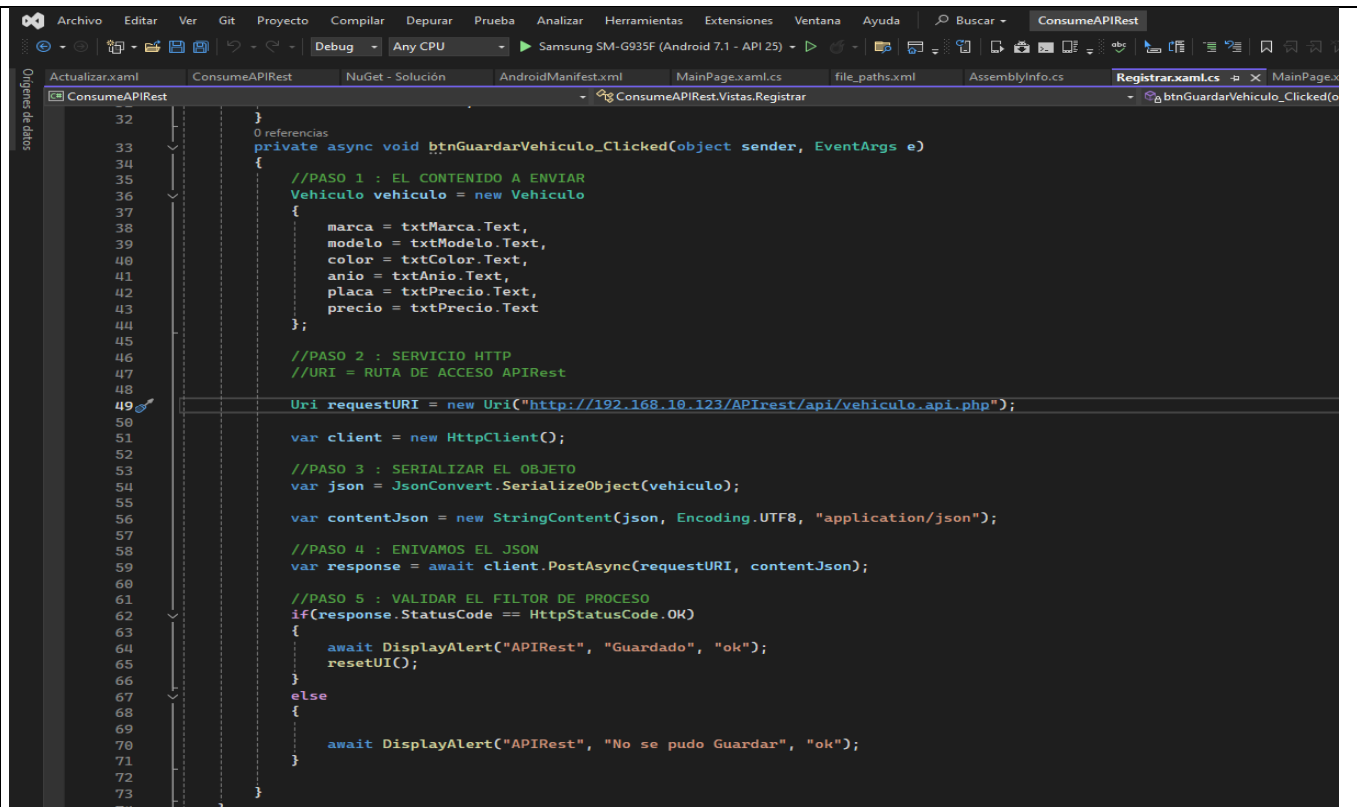


```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4             x:Class="ConsumeAPIRest.Vistas.Registrar">
5     <ContentPage.Content>
6         <StackLayout Margin="10" Spacing="5">
7             <Entry x:Name="txtMarca" Placeholder="Marca"/>
8             <Entry x:Name="txtModelo" Placeholder="Modelo"/>
9             <Entry x:Name="txtColor" Placeholder="Color"/>
10            <Entry x:Name="txtAnio" Placeholder="Año"/>
11            <Entry x:Name="txtPlaca" Placeholder="Placa"/>
12            <Entry x:Name="txtPrecio" Placeholder="Precio" Keyboard="Numeric"/>
13            <Button Text="Guardar" x:Name="btnGuardarVehiculo" Clicked="btnGuardarVehiculo_Clicked"/>
14        </StackLayout>
15    </ContentPage.Content>
16 </ContentPage>
```

En esta parte creamos los métodos necesarios para el registro de un nuevo vehículo, tales como para limpiar el formulario como o enviar los datos a la base de datos.

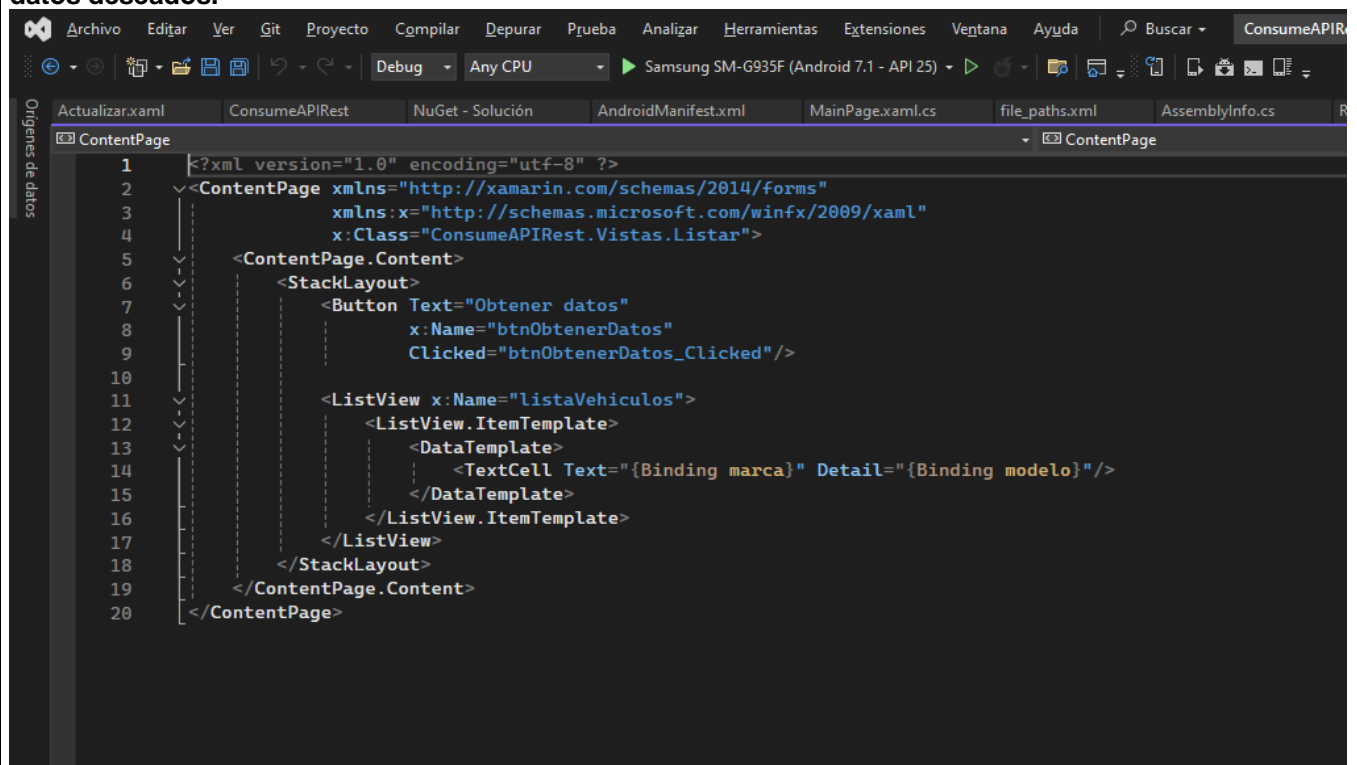


```
1 using ConsumeAPIRest.Model;
2 using Newtonsoft.Json;
3 using System;
4 using System.Collections.Generic;
5 using System.Linq;
6 using System.Net.Http;
7 using System.Text;
8 using System.Threading.Tasks;
9
10 using System.Net;
11 using Xamarin.Forms;
12 using Xamarin.Forms.Xaml;
13
14 namespace ConsumeAPIRest.Vistas
15 {
16     [XamlCompilation(XamlCompilationOptions.Compile)]
17     public partial class Registrar : ContentPage
18     {
19         public Registrar()
20         {
21             InitializeComponent();
22         }
23
24         private void resetUI()
25         {
26             txtMarca.Text = "";
27             txtModelo.Text = "";
28             txtColor.Text = "";
29             txtAnio.Text = "";
30             txtPlaca.Text = "";
31             txtPrecio.Text = "";
32         }
33
34         private async void btnGuardarVehiculo_Clicked(object sender, EventArgs e)
35         {
36             //PASO 1 : EL CONTENIDO A ENVIAR
37             Vehiculo vehiculo = new Vehiculo
38             {
39                 marca = txtMarca.Text,
40                 modelo = txtModelo.Text,
41                 color = txtColor.Text,
42                 anio = txtAnio.Text,
```

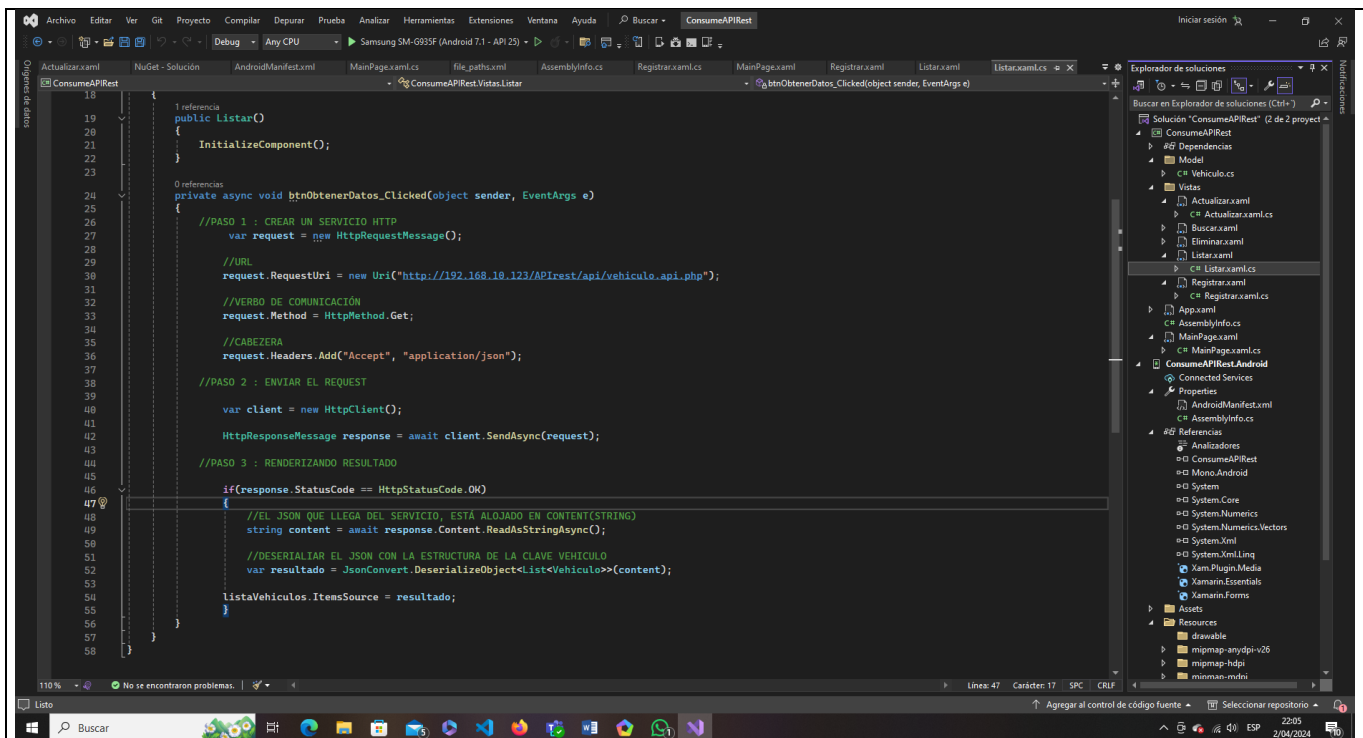


```
32 }
33 0 referencias
34 private async void btnGuardarVehiculo_Clicked(object sender, EventArgs e)
35 {
36     //PASO 1 : EL CONTENIDO A ENVIAR
37     Vehiculo vehiculo = new Vehiculo
38     {
39         marca = txtMarca.Text,
40         modelo = txtModelo.Text,
41         color = txtColor.Text,
42         anio = txtAnio.Text,
43         placa = txtPrecio.Text,
44         precio = txtPrecio.Text
45     };
46     //PASO 2 : SERVICIO HTTP
47     //URI = RUTA DE ACCESO APIRest
48
49     Uri requestURI = new Uri("http://192.168.10.123/APIrest/api/vehiculo.api.php");
50
51     var client = new HttpClient();
52
53     //PASO 3 : SERIALIZAR EL OBJETO
54     var json = JsonConvert.SerializeObject(vehiculo);
55
56     var contentJson = new StringContent(json, Encoding.UTF8, "application/json");
57
58     //PASO 4 : ENIVAMOS EL JSON
59     var response = await client.PostAsync(requestURI, contentJson);
60
61     //PASO 5 : VALIDAR EL FILTRO DE PROCESO
62     if(response.StatusCode == HttpStatusCode.OK)
63     {
64         await DisplayAlert("APIRest", "Guardado", "ok");
65         resetUI();
66     }
67     else
68     {
69
70         await DisplayAlert("APIRest", "No se pudo Guardar", "ok");
71     }
72 }
73
74 }
```

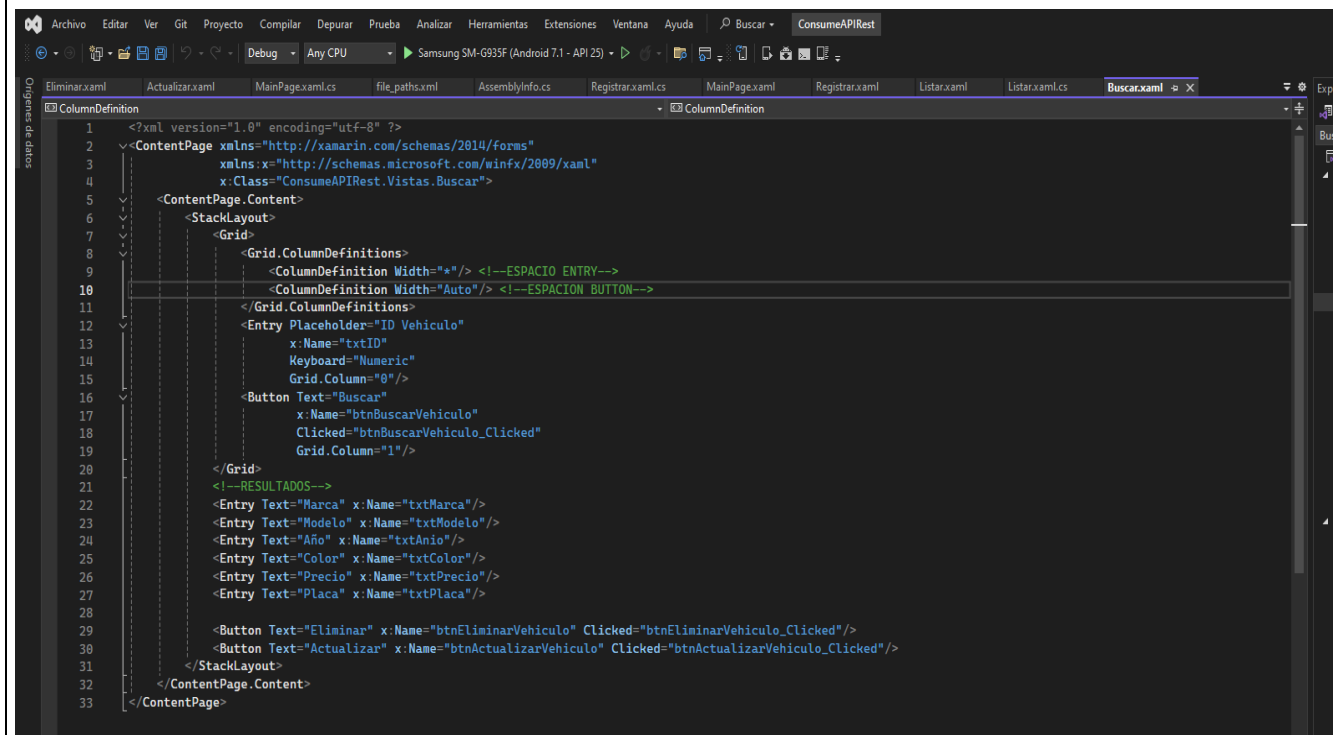
Para listar los elementos usamos la etiqueta Listviews con el atributo Binding para poder renderizar los datos deseados.



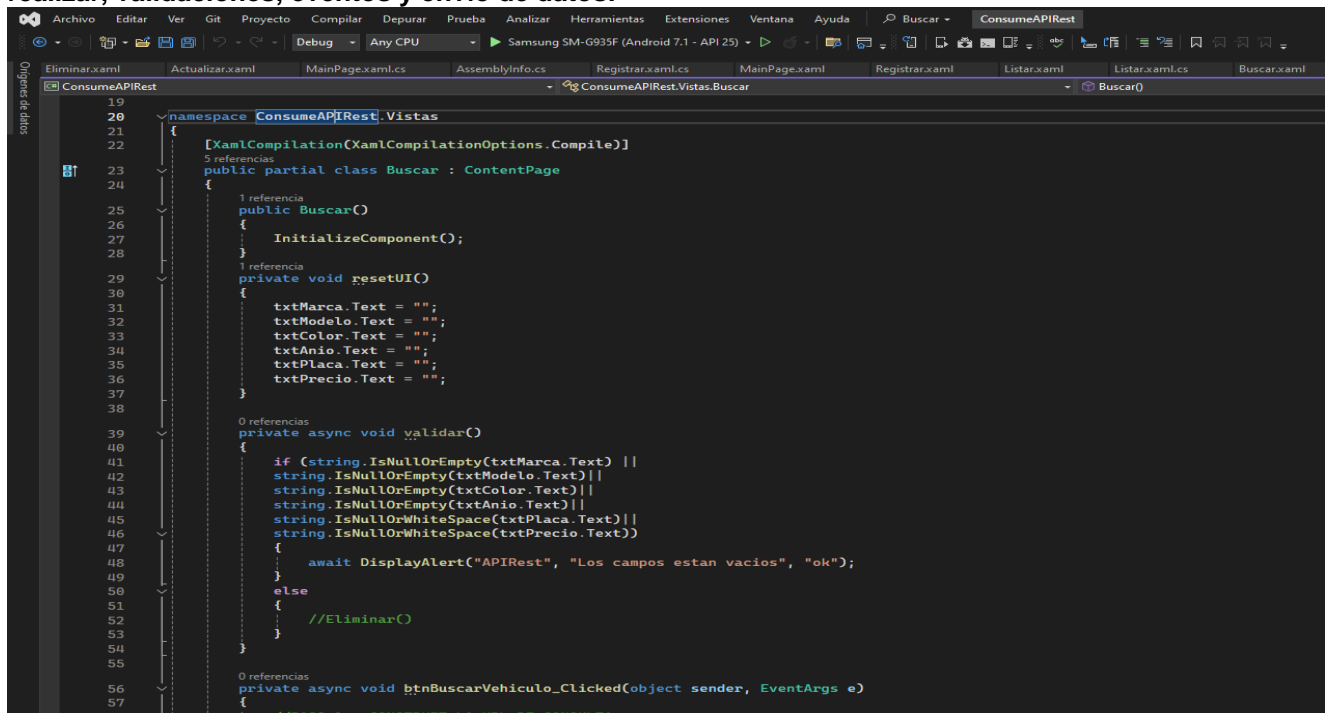
```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4             x:Class="ConsumeAPIRest.Vistas.Listar">
5     <ContentPage.Content>
6         <StackLayout>
7             <Button Text="Obtener datos"
8                   x:Name="btnObtenerDatos"
9                   Clicked="btnObtenerDatos_Clicked"/>
10
11             <ListView x:Name="listaVehiculos">
12                 <ListView.ItemTemplate>
13                     <DataTemplate>
14                         <TextCell Text="{Binding marca}" Detail="{Binding modelo}"/>
15                     </DataTemplate>
16                 </ListView.ItemTemplate>
17             </ListView>
18         </StackLayout>
19     </ContentPage.Content>
20 </ContentPage>
```



Luego a forma de hacerlo más compacto y didáctico, el formulario de búsqueda, de eliminación y actualización se realizó en un mismo formulario.

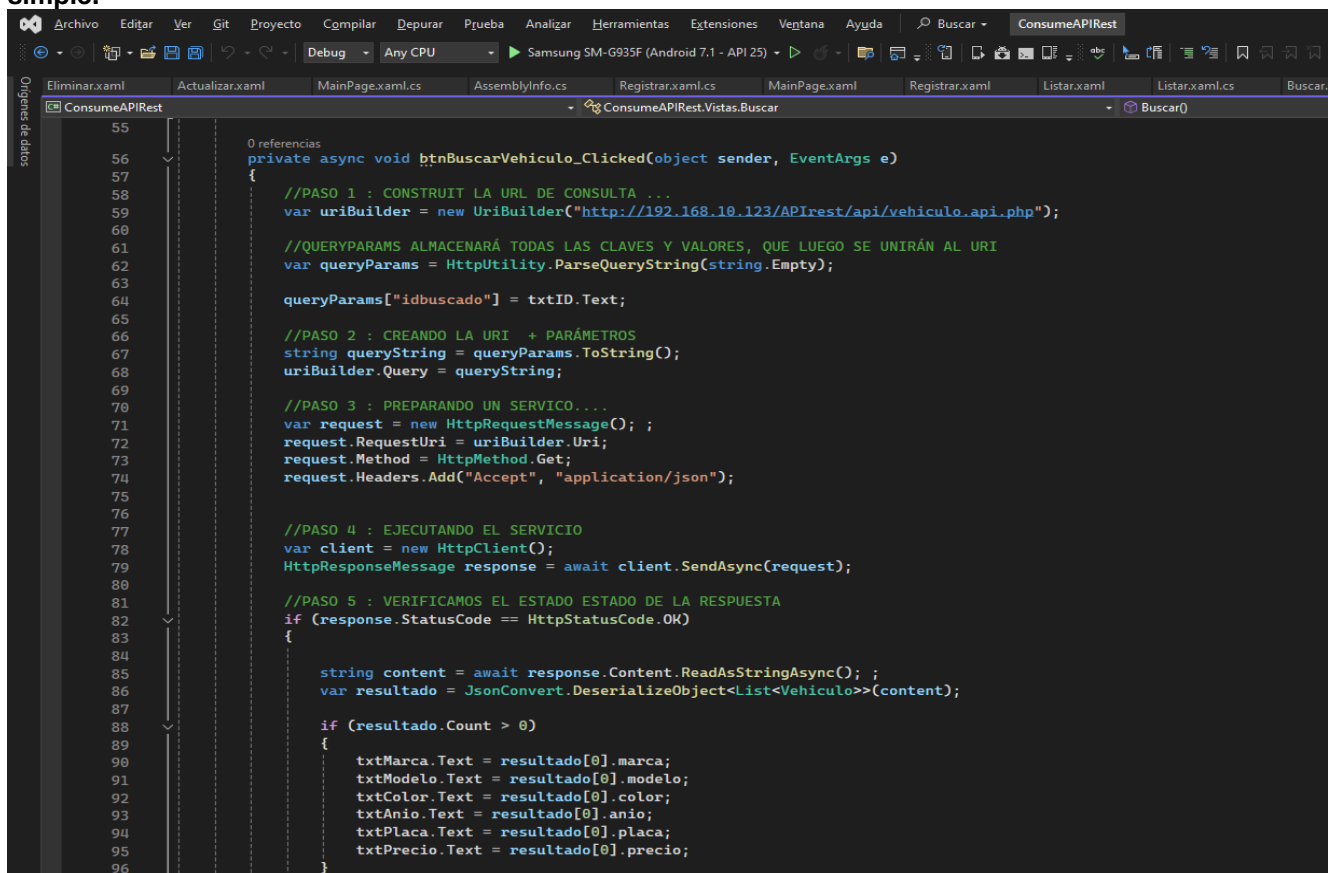


Al realizar tantas operaciones en un mismo view, eso extiende bastante las operaciones que se deben de realizar, validaciones, eventos y envío de datos.



```
19 namespace ConsumeAPIRest.Vistas
20 {
21     [XamlCompilation(XamlCompilationOptions.Compile)]
22     public partial class Buscar : ContentPage
23     {
24         1 referencia
25         public Buscar()
26         {
27             InitializeComponent();
28         }
29         1 referencia
30         private void ResetUI()
31         {
32             txtMarca.Text = "";
33             txtModelo.Text = "";
34             txtColor.Text = "";
35             txtAnio.Text = "";
36             txtPlaca.Text = "";
37             txtPrecio.Text = "";
38         }
39         0 referencias
40         private async void Validar()
41         {
42             if (string.IsNullOrEmpty(txtMarca.Text) ||
43                 string.IsNullOrEmpty(txtModelo.Text) ||
44                 string.IsNullOrEmpty(txtColor.Text) ||
45                 string.IsNullOrEmpty(txtAnio.Text) ||
46                 string.IsNullOrEmpty(txtPlaca.Text) ||
47                 string.IsNullOrEmpty(txtPrecio.Text))
48             {
49                 await DisplayAlert("APIRest", "Los campos estan vacios", "ok");
50             }
51             else
52             {
53                 //Eliminar()
54             }
55         }
56         0 referencias
57         private async void btnBuscarVehiculo_Clicked(object sender, EventArgs e)
58         {
59             //PASO 1 : CONSTRUIR LA URL DE CONSULTA
```

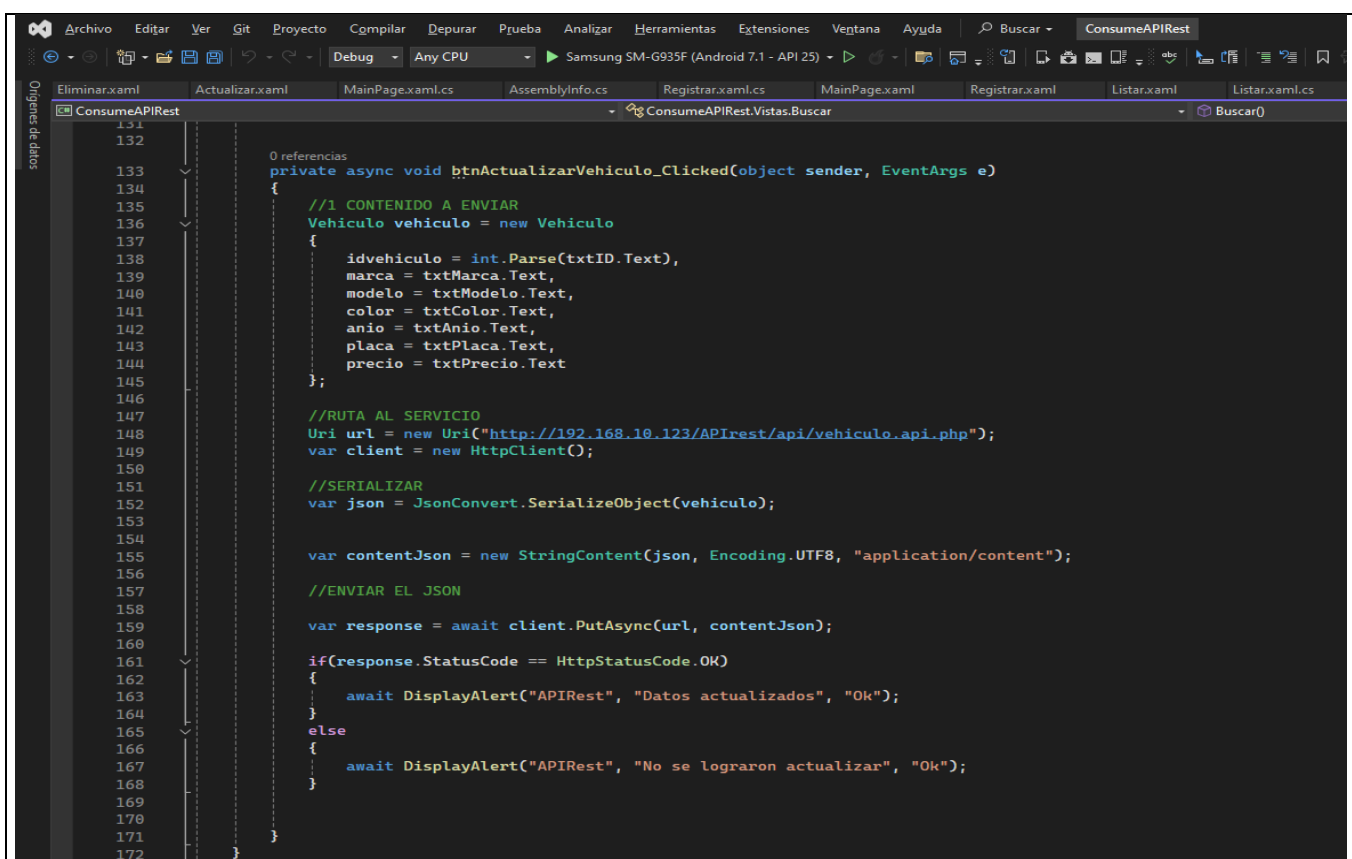
Al realizar una búsqueda, ingresamos el id, de esa forma podemos traer los datos de una manera muy simple.



```
55
56
57     0 referencias
58     private async void btnBuscarVehiculo_Clicked(object sender, EventArgs e)
59     {
60         //PASO 1 : CONSTRUIR LA URL DE CONSULTA ...
61         var uriBuilder = new UriBuilder("http://192.168.10.123/APIrest/api/vehiculo.api.php");
62
63         //QUERYPARAMS ALMACENARÁ TODAS LAS CLAVES Y VALORES, QUE LUEGO SE UNIRÁN AL URI
64         var queryParams = HttpUtility.ParseQueryString(string.Empty);
65
66         queryParams["idbuscado"] = txtID.Text;
67
68         //PASO 2 : CREANDO LA URI + PARÁMETROS
69         string queryString = queryParams.ToString();
70         uriBuilder.Query = queryString;
71
72         //PASO 3 : PREPARANDO UN SERVICIO....
73         var request = new HttpRequestMessage();
74         request.RequestUri = uriBuilder.Uri;
75         request.Method = HttpMethod.Get;
76         request.Headers.Add("Accept", "application/json");
77
78         //PASO 4 : EJECUTANDO EL SERVICIO
79         var client = new HttpClient();
80         HttpResponseMessage response = await client.SendAsync(request);
81
82         //PASO 5 : VERIFICAMOS EL ESTADO ESTADO DE LA RESPUESTA
83         if (response.StatusCode == HttpStatusCode.OK)
84         {
85             string content = await response.Content.ReadAsStringAsync();
86             var resultado = JsonConvert.DeserializeObject<List<Vehiculo>>(content);
87
88             if (resultado.Count > 0)
89             {
90                 txtMarca.Text = resultado[0].marca;
91                 txtModelo.Text = resultado[0].modelo;
92                 txtColor.Text = resultado[0].color;
93                 txtAnio.Text = resultado[0].anio;
94                 txtPlaca.Text = resultado[0].placa;
95                 txtPrecio.Text = resultado[0].precio;
96             }
97         }
98     }
99 }
```

```
Archivo Editar Ver Git Proyecto Compilar Depurar Prueba Analizar Herramientas Extensiones Ventana Ayuda Buscar ConsumeAPIRest
Debug Any CPU Samsung SM-G935F (Android 7.1 - API 25)
Eliminar.xaml Actualizar.xaml MainPage.xaml.cs AssemblyInfo.cs Registrar.xaml.cs Registrar.xaml Listar.xaml Listar.xaml.cs Buscar.xaml Buscar.xaml.cs
ConsumeAPIRest ConsumeAPIRest.Vistas.Buscar
81 //PASO 5 : VERIFICAMOS EL ESTADO ESTADO DE LA RESPUESTA
82 if (response.StatusCode == HttpStatusCode.OK)
83 {
84
85     string content = await response.Content.ReadAsStringAsync(); ;
86     var resultado = JsonConvert.DeserializeObject<List<Vehiculo>>(content);
87
88     if (resultado.Count > 0)
89     {
90         txtMarca.Text = resultado[0].marca;
91         txtModelo.Text = resultado[0].modelo;
92         txtColor.Text = resultado[0].color;
93         txtAnio.Text = resultado[0].anio;
94         txtPlaca.Text = resultado[0].placa;
95         txtPrecio.Text = resultado[0].precio;
96     }
97     else
98     {
99         txtMarca.Text = "";
100         txtModelo.Text = "";
101         txtColor.Text = "";
102         txtAnio.Text = "";
103         txtPlaca.Text = "";
104         txtPrecio.Text = "";
105     }
106 }
107
108 0 referencias
109 private async void btnEliminarVehiculo_Clicked(object sender, EventArgs e)
110 {
111
112     var cleint = new HttpClient();
113
114     string id = txtID.Text; ;
115
116     var response = await cleint.DeleteAsync($"http://192.168.10.123/APIrest/api/vehiculo.api.php/{id}");
117
118     //ESTADO DE LA RESPUESTA
119     if(response.StatusCode == HttpStatusCode.OK || response.Content.Headers.ContentLength > 0)
120     {
121         await DisplayAlert("APIRest", "Datos eliminados", "Ok");
122         resetUI();
123     }
124 }
125
126 110% No se encontraron problemas. Línea: 20
```

```
Archivo Editar Ver Git Proyecto Compilar Depurar Prueba Analizar Herramientas Extensiones Ventana Ayuda Buscar ConsumeAPIRest
Debug Any CPU Samsung SM-G935F (Android 7.1 - API 25)
Eliminar.xaml Actualizar.xaml MainPage.xaml.cs AssemblyInfo.cs Registrar.xaml.cs Registrar.xaml Listar.xaml Listar.xaml.cs Buscar.xaml
ConsumeAPIRest ConsumeAPIRest.Vistas.Buscar
108
109 0 referencias
110 private async void btnEliminarVehiculo_Clicked(object sender, EventArgs e)
111 {
112
113     var cleint = new HttpClient();
114
115     string id = txtID.Text; ;
116
117     var response = await cleint.DeleteAsync($"http://192.168.10.123/APIrest/api/vehiculo.api.php/{id}");
118
119     //ESTADO DE LA RESPUESTA
120     if(response.StatusCode == HttpStatusCode.OK || response.Content.Headers.ContentLength > 0)
121     {
122         await DisplayAlert("APIRest", "Datos eliminados", "Ok");
123         resetUI();
124     }
125     else
126     {
127         await DisplayAlert("APIRest", "Datos no eliminados", "Ok");
128     }
129 }
130
131 0 referencias
132 private async void btnActualizarVehiculo_Clicked(object sender, EventArgs e)
133 {
134
135     //1 CONTENIDO A ENVIAR
136     Vehiculo vehiculo = new Vehiculo
137     {
138         idvehiculo = int.Parse(txtID.Text),
139         marca = txtMarca.Text,
140         modelo = txtModelo.Text,
141         color = txtColor.Text,
142         anio = txtAnio.Text,
143         placa = txtPlaca.Text,
144         precio = txtPrecio.Text
145     };
146
147     //RUTA AL SERVICIO
148     Uri url = new Uri("http://192.168.10.123/APIrest/api/vehiculo.api.php");
149     var client = new HttpClient();
150 }
151
152 110% No se encontraron problemas.
```



```
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172

0 referencias
private async void btnActualizarVehiculo_Clicked(object sender, EventArgs e)
{
    //1 CONTENIDO A ENVIAR
    Vehiculo vehiculo = new Vehiculo
    {
        idvehiculo = int.Parse(txtID.Text),
        marca = txtMarca.Text,
        modelo = txtModelo.Text,
        color = txtColor.Text,
        anio = txtAnio.Text,
        placa = txtPlaca.Text,
        precio = txtPrecio.Text
    };

    //RUTA AL SERVICIO
    Uri url = new Uri("http://192.168.10.123/APIrest/api/vehiculo.api.php");
    var client = new HttpClient();

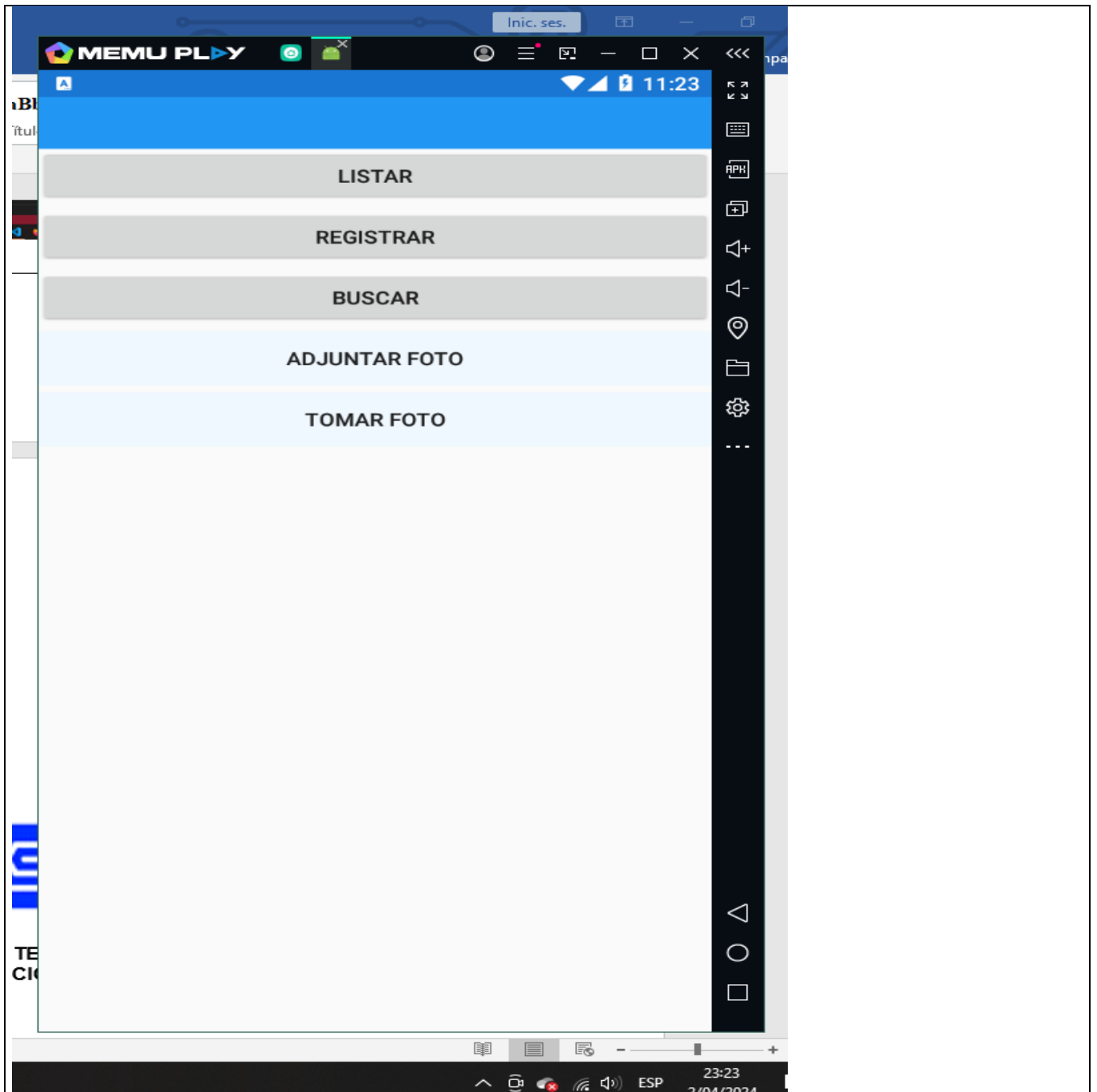
    //SERIALIZAR
    var json = JsonConvert.SerializeObject(vehiculo);

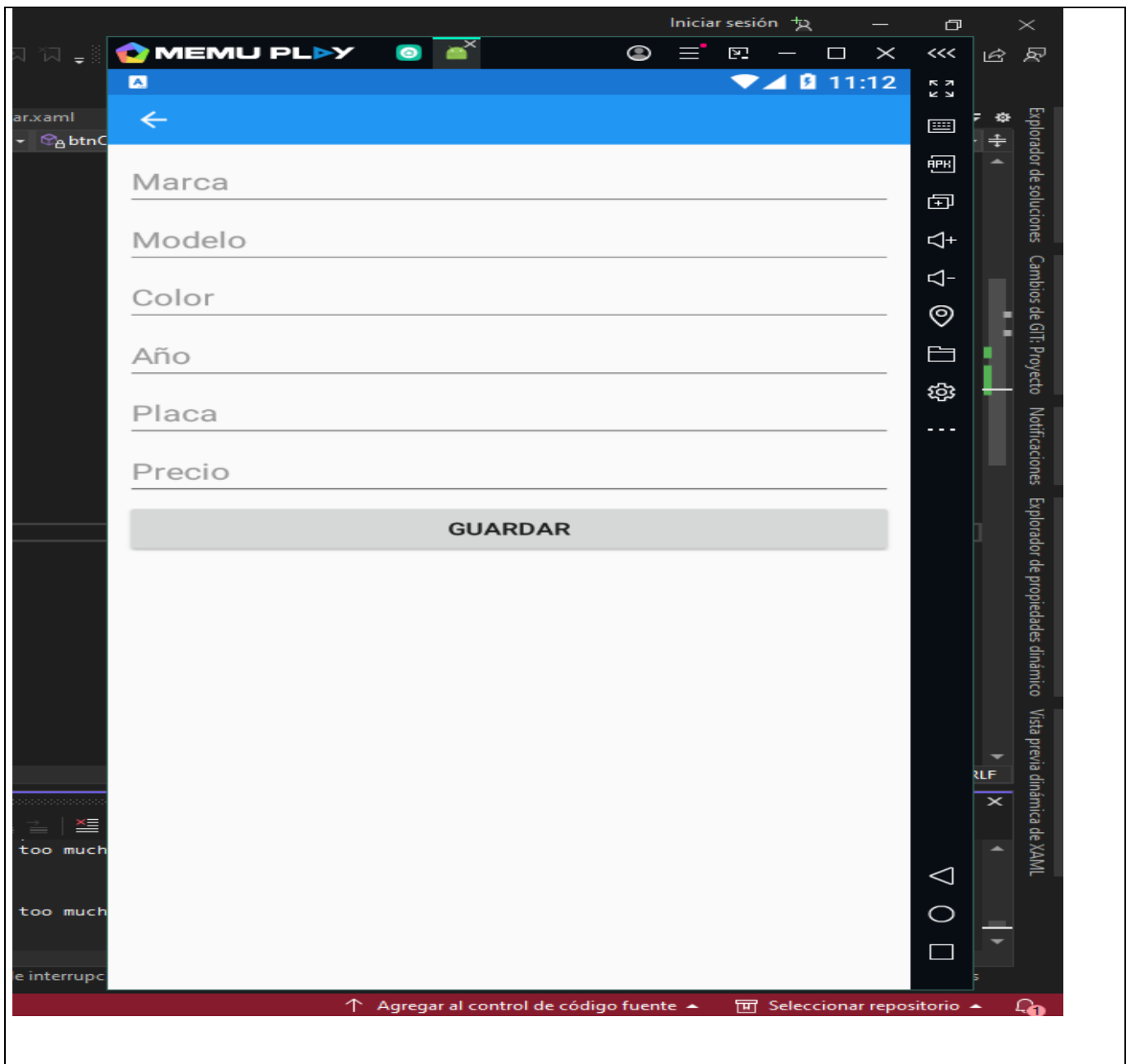
    var contentJson = new StringContent(json, Encoding.UTF8, "application/content");

    //ENVIAR EL JSON
    var response = await client.PutAsync(url, contentJson);

    if(response.StatusCode == HttpStatusCode.OK)
    {
        await DisplayAlert("APIRest", "Datos actualizados", "Ok");
    }
    else
    {
        await DisplayAlert("APIRest", "No se lograron actualizar", "Ok");
    }
}
```

Y así sería ya acabado





The image shows a complex development environment. At the top, a blue header bar contains the text 'tabla' and '¿Qué desea hacer?'. Below this, a row of tabs is visible, labeled 'Código', 'AaBbCc', 'AaBbCc', 'AaBbCc', 'AaBbCc', 'AaBbCc', and 'AaBbCc'. The main area is divided into three sections. The top section, titled 'Estilos', shows a list of styles with a '1' next to the first one. The middle section is a code editor displaying Java code for a vehicle management application. The code includes a 'precio' variable, a 'URL' for the API, a 'client' object, and a 'JSON' object. It also shows a 'contentJson' variable and a 'response' object. The bottom section is a web browser displaying the application's output. The browser shows a form with fields for 'ID Vehículo', 'Marca', 'Modelo', 'Año', 'Color', 'Precio', and 'Placa'. There are 'BUSCAR', 'ELIMINAR', and 'ACTUALIZAR' buttons. The browser also shows a 'SÁBADO' date and a 'M T' status.

EVALUACIÓN DEL INFORME DE TRABAJO SEMANAL	NOTA
--	---

OBSERVACIONES Y RECOMENDACIONES		
DEL INSTRUCTOR:		DEL MONITOR DE EMPRESA:
FIRMA DEL ESTUDIANTE:	FIRMA DE MONITOR DE EMPRESA:	FIRMA DEL INSTRUCTOR:

--	--	--



**PROPIEDAD INTELECTUAL DEL SENATI. PROHIBIDA SU
REPRODUCCIÓN Y VENTA SIN LA AUTORIZACIÓN
CORRESPONDIENTE**