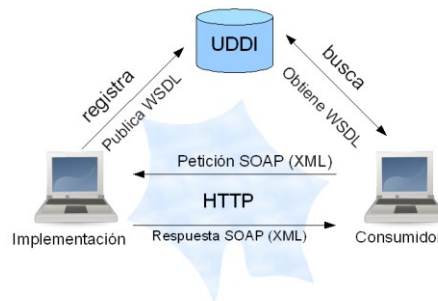
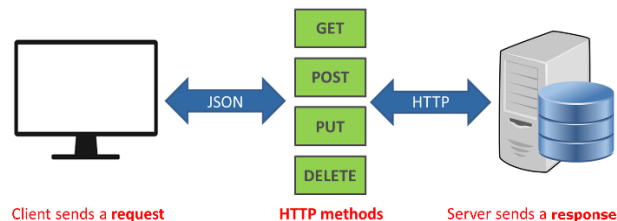


Introducción a los Servicios Web (API)

Servicio Web: es una aplicación diseñada para establecer comunicación con algún otro programa en internet, lo más importante es que no es relevante tecnología que usarán los clientes para consumir la información, es decir, no importa la plataforma, ni el lenguaje de programación del cliente, dado que el proceso establece mediante reglas de comunicación y se establece un formato para el texto que sea estándar, este puede ser una estructura JSON (JavaScript Object Notation), XML, entre otros. Para establecer la comunicación, un servicio web puede usar algún protocolo de comunicación como graphql, REST, SOAP (Simple Object Access Protocol), etc.



API Rest (Application Programming Interface-REpresentational State Transfere): una interfaz máquina-máquina que dos sistemas de computación utilizan para intercambiar información de manera segura a través de Internet. La mayoría de las aplicaciones para empresas que poseen sistemas inter-organizacionales deben comunicarse con otras aplicaciones internas o de terceros para llevar a cabo varias tareas. Cuando esto se presente la tecnología actual que de emplea es mediante de un API.



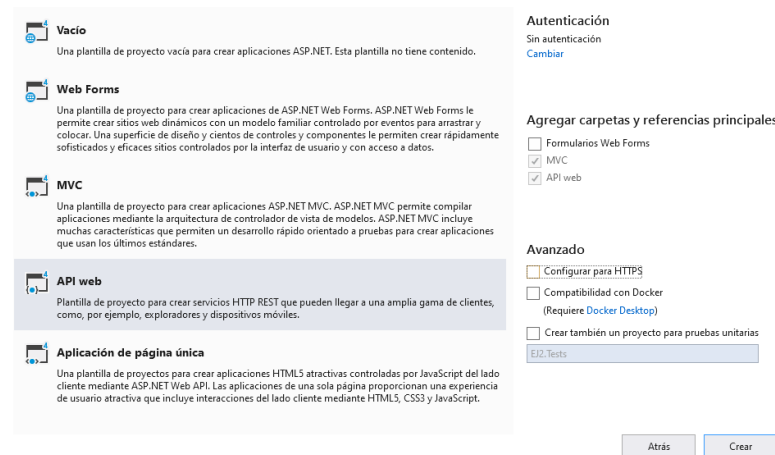
Pasos para crea una API web con Visual Studio

Para crea o consumir un API se puede hacer desde cualquier plataforma y con cualquier lenguaje, en este documento se va a hacer desde visual Studio Framework y C#.

1. Crear proyecto Web API:

- Seleccionar Archivo → Nuevo → Proyecto → Aplicación Web ASP.NET (.Net Framework, C#): definir: Web API y seleccionar: MVC, Web API y no HTTPS. Asignar nombre y ubicación en carpeta deseada → crear

Crear una aplicación web ASP.NET



Vacío
Una plantilla de proyecto vacía para crear aplicaciones ASP.NET. Esta plantilla no tiene contenido.

Web Forms
Una plantilla de proyecto para crear aplicaciones de ASP.NET Web Forms. ASP.NET Web Forms le permite crear sitios web dinámicos con un modelo familiar controlado por eventos para arrastrar y colocar. Una superficie de diseño y cientos de controles y componentes le permiten crear rápidamente sofisticados y eficaces sitios controlados por la interfaz de usuario y con acceso a datos.

MVC
Una plantilla de proyecto para crear aplicaciones ASP.NET MVC. ASP.NET MVC permite compilar aplicaciones mediante la arquitectura de controlador de vista de modelos. ASP.NET MVC incluye muchas características que permiten un desarrollo rápido orientado a pruebas para crear aplicaciones que usan los últimos estándares.

API web
Plantilla de proyecto para crear servicios HTTP REST que pueden llegar a una amplia gama de clientes, como, por ejemplo, exploradores y dispositivos móviles.

Aplicación de página única
Una plantilla de proyectos para crear aplicaciones HTML5 atractivas controladas por JavaScript del lado cliente mediante ASP.NET Web API. Las aplicaciones de una sola página proporcionan una experiencia de usuario atractiva que incluye interacciones del lado cliente mediante HTML5, CSS3 y JavaScript.

Autenticación
Sin autenticación
[Cambiar](#)

Agregar carpetas y referencias principales

- ☐ Formularios Web Forms
- ☒ MVC
- ☒ API web

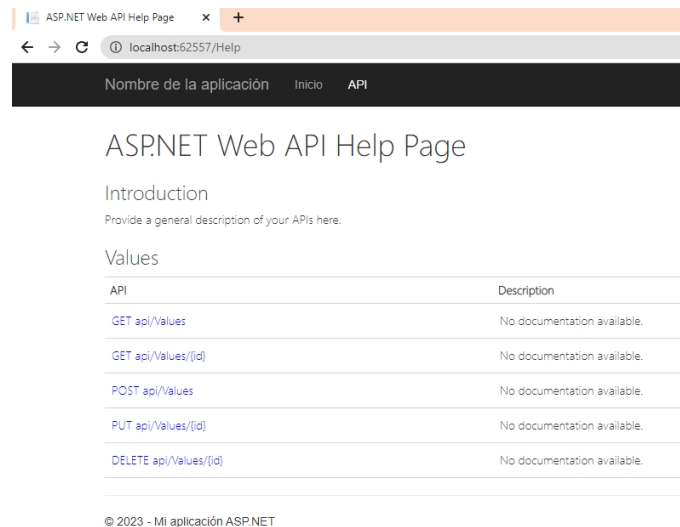
Avanzado

- ☐ Configurar para HTTPS
- ☐ Compatibilidad con Docker (Requiere [Docker Desktop](#))
- ☐ Crear también un proyecto para pruebas unitarias

[EJ2.Tests](#)

[Atrás](#) [Crear](#)

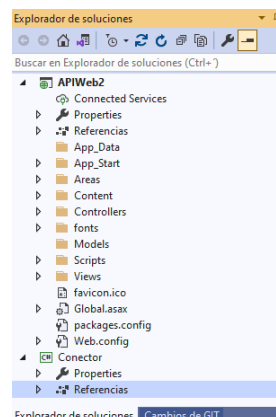
- Probar proyecto:



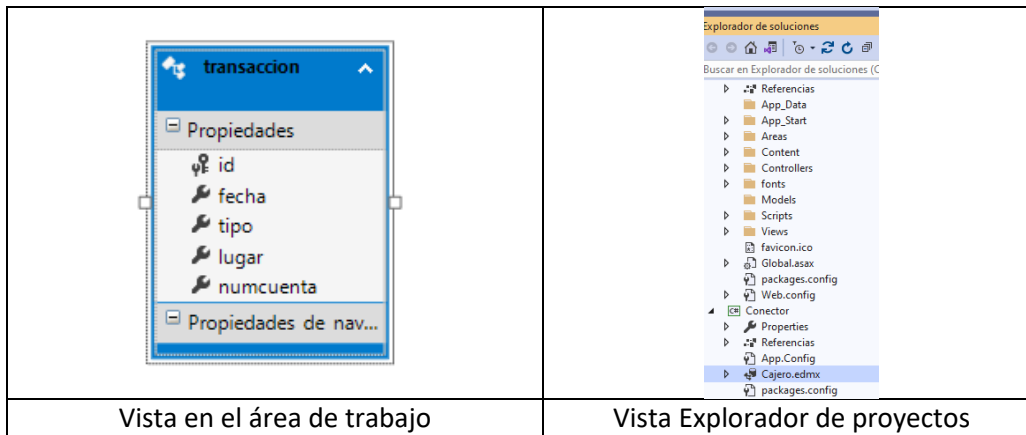
2. Conectar la B de D al API con un proyecto de Entity Framework

- *Crear proyecto de librería de clase de .Net Framework*

Desde la solución del proyecto API se da clic derecho, agregar → Nuevo Proyecto seleccionar Class Library .Net Framework (o Biblioteca de clases .Net Framework) (nombre sugerido Conector). Por defecto se crea la clase Class1.cs, eliminarla no se necesita.



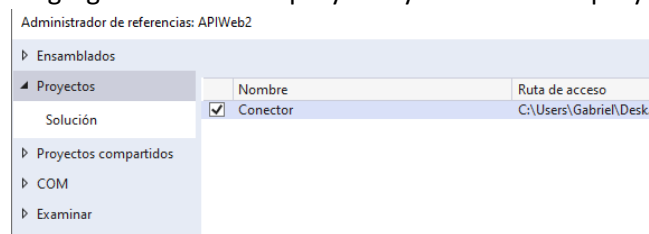
- *Se agrega un nuevo componente (Entity Data Model) al proyecto de librería de clase: se da clic derecho sobre Conector → agregar → nuevo elemento → Elemento C# → Datos y se selecciona DAO .Net Entity Data Model, se asigna nombre (sugerencia nombre base de datos) y se da clic en agregar. Se continua con el asistente de conexión a la B de D (que se ha visto en clase).*



Se compila el proyecto librería de clases con Entity Framework.

3. Relacionar los dos proyectos

- Desde el proyecto de Web API se referencia a al proyecto de Entity Framework: se da clic derecho en referencias y se selecciona agregar referencia → proyecto y seleccionar el proyecto Entity Framework



- Agregar referencia al paquete Entity Framework en el proyecto Web API: se da clic derecho en referencias y se selecciona agregar referencia → Ensamblados → se busca a Entity Framework y se selecciona. Por lo general no está por defecto, entonces hay que agregarla: desde el menú principal se da clic en Herramientas → Administrador de paquetes NuGet → Administrador de paquetes NuGet para solución: se busca y selecciona a Entity Framework y de da instalar.



- Por último, se agrega la cadena de conexión generada en el proyecto Entity Framework a el proyecto Web API: En el archivo App.Config del proyecto Entity Framework se copia la cadena de conexión:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
    <!-- For more information on Entity Framework configuration, visit https://go.microsoft.com/fwlink/?LinkID=287744 -->
    <section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection" baseDirectory="App_Data" />
  </configSections>
  <connectionStrings>
    <add name="cajeroEntities" connectionString="metadata=res://*/CajeroEntities.ssdml;provider=System.Data.SqlClient;providerSettings=;connectionString=Server=.;Initial Catalog=cajero;Integrated Security=True;User=sa;Password=;TrustServerCertificate=True;" />
  </connectionStrings>
  <entityFramework>
    <defaultConnectionFactory type="System.Data.Entity.Infrastructure.SqlConnectionFactory" />
  </entityFramework>
</configuration>
```

Y se copia en el archivo en el archivo web.config, después de cerrar la etiqueta </system web> y antes de <system.webServer>

```

Web.config  Cajero.edmx [Diagram1]  APIWeb2: Introducción
<add key="webpages:Enabled" value="false" />
<add key="ClientValidationEnabled" value="true" />
<add key="UnobtrusiveJavaScriptEnabled" value="true" />
</appSettings>
<system.web>
  <compilation debug="true" targetFramework="4.7.2" />
  <httpRuntime targetFramework="4.7.2" />
</system.web>
<connectionStrings>
  <add name="cajeroEntities" connectionString="metadata=res://*/Ca
</connectionStrings>
<system.webServer>
  <handlers>
    <remove name="ExtensionlessUrlHandler-Integrated-4.0" />

```

Se guarda todo

4. Agregar Controlador para consumir los datos de la B de D

Para realizar las transacciones sobre la conexión de Entity Framework se debe crear un controlador, para lo cual se da clic derecho en la carpeta controllers: Agregar → Controlador → Web API → Controlador Web API 2 con lectura y escritura, se le asigna un nombre y se da clic en agregar. Se genera una nueva clase los métodos Get (consulta general y filtrada), Post (insertar), Put (actualizar) y Delete (eliminar) con las opciones para poder realizar las operaciones básicas sobre la conexión de la B de D, la cual hay que programar.

```

0 referencias
public class TransaccionesController : ApiController
{
    // GET: api/Transacciones
    0 referencias
    public IEnumerable<string> Get()
    {
        return new string[] { "value1", "value2" };
    }

    // GET: api/Transacciones/5
    0 referencias
    public string Get(int id)
    {
        return "value";
    }

    // POST: api/Transacciones
    0 referencias
    public void Post([FromBody]string value)
    {
    }

    // PUT: api/Transacciones/5

```

4.1. Para consulta general: Método Get sin parámetros:

- Lo primero es importar al proyecto Entity Framework: `using Conector;`
Luego, programar el método Get en la clase CuentaController de la aplicación del API, con el siguiente código:

```

public IEnumerable<Cuenta> Get()
{
    using (cajeroEntities objEntidad = new cajeroEntities()) {
        return objEntidad.Cuenta.ToList(); }
}

```

Se prueba el funcionamiento desde la aplicación y desde postman (que se debe instalar).

Observación: Para configurar que solo sea formato Json se adiciona las siguientes líneas en el método Application_Start() de Global y se agregan las líneas:

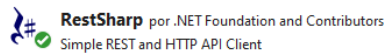
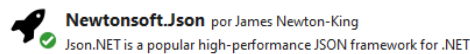
```

GlobalConfiguration.Configuration.Formatters.Clear();
GlobalConfiguration.Configuration.Formatters.Add(new JsonMediaTypeFormatter()); //e importar
paquete

```

- Para Consumir o hacer uso del API con la consulta general y los demás métodos, se puede crear cualquier tipo de aplicación (consola, escritorio, web o móvil) en este caso lo vamos a hacer con una de escritorio. Se diseña el siguiente formulario:

- Luego se deben instalar y referenciar los paquetes: Newtonsoft (para trabajar con formato Json) y RestSharp (para trabajar con API como cliente), se procede como ya se explicó anteriormente desde la opción del menú “Administrador de paquetes NuGet”. En la clase del formulario se importa `using Newtonsoft.Json;`



- Se agrega una nueva clase de propósito general (vamos a llamarla DBApi). En esta clase se van a realizar todos los métodos para conectar al API (algo similar a la clase para conectar a las B de D realizada en clase). Se realizan las siguientes importaciones:

```
using Newtonsoft.Json;
using RestSharp;
using System.IO;
using System.Net;
using System.Web;
using System.Windows.Forms;
```

Y se copia el siguiente código para el método Get en la clase DBApi:

```
public static dynamic Get(string urlApi)
{
    HttpWebRequest objPedido = (HttpWebRequest)WebRequest.Create(urlApi);
    objPedido.UserAgent = "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:23.0) Gecko/20100101 Firefox/23.0";
    //objPedido.CookieContainer = myCookie;
    objPedido.Credentials = CredentialCache.DefaultCredentials;
    objPedido.Proxy = null;
    HttpResponseMessage objRespuesta = (HttpResponseMessage)objPedido.GetResponse();
    Stream objFlujo = objRespuesta.GetResponseStream();
    StreamReader objFlujoLectura = new StreamReader(objFlujo);
    //Leemos los datos
    string datosJson = HttpUtility.HtmlDecode(objFlujoLectura.ReadToEnd());
    dynamic datos = JsonConvert.DeserializeObject(datosJson);
    return datos;
}
```

Y se programa el botón de Consulta General con el código (ojo hay que cambiar la URL por la del API) y se prueba:

```
dynamic respuesta = DBApi.Get("http://localhost:50615/Api/Cuenta");
txSalida.Text = respuesta.ToString();
```

4.2. Para consulta general: Método Get con parámetros:

4.3. Para consulta general: Método Post:

- En la clase CuentaController de la aplicación del API es programar el método Post, con el siguiente código:

```
public HttpResponseMessage Post([FromBody] cuenta objCuenta)
{
    int resp = 0;
    HttpResponseMessage objMenRespuesta = null;
    try {
        using (cajeroEntities objEntidad = new cajeroEntities())
        {
            objEntidad.Entry(objCuenta).State = EntityState.Added;
            resp = objEntidad.SaveChanges();
            objMenRespuesta = Request.CreateResponse(HttpStatusCode.OK, resp);
        }
    } catch (Exception er) {
        objMenRespuesta = Request.CreateErrorResponse(HttpStatusCode.InternalServerError, er.Message);
    }
    return objMenRespuesta;
}
```

- En la clase DBApi se copia el siguiente código para el método Post:

```
public static dynamic Post(string urlApi, string json, string autorizacion = null)
{
    try {
        var objCliente = new RestClient(urlApi);
        var objPedido = new RestRequest();
        objPedido.Method = Method.Post;
        objPedido.AddHeader("content-type", "application/json");
        objPedido.AddParameter("application/json", json, ParameterType.RequestBody);
        if (autorizacion != null)
        {
            objPedido.AddHeader("Authorization", autorizacion);
        }
        RestResponse objRespuesta = objCliente.Execute(objPedido);
        dynamic datos = JsonConvert.DeserializeObject(objRespuesta.Content);
        return datos;
    }
    catch (Exception ex) {
        MessageBox.Show(ex.Message);
        return null;
    }
}
```

- Y para finalizar en la Clase formulario se programa el botón Insertar Cuenta con el código:

```
int cod = Int32.Parse(txCod.Text);
string nom = txNom.Text;
double saldo = Double.Parse(txSaldo.Text);
int clave = Int32.Parse(txClave.Text);
int codsuc = Int32.Parse(txCodSuc.Text);
Cuenta objCuenta = new Cuenta (cod, nom, saldo, clave, codsuc);
string json = JsonConvert.SerializeObject(objCuenta);
dynamic respuesta = DBApi.Post("http://localhost:50615/Api/Cuenta", json);
if (respuesta == 1) {
    MessageBox.Show("La creación de la cuanta fue exitosa");
} else {
    MessageBox.Show("Fallo la creación de la cuanta, revise la información");
}
```

Reorganizado el código las clases quedan de la siguiente forma:

- la clase CuentaController que de la siguiente forma;

```
namespace EjemploServicios.Controllers
{
    public class CuentaController : ApiController
    {
        // GET: api/Cuenta
        public IEnumerable<cuenta> Get()
        {
            using (cajeroEntities objEntidad = new cajeroEntities())
            {
                return objEntidad.cuenta.ToList();
            }
        }
        // GET: api/Cuenta/5
        public string Get(int id)
        {
            return "value";
        }
        // POST: api/Cuenta
        public HttpResponseMessage Post([FromBody] cuenta objCuenta)
        {
            EntityState operacion = EntityState.Added;
            return operacion(objCuenta, operacion);
        }
        // PUT: api/Cuenta/5
        public HttpResponseMessage Put([FromBody] cuenta objCuenta)
        {
            EntityState operacion = EntityState.Modified;
            return operacion(objCuenta, operacion);
        }
        // DELETE: api/Cuenta/5
        public HttpResponseMessage Delete([FromBody] cuenta objCuenta)
        {
            EntityState operacion = EntityState.Deleted;
            return operacion(objCuenta, operacion);
        }
        private HttpResponseMessage operacion([FromBody] cuenta objCuenta, EntityState operacion)
        {
            int resp = 0;
            HttpResponseMessage objMenRespuesta = null;
            try
            {
                using (cajeroEntities objEntidad = new cajeroEntities())
                {
                    objEntidad.Entry(objCuenta).State = operacion;
                    resp = objEntidad.SaveChanges();
                    objMenRespuesta = Request.CreateResponse(HttpStatusCode.OK, resp);
                }
            } catch (Exception er)
            {
                objMenRespuesta = Request.CreateErrorResponse(HttpStatusCode.InternalServerError, er.Message);
            }
            return objMenRespuesta;
        }
    }
}
```

- La clase DBApi es:

```
class DBApi
{
    public static dynamic Get(string urlApi)
    {

```

```

        HttpRequest objPedido = (HttpRequest)WebRequest.Create(urlApi);
        objPedido.UserAgent = "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:23.0) Gecko/20100101
Firefox/23.0";
        //objPedido.CookieContainer = myCookie;
        objPedido.Credentials = CredentialCache.DefaultCredentials;
        objPedido.Proxy = null;
        HttpResponse objRespuesta = (HttpResponse)objPedido.GetResponse();
        Stream objFlujo = objRespuesta.GetResponseStream();
        StreamReader objFlujoLectura = new StreamReader(objFlujo);
        //Leemos los datos
        string datosJson = HttpUtility.HtmlDecode(objFlujoLectura.ReadToEnd());
        dynamic datos = JsonConvert.DeserializeObject(datosJson);
        return datos;
    }
    public static dynamic Post(string urlApi, string json, string autorizacion = null)
    {
        Method op = Method.Post;
        return operaciones(urlApi, json, autorizacion = null, op);
    }
    public static dynamic Put(string urlApi, string json, string autorizacion = null)
    {
        Method op = Method.Put;
        return operaciones(urlApi, json, autorizacion = null, op);
    }
    public static dynamic Delete(string urlApi, string json, string autorizacion = null)
    {
        Method op = Method.Delete;
        return operaciones(urlApi, json, autorizacion = null, op);
    }
    private static dynamic operaciones(string urlApi, string json, string autorizacion,
Method op)
    {
        try
        {
            var objCliente = new RestClient(urlApi);
            var objPedido = new RestRequest();
            objPedido.Method = op;
            objPedido.AddHeader("content-type", "application/json");
            objPedido.AddParameter("application/json", json, ParameterType.RequestBody);
            if (autorizacion != null)
            {
                objPedido.AddHeader("Authorization", autorizacion);
            }
            RestResponse objRespuesta = objCliente.Execute(objPedido);
            dynamic datos = JsonConvert.DeserializeObject(objRespuesta.Content);
            return datos;
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            return null;
        }
    }
}

```

La clase del formulario:

```

public partial class Ppal : Form
{
    public Ppal()
    {
        InitializeComponent();
    }
    private void btMostrarD_Click(object sender, EventArgs e)
    { //Consulta por posicion, hay que pedir cual posion busca
        dynamic respuesta = DBApi.Get("http://localhost:50615/Api/Cuenta");
    }
}

```



```

        txCod.Text = respuesta[1].numero.ToString();
        txNom.Text = respuesta[1].nombre.ToString();
        txSaldo.Text = respuesta[1].saldo.ToString();
    }

    private void btMostrarG_Click(object sender, EventArgs e)
    {
        //Consulta General
        dynamic respuesta = DBApi.Get("http://localhost:50615/Api/Cuenta");
        txSalida.Text = "Las cuentas son:\n"+respuesta.ToString();
    }

    private void btInsertar_Click(object sender, EventArgs e)
    {
        //Insertar
        Cuenta objCuenta = leerCuenta();
        string json = JsonConvert.SerializeObject(objCuenta);
        dynamic respuesta = DBApi.Post("http://localhost:50615/Api/Cuenta", json);
        if (respuesta.ToString() == "1") {
            txSalida.Text="La creacion de la cuanta fue exitosa";
        } else {
            txSalida.Text = "Fallo la creacion de la cuanta, revice la información";
        }
    }

    private void btActulizar_Click(object sender, EventArgs e)
    {
        Cuenta objCuenta = leerCuenta();
        string json = JsonConvert.SerializeObject(objCuenta);
        dynamic respuesta = DBApi.Put("http://localhost:50615/Api/Cuenta", json);
        if (respuesta.ToString() == "1")
        {
            txSalida.Text = "La actualización de la cuenta fue exitosa";
        }
        else
        {
            txSalida.Text = "Fallo la actualización de la cuanta, revice la información";
        }
    }

    private void btEliminar_Click(object sender, EventArgs e)
    {
        int numero = Int32.Parse(txCod.Text);
        Cuenta objCuenta = new Cuenta(numero, "", 0, 0, 0);
        string json = JsonConvert.SerializeObject(objCuenta);
        dynamic respuesta = DBApi.Delete("http://localhost:50615/Api/Cuenta", json);
        if (respuesta.ToString() == "1")
        {
            txSalida.Text = "La eliminación de la cuanta fue exitosa";
        }
        else
        {
            txSalida.Text = "Fallo la eliminación de la cuanta, revice la información";
        }
    }

    private Cuenta leerCuenta()
    {
        int cod = Int32.Parse(txCod.Text);
        string nom = txNom.Text;
        double saldo = Double.Parse(txSaldo.Text);
        int clave = Int32.Parse(txClave.Text);
        int codsuc = Int32.Parse(txCodSuc.Text);
        return new Cuenta(cod, nom, saldo, clave, codsuc);
    }
}

```