**¡¡ Corregir erratas !!**

**CICLO FORMATIVO SUPERIOR D.A.M.**



##### Documentación sobre .NET

**REALIZADA POR**

**David Fernández Jaramillo**

**Juan Gabriel Sánchez Galeano**

**Iván Carrillo Mateo**

**Fco. Javier Díaz Izquierdo**

**ALMENDRALEJO, JUNIO DEL 2017**

**DOCUMENTACION .NET**

Autor:

David Fernández Jaramillo

Juan Gabriel Sánchez Galeano

Iván Carrillo Mateo

Fco. Javier Díaz Izquierdo

Profesor

José María Alonso Rodríguez

ÍNDICE

1 INTRODUCCIÓN 1

1.1 Objetivo 1

1.2 Definiciones 1

1.2.1 .NET 1

1.2.2 C# 2

1.2.3 Visual Studio 3

1.2.4 Windows Forms 4

1.2.5 ASP.NET WebForms 4

1.2.6 ASP.NET 5

1.2.7 ASP.NET MVC 6

2 INSTALACIÓN DE VISUAL STUDIO 7

2.1 Instalación 7

2.1.1 Instalación mediante el instalador web 7

3 APLICACIONES DE ESCRITORIO 51

3.1 Introduccion a Aplicación Escritorio 51

3.1.1 Creación del proyecto 51

3.1.2 Desarrollo de la aplicación 54

# INTRODUCCIÓN

## Objetivo

El **objetivo** de esta documentación es ayudar a futuros alumnos introducirse al **framework** **.NET** de Microsoft.

En este documento contaremos los pasos a seguir para desarrollar distintas aplicaciones que serán necesarias en su formación. Veremos cómo realizar aplicaciones de **Windows Forms** y también de **ASP.NET** mediante el IDE **Visual Studio 2015**. El lenguaje de programación que usaremos será **C#.**

## Definiciones

Antes de comenzar a desarrollar las aplicaciones es conveniente definir y explicar los términos más **importantes** a tener en cuenta:

### .NET

**Microsoft .NET** es una plataforma de desarrollo y ejecución de aplicaciones. Esto quiere decir que no sólo nos brinda todas las **herramientas** y **servicios** que se necesitan para desarrollar modernas **aplicaciones** **empresariales** y de **misión** **crítica**, sino que también nos provee de mecanismos robustos, seguros y eficientes para asegurar que la ejecución de las mismas sea óptima. Los componentes principales de la plataforma **.NET** son:

* Un entorno de ejecución de aplicaciones, también llamado “**Runtime**”, que es un componente de software cuya función es la de **ejecutar** las **aplicaciones** **.NET** e interactuar con el sistema operativo ofreciendo sus servicios y recursos.
* Un conjunto de **bibliotecas** de **funcionalidades** y controles **reutilizables**.
* Un **conjunto** de **lenguajes** de **programación** de **alto** **nivel**, junto con sus compiladores y linkers, que permitirán el desarrollo de aplicaciones sobre la plataforma **.NET**.
* Un conjunto de utilitarios y herramientas de desarrollo para **simplificar** las **tareas** más comunes del proceso de desarrollo de aplicaciones.
* **Documentación** y **guías** **de** **arquitectura**, que describen las mejores prácticas de diseño, organización, desarrollo, prueba e instalación de **aplicaciones .NET**.
* **.NET** **no es un sistema operativo**, como si lo es Microsoft Windows en sus distintas versiones.
* **.NET no es un Lenguaje de Programación:** si bien la plataforma Microsoft **.NET** incluye lenguajes de programación de aplicaciones, su concepto es más amplio y va más allá de éstos.
* **.NET no es un Entorno de Desarrollo:** si bien la plataforma Microsoft **.NET** incluye entornos de desarrollo integrados (IDEs), su concepto es más amplio y va más allá de éstos.
* **.NET no es un servidor de aplicaciones (Application Server).**
* **.NET** no es un **producto** **empaquetado** que se pueda comprar como tal, sino que es una plataforma que engloba distintas aplicaciones, servicios y conceptos y que en conjunto permiten el **desarrollo** y la **ejecución** de **aplicaciones**.

### C#

**C#** es un **lenguaje** **orientado** **a** **objetos** **elegante** y con **seguridad** de **tipos** que permite a los desarrolladores compilar diversas aplicaciones sólidas y seguras que se ejecutan en **.NET Framework**. Puede utilizar **C#** para crear aplicaciones cliente de Windows, servicios Web XML, componentes distribuidos, aplicaciones cliente-servidor, aplicaciones de base de datos, y mucho, mucho más. Visual **C#** proporciona un editor de código avanzado, cómodos diseñadores de interfaz de usuario, depurador integrado y numerosas herramientas más para facilitar el desarrollo de aplicaciones basadas el lenguaje **C#** y **.NET** Framework.

La sintaxis de **C#** es muy expresiva, pero también es sencilla y fácil de aprender basada en signos de llave podrá ser reconocida inmediatamente por cualquier persona familiarizada con **C, C++ o Java**. Los desarrolladores que conocen cualquiera de estos lenguajes pueden empezar a trabajar de forma productiva en **C#** en un plazo muy breve.

Los programas de **C#** se ejecutan en **.NET** Framework, un componente que forma parte de Windows y que incluye un sistema de ejecución virtual denominado **Common Language Runtime** (**CLR**) y un conjunto unificado de bibliotecas de clases. CLR es la implementación comercial de Microsoft de **CLI** (**Common Language Infrastructure**), un estándar internacional que constituye la base para crear entornos de ejecución y desarrollo en los que los lenguajes y las bibliotecas trabajan juntos sin ningún problema.

El código fuente escrito en **C#** se compila en un **lenguaje intermedio (IL)** conforme con la especificación **CLI**. El código de lenguaje intermedio y recursos tales como **mapas de bits y cadenas se almacenan en disco** en un archivo ejecutable denominado ensamblado, cuya extensión es **.exe** o **.dll** generalmente. Un ensamblado contiene un manifiesto que proporciona información sobre los tipos, la versión, la referencia cultural y los requisitos de seguridad del ensamblado.

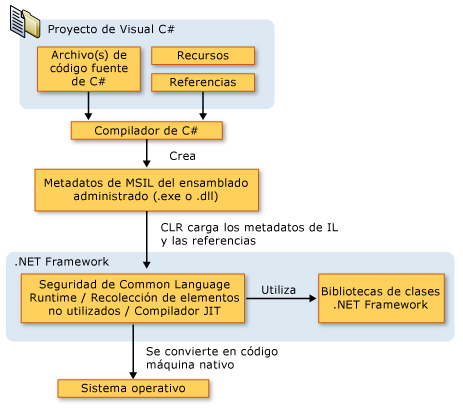


Figura 1.1 Esquema C#.

### Visual Studio

**Microsoft Visual Studio**  es un [entorno de desarrollo integrado](https://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado) (IDE, por sus siglas en inglés) para sistemas operativos [Windows](https://es.wikipedia.org/wiki/Windows). Soporta múltiples lenguajes de programación tales como [C++](https://es.wikipedia.org/wiki/Visual_C%2B%2B), [**C#**](https://es.wikipedia.org/wiki/C_sharp), [Visual Basic **.NET**](https://es.wikipedia.org/wiki/Visual_Basic_.NET), [F#](https://es.wikipedia.org/wiki/F_Sharp), [Java](https://es.wikipedia.org/wiki/Java_(lenguaje_de_programación)), [Python](https://es.wikipedia.org/wiki/Python), [Ruby](https://es.wikipedia.org/wiki/Ruby), [PHP](https://es.wikipedia.org/wiki/PHP); al igual que entornos de desarrollo web como [**ASP.NET**](https://es.wikipedia.org/wiki/ASP.NET) MVC, Django, etc., a lo cual sumarle las nuevas capacidades online bajo **Windows** **Azure** en forma del editor Monaco.

**Visual Studio** permite a los desarrolladores crear sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma **.NET** (a partir de la versión **.NET** 2002). Así se pueden crear aplicaciones que se comuniquen entre estaciones de trabajo, páginas web, dispositivos móviles, dispositivos embebidos, consolas, entre otros.



Figura 1.2 Visual Studio.

### Windows Forms

Con **Windows Forms**, puede desarrollar aplicaciones **smart client**. Las aplicaciones ***smart client*** son aplicaciones gráficamente enriquecidas, fáciles de implementar y actualizar, que pueden trabajar con o sin conexión a Internet y que pueden tener acceso a los recursos del equipo local de un modo más seguro que las aplicaciones tradicionales basadas en **Windows**.

En **Windows Forms**, un *formulario* es una superficie visual en la que se muestra información al usuario. Normalmente, las aplicaciones de **Windows** **Forms** se compilan mediante la adición de controles a los formularios y el desarrollo de respuestas a las acciones del usuario, como clics del mouse o presiones de teclas. Un *control* es un elemento de **interfaz de usuario (UI)** discreto que muestra datos o acepta la entrada de datos.

Con el Diseñador de **Windows Forms** de arrastrar y colocar de **Visual Studio**, puede crear fácilmente aplicaciones de **Windows Forms**. Simplemente seleccione los controles con el cursor y agréguelos donde desee en el formulario. El diseñador proporciona herramientas como líneas de cuadrícula y líneas de ajuste para minimizar la molestia de alinear los controles.

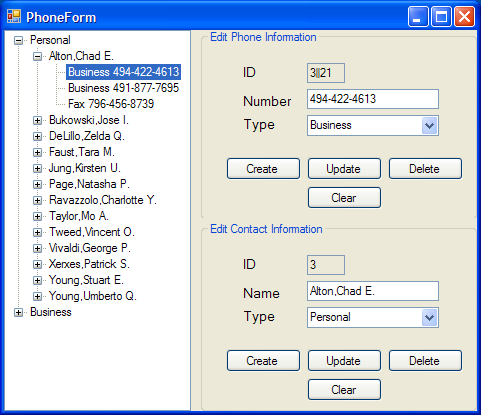


Figura 1.3 Ejemplo de Windows Forms.

### ASP.NET WebForms

**WebForms** es una parte de la tecnología **ASP.NET**, que permite crear interfaces de usuario para aplicaciones Web. Estas interfaces de usuario son independientes de la aplicación cliente (navegador) utilizada para representarlas, debido a que toda la parte de proceso que hay en éstas se ejecuta en el servidor.

Es posible utilizar WinForms en lugar de WebForms para crear la parte de usuario de una aplicación Web. En este caso, la parte de usuario sólo podrá ejecutarse en una máquina que tenga el **CLR** (no como un WebForm, que es soportado por cualquier navegador).

Un WebForm consiste en dos componentes, los elementos visuales y el código. En **Visual Studio**, por defecto cada uno de esos elementos se almacena en un fichero separado. Los elementos visuales se crean en un fichero .**ASP**x que se comporta como contenedor de los elementos **HTML** y los controles **WebForms**. El código se guarda en un fichero separado; si es **C#**, la extensión del fichero será .cs.

No obstante, es posible mantener los elementos visuales y el código en el mismo fichero: el diseñador de WebForms del **Visual Studio** no lo hace por defecto.

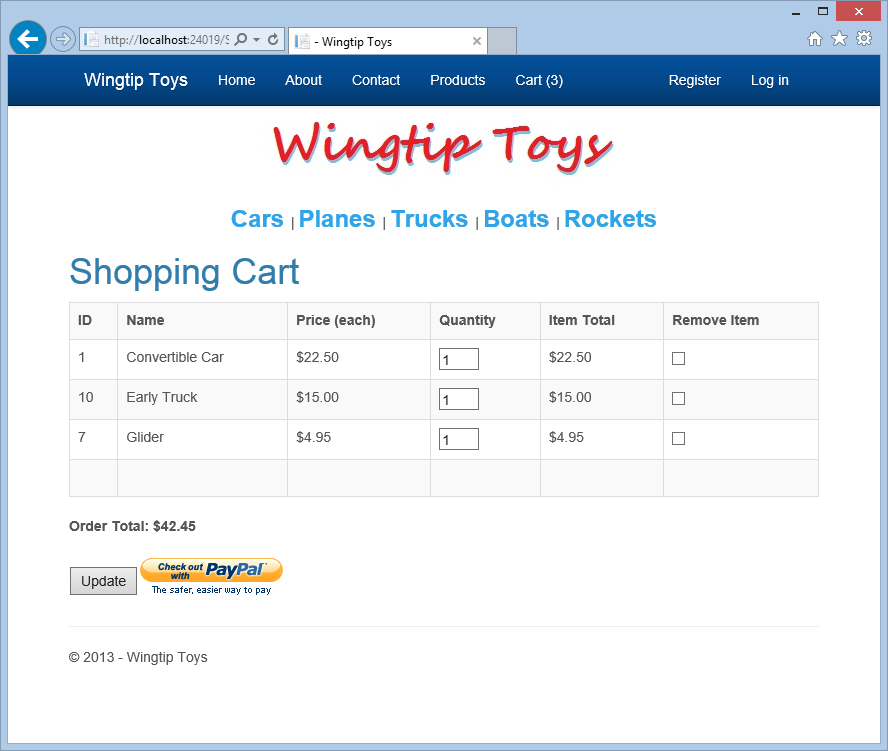


Figura 1.4 Ejemplo Web Forms.

Dada la limitación de WebForms, apareció una nuva forma de hacer los proyectos web. El modelo MVC de ASP.NET

### ASP.NET

**ASP.NET** es un modelo de desarrollo Web unificado que incluye los servicios necesarios para crear aplicaciones Web empresariales con el código mínimo. **ASP.NET** forma parte de **.NET** Framework y al codificar las aplicaciones **ASP.NET** tiene acceso a las clases en **.NET** Framework. El código de las aplicaciones puede escribirse en cualquier lenguaje compatible con el Common Language Runtime (CLR), entre ellos Microsoft Visual Basic, **C#**, JScript **.NET** y J#. Estos lenguajes permiten desarrollar aplicaciones **ASP.NET** que se benefician del Common Language Runtime, seguridad de tipos, herencia, etc.

Las páginas **Web** **ASP.NET** están completamente orientadas a objetos. En las páginas **Web** **ASP.NET** se puede trabajar con elementos **HTML** que usen propiedades, métodos y eventos. El marco de trabajo de páginas **ASP.NET** quita los detalles de implementación relacionados con la separación de cliente y servidor inherente a las aplicaciones Web presentando un modelo unificado que responde a los eventos de los clientes en el código que se ejecuta en el servidor. El marco de trabajo también mantiene automáticamente el estado de la página y de los controles que contenga durante el ciclo vital de procesamiento de la página.

### ASP.NET MVC

**MVC** es un modelo de diseño estándar con el que están familiarizados muchos desarrolladores. Algunos tipos de aplicaciones web salen beneficiados con el marco de MVC. Otras seguirán usando el modelo de la aplicación **ASP.NET** tradicional que está basado en formularios **Web** **Forms** y devoluciones. Otros tipos de aplicaciones web combinarán las dos estrategias; una no excluye a la otra.

El modelo arquitectónico **Modelo-Vista-Controlador** (**MVC**) separa una aplicación en tres componentes principales: el modelo, la vista y el controlador. El marco de **ASP.NET** **MVC** proporciona una alternativa al modelo de formularios **Web** **Forms** de **ASP.NET** para crear aplicaciones web. El marco de **ASP.NET** MVC es un marco de presentación de poca complejidad y fácil de comprobar que (como las aplicaciones basadas en formularios Web Forms) se integra con las características de **ASP.NET** existentes, tales como páginas maestras y la autenticación basada en pertenencia. El marco de MVC se define en el ensamblado System.Web.Mvc.

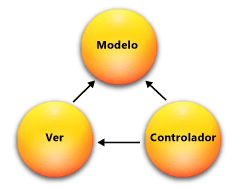


Figura 1.5 Modelo de diseño MVC.

# INSTALACIÓN DE VISUAL STUDIO

## Instalación

**.NET** es una parte integral de muchas aplicaciones que se **ejecutan** en **Windows** y proporciona la **funcionalidad** común para que dichas **aplicaciones** puedan **ejecutarse**. Para los desarrolladores, **.NET Framework** facilita un modelo de programación coherente y completa para crear aplicaciones que ofrezcan experiencias de usuario visualmente increíbles y una comunicación segura y sin problemas.

### Instalación mediante el instalador web

En **primer** **lugar**, accederemos a la web oficial de **Microsoft Visual Studio** , en la URL:

[***https://www.visualstudio.com/es-es/products/visual-studio-community-vs.ASPx***](https://www.visualstudio.com/es-es/products/visual-studio-community-vs.aspx)

Pulsaremos en **"Descargar Community 2015":**



Figura 2.6 Instalación Visual Studio

Se descargará un fichero para la instalación online. **Abriremos** el **fichero** descargado **vs\_community.exe:**

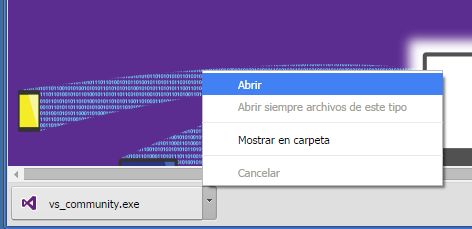


Figura 2.7 Instalación Visual Studio

Pulsaremos **"Ejecutar":**

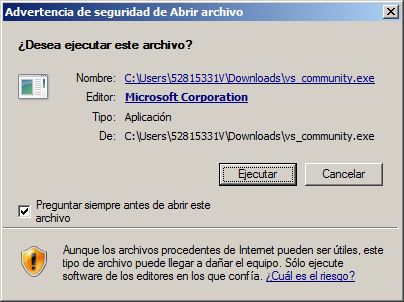


Figura 2.8 Instalación Visual Studio

Se iniciará el asistente de instalación de **Visual Studio Community 2015**, el asistente realizará una comprobación previa de requisitos, si **falta** **algún** **requisito** nos lo **indicará**. En nuestro caso estamos instalado **Visual Studio Community 2015** en un equipo con Windows 7, por lo que nos recomendará que **actualicemos** **Internet** **Explorer** a la **versión** **10**, aunque podemos continuar la instalación sin hacerlo. Pulsaremos **“Continuar”:**

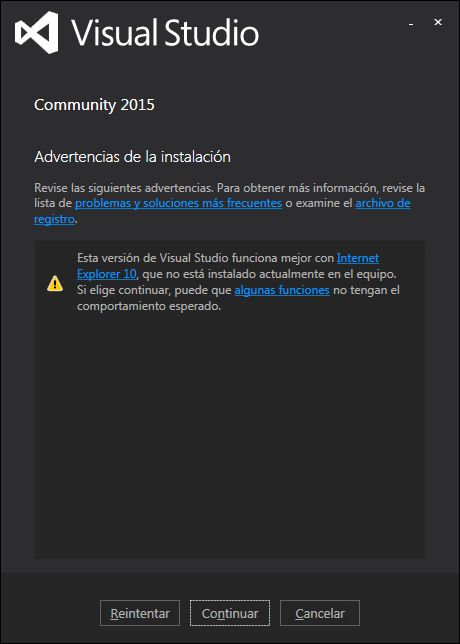


Figura 2.9 Instalación Visual Studio

Elegimos la ubicación donde queramos instalar nuestro **Visual Studio** y elegimos el tipo de instalación.

Marcaremos **“Personalizada”** para elegir las opciones de instalación de **Visual Studio** Community 2015:

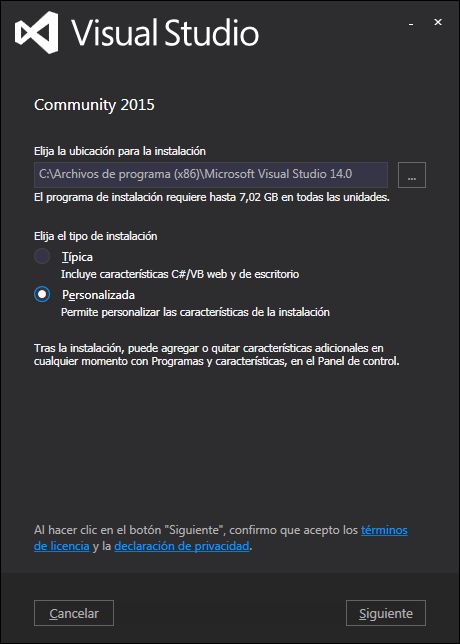


Figura 2.10 Instalación Visual Studio.

Marcaremos ahora las características que queramos instalar (Web Developer Tools, PowerShell para **Visual Studio**, Silberlight, desarrollo de aplicaciones universales, desarrollo para móviles multiplataforma (**C#**/**.NET** (Xamarin), Apache Cordova, desarrollo móvil de Visual C++, desarrollo de iOS de Visual C++, WebSocket4Net, etc.). En nuestro caso marcaremos todas las características (en caso de que queramos tener todos los componentes de **Visual Studio**, hay que tener en cuenta el tamaño de la instalación; 24,08GB) y pulsaremos **"Siguiente":**

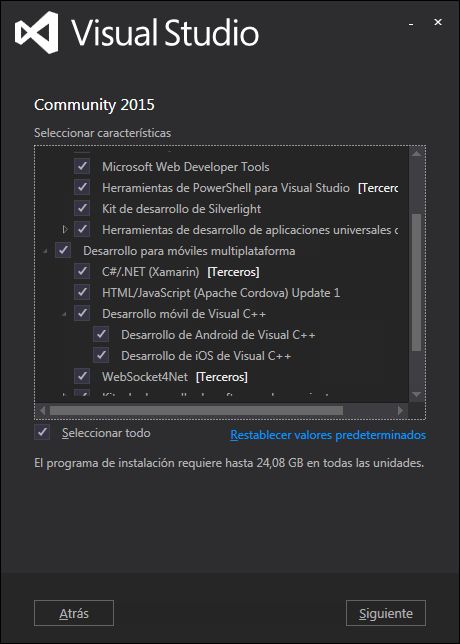


Figura 2.11 Instalación Visual Studio.

Leeremos los términos de licencia de las distintas características marcadas, si estamos de acuerdo continuaremos con la instalación de **Visual Studio** **.NET** **Community** **2015** pulsando en **“Instalar”:**

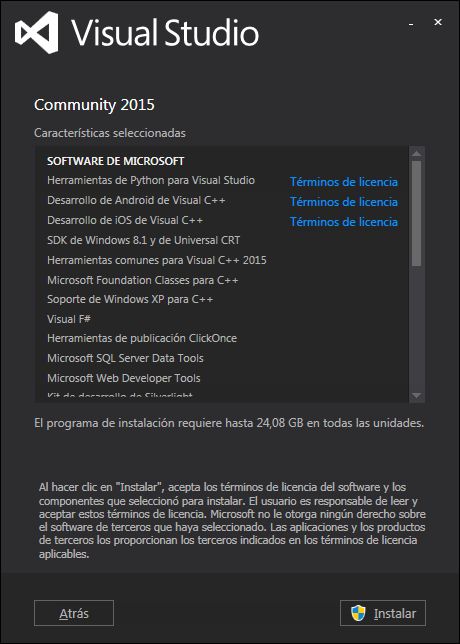


Figura 2.12 Instalación Visual Studio.

Se iniciará la descarga e instalación definitiva de **Visual Studio** **.NET** **Community** 2015:

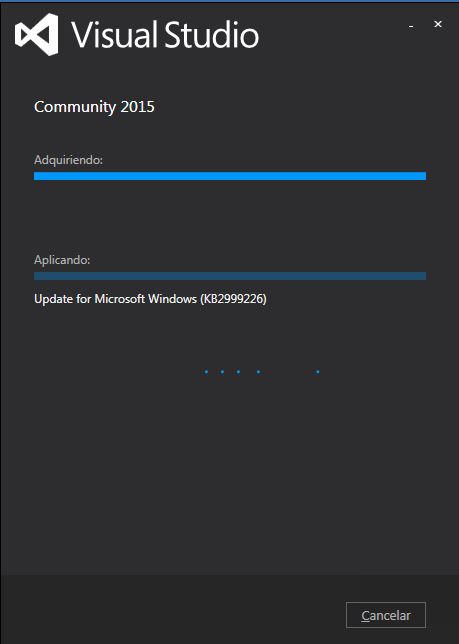


Figura 2.13 Instalación Visual Studio.

Tras varios minutos, dependiendo de las características elegidas el asistente nos indicará que la instalación de **Visual Studio** **.NET** **Community** **2015** ha concluido. Es posible que nos solicite reiniciar el equipo, si es así pulsaremos en **"Reiniciar ahora"**.

# APLICACIONES DE ESCRITORIO

## Introduccion a Aplicación Escritorio

Nos enfrentamos a nuestra primera aplicación de escritorio en **.NET**. Para ello haremos uso de **Windows** **Forms**.

Empezaremos con una aplicación simple que nos ayude a comprender el uso.

### Creación del proyecto

Comenzamos creando nuestro proyecto, para ello ejecutamos **Visual Studio** y nos abrirá la siguiente ventana:

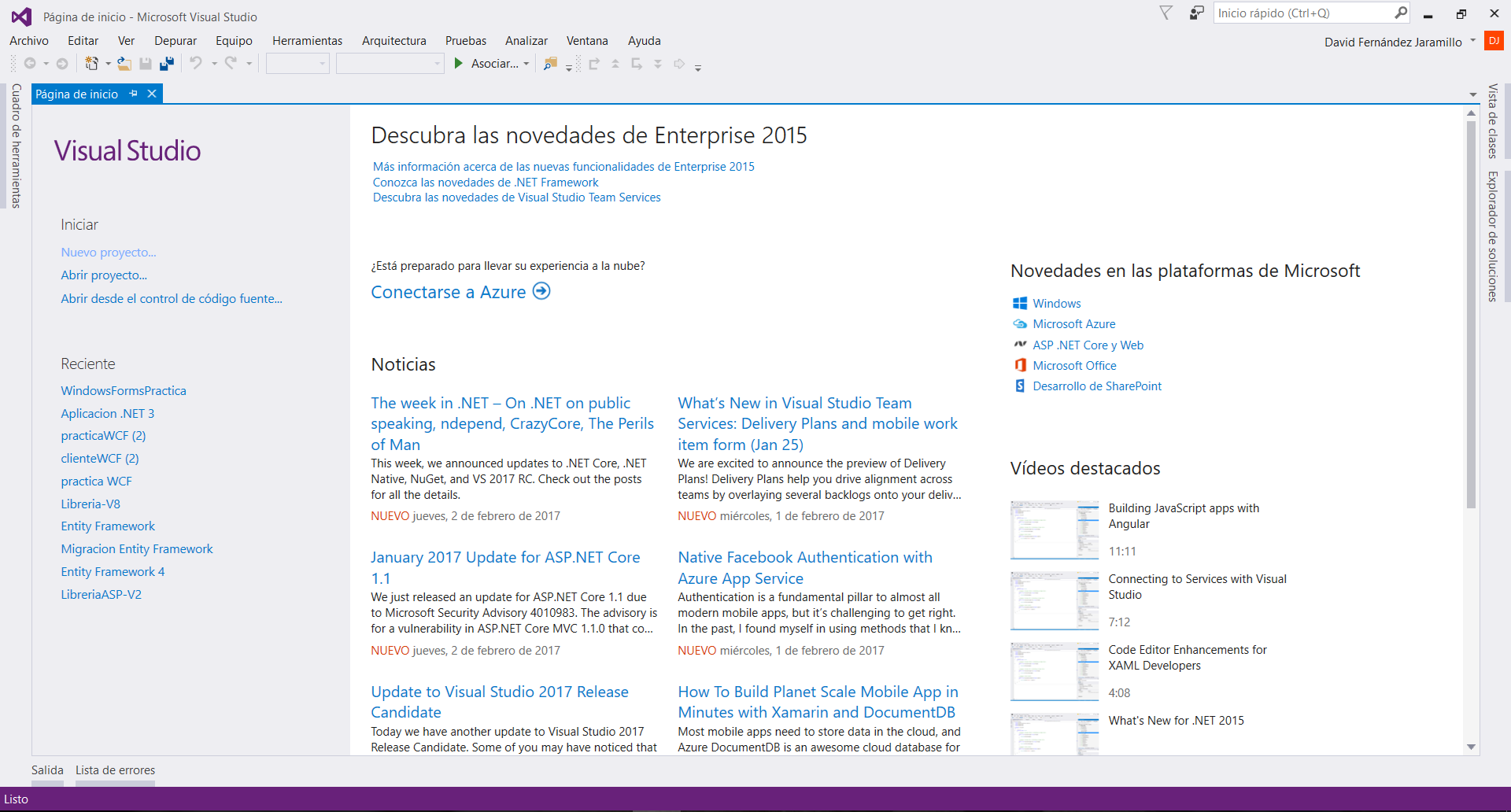


Figura 4.14 Ventana inicio.

En esta ventana nos aparecerán **noticias** **relacionadas** con **.NET**, novedades, actualizaciones… además de nuestros proyectos recientes.

Pulsamos sobre **Nuevo** **Proyecto** y aparecerá lo siguiente…

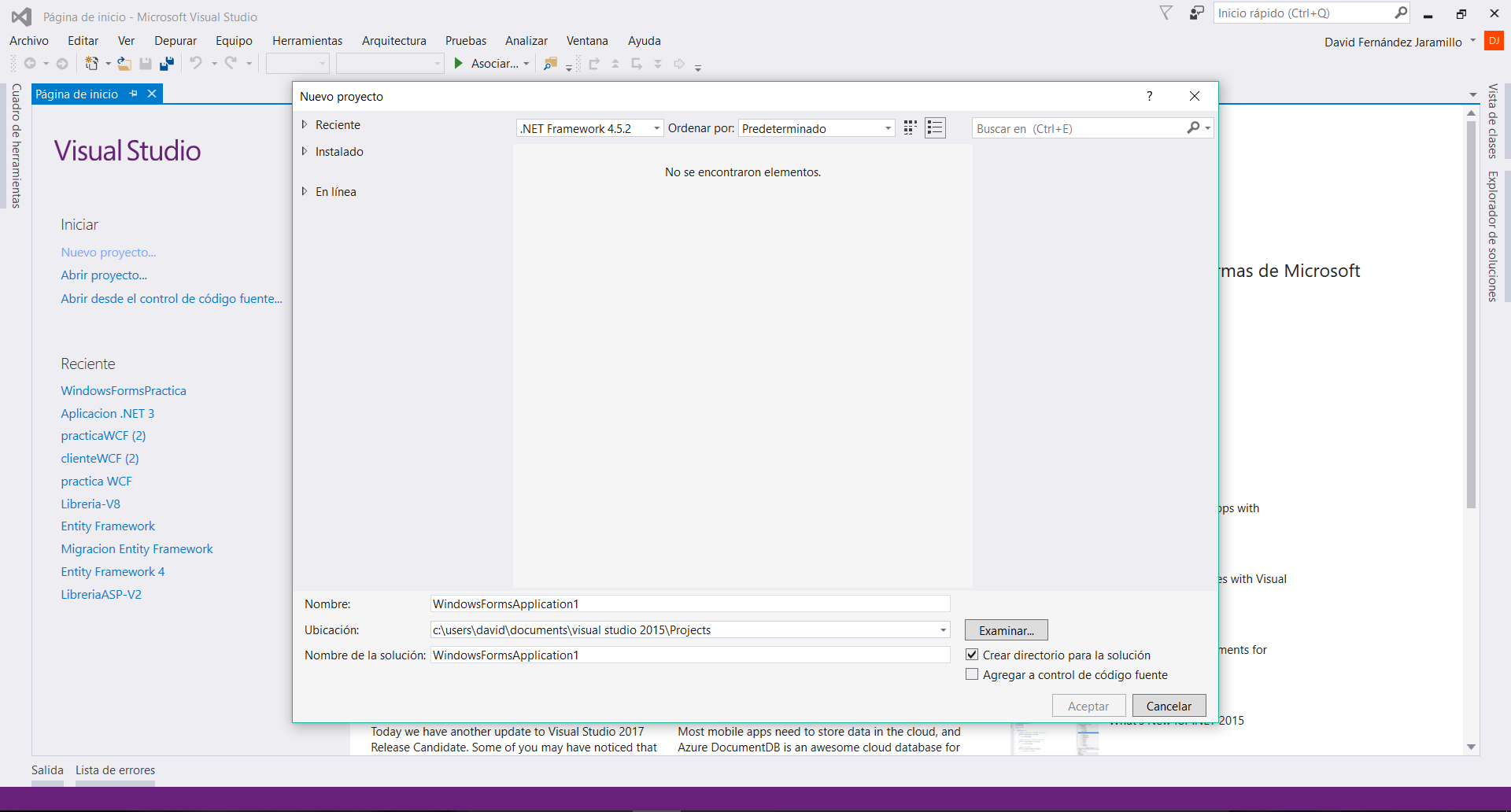


Figura 4.15 Creando el proyecto.

A continuación, pulsaremos sobre la **opción** **plantillas**, dentro de ésta, pulsamos **Visual** **C#** y posteriormente en Windows:

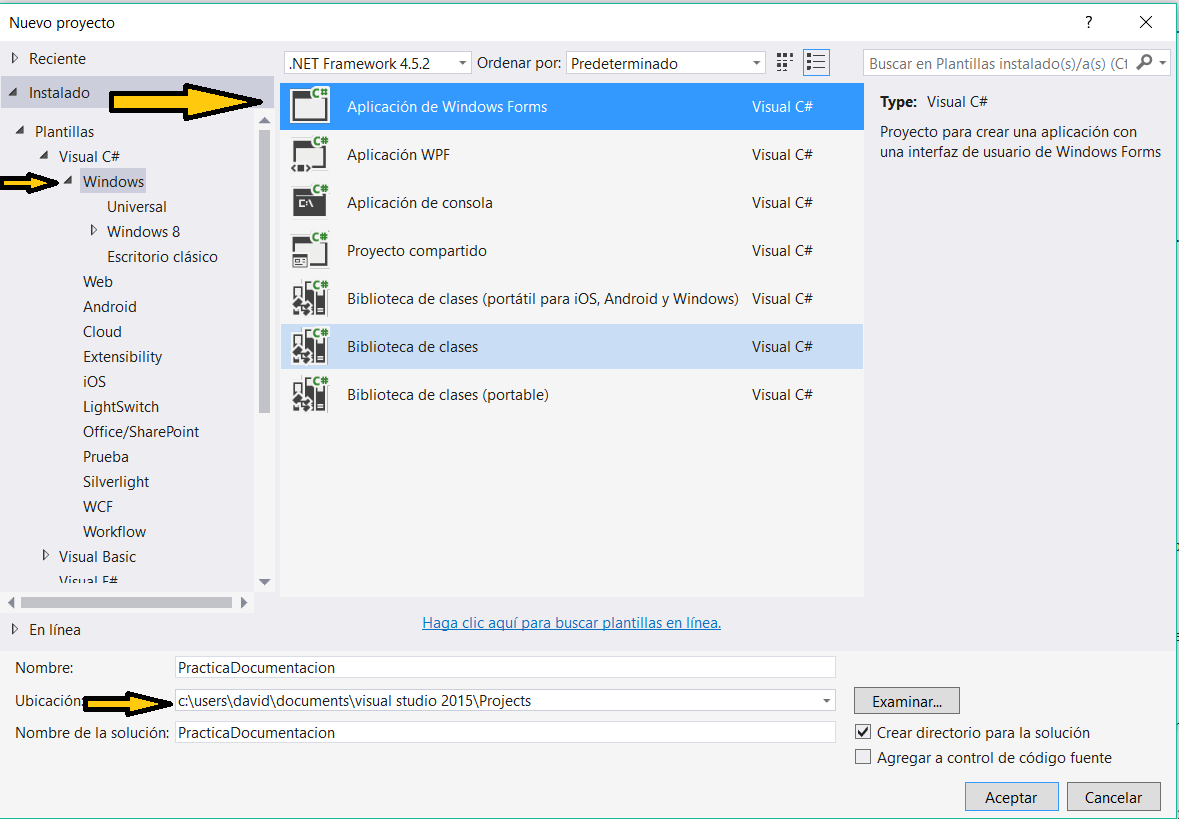


Figura 4.16 Opciones de proyecto.

De las opciones que nos aparecen elegiremos aquella que pone Aplicación de **Windows** **Forms**. En la parte inferior elegiremos el nombre de nuestro proyecto y la ruta donde guardarlo.

Le damos a **Aceptar** y nos creará el proyecto y aparecerá lo siguiente:

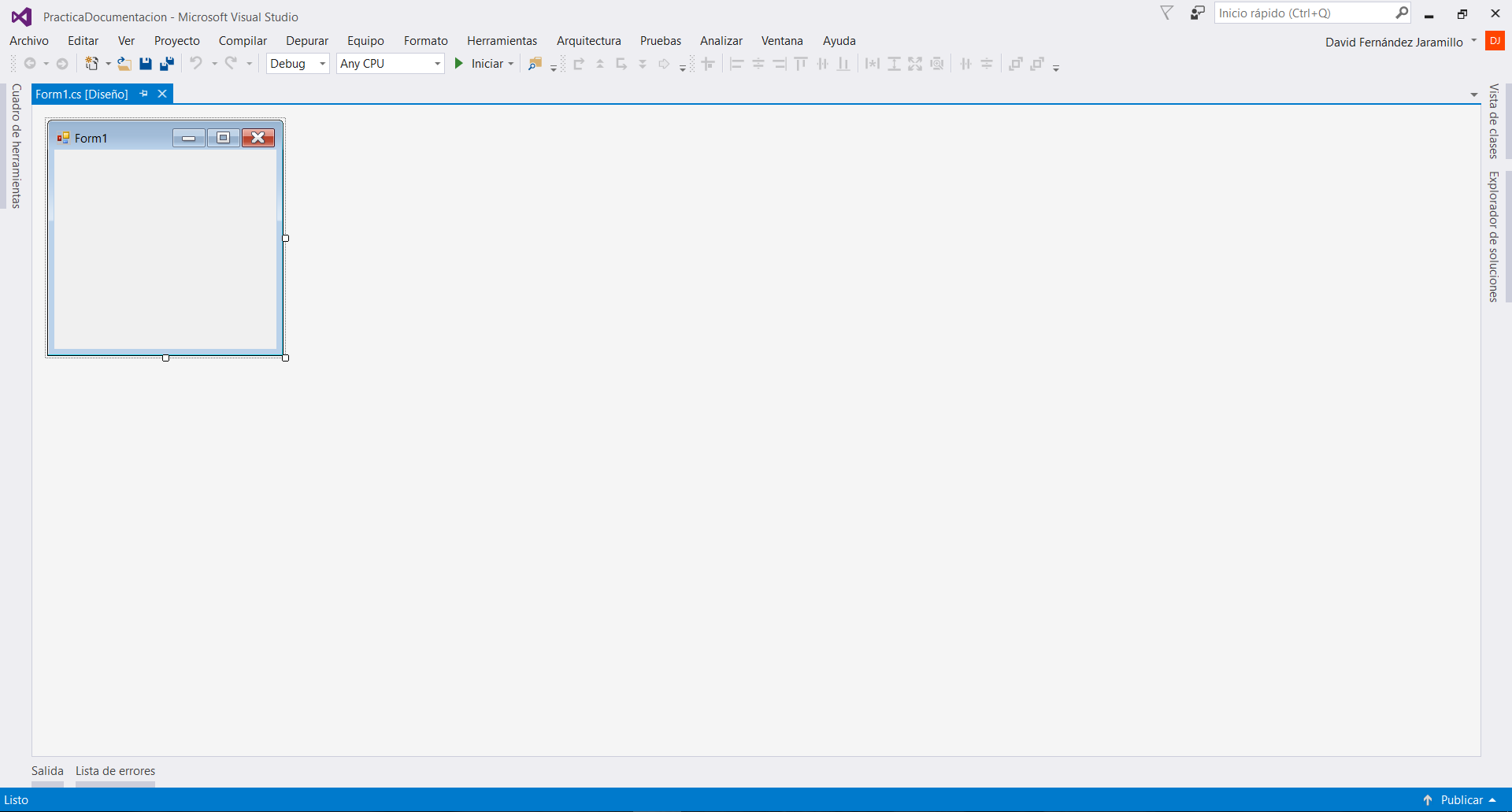


Figura 4.17 Proyecto vacío.

### Desarrollo de la aplicación

Pasamos a desarrollar la aplicación. Para ello simplemente **arrastraremos** a la **ventana** los **elementos** que **deseemos** y **posteriormente** se desarrolla el comportamiento.

Le damos a **Explorador** **de soluciones** y saldrá lo siguiente:

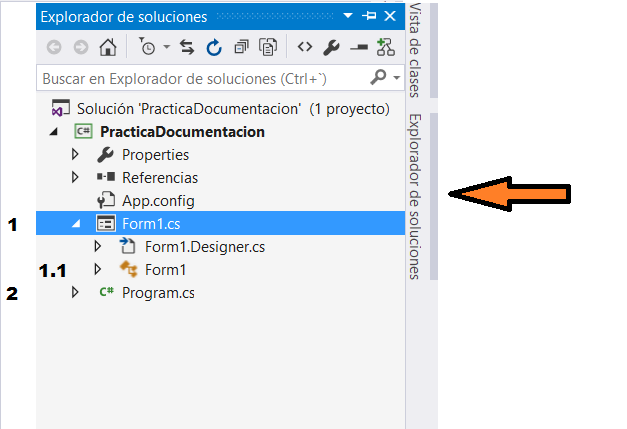


Figura 4.18 Explorador de soluciones.

1. Es el **formulario** que aparece al **crear** **la** **aplicación**, ahí es donde se diseñara la **interfaz** **gráfica**.
   1. Es la clase que gestiona las acciones de la interfaz grafica
2. Ahí se encuentra el **main**, que es desde donde se inicia la aplicación

Ahora vamos a diseñar la interfaz gráfica para ello ponemos la ventana más grande utilizando los **puntos** **blancos** que salen en los bordes.

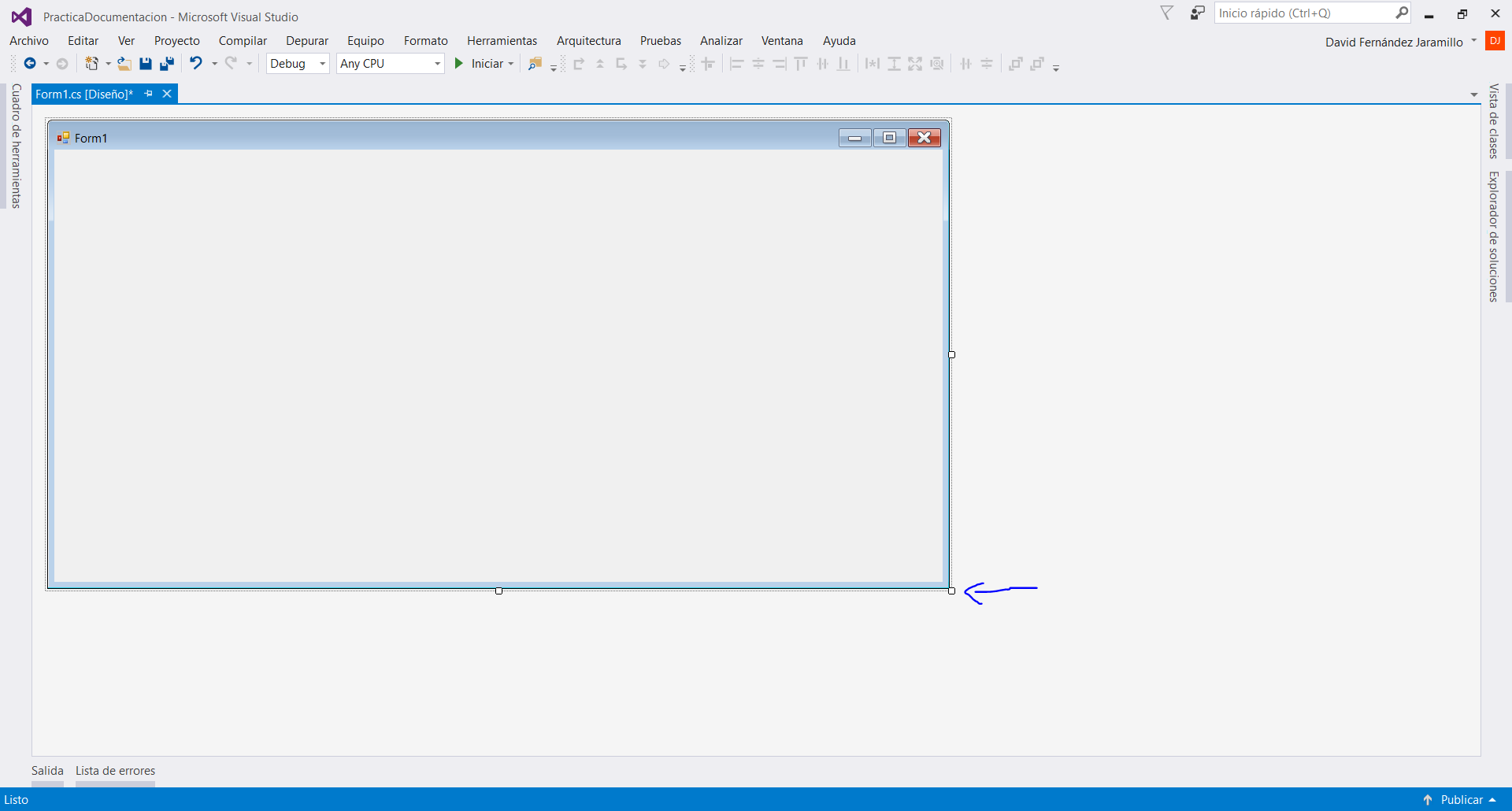


Figura 4.19 Ventana.

A continuación, **seleccionados** donde pone cuadro de herramientas y seleccionamos el elemento **Buttom**.

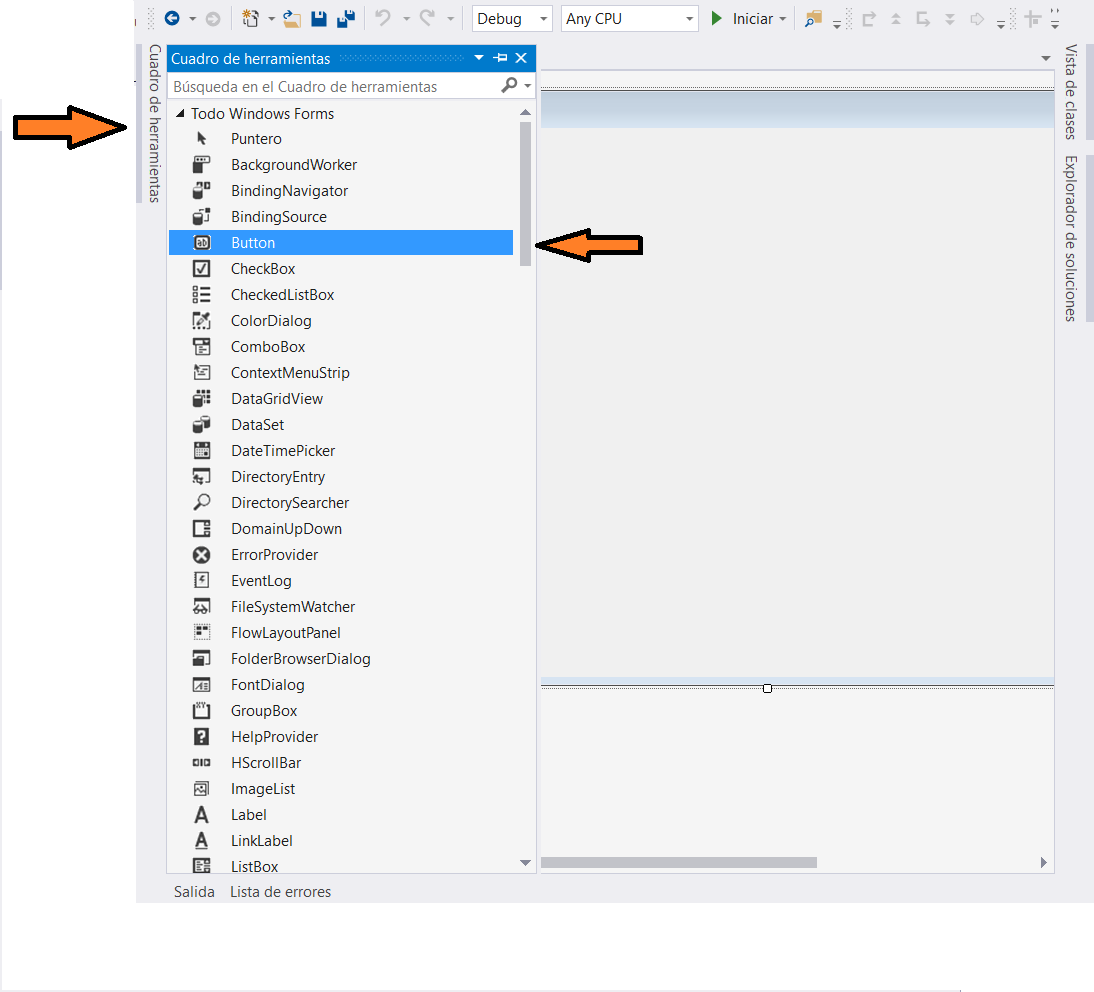


Figura 4.20 Cuadro de herramientas.

Lo arrastramos a la **pantalla** y ya tendríamos el botón **creado**.

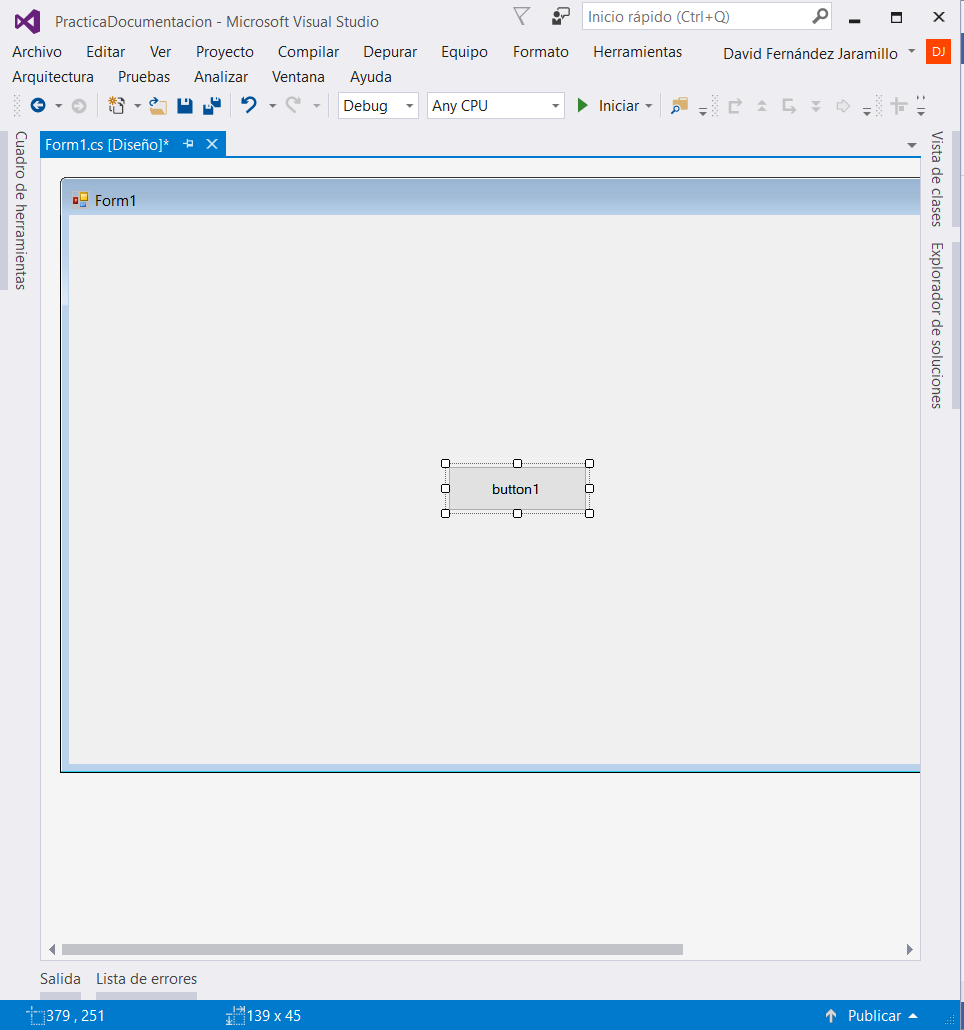


Figura 4.21 Creando un botón.

Hacemos **clic** **derecho** sobre el **botón** y vamos a las **Propiedades**.

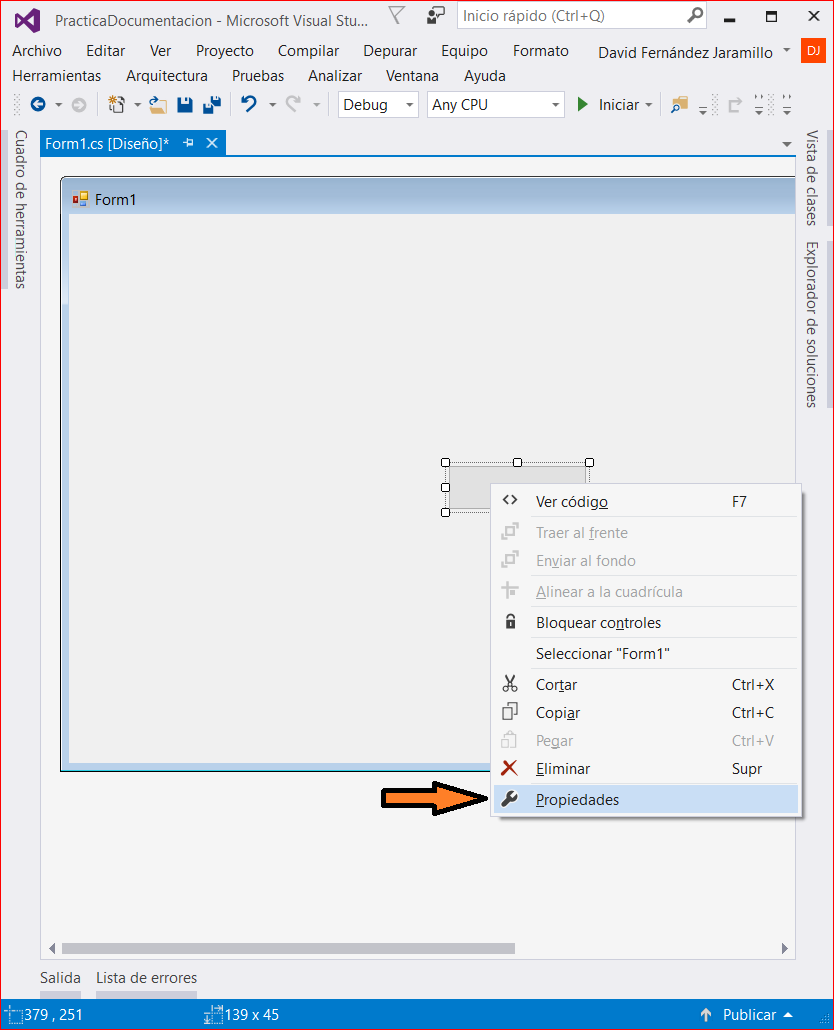


Figura 4.22 Propiedades del botón.

**Cambiamos** el **valor** del **texto** que tiene el botón, para ello **buscamos** la **propiedad** **Text** y la cambiamos:

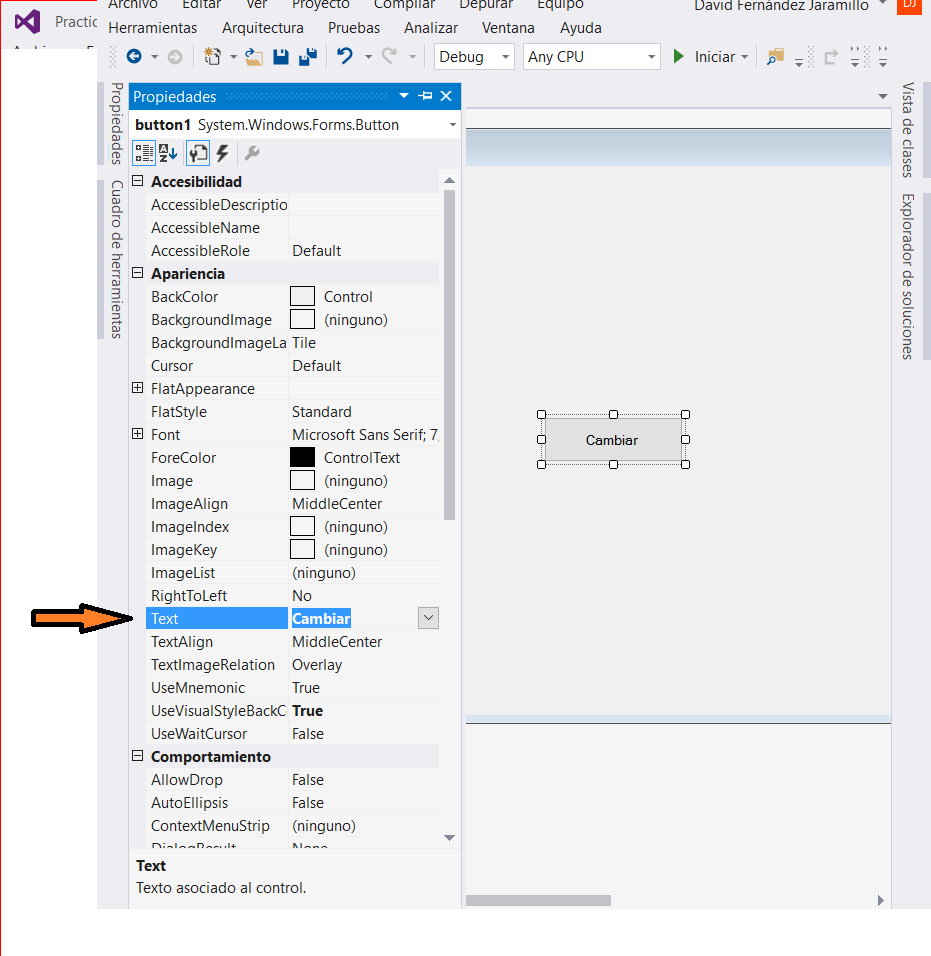


Figura 4.23 Propiedad text.

Vamos a crear una **Caja de texto**, para ello vamos al Cuadro de herramientas, y buscamos el elemento en el buscador poniendo “**text**”, seleccionamos **TextBox** y lo añadimos a nuestra aplicación como lo hemos hecho anteriormente.

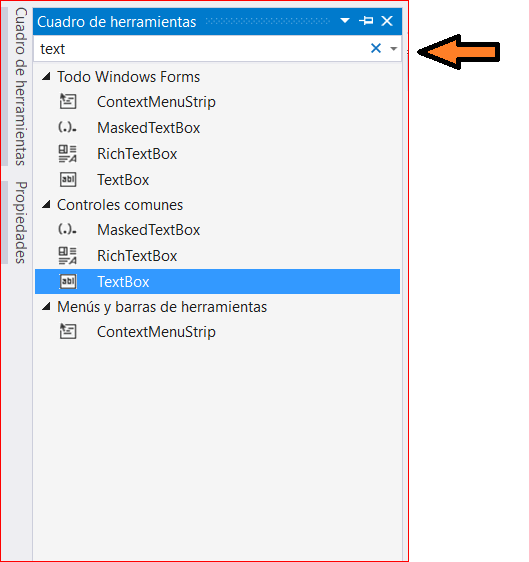


Figura 4.24 Caja de texto.

En a las **propiedades** del botón seleccionamos la **lista** **de** **eventos**:

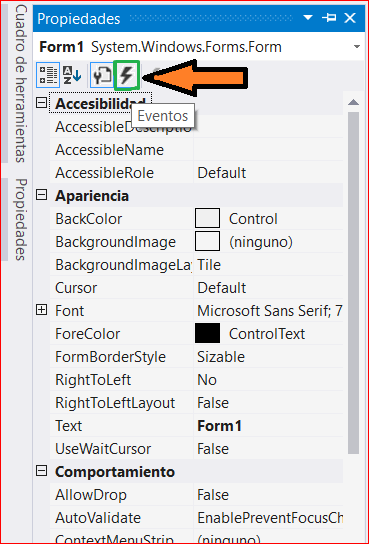


Figura 4.25 Lista de eventos.

Vamos a **Acción** donde encontraremos la propiedad **Click** que tiene lugar cuando pulsas el botón, le damos un **nombre** al **evento** y pulsamos **Intro**:

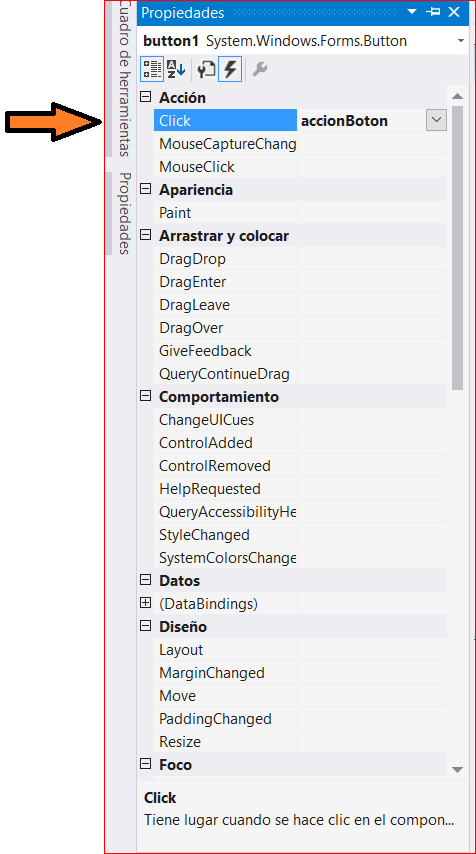


Figura 4.26 Acción clic.

Nos **lleva** a la **clase** que gestiona la **interfaz** donde nos crea el evento con el **nombre** que le hemos dado:

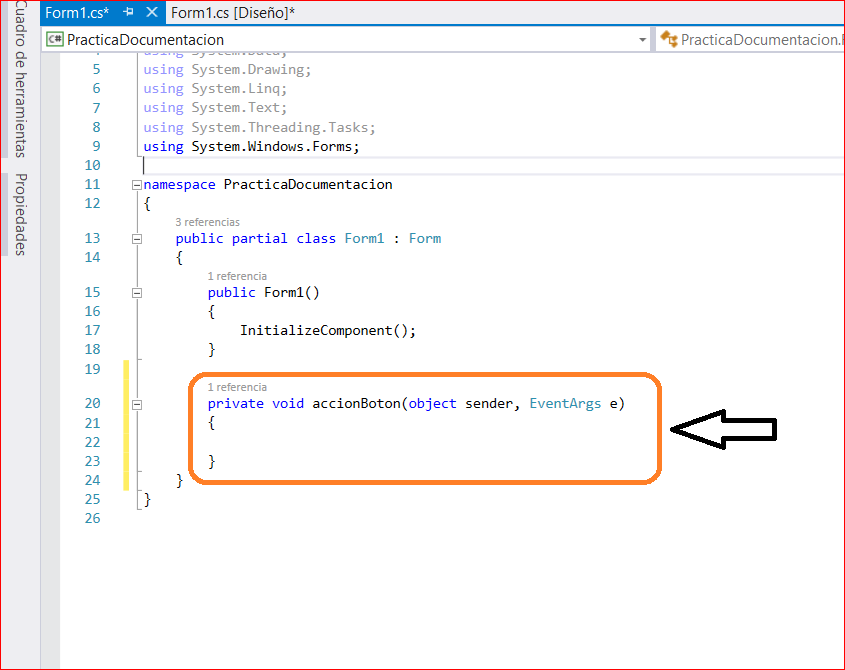


Figura 4.27 Evento del clic.

Añadimos el siguiente código.

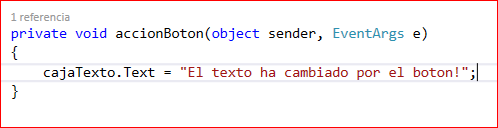


Figura 4.28 Desarrollo del evento.

**cajaTexto**: hace referencia al **TextBox** que creamos anteriormente.

**Text**: es una **propiedad** que **permite** tanto **obtener** el **texto** **como** **modificarlo**, en este caso lo **estamos** **modificando**.

Funciona como **get** y **set** a la vez, aquí no hace falta poner **getPropiedad**, **setPropiedad**, con llamar al nombre de la propiedad basta. Ahora **arrancamos** nuestra a **aplicación** pulsando el botón iniciar:

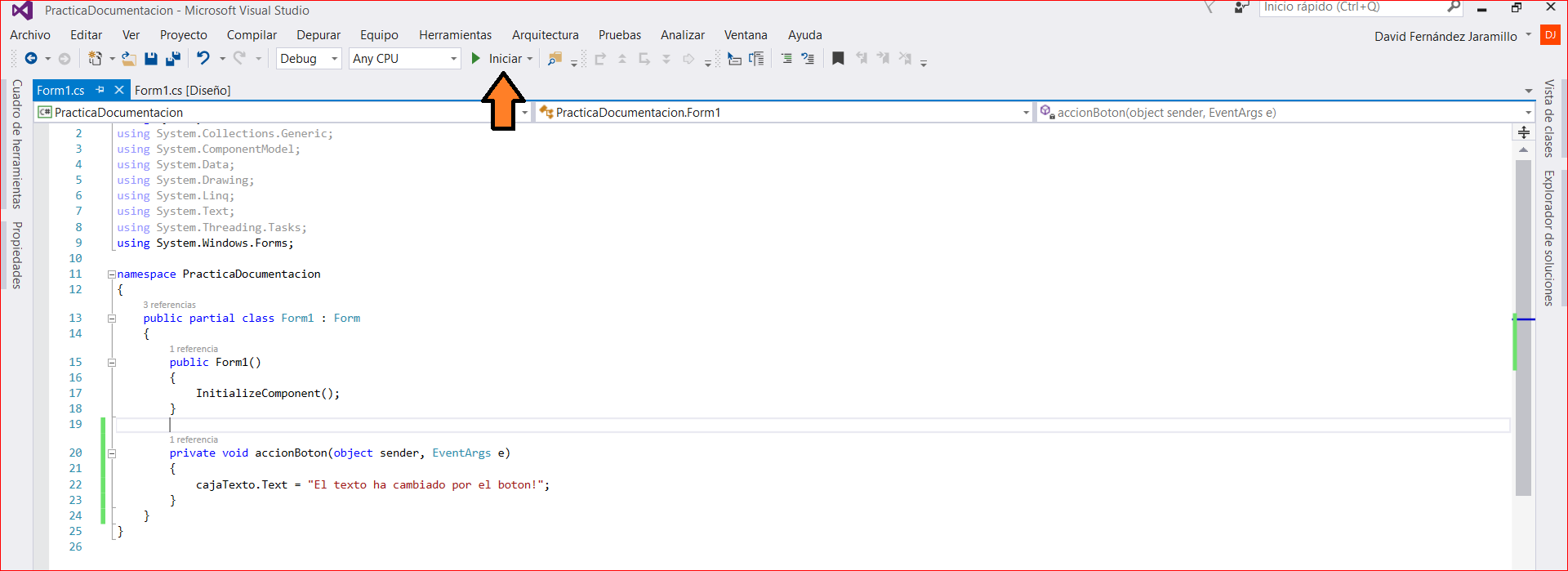


Figura 4.29 Arrancando la aplicación.

Como se observa podemos añadir un **Texto** al **TextBox**:

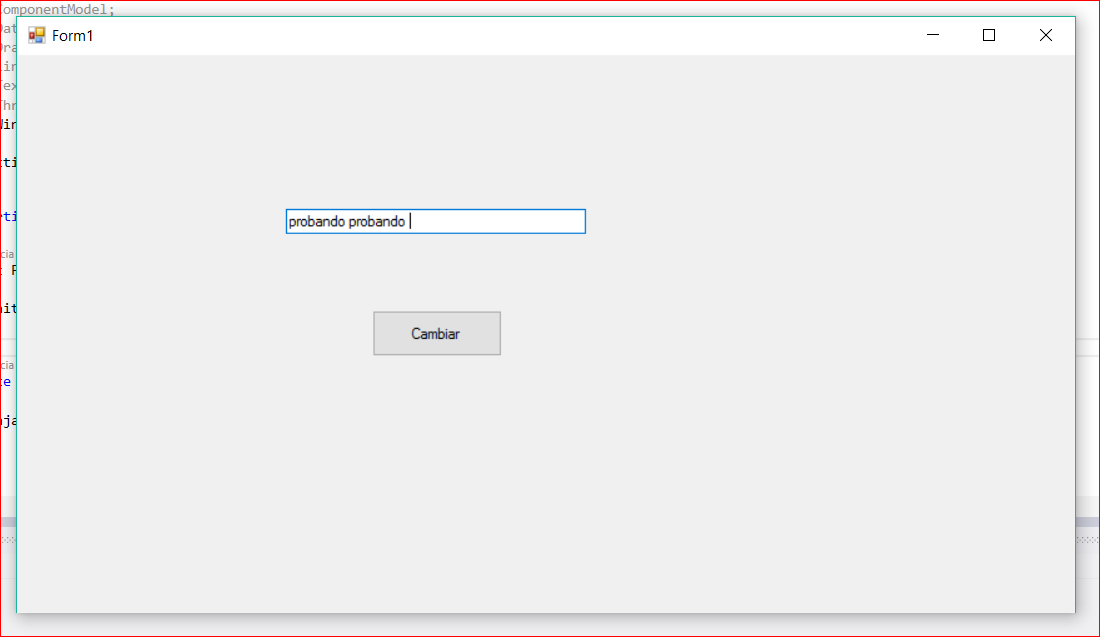


Figura 4.30 Probando.

Cuando pulsemos el **botón** este texto cambiará por el que tengamos en nuestra acción:

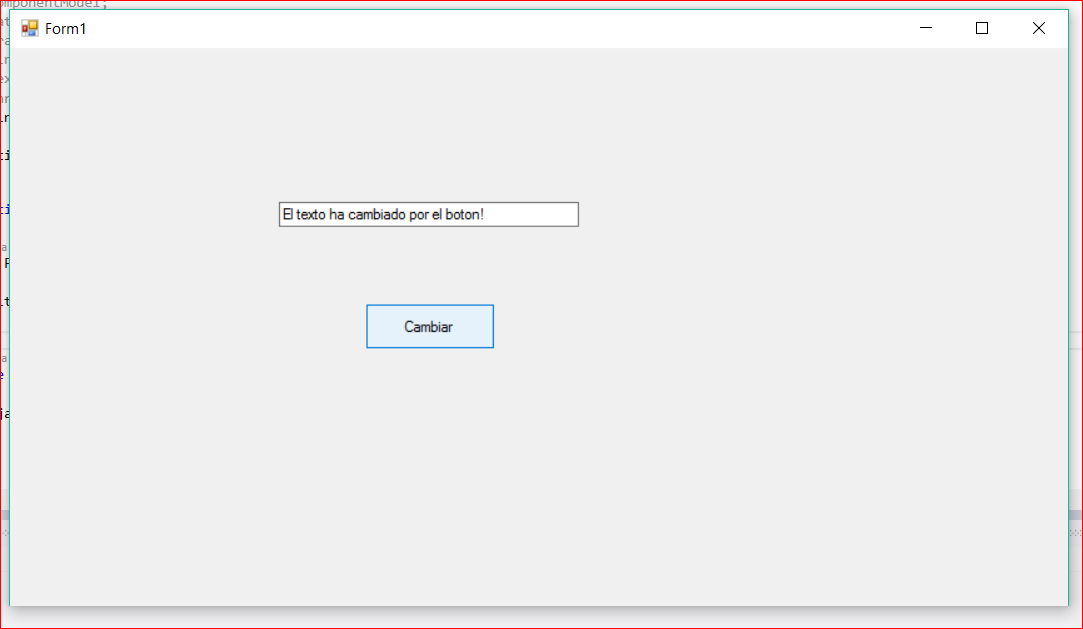


Figura 4.31 Resultado final.

# PROYECTOS

1. *Aplicación introductoria de escritorio:* [Aplicacion introduccion escritorio](file:///C:/Users/chema/Desktop/CASTELAR/DAM/1-TEMARIO/ESCRITORIO-NET/1-DOCUMENTACIÓN/Proyectos/Aplicacion introduccion Escritorio)*.*

# ANEXOS

## Anexo Sintaxis C#

El “Anexo Sintaxis” es un documento aparte a éste, en el que se desarrolla toda la sintaxis del lenguaje de programación C#.

Link: [Anexo Sintaxis C#](file:///C:/Users/chema/Desktop/CASTELAR/DAM/1-TEMARIO/ESCRITORIO-NET/1-DOCUMENTACIÓN/Anexo_Sintaxis_CSharp.docx).