**UNIVERSIDAD MARIANO GALVEZ DE GUATEMALA  
MAZATENANGO SUCHITEPEQUEZ**

**FACULTAD DE INGENIERIA EN SISTEMAS DE INFORMACION Y CIENCIAS DE LA COMPUTACION**

***Ing. Axel Aguilar Carrillo****Desarrollo Web*

**Manual Técnico - Usuario**

**Sistema de Inventarios Web**

**Norman Vinicio García Ramírez   
3090 13 3597**

**Noviembre 2018**

Descripción

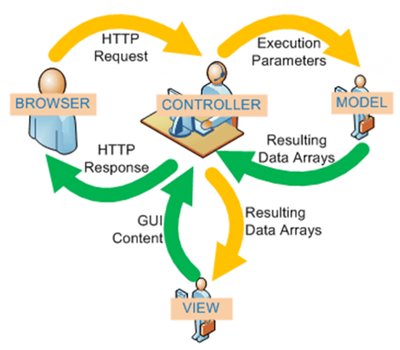
Sistema Web para el control de Inventarios, entradas y salidas de productos.

Arquitectura

El sistema se implementará bajo la arquitectura MVC (Modelo Vista Controlador) que es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

* El Modelo que contiene una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia.
* La Vista, o interfaz de usuario, que compone la información que se envía al cliente y los mecanismos interacción con éste.
* El Controlador, que actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

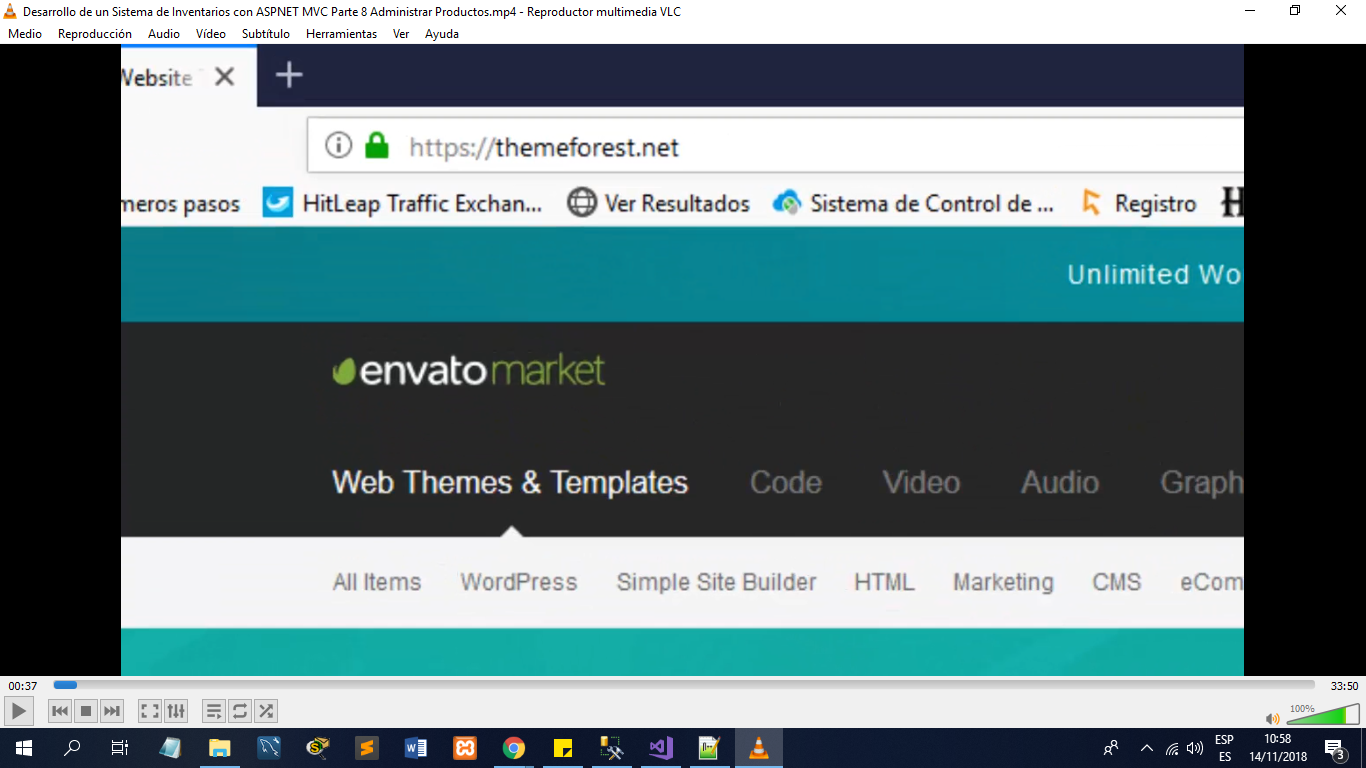
El flujo que sigue el control generalmente es el siguiente:



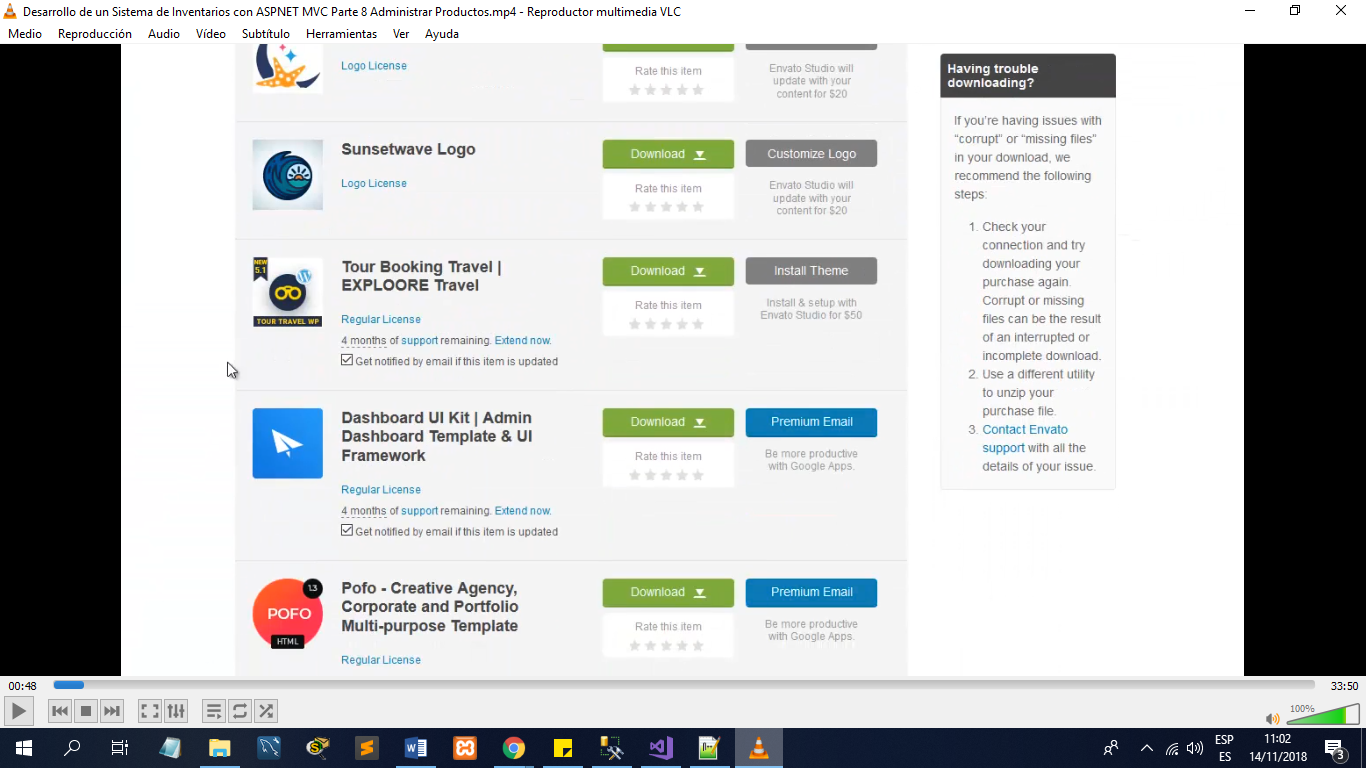
* El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc.)
* El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler).
* El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.

Componentes

* **Microsoft Visual Studio 2017**:
* **Aspx .Net con C#**: Lenguaje de programación orientada a objetos bajo el cual se desarrollará el sistema ASP.NET es un entorno para aplicaciones web desarrollado y comercializado por Microsoft. Es usado por programadores y diseñadores para construir sitios web dinámicos, aplicaciones web y servicios web XML.ASP.NET está construido sobre el Common Language Runtime, permitiendo a los programadores escribir código ASP.NET usando cualquier lenguaje admitido por el .NET Framework.
* **Java Script y Ajax**: Este lenguaje y esta tecnología se utilizarán para las autenticaciones en las acciones de los botones en todo el sistema.
* **Json**: es un formato de texto ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript aunque hoy, debido a su amplia adopción como alternativa a XML, se considera un formato de lenguaje independiente.
* **CSS**: es una hoja de estilos que nos permite tener vistas más estéticas en el sistema para dar un mejor ambiente al usuario.
* **HTML**: Lenguaje de Maquetación sobre el cual se crearán las vistas y formularios Web.
* **Framework 4.6.2**: es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar algunos usos fueron al momento de encriptar contraseñas o códigos a la base de datos.
* **Framework Entity**: Este Framework de Microsoft nos facilitará la manipulación de datos y las consultas a la base de datos, así como la inserción de datos, este Framework permite que sea de forma más general.
* **Template**: Paquete (**Deshboard UI Kit**) que contiene las vistas Home contenidas en el programa y descargadas en la siguiente página.

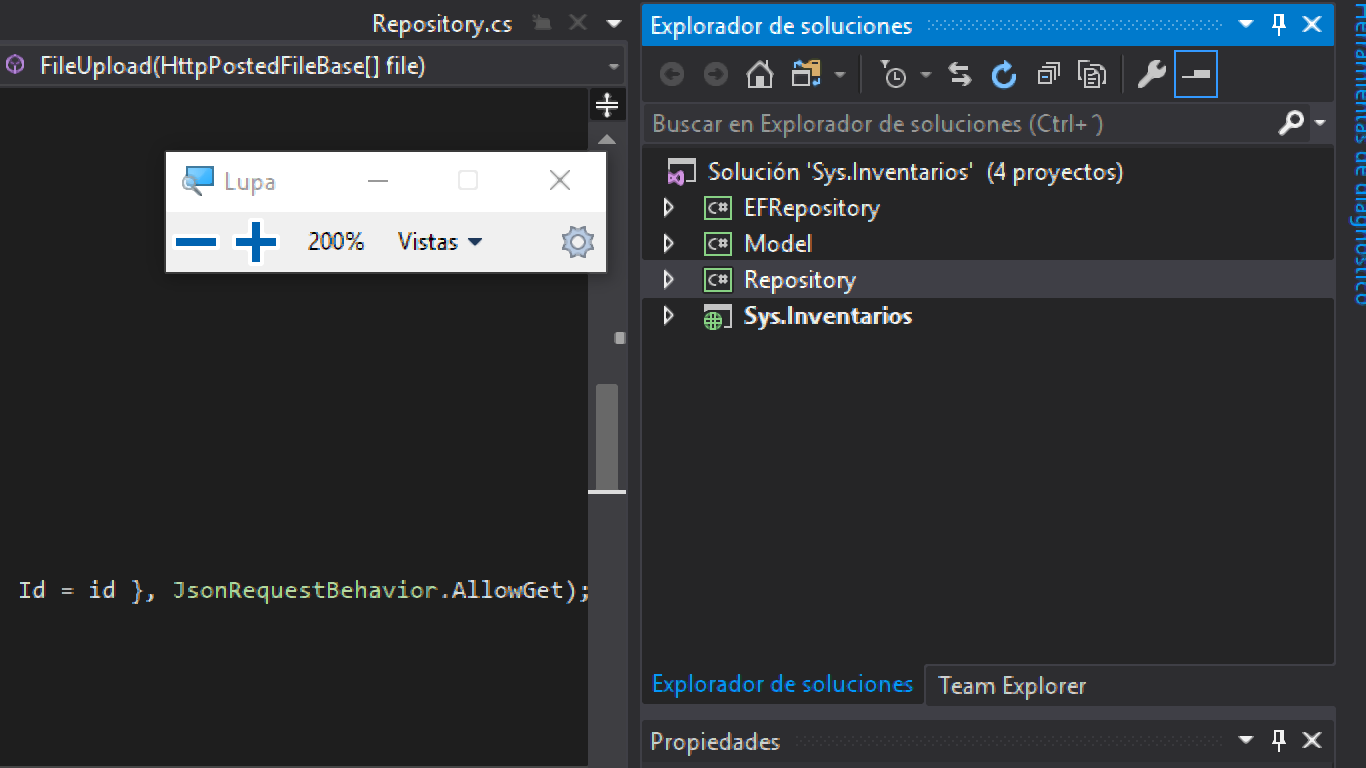


Se descarga y se modifican al gusto las vistas que contiene el kit dependiendo lo que se necesite, en este caso serán el Home, tablas e ingreso de datos.

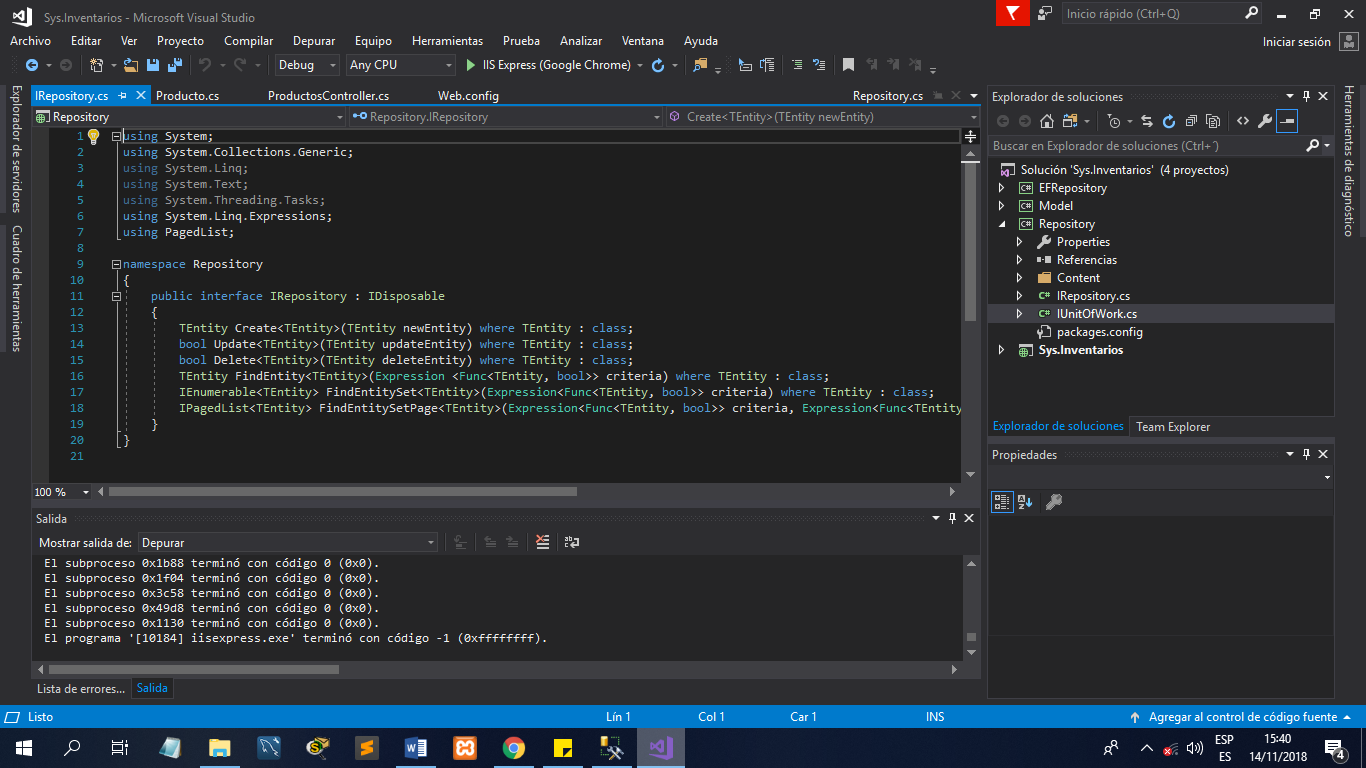


**Proyecto SysInventarios**

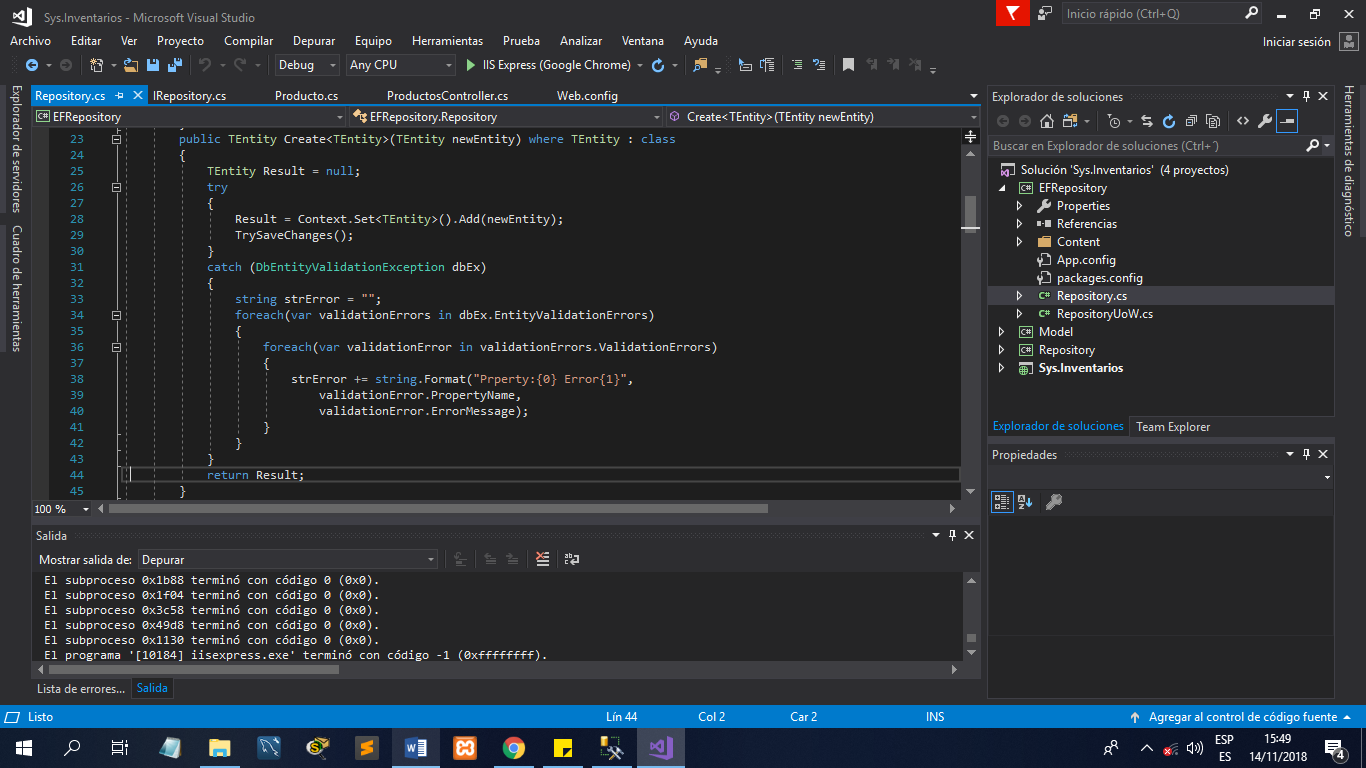
El proyecto contiene 4 carpetas 3 bibliotecas de clases y la carpeta principal que contiene todo el proyecto MVC:



* **Repository:** en el repositorio tendremos declarados los métodos con propiedades que utilizaremos para todo el CRUD, y métodos para consultas especificas (Create, Update, Delete) dentro de una interface llamada IRepository.

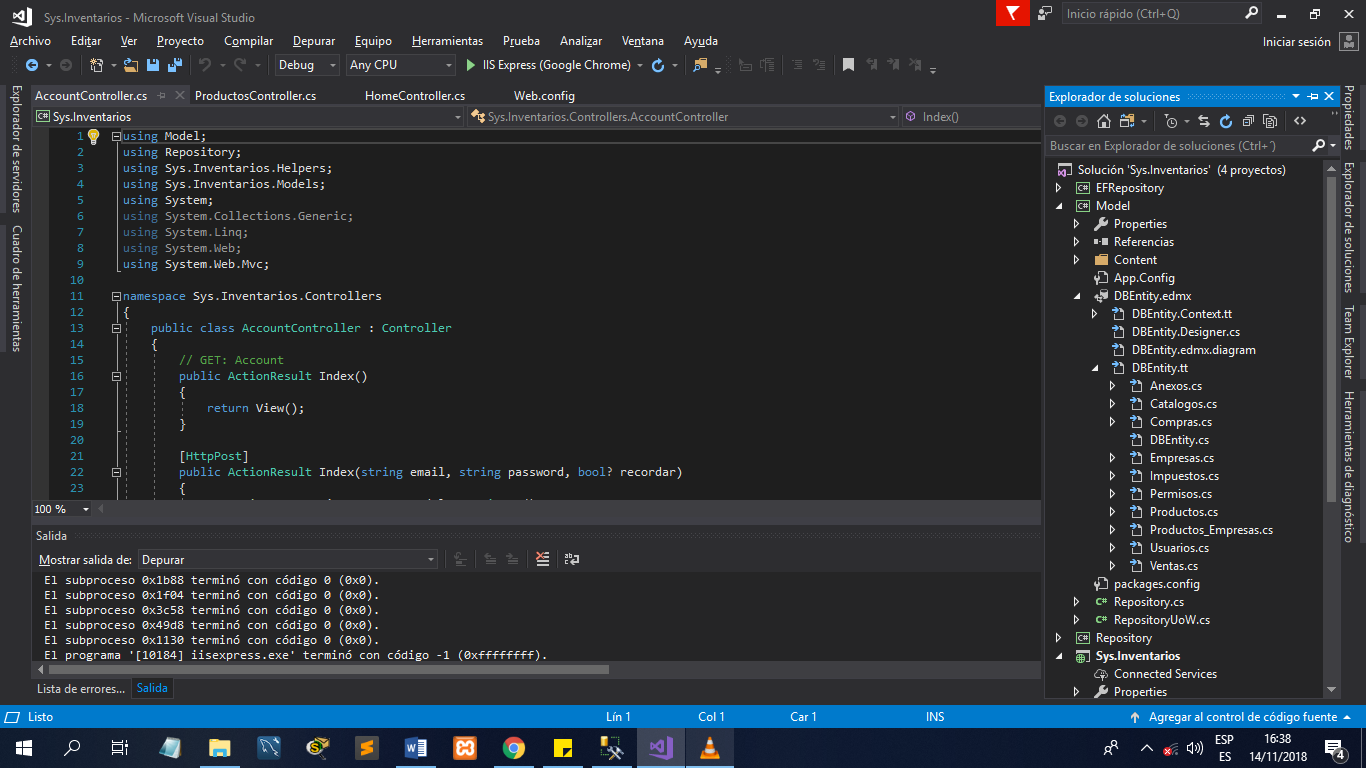


* **EFRepository**: en esta biblioteca de clases agregaremos la funcionalidad que tendrá cada método incluido en el repositorio, implementa cada una de las marcas o métodos, esta clase es la que hace los querys a la base de datos y regresa una lista de entidades por ejemplo el método Create como se muestra a continuación.



* **Model**: En el Model o modelo estará la conexión directa a la base de datos y las entidades incluidas en ella.

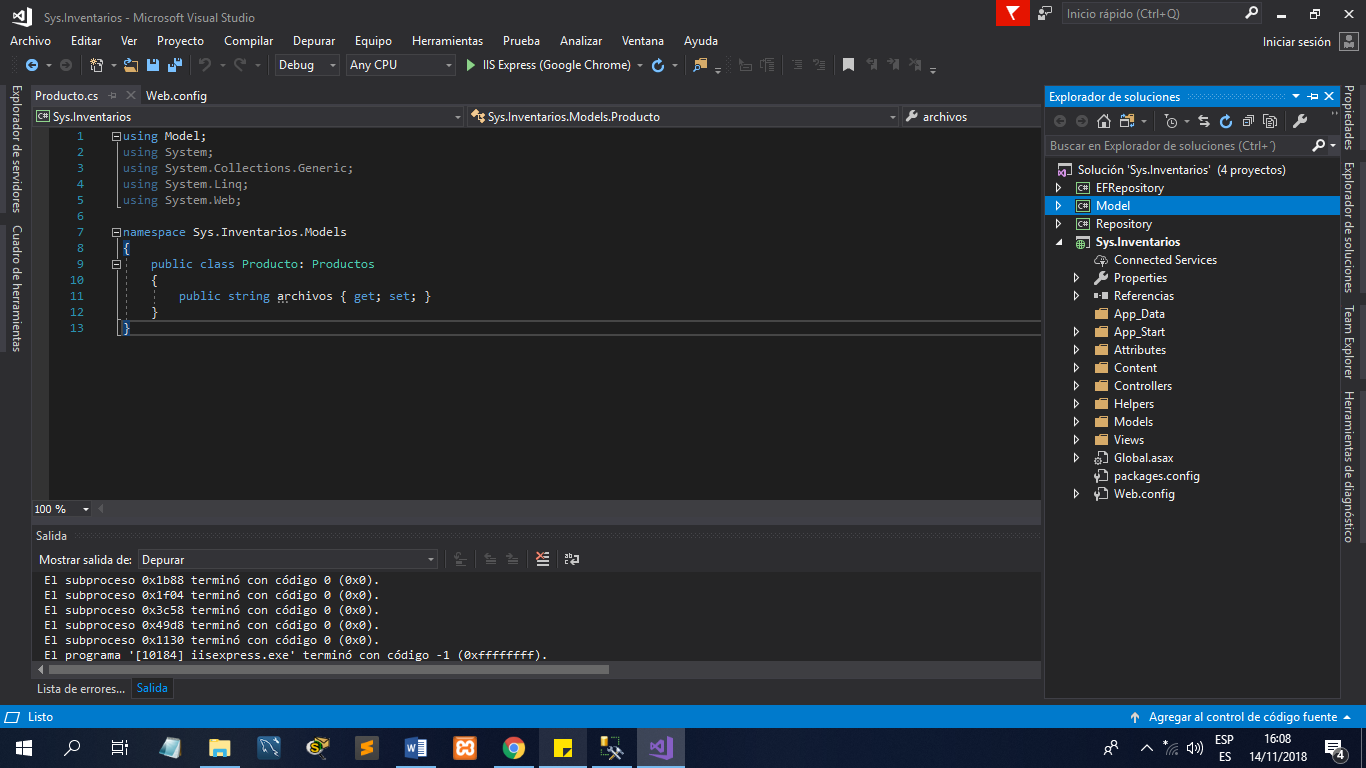
**Nota**: en el **App.Config** se creará la línea de conexión a la base de datos y en **DBEntity** estarán contenidas las tablas o entidades, cada modificación de campos en la base de datos alteran el flujo de datos en el sistema, no es necesario cargar la base de datos completa, bastará con modificar las propiedades en la entidad contenida en el Modelo y si continua el error se deberá eliminar y cargar nuevamente la entidad o tabla afectada.



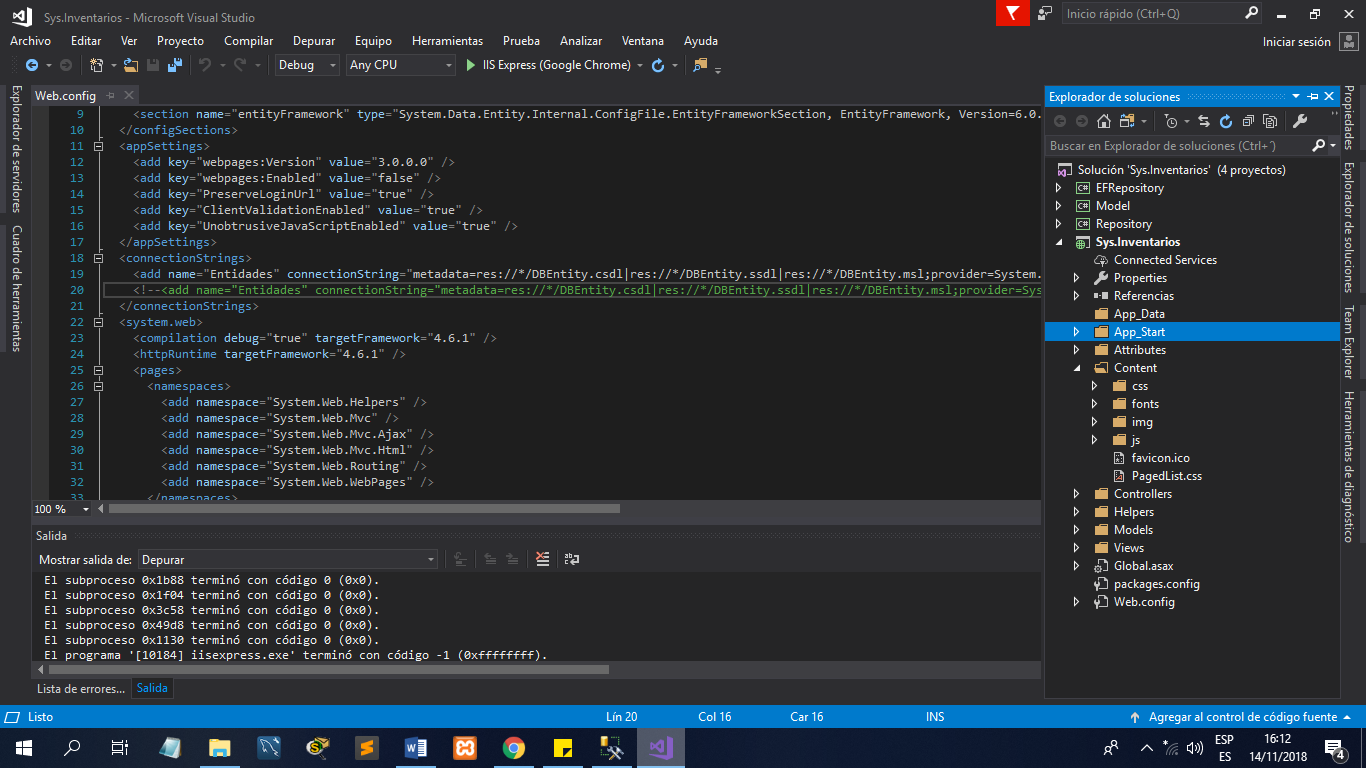
Para que los métodos tengan acceso a los datos contenidos en las entidades se deben agregar como en las referencias del Modelo.

Esta forma de trabajo nos permitirá poder re utilizar en un proyecto futuro la misma estructura modificando únicamente el nombre de la conexión que se creó a la base de datos en este caso “**Entidades**”

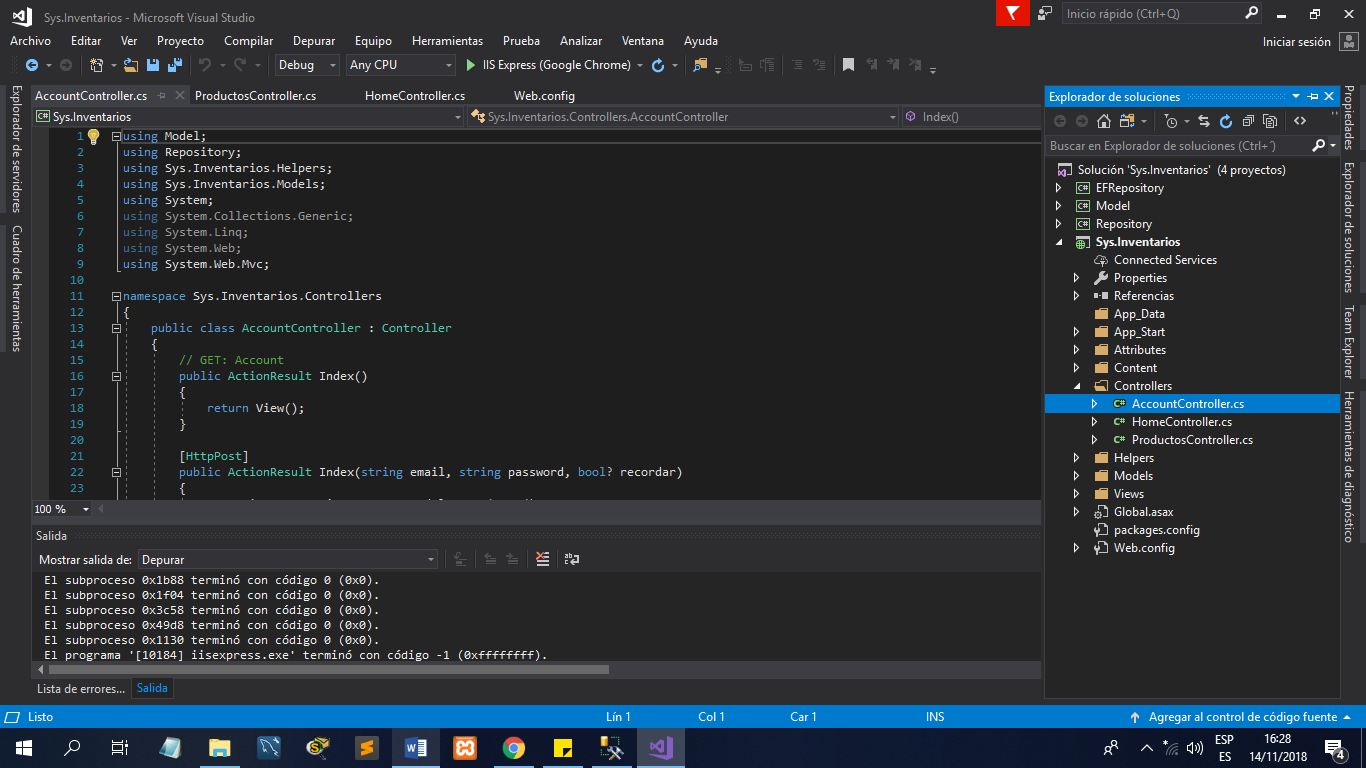
**Sys.Inventarios**: al momento de crear un proyecto MVC nos cargaran las siguientes carpetas para elaborar nuestro proyecto, adicional se crearon las carpetas Content y Helpers.



* Content: En esta carpeta se integran los CSS´s, las imágenes y los Java Script

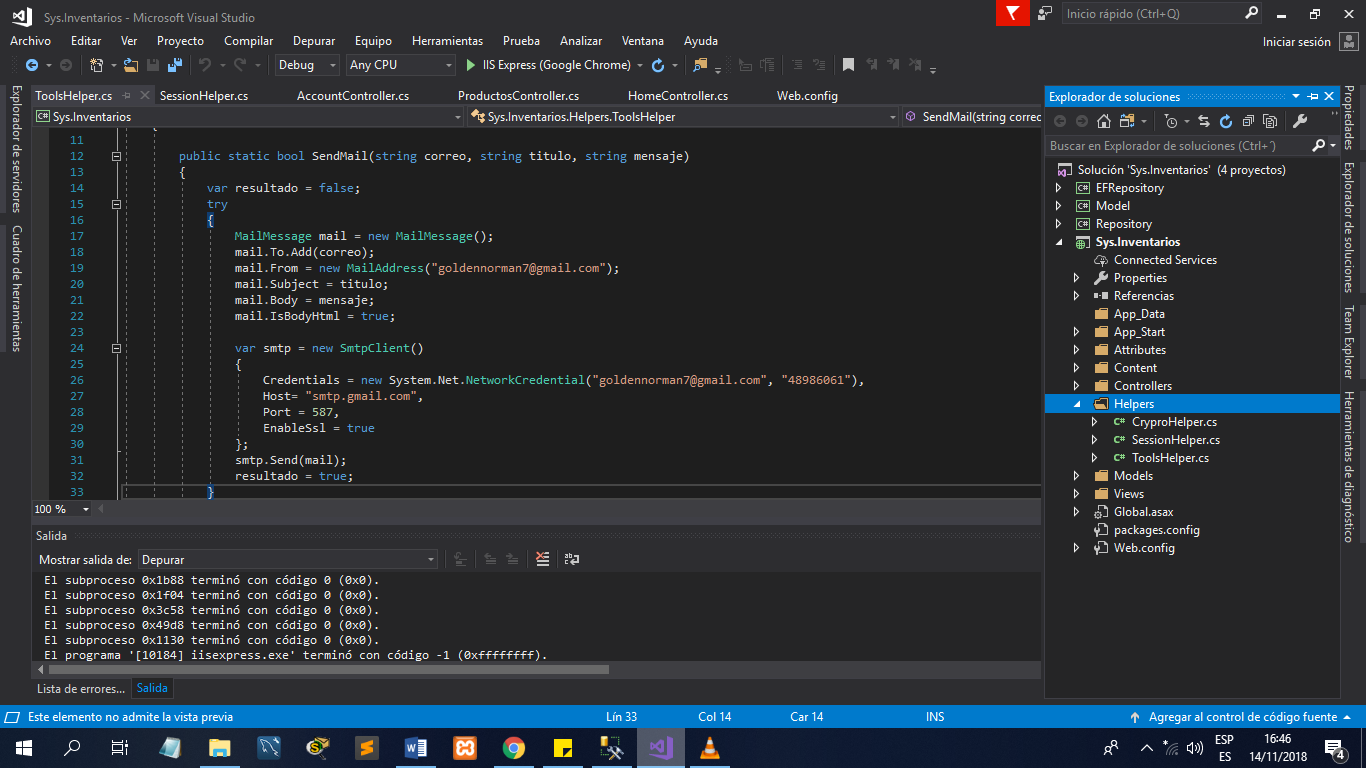


* Controllers: en los controladores se definen las reglas o la lógica de negocios y las llamadas a los modelos.

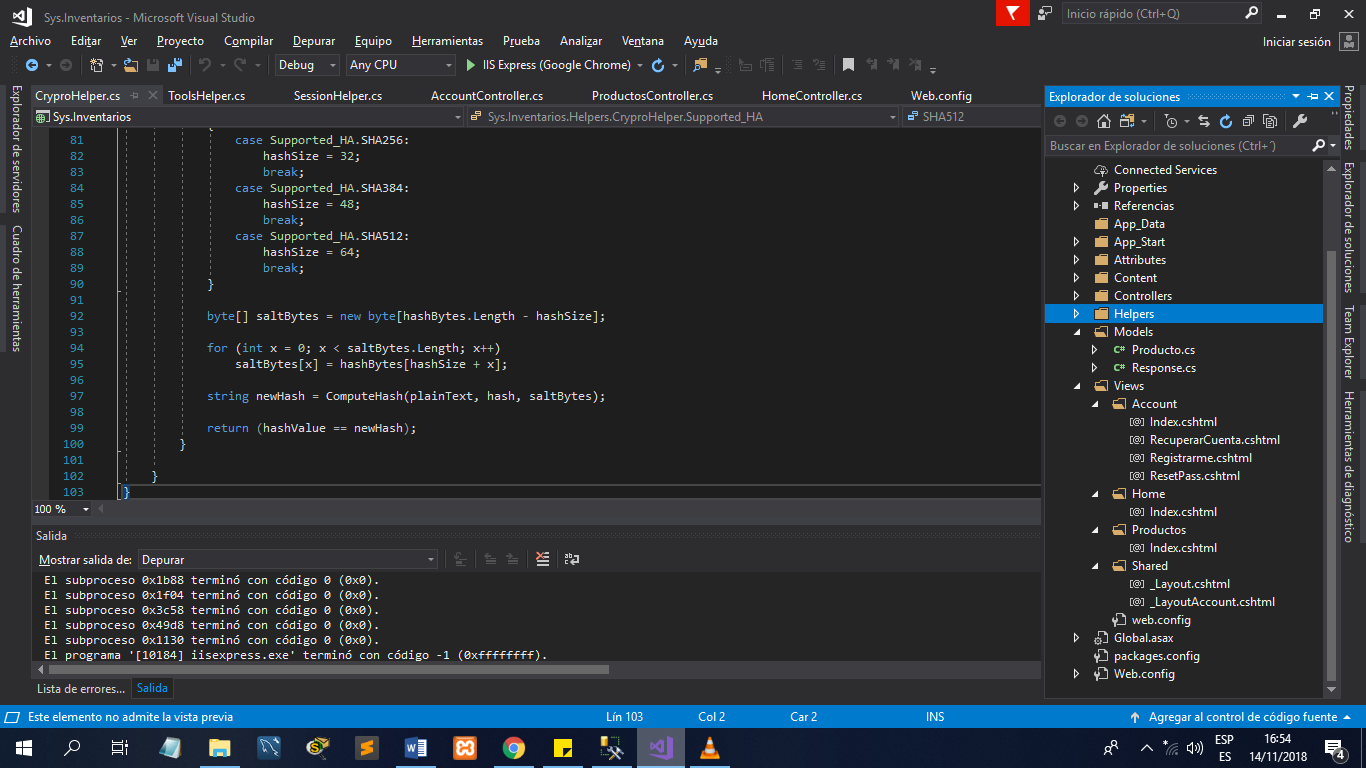


* Helpers: esta carpeta nos servirá para contener los inicios de sesiones, encriptar contraseñas, y enviar correos en caso de no recordar la contraseña, Actualizar sesiones y hacer verificaciones para insertar usuarios en caso de que ya sean existentes.

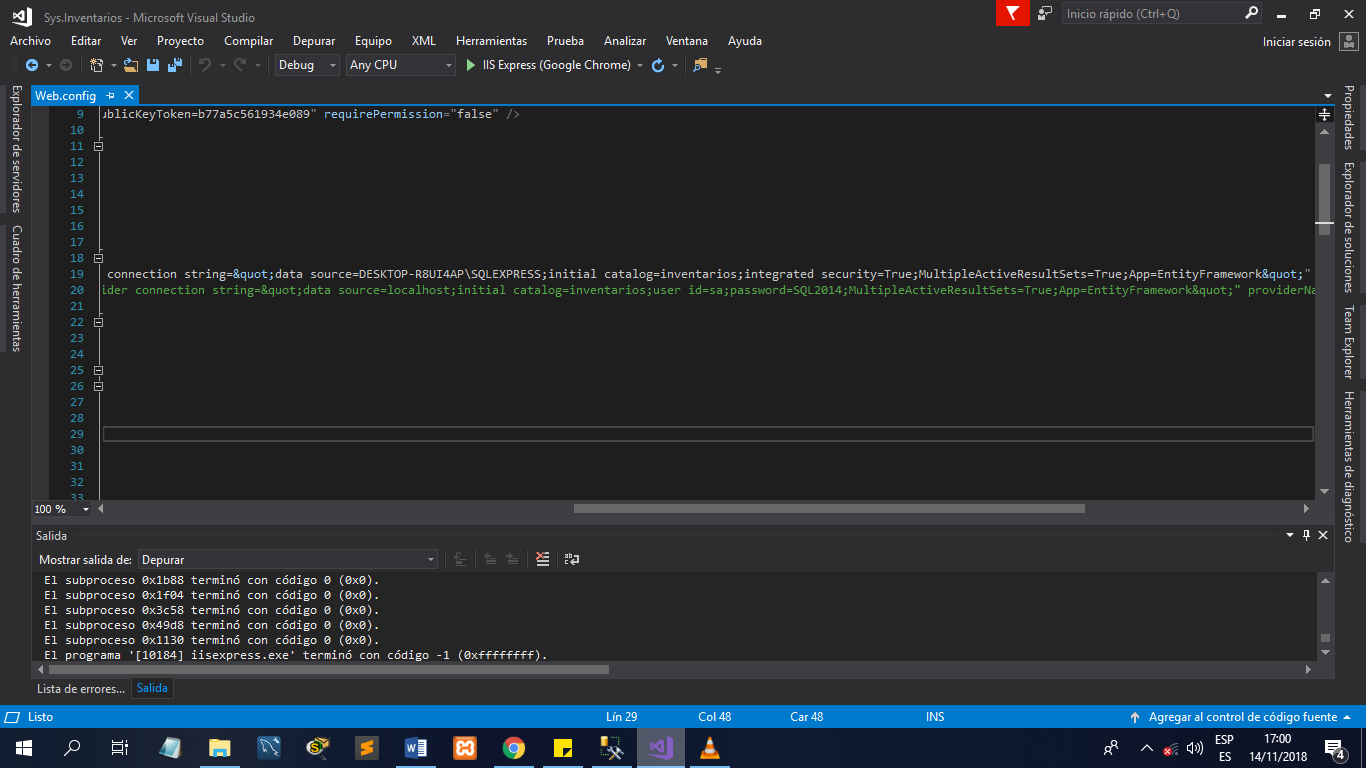
Se utilizó el método de encriptado mediante Hash en nuestro caso se utilizò el hash SHA256, SHA384, SHA512 contenidos en la clase CryptoHelper.



**Model y Views (Modelo y Vistas**): en el Modelo está la comunicación con la base de datos, todos los querys que se creen en el controlador pasarán por el modelo para llegar a la base de datos, y en las vistas están las interfaces mediante las cuales en usuario podrá interactuar con el sistema



**Nota:** en el caso de que la base de datos cambie de servidor o de lugar la cadena de conexión que se debe modificar es la que está contenida en el web.config que está en la carpeta principal del proyecto, no modificar la conexión contenida en la librería de clases Model.

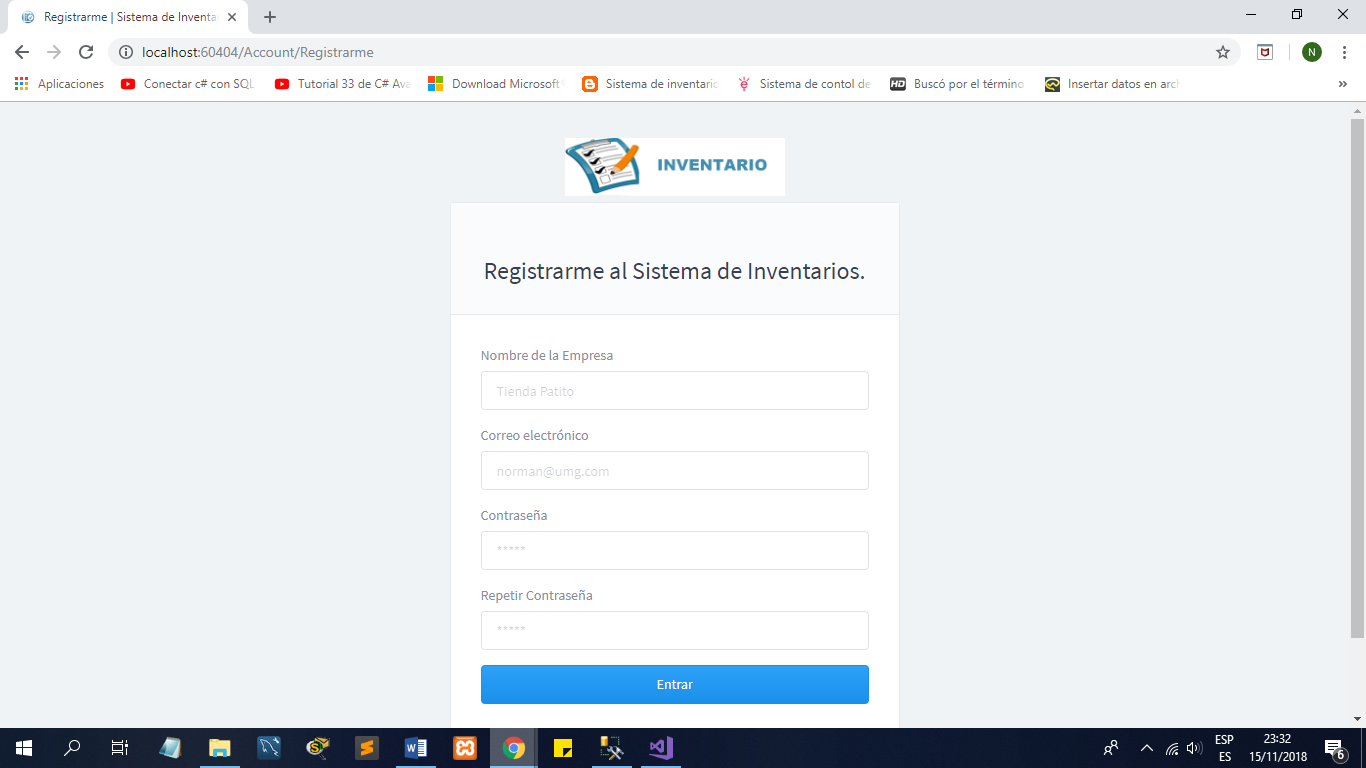


**Vistas**

* Logeo y Acceso al Sistema
* La primera vista es la ventana de acceso al sistema o Login, en el cual el usuario Administrador ingresa sus datos (Correo Electrónico y Contraseña).



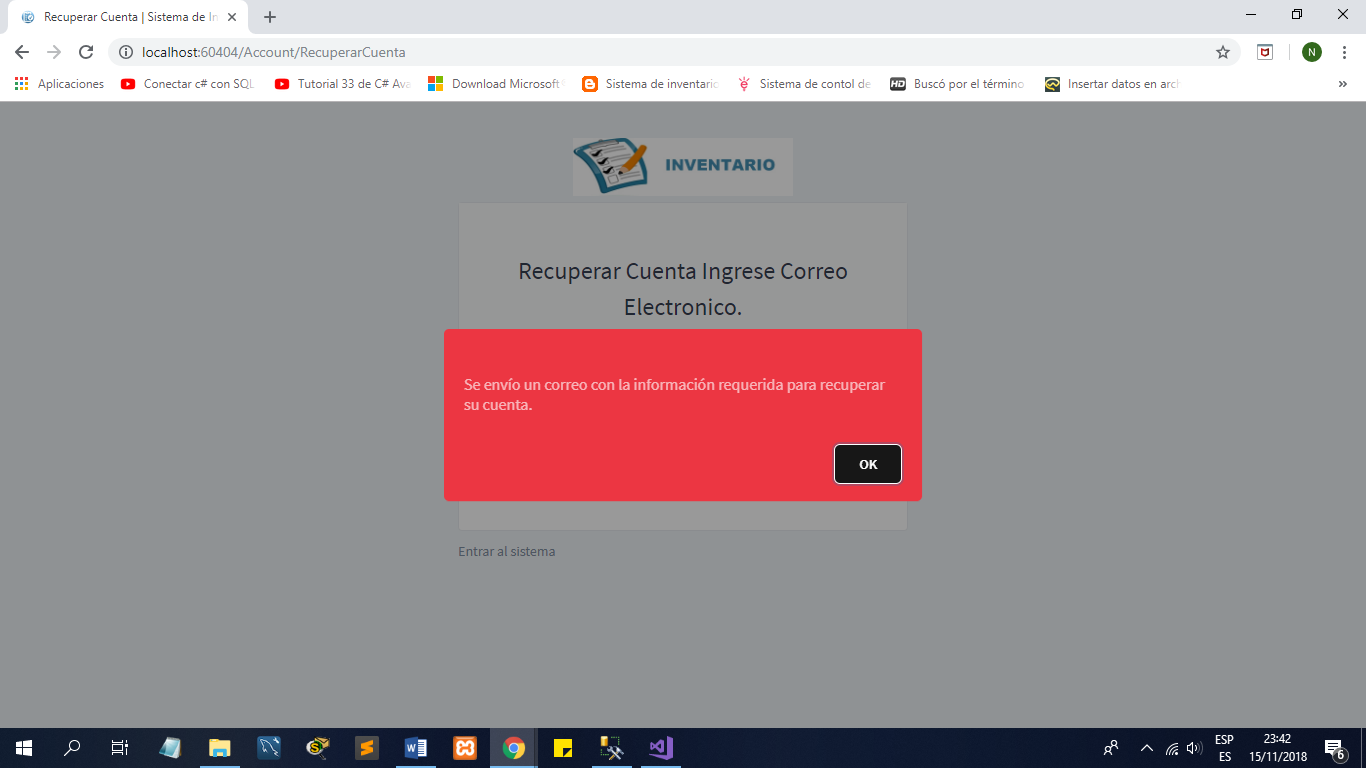
* + Nota: en caso de no estar registrado en la parte inferior de la pantalla está la opción de crear una cuenta, o en el caso de olvidar tu contraseña la opción también estará habilitada.
* Siguiendo con la vista de registro de usuarios en la cual permite registrar usuarios con el rol de Administradores que pertenezcan a la misma empresa y los proveedores de empresas terceras.



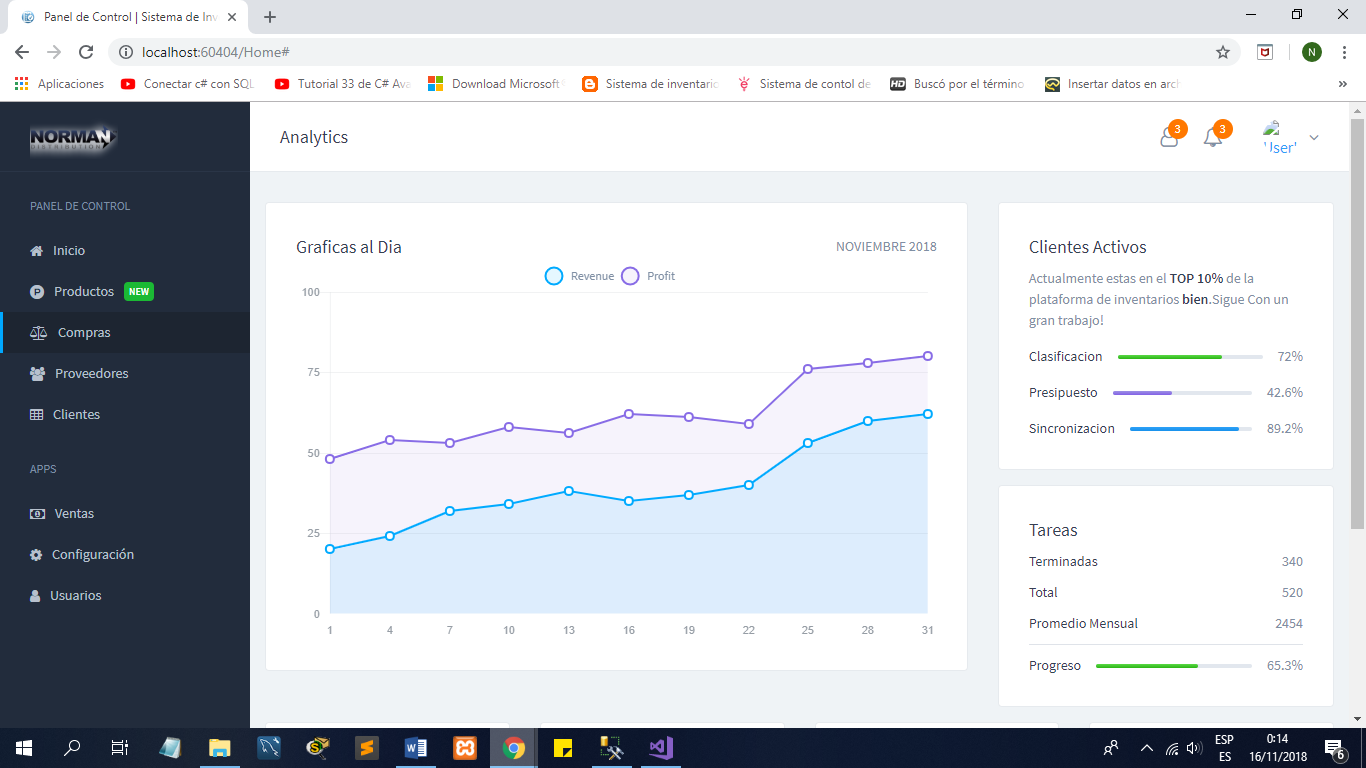
* En el caso de no recordar la contraseña el sistema permite crear una nueva mediante el correo electrónico, consiste en insertar el correo electrónico y al dar clic en el botón de enviar, estará llegando un correo al usuario en el cual estará el enlace para proceder a resetear el Password.



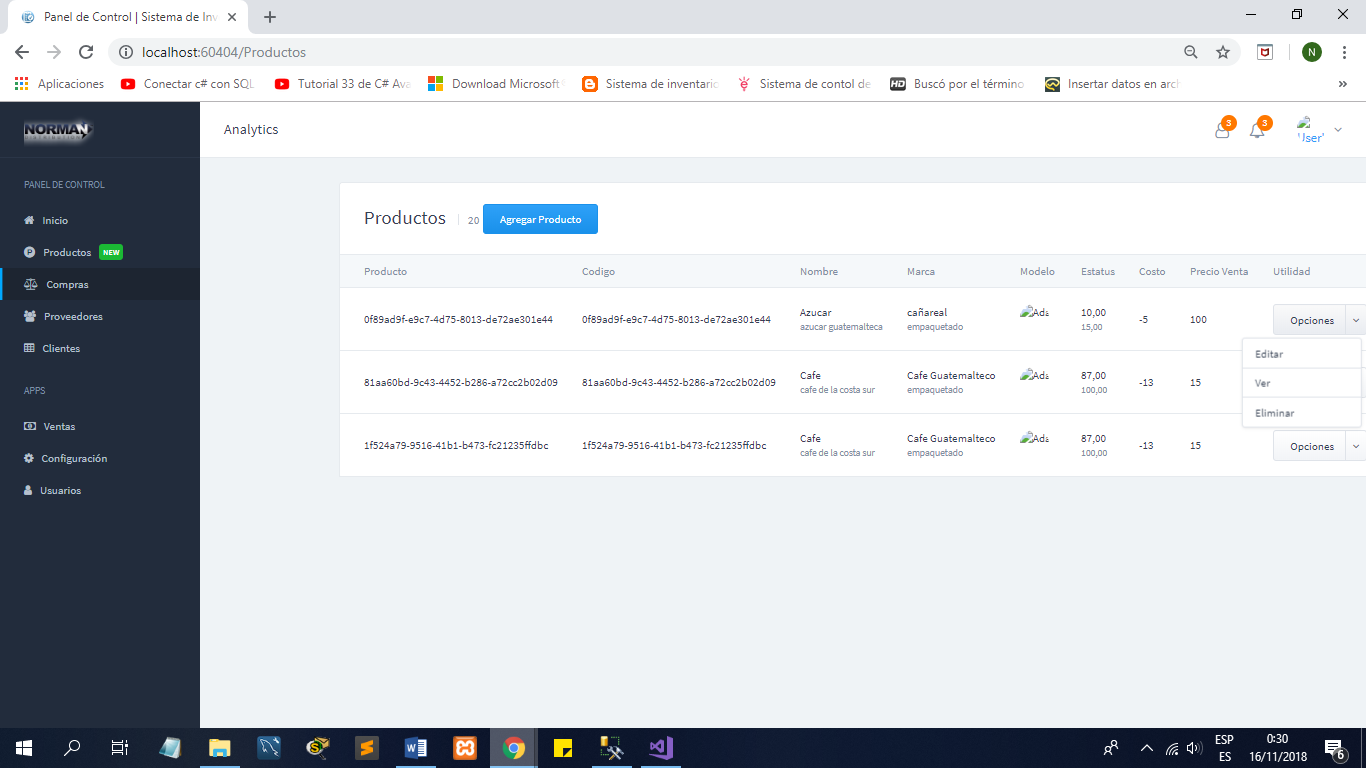
* Aparecerá un mensaje indicando que se envió el mensaje al correo correctamente como se muestra a continuación.



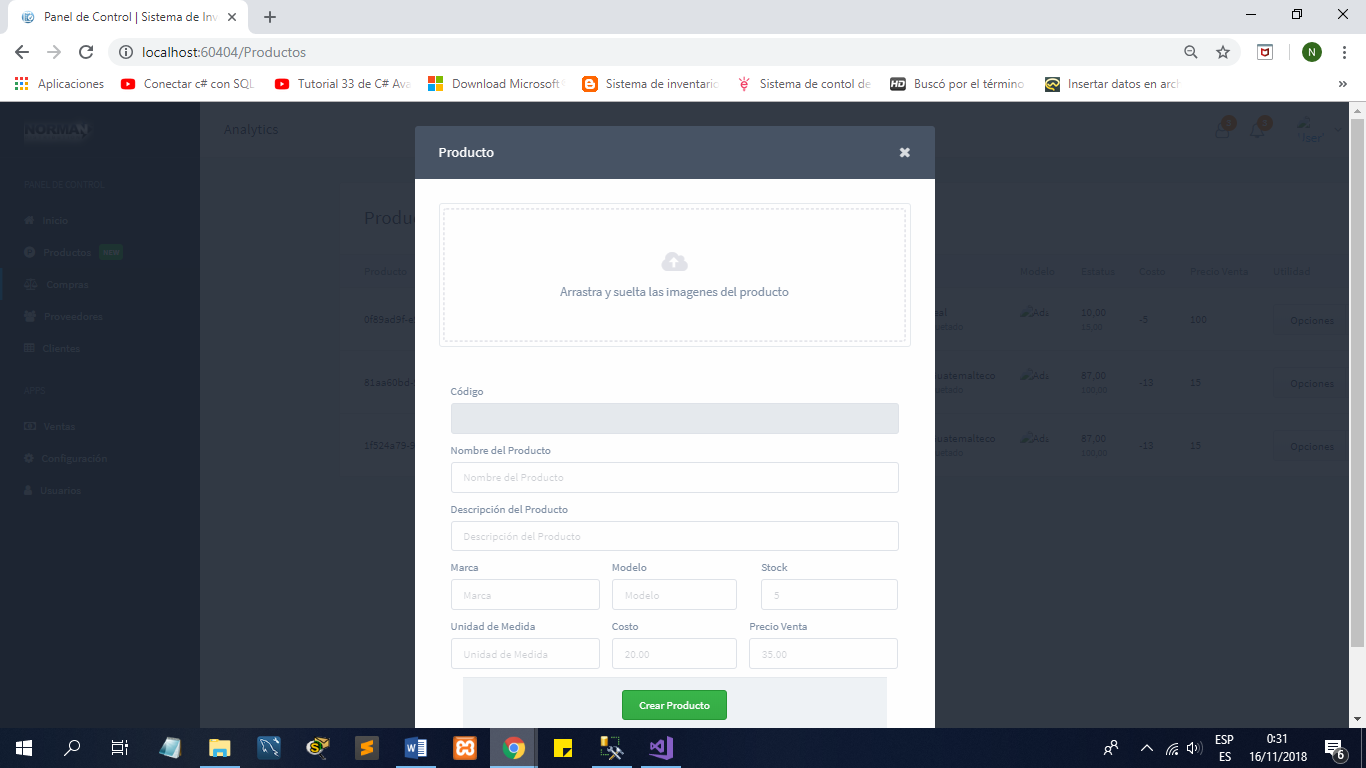
* Página Principal la cual cuenta con un menú de opciones y una serie de gráficas y resultados estadísticos.



* Módulo de Productos: se muestran los productos existentes

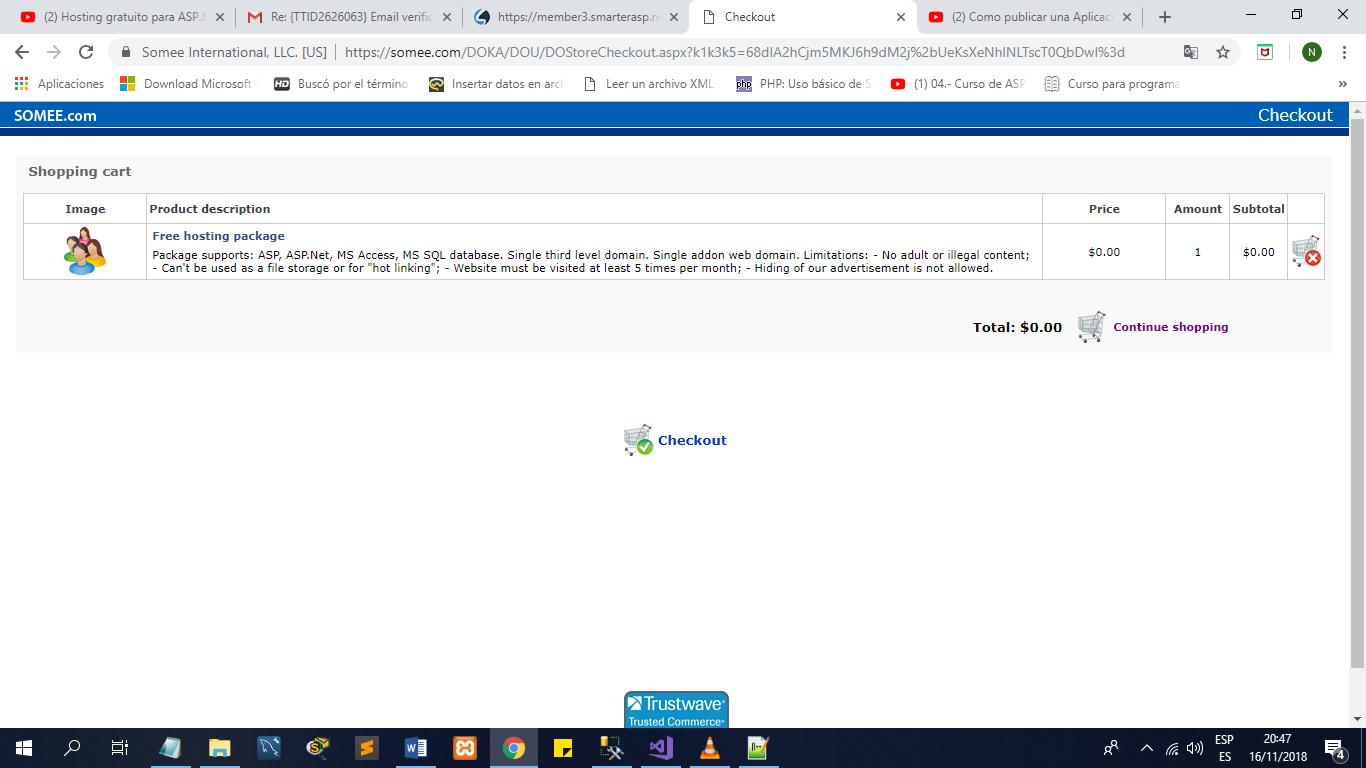


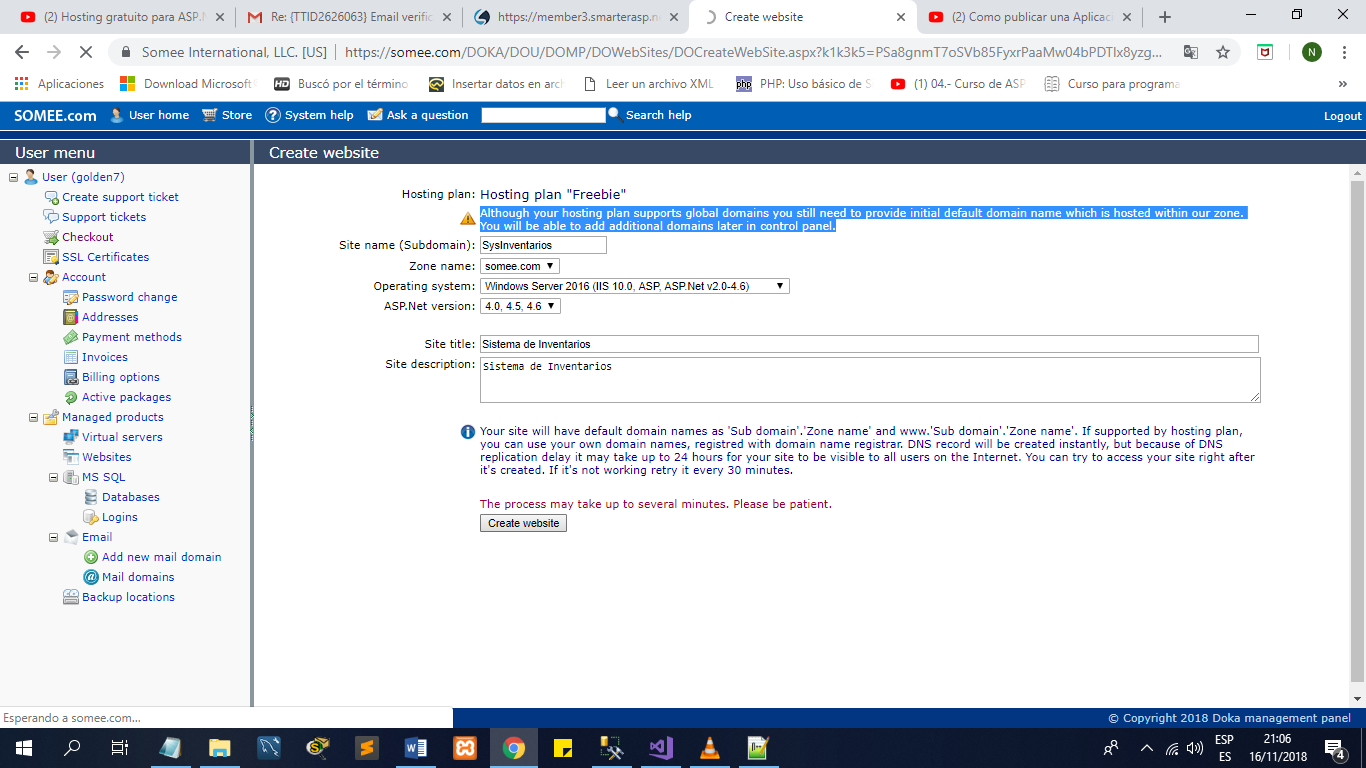
* Agregar producto: podemos agregar un nuevo producto al sistema



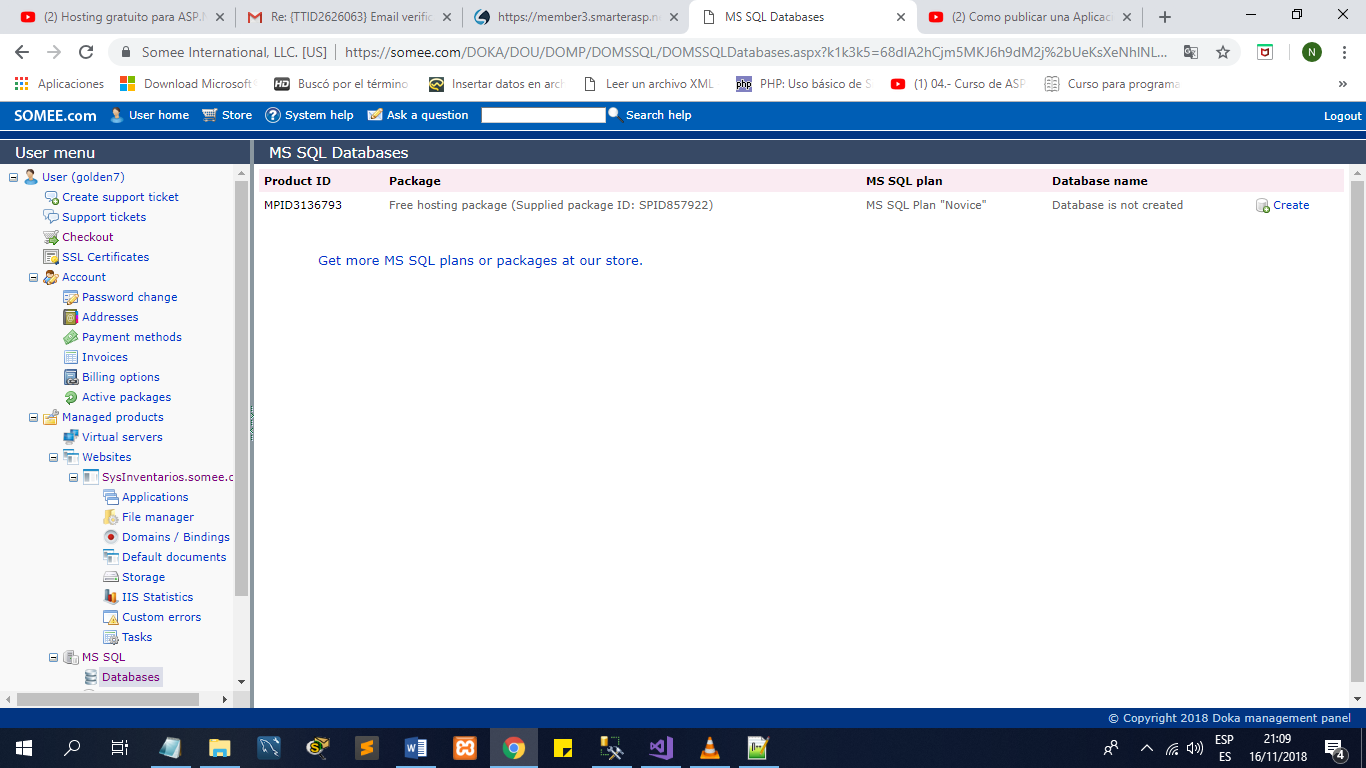
PUBLICAR PROYECTO

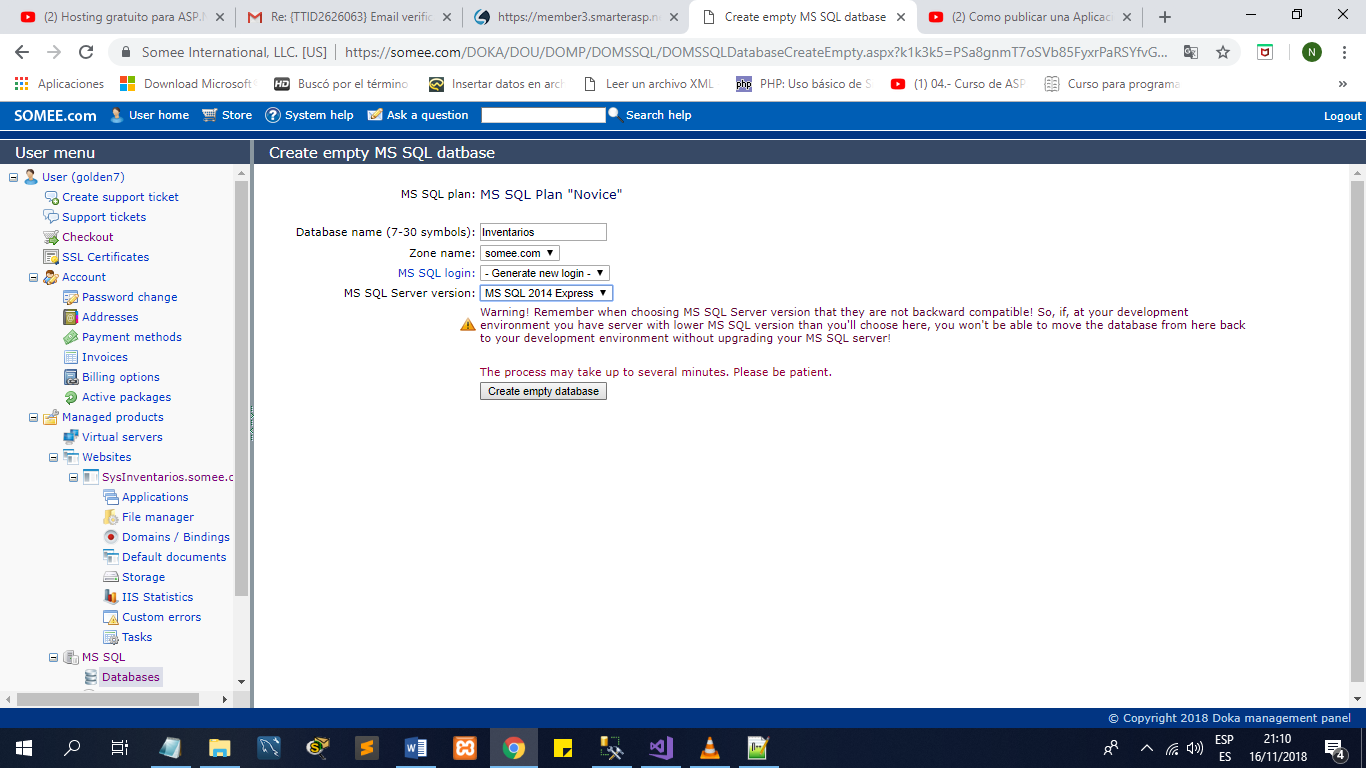
CREAMOS UNA CUENTA EN SOME HOSTING GRATUITO Y CREAMOS UN NUEVO PROYECTO Y DESDE VISUAL STUDIO VAMOS A PUBLICARLO



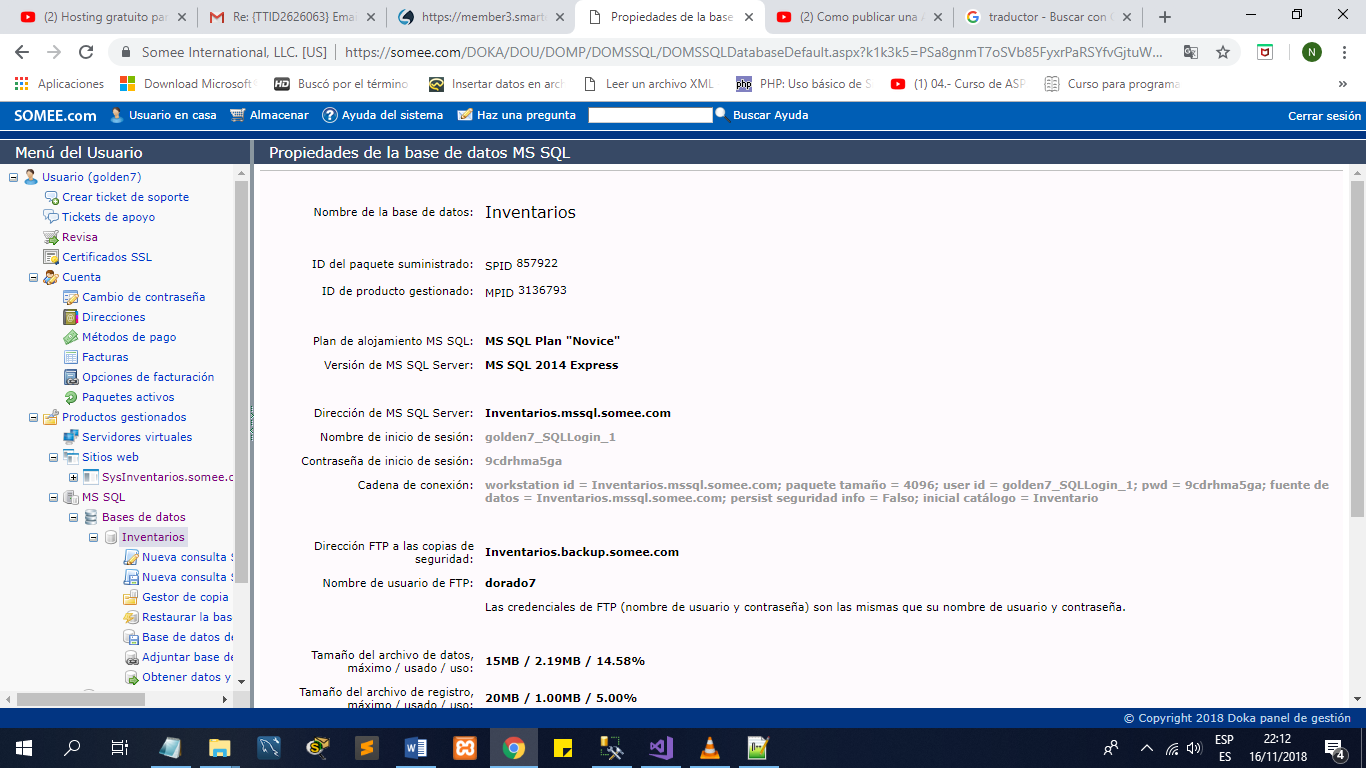


SE CREA UNA BASE DE DATOS Y UN SERVIDOR EN SOMME

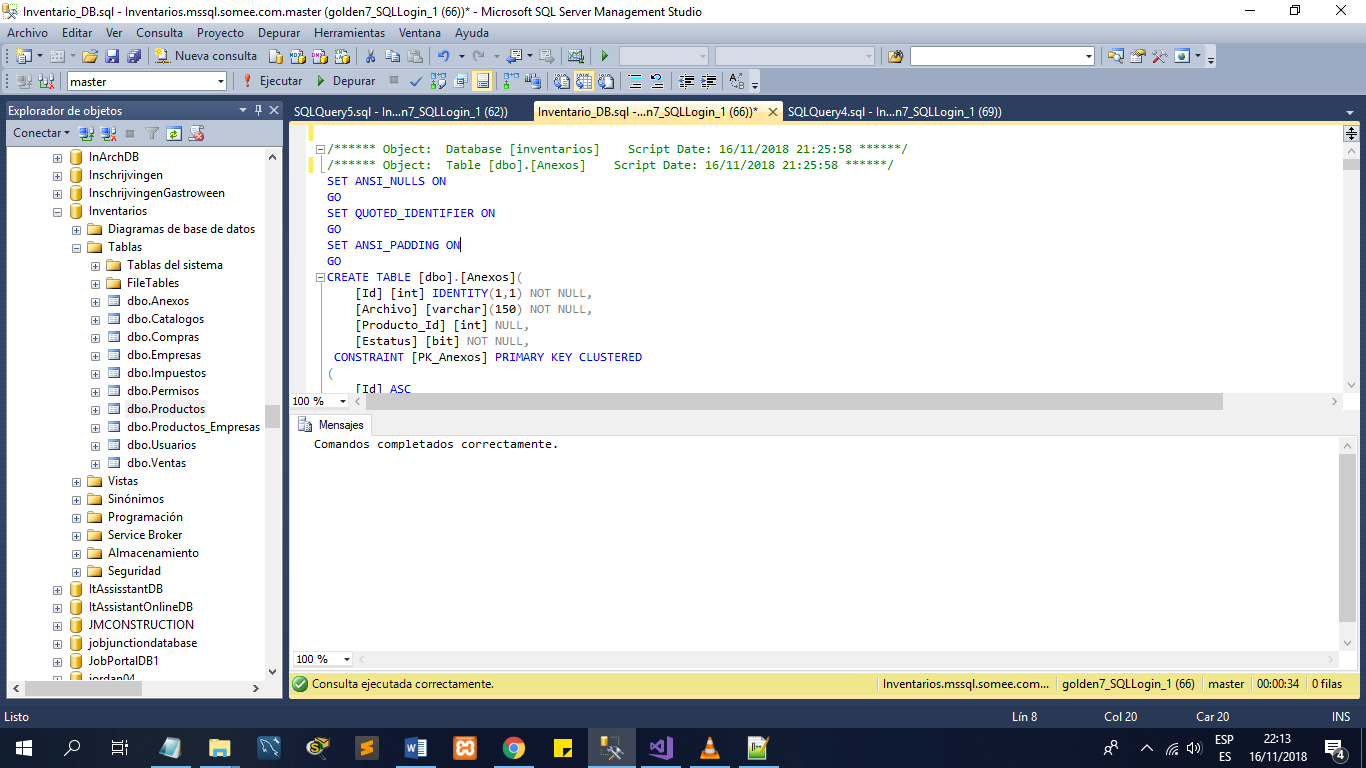




COPIAMOS LA DIRECCION DE FTP Y NOS CONECTAMOS AL SERVIDOR QUE CREAMOS EN EL SQL MANAGEMENT



PEGAR EL SCRIPT EN LA BASE DE DATOS EN LA NUBE



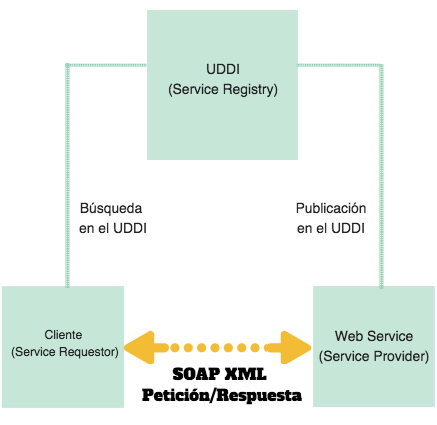
NOTA: PARA PUBLICARLO UTILIZAMOS FILEZILLA PARA PODER COPIAR LOS DATOS DE LA CARPETA PUBLICADA A LA CARPETA QUE CREAMOS EN LA NUBE MEDIANTE FTP

**Web Services**

Un Web Service, o Servicio Web, es un método de comunicación entre dos aparatos electrónicos en una red. Es una colección de protocolos abiertos y estándares usados para intercambiar datos entre aplicaciones o sistemas. Las aplicaciones escritas en varios lenguajes de programación que funcionan en plataformas diferentes pueden utilizar web services para intercambiar información a través de una red. La interoperatividad, por ejemplo, entre Java y Python o Windows y Linux se debe al uso de estándares abiertos.

Como sistema de mensajes se utiliza XML estandarizado. El protocolo más simple para el intercambio de información entre ordenadores es XML-RPC, que emplea XML para llevar a cabo RPCs. RPC, Remote Procedure Call, es un protocolo de red que permite a un programa a ejecutar código en una máquina remota. Los XML-RPC requests son una combinación entre contenido XML y headers HTTP. La simpleza de los XML-RPC hizo que el estándar evolucionase a SOAP, uno de los componentes básicos de los Web Services.

La base de comunicación entre web services es por tanto el lenguaje XML y el protocolo HTTP.



**Modelo Entidad Relación**

