Sistemas de Administración de Datos

El alumno debe de desarrollar un Proyecto de Software donde utilice los diferentes  
diagramas vistos en clase, desarrollo de Base de Datos y Programación desde Visual Studio a  
SQL Server.

Virtual tianguis

Punto de Venta (Proyecto Final)



Luis Gonzalo Soriano Crespo 145810

Christian Josué Acosta Vargas 390284

Marco Antonio Días Valdez

Contenido

[Introducción 1](#_Toc9709636)

[Resumen del Proyecto 4](#_Toc9709637)

[Descripción del Proyecto 4](#_Toc9709638)

[Objetivos del Proyecto 4](#_Toc9709639)

[Alcance del Proyecto 4](#_Toc9709640)

[Diagramas Para la Base de Datos 5](#_Toc9709641)

[Diagrama Entidad – Relación 5](#_Toc9709642)

[Diagrama Relacional 5](#_Toc9709643)

[Explicación de Funciones Específicas del Software Realizado 6](#_Toc9709644)

[Plan de Instalación 6](#_Toc9709645)

[Software de Escritorio 6](#_Toc9709646)

[Conclusiones 7](#_Toc9709647)

[Referencias 8](#_Toc9709648)

Introducción

Una base de datos es una colección organizada de información, generalmente almacenada en un sistema electrónico donde su acceso es a través de un sistema de cómputo. Para administrar una base de datos se necesita de un sistema de administración de base de datos (DBMS), un software que interactúa ya sea con los usuarios que lo utilizan, aplicaciones o con la misma base de datos para capturar y analizar la información.

Un software de este tipo es Microsoft SQL Server, el cual, es un sistema relacional de administración de base de datos (RDBMS) que es la base moderna de los lenguajes de administración de bases de datos (SQL) donde los datos se guardan en objetos de información denominados tablas, las cuales a su vez, contienen datos del mismo tipo agrupados en fila (los datos dentro de una tablas se dividen en filas) y columnas (nombran la información específica que contiene la fila); este tipo de sistema es usado en varias aplicaciones como procesos de transacciones e inteligencia de negocios.

Microsoft SQL Server está escrito a partir del lenguaje SQL (Lenguaje estructurado de peticiones) que es un lenguaje estándar para acceder y manipular bases de datos, aunque tiene sus variantes dependiendo del DBMS que lo ejecute, SQL Server se encuentra basado en T-SQL, una variante de SQL hecha por Microsoft que añade una serie de extensiones (como comandos) de programación propias al lenguaje base.

Un entorno de desarrollo integrado (IDE) es una aplicación de software que contiene herramientas y entornos visuales de desarrollo hecho para los programadores con el objetivo de facilitar el desarrollo de software. Un IDE tiene incluido normalmente un editor de código fuente, herramientas para la compilación automatizada del código fuente y un depurador de errores.

Visual Studio es un IDE hecho por Microsoft donde puedes editar código fuente, depurarlo y compilarlo para después lanzar al mercado la aplicación que ha desarrollado el programador. Visual Studio incluye compiladores, auto corrector y auto completado de código, entornos para diseños gráficos entre varias otras herramientas para facilitar el desarrollo de software. Es un IDE disponible tanto para Windows como para MAC con tres ediciones diferentes: Community, Professional y Enterprise.

Se le conoce como punto de venta a un lugar físico o conceptual donde existe una relación entre un consumidor y las marcas o compañías que ofrecen sus productos al consumidor y desea comprar, es básicamente en donde los productos se ponen a la vista para que la persona decida que comprar. El punto de venta es una parte altamente impórtate para las empresas puesto que el 80% de las decisiones de compra se realizan por este medio y, por tanto, debe de ser dominado al margen para que las marcas y/o empresas puedan subsistir.

Existen varios tipos de puntos de venta de los cuales mencionaremos a algunos:

• Retrailers: son tiendas u almacenes que ofrecen una gran gama de productos de diversas marcas y tipos como lo podrían ser los supermercados o los almacenes de ropa

• Pop up stores: Son tiendas que se montan en uno o varios días en específico y su permanencia es a corto plazo para desaparecer en días posteriores, un ejemplo de este tipo de puntos de ventas son los tianguis.

• Vending machines: Consiste en colocar refrigeradores o dispensadores para vender un producto que necesite refrigerio o preparación antes de ser distribuido al cliente donde este ingresa a la maquina dinero para después seleccionar el producto de su preferencia, los ejemplos más típicos son las maquinas dispensadoras de refrescos o las máquinas para preparar café.

• E-commerce: Consiste en vender productos por medio de una pagina de internet y los productos pueden ser virtuales como un videojuego o un álbum de canciones hasta físicos como toallas o drones, ejemplos de este tipo de punto de venta son amazon, mercado libre o hasta un grupo de Facebook de compra y venta

• Móviles: Se les puede clasificar como mercados en forma de vehículo o con ruedas que ofrecen productos y su lugar de venta no depende de un punto en especifico si no que se pueden desplazar conforme a sus necesidades, la diferencia entre un punto de venta móvil y un Pop up store es que la segunda se necesita montar primer para poder operar y su traslado no tiene mucho alcance mientras que la primera el traslado es fácil y pueden empezar a operar en cuanto se estacionan en un lugar, un ejemplo de este tipo son los puestos de tacos.

Durante el desarrollo de esta aplicación de punto de venta, donde reuniremos los requisitos que debe de cubrir la aplicación para que funcione como un portal de transacción de bienes entre el vendedor, la marca y, el consumidor, usaremos Visual Studio Community y SQL Server 2008 R2; el proyecto se editará y guardará en un repositorio privado de GitHub que sirve para realizar operaciones de desarrollo de software en tiempo real entre los programadores al igual de mantener un historial de las versiones de la aplicación a desarrollar, para realizar operaciones dentro de este repositorio usaremos GitHub Desktop.

i. Explicar que es la Programación por Capas.

En ingeniería de software, la programación por capas es un servidor entre un usuario y una base de datos donde su presentación en pantalla, procesamiento de la información y las funciones de la administración de la información se encuentran físicamente separadas unas de otras; el tipo de programación mas usado es el de las tres capas, pero se puede trabajar un modelo que utilice dos o más de tres capas. Una capa es un patrón arquitectónico dentro de la escritura de un código.

Con este tipo de modelo la programación resulta ser mas flexible a cambios puesto que se puede trabajar una capa independientemente de las otras, agregado o modificando funcionalidades sin tener que cambiar el código por completo, además, una o varias capas pueden ser reutilizadas en otros proyectos de desarrollo.

La programación de tres capas contiene las siguientes capas:

• Capa de presentación: Es el nivel más alto de la aplicación, arquitectónicamente hablando. Muestra la información relacionada a servicios como mecanismo de búsqueda, obtención y manipulación de datos. Se comunica con las otras capas para desplegar la información que el usuario/cliente pide y se envíen o realicen los comandos correctos. Es básicamente una interfaz de usuario grafica

• Capa de negocios (lógica, intermediario, etc.): Se encarga de procesar la información que recopila de la capa de datos y enviársela en un formato entendible a la capa de presentación dependiendo de las peticiones recibidas en la capa de presentación.

• Capa de datos: Incluye los mecanismos de persistencia de datos (como los servidores de bases de datos), y las capas de acceso de datos que encapsula los mecanismos de persistencia y exposición de los datos. La capa de acceso a datos debe de ser provista con una API de administración de datos almacenados para la capa de negocios, con ello, no se necesitarán de dependencias para mostrar los datos que se encuentran almacenados en los mecanismos de persistencia de datos (ejemplo: que la aplicación necesite instalar SQL Server para poder funcionar).

# Resumen del Proyecto

Descripción del Proyecto

Es un Software creado especialmente para agilizar los procesos de venta donde se interactúa directamente con el cliente, permitiendo realizar transacciones diarias y obtener información clave de todas las operaciones registradas para una toma de decisiones correcta.

## Objetivos del Proyecto

Implementación rápida: Seleccionar un sistema que sea muy sencillo y puedas empezar a operar en minutos y que puedas configurar tu mismo.

Fácil de usar: Debe tener una Interfaz amigable y fácil de usar. Que solo requieras unos minutos de capacitación y entrenar a tu personal será sumamente sencillo.

Interfaz amigable: Diseño intuitivo para mejorar la experiencia del usuario. Interfaz amigable y funcional a través de menús que optimizan la navegación en el sistema

Altamente Configurable: Desarrollado para adaptarse a las necesidades de tu negocio sin importar su giro comercial. ¡Un Sistema que crece con tu negocio!

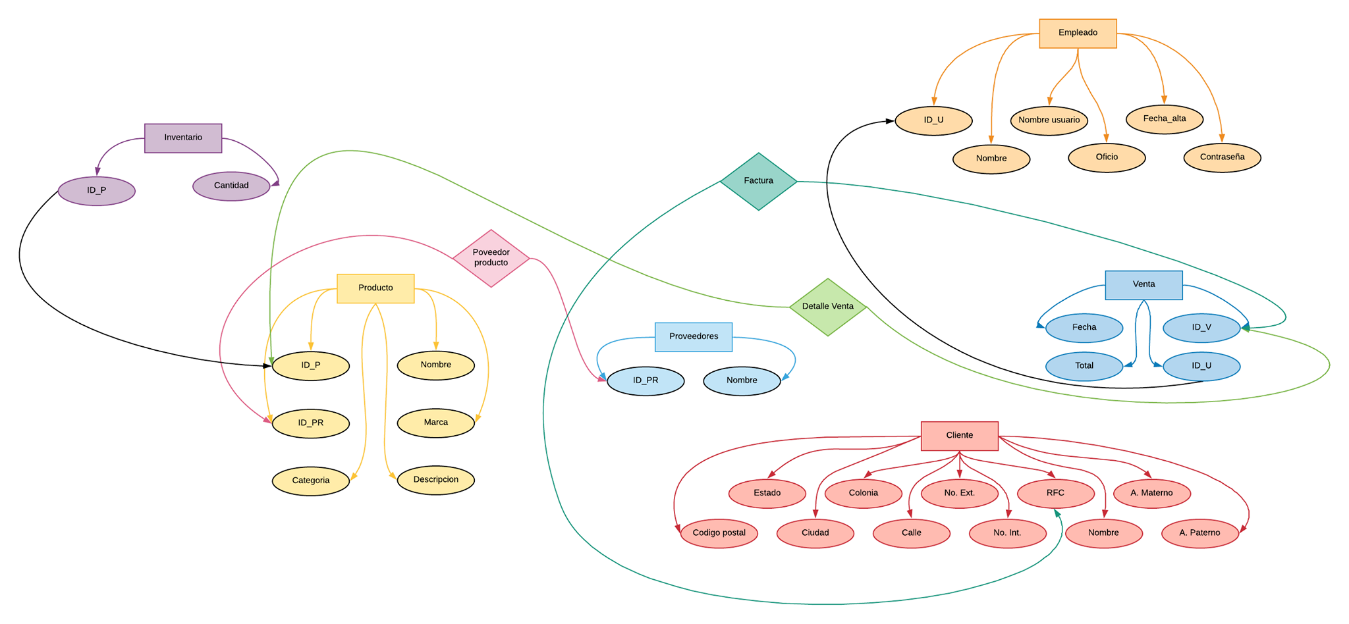
Seguridad Total: Puedes crear múltiples perfiles (gerente, supervisor, vendedor, etc) para controlar accesos y resguardar tu información.

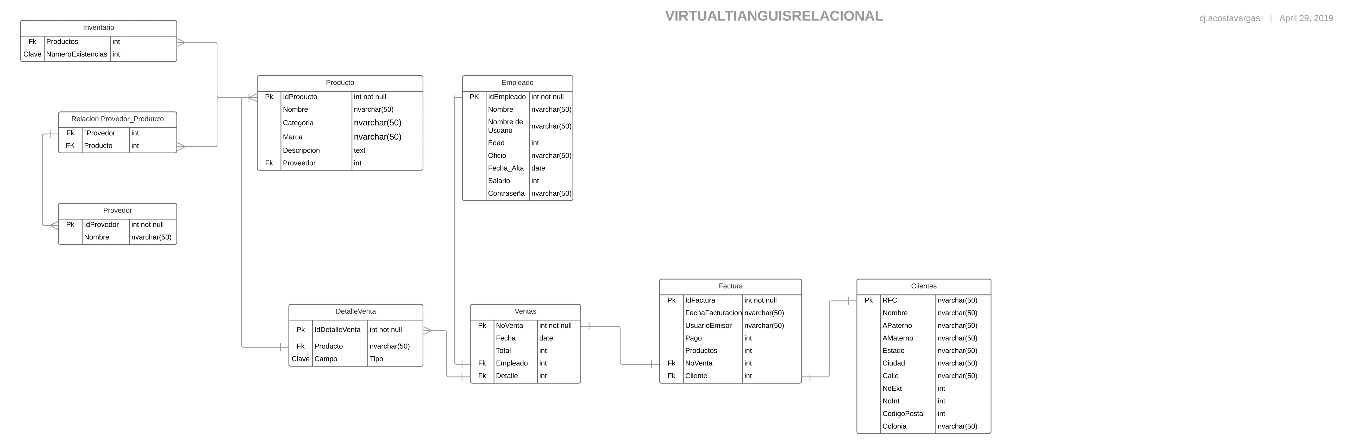
## Alcance del Proyecto

El proyecto va dirigido principalmente a los negocios que cuenten con una administración que tenga distintos tipos de puestos como un Gerente, un supervisor y varios vendedores y otros como personal de seguridad, limpieza, mantenimiento. Puede adaptarse a pequeños negocios como abarroteras o mini super en particular, tiendas de segunda más administradas ( que es la temática de nuestro proyecto), Ventas de casa de empeño, saldos, etc.

# Diagramas Para la Base de Datos

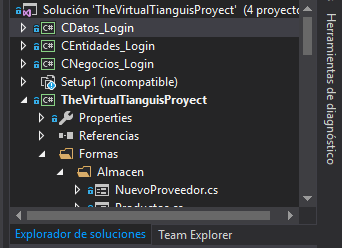
## Diagrama Entidad – Relación



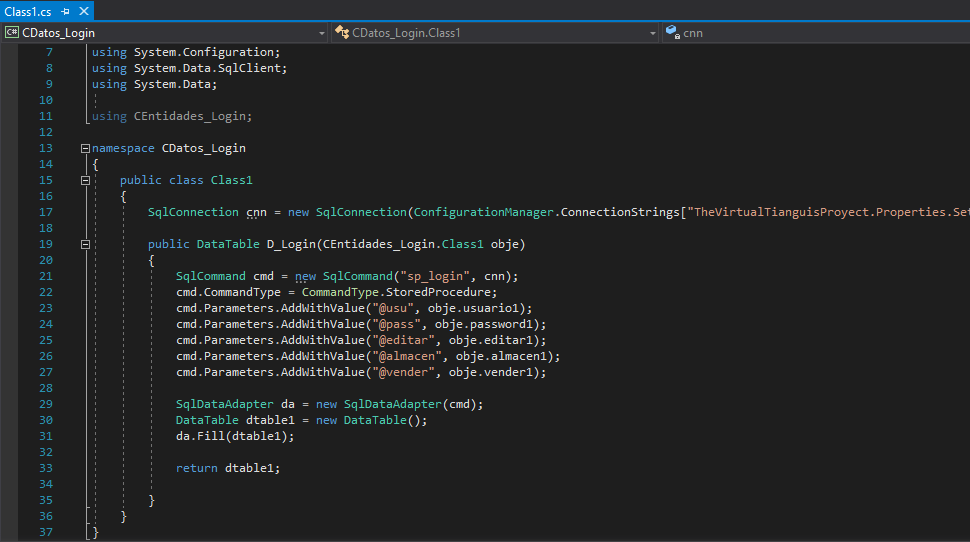
Diagrama Relacional

# Explicación de Funciones Específicas del Software Realizado

* Login

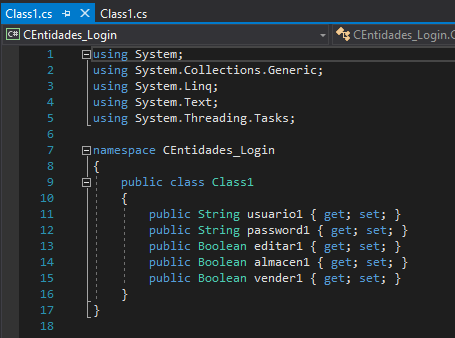


Para la función de Login creamos 3 clases dentro del proyecto/solución de Visual Studio: CDatos\_Login, CEntidades\_Login, CNegocios\_Login como se muestra en la imagen.

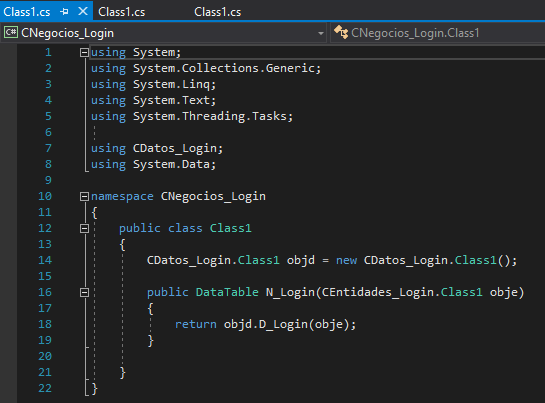


La clase CDatos\_Login contiene lo que es la conexión a la base de datos la cual se invoca con el comando SqlConnection y selecciona la conexión que esta cargada a la variable goblal del proyecto.

En seguida manda llamar un proceso llamado “sp\_login” que se encuentra en la base de datos la cual no arroja el nombre de usuario, la contraseña y 3 permisos llamado “editar”,”almacen” y “vender” estas variables booleanas se guardan en variables globales que servirán para reconocer que tipo de usuario esta iniciando sesión y que acciones tiene permitas hacer dentro del programa.



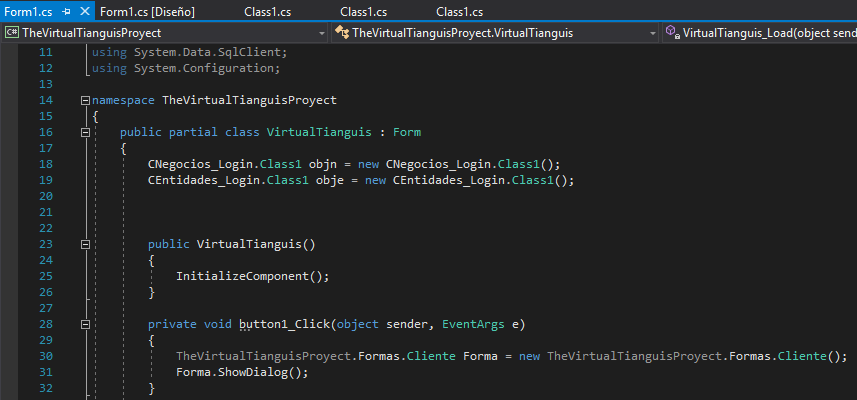
La clase CEntidades\_Login guarda lo que son las variables en las que se guardaran los datos arrojados por el proceso “sp\_login”.



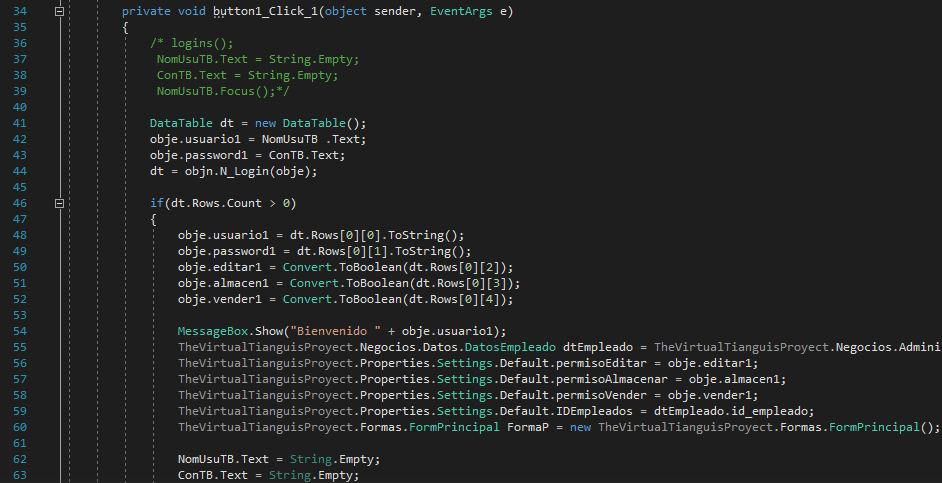
La clase CNegocios\_Login se encarga de hacer un objeto de tipo class1 de la capa CDatos\_Login y entonces crea un DataTable las cuales recibe los parámetros de todas las variables de CEntidades\_Login, y nos arroja esa tabla.



Interfaz de Login.

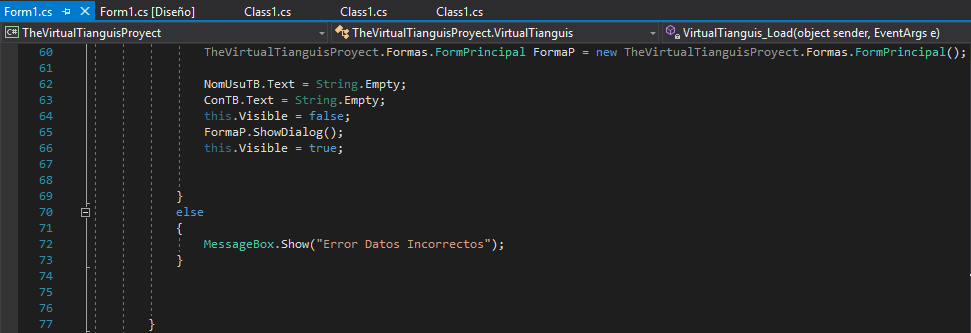


Dentro de la form se crean dos objetos tipos CNegocios\_Login y CEntidades\_Login.



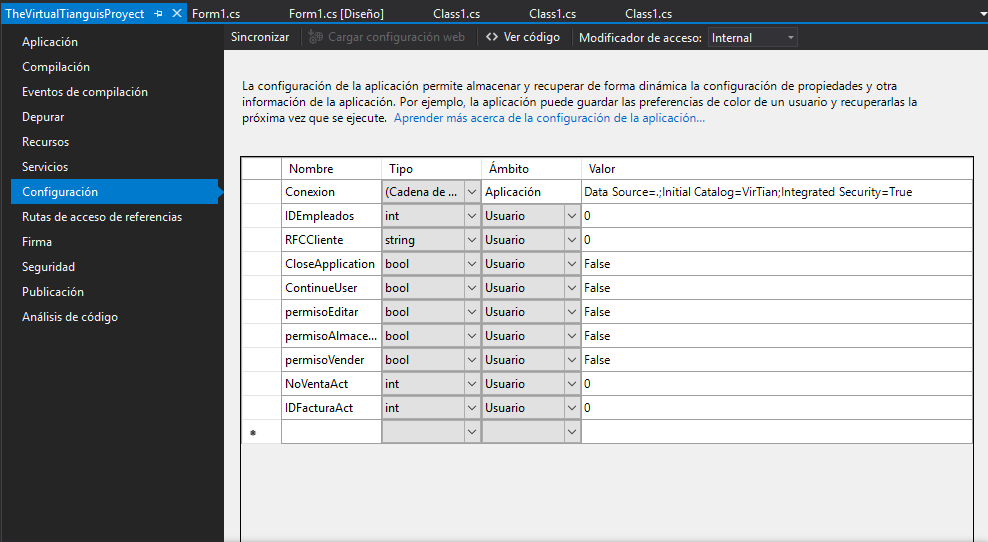
Al dar click al botón de iniciar sesión se crea una datatable y se guarda el nombre de usuario y contraseña de los text box correspondientes, entonces igualamos nuestra datatable al objeto de CNegocios\_Login que contiene la tabla cargada con los datos cargados en caso de que el usuario y la contraseña sean correctas debido al proceso que se encuentra en el query.

Entonces se empiezan a guardar esos datos de las tablas a las variables del objeto de CEntidades\_Login, a continuación te muestra un mensaje de bienvenido y el nombre de usuario guardado y en seguida se guardan los permisos de ese usuario logeado y te abre la form principal ya para empezar a hacer acciones.



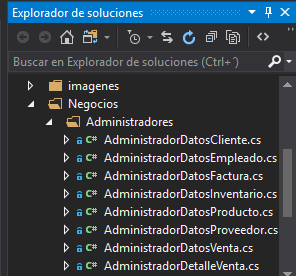
Los text box se vacían y se oculta la form de login para que no se este viendo todo el tiempo mientras interactua con la ventana abierta, una vez cerrada la ventana principal se vuelve a ver la ventana de login por si quiere iniciar sesión con otra cuenta.

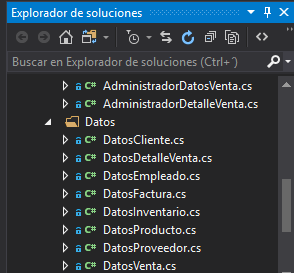
* Variables Globales



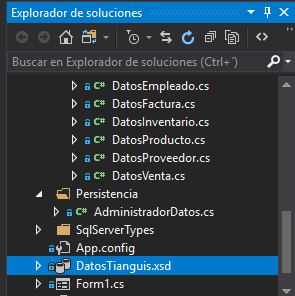
En esta sección de Configuración del Proyecto se pueden asignar las variables globales de todo el proyecto en la cual se encuentra la conexión para la base de datos, IDEmpleados, RFC Cliente, cerrar la aplicación, los permisos y entre otros que fueron de gran utilidad para tener un facilidad de manejar en el código.

* Capas

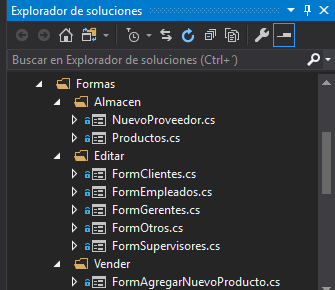


El proyecto se trabajo en capas, donde se encuentra Negocios, Administradores, Datos y Persistencia.

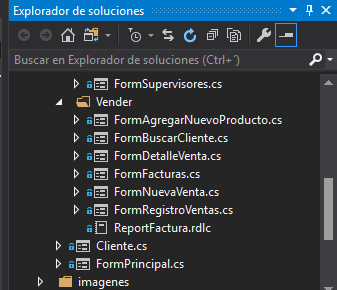
Aquí se muestran todos los archivos .cs para la Capa de Datos.



Dentro de Persitencia se encuentra el archivo AdministradoDatos.

****

Dentro de la carpeta formas se encuentras todas las forms incorporadas al proyecto como se ve en la imagen para tener un mejor orden.

****

La mayoría de funciones específicas del software están en la capa negocios la cual cuenta con la carpeta administrador de datos y datos.

Una de ellas sería una función que sea capas de extraer información de una tabla de la base de datos.

**Función ObtenerDatosProducto.**

1. **public** **static** TheVirtualTianguisProyect.Negocios.Datos.DatosProducto ObtenerDatosProducto(System.**Double** IDProducto)
2. {
3. **String** ConnectionString = **String**.Format(TheVirtualTianguisProyect.Properties.Settings.**Default**.Conexion);
4. **String** query = "select \* from Producto where Id\_producto = " + IDProducto;
6. DataTable Dt = TheVirtualTianguisProyect.Persistencia.AdministradorDatos.ExecuteQuery(ConnectionString, query, "Producto");
7. **if** (Dt.Rows.Count == 0)
8. {
9. **return** **null**;
10. }
12. **return** [new](http://www.google.com/search?q=new+msdn.microsoft.com) TheVirtualTianguisProyect.Negocios.Datos.DatosProducto(Dt.Rows[0]);
13. }

Esta función necesita recibir un valor con el cual haremos referencia a la tabla productos, en este caso se necesita un recibir una variable double con la cual accederemos a una fila.

El primer paso es crear una variable string en la cual se almacena la dirección de la conexión.

El segundo paso es crear una variable string en la cual guardaremos un query en este caso como dice la línea 4 queremos extraer todos los campos de la tabla producto cuando el id del producto coincida con la variable que recibe la función obtenerdatosproducto.

Después crearemos una variable datatable en la cual guardaremos el resultado de ejecutar la función ExecuteQuery de la capa de persistencia.

Si se detecta que el numero de filas es 0 regresa null lo que significa que no pudo obtener datos.

En caso contrario regresa los datos de la fila haciendo uso de la capa de datos.

**Función BajaProducto.**

1. **public** **static** **void** BajaProducto(TheVirtualTianguisProyect.Negocios.Datos.DatosProducto DatosProducto)
2. {
3. **String** ConnectionString = **String**.Format(TheVirtualTianguisProyect.Properties.Settings.**Default**.Conexion);
4. **String** query = "update Producto set ProductoActivo = '" + **false** + "' where Id\_producto = " + DatosProducto.Id\_producto;
6. TheVirtualTianguisProyect.Persistencia.AdministradorDatos.ExecuteNonQuery(ConnectionString, query);
7. }

Esta función recibe como parámetro un dato de tipo DatosProducto que pertenece a la capa de negocios.

El primer paso es crear una variable string en la cual se almacena la dirección de la conexión.

El segundo paso es crear una variable string en la cual guardaremos un query en este caso es que actualice el campo de ProductoActivo (que por defecto al ingresar un producto esta activo) a false cuando el Id del producto coincida con la variable id de la variable DatosProducto.

Por último queda mandar la conexión y el query a la función ExecuteNonQuery.

**Función AltaProducto**

1. **public** **static** **void** AltaProducto(TheVirtualTianguisProyect.Negocios.Datos.DatosProducto DatosProducto)
2. {
3. **String** ConnectionString = **String**.Format(TheVirtualTianguisProyect.Properties.Settings.**Default**.Conexion);
4. **String** query = "insert into Producto( Nombre, Marca, Precio, Categoria, Descripcion, Proveedor, ProductoActivo ) values ( " + "'" + DatosProducto.Nombre + "', " + "'" + DatosProducto.Marca + "', " + DatosProducto.Precio + ", '" + DatosProducto.Categoria + "' , '" + DatosProducto.Descripcion + "', " + DatosProducto.Proveedor + ", '" + DatosProducto.ProductoActivo + "') ";
6. TheVirtualTianguisProyect.Persistencia.AdministradorDatos.ExecuteNonQuery(ConnectionString, query);
7. }

Esta función recibe como parámetro un dato de tipo DatosProducto que pertenece a la capa de negocios.

El primer paso es crear una variable string en la cual se almacena la dirección de la conexión.

El segundo paso es crear una variable string en la cual guardaremos un query en este caso inserta los datos que recibe de la variable DatosProducto (que es donde estará almacenada la información que se recibe del usuario) a la tabla Producto en los campos indicados.

Por último queda mandar la conexión y el query a la función ExecuteNonQuery.

**Función VerificaProductoAlta.**

1. **public** **static** **Double** VerificaProductoAlta(System.**String** Nombre)
2. {
3. **int** Bandera;
4. SqlDataReader Encontrado;
5. **String** ConnectionString = **String**.Format(TheVirtualTianguisProyect.Properties.Settings.**Default**.Conexion);
6. **String** query = "SELECT \* FROM Producto WHERE Nombre = '" + Nombre +"' And ProductoActivo = 'true'";
8. SqlConnection myConnection = [new](http://www.google.com/search?q=new+msdn.microsoft.com) SqlConnection(ConnectionString);
9. **try**
10. {
11. myConnection.Open();
12. SqlCommand myCommand = [new](http://www.google.com/search?q=new+msdn.microsoft.com) SqlCommand(query, myConnection);
13. Encontrado = myCommand.ExecuteReader();
14. **if** (Encontrado.Read() == **false**)
15. {
16. Bandera = -1;
17. }
18. **else**
19. {
20. Bandera = 1;
21. }
22. }
23. **catch** (Exception ex)
24. {
25. **string** message = ex.Message;
26. MessageBox.Show(message);
27. **throw** ex;
28. }
29. **finally**
30. {
31. **if** (myConnection.State == ConnectionState.Open)
32. myConnection.Close();
33. }
34. **return** Bandera;
35. }

Esta función recibe de parámetro una variable string con el nombre del producto.

Lo que sigue es crear una bandera la cual retornaremos con un valor dependiendo de si se encuentra o no el producto.

El segundo paso es declarar una variable de SqlDataRaider donde se guardara el valor de la ejecución del query. True en caso de que exista el producto y false en caso contrario.

Y por ultimo retorna la bandera.

**Función ActualizaDatosProducto.**

1. **public** **static** **void** ActualizaDatosProducto(TheVirtualTianguisProyect.Negocios.Datos.DatosProducto DatosProducto)
2. {
3. **String** ConnectionString = **String**.Format(TheVirtualTianguisProyect.Properties.Settings.**Default**.Conexion);
4. **String** query = "update Producto set Nombre = '" + DatosProducto.Nombre + "', Marca = '" + DatosProducto.Marca + "', Precio = " + DatosProducto.Precio + ", Categoria = '" + DatosProducto.Categoria + "', Descripcion = '" + DatosProducto.Descripcion + "', Proveedor = " + DatosProducto.Proveedor + ", ProductoActivo = '" + DatosProducto.ProductoActivo + "' where Id\_producto = " + DatosProducto.Id\_producto;
6. TheVirtualTianguisProyect.Persistencia.AdministradorDatos.ExecuteNonQuery(ConnectionString, query);
7. }

Esta función recibe como parámetro un dato de tipo DatosProducto que pertenece a la capa de negocios.

El primer paso es crear una variable string en la cual se almacena la dirección de la conexión.

El segundo paso es crear una variable string en la cual guardaremos un query en este caso actualiza los datos que recibe de la variable DatosProducto (que es donde estará almacenada la información que se recibe del usuario) a la tabla Producto en los campos indicados.

Por último queda mandar la conexión y el query a la función ExecuteNonQuery.

**Función ExecuteQuery.**

1. **public** **static** DataTable ExecuteQuery(**string** ConnectionString, **string** query, **string** tableName)
2. {
3. **try**
4. {
5. SqlConnection myConnection = [new](http://www.google.com/search?q=new+msdn.microsoft.com) SqlConnection(ConnectionString);
6. SqlDataAdapter myAdapter = [new](http://www.google.com/search?q=new+msdn.microsoft.com) SqlDataAdapter(query, myConnection);
7. DataSet ds = [new](http://www.google.com/search?q=new+msdn.microsoft.com) DataSet();
8. myAdapter.Fill(ds, tableName);
9. ds.Tables[0].TableName = tableName;
10. **return** ds.Tables[0];
11. }
12. **catch** (Exception ex)
13. {
14. **string** message = ex.Message;
15. MessageBox.Show(message);
16. **throw** ex;
17. }
18. }
19. }

La función recibe como parámetros una variable string que almacene la dirección de la conexión, una variable string que contenga el query a ejecutar y el nombre de la tabla sobre la cual aplicaremos el query.

Lo primero que se hace es crear una conexión con una variable SqlConnection.

Declarar una variable SqlDataAdapter que funcionara como traductor entre el query y la base de datos.

La variable de tipo DataSet la usaremos para guardar un conjunto de tablas.

Después al llamar la función fill en la variable myAdapter guarda la ejecución del query junto con tabla en el DataSet.

A continuación se le asigna el nombre a la primer tabla gracias al parámetro de entrada tableName.

Y por ultimo retorna esa tabla.

**Función ExecuteNonQuery.**

1. **public** **static** **void** ExecuteNonQuery(**string** ConnectionString, **string** query)
2. {
3. SqlConnection myConnection = [new](http://www.google.com/search?q=new+msdn.microsoft.com) SqlConnection(ConnectionString);
4. **try**
5. {
6. myConnection.Open();
7. SqlCommand myCommand = [new](http://www.google.com/search?q=new+msdn.microsoft.com) SqlCommand(query, myConnection);
8. myCommand.ExecuteNonQuery();
9. }
10. **catch** (Exception ex)
11. {
12. **string** message = ex.Message;
13. MessageBox.Show(message);
14. //throw ex;
15. }
16. **finally**
17. {
18. **if** (myConnection.State == ConnectionState.Open)
19. myConnection.Close();
20. }
21. }

La función recibe como parámetros una variable string que almacene la dirección de la conexión, una variable string que contenga el query a ejecutar.

Lo primero que se hace es crear una conexión con una variable SqlConnection.

Dentro de try primero prueba la conexión que se le envia y busca abrir una base de datos.

Declarar una variable SqlCommand que va a guardar un query y la conexión y los junta en un comando y al final ejecuta el query con el método de la variable SqlCommand.

**Función SiguienteID.**

1. **public** **static** **int** SiguienteID(**string** tableName, **string** idField)
2. {
3. **int** val;
4. **String** ConnectionString = **String**.Format(TheVirtualTianguisProyect.Properties.Settings.**Default**.Conexion);
5. SqlConnection myConnection = [new](http://www.google.com/search?q=new+msdn.microsoft.com) SqlConnection(ConnectionString);
7. **string** Query = "select max(" + idField + ") from " + tableName;
8. myConnection.Open();
9. SqlCommand myCommand = [new](http://www.google.com/search?q=new+msdn.microsoft.com) SqlCommand(Query, myConnection);
11. **object** maxValue = myCommand.ExecuteScalar();
12. myConnection.Close();
13. **if** (maxValue == DBNull.**Value**) **return** 1;
14. **else**
15. val = **int**.Parse((maxValue).ToString());
16. **return** val + 1;
17. }

Esta función se encarga de abrir una nueva fila para almacenar los datos deseados.

Recibe el nombre de la tabla y una variable que nos indica la columna de los identificadores con la cual nos guiaremos para encontrar el id más grande para después retornar el valor del siguiente id.

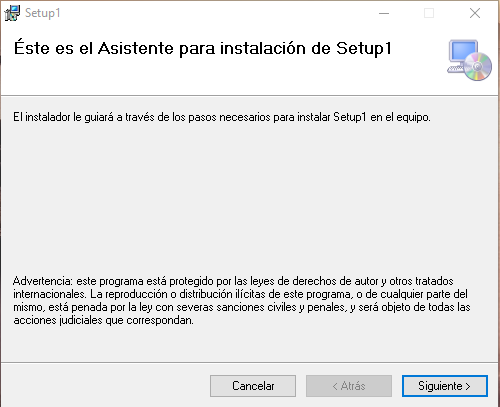
# Plan de Instalación

## Software de Escritorio

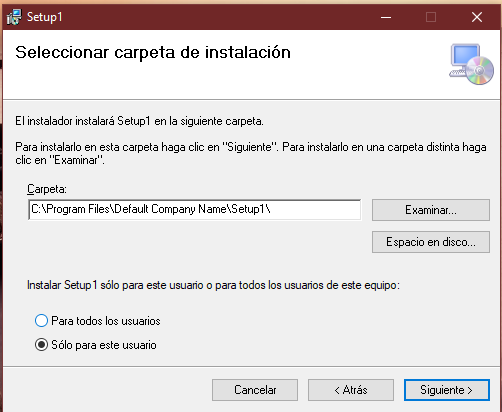
La forma de instalar el software en Windows será el siguiente:

1. Se tendrá que crear la base de datos compilando el archivo QueryDeProyectoFinal que se encontrará en la ruta \VirtualTianguis.
2. Una vez creada la base de datos se podrá instalar el archivo setup1.exe que se encontrará en la siguiente ruta \VirtualTianguis\TheVirtualTianguisProyect\Setup1\Debug.

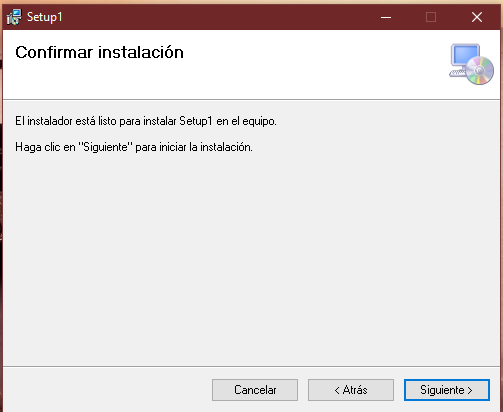
Los pasos de instalación serán los siguientes. En caso de no tener el framework adecuado se solicitará instalarlo. El peso aproximado del programa es de 50 Mb.



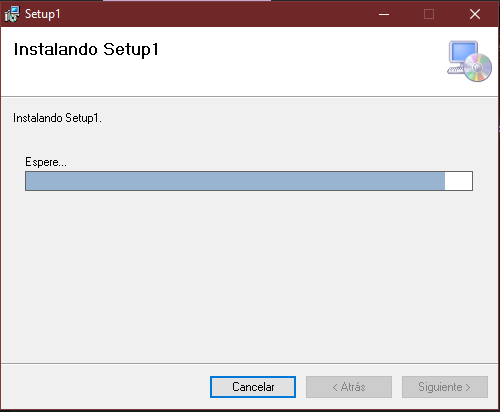
Seleccionar siguiente para continuar.



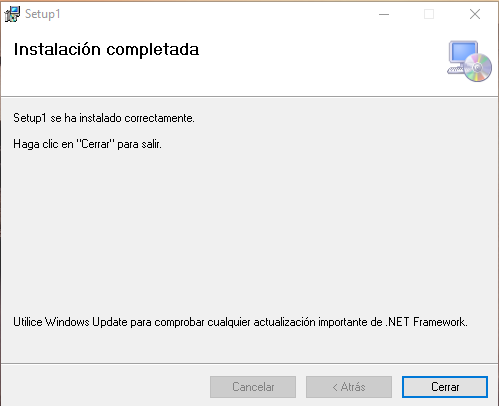
En este paso seleccione la ruta de instalación y los usuarios para los que desea instalar la aplicación. Seleccione siguiente para continuar.



Ahora confirme la instalación en el botón siguiente.



Espere a que termine la instalación.



Una vez concluida la instalación ya puede cerrar la ventana y abrir el acceso directo que queda en su escritorio.

# Conclusiones

Marco Antonio Díaz Valdez

A lo largo de este proyecto pude entender y dominar la estructura de la programación de un proyecto en forma de programación por capas, usando estructuras de datos y funciones de manipulación de los mismos por separado. Este ha sido mi primer proyecto creado en visual studio con el lenguaje C# de manera organizada, me ha servido como base y experiencia para poder trabajar con otros IDEs para programar independientemente del lenguaje o compilador que utilice el mismo. También fue mi primer proyecto en el que uso un servidor de base de datos para guardar información en lugar de guardar los datos en un archivo .txt, siempre y cuando el IDE tenga un entorno para diseñar tablas de una base de datos, no necesito de relacionarlas directamente en la base de datos puesto que al tener este entorno puedo hacer las relaciones por medio de código.

Luis Gonzalo Soriano Crespo

Me resulto muy interesante el proceso de este proyecto final para la materia de bases de datos, entendí como es que se comunica visual studio con la base de datos en este caso sql server por medio de código C# en lo personal este lenguaje me gusto mucho ya que entendí un uso más del mismo, no sabía muchos métodos que se implementan con C# por lo que me costo algo de trabajo comprender y hacer funcional algunas partes del código como la conexión del login por capas, también implemente un servidor en red sobre la conexión de mi celular para trabajar sobre la misma base de datos, pero tuvimos algunos inconvenientes con otros elementos del proyecto para hacerlo más óptimo, tuvimos algunas diferencias de ideas para la estructura de las bases de datos y el proyecto en visual studio pero al final supimos como resolverlas. La elaboración de este proyecto me ayudo a comprender los principios de otros programas que utilizan bases de datos.

Christian Josué Acosta Vargas

El proyecto final resultó bastante complicado puesto que me hacia falta bastante conocimiento sobre el uso de capas y uso de herramientas que proporciona visual studio.

El resultado final es el esperado desde que fue planteado, aunque en el transcurso si vimos que tuvimos fallos en la planeación de algunos forms, también supimos corregirlos.

En trayecto aprendí a utilizar bastantes como el lenguaje c#, herramientas de visual studio como la creación de un instalador, aprendí a utilizar la mayoría de elementos que contiene el proyecto desde la creación de forms y hasta la programación de los elementos de esta.

Algo que me costó bastante trabajo aprender fue el uso de las datagridview puesto que teníamos que llenarlas con tablas y si queríamos algo especifico de ellas tuve que hacer consultas desde las tablas de dataset.

# Referencias

- InformaBTL. (25 de Junio de 2015). ¿Qué es el Punto de Venta? Obtenido de InformaBTL Promociones, activaciones y below the line: https://www.informabtl.com/que-es-el-punto-de-venta/

- Rouse, M. (1 de Agosto de 2017). Microsoft SQL Server. Obtenido de TechTarget: https://searchsqlserver.techtarget.com/definition/SQL-Server

- The Wikimedia Foundation. (24 de Febrero de 2019). Database. Obtenido de Wikipedia, the free encyclopedia: https://en.wikipedia.org/wiki/Database

- The Wikimedia Foundation. (16 de Abril de 2019). Integrated development environment. Obtenido de Wikipedia, the free encyclopedia: https://en.wikipedia.org/wiki/Integrated\_development\_environment

- The Wikimedia Foundation. (7 de Febrero de 2019). Multitier architecture. Obtenido de Wikipedia,the free encyclopedia: https://en.wikipedia.org/wiki/Multitier\_architecture

- W3Schools. (1 de Enero de 2019). SQL Tutorial. Obtenido de w3schools.com: https://www.w3schools.com/sql/?fbclid=IwAR3o-fJoVQgN0lG7ryN-z\_xPI87KlcGwsMLnG2p8LHW63lpZKtb9goUHrbk

- Warren, G., & Lee, T. G. (18 de Marzo de 2019). Welcome to the Visual Studio IDE. Obtenido de Microsoft: https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2019

- Warren, G., Casey, L., Hogenson, G., Jones, M., & Cai, S. (11 de Febrero de 2017). Create a database and add tables in Visual Studio. Obtenido de Microsoft: https://docs.microsoft.com/en-us/visualstudio/data-tools/create-a-sql-database-by-using-a-designer?view=vs-2019&fbclid=IwAR1VuGXT-qL0ISGdjDSuTkxfbxXhfjA\_u1jRmR9w1lTLGMmge6bUAqBmQiQ