2018

SISTEMA DE CONTROL DE VENTAS Y MANEJO DE EXISTENCIAS PARA TOP TUR CASINO

TORRES LÓPEZ, NICOLÁS IGNACIO

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
SEDE VIÑA DEL MAR
JOSÉ MIGUEL CARRERA

SISTEMA DE CONTROL DE VENTAS Y MANEJO DE EXISTENCIAS PARA TOP TUR CASINO

Trabajo de Titulación
para optar al Título de:
TÉCNICO UNIVERSITARIO EN INFORMÁTICA

Alumno:
Nicolás Ignacio Torres López

Profesor Guía:
Dagoberto Cabrera Tapia.

2018

RESUMEN

Key words: Procesos, Funcionalidades, Software, Sistema,

El siguiente informe tiene como objetivo describir todos los procesos y funcionalidades del software que se proveerá a la micro empresa TOPTUR CASINO. Este software es un sistema de control de ventas y manejo de existencias desarrollado para resolver las problemáticas actuales de la empresa con respecto al manejo de sus productos tanto en la venta como en el almacenamiento de éstos.

La micro empresa TOPTUR CASINO de la cual la señora Marcia Emilia López Alvarado es la administradora y jefa se encuentra ubicada en el Cerro San Roque, Valparaíso, Región de Valparaíso, es un sector lejano a la ciudad donde los microbuses de la empresa TOPTUR S.A llegan tras finalizar sus recorridos diarios, ahí es donde se encuentra físicamente el espacio donde los choferes de estos microbuses van a disfrutar de la comida entregada por TOPTUR CASINO y a descansar.

A fin de agilizar los procesos que ocurren dentro de la empresa ya sea de ventas, atención a los clientes, pagos de deudas y administración de la bodega es que se propone crear este sistema de control de ventas y manejo de existencias que puede cumplir las exigencias planteadas por el negocio.

Para el desarrollo del sistema computacional del cual harán uso en esta empresa se ha usado el lenguaje de programación C# en el IDE (Entorno de desarrollo integrado) Visual Studio 2017, siendo ésta su última versión disponible hasta el momento. Y para almacenar los datos del sistema se ha usado el sistema de gestión de base de datos MySQL de ORACLE.

El contenido de los capítulos es el siguiente:

Capítulo uno: Se describe la empresa TOP TUR CASINO en su totalidad para poder contextualizar el ambiente del negocio y la estructura con la cual funciona actualmente, los problemas que se detectaron respecto al manejo rudimentario de sus ventas y la falta de gestión en sus productos y sus respectivas soluciones propuestas que se han incluido en el sistema.

Capítulo dos: Se mostrarán principalmente las tablas que formarán parte del sistema ya sea de clientes, ventas o productos, diagrama de Flujos administrativos que describe el proceso por el cual pasa el sistema propuesto, entradas y salidas, modelo de datos relacional y estructura de códigos utilizados por el sistema. A su vez serán descritas las estructuras funcionales del sistema de manera breve ya sea para la generación de ventas, gestión de productos y manejo de personal.

Capítulo tres: Por último, en el capítulo tres se dan a conocer el Diagrama de Menú que tendrá el sistema y el Diagrama modular con el cual se trabajará, se muestran las pantallas del sistema mostrando cada función que se tendrá el sistema de control de ventas y manejo de existencias propuesto.

Como conclusión se ha descrito el proceso por el cual pasé para poder desarrollar el sistema, como este se vio afectado por distintas influencias externas y el futuro que podría deparar a este sistema en caso de implementarse.

Al final de este informe se anexaron los códigos fuente de algunas de las funciones más importantes del sistema propuesto.

ÍNDICE

INTRODUCCIÓN

Este sistema se desarrollará para la empresa TOPTUR CASINO de la cual la señora Marcia Emilia López Alvarado es la administradora y jefa, esta empresa se encuentra localizada en el cerro San Roque de Valparaíso y se encuentra en funcionamiento durante toda la semana.

Este sistema nace de la necesidad de la organización por reducir la pérdida de dinero generada debido a la desinformación que se produce al no poseer un sistema de control de ventas y un sistema de control de existencias, además de la falta de un sistema de deudas pendientes para los clientes habituales que en momentos determinados pueden conducir a pérdidas de dinero y/o problemas directos con estas personas.

El software que se propone crear será desarrollado en el lenguaje de programación C# mediante la herramienta Microsoft Visual Studio, éste software pretende solucionar este problema mediante la implementación de 3 sub-sistemas, un sistema de control de ventas que tiene como propósito cumplir de manera eficiente con la emisión de un comprobante de venta, proporcionando información detallada de la venta producida y los actores que interactúan en ésta, un sistema de control de existencias que tiene como propósito llevar el conteo de los insumos disponibles para el uso del personal autorizado y por último un sistema de deudas pendientes para los clientes habituales que tiene como propósito dar a conocer al administrador las deudas que poseen los clientes habituales y así poder tomar decisiones al respecto, estos sistemas estarán disponibles para la plataforma Microsoft Windows.

CAPÍTULO I:
ASPECTOS RELEVANTES DEL DISEÑO LÓGICO

1. ASPECTOS RELEVANTES DEL DISEÑO LÓGICO

1.1 DESCRIPCIÓN DE LA ORGANIZACIÓN

La organización para la cual se desarrollará el sistema está bajo la administración de su dueña, la señora Marcia Emilia López Alvarado y se encuentra ubicada en la ciudad de Valparaíso.

Esta empresa se dedica a la compra y venta de productos varios y además a la venta de comida preparada. Cuenta con una cantidad de clientes fluctúa entre las 25 y 30 personas, pero éstos se mantienen comprando durante todo el día por lo que las ventas no son pocas.

La dueña de la empresa es la que maneja las finanzas en general, lleva el conteo de los productos vendidos y las deudas mediante un cuaderno, cuenta el dinero de manera manual y posteriormente invierte nuevamente en nuevos productos.

El sistema por desarrollar será instalado en un computador portátil disponible en el lugar de trabajo.

Al interior de la empresa no se maneja ningún tipo de software para administrar ésta, por lo tanto, se quiere solucionar la mayor cantidad de problemas con la integración de este sistema.

1.2 DESCRIPCION DE LA SITUACION ACTUAL

En la actualidad la empresa cuenta con un sistema de control de ventas manual en el cual se procede a escribir en un cuaderno(ver en página 5 y 6) el detalle de la venta realizada, o sea, se escriben los productos que se venden, el precio de estos productos y si el cliente desea pagar en otro momento se anota el nombre de éste en el cuaderno junto a la venta, para así identificar que éste posee una deuda en esa venta en específico, en el momento en el que el cliente pague esa deuda que posee, la vendedora que se encuentre trabajando deberá borrar su nombre de la venta que se encuentra en el cuaderno para así concretar el pago de ésta y que éste no sea identificado posteriormente por otra vendedora como un deudor.

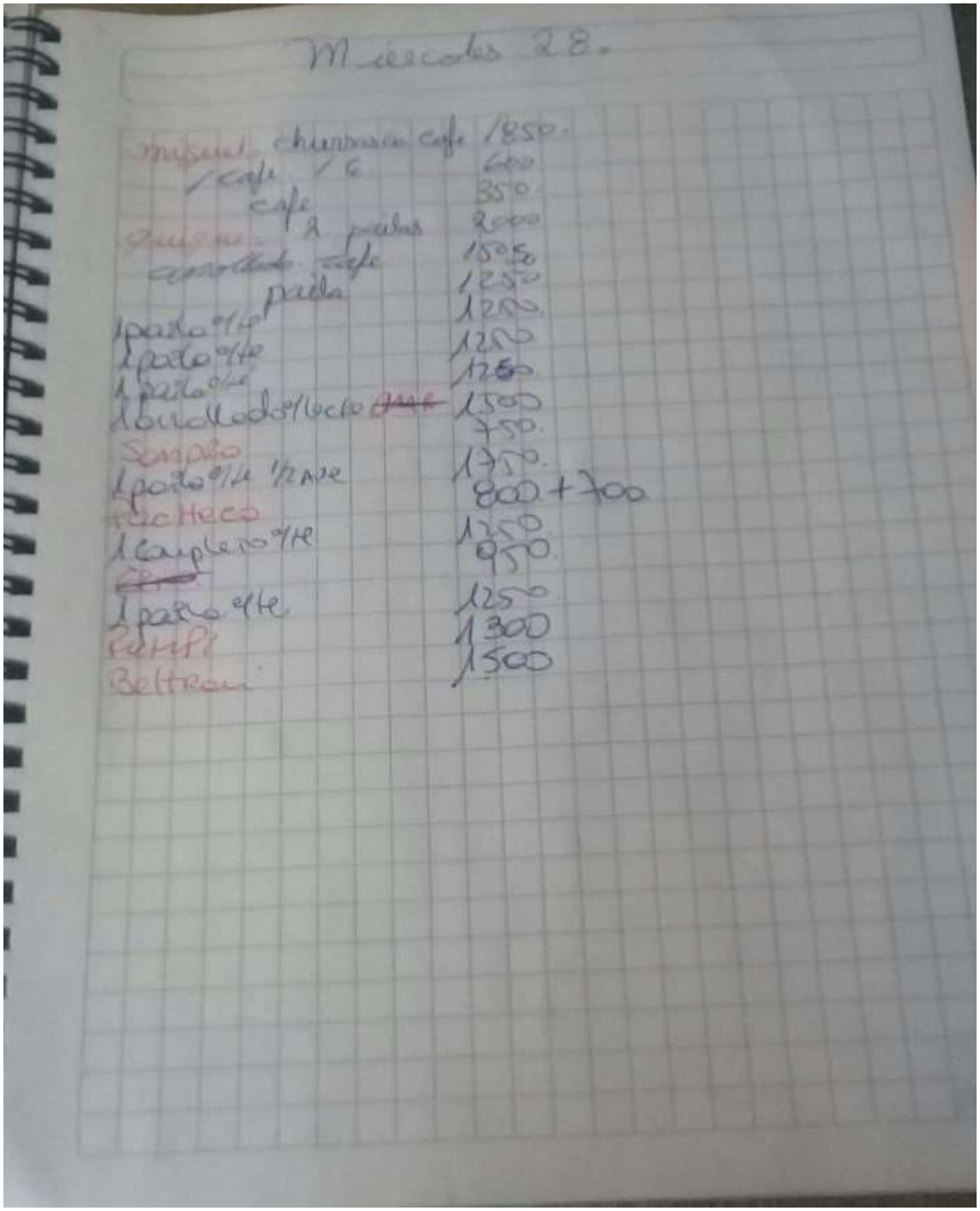
Este sistema genera muchos problemas debido a la inexactitud que provee, ya que el cuaderno se puede modificar en cualquier momento y lo puede modificar cualquier vendedora, esto trae problemas debido a que los clientes no siempre tienen una buena relación con cada una de las vendedoras por lo que en ciertos momentos algunos clientes niegan sus deudas a otras vendedoras que no son las que lo anotaron y esto conlleva a que la vendedora que lo anotó tiene que cobrar personalmente al cliente y al no tener una buena relación podría terminar en un conflicto de mayor escala.

Esta empresa además no cuenta con un sistema de manejo de existencias por lo cual hay una mala gestión al no saber qué se posee con exactitud ni que productos se venden, el problema que le genera a esta empresa no poseer este sistema es que las vendedoras tienen acceso a todos los productos que se manejan dentro de la bodega y éstas pueden hacer uso indebido de éstos(consumirlos) o incluso robarlos, esta problemática además provoca que las vendedoras duren poco en el puesto de trabajo, usualmente la dueña busca personas conocidas con tal de evitar estos problemas, este método es poco efectivo ya que al implementarlo ha disminuido las pérdidas pero siguen ocurriendo.

1.2.1 IMÁGENES CUADERNO DE VENTA

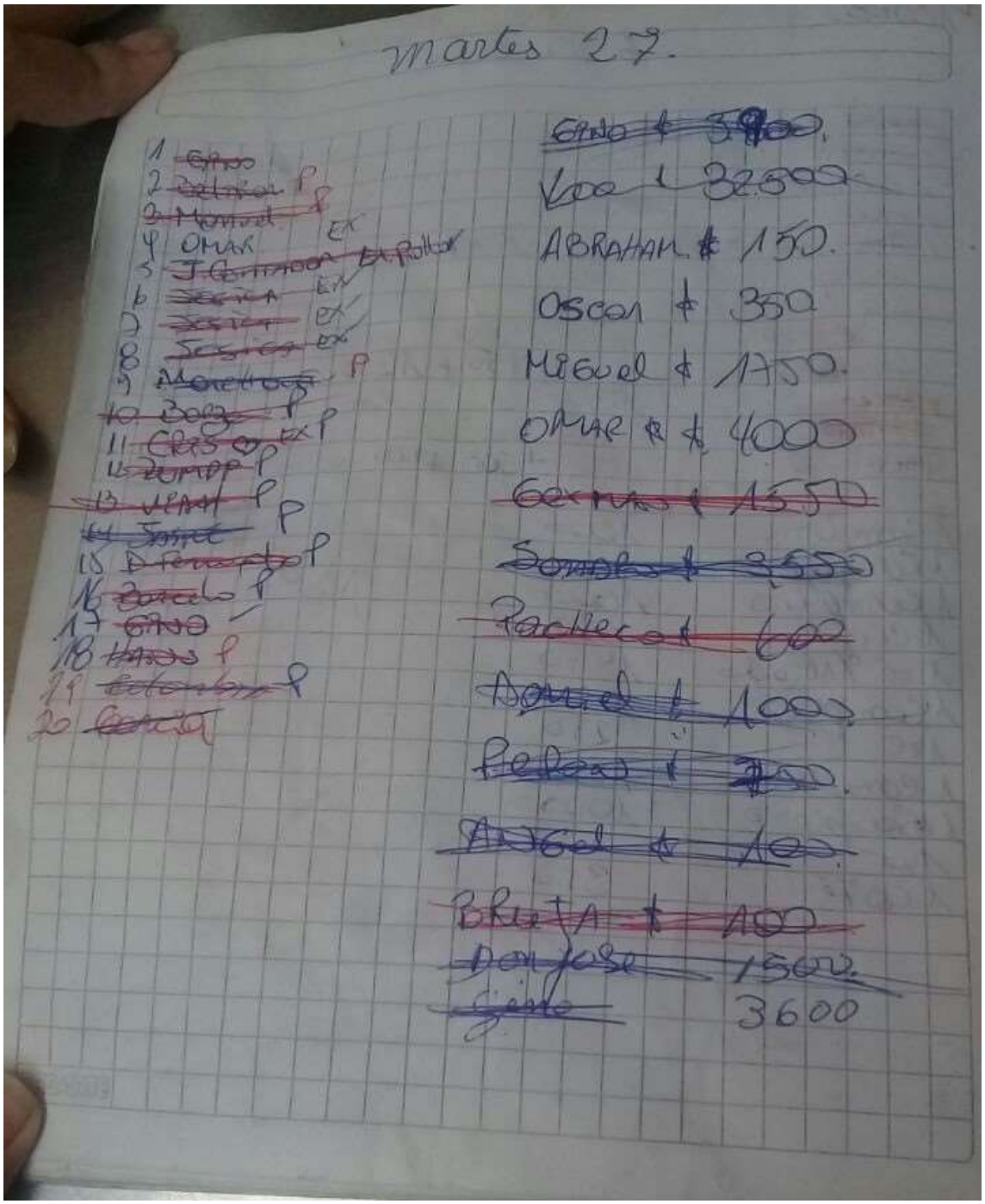
Ventas del miércoles 28

(Nombre producto / Precio)



Reserva de almuerzos y deudores del martes 27

(Nombre del cliente que reserva almuerzo) (Nombre deudor / Deuda)



1.3 PROBLEMAS DETECTADOS

Debido a que la empresa cuenta con un sistema rudimentario de control de ventas y no posee un sistema de manejo de existencias, se han detectado los siguientes problemas:

- Al momento de ingresar o consultar una venta, el tiempo de respuesta es bastante alto debido a que se debe hacer manualmente en el cuaderno correspondiente.

- No se lleva un registro de las entradas o salidas de productos que se producen en la bodega.

- Se generan pérdidas en el stock de productos al no poseer registros.

- Las transacciones no poseen un respaldo histórico al que se tenga acceso para rectificar deudas y/o cuadrar las existencias de la bodega.

- Los clientes no poseen acceso al monto de sus deudas y tampoco las confirman por lo que no se sabe realmente si las deudas existentes en el cuaderno son fidedignas.

- Pérdidas sustanciales de dinero al no cobrar las deudas pendientes que quedan en cuadernos antiguos o que los clientes se niegan a pagar.

- Problemas entre el personal y la administradora por posibles pérdidas en el stock de productos.

1.4 DESCRIPCIÓN DEL SISTEMA PROPUESTO

1.4.1 Objetivos del sistema propuesto

Luego de analizar el sistema que se está empleando y las carencias que poseen en la empresa, se decide implementar un nuevo sistema de control de ventas e interconectarlo con un sistema de manejo de existencias, además de insertar a los clientes en el sistema.

El sistema que se desarrollará tiene los siguientes objetivos:

- Agilizar y respaldar transacciones producidas en el sistema de control de ventas.

- Almacenar los datos de las transacciones de ventas en una base de datos local y vincular ésta a la nube para mantener actualizada la base de datos de backup.

- Agregar al cliente dentro del sistema para así facilitar el manejo de las deudas.

- Controlar la adquisición, venta y uso de los productos que se encuentren en bodega para así llevar una contabilidad acertada y evitar peleas internas del personal por falta de productos.

1.4.2 Descripción General de la Solución Propuesta

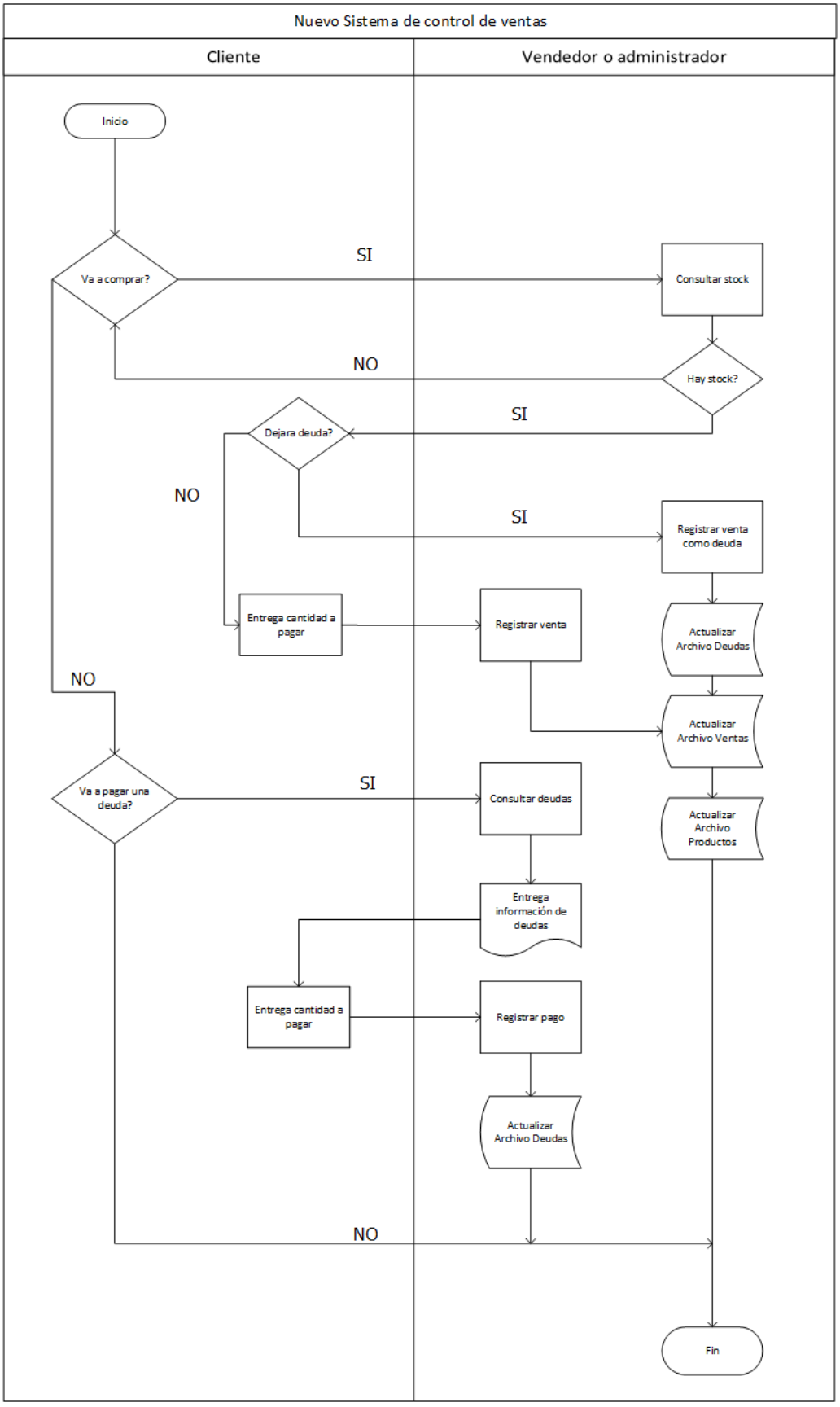
Este sistema se dedicará al control de ventas guardando la información necesaria para identificar todos los aspectos de la venta en una base de datos, además controlará la adquisición, venta y uso de las existencias de la bodega.

Para esto se reconocerá los productos mediante un identificador único y así cualquiera que fuese la acción que se tomase respecto a este, la persona que lo hizo tendría asignado este identificador a su RUT de esta manera se sabría quien fue la persona que realizó la acción sobre el producto, finalmente incluirá a los clientes en un sistema donde los vendedores podrán consultar las ventas realizadas hacia estos clientes y las deudas que pudieran existir.

Beneficios del sistema propuesto:

- Rapidez en el ingreso y consulta de las transacciones de venta producidas.
- Se reduce la inconsistencia de datos que podría haber existido con el anterior sistema.
- Interfaz de usuario amigable y fácil de entender para el correcto manejo de las ventas y el control de existencias.
- Almacenamiento interno en una base de datos para no tener pérdidas sustanciales de datos importantes para la empresa.
- Se elimina la desconfianza al tener datos de respaldo en caso de pérdidas de productos.
- Se visualizan las deudas pendientes para así poder cobrarlas en cualquier momento y no percibir pérdidas al respecto.

1.4.3 Diagrama de Flujo Administrativo (sistema propuesto)



1.4.4 Estructura Funcional del Sistema

Las funciones que conforman el sistema propuesto son las siguientes:

- Mantención de datos, es aquí donde se les dará mantención a los datos de los productos almacenados en la bodega.

- Proceso de ventas, es aquí donde se creará una nueva venta para su posterior almacenamiento en la base de datos.

- Consulta de ventas, es aquí donde se consultan las ventas producidas históricamente a la base de datos correspondiente.

- Generación de deuda, es aquí donde se le asigna el monto de una venta como deuda a cierto cliente en específico.

- Consulta de deuda, es aquí donde se puede consultar las deudas que poseen los clientes en la base de datos histórica.

1.4.5 Información que se Manejará

Salidas:

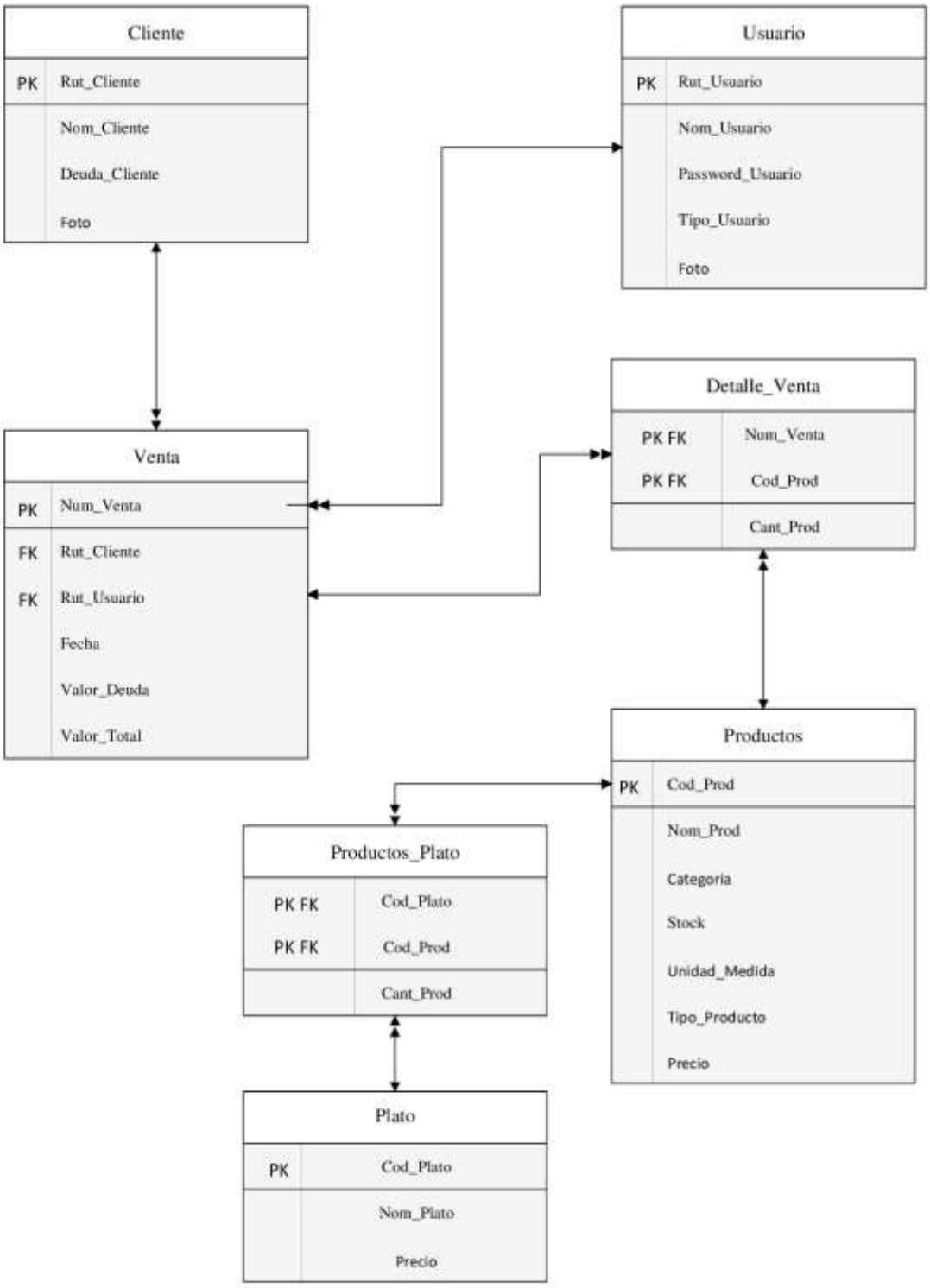
- Listado de ventas producidas.
- Listado de productos en bodega.
- Listado de clientes.
- Detalle de ventas producidas.
- Listado de deudas.

Todas las salidas generadas por el sistema propuesto pueden ser visualizadas por pantalla.

Entradas:

- Datos de los clientes.
- Datos de los vendedores.
- Datos del administrador.
- Datos de los productos.
- Datos de las ventas.
- Datos de las deudas.

1.4.6 Entidades de Información



1.4.7 Condicionantes de Diseño

Se ha optado por hacer esta aplicación para escritorio compatible con el sistema operativo Microsoft Windows 7 ya que es el sistema operativo con el que cuenta el sistema computacional que se encuentra dentro de la empresa, el software es programado en el lenguaje de programación C#, las bases de datos que se usarán en el sistema son MySQL y se usarán instrucciones SQL para manejar las tablas de éstas y la exportación de datos a archivos locales.

CAPÍTULO 2:

MEDIO AMBIENTE COMPUTACIONAL Y DESCRIPCIÓN DE ARCHIVOS

2. MEDIO AMBIENTE COMPUTACIONAL Y DESCRIPCIÓN DE ARCHIVOS

2.1 DESCRIPCIÓN DE LAS CARACTERÍSTICAS DEL RECURSO COMPUTACIONAL

A continuación, se detallan las principales características del recurso computacional, utilizado para el desarrollo del sistema de Control de Ventas y Manejo de Existencias, para la empresa TOPTUR CASINO.

2.1.1 Descripción del hardware utilizado

A continuación, se detallan las características del hardware utilizado en el desarrollo del sistema y del hardware en que se manejará el Sistema de Control de Venta y Manejo de Existencias.

Hardware utilizado en el desarrollo

Notebook Asus ROG GL552VW:

- Procesador INTEL Core I7 de 2.6~ GHz
- 12 GB de memoria RAM
- SSD de 120 GB de capacidad
- HDD de 1 TB de capacidad
- Monitor integrado
- Mouse óptico de 2 botones USB
- Teclado integrado

Hardware donde se utilizará

Notebook Hp Stream:

- Procesador Intel Celeron N3060 1.6 ~GHz
- 4 GB de memoria RAM
- HDD de 1 TB de capacidad
- Monitor integrado
- Teclado integrado

- Mousepad integrado
- Pantalla de 11.6”

2.1.2 Descripción del software utilizado

Windows 10 Home 64 bits:

Es la última versión del sistema operativo Windows de Microsoft, que presenta una cómoda interfaz y fácil manejo.

Microsoft Visual Studio 2017:
Microsoft Visual Studio 2017 es un IDE de programación para distintos lenguajes, se usará específicamente el lenguaje de programación C# que es nativo de Microsoft.

Como plataforma de desarrollo se usará Windows Forms con la ayuda de los controles personalizados para mejorar el aspecto de la aplicación.

También se usará el patrón de arquitectura de software MVC donde tendremos como modelo la BD en MySQL, como vista tendremos los formularios de Windows y por último como controlador estarán los scripts en C# que usarán a la vez una distribución en capas.

Lenguaje de programación C#:

Es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA (ECMA-334) e ISO (ISO/IEC 23270). C# es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común.

Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes.

MySQL:

MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual GPL/Licencia comercial por Oracle Corporation y está considerada como la base datos open source más popular del mundo, y una de las más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web.

MySQL fue inicialmente desarrollado por MySQL AB (empresa fundada por David Axmark, Allan Larsson y Michael Widenius). MySQL A.B. fue adquirida por Sun Microsystems en 2008, y ésta a su vez fue comprada por Oracle Corporation en 2010, la cual ya era dueña desde 2005 de Innobase Oy, empresa finlandesa desarrolladora del motor InnoDB para MySQL.

Está desarrollado en su mayor parte en ANSI C y C++. Tradicionalmente se considera uno de los cuatro componentes de la pila de desarrollo LAMP y WAMP.

MySQL es usado por muchos sitios web grandes y populares, como Wikipedia, Google (aunque no para búsquedas), Facebook, Twitter, Flickr, y YouTube.

Tipos de datos que soporta MySQL:

• TINYINT 1 byte (8 bit) • SMALLINT 2 bytes (16 bit) • MEDIUMINT 3 bytes (24 bit) • INTEGER 4 bytes (32 bit) • BIGINT 8 bytes (64 bit). • FLOAT • DECIMAL • DOUBLE	NUMERIC
• CHAR • VARCHAR • TINYBLOB, TINYTEXT • BLOB, TEXT • MEDIUMBLOB	TEXT

• MEDIUMTEXT • LONGBLOB, LONGTEXT • ENUM • SET	
• TIMESTAMP • TIME • YEAR • DATE • DATETIME	DATE

2.2 DESCRIPCIÓN DE TABLAS DEL SISTEMA

En este punto se detallan todas las tablas que fueron utilizadas en el desarrollo del sistema de Ventas y Control de Existencias para la empresa TOPTUR CASINO. Describiendo la estructura de cada una de ellas.

2.2.1 Usuario

Nombre : Usuario
Descripción : En esta tabla se almacenan los datos de los vendedores y el administrador de la empresa.
Clave primaria : Rut_Usuario

Nombre	Tipo	Descripción
Rut_Usuario	String (9)	Rut del usuario
Nom_Usuario	String (30)	Nombre del usuario
Password_Usuario	String (30)	Contraseña del usuario
Tipo_Usuario	String (20)	Define el tipo de usuario, ya sea Vendedor o Administrador
Foto	Boolean	0 = sin foto, 1 = con foto

2.2.2 Cliente

Nombre : Cliente

Descripción : En esta tabla se almacenan los datos de los clientes incluyendo el valor total de su deuda si posee una.

Clave primaria : Rut_Cliente

Nombre	Tipo	Descripción
Rut_Cliente	String (9)	Rut del cliente
Nom_Cliente	String (30)	Nombre del cliente
Deuda_Cliente	Integer	Valor total de su deuda
Foto	Boolean	0 = sin foto, 1 = con foto

2.2.3 Registro de ventas

Nombre : Venta

Descripción : En esta tabla se almacenan los datos de las ventas registradas en el sistema.

Clave primaria : Num_Venta

Claves foráneas: Rut_Cliente (Referencia a tabla Cliente)
Rut_Usuario (Referencia a tabla Usuario)

Nombre	Tipo	Descripción
Num_Venta	Integer	Identificador único de la venta
Rut_Cliente	String (9)	Rut del cliente que realizo la compra
Rut_Usuario	String (9)	Rut del usuario que realizo la venta
Fecha	Date Time	Fecha en que se produjo la venta
Valor_Deuda	Integer	Valor de la deuda que se registra, puede ser parte del valor total de la venta
Valor_Total	Integer	Valor total de la venta realizada

2.2.4 Detalle de venta

Nombre : Detalle_Venta
Descripción : En esta tabla se almacenan los productos envueltos en la venta y la cantidad de estos que fue requerida.
Clave primaria : Num_Venta + Cod_Prod
Claves foráneas: Num_Venta (Referencia a tabla Venta)
Cod_Prod (Referencia a tabla Productos)

Nombre	Tipo	Descripción
Num_Venta	Integer	Identificador único de la venta a la que pertenece este detalle
Cod_Prod	Integer	Identificador único del producto envuelto en la venta
Cant_Prod	Integer	Cantidad del producto que fue vendida

2.2.5 Productos

Nombre : Productos
Descripción : En esta tabla se almacenan los datos de los productos que se encuentran en inventario.
Clave primaria : Cod_Prod

Nombre	Tipo	Descripción
Cod_Prod	Integer	Identificador único del producto
Nom_Prod	String (30)	Nombre del producto
Categoria	String (30)	Categoria del producto
Stock	Float	Cantidad del producto almacenada en inventario
Unidad_Medida	String (10)	Unidad en la que se mide el producto.
Tipo_Producto	String	Identificador para saber si el producto se vende

	(20)	o se usa para producir algún plato
Precio	Integer	Precio del producto, Preparación = 0, Venta > 0

2.2.6 Productos por plato

Nombre : Productos_Plato

Descripción : En esta tabla se almacenan los datos de los productos que se encuentran en inventario.

Clave primaria : Cod_Prod + Cod_Plato

Claves foráneas: Cod_Prod (Referencia a tabla Productos)
Cod_Plato (Referencia a tabla Platos_Preparados)

Nombre	**Tipo**	**Descripción**
Cod_Prod	Integer	Identificador único del producto
Cod_Plato	Integer	Identificador único del plato al que pertenece el producto
Cant_Prod	String (30)	Cantidad del producto que requiere el plato

2.2.7 Platos Preparados

Nombre : Platos

Descripción : En esta tabla se almacenan los datos de los platos que se preparan en la empresa.

Clave primaria : Cod_Plato

Nombre	**Tipo**	**Descripción**
Cod_Plato	Integer	Identificador único del plato
Nom_Plato	String (30)	Nombre del plato.
Precio	Integer	Precio del plato.

CAPÍTULO 3:
ESTRUCTURA GENERAL DEL SISTEMA

3. ESPECIFICACIONES DEL SISTEMA

En este capítulo se revisará la estructura funcional del sistema propuesto profundizando en las funcionalidades especificas más importantes de este, describiéndolas brevemente y mostrando imágenes referenciales. Igualmente se mostrarán los diagramas: modular y de menús, como también una lista con los programas que posee el sistema.

3.1 DIAGRAMA MODULAR

A continuación, se presenta un diagrama modular que muestra las funcionalidades del sistema separadas por módulos.

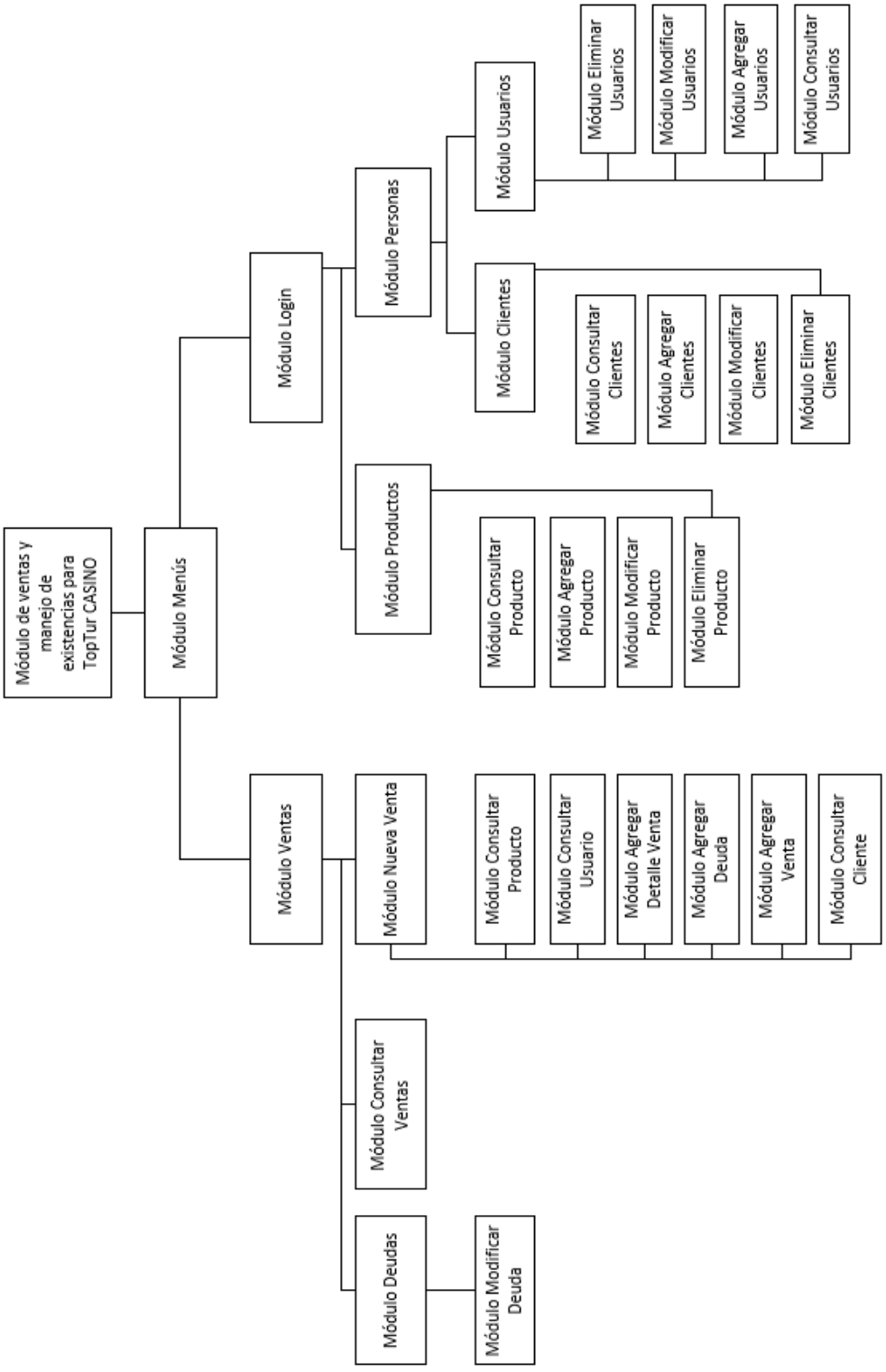


Figura 3.1 Diagrama modular del sistema.

3.2 DIAGRAMA DE MENÚS

El siguiente diagrama de menús muestra la distribución del menú lateral del sistema propuesto.

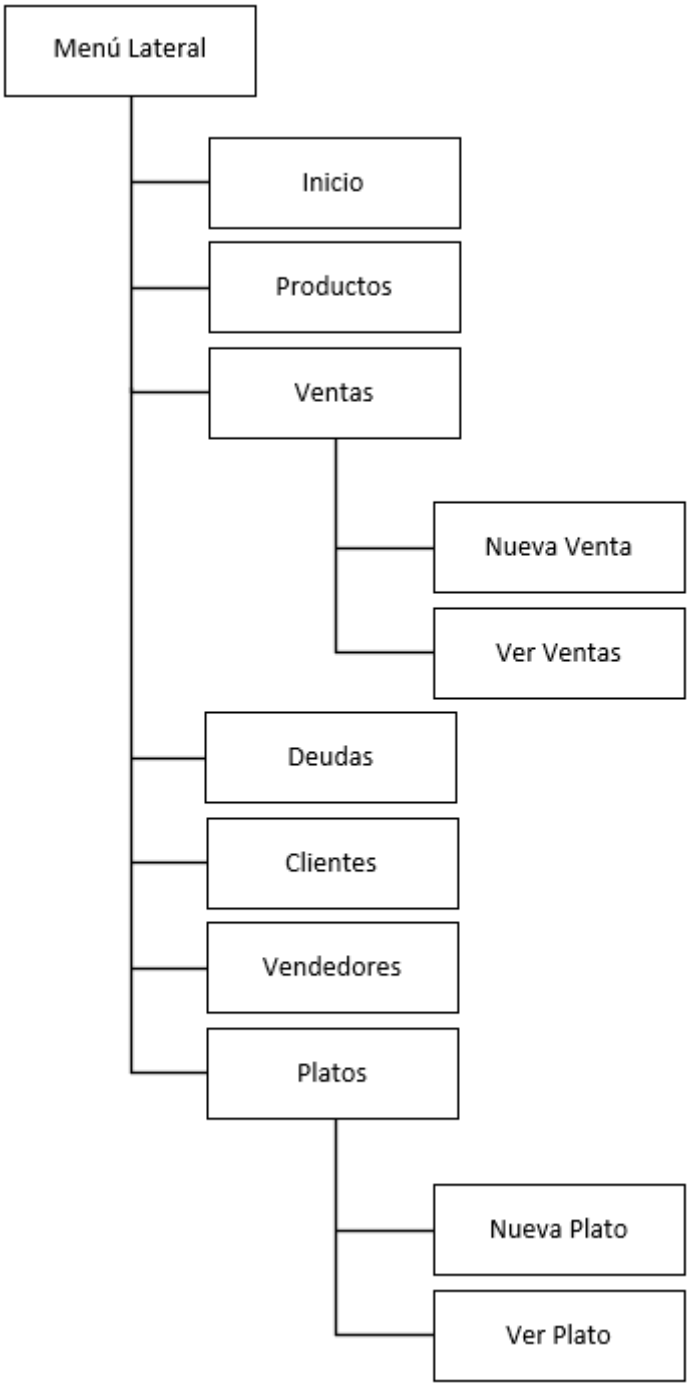


Figura 3.2 Diagrama de menús del sistema.

3.3 LISTA DE PROGRAMAS

La siguiente tabla contiene la lista de programas que pertenecen al sistema propuesto (el * marca los programas que serán explicados posteriormente).

	NOMBRE DEL PROGRAMA	OBJETIVO
1	Inicio de Sesión (*)	Permite al usuario registrado tener acceso a ciertas características del sistema
2	Listar Productos	Muestra un listado de todos los productos ya registrados en el sistema
3	Agregar Productos	Permite agregar nuevos productos al sistema
4	Modificar Productos (*)	Permite modificar los datos de los productos
5	Eliminar Productos	Permite eliminar productos ya registrados
6	Nueva Venta (*)	Permite al vendedor generar una nueva venta
7	Ver Ventas	Lista las ventas realizadas
8	Listar Deudas	Muestra un listado de todas las deudas pendientes de pago
9	Pagar Deudas (*)	Permite pagar una deuda pendiente
10	Listar Clientes	Muestra un listado de todos los clientes ya registrados en el sistema
11	Agregar Clientes	Agrega un nuevo cliente al sistema
12	Modificar Clientes	Permite modificar los datos de los clientes
13	Eliminar Clientes	Elimina un cliente del sistema
14	Listar Vendedores	Muestra un listado de todos los vendedores ya registrados en el sistema
15	Agregar Vendedores	Agrega un nuevo vendedor al sistema
16	Modificar Vendedores	Permite modificar los datos de los vendedores
17	Eliminar Vendedores	Elimina un vendedor del sistema
18	Listar Platos	Muestra un listado de todos los platos ya registrados en el sistema
19	Agregar Platos (*)	Agrega un nuevo plato al sistema
20	Modificar Platos	Permite modificar los datos de los platos

21	Eliminar Platos	Elimina un plato del sistema

Figura 3.3 Lista de programas.

3.4 DESCRIPCIÓN DE PROGRAMAS

3.4.1 Inicio de Sesión

Nombre: LoginControl

Objetivo: Identificar a un vendedor en el sistema para otorgarle acceso a características reservadas mediante el uso de un usuario y contraseña.

Diagrama de bloques

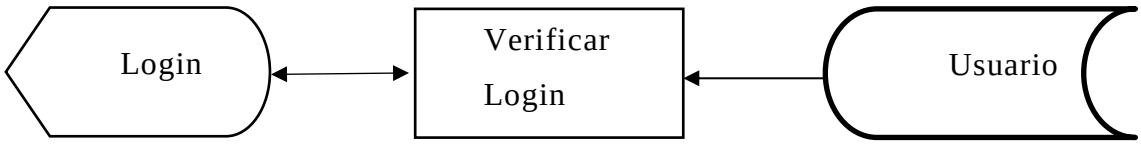


Figura 3.4 Diagrama de bloque de “Inicio de Sesión”.

Reglas del proceso: Se solicitan los datos del vendedor, si estos están correctos se le permite acceder a la funcionalidad seleccionada.

Código Fuente: Ver anexo 1 (página 43).

Diseño de pantalla

Figura 3.5 Pantalla de “Inicio de Sesión”.

3.4.2 Menú principal

Nombre: frmPrincipal

Objetivo: Navegar por los distintos controles de usuario mediante un menú lateral desplegable.

Diagrama de bloques:

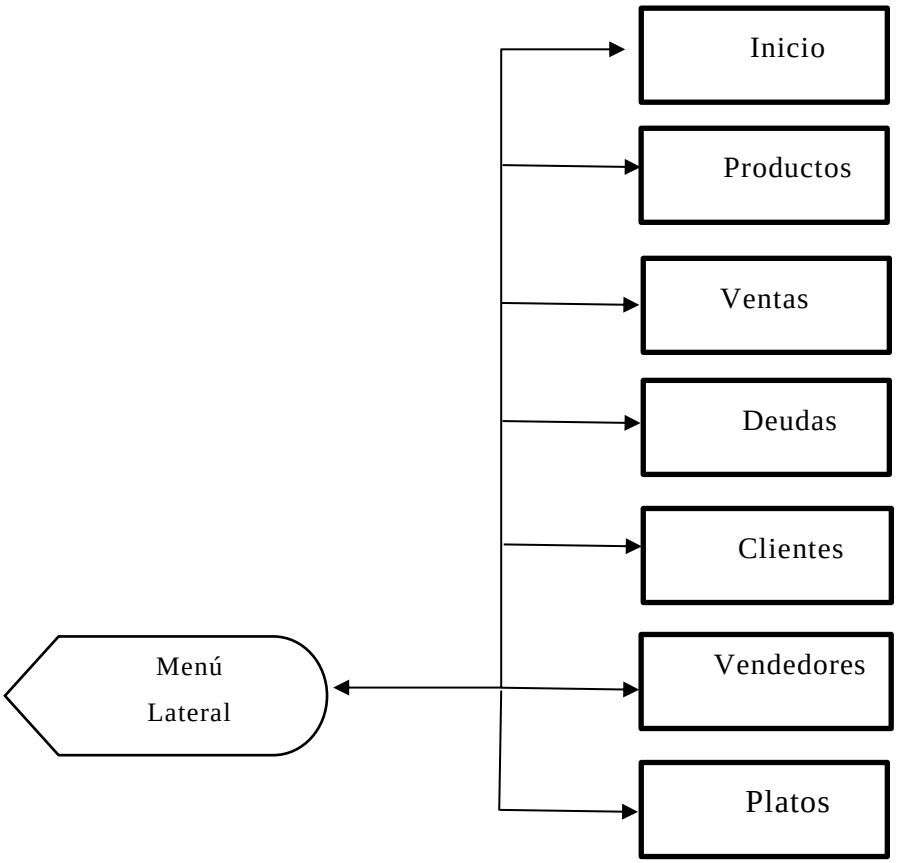


Figura 3.6 Diagrama de bloques del “menú principal”.

Reglas del proceso: Menú principal que se despliega al iniciar la aplicación.

Código Fuente: Ver anexo 2 (página 45).

Diseño de pantalla:

Figura 3.7 Menú principal

3.4.3 Modificar Productos

Nombre: ProductosControl

Objetivo: Modificar los datos de los productos registrados en el sistema.

Diagrama de bloques:

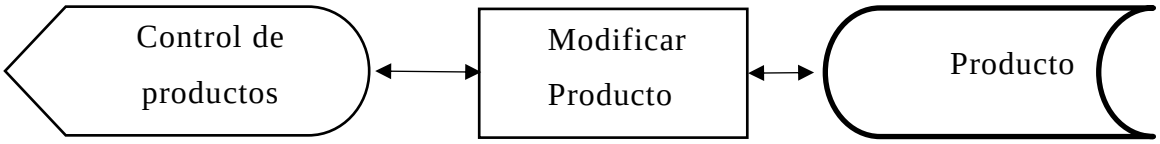


Figura 3.8 Diagrama de bloques de “modificar productos”.

Reglas del proceso: Al presionar el botón Productos, se desplegará el control completo del mantenedor de Productos. Para poder modificar un producto se debe seleccionar desde la lista que se muestra en la pantalla y presionar el botón “Modificar”. Posterior al seleccionarlo automáticamente se llenarán los campos con la información del producto que se seleccionó para modificarlo se deben cambiar las cajas de textos o Combo box correspondientes, dependiendo de lo que se desea modificar. Para realizar la modificación simplemente se debe hacer clic en el botón “Aceptar”. Para que se realice la modificación se validará previamente, que, en caso de modificar el stock del producto, éste sea mayor a 0, por otro lado, se valida que al seleccionar tipo “Preparación” no se pueda ingresar un precio ya que estos productos no se venden directamente. En caso de pasar las validaciones se procede a realizar la modificación y se muestra un mensaje emergente con un mensaje de éxito.

Código Fuente: Ver anexo 3 (página 49).

Diseño de pantalla:

Figura 3.9 Pantalla de “productos”.

3.4.4 Nueva Venta

Nombre: VentasControl

Objetivo: Generar una nueva venta de productos asociados a un cliente y un vendedor.

Diagrama de bloques:

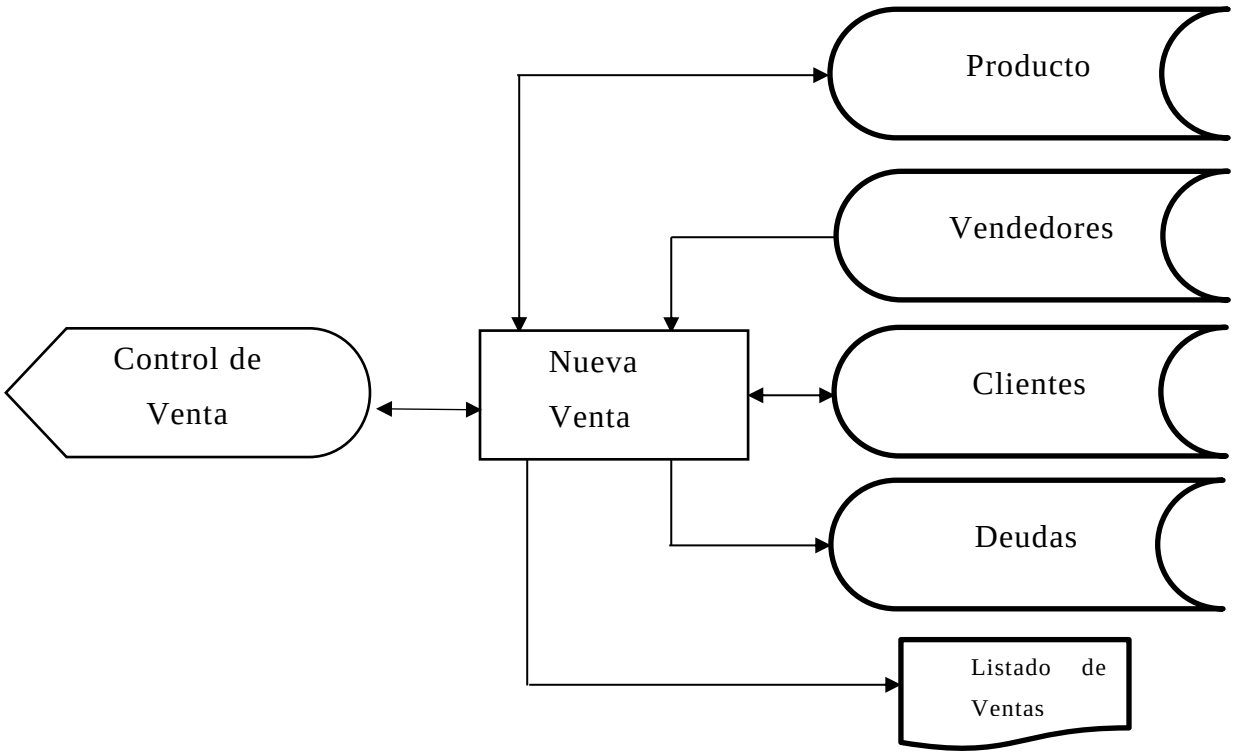


Figura 3.10 Diagrama de bloques de “nueva venta”.

Reglas del proceso: Al presionar el botón “Ventas” en el menú lateral se desplegarán 2 opciones, al seleccionar “Nueva Venta” se desplegará el control de ventas. Para poder efectuar una venta satisfactoriamente se debe seleccionar primero el usuario del cual se posee la contraseña, siguiente a esto se debe seleccionar al cliente al que se le efectúa la venta(de no estar registrado el cliente se debe seleccionar “(ninguno)”), ahora se procede a agregar los productos a la lista del detalle presionando el botón “+”, una vez agregados los productos que se solicitan se puede agregar opcionalmente un monto de deuda en la casilla de “Deuda”, finalmente para terminar el proceso de venta se debe escribir la contraseña del usuario en el campo “Contraseña” y presionar el botón “Terminar”. Cuando el sistema termine de procesar los datos aparecerá un mensaje señalando el ingreso exitoso de la venta.

Código Fuente: Ver anexo 4 (página 55).

Diseño de pantalla:

Figura 3.11 Pantalla de “nueva venta”.

3.4.5 Pagar Deudas

Nombre: DeudaControl

Objetivo: Modificar las ventas que poseen deudas vigentes para dejarlas pagadas.

Diagrama de bloques:

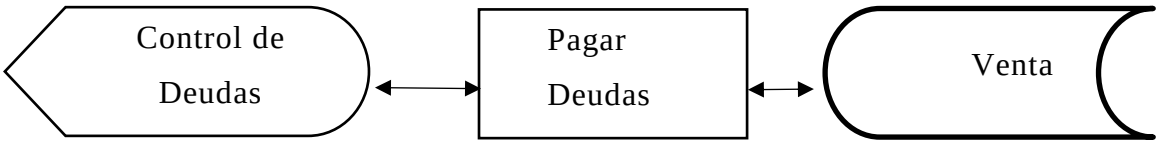


Figura 3.12 Diagrama de bloques de “Pagar Deudas”.

Reglas del proceso: Al presionar el botón “Deudas”, se desplegará el control completo de Deudas. Para poder modificar una venta y pagar su deuda se debe seleccionar desde la lista que se muestra en la pantalla, para que aparezcan ventas con deudas en la lista se debe seleccionar una fecha anterior a la actual, de esta manera se desplegarán todas las ventas realizadas entre la fecha seleccionada y el día actual, una vez seleccionada la venta debe modificar el monto a pagar en el campo “Monto a Pagar” y presionar el botón “Pagar Deuda”. Finalmente se desplegará un mensaje de éxito para comprobar el éxito del pago.

Código Fuente: Ver anexo 5 (página 61).

Diseño de pantalla:

Figura 3.13 Pantalla de “Pagar Deudas”.

3.4.6 Agregar Platos

Nombre: PlatosControl

Objetivo: Agregar un nuevo plato al sistema.

Diagrama de bloques:

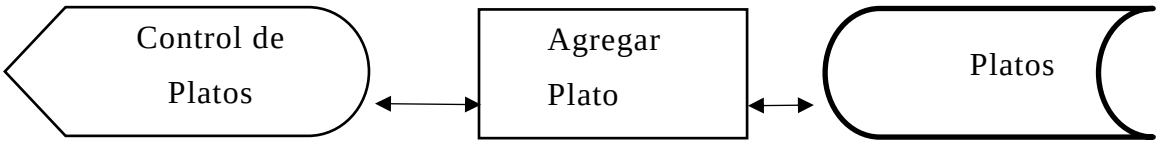


Figura 3.14 Diagrama de bloques de “Agregar Platos”.

Reglas del proceso: Al presionar el botón “Platos” en el menú lateral se desplegarán 2 opciones, al seleccionar “Nuevo Plato” se desplegará el control de platos. Para poder agregar un plato satisfactoriamente se deben seleccionar primero los productos del cual se compone este plato con sus cantidades respectivas, posterior a esto se deben llenar los siguientes campos con la información solicitada, “Nombre del plato”, “Stock” y “Precio”. Finalmente se debe presionar el botón “Agregar”. Cuando el sistema termine de procesar los datos aparecerá un mensaje señalando el ingreso exitoso del plato en el sistema.

Código Fuente: Ver anexo 6 (página 65).

Diseño de pantalla:

Figura 3.15 Pantalla de “Agregar Platos”.

CONCLUSIONES

Para poder realizar este trabajo se necesitaron aproximadamente nueve meses de trabajo desde que comenzó el análisis hasta finalizar con la construcción total del software, cabe destacar que para poder llegar a la construcción del software fue necesario interiorizarse en el lenguaje de desarrollo en este caso C#, además de buscar cada error que se producía mediante la implementación de distintos módulos, esto fue investigado durante la construcción con documentación de foros y archivos de Microsoft orientada al desarrollo.

La influencia de la formación profesional adquirida para enfrentar el desarrollo de un software desde cero es primordial ya que el desarrollo puede realizarse mediante cualquier lenguaje de programación sin importancia, mientras cumpla los requisitos para el desarrollo. Pero es importante destacar que en la formación profesional es importante el adquirir los conocimientos necesarios en la lógica de programación y en el análisis del sistema, ya que un mal análisis nos llevará a un software con errores y quizás implementaciones innecesarias.

Este sistema pretende entrar en funcionamiento a finales del año 2018 siendo instalado en la máquina personal de la administradora y configurado para su óptima utilización.

En un futuro este proyecto puede ampliarse adquiriendo distintas características que pudiesen resolver otras problemáticas específicas del negocio en el cual se integrará, como identificación de clientes mediante otra aplicación específica para ellos, pago online, reserva de almuerzos, etc.

Para finalizar podemos decir que este proyecto ayudó al aumento de otras habilidades poco desarrolladas a lo largo del camino universitario que plantea la maya curricular, como el análisis y resolución de problemas reales mediante el desarrollo de un sistema en base a programación orientada a objetos y manejo de capas, que realmente en el mundo laboral actual son las bases fundamentales para crear soluciones.

BIBLIOGRAFÍA

Plantilla de diagrama de flujos para WORD [en línea]
<Https://www.lucidchart.com/pages/es/plantilla-de-diagrama-de-flujo-para-word>

Desarrollo en C# [en línea]
<https://msdn.microsoft.com/es-es/library/ms173063.aspx>

Stack Overflow – Where Developers Learn, Share and Build Careers
< https://stackoverflow.com/>

ANEXO 1: CODIGO FUENTE: INICIO DE SESIÓN

```
using System;
using System.Windows.Forms;

namespace TopTurCasino.Presentación
{
    public partial class LoginControl : UserControl
    {
        private UsuarioNeg userNeg;
        private Negocio negocio = new Negocio();
        private Usuario usuario = new Usuario();
        private string tipoLogin = "";
        public LoginControl()
        {
            InitializeComponent();
        }

        private void btnIngresar_Click(object sender, EventArgs e)
        {
            try
            {
            Ingresar();
            }
            catch (Exception ex)
            {
                negocio.Error("Error de BD", ex.Message);
            }
        }

        private void Ingresar()
        {
            try
            {
            if (txtRut.Text == "" || txtPassword.Text == "")
            {
                negocio.Error("Error de Ingreso", "Ingrese todos los datos");
            }
            else
            {
                if (negocio.validarRut(txtRut.Text))
                {
                    string rut = negocio.LimpiarRut(txtRut.Text);
                    usuario = userNeg.ObtenerUsuario(rut, txtPassword.Text);
                    if (usuario.NomUsuario != null)
                    {
                        if (usuario.TipoUsuario == "Administrador")
                        {
                            txtRut.Text = "";
                            txtPassword.Text = "";
                            userNeg.LoginExitoso(usuario, tipoLogin);
                        }
                        else
                        {
                            negocio.Error("Error de Ingreso", "Debe ser
administrador para poder acceder");
                        }
                    }
                    else
                    {
```

```
                        negocio.Error("Error de Ingreso", "Rut o contraseña
incorrectas");
                    }
                }
                else
                {
                    negocio.Error("Error de Ingreso", "Rut inválido");
                }
            }
            }
            catch (Exception ex)
            {
                negocio.Error("Error de BD", ex.Message);
            }
        }

        private void LoginControl_Load(object sender, EventArgs e)
        {
            try
            {
            frmPrincipal principal = this.Parent as frmPrincipal;
            userNeg = new UsuarioNeg(principal);
            }
            catch (Exception ex)
            {
                negocio.Error("Error de BD", ex.Message);
            }
        }
        public void AgregarTipo(string tipo)
        {
            this.tipoLogin = tipo;
        }

        private void txtPassword_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (e.KeyChar == (char)Keys.Enter)
            {
                Ingresar();
            }
        }

        private void txtRut_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!(char.IsNumber(e.KeyChar)) && (e.KeyChar != (char)Keys.K) &&
(e.KeyChar != (char)Keys.Back) && (e.KeyChar != (char)45) && (e.KeyChar !=
(char)46))
            {
                e.Handled = true;
                return;
            }
        }
    }
   }
```

ANEXO 2: CODIGO FUENTE: MENÚ

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TopTurCasino.Presentación
{
    public partial class frmPrincipal : Form
    {
        public frmPrincipal()
        {
            InitializeComponent();
        }

        private void btnCerrar_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }
        public void LoginExitoso(Usuario usuario,string tipoLogin)
        {
            lblBienvenido.Text = "Bienvenido " + usuario.NomUsuario + " a Top Tur
Casino";
            switch (tipoLogin)
            {
                case "Productos":
                    productosControl1.BringToFront();
                    break;
                case "Clientes":
                    clientesControl1.BringToFront();
                    break;
                case "Vendedores":
                    usuariosControl1.BringToFront();
                    break;
                case "NuevoPlato":
                    platosControl1.BringToFront();
                    break;
                case "VerPlato":
                    platosControl1.BringToFront();
                    break;
            }
        }

        private void btnInicio_Click(object sender, EventArgs e)
        {
            MenuTransition.HideSync(pnlVentas);
            MenuTransition.HideSync(pnlPlatos);
            pnlLateral.Visible = false;
            pnlLateral.Height = btnInicio.Height;
            pnlLateral.Top = btnInicio.Top;
            SideBarTransition.ShowSync(pnlLateral);
            inicioControl1.BringToFront();
        }

        private void btnProductos_Click(object sender, EventArgs e)
```

```
{
    MenuTransition.HideSync(pnlVentas);
    MenuTransition.HideSync(pnlPlatos);
    pnlLateral.Visible = false;
    pnlLateral.Height = btnProductos.Height;
    pnlLateral.Top = btnProductos.Top;
    SideBarTransition.ShowSync(pnlLateral);
    loginControl1.AgregarTipo(((Control)sender).Tag.ToString());
    loginControl1.BringToFront();
}

private void btnVentas_Click(object sender, EventArgs e)
{
    MenuTransition.HideSync(pnlPlatos);
    pnlLateral.Visible = false;
    pnlLateral.Height = btnVentas.Height;
    pnlLateral.Top = btnVentas.Top;
    SideBarTransition.ShowSync(pnlLateral);
    pnlVentas.BringToFront();
    MenuTransition.ShowSync(pnlVentas);
}

private void btnClientes_Click(object sender, EventArgs e)
{
    MenuTransition.HideSync(pnlVentas);
    MenuTransition.HideSync(pnlPlatos);
    pnlLateral.Visible = false;
    pnlLateral.Height = btnClientes.Height;
    pnlLateral.Top = btnClientes.Top;
    SideBarTransition.ShowSync(pnlLateral);
    loginControl1.AgregarTipo(((Control)sender).Tag.ToString());
    loginControl1.BringToFront();
}

private void btnVendedores_Click(object sender, EventArgs e)
{
    MenuTransition.HideSync(pnlVentas);
    MenuTransition.HideSync(pnlPlatos);
    pnlLateral.Visible = false;
    pnlLateral.Height = btnVendedores.Height;
    pnlLateral.Top = btnVendedores.Top;
    SideBarTransition.ShowSync(pnlLateral);
    loginControl1.AgregarTipo(((Control)sender).Tag.ToString());
    loginControl1.BringToFront();
}

private void btnMenu_Click(object sender, EventArgs e)
{
    MenuTransition.HideSync(pnlVentas);
    MenuTransition.HideSync(pnlPlatos);
    if (pnlMenu.Width == 100)
    {
        pnlMenu.Hide();
        pnlMenu.Width = 300;
        pnlVentas.Left = 300;
        btnInicio.Text = "Inicio";
        btnProductos.Text = "Productos";
        btnVentas.Text = "Ventas";
        btnDeudas.Text = "Deudas";
        btnClientes.Text = "Clientes";
        btnVendedores.Text = "Vendedores";
        btnMenu.Dock = DockStyle.Right;
```

```
            MenuTransition.ShowSync(pnlMenu);
            LogoTransition.ShowSync(imgLogo);
        }
        else
        {
            LogoTransition.HideSync(imgLogo);
            pnlMenu.Hide();
            pnlMenu.Width = 100;
            pnlVentas.Left = 100;
            btnInicio.Text = "";
            btnProductos.Text = "";
            btnVentas.Text = "";
            btnDeudas.Text = "";
            btnClientes.Text = "";
            btnVendedores.Text = "";
            btnMenu.Dock = DockStyle.Fill;
            MenuTransition.ShowSync(pnlMenu);
        }
    }

    private void btnNuevaVenta_Click(object sender, EventArgs e)
    {
        MenuTransition.HideSync(pnlVentas);
        ventasControl1.BringToFront();
    }

    private void deudaControl1_Load(object sender, EventArgs e)
    {

    }

    private void btnDeudas_Click(object sender, EventArgs e)
    {
        MenuTransition.HideSync(pnlVentas);
        MenuTransition.HideSync(pnlPlatos);
        pnlLateral.Visible = false;
        pnlLateral.Height = btnDeudas.Height;
        pnlLateral.Top = btnDeudas.Top;
        SideBarTransition.ShowSync(pnlLateral);
        deudaControl1.BringToFront();
    }

    private void btnPlatos_Click(object sender, EventArgs e)
    {
        MenuTransition.HideSync(pnlVentas);
        pnlLateral.Visible = false;
        pnlLateral.Height = btnPlatos.Height;
        pnlLateral.Top = btnPlatos.Top;
        SideBarTransition.ShowSync(pnlLateral);
        pnlPlatos.BringToFront();
        MenuTransition.ShowSync(pnlPlatos);
    }

    private void btnVerVentas_Click(object sender, EventArgs e)
    {
        MenuTransition.HideSync(pnlVentas);
        verVentasControl1.BringToFront();
    }

    private void btnNuevoPlato_Click(object sender, EventArgs e)
    {
        MenuTransition.HideSync(pnlPlatos);
```

```
            loginControl1.AgregarTipo(((Control)sender).Tag.ToString());
            loginControl1.BringToFront();
        }

        private void btnVerPlatos_Click(object sender, EventArgs e)
        {
            MenuTransition.HideSync(pnlPlatos);
            loginControl1.AgregarTipo(((Control)sender).Tag.ToString());
            loginControl1.BringToFront();
        }
    }
}
```

ANEXO 3: CODIGO FUENTE: MODIFICAR PRODUCTOS

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TopTurCasino.Presentación
{
    public partial class ProductosControl : UserControl
    {
        private ProductoNeg productoNeg = new ProductoNeg();
        private Producto producto = new Producto();
        private Negocio negocio = new Negocio();
        private string accion = "";
        public ProductosControl()
        {
            InitializeComponent();
        }

        private void ProductosControl_Load(object sender, EventArgs e)
        {
            try
            {
                dgProductos.DataSource = productoNeg.ObtenerProductos();
                dgProductos.Columns[0].Visible = false;
                dgProductos.Columns[1].HeaderText = "Nombre Producto";
                dgProductos.Columns[2].HeaderText = "Categoría";
                dgProductos.Columns[3].HeaderText = "Stock";
                dgProductos.Columns[4].HeaderText = "Unidad de Medida";
                dgProductos.Columns[5].Visible = false;
                dgProductos.Columns[6].HeaderText = "Precio";
                producto = productoNeg.ObtenerProductoPorId(1);
                txtProducto.Text = producto.NomProducto;
                txtCategoria.Text = producto.Categoria;
                txtStock.Text = producto.Stock.ToString();
                txtUnidadMedida.Text = producto.UnidadMedida;
                if(producto.TipoProducto.Equals("Venta"))
                {
                    ddTipo.selectedIndex = 0;
                }
                else
                {
                    ddTipo.selectedIndex = 1;
                }
                txtPrecio.Text = producto.Precio.ToString();
            }
            catch (Exception ex)
            {
                negocio.Error("Error de BD", ex.Message);
            }
        }

        private void dgProductos_Click(object sender, EventArgs e)
        {
            try
```

```
            {
                if (dgProductos.Rows.Count > 0)
                {
                    producto = dgProductos.CurrentRow.DataBoundItem as Producto;
                    txtProducto.Text = producto.NomProducto;
                    txtCategoria.Text = producto.Categoria;
                    txtStock.Text = producto.Stock.ToString();
                    txtUnidadMedida.Text = producto.UnidadMedida;
                    if (producto.TipoProducto.Equals("Venta"))
                    {
                        ddTipo.selectedIndex = 0;
                    }
                    else
                    {
                        ddTipo.selectedIndex = 1;
                    }
                    txtPrecio.Text = producto.Precio.ToString();
                }
            }
            catch (Exception ex)
            {
                negocio.Error("Error de BD", ex.Message);
            }
        }

        private void txtBuscar_OnValueChanged(object sender, EventArgs e)
        {
            try
            {
            dgProductos.DataSource =
productoNeg.ObtenerProductosFiltrados(txtBuscar.Text);
            }
            catch (Exception ex)
            {
                negocio.Error("Error de BD", ex.Message);
            }
        }

        private void btnAgregar_Click(object sender, EventArgs e)
        {
            try
            {
                txtProducto.Text = "";
                txtCategoria.Text = "";
                txtStock.Text = "";
                txtUnidadMedida.Text = "";
                ddTipo.selectedIndex = 0;
                txtPrecio.Text = "";
                txtProducto.Enabled = true;
                txtCategoria.Enabled = true;
                txtStock.Enabled = true;
                txtUnidadMedida.Enabled = true;
                ddTipo.Enabled = true;
                txtPrecio.Enabled = true;
                btnAceptar.Enabled = true;
                btnCancelar.Enabled = true;
                btnModificar.Enabled = false;
                btnEliminar.Enabled = false;
                dgProductos.Enabled = false;
                accion = "Agregar";
            }
            catch (Exception ex)
```

```
            {
                negocio.Error("Error de BD", ex.Message);
            }
        }

        private void txtStock_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!(char.IsNumber(e.KeyChar)) && (e.KeyChar != (char)Keys.Back) &&
(e.KeyChar != (char)44))
            {
                e.Handled = true;
                return;
            }
            else
            {
                if (e.KeyChar == (char)44)
                {
                    if (txtStock.Text.Contains(","))
                    {
                        e.Handled = true;
                        return;
                    }
                }
            }
        }

        private void btnAceptar_Click(object sender, EventArgs e)
        {
            try
            {
                if (txtProducto.Text == "" || txtCategoria.Text == "" ||
txtStock.Text == "" || txtUnidadMedida.Text == "" || txtPrecio.Text == "")
                {
                    negocio.Error("Aviso de Productos", "Ingrese todos los
datos");
                }
                else
                {
                    if (txtStock.Text == "0")
                    {
                        negocio.Error("Aviso de Productos", "Ingrese un stock
válido");
                    }
                    else
                    {
                        if (txtPrecio.Enabled && txtPrecio.Text == "0")
                        {
                            negocio.Error("Aviso de Productos", "Ingrese un
precio válido");
                        }
                        else
                        {
                            if (accion == "Agregar")
                            {
                                Producto producto = new Producto();
                                producto.NomProducto = txtProducto.Text;
                                producto.Categoria = txtCategoria.Text;
                                producto.Stock = float.Parse(txtStock.Text);
                                producto.UnidadMedida = txtUnidadMedida.Text;
                                producto.TipoProducto =
ddTipo.Items[ddTipo.selectedIndex];
```

```
                                producto.Precio =
Convert.ToInt32(txtPrecio.Text);
                                int i = productoNeg.AgregarProducto(producto);
                                dgProductos.DataSource =
productoNeg.ObtenerProductos();
                                negocio.Error("Aviso de Producto","Producto
agregado con éxito");
                            }
                            if (accion == "Modificar")
                            {
                                Producto prod = new Producto();
                                prod.NomProducto = txtProducto.Text;
                                prod.Categoria = txtCategoria.Text;
                                prod.Stock = float.Parse(txtStock.Text);
                                prod.UnidadMedida = txtUnidadMedida.Text;
                                prod.TipoProducto =
ddTipo.Items[ddTipo.selectedIndex];
                                prod.Precio = Convert.ToInt32(txtPrecio.Text);
                                producto = dgProductos.CurrentRow.DataBoundItem
as Producto;
                                int i = productoNeg.ModificarProducto(prod,
producto.CodProducto);
                                dgProductos.DataSource =
productoNeg.ObtenerProductos();
                                negocio.Error("Aviso de Producto","Producto
modificado con éxito");
                            }
                            txtProducto.Text = "";
                            txtCategoria.Text = "";
                            txtStock.Text = "";
                            txtUnidadMedida.Text = "";
                            ddTipo.selectedIndex = 0;
                            txtPrecio.Text = "";
                            txtProducto.Enabled = false;
                            txtCategoria.Enabled = false;
                            txtStock.Enabled = false;
                            txtUnidadMedida.Enabled = false;
                            ddTipo.Enabled = false;
                            txtPrecio.Enabled = false;
                            btnAceptar.Enabled = false;
                            btnCancelar.Enabled = false;
                            btnAgregar.Enabled = true;
                            btnModificar.Enabled = true;
                            btnEliminar.Enabled = true;
                            dgProductos.Enabled = true;
                        }
                    }
                }
            }
            catch (Exception ex)
            {
                negocio.Error("Error de BD", ex.Message);
            }
        }

        private void btnModificar_Click(object sender, EventArgs e)
        {
            if(txtProducto.Text == "" || txtCategoria.Text == "" || txtStock.Text
== "" || txtUnidadMedida.Text == "" || txtPrecio.Text == "")
            {
                negocio.Error("Aviso de Productos", "Selecciona un producto para
modificar");
```

```
            }
            else
            {
                txtProducto.Enabled = true;
                txtCategoria.Enabled = true;
                txtStock.Enabled = true;
                txtUnidadMedida.Enabled = true;
                ddTipo.Enabled = true;
                if (ddTipo.selectedIndex == 1)
                {
                    txtPrecio.Text = "0";
                    txtPrecio.Enabled = false;
                }
                else
                {
                    txtPrecio.Enabled = true;
                }
                btnAceptar.Enabled = true;
                btnCancelar.Enabled = true;
                btnAgregar.Enabled = false;
                btnEliminar.Enabled = false;
                dgProductos.Enabled = false;
                accion = "Modificar";
            }
        }

        private void btnEliminar_Click(object sender, EventArgs e)
        {
            try
            {
                if (txtProducto.Text == "" || txtCategoria.Text == "" ||
txtStock.Text == "" || txtUnidadMedida.Text == "" || txtPrecio.Text == "")
                {
                    negocio.Error("Aviso de Productos", "Selecciona un producto
para eliminar");
                }
                else
                {
                    producto = dgProductos.CurrentRow.DataBoundItem as Producto;
                    int i = productoNeg.EliminarProducto(producto.CodProducto);
                    dgProductos.DataSource = productoNeg.ObtenerProductos();
                    txtProducto.Text = "";
                    txtCategoria.Text = "";
                    txtStock.Text = "";
                    txtUnidadMedida.Text = "";
                    ddTipo.selectedIndex = 0;
                    txtPrecio.Text = "";
                }
            }
            catch (Exception ex)
            {
                negocio.Error("Error de BD", ex.Message);
            }
        }

        private void btnCancelar_Click(object sender, EventArgs e)
        {
            txtProducto.Text = "";
            txtCategoria.Text = "";
            txtStock.Text = "";
            txtUnidadMedida.Text = "";
            ddTipo.selectedIndex = 0;
```

```
            txtPrecio.Text = "";
            txtProducto.Enabled = false;
            txtCategoria.Enabled = false;
            txtStock.Enabled = false;
            txtUnidadMedida.Enabled = false;
            ddTipo.Enabled = false;
            txtPrecio.Enabled = false;
            btnAceptar.Enabled = false;
            btnCancelar.Enabled = false;
            btnAgregar.Enabled = true;
            btnModificar.Enabled = true;
            btnEliminar.Enabled = true;
            dgProductos.Enabled = true;
        }

        private void txtPrecio_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!(char.IsNumber(e.KeyChar)) && (e.KeyChar != (char)Keys.Back))
            {
                e.Handled = true;
                return;
            }
        }

        private void ddTipo_onItemSelected(object sender, EventArgs e)
        {
            if (ddTipo.selectedIndex == 1)
            {
                txtPrecio.Text = "0";
                txtPrecio.Enabled = false;
            }
            else
            {
                txtPrecio.Text = "1";
                txtPrecio.Enabled = true;
            }
        }
    }
}
```

ANEXO 4: CODIGO FUENTE: NUEVA VENTA

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TopTurCasino.Presentación
{
    public partial class VentasControl : UserControl
    {
        private ClienteNeg clienteNeg = new ClienteNeg();
        private ProductoNeg productoNeg = new ProductoNeg();
        private UsuarioNeg usuarioNeg;
        private VentaNeg ventaNeg = new VentaNeg();
        private Negocio negocio = new Negocio();
        private List<Detalle> listaDetalle = new List<Detalle>();
        public VentasControl()
        {
            InitializeComponent();
        }

        private void VentasControl_Load(object sender, EventArgs e)
        {
            try
            {
            usuarioNeg = new UsuarioNeg(this.Parent as frmPrincipal);
            dgDetalle.DataSource = null;
            dgProductos.DataSource = productoNeg.ObtenerProductosVenta();
            dgProductos.Columns[0].Visible = false;
            dgProductos.Columns[1].HeaderText = "Producto";
            dgProductos.Columns[2].Visible = false;
            dgProductos.Columns[3].HeaderText = "Stock";
            dgProductos.Columns[4].Visible = false;
            dgProductos.Columns[5].Visible = false;
            dgProductos.Columns[6].HeaderText = "Precio";

            ddUsuario.Items = usuarioNeg.ObtenerUsuariosString();
            ddUsuario.selectedIndex = 0;
            ddCliente.Items = clienteNeg.ObtenerClientesString();
            ddCliente.selectedIndex = 0;
            lblFecha.Text = "Fecha de hoy " + DateTime.Now.ToString("dd-MM-
yyyy");
            Usuario usuario =
usuarioNeg.ObtenerUsuarioPorNombre(ddUsuario.Items[0]);
            Cliente cliente =
clienteNeg.ObtenerClientePorNombre(ddCliente.Items[0]);

            imgUsuario.Image = usuario.Foto ?
Image.FromFile("D:/Projects/TopTurCasino/TopTurCasino/Fotos/" +
usuario.RutUsuario + ".jpg") :
Image.FromFile("D:/Projects/TopTurCasino/TopTurCasino/Fotos/sinfoto.jpg");
            imgCliente.Image = cliente.Foto ?
Image.FromFile("D:/Projects/TopTurCasino/TopTurCasino/Fotos/" +
cliente.RutCliente + ".jpg") :
Image.FromFile("D:/Projects/TopTurCasino/TopTurCasino/Fotos/sinfoto.jpg");
```

```
            }
            catch (Exception ex)
            {
                negocio.Error("Error de BD", ex.Message);
            }
        }

        private void btnAgregar_Click(object sender, EventArgs e)
        {
            try
            {
            int i = 1;
            Detalle det = new Detalle();
            List<Producto> lista = dgProductos.DataSource as List<Producto>;
            if(listaDetalle.Count > 0)
            {
                foreach(Detalle d in listaDetalle)
                {
                    if (d.NomProducto ==
lista[(dgProductos.CurrentRow.Index)].NomProducto)
                    {
                        if (d.Cantidad <
lista[(dgProductos.CurrentRow.Index)].Stock)
                        {
                            d.Cantidad++;
                            i = 0;
                            int total = Convert.ToInt32(lblTotal.Text);
                            total +=
lista[(dgProductos.CurrentRow.Index)].Precio;
                            lblTotal.Text = total.ToString();
                        }
                        else
                        {
                            negocio.Error("Aviso de Stock", "Stock insuficiente
para seguir agregando");
                            i = 0;
                        }

                    }
                }

            }
            if(i == 1)
            {
                det.NomProducto =
lista[(dgProductos.CurrentRow.Index)].NomProducto;
                det.Cantidad = 1;
                det.Precio = lista[(dgProductos.CurrentRow.Index)].Precio;
                listaDetalle.Add(det);
                int total = Convert.ToInt32(lblTotal.Text);
                total += lista[(dgProductos.CurrentRow.Index)].Precio;
                lblTotal.Text = total.ToString();
            }
            dgDetalle.DataSource = null;
            dgDetalle.DataSource = listaDetalle;
            dgDetalle.Columns[0].HeaderText = "Producto";
            dgDetalle.Columns[1].HeaderText = "Cantidad";
            dgDetalle.Columns[2].HeaderText = "Precio";
            dgDetalle.Columns[3].Visible = false;
            dgDetalle.Columns[4].Visible = false;
            dgDetalle.Rows[0].Selected = true;
            }
```

```
            catch (Exception ex)
            {
                negocio.Error("Error de BD", ex.Message);
            }
        }

        private void txtBuscarProducto_OnValueChanged(object sender, EventArgs e)
        {
            try
            {
            //Filtrar
            dgProductos.DataSource =
productoNeg.ObtenerProductosFiltrados(txtBuscarProducto.Text, true);
            dgProductos.Columns[0].Visible = false;
            dgProductos.Columns[1].HeaderText = "Producto";
            dgProductos.Columns[2].Visible = false;
            dgProductos.Columns[3].HeaderText = "Stock";
            dgProductos.Columns[4].Visible = false;
            dgProductos.Columns[5].Visible = false;
            dgProductos.Columns[6].HeaderText = "Precio";
            }
            catch (Exception ex)
            {
                negocio.Error("Error de BD", ex.Message);
            }
        }

        private void txtDeuda_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!(char.IsNumber(e.KeyChar)) && (e.KeyChar != (char)Keys.Back))
            {
                e.Handled = true;
                return;
            }
        }

        private void ddUsuario_onItemSelected(object sender, EventArgs e)
        {
            try
            {
            Usuario usuario =
usuarioNeg.ObtenerUsuarioPorNombre(ddUsuario.Items[ddUsuario.selectedIndex]);
            imgUsuario.Image = usuario.Foto ?
Image.FromFile("D:/Projects/TopTurCasino/TopTurCasino/Fotos/" +
usuario.RutUsuario + ".jpg") :
Image.FromFile("D:/Projects/TopTurCasino/TopTurCasino/Fotos/sinfoto.jpg");
            }
            catch (Exception ex)
            {
                negocio.Error("Error de BD", ex.Message);
            }
        }

        private void ddCliente_onItemSelected(object sender, EventArgs e)
        {
            try
            {
            Cliente cliente =
clienteNeg.ObtenerClientePorNombre(ddCliente.Items[ddCliente.selectedIndex]);
            imgCliente.Image = cliente.Foto ?
Image.FromFile("D:/Projects/TopTurCasino/TopTurCasino/Fotos/" +
```

```
cliente.RutCliente + ".jpg") :
Image.FromFile("D:/Projects/TopTurCasino/TopTurCasino/Fotos/sinfoto.jpg");
            }
            catch (Exception ex)
            {
                negocio.Error("Error de BD", ex.Message);
            }
        }

        private void btnEliminar_Click(object sender, EventArgs e)
        {
            try
            {
                if(listaDetalle.Count > 0)
                {
                    if (dgDetalle.CurrentRow == null)
                    {
                        negocio.Error("Error de Producto", "Seleccione un
producto para eliminar");
                    }
                    int total = Convert.ToInt32(lblTotal.Text);
                    total -= listaDetalle[(dgDetalle.CurrentRow.Index)].Precio;
                    if (listaDetalle[dgDetalle.CurrentRow.Index].Cantidad > 1)
                    {
                        listaDetalle[dgDetalle.CurrentRow.Index].Cantidad--;
                    }
                    else
                    {

listaDetalle.Remove(listaDetalle[dgDetalle.CurrentRow.Index]);
                    }
                    dgDetalle.DataSource = null;
                    dgDetalle.DataSource = listaDetalle;
                    dgDetalle.Columns[0].HeaderText = "Producto";
                    dgDetalle.Columns[1].HeaderText = "Cantidad";
                    dgDetalle.Columns[2].HeaderText = "Precio";
                    dgDetalle.Columns[3].Visible = false;
                    dgDetalle.Columns[4].Visible = false;
                    lblTotal.Text = total.ToString();
                }
            }
            catch (Exception ex)
            {
                negocio.Error("Error de BD", ex.Message);
            }
        }

        private void btnTerminar_Click(object sender, EventArgs e)
        {
            try
            {
                Usuario usuario =
usuarioNeg.ObtenerUsuarioPorNombre(ddUsuario.Items[ddUsuario.selectedIndex]);
                Cliente cliente =
clienteNeg.ObtenerClientePorNombre(ddCliente.Items[ddCliente.selectedIndex]);
                Usuario user = usuarioNeg.ObtenerUsuario(usuario.RutUsuario,
txtPassword.Text);
                if (user.NomUsuario != null)
                {
                    if(listaDetalle.Count > 0)
                    {
```

```
                        int deuda = txtDeuda.Text == "" ? 0 :
Convert.ToInt32(txtDeuda.Text);
                        int total = Convert.ToInt32(lblTotal.Text);
                        if (deuda <= total)
                        {
                            Venta venta = new Venta();
                            venta.RutCliente = cliente.RutCliente;
                            venta.RutUsuario = usuario.RutUsuario;
                            venta.Fecha = DateTime.Now;
                            venta.ValorDeuda = deuda;
                            venta.ValorTotal = total;
                            ventaNeg.AgregarVenta(venta);
                            int numVenta = ventaNeg.ObtenerUltimaVenta();
                            foreach(Detalle d in listaDetalle)
                            {

                                d.NumVenta = numVenta;
                                Producto prod =
productoNeg.ObtenerProductoPorNombre(d.NomProducto);
                                d.CodProducto = prod.CodProducto;
                                int i = ventaNeg.AgregarDetalleVenta(d);
                                prod.Stock = Convert.ToInt32(prod.Stock) -
d.Cantidad;
                                i =
productoNeg.ModificarProducto(prod,prod.CodProducto);
                                cliente =
clienteNeg.ObtenerClientePorRut(venta.RutUsuario);
                                cliente.DeudaCliente += deuda;
                                i = usuarioNeg.ModificarUsuario(usuario);
                                dgDetalle.DataSource = null;
                                dgProductos.DataSource =
productoNeg.ObtenerProductosVenta();
                                listaDetalle = new List<Detalle>();
                                txtPassword.Text = "";
                                txtBuscarProducto.Text = "";
                                txtDeuda.Text = "";
                                lblTotal.Text = "0";
                                negocio.Error("Aviso de Venta", "La venta ha sido
realizada con éxito");
                            }
                        }
                        else
                        {
                            negocio.Error("Aviso de deuda", "La deuda asignada no
puede ser mas alta que el total de la compra");
                        }
                    }
                    else
                    {
                        negocio.Error("Aviso de productos", "Agregue algún
producto a la venta");
                    }
                }
                else
                {
                    negocio.Error("Aviso de validación", "Error en la
contraseña");
                }
            }
            catch (Exception ex)
            {
                negocio.Error("Error de BD", ex.Message);
```

```
            }
        }

        private void txtDeuda_OnValueChanged(object sender, EventArgs e)
        {

        }
    }
}
```

ANEXO 5: CODIGO FUENTE: PAGAR DEUDA

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TopTurCasino.Presentación
{
    public partial class DeudaControl : UserControl
    {
        private UsuarioNeg usuarioNeg = new UsuarioNeg();
        private ClienteNeg clienteNeg = new ClienteNeg();
        private VentaNeg ventaNeg = new VentaNeg();
        private Negocio negocio = new Negocio();
        public DeudaControl()
        {
            InitializeComponent();
        }

        private void DeudaControl_Load(object sender, EventArgs e)
        {
            try
            {
                dgVentas.DataSource =
ventaNeg.ObtenerDeudasFiltradas(txtBuscarVendedor.Text, txtBuscarCliente.Text,
dpFecha.Value);
                dgVentas.Columns[0].HeaderText = "RUT Cliente";
                dgVentas.Columns[1].HeaderText = "RUT Vendedor";
                dgVentas.Columns[2].HeaderText = "Fecha de venta";
                dgVentas.Columns[3].HeaderText = "Deuda";
                dgVentas.Columns[4].HeaderText = "Total";
                dpFecha.Value = DateTime.Now;
            }
            catch (Exception ex)
            {
                negocio.Error("Error de BD", ex.Message);
            }
        }

        private void dgVentas_Click(object sender, EventArgs e)
        {
            try
            {
                if (dgVentas.Rows.Count > 0)
                {
                    Venta venta = dgVentas.CurrentRow.DataBoundItem as Venta;
                    txtDeuda.Text = venta.ValorDeuda.ToString();
                }
            }
            catch (Exception ex)
            {
                negocio.Error("Error de BD", ex.Message);
            }
        }
```

```
        private void txtDeuda_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!(char.IsNumber(e.KeyChar)) && (e.KeyChar != (char)Keys.Back))
            {
                e.Handled = true;
                return;
            }
        }

        private void btnDeuda_Click(object sender, EventArgs e)
        {
            try
            {
                if (dgVentas.CurrentRow == null)
                {
                    negocio.Error("Aviso de Deuda", "Debe seleccionar una
deuda");
                }
                else
                {
                    if (txtDeuda.Text == "")
                    {
                        Venta venta = dgVentas.CurrentRow.DataBoundItem as Venta;
                        Cliente cliente =
clienteNeg.ObtenerClientePorRut(venta.RutCliente);
                        cliente.DeudaCliente -= Convert.ToInt32(txtDeuda.Text);
                        int i = clienteNeg.ModificarCliente(cliente);
                        venta.ValorDeuda -= Convert.ToInt32(txtDeuda.Text);
                        i = ventaNeg.ModificarVenta(venta);
                        dgVentas.DataSource =
ventaNeg.ObtenerDeudasFiltradas(txtBuscarVendedor.Text, txtBuscarCliente.Text,
dpFecha.Value);
                        dgVentas.Columns[0].HeaderText = "RUT Cliente";
                        dgVentas.Columns[1].HeaderText = "RUT Vendedor";
                        dgVentas.Columns[2].HeaderText = "Fecha de venta";
                        dgVentas.Columns[3].HeaderText = "Deuda";
                        dgVentas.Columns[4].HeaderText = "Total";
                    }
                    else
                    {
                        negocio.Error("Aviso de Deuda", "Debe ingresar un monto a
pagar");
                    }
                }
            }
            catch (Exception ex)
            {
                negocio.Error("Error de BD", ex.Message);
            }
        }

        private void dpFecha_onValueChanged(object sender, EventArgs e)
        {
            try
            {
                dgVentas.DataSource =
ventaNeg.ObtenerDeudasFiltradas(txtBuscarVendedor.Text,txtBuscarCliente.Text,dpFe
cha.Value);
            }
            catch (Exception ex)
            {
                negocio.Error("Error de BD", ex.Message);
```

```
            }
        }

        private void txtBuscarCliente_OnValueChanged(object sender, EventArgs e)
        {
            try
            {
                dgVentas.DataSource =
ventaNeg.ObtenerDeudasFiltradas(txtBuscarVendedor.Text, txtBuscarCliente.Text,
dpFecha.Value);
            }
            catch(Exception ex)
            {
                negocio.Error("Error de BD", ex.Message);
            }
}

        private void txtBuscarVendedor_OnValueChanged(object sender, EventArgs e)
        {
            try
            {
                dgVentas.DataSource =
ventaNeg.ObtenerDeudasFiltradas(txtBuscarVendedor.Text, txtBuscarCliente.Text,
dpFecha.Value);
            }
            catch (Exception ex)
            {
                negocio.Error("Error de BD", ex.Message);
            }
        }

        private void txtBuscarVendedor_KeyPress(object sender, KeyPressEventArgs
e)
        {
            if (!(char.IsNumber(e.KeyChar)) && (e.KeyChar != (char)Keys.Back) &&
(e.KeyChar != (char)Keys.K))
            {
                e.Handled = true;
                return;
            }
            else
            {
                if (e.KeyChar == (char)45)
                {
                    if (txtBuscarCliente.Text.Contains("-"))
                    {
                        e.Handled = true;
                        return;
                    }
                }
            }
        }

        private void txtBuscarCliente_KeyPress(object sender, KeyPressEventArgs
e)
        {
            if (!(char.IsNumber(e.KeyChar)) && (e.KeyChar != (char)Keys.Back) &&
(e.KeyChar != (char)Keys.K) && (e.KeyChar != (char)45))
            {
                e.Handled = true;
                return;
            }
```

```
            else
            {
                if (e.KeyChar == (char)45)
                {
                    if (txtBuscarCliente.Text.Contains("-"))
                    {
                        e.Handled = true;
                        return;
                    }
                }
            }
        }
    }
}
```

ANEXO 6: CODIGO FUENTE: AGREGAR PLATO

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TopTurCasino.Presentación
{
    public partial class PlatosControl : UserControl
    {
        private ProductoNeg productoNeg = new ProductoNeg();
        private PlatoNeg platoNeg = new PlatoNeg();
        private Negocio negocio = new Negocio();
        private List<Producto_Plato> listaPlatos = new List<Producto_Plato>();
        public PlatosControl()
        {
            InitializeComponent();
        }

        private void lblDeuda_Click(object sender, EventArgs e)
        {

        }

        private void PlatosControl_Load(object sender, EventArgs e)
        {
            try
            {
                dgProductos.DataSource =
productoNeg.ObtenerProductosPreparacion();
                dgProductos.Columns[0].Visible = false;
                dgProductos.Columns[1].HeaderText = "Producto";
                dgProductos.Columns[2].Visible = false;
                dgProductos.Columns[3].HeaderText = "Stock";
                dgProductos.Columns[4].Visible = false;
                dgProductos.Columns[5].Visible = false;
                dgProductos.Columns[6].Visible = false;
                dgPlato.DataSource = null;
            }
            catch (Exception ex)
            {
                negocio.Error("Error de BD", ex.Message);
            }
        }

        private void btnAgregar_Click(object sender, EventArgs e)
        {
            try
            {
                if (txtCantidad.Text == "" || txtCantidad.Text == "0")
                {
                    negocio.Error("Aviso de Cantidad", "Ingrese una cantidad");
                }
                else
                {
```

```
                    int i = 1;
                    Detalle det = new Detalle();
                    List<Producto> lista = dgProductos.DataSource as
List<Producto>;
                    Producto_Plato productoPlato = new Producto_Plato();
                    Producto_Plato productoLista = new Producto_Plato();
                    productoPlato.NomProducto =
lista[dgProductos.CurrentRow.Index].NomProducto;
                    productoPlato.CodProducto =
lista[dgProductos.CurrentRow.Index].CodProducto;
                    productoPlato.Cantidad = float.Parse(txtCantidad.Text);
                    productoPlato.UnidadMedida =
lista[dgProductos.CurrentRow.Index].UnidadMedida;
                    if (listaPlatos.Count > 0)
                    {
                        foreach (Producto_Plato p in listaPlatos)
                        {
                            if (p.CodProducto == productoPlato.CodProducto)
                            {
                                productoLista = p;
                            }
                        }
                        if (productoLista.CodProducto != 0)
                        {
                            listaPlatos.Remove(productoLista);
                        }
                    }
                    listaPlatos.Add(productoPlato);
                    dgPlato.DataSource = null;
                    dgPlato.DataSource = listaPlatos;
                    dgPlato.Columns[0].HeaderText = "Producto";
                    dgPlato.Columns[1].Visible = false;
                    dgPlato.Columns[2].Visible = false;
                    dgPlato.Columns[3].HeaderText = "Stock";
                    dgPlato.Columns[4].HeaderText = "Unidad de Medida";
                }
            }
            catch (Exception ex)
            {
                negocio.Error("Error de BD", ex.Message);
            }
        }

        private void btnEliminar_Click(object sender, EventArgs e)
        {
            if (listaPlatos.Count == 0)
            {
                negocio.Error("Aviso de Producto", "No hay productos para
eliminar");
            }
            else
            {
                if (dgPlato.CurrentRow == null)
                {
                    negocio.Error("Aviso de Producto", "Seleccione un producto
para eliminar");
                }
                else
                {
                    listaPlatos.Remove(listaPlatos[dgPlato.CurrentRow.Index]);
                    dgPlato.DataSource = null;
                    dgPlato.DataSource = listaPlatos;
```

```
                    dgPlato.Columns[0].Visible = false;
                    dgPlato.Columns[1].HeaderText = "Producto";
                    dgPlato.Columns[2].Visible = false;
                    dgPlato.Columns[3].HeaderText = "Stock";
                    dgPlato.Columns[4].HeaderText = "Unidad de Medida";
                }
            }
        }

        private void btnEnviar_Click(object sender, EventArgs e)
        {
            try
            {
                if (listaPlatos.Count == 0)
                {
                    negocio.Error("Aviso de Plato", "El plato debe contener
productos");
                }
                else
                {
                    if (txtNombre.Text.Equals("") || txtStock.Text.Equals("") ||
txtPrecio.Text.Equals(""))
                    {
                        negocio.Error("Aviso de Plato", "Ingrese todos los
datos");
                    }
                    else
                    {
                        Plato plato = new Plato();
                        plato.CodPlato = 0;
                        plato.NomPlato = txtNombre.Text;
                        plato.Precio = Convert.ToInt32(txtPrecio.Text);
                        platoNeg.AgregarPlato(plato);
                        plato = platoNeg.ObtenerPlato(txtNombre.Text);
                        foreach (Producto_Plato p in listaPlatos)
                        {
                            p.CodPlato = plato.CodPlato;
                            platoNeg.AgregarProductoPlato(p);
                        }
                        txtNombre.Text = "";
                        txtStock.Text = "";
                        txtPrecio.Text = "";
                        txtCantidad.Text = "1";
                        dgPlato.DataSource = null;
                        listaPlatos = new List<Producto_Plato>();
                        negocio.Error("Aviso de Plato", "Plato ingresado con
éxito");
                    }
                }
            }
            catch (Exception ex)
            {
                negocio.Error("Error de BD", ex.Message);
            }
        }

        private void txtCantidad_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!(char.IsNumber(e.KeyChar)) && (e.KeyChar != (char)Keys.Back) &&
(e.KeyChar != (char)44))
            {
                e.Handled = true;
```

```
                return;
            }
            else
            {
                if (e.KeyChar == (char)44)
                {
                    if (txtCantidad.Text.Contains(","))
                    {
                        e.Handled = true;
                        return;
                    }
                }
            }
        }

        private void txtStock_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!(char.IsNumber(e.KeyChar)) && (e.KeyChar != (char)Keys.Back))
            {
                e.Handled = true;
                return;
            }
        }

        private void txtPrecio_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!(char.IsNumber(e.KeyChar)) && (e.KeyChar != (char)Keys.Back))
            {
                e.Handled = true;
                return;
            }
        }

        private void txtBuscarProducto_OnValueChanged(object sender, EventArgs e)
        {
            try
            {
                dgProductos.DataSource =
productoNeg.ObtenerProductosFiltrados(txtBuscarProducto.Text);
            }
            catch (Exception ex)
            {
                negocio.Error("Error de BD", ex.Message);
            }
        }
    }
}
```