INSTITUTO TECNOLOGICO
"CORDILLERA"

CARRERA DE ANALISIS Y SISTEMAS

MEJORAMIENTO DEL PROCESO DE FACTURACIÓN MEDIANTE UNA APLICACIÓN WEB DE FACTURACIÓN ELECTRÓNICA PARA LA CAFETERÍA DELICIAS DE VERDE UBICADA EN LA PARROQUIA CALDERÓN

Proyecto de investigación previo a la obtención del título de Tecnólogo en Análisis de Sistemas.

Autora: Tibán Oña Alba Marina

Tutor: Ing. Juan Minango

Quito, Abril 2015

APROBACIÓN DEL TUTOR Y LECTOR

En mi calidad de tutor del trabajo sobre el tema: "MEJORAMIENTO DEL PROCESO DE FACTURACIÓN MEDIANTE UNA APLICACIÓN WEB DE FACTURACIÓN ELECTRÓNICA PARA LA CAFETERÍA DELICIAS DE VERDE UBICADA EN LA PARROQUIA CALDERÓN" presentado por la ciudadana: Alba Marina Tibán Oña, estudiante de la Escuela de Sistemas, considero que dicho informe reúne los requisitos y méritos suficientes para ser sometido a la evaluación por parte del Tribunal de Grado, que el Honorable Consejo de Escuela designe, para su correspondiente estudio y calificación.

Quito, Marzo del 2015

----------	----------
Ing. Juan Minango	**Ing. Johnny Coronel**
TUTOR	**LECTOR**

DECLARATORIA

Declaro que la investigación es absolutamente original, autentica, personal, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes. Las ideas, doctrinas, resultados y conclusiones a los que he llegado son de mi absoluta responsabilidad.

Alba Marina Tibán Oña
C.C.172405716-9

CONTRATO DE CESIÓN SOBRE DERECHOS PROPIEDAD INTELECTUAL

Comparecen a la celebración del presente contrato de cesión y transferencia de derechos de propiedad intelectual, por una parte, el estudiante Alba Marina Tibán Oña, por sus propios y personales derechos, a quien en lo posterior se le denominará el "CEDENTE"; y, por otra parte, el INSTITUTO SUPERIOR TECNOLÓGICO CORDILLERA, representado por su Rector el Ingeniero Ernesto Flores Córdova, a quien en lo posterior se lo denominará el "CESIONARIO". Los comparecientes son mayores de edad, domiciliados en esta ciudad de Quito Distrito Metropolitano, hábiles y capaces para contraer derechos y obligaciones, quienes acuerdan al tenor de las siguientes cláusulas:

PRIMERA: ANTECEDENTE.- a) El Cedente dentro del pensum de estudio en la carrera de análisis de sistemas que imparte el Instituto Superior Tecnológico Cordillera, y con el objeto de obtener el título de Tecnólogo en Análisis de Sistemas, el estudiante participa en el proyecto de grado denominado "MEJORAMIENTO DEL PROCESO DE FACTURACIÓN MEDIANTE UNA APLICACIÓN WEB DE FACTURACIÓN ELECTRÓNICA PARA LA CAFETERÍA DELICIAS DE VERDE UBICADA EN LA PARROQUIA CALDERÓN.", el cual incluye la creación y desarrollo del programa de ordenador o software, para lo cual ha implementado los conocimientos adquiridos en su calidad de alumno. b) Por iniciativa y responsabilidad del Instituto Superior Tecnológico Cordillera se desarrolla la creación del programa de ordenador, motivo por el cual se regula de forma clara la cesión de los derechos de autor que genera la obra literaria y que es

producto del proyecto de grado, el mismo que culminado es de plena aplicación técnica, administrativa y de reproducción.

SEGUNDA: CESIÓN Y TRANSFERENCIA.- Con el antecedente indicado, el Cedente libre y voluntariamente cede y transfiere de manera perpetua y gratuita todos los derechos patrimoniales del programa de ordenador descrito en la cláusula anterior a favor del Cesionario, sin reservarse para sí ningún privilegio especial (código fuente, código objeto, diagramas de flujo, planos, manuales de uso, etc.). El Cesionario podrá explotar el programa de ordenador por cualquier medio o procedimiento tal cual lo establece el Artículo 20 de la Ley de Propiedad Intelectual, esto es, realizar, autorizar o prohibir, entre otros: a) La reproducción del programa de ordenador por cualquier forma o procedimiento; b) La comunicación pública del software; c) La distribución pública de ejemplares o copias, la comercialización, arrendamiento o alquiler del programa de ordenador; d) Cualquier transformación o modificación del programa de ordenador; e) La protección y registro en el IEPI el programa de ordenador a nombre del Cesionario; f) Ejercer la protección jurídica del programa de ordenador; g) Los demás derechos establecidos en la Ley de Propiedad Intelectual y otros cuerpos legales que normen sobre la cesión de derechos de autor y derechos patrimoniales.

TERCERA: OBLIGACIÓN DEL CEDENTE.- El cedente no podrá transferir a ningún tercero los derechos que conforman la estructura, secuencia y organización del programa de ordenador que es objeto del presente contrato, como tampoco emplearlo o utilizarlo a título personal, ya que siempre se deberá guardar la exclusividad del programa de ordenador a favor del Cesionario.

CUARTA: CUANTIA.- La cesión objeto del presente contrato, se realiza a título gratuito y por ende el Cesionario ni sus administradores deben cancelar valor alguno o regalías por este contrato y por los derechos que se derivan del mismo.

QUINTA: PLAZO.- La vigencia del presente contrato es indefinida.

SEXTA: DOMICILIO, JURISDICCIÓN Y COMPETENCIA.- Las partes fijan como su domicilio la ciudad de Quito. Toda controversia o diferencia derivada de éste, será resuelta directamente entre las partes y, si esto no fuere factible, se solicitará la asistencia de un Mediador del Centro de Arbitraje y Mediación de la Cámara de Comercio de Quito. En el evento que el conflicto no fuere resuelto mediante este procedimiento, en el plazo de diez días calendario desde su inicio, pudiendo prorrogarse por mutuo acuerdo este plazo, las partes someterán sus controversias a la resolución de un árbitro, que se sujetará a lo dispuesto en la Ley de Arbitraje y Mediación, al Reglamento del Centro de Arbitraje y Mediación de la Cámara de comercio de Quito, y a las siguientes normas: a) El árbitro será seleccionado conforme a lo establecido en la Ley de Arbitraje y Mediación; b) Las partes renuncian a la jurisdicción ordinaria, se obligan a acatar el laudo arbitral y se comprometen a no interponer ningún tipo de recurso en contra del laudo arbitral; c) Para la ejecución de medidas cautelares, el árbitro está facultado para solicitar el auxilio de los funcionarios públicos, judiciales, policiales y administrativos, sin que sea necesario recurrir a juez ordinario alguno; d) El procedimiento será confidencial y en derecho; e) El lugar de arbitraje serán las instalaciones del centro de arbitraje y mediación de la Cámara de Comercio de Quito; f) El idioma del arbitraje será el español; y, g) La reconvención, caso de haberla, seguirá los mismos procedimientos antes indicados para el juicio principal.

SÉPTIMA: ACEPTACIÓN.- Las partes contratantes aceptan el contenido del presente contrato, por ser hecho en seguridad de sus respectivos intereses.

En aceptación firman a los 06 días del mes de abril del dos mil quince.

f) _____
C.C. 1724428980
CEDENTE

f) _____
Instituto Superior Tecnológico Cordillera
CESIONARIO

AGRADECIMIENTO

Agradesco a Dios, quien me protege y guía mi vida. Quien no me deja rendirme quien me muestra que con esfuerzo, constancia y perseverancia podemos cumplir todos nuestros sueño.

A mi familia por apoyarme incondicionalmente en todo momento

Al Instituto Tecnológico Superior "CORDILLERA" por guiarme en mi estudio y permitirme formarme como profesional.

A mis amigos quienes me , motivaron y dieron las fuerzas para seguir en este reto.

A mi tutor y lector: Ing. Juan Minango e Ing. Johnny Coronel quienes fueron guías para la elaboración del proyecto que presento.

A la Empresa la cual fue la herramienta principal para este trabajo, la misma que me facilitó la información necesaria para poder hacer realidad la presente investigación

DEDICATORIA

A Dios por brindarme la Fe, Fortaleza, y esperanza.

A mis hermanas por el apoyo incondicional.

A mi pequeño querubín Carlitos Daniel, quien vino a completar mi vida y forma parte de este gran sueño a pesar de las dificultades , que la vida nos ha puesto en el camino

A mis verdaderos amigos que siempre estuvieron apoyándome en todo momento.

INDICE GENERAL

Indice de Tablas

Indice de Figuras

Resumen Ejecutivo

El presente proyecto de Investigación, tiene como finalidad mejorar el proceso de facturación.La intención de mejorar el proceso de facturación en la Cafeteria Delicias de verde a sido un cambio de organización y producción a nivel de tecnología. La aplicación es amigable para el usuario ya que permitirá llevar todos los registros de una mejor manera. El cual se encarga que los empleados interactúen con la aplicación y puedan realizar ingresos de producto, facturar, y generar reportes.Los grandes beneficiarios de la aplicación, son los cajeros, Gerentes propietarios , ya que, obtendrán una información más real y ordenada , de las ventas realizadas en el día .Los objetivos planteados se cumplen a cabalidad, dejando una gran satisfacción al cliente. Dentro de la aplicación se maneja de manera ágil la información necesaria para la Cafeteria.Para realizar este trabajo de forma personal a sido una construcción de varias investigaciones para lograr un adecuado nivel de satisfacción para la Cafeteria Delicias de Verde.Con el fin de rendir en el tiempo antes acostumbrado de una mejor manera de la exitencia de un nuevo habito al que ya estaban acostumbrados a manejar.

Abstract

This research project, aims to improve the process of facturación.La intention to improve the billing process in Delicias green Cafeteria been a change of organization and production technology level. The application is user friendly as it will maintain all records in a better way. Which ensures that employees interact with the application and can make product revenue, billing, and generate reportes.Los major beneficiaries of the application are cashiers, managers owners, as will obtain a more realistic and orderly information, sales on the day .The objectives are met fully, leaving a large customer satisfaction. Within the application is handled agile information needed for this work Cafeteria.Para personally been to a construction of several investigations to achieve an adequate level of satisfaction for Delicias Cafeteria Verde.Con to perform in I used the time before a better way of exitencia a new habit to those already accustomed to handling.

Capítulo I: Antecedentes

1.01. Contexto

El Servicio De Rentas Internas SRI define a la factura electrónica como un "comprobante de venta autorizado previamente por el mismo que se respalda en las transacciones efectuadas por los contribuyentes en la trasferencia de bienes o por la prestación de servicios o la realización de otras transacciones asignadas con tributos.

Estos podrán ser llenados de forma manual ,mecánica o a través de sistemas computarizados .Las facturas en original y copia deben ser llenados en forma simultanea mediante el uso del papel carbón, carbonato o autocopiado químico; en cualquier caso contrario no serán válidas .La falta de emisión o entrega de documentos autorizados ,la emisión incompleta o falsa de estos, constituyen casos defraudación que serán sancionados de conformidad con el código Tributario".(Servicio de Rentas Internas,2012,pág.15,Comprovante de venta y retención, Extraído el 22 de marzo del 2013,http://cef.sri.gob.ec/virtualcef/file.php/1/MaterialCursosVirtuales/CVR Sistema de Facturacion.pdf).

"Vender es una premisa indispensable en cualquier negocio, pero no gestionar adecuadamente los efectos positivos de esas ventas, que al final no es otra cosa que dinero, puede resultar un problema, que mal gestionado puede llevar a la empresa a morir de éxito".(Esker,2012, Hablado de soluciones :automatización de

facturas a clientes, Extraído el 22 de marzo del 2013,http://perderlospapelespuntocom.wordpress.com/2013/12/12,hablando-de-soluciones-automatizacion-de-facturas-a-clientes/).

Realizar el proceso de facturación inadecuadamente conlleva unos procesos tediosos contratiempos e inconsistencias en la información procesada. Hasta el mínimo error puede hacer que este proceso sea ineficaz, consumiendo rápidamente el capital y los recursos disponibles.

Como consecuencia de los métodos empleados por las empresa Negocios y Servicios hay facturas acumuladas, extraviadas y en algunos casos es necesario realizar una nueva factura, lo cual implica un desajuste en la secuencia de facturas .Cabe mencionar el deterioro del material empleado en la elaboración de las actividades administrativas y la necesidad de mayor espacio físico, consecuencia de la acumulación de documentos en la utilización de este método de trabajo. Además de la ineficiencia de los procesos el costo asociado a la papelería y mobiliario necesario es cada vez mayor.

1.02. **Justificación**

La organización de la información y el rápido acceso resulta muy útil para una empresa, debido a que esta permite una eficaz toma de decisiones para la organización, minimiza, errores y disminuye la pérdida de Información.

La facturación permite mejorar la rentabilidad, obtener mayor eficiencia en los procesos mejorando la gestión de cobros y la calidad del servicio que presta esta empresa. Al llevar un correcto proceso de generación de las facturas se logrará una

reducción del tiempo de gestión de los documentos y de errores de facturación y a la vez mejorar la atención del cliente.

Por otra parte se garantizará que la información correcta estará disponible para las personas en el momento adecuado. La factura es un documento de mucha importancia y debe ser emitida con la mayor precisión ,claridad y exactitud que aumente la seguridad.

1.03. Definición del Problema Central. (Matriz T)

Tabla 1

Matriz de análisis de Fuerzas T **(Ver Anexo A.01)**

Capítulo II: Análisis de Involucrados

2.01. Requerimientos

2.01.1 Descripción del sistema actual

En la actualidad La Cafetería Delicias de Verde realiza el proceso de una manera no automatizado, cuentan con una caja registradora la cual les permite saber la venta y el gasto total que se han realizado al final del día. Las facturas las llenan manualmente cuentan con hojas de registro para llevar el registro de los productos que ingresan y egresan del inventario.

2.01.2 Visión y alcance

Visión

El propósito del presente trabajo es poder mejorar el proceso de facturación para agilizar la emisión de información requerida y así brindar un mejor servicio.

Alcance

Mejorar el proceso de facturación e inventario, será de mucha utilidad ya que permitirá registrar y guardar la información con mayor rapidez. El alcance de este proyecto es que se pueda generar un reporte de productos existentes reporte de ventas y de todos los gastos realizados en el día.

2.01.3 Entrevistas

Tabla 2

Diseño de entrevistas

Diseño Entrevista		
identificador: 001		
PREGUNTAS	**OBJETIVOS**	**ANALISIS POSTERIOR**
Con que sistema trabaja actualmente en la cafetería?	Definir qué es lo que realmente necesita el usuario.	Actualmente la Empresa cuenta con una caja registradora que de una u otra forma nos ayuda con el ingreso y egreso de dinero las facturas llenamos utilizando el factor Humano un proceso mejorado serviría de mucha ayudando a tener la información mejor ordenada
Cuanto Aportaría ejecutar una aplicación de facturación electrónica en la cafetería?	Saber si Realmente Necesita la Aplicación.	Agilitara los procesos de ventas, ingresos y egresos que se realizan en el día.
Quién tendrá acceso al proceso de facturación?	Determinar la persona que manejaran el sistema.	La persona que maneja la aplicación genera un reporte diario.

2.01.4 Matriz de requerimientos

Tabla 3

Matriz de requerimiento Funcionales

MATRIZ DE REQUERIMIENTOS						
Identificador	**Descripción**	**Fuente**	**Prioridad**	**Tipo**	**Estado**	**Usuarios Involucrados**
REQUERIMIENTOS FUNCIONALES						
RF 001	Generar la factura	empresa	Alta	Sistema	Aprobado	Cliente/Cajero
RF 002	Se necesita generar diversas formas de pago	empresa	Medio	Sistema	Aprobado	Gerente Propietario
RF 003	Registrar detalle del Pedido	empresa	Alta	Sistema	Aprobado	Gerente Propietario /Cajero
RF 004	Registrar productos en el inventario	empresa	Alta	Sistema	Aprobado	Proveedor/Cajero

En estos cuadros determinamos los requerimientos más necesarios para la empresa.

Tabla 4

Matriz de requerimiento No Funcionales

MATRIZ DE REQUERIMIENTOS

Identificador	Descripción	Fuente	Prioridad	Tipo	Estado	Usuarios Involucrados
REQUERIMIENTOS NO FUNCIONALES						
RNF 001	Confirmar el cambio de clave	empresa	medio	Sistema	Aprobado	Gerente propietario /Empleados
RNF 002	Generar reporte de Ingresos y Egresos	empresa	medio	Sistema	Aprobado	Gerente propietario/Contador/Empleados
RNF 003	Los reportes deben generarse pdf	empresa	alto	Sistema	Aprobado	Contador/Empleados

En estos cuadros determinamos los requerimientos más necesarios para la empresa.

2.01.5 Descripción detallada

Tabla 5

Descripción Generar la Factura

Generar la Factura		**Estado**	**Análisis**
Creado por	Marina Tibán	**Actualizado por**	Marina Tibán
Fecha Creación	22-12-2014	**Fecha de Actualización**	07-02-2015
Identificador		RF 001	
Tipo de Requerimiento	Crítico	**Tipo de Requerimiento**	Funcional
Datos de Entrada	• Ingresar datos del cliente por única vez • Guardar datos al momento de generar la factura		
Descripción	Mantener datos y fechas del cliente		
Datos de salida	Genera la Factura		
Resultados Esperados	Tener un mejor control		
Origen	Empresa		
Dirigido a	• Cajero • Administrador		
Prioridad	Alta		
Requerimientos Asociados	RF 004		
Especificación			
Precondiciones	Para que se ejecute el RF el cliente debe estar registrado		
Pos condiciones	Si el usuario olvida la clave de ingreso no podrá realizar el requerimiento.		
Criterios de Aceptación	Mejoría en la búsqueda		

En este cuadro determinamos el proceso de facturación.

Tabla 6

Descripción genera formas de Pago

Se necesita generar diversas formas de pago		Estado	**Análisis**
Creado por	Marina Tibán	**Actualizado por**	Marina Tibán
Fecha Creación	22-12-2014	**Fecha de Actualización**	07-02-2015
Identificador		RF 002	
Tipo de Requerimiento	Crítico	**Tipo de Requerimiento**	Funcional
Datos de Entrada	• Ingresar el monto recibido • Los pagos solo son en efectivo		
Descripción	Mantener bloqueado las formas de pago		
Datos de salida	Información detallada del valor cancelado		
Resultados Esperados	Cumplir con la forma de pago		
Origen	Empresa		
Dirigido a	• Cajero • Administrador		
Prioridad	Alta		
Requerimientos Asociados	RF 001		
Especificación			
Precondiciones	El sistema guarda información sin dificultad		
Pos condiciones	El cajero será el único responsable del manejo del sistema		
Criterios de Aceptación	Permite analizar momentáneamente las ventas generadas al gerente propietario y al contador		

En este cuadro determinamos las formas de pago

Tabla 7

Descripción Registro de Pedido

Registrar detalle del Pedido		Estado	Análisis
Creado por	**Marina Tibán**	**Actualizado por**	**Marina Tibán**
Fecha Creación	**22-12-2014**	**Fecha de Actualización**	**07-02-2015**
Identificador		**RF 003**	
Tipo de Requerimiento	**Crítico**	**Tipo de Requerimiento**	**Funcional**
Datos de Entrada	• **Ingresar detalladamente los productos consumidos**		
Descripción	**Cada cajero mantiene la responsabilidad al momento de registrar el detalle de los productos**		
Datos de salida	**Información detallada con éxito**		
Resultados Esperados	**El sistema guarda con éxito el registro.**		
Origen	**Empresa**		
Dirigido a	• **Cajero** • **Administrador**		
Prioridad	**Alta**		
Requerimientos Asociados	**RF 002**		
Especificación			
Precondiciones	**Los datos no son únicos en el registro detallado**		
Pos condiciones	**Si el cajero no detalla los productos tendrá una alteración en el reporte de productos**		
Criterios de Aceptación	**Obtener Flexibilidad en los informes**		

Tabla 8

Descripción Registro de Productos en el Inventario

Registrar productos en el inventario		Estado	**Análisis**
Creado por	Marina Tibán	**Actualizado por**	Marina Tibán
Fecha Creación	22-12-2014	**Fecha de Actualización**	22-12-2014
Identificador		RF 004	
Tipo de Requerimiento	Crítico	**Tipo de Requerimiento**	Funcional
Datos de Entrada	Registrar los productos de ingresan en el dia		
Descripción	El cajero mantiene la responsabilidad recibir y registrar los productos en el inventario		
Datos de salida	Productos ingresados correctamente.		
Resultados Esperados	Mayor eficiencia al momento de realizar una búsqueda especifica.		
Origen	Empresa		
Dirigido a	• Cajero • Administrador		
Prioridad	Alta		
Requerimientos Asociados	RF 001		
Especificación			
Precondiciones	Los códigos son únicos para cada producto		
Pos condiciones	Si el cajero confunde los códigos de los productos tendrá una alteración en el reporte de productos		
Criterios de Aceptación	Una vez ingresado los productos el usuario podrá visualizar el listado de los productos para que pueda seleccionar		

En estos cuadros hemos determinado de manera individual el análisis de los requerimientos funcionales necesarios para la empresa.

2.02. Mapeo de Involucrados

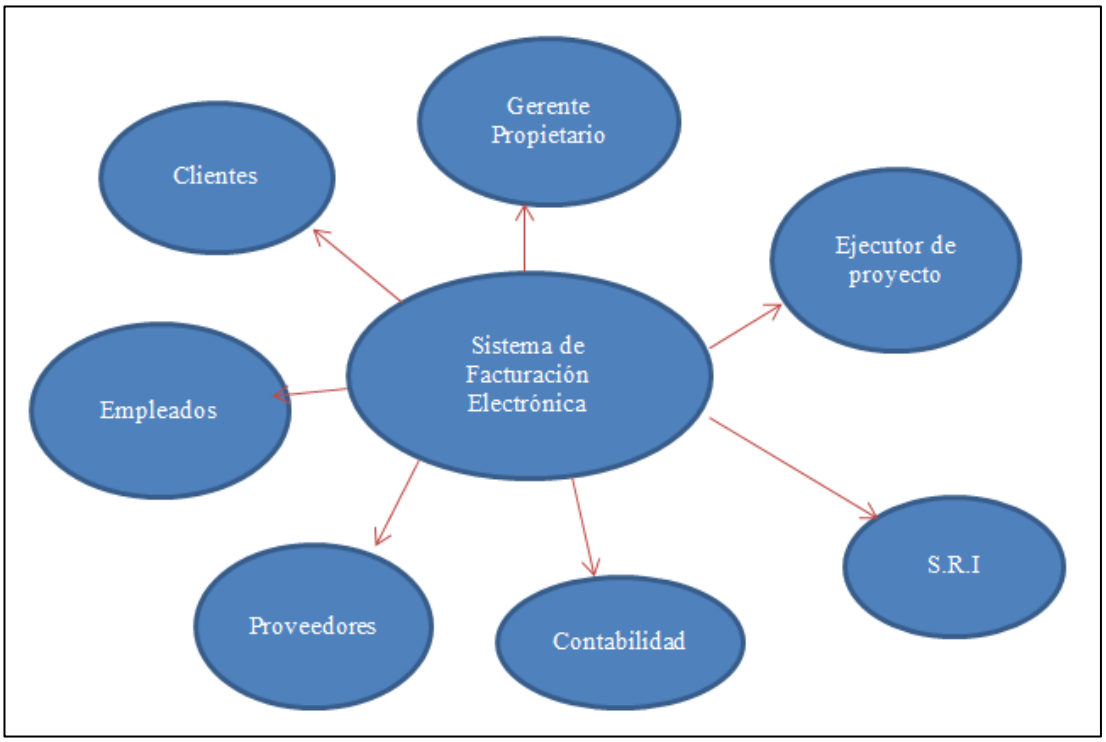


Figura 1 Involucrados para el manejo del sistema

En el siguiente mapa de Involucrados se detalla de forma gráfica las personas e instituciones que se ven afectados en el proceso o se mueven en el entorno del mismo y que pueden participar en la solución mediante la intervención efectiva del proyecto.

2.03. Matriz de Involucrados (Ver Anexo A.02)

3.01. Árbol de Problemas

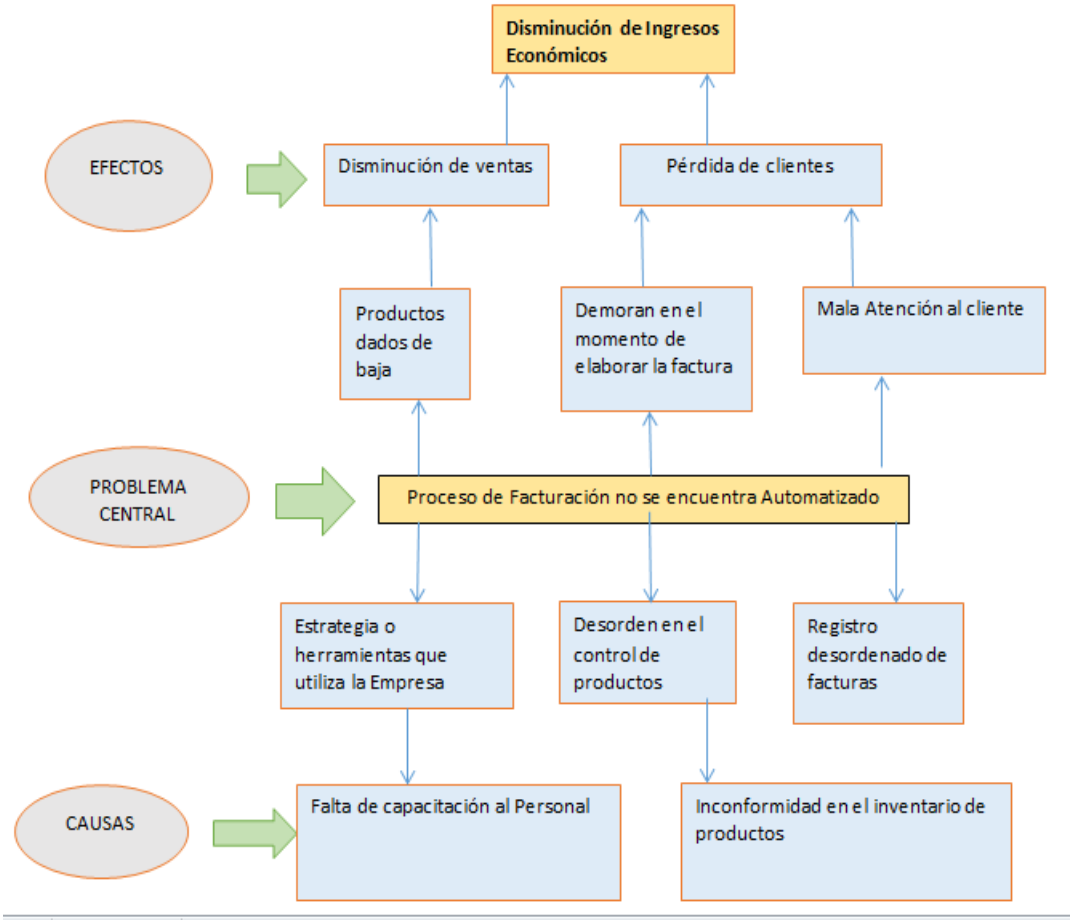


Figura 2 Árbol de problemas

En el árbol de problema se detalla de forma clara las diferencias entre efectos y causas que tiene la Empresa.

3.01.01 Análisis del árbol de problemas

La Empresa ha venido trabajando de manera manual en el proceso de facturación. Hoy en día las nuevas tendencias tecnológicas van aumentando lo que obliga a los ecuatorianos adaptarse a dichas tendencias.

3.02. Árbol de Objetivos

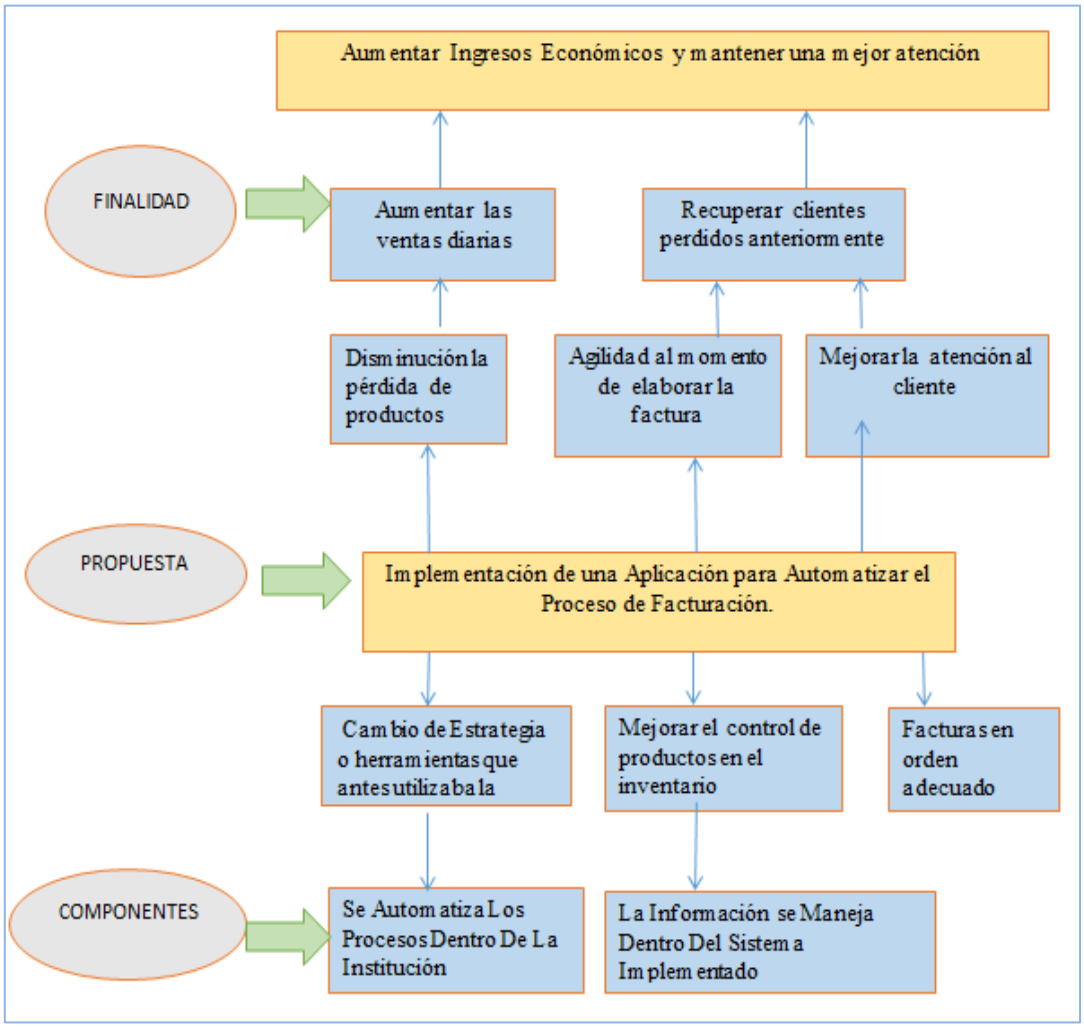


Figura 3 Arbol de objetivos

3.02.01 Análisis del árbol de objetivos

En este árbol de objetivos se detecta el propósito del proyecto, los componentes que en la actualidad se dan ayudará a la Empresa con la herramientas tecnológicas optimizando los procesos responderán de una manera eficiente al personal capacitado, el cual beneficiará a la Empresa ayudando en la organización y coordinación para que el proceso de facturación sea un éxito.

3.03. Diagramas de casos de uso

Figura 4 Diagrama Caso de uso general del sistema (Ver Anexo A.03)

3.04. Casos de uso de realización

3.04.1 Cancela Servicio

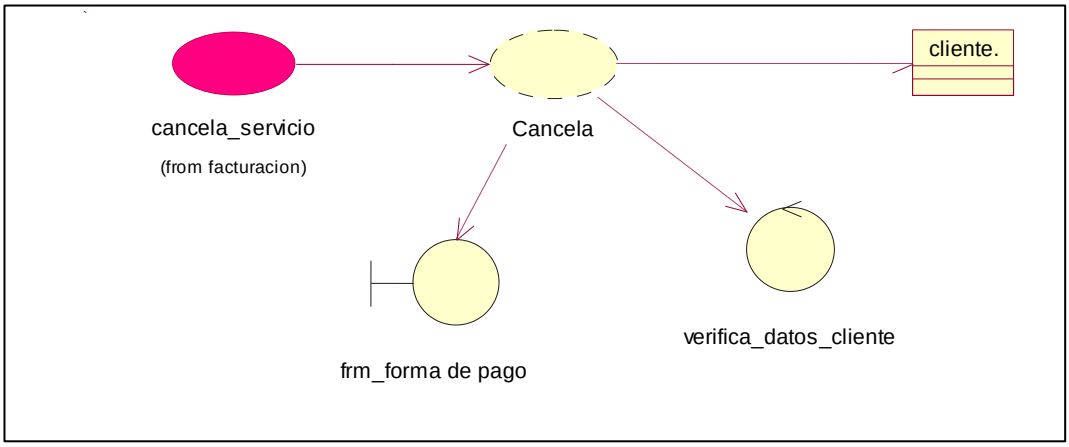


Figura 5 Diagrama Cancela Servicio

Esta Figura nos muestra la forma de pago que el cliente va a relizar pero solo para registro

3.04.2 Registra Pedido

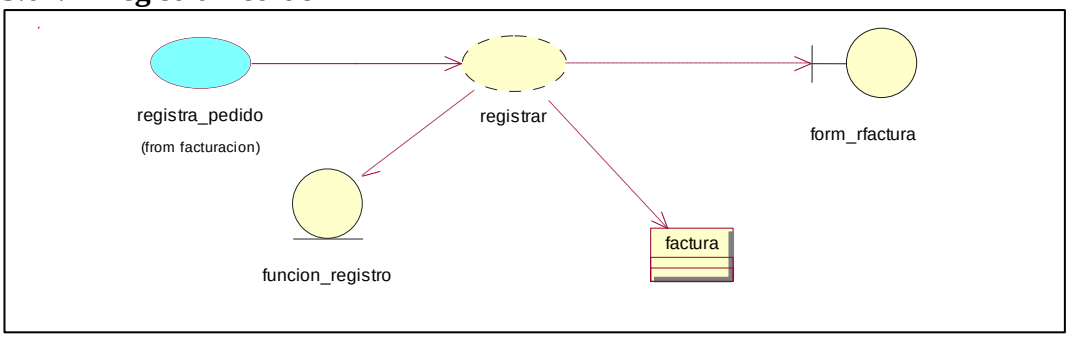


Figura 6 Diagrama Registra Pedido

En esta figura nos muestra el registro del pedido

3.04.3 Ingreso de productos

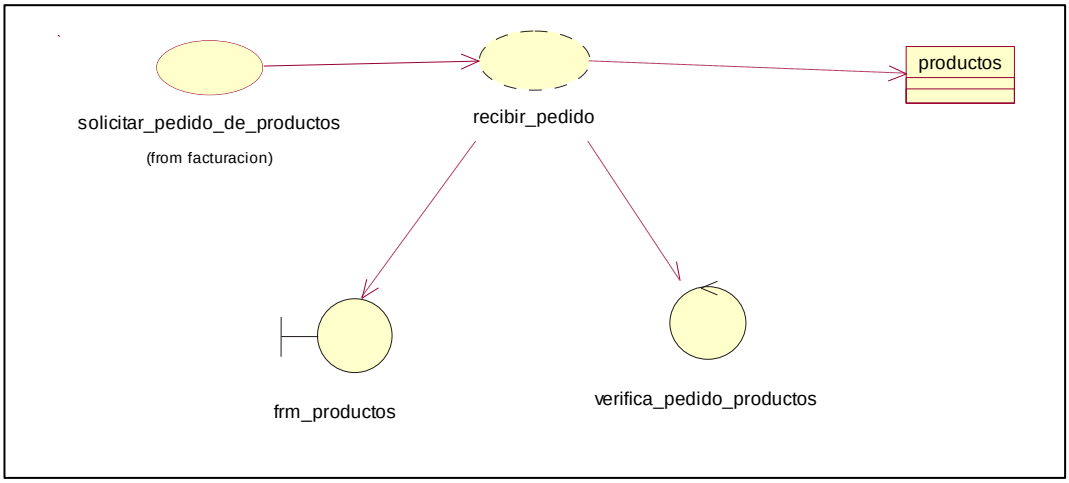


Figura 7 Ingreso de productos

Esta figura nos muestra el proceso que se realiza al momento de ingresar un producto

3.05. Diagrama de secuencias del sistema

3.05.1 Cancela servicio

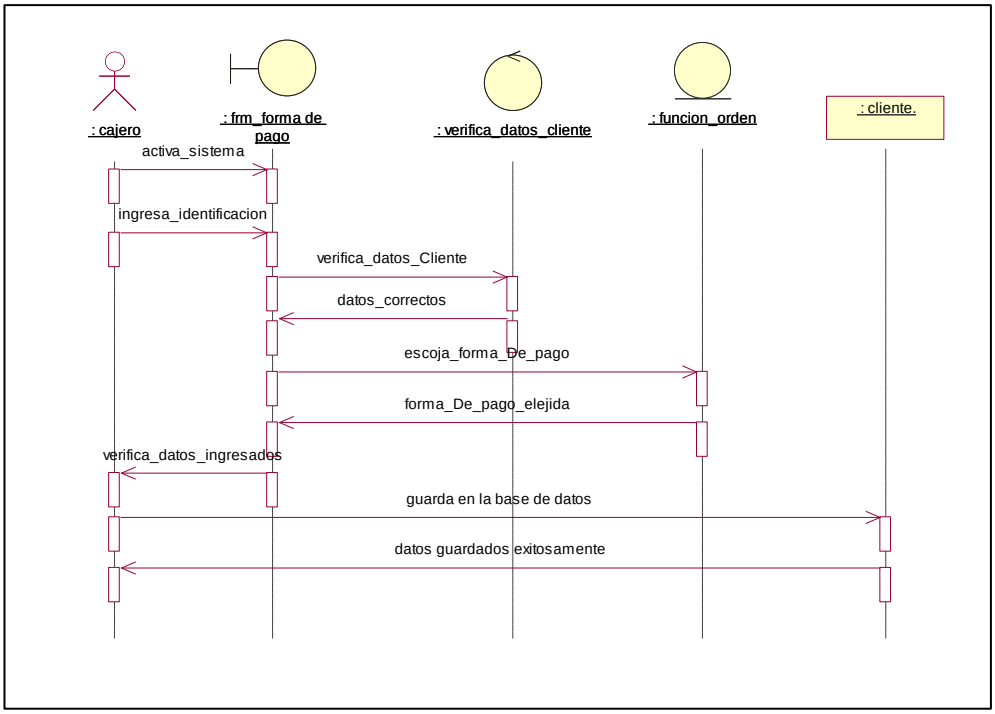


Figura 8 Diagrama Cancela servicio

3.05.2 Registra Pedido

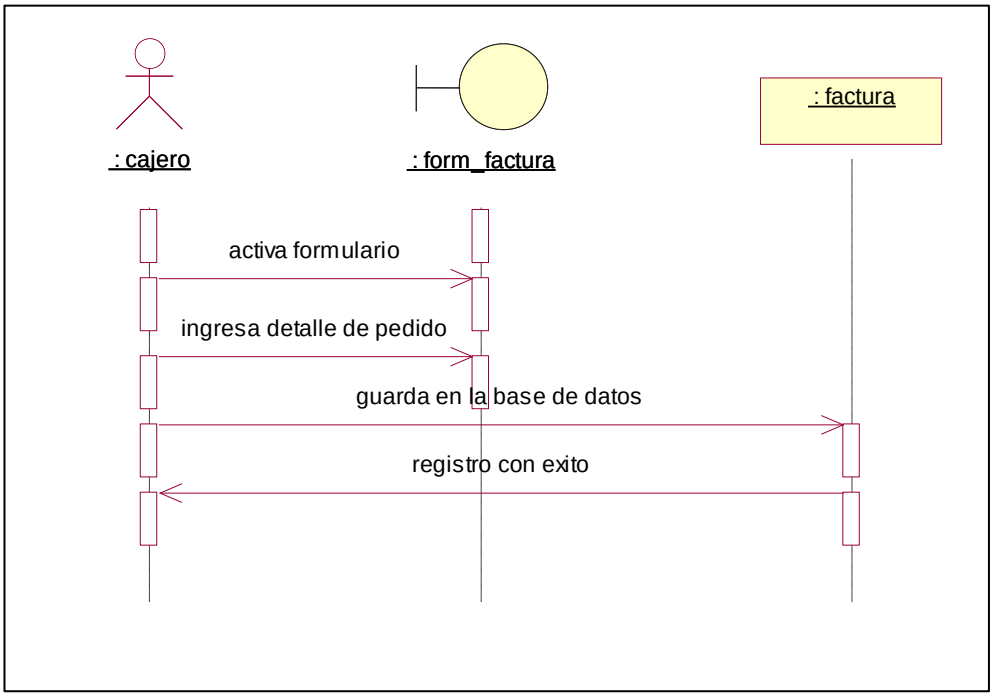


Figura 9 Diagrama Registra Pedido

3.05.3 Ingreso de Productos

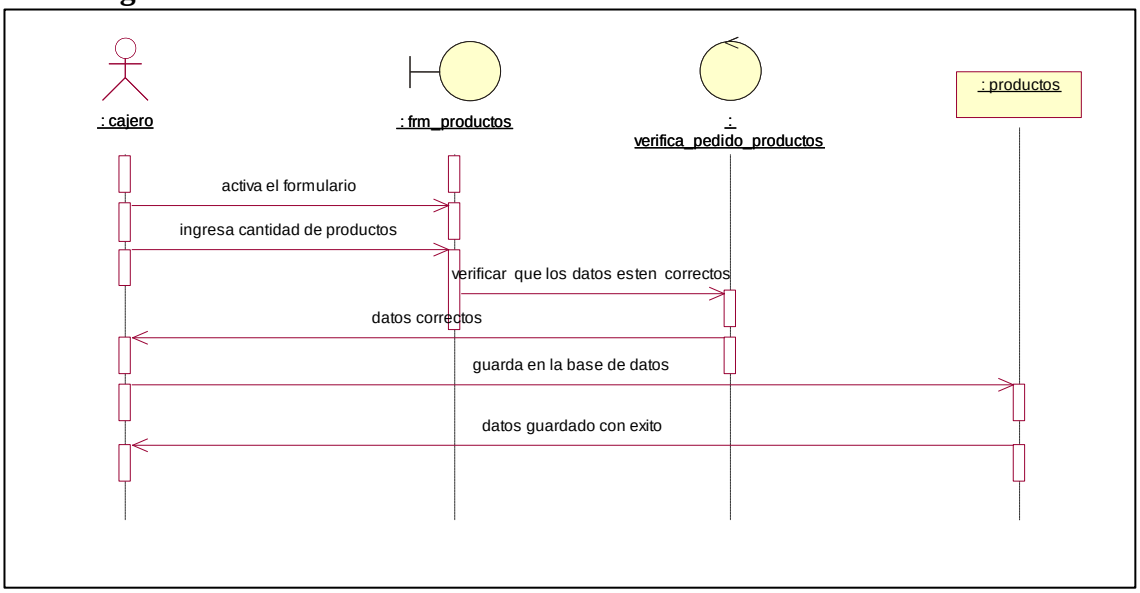


Figura 10 Diagrama Ingreso de Productos

3.06. Especificación de casos de uso

Tabla 8

Descripción Cancela Servicio

Caso de Uso:	**Cancela servicio**
Identificador	UC001
CURSO TIPICO DE EVENTOS	
USUARIO	SISTEMA
1. El Caso de uso se activa cuando el cajero digita cancelar	1. Sistema despliega la forma de pago efectivo
2. El cajero debe verificar la información del cliente para generar la factura	2.- Desplegará la interfaz para registrar la información o datos del cliente para la emisión e la factura
3. El cajero selecciona la forma de pago efectivo luego presionar botón crear factura.	3.- Registra la forma de pago efectivo por el cajero

Tabla 9

Descripción Registro de pedido

Caso de Uso:	**Registra Pedido**
Identificador	UC002
CURSO TIPICO DE EVENTOS	
USUARIO	SISTEMA
1. El caso de uso se activa cuando el usuario digita factura.	1. Sistema despliega el formulario de la factura
2. El cajero debe verificar ingresar la orden del pedido para generar la factura	2.- Se mantiene en el formulario de la factura para ingresar la orden del pedido
3. El cajero llena el botón que dice orden de pedido	3.- Guarda los datos y verifica si los datos son correctos caso contrario le aparecerá una pantalla indicándole que le falta llenar el campo.

Tabla 10

Descripción Ingreso de productos

Caso de Uso:	**Ingreso de Productos**
Identificador	UC003
CURSO TIPICO DE EVENTOS	
USUARIO	SISTEMA
1. El caso de uso se activa cuando el usuario digita ingreso de productos	1. Sistema despliega el formulario de productos
2. El cajero debe ingresar detalladamente los productos	2.- Se mantiene en el formulario de productos para ingresar los datos.
3. El cajero digita el botón que dice Guardar	3.-El Sistema nos muestra una ventana que dice datos guardados con éxito caso contrario nos mostrara una ventana con los errores a corregir.

Capítulo IV: Análisis y Alternativas

4.01. Matriz de Análisis de Alternativas

ANÁLISIS Y MATRIZ DE ALTERNATIVAS

Escala de Valores

1= Bajo
2=Medio Bajo
3=Medio
4=Medio Alto
5=Alto

VARIABLES / ESTRATEGIAS	NECESIDAD DE RECURSOS			FACTIBILIDAD POLÍTICA	DURACIÓN DEL PROYECTO	PUNTAJE	PRIORIDAD
	HUMANOS	TÉCNICOS	FINANCIER.				
Analizar el mejoramiento del proceso de facturación en la empresa una vez aplicado el sistema	4	2	1	3	3	13	2da.
Tiempo aprovechado por el cajero	4	4	2	2	3	15	2da.
Aumentar los Ingresos económicos.	5	4	5	2	3	19	1ra.

Figura 11 Matriz de Análisis de Alternativas

4.02. Matriz de Impactos de Objetivos

Tabla 11

Matriz de Impactos de Objetivos

OBJETIVOS	FACTIBILIDAD A LOGRARSE	IMPACTO DE GENERO	IMPACTO AMBIENTAL	RELEVANCIA	SOSTENIBILIDAD
Mantener los beneficios que se adquieren, ya que son de mucha importancia	Se cuenta con el apoyo de la Empresa	Fortalecimiento en el grupo de trabajo incrementando orden.	Contribuye con la disminución de recursos físicos	Responde a las expectativas de los beneficiarios	Fortalece la Organización en la Empresa
Aceptable y apto para todos los beneficiarios			Ninguno	Ninguno	Gran acogida por todos los beneficiarios

4.03. Estándares para el Diseño de Clases

Tabla 12

Estándares de Diseño de clases **(Ver Anexo A.04)**

4.04. Diagrama de clases

Figura 12 Diagrama de clases

(Ver Anexo A.05)

4.05. Modelo Lógico - Físico

Figura 13 Modelo Logico-Fisico **(Ver Anexo A.06)**

4.06. Diagrama de Componentes

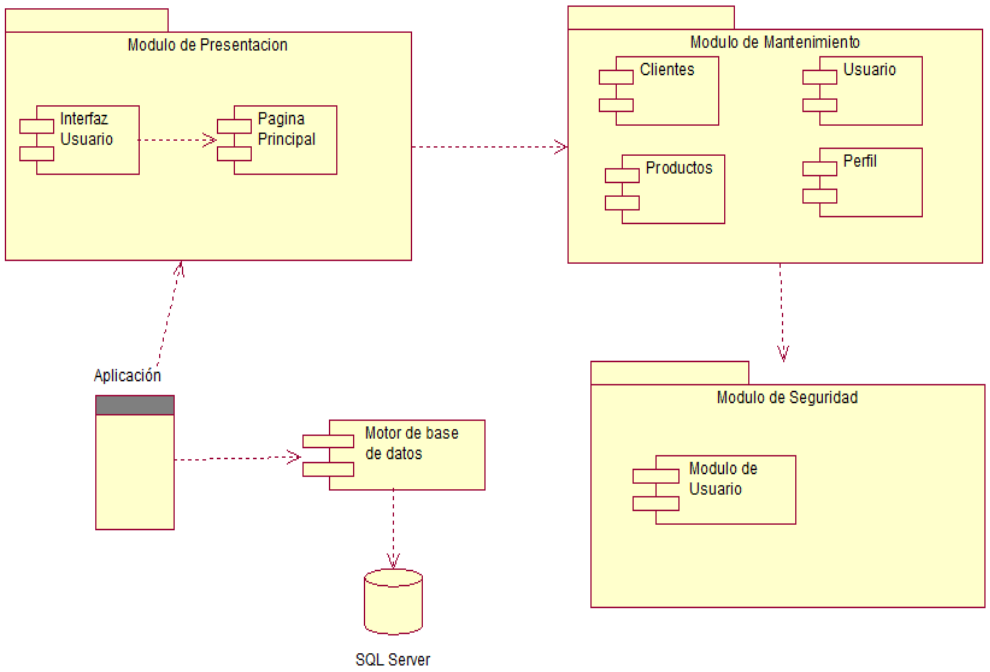


Figura 14 Diagrama de Componentes

En esta figura mostramos el diseño del software como esta conformada la estructura general

4.07. Diagramas de Estrategias

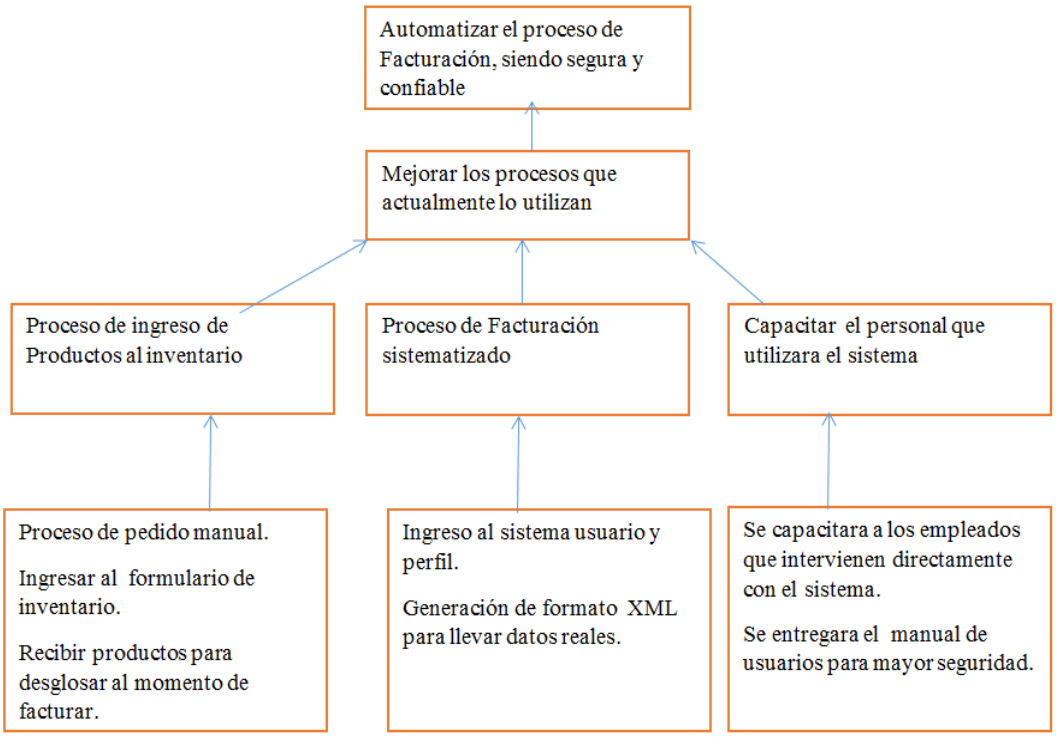


Figura 15 Diagrama de Estrategias

En esta figura mostramos el diagrama de Estrategias ya que es una alternativa para visualizar y razonar el proyecto digital (Según Juan Puebla).

4.08. Matriz de Marco Lógico.

Tabla 13

Matriz de Marco Lógico

Matriz del Marco Lógico

Finalidad **Mayor Productividad y mejor atención al cliente.**	**Indicadores** Se Requiere estudios económicos técnicos y financieros con el fin de garantizar su operatividad	**Medios de Verificación** -Registros administrativos -Informes semanales	**Supuestos** -No existe los recursos necesarios para la realización de soportes
Propósito **Disminuir la pérdida de producto y de dinero**	-Controlar adecuadamente los procesos y registros	Registros de Actividades realizadas	Utilización inadecuada de los materiales y suministros entregados a los técnicos.
Componentes **-Registrar y controlar el proceso** **-Clientes satisfechos por la atención brindada.**	Aumento de costos en los suministros y equipos entregados al área de soporte	Informes semanales y diarios sobre los inconvenientes generados	Clientes insatisfecho por el servicio
Actividades **-Implementación de nueva tecnología** **-Capacitar al personal para el nuevo uso de la aplicación.** **-Registrar el stock de productos**	-Eficiencia en el despacho de productos	Verificación por medios de reportes kardex e informes de ventas diarias	Establecer adecuadas técnicas para la realización de los soportes del sistema

4.09. Vistas arquitectónicas

4.09.01. Vista lógica

Apoya principalmente los requisitos funcionales ,lo que el sistema debe brindar en términos de servicio a sus usuarios

El sistema se descompone en una serie de abtracciones primarias ,tomadas principalmente del dominio del problema en forma de objetos clases de objetos, Aquí se Aplican los principios de abstracción,encapsulación y herencia.

Esta descomposición no solo se hace para potenciar el análisis funcional,sino también sirve para identificar mecanismos y elementos de diseño comunes a diversas partes del sistema.

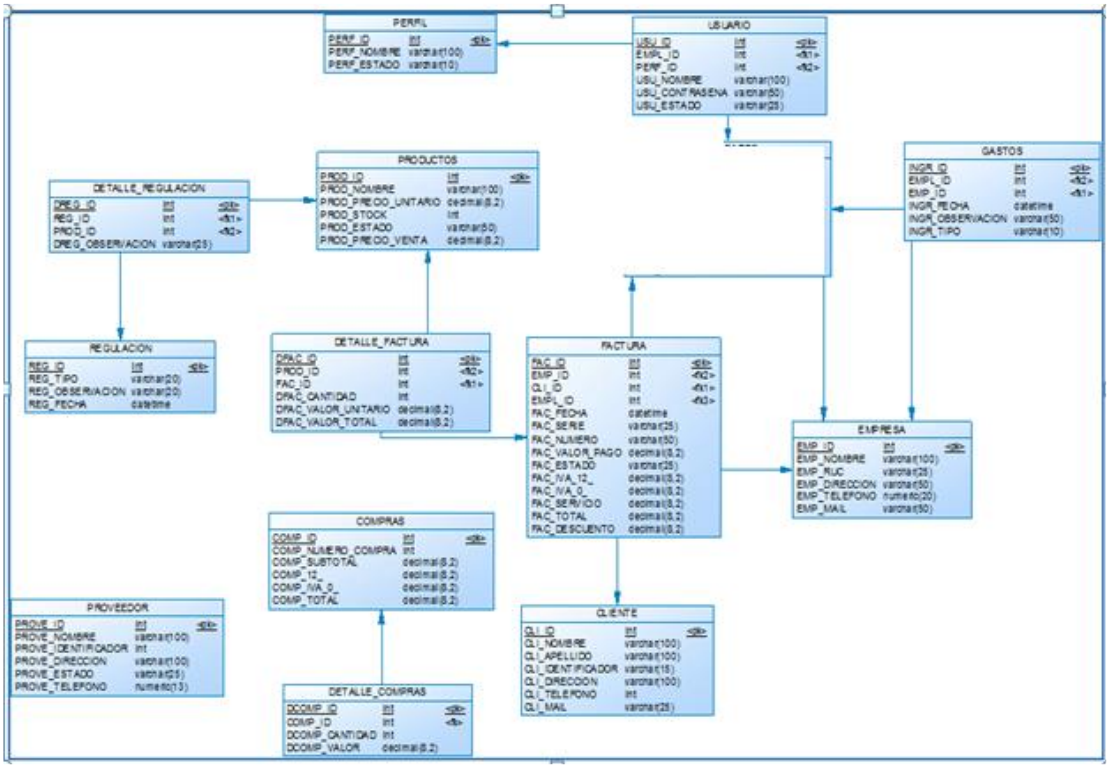


Figura 16 Vista Lógica

4.09.02. Vista física

Se toma en cuenta los requisitos no funcionales del sistema tales como, disponibilidad, confiabilidad, desempeño entre otros más. El sistema se ejecuta sobre varios nodos de procesamientos (hardware).Estos nodos son relacionados con elementos identificados de las vistas anteriores.

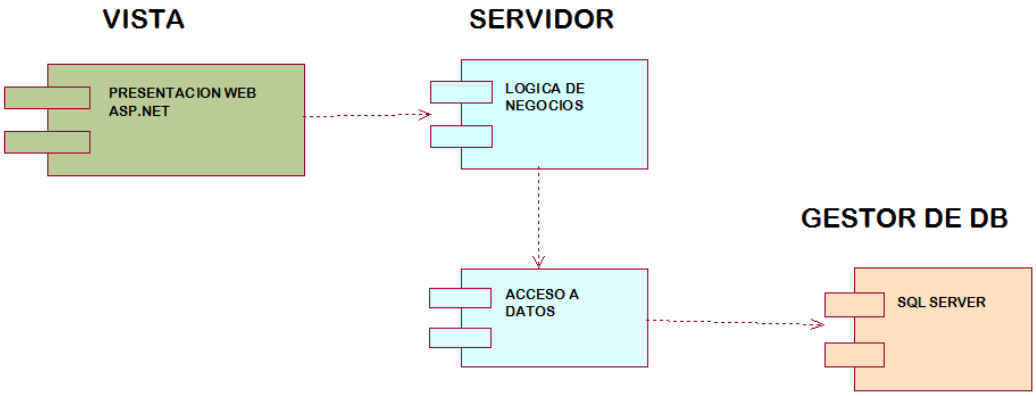


Figura 17 Vista Física

En esta figura mostramos la interfaz del software

4.09.03. Vista de desarrollo

Se encuentra en la organización real de los módulos de software en el ambiente de desarrollo. El software se empaqueta en partes pequeñas que pueden ser bibliotecas o subsistemas que son desarrollados por uno o un grupo de programadores.

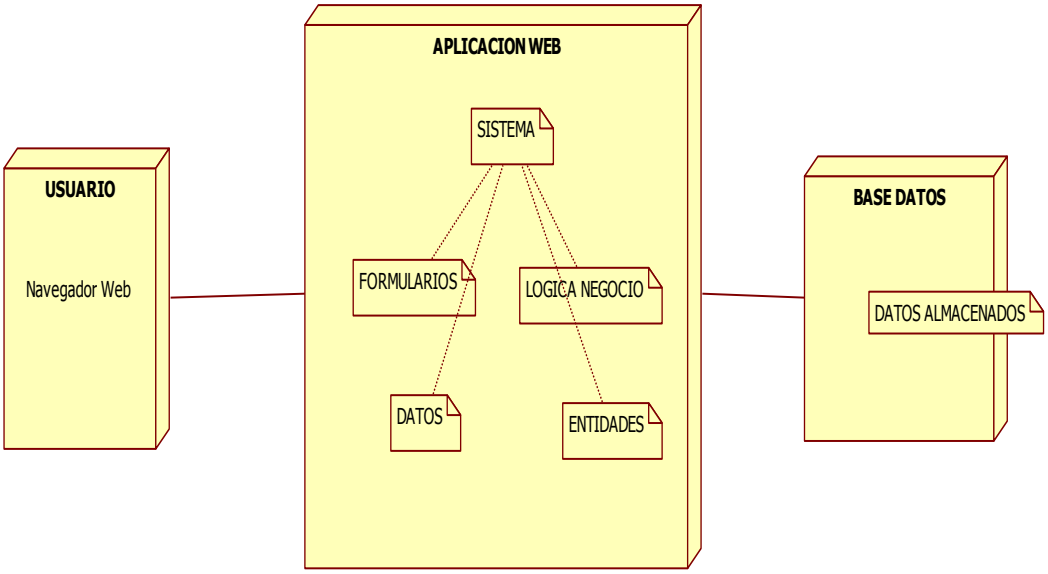


Figura 18 Vista de desarrollo

4.09.04. Vista de procesos

Se tratan los aspectos de concurrencia y distribución, Integridad del sistema y tolerancia a fallos. Se especifica en cual hilo de control se ejecuta efectivamente una operación de una clase identificada en la vista lógica

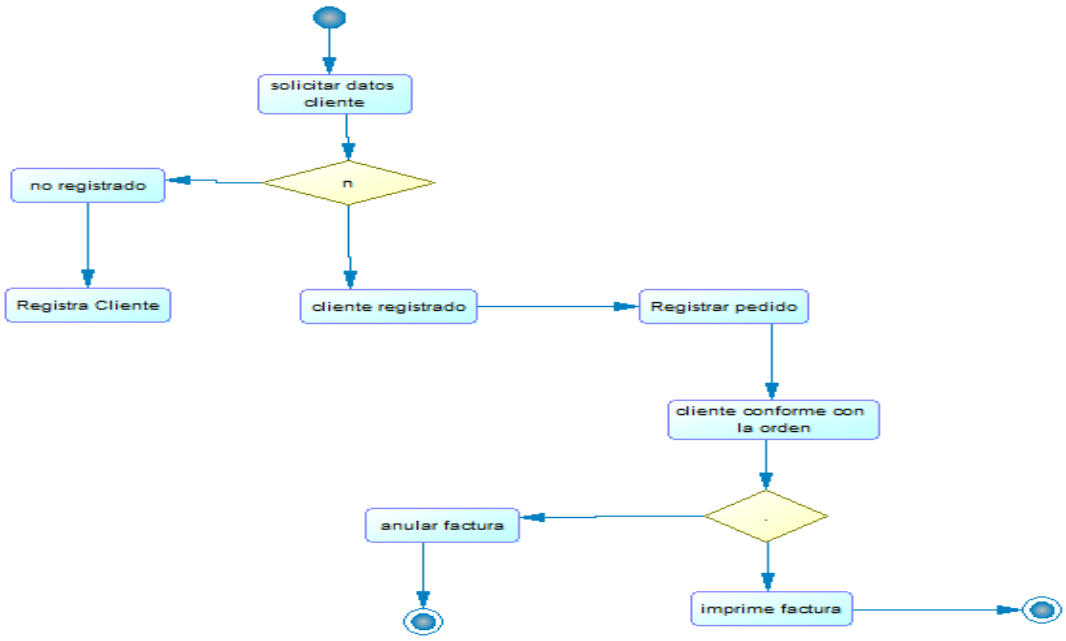


Figura 19 Vista de Proceso

Capítulo V: Propuesta

5.01. Especificación de estándares de programación

Según (Real) En su publicación de estándares de programación Redacta.Durante el desarrollo del aplicativo se hará uso de varios objetos según sea la necesidad en el proceso.

Estándar 1: Declaración de Variables, se ajusten al motivo para la que se requieran .El nemotécnico definido se establece tomando en consideración principalmente lo siguiente:

Tabla 14

Detalle de Objetos

Tipos de Control	Prefijo	Especificación y Nomenclatura
Label	Lbl	lblTitulo
TextBox	Txt	txtNombre
Button	Btn	btnAceptar
RadioButton	Rdo	droImagen
CheckBox	Chk	chkEstado
Select	Slc	slcNombre
PassWordBox	Psw	pswClave

Tabla 15

Tipo de Datos utilizados

Tipos de Variable	Abreviatura	Descripción
Char	Ch	Carácter de 16 bits
String	St	Cadena de caracteres
Int	In	Carácter entero de 32 bits
DateTime	Dt	Carácter de Fecha y hora
Boolean	Bl	Valor lógico de verdadero o falso
Float	Fl	Comas flotantes de 11-12 dígitos
Double	Dl	Comas flotantes de 64bits (15-16 dígitos)
Byte	Bt	Entero de 8 bits sin signo
Array	Ar	Tipo de datos compuesto que puede contener múltiples tipos de datos

Tabla 16

Estándares en Base de datos

En la modelación y diseño de la base de datos se utilizó los siguientes tipos de datos:

Tipos de Datos	Descripción
INT	Utilizado como identificador de Códigos PrimaryKey de cada tabla y sus relaciones.
VARCHAR	Utilizado en todos los campos que contienen texto y campos numéricos especiales (ejemplo: 00020)
CHAR	Utilizado en campos de tipo texto que contienen un solo carácter para identificar.
DECIMAL	Utilizado en campos numéricos que contienen decimales.
DATE	Utilizado en campos que solo contienen valores de fechas
DATETIME	Utilizado en campos que contienen valores de fecha y hora.

Especificación de nombres de Tablas

Se nombra a las tablas de la Base de datos de la siguiente manera:

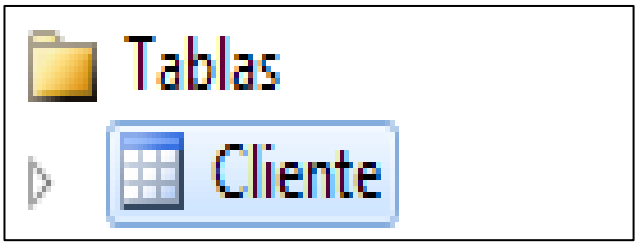


Figura 20 Especificación de nombre de Tablas

Se usa la primera letra con Mayúscula, seguida del nombre completo de la tabla.

Ejemplo: **Cliente**

Especificación de campos de Tablas

En la designación de campos para las distintas tablas se usa el siguiente tipo de descripción

id_cliente	int	☐
cedula	int	☑
nombre	varchar(50)	☑
apellidos	varchar(50)	☑
direccion	text	☑
telefono	int	☑
mail	varchar(50)	☑
		☐

Figura 21 Especificación de campos de Tablas

El campo de identificación de clave primaria siempre se usa la palabra **"id"** seguido del nombre de la tabla. Para todos los que se usa un detalle de campo en minúsculas seguido del nombre específico. Ejemplo: id_cliente.

5.02. Diseño de Interfaces de Usuario

Diseño de Login

Figura 22 Diseño de Login

Tabla 17

Descripción de Login

ITEMS	DESCRIPCION
A	Iniciar Para ingresar como usuario o administrador
B	Menú Principal
C	Nombre de la Empresa
D	Casillero para ingresar nombre de Usuario/administrador del sistema
E	Casillero para ingresar la contraseña
F	Casillero para elegir si queremos recordar la contraseña en futuras ingresos al sistema
G	Ingreso al sistema

5.02.01 Diseño de Página Principal

Diseño Página Principal

Figura 23 Diseño Página Principal

Tabla 18

Descripción Diseño Página Principal

ITEMS	DESCRIPCION
A	Nombre de usuario Registrado
B	Imagen de la empresa

5.03. Especificación de pruebas de unidad

5.03.01 *Especificación de pruebas de unidad. Registro del Cliente*

Tabla 19

Identificador Crear Cliente

Identificador de la Prueba	**PU001**
Método a Probar	Crear
Objetivo de la Prueba	Crear Cliente
Datos de Entrada	
Nombre Apellido Cedula Teléfono E-mail Dirección	
Resultado Esperado	
El Cliente se crea sin dificultad	
Comentarios	
Al crear el cliente el tiempo de es minino	

Figura 24 Crear Cliente

5.03.02 Especificación de pruebas de unidad. Modificar Cliente

Tabla 20

Prueba Modificar Cliente

Identificador de la Prueba	**PU002**
Método a Probar	Modificar
Objetivo de la Prueba	Realizar Modificaciones a Clientes que ya están registrados
Datos de Entrada	
Datos del Cliente	
Resultado Esperado	
Los Datos Modificados de guardan con éxito	
Comentarios	
El tiempo de la modificación es mínima	

Figura 25 *Modificar Cliente*

Esta figura nos muestra la interfaz para modificar el cliente

5.03.03 Especificación de pruebas de unidad. Eliminar Cliente

Tabla 21

Prueba Eliminar cliente

Identificador de la Prueba	**PU003**
Método a Probar	Eliminar Cliente
Objetivo de la Prueba	Verificar el funcionamiento en el proceso eliminar.
Datos de Entrada	
Escoger la opción eliminar	
Resultado Esperado	
Está seguro de querer eliminar el cliente #: MAYRA FUENTES?	
Comentarios	
El tiempo de espera es mínimo	

Figura 26 Eliminar Cliente

Esta figura nos muestra la interfaz para eliminar el cliente no deseado

5.04. Especificación de pruebas de aceptación

5.04.01 Especificación de pruebas de aceptación. Iniciar Sesión

Tabla 22

Prueba Iniciar Sesión

Identificador de la Prueba	**PA001**
Caso de uso	Iniciar sesión
Tipo de usuario	Administrador, cajero
Objetivo de la prueba	Registrar Usuarios
Secuencia de Eventos	
Iniciar sesión Registrar usuario Aplicar roles según el usuario creado	
Resultados Esperados	
Visualizar los datos ingresados	
Estado	
Activo	

5.04.02 Especificación de pruebas de aceptación Facturación

Tabla 23

Prueba Facturación

Identificador de la Prueba	**PA002**
Caso de uso	Facturación
Tipo de usuario	Cajero
Objetivo de la prueba	Verificar el proceso de facturación
Secuencia de Eventos	
Iniciar sesión Registrar cliente Registrar la Orden Genera e imprime factura Generar XML	
Resultados Esperados	
Información correcta del cliente Almacenar factura Impresión o generar XML de la factura	
Estado	
Aceptado	

5.04.03 *Especificación de pruebas Ingreso de Producto*

Tabla 24

Prueba de ingreso de Producto

Identificador de la Prueba	**PA003**
Caso de uso	Ingreso de Producto
Tipo de usuario	Cajero
Objetivo de la prueba	Registrar la orden de la compra
Secuencia de Eventos	
Iniciar sesión Ingreso al módulo de productos Registrar el ingreso de la materia prima	
Resultados Esperados	
Almacenar productos sin problema	
Estado	
Aceptado	

5.05. Especificación de pruebas de carga

5.05.01 ***Especificación de pruebas de carga. Compras***

Tabla 25

Prueba Ingreso de Compras

Identificador de la Prueba	**PC001**
Tipo de prueba	Ingreso de compra
Objetivo de la prueba	
Verificar los errores	
Descripción	
Ingresar las compras y revisar si genera algún error	
Resultados Esperados	

5.05.02 ***Especificación de pruebas de carga. Stock de productos***

Tabla 26

Prueba Ingreso de Productos

Identificador de la Prueba	**PC002**
Tipo de prueba	Ingreso de productos
Objetivo de la prueba	
Ingresar varios productos para probar el correcto funcionamiento del proceso	
Descripción	
Ingresar el producto y revisar si el stock aumenta	
Resultados Esperados	
El módulo de productos funciona correctamente	

5.06. Configuración del Ambiente mínima/ideal

El ambiente mínimo ideal para que la aplicación se desempeñe de una manera correcta, debe prevalecer con las características que se detallan a continuación:

Servidor de aplicación y de información.

-
- Procesador de 2,4 GHz
- Sistema Operativo Windows 7 de 32 o 64 bits
- SQL server 2008
- Cristal Report de 32x o 64x

Capítulo VI: Aspectos Administrativos

6.01. Recursos

6.01.01 Recurso Humano

Es la persona encargada de llevar a cabo una gestión con un perfil de liderazgo, en la organización, que trabajen y den el máximo de sí mismas con una actitud positiva y favorable cuya finalidad es proporcionar a la organización una fuerza laboral eficiente.

Se refiere a las personas o colaboradores en la planeación, organización, desarrollo, y coordinación de un objetivo.

a. Responsable del Proyecto

b. Supervisor de sistemas de la empresa

Tabla 27

Descripción de Recurso Humano

RECURSO HUMANO			
Humano	**Nombre**	**Actividad**	**Responsabilidad**
Tutor	Ing. Juan Minango	Guía el desarrollo del Proyecto	Revisar el progreso del desarrollo del sistema y la documentación.
Lector	Ing.Jhonny Coronel	Revisa el desarrollo del Proyecto	Asegurar que el desarrollo del proyecto elaborado cumple con las normas establecidas.

6.01.02 Recursos Físicos:

Los recursos físicos tradicionalmente, comprenden varios ítems como terrenos, edificios, maquinaria, equipos, infraestructura, bibliografía, documentación, medios de transporte, etc. Sin embargo, este tipo de recursos no siempre deben ser adquiridos, pero sí puede ser cubiertos o suplidos con lo que se tiene. Cuando hay convencimiento y suficiente motivación para emprender una misión, es importante fomentar la movilización de recursos en donde todos ponen lo que puedan.

6.01.03 Recursos Técnicos:

En caso de que el proyecto contemple este tipo de componente, es necesario establecer las alternativas técnicas elegidas y las tecnologías a utilizar. Cuando un proyecto contempla la adopción de innovaciones tecnológicas, es bueno tener presente, que muy probablemente, la adopción de la innovación no se va a producir en un su totalidad. El proceso de transferencia de tecnología es de doble vía, es decir, la propuesta generalmente presentada por el grupo de agentes de desarrollo, al encontrarse con la tecnología tradicionalmente implementada en las comunidades, entra en un procesos de diálogo en donde ambas se transforman para evolucionar a una tercera propuesta producto de su conjunción.

6.02. Presupuesto

Tabla 28

Descripción de Presupuesto

Recurso	Cantidad	Valor Unitario	Valor Total
Suministros de oficina	--------	----------	10,00
Servicios Básicos	6meses	3,00	18,00
Alimentación	60 días	2,75	165,00
Transporte	60 días	1,25	75,00
Computador	1	600,00	600,00
Internet	6meses	22,00	132,00
Impresiones	300	0,5	15,00
Empastado	1	---------	30,00
Anillado	1	10,00	10,00
Capacitación	10 días	35,00	350,00
	TOTAL		1.405,00

6.03. Cronograma

Se proyecta el tiempo de las actividades generadas en la ejecución del sistema, de la base para la evaluación y control del avance del proyecto. Se realiza su diagrama en Microsoft Project 2010.

Figura 27 Cronograma de Proyecto **Ver (Anexo A.06)**

Capítulo VII: Conclusiones y Recomendaciones

7.01. Conclusiones

- La Cafetería Delicias de Verde cuenta con una caja registradora el cual les permite controlar las ventas y los gastos realizados en el día.
- El aplicativo cuenta con una interfaz amigable para el usuario lo cual facilita el proceso de facturación
- Como lenguaje de programación se utilizó asp.net herramienta que me ha permitido desarrollar de manera ordenada dividiendo la aplicación en tres capas como la presentación, datos y la lógica de negocio donde se recopila todas las funciones del sistema.
- El aspecto más importante que se va a lograr con el proceso de facturación es disminuir el tiempo del servicio hacia los clientes, además la información registrada será más organizada.

7.02. Recomendaciones

- Desarrollar las aplicaciones en base a los requerimientos del usuario para así obtener un producto óptimo, de calidad y seguro que permita resolver los pequeños problemas que conlleva la información.
- Tener una buena relación con los usuarios directos de la empresa para que nos den a conocer sus inquietudes y necesidades.
- La seguridad es uno de los aspectos más importantes en una aplicación por lo que se recomienda tener un administrador de usuario quien asignara

correctamente los perfiles de cada uno y en un futuro evitar inconvenientes con el proceso de la aplicación.

- Las personas que utilicen la aplicación deben tener una adecuada y amplia capacitación, para que así puedan conocer y comprender las diferentes funcionalidades del sistema de esta manera se evitara que den ingresos erróneos al sistema.

Bibliografía

Real, A. (s.f.) estándares de Programacion.Costa Rica.

Diseño de Bases de Datos :http://escritura.proyectolatin.org/topicos-avanzados-de-bases-de-datos/diseno-y-modelado-de-bases-de-datos/

Server, Windows. Windows Server . s.f. 05 de 10 de 2014 <http://www.ajpdsoft.com/modules.php?name=News&file=article&sid=304>.

Diagramas de estrategias :https://www.google.com.ec/#q=que+es+un+diagrama+de+componentes+en+uml

Juan Puebla Pons

Víctor Manuel Martínez López

http://upcommons.upc.edu/e-prints/bitstream/2117/13663/1/El%20diagrama%20como%20estrategia%20del%20proyecto%20arquitect%C3%B3nico%20contempor%C3%A1neo-Puebla-Mart%C3%ADnez.pdf

".(Servicio de Rentas Internas,2012,pág.15,Comprovante de venta y retención, Extraído el 22 de marzo del 2013,http://cef.sri.gob.ec/virtualcef/file.php/1/MaterialCursosVirtuales/CVR Sistema de Facturacion.pdf)

ANEXOS

Anexo A.01

Tabla 1

Matriz de análisis de Fuerzas T

ANÁLISIS DE FUERZAS					
Situación Empeorada	**Situación Actual**				**Situación Mejorada**
Perdidas Económicas para la empresa Cafetería Delicias de Verde	La Empresa Cafetería Delicias de Verde tiene un inadecuado control de facturación				Mejorar la utilización de facturas mediante una aplicación informática orientada a la web.
Fuerzas Impulsadoras	**I**	**PC**	**I**	**PC**	**Fuerzas Bloqueadoras**
Utilización de hojas pre impreso según lo necesitado	3	4	3	4	No usar hojas pre impresas
Asignar persona responsable	3	4	2	4	Asignación incorrecta de la persona encargada
Manejar adecuadamente el ingreso de productos	3	4	3	4	No mantener la organización del registro por desinformación.
Capacitación de apoyo	3	4	3	4	Falta de Puntualidad ,Ausencia del personal
Coordinación de productos en stock entre el personal de turno	3	4	3	4	Confusión por desinformación sobre la sistematización por la web.

El análisis de la Matriz T permite visualizar la problemática presente, optimizando la búsqueda de soluciones para las fuerzas bloqueadoras presentes mediante las fuerzas impulsadoras, logrando así mejorar los aspectos deficientes percibidos en la empresa.

Anexo A.02

Matriz de Involucrados

Actores Involucrados	Intereses sobre el problema central	Problemas Percibidos	Recursos, Mandatos y Capacidades	Intereses sobre el Proyecto	Conflictos Potenciales
Gerente Propietario	Llevar de mejor manera el proceso de facturación, bajo las normas que establece el S.R.I	Poca Organización en el proceso de facturación	Reglamentos internos Recurso Humano, Recurso Tecnológico	Control Adecuado del proceso de facturación.	Que exista datos duplicados
Empleados	Generar la factura en menos tiempo	No tener facturas autorizadas por el S.R.I	Conocimientos claros para realizar el proceso	Mantener un control de clientes satisfechos.	Ninguno
Cliente	Los datos deben ser llenados con letra legible.	No recibir documento	De acuerdo a la Ley orgánica de defensa al consumidor	Mantener un buen servicio.	Factura caducada
Contabilidad	Cumplir con los requisitos legales del S.R.I	Información incoherente	Régimen de facturas	Cumplir adecuadamente con la declaración de impuestos	Duplicidad de información
Proveedores	Subir el porcentaje de productos	Los pedidos ya no son frecuentes	Recursos económicos Recurso Humano	Pedido recibido satisfactoriamente.	Escases de productos.

Esta tabla muestra el análisis y el interés de cada uno de los involucrados con la problemática al igual que las contrariedades y conflictos potenciales percibidos en el proyecto.

Anexo A.03
Caso de uso general del sistema

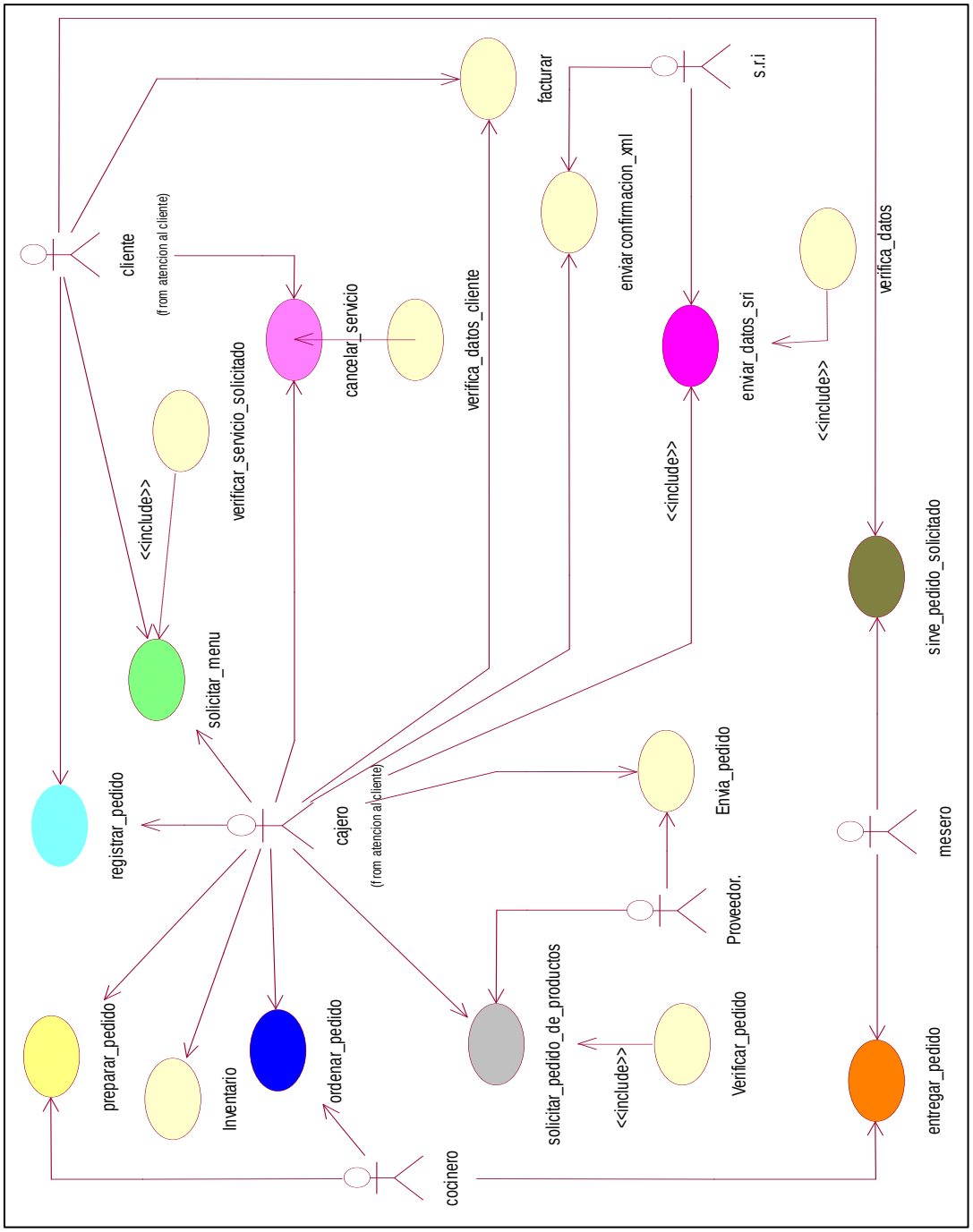


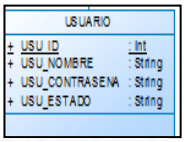
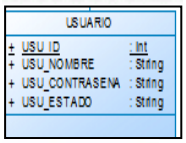
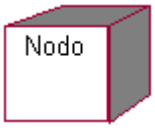
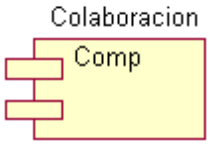
Figura 4.*Diagrama Caso de uso general del sistema*

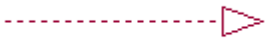
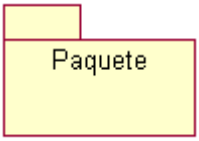
Anexo A.04

Tabla 12

Estándares de Diseño de clases

Fuente: http://www.monografias.com/trabajos28/proyecto-uml/proyecto-uml.shtml#element#ixzz3TifBkEDf

Clase		Define los atributos y los métodos de una serie de objetos. La clase implementa una o mas interfaces
Interfaz		Permite emplear un circulo para representar las interfaces, empleando la clase con el nombre de la cursiva.
Atributos		Son características de una clase pueden ser de tres tipos : public,private,protected
Atributo privado		Indica que el atributo solo será accesible desde dentro de la clase solo con sus métodos
Atributo Protected		Indica que el atributo no será accesible desde la fuera de la clase, pero si podrá ser accedido por métodos de la clase además de las subclases que se deriven.
Nodo		Elemento Físico que existe en tiempo de ejecución y representa un recurso computacional con capacidad de procesar.
Colaboración		Define una interacción entre elementos que cooperan para proporcionar un comportamiento mayor
Componentes		Parte física y por tanto reemplazable de un modelo, que agrupa un conjunto de interfaces, archivos de código fuente, clases, colaboraciones y proporciona la implementación de dichos elementos.
Dependencia		Es una relación entre dos elementos, tal que un cambio en uno puede afectar al otro.
Asociación		Es una relación estructural que resume un conjunto de enlaces que son conexiones entre objetos

Generalización		Es una relación en la que el elemento generalizado puede ser substituido por cualquiera de los elementos hijos, ya que comparten su estructura y comportamiento.
Realización		Es una relación que implica que la parte realizante cumple con una serie de especificaciones propuestas por la clase realizada (interfaces).
Elementos de Agrupación		Se emplea para organizar otros elementos en grupos.

Anexo A.05

Diagrama de clases

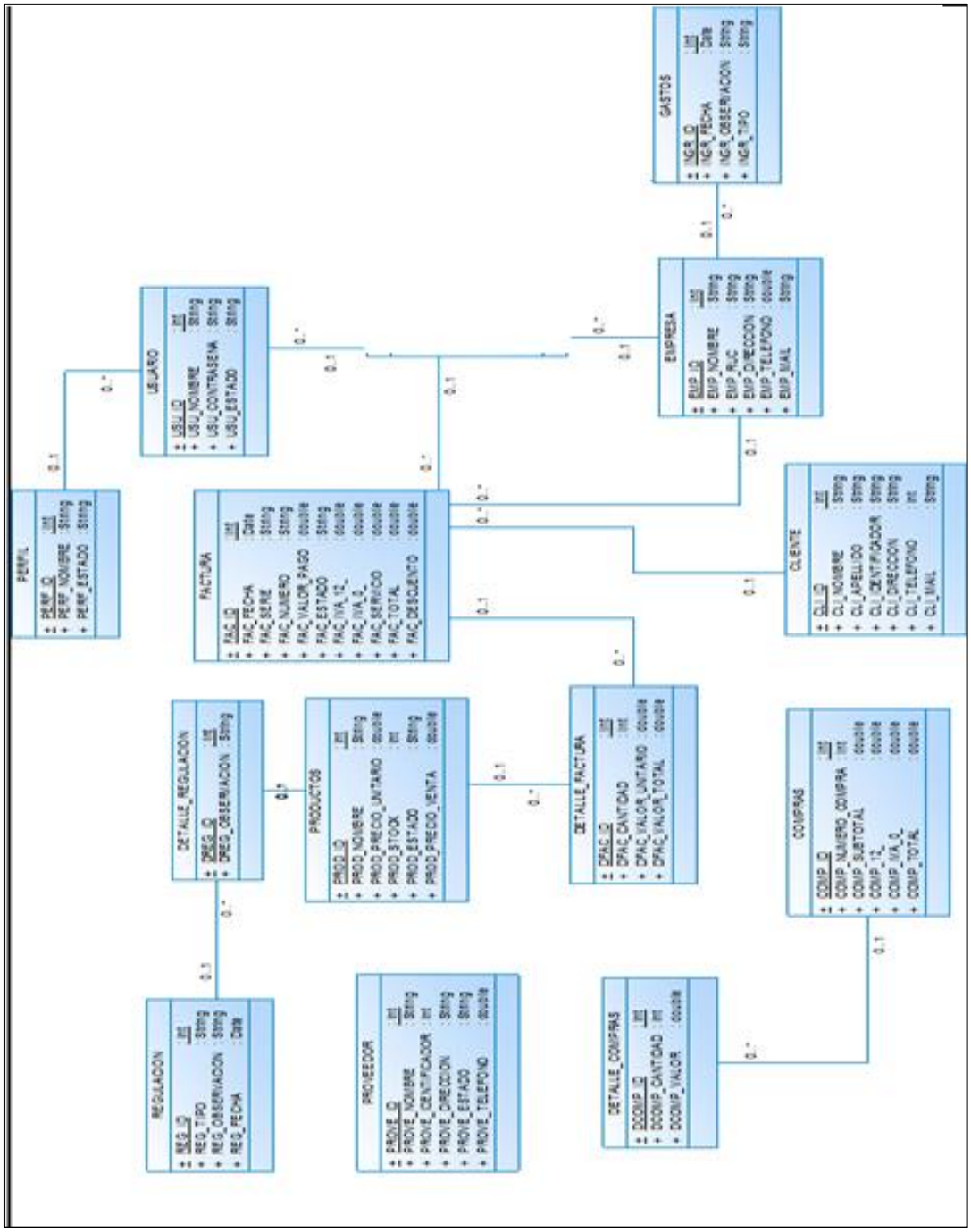


Figura 11:*Diagrama de clases*

Anexo A.06

Modelo Lógico - Físico

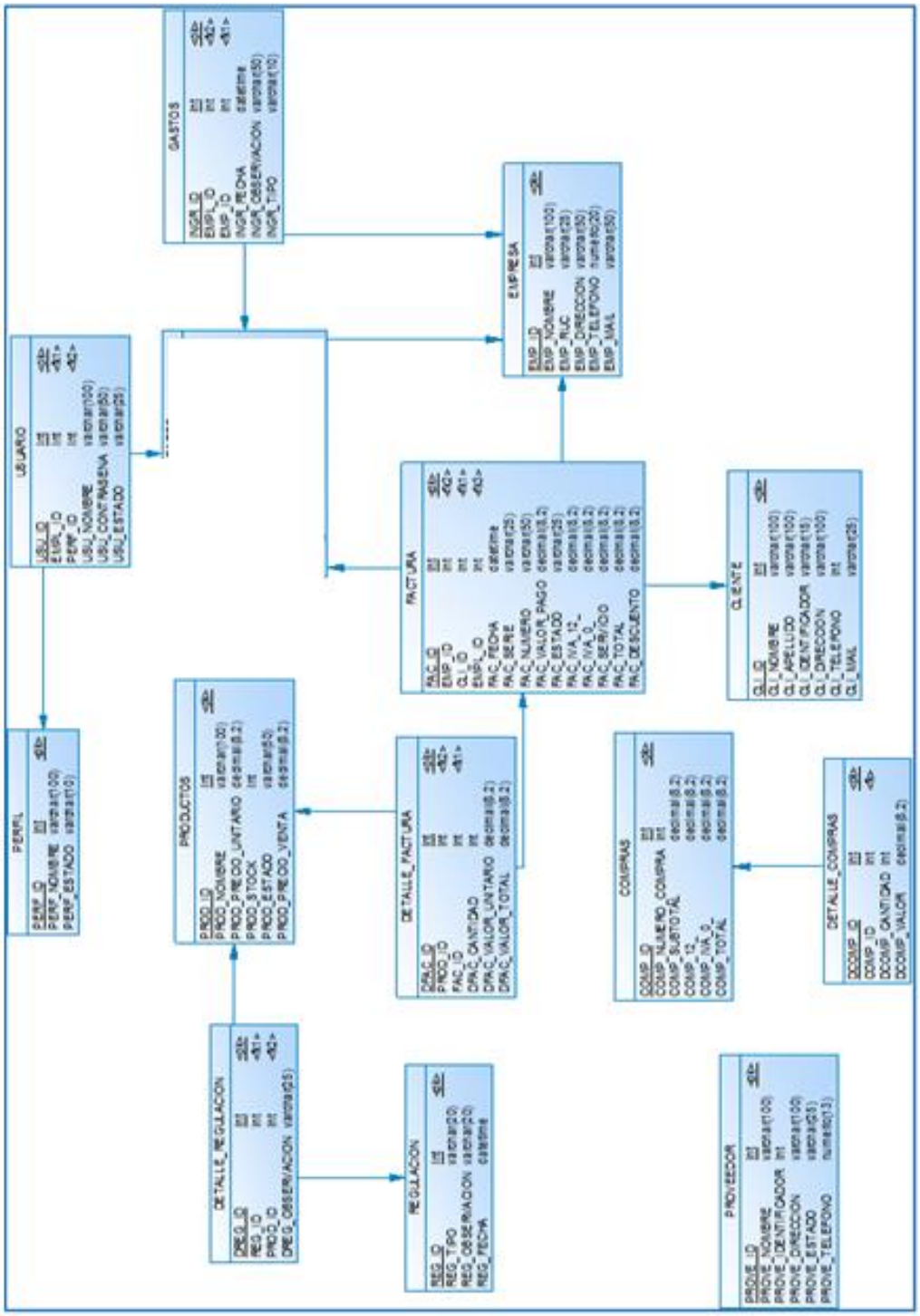


Figura 12: *Modelo Logico-Fisico*

Anexo A.07

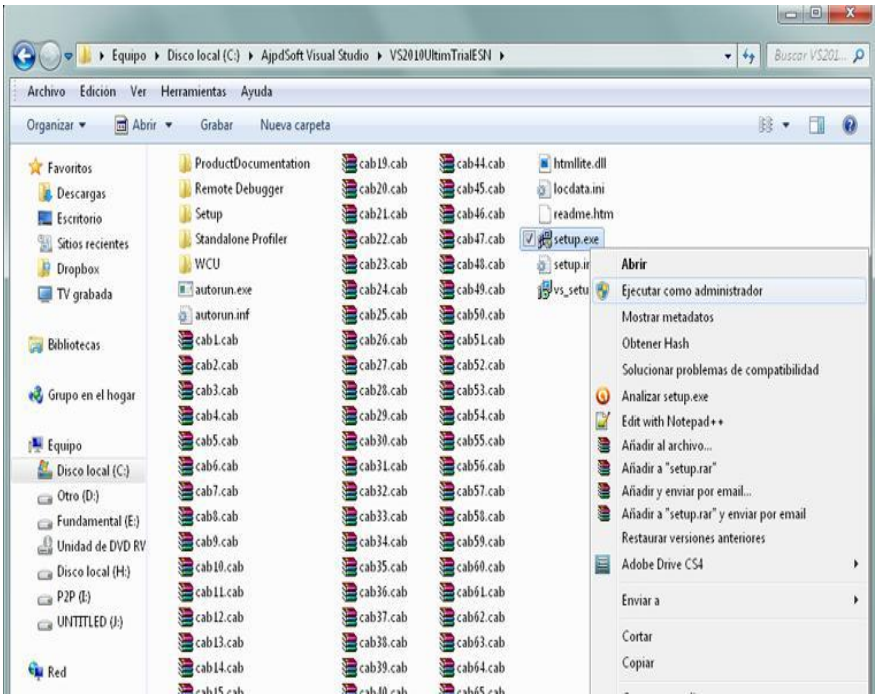
Cronograma

Figura 26: *Cronograma de Proyecto.*

MANUAL DE INSTALACIÓN

1Instalación de Visual Studio 2010

1. Ahora, realizamos click derecho en el archivo setup.exe y a continuación seleccionamos ejecutar como administrador.

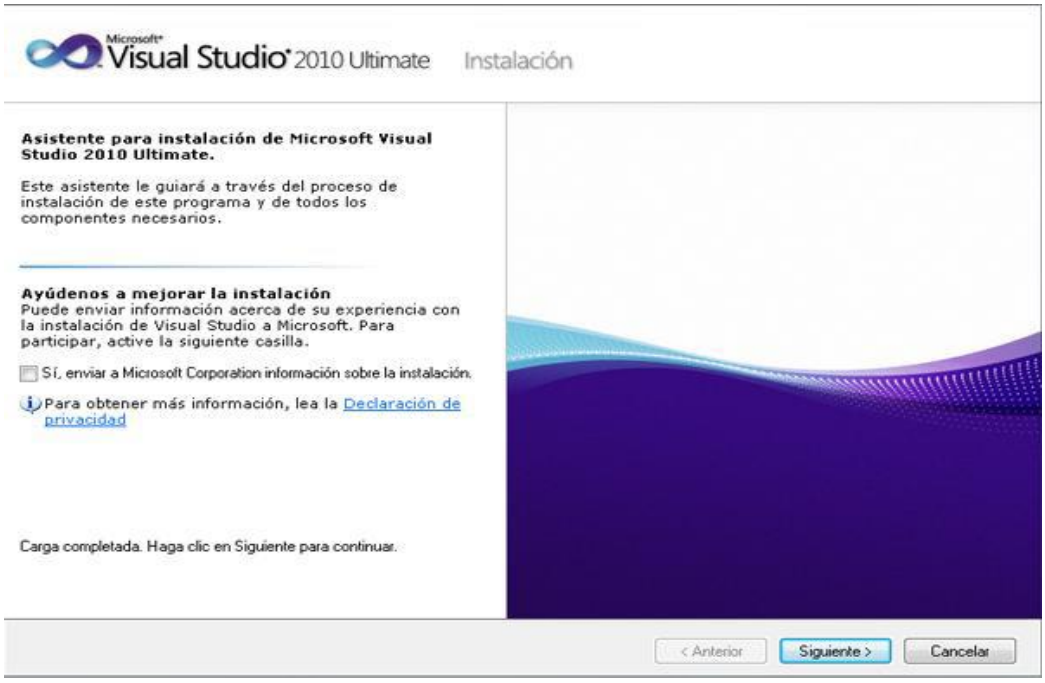


1.01 Figura 1 *Archivos instalador visual*

2. Se iniciará nuestro asistente de instalación, click en la primera opción: *instalar 2 Microsoft Visual Studio 2010*

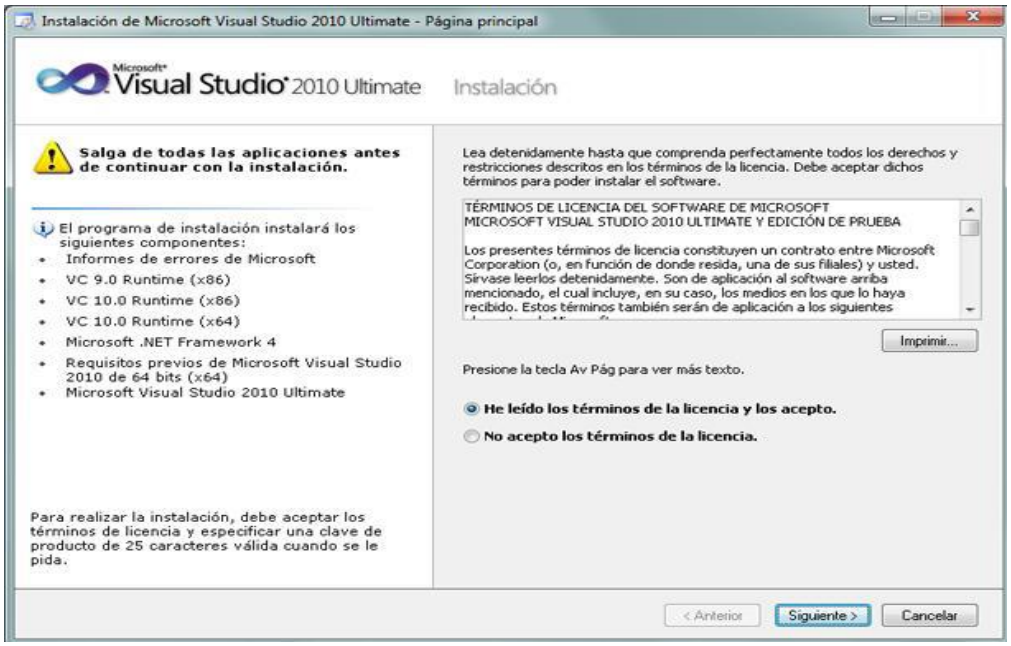
1.02 Figura 2 *Primer paso de Instalación*

2 Si nosotros deseamos habilitamos la opción de enviar a Microsoft información de la instalación, click en siguiente



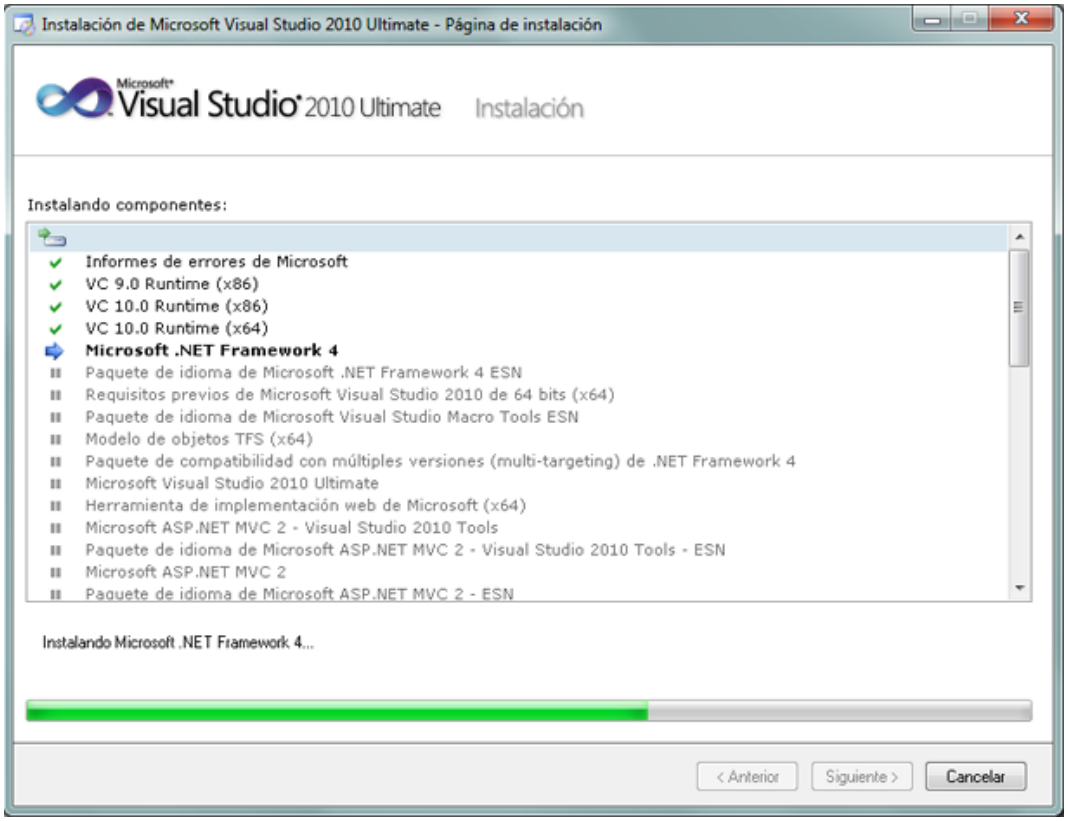
1.03 Figura 3 *Segundo paso de Instalación*

Nos desplegará una ventana, donde nos indica los términos y condiciones de uso y de instalación del programa, leemos los derechos planteados, click en aceptar los términos de contrato y presionamos Enter para ir al siguiente paso



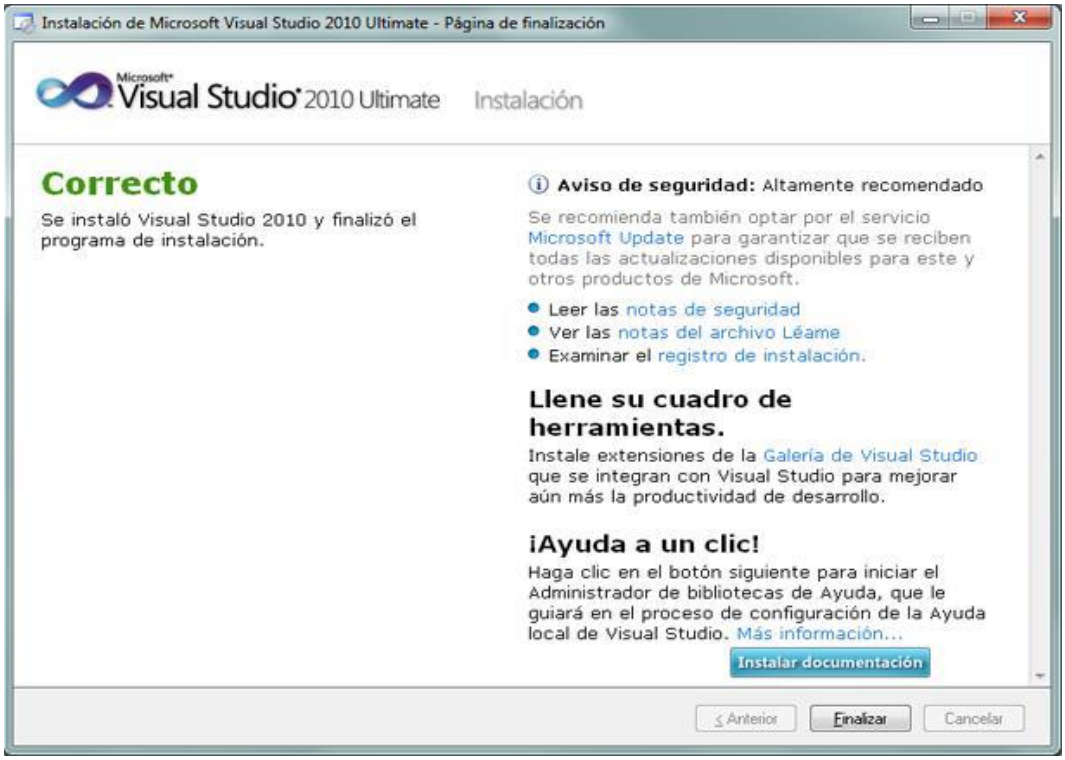
1.04 Figura 4 *Términos y Condiciones*

4.Se inicia la instalación, nos despliega un cuadro en donde nos indica todas las herramientas que se instalan



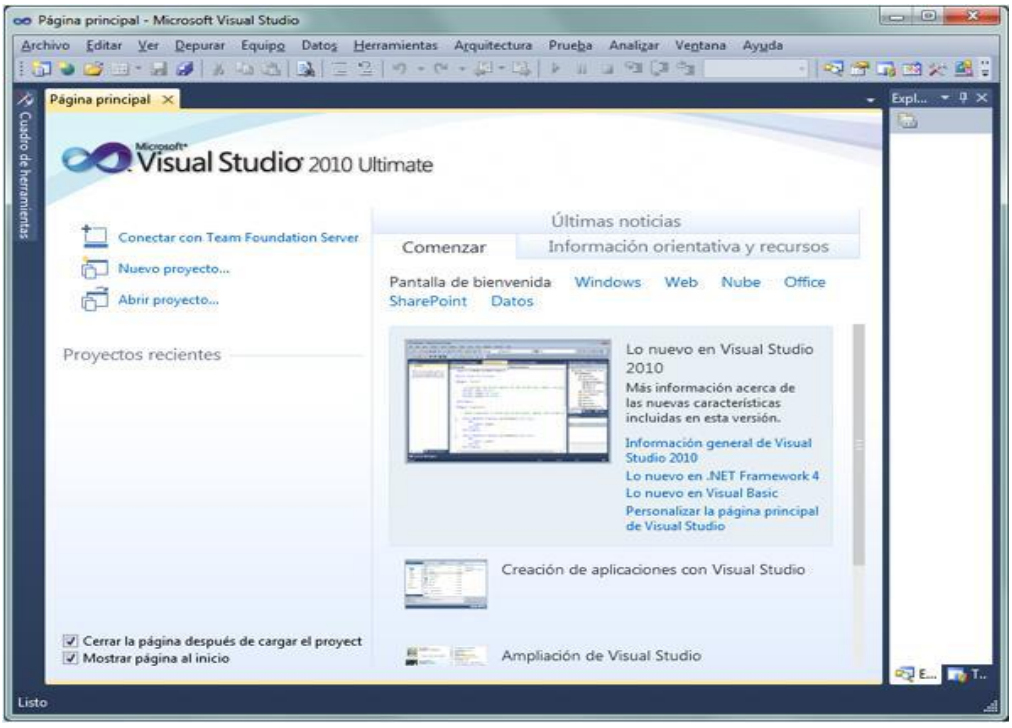
1.05 Figura 5 Instalación de Complementos

5.Una vez instalados todos los paquetes click en siguiente para continuar con la instalación, si la instalación se hizo de manera correcta, nos despliega un cuadro final indicando la finalización de la instalación y click en finalizar



1.06 Figura 6 Finalización de Instalación del Visual Studio 2010

6.Procedemos a reiniciar el equipo, para que toda la instalación sea la correcta.

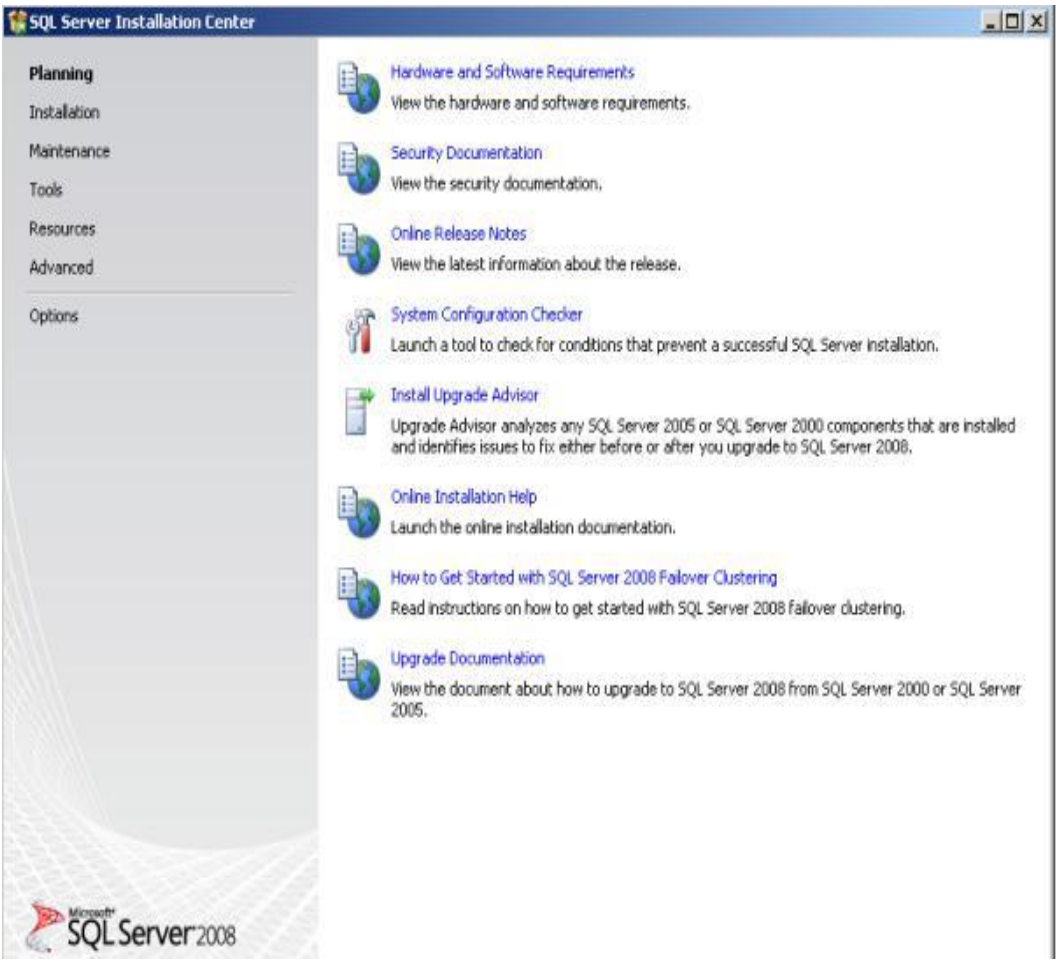


1.07 Figura 7 Ventana del Inicio del Sistema

Instalación de Nuestro Gestor de Base de Datos

Para nuestro fin y por su facilidad de conexión con Visual Studio 2010, trabajaremos como gestor de nuestra base de datos con SQL SERVER 2008

7.Click derecho en el setup.exe ejecutar como administrador

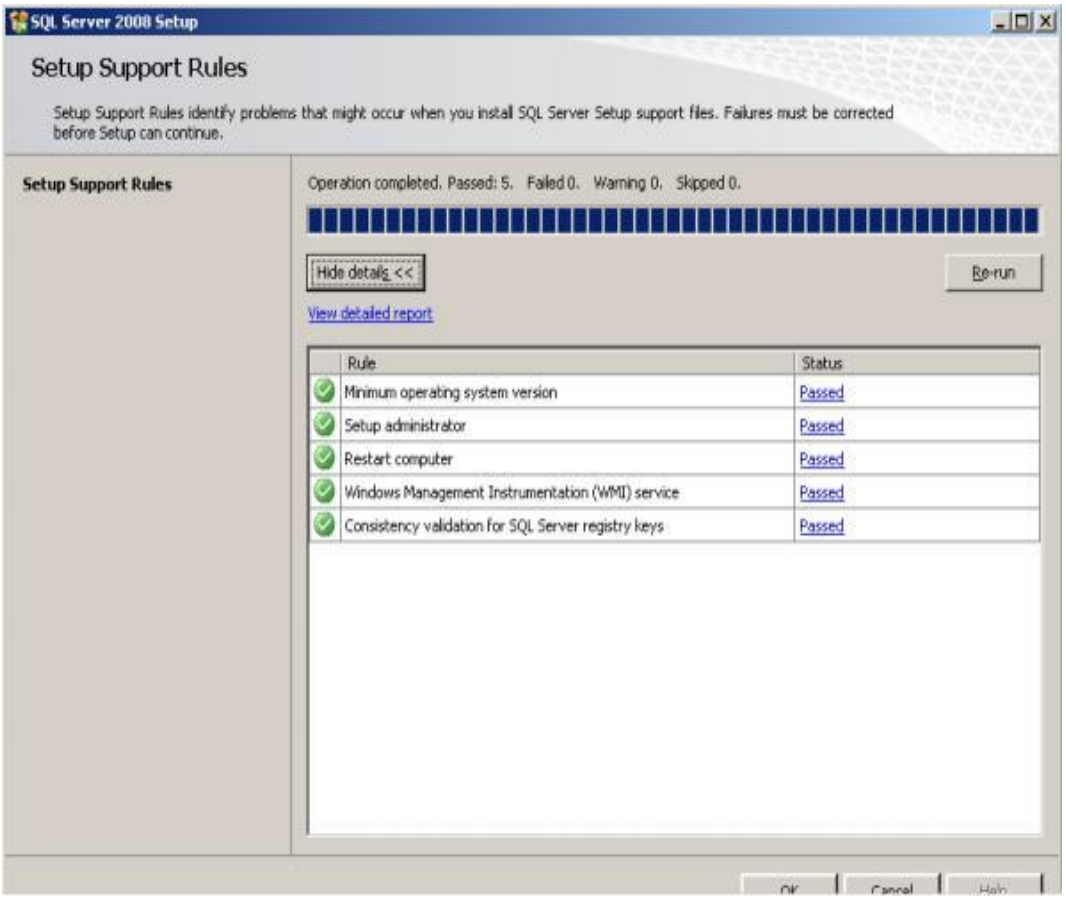


1.08 Figura 8Primer paso de Instalación Sql Server 2008

8.Click en la opción **Installation,** y seleccionamos New Sql Sever, click en siguiente.

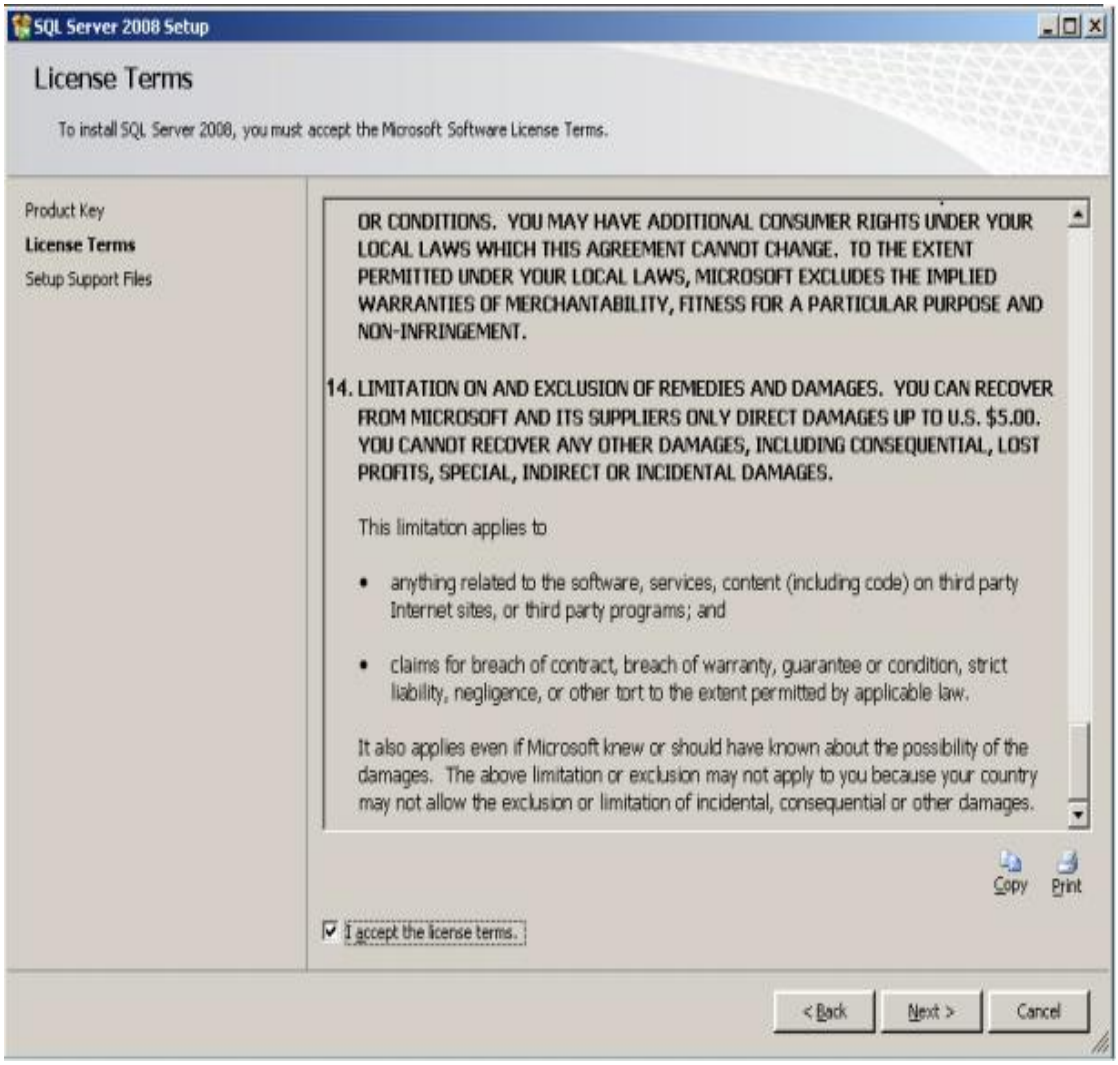
1.09 Figura 9 Elección de la Base de Datos

9.En el recuadro, realizamos click en siguiente para continuar con el proceso de instalación.



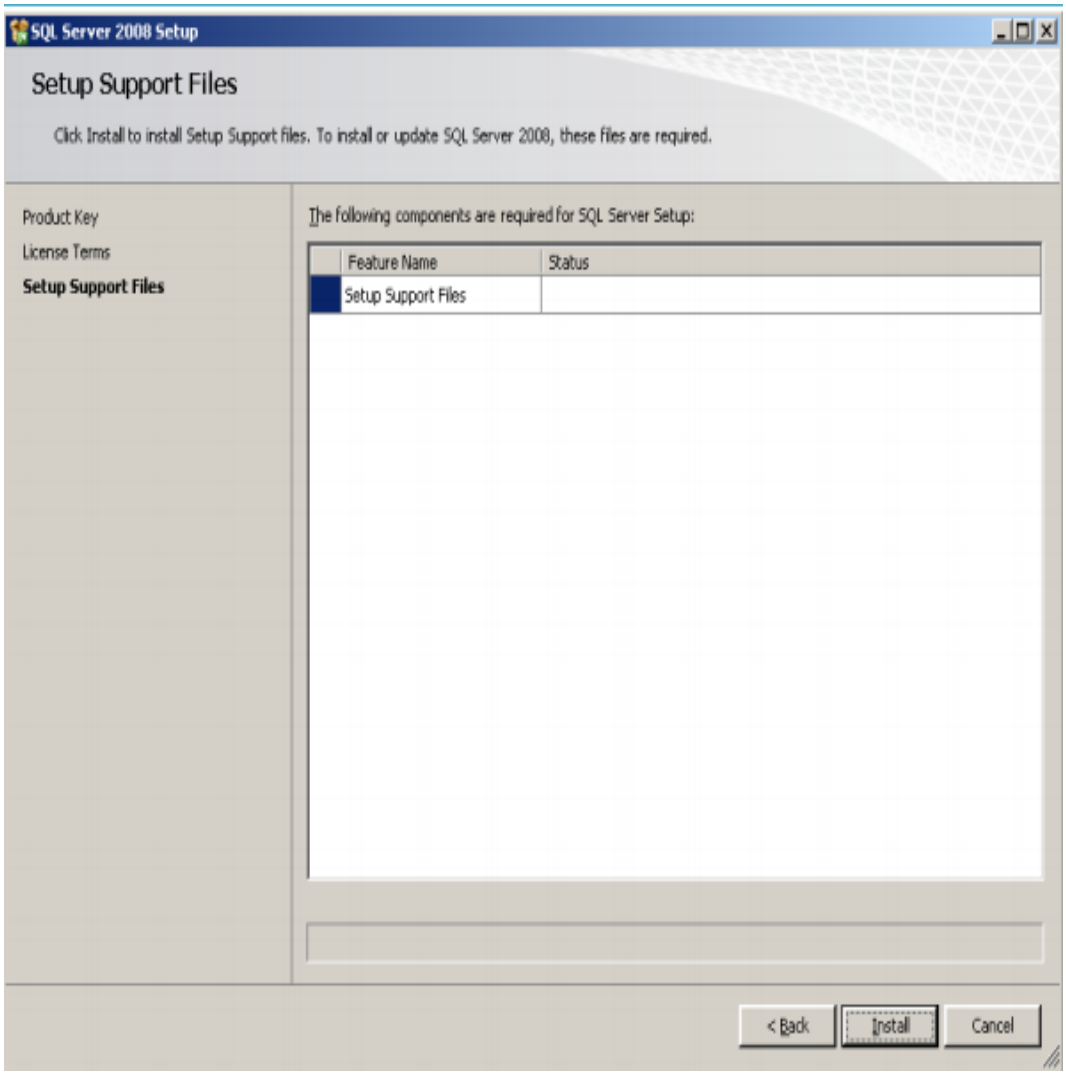
1.10Figura 10 Instalación de Complementos

10.Aceptamos los términos y condiciones que nos indican a continuación y click en siguiente



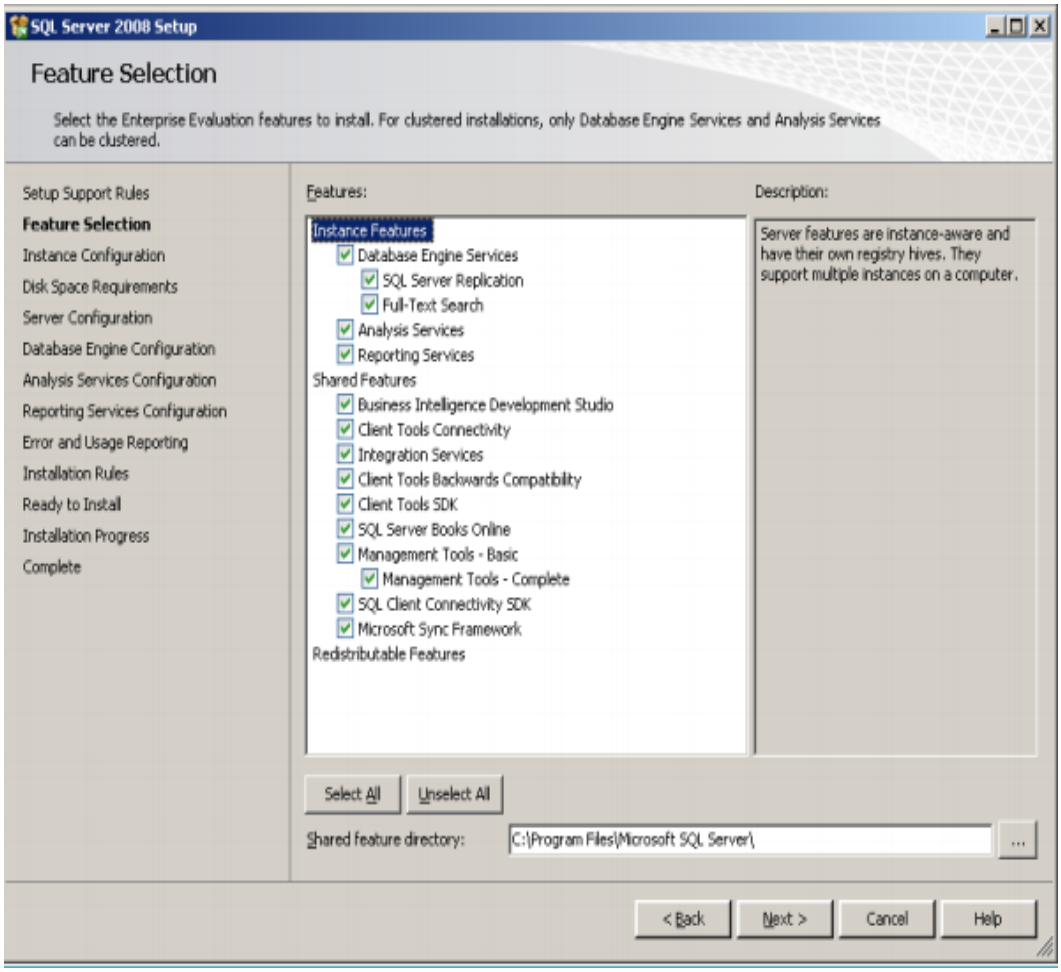
1.11Figura 11Términos y Condiciones de Uso

11.Click en Install para cargar las herramientas necesarias.



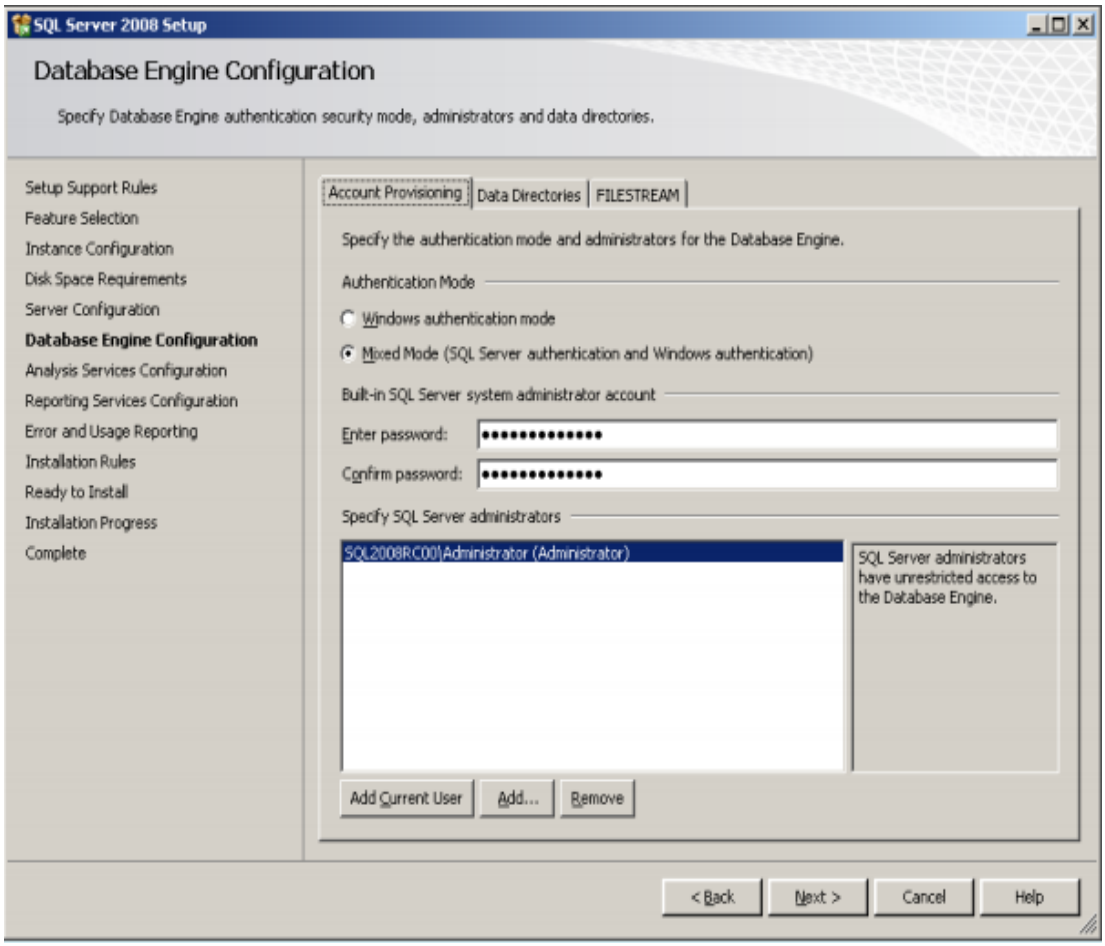
1.12 Figura 12 Instalación de las Herramientas

12.Ahora seleccionamos las características que vamos a utilizar en nuestro gestor.



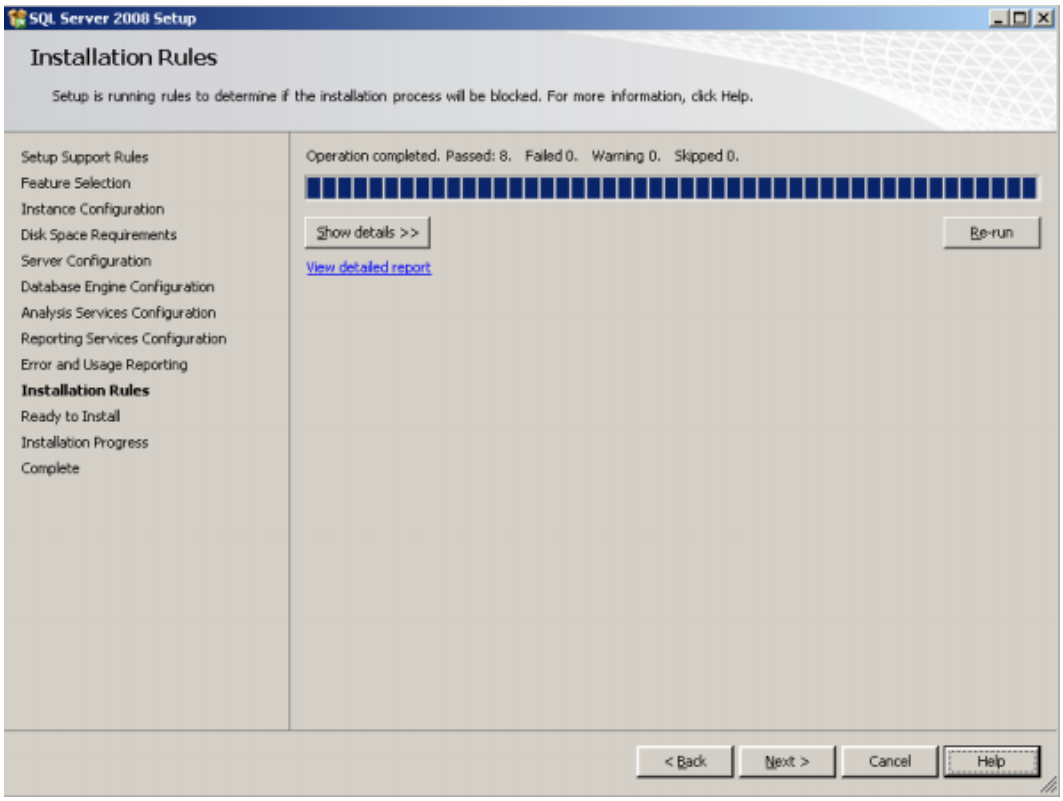
1.13 ***Figura 13*** Selección de Complementos a Instalar

13.Realizamos click en las ventanas siguientes hasta llegar a la ventana que indica si el usuario desea ingresar una contraseña propia o usar la de autentificación de windows, la decisión la tiene el usuario, nosotros usaremos la autentificación de Windows.



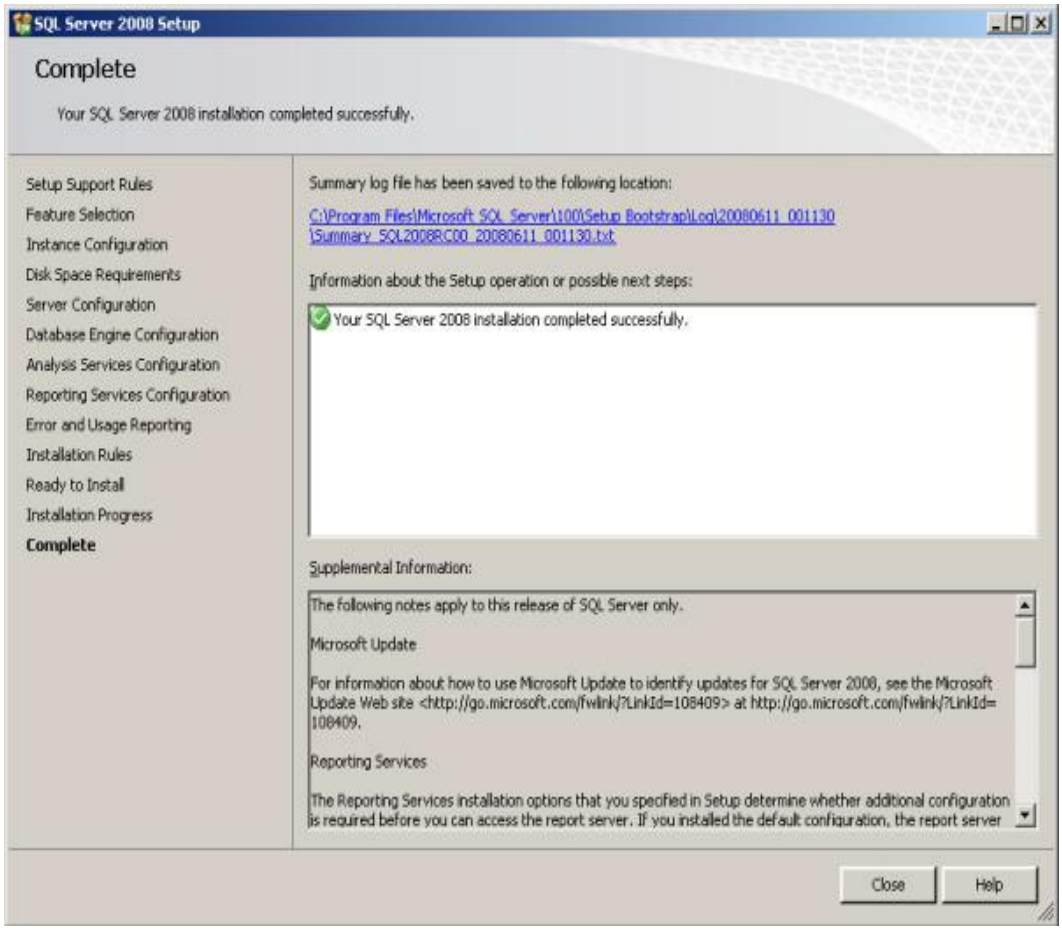
1.14 ***Figura 14*** Usuario y Contraseña del Servidor

14.Click en siguiente en las demás ventanas, hasta llegar al siguiente recuadro



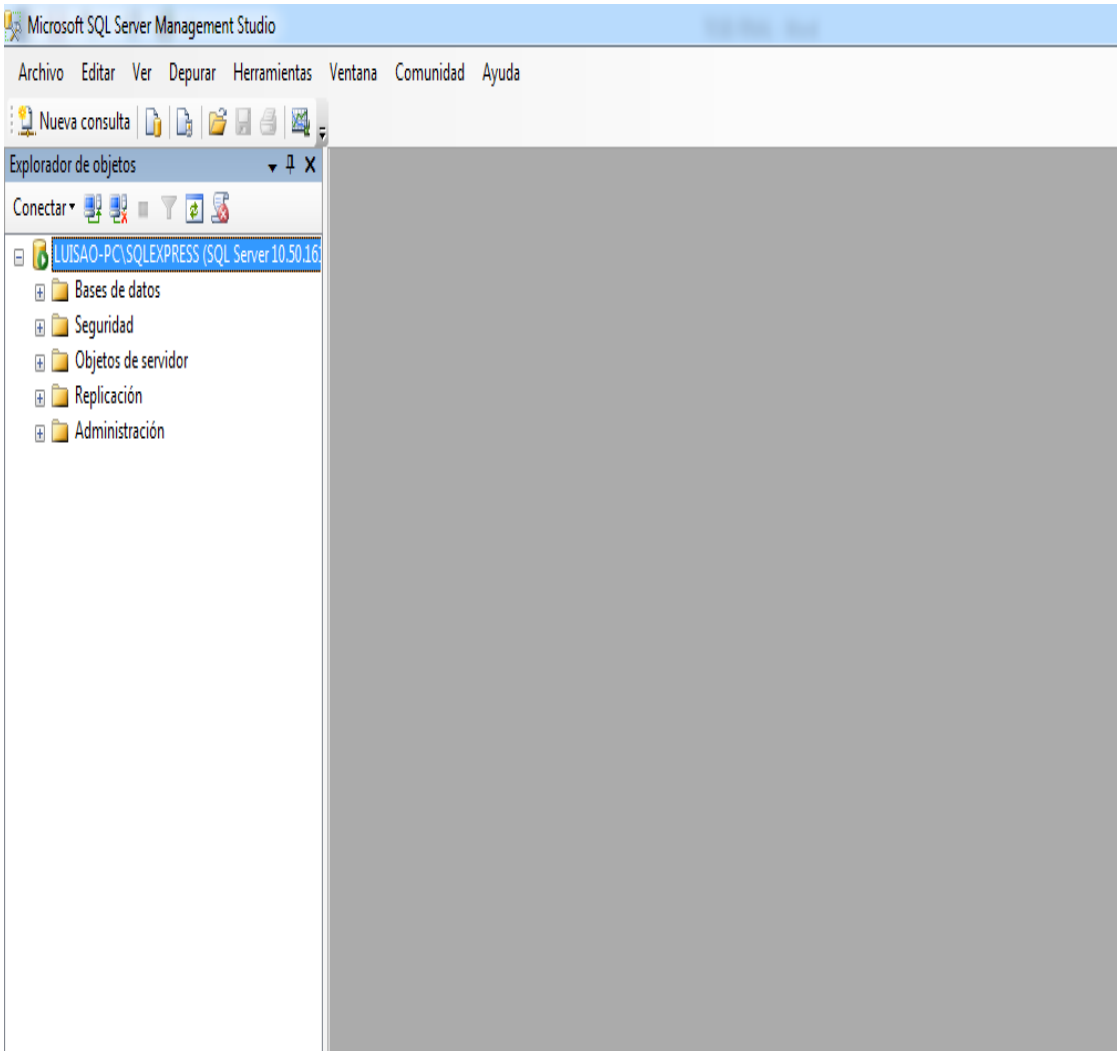
1.15 ***Figura 15*** Instalación de las Reglas del Servidor

15. Click en Next hasta culminar la instalación, al final veremos un recuadro que nos indicará la finalización del a instalación y click en Close.



1.16 Figura 16 Finalización de la Instalación del Gestor de Base de Datos

16.Para ver su correcta instalación, en nuestra barra de herramientas, buscamos Sql Server Management, lo ejecutamos y visualizaremos la siguiente pantalla.



1.17 Figura 17 Inicio de Nuestro Servidor

MANUAL DE USUARIO

- **Inicio de Login**

1.01 Figura 18. Inicio de sesión

En esta figura nos muestra los campos para ingresar al sistema con un usuario y contraseña.

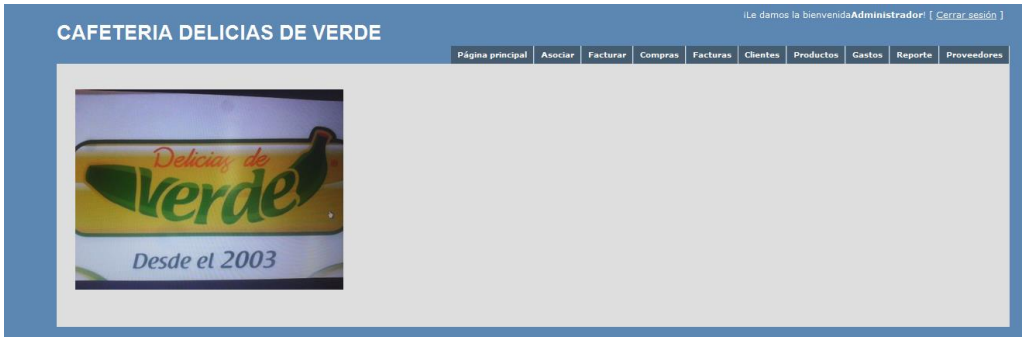
1er paso seleccione iniciar sesión

2 do paso digite su nombre de usuario y la contraseña asignada

3 er paso el sistema cuenta con la opción que dice recordar contraseña si el campo a sido seleccionado laproxima vez que el usuario digite su nombre la clave aparecerá automáticamente sin necesidad de volver a escribir

4 to paso Iniciar sesion

- **Pagina Principal**



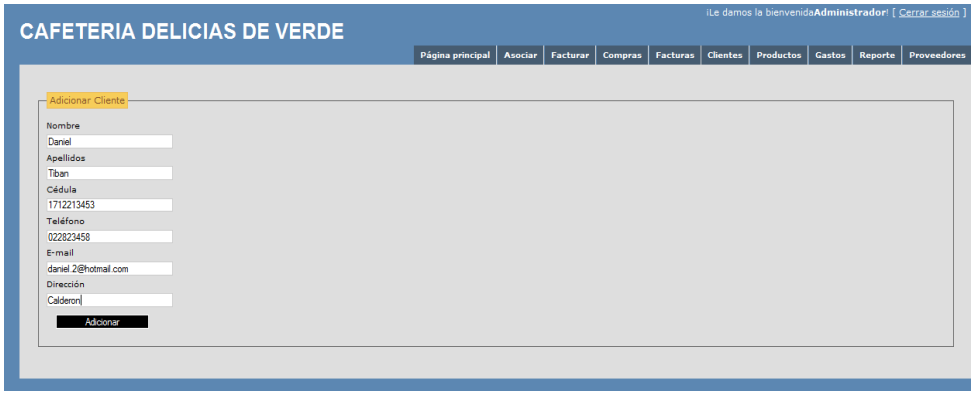
1.02 Figura 19. Pagina Principal

En esta figura nos muestra la página principal de la aplicación, esta nos aparecerá cuando el usuario a ya iniciado la sesión.

5 to Paso cuando ya este iniciada la aplicación nos aparecerá la pantalla principal

6 to Paso tendremos varias opciones para escoger como ingresar clientes,productos,compras,gastos,reporte de ventas .

Ingresar Cliente



1.03 Figura 20. Registro de cliente

Esta figura nos muestra todos los campos disponibles y obligatorios para registrar el cliente.

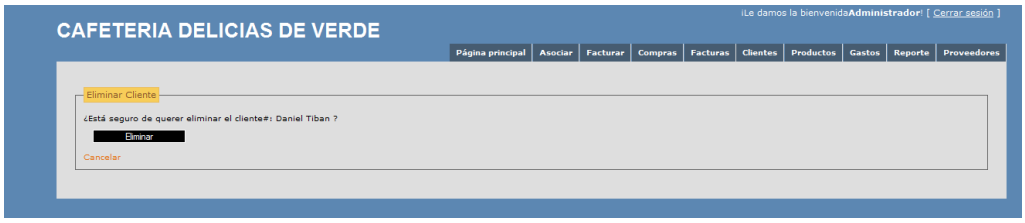
1er Paso Presionamos en la opción que dice Clientes

2do Paso nos aparecerá una nueva ventana la cual nos pide ingresar datos personales del cliente que se desea guardar

3 er Paso Recordar que todos los campos son obligatorios

4to Paso unavez llenados todos los campos digitamos el botón que dice adicionar.

- **Mensaje Eliminar Cliente**



1.04 Figura 21. Eliminar de cliente

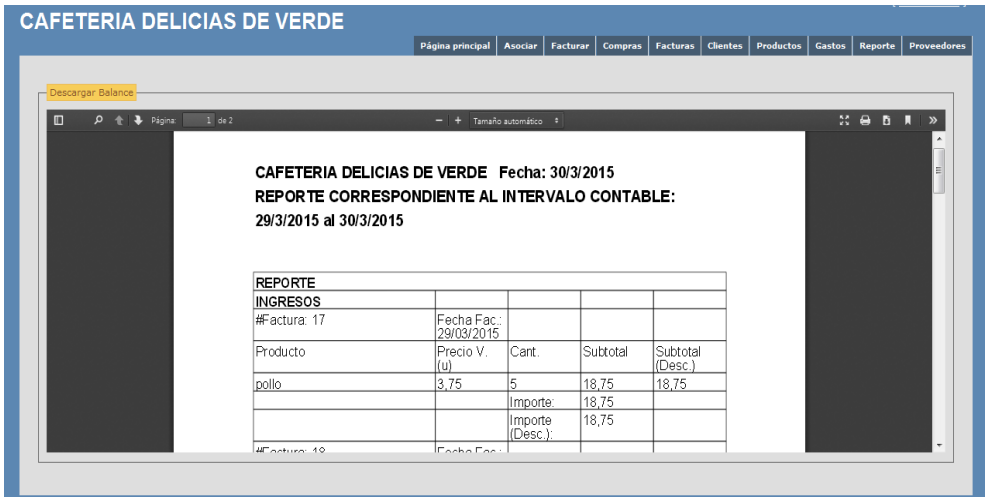
Esta figura nos muestra si deseamos eliminar un cliente que no a realizado una compra caso contrario no se podrá eliminar.

5to Paso en la interfaz del cliente tenemos la opción de eliminar al cliente

6to Paso Presione en el botón que dice eliminar le aparecerá una mensaje indicándole si esta seguro de querer eliminar el cliente si esta seguro de eliminarlo presione en aceptar caso contrario presione cancelar

7mo Paso Recordar que solo se podrán eliminar a los clientes que no se les a facturado una vez facturado el cliente automáticamente el sistema no le permitirá eliminar el cliente.

- **Selección de la Fecha del Reporte Diario**
- **Reporte Generado**



1.05Figura 22. Gererar Reporte

En esta figura nos muestra el reporte que solicitamos según el rango de fechas seleccionadas.

1er Paso para generar el reporte presione en el botón que dice reporte

2do paso le aparecerá una ventana donde debe definir la fecha de inicio y la fecha final del reporte

3er paso una vez seleccionado la fecha presionar la opción que dice generar y nos mostrara el reporte deseado.

- **Crear Factura**

1 er Paso Digitamos en el icono que dice Facturar

2do Paso Nos aparecerá una nueva ventana donde debemos ingresar el ruc del cliente

3er Paso presionamos el botón buscar y se nos cargara automáticamente los datos del cliente

4to Paso Ingresamos los productos que el cliente consumio

5to Paso En la ventana nos mostrara un detalle la factura que se le va a facturar

- **Crear Gastos**

  1 er Paso Digitamos en el icono que dice Gastos.

  2do Paso Nos aparecerá una nueva ventana donde debemos ingresar la descripción del gasto.

  3er Paso ingresar el valor del gasto

  4to Paso guardar el gasto

- **Crear Compra**

  1 er Paso Digitamos en el icono que dice Compra.

  2do Paso Nos aparecerá una nueva ventana donde debemos ingresar el proveedor.

  3er Paso ingresar el ruc del proveedor

  4to Paso ingresamos el producto

  5to Paso ingresamos el precio compra

  6to Paso ingresamos el precio a vender

  7mo Paso Guardamos la compra

MANUAL TÉCNICO

1.01 Manual técnico

En el manual técnico se detalla parte de la estructura de la programación realizada.

Con la finalidad que el departamento técnico pueda comprender la lógica de programación empleada.

2.01 Diccionario Técnico

Table	column	Type	Precisión	max length	Permits Nulls	Es Autonumerico
Cliente	id_cliente	int	NULL	4	NO	SI
Cliente	cedula	int	NULL	4	SI	NO
Cliente	nombre	varchar	NULL	50	SI	NO
Cliente	apellidos	varchar	NULL	50	SI	NO
Cliente	direccion	text	NULL	16	SI	NO
Cliente	telefono	int	NULL	4	SI	NO
Cliente	mail	varchar	NULL	50	SI	NO
Comp_Prod	id_comp_prod	int	NULL	4	NO	SI
Comp_Prod	id_comp	int	NULL	4	SI	NO
Comp_Prod	id_prod	int	NULL	4	SI	NO
Comp_Prod	stock	int	NULL	4	SI	NO
Compra	id_compra	int	NULL	4	NO	SI
Compra	fecha	datetime	NULL	8	SI	NO
Empleado	id_empleado	int	NULL	4	NO	SI
Empleado	cedula	int	NULL	4	SI	NO
Empleado	nombre	varchar	NULL	50	SI	NO
Empleado	apellidos	varchar	NULL	50	SI	NO
Empleado	direccion	text	NULL	16	SI	NO
Empleado	mail	varchar	NULL	50	SI	NO
Empleado	telefono	int	NULL	4	SI	NO
Empleado	estado	varchar	NULL	50	SI	NO
Empleado	salario	real	NULL	4	SI	NO
Empleado	usuario	varchar	NULL	50	SI	NO
Empleado	rol	varchar	NULL	50	SI	NO

Fac_Prod	id_fac_prod	int	NULL	4	NO	SI
Fac_Prod	id_fac	int	NULL	4	SI	NO
Fac_Prod	id_prod	int	NULL	4	SI	NO
Fac_Prod	cantidad	int	NULL	4	SI	NO
Factura	id_factura	int	NULL	4	NO	SI
Factura	id_cliente	int	NULL	4	SI	NO
Factura	id_empleado	varchar	NULL	50	SI	NO
Factura	fecha	datetime	NULL	8	SI	NO
Factura	estado	varchar	NULL	50	SI	NO
Factura	servicio	real	NULL	4	SI	NO
Factura	descuento	real	NULL	4	SI	NO
Gasto	id_gasto	int	NULL	4	NO	SI
Gasto	fecha	datetime	NULL	8	SI	NO
Gasto	descripcion	text	NULL	16	SI	NO
Gasto	importe	float	53	8	SI	NO
Producto	id_producto	int	NULL	4	NO	SI
Producto	nombre	varchar	NULL	50	SI	NO
Producto	precio_comprar	real	NULL	4	SI	NO
Producto	stock	int	NULL	4	SI	NO
Producto	precio_venta	int	NULL	4	SI	NO
Producto	descripcion	text	NULL	16	SI	NO
Producto	descuento	float	53	8	SI	NO
Prov_Comp	id_prov_comp	int	NULL	4	NO	SI
Prov_Comp	id_prov	int	NULL	4	SI	NO
Prov_Comp	id_comp	int	NULL	4	SI	NO
Prov_Prod	id_prov_prod	int	NULL	4	NO	SI
Prov_Prod	id_prov	int	NULL	4	SI	NO
Prov_Prod	id_prod	int	NULL	4	SI	NO
Proveedor	id_provedor	int	NULL	4	NO	SI
Proveedor	ruc	int	NULL	4	SI	NO
Proveedor	nombre	varchar	NULL	50	SI	NO
Proveedor	direccion	text	NULL	16	SI	NO
Proveedor	estado	varchar	NULL	50	SI	NO
Proveedor	telefono	int	NULL	4	SI	NO
Proveedor	descripcion	text	NULL	16	SI	NO

Proveedor	mail	varchar	NULL	50	SI	NO	

3.01 Codigo General del Sistema

Codigo Cliente

```
{
   [Authorize(Roles = "admin, member")]
   public class ClienteController : Controller
   {
      Clientes dbC = new Clientes();
      Facturas dbF = new Facturas();
      Validacion valid = new Validacion();
      //
      // GET: /Cliente/

      public ActionResult Index()
      {
         ViewData["errores"] = valid.Errores;
         List<Cliente> lista = (from s in dbC.Cliente select s).ToList();
         List<bool> listDel = new List<bool>();
         foreach (Cliente item in lista)
         {
            List<Factura> lF = (from s in dbF.Factura where s.id_cliente == item.id_cliente select
s).ToList();
            if (lF.Count > 0)
               listDel.Add(true);
            else
               listDel.Add(false);
         }
         ViewData["eliminar"] = listDel;
         return View(lista);
      }

      //
      // GET: /Cliente/Details/5

      public ActionResult Details(int id)
      {
         Cliente cli = (from s in dbC.Cliente where s.id_cliente == id select s).FirstOrDefault();
         return View(cli);
      }

      //
      // GET: /Cliente/Create

      public ActionResult Create()
      {
         ViewData["errores"] = valid.Errores;
         return View();
      }

      //
      // POST: /Cliente/Create

      [HttpPost]
      public ActionResult Create(Cliente cli)
      {
         try
```

```
        {
            // TODO: Add insert logic here
            valid.validate_form_cliente(cli);
            List<Cliente> list = (from s in dbC.Cliente where s.cedula ==cli.cedula select s).ToList();
            if (list.Count > 0)
                valid.Errores.Add("El cliente ya existe ");
            if (valid.isValid())
            {
                dbC.AddToCliente(cli);
                dbC.SaveChanges();
                return RedirectToAction("Index");
            }
            else
            {
                ViewData["errores"] = valid.Errores;
                return View();
            }
        }
        catch
        {
            return View();
        }
    }

    //
    // GET: /Cliente/Edit/5

    public ActionResult Edit(int id)
    {
        ViewData["errores"] = valid.Errores;
        Cliente cli = (from s in dbC.Cliente where s.id_cliente == id select s).FirstOrDefault();
        return View(cli);
    }

    //
    // POST: /Cliente/Edit/5

    [HttpPost]
    public ActionResult Edit(int id, Cliente c)
    {
        try
        {
            // TODO: Add update logic here
            valid.validate_form_cliente(c);
            List<Cliente> list = (from s in dbC.Cliente where s.cedula == c.cedula select s).ToList();
            Cliente c1 = (from s in dbC.Cliente where s.id_cliente == id select s).FirstOrDefault();
            if (c.cedula != c1.cedula && list.Count > 0)
                valid.Errores.Add("Ya existe un cliente con el número de cédula entrado");
            if (valid.isValid())
            {
                dbC.ContextOptions.LazyLoadingEnabled = false;
                var cli = (from s in dbC.Cliente
                           where s.id_cliente == id
                           select s).FirstOrDefault();
                cli.cedula = c.cedula;
                cli.nombre = c.nombre;
                cli.apellidos = c.apellidos;
                cli.direccion = c.direccion;
                cli.telefono = c.telefono;
                cli.mail = c.mail;
                dbC.SaveChanges();
                return RedirectToAction("Index");
```

```
            }
            else
            {
                ViewData["errores"] = valid.Errores;
                return View(c);
            }
        }
        catch
        {
            return View();
        }
    }

    //
    // GET: /Cliente/Delete/5

    public ActionResult Delete(int id)
    {
        Cliente cli = (from s in dbC.Cliente where s.id_cliente == id select s).FirstOrDefault();
        return View(cli);
    }

    //
    // POST: /Cliente/Delete/5

    [HttpPost]
    public ActionResult Delete(int id, Cliente c)
    {
        try
        {
            // TODO: Add delete logic here
            dbC.ContextOptions.LazyLoadingEnabled = false;
            var cli = (from s in dbC.Cliente
                       where s.id_cliente == id
                       select s).FirstOrDefault();
            dbC.DeleteObject(cli);
            dbC.SaveChanges();
            return RedirectToAction("Index");
        }
        catch
        {
            return View();
        }
    }
  }
}
```

Codigo Factura

```
{
  [Authorize(Roles = "admin, member")]
  public class FacturaController : Controller
  {
    Facturas dbF = new Facturas();
    Clientes dbC = new Clientes();
    Productos dbP = new Productos();
    FacProds dbFP = new FacProds();
    Validacion valid = new Validacion();
    //
    // GET: /Factura/

    public ActionResult Index()
    {
```

```
            List<Factura> lista = (from s in dbF.Factura select s).ToList();
            List<FacturaListar> listaF = new List<FacturaListar>();

            foreach (Factura item in lista)
            {
                Cliente cli = (from s in dbC.Cliente where s.id_cliente == item.id_cliente select
s).FirstOrDefault();
                FacturaListar f = new FacturaListar();
                f.cli = cli;
                f.fac = item;
                listaF.Add(f);
            }
            return View(listaF);
        }


        public FacturaMostrar showFactura(int id)
        {
            Factura fac = (from s in dbF.Factura where s.id_factura == id select s).FirstOrDefault();
            List<Fac_Prod> lFP = (from s in dbFP.Fac_Prod where s.id_fac == id select s).ToList();

            List<Producto> lP = new List<Producto>();

            for (int i = 0; i < lFP.Count; i++)
            {
                int idP = int.Parse(lFP[i].id_prod.ToString());
                Producto prod = (from s in dbP.Producto where s.id_producto == idP select s).FirstOrDefault();
                lP.Add(prod);

            }

            Cliente cli = (from s in dbC.Cliente where s.id_cliente == fac.id_cliente select s).FirstOrDefault();

            FacturaMostrar fm = new FacturaMostrar();
            fm.fac = fac;
            fm.cli = cli;
            fm.prods = lP;
            fm.facpros = lFP;

            return fm;
        }


        public void generarXML(FacturaMostrar fm, float totSI, float tot)
        {
            //Declaramos el documento y su definición
            XDocument factura = new XDocument(
                new XDeclaration("1.0", "utf-8", null));

            //Creamos el nodo raiz y lo añadimos al documento
            XElement nodofactura = new XElement("factura");
            factura.Add(nodofactura);

            //Creamos el nodo infotributaria y el contenido con sus nodos hijos
            XElement infoTributaria = new XElement("infoTributaria");
            infoTributaria.Add(new XElement("ambiente", 1));
            infoTributaria.Add(new XElement("tipoEmision", 1));
            infoTributaria.Add(new XElement("razonSocial", "LOURDES JESSENIA GAON SABANDO"));
            infoTributaria.Add(new XElement("nombreComercial", "DELICIAS DE VERDE"));
            infoTributaria.Add(new XElement("ruc", "1709643736001"));
            infoTributaria.Add(new XElement("claveAcceso",
"0000000000000000000000000000000000000000000000000000"));
```

```
        infoTributaria.Add(new XElement("codDoc", "01"));
        infoTributaria.Add(new XElement("estab", "001"));
        infoTributaria.Add(new XElement("ptoEmi", "001"));
        infoTributaria.Add(new XElement("secuencial", "000000001"));
        infoTributaria.Add(new XElement("dirMatriz", "CHURULOMA ALTO, GEOVANNY CALLES
S/N Y DERBY"));

        //Creamos el nodo infoFactura
        XElement infoFactura = new XElement("infoFactura");
        infoFactura.Add(new XElement("fechaEmision", DateTime.Today.ToString("dd/MM/yyyy")));
        infoFactura.Add(new XElement("dirEstablecimiento", "CHURULOMA ALTO, GEOVANNY
CALLES S/N Y DERBY"));
        infoFactura.Add(new XElement("contribuyenteEspecial", 1001));
        infoFactura.Add(new XElement("obligadoContabilidad", "NO"));
        infoFactura.Add(new XElement("tipoIdentificacionComprador", "PASAPORTE"));
        infoFactura.Add(new XElement("razonSocialComprador", fm.cli.nombre + " "+fm.cli.apellidos));
        infoFactura.Add(new XElement("identificacionComprador", fm.cli.cedula));
        infoFactura.Add(new XElement("totalSinImpuestos", totSI - totSI*0.12));
        infoFactura.Add(new XElement("totalDescuento", 0));
        infoFactura.Add(new XElement("totalConImpuestos",
          new XElement("totalImpuesto",
            new XElement("codigo", 2),
            new XElement("baseImponible", 12.12),
            new XElement("tarifa", 0),
            new XElement("valor", totSI*0.12))));
        infoFactura.Add(new XElement("propina", 0));
        infoFactura.Add(new XElement("importeTotal", tot));
        infoFactura.Add(new XElement("moneda", "DOLAR"));

        //nodo detalles factura
        int cont = 0;
        XElement detalles = new XElement("detalles");
        foreach (Producto item in fm.prods)
        {
          double? prodsub = ((item.precio_venta * fm.facpros[cont].cantidad) - item.descuento / 100 *
(item.precio_venta * fm.facpros[cont].cantidad));
          detalles.Add(new XElement("detalle",
          new XElement("codigoPrincipal", item.id_producto),
          new XElement("descripcion", item.nombre),
          new XElement("cantidad",fm.facpros[cont].cantidad),
          new XElement("precioUnitario", item.precio_venta),
          new XElement("descuento", 0),
          new XElement("precioTotalSinImpuesto", prodsub-prodsub*0.12),
          new XElement("impuestos", new XElement("impuestos", new XElement("codigo", 2),
            new XElement("codigoPorcentaje", 0),
            new XElement("tarifa", 0),
            new XElement("baseImponible", 10.36),
            new XElement("valor", prodsub*0.12)))));
        }


        //nodo informacion adicional

        XElement infoAdicional = new XElement("infoAdicional");
        infoAdicional.Add(new XElement("campoAdicional", "nombre=caja"));

        //Añadimos los nodos y escribimos en el documento
        nodofactura.Add(infoTributaria);
        nodofactura.Add(infoFactura);
        nodofactura.Add(detalles);
        nodofactura.Add(infoAdicional);
```

```
        //string cedula = "Factura";
        var path = Path.Combine(Server.MapPath("~/Content"), fm.fac.id_factura + ".xml");
        factura.Save(@path);
        //factura.Save(Server.MapPath("/Xml"+cedula+".xml"));
    }

    public void generarPdf(FacturaMostrar fm)
    {

        Factura fac = fm.fac;
        Cliente cli = fm.cli;
        List<Producto> listaP = fm.prods;
        List<Fac_Prod> listaFP = fm.facpros;

        // step 1: creation of a document-object
        Document document = new Document(PageSize.LETTER, 20, 20, 20, 20);

        try
        {
            // step 2:
            // we create a writer that listens to the document
            // and directs a PDF-stream to a file
            var path = Path.Combine(Server.MapPath("~/Content"), fac.id_factura + ".pdf");
            PdfWriter writer = PdfWriter.GetInstance(document,
                        new FileStream(path, FileMode.Create));

            // step 3: we open the document
            document.Open();
            Font arial = FontFactory.GetFont("Arial", 12, Font.BOLD);
            Font cabecera = FontFactory.GetFont("Arial", 14, Font.BOLD);
            Font DESC = FontFactory.GetFont("Arial", 9);
            DateTime fecha = DateTime.Now;
            document.Add(new Paragraph("            DELICIAS DE VERDE CAFETERIA            R.U.C.
1709643736001", cabecera));
            document.Add(new Paragraph("                VENTA DE COMIDA Y BEBIDAS EN
CAFETERIAS PARA SU", DESC));
            document.Add(new Paragraph("                                CONSUMO INMEDIATO
FAC S 001-001- ", DESC));
            document.Add(new Paragraph("                GAON SABANDO LOURDES JESSENIA
No.: "+fac.id_factura, DESC));
            document.Add(new Paragraph("                Churoloma Alto, Geovanny Calles S/n y Derby
Aut. SRI 1116365558", DESC));
            document.Add(new Paragraph("                Telfs: 09 9842 3586 / 09 8340 1030
Fecha de aut. de Impresión: "+DateTime.Now.Day+" de "+DateTime.Now.Month+" del
"+DateTime.Now.Year, DESC));
            document.Add(new Paragraph("                Quito, Ecuador
Válida su emisión hasta: "+DateTime.Now.Day+" de "+DateTime.Now.Month+" del "+
(int.Parse((DateTime.Now.Year).ToString())+1), DESC));
            document.Add(new Paragraph("                Fecha: " + fecha.Day + "/" + fecha.Month + "/" +
fecha.Year, DESC));
            document.Add(new Paragraph( "            Cliente: " + cli.nombre + " "+ cli.apellidos + "  RUC /
CI : "+cli.cedula, cabecera));
            document.Add(new Paragraph( "            Dirección: " + cli.direccion + "  Teléfono:
"+cli.telefono, cabecera));
            document.Add(new Paragraph("              ", DESC));
            document.Add(new Paragraph("          "));
            // step 4: we create a table and add it to the document
            PdfPTable aTable = new PdfPTable(4);
            float[] widths = { 20, 50, 20, 20};
            aTable.SetTotalWidth(widths);
            // 2 rows, 2 columns
```

```
            String show = "";
            string[] cmp;
            aTable.AddCell("CANT.");
            aTable.AddCell("Descripción");
            aTable.AddCell("V. Unitario");
            aTable.AddCell("V TOTAL");
            int cont = 0;
            float suma = 0;
            float sumaD = 0;
            foreach (Producto item in fm.prods) {
               aTable.AddCell(fm.facpros[cont].cantidad.ToString());
               aTable.AddCell( item.nombre);
               aTable.AddCell( item.precio_venta.ToString());
               //aTable.AddCell( (item.precio_venta * fm.facpros[cont].cantidad).ToString());
               string ptemp = ((item.precio_venta * fm.facpros[cont].cantidad) - item.descuento / 100 *
(item.precio_venta * fm.facpros[cont].cantidad)).ToString(); ;
               double prodSub = Double.Parse(ptemp);
               show=Math.Round((prodSub - prodSub*0.12 ), 2,
MidpointRounding.AwayFromZero).ToString();
               cmp = show.Split(',');
               if(cmp[1]!=null)
                  if(cmp[1].Length==1)
                     aTable.AddCell(show+'0');
                  else
                     aTable.AddCell(show);
               else
                  aTable.AddCell(show);
               suma += float.Parse((item.precio_venta * fm.facpros[cont].cantidad).ToString());
               sumaD += float.Parse(((item.precio_venta * fm.facpros[cont].cantidad) - item.descuento / 100
* (item.precio_venta * fm.facpros[cont].cantidad)).ToString());
               cont++;
            }
            aTable.AddCell("");
            aTable.AddCell("");
            aTable.AddCell("SubTotal: ");
            show = Math.Round((sumaD - (sumaD * 0.12)), 2,
MidpointRounding.AwayFromZero).ToString();
            cmp = show.Split(',');
               if(cmp[1]!=null)
                  if(cmp[1].Length==1)
                     aTable.AddCell(show+'0');
                  else
                     aTable.AddCell(show);
               else
                  aTable.AddCell(show);

            aTable.AddCell("");
            aTable.AddCell("");
            aTable.AddCell("IVA 0%: ");
            aTable.AddCell("");

            aTable.AddCell("");
            aTable.AddCell("");
            aTable.AddCell("IVA 12%: ");
            show = Math.Round((sumaD * 0.12), 2, MidpointRounding.AwayFromZero).ToString();
            cmp = show.Split(',');
               if(cmp[1]!=null)
                  if(cmp[1].Length==1)
                     aTable.AddCell(show+'0');
                  else
                     aTable.AddCell(show);
               else
```

```
                aTable.AddCell(show);

            aTable.AddCell("");
            aTable.AddCell("");
            aTable.AddCell("Total:");
            show = Math.Round(sumaD, 2, MidpointRounding.AwayFromZero).ToString();
            cmp = show.Split(',');
            if (cmp[1] != null)
               if (cmp[1].Length == 1)
                  aTable.AddCell(show + '0');
               else
                  aTable.AddCell(show);
            else
               aTable.AddCell(show);

            document.Add(aTable);
            document.Add(new Paragraph("      ", cabecera));
            document.Add(new Paragraph("      ", cabecera));
            document.Add(new Paragraph("      ", cabecera));
            document.Add(new Paragraph("      ", cabecera));
            document.Add(new Paragraph("                  _____
_____", cabecera));
            document.Add(new Paragraph("              FIRMA AUTORIZADA
RECIBIDO", cabecera));
            this.generarXML(fm, sumaD, sumaD);
         }
         catch (DocumentException de)
         {

         }
         catch (IOException ioe)
         {

         }
         // step 5: we close the document
         document.Close();
      }
      //
      // GET: /Factura/Details/5

      public ActionResult Details(int id)
      {
         FacturaMostrar fm = this.showFactura(id);
         ViewData["path"] = Path.Combine(Server.MapPath("~/Content"), fm.fac.id_factura.ToString() );
         return View(fm);
      }

      //
      // GET: /Factura/Create

      public ActionResult Create()
      {
         List<Producto> all_prod = (from s in dbP.Producto where s.stock > 0 select s).ToList();
         ViewData["errores"] = valid.Errores;
         Facturar fac = new Facturar();
         fac.prods = all_prod;
         fac.f_c = false;
         return View(fac);
      }

      //
      // POST: /Factura/Create
```

```
[HttpPost]
public ActionResult Create(FormCollection fc)
{
    Facturar fac = new Facturar();
    try
    {
        // TODO: Add insert logic here
        if (HttpContext.Request.Form.AllKeys[1].Equals("ButtonB"))
        {

            try
            {
                valid.validate_factura(fc["clientes"]);
            }

            catch {
                valid.Errores.Add("Solo puede entrar valores numéricos");
            }
            if (!valid.isValid())
            {
                List<Producto> all_prod = (from s in dbP.Producto where s.stock > 0 select s).ToList();
                ViewData["errores"] = valid.Errores;

                fac.f_c = false;
                fac.prods = all_prod;
                return View(fac);
            }
            else
            {
                string ced = fc["clientes"];
                Cliente cli = (from s in dbC.Cliente where s.cedula.Equals(ced) select s).FirstOrDefault();
                List<Producto> all_prod = (from s in dbP.Producto where s.stock > 0 select s).ToList();
                ViewData["errores"] = valid.Errores;
                fac.f_c = true;
                fac.client = cli;
                fac.prods = all_prod;
                fac.fc = fc;
                return View(fac);
            }
        }

        if (fc["idC"] == null) {
            valid.Errores.Add("Debe buscar un cliente para facturarle");
            List<Producto> all_prod = (from s in dbP.Producto where s.stock > 0 select s).ToList();
            ViewData["errores"] = valid.Errores;
            fac.f_c = false;
            fac.prods = all_prod;
            return View(fac);
        }

        int idC1 = Convert.ToInt16(fc["idC"]);

        // tomando los productos y las cantidades a modificar
        List<ProductoStock> prod_stock = new List<ProductoStock>();
        foreach (string key in fc.AllKeys)
        {
            if (fc[key].Contains("true"))
            {
                string llave = "cantidad_" + key;
                ProductoStock aux = new ProductoStock();
                aux.id_producto = Convert.ToInt16(key);
```

```
                try
                {
                    aux.cant_stock = Convert.ToInt16(fc[llave]);
                }
                catch
                {

                    valid.Errores.Add("La cantidad de productos debe ser un valor númerico");
                }
                Producto p1 = (from s in dbP.Producto where s.id_producto == aux.id_producto select
s).FirstOrDefault();
                int stock = int.Parse((p1.stock - aux.cant_stock).ToString());
                if (stock < 0)
                    valid.Errores.Add("No puede facturar más productos de la cantidad en existencia");
                prod_stock.Add(aux);
            }
        }
        valid.validate_prod_compra(prod_stock);
        if (valid.isValid())
        {
            // adicionando la factura
            Factura fac1 = new Factura();
            fac1.fecha = DateTime.Now;
            fac1.id_cliente = idC1;
            fac1.id_empleado = System.Web.HttpContext.Current.User.Identity.Name;
            fac1.estado = "Por Cobrar";
            dbF.AddToFactura(fac1);
            dbF.SaveChanges();

            //modificando las cantidades en existencia
            dbP.ContextOptions.LazyLoadingEnabled = false;

            foreach (ProductoStock item in prod_stock)
            {
                Producto p = (from s in dbP.Producto where s.id_producto == item.id_producto select
s).FirstOrDefault();
                p.stock = p.stock - item.cant_stock;
                dbP.SaveChanges();
                Fac_Prod FP = new Fac_Prod();
                FP.id_fac = fac1.id_factura;
                FP.id_prod = item.id_producto;
                FP.cantidad = item.cant_stock;
                dbFP.AddToFac_Prod(FP);
            }
            dbFP.SaveChanges();
            this.generarPdf(this.showFactura(fac1.id_factura));
            return RedirectToAction("Index");
        }
        else
        {
            int idC = int.Parse(fc["idC"].ToString());
            List<Producto> all_prod = (from s in dbP.Producto where s.stock > 0 select s).ToList();
            Cliente cli = (from s in dbC.Cliente where s.id_cliente == idC select s).FirstOrDefault();
            fac.f_c = true;
            fac.fc = fc;
            fac.client = cli;
            fac.prods = all_prod;
            ViewData["errores"] = valid.Errores;
            return View(fac);
        }
```

```
        }
        catch
        {
            return View();
        }
    }

    //
    // GET: /Factura/Edit/5

    public ActionResult doFactura()
    {

        int idC = int.Parse(TempData["id_cliente"].ToString());

        List<Producto> all_prod = (from s in dbP.Producto where s.stock > 0 select s).ToList();
        Cliente cli = (from s in dbC.Cliente where s.id_cliente == idC select s).FirstOrDefault();
        ViewData["id_cliente"] = idC;
        ViewData["nombre_cliente"] = cli.nombre + " "+ cli.apellidos;
        ViewData["errores"] = valid.Errores;
        return View(all_prod);
    }

    //
    // POST: /Factura/Edit/5

    [HttpPost]
    public ActionResult doFactura(FormCollection fc)
    {
        int idC1 = Convert.ToInt16(fc["id_cliente"]);

        // tomando los productos y las cantidades a modificar
        List<ProductoStock> prod_stock = new List<ProductoStock>();
        foreach (string key in fc.AllKeys)
        {
            if (fc[key].Contains("true"))
            {
                string llave = "cantidad_" + key;
                ProductoStock aux = new ProductoStock();
                aux.id_producto = Convert.ToInt16(key);
                try
                {
                    aux.cant_stock = Convert.ToInt16(fc[llave]);
                }
                catch
                {

                    valid.Errores.Add("La cantidad de productos debe ser un valor númerico");
                }
                Producto p1 = (from s in dbP.Producto where s.id_producto == aux.id_producto select
s).FirstOrDefault();
                p1.stock = p1.stock - aux.cant_stock;
                if (p1.stock < 0)
                    valid.Errores.Add("No puede facturar más productos de la cantidad en existencia");
                prod_stock.Add(aux);
            }
        }
        valid.validate_prod_compra(prod_stock);
        if (valid.isValid())
        {
            // adicionando la factura
```

```
            Factura fac = new Factura();
            fac.fecha = DateTime.Now;
            fac.id_cliente = idC1;
            fac.id_empleado = System.Web.HttpContext.Current.User.Identity.Name;
            fac.estado = "Por Cobrar";
            dbF.AddToFactura(fac);
            dbF.SaveChanges();

            //modificando las cantidades en existencia
            dbP.ContextOptions.LazyLoadingEnabled = false;

            foreach (ProductoStock item in prod_stock)
            {
               Producto p = (from s in dbP.Producto where s.id_producto == item.id_producto select
s).FirstOrDefault();
               p.stock = p.stock - item.cant_stock;
               dbP.SaveChanges();
               Fac_Prod FP = new Fac_Prod();
               FP.id_fac = fac.id_factura;
               FP.id_prod = item.id_producto;
               FP.cantidad = item.cant_stock;
               dbFP.AddToFac_Prod(FP);
            }
            dbFP.SaveChanges();
            this.generarPdf(this.showFactura(fac.id_factura));
            return RedirectToAction("Index");
         }
         else
         {
            int idC = int.Parse(fc["id_cliente"].ToString());

            List<Producto> all_prod = (from s in dbP.Producto where s.stock > 0 select s).ToList();
            Cliente cli = (from s in dbC.Cliente where s.id_cliente == idC select s).FirstOrDefault();
            ViewData["id_cliente"] = idC;
            ViewData["nombre_cliente"] = cli.nombre + " " + cli.apellidos;
            ViewData["errores"] = valid.Errores;
            return View(all_prod);
         }
      }

      //
      // GET: /Factura/Delete/5

      public ActionResult Delete(int id)
      {
         return View(this.showFactura(id));
      }

      //
      // POST: /Factura/Delete/5

      [HttpPost]
      public ActionResult Delete(int id, FormCollection collection)
      {
         try
         {
            // TODO: Add delete logic here
            Factura fac = (from s in dbF.Factura where s.id_factura == id select s).FirstOrDefault();
            List<Fac_Prod> lFP = (from s in dbFP.Fac_Prod where s.id_fac == id select s).ToList();
            foreach (Fac_Prod item in lFP)
            {
```

```
                Producto p = (from s in dbP.Producto where s.id_producto == item.id_prod select
s).FirstOrDefault();
                p.stock = p.stock + item.cantidad;
                dbP.SaveChanges();

                dbFP.DeleteObject(item);
            }
            dbFP.SaveChanges();
            dbF.DeleteObject(fac);
            dbF.SaveChanges();

            return RedirectToAction("Index");
        }
        catch
        {
            return View();
        }
    }
}
```

Codigo Producto

```
{
    [Authorize(Roles = "admin, member")]
    public class ProductoController : Controller
    {
        Productos dbP = new Productos();
        Validacion valid = new Validacion();
        //
        // GET: /Producto/

        public ActionResult Index()
        {
            List<Producto> lista = (from s in dbP.Producto select s).ToList();
            return View(lista);
        }

        //
        // GET: /Producto/Details/5

        public ActionResult Details(int id)
        {
            Producto pro = (from s in dbP.Producto where s.id_producto == id select s).FirstOrDefault();
            return View();
        }

        //
        // GET: /Producto/Create

        public ActionResult Create()
        {
            ViewData["errores"] = valid.Errores;
            return View();
        }

        //
        // POST: /Producto/Create

        [HttpPost]
        public ActionResult Create(Producto pro)
        {
```

```
        try
        {
            // TODO: Add insert logic here
            valid.validate_form_producto(pro);
            if (valid.isValid())
            {
                pro.stock = 0;
                if (pro.descuento == null || pro.descuento.Equals(""))
                    pro.descuento = 0;
                dbP.AddToProducto(pro);
                dbP.SaveChanges();
                return RedirectToAction("Index");
            }
            else
            {
                ViewData["errores"] = valid.Errores;
                return View();
            }
        }
        catch
        {
            return View();
        }
    }

    //
    // GET: /Producto/Edit/5

    public ActionResult Edit(int id)
    {
        ViewData["errores"] = valid.Errores;
        Producto p = (from s in dbP.Producto where s.id_producto == id select s).FirstOrDefault();
        return View(p);
    }

    //
    // POST: /Producto/Edit/5

    [HttpPost]
    public ActionResult Edit(int id, Producto p)
    {
        try
        {
            // TODO: Add update logic here
            valid.validate_form_producto(p);
            if (valid.isValid())
            {
                dbP.ContextOptions.LazyLoadingEnabled = false;
                var prov = (from s in dbP.Producto
                            where s.id_producto == id
                            select s).FirstOrDefault();
                prov.nombre = p.nombre;
                prov.precio_comprar = p.precio_comprar;
                prov.precio_venta = p.precio_venta;
                prov.descripcion = p.descripcion;
                prov.descuento = p.descuento;
                dbP.SaveChanges();

                return RedirectToAction("Index");
            }
            else {
                ViewData["errores"] = valid.Errores;
```

```
                Producto p1 = (from s in dbP.Producto where s.id_producto == id select s).FirstOrDefault();
                return View(p1);

            }
        }
        catch
        {
            return View();
        }
    }

    //
    // GET: /Producto/Delete/5

    public ActionResult Delete(int id)
    {
        Producto p = (from s in dbP.Producto where s.id_producto == id select s).FirstOrDefault();
        return View(p);
    }

    //
    // POST: /Producto/Delete/5

    [HttpPost]
    public ActionResult Delete(int id, Producto p)
    {
        try
        {
            // TODO: Add delete logic here
            dbP.ContextOptions.LazyLoadingEnabled = false;
            var pro = (from s in dbP.Producto
                       where s.id_producto == id
                       select s).FirstOrDefault();
            dbP.DeleteObject(pro);
            dbP.SaveChanges();
            return RedirectToAction("Index");
        }
        catch
        {
            return View();
        }
    }
  }
}
```

Codigo Reporte

```
{
  [Authorize(Roles = "admin, member")]
  public class BalanceController : Controller
  {
    private Facturas dbF = new Facturas();
    private Gastos dbG = new Gastos();
```

```
private Productos dbP = new Productos();
private Compras dbC = new Compras();
private FacProds dbFP = new FacProds();
private CompProds dbCP = new CompProds();
private Clientes dbCL = new Clientes();
private Proveedores dbPR = new Proveedores();
private ProvProds dbPP = new ProvProds();
Validacion valid = new Validacion();
//
// GET: /LibroDiario/

public ActionResult Index()
{
   ViewData["errores"] = valid.Errores;
   return View();
}


[HttpPost]
public ActionResult Index(FormCollection fc)
{
   valid.validate_form_balance(fc);
   if (valid.isValid())
   {
      string fi = fc["TestDatePicker1"];
      string ff = fc["TestDatePicker2"];
      string format = "MM/dd/yyyy";
      DateTime di = DateTime.ParseExact(fi, format, CultureInfo.InvariantCulture);
      DateTime dd = DateTime.ParseExact(ff, format, CultureInfo.InvariantCulture);
      dd = dd.AddMinutes(1439);
      dd = dd.AddSeconds(59);

      List<Factura> facs1 = (from s in dbF.Factura select s).ToList();
      List<Compra> cmps1 = (from s in dbC.Compra select s).ToList();
      List<Gasto> gass1 = (from s in dbG.Gasto select s).ToList();

      List<Factura> facs = new List<Factura>();
      List<Compra> cmps = new List<Compra>();
      List<Gasto> gass = new List<Gasto>();

      foreach (Factura i in facs1)
         if (((DateTime)i.fecha) >= di && ((DateTime)i.fecha) <= dd)
            facs.Add(i);
      foreach (Compra i in cmps1)
         if (((DateTime)i.fecha) >= di && ((DateTime)i.fecha) <= dd)
            cmps.Add(i);
      foreach (Gasto i in gass1)
         if (((DateTime)i.fecha) >= di && ((DateTime)i.fecha) <= dd)
            gass.Add(i);
      // step 1: creation of a document-object
      Document document = new Document(PageSize.LETTER, 20, 20, 20, 20);

      try
      {
         // step 2:
         // we create a writer that listens to the document
         // and directs a PDF-stream to a file
         var name = DateTime.Now.ToBinary() + ".pdf";
         var path = Path.Combine(Server.MapPath("~/Content"), name );
         PdfWriter writer = PdfWriter.GetInstance(document,
                        new FileStream(path, FileMode.Create));
```

```
            // step 3: we open the document
            document.Open();
            Font arial = FontFactory.GetFont("Arial", 12, Font.BOLD);
            Font cabecera = FontFactory.GetFont("Arial", 14, Font.BOLD);
            Font DESC = FontFactory.GetFont("Arial", 9);
            DateTime fecha = DateTime.Now;
            document.Add(new Paragraph("          CAFETERIA DELICIAS DE VERDE   Fecha: " +
fecha.Day + "/" + fecha.Month + "/" + fecha.Year, cabecera));
            document.Add(new Paragraph("          REPORTE CORRESPONDIENTE AL
INTERVALO CONTABLE:", cabecera));
            document.Add(new Paragraph("          " + di.Day+"/" +di.Month+"/"+di.Year+ " al " +
dd.Day+"/" +dd.Month+"/"+dd.Year, cabecera));
            document.Add(new Paragraph("            ", DESC));
            document.Add(new Paragraph("            ", DESC));
            document.Add(new Paragraph("            ", DESC));

            // step 4: we create a table and add it to the document
            PdfPTable aTable = new PdfPTable(5);
            float[] widths = { 50, 20, 20, 20, 20 };
            aTable.SetTotalWidth(widths);
            // 2 rows, 2 columns
            PdfPCell headerG = new PdfPCell(new Phrase("REPORTE", arial));
            headerG.Colspan = 5;
            aTable.AddCell(headerG);

            //ADICIONANDO LOS INGRESOS AL BALANCE
            aTable.AddCell(new Phrase("INGRESOS", arial));
            aTable.AddCell("");
            aTable.AddCell("");
            aTable.AddCell("");
            aTable.AddCell("");

            float ingresos = 0;
            float ingresosD = 0;
            foreach (Factura fac in facs)
            {
                Cliente cli = (from s in dbCL.Cliente where s.id_cliente == fac.id_cliente select
s).FirstOrDefault();

                aTable.AddCell("#Factura: " + fac.id_factura);
                aTable.AddCell("Fecha Fac.: " + ((DateTime)fac.fecha).ToShortDateString());
                aTable.AddCell("");
                aTable.AddCell("");
                aTable.AddCell("");
                aTable.AddCell("Producto");
                aTable.AddCell("Precio V. (u)");
                aTable.AddCell("Cant.");
                aTable.AddCell("Subtotal");
                aTable.AddCell("Subtotal (Desc.)");
                float ingresosP = 0;
                float ingresosPD = 0;
                List<Fac_Prod> lFP = (from s in dbFP.Fac_Prod where s.id_fac == fac.id_factura select
s).ToList();
                foreach (Fac_Prod fp in lFP)
                {
                    Producto item = (from s in dbP.Producto where s.id_producto == fp.id_prod select
s).FirstOrDefault();
                    aTable.AddCell(item.nombre);
                    aTable.AddCell(item.precio_venta.ToString());
                    aTable.AddCell(fp.cantidad.ToString());
                    aTable.AddCell((item.precio_venta * fp.cantidad).ToString());
```

```
                    aTable.AddCell(((item.precio_venta * fp.cantidad) - item.descuento / 100 *
(item.precio_venta * fp.cantidad)).ToString());
                    ingresosP += float.Parse((item.precio_venta * fp.cantidad).ToString());
                    ingresosPD += float.Parse(((item.precio_venta * fp.cantidad) - item.descuento / 100 *
(item.precio_venta * fp.cantidad)).ToString());
                }
                aTable.AddCell("");
                aTable.AddCell("");
                aTable.AddCell("Importe:");
                aTable.AddCell(ingresosP.ToString());
                aTable.AddCell("");
                aTable.AddCell("");
                aTable.AddCell("");
                aTable.AddCell("Importe (Desc.):");
                aTable.AddCell(ingresosPD.ToString());
                aTable.AddCell("");
                ingresos += ingresosP;
                ingresosD += ingresosPD;
            }
            aTable.AddCell("");
            aTable.AddCell("");
            aTable.AddCell("Ingresos:");
            aTable.AddCell(ingresos.ToString());
            aTable.AddCell("");
            aTable.AddCell("");
            aTable.AddCell("");
            aTable.AddCell("Ingresos (Desc.):");
            aTable.AddCell("");
            aTable.AddCell(ingresosD.ToString());

            //ADICIONANDO LAS COMPRAS AL BALANCE
            aTable.AddCell(new Phrase("COMPRAS", arial));
            aTable.AddCell("");
            aTable.AddCell("");
            aTable.AddCell("");

            float compras = 0;

            foreach (Compra cmp in cmps)
            {
                int idProd = int.Parse((from s in dbCP.Comp_Prod where s.id_comp == cmp.id_compra
select s).FirstOrDefault().id_prod.ToString());
                int idProv = int.Parse((from s in dbPP.Prov_Prod where s.id_prod == idProd select
s).FirstOrDefault().id_prov.ToString());
                Proveedor prov = (from s in dbPR.Proveedor where s.id_provedor == idProv select
s).FirstOrDefault();

                aTable.AddCell("#Compra: " + cmp.id_compra);
                aTable.AddCell("Fecha Comp.: " + ((DateTime)cmp.fecha).ToShortDateString());
                aTable.AddCell("Proveedor: " + prov.nombre);
                aTable.AddCell("");
                aTable.AddCell("");
                aTable.AddCell("Producto");
                aTable.AddCell("Precio C. (u)");
                aTable.AddCell("Cant.");
                aTable.AddCell("");
                aTable.AddCell("Subtotal");
                float comprasP = 0;
                List<Comp_Prod> lCP = (from s in dbCP.Comp_Prod where s.id_comp == cmp.id_compra
select s).ToList();
                foreach (Comp_Prod cp in lCP)
```

```
                {
                    Producto item = (from s in dbP.Producto where s.id_producto == cp.id_prod select
s).FirstOrDefault();
                    aTable.AddCell(item.nombre);
                    aTable.AddCell(item.precio_comprar.ToString());
                    aTable.AddCell(cp.stock.ToString());
                    aTable.AddCell((item.precio_comprar * cp.stock).ToString());
                    aTable.AddCell(((item.precio_comprar * cp.stock) - item.descuento / 100 *
(item.precio_venta * cp.stock)).ToString());
                    comprasP += float.Parse((item.precio_comprar * cp.stock).ToString());
                }
                aTable.AddCell("");
                aTable.AddCell("");
                aTable.AddCell("Importe:");

                aTable.AddCell("");
                aTable.AddCell(comprasP.ToString());
                compras += comprasP;
            }
            aTable.AddCell("");
            aTable.AddCell("");
            aTable.AddCell("Importe Compras");
            aTable.AddCell("");
            aTable.AddCell(compras.ToString());

            //ADICIONANDO LOS GASTOS AL BALANCE
            aTable.AddCell(new Phrase("GASTOS", arial));
            aTable.AddCell("");
            aTable.AddCell("");
            aTable.AddCell("");
            aTable.AddCell("");
            aTable.AddCell("Descripcion");
            aTable.AddCell("Fecha");
            aTable.AddCell("");
            aTable.AddCell("");
            aTable.AddCell("Importe G.");

            float gastos = 0;

            foreach (Gasto gas in gass)
            {


                aTable.AddCell(gas.descripcion);
                aTable.AddCell(((DateTime)gas.fecha).ToShortDateString());
                aTable.AddCell("");
                aTable.AddCell("");
                aTable.AddCell(gas.importe.ToString());
                gastos += float.Parse(gas.importe.ToString());
            }
            aTable.AddCell("");
            aTable.AddCell("");
            aTable.AddCell("Importe Gastos");
            aTable.AddCell("");
            aTable.AddCell(gastos.ToString());


            aTable.AddCell("");
            aTable.AddCell("");
            aTable.AddCell("COMPRAS TOTALES");
            aTable.AddCell(compras.ToString());
            aTable.AddCell("");
```

```
                aTable.AddCell("");
                aTable.AddCell("");
                aTable.AddCell("GASTOS TOTALES");
                aTable.AddCell(gastos.ToString());
                aTable.AddCell("");

                aTable.AddCell("");
                aTable.AddCell("");
                aTable.AddCell("INGRESOS TOTALES");
                aTable.AddCell("");
                aTable.AddCell(ingresos.ToString());

                aTable.AddCell("");
                aTable.AddCell("");
                aTable.AddCell("INGRESOS (DESC.)");
                aTable.AddCell("");
                aTable.AddCell(ingresosD.ToString());

                aTable.AddCell("");
                aTable.AddCell("");
                aTable.AddCell("BALANCE");
                aTable.AddCell("");
                aTable.AddCell((ingresosD - compras - gastos).ToString());
                TempData["path"] = name;
                document.Add(aTable);
            }
            catch (DocumentException de)
            {

            }
            catch (IOException ioe)
            {

            }
            // step 5: we close the document
            document.Close();

            return RedirectToAction("downloadBalance");
        }
        else {
            ViewData["errores"] = valid.Errores;
            return View();
        }
    }

    public ActionResult downloadBalance()
    {
        ViewData["path"] = TempData["path"].ToString();
        return View();
    }
  }
}
```

Codigo Compra

```
{
  [Authorize(Roles = "admin, member")]
  public class CompraController : Controller
  {
    //
    // GET: /Compra/
```

```
        Compras cmpcontex = new Compras();
        Proveedores provscontext = new Proveedores();
        ProvProds provprodcontext = new ProvProds();
        Productos prodcontext = new Productos();
        CompProds dbCP = new CompProds();
        Validacion valid = new Validacion();

        public ActionResult Index()
        {
            List<Compra> comp = (from s in cmpcontex.Compra select s).ToList();
            return View(comp);
        }
        public ActionResult Create()
        {
            List<Proveedor> provs = (from s in provscontext.Proveedor select s).ToList();
            ViewData["errores"] = valid.Errores;
            return View(provs);
        }
        [HttpPost]
        public ActionResult Create(FormCollection fc_proveedor)
        {

            TempData["id_proveedor"] = fc_proveedor["proveedores"];
            ViewData["errores"] = valid.Errores;
            int id = Convert.ToInt16(fc_proveedor["proveedores"]);
            valid.validate_proveedor(id);
            if (valid.isValid())
                return RedirectToAction("doCompra");
            else
            {
                List<Proveedor> provs = (from s in provscontext.Proveedor select s).ToList();
                ViewData["errores"] = valid.Errores;
                return View(provs);
            }
        }
        public ActionResult doCompra()
        {
            string idprov = TempData["id_proveedor"].ToString();
            int finalid = Convert.ToInt16(idprov);
            List<Producto> toshow = aux_doCompra(finalid);
            string nombre = (from s in provscontext.Proveedor where s.id_provedor == finalid select
s.nombre).FirstOrDefault();
            ViewData["id_proveedor"] = finalid;
            ViewData["nombre_proveedor"] = nombre;
            ViewData["errores"] = valid.Errores;
            return View(toshow);
        }
        public List<Producto> aux_doCompra(int finalid) {
            List<Producto> toshow = new List<Producto>();
            List<Prov_Prod> provprods = (from s in provprodcontext.Prov_Prod where s.id_prov == finalid
select s).ToList();
            List<Producto> all_prod = (from s in prodcontext.Producto select s).ToList();

            foreach (Producto p in all_prod)
            {
                foreach (Prov_Prod item in provprods)
                    if (p.id_producto == item.id_prod)
                    {
                        toshow.Add(p);
                        break;
                    }
            }
```

```
            return toshow;
        }
        [HttpPost]
        public ActionResult doCompra(FormCollection fc_compra)
        {
            int id_provedor = Convert.ToInt16(fc_compra["id_proveedor"]);
            // tomando los productos y las cantidades a modificar
            List<ProductoStock> prod_stock = new List<ProductoStock>();
            foreach (string key in fc_compra.AllKeys)
            {
                if (fc_compra[key].Contains("true"))
                {
                    string llave = "precio_" + key;
                    ProductoStock aux = new ProductoStock();
                    aux.id_producto = Convert.ToInt16(key);
                    try
                    {
                        aux.cant_stock = Convert.ToInt16(fc_compra[llave]);
                    }
                    catch
                    {

                        valid.Errores.Add("La cantidad de productos debe ser un valor númerico");
                    }
                    prod_stock.Add(aux);
                }
            }
            valid.validate_prod_compra(prod_stock);
            if (valid.isValid())
            {
                // adicionando la compra
                Compra modelo_compra = new Compra();
                modelo_compra.fecha = DateTime.Now;
                cmpcontex.AddToCompra(modelo_compra);
                cmpcontex.SaveChanges();

                //modificando las cantidades en existencia
                prodcontext.ContextOptions.LazyLoadingEnabled = false;

                foreach (ProductoStock item in prod_stock)
                {
                    Producto p = (from s in prodcontext.Producto where s.id_producto == item.id_producto select
s).FirstOrDefault();
                    p.stock = p.stock + item.cant_stock;
                    prodcontext.SaveChanges();
                    Comp_Prod PC = new Comp_Prod();
                    PC.id_comp = modelo_compra.id_compra;
                    PC.id_prod = item.id_producto;
                    PC.stock = item.cant_stock;
                    dbCP.AddToComp_Prod(PC);
                }
                dbCP.SaveChanges();
                return RedirectToAction("Index");
            }
            else
            {
                TempData["id_proveedor"] = fc_compra["id_proveedor"];
                string idprov = TempData["id_proveedor"].ToString();
                int finalid = Convert.ToInt16(idprov);
                List<Producto> toshow = aux_doCompra(finalid);
                string nombre = (from s in provscontext.Proveedor where s.id_provedor == finalid select
s.nombre).FirstOrDefault();
```

```
            ViewData["id_proveedor"] = finalid;
            ViewData["nombre_proveedor"] = nombre;
            ViewData["errores"] = valid.Errores;
            return View(toshow);
        }
    }

    public MostrarCompra ShowCompra(int id)
    {
        Compra comp = (from s in cmpcontex.Compra where s.id_compra == id select s).FirstOrDefault();
        List<Comp_Prod> lCP = (from s in dbCP.Comp_Prod where s.id_comp == id select s).ToList();

        List<Producto> lP = new List<Producto>();
        for (int i = 0; i < lCP.Count; i++)
        {
            int idP = int.Parse(lCP[i].id_prod.ToString());
            Producto prod = (from s in prodcontext.Producto where s.id_producto == idP select
s).FirstOrDefault();
            lP.Add(prod);
        }
        int idPr = lP[0].id_producto;
        Prov_Prod pp = (from s in provprodcontext.Prov_Prod where s.id_prod == idPr select
s).FirstOrDefault();
        Proveedor prov = (from s in provscontext.Proveedor where s.id_provedor == pp.id_prov select
s).FirstOrDefault();

        MostrarCompra mc = new MostrarCompra();
        mc.compra = comp;
        mc.prov = prov;
        mc.prods = lP;
        mc.cmppros = lCP;
        return mc;
    }

    public ActionResult Delete(int id)
    {

        return View(this.ShowCompra(id));
    }

    [HttpPost]
    public ActionResult Delete(int id, MostrarCompra mc)
    {

        Compra comp = (from s in cmpcontex.Compra where s.id_compra == id select s).FirstOrDefault();
        List<Comp_Prod> lCP = (from s in dbCP.Comp_Prod where s.id_comp == id select s).ToList();
        foreach (Comp_Prod item in lCP)
        {
            dbCP.DeleteObject(item);
        }
        dbCP.SaveChanges();
        cmpcontex.DeleteObject(comp);
        cmpcontex.SaveChanges();
        return RedirectToAction("Index");
    }

    public ActionResult Details(int id)
    {
        return View(this.ShowCompra(id));
    }
  }
}
```

Codigo Gasto

```
{
   [Authorize(Roles = "admin, member")]
   public class GastoController : Controller
   {
      private Gastos dbG = new Gastos();
      Validacion valid = new Validacion();
      //
      // GET: /Gasto/

      public ActionResult Index()
      {
         List<Gasto> lista = (from s in dbG.Gasto select s).ToList();
         return View(lista);
      }

      //
      // GET: /Gasto/Details/5

      public ActionResult Details(int id)
      {
         Gasto g = (from s in dbG.Gasto where s.id_gasto == id select s).FirstOrDefault();
         return View(g);
      }

      //
      // GET: /Gasto/Create

      public ActionResult Create()
      {
         ViewData["errores"] = valid.Errores;
         return View();
      }

      //
      // POST: /Gasto/Create

      [HttpPost]
      public ActionResult Create(Gasto g)
      {
         try
         {
            // TODO: Add insert logic here
            valid.valid_form_gastos(g);
            if (valid.isValid())
            {
               g.fecha = DateTime.Now;
               dbG.AddToGasto(g);
               dbG.SaveChanges();
               return RedirectToAction("Index");
            }
            else {
               ViewData["errores"] = valid.Errores;
               return View();
            }
         }
         catch
         {
            return View();
         }
```

```
    }

    //
    // GET: /Gasto/Edit/5

    public ActionResult Edit(int id)
    {
        ViewData["errores"] = valid.Errores;
        Gasto g = (from s in dbG.Gasto where s.id_gasto == id select s).FirstOrDefault();
        return View(g);
    }

    //
    // POST: /Gasto/Edit/5

    [HttpPost]
    public ActionResult Edit(int id, Gasto g)
    {
        try
        {
            // TODO: Add update logic here
            valid.valid_form_gastos(g);
            if (valid.isValid())
            {
                dbG.ContextOptions.LazyLoadingEnabled = false;
                var gas = (from s in dbG.Gasto
                           where s.id_gasto == id
                           select s).FirstOrDefault();
                gas.descripcion = g.descripcion;
                gas.importe = g.importe;
                dbG.SaveChanges();
                return RedirectToAction("Index");
            }
            else {
                ViewData["errores"] = valid.Errores;
                return View(g);
            }
        }
        catch
        {
            return View();
        }
    }

    //
    // GET: /Gasto/Delete/5

    public ActionResult Delete(int id)
    {
        Gasto g = (from s in dbG.Gasto where s.id_gasto == id select s).FirstOrDefault();
        return View(g);
    }

    //
    // POST: /Gasto/Delete/5

    [HttpPost]
    public ActionResult Delete(int id, Gasto g)
    {
        try
        {
            // TODO: Add delete logic here
```

```
            dbG.ContextOptions.LazyLoadingEnabled = false;
            var gas = (from s in dbG.Gasto
                       where s.id_gasto == id
                       select s).FirstOrDefault();
            dbG.DeleteObject(gas);
            dbG.SaveChanges();
            return RedirectToAction("Index");
          }
          catch
          {
            return View();
          }
        }
    }
}
```

Código Proveedor

```
{
    [Authorize(Roles = "admin, member")]
    public class ProveedorProductosController : Controller
    {
        //
        // GET: /ProveedorProductos/
        Proveedores prvcontext = new Proveedores();
        Productos prodcontext = new Productos();
        ProvProds prodprovs = new ProvProds();
        Validacion valid = new Validacion();


        public ActionResult Index()
        {
            List<Proveedor> provs = (from s in prvcontext.Proveedor select s).ToList();
            ViewData["errores"] = valid.Errores;
            return View(provs);
        }
        [HttpPost]
        public ActionResult Index(FormCollection fc_proveedor)
        {
            try
            {

                TempData["id_proveedor"] = fc_proveedor["proveedores"];
                ViewData["errores"] = valid.Errores;
                int id = Convert.ToInt16(fc_proveedor["proveedores"]);
                valid.validate_proveedor(id);
                if (valid.isValid())
                    return RedirectToAction("selectProductos");
                else {
                    List<Proveedor> provs = (from s in prvcontext.Proveedor select s).ToList();
                    ViewData["errores"] = valid.Errores;
                    return View(provs);
                }

            }
```

```
            catch
            {

                return View();
            }
        }
        public ActionResult selectProductos() {

            ViewData["errores"] = valid.Errores;
            string idprov = TempData["id_proveedor"].ToString();
            List<Producto> prods = new List<Producto>();

            string nombre = aux_selectProductos(idprov, out prods);
            ViewData["id_proveedor"] = TempData["id_proveedor"];
            ViewData["nombre_proveedor"] = nombre;
            return View(prods);
        }
        public string aux_selectProductos(string idprov, out List<Producto> prods)
        {
            int finalid = Convert.ToInt16(idprov);

            List<Prov_Prod> productos = (from s in prodprovs.Prov_Prod where s.id_prov == finalid select
s).ToList();
            List<Producto> all_prods = (from s in prodcontext.Producto select s).ToList();
            List<Producto> prod = new List<Producto>();
            prods = prod;
            foreach (Producto p in all_prods)
            {
                bool flag = false;
                foreach (Prov_Prod item in productos)
                    if (p.id_producto == item.id_prod)
                    {
                        flag = true;
                        break;
                    }
                if (!flag)
                    prod.Add(p);
            }
            return (from s in prvcontext.Proveedor where s.id_provedor == finalid select
s.nombre).FirstOrDefault();
        }
        [HttpPost]
        public ActionResult selectProductos(FormCollection fc_productos)
        {
            Prov_Prod modelo;
            int idprove = Convert.ToInt16(fc_productos["id_proveedor"]);
            List<int> prodtoadd = new List<int>();
            foreach (string key in fc_productos.AllKeys)
                if (fc_productos[key].Contains("true"))
                    prodtoadd.Add(Convert.ToInt16(key));
            valid.validate_check_product(prodtoadd);
            if (valid.isValid())
            {
                foreach (int item in prodtoadd)
                {
                    modelo = new Prov_Prod();
                    modelo.id_prov = idprove;
                    modelo.id_prod = item;
                    prodprovs.AddToProv_Prod(modelo);
                }
                prodprovs.SaveChanges();
                TempData["id_proveedor"] = idprove;
```

```
            TempData["list_idproductos"] = prodtoadd;
            return RedirectToAction("resumenAsociacion");
        }
        else
        {
            ViewData["errores"] = valid.Errores;
            string idprov = fc_productos["id_proveedor"].ToString();
            List<Producto> prods = new List<Producto>();

            string nombre = aux_selectProductos(idprov, out prods);
            ViewData["id_proveedor"] = fc_productos["id_proveedor"];
            ViewData["nombre_proveedor"] = nombre;
            return View(prods);

        }
    }
    public ActionResult resumenAsociacion()
    {
        int idp = Convert.ToInt16(TempData["id_proveedor"]);
        string nombre = (from s in prvcontext.Proveedor where s.id_provedor == idp select
s.nombre).FirstOrDefault();
        ViewData["nombre_proveedor"] = nombre;
        List<Producto> all_prods = (from s in prodcontext.Producto select s).ToList();
        List<int> asociados = (List<int>)TempData["list_idproductos"];
        List<Producto> prodAdd = new List<Producto>();
        foreach (Producto item in all_prods)
            if (asociados.Exists(x => x == item.id_producto))
                prodAdd.Add(item);
        return View(prodAdd);
    }
  }
}
```