INSTITUTO TECNOLOGICO
"CORDILLERA"

CARRERA ANALISIS DE SISTEMAS

AUTOMATIZACIÓN DEL PROCESO DE FACTURACIÓN DE RESERVAS MEDIANTE UNA APLICACIÓN WEB DE FACTURACIÓN ELECTRÓNICA PARA EL HOTEL METROPOLIS DE LA CIUDAD DE QUITO.

Proyecto de investigación previo a la obtención del título de Tecnólogo de

Analista de Sistemas

Autora: Fuentes Yancha Mayra Zulema

Tutor: Ing. Marco Obando

Quito, Abril 2015

DECLARACIÓN DE APROBACIÓN TUTOR Y 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.

_____

MAYRA ZULEMA FUENTES YANCHA
C.C. 0803503911

CONTRATO DE CESIÒN 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. **FUENTES YANCHA MAYRA ZULEMA**, 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 "**AUTOMATIZACIÓN DEL PROCESO DE FACTURACIÓN DE RESERVAS MEDIANTE UNA APLICACIÓN WEB DE FACTURACIÓN ELECTRÓNICA PARA EL HOTEL METROPOLIS DE LA CIUDAD DE QUITO".** 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 6 días del mes de abril del dos mil quince

f)_____ f)_____

C.C. 0803503911 Instituto Superior Tecnológico Cordillera

CEDENTE **CESIONARIO**

AGRADECIMIENTO

En primer lugar a Dios por haberme guiado por un buen camino a mi querida Madre y mi Padre con mucho cariño le dedico todo mi esfuerzo y a mis hermanos por siempre haberme dado su fuerza y apoyo incondicional que me han ayudado y llevado hasta donde estoy ahora. Por ultimo a mi Tutor y Lector por haberme dedicado tiempo y paciencia en la elaboración de mi Tesis, también a mis compañeros de tesis porque en esa armonía grupal lo hemos logrado.

DEDICATORIA

Dedico esta tesis A. DIOS, y a mis queridos Padres quienes inspiraron mi espíritu para la conclusión de esta tesis., a mis profesores en especial a mi Tutor quien fue tan paciente y comprensivo sin su apoyo no hubiera concluido con mi tesis. A todos ellos se los agradezco desde el fondo de mi alma. Para todos ellos hago esta dedicatoria.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN EJECUTIVO

El siguiente proyecto de Investigación tiene como finalidad agilizar los procesos de recepción y de los servicios que se brinda en el hotel como Reservaciones es uno de los procesos, más complejo ya que se reciben las reservaciones de diferentes maneras.

Los grandes beneficiaros de la aplicación, son el personal Administrativo, clientes, ya que se disminuirá el largo tiempo de espera para realizar su respectiva reserva.

Los objetivos planteados se cumplen a cabalidad, dejando una gran satisfacción al cliente, en el apoyo de la administración de los diferentes procesos que fluyen en el hotel como son de recepción, Reservaciones y servicios, en el manejo eficiente y efectivo de la información para una adecuada toma de decisiones, y una mejor atención al cliente. Dentro de nuestro sistema se maneja ágilmente reservas mediante la web registrar fechas de entrada y salida de las reservas de habitaciones, visualizar los reportes de sus reservas, habitaciones disponibles, información de las habitaciones para que el cliente pueda elegir la que estime conveniente, además puede visualizar mediante la web su factura electrónica y con su respectivo XML.

ABSTRACT

The next research project aims to streamline the processes of reception and service that is offered at the hotel as Reservation is one of the processes more complex since reservations are received in different ways.

The major beneficiaries of the application, are the administrative staff, customers, since they decrease the long waiting for their respective reservation.

The objectives are met fully, leaving a large customer satisfaction, in support of the administration of the different processes that flow into the hotel as they are receiving, Reservation and services for the efficient and effective management of information for appropriate decisions, and better customer service. Within our system reserves nimbly handled by web log dates of entry and exit of room bookings, view reports reservations, rooms available information room for the customer to choose it deems appropriate, also can displayed by the electronic web and its respective XML invoice.

Capítulo I: Antecedentes

1.01. Contexto

El sistema WEB de facturación electrónica soporta todos los comprobantes fiscales digitales que requiera: (facturas).

Es un documento que cumple con los requisitos legales y reglamentarios exigibles para todos comprobantes de venta, garantizando la autenticidad de su origen y la integridad de su contenido.

Reducción significativa de costos mensualmente en suministros de oficina y en tiempo de distribución por documento emitido.

La factura electrónica cumple con los requisitos legales de los comprobantes tradicionales y garantiza, entre otras cosas, la autenticidad de su origen y la integridad de su contenido, lo que genera una mayor seguridad jurídica, y disminuye los riesgos de fraude y de evasión fiscal.

El presente proyecto es un Sistema Web de Reservación de Habitaciones del Hotel METROPOLIS de la Ciudad de Quito se encuentra ubicado en la Panamericana Norte KM 11 y medio calle Corazón de Jesús es un lugar tranquilo sobre todo de mucha acogida de los visitantes de las diferentes provincias por lo que la información de las personas se es muy difícil de ser manejado en forma manual, y la complejidad de información que se requiere y en el corto tiempo para una adecuada toma de decisiones por lo que es muy importante un Sitio Web que ayude a una adecuada administración del hotel.

1.02. Justificación

La solución que le ofrezco al hotel METROPILIS es muy importante porque mejorara el funcionamiento del hotel ofreciendo una mayor seguridad en sus procesos de control de reservas, registros y facturación.

El Software de Reservación de habitaciones Vía Internet logrará la satisfacción del cliente automatizando los procesos.

Con el Sitio Web se pretende agilizar los procesos de recepción y de los servicios que se brinda en el hotel como Reservaciones es uno de los procesos, más complejo ya que se reciben las reservaciones de diferentes maneras mediante la web, también podrán efectuar sus llamadas para realizar sus respectivas reservas podrán visualizar sus reservas.

1.03. Definición del Problema Central.

El problema se presenta en los procesos de reserva, registro y facturación de los huéspedes. Sucede que cuando el huésped llama al hotel y se le toman los datos personales necesarios para proceder a la reserva. Estos datos se guardan en un papel y es archivado donde se encuentran todas las reservas realizadas, pero como no se cuenta con un sistema para guardar dicha información. Otra dificultad que se presenta es en la facturación no se registra el consumo del cliente este proceso se lleva a mano y muchas veces no se sabe que es lo que el cliente ha consumido durante su hospedaje y se suele preguntar, pero no toda información dada por el cliente es cierta y por lo tanto se registra pérdidas para el hotel por falta de registro de la información.

1.03.01 Matriz T del Problema Central

Tabla 1

Matriz T del Problema Central

ANÁLISIS DE FUERZAS T					
Situación Empeorada	**Situación Actual**				**Situación Mejorada**
Pérdidas de información de datos de clientes por falta de un sistema de reservaciones.	Demora en el registro de hospedajes, reservaciones y facturación.				Correcta gestión de reservaciones y facturación.
Fuerzas Impulsadoras	**I**	**PC**	**I**	**PC**	**Fuerzas Bloqueadoras**
El financiamiento de la aplicación web satisface al alumno para su titulación.	2	4	4	2	Bajos ingresos en las reservaciones y hospedajes.
Capacitación sobre los beneficios al utilizar un sistema de reservaciones vía web.	3	4	4	2	Ignorar los beneficios de las reservaciones mediante aplicación web.
Capacidad y conocimiento de toda la aplicación web.	3	4	3	2	Desconocimiento de la aplicación web.
Crecimiento empresarial	3	4	3	2	Falta de visión empresarial.
Información confiable	2	4	4	2	Información no actualizada y por ende no confiable.
Resguardo de toda la información de huéspedes	2	3	3	2	No hay resguardo de información de huéspedes.
Actualización de programas a ser utilizados.	2	4	3	2	Programas obsoletos, disminuyendo el rendimiento de los mismos.

Nota: Determinación de las fuerzas bloqueadoras e impulsadoras.

PC. Potencial de Cambio; I = Intensidad

Capítulo II: Análisis de Involucrados

2.01 Requerimientos

El **"Hotel Metrópolis",** para realizar el estudio y el levantamiento de requerimientos corresponden con: Sistema de reservaciones y facturación electrónica.

2.01.1 Descripción del Sistema Actual

Se describen los procesos actuales que inciden directamente en el desarrollo del proyecto.

2.01.1.01 Involucrados Directos

- Recepción:
- Área administrativa:
- Cliente:
- Guardias.
- Personal de limpieza.

2.01.1.02 Involucrados indirectos

- Proveedores.
- SRI
- Turistas

2.01.2 Visión y Alcance

Este documento tiene como objetivo entregar al usuario final una visión del producto, así como los límites y alcances del proyecto. Es de suma importancia definir el alcance del software a desarrollar a modo de crear la expectativa necesaria y no contribuir a la insatisfacción del cliente.

El propósito de este documento es analizar y definir los requerimientos del Software de Aplicación Web para mejorar las necesidades del usuario final (cliente).

El software podrá ser implementado en una empresa de reservación de habitaciones, manejando principalmente los procesos de un formulario de reservas, como registrar las fechas de entradas y salidas de la reservas de habitaciones, además podrá visualizar sus reservas mediante la web, así mismo podrá ver el tipo de habitación para realizar su respectiva reserva y proceder con su facturación generando su respectivo XML.

2.01.3 Entrevistas

Tabla 2

Entrevistas

ENTREVISTA		
Identificador: Gerente		
PREGUNTAS	**OBJETIVOS**	**ANÁLISIS POSTERIOR**
¿Qué se busca mejorar con el sistema de reservación y facturación electrónica?	• Automatizar el proceso de reservaciones • Agilizar las reservaciones, mediante una aplicación web	Lo que se busca con el sistema de reservación es automatizar este proceso ya que lo realizan de forma manual.
¿Requiere de diferentes formas de pago mediante la aplicación web?	• Agilizar el proceso de reserva	Establecer diferentes usuarios con diferentes roles para poder acceder a los diferentes módulos que posee el sistema.
¿Busca mejorar el control de reservaciones?	• Llevar un adecuado control de reservaciones de cada cliente. • Contabilizar el número de reservaciones.	Llevar un adecuado control de las reservas vía web para satisfacer a los clientes.
¿Necesita reportes del incremento de huéspedes?	• Generar reportes de clientes.	Reportes de clientes Nuevos ingresados por mes.

Nota: *Las entrevistas muestra las necesidades realizadas a la empresa para el levantamiento de requerimientos.*

2.01.4 Matriz de requerimientos

Tabla 3
Matriz de requerimientos

MATRIZ DE REQUERIMIENTOS						
Identificador	**Descripción**	**Fuente**	**Prioridad**	**Tipo**	**Estado**	**Usuarios Involucrados**
REQUERIMIENTOS FUNCIONALES						
RF 001	Los clientes se registraran por primera vez al sistema mediante la web.	Gerente	Alta	Sistema	Revisión	Gerente Administrador Recepcionista Cliente
RF 002	Los clientes realizar sus reservas y buscaran las habitaciones que deseen mediante la web.	Gerente	Alta	Sistema	Revisión	Gerente Administrador Recepcionista Cliente
RF 003	Emitir Facturas Electrónica.	Gerente	Alta	Sistema	Revisión	Gerente Administrador Recepcionista
REQUERIMIENTOS NO FUNCIONALES						
NRF 001	Colores de base del sistema	Administrador	Media	Usuario	Revisión	Administrador

Nota: *La matriz muestra los requerimientos principales para la funcionalidad del sistema.*

2.01.5 Descripción detallada

Tabla 4

Detalle de Requerimientos RF 001

Los clientes se registraran por primera vez al sistema mediante la web.		**Estado**	**Implementación**
Creado por	Mayra Fuentes	**Actualizado por**	Mayra Fuentes
Fecha Creación	22-11-2014	**Fecha de Actualización**	18-01-2015
Identificador	RF 001		
Tipo de Requerimiento	Critico	**Tipo de Requerimiento**	Funcional
Datos de Entrada	Nombre de cliente y contraseña		
Descripción	Para acceder al sistema previamente deberá registrarse como usuario.		
Datos de Salida	Permisos a los diferentes módulos según el perfil de usuario.		
Resultados Esperados	Seguridad al momento de manipular el sistema.		
Origen	Administrador		
Dirigido a	Administrador, Recepcionista, Clientes.		
Prioridad	Alta		
Requerimientos Asociados	Ninguno		
ESPECIFICACIÓN			
Precondiciones	Para registrarse al sistema debe realizar sus respectivas reservas.		
Poscondiciones	En caso de olvidar su contraseña el cliente podrá recuperarla.		
Criterios de Aceptación	Dependiendo del perfil este podrá modificar, eliminar.		

Nota: se describe el registro de los clientes nuevos mediante el sistema web.

Tabla 5

Detalle de Requerimientos RF 002

Los clientes realizar sus reservas y buscaran las habitaciones que deseen mediante la web.		**Estado**	**Implementación**
Creado por	Mayra Fuentes	**Actualizado por**	Mayra Fuentes
Fecha Creación	22-11-2014	**Fecha de Actualización**	18-01-2015
Identificador	RF 002		
Tipo de Requerimiento	Critico	**Tipo de Requerimiento**	Funcional
Datos de Entrada	Nombre de la habitación.		
Descripción	Realizar la reserva mediante la web.		
Datos de Salida	Detalle de reserva valores a cancelar.		
Resultados Esperados	Incrementar clientes reducir tiempo , agilidad en la reserva de las habitaciones		
Origen	Gerente		
Dirigido a	Administrador, Recepcionista, Clientes.		
Prioridad	Alta		
Requerimientos Asociados	RF 001		
ESPECIFICACIÓN			
Precondiciones	1. Para ejecutar con el procedimiento el cliente debe estar registrado en el sistema. 2. Una vez ingresado al sistema el cliente debe realizar sus reservas y añadir sus productos a consumir.		
Poscondiciones	Una vez realizado todo el proceso de reserva procede al pago.		
Criterios de Aceptación	Permite que un cliente pueda establecer sus búsquedas de acuerdo a sus preferencias.		

Nota: Se describe las reservaciones mediante la web de acuerdo al cliente.

Tabla 6

Detalle de Requerimientos RF 003

Emitir Facturas Electrónica.		**Estado**	**Implementación**
Creado por	Mayra Fuentes	**Actualizado por**	Mayra Fuentes
Fecha Creación	22-11-2014	**Fecha de Actualización**	18-01-2015
Identificador	RF 003		
Tipo de Requerimiento	Critico	**Tipo de Requerimiento**	Funcional
Datos de Entrada	Información del cliente y habitación a reservar.		
Descripción	Se ingresa información del cliente y habitación reservada.		
Datos de Salida	Detalle de la factura.		
Resultados Esperados	Envió de factura electrónica.		
Origen	Gerente, Administrador, Recepcionista.		
Dirigido a	Recepcionista		
Prioridad	Alta		
Requerimientos Asociados	RF 001 RF 002		
ESPECIFICACIÓN			
Precondiciones	Que el cliente este de acuerdo con sus reservas caso contrario no habrá devolución de dinero.		
Poscondiciones	Una vez realizado proceda a cancelar.		
Criterios de Aceptación	Generación de la factura electrónica.		

Nota: *Se describe la generación de la factura electrónica mediante la web.*

Tabla 7

Detalle de Requerimientos NRF 001

Colores de base del sistema		**Estado**	**Implementación**
Creado por	Mayra Fuentes	**Actualizado por**	Mayra Fuentes
Fecha Creación	22-11-2014	**Fecha de Actualización**	18-01-2015
Identificador	NRF 001		
Tipo de Requerimiento	Leve	**Tipo de Requerimiento**	No Funcional
Datos de Entrada	Diseño del sistema.		
Descripción	Manejo de colores que llamen la atención del usuario.		
Datos de Salida	Visualización del sistema		
Resultados Esperados	Interfaz amigable y fácil de usar.		
Origen	Administrador		
Dirigido a	Administrador		
Prioridad	Media		
Requerimientos Asociados	RF 001, RF 002, RF003		
ESPECIFICACIÓN			
Precondiciones	Ninguno		
Poscondiciones	Ninguno		
Criterios de Aceptación	Visualización y diseño del sistema.		

Nota: *Se describe los colores de base del sistema del diseño web.*

2.02. Mapeo de Involucrados

Figura 1

Mapeo de Involucrados

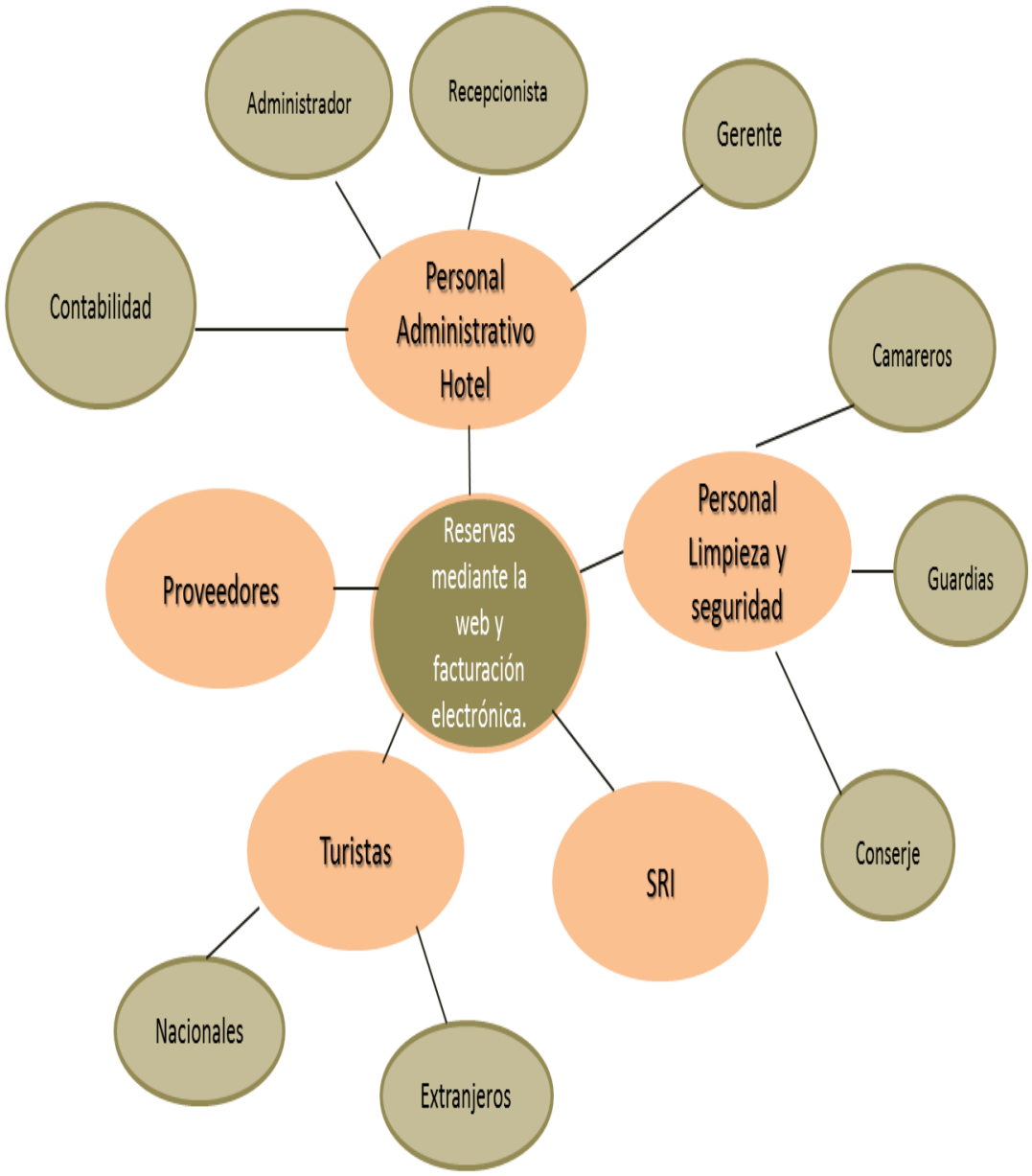


Figura 1 Mapa de involucrados: *demuestra a las personas y organismos que intervienen en el desarrollo del proyecto tanto interno como externo.*

2.03 Análisis de involucrados

Tabla 8

Análisis de involucrados

Actores Involucrados	Intereses sobre el problema central	Problemas Percibidos	Recursos, mandatos y capacidades	Intereses sobre el proyecto	Conflictos potenciales
SRI	Mejorar las declaraciones de los contribuyentes y la validación de los archivos XML.	Problema con el sistema de tributación digital y la verificación de los archivos XML.	Proponer reformas tributarias y contra con servidores públicos capacitados .	Controlar al contribuyente	Rechazo de los contribuyentes y archivos XML devueltos por estructura invalida.
Hotel	Reducción de tiempo en que se lleva a cabo los procesos de reservaciones y facturación.	Perdida de información de clientes y archivos físicos amontonados.	Diseñar estrategias y crear un presupuesto para este proyecto de reservaciones mediante la web y facturación electrónica.	Entrega oportuna de registros de los clientes.	Resistencia al cambio por parte del personal a cargo del proyecto.
Clientes o Turistas	Eliminar completamente el registro llevado a mano y el procesamiento de facturas físicas.	No contar con un sistema de reservas y facturación mediante la web.	Capacitarse continuamente sobre este sistema.	Agilidad al momento de realizar sus reservas.	Proceso de aprobación de los archivos XML y proceder enviar a los clientes y la rapidez de las reservaciones.

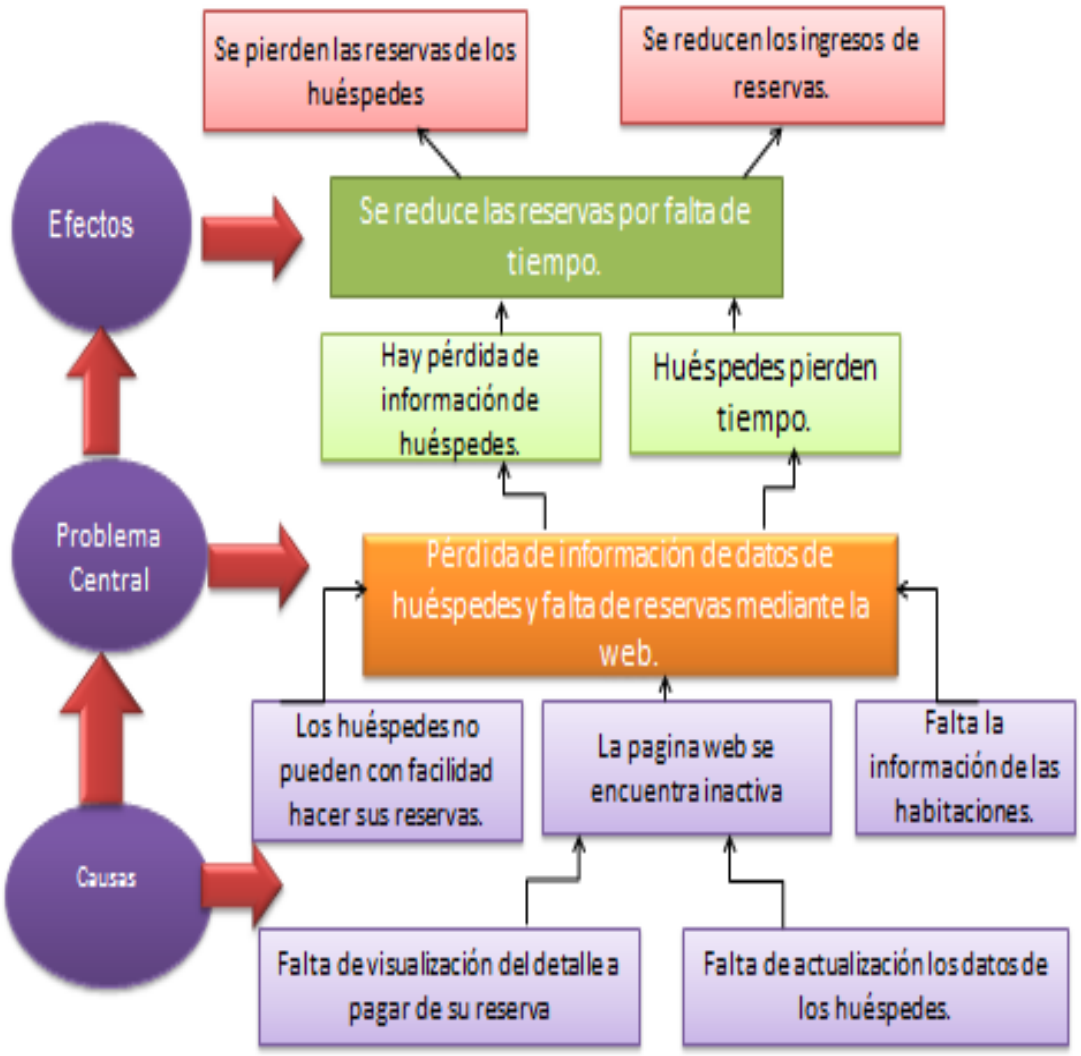
Nota: *Se describe los actores involucrados directos e indirectos.*

Capítulo III: Problemas y Objetivos

3.01. Árbol de Problemas

Figura 2

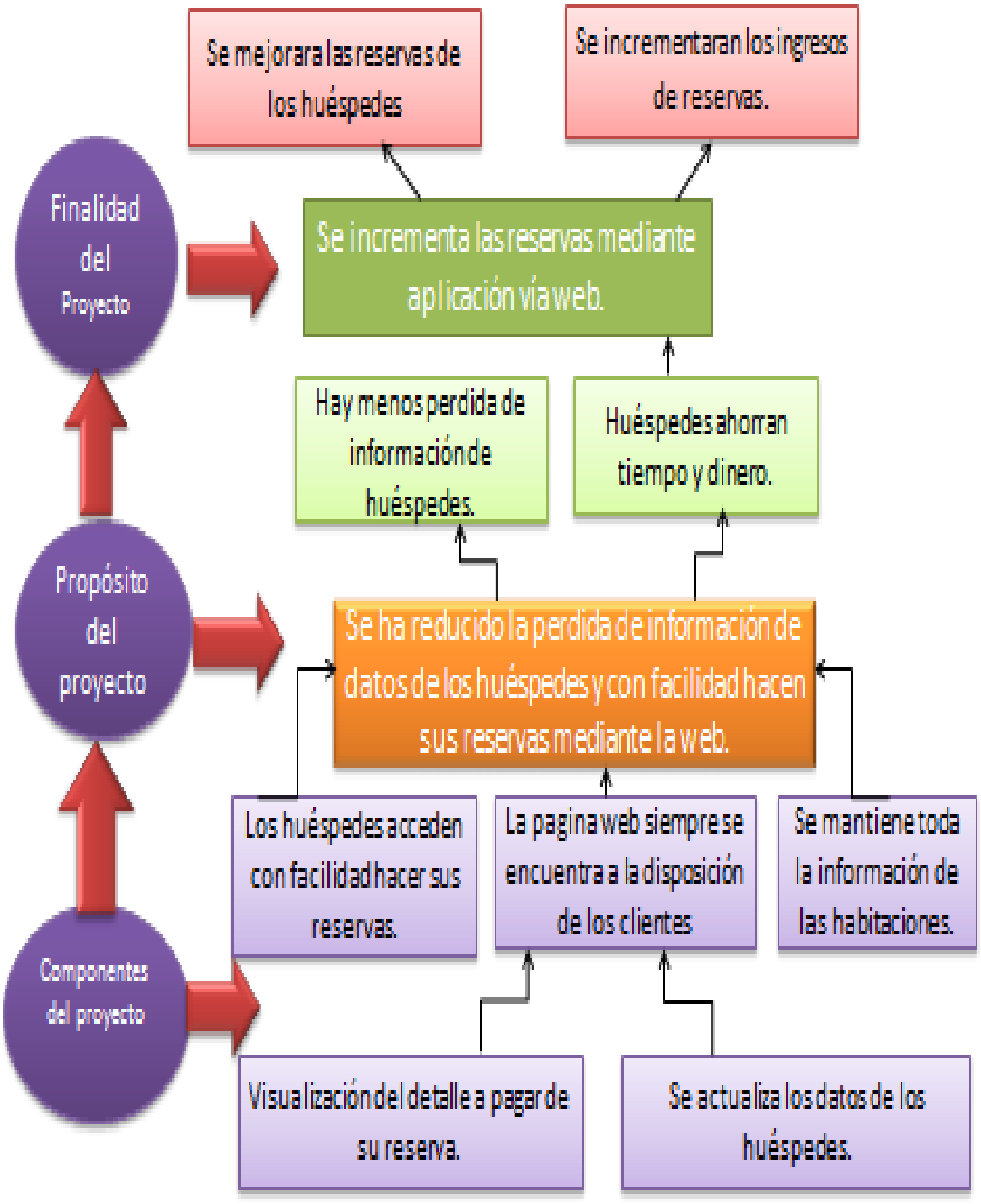
Árbol de Problemas



En esta Figura Se detallan las causas que originan el problema central que es el incorrecto proceso de reservaciones con las causas que contribuyen al inconveniente y los efectos que tiene con el HOTEL METROPOLIS

3.02. Árbol de Objetivos

Figura 3 Árbol de Objetivos



En esta figura muestra el diagrama del árbol de objetivos representa los componentes del control en el manejo de información de reservas y cliente junto con las finalidades que se desea alcanzar.

3.03. Diagramas de casos de uso

Los diagramas de casos de uso nos ayudan a representar gráficamente y describir paso a paso los procesos que se ejecutaran entre el usuario y el sistema el caos de uso están constituidos por:
Caso de Uso: Está constituido por unas figuras.

Inclusión (include o use)

Es una forma de interacción o creación, un caso de uso dado puede "incluir" otro caso de uso. El primer caso de uso a menudo depende del resultado del caso de uso incluido. Esto es útil para extraer comportamientos verdaderamente comunes desde múltiples casos de uso a una descripción individual (si el actor realiza el caso de uso base tendrá que realizar también el caso de uso incluido), desde el *caso de uso*.

Extend:

Es otra forma de interacción, un caso de uso dado (la extensión) puede *extender* a otro. Esta relación indica que el comportamiento del caso de la extensión se utiliza en casos de uso, un caso de uso a otro caso siempre debe tener extensión o inclusión. El caso de uso extensión puede ser insertado en el caso de uso extendido bajo ciertas condiciones. La notación, es una flecha de punta abierta con

línea discontinua, desde el caso de uso extensión al caso de uso extendido, con la etiqueta «extend».

Generalización:

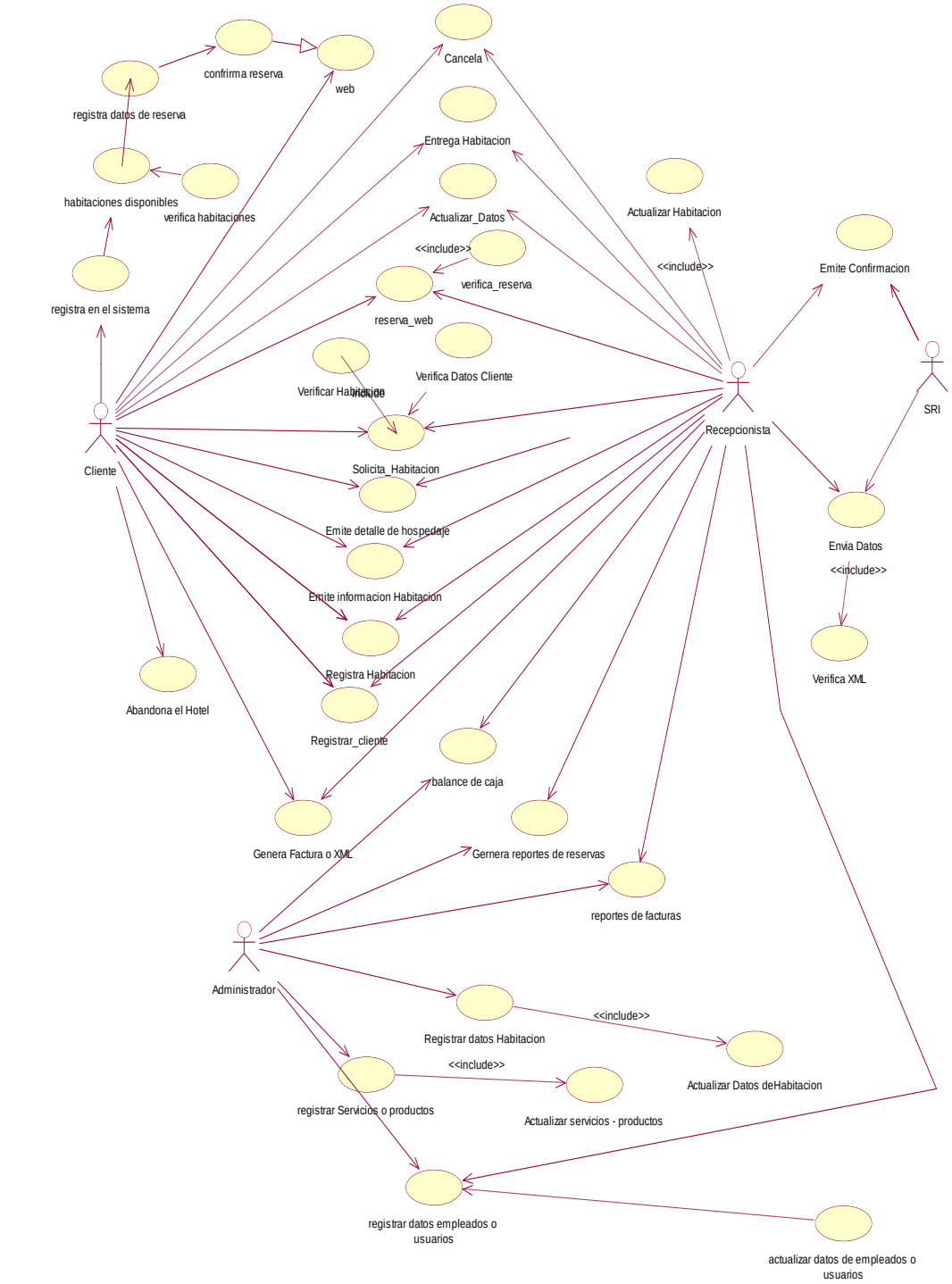
"Entonces la Generalización es la actividad de identificar elementos en común entre conceptos y definir las relaciones de una superclase (concepto general) y subclase (concepto especializado). Es una manera de construir clasificaciones taxonómicas entre conceptos que entonces se representan en jerarquías de clases. Las subclases conceptuales son conformes con las superclases conceptuales en cuanto a la intención y extensión."

Actores

Se le llama actor a toda entidad externa al sistema que guarda una relación con éste y que le demanda una funcionalidad. Esto incluye a los operadores humanos pero también incluye a todos los sistemas externos, además de entidades abstractas, como el tiempo.

En el caso de los seres humanos se pueden ver a los actores como definiciones de rol por lo que un mismo individuo puede corresponder a uno o más Actores.

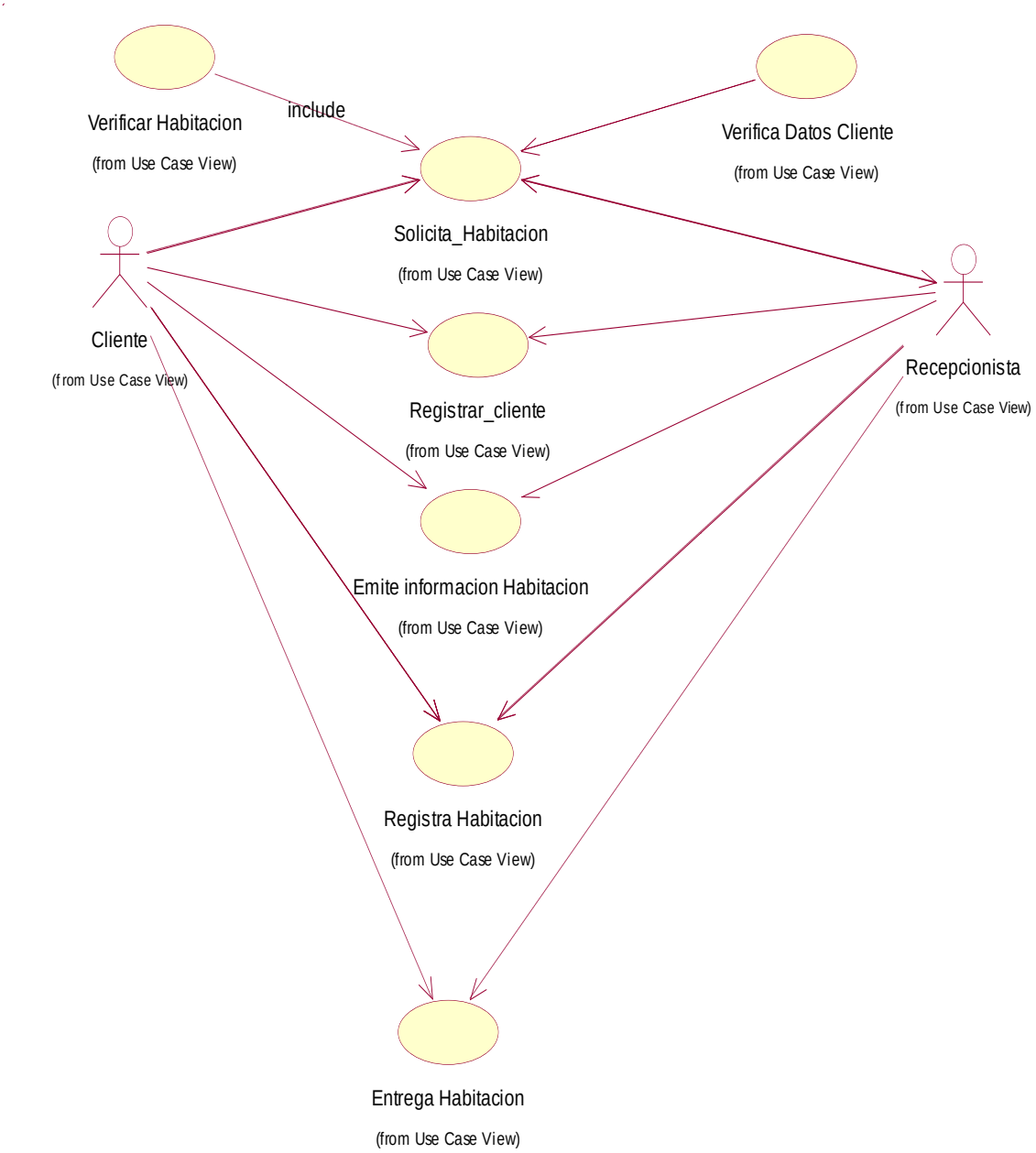
Figura 4: Caso de uso general



En esta figura hace referencia al ***Caso de uso general.*** Muestra como está estructurado el sistema de manera fácil de entender por el usuario.

Diagrama de Casos de Uso: Realizar reserva (check-in del cliente)

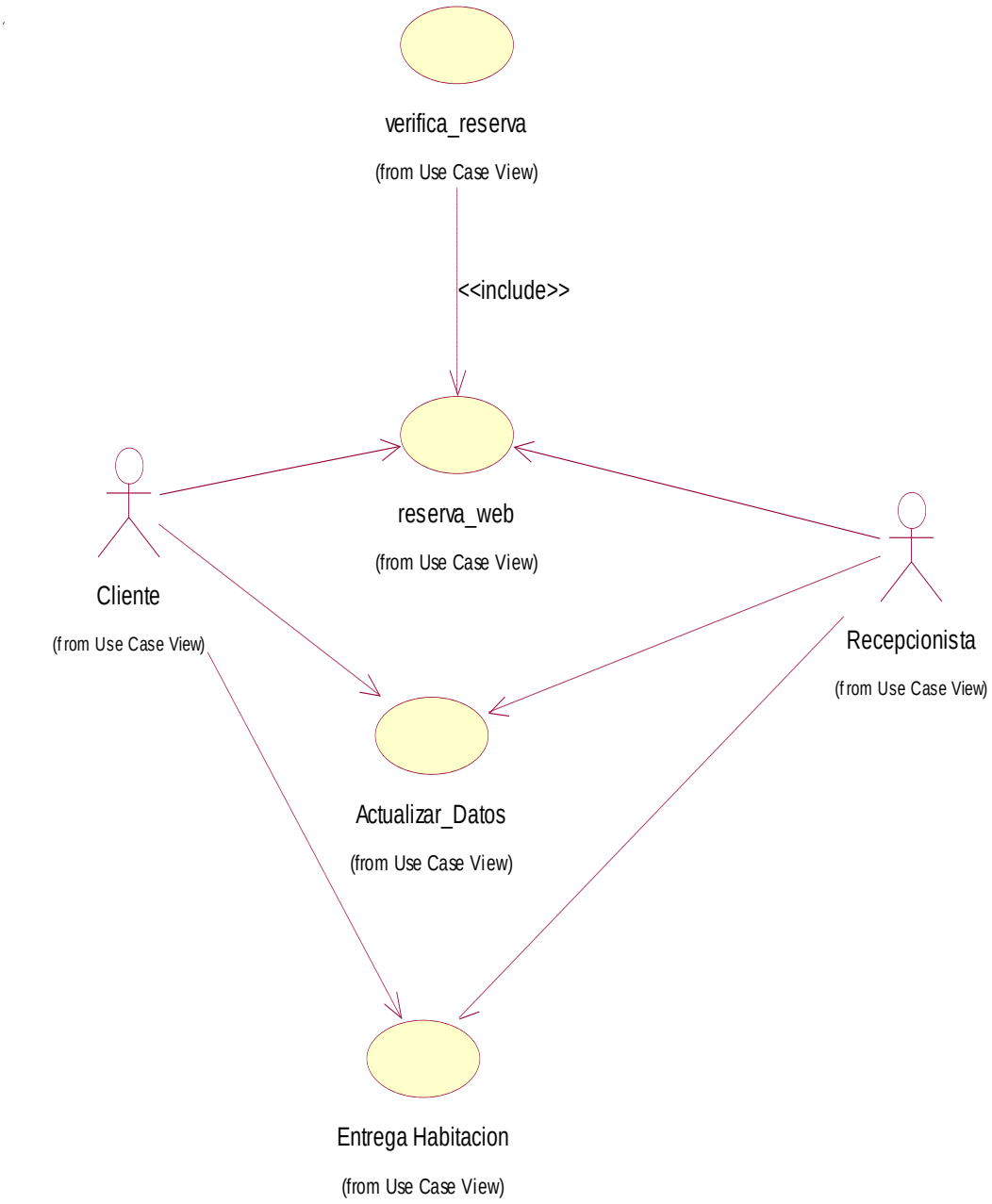
Figura 5: Casos de Uso



***CUOO1**: Esta figura muestra la solicitud de una habitación, registro de datos del cliente (check-in del cliente).*

Diagrama Caso de Uso: Verificar datos de clientes con reserva.

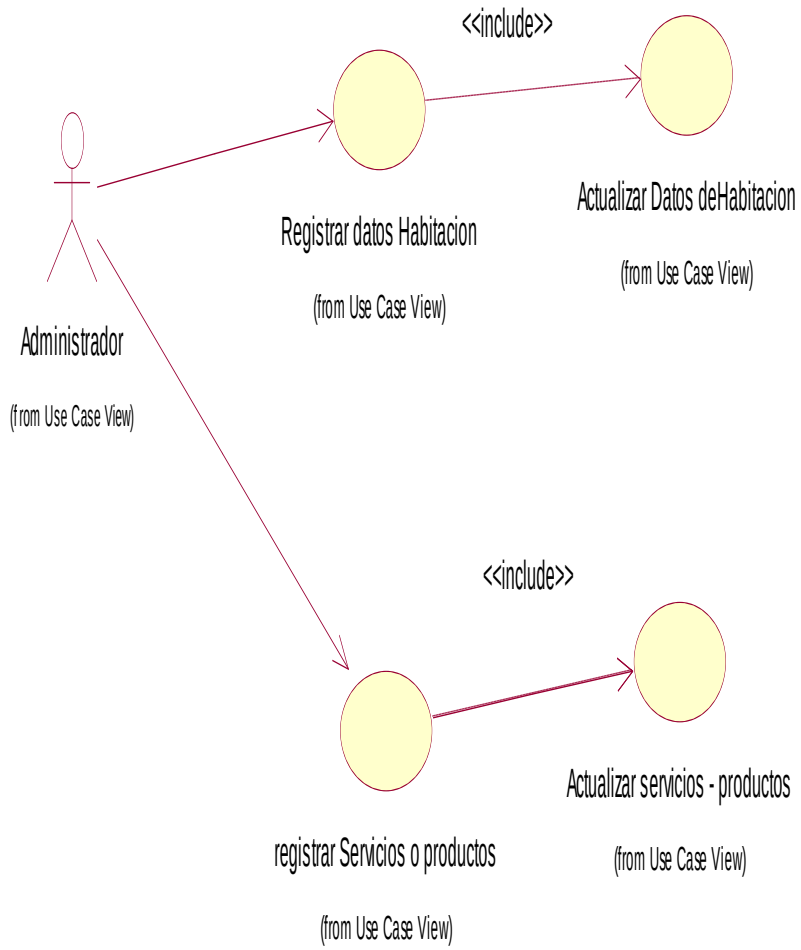
Figura 6: CU002



CU002 *En esta figura muestra la llegada de un cliente con reserva al hotel y posterior a la verificación de reserva y actualización de datos.*

Diagrama de Caso de Uso: Registrar datos de Habitación.

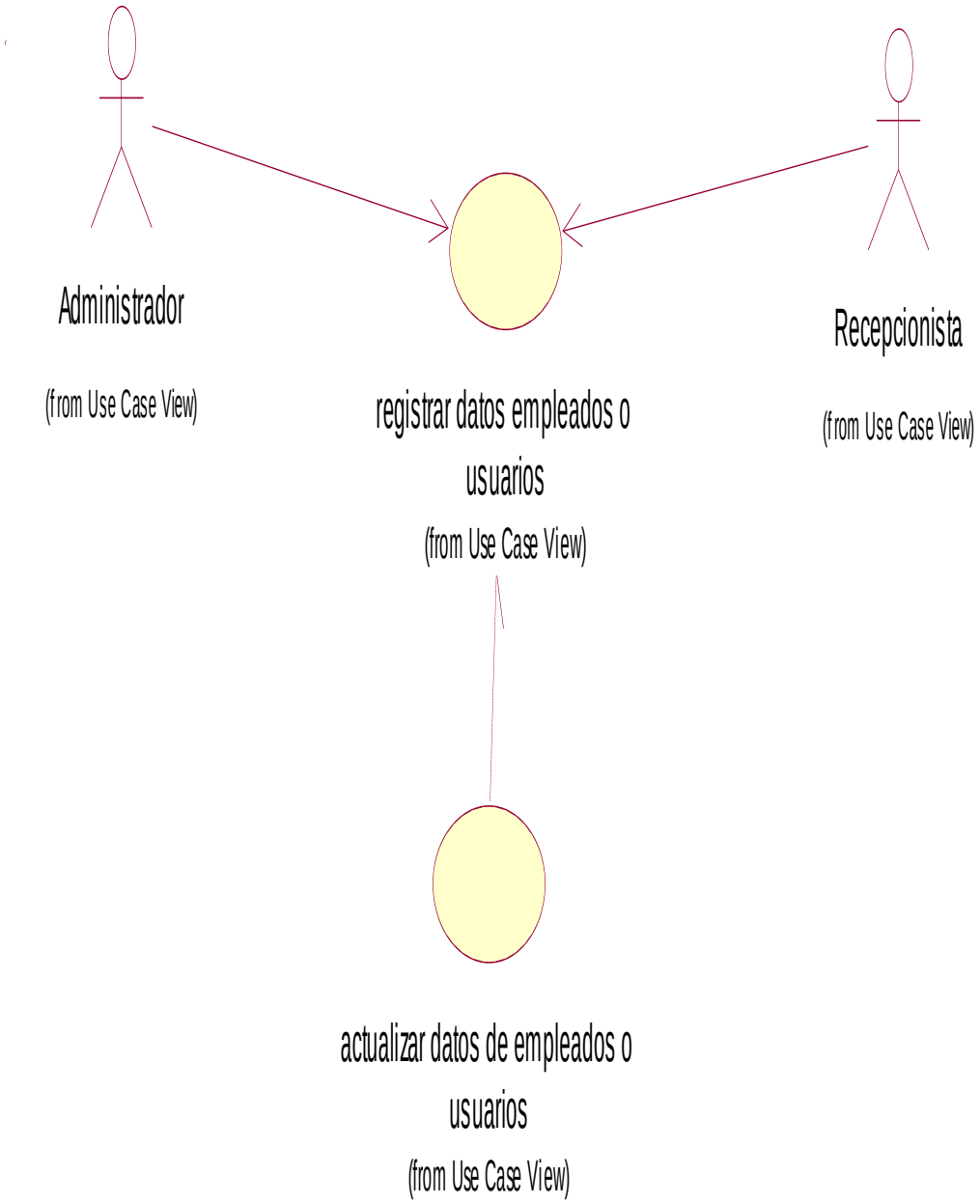
Figura 7 CU003



CU003 *En esta figura muestra registro de datos de habitación, servicios y la actualización de los mismos.*

Diagrama Caso de Uso: Registrar datos de Empleados o usuarios.

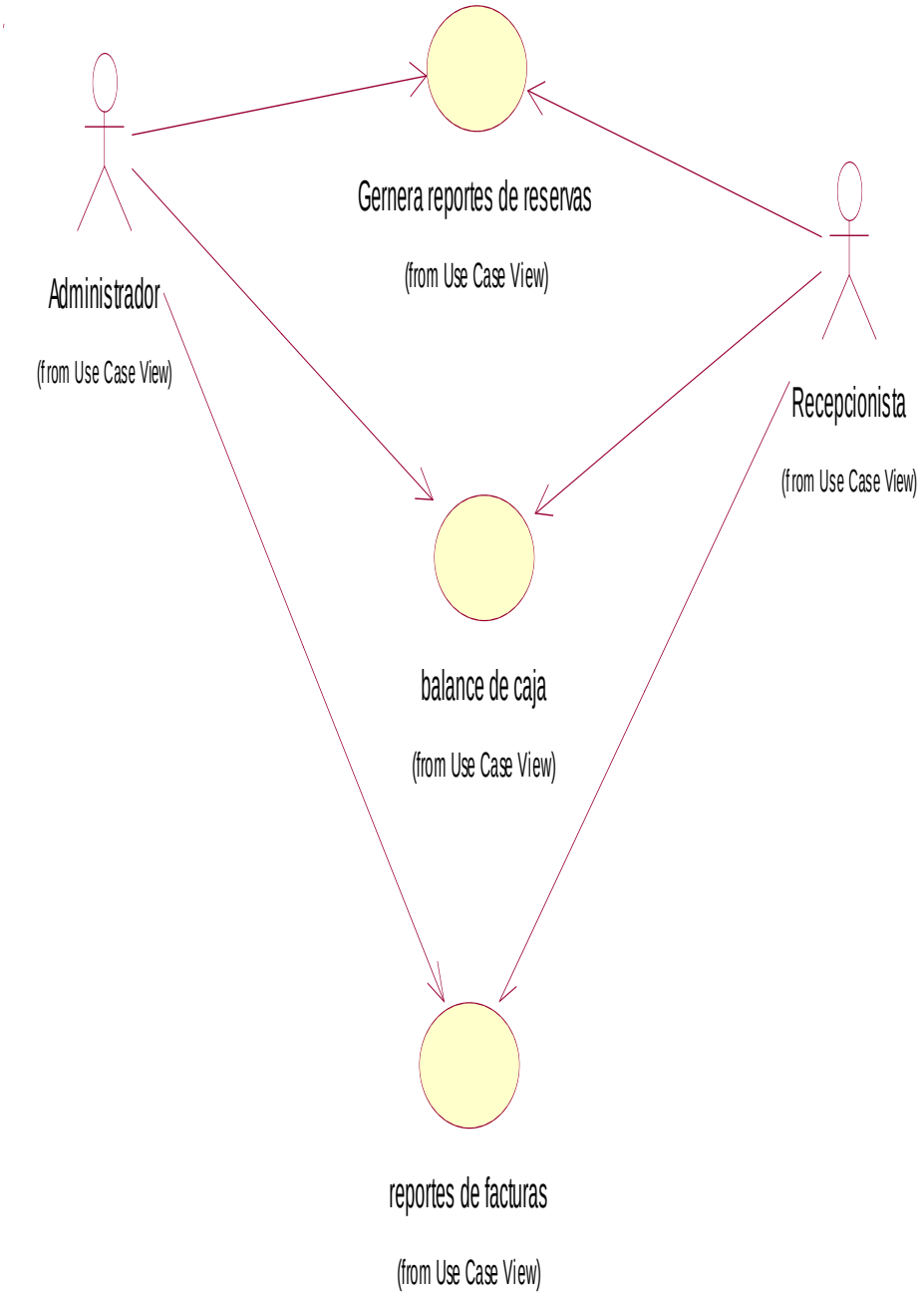
Figura 8 CU004



CU004: *En esta figura muestra el registro de empleados o usuarios y su respectiva actualización.*

Diagrama de Caso de Uso: Generar reportes de reservas y balance general.

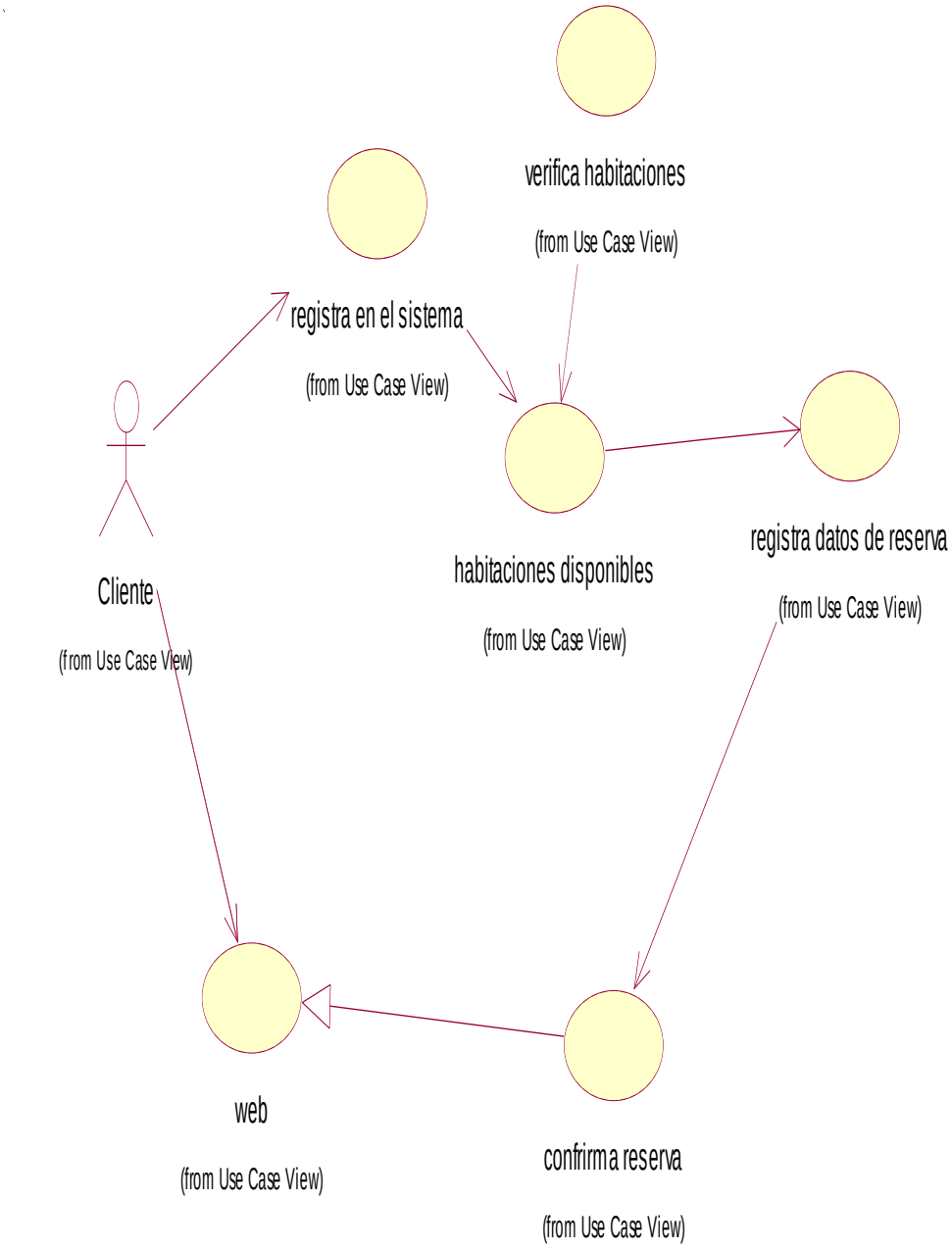
Figura 9 CU005



CU005 *En esta figura muestra los reportes de reservas y facturas.*

Diagrama de Caso de Uso: Realizar Reserva mediante la web.

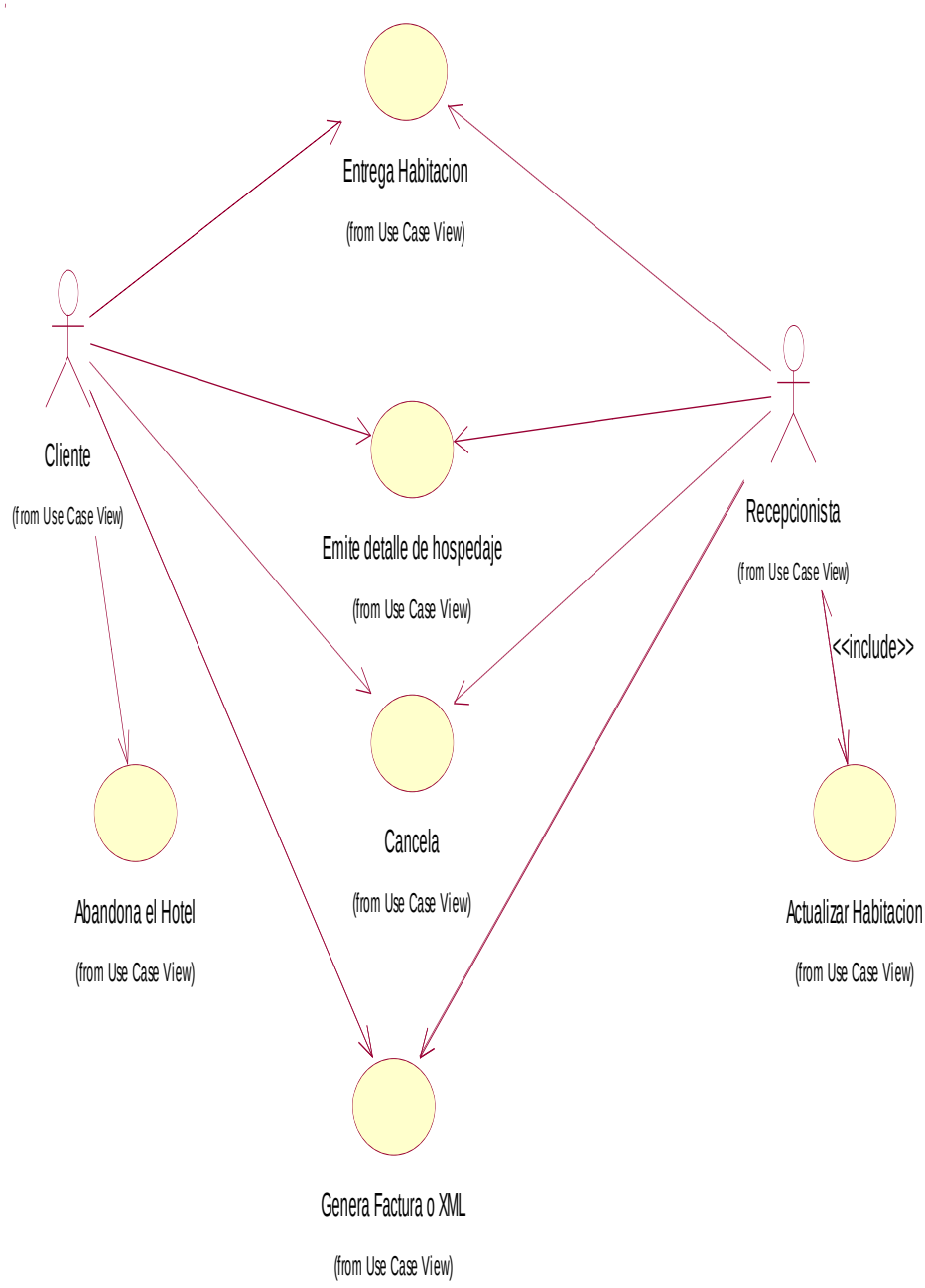
Figura 10 CU006



CU006 El cliente realiza su reserva mediante la web.

Diagrama de Caso de Uso: Entregar Habitación (Check-out)

Figura 11 CU007

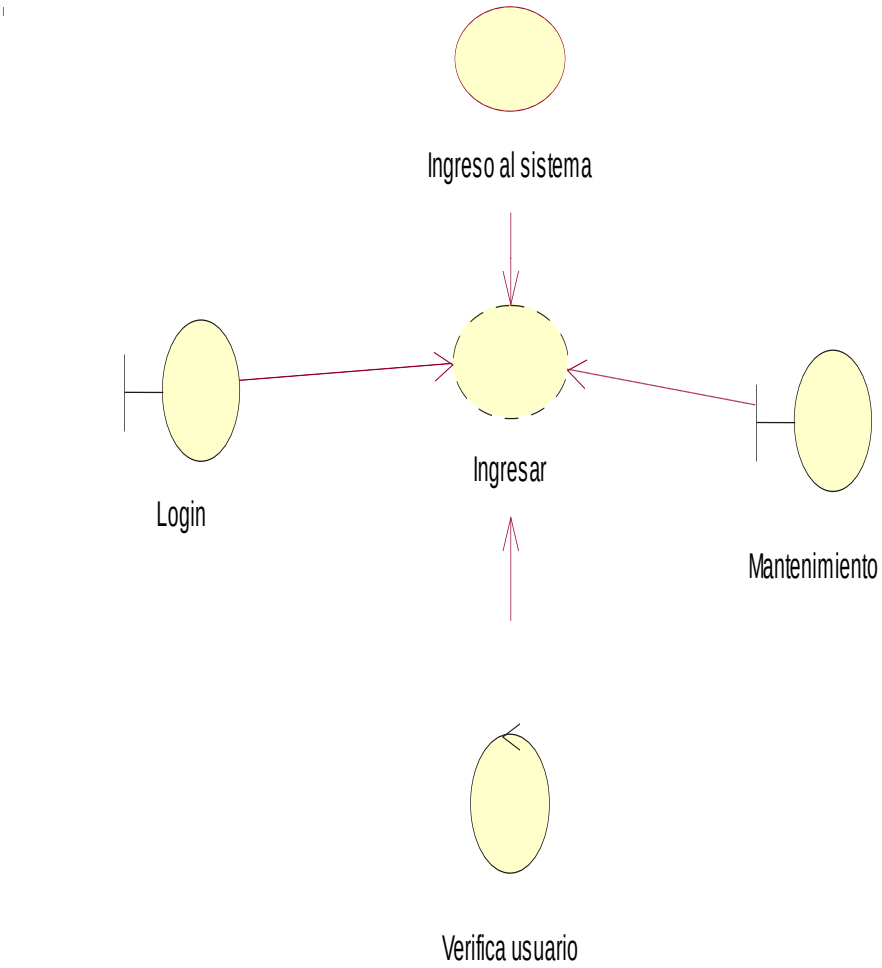


CU007 *En este diagrama muestra la entrega de habitación (check-out) y su cancelación.*

3.04. Casos de uso de realización

Para ingresar al sistema el administrador debe hacer uso del Login y el sistema verificara el nombre de usuario y contraseña y dará acceso dependiendo del tipo de usuario.

Figura 12 CU001



CU001 Ingreso al sistema.

3.04.1 Especificación de Casos de Uso de Realización

Tabla 9

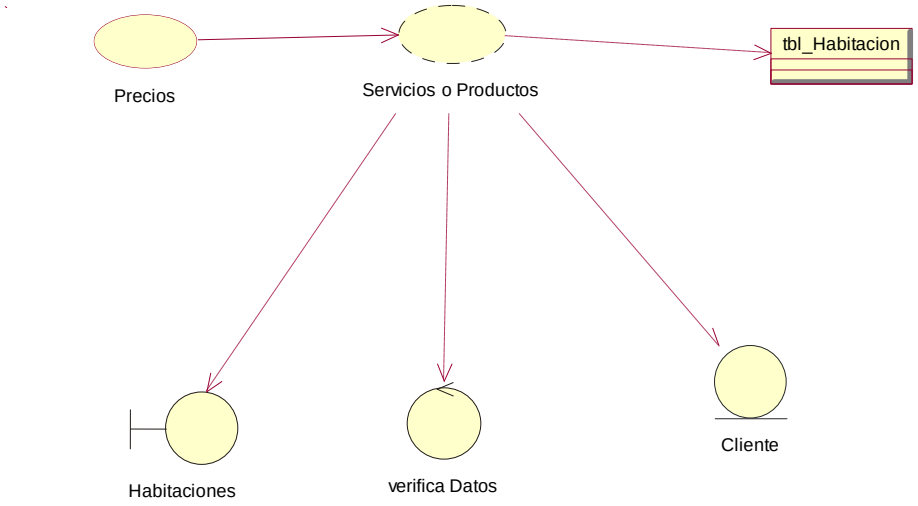
Especificación caso de uso de realización ingreso al sistema.

Nombre	**Ingreso al sistema.**
Identificador	CU001.
Responsabilidades	Permite el acceso de los usuarios a la interface que desee.
Tipo	Usuario(Cliente-Recepcionista)
Referencias Casos de Uso	Ingresar al sistema
Referencias Requisitos	Registrarse como usuario debe ingresar con nombre y clave de acceso.
PRECONDICIONES	
Estar registrados con un tipo de usuario.	
POSCONDICIONES	
Visualización de la interface que necesite del sistema.	
SALIDAS DE PANTALLA	
Interface adquirida.	

Nota: *Muestra como el usuario tiene que identificarse para ingresa al sistema y realizar lo que estime conveniente.*

El siguiente diagrama representa consulta de los precios de habitaciones y productos / servicios, en donde el cliente ingresa a la página y puede visualizar en el listado de habitaciones.

Figura 13 CU002



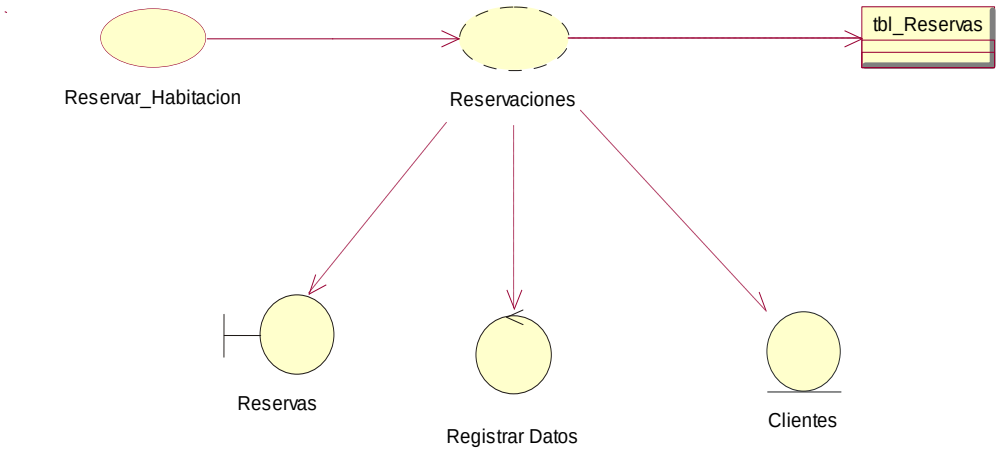
CU002. Consulta de habitaciones, productos y sus respectivos precios.

Tabla 10
Especificación caso de uso de realización consulta de habitaciones.

Nombre	**Consulta de habitaciones.**
Identificador	CU002.
Responsabilidades	Permite consultar las habitaciones, productos y precios de los mismos.
Tipo	Cliente-Recepcionista
Referencias Casos de Uso	Consultar habitaciones disponibles con sus respectivos precios.
Referencias Requisitos	Debe haber ingresado con su nombre de usuario y contraseña.
PRECONDICIONES	
Consultar sus habitaciones de acuerdo a sus necesidades.	
POSCONDICIONES	
Asignación de habitaciones	
SALIDAS DE PANTALLA	
Consulta satisfactoria.	

Nota: *Descripción de la consulta de disponibilidad de habitaciones*

Figura 14 CU003



CU003. El cliente debe realizar su reservación escoger habitación disponible e adicionar productos para su consumo y luego proceder a llenar sus datos y proceder a la confirmación de su respectiva reserva.

Tabla 11

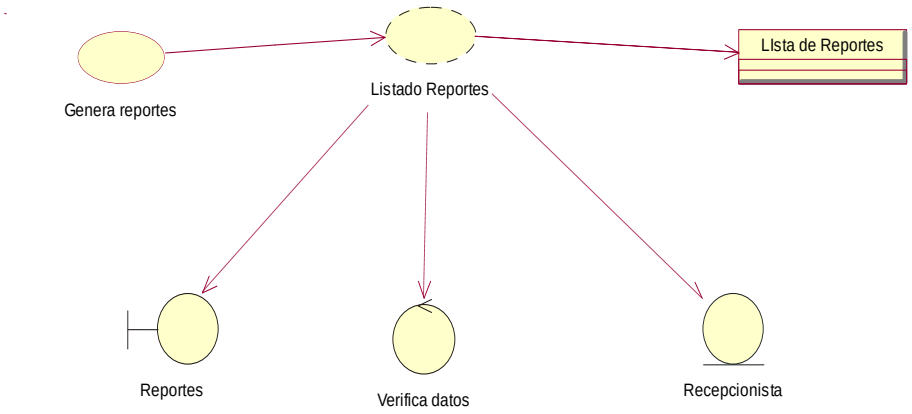
Especificación caso de uso de realización realizar su reserva.

Nombre	**Realizar reservas.**
Identificador	CU003.
Responsabilidades	Permite realizar su reserva mediante la web.
Tipo	Cliente
Referencias Casos de Uso	Realizar su respectiva reserva de acuerdo a su disponibilidad de habitación y productos a consumir.
Referencias Requisitos	Debe haberse registrado en el sistema.
PRECONDICIONES	
Luego de haber realizado su consulta de habitación el cliente procede a reservar.	
POSCONDICIONES	
Reserva mediante la web.	
SALIDAS DE PANTALLA	
Reserva en menor tiempo posible.	

***Nota**: Descripción de la realización de reservas.*

El siguiente diagrama consiste en generar reportes ya sea el administrador o la recepcionista se encarga de verificar las habitaciones genera reportes necesarios de los clientes ingresados.

Figura 15 CU004



CU004 El sistema emite reportes de clientes que han realizado sus reservas.

Tabla 12

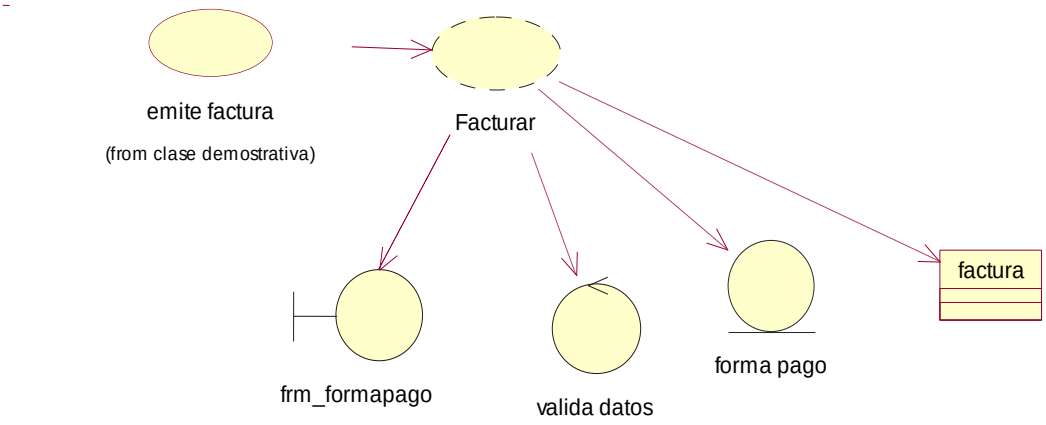
Especificación caso de uso de realización reportes de reservas

Nombre	**Reporte de reservas realizadas**
Identificador	CU004.
Responsabilidades	Permite visualizar sus respectivas reservas.
Tipo	Recepcionista-Administrador
Referencias Casos de Uso	Consultar las lista de reservas de los clientes.
Referencias Requisitos	Debe estar pendiente de la fecha de ingreso y salida.
PRECONDICIONES	
Tener reservas realizadas para visualizar el listado de reportes.	
POSCONDICIONES	
Visualización del interface que necesite para poder sacar el listado de reservaciones.	
SALIDAS DE PANTALLA	
Interface solicitada para el listado de reservas.	

Nota: *Muestra dentro del sistema el listado de reservaciones.*

El siguiente diagrama representa el proceso de facturación de la reserva y Productos adicionados.

Figura 16 CU005



CU005 *En este diagrama se muestra como está estructurado el proceso para obtener la factura.*

Tabla 12

Especificación caso de uso de realización de facturación

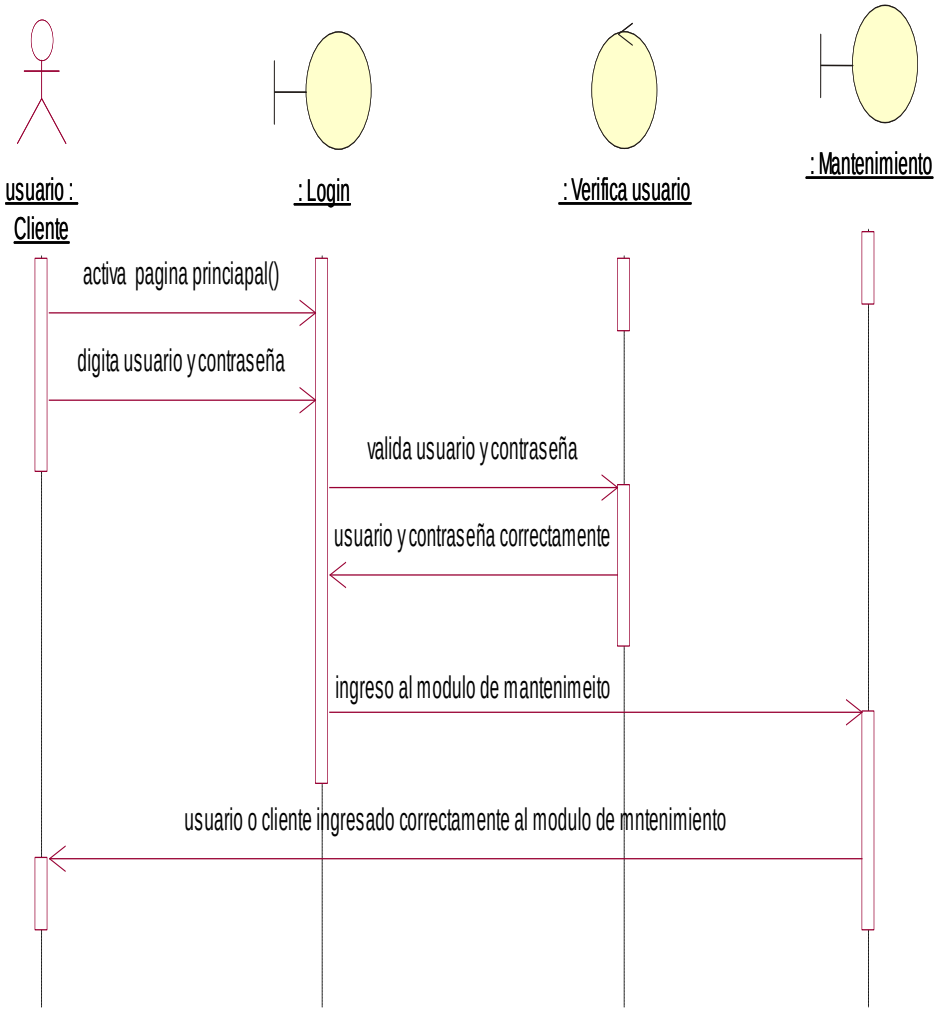
Nombre	**Factura**
Identificador	CU005.
Responsabilidades	Permite a los usuarios realizar una factura.
Tipo	Recepcionista-Administrador
Referencias Casos de Uso	Realizar factura y cobrarla.
Referencias Requisitos	Tener acceso a realizar y cobrar una factura. Tener en cuenta facturas caducadas y el cobro de la misma.
PRECONDICIONES	
Tener registro de reservas y productos.	
POSCONDICIONES	
Visualización de la interface que necesite el sistema.	
SALIDAS DE PANTALLA	
Interface solicitada para facturar.	

Nota: Muestra dentro del sistema como se realiza el proceso de facturación.

3.05. Diagrama de secuencias del sistema.

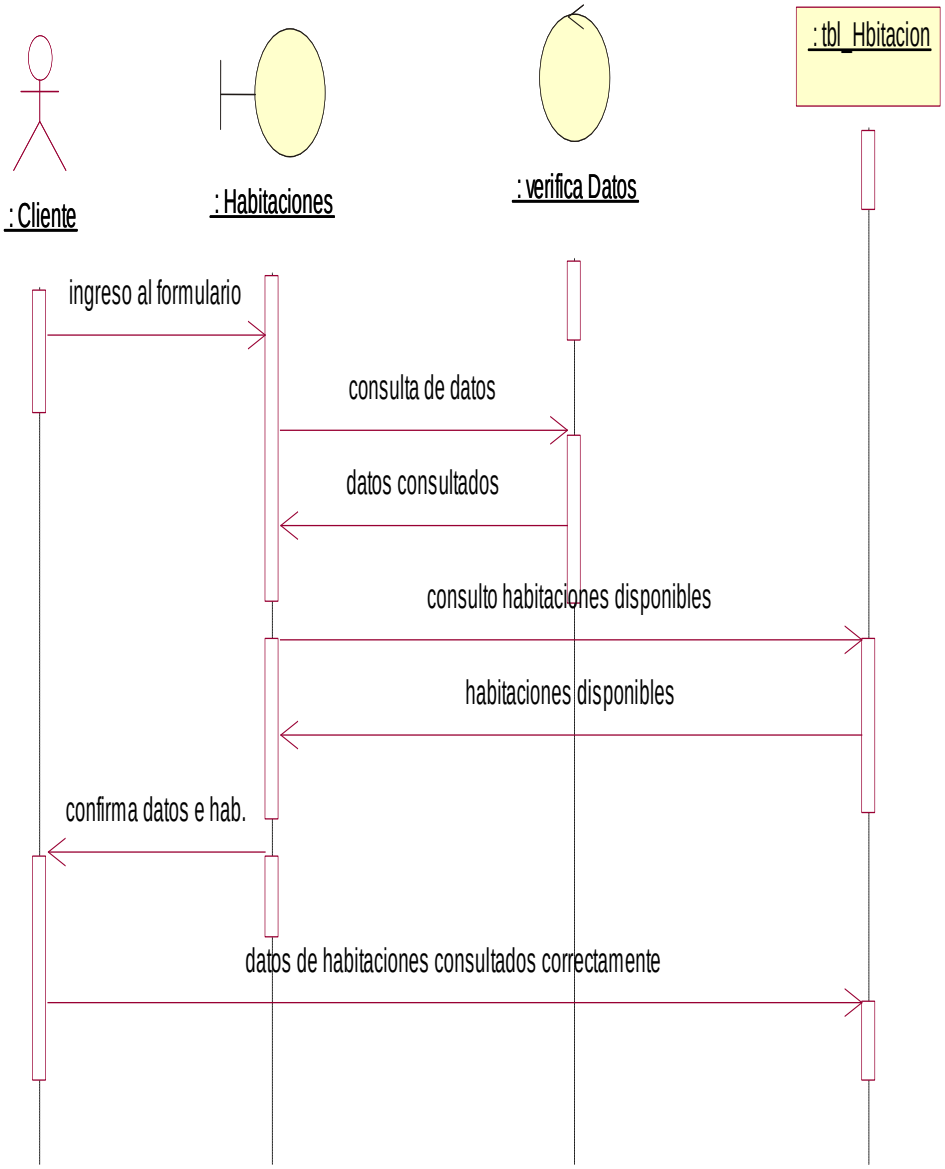
Para ingresar al sistema el usuario debe ingresar a la página principal, y luego dirigirse al login, donde se debe autentificarse con su nombre de usuario y contraseña, los cuales serán verificados si se encuentra registrado.

Figura 17 CU001



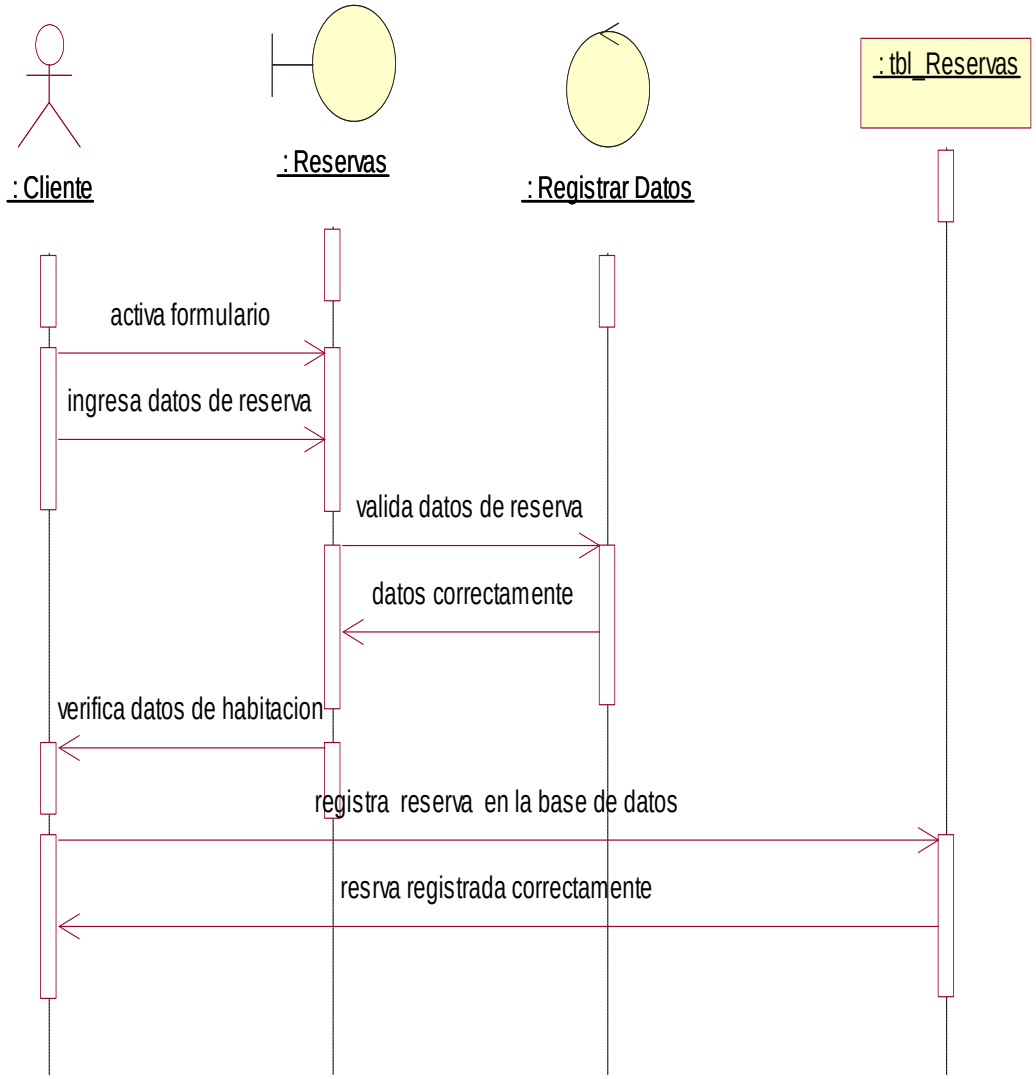
CU001. Ingreso al sistema

Figura 18 CU002



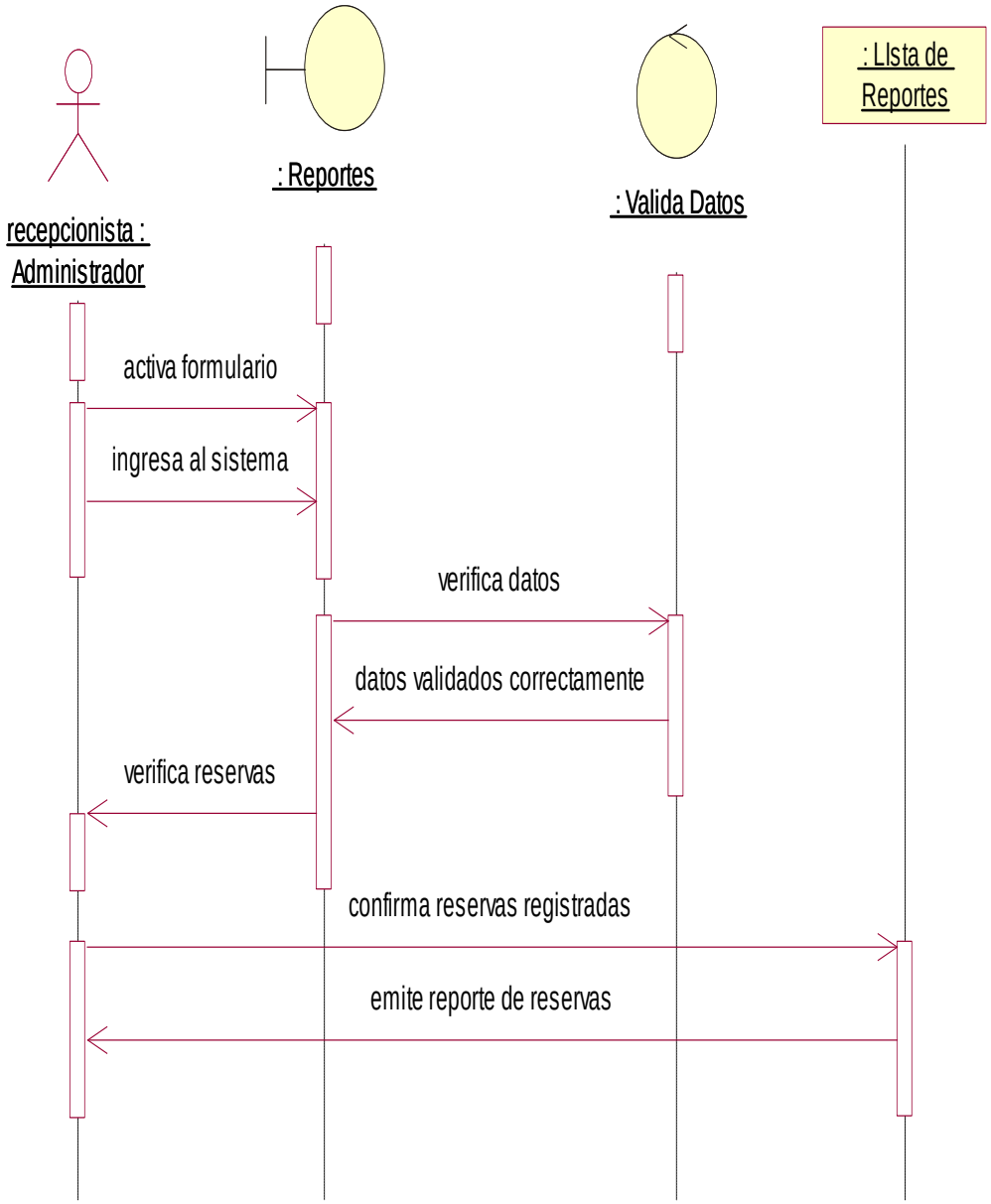
CU002. *Este diagrama representa consulta de habitaciones, precios en donde el cliente ingresa a la página y puede visualizar en el listado de habitaciones su disponibilidad.*

Figura 19 CU003



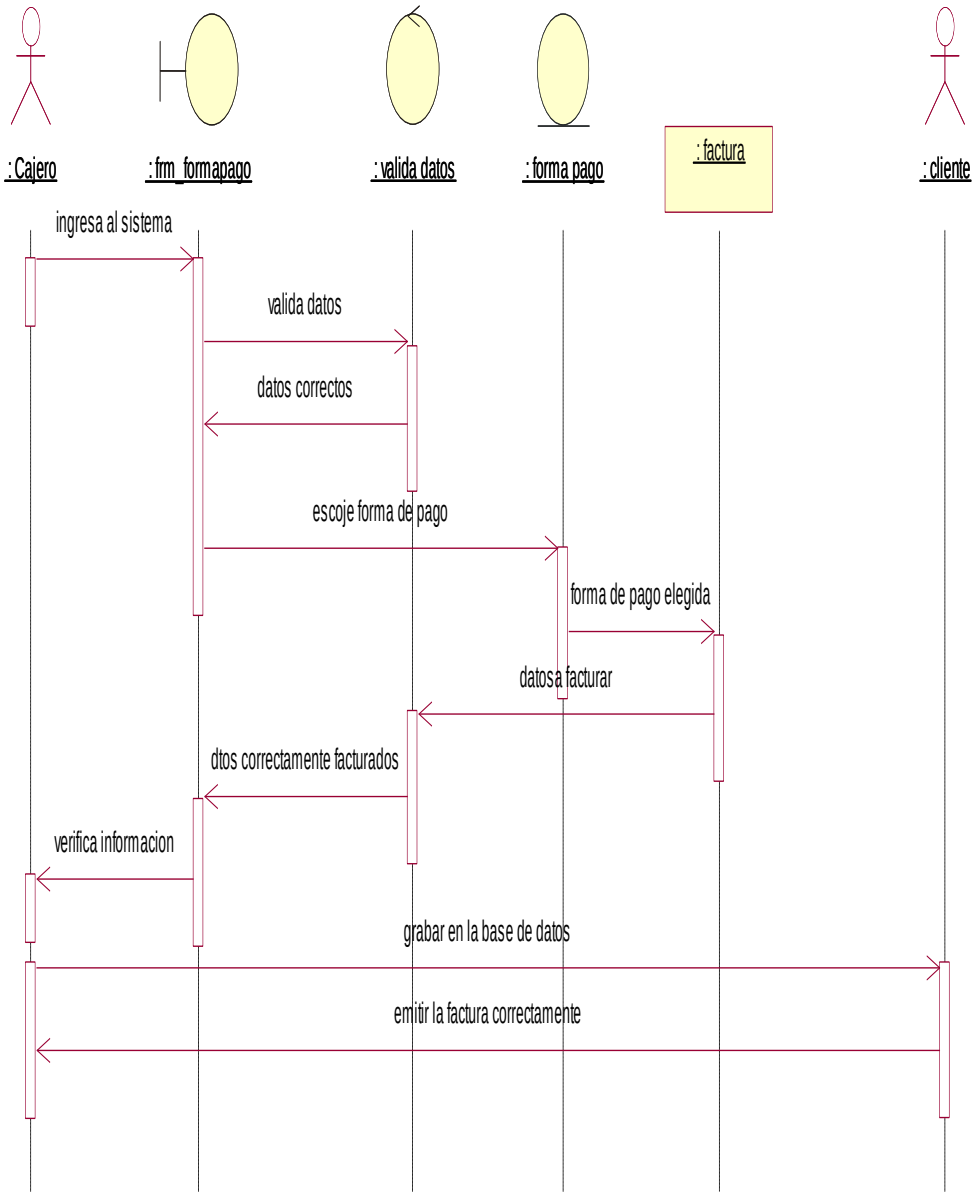
CU003. *Ingreso de datos de cliente y su respectiva reserva.*

Figura 20 CU004



CU004 Reportes de reservas.

Figura 21 CU005



CU005. Caso de realización donde la recepcionista cobra la reservación y emite su factura.

3.06. Especificación de casos de uso

Tabla 13

Especificación de casos de uso

Caso de Uso	**Realizar reserva(check-in del cliente)**
Identificador	UC001
CURSO TIPICO DE EVENTOS	
Usuario	**Sistema**
1. El caso de uso se activa cuando el cliente solicita una habitación.	**Si el cliente se encuentra ya registrado se procede actualizar datos.**
2. El cliente podrá elegir las habitaciones y servicios que estén disponibles.	**El sistema una vez consultado la disponibilidad de habitaciones procede a confirmar la reservación.**
CURSOS ALTERNATIVOS	

Tabla 14

Verificar datos de clientes con reserva.

Caso de Uso	**Verificar datos de clientes con reserva.**
Identificador	**UC002**
CURSO TIPICO DE EVENTOS	
Usuario	**Sistema**
1. El caso de uso se activa cuando la recepcionista verifique la reserva.	**Si tiene una reserva se procederá con la actualización de datos y entrega de habitación.**
CURSOS ALTERNATIVOS	

Tabla 15

Registrar datos de Habitación.

Caso de Uso	**Registrar datos de Habitación.**
Identificador	**UC003**
CURSO TIPICO DE EVENTOS	
Usuario	**Sistema**
1. El caso de uso se activa el administrador registre los datos de las habitaciones y servicios.	**Si las habitaciones ya están registradas se procede actualizarlas.**
CURSOS ALTERNATIVOS	

Tabla 16

Registrar datos de Empleados o usuarios.

Caso de Uso	**Registrar datos de Empleados o usuarios.**
Identificador	**UC004**
CURSO TIPICO DE EVENTOS	
Usuario	**Sistema**
1. **El caso de uso se activa cuando el administrador registre los datos** de los empleados o usuarios. 2. El caso de uso se activa para la recepcionista cuando el administrador autorice el ingreso de usuarios.	**Si los empleados ya están registrados se procede actualizarlas.** **Empleados que ya no laboren en el hotel se procede a eliminarlos.**
CURSOS ALTERNATIVOS	

Tabla 17

Generar reportes de reservas y balance general.

Caso de Uso	**Generar reportes de reservas y balance general.**
Identificador	**UC005**
CURSO TIPICO DE EVENTOS	
Usuario	**Sistema**
1. **El caso de uso se activa cuando el administrador** o la recepcionista genere reportes de reservas, facturas y balance de caja.	**El sistema permitirá visualizar el listado de reservas.** **Permitirá generar balance de caja y visualizarlo en PDF.**
CURSOS ALTERNATIVOS	

Tabla 18

Realizar Reserva mediante la web.

Caso de Uso	**Realizar Reserva mediante la web.**
Identificador	**UC006**
CURSO TIPICO DE EVENTOS	
Usuario	**Sistema**
1. **El caso de uso se activa cuando** el cliente se registre en el sistema.	**El sistema permitirá visualizar habitaciones disponibles con sus respectivos precios.**
2. El cliente registra sus datos una vez que haya habitaciones disponibles y procede a confirmar la reserva.	**El sistema guardara los datos del cliente y su respectiva reserva.**
CURSOS ALTERNATIVOS	

Tabla 19

Entregar Habitación (Check-out)

Caso de Uso	**Entregar Habitación (Check-out)**
Identificador	**UC007**
CURSO TIPICO DE EVENTOS	
Usuario	**Sistema**
1. El caso de uso se activa el cliente proceda hacer la entrega de la habitación.	**El sistema permitirá visualizar el detalle de sus consumos durante el hospedaje.**
2. El cliente cancelara su factura o será enviada a su correo o como estime conveniente se procederá con el check-out, el cliente abandonará el hotel.	**Se genera el XML y posterior será enviado a su correo con su respectiva validación.** **El sistema actualizara la habitación para su disponibilidad.**
CURSOS ALTERNATIVOS	

Tabla 20

Especificación de casos de uso de realización

Nombre	Registrar
Identificador	**UCR001**
Responsabilidades	**Realizar el registro del cliente**
Tipo	**Sistema**
Referencias Casos de uso	**UC001**
Referencia a requisitos	**RF001**
RECOMENDACIONES	
Necesita una identidad para ir guardando la información.	
POSCONDICIONES	
Validar las formas de pago	
SALIDAS PANTALLA	
Factura	

Tabla 21

Registrar habitación

Nombre	Registrar habitación
Identificador	**UCR002**
Responsabilidades	**Realizar el registro de la habitación**
Tipo	**Sistema**
Referencias Casos de uso	**UC002**
Referencia a requisitos	**RF002**
RECOMENDACIONES	
De instancia necesita una habitación disponible para asignar al huésped. POSCONDICIONES	Se
SALIDAS PANTALLA	

Tabla 22

Entregar Factura

Nombre	Entregar Factura
Identificador	**UCR003**
Responsabilidades	**Validar datos del cliente**
Tipo	**Sistema**
Referencias Casos de uso	**UC003**
Referencia a requisitos	**RF003**
RECOMENDACIONES	
De instancia se necesita validar los datos del cliente para emitir la factura correspondiente.	

Capítulo IV: Análisis de Alternativas

4.01. Matriz de Análisis de Alternativas

Tabla 23

Matriz de Análisis de Alternativas

Matriz de Análisis de alternativas							
Objetivos	Impacto sobre el propósito	Factibilidad Técnica	Factibilidad Financiera	Factibilidad Social	Factibilidad Política	Total	Categorías
Clientes aplicando conocimientos sobre reservas vía on-line	4	4	4	2	4	18	Alta
Usuarios han organizado y están manejando la web	3	3	2	2	2	12	Media Alta
Personal ha sido capacitado para el manejo de la aplicación	4	3	2	2	4	15	Media Alta
Se han mejorado el proceso de reservas	2	2	2	2	2	10	Media Baja
Clientes reservando habitaciones mediante vía web.	3	2	3	4	2	14	Media Alta
Total	15	14	13	12	14	69	

Nota: Muestra las alternativas y factibilidades para determinar su viabilidad.

4.02. Matriz de Impactos de Objetivos

Tabla 24

Matriz de Impactos de Objetivos

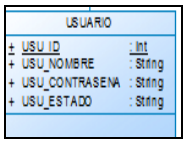
	Factibilidad de Lograse (Alta.Media-Baja) (4 - 2 - 1)	Impacto en Género (Alta–Media–Baja) (4 - 2 - 1)	Impacto Ambiental (Alta-Media-Baja) (4 - 2 - 1)	Relevancia (Alta-Media-Baja) (4 - 2 - 1)	Sostenibilidad (Alta-Media-Baja) (4 - 2 - 1)	Total
Objetivos	• Los beneficios son mayores que los costos • Cuenta con financiamiento • Es aceptable y conveniente para los beneficiarios • Existe tecnología adecuada para su realización.	1.Incrementa la reservación de huéspedes. 2 Incrementa los ingresos de las reservas 3 Incrementa el proceso de reservas y facturación. • Fortalece la aplicación de las reservas mediante la web.	• Contribuye a proteger el entorno físico. • Mejora el entorno social. • Mejora el entorno cultural. • Protege el uso de los recursos. • Favorece la tecnología.	• Responde a las expectativas de los beneficiarios • Es una prioridad sentida por los beneficiarios • Beneficia a los usuarios y disminuye el tiempo. • Los beneficios son deseados por los beneficiarios.	• Fortalece la incrementación de las reservas para huéspedes. • Fortalece la Organización local • La población está en posibilidades de aportar medios • Se puede conseguir financiamiento a futuro	88 puntos 22 a 32 BAJA 33 a 44 MEDIA BAJA 45 a 66 MEDIA ALTA 67 a 88 ALTA
Incrementada las reservas mediante la web	20 puntos	16 puntos	20 puntos	16 puntos	16 puntos	

Nota: Muestra las alternativas y factibilidades para determinar su viabilidad.

4.03. Estándares para el Diseño de Clases

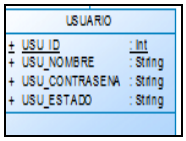
Clase

Define los atributos y los métodos de una serie de objetos. La clase implementa una o más interfaces



Atributos

Son características de una clase pueden ser de tres tipos: public,private,protected



Dependencia

Es una relación entre dos elementos, tal que un cambio en uno puede afectar al otro.

Atributo protected

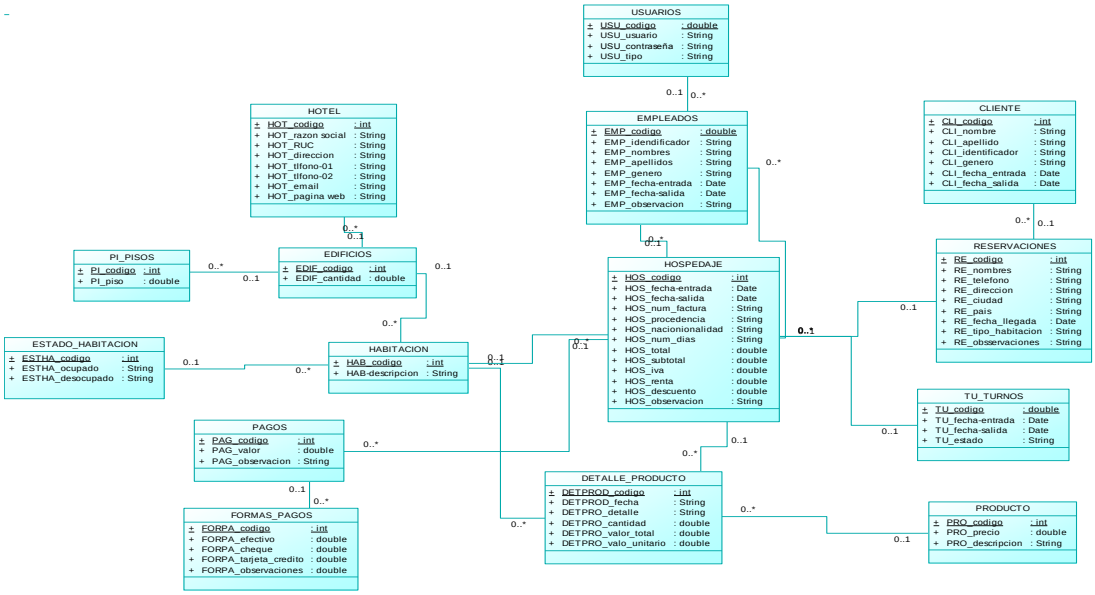
Indica que el atributo no será accesible desde la fuera de la clase, pero si podrá ser accedido por métodos de la clase además de las subclases que se deriven.

Asociaciòn

Es una relación estructural que resume un conjunto de enlaces que son conexiones entre objetos.

4.04. Diagrama de clases

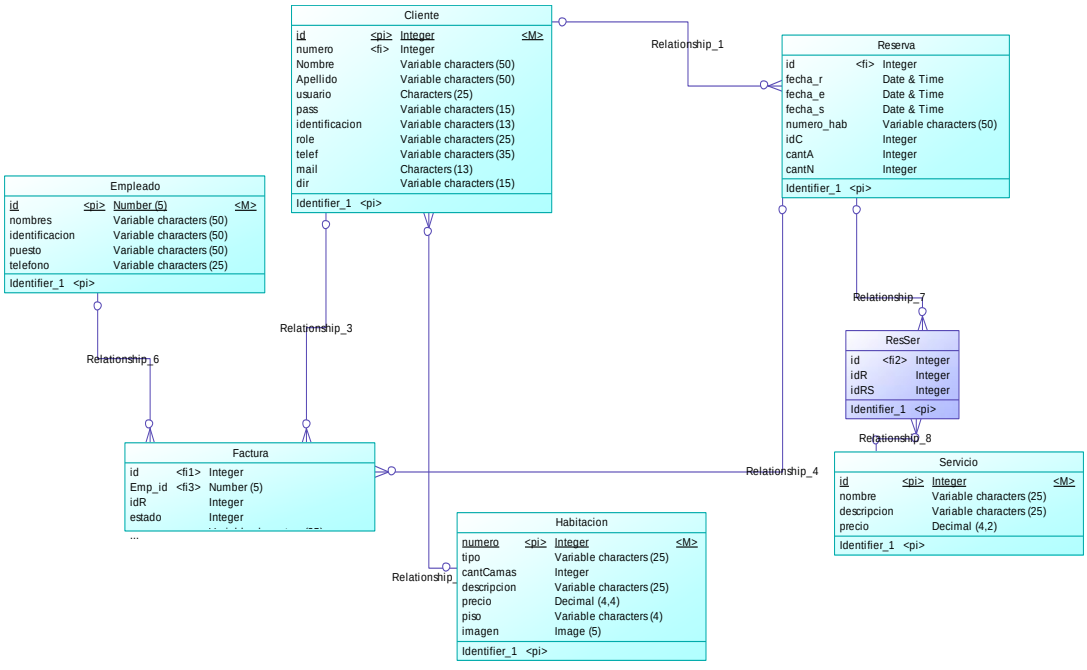
Figura 22 Diagrama de clases



En esta figura muestra el modelo de clases

4.05. Modelo Lógico - Físico

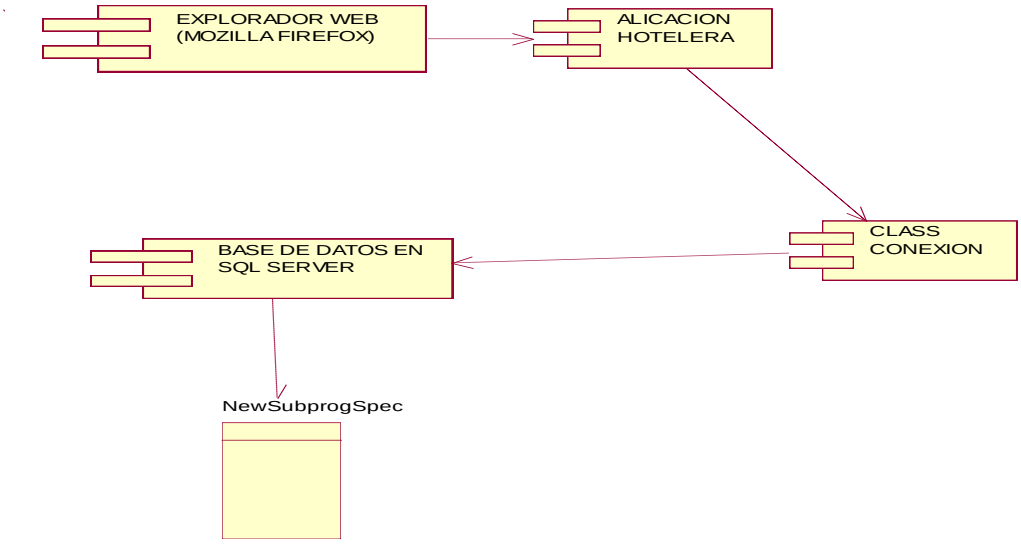
Figura 23 Modelo Lógico – Físico



En esta figura muestra del diagrama Lógico - Físico de la base de datos.

4.06. Diagrama de Componentes

Figura 24 Diagrama de Componentes



En este diagrama representa como el sistema está dividido en componentes

4.07. Diagramas de Estrategias

Figura 25 Diagramas de Estrategias

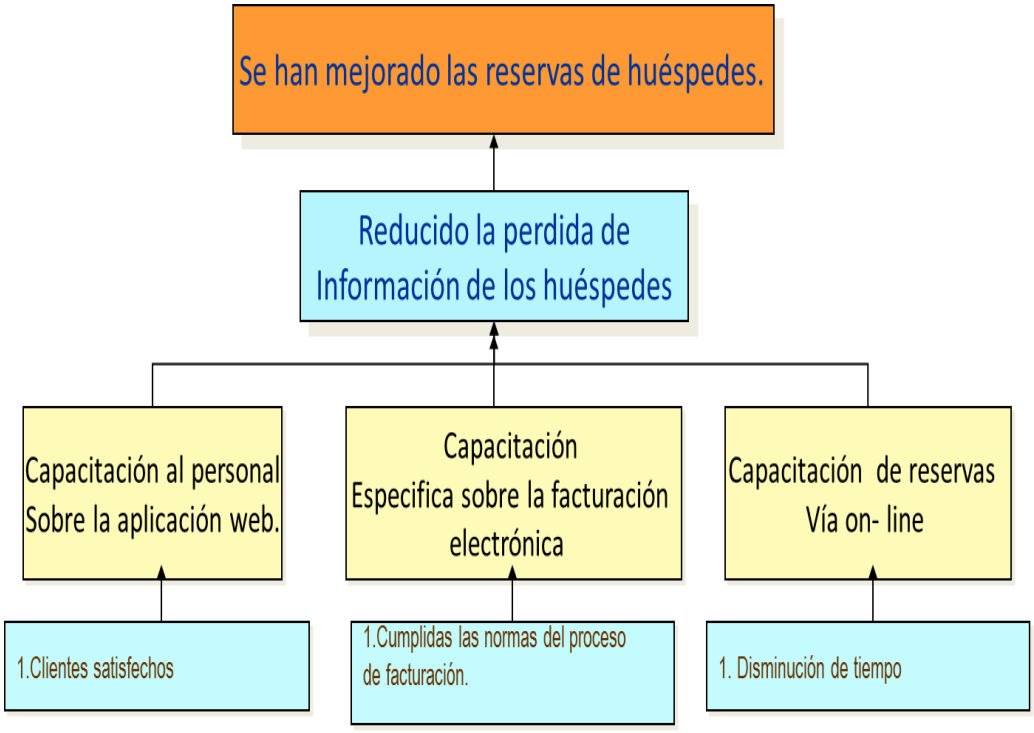


Diagrama de estrategias se detalla cada una de las estrategias y la forma que se va realizar.

4.08. Matriz de Marco Lógico.

Figura 26

Matriz de Marco Lógico

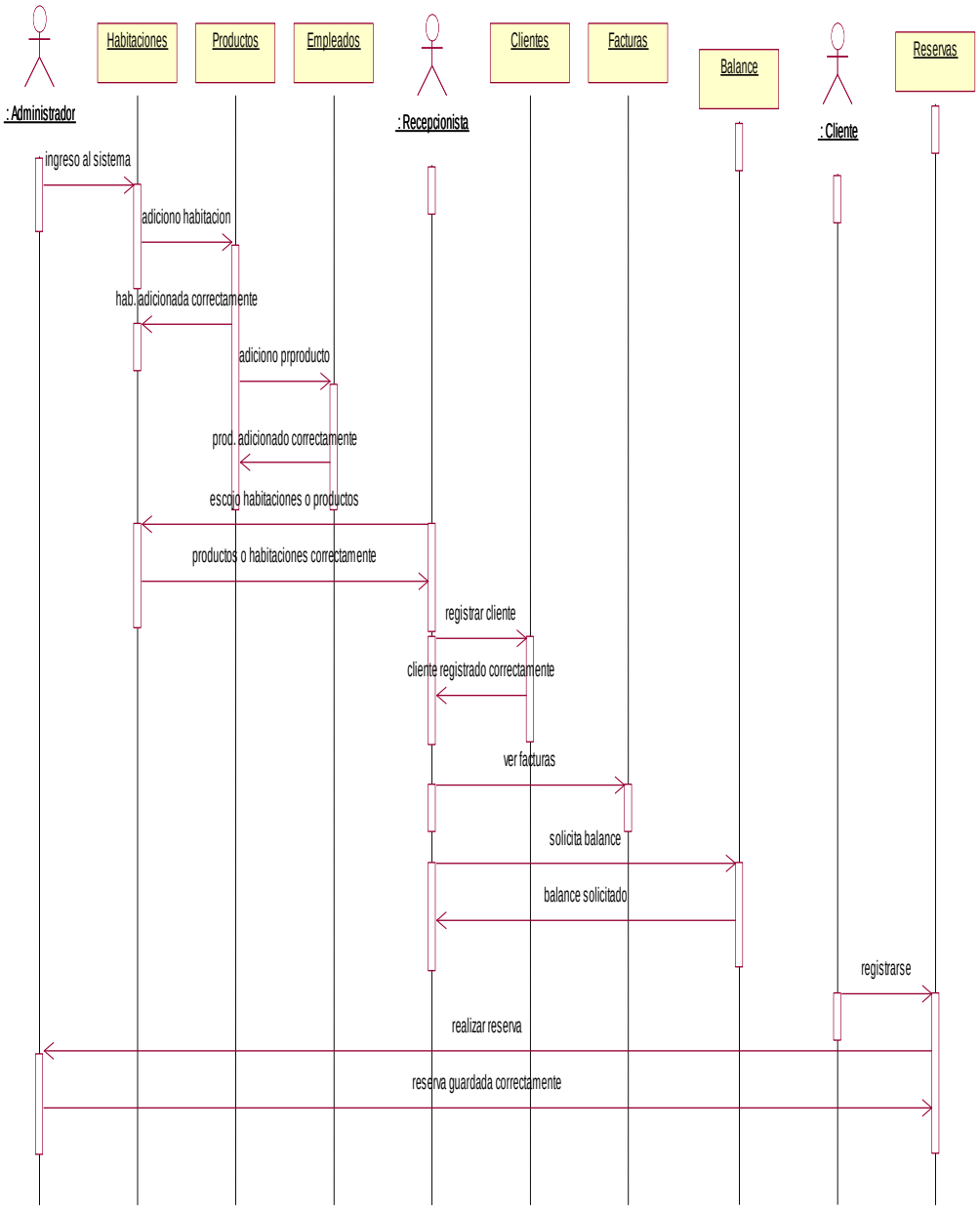
Finalidad Reservas de huéspedes y facturación electrónica.	**Indicadores**	**Medios de Verificación** La encuesta fue realizada en el hotel METROPOLIS quienes son los encargados de las reservas.	**Supuestos** El administrador y cajera no tendrán problema para realizar una reserva.
Propósito Incrementar la reservas de huéspedes mediante la web	El incremento de reservas aumentan un 90%	Mediante la entrevista realizada intervenían directamente con el sistema.	Se lograra realizar el proceso de reservación y facturación electrónica.
Componentes Ahorro de tiempo y dinero.	El ahorro de tiempo disminuye un 95%	El análisis realizado permitió crear nuevas estrategias para la reserva de habitaciones y brindar un servicio mejor a los clientes.	Los clientes están dispuestos a utilizar la tecnología para realizar sus reservas
Actividades A través de las reservas mediante la web.	las necesidades y requerimientos que el desarrollador requiere se basa en función el cual el proyecto esta desarrollado.	Realizar un reporte de ventas semanal, mensual.	Establecer técnicas adecuadas para la realización de las reservas .

***En** esta figura muestra la Matriz del Marco-Lógico.*

4.09. Vistas arquitectónicas

4.01.01. Vista lógica

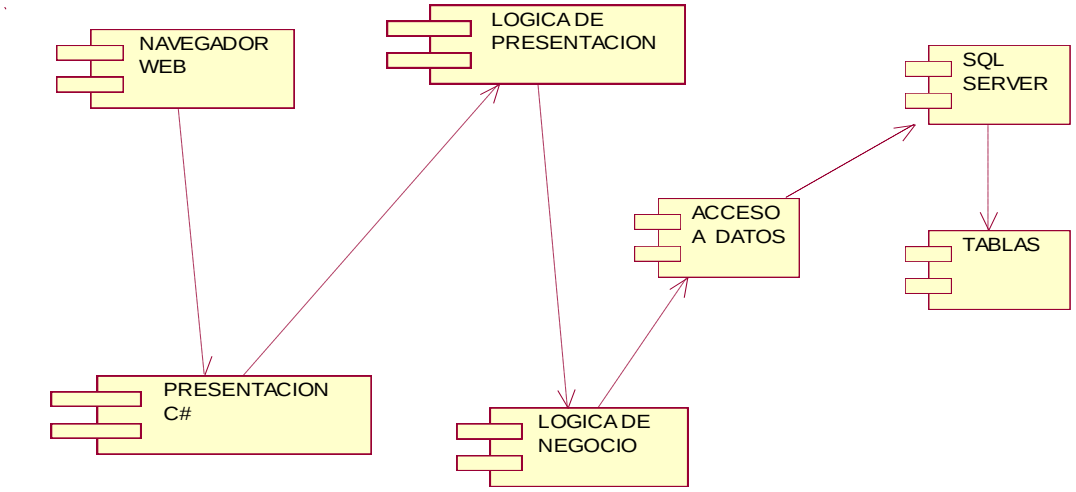
Figura 27 Vista lógica



Vista Arquitectónica-Lógica, describe la estructura de cada una de las funciones del sistema.

4.01.02. Vista física

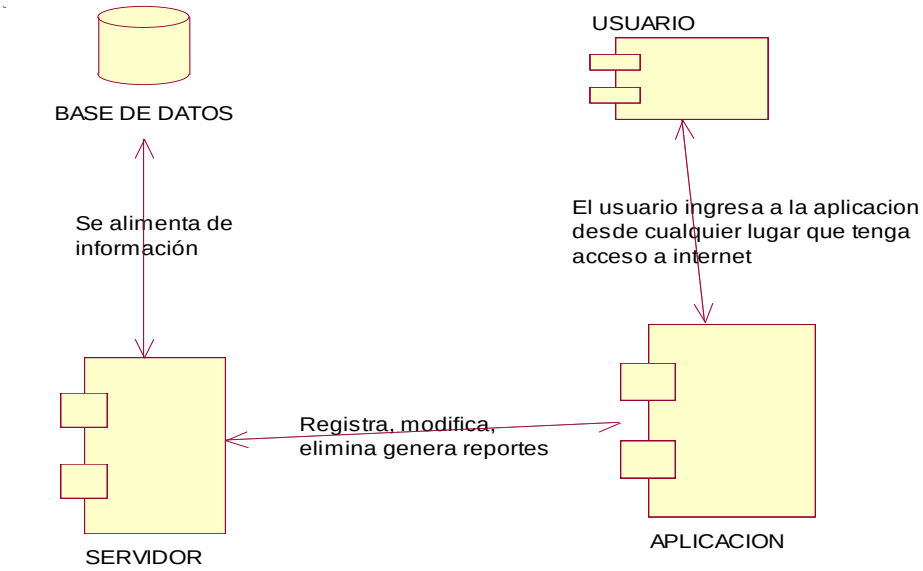
Figura 28 Vista física



Vista Arquitectónica. Muestra en un diagrama de componentes la solución del sistema realizado.

4.01.03. Vista de desarrollo

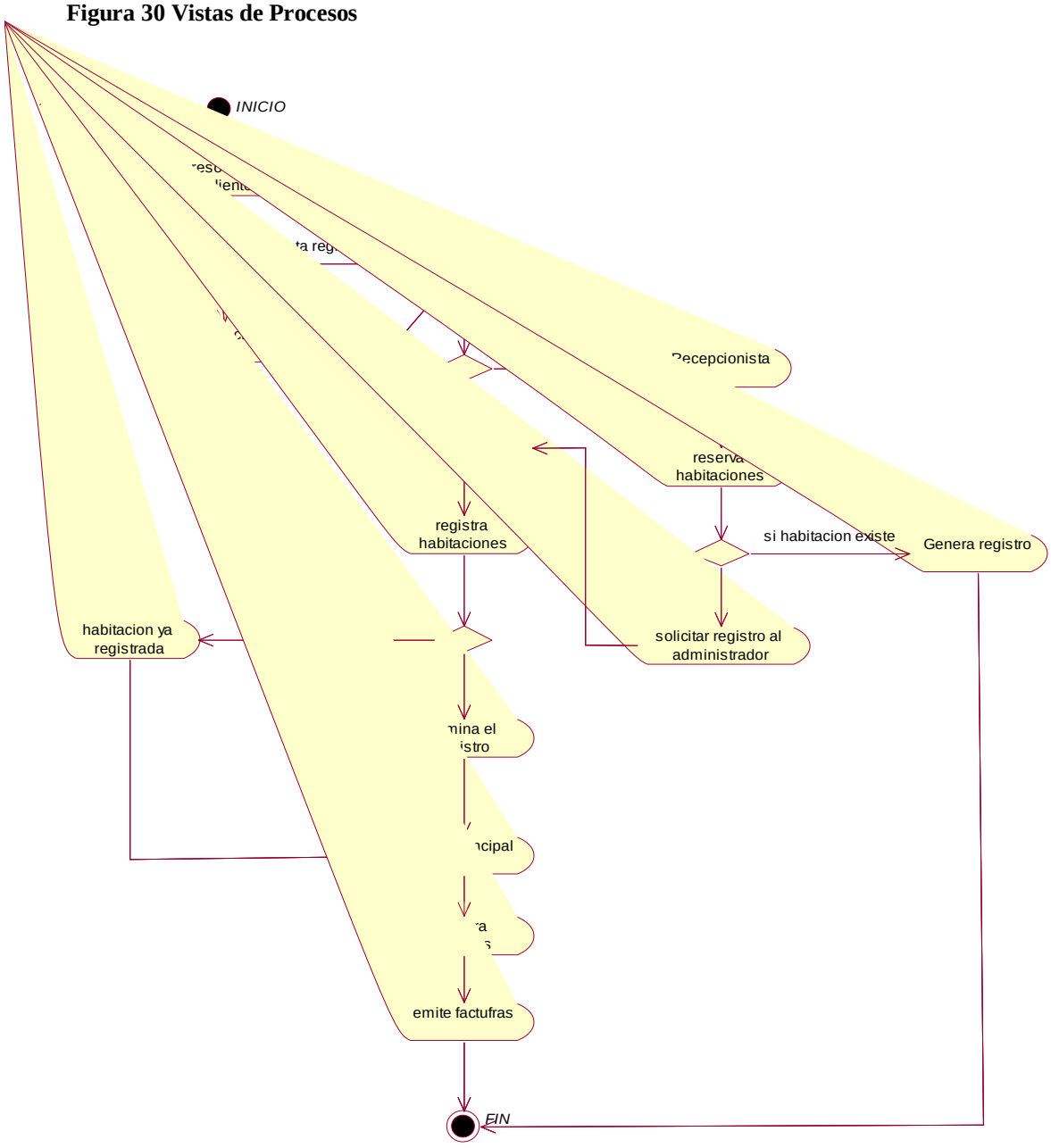
Figura 29Vista de desarrollo



Vista arquitectónica de Desarrollo. Muestra los módulos en el ambiente de desarrollo.

4.01.04 Vistas de Procesos

Figura 30 Vistas de Procesos



En esta figura muestra como está estructurado el sistema vista Arquitectónica de Proceso.

Capítulo V: Propuesta

5.01. Especificación de estándares de programación

Al programar en un lenguaje específico se debe seguir reglas que permitirán interpretar de manera eficiente la escritura del código lo cual procedemos de esta manera.

La Capa de Modelo

El modelo representa la parte de la aplicación que implementa la lógica de negocio. Esto significa que es responsable de la recuperación de datos convirtiéndolos en conceptos significativos para la aplicación, así como su procesamiento, validación, asociación y cualquier otra tarea relativa a la manipulación de dichos datos.

La Capa de la Vista

La vista hace una presentación de los datos del modelo estando separada de los objetos del modelo. Es responsable del uso de la información de la cual dispone para producir cualquier interfaz de presentación de cualquier petición que se presente.

Por ejemplo, como la capa de modelo devuelve un conjunto de datos, la vista los usaría para hacer una página HTML que los contenga. O un resultado con formato XML para que otras aplicaciones puedan consumir.

La Capa del Controlador

La capa del controlador gestiona las peticiones de los usuarios. Es responsable de responder la información solicitada con la ayuda tanto del modelo como de la vista.

Los controladores pueden ser vistos como administradores cuidando de que todos los recursos necesarios para completar una tarea se deleguen a los trabajadores más adecuados. Espera peticiones de los clientes, comprueba su validez de acuerdo a las normas de autenticación o autorización, delega la búsqueda de datos al modelo y selecciona el tipo de respuesta más adecuado según las preferencias del cliente. Finalmente delega este proceso de presentación a la capa de la Vista.

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

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

5.02. Diseño de Interfaces de Usuario

Figura 31 Diseño de Interfaces de Usuario

En esta figura muestra el Login. Nos permite ingresar al sistema ingresando Usuario y contraseña.

Habitaciones y productos

- Adicionar habitaciones.
- Adicionar productos.

Clientes

- Consultar clientes registrados.

Facturas

- Ver facturas vencidas,
- Generar PDF. XML.
- Cobrar facturas.

Empleados

- Adicionar empleado eliminar, editar.

Balance.

- Reportes de ventas.

Figura 32 Muestra el listado de Habitaciones

Figura 33 Muestra para adicionar datos por habitación

Muestra para adicionar datos por habitación

Figura 34 Muestra adicionar producto con su respectivo precio.

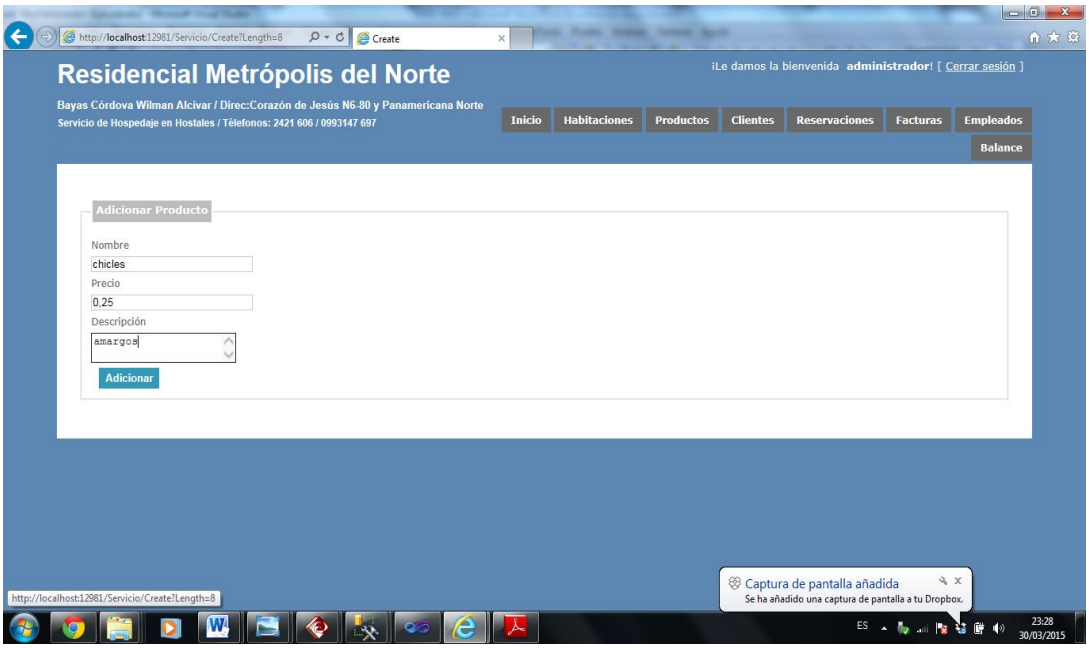


Figura 35Muestra el listado de clientes registrados para realizar su respectiva reserva.

Figura 36 Muestra el listado de las facturas guarda en PDF y genera XML

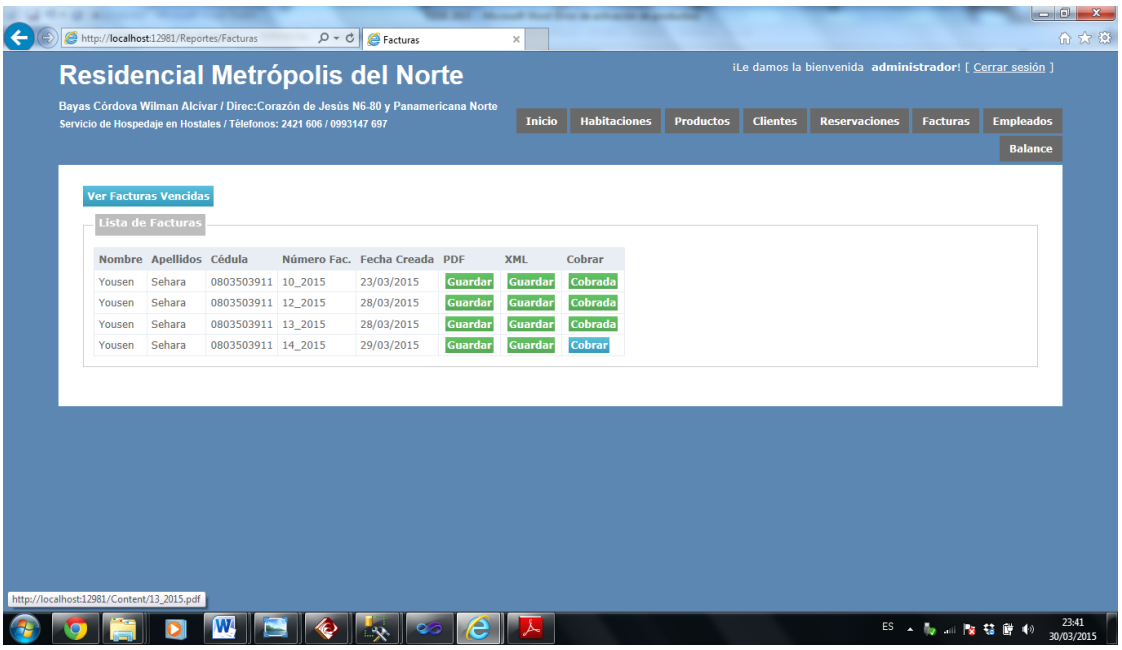


Figura 37Muestra el total a cancelar la factura.

Figura 38 Muestra para adicional empleado, eliminar, editar.

Figura 39 Muestra el arqueo de caja semanal, mensual.

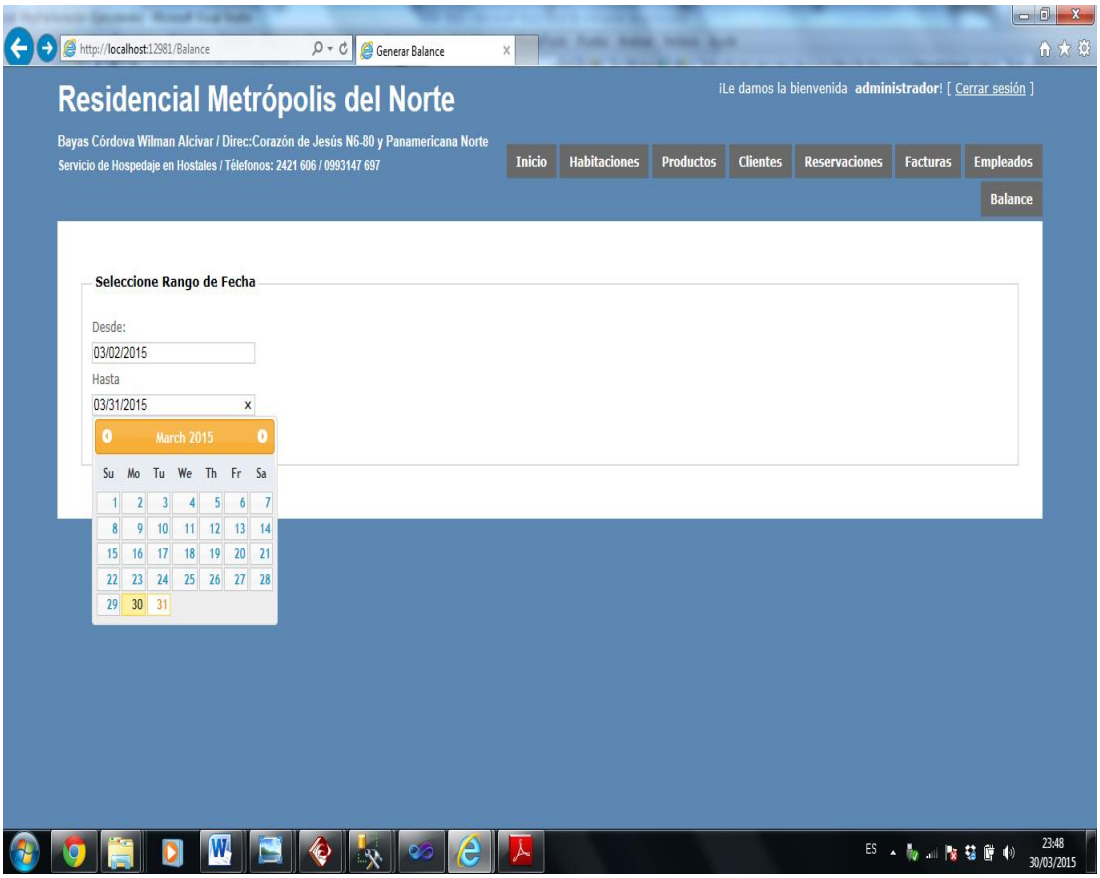


Figura 40 Muestra el PDF de los reportes semanales, mensuales

Figura 41Muestra las habitaciones disponibles para realizar la reserva

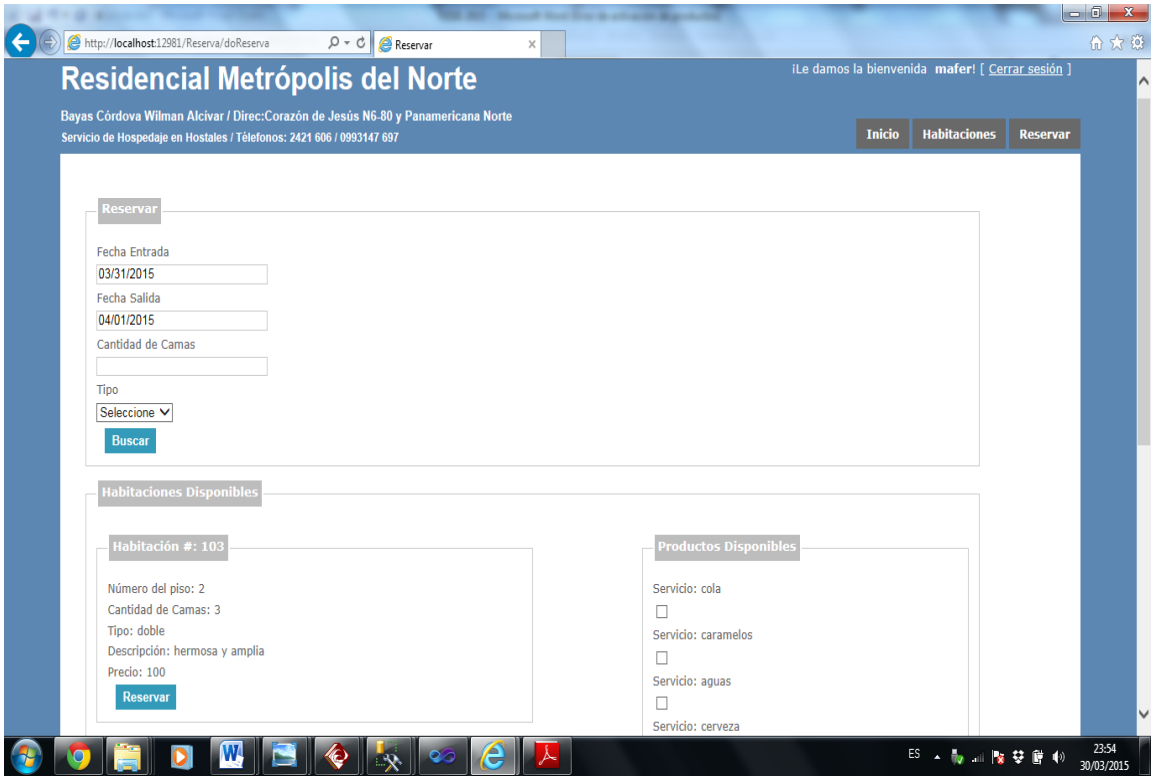
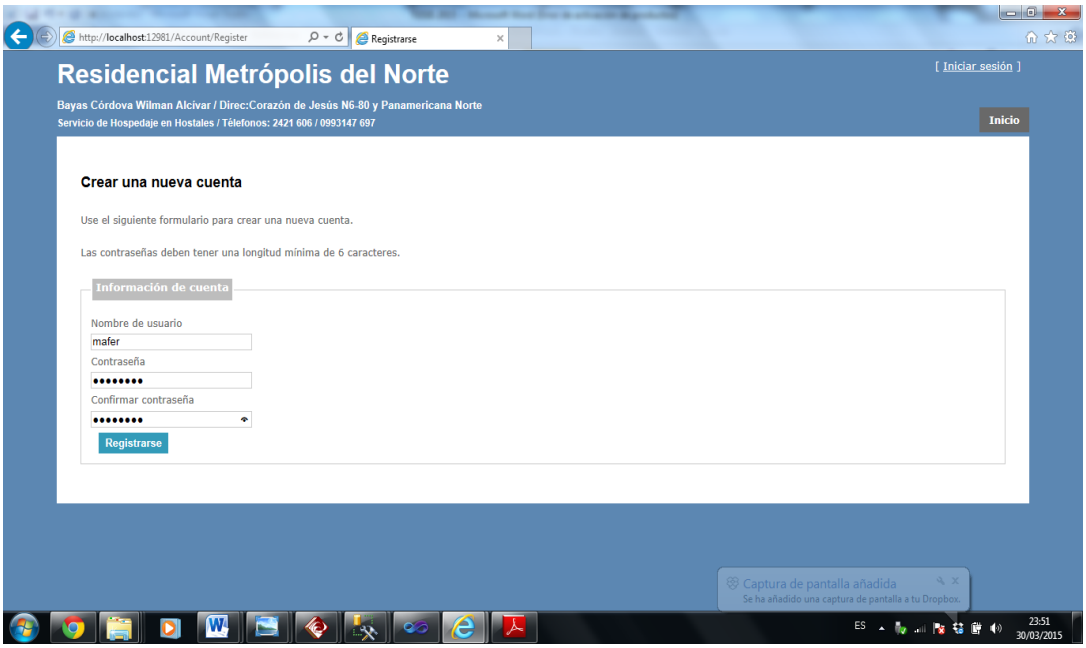


Figura 42 Muestra una ventana para registrarse por primera vez



5.03. Especificación de pruebas de unidad

Tabla 25

Especificación de pruebas de unidad

Identificador de la prueba	**Login de acceso al sistema.**
Método a probar	La forma en que los distintos usuarios acceden.
Objetivo de la prueba	Consiste si el método cumple en los estándares del manejo.
Datos de la entrada	
Nombre de usuario y contraseña	
Resultados esperados	
Acceso al sistema correctamente	
Comentarios	
Su acceso a los distintos módulos será de acuerdo al perfil de usuario.	

Nota: Muestra los resultados las pruebas de ingreso ala sistema.

Tabla 26

Registro clientes mediante la web

Identificador de la prueba	**Registro clientes mediante la web**
Método a probar	Registro de nuevo cliente.
Objetivo de la prueba	Comprobar el manejo de cliente y sus limitaciones con el sistema.
Datos de la entrada	
Datos de clientes y reservas	
Resultados esperados	
Limitar el rango de acceso.	
Comentarios	
Registro de cliente de acuerdo a sus necesidades.	

Nota: Muestra los resultados esperados luego de realizar las pruebas de registro de clientes mediante la web.

Tabla 27

Emisión y registro de factura

Identificador de la prueba	**Emisión y lista de facturas por cobrar.**
Método a probar	Calculo de valores a facturar.
Objetivo de la prueba	Emisión de registro de factura y generación del XML
Datos de la entrada	
Selección de reservas para ser cobradas.	
Resultados esperados	
Emisión de factura digital.	
Comentarios	
Su factura será enviada al correo de acuerdo al cliente.	

Nota: Muestra el registro de facturas por cobrar y facturas vencidas.

5.04. Especificación de pruebas de aceptación

Tabla 28

Especificación de pruebas de aceptación

Identificador de la prueba	**Registrar datos de usuario.**
Caso de uso	Registrar datos.
Tipo de Usuario	Personal Administrativo.
Objetivo de la Prueba	Se debe verificar que el usuario no exista para realizar su registro.
Secuencia de eventos.	
Resultados esperados	
El sistema debe emitir un mensaje si ingresa un nombre ya existente.	
Comentarios	
Para este registro debe ir al módulo cliente.	
Estado: **Aceptado**	

NOTA: Muestra los resultados esperados una vez realizadas las pruebas de aceptación

Tabla 29

Habitaciones disponibles.

Identificador de la prueba	**Habitaciones disponibles.**
Caso de uso	Consulta de habitaciones, productos y sus respectivos precios
Tipo de Usuario	Cliente –Recepcionista.
Objetivo de la Prueba	Se debe verificar habitaciones registradas.
Secuencia de eventos.	
Resultados esperados	
El sistema debe visualizar las habitaciones disponibles	
Comentarios	
Para este registro debe ir al listado de habitaciones	
Estado: Aceptado**.**	

Nota: muestra los resultados esperados de las consulta de habitaciones y productos.

Tabla 30

El cliente registra su reservación

Identificador de la prueba	**El cliente registra su reservación**
Caso de uso	Realizar reservas
Tipo de Usuario	Cliente
Objetivo de la Prueba	Se debe registrar y escoger habitaciones disponibles para su respectiva reserva.
Secuencia de eventos.	
Resultados esperados	
El sistema debe realizar su respectiva reserva de acuerdo a los días de hospedaje del cliente.	
Comentarios	
Para este registro el cliente debe llenar sus respectivos datos.	
Estado: **Aceptado.**	

Nota: Muestra los resultados esperados de reserva.

Tabla 31

Consulta de reservas realizadas

Identificador de la prueba	**Consulta de reservas realizadas**
Caso de uso	Reporte de reservas realizadas.
Tipo de Usuario	Recepcionista- Administrador.
Objetivo de la Prueba	Debe visualizar las respectivas reservas de los distintos clientes.
Secuencia de eventos.	
Resultados esperados	
El sistema debe mostrar todas las reservas ya que reservas caducadas serán eliminadas.	
Comentarios	
Para este registro el cliente debe estar a la fecha actual de su reserva.	
Estado: Aceptado**.**	

Nota: Muestra los resultados de las reservas realizadas.

Tabla 32

Proceso de facturación de la reserva.

Identificador de la prueba	**Proceso de facturación de la reserva.**
Caso de uso	Factura
Tipo de Usuario	Administración
Objetivo de la Prueba	Realizar su respectiva factura y procederla a cobrar.
Secuencia de eventos.	
Resultados esperados	
El sistema debe mostrar todas la facturas por cobrar y cobradas y generar el XML enviarla al correo de acuerdo al cliente.	
Comentarios	
Para este registro el cliente debe haber generado una factura.	
Estado:	

Nota: Muestra los resultados del proceso de facturación.

5.05. Especificación de pruebas de carga

Tabla 33

Especificación de pruebas de carga

Identificador de la prueba	**Carga del sistema.**
Tipo de prueba	Simulación de rendimiento en tiempo real, agregando varios usuarios, emitiendo facturas simultáneamente.
Objetivo de la Prueba	Simular carga de clientes y registro y comprobar rendimiento del sistema.
Descripción	
Ingresar valores de prueba y herramientas del sistema.	
Resultados esperados	
Que el sistema mantenga su nivel de rapidez.	
Comentarios	
El sistema se limita al anche de banda, se recomienda aumenta el ancho de banda.	

Nota: Muestra los resultados esperados una vez realizada las pruebas de carga.

5.06. Configuración del Ambiente mínima/ideal

Nuestro software es uso y ejecución en el portal web mínimo de uso son:

Para funcionalidad del software.

- Mínimo de espacio en el disco de 3 Mb para la instalación
- Un ancho de banda mínimo 1000 Mb para garantizar la ejecución completa de todo el desarrollo y pueda ejecutarse todo lo requerido con fluidez.

Para ejecución en punto de trabajo.

- Sistema operativo Windows, Linux que tengan un navegador web.
- Computador con un mínimo de procesador ATON de 2.2 GHz

Capítulo VI: Aspectos Administrativos

6.01. Recursos

Tenemos en consideración los siguientes recursos:

- Recursos Humanos.
- Recaudación de la información.
- Recursos materiales.

Recursos Humanos

- Tutor.
- Estudiante.

Recursos Materiales

- Computador.
- Internet.

6.02. Presupuesto

En el transcurso de la elaboración de la tesis se fueron consumiendo todos los recursos mencionados en la tabla.

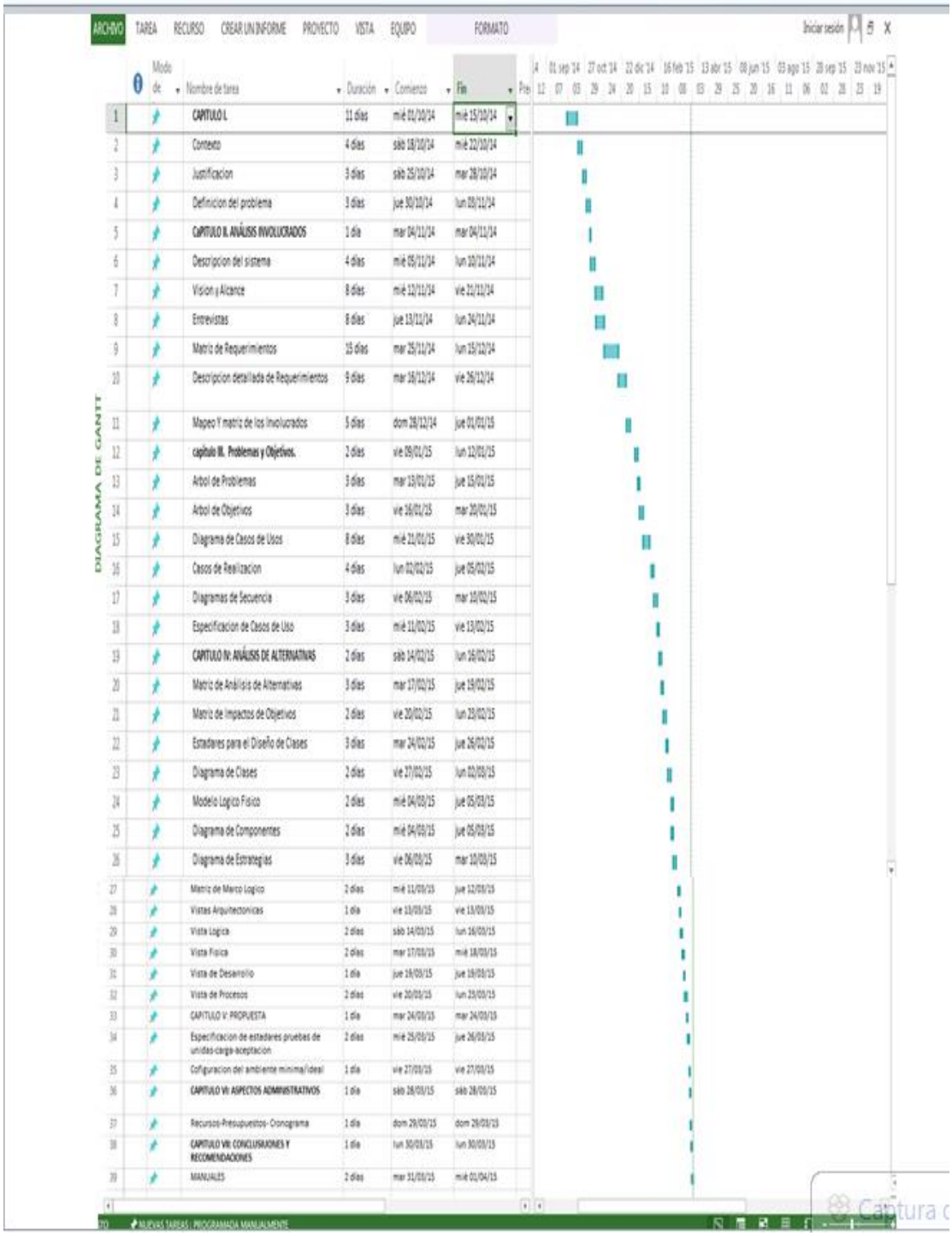
Tabla 34 Presupuesto

Presupuesto **(Ver Anexo A.01)**

6.03. Cronograma

Tabla 35

cronograma



	Modo de	Nombre de tarea	Duración	Comienzo	Fin
1		CAPITULO I.	11 días	mié 01/10/14	mié 15/10/14
2		Contexto	4 días	sáb 18/10/14	mié 22/10/14
3		Justificacion	3 días	sáb 25/10/14	mar 28/10/14
4		Definicion del problema	3 días	jue 30/10/14	lun 03/11/14
5		CAPITULO II. ANÁLISIS INVOLUCRADOS	1 día	mar 04/11/14	mar 04/11/14
6		Descripcion del sistema	4 días	mié 05/11/14	lun 10/11/14
7		Vision y Alcance	8 días	mié 12/11/14	vie 21/11/14
8		Entrevistas	8 días	jue 13/11/14	lun 24/11/14
9		Matriz de Requerimientos	15 días	mar 25/11/14	lun 15/12/14
10		Descripcion detallada de Requerimientos	9 días	mar 16/12/14	vie 26/12/14
11		Mapeo Y matriz de los Involucrados	5 días	dom 28/12/14	jue 01/01/15
12		capitulo III. Problemas y Objetivos.	2 días	vie 09/01/15	lun 12/01/15
13		Arbol de Problemas	3 días	mar 13/01/15	jue 15/01/15
14		Arbol de Objetivos	3 días	vie 16/01/15	mar 20/01/15
15		Diagrama de Casos de Usos	8 días	mié 21/01/15	vie 30/01/15
16		Casos de Realizacion	4 días	lun 02/02/15	jue 05/02/15
17		Diagramas de Secuencia	3 días	vie 06/02/15	mar 10/02/15
18		Especificacion de Casos de Uso	3 días	mié 11/02/15	vie 13/02/15
19		CAPITULO IV. ANÁLISIS DE ALTERNATIVAS	2 días	sáb 14/02/15	lun 16/02/15
20		Matriz de Análisis de Alternativas	3 días	mar 17/02/15	jue 19/02/15
21		Matriz de Impactos de Objetivos	2 días	vie 20/02/15	lun 23/02/15
22		Estadares para el Diseño de Clases	3 días	mar 24/02/15	jue 26/02/15
23		Diagrama de Clases	2 días	vie 27/02/15	lun 02/03/15
24		Modelo Logico Fisico	2 días	mié 04/03/15	jue 05/03/15
25		Diagrama de Componentes	2 días	mié 04/03/15	jue 05/03/15
26		Diagrama de Estrategias	3 días	vie 06/03/15	mar 10/03/15
27		Matriz de Marco Logico	2 días	mié 11/03/15	jue 12/03/15
28		Vistas Arquitectonicas	1 día	vie 13/03/15	vie 13/03/15
29		Vista Logica	2 días	sáb 14/03/15	lun 16/03/15
30		Vista Fisica	2 días	mar 17/03/15	mié 18/03/15
31		Vista de Desarrollo	1 día	jue 19/03/15	jue 19/03/15
32		Vista de Procesos	2 días	vie 20/03/15	lun 23/03/15
33		CAPITULO V. PROPUESTA	1 día	mar 24/03/15	mar 24/03/15
34		Especificacion de estadares pruebas de unidad-carga-aceptacion	2 días	mié 25/03/15	jue 26/03/15
35		Configuracion del ambiente minima/ideal	1 día	vie 27/03/15	vie 27/03/15
36		CAPITULO VI. ASPECTOS ADMINISTRATIVOS	1 día	sáb 28/03/15	sáb 28/03/15
37		Recursos-Presupuestos- Cronograma	1 día	dom 29/03/15	dom 29/03/15
38		CAPITULO VII. CONCLUSIONES Y RECOMENDACIONES	1 día	lun 30/03/15	lun 30/03/15
39		MANUALES	2 días	mar 31/03/15	mié 01/04/15

Nota: Esta tabla muestra el cronograma de actividades las fechas como se ha desarrollado el proyecto.

Capítulo VII: Conclusiones y Recomendaciones

7.01. Conclusiones

- Con la implementación del sistema lograremos optimización y la innovación en los procesos de reservas en los diferentes módulos.

- Los administradores tendrán la opción de adicionar habitaciones realizar búsqueda de las mismas.

- Los clientes se registraran para realizar sus reservas mediante la wen sin ningún inconveniente ya que la interface es fácil de entender.

7.02. Recomendaciones

- El personal administrativo tiene la obligación de revisar los manuales para el buen desempeño del sistema.

- Cualquier tipo de error que se presente en el sistema el personal deberá reportar al técnico para la solución.

ANEXOS

Anexo A.01 Presupuesto

Tabla 34
Presupuesto

Descripción	*Cantidad*	*Valor unitario*	*Valor total*
Seminario	*1*	*739,41*	*739,41*
Alimentación	*50*	*2,00*	*100,00*
Servicio Web Hosting	*1*	*180,00*	*180,00*
Servicio de internet	*45*	*0,80*	*36,00*
Impresiones	*6*	*15,00*	*90,00*
Anillados	*4*	*20,00*	*80,00*
Empastados	*1*	*35,00*	*35,00*
Servicio de energía eléctrica	*5*	*7*	*35,00*
Resmas de papel bond.	*2*	*4,20*	*8,40*
Equipos PC.	*2*	*620,00*	*1400,00*
Servidor	*1*	*700,00*	*700,00*
Impresora	*1*	*320,00*	*320,00*
TOTAL			*3723,81*

A.02 Manual de Instalaciòn

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

Figura **Página**

1.1 Instalación de SQL SERVER 2008

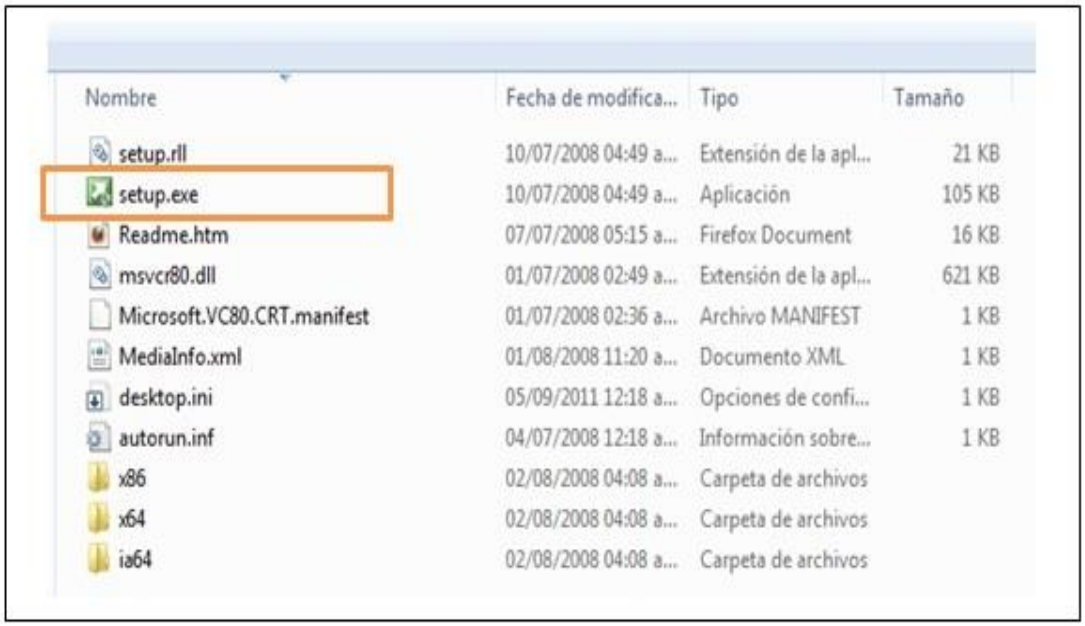
Microsoft SQL Server es un sistema para la gestión de bases de datos producido por Microsoft basado en el modelo relacional. Sus lenguajes para consultas son T-SQL y ANSI SQL.

http://es.wikipedia.org/wiki/Microsoft_SQL_Server

1.2 Dirección de la descarga del instalador.

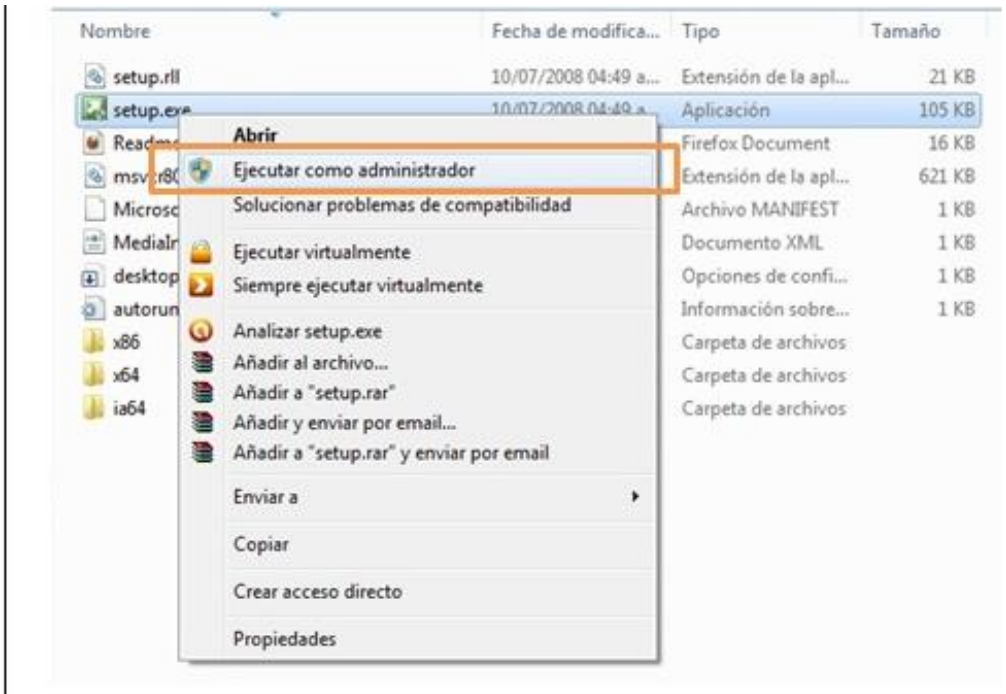
http://sql-server.softonic.com/

***Figura* 1**: seleccionar el setup.exe para ejecutarlo.



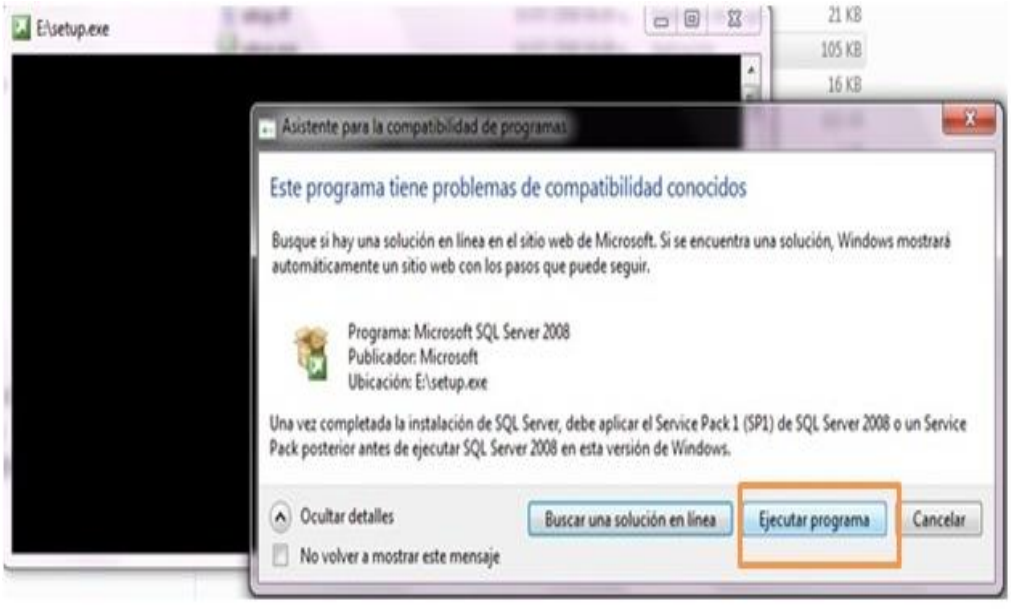
1. *seleccionamos la carpeta del setup.exe para luego ejecutarlo.*

***Figura* 2:** Ejecutar como administrador.



2. Seleccionamos click derecho en setup.exe y procedemos ejecutar como administrador.

***Figura* 3:** Ejecutar el programa.

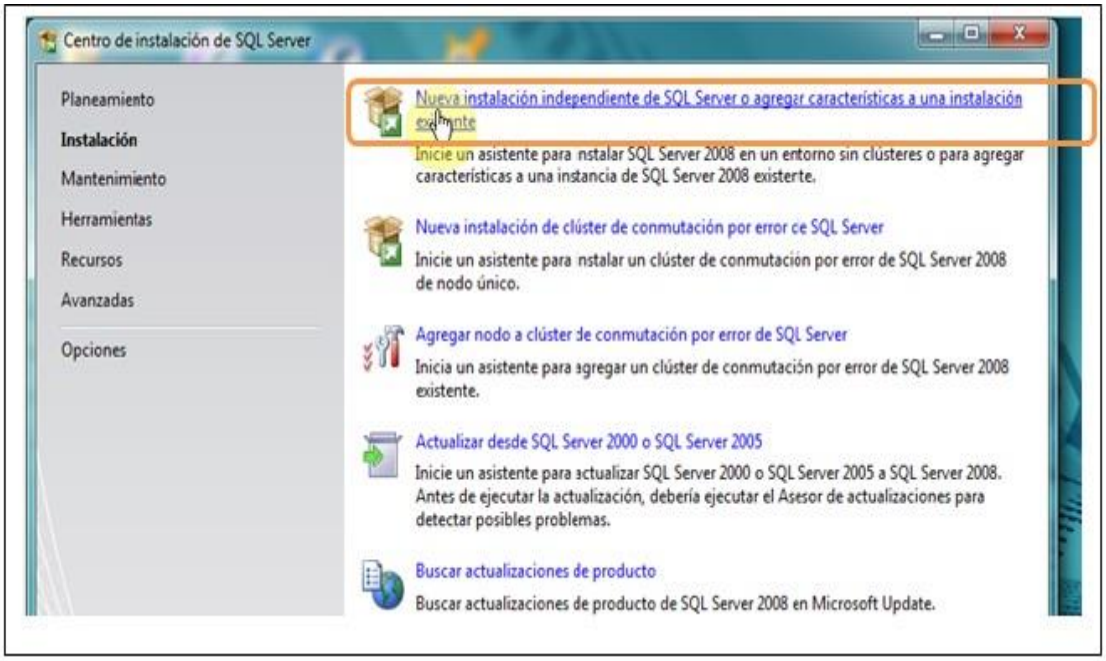


3. En esta ventana se procede a ejecutar el programa para su instalación.

***Figura* 4:** Click en Instalación.

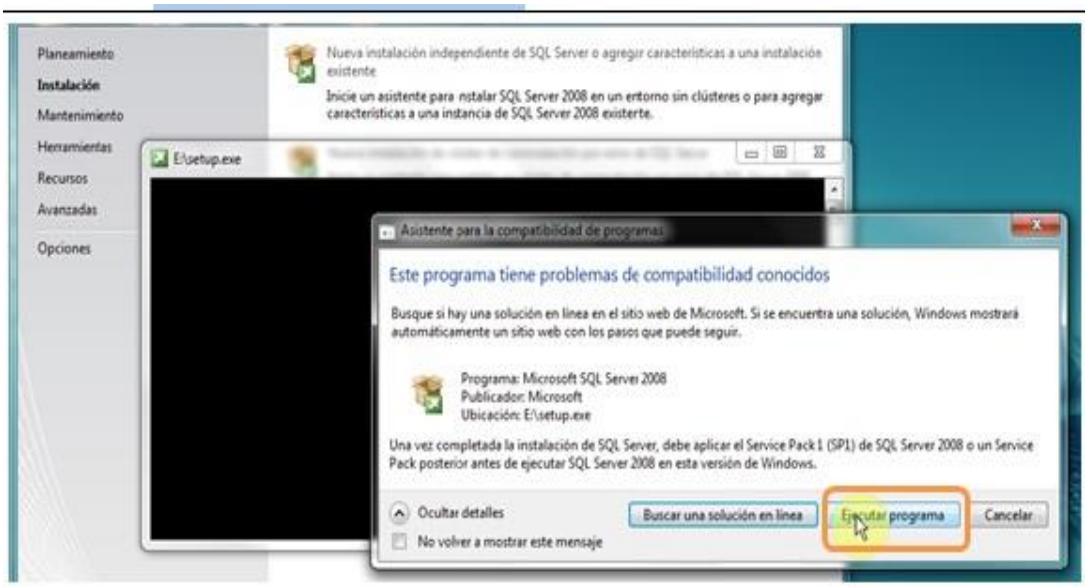
4. Esta figura muestra la ventana de la instalación.

Figura 5: Seleccionar SQL Server o agregar características.



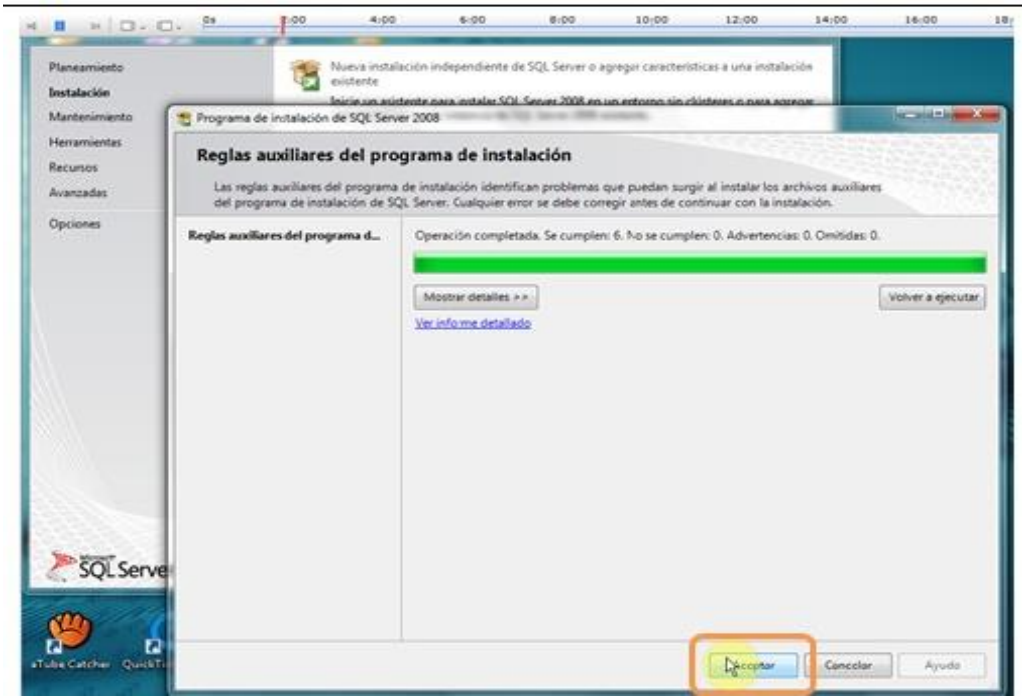
5. En esta figura escogemos la primera opción, Nueva instalación independiente de SQL Server o agregar características a una instalación existente.

Figura 6: Ejecutar la instalación para su respectiva instalación.



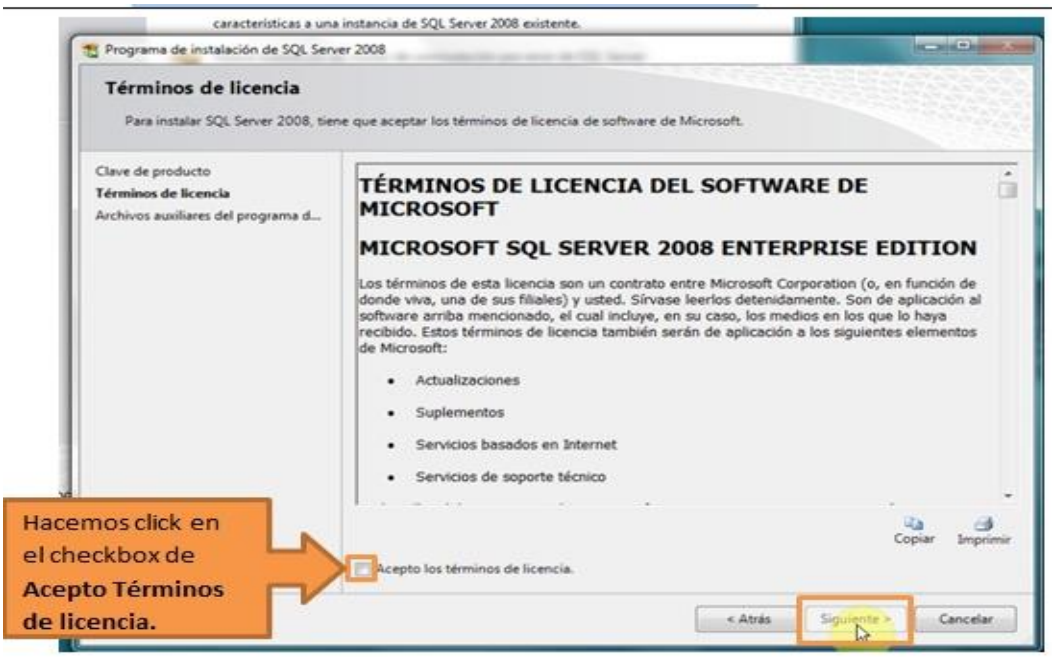
6. Seleccionamos la ejecución del programa para su instalación

Figura 7: instalación de las reglas auxiliares del programa.



7. Una vez terminando la instalación. Se procede con la instalación de las reglas auxiliares del programa luego presionamos ejecutar.

Figura 8: Aceptación de los términos de licencia.



8. Luego muestra una ventana de términos de licencia y damos click para aceptar los términos luego siguiente.

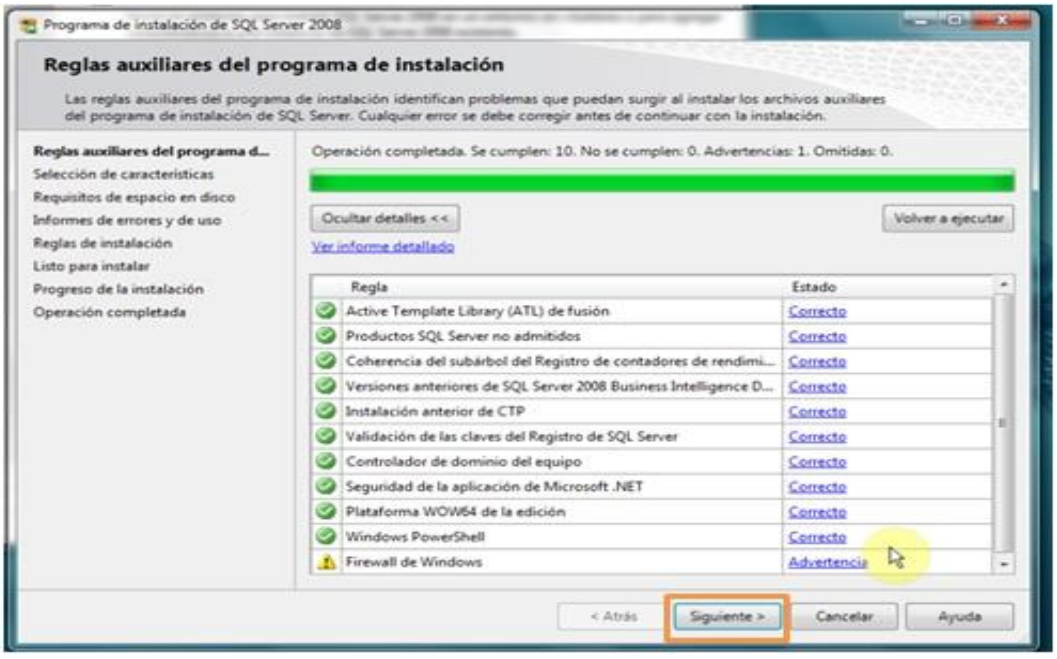
Figura 9: Instalación de archivos auxiliares del programa de instalación.

9. *Presionamos click en la instalación, esperamos que se termine la instalación.*

***Figura 10**: Completando la comprobación de reglas auxiliares.*

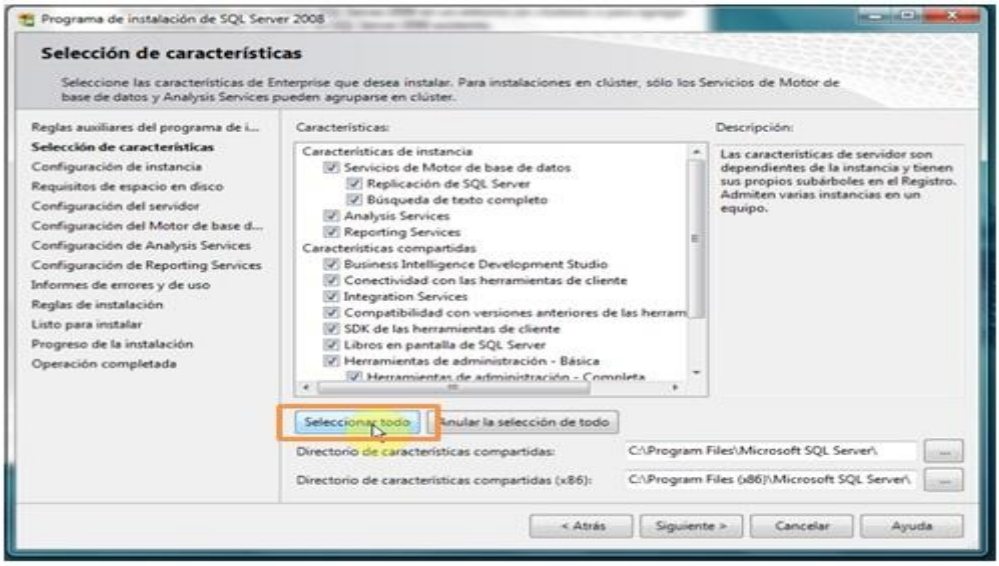
10. Esperamos que termine la instalación y luego presionamos siguiente.

***Figura 11**: Completado la instalación de reglas auxiliares.*



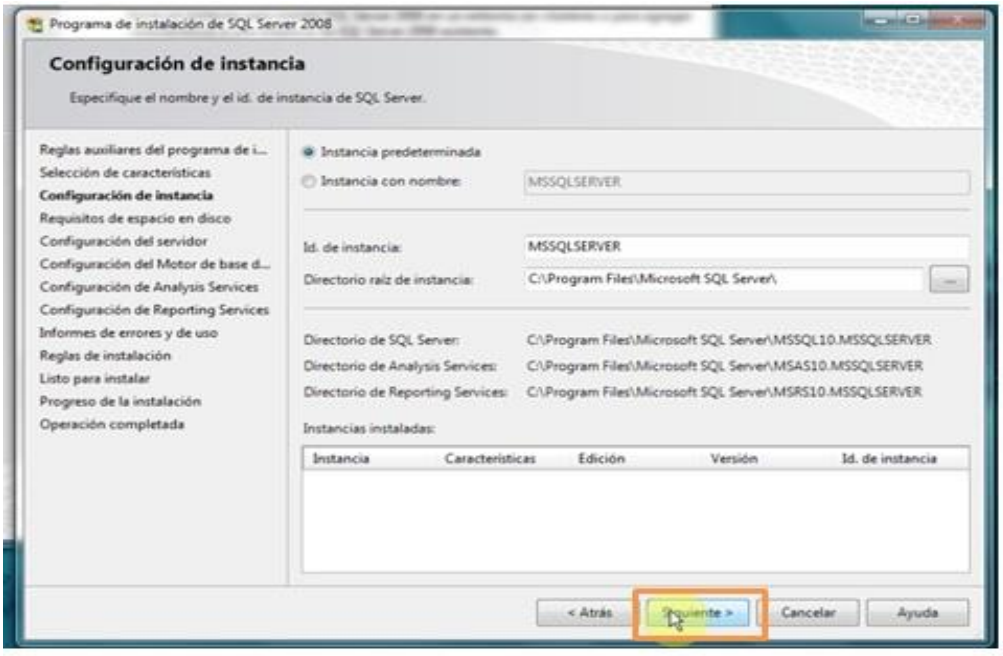
11. Una vez completado muestra una ventana, no importa si nos da error en Firewall de Windows, damos siguiente.

***Figura 12**: Selección de características.*



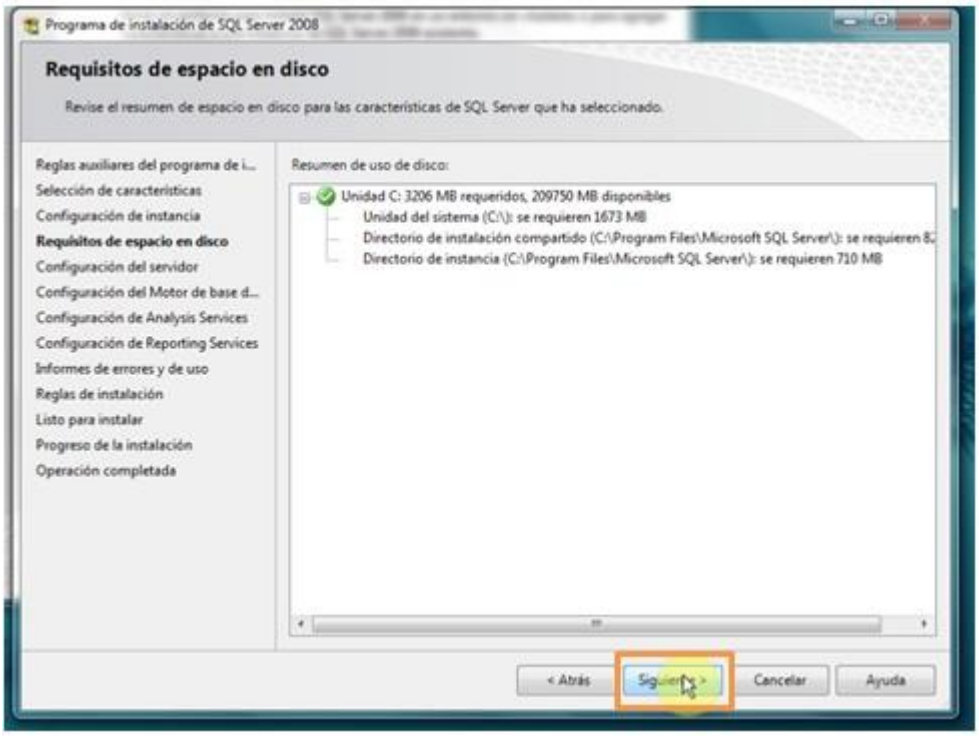
12. Muestra la ventana, donde seleccionamos todas las características, luego damos siguiente.

***Figura 13**: configuración de instancia.*



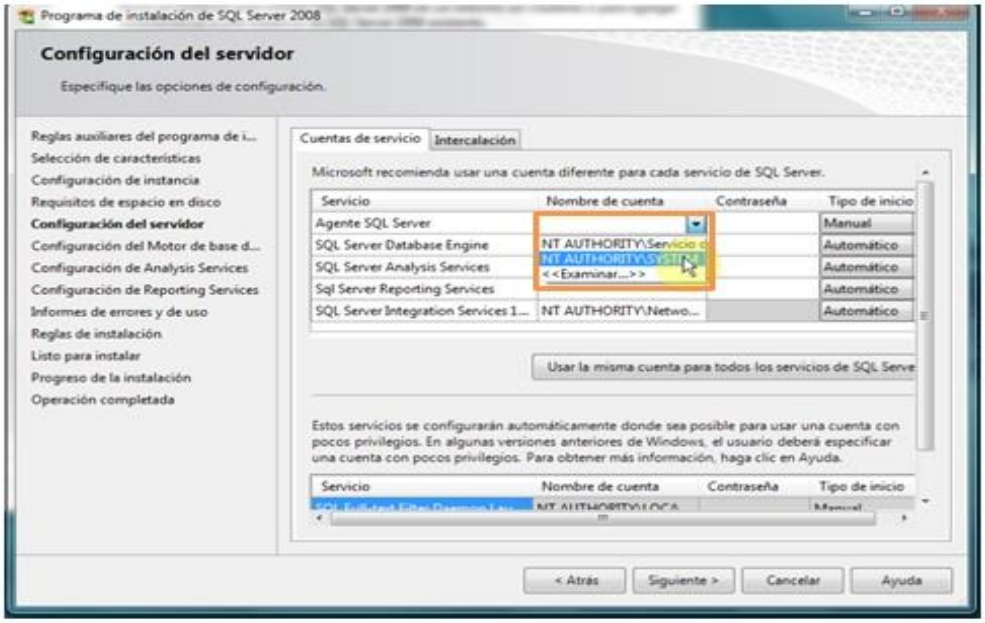
13. Luego apareceré una ventana de la configuración de instancia, hacemos click en el botón siguiente.

Figura 14: Requisitos de espacio en disco.



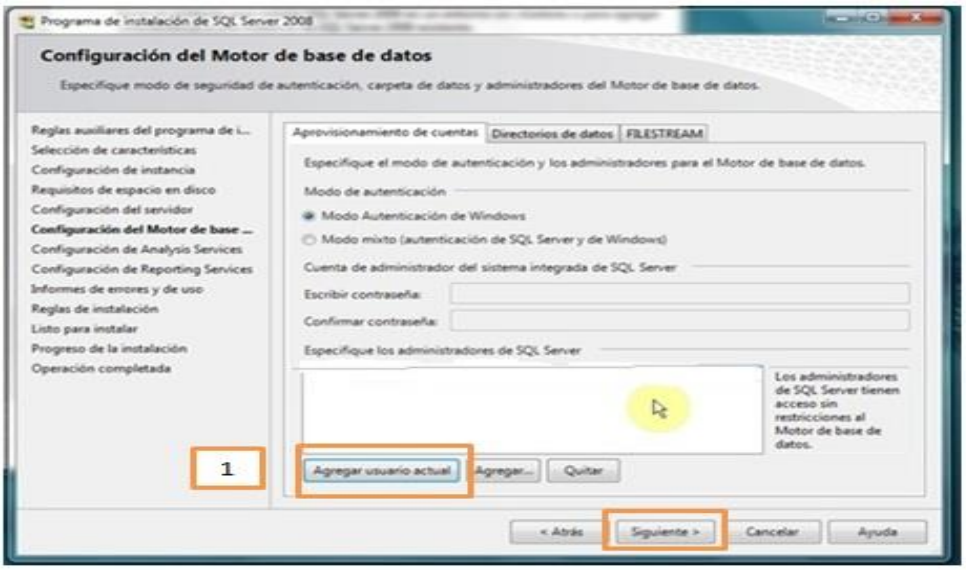
14. Luego aparece esta ventana y damos click en el botón siguiente.

Figura 15: *Configuración del Servidor.*



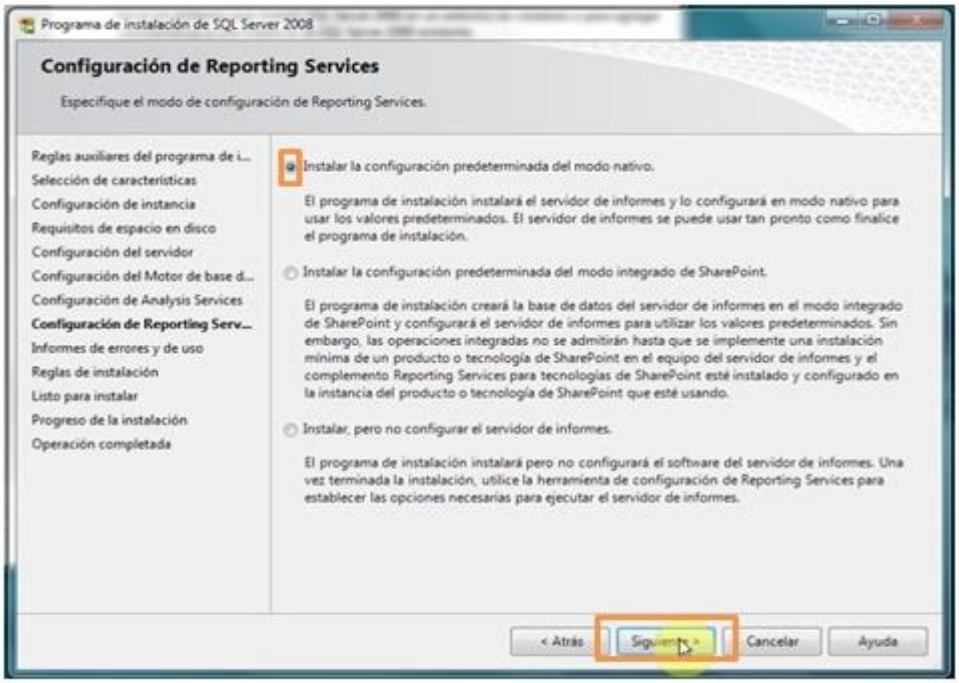
15. Esta ventana muestra la configuración del servidor Y luego seleccionamos en cada uno NT AUTHORITYSYSTEM. Presionamos en el botón siguiente.

Figura 16: configuración del Motor de base de datos.



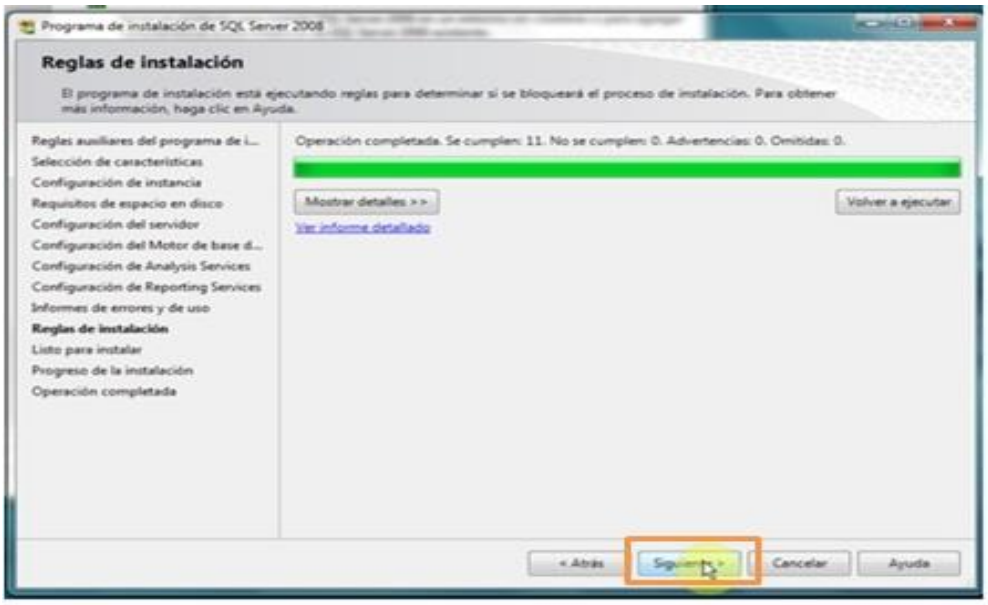
16. En la siguiente ventana damos click en el botón agregar usuario actual.

Figura 17: Configuración de Reporting Services.



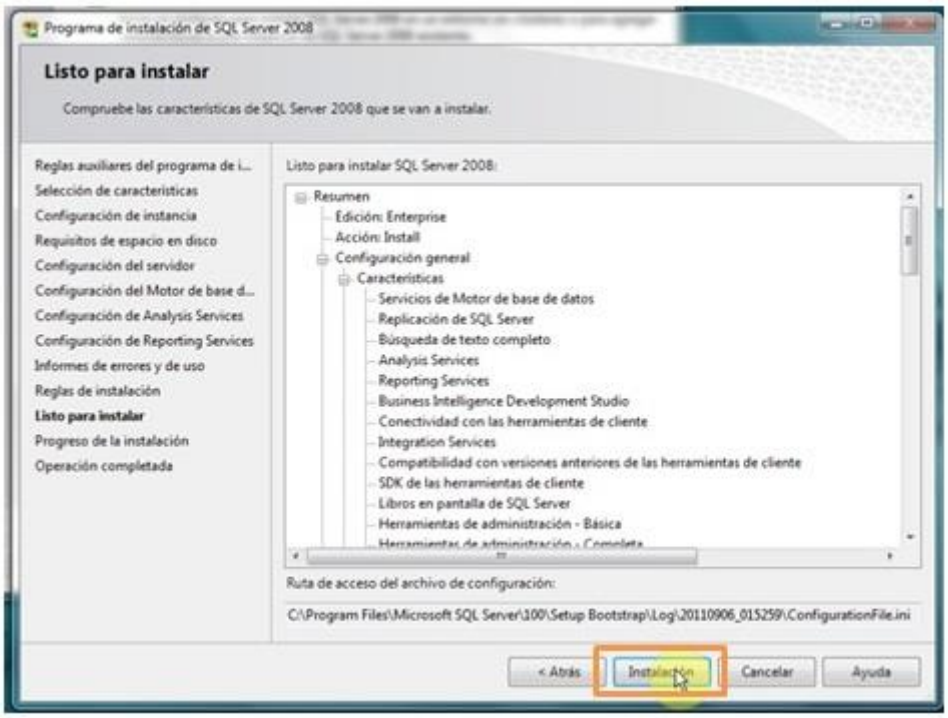
17. En esta ventana hacemos click en el botón siguiente.

Figura 18: Reglas de instalación.



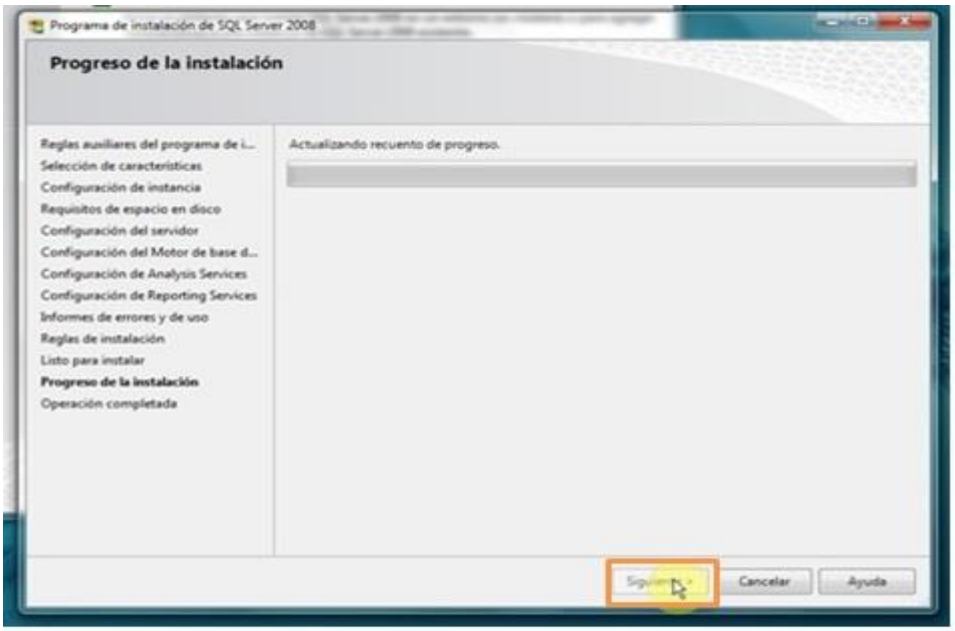
18. Luego esperamos que carguen las reglas de instalación y damos click rn siguiente.

Figura 19: Listo para instalar.



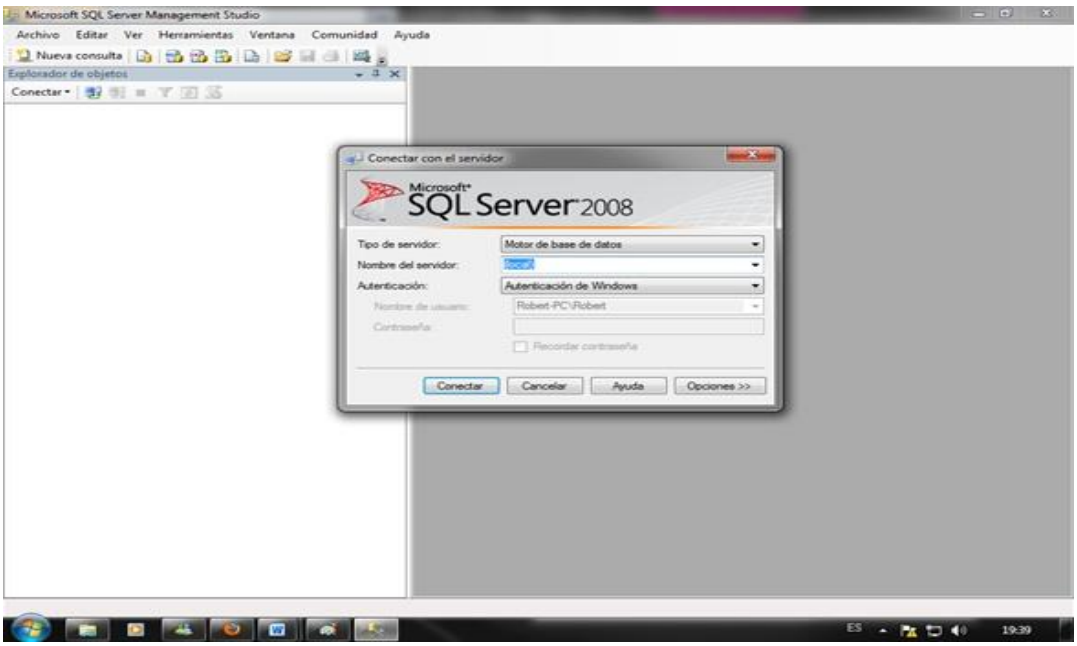
19. Esta ventana muestra que está listo para su respectiva instalación damos click en el botón instalación.

Figura 20: Progreso de la Instalación



20. Esperamos que se termine la instalación y luego damos click en siguiente.

Figura 21: listo para su ejecución SQL Server 2008



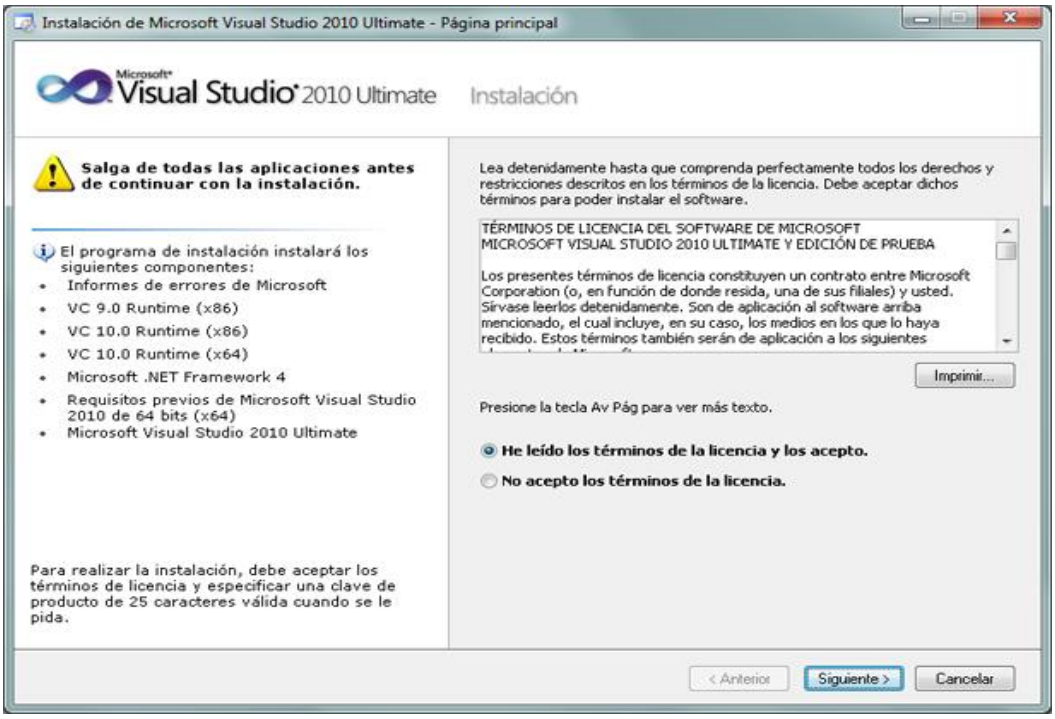
21. Esta ventana muestra el inicio cuando accedemos al programa SQL. Y listo para trabajar.

1.3 Instalación de Visual Studio 2010

Figura 22: instalación de Visual Studio 2010

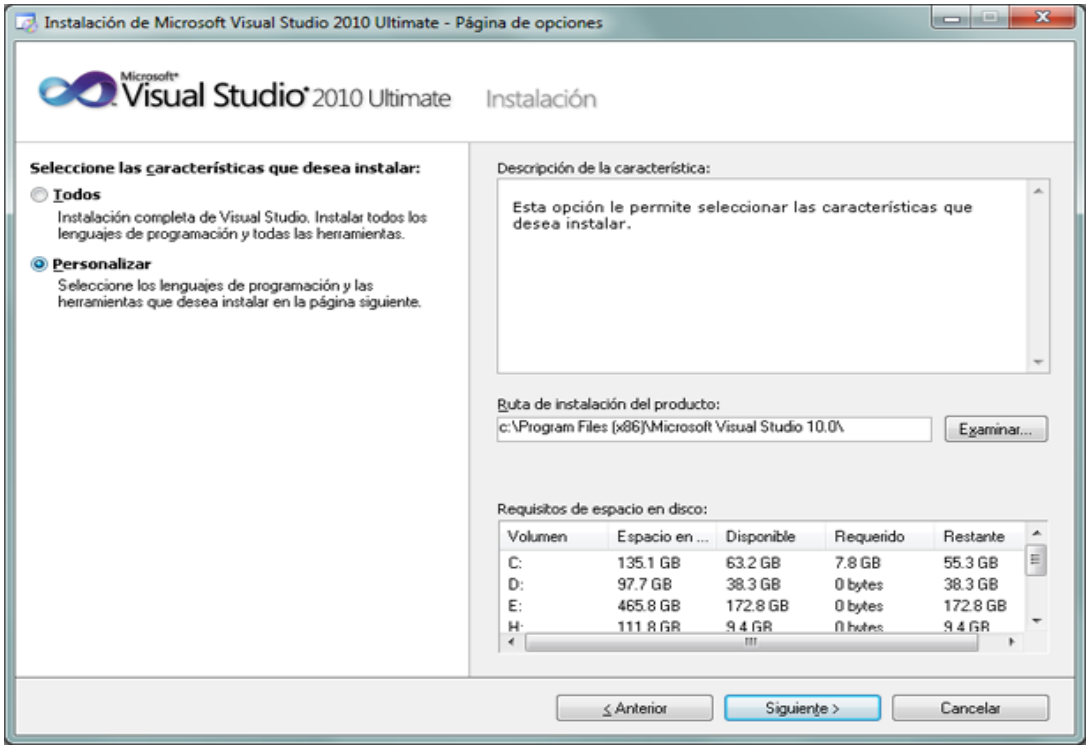
1. Damos click en instalar Microsoft visual studio 2010

Figura 23: Aceptar términos de licencia.



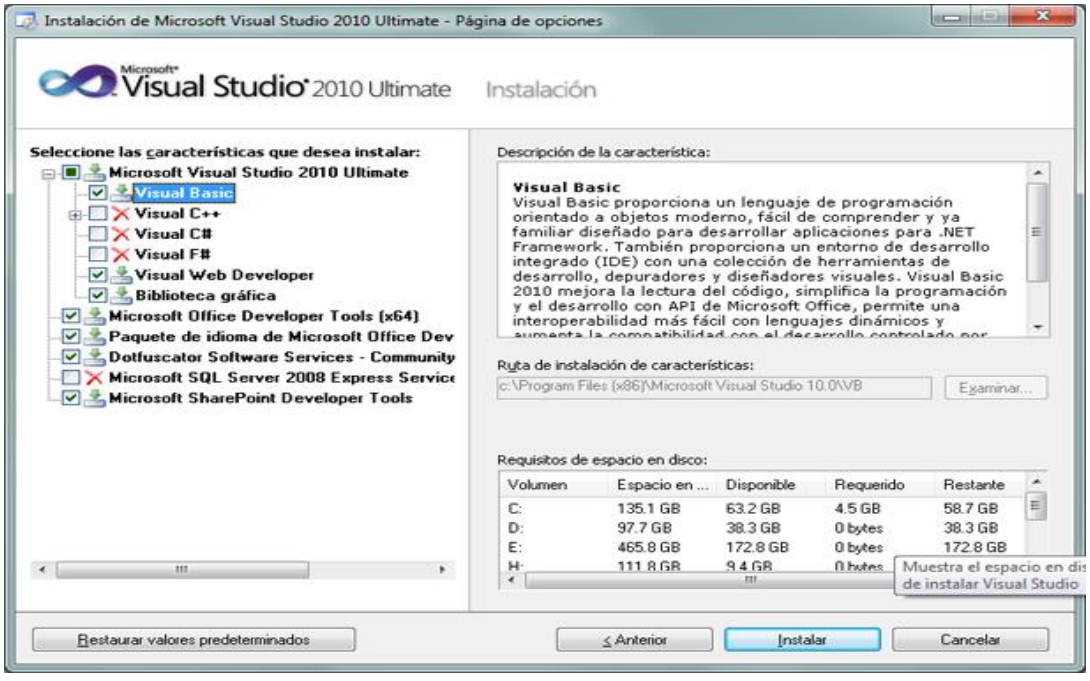
2. Seleccionamos los términos de licencia luego siguiente.

Figura 24: Personalizar.



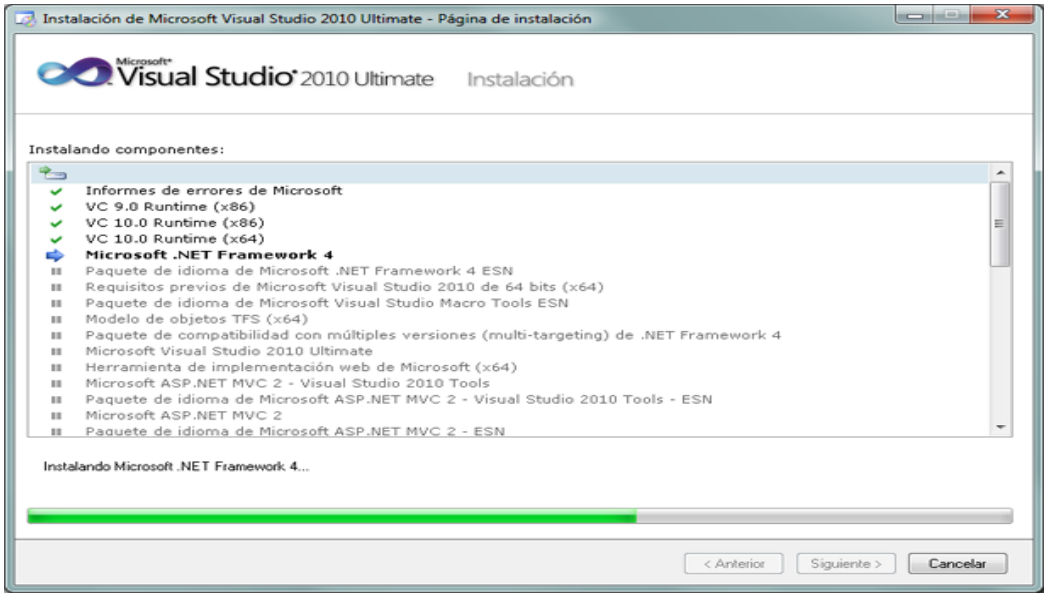
3. Marcamos la opción Personalizar para seleccionar los programas que necesitamos instalar

Figura 25: seleccionar las características de instalación



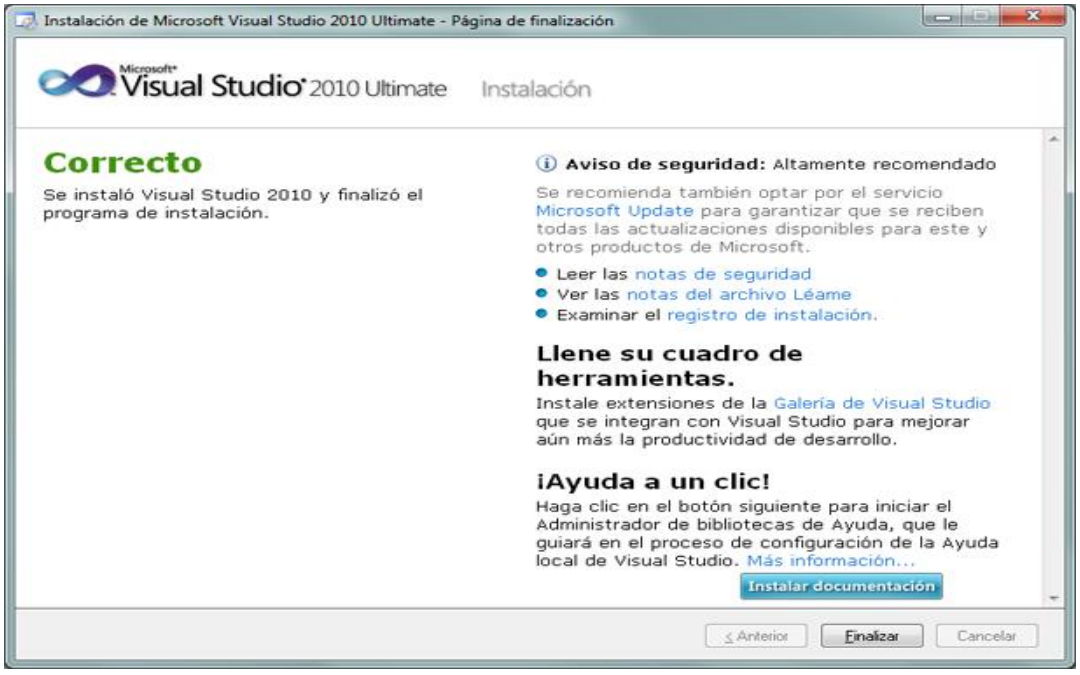
4. Seleccionamos los lenguajes que deseamos instalar.

Figura 26: Instalación de componentes.



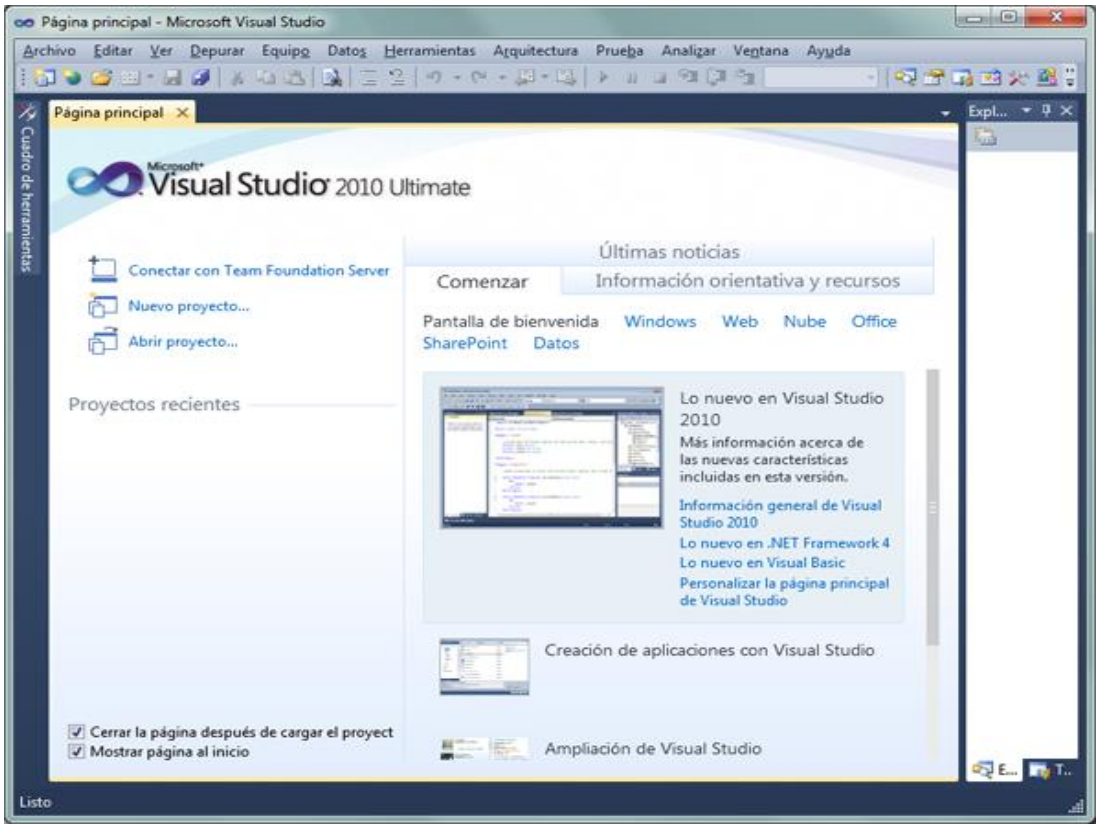
5. Instalación de los lenguajes de programación y esperamos que se complete luego se reiniciara.

Figura 27: instalación correcta de visual studio 2010



6. Finalización de la instalación damos click en finalizar.

Figura 28: Visual Studio 2010 instalado.



7. Listo para para iniciar por primera vez y empezar a trabajar.

A.03 Manual de Usuario

ÍNDICE GENERAL

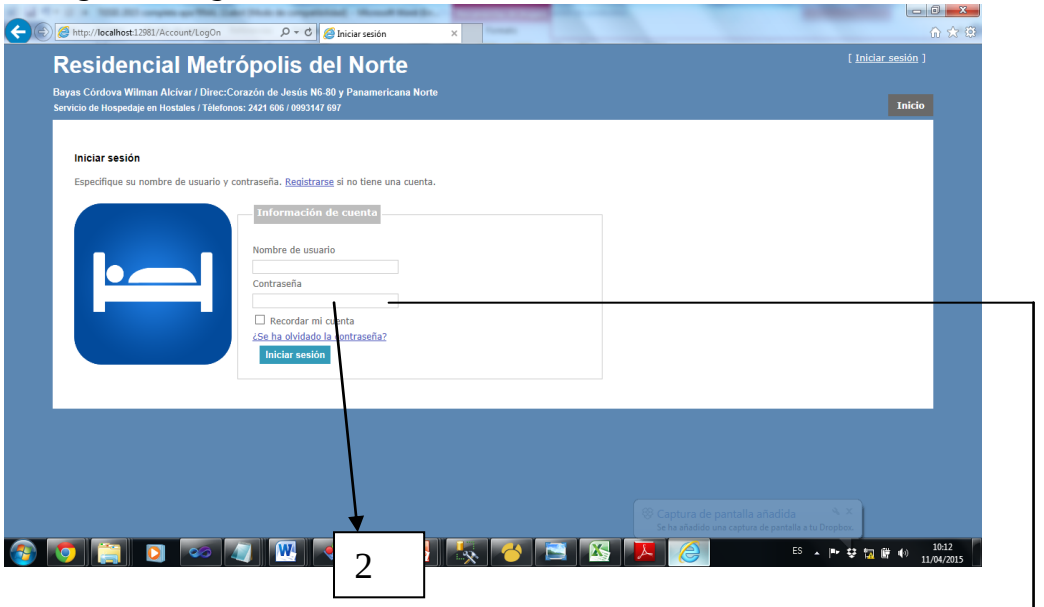
Título **Página**

INDICE DE FIGURAS

Figura **Página**

2.1 Ingreso del Sistema

Figura 1: *Ingreso del sistema*

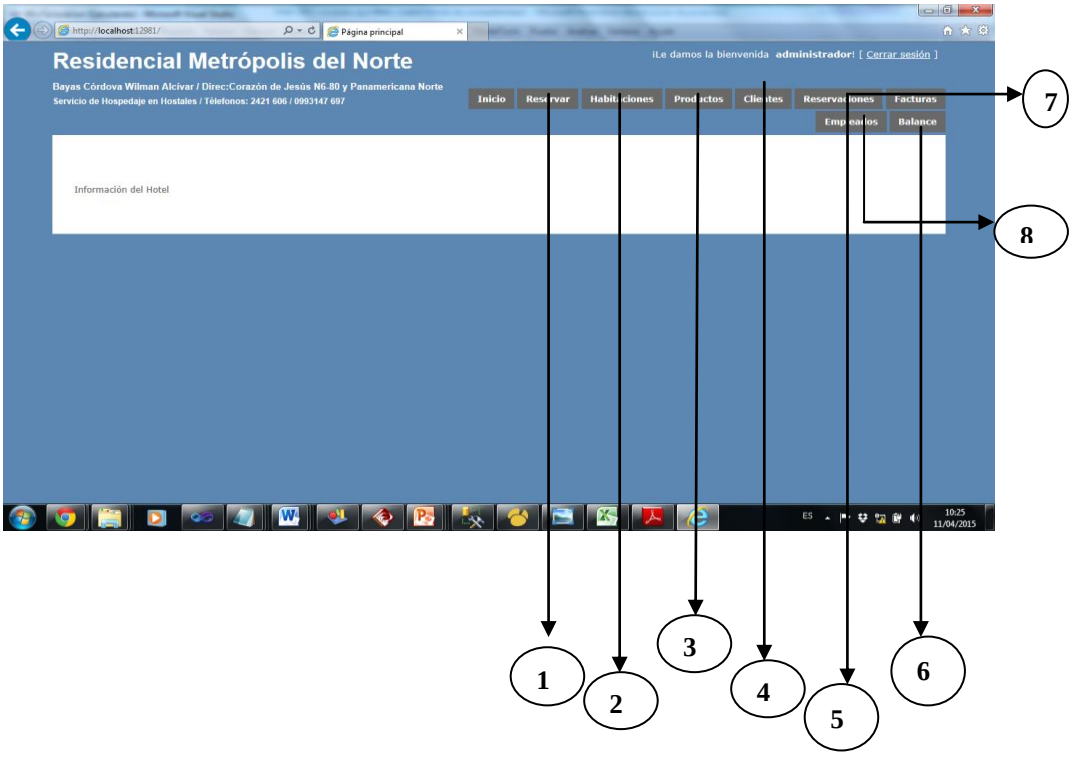


1.-Ingresar al sistema con el nombre de usuario

2.-Ingresar de la contraseña y ENTER

2.2 Interfaz de Trabajo

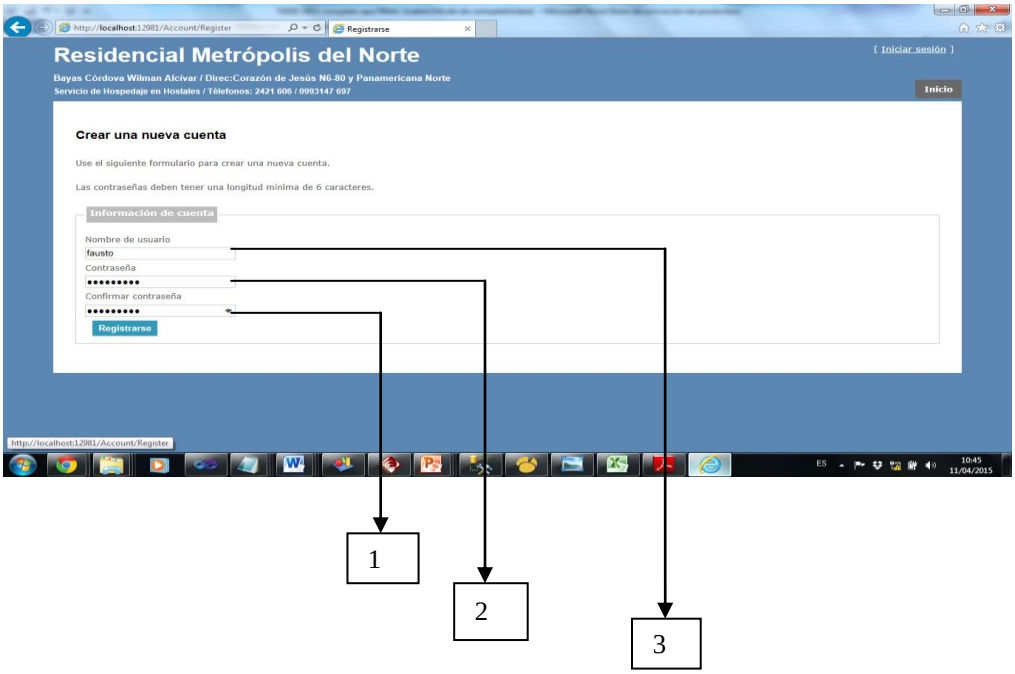
Figura 2: Muestra los Módulos



1. Menú de Reservar una reservas
2. Menú de habitaciones.
3. Menú de productos.
4. Menú de clientes registrado.
5. Listado de reservaciones.
6. Balance reportes de ventas.
7. Facturas por cobrar y cobradas visualización del XML.
8. Modulo Empleados registrar, eliminar, modificar.

2.3 Registrarse Cliente por primera vez mediante la web

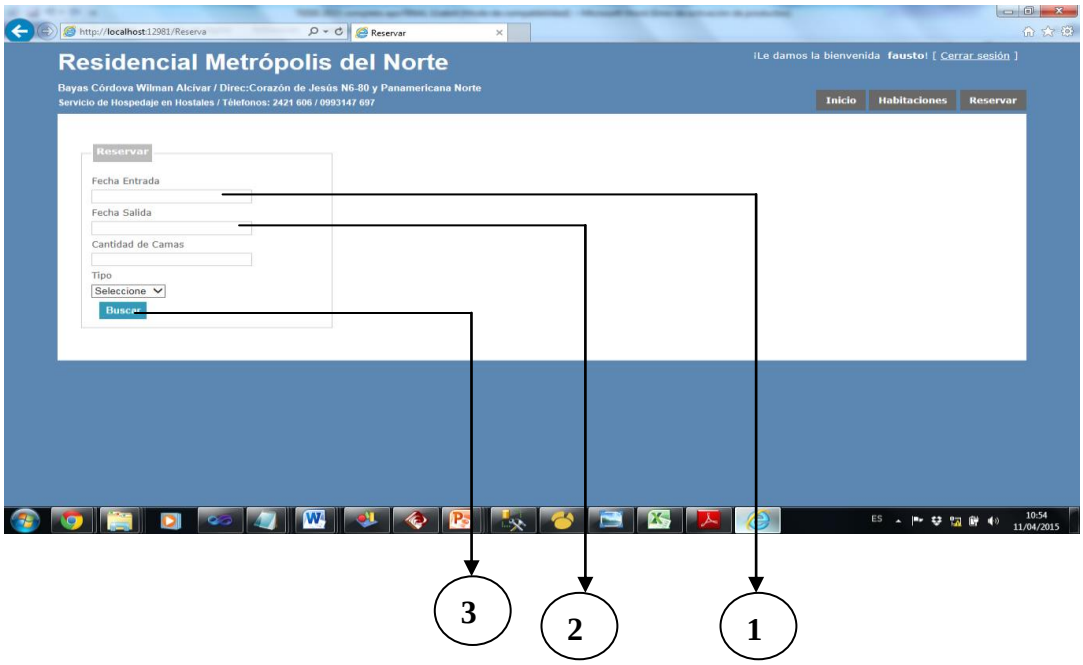
Figura 3: Registrarse como cliente por primera vez al sistema.



1. Escribo nombre de usuario que desee
2. Ingreso contraseña mínimo 6 caracteres.
3. Confirmo contraseña y luego click en registrarse

2.4 Realizar reserva mediante la web.

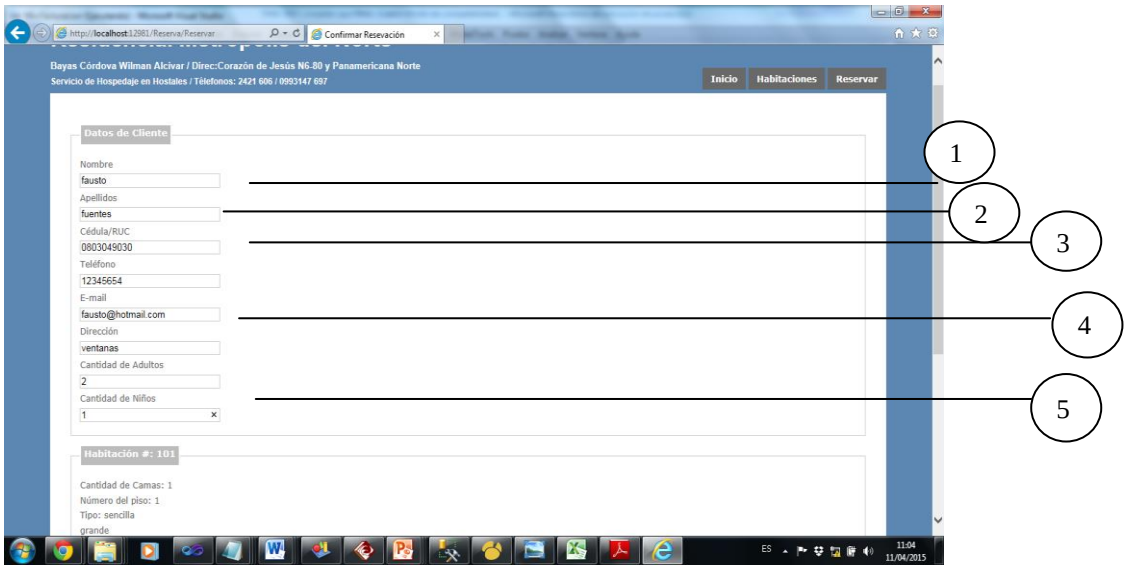
Figura 4: Reservar



1. Selecciono fecha de entrada
2. Selecciono fecha de salida.
3. Busco la disponibilidad de habitaciones.

2.5 Confirmar reservación

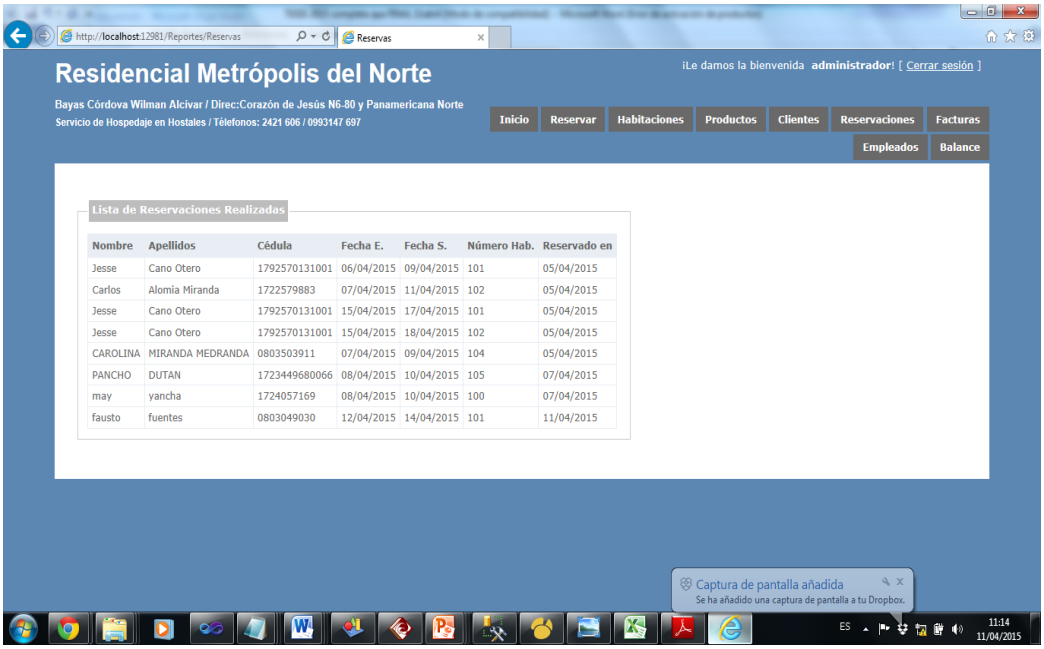
Figura 5: Ingresar datos para su respectiva reservación.



1. Ingreso nombres
2. Ingreso apellidos
3. Identificación
4. Correo
5. Cantidad de niños y adultos a hospedarse.

2.6 Reporte de reservas realizadas.

Figura 6: Listado de reservas



Muestra el listado de los clientes registrados para su respectiva reserva.

A.04 Manual Técnico

ÍNDICE GENERAL

Titulo **Página**

3.2 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
Prod	Prod_id	int	NULL	4	NO	SI
precio	prec	float	NULL	4	SI	NO
cantidad	cant	int	NULL	4	SI	NO
Nom_pro	prod	varchar	NULL	4	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	rol	varchar	NULL	50	SI	NO
reserva	Id_reserva	int	NULL	4	NO	SI
Fecha_r	id_fec_r	datetime	NULL	4	SI	NO
Fecha_E	Fec_s	datetime	NULL	4	SI	NO
Fecha_S	Fec_s	datetime	NULL	4	SI	NO
Factura	id_factura	int	NULL	4	NO	SI
Factura	id_cliente	int	NULL	4	SI	NO
Factura	id_reserva	varchar	NULL	50	SI	NO
Factura	fecha	datetime	NULL	8	SI	NO
Factura	estado	varchar	NULL	50	SI	NO
Factura	producto	real	NULL	4	SI	NO

Habitaciones	Hab_id	int	NULL	4	SI	NO
Hab_tipo	Tipo_hab	int	NULL	4	NO	SI
cant	cant	int	NULL	8	SI	NO
piso	piso	int	NULL	16	SI	NO
precio	precio	float	53	8	SI	NO
reserva	id_resrve	int	NULL	4	NO	SI
reserva	nombre	varchar	NULL	50	SI	NO
reserva	apellido	varchar	NULL	4	SI	NO
reserva	telefono	int	NULL	4	SI	NO
reserva	email	varchar	NULL	4	SI	NO
reserva	direccion	varchar	NULL	16	SI	NO
reserva	Num_hab	int	53	8	SI	NO
reserva	Reservado_en	Datetim	NULL	4	NO	SI
balance	id_balance	int	NULL	4	SI	NO
Fecha_d	id_comp	int	NULL	4	SI	NO
Fecha_h	id_prov_prod	int	NULL	4	NO	SI

3.3 Script de la Base de Datos

```
/****** Object:  Table [dbo].[bdfacturacion1]    Script Date: 03/12/2015
17:29:55 ******/
IF  EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[bdfacturacion]') AND type in (N'U'))
DROP TABLE [dbo].[bdfacturacion]
GO
/****** Object:  Table [dbo].[Cliente]    Script Date: 03/12/2015
17:29:55 ******/
IF  EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[Cliente]') AND type in (N'U'))
DROP TABLE [dbo].[Cliente]
GO
/****** Object:  Table [dbo].[Factura]    Script Date: 03/12/2015
17:29:55 ******/
IF  EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[Factura]') AND type in (N'U'))
DROP TABLE [dbo].[Factura]
GO
/****** Object:  Table [dbo].[Habitacion]    Script Date: 03/12/2015
17:29:55 ******/
IF  EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[Habitacion]') AND type in (N'U'))
DROP TABLE [dbo].[Habitacion]
GO
```

```
/****** Object:  Table [dbo].[Reserva]    Script Date: 03/12/2015
17:29:55 ******/
IF  EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[Reserva]') AND type in (N'U'))
DROP TABLE [dbo].[Reserva]
GO
/****** Object:  Table [dbo].[ResSer]    Script Date: 03/12/2015
17:29:55 ******/
IF  EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[ResSer]') AND type in (N'U'))
DROP TABLE [dbo].[ResSer]
GO
/****** Object:  Table [dbo].[Servicio]    Script Date: 03/12/2015
17:29:55 ******/
IF  EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[Servicio]') AND type in (N'U'))
DROP TABLE [dbo].[Servicio]
GO
/****** Object:  Table [dbo].[Servicio]    Script Date: 03/12/2015
17:29:55 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
```

```
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[Servicio]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[Servicio](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [nombre] [varchar](50) COLLATE
SQL_Latin1_General_CP1_CI_AS NULL,
    [descripcion] [text] COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
    [precio] [real] NOT NULL,
 CONSTRAINT [PK_Servicio] PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON,
ALLOW_PAGE_LOCKS  = ON)
)
END
GO
SET IDENTITY_INSERT [dbo].[Servicio] ON
INSERT [dbo].[Servicio] ([id], [nombre], [descripcion], [precio])
VALUES (1, N'Spa', N'por seccion es el precio', 50)
INSERT [dbo].[Servicio] ([id], [nombre], [descripcion], [precio])
VALUES (2, N'gim', N'x dias es el precio', 23)
```

```
INSERT [dbo].[Servicio] ([id], [nombre], [descripcion], [precio])
VALUES (3, N'almuerzo', NULL, 12)
SET IDENTITY_INSERT [dbo].[Servicio] OFF
/****** Object:  Table [dbo].[ResSer]    Script Date: 03/12/2015
17:29:55 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[ResSer]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[ResSer](
    [idS] [int] NOT NULL,
    [idR] [int] NOT NULL,
    [idRS] [int] IDENTITY(1,1) NOT NULL,
 CONSTRAINT [PK_ResSer] PRIMARY KEY CLUSTERED
(
    [idRS] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON,
ALLOW_PAGE_LOCKS  = ON)
)
END
GO
```

```
SET IDENTITY_INSERT [dbo].[ResSer] ON
INSERT [dbo].[ResSer] ([idS], [idR], [idRS]) VALUES (1, 5, 6)
INSERT [dbo].[ResSer] ([idS], [idR], [idRS]) VALUES (2, 5, 7)
INSERT [dbo].[ResSer] ([idS], [idR], [idRS]) VALUES (1, 6, 8)
INSERT [dbo].[ResSer] ([idS], [idR], [idRS]) VALUES (2, 6, 9)
INSERT [dbo].[ResSer] ([idS], [idR], [idRS]) VALUES (1, 7, 10)
INSERT [dbo].[ResSer] ([idS], [idR], [idRS]) VALUES (2, 7, 11)
INSERT [dbo].[ResSer] ([idS], [idR], [idRS]) VALUES (1, 8, 12)
INSERT [dbo].[ResSer] ([idS], [idR], [idRS]) VALUES (2, 8, 13)
SET IDENTITY_INSERT [dbo].[ResSer] OFF
/****** Object:  Table [dbo].[Reserva]    Script Date: 03/12/2015
17:29:55 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[Reserva]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[Reserva](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [fecha_r] [datetime] NOT NULL,
    [fecha_e] [datetime] NOT NULL,
    [fecha_s] [datetime] NOT NULL,
```

```
    [numero_hab] [nchar](10) COLLATE
SQL_Latin1_General_CP1_CI_AS NOT NULL,
    [idC] [int] NOT NULL,
 CONSTRAINT [PK_Reserva] PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON,
ALLOW_PAGE_LOCKS  = ON)
)
END
GO
SET IDENTITY_INSERT [dbo].[Reserva] ON
INSERT [dbo].[Reserva] ([id], [fecha_r], [fecha_e], [fecha_s],
[numero_hab], [idC]) VALUES (3, CAST(0x0000A4560133C588 AS
DateTime), CAST(0x0000A45700000000 AS DateTime),
CAST(0x0000A45900000000 AS DateTime), N'124       ', 2)
INSERT [dbo].[Reserva] ([id], [fecha_r], [fecha_e], [fecha_s],
[numero_hab], [idC]) VALUES (4, CAST(0x0000A456013475F2 AS
DateTime), CAST(0x0000A45700000000 AS DateTime),
CAST(0x0000A45B00000000 AS DateTime), N'126       ', 2)
INSERT [dbo].[Reserva] ([id], [fecha_r], [fecha_e], [fecha_s],
[numero_hab], [idC]) VALUES (5, CAST(0x0000A456013549EA AS
DateTime), CAST(0x0000A45700000000 AS DateTime),
CAST(0x0000A45900000000 AS DateTime), N'715       ', 2)
```

```
INSERT [dbo].[Reserva] ([id], [fecha_r], [fecha_e], [fecha_s],
[numero_hab], [idC]) VALUES (6, CAST(0x0000A4560139B5A0 AS
DateTime), CAST(0x0000A45900000000 AS DateTime),
CAST(0x0000A45A00000000 AS DateTime), N'125       ', 2)
INSERT [dbo].[Reserva] ([id], [fecha_r], [fecha_e], [fecha_s],
[numero_hab], [idC]) VALUES (7, CAST(0x0000A456013ABEF2 AS
DateTime), CAST(0x0000A46100000000 AS DateTime),
CAST(0x0000A46800000000 AS DateTime), N'124       ', 2)
INSERT [dbo].[Reserva] ([id], [fecha_r], [fecha_e], [fecha_s],
[numero_hab], [idC]) VALUES (8, CAST(0x0000A456013C35C8 AS
DateTime), CAST(0x0000A45C00000000 AS DateTime),
CAST(0x0000A45E00000000 AS DateTime), N'124       ', 2)
INSERT [dbo].[Reserva] ([id], [fecha_r], [fecha_e], [fecha_s],
[numero_hab], [idC]) VALUES (9, CAST(0x0000A45700F6F9E0 AS
DateTime), CAST(0x0000A45800000000 AS DateTime),
CAST(0x0000A46500000000 AS DateTime), N'139       ', 1)
SET IDENTITY_INSERT [dbo].[Reserva] OFF
/****** Object:  Table [dbo].[Habitacion]    Script Date: 03/12/2015
17:29:55 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[Habitacion]') AND type in (N'U'))
```

```
BEGIN
CREATE TABLE [dbo].[Habitacion](
    [numero] [nvarchar](50) COLLATE
SQL_Latin1_General_CP1_CI_AS NOT NULL,
    [tipo] [nvarchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
    [cantCamas] [int] NOT NULL,
    [descripcion] [nvarchar](max) COLLATE
SQL_Latin1_General_CP1_CI_AS NULL,
    [precio] [float] NOT NULL,
 CONSTRAINT [PK_Habitacion] PRIMARY KEY CLUSTERED
(
    [numero] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON,
ALLOW_PAGE_LOCKS  = ON)
)
END
GO
INSERT [dbo].[Habitacion] ([numero], [tipo], [cantCamas],
[descripcion], [precio]) VALUES (N'124', N'duplex', 2, N'mi hotel en
duplex', 50)
INSERT [dbo].[Habitacion] ([numero], [tipo], [cantCamas],
[descripcion], [precio]) VALUES (N'125', N'matrimonio', 1, NULL, 60)
```

```
INSERT [dbo].[Habitacion] ([numero], [tipo], [cantCamas],
[descripcion], [precio]) VALUES (N'126', N'Sencilla', 1, NULL, 35)
INSERT [dbo].[Habitacion] ([numero], [tipo], [cantCamas],
[descripcion], [precio]) VALUES (N'139', N'Matrimonial', 1, NULL, 67)
INSERT [dbo].[Habitacion] ([numero], [tipo], [cantCamas],
[descripcion], [precio]) VALUES (N'715', N'Sencilla', 1, N'Buena
Habitacion', 100)
/****** Object:  Table [dbo].[Factura]    Script Date: 03/12/2015
17:29:55 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[Factura]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[Factura](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [idR] [int] NOT NULL,
    [estado] [int] NOT NULL,
    [numero] [varchar](50) COLLATE
SQL_Latin1_General_CP1_CI_AS NOT NULL,
 CONSTRAINT [PK_Factura] PRIMARY KEY CLUSTERED
(
    [id] ASC
```

```
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON,
ALLOW_PAGE_LOCKS  = ON)
)
END
GO
SET IDENTITY_INSERT [dbo].[Factura] ON
INSERT [dbo].[Factura] ([id], [idR], [estado], [numero]) VALUES (3, 5,
0, N'5_2015')
INSERT [dbo].[Factura] ([id], [idR], [estado], [numero]) VALUES (4, 6,
1, N'6_2015')
INSERT [dbo].[Factura] ([id], [idR], [estado], [numero]) VALUES (5, 7,
0, N'7_2015')
INSERT [dbo].[Factura] ([id], [idR], [estado], [numero]) VALUES (6, 8,
0, N'8_2015')
INSERT [dbo].[Factura] ([id], [idR], [estado], [numero]) VALUES (7, 9,
0, N'9_2015')
SET IDENTITY_INSERT [dbo].[Factura] OFF
/****** Object:  Table [dbo].[Cliente]    Script Date: 03/12/2015
17:29:55 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
```

```
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[Cliente]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[Cliente](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [Nombre] [varchar](50) COLLATE
SQL_Latin1_General_CP1_CI_AS NULL,
    [Apelllidos] [varchar](50) COLLATE
SQL_Latin1_General_CP1_CI_AS NULL,
    [usuario] [nvarchar](50) COLLATE
SQL_Latin1_General_CP1_CI_AS NOT NULL,
    [pass] [nvarchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS
NOT NULL,
    [identificacion] [nvarchar](50) COLLATE
SQL_Latin1_General_CP1_CI_AS NULL,
    [role] [varchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
    [telef] [varchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
    [mail] [varchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
    [dir] [varchar](50) COLLATE SQL_Latin1_General_CP1_CI_AS
NULL,
 CONSTRAINT [PK_Cliente] PRIMARY KEY CLUSTERED
(
```

```
    [id] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON,
ALLOW_PAGE_LOCKS  = ON)
)
END
GO
SET IDENTITY_INSERT [dbo].[Cliente] ON
INSERT [dbo].[Cliente] ([id], [Nombre], [Apelllidos], [usuario], [pass],
[identificacion], [role], [telef], [mail], [dir]) VALUES (1, N'marlon',
N'remedios', N'Administrador', N'blister', N'1722579883', N'admin',
N'222', N'mjr@gmail.com', N'fsfdf')
INSERT [dbo].[Cliente] ([id], [Nombre], [Apelllidos], [usuario], [pass],
[identificacion], [role], [telef], [mail], [dir]) VALUES (2, N'mayra',
N'Sehara', Nmayral', N'mayra', N'1722579883', N'members',
N'099999999', N'yhdrigs@gmail.com', N'Canada N167')
INSERT [dbo].[Cliente] ([id], [Nombre], [Apelllidos], [usuario], [pass],
[identificacion], [role], [telef], [mail], [dir]) VALUES (3, N'Marlon
Jorge', N'Remedios Gonzalez', N'jessedaniel', N'blister', N'1234567',
N'members', NULL, NULL, NULL)
SET IDENTITY_INSERT [dbo].[Cliente] OFF
/****** Object:  Table [dbo].[bdfacturacion1]    Script Date: 03/12/2015
17:29:55 ******/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[bdfacturacion]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[bdfacturacion](
    [Column 0] [nvarchar](50) COLLATE
SQL_Latin1_General_CP1_CI_AS NULL
)
END
GO
```

3.4 Código de Reserva

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Xml.Linq;
using System.Web;
using System.Web.Mvc;
using MvcFacturacion.Models;
using System.Globalization;
using System.Text.RegularExpressions;
using iTextSharp;
using iTextSharp.text;
using iTextSharp.text.api;
using iTextSharp.text.pdf;
using iTextSharp.text.factories;
using iTextSharp.text.html;
using iTextSharp.text.xml;
using iTextSharp.text.exceptions;
using iTextSharp.text.error_messages;
using System.IO;

namespace MvcFacturacion.Controllers
{
   [Authorize(Roles = "members")]
   public class ReservaController : Controller
   {
      facturacionEntities1 dbH = new facturacionEntities1();
      Servicios dbS = new Servicios();
      ClientesContext dbC = new ClientesContext();
      Reservas dbR = new Reservas();
      ResSers dbRS = new ResSers();
      Facturas dbF = new Facturas();
      bool invalid = false;

      public bool IsValidEmail(string strIn)
      {
         invalid = false;
         if (String.IsNullOrEmpty(strIn))
            return false;

         // Use IdnMapping class to convert Unicode domain names.
         try
         {

            strIn = Regex.Replace(strIn, @"(@)(.+)$", this.DomainMapper,
                              RegexOptions.None);
         }
         catch
```

```
            {
                return false;
            }

            if (invalid)
                return false;

            // Return true if strIn is in valid e-mail format.
            try
            {
                return Regex.IsMatch(strIn,
                      @"^(?(""")(""".+?(?<!\\)""@)|(([0-9a-z]((\.(?!\.))|[-
!#\$%&'\*\+/=\?\^`\{\}\|~\w])*)(?<=[0-9a-z])@))" +
                      @"(?(\[)(\[(\d{1,3}\.){3}\d{1,3}\])|(([0-9a-z][-\w]*[0-9a-z]*\.)+[a-z0-
9][\-a-z0-9]{0,22}[a-z0-9]))$",
                      RegexOptions.IgnoreCase);
            }
            catch
            {
                return false;
            }
        }

        private string DomainMapper(Match match)
        {
            // IdnMapping class with default property values.
            IdnMapping idn = new IdnMapping();

            string domainName = match.Groups[2].Value;
            try
            {
                domainName = idn.GetAscii(domainName);
            }
            catch (ArgumentException)
            {
                invalid = true;
            }
            return match.Groups[1].Value + domainName;
        }
        //
        // GET: /Reserva/

        public ActionResult Index()
        {
            ViewData["errores"] = new List<string>();
            if (TempData["FC"] == null)
            {
                var tipos = dbH.ExecuteStoreQuery<String>("SELECT h.tipo FROM
Habitacion h GROUP BY h.tipo").ToList();
```

```
            tipos.Insert(0, "Seleccione");
            if(TempData["errores"] != null)
                ViewData["errores"] = TempData["errores"];
            TempData["errores"] = null;
            return View(tipos);
        }
        else if (TempData["AD"] == null)
        {
            try
            {
                TempData["EF"] =
this.AuxiliarReservar((FormCollection)TempData["FC"]);
                TempData["FC"] = null;
                return RedirectToAction("doReserva");
            }
            catch
            {
                return View();
            }
        }
        try
        {
            TempData["EF"] =
this.AuxiliarReservar((FormCollection)TempData["FC"],
(List<Servicio>)TempData["AD"]);
            TempData["FC"] = null;
            TempData["AD"] = null;
            return RedirectToAction("doReserva");
        }
        catch
        {
            return View();
        }

    }

    public bool isOccupied(List<Reserva> ress, Habitacion hab, DateTime fi,
DateTime ff)
    {
        for (int i = 0; i < ress.Count; i++)
            if (ress[i].numero_hab.Contains(hab.numero))
                if((fi >= ress[i].fecha_e && fi <= ress[i].fecha_s) || (ff >= ress[i].fecha_e
&& fi <= ress[i].fecha_s))
                    return true;
        return false;
    }

    public EfectuarReservaDatos AuxiliarReservar(FormCollection fc)
    {
```

```
            string fi = fc["TestDatePicker1"];
            string ff = fc["TestDatePicker2"];
            int camas= 0;
            if(!fc["camas"].Equals(""))
               camas=int.Parse(fc["camas"]);
            int tipo = int.Parse(fc["tipos"]);
            string format = "MM/dd/yyyy";
            DateTime di =
DateTime.ParseExact(fi,format,CultureInfo.InvariantCulture);
            DateTime df = DateTime.ParseExact(ff, format,
CultureInfo.InvariantCulture);
            List<Reserva> ress = (from s in dbR.Reserva select s).ToList();
            List<Habitacion> habs = (from s in dbH.Habitacion select s).ToList();
            List<Servicio> servs = (from s in dbS.Servicio select s).ToList();
            List<Habitacion> habsD = new List<Habitacion>();
            List<String> tipos = dbH.ExecuteStoreQuery<String>("SELECT h.tipo
FROM Habitacion h GROUP BY h.tipo").ToList();
            tipos.Insert(0, "Seleccione");
            for (int i = 0; i < habs.Count;i++ )
               if (!this.isOccupied(ress, habs[i], di, df) && (habs[i].cantCamas ==
camas || camas == 0) && (habs[i].tipo.Equals(tipos[tipo]) || tipo==0 ))
                  habsD.Add(habs[i]);
            List<double> precios = new List<double>();
            for (int i = 0; i < habsD.Count; i++)
               precios.Add(habsD[i].precio * (df - di).Days);
            List<bool> servsAd = new List<bool>();
            for (int i = 0; i < servs.Count; i++)
               servsAd.Add(false);
            EfectuarReservaDatos ef = new EfectuarReservaDatos();
            ef.tipos = tipos;
            ef.fc = fc;
            ef.HD = habsD;
            ef.servicios = servs;
            ef.precios = precios;
            ef.servsADD = servsAd;
            return ef;
         }

         public bool isAdded(List<Servicio> servs, Servicio serv)
         {
            for (int i = 0; i < servs.Count; i++)
               if (servs[i].nombre.Equals(serv.nombre))
                  return true;
            return false;
         }

         public EfectuarReservaDatos AuxiliarReservar(FormCollection fc,
List<Servicio> servsAdd)
```

```
        {
            string fi = fc["TestDatePicker1"];
            string ff = fc["TestDatePicker2"];
            int camas = 0;
            if (!fc["camas"].Equals(""))
                camas = int.Parse(fc["camas"]);
            int tipo = int.Parse(fc["tipos"]);
            string format = "MM/dd/yyyy";
            DateTime di = DateTime.ParseExact(fi, format,
CultureInfo.InvariantCulture);
            DateTime df = DateTime.ParseExact(ff, format,
CultureInfo.InvariantCulture);
            List<Reserva> ress = (from s in dbR.Reserva select s).ToList();
            List<Habitacion> habs = (from s in dbH.Habitacion select s).ToList();
            List<Servicio> servs = (from s in dbS.Servicio select s).ToList();
            List<Habitacion> habsD = new List<Habitacion>();
            List<String> tipos = dbH.ExecuteStoreQuery<String>("SELECT h.tipo
FROM Habitacion h GROUP BY h.tipo").ToList();
            tipos.Insert(0, "Seleccione");
            for (int i = 0; i < habs.Count; i++)
                if (!this.isOccupied(ress, habs[i], di, df) && (habs[i].cantCamas == camas
|| camas == 0) && (habs[i].tipo.Equals(tipos[tipo]) || tipo == 0))
                    habsD.Add(habs[i]);
            double contPS = 0;
            for (int i = 0; i < servsAdd.Count; i++)
                contPS += servsAdd[i].precio;
            List<double> precios = new List<double>();
            for (int i = 0; i < habsD.Count; i++)
                precios.Add(habsD[i].precio*(df - di).Days + contPS);
            List<bool> servsAd = new List<bool>();
            for (int i = 0; i < servs.Count; i++)
                if(this.isAdded(servsAdd,servs[i]))
                    servsAd.Add(true);
                else
                    servsAd.Add(false);
            EfectuarReservaDatos ef = new EfectuarReservaDatos();
            ef.tipos = tipos;
            ef.fc = fc;
            ef.HD = habsD;
            ef.servicios = servs;
            ef.precios = precios;
            ef.servsADD = servsAd;
            return ef;
        }
        //
        // GET: /Reserva/
        [HttpPost]
        public ActionResult Index(FormCollection fc)
        {
```

```
            List<string> err = this.IsValidate(fc);

            if (err.Count == 0)
            {
                try
                {
                    TempData["EF"] = this.AuxiliarReservar(fc);
                    return RedirectToAction("doReserva");
                }
                catch
                {
                    return View();
                }
            }
            else {
                ViewData["errores"] = err;
                var tipos = dbH.ExecuteStoreQuery<String>("SELECT h.tipo FROM
Habitacion h GROUP BY h.tipo").ToList();
                tipos.Insert(0, "Seleccione");
                return View(tipos);
            }

        }

        //
        // GET: /Reserva/doReserva
        public ActionResult doReserva()
        {
            EfectuarReservaDatos ef = (EfectuarReservaDatos)TempData["EF"];
            return View(ef);
        }


        // GET: /Reserva/doReserva
        [HttpPost]
        public ActionResult doReserva(FormCollection fc)
        {
            try
            {
                var formCollection = new FormCollection(HttpContext.Request.Form);
                if(HttpContext.Request.Form.AllKeys[4].Equals("ButtonB"))
                    {
                        TempData["FC"] = formCollection;
                        List<String> errs = this.IsValidate(fc);
                        if (errs.Count == 0)
                            return RedirectToAction("Index");
                        else {
                            TempData["FC"] = null;
```

```
                TempData["errores"] = errs;
                return RedirectToAction("Index");
            }


        }
    else if
(HttpContext.Request.Form.AllKeys[HttpContext.Request.Form.AllKeys.Count() -
1].Equals("ButtonS"))
    {
        List<Servicio> servs = (from s in dbS.Servicio select s).ToList();
        List<Servicio> servsAdd = new List<Servicio>();
        for (int i = 0; i < servs.Count; i++)
            if (fc[servs[i].nombre].Contains("true"))
                servsAdd.Add(servs[i]);
        TempData["FC"] = fc;
        TempData["AD"] = servsAdd;
        return RedirectToAction("Index");
    }
    string numH= HttpContext.Request.Form.AllKeys[4].Split('_')[1];
    TempData["FC"] = formCollection;
    TempData["HA"] = numH;
    return RedirectToAction("Reservar");
}
catch
{
    return View();
}
}
//
// GET: /Reserva/Reservar/
//
public ActionResult Reservar()
{
    bool flag = false;
    List<string> errores = new List<string>();
    if (TempData["errores"] == null)
        ViewData["errores"] = errores;
    else
    {
        ViewData["errores"] = TempData["errores"];
        flag = true;
    }
    FormCollection fc = (FormCollection)TempData["FC"];
    String numH = (string)TempData["HA"];
    TempData["FC"] = null;
    TempData["HA"] = null;
    Habitacion hab = (from s in dbH.Habitacion where s.numero.Equals(numH)
select s).FirstOrDefault();
```

```
            List<Servicio> servs = (from s in dbS.Servicio select s).ToList();
            List<Servicio> servsAdd = new List<Servicio>();
            double suma = 0;
            for (int i = 0; i < servs.Count; i++)
                if (fc[servs[i].nombre]!= null)
                    if (fc[servs[i].nombre].Contains("true"))
                    {
                        servsAdd.Add(servs[i]);
                        suma += servs[i].precio;
                    }
            string fi = fc["TestDatePicker1"];
            string ff = fc["TestDatePicker2"];
            string format = "MM/dd/yyyy";
            DateTime di;
            DateTime df;
            if (!flag)
            {
                di = DateTime.ParseExact(fi, format, CultureInfo.InvariantCulture);
                df = DateTime.ParseExact(ff, format, CultureInfo.InvariantCulture);
            }
            else {
                di = DateTime.Parse(fi);
                df = DateTime.Parse(ff);
            }

            var cliente1 = (from s in dbC.Cliente
                            where
s.usuario.Equals(System.Web.HttpContext.Current.User.Identity.Name)
                            select s).FirstOrDefault();
            Reservar r = new Reservar();
            r.hab = hab;
            r.servsADD = servsAdd;
            r.fi = di;
            r.ff = df;
            r.precioT = suma;
            r.cli = cliente1;
            return View(r);
        }


        // GET: /Reserva/Reservar
        [HttpPost]
        public ActionResult Reservar(FormCollection fc)
        {
            try
            {
                List<string> errores = new List<string>();
                errores = this.ConfirmValidate(fc);
                if (errores.Count == 0)
```

```
            {
                dbC.ContextOptions.LazyLoadingEnabled = false;
                var cliente1 = (from s in dbC.Cliente
                                where
s.usuario.Equals(System.Web.HttpContext.Current.User.Identity.Name)
                                select s).FirstOrDefault();
                Reserva res = new Reserva();
                res.fecha_e = DateTime.Parse(fc["TestDatePicker1"]);
                res.fecha_s = DateTime.Parse(fc["TestDatePicker2"]);
                res.fecha_r = DateTime.Now;
                res.numero_hab = fc["numH"];
                res.idC = cliente1.id;
                res.cantA = int.Parse(fc["adultos"]);
                res.cantN = int.Parse(fc["ninnos"]);
                dbR.AddToReserva(res);
                dbR.SaveChanges();
                cliente1.Nombre = fc["nombre"];
                cliente1.Apelllidos = fc["apellidos"];
                cliente1.identificacion = fc["cedula"];
                cliente1.telef = fc["telf"];
                cliente1.mail = fc["mail"];
                cliente1.dir = fc["dir"];
                dbC.SaveChanges();
                for (int i = 12; i < fc.AllKeys.Count(); i++)
                {
                    ResSer rs = new ResSer();
                    rs.idR = res.id;
                    rs.idS = int.Parse(fc[fc.AllKeys[i]].Split('_')[0]);
                    dbRS.AddToResSer(rs);
                    dbRS.SaveChanges();
                }
                Factura fac = new Factura();
                fac.idR = res.id;
                fac.estado = 0;
                fac.numero = res.id + "_" + DateTime.Now.Year;
                dbF.AddToFactura(fac);
                dbF.SaveChanges();

                this.GenerarPdf(res.id, fac);
                return RedirectToAction("ReservaRealizada");
            }
            else {


                TempData["FC"] = fc;
                TempData["HA"] = fc["numH"];
                TempData["errores"] = errores;
                return RedirectToAction("Reservar");
            }
```

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

        private List<string> ConfirmValidate(FormCollection fc) {
            List<string> errores = new List<string>();
            if (Convert.ToString(fc["nombre"]) == "" || Convert.ToString(fc["apellidos"])
== "" || Convert.ToString(fc["cedula"]) == "" || Convert.ToString(fc["telf"]) == "" ||
Convert.ToString(fc["mail"]) == "" || Convert.ToString(fc["dir"]) == "" ||
Convert.ToString(fc["adultos"]) == "" || Convert.ToString(fc["ninnos"]) == "")
            {
                errores.Add("Los campos Nombre, Apellido, Cédula, Teléfono, E-mail y
Dirección son obligatorios.");
            }
            if (fc["nombre"].Any(Char.IsDigit))
                errores.Add("El campo Nombre solo admite letras");
            if (fc["apellidos"].Any(Char.IsDigit))
                errores.Add("El campo Apellidos solo admite letras");
            if (!fc["telf"].All(Char.IsDigit))
                errores.Add("El campo Teléfono solo admite números");
            if (!this.IsValidEmail(fc["mail"])&&!fc["mail"].Equals(""))
                errores.Add("El E-mail entrado tiene un formato incorrecto");
            if (!fc["adultos"].All(Char.IsDigit))
                errores.Add("El campo Cantidad de Adultos solo admite números");
            if (!fc["ninnos"].All(Char.IsDigit))
                errores.Add("El campo Cantidad de Niños solo admite números");
            string errorMessageTemplate;
            if (!this.IsValidRUC(Convert.ToString(fc["cedula"]), out
errorMessageTemplate))
                errores.Add(errorMessageTemplate);
            return errores;

        }

        public bool IsValidRUC(string ruc, out string errorMessageTemplate)
        {

            if (string.IsNullOrEmpty(ruc))
            {
                errorMessageTemplate = "Ruc no Ingresado.";
                return false;
            }
            else
            {

                string cioriginal = ruc;
```

```
//string cisindigito = ciruc.Substring(0, 9);
int suma = 0;
int residuo = 0;
bool priva = false;
bool pub = false;
bool nat = false;
//int numprovi = 24;
int modulo = 11;
int digveri;

int d1,d2,d3,d4,d5,d6,d7,d8,d9,d10;
int p1,p2,p3,p4,p5,p6,p7,p8,p9;

// Verifico que el campo no contenga letras
for (int i = 0; i < cioriginal.Length; i++)
{
   //var n = int.Parse(cioriginal.ToCharArray(i));
   if (!char.IsDigit(cioriginal[i]))
   {
      errorMessageTemplate = "Ruc o C.I. Incorrecto";
      return false;
   }

}

if (cioriginal.Length < 10)
{
   errorMessageTemplate = "Ruc o C.I. Incorrecto";
   return false;
}
else
{
   string id = cioriginal.Substring(0,10);
   if (id == "0000000000")      //No validado en los cálculos
   {
      errorMessageTemplate = "Ruc o C.I. Incorrecto";
      return false;
   }
   else if (id == "9999999999")  //Válido Para usuario final
   {
      errorMessageTemplate = "Número C.I. Correcto";
      return true;
   }

   // almacenamos los digitos de la cedula
   d1 = int.Parse(cioriginal.Substring(0, 1));
   d2 = int.Parse(cioriginal.Substring(1, 1));
   d3 = int.Parse(cioriginal.Substring(2, 1));
   d4 = int.Parse(cioriginal.Substring(3, 1));
```

```
d5 = int.Parse(cioriginal.Substring(4, 1));
d6 = int.Parse(cioriginal.Substring(5, 1));
d7 = int.Parse(cioriginal.Substring(6, 1));
d8 = int.Parse(cioriginal.Substring(7, 1));
d9 = int.Parse(cioriginal.Substring(8, 1));
d10 = int.Parse(cioriginal.Substring(9, 1));

/* El tercer digito es:
   9 Para sociedades y extranjeras
   6 Para sociedades publicas
   menor que 6 para personas naturales*/
if (d3 == 7 || d3 == 8)
{
   errorMessageTemplate = "Ruc o C.I. Incorrecto";
   return false;
}
else if (d3 < 6)
{
   nat = true;
   p1 = d1 * 2;
   if (p1 >= 10)
   {
      p1 -= 9;
   }
   p2 = d2 * 1;
   if (p2 >= 10)
   {
      p2 -= 9;
   }
   p3 = d3 * 2;
   if (p3 >= 10)
   {
      p3 -= 9;
   }
   p4 = d4 * 1;
   if (p4 >= 10)
   {
      p4 -= 9;
   }
   p5 = d5 * 2;
   if (p5 >= 10)
   {
      p5 -= 9;
   }
   p6 = d6 * 1;
   if (p6 >= 10)
   {
      p6 -= 9;
   }
```

```
            p7 = d7 * 2;
            if (p7 >= 10)
            {
                p7 -= 9;
            }
            p8 = d8 * 1;
            if (p8 >= 10)
            {
                p8 -= 9;
            }
            p9 = d9 * 2;
            if (p9 >= 10)
            {
                p9 -= 9;
            }
            modulo = 10;
            suma = p1 + p2 + p3 + p4 + p5 + p6 + p7 + p8 + p9;

        }
        else if (d3 == 6)
        {
            pub = true;
            p1 = d1 * 3;
            p2 = d2 * 2;
            p3 = d3 * 7;
            p4 = d4 * 6;
            p5 = d5 * 5;
            p6 = d6 * 4;
            p7 = d7 * 3;
            p8 = d8 * 2;
            p9 = 0;
            suma = p1 + p2 + p3 + p4 + p5 + p6 + p7 + p8 + p9;
        }
        else if (d3 == 9)
        {
            priva = true;
            p1 = d1 * 4;
            p2 = d2 * 3;
            p3 = d3 * 2;
            p4 = d4 * 7;
            p5 = d5 * 6;
            p6 = d6 * 5;
            p7 = d7 * 4;
            p8 = d8 * 3;
            p9 = d9 * 2;
            suma = p1 + p2 + p3 + p4 + p5 + p6 + p7 + p8 + p9;
        }

        residuo = suma % modulo;
```

```
digveri = residuo == 0 ? 0 : modulo - residuo;
//Para CI o Ruc

if (pub == true)
{
   if (digveri != d9)
   {
      errorMessageTemplate = "Ruc o C.I. Incorrecto";
      return false;
   }

   if (cioriginal.Length == 13)
   {
      if (cioriginal.Substring(9, 4) != "0001")
      {
         errorMessageTemplate = "Ruc o C.I. Incorrecto";
         return false;
      }

   }


}
else if (priva == true)
{
   if (digveri != d10)
   {
      errorMessageTemplate = "Ruc o C.I. Incorrecto";
      return false;
   }


   if (cioriginal.Length == 13)
   {
      if (cioriginal.Substring(10, 3) != "001")
      {
         errorMessageTemplate = "Ruc o C.I. Incorrecto";
         return false;
      }

   }
}
else if (nat == true)
{
   if (digveri != d10)
   {
      errorMessageTemplate = "Ruc o C.I. Incorrecto";
      return false;
   }
```

```
                if (cioriginal.Length == 13 )
                {
                    if (cioriginal.Substring(10, 3) != "001")
                    {
                        errorMessageTemplate = "Ruc o C.I. Incorrecto";
                        return false;
                    }

                }
            }

        }
        errorMessageTemplate = "Número C.I. Correcto";
        return true;

    }
}


private List<string> IsValidate(FormCollection fc) {

    List<string> errores = new List<string>();
    /*DateTime output;
    if (!DateTime.TryParse(fc["TestDatePicker1"], out output))
    {
        errores.Add("La fecha de inicio no tiene un formato válido");
        return errores;
    }
    DateTime output1;
    if (!DateTime.TryParse(fc["TestDatePicker2"], out output1))
    {
        errores.Add("La fecha de fin no tiene un formato válido");
        return errores;
    }*/
    if (fc["TestDatePicker1"] == "" || fc["TestDatePicker2"] == "") {
        errores.Add("No puede dejar en blanco los valores para la fecha de inicio y
fin de la reserva");
        return errores;
    }
    string format = "MM/dd/yyyy";
    DateTime di = DateTime.ParseExact(fc["TestDatePicker1"], format,
CultureInfo.InvariantCulture);
    DateTime df = DateTime.ParseExact(fc["TestDatePicker2"], format,
CultureInfo.InvariantCulture);
    if (di < DateTime.Now)
        errores.Add("La fecha de inicio debe de ser mayor a la fecha actual");
    if (df < DateTime.Now)
        errores.Add("La fecha de fin debe de ser mayor a la fecha actual");
```

```
            if (df <= di)
                errores.Add("La fecha de fin debe de ser mayor a la fecha de inicio");
            if (!fc["camas"].All(Char.IsDigit) && int.Parse(fc["cama"]) < 0)
                errores.Add("El número de camas debe ser un valor númerico mayor que
0");
            return errores;
        }
```

3.5 Código de factura generar PDF

```
public String GenerarPdf(int idRes, Factura fac)
{
    Reserva res = (from s in dbR.Reserva where s.id == idRes select
s).FirstOrDefault();
    Cliente cli = (from s in dbC.Cliente where s.id == res.idC select
s).FirstOrDefault();
    Habitacion hab = (from s in dbH.Habitacion where
s.numero.Equals(res.numero_hab) select s).FirstOrDefault();
    List<ResSer> servsAd = (from s in dbRS.ResSer where s.idR == res.id select
s).ToList();
    List<Servicio> servs = new List<Servicio>();
    for (int i = 0; i < servsAd.Count; i++)
    {
        int idS = servsAd[i].idS;
        Servicio serv = (from s in dbS.Servicio where s.id == idS select
s).FirstOrDefault();
        servs.Add(serv);
    }

    bool bRet = false;



    // 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.numero +
".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("            HOSTAL RESIDENCIAL
METROPOLIS DEL NORTE   Fecha: " + fecha.Day + "/" + fecha.Month + "/" +
fecha.Year , cabecera));
```

```
            document.Add(new Paragraph( "            Factura (001 - 001)#: " +
fac.numero + "            R.U.C. 17136525574001", cabecera));
            document.Add(new Paragraph("            Bayas Córdova Wilman
Alcívar", DESC));
            document.Add(new Paragraph("            SERVICIO DE
HOSPEDAJE EN HOSTALES", DESC));
            document.Add(new Paragraph("            Direc: Corazón de Jesús N6-
80 y Panamericana Norte", DESC));
            document.Add(new Paragraph("            Teléfonos: 2421 606 /
0993147 697", DESC));
            document.Add(new Paragraph("            Quito - Ecuador", DESC));
            document.Add(new Paragraph("            ", DESC));
            document.Add(new Paragraph("            Fecha de la Reserva: " +
res.fecha_r.Day + "/" + res.fecha_r.Month + "/" + res.fecha_r.Year));
            document.Add(new Paragraph("            Cliente: " + cli.Nombre + " " +
cli.Apelllidos + "  Cédula: " + cli.identificacion));
            document.Add(new Paragraph("            Teléfono: " + cli.telef + " E-
mail: " + cli.mail + "  Dirección: " + cli.dir));
            document.Add(new Paragraph("            Fecha de Entrada: " +
res.fecha_e.Day + "/" + res.fecha_e.Month + "/" + res.fecha_e.Year + "     Fecha de
Salida: " + res.fecha_s.Day + "/" + res.fecha_s.Month + "/" + res.fecha_s.Year));
            document.Add(new Paragraph("            "));
            // step 4: we create a table and add it to the document
            PdfPTable aTable = new PdfPTable(6);
            float[] widths = { 30, 25, 20, 30, 30, 20 };
            aTable.SetTotalWidth(widths);
            // 2 rows, 2 columns
            PdfPCell headerG = new PdfPCell(new Phrase("Detalles de la
reservación", arial));
            headerG.Colspan = 6;
            aTable.AddCell(headerG);
            aTable.AddCell("# de Habitación");
            aTable.AddCell("Descripción");
            aTable.AddCell("Tipo");
            aTable.AddCell("Precio por Noche");
            aTable.AddCell("Cantidad de Noches");
            aTable.AddCell("Importe");
            aTable.AddCell(res.numero_hab);
            aTable.AddCell("Adultos: " + res.cantA + ". Niños: " + res.cantN);
            aTable.AddCell(hab.tipo);
            aTable.AddCell(hab.precio.ToString());
            aTable.AddCell((res.fecha_s - res.fecha_e).Days.ToString());
            aTable.AddCell((hab.precio * (res.fecha_s -
res.fecha_e).Days).ToString());
            PdfPCell headerS = new PdfPCell(new Phrase("Detalles de los Productos",
arial));
            headerS.Colspan = 6;
            aTable.AddCell(headerS);
            PdfPCell ts = new PdfPCell(new Phrase("Productos"));
```

```
            ts.Colspan = 5;
            aTable.AddCell(ts);
            aTable.AddCell("Precio");
            double suma = 0;
            for (int i = 0; i < servs.Count; i++)
            {
               PdfPCell celS = new PdfPCell(new Phrase(servs[i].nombre));
               celS.Colspan = 5;
               aTable.AddCell(celS);
               aTable.AddCell(servs[i].precio.ToString());
               suma += servs[i].precio;
            }
            PdfPCell padd = new PdfPCell(new Phrase(""));
            padd.Colspan = 4;
            aTable.AddCell(padd);
            aTable.AddCell("Subtotal:");
            aTable.AddCell((hab.precio * (res.fecha_s - res.fecha_e).Days +
suma).ToString());
            aTable.AddCell(padd);
            aTable.AddCell("IVA:");
            aTable.AddCell(((hab.precio * (res.fecha_s - res.fecha_e).Days + suma) *
0.12).ToString());
            aTable.AddCell(padd);
            aTable.AddCell("Total:");
            aTable.AddCell((((hab.precio * (res.fecha_s - res.fecha_e).Days + suma) *
0.12) + (hab.precio * (res.fecha_s - res.fecha_e).Days + suma)).ToString());
            document.Add(aTable);
            this.generarXML(res, cli, hab, servs, fac, suma);

            bRet = true;
         }
         catch (DocumentException de)
         {
            return de.Message;
         }
         catch (IOException ioe)
         {
            return ioe.Message;
         }

         // step 5: we close the document
         document.Close();
         String sFilePDF = "";
         if (bRet)
            sFilePDF += " has been created";

         return sFilePDF;
      }
```

```
public string generarClave(int fac)
{
   DateTime dt = DateTime.Now;
   string[] arr = dt.ToString("dd/MM/yyyy").Split('/');
   string s = arr[0] + arr[1] + arr[2];
   s+="0117136525740011001001";

   int nfac = fac;
   int cont=0;
   while(nfac !=0)
   {
      nfac/=10;
      cont++;
   }

   for(int i=cont;i<9;i++)
      s+='0';

   s+=fac;


   Random ram = new Random();

   int ran = ram.Next(100000000);
   int nran = ran;
   cont=0;
   while(nran !=0)
   {
      nran/=10;
      cont++;
   }

   for(int i=cont;i<8;i++)
      s+='0';

   s+=ran+"1";


   int suma=0;
   int incremento = 2;
   for(int i = 47;i>=0;i--)
   {
      int dig = int.Parse(s.ElementAt(i).ToString());
      suma+= dig*incremento;
      incremento++;
      if(incremento==8)
         incremento=2;
   }
```

```
        int ver = suma%11;
        ver = 11-ver;
        if(ver==11)
            ver = 0;
        else if(ver == 10)
            ver = 1;
        s+=ver;

        return s;

    }
```

3.6 Código de factura generar XML

```
        public void generarXML(Reserva res, Cliente cli, Habitacion hab,
    List<Servicio> servs, Factura fac, double sumaS)
     {
        //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", "WILMAR ALCIVAR
BAYAS CORDOVA"));
        infoTributaria.Add(new XElement("nombreComercial", "HOSTAL
RESIDENCIAL METROPOLIS DEL NORTE"));
        infoTributaria.Add(new XElement("ruc", "1713652574001"));
        infoTributaria.Add(new XElement("claveAcceso",
this.generarClave(fac.id)));
        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", "CORAZON DE JESUS N6-
80 Y PANAMERICANA NORTE"));

        //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", "CORAZON DE
JESUS N6-80 Y PANAMERICANA NORTE"));
        infoFactura.Add(new XElement("contribuyenteEspecial", 1001));
        infoFactura.Add(new XElement("obligadoContabilidad", "NO"));
        infoFactura.Add(new XElement("tipoIdentificacionComprador",
"CEDULA"));
        infoFactura.Add(new XElement("razonSocialComprador", cli.Nombre + " "
+ cli.Apelllidos));
        infoFactura.Add(new XElement("identificacionComprador",
cli.identificacion));
        infoFactura.Add(new XElement("totalSinImpuestos", (hab.precio *
(res.fecha_s - res.fecha_e).Days + sumaS)));
        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", (hab.precio * (res.fecha_s - res.fecha_e).Days +
sumaS) * 0.12))));
        infoFactura.Add(new XElement("propina", 0));
        infoFactura.Add(new XElement("importeTotal", (((hab.precio * (res.fecha_s
- res.fecha_e).Days + sumaS) * 0.12) + (hab.precio * (res.fecha_s -
res.fecha_e).Days + sumaS)).ToString()));
        infoFactura.Add(new XElement("moneda", "DOLAR"));

        //nodo detalles factura

        XElement detalles = new XElement("detalles");
        detalles.Add(new XElement("detalle",
           new XElement("codigoPrincipal", hab.numero),
           new XElement("descripcion", hab.descripcion),
           new XElement("cantidad", (res.fecha_s - res.fecha_e).Days),
           new XElement("precioUnitario", hab.precio),
           new XElement("descuento", 0),
           new XElement("precioTotalSinImpuesto", hab.precio * (res.fecha_s -
res.fecha_e).Days),
           new XElement("impuestos", new XElement("impuestos", new
XElement("codigo", 2),
              new XElement("codigoPorcentaje", 0),
              new XElement("tarifa", 0),
              new XElement("baseImponible", 12.12),
              new XElement("valor", hab.precio * (res.fecha_s - res.fecha_e).Days *
0.12)))));
        foreach (Servicio item in servs)
        {
           detalles.Add(new XElement("detalle",
           new XElement("codigoPrincipal", item.id),
           new XElement("descripcion", item.nombre),
           new XElement("cantidad", 1),
           new XElement("precioUnitario", item.precio),
           new XElement("descuento", 0),
           new XElement("precioTotalSinImpuesto", item.precio),
           new XElement("impuestos", new XElement("impuestos", new
XElement("codigo", 2),
              new XElement("codigoPorcentaje", 0),
              new XElement("tarifa", 0),
              new XElement("baseImponible", 12.12),
              new XElement("valor", item.precio * 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"), fac.numero +
".xml");
            factura.Save(@path);
            //factura.Save(Server.MapPath("/Xml"+cedula+".xml"));
        }



        // GET: /Reserva/ReservaRealizada
        public ActionResult ReservaRealizada()
        {
            return View();
        }
        // GET: /Reserva/Details/5

        public ActionResult Details(int id)
        {
            return View();
        }

        //
        // GET: /Reserva/Create

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

        //
        // POST: /Reserva/Create

        [HttpPost]
        public ActionResult Create(FormCollection collection)
        {
            try
            {
```

```
            // TODO: Add insert logic here

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

    //
    // GET: /Reserva/Edit/5

    public ActionResult Edit(int id)
    {
        return View();
    }

    //
    // POST: /Reserva/Edit/5

    [HttpPost]
    public ActionResult Edit(int id, FormCollection collection)
    {
        try
        {
            // TODO: Add update logic here

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

    //
    // GET: /Reserva/Delete/5

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

    //
    // POST: /Reserva/Delete/5

    [HttpPost]
    public ActionResult Delete(int id, FormCollection collection)
```

```
        {
            try
            {
                // TODO: Add delete logic here

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

BIBLIOGRAFÍA

Cortez Còrdova, Miguel, Lopez Puscàn, Javier Nunja Melquiades, Esther Revilla Pachamora,Rosa

http://files.teamzirt.webnode.es/200000332cb2d2cc270/PROYECTO%20web%20de%20hoteleria.docx

http://es.wikipedia.org/wiki/Microsoft_SQL_Server

http://sql-server.softonic.com/

https://www.google.com.tr/?gws_rd=cr&ei=wUcaVZ6yNYqkgwT2hgM#q=diagramas+rational+rose+

https://msdn.microsoft.com/es-es/library/debza5t0(v=**vs**.100).aspx

https://geeks.ms/blogs/rduarte/.../controles-de-validaci-243-n-en-asp-net.aspx

http://es.wikipedia.org/wiki/Diagrama_de_casos_de_uso