INSTITUTO TECNOLOGICO
"CORDILLERA"

CARRERA DE ANÁLIS DE SISTEMAS

OPTIMIZACIÓN DE LOS REGISTROS DE RESERVACIONES Y CONTROL DE HOSPEDAJE EN EL HOSTAL ISRAEL MEDIANTE UNA APLICACIÓN ORIENTADA A LA WEB.

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

Autor: Rivilla Mogro Gabriel David

Tutor: Ing. Carlos Adrián Nieto Sarmiento

Quito, Mayo 2016

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.

_____

GABRIEL DAVID RIVILLA MOGRO

C.C. 110511379-7

CONTRATO DE CESIÓN SOBRE DERECHOS PROPIEDAD INTELECTUAL

Comparecen a la celebración del presente contrato de cesión y transferencia de derechos de propiedad intelectual, por una parte, el estudiante GABRIEL DAVID RIVILLA MOGRO 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 administración bancaria y financiera que imparte el Instituto Superior Tecnológico Cordillera, y con el objeto de obtener el título de Tecnólogo en Analista de Sistemas, el estudiante participa en el proyecto de grado denominado "OPTIMIZACIÓN DE LOS REGISTROS DE RESERVACIONES Y CONTROL DE HOSPEDAJE EN EL HOSTAL ISRAEL MEDIANTE UNA APLICACIÓN ORIENTADA A LA WEB.".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 dieciocho días del mes de mayo del dos mil dieciséis

f) _____	f) _____
C.C. Nº 1105113797	Instituto Superior Tecnológico Cordillera
CEDENTE	CESIONARIO

AGRADECIMIENTO

Mis sinceros agradecimientos a las autoridades, profesores del Instituto Tecnológico Cordillera, por acogerme y compartir sus conocimientos y sobre todo por su invalorable ayuda que lograron guiarme y hacer de mí una mejor persona y profesional.

Al personal docente de mi apreciado Instituto, que siempre apoyan al desarrollo del estudiante, brindando una educación digna y de excelencia, asimismo al Ing. Carlos Nieto, por su excelente orientación.

DEDICATORIA

El esfuerzo de la realización de este proyecto con profundo amor y gratitud dedico primeramente al Ser Celestial que me dio la vida y que nunca me ha desamparado, a mis queridos padres Zoilo Rivilla y Dolores Mogro, que han depositado en mí la confianza el deseo y el impulso para conseguir mis sueños y a toda mi familia que de una u otra manera siempre me estuvieron apoyando.

ÍNDICE GENERAL

ÍNDICE TABLAS

ÍNDICE DE ILUSTRACIONES

RESUMEN EJECUTIVO

El presente proyecto se lo ha realizado con la finalidad de optimizar los procesos de registro de reserva y control de hospedaje mediante una aplicación web para el hostal ISRAEL y sobre todo para el crecimiento del mismo. La entrevista fue fundamental para obtener información de los procesos con los que contaban actualmente.

Se ha empleado el Lenguaje Unificado de Modelado conocido por sus siglas (UML) para realizar el modelamiento y la edificación del aplicativo como tal, a través de la elaboración de los diferentes diagramas.

.

Para el desarrollo del aplicativo se ha utilizado varias herramientas tales como Visual Studio 2013, Framework 5.0, su codificación en lenguaje C# y con un motor de Base de Datos SLQ Server 2013.

El aplicativo como primer paso a seguir fue con la obtención de información para el desarrollo y levantamientos tantos funcionales como no funcionales definiendo de esta forma el alcance del proyecto.

ABSTRACT

This project has been performed in order to optimize the registration process booking and hosting control using a web application for the hostel ISRAEL and especially for its growth. The interview was essential to obtain information processes that currently had.

It has been used Unified Modeling Language known by its acronym (UML) for modeling and building the application as such, through the development of different diagrams.

To develop the application has been used several tools such as Visual Studio 2013, Framework 5.0, coding in C # language and a database engine SLQ Server 2013.

The application as a first step to take was to obtain information for development and many functional and non-functional thereby defining the project scope uprisings.

INTRODUCCIÓN

El presente trabajo de investigación "OPTIMIZACIÓN DE LOS REGISTROS DE RESERVACIONES Y CONTROL DE HOSPEDAJE EN EL HOSTAL ISRAEL MEDIANTE UNA APLICACIÓN ORIENTADA A LA WEB" tiene como finalidad ser una herramienta de gran utilidad para el Hostal Israel ya que en la actualidad dicho Hostal no existe un control óptimo.

Por tal motivo nace la necesidad de optimizar todos los procesos de control de hospedaje por medio de un software permitiendo obtener resultados de manera rápida, precisa y eficaz para el beneficio del Hostal.

Toda la información ingresada al sistema se realizará de forma segura y podrá visualizar al momento de generar los reportes de acuerdo a la información solicitada por el encargado del hostal, además todo usuario que intente ingresar al sistema deberá registrarse previamente en el mismo.

Capítulo I: ANTECEDENTES

1.01. CONTEXTO

En el Ecuador la existencia de grandes redes hoteleras se han visto en la necesidad de buscar e implementar nuevos sistemas tecnológicos con el fin de mejorar sus servicios y consigo ganar mayor prestigio dentro del mercado hotelero tal es el caso que al norte del Distrito Metropolitano de Quito la existencia de hoteles y hostales no son la excepción, quedando de lado demás hoteles incluido el Hostal Israel ya que no cuentan con ese servicio es decir el registro y reservaciones de sus clientes se hacen de forma manual.

Hostal Israel posee gran acogida y prestigio dentro de la ciudad de Quito, tiene la necesidad, vender, gestionar y distribuir sus servicios con la mejor calidad a todos sus huéspedes o nuevos clientes todo esto encaminado a alcanzar gran competitividad dentro del mercado hotelero. Al tener una gran lista de huéspedes se hace complicado el registro de todas las reservaciones teniendo como consecuencia que varios clientes queden sin cupo.

La ausencia de un elemento tecnológico en el Hostal Israel impide que exista una buena organización, todo esto hace un gran problema ya que frecuentemente se pierde información, esto se debe a que maneja archivos planos o manuales para el registro de huéspedes, reservaciones.

Hoy en día los usuarios que utilizan servicios de hospedaje exigen ser atendidos de forma ágil y altamente eficiente sintiéndose seguros del servicio adquirido, disminuyendo el tiempo y riesgo de quedarse sin reservaciones.

1.02. JUSTIFICACIÓN

Una herramienta web recoge todas las necesidades del mercado hotelero, los problemas que generan insatisfacción a los clientes y producen notables perdidas económicas, el hecho de que las reservas se manejen en forma personal puede provocar que un gran número de usuarios, sobre todo en época vacacional se vean impedidos para acceder al servicio.

Para el mercado hotelero es de gran importancia no ser afectado y no cumplir con estándares de calidad de servicio en la atención al cliente, por otro lado el hecho de que sus empleados manejen procesos poco competitivos produce dificultades y la hace menos productiva es por ello que la optimización de proceso de reservaciones hace posible que mejore la prestación de servicios.

La presente investigación se constituye en una herramienta base para mejorar el registro de la información del Hostal, es decir facilitara la prestación de servicios en lo concerniente a hospedaje, reservaciones, consulta de habitaciones disponibles.

Mediante la aplicación del proyecto planteado se realizará los procesos de guardar, almacenar y registrar de manera adecuada la información de todos sus

clientes. El hostal Israel al no contar con una herramienta exclusivo para su operatividad seguirá perdiendo clientes, no podrá dar seguimiento a sus reservaciones o a sus vez habrán reservaciones paralelas por lo cual no cumplirán con todos sus clientes.

El Hostal Israel tendrá una importante herramienta de control y registro de todos sus huéspedes, mejorando sus servicios, ganando prestigio, gran competitividad dentro del mercado hotelero.

1.03. DEFINICIÓN DEL PROBLEMA CENTRAL

Mediante la aplicación de la matriz T podemos analizar la situación actual de dónde partiremos para conocer en qué situación se encuentra los procesos dentro de la empresa y de la misma manera para llegar a dar una solución para el mejoramiento de los mismos.

Tabla 1: Matriz T

SITUACIÓN EMPEORADA	SITUCIÓN ACTUAL				SITUACIÓN MEJORADA
Pérdida de los registros de ingreso de los huéspedes.	Inadecuada organización de los registros de reservaciones.				Organización y Eficiencia al momento de ingresar la información de los huéspedes.
FUERZAS IMPULSADORAS	**CALIFICACIÓN**				**FUERZAS BLOQUEADORAS**
	I	PC	I	PC	
Establecer una organización innovadora para los procesos de ingreso de información.	2	4	2	4	Bajo control en el registro de información ingresada.
Determinar procesos óptimos para registrar la información al momento de ingresar los huéspedes.	2	4	2	4	Métodos inadecuados para los registros de información de los huéspedes.
Revisión y control meticuloso de la información al ser ingresados	2	4	2	4	Impericia en el manejo de control de la información.

Nota: Matriz de análisis de fuerzas T. en esta matriz detallamos las fuerzas bloqueadoras que nos impiden lograr la situación mejorada y las fuerzas impulsadoras que nos ayudaran a cumplir con la propuesta del proyecto.
"I" = Intensidad
"PC"= Potencial de cambio
1=Bajo
2= Medio Bajo
3= Medio
4= Medio Alto
5= Alto

Capítulo II: ANÁLISIS DE INVOLUCRADOS

2.01. REQUERIMIENTOS

2.01.1. DESCRIPCIÓN DEL SISTEMA ACTUAL

La parte administrativa del hostal actualmente cuenta con un proceso de ingreso de información manual archivándolo en hojas de cálculo, por ejemplo el huésped se dirige a recepción, a quien le dice que va a reservar una habitación, luego el recepcionista pregunta el tiempo que va a permanecer en la instalación, entonces le pide que escriba sus datos en la hoja de cálculo de Excel y posteriormente le brinda los accesorios adecuados para su instancia.

Posteriormente estas hojas de cálculo son archivadas y no brindan una organización adecuada, además de convertirse en una tarea muy tediosa al momento de realizar las consultas y obtener reportes de información requerida hostal Israel realiza sus búsquedas mediante archivos de documentación la cuál es muy tediosa es por ello que el actual proyecto va lograr que se mejore los procesos de reservaciones y control en cuanto a calidad, rapidez, servicio y eficacia que este proceso requiere.

2.01.2. VISIÓN

El presente proyecto va a permitir al hostal Israel llevar los procesos de reservaciones y control de una manera óptima que pueda registrar la información de los huéspedes realizar búsquedas y generar reportes de tal manera que el equipo administrativo pueda llevar adecuadamente los procesos antes mencionados.

2.01.2.1. ALCANCE

MÓDULO DE SEGURIDAD

Este módulo se caracteriza porque va permitir definir los usuarios de administrador y empleado de esta manera gestionar las funciones o tareas a las que podrán tener acceso, logrando con ello una mejor seguridad al momento de manipular la información

MÓDULO DE MANTENIMIENTO

Nos va a permitir guardar información de una manera segura y adecuada, podemos modificarla o eliminarla de acuerdo a las necesidades que el hostal las requiera permitiendo el buen funcionamiento de nuestro sistema.

MÓDULO DE NEGOCIO

Se hace referencia a las reglas del negocio en dónde debemos tenerlas muy claras y las cuales nos van a permitir llegar a nuestro objetivo en este proyecto vamos a optimizar el proceso de reservaciones para así poder brindar un servicio de primera y que el hostal vaya aumentando su prestigio.

MÓDULO DE REPORTES

En este módulo va a permitir que el administrador genere reportes de las reservas diarias y de los huéspedes según su nombre cada que lo requiera, así permitiendo una mejor gestión y organización de la información.

2.01.3. ENTREVISTAS

Para la obtención de la siguiente información se realizó una entrevista a los recepcionistas y al gerente del Hostal Israel.

Tabla 2: Entrevista

ENTREVISTA		
PREGUNTAS	**OBJETIVOS**	**ANÁLISIS POSTERIOR**
¿Cuál es la manera en que se registran a los huéspedes en el hostal Israel?	Determinar los procesos a mejorar con el aplicativo web.	Optimizar los procesos de reservaciones y control de los huéspedes.
¿Cómo considera usted la manera de acceder a la información?	Controlar de manera sistematizada las reservaciones y control de la información.	Generar reportes diarios para tener un seguimiento único por cada huésped.
¿Cree usted que el registro computarizado del cliente puede mejorar el servicio brindado?	Optimizar el registro de reservaciones de los huéspedes.	Se requiere que se mejore el proceso de reservaciones en el hostal.
¿Con qué frecuencia usted realiza consultas de los registros?	Generar reportes para mantener informado al administrador para saber la cantidad de clientes llega cada día.	Se requiere que se genere reportes de los registros diarios.

Nota: En esta entrevista recolectamos información acerca de cómo se maneja los procesos dentro del hostal.

2.01.1. MATRIZ DE REQUERIMIENTOS

Mediante la aplicación de la matriz podemos identificar y especificar los requerimientos tanto funcionales como no funcionales que se va incluir en nuestro sistema.

Tabla 3: Requerimientos Funcionales y no funcionales

MATRIZ DE REQUERIMIENTOS						
Identificador	**Descripción**	**Fuente**	**Prioridad**	**Tipo**	**Estado**	**Usuarios Involucrados**
REQUERIMIENTOS FUNCIONALES						
RF001	El aplicativo deberá registrar al empleado asignando un rol.	Gerente	Alta	Funcional	Análisis	Gerente Recepcionista Administrador
RF002	El aplicativo debe tener el control de empleados para el acceso según su rol	Gerente	Alta	Funcional	Análisis	Gerente Recepcionista Administrador
RF003	El aplicativo debe controlar el ingreso de reservaciones con los siguientes estados: reservada, activa o cancelada.	Recepcionista	Alta	Funcional	Análisis	Recepcionista Administrador
RF004	El aplicativo al momento de guardar la reservación deberá generar una ficha de reservación.	Recepcionista	Media Alta	Funcional	Análisis	Recepcionista Administrador
RF005	El aplicativo deberá generar reportes de reservaciones diarias.	Recepcionista	Media Alta	Funcional	Análisis	Gerente Administrador
RF006	El aplicativo debe realizar búsquedas de las reservaciones por nombre del cliente	Gerente	Media Alta	Funcional	Análisis	Gerente Administrador
REQUERIMIENTOS NO FUNCIONALES						
NRF001	El aplicativo deberá tener una interfaz amigable para que el cliente pueda usarla con facilidad	Gerente	Media	No funcional	Análisis	Recepcionista
NRF002	El aplicativo debería ser multiplataforma	Gerente	Media	No funcional	Análisis	Recepcionista

Nota: En esta matriz detallamos los requerimientos funcionales como no funcionales de la información que adquirimos en la entrevista

2.01.2. DESCRIPCIÓN DETALLADA

En los siguientes anexos describiremos detalladamente tanto los requerimientos funcionales como los requerimientos no funcionales de la información que hemos adquirido.

Tabla 4: Requerimiento Funcional

DESCRIPCIÓN DETALLADA DEL REQUERIEMIENTO FUNCIONAL			
El aplicativo deberá registrar al empleado asignando un rol.		**Estado:**	En análisis
Creado por:	Gabriel Rivilla	**Actualizado por:**	Gabriel Rivilla
Fecha de creación:	04-01-2016	**Fecha de Actualización:**	13-01-2016
Identificador:	RF001		
Estado de Requerimiento:	Crítico	**Tipo de Requerimiento**	Funcional
Datos de entrada:	Registro del empleado Asignación de rol.		
Descripción:	Se realizará el registro del empleado y se le asignará su respectivo rol.		
Datos de salida:	Se desplegará un mensaje: empleado guardado exitosamente.		
Resultados esperados:	Registro del empleado.		
Origen:	Gerente del hostal.		
Dirigido a:	Recepcionista, administrador.		
Prioridad:	5		
Requerimientos asociados:	Ninguno		
ESPECIFICACIÓN			
Precondiciones:	1.- El Empleado aporta con sus datos personales. 2.- Se procede a ingresar su información al sistema. 3.- Se le asigna un rol. 4.- Seguidamente se registra al empleado. 5.- Se despliega un mensaje: empleado guardado exitosamente.		
Pos **condiciones:**	1.- Podrá manipular el sistema con respecto a su rol.		
Criterios de aceptación:	Luego de registrar al usuario se procede a ingresar al sistema de acuerdo a su rol.		

Nota: En esta matriz se detalla los requerimientos funcionales identificados.

Escala:

1 = baja

2 = media baja

3 = media

4 = media alta

5 = alta

Tabla 5: Requerimiento Funcional

DESCRIPCIÓN DETALLADA DEL REQUERIEMIENTO FUNCIONAL			
El aplicativo debe tener el control de empleados para el acceso según su rol		**Estado:**	En análisis
Creado por:	Gabriel Rivilla	**Actualizado por:**	Gabriel Rivilla
Fecha de creación:	04-01-2016	**Fecha de Actualización:**	13-01-2016
Identificador:	RF002		
Estado de Requerimiento:	Crítico	**Tipo de Requerimiento**	Funcional
Datos de entrada:	Rol de entrada, username de usuario, password de usuario.		
Descripción:	Para obtener el acceso al sistema los usuarios deberán loguearse con su respectivo username y password.		
Datos de salida:	Se despliega un mensaje: username y password ingresados correctamente caso contrario username o password incorrectos.		
Resultados esperados:	Ingreso al sistema según su rol asignado		
Origen:	Gerente		
Dirigido a:	Recepcionista, administrador.		
Prioridad:	5		
Requerimientos asociados:	RF002		
ESPECIFICACIÓN			
Precondiciones:	1.- Para tener acceso al sistema los usuarios deben ser debidamente registrados. 2.- Una vez registrados solo tendrán acceso a ciertos módulos según su rol.		
Pos **condiciones:**	Si se ingresa correctamente podrá manipular el sistema según su rol asignado.		
Criterios de aceptación:	Permite que los empleados realicen consultas y generen reportes según sus necesidades.		

Notas: En esta matriz se detalla los requerimientos funcionales identificados.
Escala:
1 = baja
2 = media baja
3 = media
4 = media alta
5 = alta

Tabla 6: Requerimiento Funcional

DESCRIPCIÓN DETALLADA DEL REQUERIEMIENTO FUNCIONAL			
El aplicativo debe controlar el ingreso de reservaciones con los siguientes estados: reservada, activa o cancelada.		**Estado:**	En análisis
Creado por:	Gabriel Rivilla	**Actualizado por:**	Gabriel Rivilla
Fecha de creación:	04-01-2016	**Fecha de Actualización:**	13-01-2016
Identificador:	RF003		
Estado de Requerimiento:	Crítico	**Tipo de Requerimiento**	Funcional
Datos de entrada:	Registro del huésped, datos personales, asignación de habitación.		
Descripción:	Se realizará el registro del cliente con sus datos personales.		
Datos de salida:	Se desplegará un mensaje: reserva realizada exitosamente, generar una ficha de reservación.		
Resultados esperados:	Registro del huésped con su respectiva asignación de habitación.		
Origen:	Recepcionista		
Dirigido a:	Recepcionista, administrador.		
Prioridad:	5		
Requerimientos asociados:	Ninguno		
ESPECIFICACIÓN			
Precondiciones:	1.- El huésped aporta con sus datos personales. 2.- Se procede a ingresar su información al sistema. 3.- Se le asigna una respectiva habitación. 4.- Seguidamente se registra al huésped. 5.- Se despliega un mensaje: reserva realizada exitosamente guardado exitosamente.		
Pos **condiciones:**	Una vez registrados los huéspedes se les asigna su respectiva habitación.		
Criterios de aceptación:	Permite que el administrador realice las consultas de los clientes.		

Notas: En esta matriz se detalla los requerimientos funcionales identificados.

Escala:

1 = baja

2 = media baja

3 = media

4 = media alta

5 = alta

Tabla 7: Requerimiento Funcional

DESCRIPCIÓN DETALLADA DEL REQUERIEMIENTO FUNCIONAL			
El aplicativo al momento de guardar la reservación deberá generar una ficha de reservación.		**Estado:**	En análisis
Creado por:	Gabriel Rivilla	**Actualizado por:**	Gabriel Rivilla
Fecha de creación:	04-01-2016	**Fecha de Actualización:**	13-01-2016
Identificador:	RF004		
Estado de Requerimiento:	Crítico	**Tipo de Requerimiento**	Funcional
Datos de entrada:	Registro de reservación.		
Descripción:	Se realizará el registro del cliente con sus datos personales.		
Datos de salida:	Se desplegará un mensaje: reserva realizada exitosamente, generar una ficha de reservación.		
Resultados esperados:	Registro del huésped con su respectiva asignación de habitación.		
Origen:	Recepcionista		
Dirigido a:	Recepcionista, administrador.		
Prioridad:	4		
Requerimientos asociados:	RF003		
ESPECIFICACIÓN			
Precondiciones:	1.- Se registra al huésped.		
Pos **condiciones:**	Al momento de registrar al huésped se despliega un mensaje: reserva realizada exitosamente		
Criterios de aceptación:	Permite que el administrador realice las consultas de los huéspedes		

Notas: En esta matriz se detalla los requerimientos funcionales identificados.
Escala:
1 = baja
2 = media baja
3 = media
4 = media alta
5 = alta

Tabla 8: Requerimiento Funcional

DESCRIPCIÓN DETALLADA DEL REQUERIEMIENTO FUNCIONAL			
El aplicativo deberá generar reportes de reservaciones diarias.		**Estado:**	En análisis
Creado por:	Gabriel Rivilla	**Actualizado por:**	Gabriel Rivilla
Fecha de creación:	04-01-2016	**Fecha de Actualización:**	13-01-2016
Identificador:	RF005		
Estado de Requerimiento:	Crítico	**Tipo de Requerimiento**	Funcional
Datos de entrada:	Registro de reservación.		
Descripción:	Se realizará el registro del cliente con sus datos personales y la asignación de habitación.		
Datos de salida:	Brinda información de la reservación		
Resultados esperados:	Generar reporte de las reservaciones.		
Origen:	Recepcionista		
Dirigido a:	Recepcionista, administrador.		
Prioridad:	4		
Requerimientos asociados:	RF003, RF004.		
ESPECIFICACIÓN			
Precondiciones:	1.- Se registra al huésped. 2.- Se realiza la reservación.		
Pos **condiciones:**	Al momento de registrar al huésped se despliega un mensaje: reserva realizada exitosamente y se genera el reporte		
Criterios de aceptación:	Permite que el administrador realice los reportes de las reservaciones.		

Notas: En esta matriz se detalla los requerimientos funcionales identificados.

Escala:

1 = baja

2 = media baja

3 = media

4 = media alta

5 = alta

Tabla 9: Requerimiento Funcional

DESCRIPCIÓN DETALLADA DEL REQUERIEMIENTO FUNCIONAL			
El aplicativo debe realizar búsquedas de las reservaciones por nombre del cliente		**Estado:**	En análisis
Creado por:	Gabriel Rivilla	**Actualizado por:**	Gabriel Rivilla
Fecha de creación:	04-01-2016	**Fecha de Actualización:**	13-01-2016
Identificador:	RF006		
Estado de Requerimiento:	Crítico	**Tipo de Requerimiento**	Funcional
Datos de entrada:	Registro de reservación.		
Descripción:	Se realizará el registro del cliente con sus datos personales y la asignación de habitación.		
Datos de salida:	Brinda información de la reservación		
Resultados esperados:	Generar reporte de las reservaciones.		
Origen:	Recepcionista		
Dirigido a:	Recepcionista, administrador.		
Prioridad:	4		
Requerimientos asociados:	RF003, RF004.		
ESPECIFICACIÓN			
Precondiciones:	1.- Se registra al huésped. 2.- Se realiza la reservación.		
Pos **condiciones:**	Al momento de registrar al huésped se despliega un mensaje: reserva realizada exitosamente y se genera el reporte		
Criterios de aceptación:	Permite que el administrador realice los reportes de las reservaciones.		

Nota: En esta matriz se detalla los requerimientos funcionales identificados.
Escala:
1 = baja
2 = media baja
3 = media
4 = media alta
5 = alta

Tabla 10: Requerimiento no Funcional

DESCRIPCIÓN DETALLADA DEL REQUERIEMIENTO FUNCIONAL			
El aplicativo deberá tener una interfaz amigable para que el cliente pueda interactuar con facilidad.		**Estado:**	En análisis
Creado por:	Gabriel Rivilla	**Actualizado por:**	Gabriel Rivilla
Fecha de creación:	04-01-2016	**Fecha de Actualización:**	13-01-2016
Identificador:	NRF001		
Estado de Requerimiento:	Moderado	**Tipo de Requerimiento**	No Funcional
Datos de entrada:	Impacto visual		
Descripción:	Interfaz gráfica amigable.		
Datos de salida:	Atracción por el huésped.		
Resultados esperados:	Interactuar con facilidad.		
Origen:	Recepcionista		
Dirigido a:	Recepcionista, administrador, cliente		
Prioridad:	3		
Requerimientos asociados:	NINGUNO		
ESPECIFICACIÓN			
Precondiciones:			
Pos **condiciones:**	Al momento de registrar al huésped se despliega un mensaje: reserva realizada exitosamente y se genera el reporte		
Criterios de aceptación:	Permite que el administrador realice los reportes de las reservaciones.		

Nota: En esta matriz se detalla los requerimientos funcionales identificados.

Escala:

1 = baja

2 = media baja

3 = media

4 = media alta

5 = alta

Tabla 11: Requerimiento no Funcional

DESCRIPCIÓN DETALLADA DEL REQUERIEMIENTO FUNCIONAL			
El aplicativo debería ser multiplataforma		**Estado:**	En análisis
Creado por:	Gabriel Rivilla	**Actualizado por:**	Gabriel Rivilla
Fecha de creación:	04-01-2016	**Fecha de Actualización:**	13-01-2016
Identificador:	NRF002		
Estado de Requerimiento:	Moderado	**Tipo de Requerimiento**	No Funcional
Datos de entrada:	Ingreso al navegador		
Descripción:	Debe ser compatible con cualquier tipo de navegador ya sea Chrome, Firefox entre otros.		
Datos de salida:	Visualización de la aplicación		
Resultados esperados:	Interactuar de una manera fácil con la aplicación.		
Origen:	Gerente		
Dirigido a:	Recepcionista, administrador, cliente		
Prioridad:	3		
Requerimientos asociados:	NRF001		
ESPECIFICACIÓN			
Precondiciones:	1.- Se obtiene un navegador 2.- Se ingresa al navegador		
Pos **condiciones:**	Visualización de la aplicación.		
Criterios de aceptación:	Agradable y fácil manejo de la aplicación.		

Nota: En esta matriz se detalla los requerimientos funcionales identificados.

Escala:

1 = baja

2 = media baja

3 = media

4 = media alta

5 = alta

2.02. MAPEO DE INVOLUCRADOS

Se visualiza detalladamente al personal involucrado en el proceso del desarrollo del aplicativo tanto directamente como indirectamente.

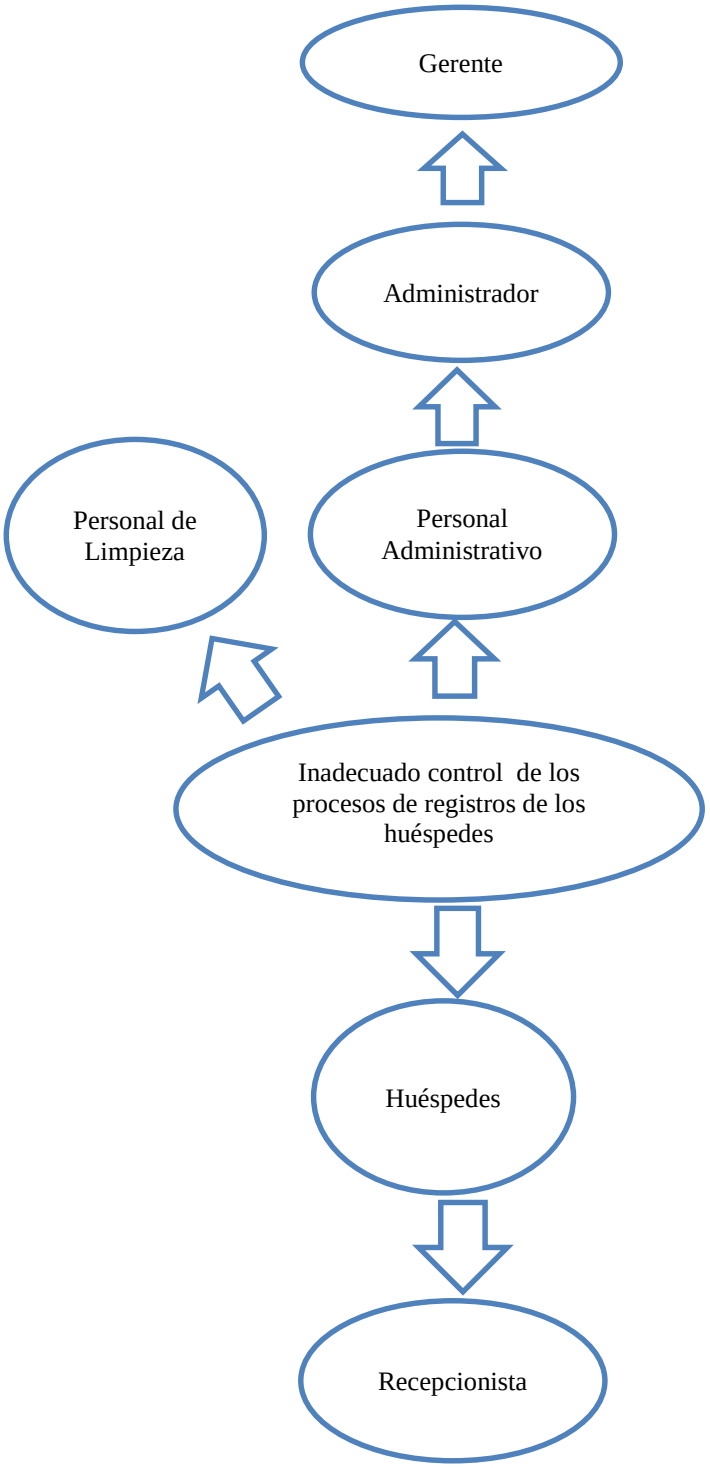


Figura 1: Mapeo de Involucrados.
En esta figura se muestra la participación de los involucrados directos e indirectos.

2.03. MATRIZ DE INVOLUCRADOS

En la siguiente matriz hace referencia a los actores involucrados directamente e indirectamente tomando en referencia su interés, recursos y los problemas percibidos dentro del hostal Israel.

Tabla 12: Matriz de involucrados

ACTORES INVOLUCRADOS	INTERÉS	RECURSOS	PROBLEMA PERCIBIDO
Gerente	Mejorar la gestión del registro de reservaciones.	Controlar de una manera eficiente las reservaciones.	Inexistencia de un software que permita optimizar el proceso de registro.
Administrador	Acceder a toda la información	Verificar la información ingresada al sistema.	Información errónea entregada.
Recepcionista	Optimizar el proceso de registro de reservaciones	Manejar con mayor eficacia exactitud los datos	Información ingresada de manera inadecuada
Personal Administrativo	Controlar el ingreso de información	Manejar con integridad la información	Pérdida de información de las reservas
Huésped	Tener una buena acogida dentro del hostal	Brindar sus datos personales	Insatisfacción al momento de realizar su reserva.

Nota: En esta matriz se detalla la matriz de involucrados.

Capítulo III: PROBLEMAS Y OBJETIVOS

3.01 ÁRBOL DE PROBLEMAS

En el siguiente árbol de problemas podemos identificar los posibles problemas que tenemos que resolver con sus respectivas causas y efectos.

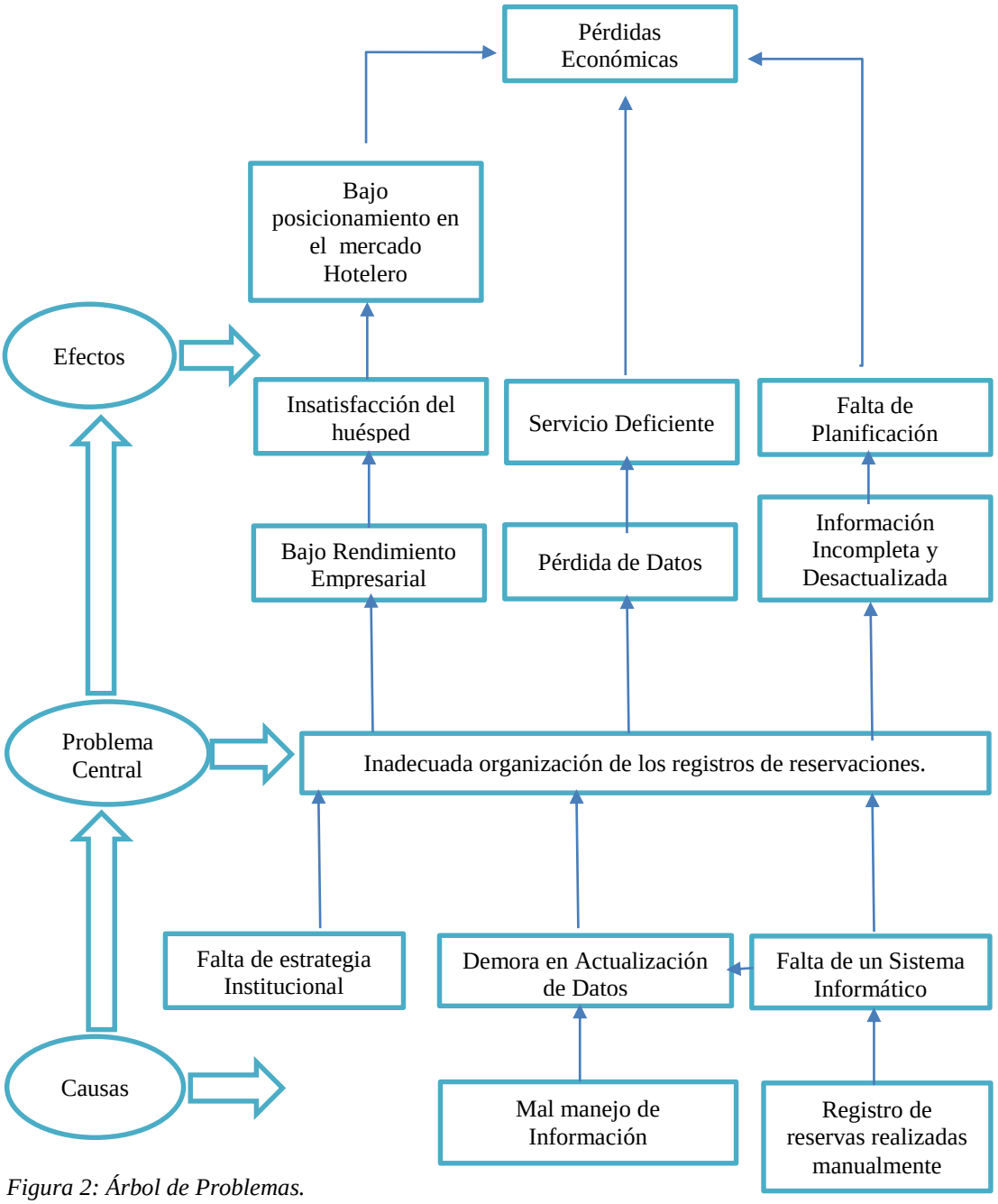


Figura 2: Árbol de Problemas.
En ésta figura se visualiza el problema central con sus respectivas causas y efectos.

3.02 ÁRBOL DE OBJETIVOS

En el siguiente árbol de Objetivos podemos identificar las posibles soluciones que vamos a desarrollar para las mejoras de los procesos del hostal.

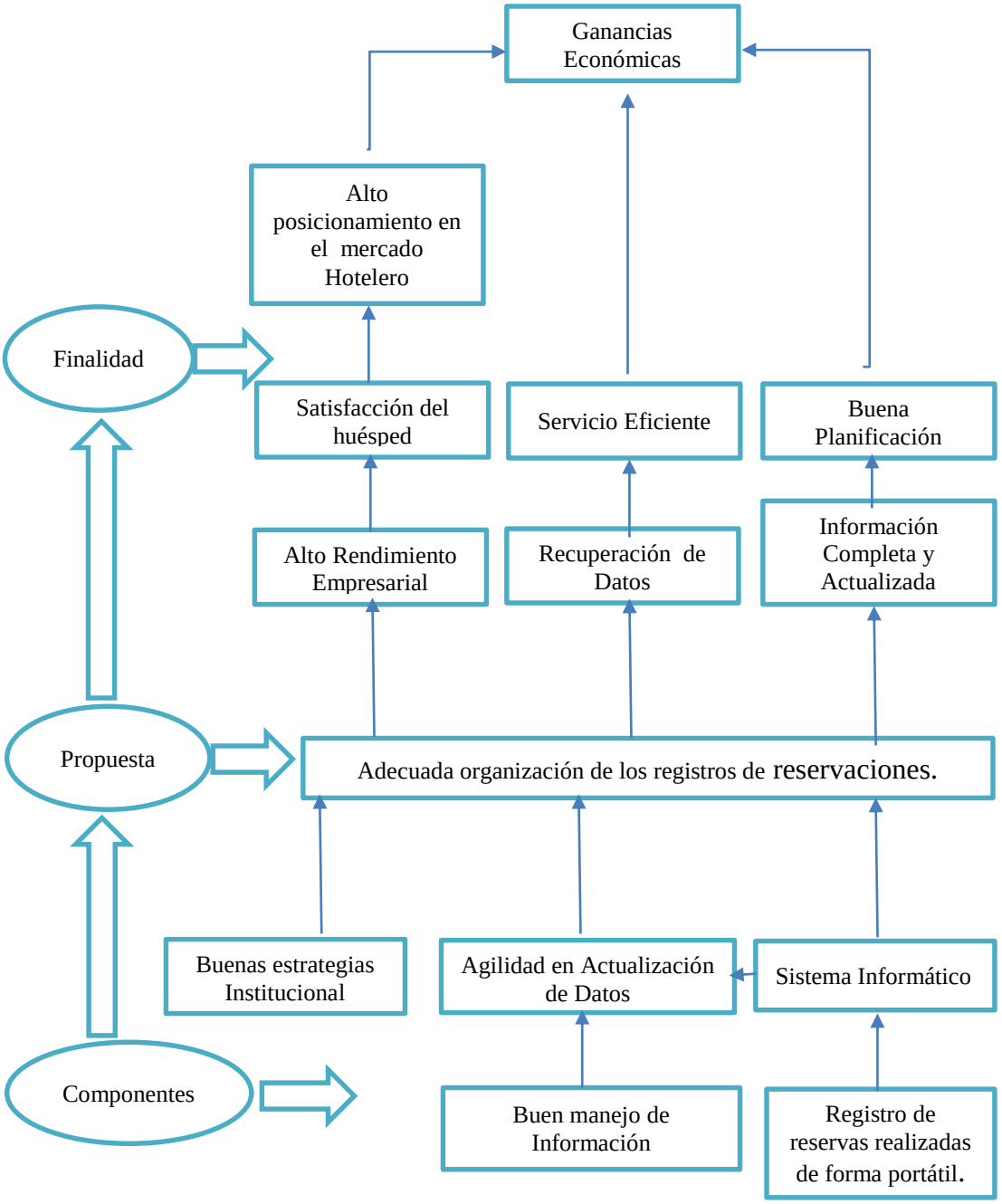


Figura 3: Árbol de Objetivos.
En la presente figura podemos visualizar la propuesta con sus respectivos componentes y finalidades
(Escobar, 2015)
.

3.03 DIAGRAMA DE CASO DE USO

(Wikipedia, 2010) Señala:

En el Diagrama de Casos de Uso se especifica las actividades que se va a realizar en los distintos procesos, en los cuales se va a permitir la interacción entre los actores externos, los que son parte del negocio sin dependencia y los que son parte del negocio con dependencia.

3.03.1 CASO DE USO GENERAL

En el siguiente Diagrama de Caso de Uso describe las tareas que van a realizar los involucrados en el presente proyecto y la interacción con el mismo. (Ver ANEXO A001).

3.03.2 CASO DE USO INDIVIDUAL

En los Diagramas de Caso de Uso se describe las tareas que se va a realizar dentro del proyecto en forma individual en cada uno de los procesos identificados.

OC001 INGRESAR USUARIO

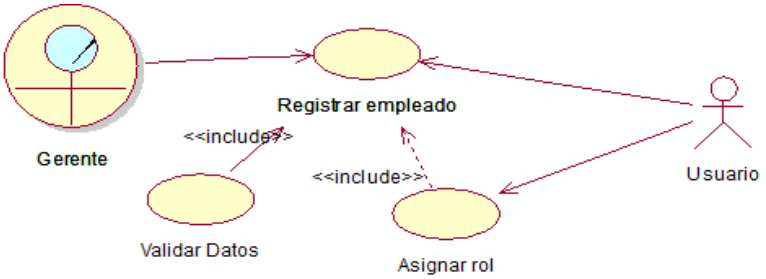


Figura 4: OC001 Ingresar Usuario.
En esta figura se muestra las tareas que se va a realizar en el proceso de ingreso de usuario donde el gerente va ingresar los datos del usuario.

OC002 REALIZAR RESERVA

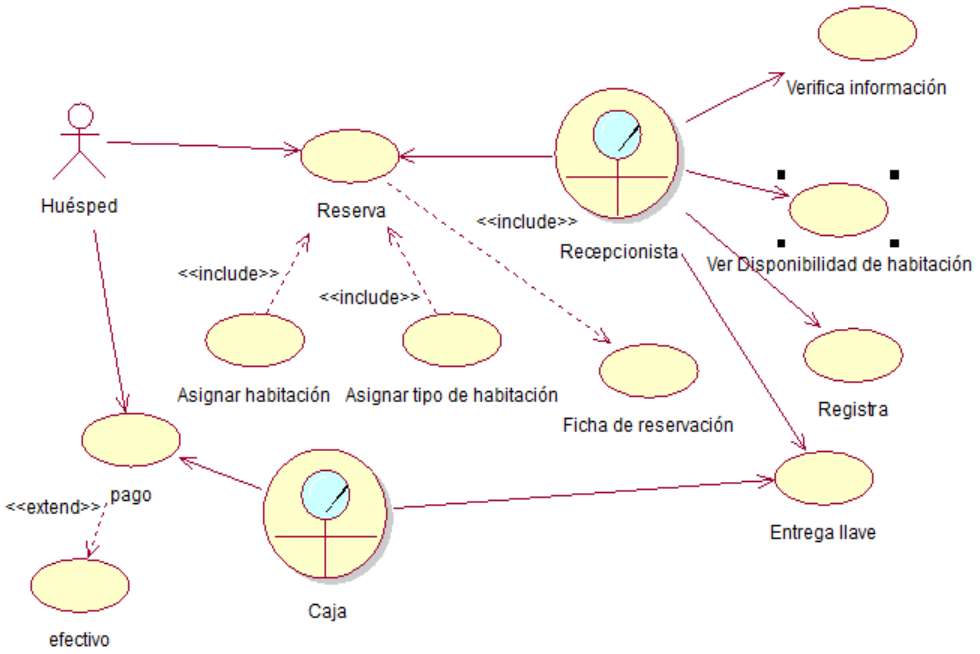


Figura 5: OC002 Realizar Reserva.
En esta figura se muestra las tareas que se va a realizar en el proceso del registro de reserva donde el Recepcionista ingresa Datos del huésped y le asigna su respectiva habitación.

OC003 GENERAR REPORTE

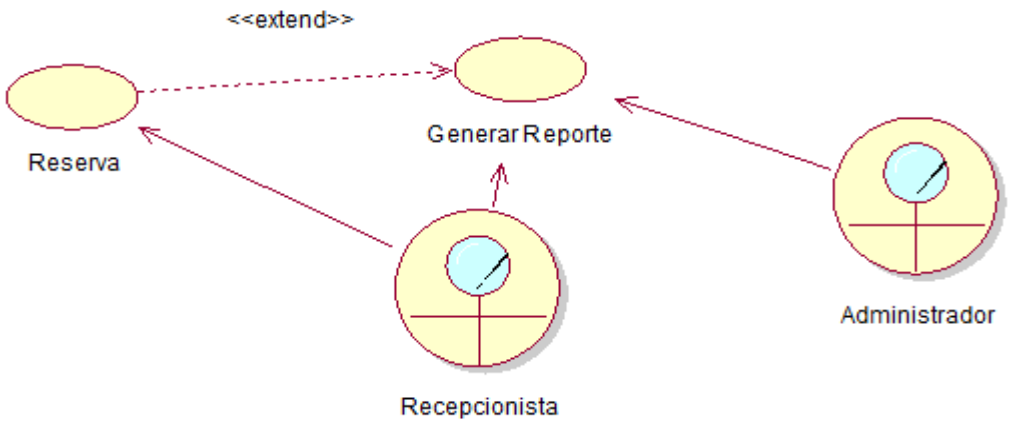


Figura 6: OC003 Generar Reporte.
En esta figura se muestra las tareas que se va a realizar en el proceso de generar reporte de las reservas. Donde el Administrador verifica la información existente.

OC004 GENERAR BÚSQUEDA

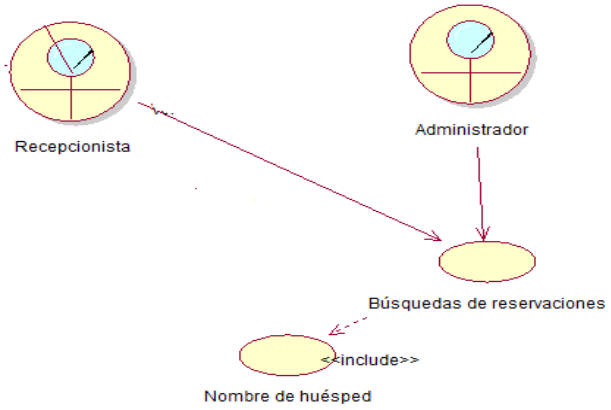


Figura 7: OC004 Generar Búsqueda.
En esta figura se muestra las tareas que se va a realizar en el proceso de generar búsquedas de las reservas. Donde el administrador va buscar información que necesite.

3.03.3 ESPECIFICACIONES DE CASO DE USO

En las siguientes tablas se especifica detalladamente las tareas que se va a realizar en el proyecto en forma individual en cada uno de los procesos identificados.

Tabla 13: OC001 INGRESAR USUARIO

Caso de Uso	**Ingresar Usuario**
Identificador	**OC001**
Usuario	**Sistema**
1. Registrar los datos.	1. Verifica si los datos ingresados ya existen si existen visualizara datos existente sino el registro guardado.
2. Asignar rol	2. Se debe seleccionar un rol como administrador o recepcionista sino se visualizará un mensaje: "Debe seleccionar un rol".
CURSOS ALTERNATIVOS	

Nota: En este caso detallamos el proceso de ingresar empleado.

Tabla 14: OC002 REALIZAR RESERVA

Caso de Uso	**Realizar reserva**
Identificador	**OC002**

Usuario	**Sistema**
1. Solicita reserva.	1. Buscará el tipo habitación requerido si hay disponibles se procede a la reserva si no se le recomendará otro tipo de habitación.
2. Confirma reserva	2. Se va a visualizar el formulario de reserva, verifica los datos ingresados si se ingresa correctamente se despliega un mensaje "Reserva exitosa" sino se despliega un mensaje: "verificar información."
3. Imprimir formulario de reserva	3. Si el formulario de reserva es generado puede guardar sino imprimir directamente.

CURSOS ALTERNATIVOS

La reserva también se la puede realizar vía telefónica y se le envía a su correo personal el formulario de reservación.

Nota: En este Caso de Uso detallamos el proceso de realizar reserva.

Tabla 15: OC003 GENERAR REPORTE

Caso de Uso	**Generar reportes.**
Identificador	**OC003**

Usuario	**Sistema**
1. Selecciona los datos a generar el reporte.	1. Verifica información seleccionada si existe genera reporte sino se visualizará un mensaje: "verificar datos"
2. Seleccionar las fechas que desea generar el reporte.	2. Si la información existe dentro de las fechas seleccionadas generar reporte.
3. Seleccionar si desea imprimir	3. Si el reporte fue generado puede exportar a Excel sino imprimir directamente.

CURSOS ALTERNATIVOS

Nota: En este Caso de Uso detallamos el proceso de generar reportes.

Tabla 16: OC004 GENERAR BÚSQUEDA

Caso de Uso	Buscar reservaciones
Identificador	**OC004**
Usuario	**Sistema**
1. Buscar por nombre de huésped.	2. Si la reservación existe se visualizará la información sino se visualizará un mensaje: " no existe datos"
CURSOS ALTERNATIVOS	

Nota: En este Caso de Uso detallamos el proceso de Buscar reservaciones.

3.05 CASOS DE USO DE REALIZACIÓN

En los siguientes casos de uso de realización se describe la manera de cómo se realiza un proceso dentro del modelo de diseño.

UCR001 INGRESAR USUARIO

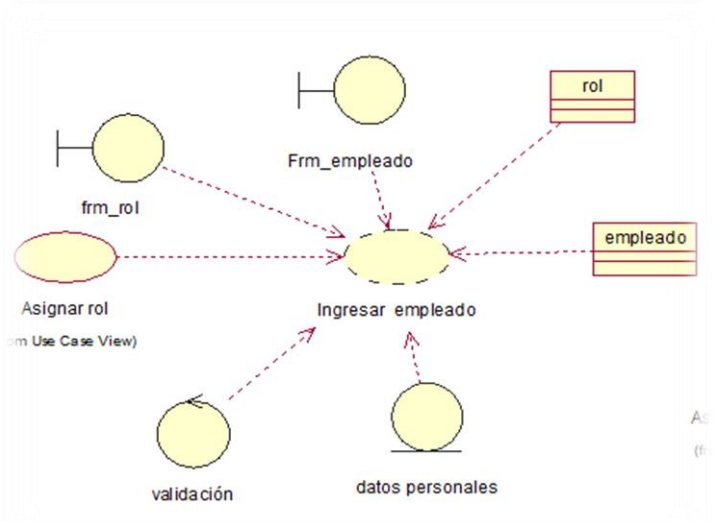


Figura 8: Caso de Uso de Realización (UCR001 Ingresar Usuario).
Se describe detalladamente el proceso de ingreso de empleado. Para cumplir con este proceso se va a necesitar el formulario de empleado y las tablas de empleado.

Tabla 17: UCR001 INGRESAR USUARIO

Nombre	**Ingreso de empleado**
Identificador	**UCR001**
Responsabilidades	**Gerente, Empleado**
Tipo	**Sistema**
Referencias Casos de Uso	**UC001, UC002**
Referencias Requisitos	**RF001, RF002**
PRECONDICIONES	
De Instancia	
El empleado debe contar con sus papeles en regla para ser registrado.	
POSCONDICIONES	
De Instancia	
Se registrara al empleado y se le asignará su respectivo rol.	
SALIDAS PANTALLA	
Se visualizará la información ingresada	

Nota: En este Caso de Uso de Realización se detalla el proceso de ingreso de Empleado.

UCR002 REALIZAR RESERVA

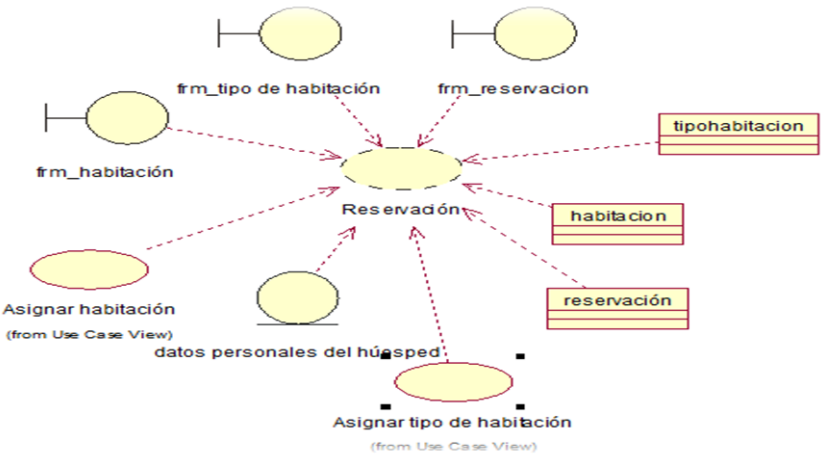


Figura 9: Caso de uso de Realización (UCR002 Realizar Reserva).
Se describe detalladamente el proceso de registro de reserva. Para cumplir con el actual el proceso se va a necesitar el formulario de reserva y las tablas de habitación, tipo habitación y la de reservación

Tabla 18: UCR002 REALIZAR RESERVA

Nombre	**Realizar Reserva**
Identificador	**UCR002**
Responsabilidades	**Recepcionista, Huésped**
Tipo	**Sistema**
Referencias Casos de Uso	**UC003**
Referencias Requisitos	**RF005**
PRECONDICIONES	
De Instancia **Debe estar debidamente registrada la reserva**	
POSCONDICIONES	
De Instancia **Debe haber datos**	
SALIDAS PANTALLA	
Se visualizará la información requerida.	

Nota: En este Caso de Uso de Realización se detalla el proceso de reserva
.

UCR003 GENERAR REPORTE

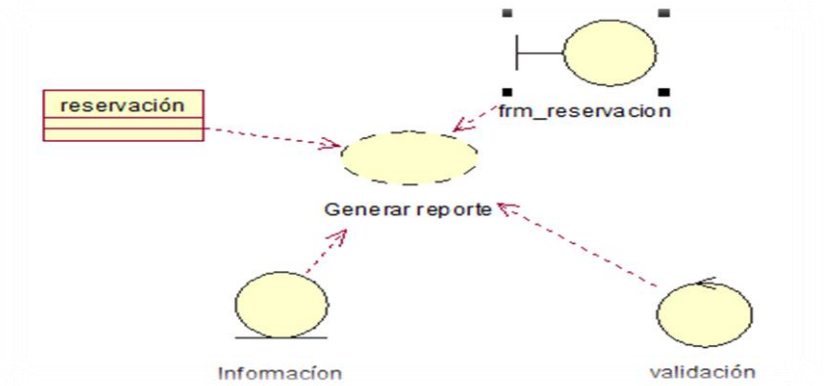


Figura 10: Caso de Uso de Realización (UCR003 Generar Reporte).
Se describe detalladamente el proceso de generación de reporte. Para cumplir con el actual proceso se va a necesitar el formulario y la tabla de reservación y se procede a validar información.

Tabla 19: UCR003 GENERAR REPORTE

Nombre	**Generar reporte**
Identificador	**UCR003**
Responsabilidades	**Administrador**
Tipo	**Sistema**
Referencias Casos de Uso	**UC002**
Referencias Requisitos	**RF003**
PRECONDICIONES	
De Instancia **El huésped debe solicitar la reserva.**	
POSCONDICIONES	
De Instancia **Se llenaran los campos de reserva.**	
SALIDAS PANTALLA **Al momento de registrar la reserva se visualizará el formulario de reserva de habitación**	

Nota: En este Caso de Uso de Realización se detalla el proceso de generar reporte.

3.06. DIAGRAMA DE SECUENCIAS DEL SISTEMA

En los siguientes Diagramas de Secuencia se detalla las interacciones que tienen entre sí los involucrados encontrados en el presente proyecto.

SEQ 001 REGISTRAR USUARIO

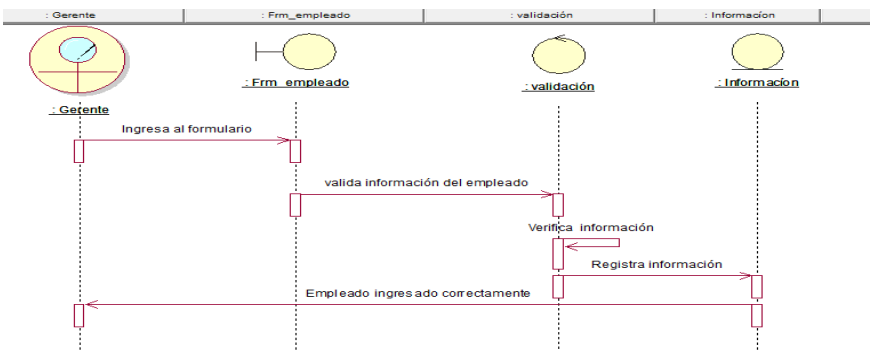


Figura 11: Diagrama de Secuencia (SEQ 001 Registrar Usuario).
En este diagrama se describe las tareas que se va a realizar en el proceso de registro de empleado.

SEQ 002 REALIZAR RESERVA

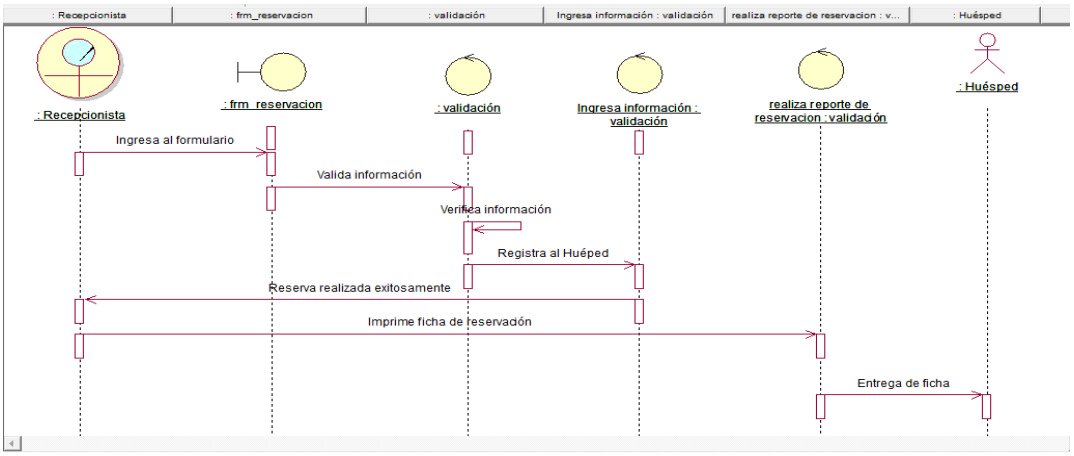


Figura 12: Diagrama de Secuencia (SEQ 002 Realizar Reserva).
En este diagrama se describe las tareas que se va a realizar en el proceso de realizar reserva.

SEQ 003 GENERAR REPORTE

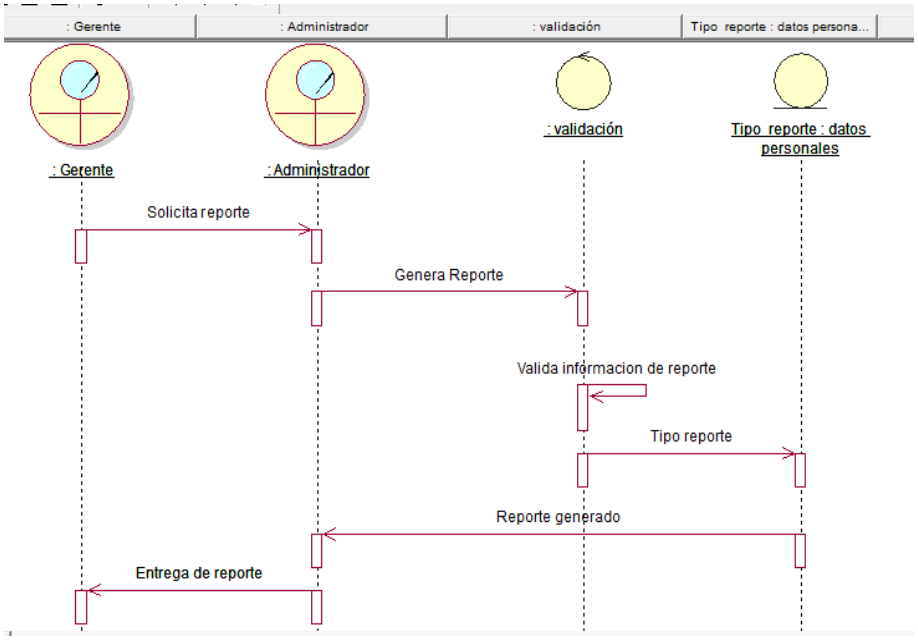


Figura 13: Diagrama de Secuencia (SEQ 003 Generar Reporte).
En este diagrama se describe las tareas que se va a realizar en el proceso de generar reporte.

SEQ 004 REALIZAR BÚSQUEDA

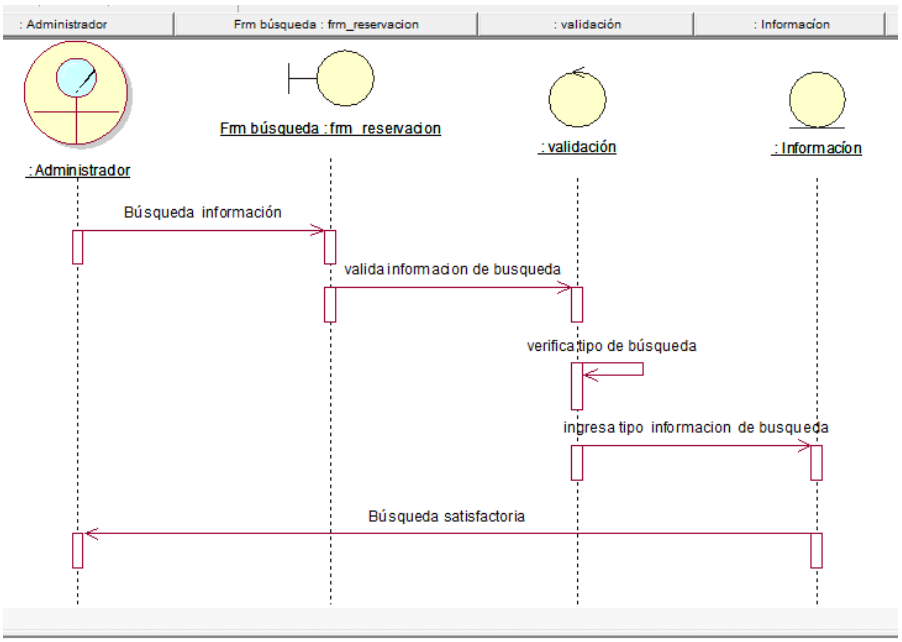


Figura 14: Diagrama de Secuencia (SEQ 004 Realizar Búsqueda).
En este diagrama se describe las tareas que se va a realizar en el proceso de generar búsquedas.

Capítulo IV: ANÁLISIS DE ALTERNATIVAS

4.01. MATRIZ DE ANÁLISIS DE ALTERNATIVAS

En la presente matriz se identifica las alternativas a partir del árbol de objetivos y cada una de ellas se cataloga de acuerdo al Impacto sobre el propósito y la factibilidad técnica, financiera, social y política con una escala del 1 al 5.

Tabla 20: Matriz de Análisis de Alternativas

Metas	Impacto sobre el propósito	Factibilidad Técnica	Factibilidad Financiera	Factibilidad Social	Factibilidad Política	Puntaje	Prioridad
Registro de reservas realizadas de forma portátil	5	5	3	4	5	22	Alta
Sistema Informático	5	4	4	3	5	21	Media Alta
Agilidad en Actualización de Datos	5	4	3	4	5	21	Alta
Buen manejo de Información	5	5	3	4	5	22	Alta
Buenas estrategias Institucional	5	4	4	3	4	20	Media
Alto Rendimiento	5	4	4	3	5	21	Media Alta

Empresarial							
Recuperación de Datos	5	4	3	4	4	20	Media
Información Completa y Actualizada	5	4	3	4	5	21	Media Alta
Buena Planificación	5	4	4	4	5	22	Alta
Servicio Eficiente	5	4	4	3	5	21	Alta
Satisfacción del huésped	5	4	4	3	4	20	Alta
Alto posicionamiento en el mercado Hotelero	5	5	3	4	5	22	Alta
Ganancias Económicas	5	5	4	4	5	23	Media Alta
TOTAL	65	56	46	47	62	276	

Nota: En la presente matriz se califica mediante una escala del 1 al 5
Escala:
1= Bajo
2= Medio Bajo
3= Medio
4= Medio Alto
5= Alto

4.02. MATRIZ DE IMPACTOS DE OBJETIVOS

En la siguiente matriz se detalla la factibilidad, Impacto de Género y Ambiental entre otros aspectos como la relevancia y a sostenibilidad de cada uno de los objetivos ya que son de gran importancia para llegar al cambio deseado.

Tabla 21: Matriz de Impactos de Objetivos

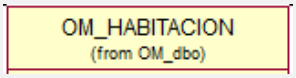
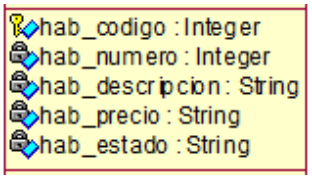
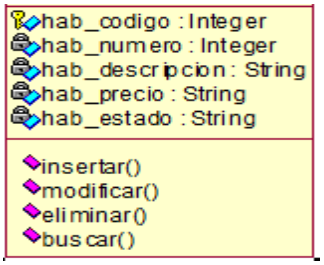
Objetivos	Factibilidad a Lograrse	Impacto de género	Impacto Ambiental	Relevancia	Sostenibilidad
Registrar correctamente las reservas en el sistema	Mantener de una manera organizada las reservas en el Hostal.	El registro de reservas se podrá hacerlo el encargado de recepción.	Mejora en el ámbito laboral. Optimizar el tiempo de registro.	Permitirá tener un orden adecuado.	El registro de reservas es vital ya que gracias a ello se puede tener una mejor organización.
Agilizar la Actualización de los Datos.	La información será verídica y eficaz.	Contratación de personal para la manipulación de datos	El proyecto impulsara el reciclaje tecnológico y uso eficiente de energía.	Utilización adecuada de la Tecnología	El personal del Hostal será debidamente capacitado.
Evitar pérdidas de información.	Los Datos ingresados deben ser confiables para su posterior uso	Se han contratado personal que sean responsables para el manejo de información.	Mejorar el tiempo de búsqueda de información.	Evitará la pérdida de información en un futro inmediato.	El personal del Hostal será debidamente capacitado para el manejo de información.
Incrementar la demanda de huéspedes en el hostal.	Los Datos de los Huéspedes deben ser verdaderos	El registro de reservas se podrá hacerlo el encargado de recepción.	Mejorar el ambiente laboral.	Permite un control adecuado de los datos de los huéspedes	Los huéspedes son vitales para el funcionamiento del hostal.
Evitar pérdidas económicas	Comparar los precios de los diferentes Hostales.	Se han contratado personal para la administración de la empresa.	Mejorar el ambiente laboral.	Permitirá un control adecuado de los registros.	Con el sistema se pretende mejorar la calidad de servicio.

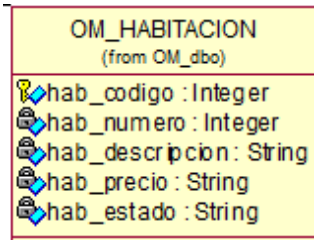
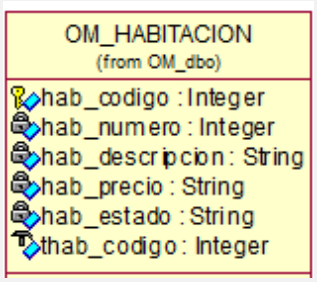
Nota: Matriz de Impactos de Objetivos. En esta matriz se detalla los objetivos con cada uno de los aspectos descritos.

4.03. ESTÁNDARES PARA EL DISEÑO DE CLASES

La presente tabla se detalla de cómo se va a diseñar el diagrama de clases adecuadamente teniendo en cuenta de cómo se va organizando los estándares a utilizar para el desarrollo del mismo.

Tabla 22: Matriz de Análisis de Alternativas

Nombre	Gráfico	Estándar
Clase		Las clases nos permite representar entidades lo cual será detallado como el nombre principal de cada una de las tablas. Ej.: OM_HABITACION
Atributos		Se detalla con palabras claves los atributos. Ej.: hab_numero
Atributos Públicos		Indica que el atributo será visible tanto dentro como fuera de la clase
Atributos Privados		Indica que el atributo sólo será accesible desde dentro de la clase
Atributos Protegidos		Indica que el atributo no será accesible desde fuera de la clase, pero si podrá ser accesado por métodos de la clase
Métodos		Los métodos se escriben en letras minúsculas detallando lo que se va a realizar en cada tabla. Ej.: insertar()

Métodos Públicos		Indica que el método será visible tanto dentro como fuera de la clase
Método Privados		Indica que el método sólo será accesible desde dentro de la clase
Métodos Protegidos		Indica que el método no será accesible desde fuera de la clase, pero si podrá ser accesado por métodos de la clase
Primary Key		Se denomina Primary Key al atributo que por lo general siempre va marcado como una especie de llave de color dorado, de tal manera de que todas las tablas tienen que llevar una. Ej.: hab_codigo
Foreing Key		Se denomina Foreing Key al atributo que por lo general siempre va marcado como una especie de t, de tal manera que cada clave foránea lleva la misma identificación de la clave primaria que hereda Ej.: thab_codigo
Relación		Nos permite relacionar las tablas para obtener la herencia de las mismas.

Nota: Estándares para el diseño de clases. Son de vital importancia ya que nos permiten diseñar el diagrama de Clases.

4.04. DIAGRAMA DE CLASES

El actual Diagrama se especifica la manera de como se constituye la base de datos importada al la herramienta Rational Rose. Se visualiza las tablas con sus respectivos atributos y métodos a utilizar.

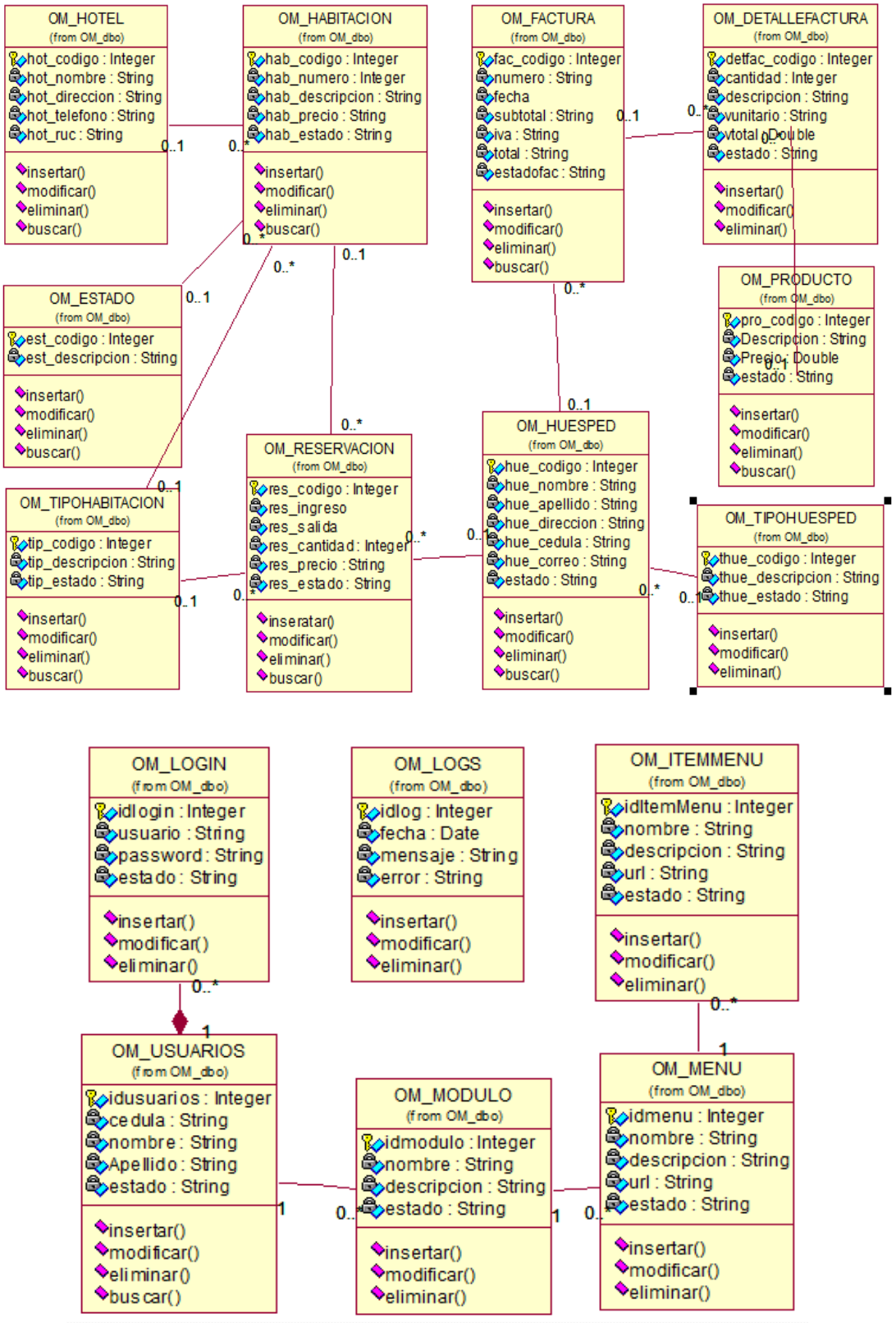
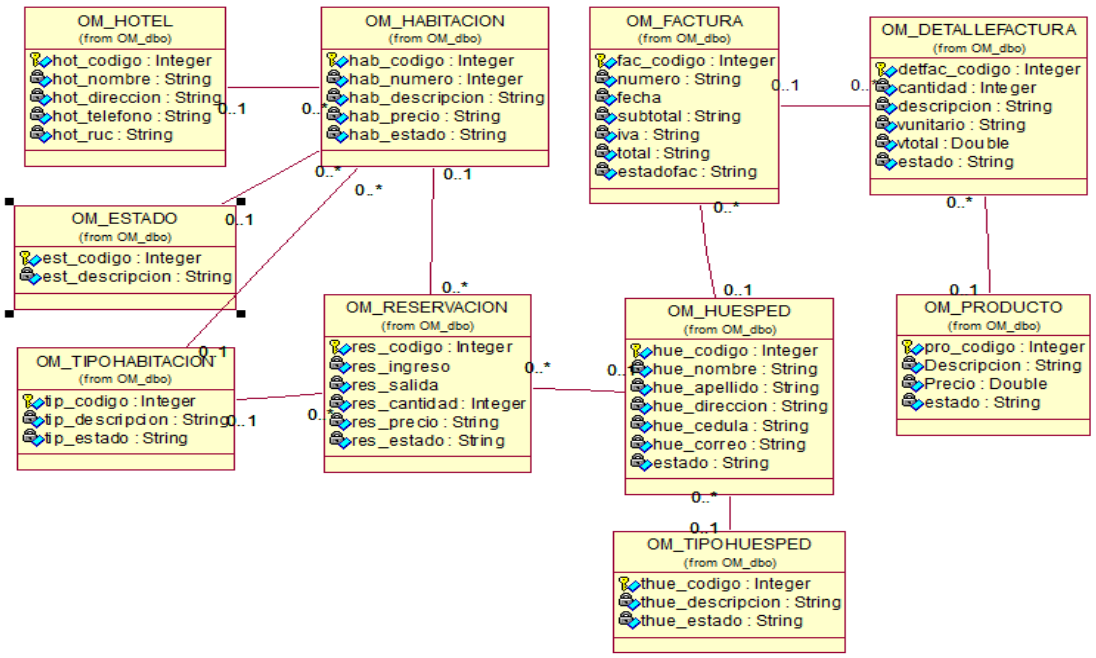


Figura 15: Diagrama de Clases.
Especifica cómo se constituye el sistema donde se visualiza sus clases respectivos métodos y atributos.

4.05. MODELO LÓGICO – FÍSICO

(Tuza, 2014) Señala que:

En los presentes diagramas nos permite visualizar las tablas del sistema tanto lógico como fisicamente con sus respectivos atributos y relaciones, detallando cada atributo como tipo de variable si es PRIMARY KEY (PK) o FOREING KEY (FK) respectivamente. Modelo Físico (Ver ANEXO A002).



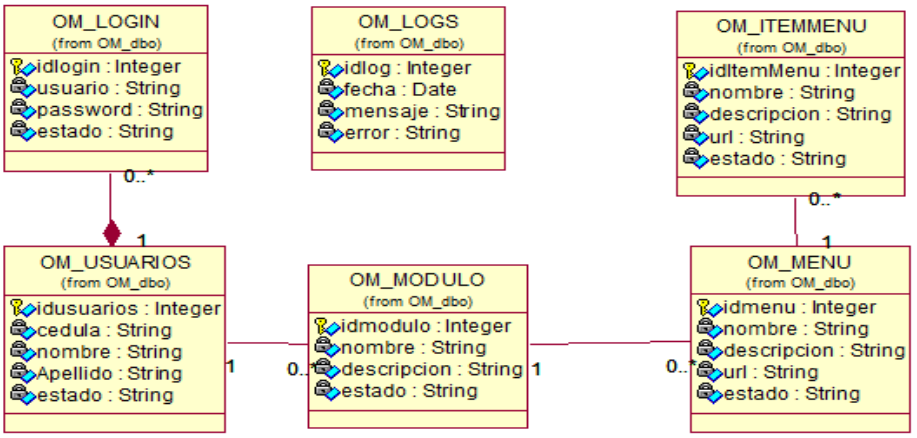


Figura 16: Diagrama de Clases (Modelo Lógico).
El presente diagrama especifica cómo se constituye el sistema de forma física lógica.

4.06. DIAGRAMA DE COMPONENTES

El presente Diagrama de Componentes se realiza un mapeo de diseño de todos los componentes con los que se va a trabajar tales como el Framework, Base de Datos entre otros los cuales nos permite el desarrollo del proyecto.

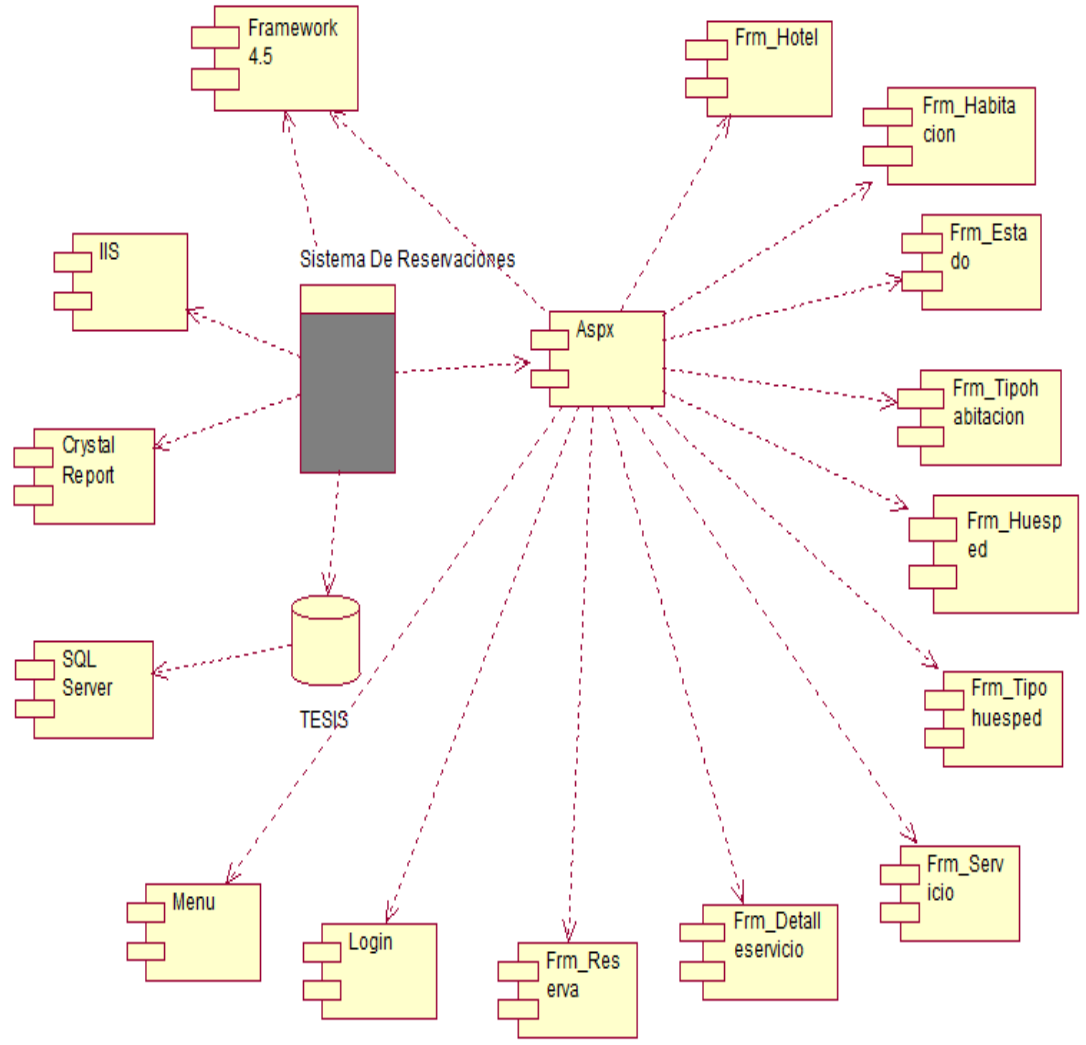


Figura 17: Diagrama de Componentes.
El presente diagrama especifica cómo se constituye el sistema de forma física conjuntamente con sus atributos.

4.07. DIAGRAMA DE ESTRATEGIAS

El presente Diagrama de Estrategias parte como punto principal el objetivo esto proviene de un propósito que se ha determinado en los distintos componentes anteriormente.

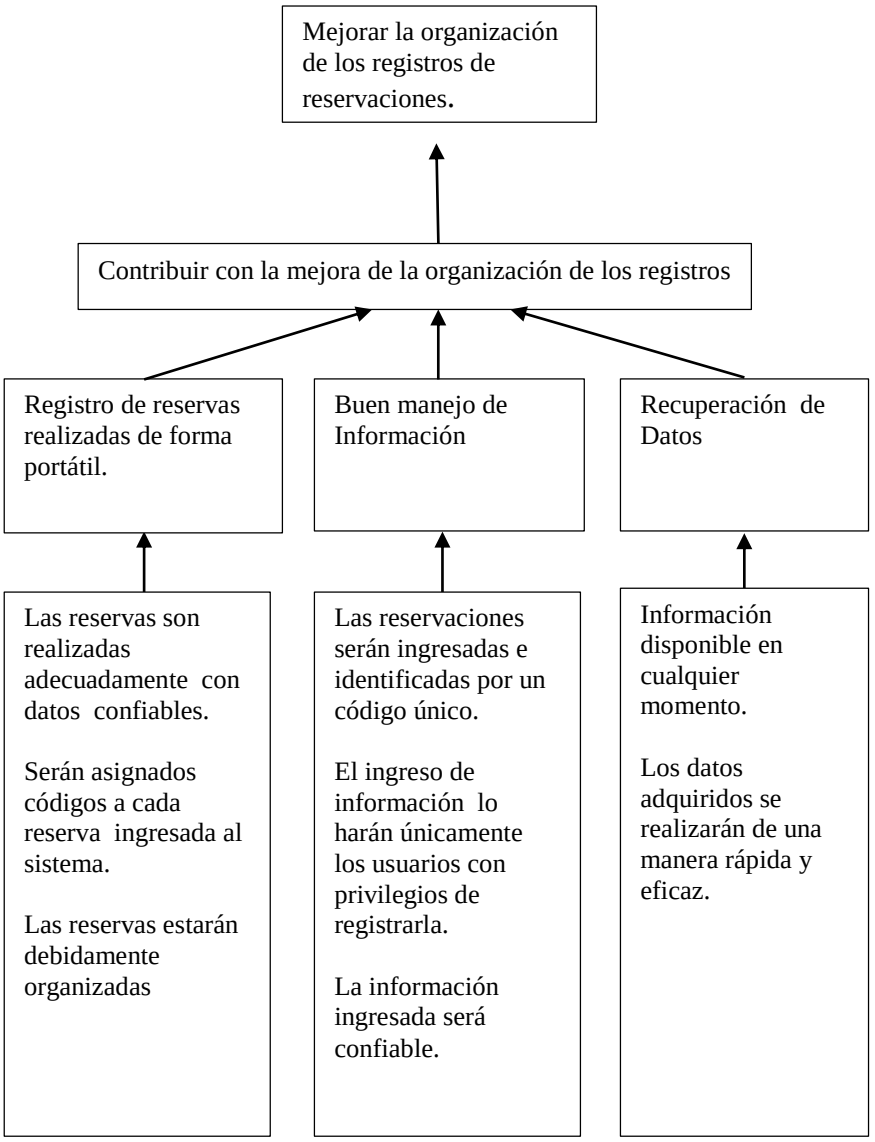


Figura 18: Diagrama de Estrategias.
El presente diagrama se detalla el objetivo planteado anteriormente.

4.08. MATRIZ DE MARCO LÓGICO

La Matriz de Marco Lógico nos permite verificar el resumen narrativo, indicadores, medios de verificación y los supuestos a través de la finalidad, propósito, componentes y las actividades del presente proyecto.

Tabla 23: Matriz de Marco Lógico

Resumen Narrativo	Indicadores	Medios de Verificación	Supuestos
Finalidad			
Registrar correctamente las reservas en el sistema.	Incremento de los huéspedes.	Lo efectuará el recepcionista.	Buen control de los registros de Reservaciones.
Propósito			
Eficiente gestión del proceso de reservación.	Satisfacción por parte de los huéspedes.	Información real de los clientes.	Información disponible.
Componentes			
Falta de control del proceso de reservación.	Número de registros de reservaciones en el periodo.	Registro de reservaciones de una forma adecuada.	Aplicación continúa del sistema informático en el hostal.
Inexistencia de un Sistema informático.	Registros auténticos.	Sistema Hotelero.	
Actividades			
Asegurar un buen registro de reservaciones.	Disponibilidad de la información en los momentos requeridos.	Registros. Reportes.	Reportes constantes. Control de las reservaciones.
Asegurar la existencia real de la información brindada por los huéspedes.	Agilidad al momento de adquirir información.		

Nota: Matriz de Marco Lógico. Nos permite identificar ciertos componentes para nuestro proyecto.

4.09. VISTAS ARQUITECTÓNICAS

4.01.01. VISTA LÓGICA

El presente figura nos permite visualizar como va estar desarrollado el sistema lógicamente, tomando como punto de partida la Base de datos,

seguidamente las capas a utilizarse y por último la interfaz que se va a utilizar para el desarrollo del mismo.

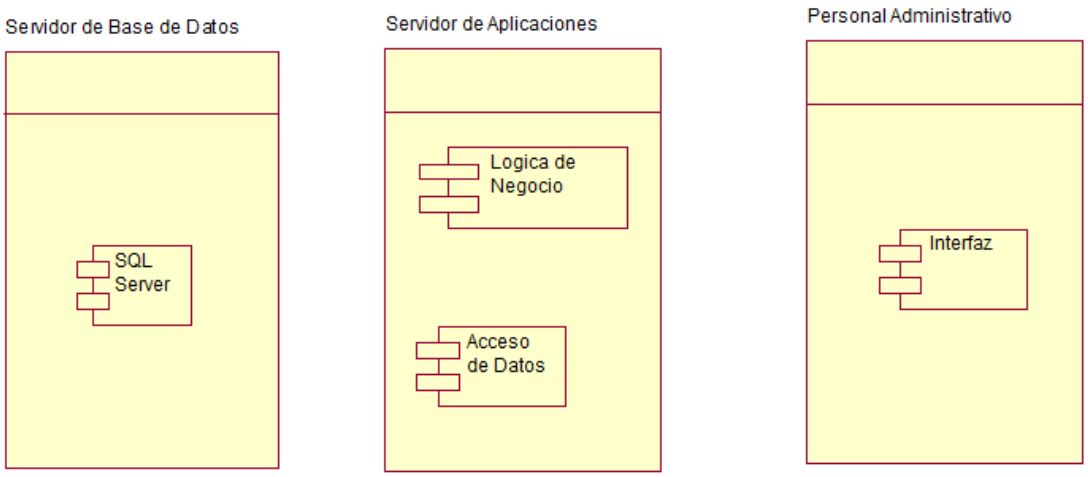


Figura 19: Vista Lógica.
El presente figura se detalla de cómo va estar compuesta nuestro sistema lógicamente.

4.01.02. VISTA FÍSICA

En la vista física se detalla más preciso los componentes qu es eva a utilizar como el motor de Base de Datos y mediante las flechas nos mustra cual va ser el flujo que va a seguir el sistema

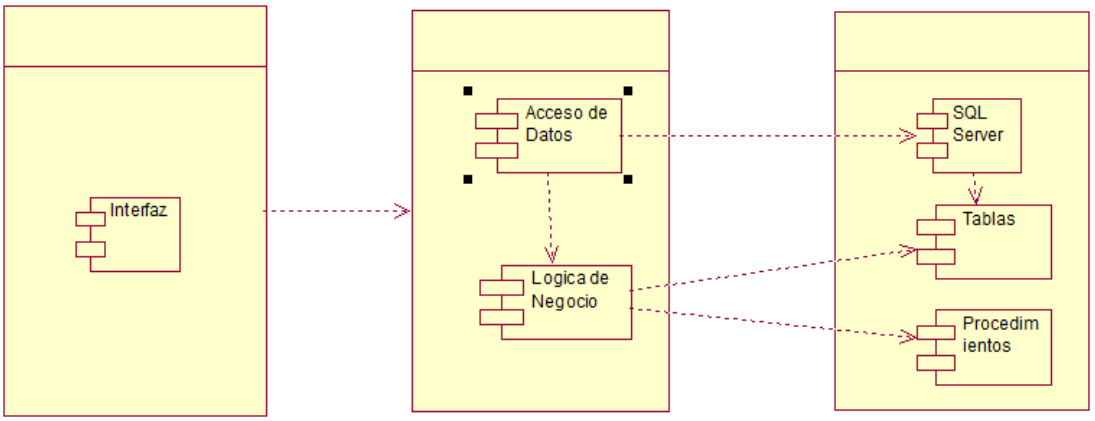


Figura 20: Vista Física.
El presente figura se detalla de cómo va estar compuesta nuestro sistema físicamente.

4.01.03. VISTA DE DESARROLLO

La aplicación web se la plantea por capas, teniendo los módulos tales como, seguridad, mantenimiento entre otros, los cuales van a desarrollar los procesos de reservaciones del hostal.

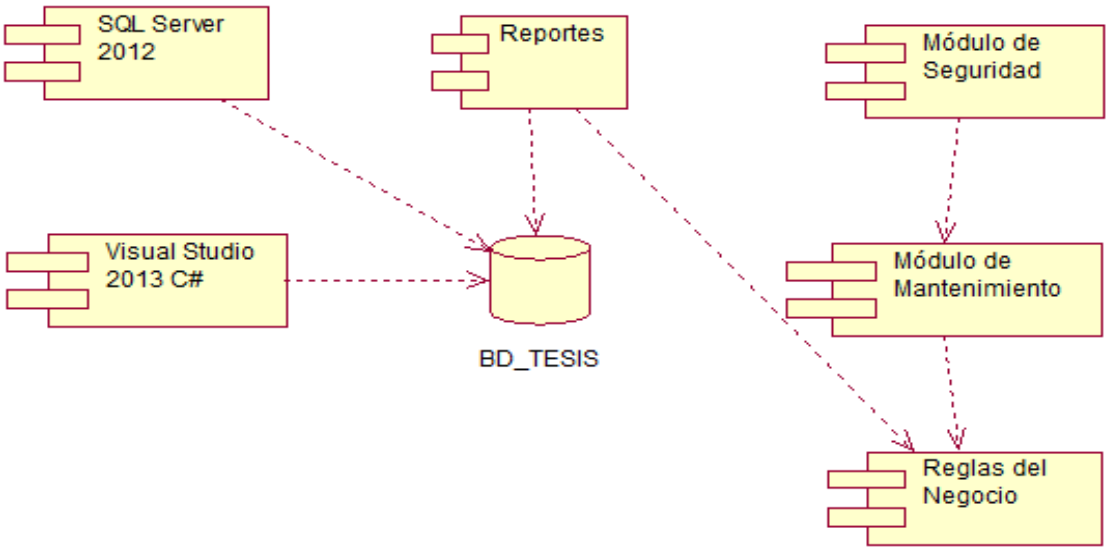


Figura 21: Vista de Desarrollo.
En la presente figura se detalla de cómo va a estar desarrollado nuestro sistema.

4.01.04. VISTA DE PROCESOS

En las siguientes figuras se detallan y se visualizan cada una de las actividades que se va a realizar dentro de cada proceso de una manera ordenada y sistemática los cuales nos permiten comprender mejor el desarrollo del proyecto.

Vista de Procesos Registrar Usuario

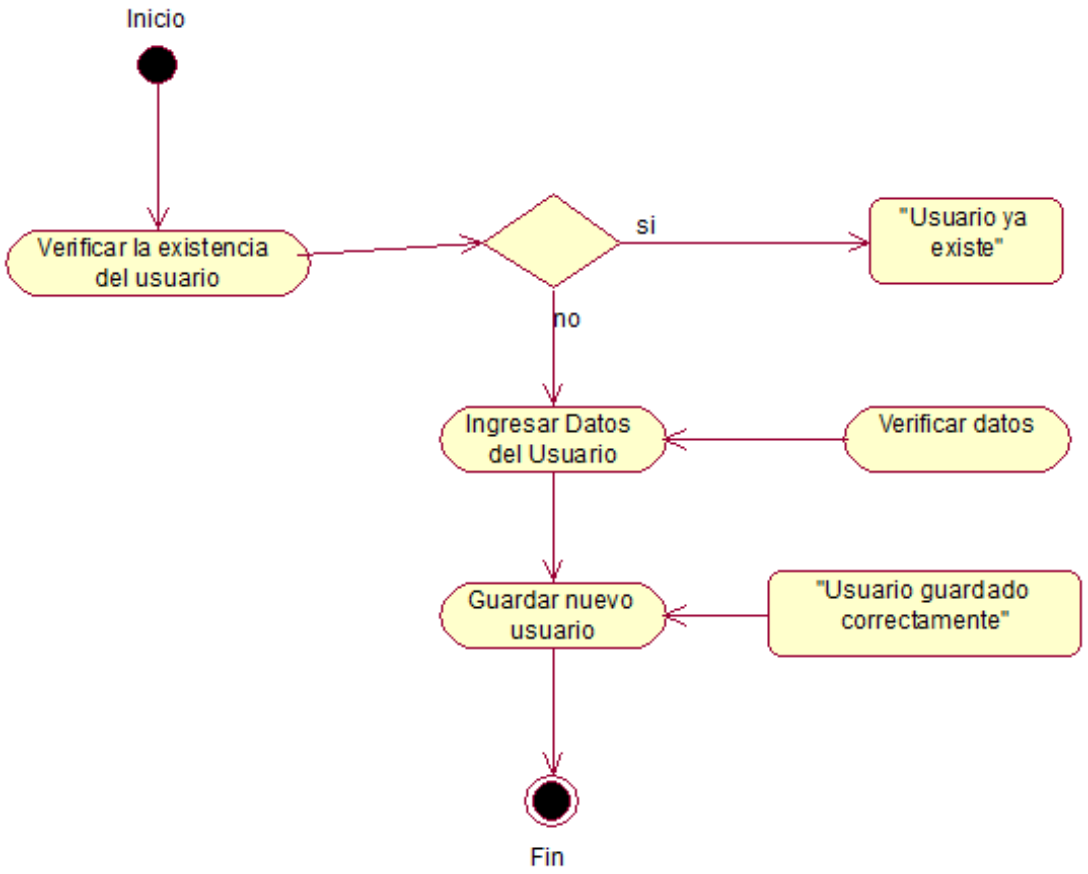


Figura 22: Vista de Proceso (Registrar Usuario)
En la presente figura se detalla las actividades a realizar para el registro de usuario.

Vista de Proceso Realizar Reserva

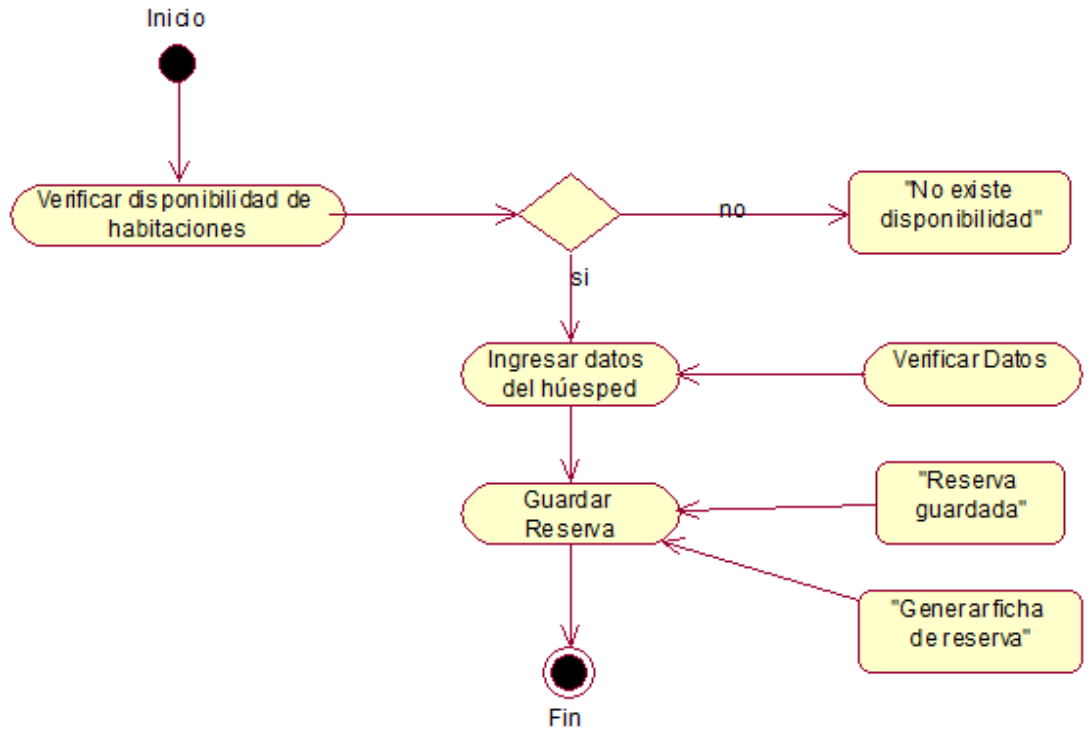


Figura 23: Vista de Proceso (Realizar Reserva).
En la presente figura se detalla las actividades a realizar para el registro de reserva.

Vista de Proceso Realizar Reservación.

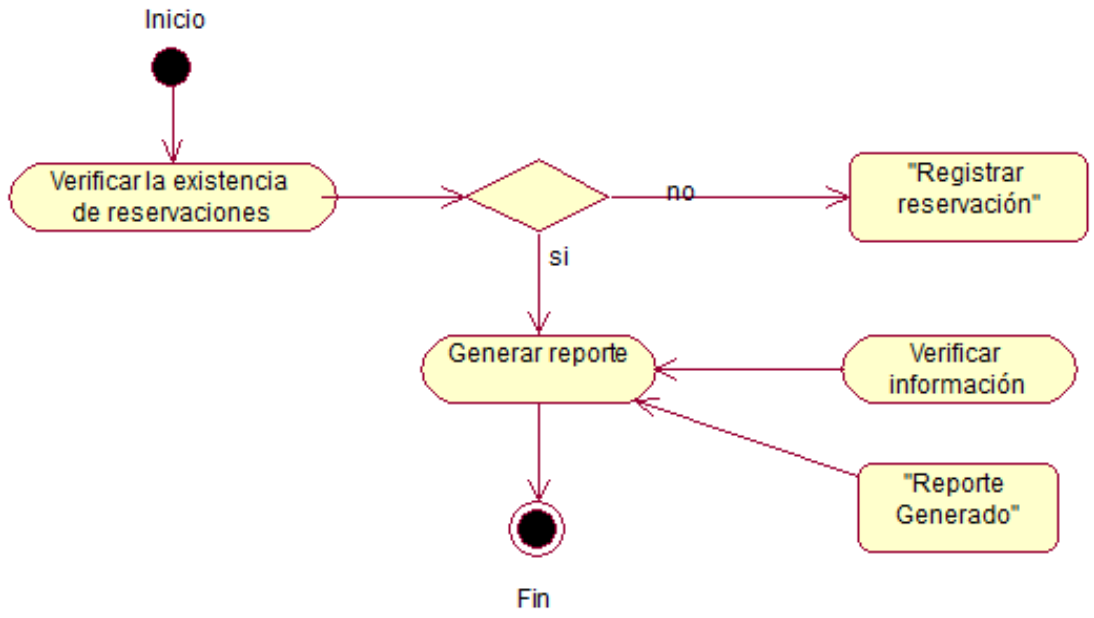


Figura 24: Vista de Proceso (Generar Reporte).
En la presente figura se detalla las actividades a realizar para generar el reporte

Capítulo V: PROPUESTA

5.01. ESPECIFICACIÓN DE ESTÁNDARES DE PROGRAMACIÒN

Los Estándares de Programación nos permiten identificar el desarrollo de nuestro proyecto además la forma de cómo se implementa el código fuente en el mismo.

"Un estándar de programación es: Una forma de "normalizar" la programación de forma tal que al trabajar en un proyecto, cualquier persona involucrada en el mismo tenga acceso y comprenda el código. (Will.i., 2010)

Tabla 24: Estándares de Programación

OBJETO	PREFIJO	DESCRIPCIÓN
Button	Btn	Esto causa una acción al momento de pulsar el botón. Ejm: Btn_Guardar
Checkbox	Chk	Se usa para visualizar al usuario un tipo de selección. Ejm: Chk_Estado
Textbox	Txt	Conocida como caja de texto se utiliza para insertar y reflejar datos. Ejm: Txt_Nombre
Label	Lbl	El objeto Label es un elemento que nos permite insertar texto. Ejm: Lbl_Nombre
ImageButton	Imb	Estos Imagen-Botón se puede configurar y esto causa una acción al momento de presionar. Ejm: Imb_Nuevo
Dropdownlist	Cmb	Se usa para visualizar datos en un cuadro

		Ejm: Cmb_TipoHotel
Image	Img	El objeto image representa a las imágenes gráficas. Ejm: Img_ingreso.
GridView	Gvd	Muestra de una forma adecuada los datos en forma de una tabla. Ejm: Gdv_Datos
Clase	Cls	Se utiliza como un control de la aplicación.
Formulario	Frm	Muestra de una forma adecuada los datos en forma de una tabla. Ejm: Frm_Hotel
CrystalReports	Rpt	Se utiliza para representar de una manera eficaz y verdadera los datos.

Nota: Estándares de Programación. Nos permite identificar la manera de cómo se desarrolla nuestro proyecto.

Arquitectura de Desarrollo

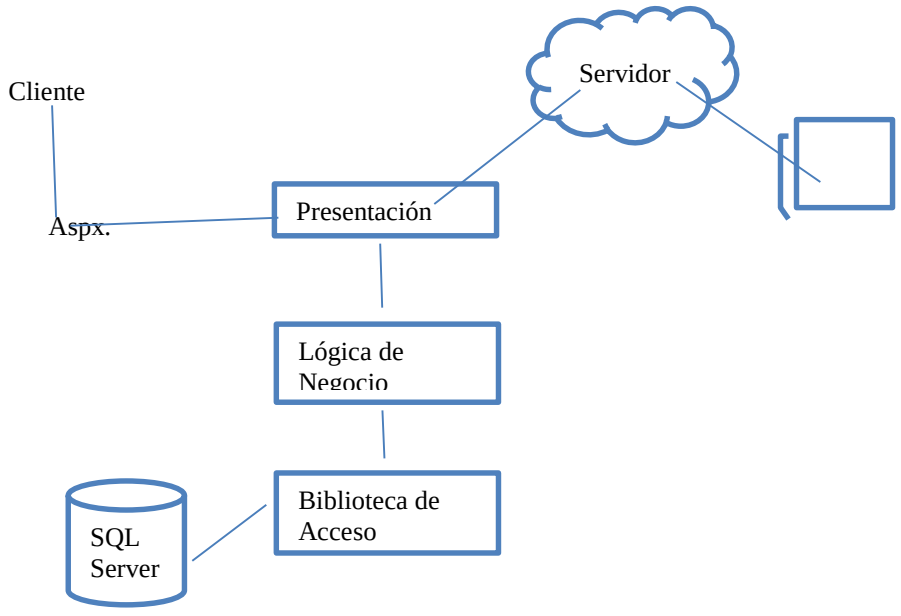


Figura 25: Arquitectura de Desarrollo
En la presente figura se detalla la arquitectura que vamos a utilizar para el desarrollo de nuestro proyecto

Arquitectura en 3-Capas

(Luis, 2014) Señala que:

Capa de Presentación: La capa de presentación proporciona la interfaz gráfica quien es la que se ocupa de interactuar con el cliente, es decir, son aquellas ventanas, avisos, cuadros de diálogos, páginas web entre otros, que el cliente final emplea para interactuar con el aplicativo web; dicha capa se interactúa con la capa de Lógica de Negocio, enviando y solicitando información.

Capa Lógica de Negocio: La capa de Lógica de Negocio es la encargada de establecer valga la redundancia la lógica del negocio , de tal manera, que todo lo que el Aplicativo debe tener en cuenta precedentemente de ejecutar una operación, dicha capa se comunica con la capa de acceso de datos realizando solicitudes a la misma.

Capa Acceso de Datos: La Capa de Acceso de Datos es quien está encargada de establecer la relación con la base de datos, en esta capa reposaran todas las operaciones tales como, Crear, Leer, Modificar y Eliminar conocido por sus siglas (CRUD); además es quien da a conocer que motor de base de datos estamos utilizando, asimismo es la encargad de recibir las solicitudes de la Capa de Lógica de Negocio, la cual efectúa dichas operaciones y devuelve los resultados a la misma capa quien emitió las solicitudes.

5.02. DISEÑO DE INTERFACES DE USUARIO

En las presentes interfaces se visualiza el diseño del proyecto y la manera de cómo va interactuar con el Usuario además de los objetos que se utiliza para el desarrollo del mismo y esto debe de ser de forma comprensiva y llamativa.

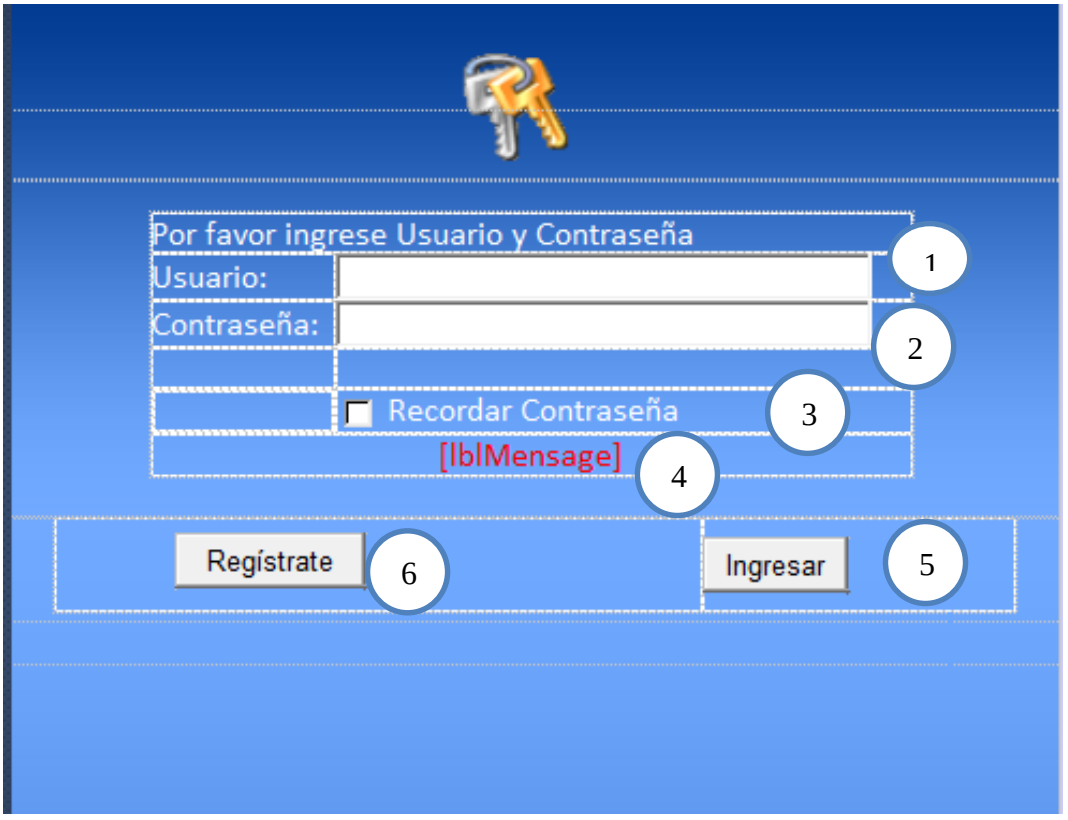


Figura 26: Diseño de Interface (Ingreso al Sistema)
En la presente figura se visualiza la interfaz gráfica para el ingreso al sistema de una forma adecuada y sistemática.

Tabla 25: Ingreso al Sistema

NUMERACIÓN	REPRESENTACIÓN	PREFIJO	DESCRIPCIÓN
1	Textbox	Txt	Nombre de Usuario
2	Textbox	Txt	Contraseña
3	Checkbox	Chk	Recordar Contraseña
4	Label	Lbl	Mensaje
5	Button	Btn	Inicio de Sesión
6	Button	Btn	Regístrate

Nota: Ingreso al Sistema. Nos permite identificar la manera de cómo se desarrolla nuestro proyecto para el ingreso al sistema.

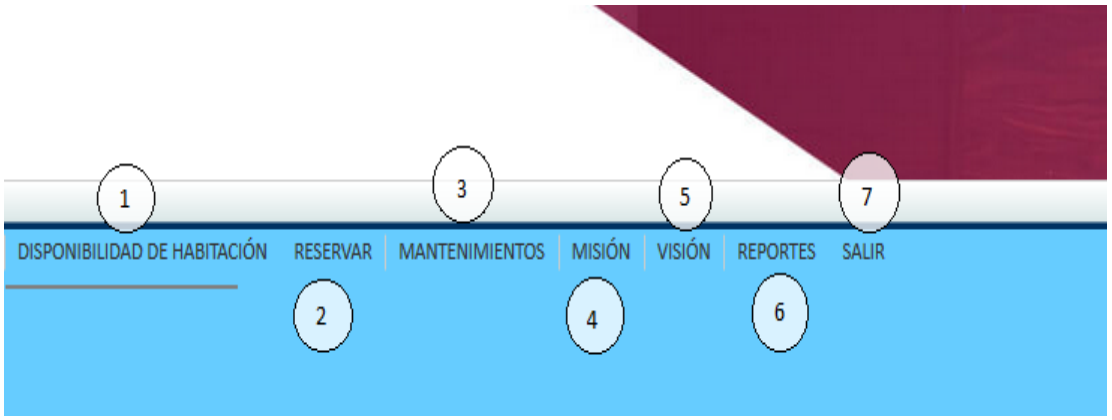


Figura 27: Diseño de Interface (Pantalla de Inicio)
En la presente figura se visualiza la interfaz gráfica de la pantalla de Inicio del Sistema de una manera agradable.

Tabla 26: Pantalla de Inicio

NUMERACIÓN	REPRESENTACIÓN	DESCRIPCIÓN
1	Textbox	Nombre de Usuario
2	Textbox	Disponibilidad de Habitación
3	Textbox	Reservar
4	Textbox	Mantenimientos
5	Textbox	Misión
6	Textbox	Visión
7	Textbox	Salir

Nota: Pantalla de Inicio. Nos permite identificar la manera de cómo se desarrolla nuestro proyecto para la pantalla de inicio.

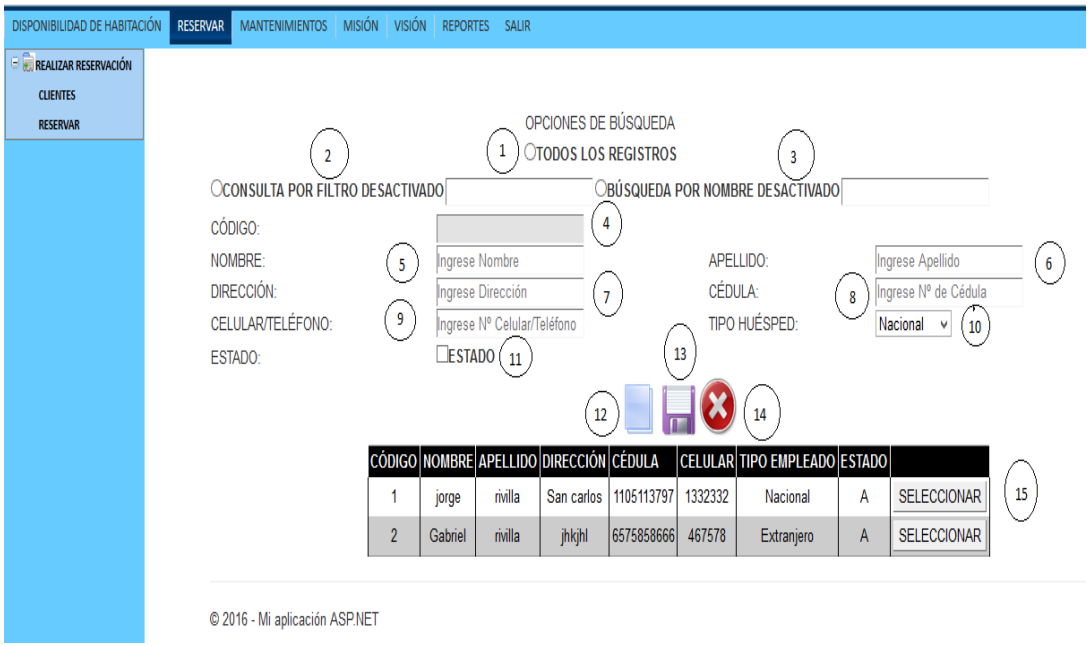


Figura 28: Diseño de Interface (Registro del Huésped)
En la presente figura se visualiza la interfaz gráfica de la pantalla de ingreso de Huésped una manera didáctica.

Tabla 27: Registro De Huésped

NUMERACIÓN	REPRESENTACIÓN	DESCRIPCIÓN
1	RadioButton	Todos los Registros
2	RadioButton ,Textbox	Consulta por Filtro
3	RadioButton ,Textbox	Consulta por Nombre
4	Textbox	Código
5	Textbox	Nombre
6	Textbox	Apellido
7	Textbox	Dirección
8	Textbox	Cédula
9	Textbox	Celular/Teléfono
10	Dropdownlist	Tipo Huésped
11	CheckList	Estado
12	ImageButton	Nuevo
13	ImageButton	Guardar
14	ImageButton	Eliminar
15	GridView	Datos

Nota: Registro de Huésped. Nos permite identificar la manera de cómo se desarrolla nuestro proyecto para el ingreso del Huésped.

Figura 29: Diseño de Interface (Realizar Reserva)
En la presente figura se visualiza la interfaz gráfica de la pantalla de ingreso de Reserva una manera didáctica.

Tabla 28: Realizar Reserva

NUMERACIÓN	REPRESENTACIÓN	DESCRIPCIÓN
1	Textbox	Código
2	Textbox	Número
3	Dropdownlist	Nombre de Huésped
4	Dropdownlist	Tipo habitación
5	Textbox	Fecha Ingreso
6	Textbox	Fecha Salida
7	Textbox	Número de Personas
8	Textbox	Precio
9	CheckList	Estado
10	ImageButton	Nuevo
11	ImageButton	Guardar
12	ImageButton	Eliminar

Nota: Realizar Reserva. Nos permite identificar la manera de cómo se desarrolla nuestro proyecto para el registro de reserva

5.03. ESPECIFICACIÓN DE PRUEBAS DE UNIDAD

En las siguiente tablas de muestra de forma detallada la implementación de pruebas de unidad sobre la aplicación web y se evalúan las distintas actividades realizadas durante el funcionamiento del Sistema.

Tabla 29: Ingreso al Sistema

Identificador de la Prueba:	EPU001
Método a Probar	Ingreso al sistema con autentificación de Usuario
Objetivo de la Prueba:	Garantizar que al acceso al Sistema lo realice el personal registrado para la manipulación del mismo.
Datos De Entrada	
Usuario y Contraseña	
Resultado Esperado	
Acceso al Sistema y manipulación de una manera segura.	
Comentarios	
Los campos a insertar son obligatorios	

Nota: Ingreso al Sistema. Nos permite identificar detalladamente la forma adecuada de evaluación del ingreso al Sistema.

Tabla 30: Ingreso de Información

Identificador de la Prueba:	EPU002
Método a Probar	Ingreso de Información
Objetivo de la Prueba:	Asegurar que el ingreso de información se realice de manera eficaz y correcta, teniendo en cuenta los campos restringidos a números, letras, fechas.
Datos De Entrada	
Datos Varios	
Resultado Esperado	
Adecuado Ingreso de Información en los diferentes campos.	
Comentarios	
Solo se Aceptan Letras; Solo se Aceptan Números.	

Nota: Ingreso de Información. Nos permite identificar detalladamente la forma adecuada de evaluación de ingreso de información.

Tabla 31: Modificar Información

Identificador de la Prueba:	EPU003
Método a Probar	Modificar Información
Objetivo de la Prueba:	Asegurar que la modificación de información se realice de una manera adecuada, teniendo en cuenta los campos restringidos a números, letras, fechas.
Datos De Entrada	
Seleccionar	
Resultado Esperado	
Adecuada modificación de los datos teniendo en cuanta las restricciones.	
Comentarios	
Solo se Aceptan Letras; Solo se Aceptan Números	

Nota: Modificar Información. Nos permite identificar detalladamente la forma adecuada de evaluación de modificación de información.

Tabla 32: Eliminar Información

Identificador de la Prueba:	EPU004
Método a Probar	Eliminar Registros
Objetivo de la Prueba:	Garantizar que el borrado de registros se realiza de una forma adecuada, teniendo presente que se lo realiza de manera lógica.
Datos De Entrada	
Seleccionar	
Resultado Esperado	
Registros Existentes en la base de Datos de forma inactiva.	
Comentarios	
Registro Eliminado Correctamente	

Nota: Eliminar Información. Nos permite identificar detalladamente la forma adecuada de evaluación de como eliminar la información.

5.04. ESPECIFICACIÓN DE PRUEBAS DE ACEPTACIÓN

En las presentes tablas se especifica detalladamente las pruebas de aceptación las cuales son de muy importancia ya que son los que nos van a poder permitir identificar si cumplen con los estándares necesarios y poder satisfacer los requerimientos propuestos anteriormente.

Tabla 33: Registro de Usuario

Identificador de la Prueba:	EPA001
Caso de uso	UC001
Tipo de usuario	Administrador
Objetivo de la Prueba:	Ingresar a los Usuarios de forma correcta al Sistema
Secuencia de eventos	
Registro, Autentificación, Validación, Ingreso	
Resultados Esperados	
Ingreso de los Usuarios de una manera segura	
Comentarios	
El Sistema permitirá el Ingreso únicamente el personal registrado	
Estado : No Aceptado	

Nota: Registro de Usuario. Nos permite identificar detalladamente la forma adecuada de aceptación del registro de Usuario.

Tabla 34: Ingreso de Reserva

Identificador de la Prueba:	EPA002
Caso de uso	UC002
Tipo de usuario	Recepcionista
Objetivo de la Prueba:	Verificar las validaciones de los campos estén funcionando y realizar correctamente la reserva.
Secuencia de eventos	
Registro, Autentificación, Ingreso, Almacenamiento.	
Resultados Esperados	
Ingreso de información de una manera segura.	
Comentarios	
El Sistema muestra una ficha al momento de ingresar a información.	
Estado : No Aceptado	

Nota: Registro de Reserva. Nos permite identificar detalladamente la forma adecuada de aceptación del registro de Reserva.

Tabla 35: Generar Reportes

Identificador de la Prueba:	EPA003
Caso de uso	UC003
Tipo de usuario	Recepcionista
Objetivo de la Prueba:	Evidenciar la forma eficaz al momento de generar el reporte.
Secuencia de eventos	
Búsqueda e Impresión.	
Resultados Esperados	
Reporte generado Exitosamente.	
Comentarios	
Ninguno	
Estado : Aceptado	

Nota: Generar Reporte, Nos permite identificar detalladamente la forma adecuada de aceptación al momento de generar reporte.

5.05. ESPECIFICACIÓN DE PRUEBAS DE CARGA

En las Presentes Tablas se muestran de una forma detallada las pruebas de carga que se le va a realizar al Sistema y de esta manera evaluar la capacidad que presenta para el almacenamiento de la información.

Tabla 36: Subida Masiva de Información

Identificador de la Prueba:	EPC001
Tipo de prueba	Funcionamiento del sistema con una subida masiva de información.
Objetivo de la Prueba:	Verificar el comportamiento del sistema al subir gran cantidad de información en el mismo.
Descripción	
Se requiere de una plataforma de pruebas de carga para simular el escenario para aplicaciones web.	
Resultados Esperados	
Saber la cantidad de información que permite subir al sistema.	
Comentarios	
En Revisión	

Nota: Subida Masiva de Información. Nos permite identificar detalladamente la capacidad que nos permite para guardar información.

5.06. CONFIGURACIÓN DEL AMBIENTE MÍNIMA/IDEAL

Es fundamental tener en cuenta ciertos parámetros mínimos tales como sistema operativo, memoria RAM, tamaño de disco, capacitación del personal para un correcto funcionamiento de nuestro aplicativo web.

Tabla 37: Requisitos Mínimos

Parámetros	Descripción
Sistema Operativo	Windows 7 32 o 64 bits (o superior)
Memoria RAM	2 Gb
Disco Duro	100 Gb
Procesador	100 GHz (o superior)
Capacitación al personal	15 Horas
Pruebas	15 días

Nota: Requisitos Mínimos: En la presente tabla se detalla los requerimientos mínimos para poner en marcha el sistema.

Capítulo VI: ASPECTOS ADMINISTRATIVOS

6.01. RECURSOS

(Martin, 2013) Señala que:

Los recursos son la base que todo proyecto requiere para su desarrollo, tales que pudiéndolos usar de manera adecuada nos permitirá llegar a resultados excelentes y favorables del miso.

En el presente proyecto se ha utilizado cuatro recursos esenciales los cuales se describen enseguida:

RECURSOS HUMANOS

Es de vital importancia los recursos humanos para poder emprender un proyecto establecido con anterioridad.

Se debe establecer personal necesario preliminarmente preparado, para efectuar las labores que a los mismos se les determine, una vez que el personal se encuentre debidamente capacitados están listos para el manejo de información.

Tabla 38: Recursos Humanos

HUMANO	NOMBRE	ACTIVIDAD	RESPONSABILIDAD
Tutor	Ing. Carlos Nieto	Director del proyecto	Es el encargado de orientar a los tutorados en todo el proceso de desarrollo del sistema
Propietario	Sr. Fernando Chávez	Autoriza desarrollo del sistema.	Toma de decisiones

Estudiante	Gabriel Rivilla	Desarrollador	Ejecuta proceso

Nota: Recursos Humanos. Se detallan algunos involucrados para el desarrollo

RECURSOS TÉCNICOS

Estos recursos se considera el uso de recursos tecnológicos, métodos y mecanismos que nos permiten complementar el desarrollo del proyecto, tales como: Rational Rose donde se elaboraron diagramas de caso de uso, secuencia, actividades, componentes, clases además modelo lógico y físico de la base de datos.

Tabla 39: Recursos Tecnológicos

CANTIDAD	EQUIPO	DESCRIPCION
1	Laptop	Hp core i3 2.4Ghz, Ram de 8 GB Dvd writer, 500 GB disco duro
1	Impresora	Canon
1	Visual Studio 2013	Software para desarrollo del sistema.
1	Crystal Report	Generación de reportes
1	SQL Server 2012	Motor de Base de Datos.

Nota: Recursos Tecnológicos. Se detalla los recursos tecnológicos que se ha utilizado

6.02. PRESUPUESTO

En el presupuesto se encarga de efectuar el alcance de la delegación dentro de las zonas administrativas para el adecuado manejo de los recursos, el presupuesto se establece como origen a todos y cada uno de los costos y gastos que se presentan

del instante de que se pone en marcha el desarrollo del análisis cualquier tipo de proyecto.

Tabla 40: Presupuesto

SERVICIOS BÁSICOS				
	COSTO	TIEMPO (MESES)	TOTAL	20% DE DESCUENTO
Luz	$ 25	6	$ 150	$ 30
Agua	$ 10	6	$ 60	$ 12
Teléfono	$ 5	6	$ 30	$ 6
Internet	$ 25	6	$ 150	$ 30
Tv Cable	$ 25	6	$ 150	$ 30
		TOTAL		$ 108
RECURSOS TECNOLÓGICOS				
Laptop	$ 750	-	$ 750	-
Impresora	$ 70	-	$ 70	-
Teléfono Móvil	$ 60	-	$ 60	-
		TOTAL	$ 880	
RECURSOS MATERIALES				
Papel de impresora	$ 10	6	$ 60	-
Tinta de impresora	$ 5	6	$ 30	-
Transporte	$ 5	6	$ 30	-
		TOTAL	$ 120	
RECURSOS HUMANOS				

Desarrollador	$ 100	-	$ 100	-
Seminario de Titulación	$ 750	-	$ 750	-
Empastados	$ 50	-	$ 50	-
		TOTAL	$ 900	
	TOTAL		$ 2008	

Nota: Recursos: En la presente tabla se detalla los costos que se identificaron durante todo el proceso del proyecto

6.03. CRONOGRAMA

En el presente cronograma se determina el análisis de la perspectiva del tiempo empleado en las actividades que se realizan durante todo el proceso de desarrollo del proyecto. (Ver ANEXO A003).

Capítulo VII: CONCLUSIONES Y RECOMENDACIONES

7.01. CONCLUSIONES

Como resultado del presente proyecto podemos concluir que:

- El aplicativo web ha sido culminado satisfactoriamente gracias a las enseñanzas recibidas por parte de los docentes del Instituto Tecnológico Superior Cordillera, asimismo de la investigación y algunas de las sugerencias del usuario final.

- La forma manejable empleada para el control de acogida y reservación del hostal

  ISRAEL, se establece un ineficiente proceso de la información y no proporciona un adecuado manejo de organización de los procesos.

- El aplicativo web nos brinda un eficaz manejo de la organización de los procesos de inscripción de reservaciones, supervisión de alojamiento de las personas hospedadas mientras su estadía, comprendido hasta la partida del establecimiento del hostal ISRAEL.

- La implementación del diseño de la base de datos nos proporciona tener una integridad de los datos; además el diseño de interfaz nos concede una

observación anticipada de las pantallas y reportes que se va a emplear en el aplicativo web.

- El aplicativo desarrollado como planteamiento de solución se adapta cómodamente a las distintas maneras para el control de hospedaje y registro de reservaciones, esto gracias a los elementos establecidos con los que cuenta el aplicativo.

- Al implementar el sistema web en el hostal nos permitió optimizar y aligerar los procesos de organización y control de alojamiento en el hostal.

7.02. RECOMENDACIONES

- Se le sugiere a la persona delegada del manejo del aplicativo web realizar de manera persistente y habitual respaldar la base de datos para prevenir extravío de información.

- Todos los empleados que utilizan la aplicación deberán ser meticulosos con el uso de contraseñas para impedir que las mismas lleguen a manos de usuarios no autorizados, y de esta manera no tener vulnerabilidades de la información.

- Se sugiere a los usuarios del sistema verificar los manuales de funcionamiento frente a las diversas dudas que tienen sobre el manejo de los procesos.

BIBLIOGRAFÍA

(9 de Marzo de 2010). Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Caso_de_uso

Escobar, A. E. (3 de Abril de 2015). *WEB*. Obtenido de http://repo.uta.edu.ec/bitstream/123456789/10388/1/Tesis_t991si.pdf

Luis, J. (24 de Septiembre de 2014). *Parvulos.Net*. Obtenido de http://joseluisgarciab.blogspot.com/2014/09/programacion-en-3-capas.html

Martin, L. (28 de Febrero de 2013). *BusinessInFact*. Obtenido de http://www.todostartups.com/bloggers/recursos-necesarios-para-poner-en-marcha-un-proyecto-emprendedor-por-businessinfact

Tuza, J. F. (2014). Quito.

Will.i. (28 de Junio de 2010). *Yo lo puedo hacer*. Obtenido de http://yolopuedohacer.blogspot.com/2010/06/estandares-de-programacion-manana-hoy-y.html

ANEXOS

ANEXO A001

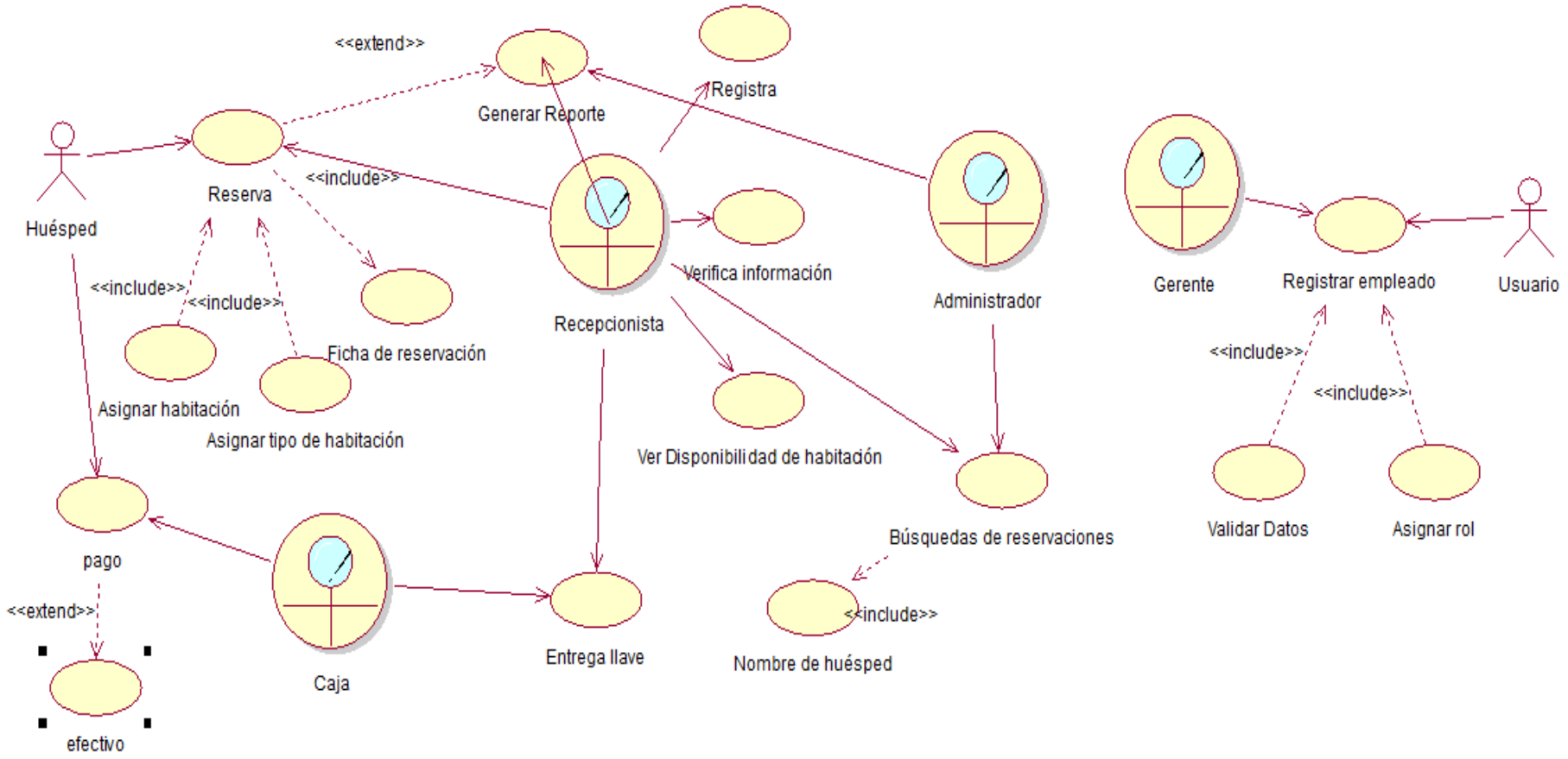


Figura 30: Diagrama de Caso de Uso General
En este diagrama se describe todas las tareas que se va a realizar en el proyecto conjuntamente con todos los procesos

ANEXO A002

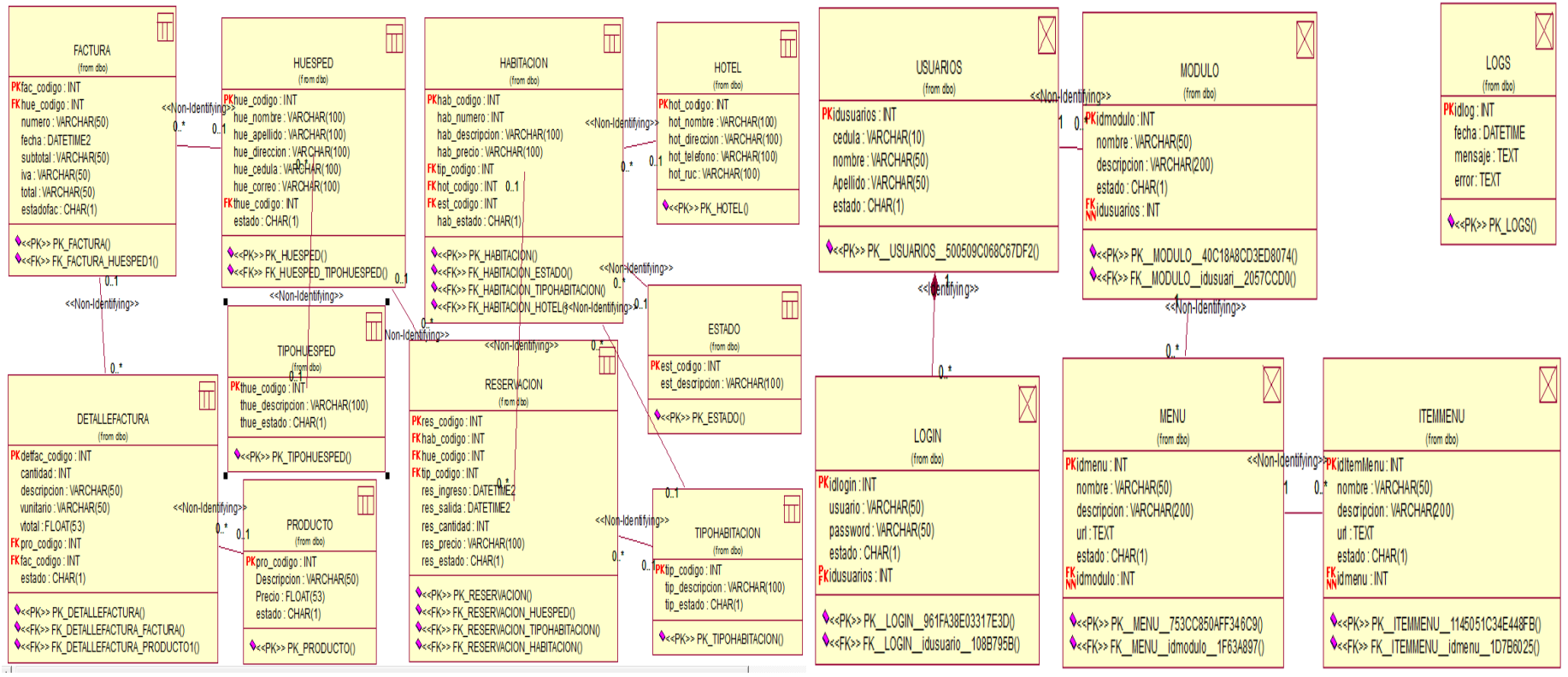


Figura 31: Diagrama de Clases (Modelo físico).
El presente diagrama especifica cómo se constituye el sistema de forma física conjuntamente con sus atributos.

ANEXO A003

DIAGRAMA DE GANTT

	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1		Inicio	0 días	lun 03/08/15	lun 03/08/15	
2		Levantamiento de requerimientos	15 días	lun 03/08/15	lun 24/08/15	1
3		Analisis de metodologia	5 días	mar 25/08/1	lun 31/08/15	2
4		selección de lenguaje de programacion	5 días	mar 01/09/15	lun 07/09/15	3
5		Casos de Uso	8 días	mar 08/09/15	jue 17/09/15	4
6		Diseño de la Base de Datos	12 días	vie 18/09/15	lun 05/10/15	5
7		Codificacion del Sistema	100 días	mar 06/10/1	mié 02/03/1	6
8		Pruebas del sistema	5 días	jue 03/03/16	mié 09/03/1	7
9		Implementacion	5 días	jue 10/03/16	mié 16/03/1	8
10		Elaboracion del Manual de Usuarios	8 días	jue 17/03/16	lun 28/03/16	9
11		Elaboracion del Manual de Instalacion	8 días	mar 29/03/16	vie 08/04/16	10
12		Fin	0 días	vie 08/04/16	vie 08/04/16	11

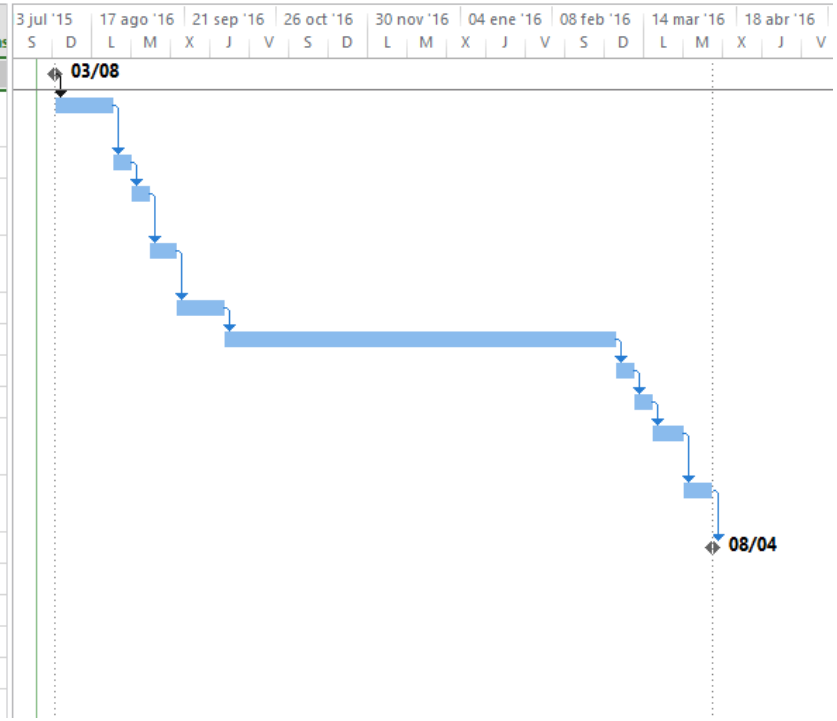


Figura 32: Cronograma
En el presente cronograma se detalla las actividades que se realizan Durante todo el proceso de desarrollo del proyecto desde la planificación hasta el desarrollo.

MANUAL DE INSTALACIÓN

ÍNDICE DE FIGURAS

El presente manual, tiene como propósito establecer una orientación para la instalación de los programas necesarios para poner en marcha nuestro sistema Web para el hostal ISRAEL.

Instalación SQL Server 2012

Se debe descargar la versión del SQL Server 2012 Full de la página oficial del siguiente link:

https://www.microsoft.com/es-es/download/details.aspx?id=29062

Se descomprime el archivo descargado para empezar con la instalación del SQL Server 2012

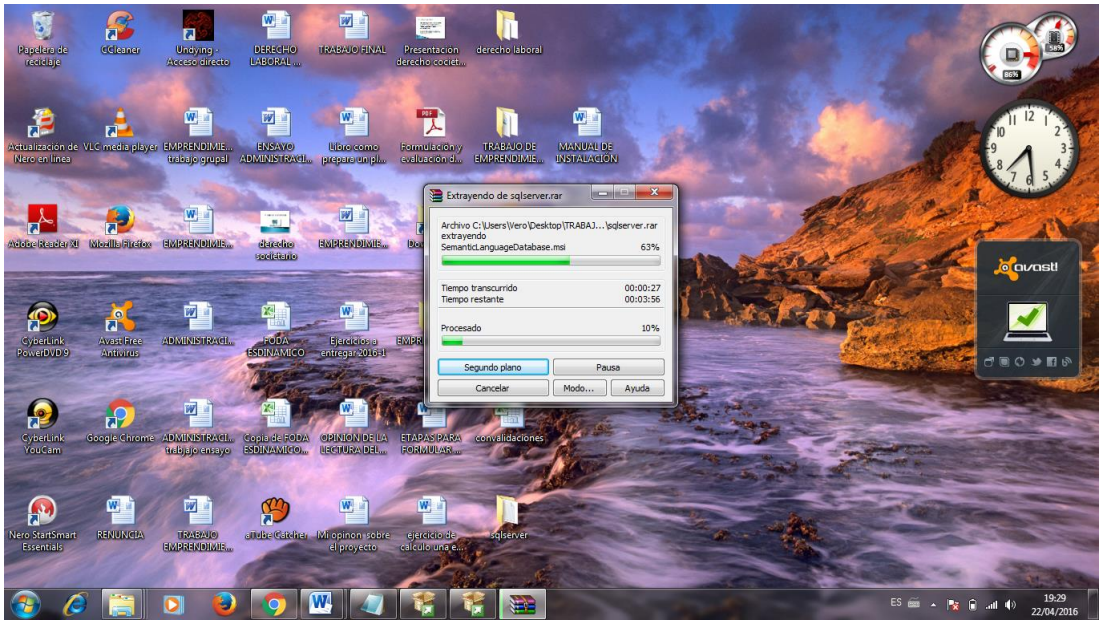


Figura 33: Instalador SQL Server 2012

Se debe dar click derecho en el setup y ejecutar como administrador

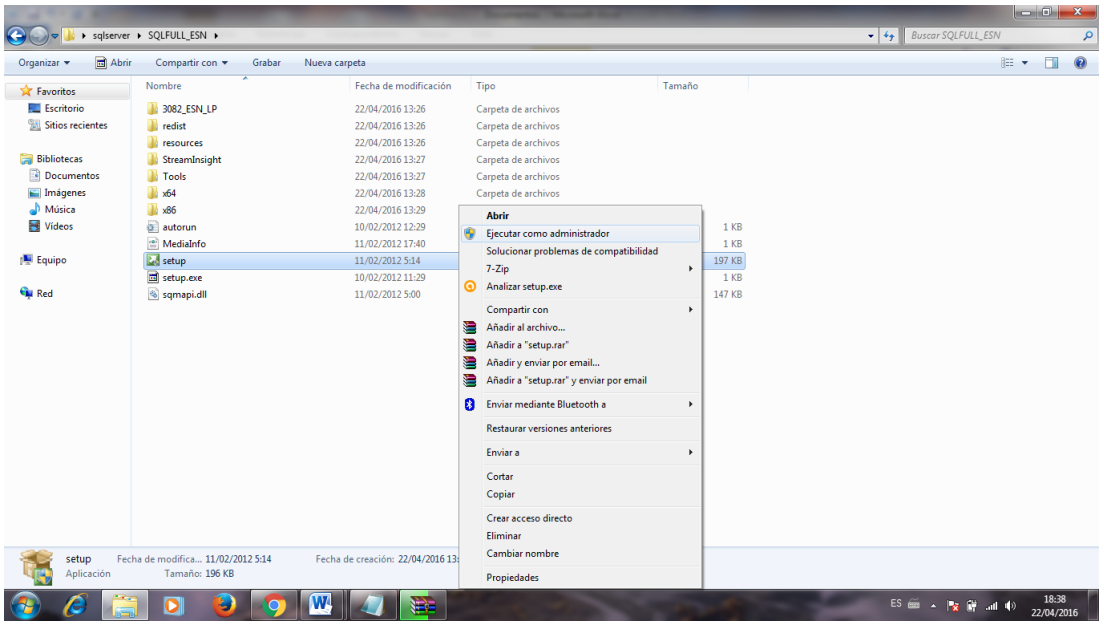


Figura 34: Ejecutamos el Instalador

Si al momento de instalar SQL Server 2012, aparece un error como el siguiente:

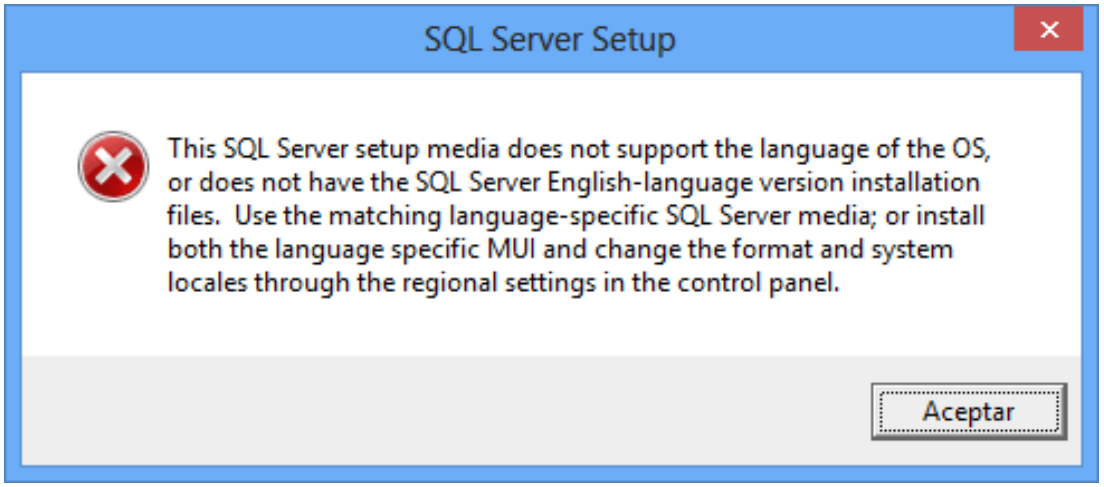


Figura 35: Error De Instalación

Se debe realizar los siguientes pasos:

Ir a panel de control, Configuración regional y de idioma, Formatos: Seleccionar Español(España), Aplicar Cambios.

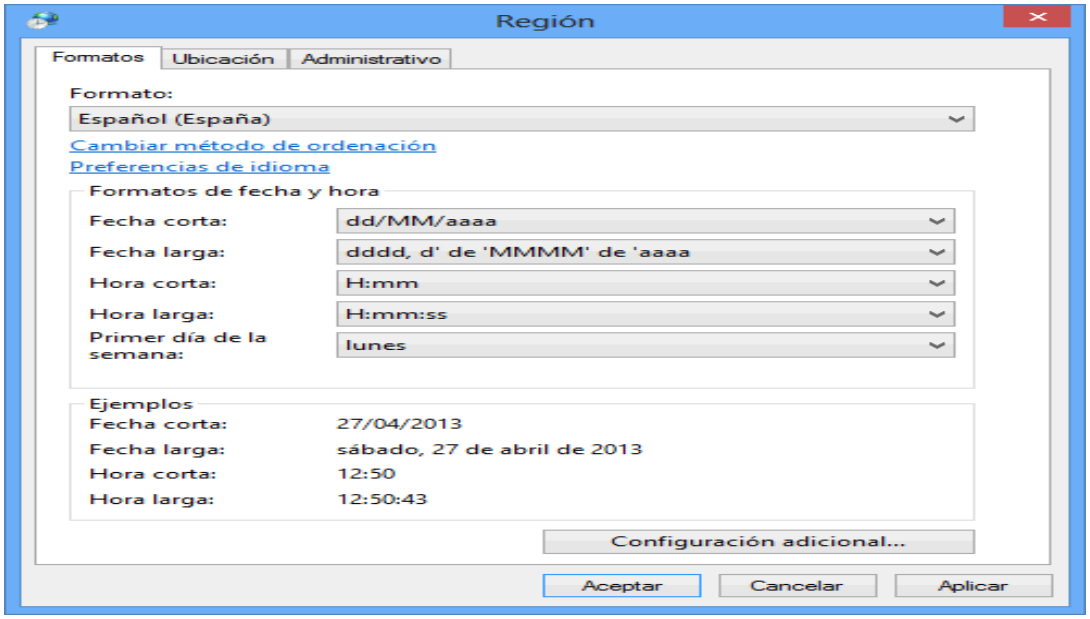


Figura 36: Formatos

Ahora si procedemos a instalar en el Centro de instalación de SQL Server damos clic en instalación y en Nueva instalación independiente de SQL Server o agregar características a una instalación existente.

Figura 37: Centro de instalación de SQL Server

Esto permite que la instalación se inicie y se nos muestra las Reglas auxiliares del programa de instalación, la cual se encarga de identificar posibles problemas que pueden surgir al momento de instalar. Una vez finalizado damos click en Aceptar.

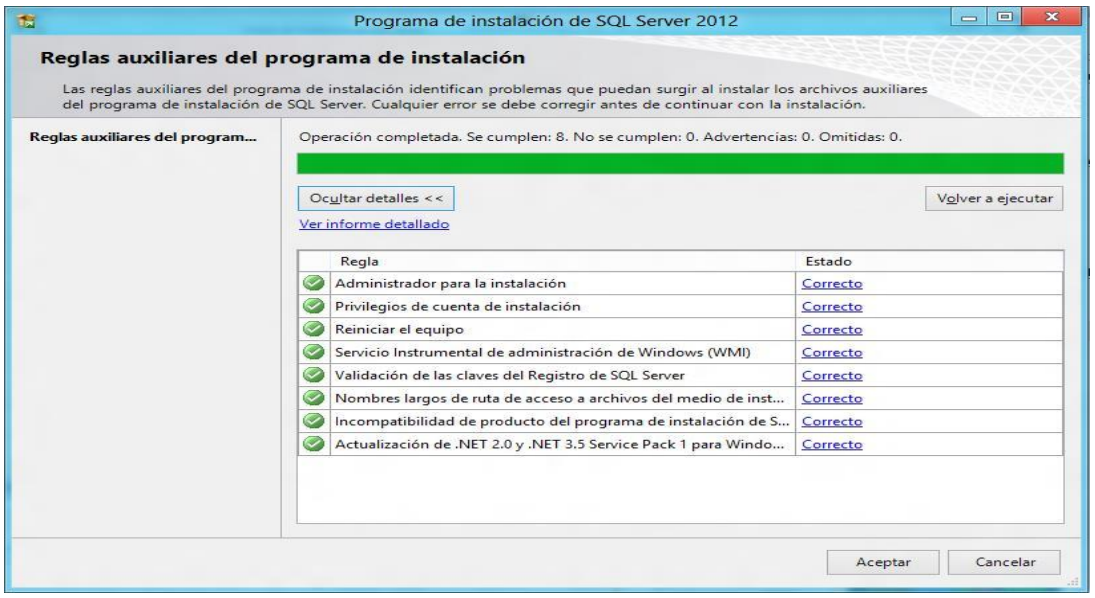


Figura 38: Reglas auxiliares del programa de instalación

Esperamos y nos pide la clave del producto la cual la vamos encontrar en el archivo descomprimido y procedemos a escribir la clave del producto y damos click en siguiente.

Figura 39: Clave del Producto

Debemos aceptar los Términos de Licencia del Software de Microsoft y damos click en siguiente.

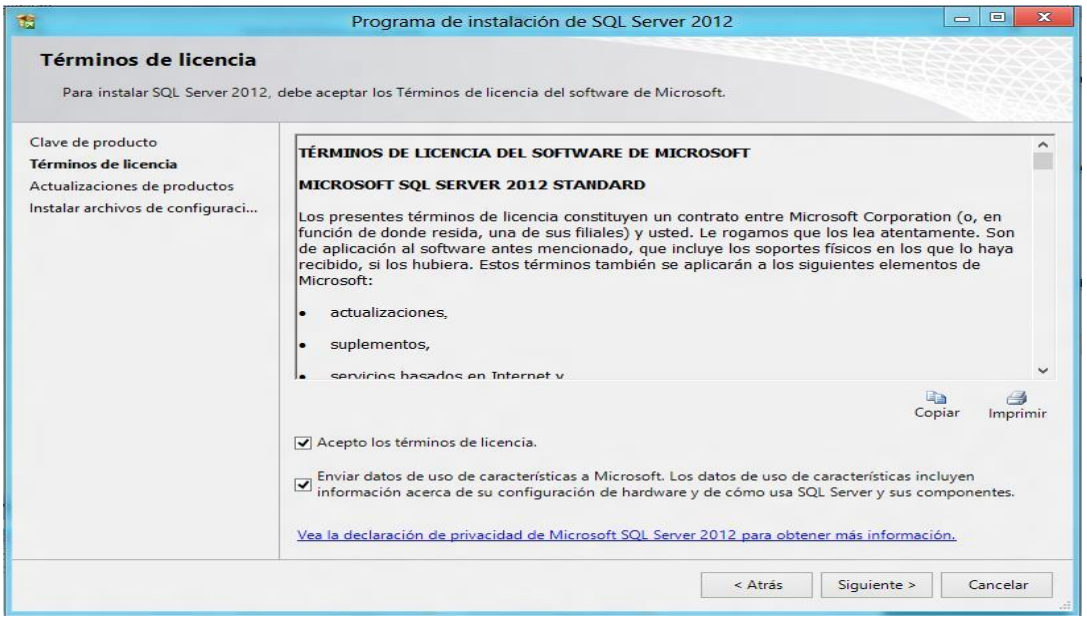


Figura 40: Términos de Licencia

Una vez seleccionado las casillas correspondientes nos llevará a la ventana de Actualizaciones de Productos, la cual realiza una comprobación de actualizaciones que podamos instalar para mejorar el rendimiento del SQL Server.

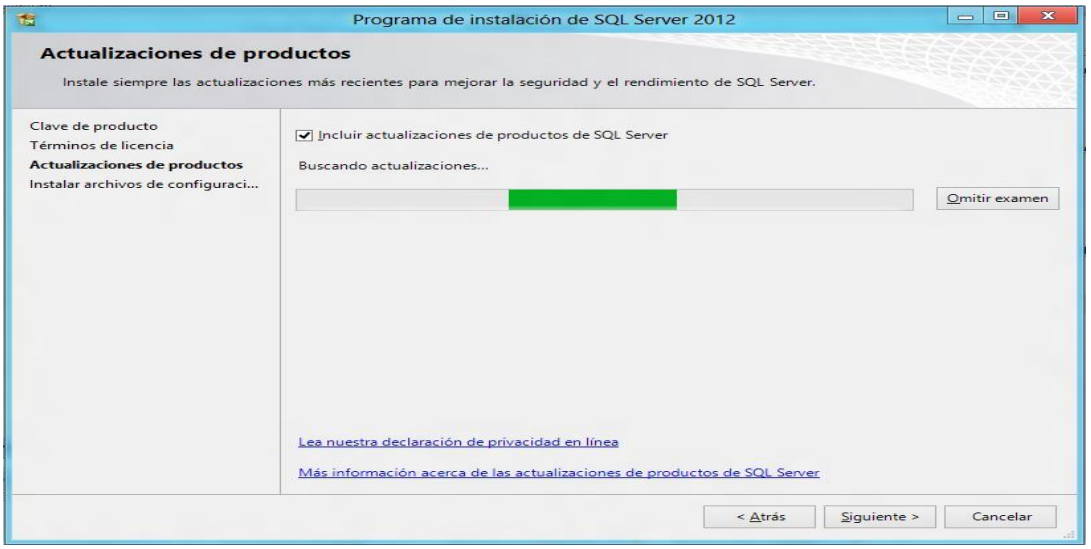


Figura 41: Actualizaciones de Productos.

En el caso de no existir ninguna actualización disponible nos mostrará la siguiente ventana y le damos clic en Siguiente.

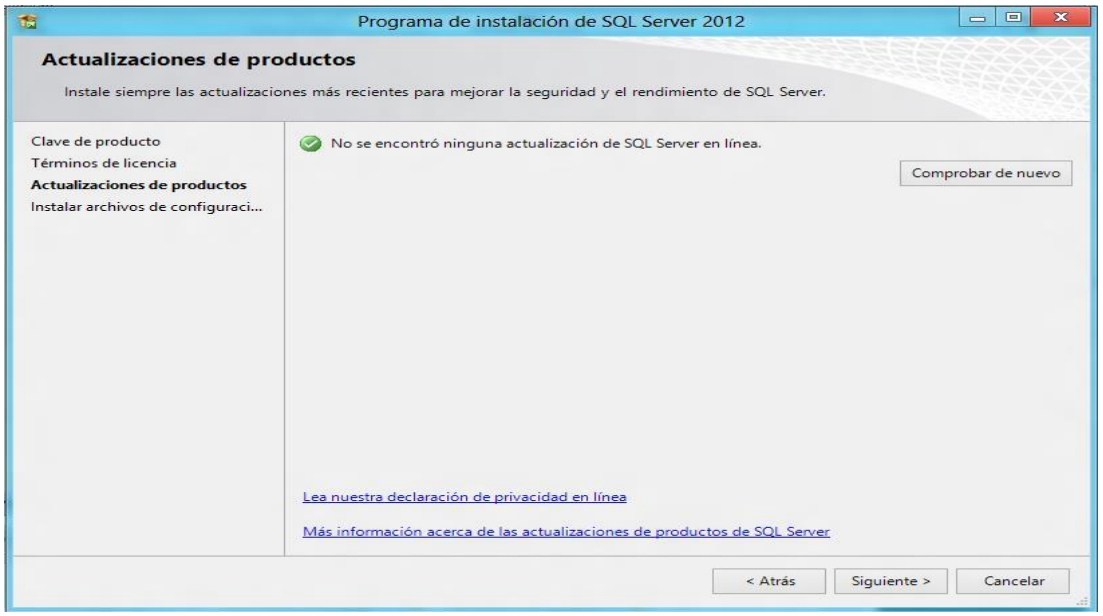


Figura 42: Mensaje del registro de actualizaciones

Seguidamente se abrirá la ventana siguiente en donde procede a instalar actualizaciones en caso que las haya, así como también los archivos del programa.

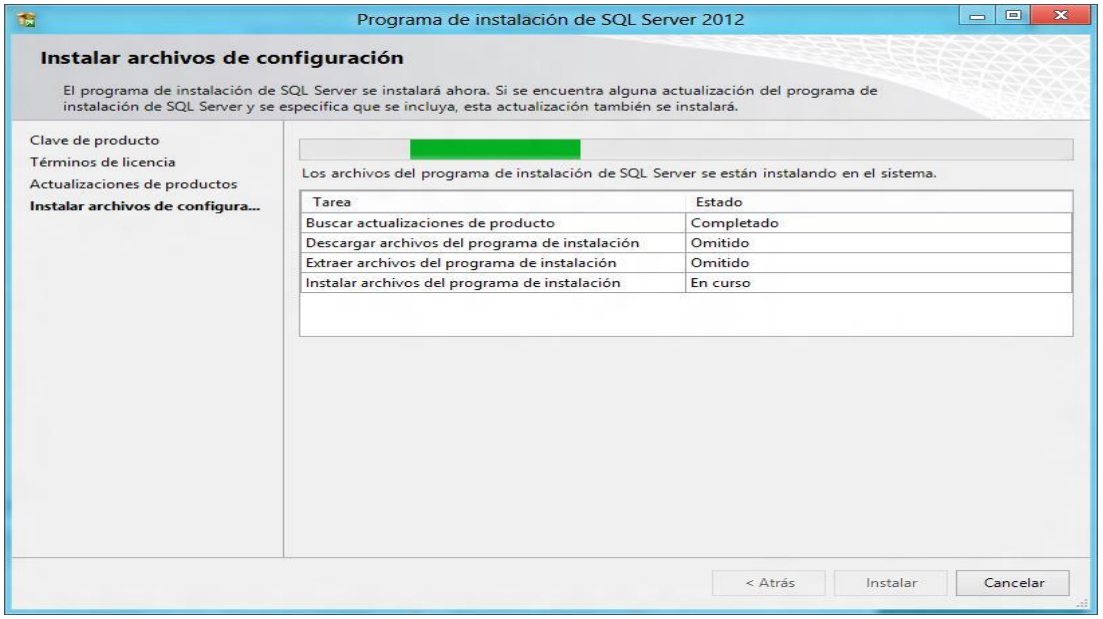


Figura 43: Instalación de Archivos de Configuración

A continuación nuevamente nos va aparecer la ventana de Reglas auxiliares de programa de instalación, pero en este caso verifica las reglas necesarias para la instalación del SQL Server. Y damos clic en Siguiente.

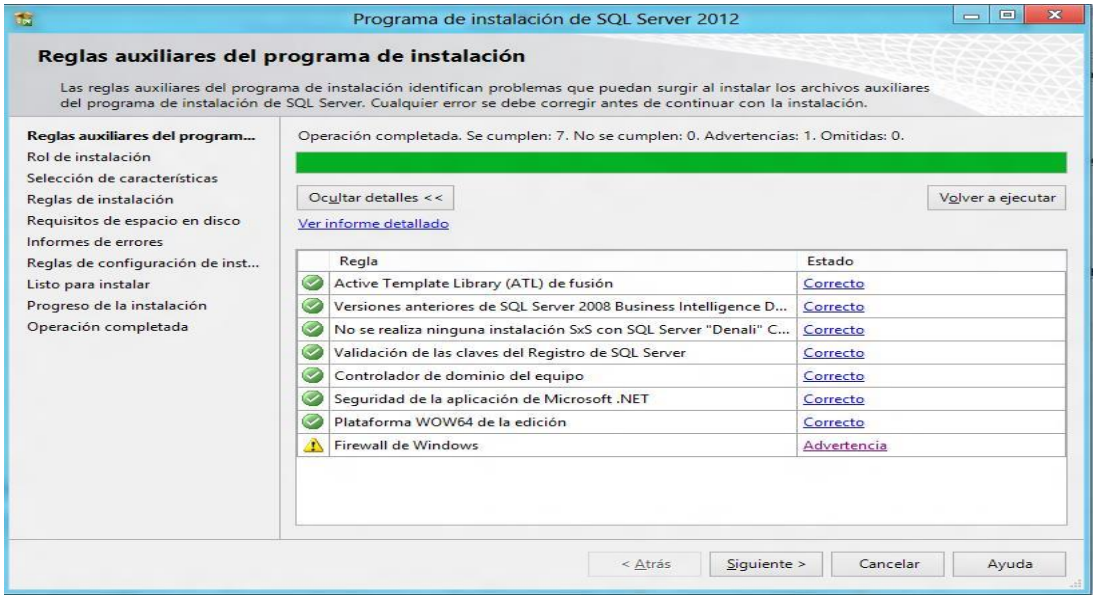


Figura 44: Reglas auxiliares del programa de instalación.

En la presente ventana seleccionamos la opción de Instalación de Características de SQL Server y damos clic en Siguiente.

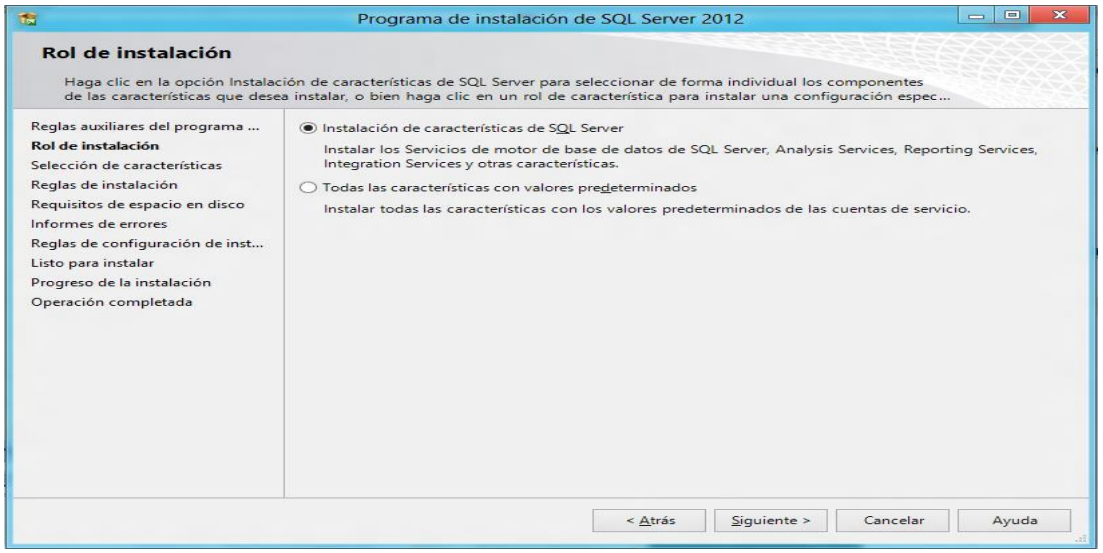


Figura 45: Rol de Instalación.

En esta ventana seleccionamos todas las características que vamos a instalar y damos clic en Siguiente

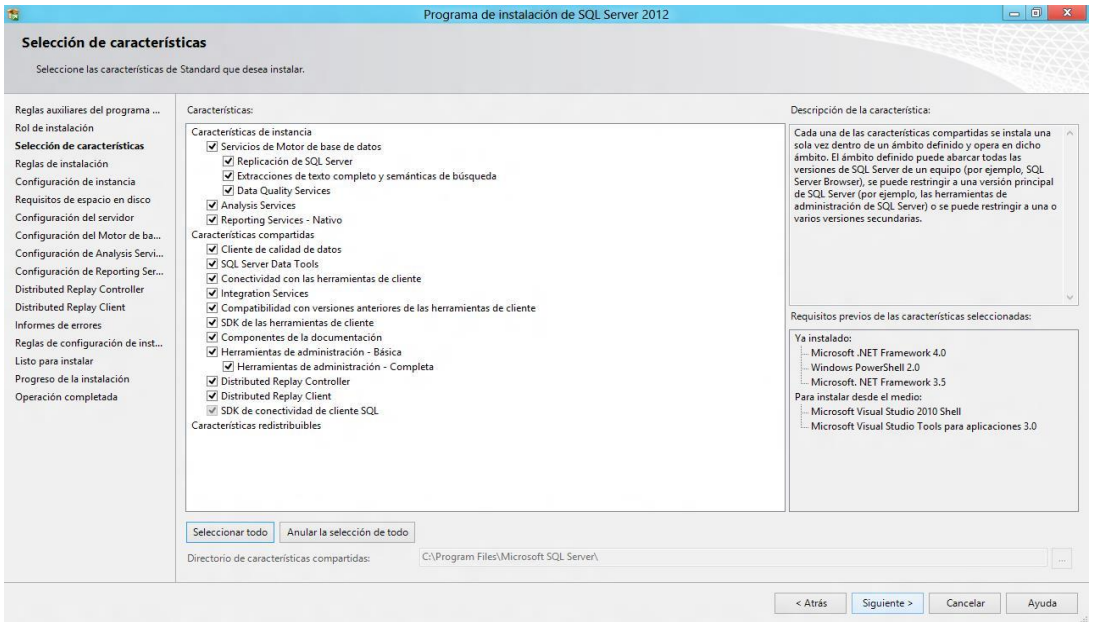


Figura 46: Selección de Características.

Una vez seleccionado las características a instalar procedemos a verificar las reglas de instalación y damos clic en Siguiente.

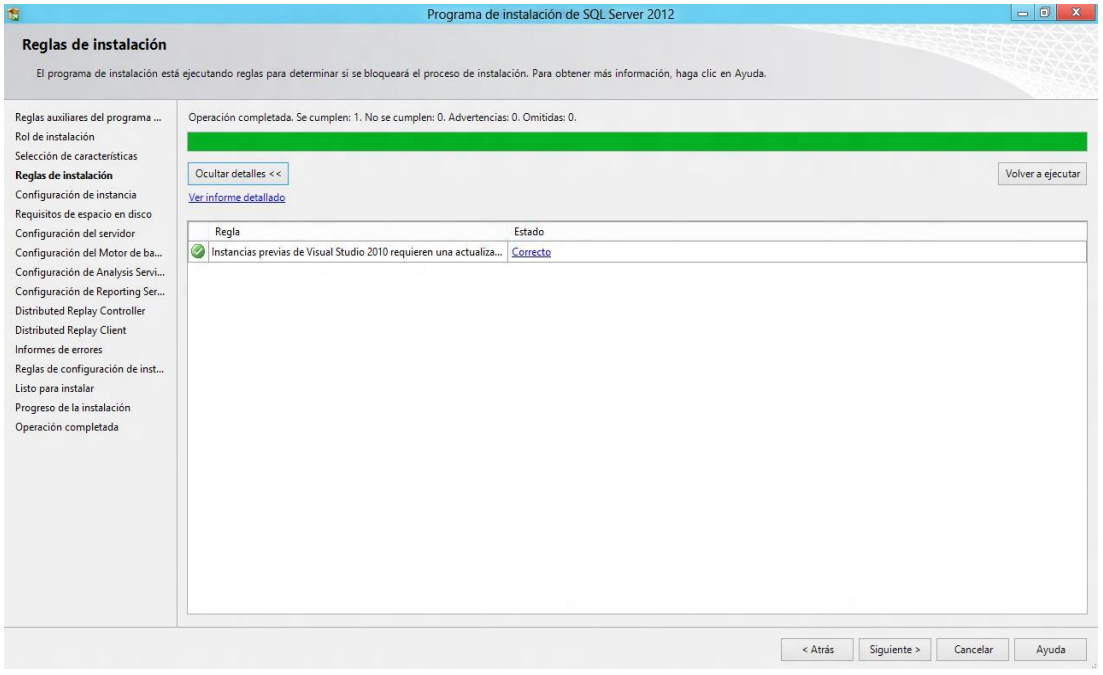


Figura 47: Reglas de Instalación

En la presente ventana crearemos una nueva instancia y le dejamos como esta predeterminado y damos clic en Siguiente.

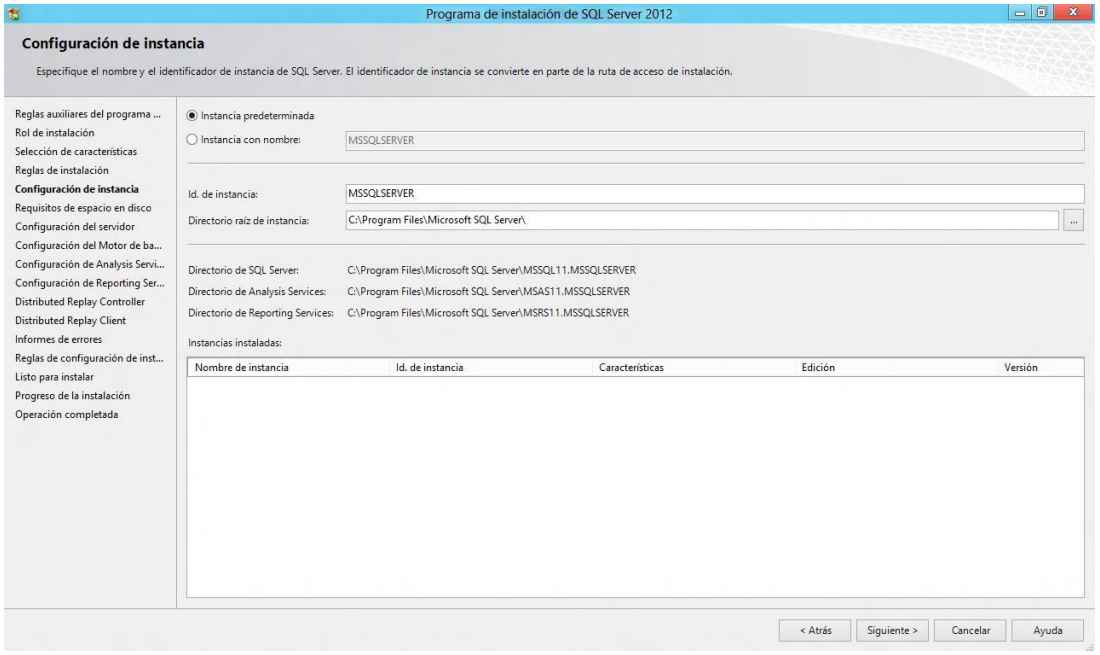


Figura 48: Configuración de Instancia.

En esta ventana se puede visualizar los datos de espacio requerido y el espacio disponible para la instalación, verificamos y damos clic en Siguiente.

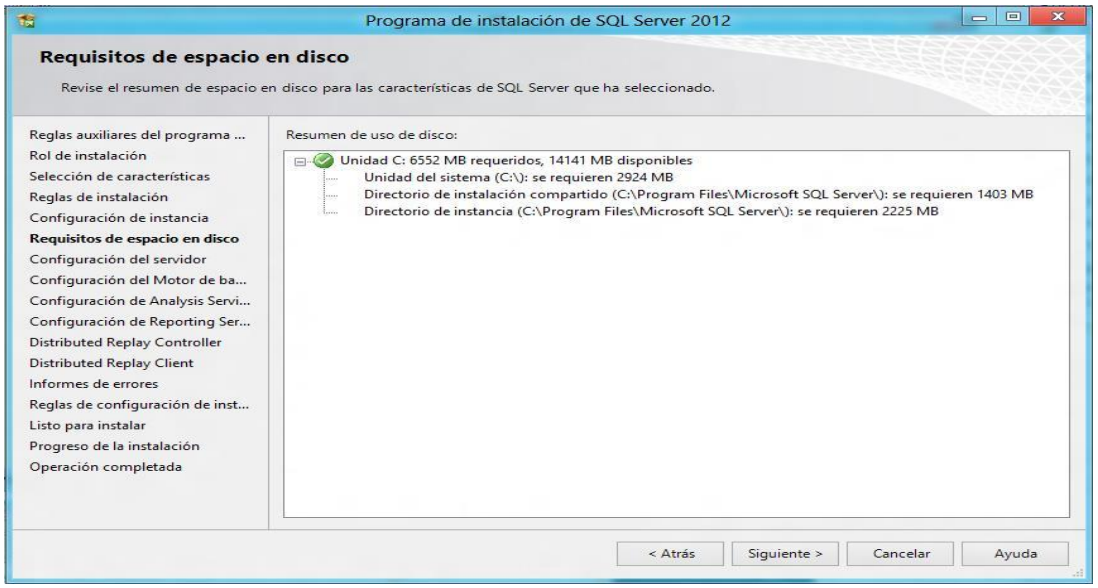


Figura 49: Requisitos de espacio en Disco.

A continuación procedemos a especificar las cuentas de servicio y la configuración de intercalación una vez hecho le damos clic en Siguiente.

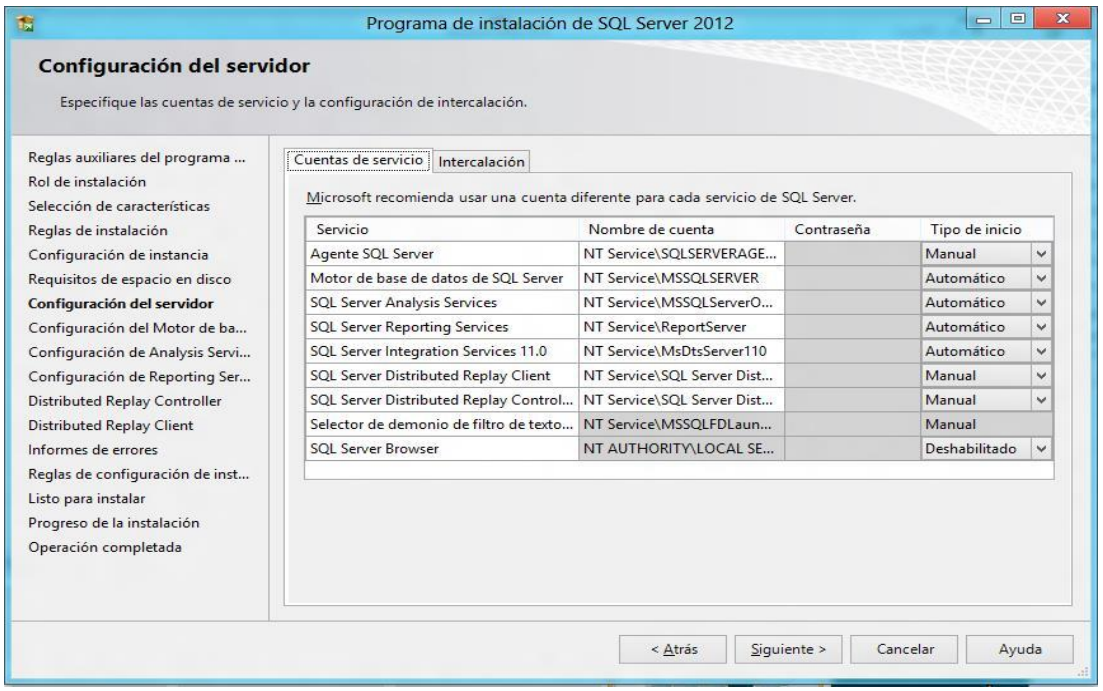


Figura 50: Configuración del Servidor

En esta ventana especificamos el modo de seguridad de autentificación, carpeta de datos y os administradores del motor de base de datos y damos clic en Siguiente.

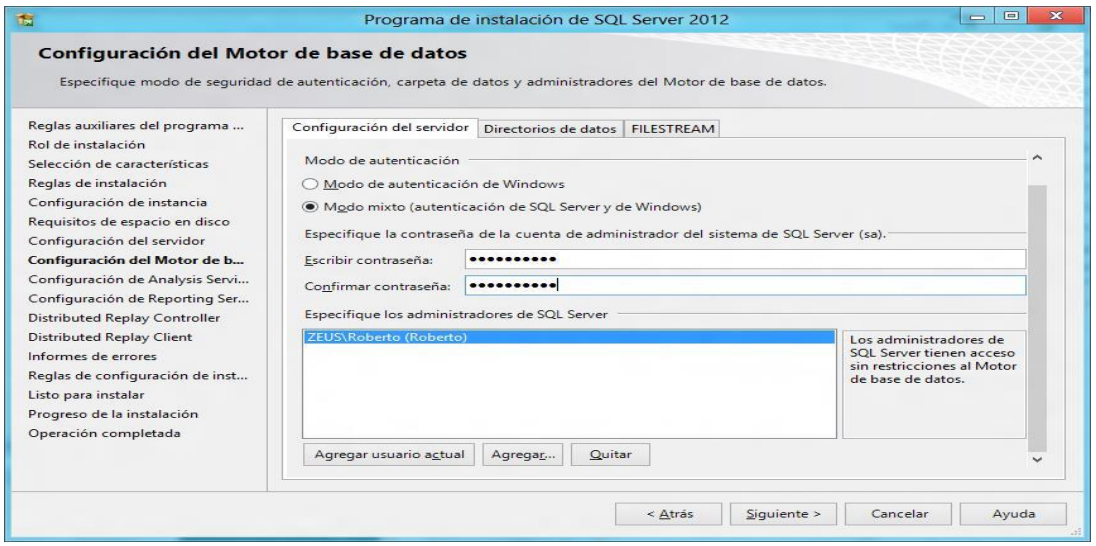


Figura 51: Configuración del Motor de Base de Datos.

En la presente ventana dejamos los valores que están predeterminados y damos clic en Siguiente.

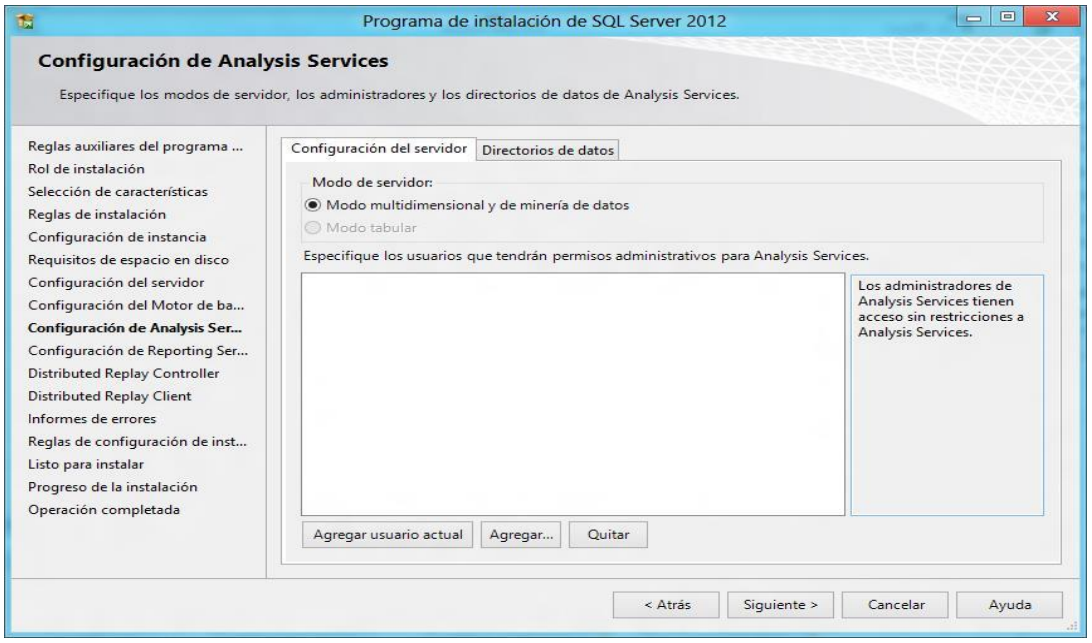


Figura 52: Configuración de Análisis de Servicios

A continuación damos clic en la opción Instalar y Configurar y damos clic en Siguiente.

Figura 53: Configuración de Reporting Services.

Seguidamente asignamos a los usuarios que tendrán permiso de acceso y damos clic en Siguiente.

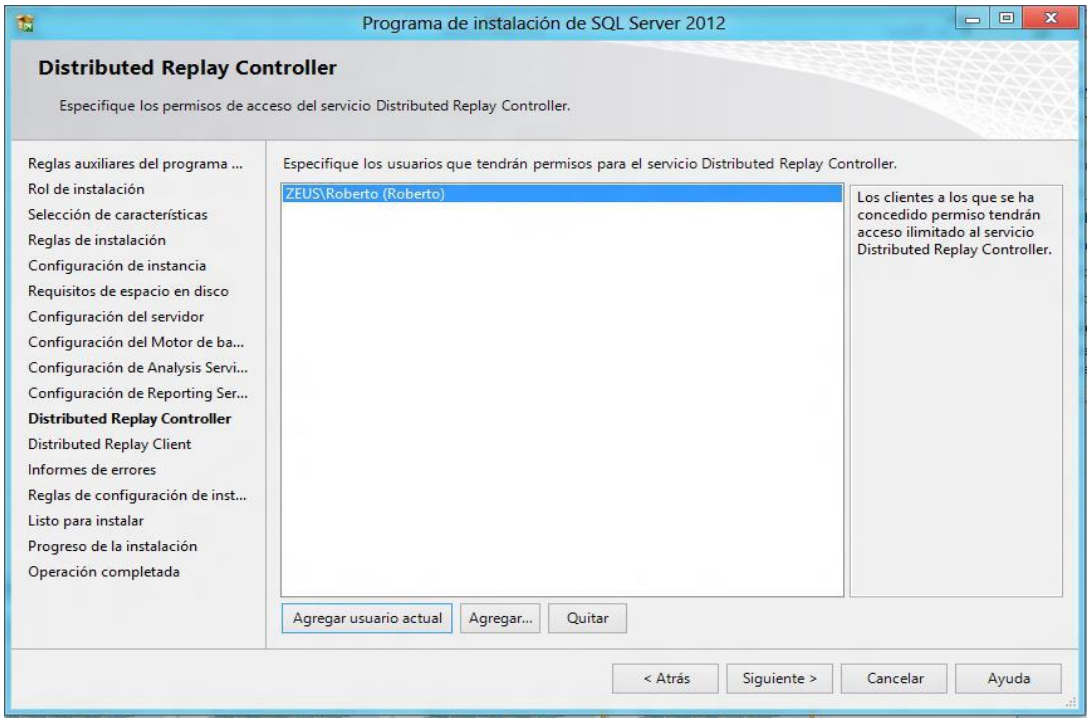


Figura 54: Especificación de usuarios quienes tendrán permisos.

En esta ventana procedemos a especificar el controlador adecuado y los directorios de datos y damos clic en Siguiente.

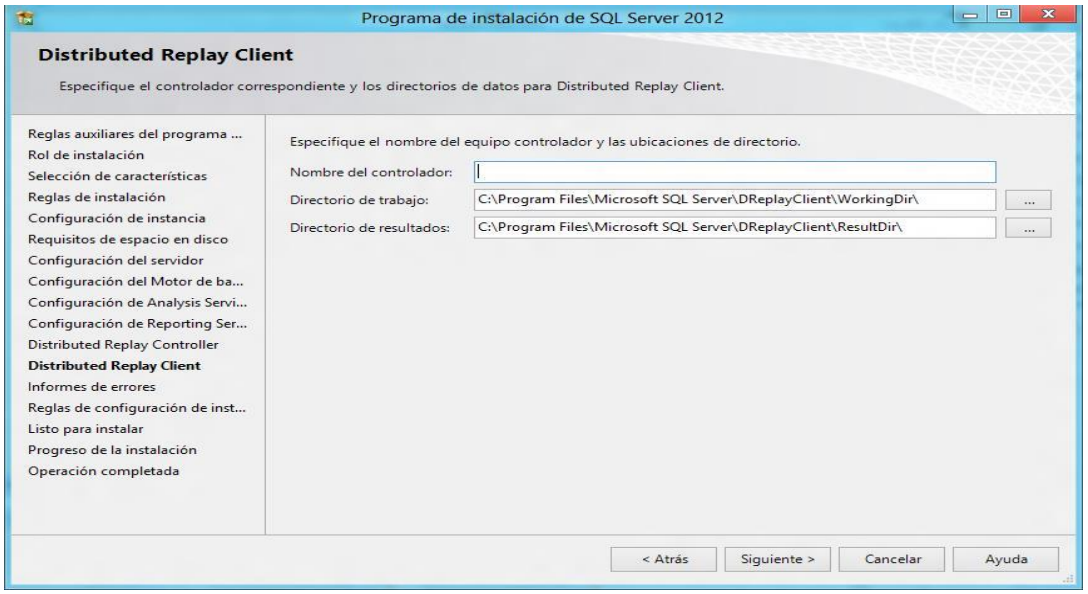


Figura 55: Especificación del Controlador.

Seguidamente llegamos a los informes de errores lo cual la opción es opcional y damos clic en Siguiente.

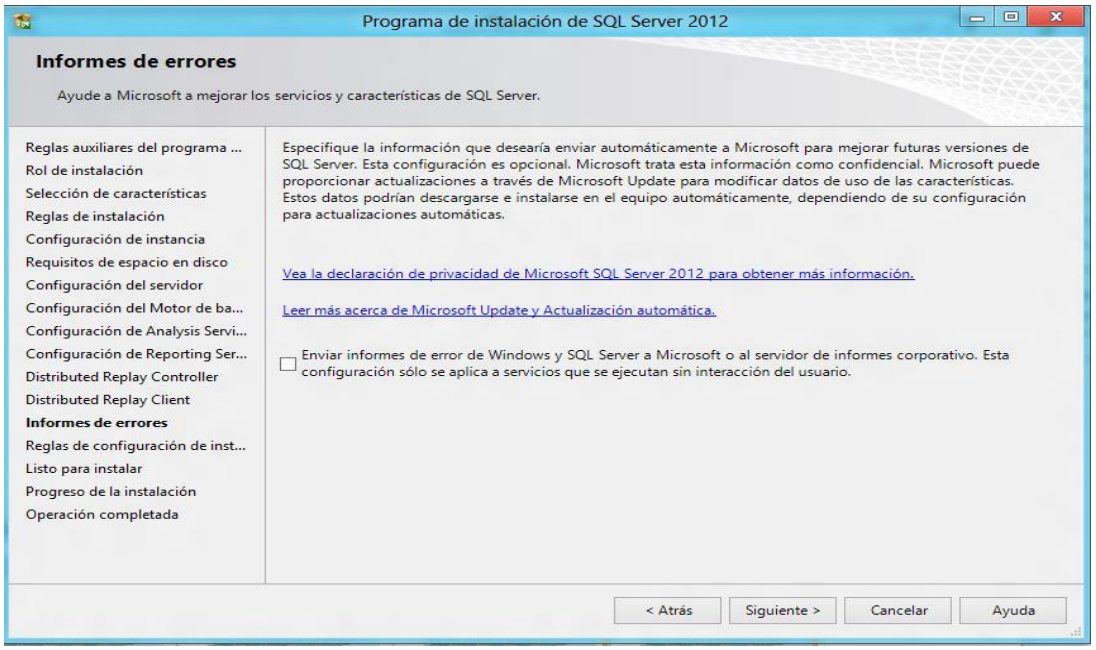


Figura 56: Informe de Errores.

En la presente ventana verificamos que no existan errores y damos clic en Siguiente.

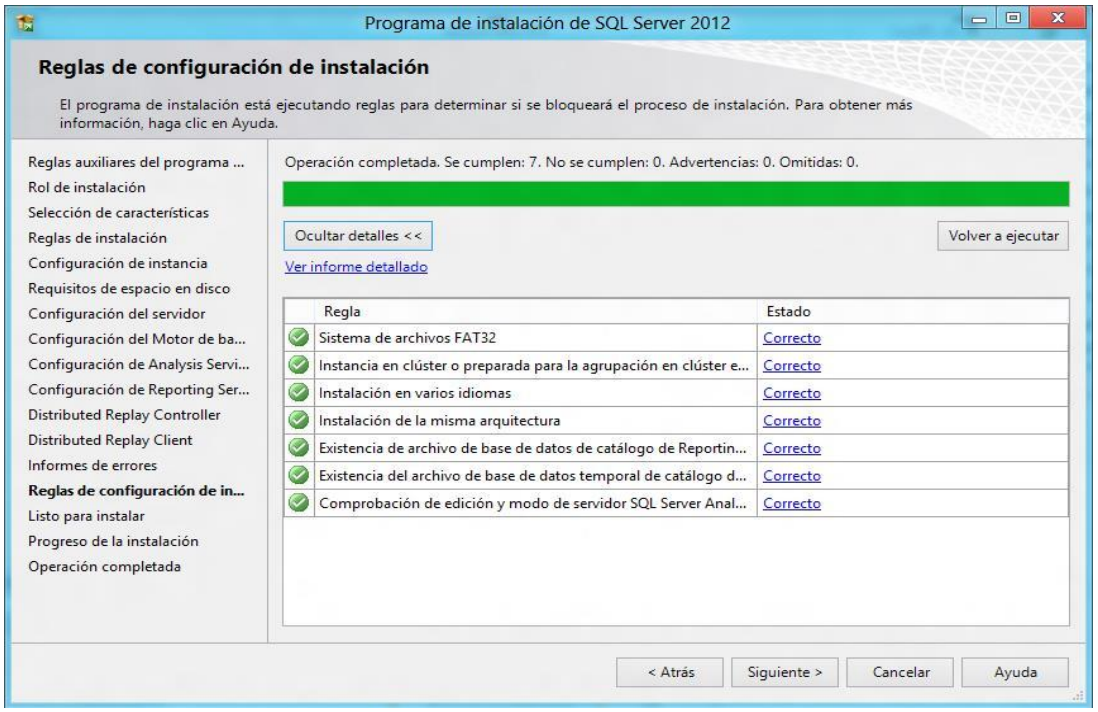


Figura 57: Reglas de configuración de Instalación.

En esta ocasión verificaremos todas las características que vamos a instalar y procedemos a dar clic en Instalar.

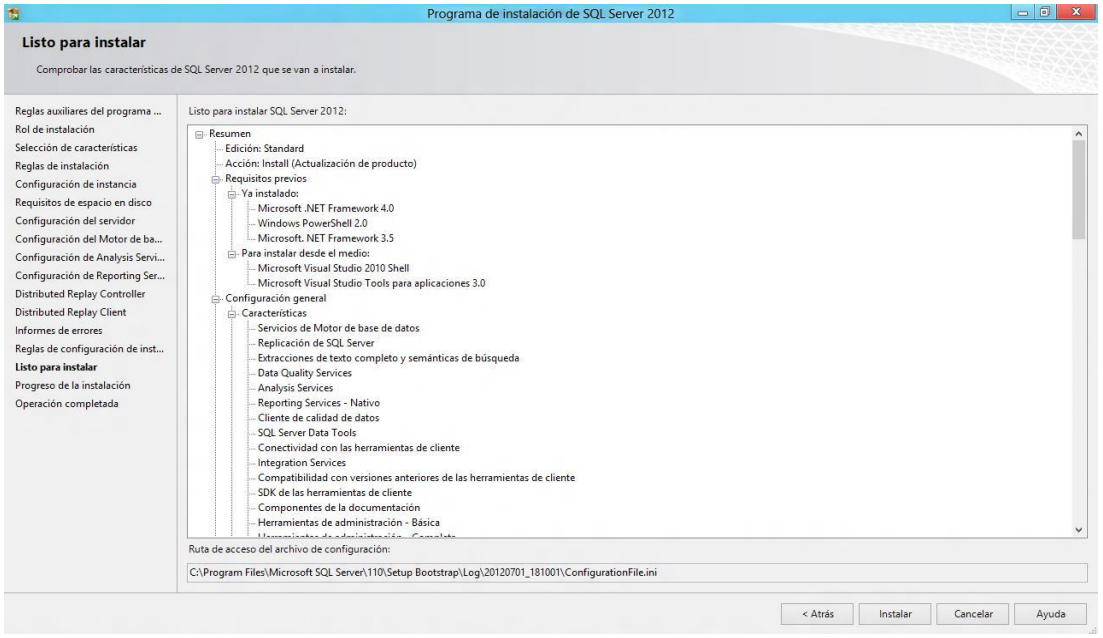


Figura 58: Listo para Instalar.

En la presente ventana vemos el progreso de instalación.

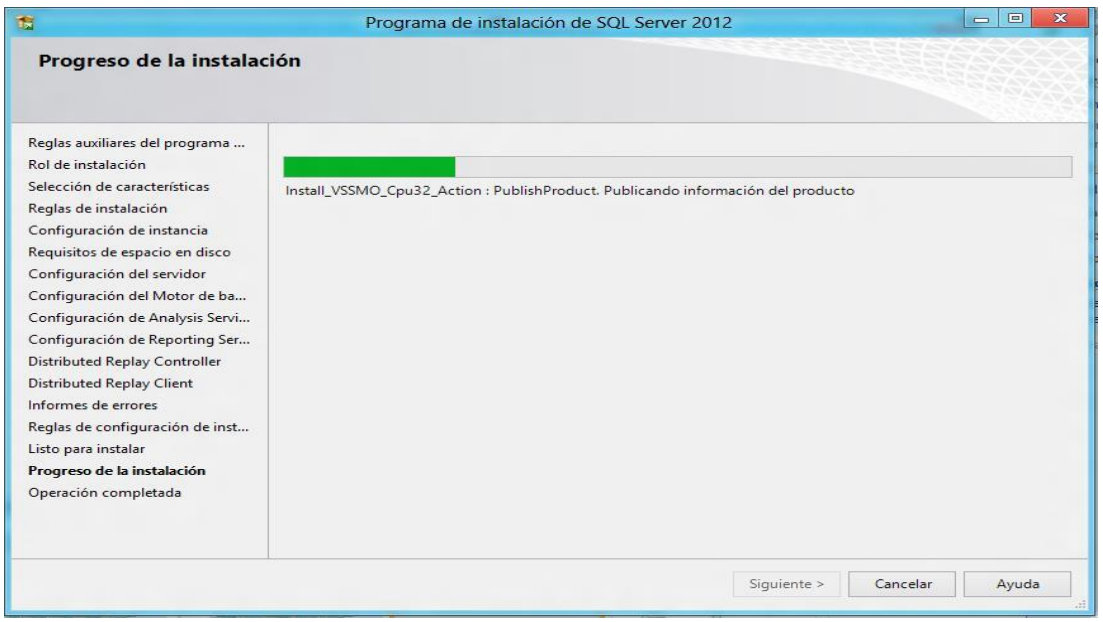


Figura 59: Progreso de Instalación.

Una vez finalizada la Instalación nos aparece una ventana con el resumen de las características instaladas con su respectivo estado y procedemos a dar clic en cerrar y con este paso terminamos la instalación del SQL Server 2012, el cual ya podemos trabajar.

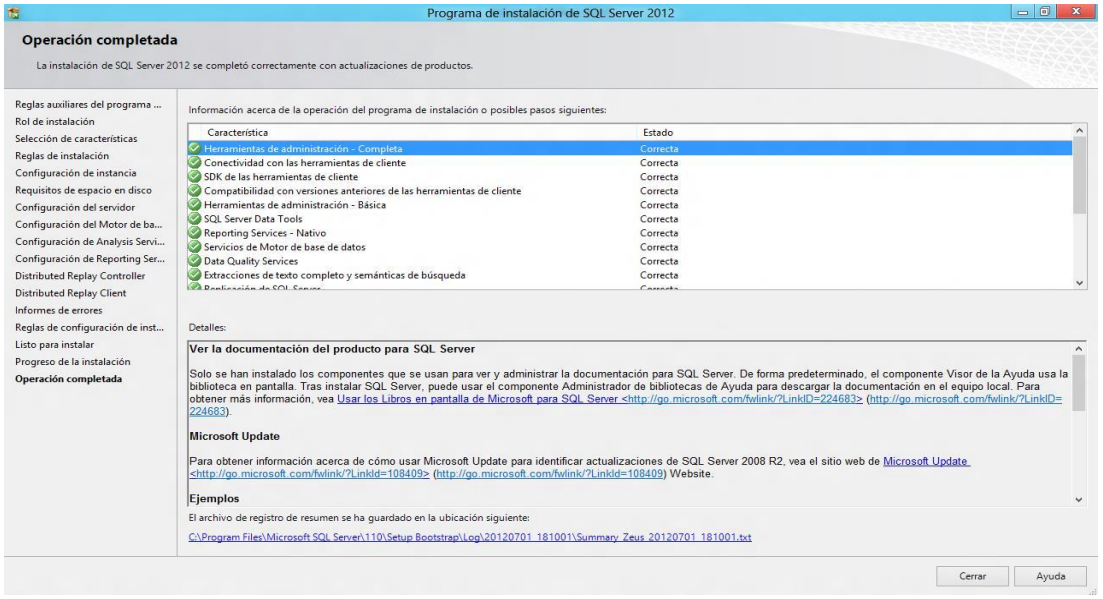


Figura 60: Operación Completada.

Instalación Visual Studio 2013

Se debe descargar la versión del Visual Studio 2013 de la página oficial del siguiente link:

https://www.microsoft.com/es-es/download/details.aspx?id=44915

Se descomprime el archivo descargado para empezar con la instalación del Visual Studio 2013.

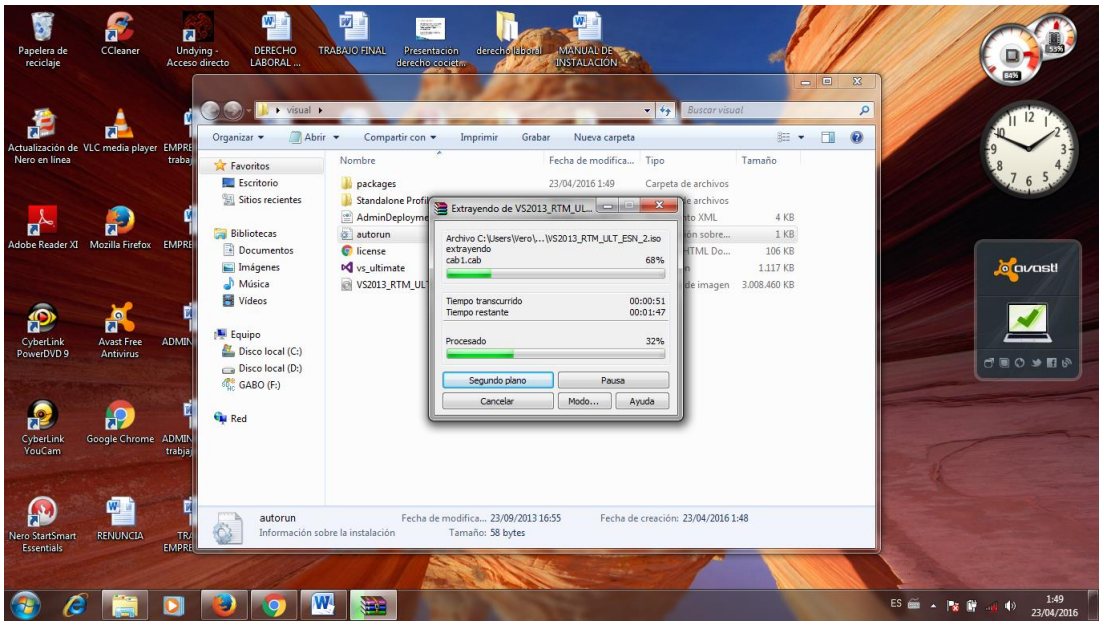


Figura 61: Descomprimir.

Se debe dar clic derecho en el icono vs_ultimate y ejecutamos como administrador.

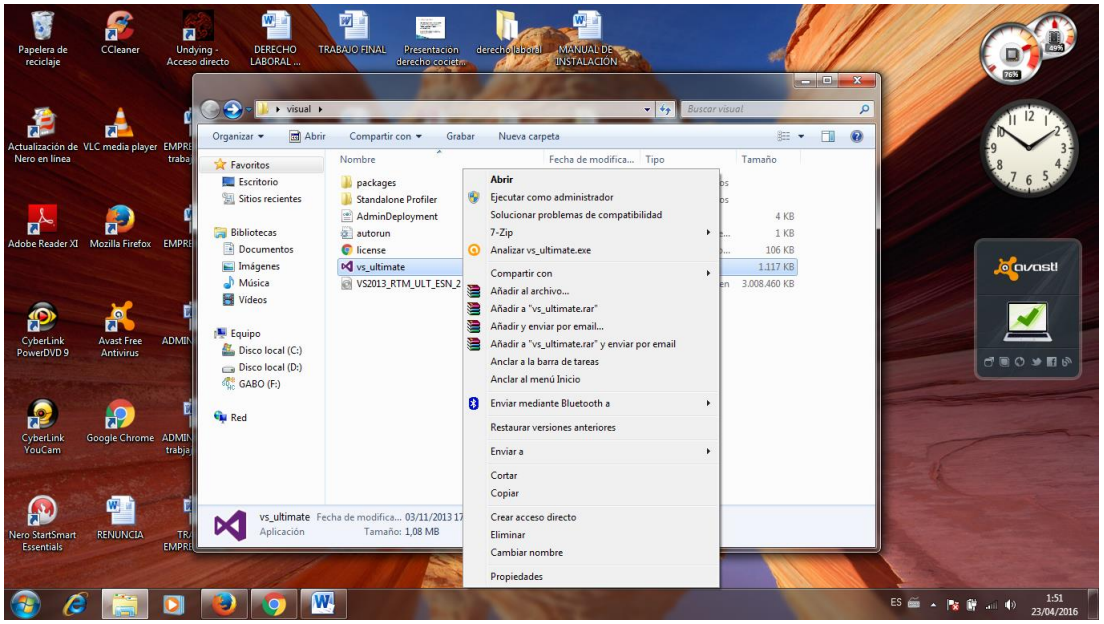


Figura 62: Instalador del Visual Studio 2013

Procedemos a instalar, omitimos el mensaje visualizado y damos clic en continuar.

Figura 63: Pantalla de Inicio

En la presente ventana damos clic en la opción: Acepto términos de licencia y la declaración de privacidad y damos clic en Siguiente.

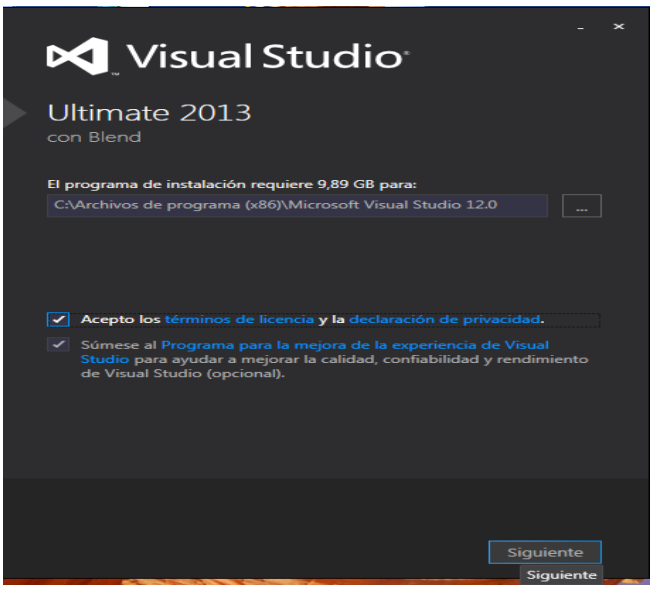


Figura 64: Términos y Condiciones del Servicio

En esta figura seleccionamos las características a instalar y damos clic en INSTALAR.

Figura 65: Características.

A continuación se nos abre la ventada de progreso de instalación.

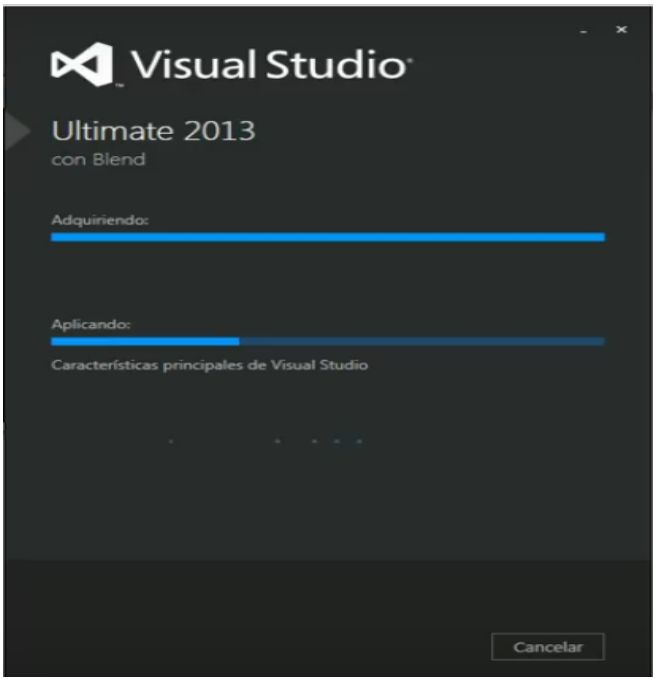


Figura 66: Progreso de Instalación

Esperamos cierto tiempo que se termine de instalar, una vez instalado aparece esta ventana y damos clic en Reiniciar ahora.

Figura 67: Proceso de reinicio.

Una vez reiniciada nuestra computadora procedemos abrir el Visual Studio 2013 y damos clic en Iniciar Sesión.

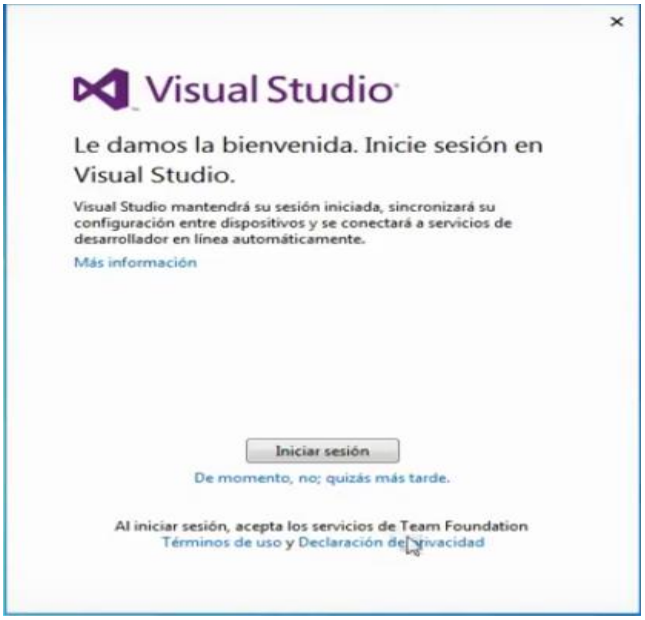


Figura 68: Inicio de Sesión

A continuación le personalizamos nuestro Visual Studio y escogemos las características que deseamos y en la configuración de desarrollo seleccionamos: General y damos clic en Iniciar Visual Studio.

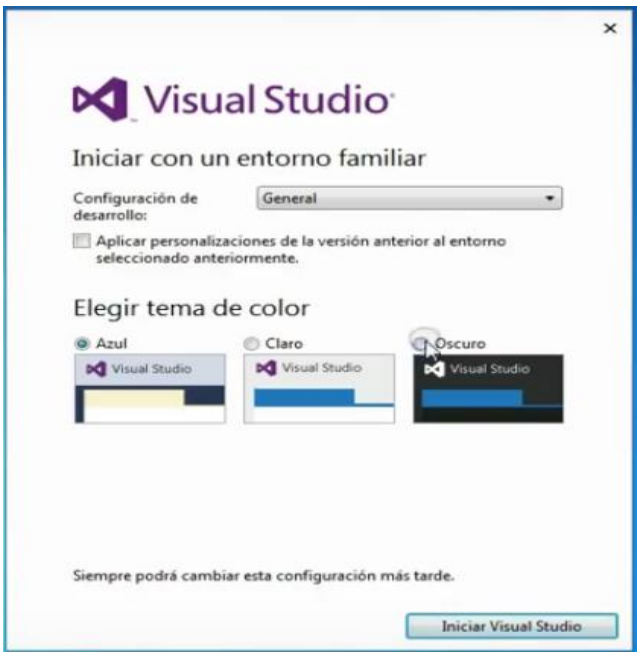


Figura 69: Características de la interfaz del Visual Studio.

Seguidamente nos aparece la siguiente ventana y esperamos

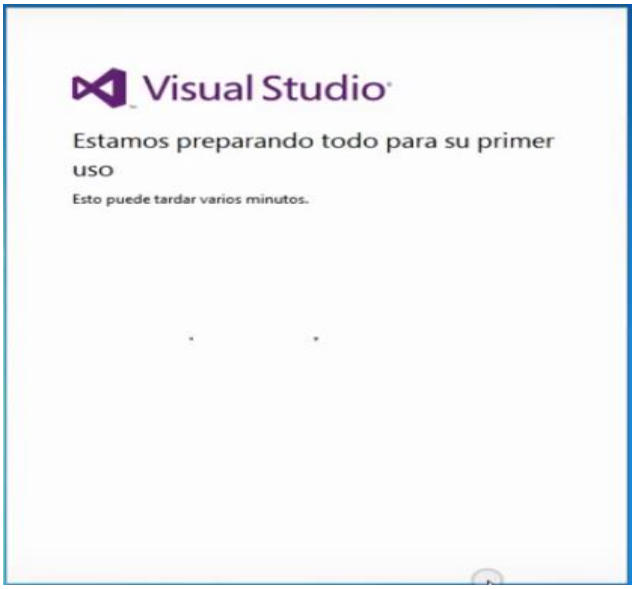


Figura 70: Preparación del Visual Studio

Finalmente nuestro Visual Studio 2013 se instaló correctamente.

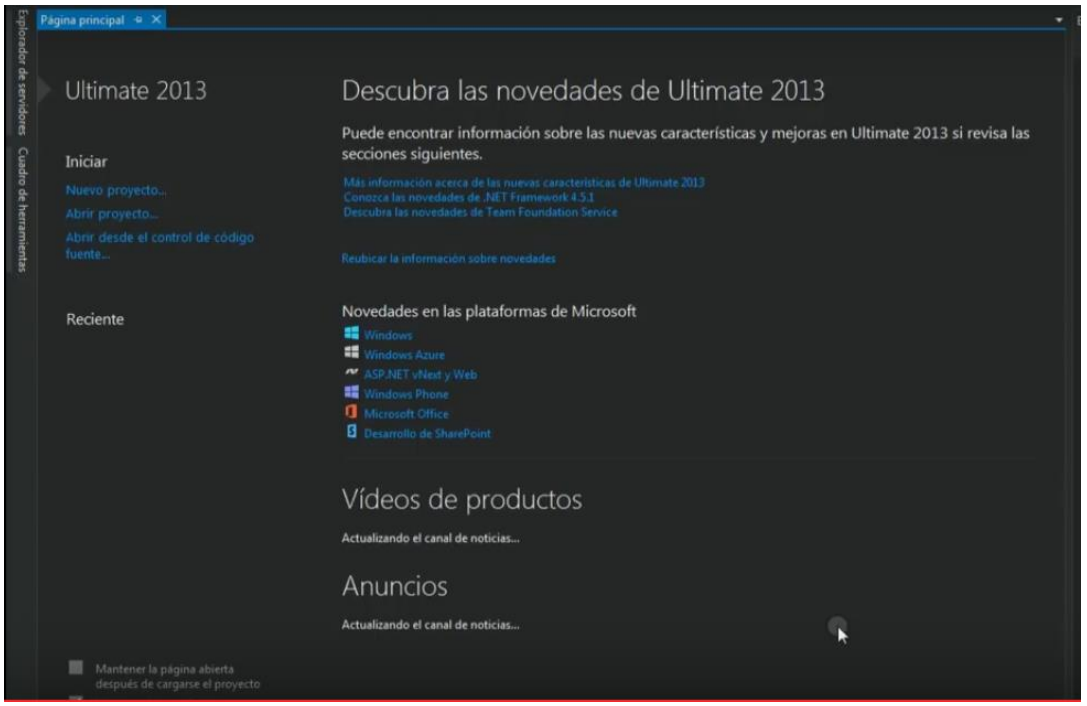


Figura 71: Pantalla de Inicio del Visual Studio 2013.

MANUAL DE USUARIO FINAL

ÍNDICE TABLAS

ÍNDICE DE FIGURAS

El presente manual, tiene como objetivo orientar al usuario para un mejor entendimiento del funcionamiento del aplicativo

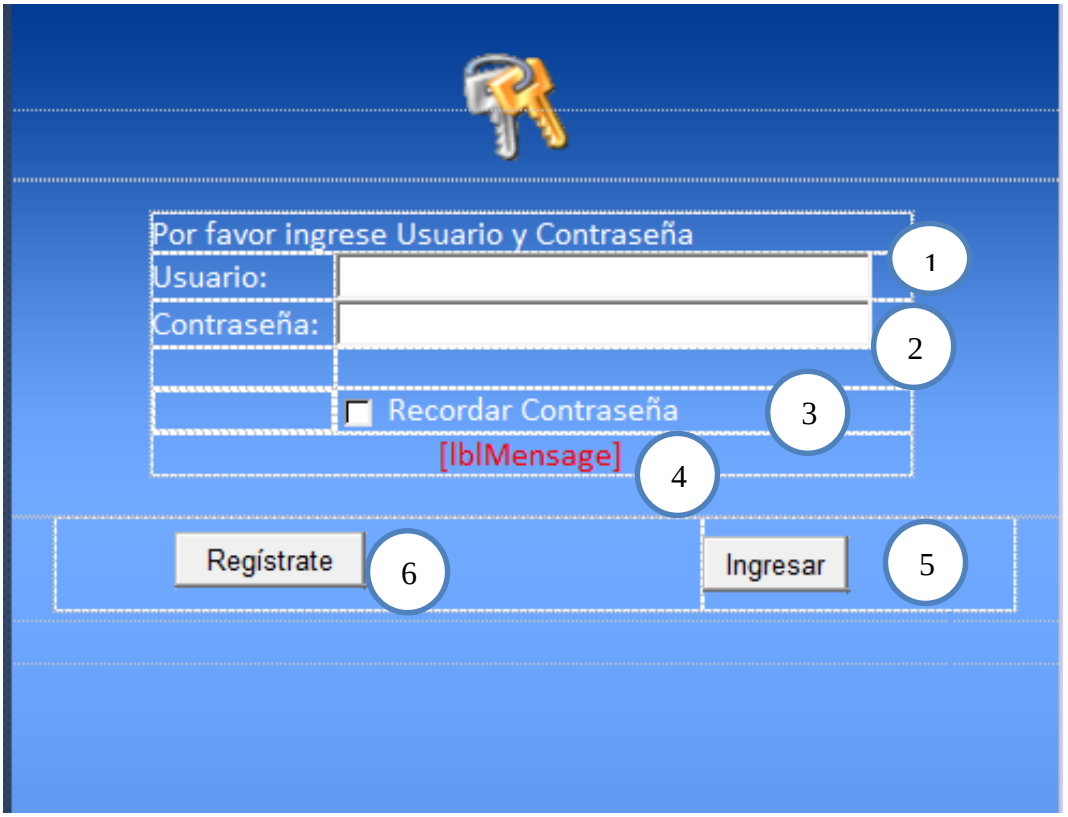


Figura 72: Ingreso al Sistema.
En la presente figura se visualiza la interfaz gráfica para el ingreso al sistema de una forma adecuada y sistemática.

Tabla 41: Ingreso al Sistema

NUMERACIÓN	REPRESENTACIÓN	PREFIJO	DESCRIPCIÓN
1	Textbox	Txt	Nombre de Usuario
2	Textbox	Txt	Contraseña
3	Checkbox	Chk	Recordar Contraseña
4	Label	Lbl	Mensaje
5	Button	Btn	Inicio de Sesión
6	Button	Btn	Regístrate

Nota: Ingreso al Sistema. Nos permite identificar la manera de cómo se desarrolla nuestro proyecto para el ingreso al sistema.

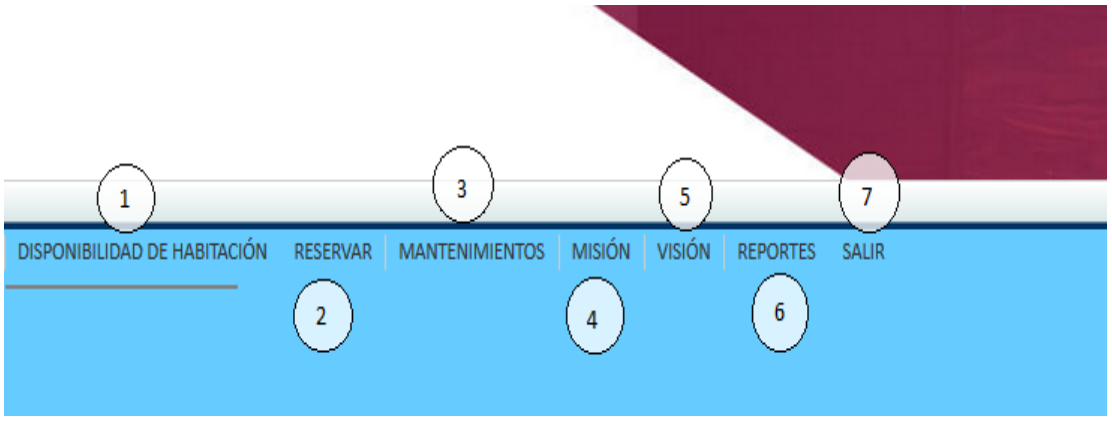


Figura 73: Pantalla de Inicio
En la presente figura se visualiza la interfaz gráfica de la pantalla de Inicio del Sistema de una manera agradable.

Tabla 42: Pantalla de Inicio

NUMERACIÓN	REPRESENTACIÓN	DESCRIPCIÓN
1	Textbox	Nombre de Usuario
2	Textbox	Disponibilidad de Habitación
3	Textbox	Reservar
4	Textbox	Mantenimientos
5	Textbox	Misión
6	Textbox	Visión
7	Textbox	Salir

Nota: Pantalla de Inicio. Nos permite identificar la manera de cómo se desarrolla nuestro proyecto para la pantalla de inicio.

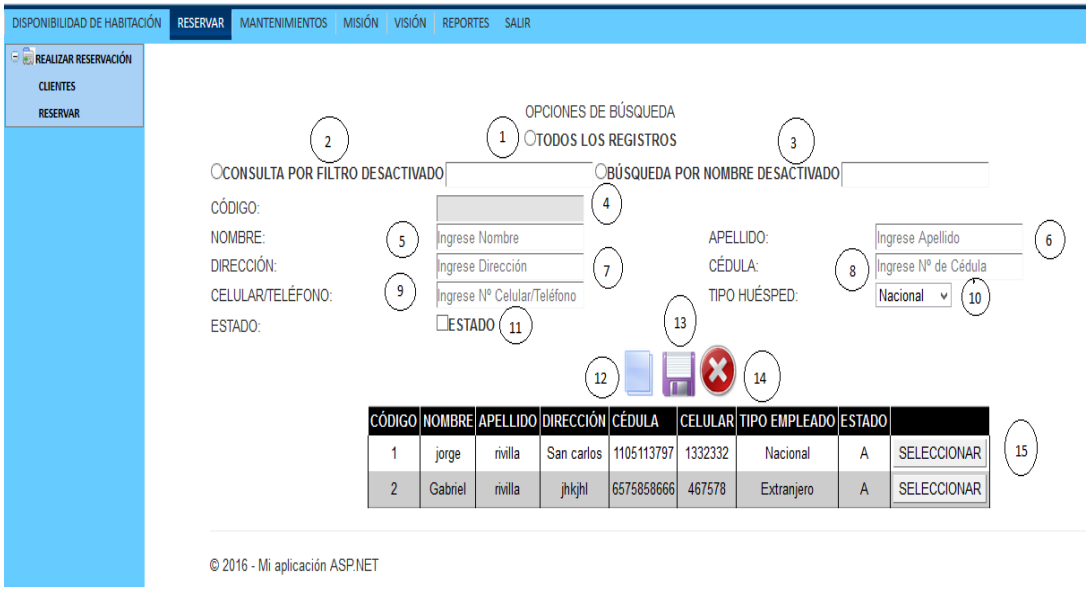


Figura 74: Registro del Huésped
En la presente figura se visualiza la interfaz gráfica de la pantalla de ingreso de Huésped una manera didáctica.

Tabla 43: Registro del Huésped

NUMERACIÓN	REPRESENTACIÓN	DESCRIPCIÓN
1	RadioButton	Todos los Registros
2	RadioButton ,Textbox	Consulta por Filtro
3	RadioButton ,Textbox	Consulta por Nombre
4	Textbox	Código
5	Textbox	Nombre
6	Textbox	Apellido
7	Textbox	Dirección
8	Textbox	Cédula
9	Textbox	Celular/Teléfono
10	Dropdownlist	Tipo Huésped
11	CheckList	Estado
12	ImageButton	Nuevo
13	ImageButton	Guardar
14	ImageButton	Eliminar
15	GridView	Datos

Nota: Registro de Huésped. Nos permite identificar la manera de cómo se desarrolla nuestro proyecto para el ingreso del Huésped.

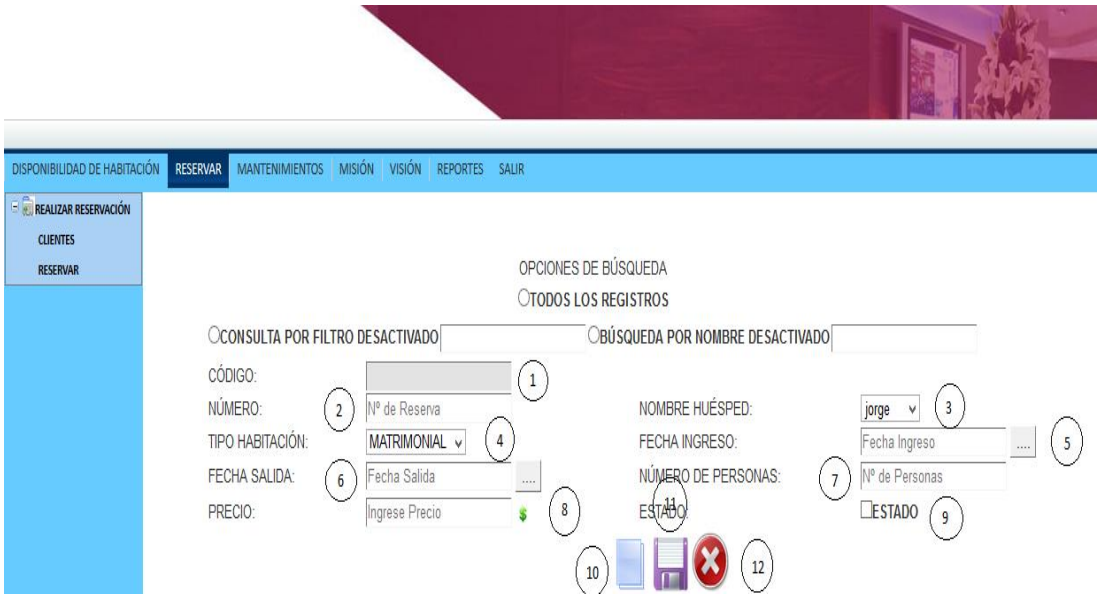


Figura 75: Realizar Reserva
En la presente figura se visualiza la interfaz gráfica de la pantalla de ingreso de Reserva una manera didáctica.

Tabla 44: Realizar Reserva

NUMERACIÓN	REPRESENTACIÓN	DESCRIPCIÓN
1	Textbox	Código
2	Textbox	Número
3	Dropdownlist	Nombre de Huésped
4	Dropdownlist	Tipo habitación
5	Textbox	Fecha Ingreso
6	Textbox	Fecha Salida
7	Textbox	Número de Personas
8	Textbox	Precio
9	CheckList	Estado
10	ImageButton	Nuevo
11	ImageButton	Guardar
12	ImageButton	Eliminar

Nota: Realizar Reserva. Nos permite identificar la manera de cómo se desarrolla nuestro proyecto para el registro de reserva

MANUAL TÉCNICO

ÍNDICE TABLAS

El presente manual tiene como objetivo visualizar las tablas que hemos utilizado para el desarrollo del proyecto y el lenguaje de programación que hemos utilizado

Diccionario de Datos

Tabla 45: Diccionario de Datos (Usuario)

NOMBRE	TIPO	CLAVE	DESCRIPCIÓN
idusuarios	Int	PK	Clave primaria de la tabla.
cedula	varchar		Número de cedula del usuario.
nombre	varchar		Nombre del usuario
Apellido	varchar		Apellido del usuario
estado	char		Estado del usuario.

Nota: PK Clave primaria (Primary Key)

Tabla 46: Diccionario de Datos (Login)

NOMBRE	TIPO	CLAVE	DESCRIPCIÓN
idlogin	Int	PK	Clave primaria de la tabla.
usuario	Varchar		Nombre de usuario
password	Varchar		Contraseña de usuario
estado	Char		Estado de la tabla
idusuarios	Int	FK	Clave Foránea de la tabla de datos usuario.

Nota: PK Clave primaria (Primary Key), FK Clave Foránea (Foreing Key).

Tabla 47: Diccionario de Datos (Huésped)

NOMBRE	TIPO	CLAVE	DESCRIPCIÓN
hue_codigo	int	PK	Clave primaria de la tabla.
hue_nombre	varchar		Nombre del huésped
hue_apellido	varchar		Apellido del huésped
hue_direccion	varchar		Dirección del huésped
hue_cedula	varchar		Número de cedula del huésped
hue_correo	varchar		Correo electrónico del huésped
hue_codigo	int	FK	Clave foránea de la tabla de datos tipo huésped,
estado	char		Estado de la tabla huésped

Nota: PK Clave primaria (Primary Key), FK Clave Foránea (Foreing Key).

Tabla 48: Diccionario de Datos (Habitación)

NOMBRE	TIPO	CLAVE	DESCRIPCIÓN
hab_codigo	int	PK	Clave primaria de la tabla.
hab_numero	int		Número de habitación
hab_descripcion	varchar		Descripción de habitación
hab_precio	varchar		Precio de habitación
tip_codigo	int	FK	Clave foránea de la tabla de datos tipohabitación
hot_codigo	int	FK	Clave foránea de la tabla de datos hotel
est_codigo	int	FK	Clave foránea de la tabla de datos estado
hab_estado	char		Estado de la tabla

Nota: PK Clave primaria (Primary Key), FK Clave Foránea (Foreing Key).

Tabla 49: Diccionario de Datos (Reservación)

NOMBRE	TIPO	CLAVE	DESCRIPCIÓN
res_codigo	int	PK	Clave primaria de la tabla.
hab_codigo	int	FK	Clave foránea de la tabla de datos habitación
hue_codigo	int	FK	Clave foránea de la tabla de datos huésped
tip_codigo	int	FK	Clave foránea de la tabla de datos tipohabitación
res_ingreso	datetime		Fecha de ingreso
res_salida	datetime		Fecha de salida
res_cantidad	int		Número de personas
res_precio	varchar		Precio
res_estado	char		Estado de la tabla

Nota: PK Clave primaria (Primary Key), FK Clave Foránea (Foreing Key).

Script de la Base de Datos

```
CREATE DATABASE TESIS
GO
USE [TESIS]
GO


CREATE PROCEDURE  [dbo].[SPSA_DETALLESERVICIO]

    @CODIGO INT,
        @FECHA DATETIME2(7),
    @SERVICIO INT,
        @TIPOHABITACION INT,
        @ESTADO Char(1)
as
begin
 if EXISTS ( SELECT *  FROM DETALLESERVICIO WHERE det_codigo =
@CODIGO)
        BEGIN
```

```
        UPDATE  DETALLESERVICIO SET

        det_fechaconsumo=@FECHA,
        ser_codigo=@SERVICIO,
        tip_codigo=@TIPOHABITACION,
        det_estado=@ESTADO


         WHERE det_codigo=@CODIGO
       END
       ELSE
       BEGIN
        INSERT INTO
DETALLESERVICIO(det_codigo,det_fechaconsumo,ser_codigo, tip_codigo,
det_estado)
         VALUES (@CODIGO,@FECHA, @SERVICIO,
@TIPOHABITACION,@ESTADO)
                 END
                 END

GO

CREATE PROCEDURE  [dbo].[SPSA_EMPLEADO]

    @CODIGO INT,
        @CEDULA VARCHAR (10),
    @NOMBRE VARCHAR (50),
        @APELLIDO VARCHAR (50),
        @ESTADO CHAR(1)
as
begin
 if EXISTS ( SELECT *  FROM USUARIOS WHERE idusuarios = @CODIGO)
       BEGIN
        UPDATE  USUARIOS SET


        cedula=@CEDULA,
        nombre=@NOMBRE,
        Apellido=@APELLIDO,
        estado=@ESTADO

         WHERE idusuarios=@CODIGO
       END
       ELSE
       BEGIN
        INSERT INTO USUARIOS (idusuarios,cedula,nombre, Apellido, estado)
         VALUES (@CODIGO, @CEDULA, @NOMBRE,
@APELLIDO,@ESTADO)
                 END
```

```
                END

GO


CREATE PROCEDURE  [dbo].[SPSA_ESTADO]

    @CODIGO INT,
    @DESCRIPCION VARCHAR (100)

as
begin
 if EXISTS ( SELECT *  FROM ESTADO  WHERE est_codigo = @CODIGO)
        BEGIN
         UPDATE  ESTADO SET


         est_descripcion=@DESCRIPCION


          WHERE est_codigo=@CODIGO
        END
        ELSE
        BEGIN
         INSERT INTO ESTADO (est_codigo,est_descripcion)
          VALUES (@CODIGO,@DESCRIPCION)
                END
                END
GO


CREATE PROCEDURE  [dbo].[SPSA_HABITACION]

    @CODIGO INT,
          @NUMERO INT,
    @DESCRIPCION VARCHAR (100),
          @PRECIO  VARCHAR (100),
    @TIPO INT,
          @HOTEL INT,
          @ESTADOHAB INT,
          @ESTADO Char(1)
as
begin
 if EXISTS ( SELECT *  FROM HABITACION WHERE hab_codigo =
@CODIGO)
        BEGIN
         UPDATE  HABITACION SET

         hab_numero=@NUMERO,
```

```
            hab_descripcion=@DESCRIPCION,
            hab_precio=@PRECIO,
            tip_codigo=@TIPO,
            hot_codigo=@HOTEL,
            est_codigo=@ESTADOHAB,
            hab_estado=@ESTADO


             WHERE hab_codigo=@CODIGO
           END
           ELSE
           BEGIN
            INSTERT INTO HABITACION(hab_codigo,hab_numero,hab_descripcion,
hab_precio, tip_codigo, hot_codigo,est_codigo, hab_estado)
             VALUES (@CODIGO,@NUMERO, @DESCRIPCION, @PRECIO,
@TIPO,@HOTEL, @ESTADOHAB,@ESTADO)
                    END
                    END

GO


CREATE PROCEDURE  [dbo].[SPSA_HOTEL]

    @CODIGO INT,
    @NOMBRE VARCHAR (100),
    @DIRECCION VARCHAR (100),
          @TELEFONO VARCHAR (100),
          @RUC VARCHAR (100)
          as
begin

 if EXISTS ( SELECT *  FROM HOTEL WHERE hot_codigo = @CODIGO)
          BEGIN
          UPDATE  HOTEL SET


          hot_nombre=@NOMBRE,
          hot_direccion=@DIRECCION,
          hot_telefono=@TELEFONO,
          hot_ruc=@RUC


           WHERE hot_codigo=@CODIGO
          END
          ELSE
          BEGIN
          INSERT INTO HOTEL (hot_codigo,hot_nombre,
hot_direccion,hot_telefono,hot_ruc )
```

```
        VALUES (@CODIGO, @NOMBRE, @DIRECCION, @TELEFONO,
@RUC)
                END
                END


GO


CREATE PROCEDURE  [dbo].[SPSA_HUESPED]

    @CODIGO INT,
    @NOMBRE VARCHAR (100),
        @APELLIDO VARCHAR (100),
        @DIRECCION VARCHAR (100),
        @CEDULA VARCHAR (100),
        @CELULAR VARCHAR (100),
        @TIPO INT,
        @ESTADO CHAR(1)
as
begin
 if EXISTS ( SELECT *  FROM HUESPED WHERE hue_codigo = @CODIGO)
        BEGIN
         UPDATE  HUESPED SET


         hue_nombre=@NOMBRE,
         hue_apellido=@APELLIDO,
         hue_direccion=@DIRECCION,
         hue_cedula=@CEDULA,
         hue_correo=@CELULAR,
         thue_codigo=@TIPO,
         estado=@ESTADO

          WHERE hue_codigo=@CODIGO
        END
        ELSE
        BEGIN
         INSERT INTO HUESPED(hue_codigo,hue_nombre,hue_apellido,
hue_direccion, hue_cedula, hue_correo,thue_codigo,estado)
         VALUES (@CODIGO, @NOMBRE,@APELLIDO,@DIRECCION,
@CEDULA, @CELULAR, @TIPO,@ESTADO)
                END
                END

GO


CREATE PROCEDURE  [dbo].[SPSA_ITEMMODULO]
```

```
    @CODIGO INT,
    @DESCRIPCION VARCHAR (100),
    @URL VARCHAR (100),
          @MODULO INT,
          @ESTADO CHAR(1)
as
begin
 if EXISTS ( SELECT *  FROM ITEMMODULO WHERE imod_codigo =
@CODIGO)
         BEGIN
          UPDATE  ITEMMODULO SET


          imod_descripcion=@DESCRIPCION,
          imod_url=@URL,
          mod_codigo=@MODULO,
          imod_estado=@ESTADO


           WHERE imod_codigo=@CODIGO
         END
         ELSE
         BEGIN
          INSERT INTO ITEMMODULO (imod_codigo,imod_descripcion,imod_url,
mod_codigo, imod_estado )
           VALUES (@CODIGO, @DESCRIPCION, @URL, @MODULO,
@ESTADO)
                  END
                  END

GO

CREATE PROCEDURE  [dbo].[SPSA_LOGIN]

    @CODIGO INT,
          @EMPLEADO VARCHAR(50),
    @PASS VARCHAR (50),
          @ESTADO CHAR(1),
          @USUARIO INT
as
begin
 if EXISTS ( SELECT *  FROM LOGIN WHERE idlogin = @CODIGO)
         BEGIN
          UPDATE  LOGIN SET


          usuario=@PASS,
          password=@EMPLEADO,
```

```
        estado=@ESTADO,
        idusuarios=@USUARIO


         WHERE idlogin=@CODIGO
       END
       ELSE
       BEGIN
        INSERT INTO LOGIN(idlogin,usuario,password, estado, idusuarios )

         VALUES (@CODIGO,  @EMPLEADO, @PASS, @ESTADO,
@USUARIO)
                END
                END

GO


CREATE PROCEDURE  [dbo].[SPSA_MODULO]

    @CODIGO INT,
    @NOMBRE VARCHAR (100),
    @DESCRIPCION VARCHAR (100),
        @ROL INT,
        @ESTADO CHAR(1)
as
begin
 if EXISTS ( SELECT *  FROM MODULO WHERE mod_codigo = @CODIGO)
        BEGIN
        UPDATE  MODULO SET


        mod_nombre=@NOMBRE,
        mod_descripcion=@DESCRIPCION,
        rol_codigo=@ROL,
        mod_estado=@ESTADO


         WHERE rol_codigo=@CODIGO
       END
       ELSE
       BEGIN
        INSERT INTO MODULO (mod_codigo,mod_nombre,mod_descripcion,
rol_codigo, mod_estado )
         VALUES (@CODIGO, @NOMBRE, @DESCRIPCION, @ROL,
@ESTADO)
                END
                END
```

```
GO

CREATE PROCEDURE  [dbo].[SPSA_RESERVA]

    @CODIGO INT,
    @NUMERO INT,
    @HUESPED INT,
          @TIPO INT,
          @INGRESO DATETIME2 (7),
          @SALIDA DATETIME2 (7),
          @CANTIDAD INT,
          @PRECIO VARCHAR (100),
          @ESTADO CHAR(1)
as
begin
 if EXISTS ( SELECT *  FROM RESERVACION WHERE res_codigo =
@CODIGO)
         BEGIN
          UPDATE  RESERVACION SET


          hab_codigo=@NUMERO,
          hue_codigo=@HUESPED,
          tip_codigo=@TIPO,
          res_ingreso=@INGRESO,
          res_salida=@SALIDA,
          res_cantidad=@CANTIDAD,
          res_precio=@PRECIO,
          res_estado=@ESTADO

           WHERE res_codigo=@CODIGO
         END
         ELSE
         BEGIN
          INSERT INTO RESERVACION(res_codigo,hab_codigo,hue_codigo,
tip_codigo, res_ingreso, res_salida,res_cantidad, res_precio, res_estado)
           VALUES (@CODIGO, @NUMERO, @HUESPED, @TIPO, @INGRESO,
@SALIDA, @CANTIDAD, @PRECIO, @ESTADO)
                  END
                  END
GO


CREATE PROCEDURE  [dbo].[SPSA_SERVICIO]

    @CODIGO INT,
    @NOMBRE VARCHAR (100),
```

```
            @COSTO VARCHAR(100),
            @ESTADO CHAR(1)
as
begin
 if EXISTS ( SELECT *  FROM SERVICIO WHERE ser_codigo = @CODIGO)
          BEGIN
           UPDATE  SERVICIO SET


           ser_nombre=@NOMBRE,
           ser_costo=@COSTO,
           ser_estado=@ESTADO

            WHERE ser_codigo=@CODIGO
          END
          ELSE
          BEGIN
           INSERT INTO SERVICIO(ser_codigo,ser_nombre,ser_costo, ser_estado )
            VALUES (@CODIGO, @NOMBRE,@COSTO, @ESTADO)
                  END
                  END

GO


CREATE PROCEDURE  [dbo].[SPSA_TIPOEMPLEADO]

    @CODIGO INT,
    @DESCRIPCION VARCHAR (100),
    @ESTADO CHAR (1)
as
begin
 if EXISTS ( SELECT *  FROM TIPOEMPLEADO WHERE temp_codigo =
@CODIGO)
          BEGIN
           UPDATE  TIPOEMPLEADO SET


           temp_descripcion=@DESCRIPCION,
           temp_estado=@ESTADO

            WHERE temp_codigo=@CODIGO
          END
          ELSE
          BEGIN
           INSERT INTO TIPOEMPLEADO
(temp_codigo,temp_descripcion,temp_estado)
            VALUES (@CODIGO,@DESCRIPCION,@ESTADO)
                  END
```

```
                END

GO


CREATE PROCEDURE  [dbo].[SPSA_TIPOHABITACION]

    @CODIGO INT,
    @DESCRIPCION VARCHAR (100),
         @ESTADO CHAR(1)
as
begin
 if EXISTS ( SELECT *  FROM TIPOHABITACION WHERE tip_codigo =
@CODIGO)
        BEGIN
         UPDATE  TIPOHABITACION SET


         tip_descripcion=@DESCRIPCION,
         tip_estado=@ESTADO


          WHERE tip_codigo=@CODIGO
        END
        ELSE
        BEGIN
         INSERT INTO TIPOHABITACION(tip_codigo,tip_descripcion, tip_estado
)
          VALUES (@CODIGO, @DESCRIPCION, @ESTADO)
                END

GO

CREATE PROCEDURE  [dbo].[SPSA_TIPOHUESPED]

    @CODIGO INT,
    @DESCRIPCION VARCHAR (100),
    @ESTADO CHAR (1)
as
begin
 if EXISTS ( SELECT *  FROM TIPOHUESPED WHERE thue_codigo =
@CODIGO)
        BEGIN
         UPDATE  TIPOHUESPED SET


         thue_descripcion=@DESCRIPCION,
         thue_estado=@ESTADO
```

```
        WHERE thue_codigo=@CODIGO
    END
    ELSE
    BEGIN
     INSERT INTO TIPOHUESPED(thue_codigo,thue_descripcion,thue_estado)
      VALUES (@CODIGO,@DESCRIPCION,@ESTADO)
          END
          END

GO
CREATE TABLE [dbo].[DETALLESERVICIO](
    [det_codigo] [int] NOT NULL,
    [det_fechaconsumo] [datetime2](7) NOT NULL,
    [ser_codigo] [int] NULL,
    [tip_codigo] [int] NULL,
    [det_estado] [char](1) NULL,
 CONSTRAINT [PK_DETALLESERVICIO_1] PRIMARY KEY CLUSTERED
(
    [det_codigo] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO

CREATE TABLE [dbo].[ESTADO](
    [est_codigo] [int] NOT NULL,
    [est_descripcion] [varchar](100) NULL,
 CONSTRAINT [PK_ESTADO] PRIMARY KEY CLUSTERED
(
    [est_codigo] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO

CREATE TABLE [dbo].[HABITACION](
    [hab_codigo] [int] NOT NULL,
    [hab_numero] [int] NULL,
    [hab_descripcion] [varchar](100) NULL,
    [hab_precio] [varchar](100) NULL,
    [tip_codigo] [int] NULL,
    [hot_codigo] [int] NULL,
    [est_codigo] [int] NULL,
    [hab_estado] [char](1) NULL,
 CONSTRAINT [PK_HABITACION] PRIMARY KEY CLUSTERED
```

```
(
	[hab_codigo] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO

CREATE TABLE [dbo].[HOTEL](
	[hot_codigo] [int] NOT NULL,
	[hot_nombre] [varchar](100) NULL,
	[hot_direccion] [varchar](100) NULL,
	[hot_telefono] [varchar](100) NULL,
	[hot_ruc] [varchar](100) NULL,
 CONSTRAINT [PK_HOTEL] PRIMARY KEY CLUSTERED
(
	[hot_codigo] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO

CREATE TABLE [dbo].[HUESPED](
	[hue_codigo] [int] NOT NULL,
	[hue_nombre] [varchar](100) NULL,
	[hue_apellido] [varchar](100) NULL,
	[hue_direccion] [varchar](100) NULL,
	[hue_cedula] [varchar](100) NULL,
	[hue_correo] [varchar](100) NULL,
	[thue_codigo] [int] NULL,
	[estado] [char](1) NULL,
 CONSTRAINT [PK_HUESPED] PRIMARY KEY CLUSTERED
(
	[hue_codigo] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO

CREATE TABLE [dbo].[ITEMMENU](
	[idItemMenu] [int] NOT NULL,
	[nombre] [varchar](50) NULL,
	[descripcion] [varchar](200) NULL,
	[url] [text] NULL,
```

```
	[estado] [char](1) NULL,
	[idmenu] [int] NOT NULL,
PRIMARY KEY CLUSTERED
(
	[idItemMenu] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO

CREATE TABLE [dbo].[LOGIN](
	[idlogin] [int] NOT NULL,
	[usuario] [varchar](50) NULL,
	[password] [varchar](50) NULL,
	[estado] [char](1) NULL,
	[idusuarios] [int] NOT NULL,
 CONSTRAINT [PK__LOGIN__961FA38E03317E3D] PRIMARY KEY
CLUSTERED
(
	[idlogin] ASC,
	[idusuarios] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO

CREATE TABLE [dbo].[LOGS](
	[idlog] [int] NOT NULL,
	[fecha] [datetime] NULL,
	[mensaje] [text] NULL,
	[error] [text] NULL,
 CONSTRAINT [PK_LOGS] PRIMARY KEY CLUSTERED
(
	[idlog] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO

CREATE TABLE [dbo].[MENU](
	[idmenu] [int] NOT NULL,
	[nombre] [varchar](50) NULL,
	[descripcion] [varchar](200) NULL,
```

```
	[url] [text] NULL,
	[estado] [char](1) NULL,
	[idmodulo] [int] NOT NULL,
PRIMARY KEY CLUSTERED
(
	[idmenu] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO

CREATE TABLE [dbo].[MODULO](
	[idmodulo] [int] NOT NULL,
	[nombre] [varchar](50) NULL,
	[descripcion] [varchar](200) NULL,
	[estado] [char](1) NULL,
	[idusuarios] [int] NOT NULL,
PRIMARY KEY CLUSTERED
(
	[idmodulo] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO

CREATE TABLE [dbo].[PAGO](
	[pag_codigo] [int] NOT NULL,
	[pag_descripcion] [varchar](50) NULL,
	[pag_estado] [char](1) NULL,
	[hue_codigo] [int] NULL,
 CONSTRAINT [PK_PAGO] PRIMARY KEY CLUSTERED
(
	[pag_codigo] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO


CREATE TABLE [dbo].[RESERVACION](
	[res_codigo] [int] NOT NULL,
	[hab_codigo] [int] NULL,
	[hue_codigo] [int] NULL,
```

```
	[tip_codigo] [int] NULL,
	[res_ingreso] [datetime2](7) NULL,
	[res_salida] [datetime2](7) NULL,
	[res_cantidad] [int] NULL,
	[res_precio] [varchar](100) NULL,
	[res_estado] [char](1) NULL,
 CONSTRAINT [PK_RESERVACION] PRIMARY KEY CLUSTERED
(
	[res_codigo] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO

CREATE TABLE [dbo].[SERVICIO](
	[ser_codigo] [int] NOT NULL,
	[ser_nombre] [varchar](100) NULL,
	[ser_costo] [varchar](100) NULL,
	[ser_estado] [char](1) NULL,
 CONSTRAINT [PK_SERVICIO] PRIMARY KEY CLUSTERED
(
	[ser_codigo] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO

CREATE TABLE [dbo].[TIPOHABITACION](
	[tip_codigo] [int] NOT NULL,
	[tip_descripcion] [varchar](100) NULL,
	[tip_estado] [char](1) NULL,
 CONSTRAINT [PK_TIPOHABITACION] PRIMARY KEY CLUSTERED
(
	[tip_codigo] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO

CREATE TABLE [dbo].[TIPOHUESPED](
	[thue_codigo] [int] NOT NULL,
	[thue_descripcion] [varchar](100) NULL,
	[thue_estado] [char](1) NULL,
```

```
 CONSTRAINT [PK_TIPOHUESPED] PRIMARY KEY CLUSTERED
(
	[thue_codigo] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO

CREATE TABLE [dbo].[USUARIOS](
	[idusuarios] [int] NOT NULL,
	[cedula] [varchar](10) NULL,
	[nombre] [varchar](50) NULL,
	[Apellido] [varchar](50) NULL,
	[estado] [char](1) NULL,
PRIMARY KEY CLUSTERED
(
	[idusuarios] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO

CREATE VIEW [dbo].[Empleadovista]
AS
SELECT       dbo.EMPLEADOS.emp_codigo, dbo.EMPLEADOS.emp_nombre,
dbo.EMPLEADOS.emp_nombres, dbo.EMPLEADOS.emp_apellido,
dbo.EMPLEADOS.emp_apellidos, dbo.EMPLEADOS.emp_fechaingreso,
                  dbo.EMPLEADOS.emp_fechasalida,
dbo.EMPLEADOS.emp_cedula, dbo.EMPLEADOS.emp_direccion,
dbo.EMPLEADOS.emp_correo, dbo.EMPLEADOS.emp_estado,
dbo.TIPOEMPLEADO.temp_codigo,
                  dbo.TIPOEMPLEADO.temp_descripcion,
dbo.TIPOEMPLEADO.temp_estado
FROM          dbo.EMPLEADOS INNER JOIN
                  dbo.TIPOEMPLEADO ON dbo.EMPLEADOS.temp_codigo =
dbo.TIPOEMPLEADO.temp_codigo

GO

CREATE VIEW [dbo].[vistadetalleservicio]
AS
SELECT       dbo.DETALLESERVICIO.det_codigo,
dbo.DETALLESERVICIO.det_fechaconsumo, dbo.SERVICIO.ser_nombre,
dbo.TIPOHABITACION.tip_descripcion, dbo.DETALLESERVICIO.det_estado
FROM          dbo.DETALLESERVICIO INNER JOIN
```

```
                dbo.TIPOHABITACION ON dbo.DETALLESERVICIO.tip_codigo
= dbo.TIPOHABITACION.tip_codigo INNER JOIN
                dbo.SERVICIO ON dbo.DETALLESERVICIO.ser_codigo =
dbo.SERVICIO.ser_codigo

GO

CREATE VIEW [dbo].[vistaficha]
AS
SELECT      dbo.RESERVACION.res_codigo, dbo.RESERVACION.res_numero,
dbo.RESERVACION.res_ingreso, dbo.RESERVACION.res_salida,
dbo.RESERVACION.res_cantidad, dbo.RESERVACION.res_precio,
                dbo.RESERVACION.res_estado, dbo.HUESPED.hue_codigo,
dbo.HUESPED.hue_nombre + ' ' + dbo.HUESPED.hue_apellido AS Nombre,
dbo.HUESPED.hue_direccion, dbo.HUESPED.hue_cedula,
                dbo.HUESPED.hue_celeular, dbo.HUESPED.thue_codigo,
dbo.HUESPED.estado, dbo.PAGO.pag_codigo, dbo.PAGO.pag_descripcion,
dbo.PAGO.pag_estado, dbo.RESERVACION.tip_codigo
FROM        dbo.RESERVACION INNER JOIN
                dbo.HUESPED ON dbo.RESERVACION.hue_codigo =
dbo.HUESPED.hue_codigo CROSS JOIN
                dbo.PAGO

GO

CREATE VIEW [dbo].[vistahabitacion]
AS
SELECT      dbo.HABITACION.hab_codigo, dbo.HABITACION.hab_numero,
dbo.HABITACION.hab_descripcion, dbo.HABITACION.hab_precio,
dbo.TIPOHABITACION.tip_descripcion, dbo.HOTEL.hot_nombre,
                dbo.ESTADO.est_descripcion, dbo.HABITACION.hab_estado
FROM        dbo.HABITACION INNER JOIN
                dbo.HOTEL ON dbo.HABITACION.hot_codigo =
dbo.HOTEL.hot_codigo INNER JOIN
                dbo.ESTADO ON dbo.HABITACION.est_codigo =
dbo.ESTADO.est_codigo INNER JOIN
                dbo.TIPOHABITACION ON dbo.HABITACION.tip_codigo =
dbo.TIPOHABITACION.tip_codigo

GO

CREATE VIEW [dbo].[vistahuesped]
AS
SELECT      dbo.HUESPED.hue_codigo, dbo.HUESPED.hue_nombre,
dbo.HUESPED.hue_apellido, dbo.HUESPED.hue_direccion,
dbo.HUESPED.hue_cedula, dbo.HUESPED.hue_correo,
dbo.TIPOHUESPED.thue_descripcion,
                dbo.HUESPED.estado
FROM        dbo.HUESPED INNER JOIN
```

```
                dbo.TIPOHUESPED ON dbo.HUESPED.thue_codigo =
dbo.TIPOHUESPED.thue_codigo

GO

CREATE VIEW [dbo].[vistaitemmodulo]
AS
SELECT      dbo.ITEMMODULO.imod_codigo,
dbo.ITEMMODULO.imod_descripcion, dbo.ITEMMODULO.imod_url,
dbo.MODULO.mod_nombre, dbo.ITEMMODULO.imod_estado
FROM        dbo.MODULO INNER JOIN
                dbo.ITEMMODULO ON dbo.MODULO.mod_codigo =
dbo.ITEMMODULO.mod_codigo

GO

CREATE VIEW [dbo].[vistaLogin]
AS
SELECT      dbo.LOGIN.idlogin, dbo.LOGIN.usuario, dbo.LOGIN.password,
dbo.LOGIN.estado, dbo.USUARIOS.nombre
FROM        dbo.USUARIOS INNER JOIN
                dbo.LOGIN ON dbo.USUARIOS.idusuarios =
dbo.LOGIN.idusuarios

GO

CREATE VIEW [dbo].[vistamodulo]
AS
SELECT      dbo.MODULO.mod_codigo, dbo.MODULO.mod_nombre,
dbo.MODULO.mod_descripcion, dbo.ROL.rol_descripcion,
dbo.MODULO.mod_estado
FROM        dbo.ROL INNER JOIN
                dbo.MODULO ON dbo.ROL.rol_codigo =
dbo.MODULO.rol_codigo

GO

CREATE VIEW [dbo].[vistareserva]
AS
SELECT      dbo.RESERVACION.res_codigo, dbo.HABITACION.hab_numero,
dbo.HUESPED.hue_nombre, dbo.TIPOHABITACION.tip_descripcion,
dbo.RESERVACION.res_ingreso, dbo.RESERVACION.res_salida,
                dbo.RESERVACION.res_cantidad,
dbo.RESERVACION.res_precio, dbo.RESERVACION.res_estado
FROM        dbo.TIPOHABITACION INNER JOIN
                dbo.RESERVACION ON dbo.TIPOHABITACION.tip_codigo =
dbo.RESERVACION.tip_codigo INNER JOIN
                dbo.HUESPED ON dbo.RESERVACION.hue_codigo =
dbo.HUESPED.hue_codigo INNER JOIN
```

```
                dbo.HABITACION ON dbo.TIPOHABITACION.tip_codigo =
dbo.HABITACION.tip_codigo AND dbo.RESERVACION.hab_codigo =
dbo.HABITACION.hab_codigo

GO

ALTER TABLE [dbo].[DETALLESERVICIO]  WITH CHECK ADD
CONSTRAINT [FK_DETALLESERVICIO_SERVICIO] FOREIGN
KEY([ser_codigo])
REFERENCES [dbo].[SERVICIO] ([ser_codigo])
GO
ALTER TABLE [dbo].[DETALLESERVICIO] CHECK CONSTRAINT
[FK_DETALLESERVICIO_SERVICIO]
GO
ALTER TABLE [dbo].[DETALLESERVICIO]  WITH CHECK ADD
CONSTRAINT [FK_DETALLESERVICIO_TIPOHABITACION] FOREIGN
KEY([tip_codigo])
REFERENCES [dbo].[TIPOHABITACION] ([tip_codigo])
GO
ALTER TABLE [dbo].[DETALLESERVICIO] CHECK CONSTRAINT
[FK_DETALLESERVICIO_TIPOHABITACION]
GO
ALTER TABLE [dbo].[HABITACION]  WITH CHECK ADD  CONSTRAINT
[FK_HABITACION_ESTADO] FOREIGN KEY([est_codigo])
REFERENCES [dbo].[ESTADO] ([est_codigo])
GO
ALTER TABLE [dbo].[HABITACION] CHECK CONSTRAINT
[FK_HABITACION_ESTADO]
GO
ALTER TABLE [dbo].[HABITACION]  WITH CHECK ADD  CONSTRAINT
[FK_HABITACION_HOTEL] FOREIGN KEY([hot_codigo])
REFERENCES [dbo].[HOTEL] ([hot_codigo])
GO
ALTER TABLE [dbo].[HABITACION] CHECK CONSTRAINT
[FK_HABITACION_HOTEL]
GO
ALTER TABLE [dbo].[HABITACION]  WITH CHECK ADD  CONSTRAINT
[FK_HABITACION_TIPOHABITACION] FOREIGN KEY([tip_codigo])
REFERENCES [dbo].[TIPOHABITACION] ([tip_codigo])
GO
ALTER TABLE [dbo].[HABITACION] CHECK CONSTRAINT
[FK_HABITACION_TIPOHABITACION]
GO
ALTER TABLE [dbo].[HUESPED]  WITH CHECK ADD  CONSTRAINT
[FK_HUESPED_TIPOHUESPED] FOREIGN KEY([thue_codigo])
REFERENCES [dbo].[TIPOHUESPED] ([thue_codigo])
GO
ALTER TABLE [dbo].[HUESPED] CHECK CONSTRAINT
[FK_HUESPED_TIPOHUESPED]
```

```
GO
ALTER TABLE [dbo].[ITEMMENU]  WITH CHECK ADD FOREIGN
KEY([idmenu])
REFERENCES [dbo].[MENU] ([idmenu])
GO
ALTER TABLE [dbo].[LOGIN]  WITH CHECK ADD  CONSTRAINT
[FK__LOGIN__idusuario__108B795B] FOREIGN KEY([idusuarios])
REFERENCES [dbo].[USUARIOS] ([idusuarios])
GO
ALTER TABLE [dbo].[LOGIN] CHECK CONSTRAINT
[FK__LOGIN__idusuario__108B795B]
GO
ALTER TABLE [dbo].[MENU]  WITH CHECK ADD FOREIGN KEY([idmodulo])
REFERENCES [dbo].[MODULO] ([idmodulo])
GO
ALTER TABLE [dbo].[MODULO]  WITH CHECK ADD FOREIGN
KEY([idusuarios])
REFERENCES [dbo].[USUARIOS] ([idusuarios])
GO
ALTER TABLE [dbo].[PAGO]  WITH CHECK ADD  CONSTRAINT
[FK_PAGO_HUESPED] FOREIGN KEY([hue_codigo])
REFERENCES [dbo].[HUESPED] ([hue_codigo])
GO
ALTER TABLE [dbo].[PAGO] CHECK CONSTRAINT [FK_PAGO_HUESPED]
GO
ALTER TABLE [dbo].[RESERVACION]  WITH CHECK ADD  CONSTRAINT
[FK_RESERVACION_HABITACION] FOREIGN KEY([hab_codigo])
REFERENCES [dbo].[HABITACION] ([hab_codigo])
GO
ALTER TABLE [dbo].[RESERVACION] CHECK CONSTRAINT
[FK_RESERVACION_HABITACION]
GO
ALTER TABLE [dbo].[RESERVACION]  WITH CHECK ADD  CONSTRAINT
[FK_RESERVACION_HUESPED] FOREIGN KEY([hue_codigo])
REFERENCES [dbo].[HUESPED] ([hue_codigo])
GO
ALTER TABLE [dbo].[RESERVACION] CHECK CONSTRAINT
[FK_RESERVACION_HUESPED]
GO
ALTER TABLE [dbo].[RESERVACION]  WITH CHECK ADD  CONSTRAINT
[FK_RESERVACION_TIPOHABITACION] FOREIGN KEY([tip_codigo])
REFERENCES [dbo].[TIPOHABITACION] ([tip_codigo])
GO
ALTER TABLE [dbo].[RESERVACION] CHECK CONSTRAINT
[FK_RESERVACION_TIPOHABITACION]
GO
```

Conexión a la Base de Datos

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DATO
{
   class Conexion
   {


       public string conexionC()
       {

           // leemos la ruta  y el archivo de conexion
           StreamReader leerArchivo = new StreamReader("C:\\RESERVACION" +
"\\proyecto.txt");

           //tomar esa informacion y dirigirme a la base de datos


           string datos;
           datos = leerArchivo.ReadToEnd();
           return @"Data source = " + datos + "; Integrated Security=SSPI";


       }


   }
}
```

Código fuente del Sistema

Capa de Datos (huespedDALC.cs)

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace DATO
{
  public class huespedDALC
   {


      //referenciamos la clase de conexion
      Conexion conection = new Conexion();

      //creamos variables de conexion
      SqlConnection cnn = null;
      SqlCommand cmd = null;

      public DataSet traerHuesped(string[] dato)
      {

         //iniciamos la conexion
         cnn = new SqlConnection(conection.conexionC());
         DataSet dsDataSet = new DataSet();


         //generamos codigo de consulta general

         if (dato[0] == "*")
         {
            cmd = new SqlCommand("  select * from vistahuesped", cnn);

         }



         else if (dato[0] == "n")
         {
            cmd = new SqlCommand("select * from vistahuesped  " + " where
hue_nombre ='" + dato[1] + "'", cnn);
```

```
        }

        ////// por filtro
        else if (dato[0] == "f")
        {
            cmd = new SqlCommand("select * from vistahuesped  " + " where
hue_nombre like '%" + dato[1] + "%'", cnn);
        }



        //ejecutamos la consulta
        SqlDataAdapter sdaSqlDataAdapter = new SqlDataAdapter(cmd);

        // abrimos la conexion de sqlserver
        cnn.Open();


        // enlazamos la consulta
        sdaSqlDataAdapter.Fill(dsDataSet);
        cnn.Close();


        //retornamos la informacion mediante el data set
        return dsDataSet;

    }



    public DataSet Generar_Codigo()
    {

        //iniciamos la conexion
        cnn = new SqlConnection(conection.conexionC());
        DataSet dsDataSet = new DataSet();


        //generamos codigo de consulta general


        cmd = new SqlCommand("select max(hue_codigo +1) AS Codigo from
HUESPED ", cnn);


        //ejecutamos la consulta
        SqlDataAdapter sdaSqlDataAdapter = new SqlDataAdapter(cmd);
        // abrimos la conexion de sqlserver
```

```
        cnn.Open();
        // enlazamos la consulta
        sdaSqlDataAdapter.Fill(dsDataSet);
        cnn.Close();
        //retornamos la informacion mediante el data set
        return dsDataSet;

    }



    public bool GuardarHuesped(string[] datos)
    // cnn = new SqlConnection(conection.conexionC());
    {
        SqlConnection con = new SqlConnection(conection.conexionC());
        con.Open();




        SqlCommand cmd = new SqlCommand("SPSA_HUESPED", con);

        cmd.Parameters.Add("@CODIGO", SqlDbType.Int).Value = datos[0];
        cmd.Parameters.Add("@NOMBRE", SqlDbType.VarChar, 100).Value =
datos[1];
        cmd.Parameters.Add("@APELLIDO", SqlDbType.VarChar, 100).Value =
datos[2];
        cmd.Parameters.Add("@DIRECCION", SqlDbType.VarChar, 100).Value =
datos[3];
        cmd.Parameters.Add("@CEDULA", SqlDbType.VarChar, 100).Value =
datos[4];
        cmd.Parameters.Add("@CELULAR", SqlDbType.VarChar, 100).Value =
datos[5];
        cmd.Parameters.Add("@TIPO", SqlDbType.Int).Value = datos[6];
        cmd.Parameters.Add("@ESTADO", SqlDbType.Char, 1).Value = datos[7];




        cmd.CommandType = CommandType.StoredProcedure;
        int exec = cmd.ExecuteNonQuery();
        if (exec == 0)
        {

            return false;
        }
```

```
            else
            {
                return true;
            }


        }



        public bool EliminarHuesped(string[] datos)
        {
            SqlConnection con = new SqlConnection(conection.conexionC());
            con.Open();
            SqlCommand cmd = new SqlCommand("SPSD_HUESPED", con);
            cmd.Parameters.Add("@CODIGO", SqlDbType.Int).Value = datos[0];

            cmd.CommandType = CommandType.StoredProcedure;
            int exec = cmd.ExecuteNonQuery();
            if (exec == 0)
            {

                return false;
            }
            else
            {
                return true;
            }


        }

    }
}
```

Capa de Datos (reservaDALC.cs)

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;


namespace DATO
{
```

```
public  class reservaDALC
  {



      //referenciamos la clase de conexion
      Conexion conection = new Conexion();

      //creamos variables de conexion
      SqlConnection cnn = null;
      SqlCommand cmd = null;

      public DataSet traerReserva(string[] dato)
      {

         //iniciamos la conexion
         cnn = new SqlConnection(conection.conexionC());
         DataSet dsDataSet = new DataSet();


         //generamos codigo de consulta general

         if (dato[0] == "*")
         {
            cmd = new SqlCommand("select * from vistareserva ", cnn);

         }



         else if (dato[0] == "n")
         {
            cmd = new SqlCommand("select * from vistareserva  " + " where
hab_numero ='" + dato[1] + "'", cnn);

         }

         ////// por filtro
         else if (dato[0] == "f")
         {
            cmd = new SqlCommand("select * from vistareserva  " + " where
hue_nombre like '%" + dato[1] + "%'", cnn);
         }



         //ejecutamos la consulta
         SqlDataAdapter sdaSqlDataAdapter = new SqlDataAdapter(cmd);
```

```
        // abrimos la conexion de sqlserver
        cnn.Open();


        // enlazamos la consulta
        sdaSqlDataAdapter.Fill(dsDataSet);
        cnn.Close();


        //retornamos la informacion mediante el data set
        return dsDataSet;

    }


    public DataSet Generar_Codigo()
    {

        //iniciamos la conexion
        cnn = new SqlConnection(conection.conexionC());
        DataSet dsDataSet = new DataSet();


        //generamos codigo de consulta general


        cmd = new SqlCommand("select max(res_codigo +1) AS Codigo from
RESERVACION ", cnn);


        //ejecutamos la consulta
        SqlDataAdapter sdaSqlDataAdapter = new SqlDataAdapter(cmd);
        // abrimos la conexion de sqlserver
        cnn.Open();
        // enlazamos la consulta
        sdaSqlDataAdapter.Fill(dsDataSet);
        cnn.Close();
        //retornamos la informacion mediante el data set
        return dsDataSet;

    }



    public bool GuardarReserva(string[] datos)
    // cnn = new SqlConnection(conection.conexionC());
    {
        SqlConnection con = new SqlConnection(conection.conexionC());
        con.Open();
```

```
        SqlCommand cmd = new SqlCommand("SPSA_RESERVA", con);

        cmd.Parameters.Add("@CODIGO", SqlDbType.Int).Value = datos[0];
        cmd.Parameters.Add("@NUMERO", SqlDbType.Int).Value = datos[1];
        cmd.Parameters.Add("@HUESPED", SqlDbType.Int).Value = datos[2];
        cmd.Parameters.Add("@TIPO", SqlDbType.Int).Value = datos[3];
        cmd.Parameters.Add("@INGRESO", SqlDbType.DateTime2, 7).Value =
datos[4];
        cmd.Parameters.Add("@SALIDA", SqlDbType.DateTime2, 7).Value =
datos[5];
        cmd.Parameters.Add("@CANTIDAD", SqlDbType.Int).Value = datos[6];
        cmd.Parameters.Add("@PRECIO", SqlDbType.VarChar, 100).Value =
datos[7];
        cmd.Parameters.Add("@ESTADO", SqlDbType.Char, 1).Value = datos[8];




        cmd.CommandType = CommandType.StoredProcedure;
        int exec = cmd.ExecuteNonQuery();
        if (exec == 0)
        {

            return false;
        }
        else
        {
            return true;
        }


    }



    public bool EliminarReserva(string[] datos)
    {
        SqlConnection con = new SqlConnection(conection.conexionC());
        con.Open();
        SqlCommand cmd = new SqlCommand("SPSD_RESERVA", con);
        cmd.Parameters.Add("@CODIGO", SqlDbType.Int).Value = datos[0];

        cmd.CommandType = CommandType.StoredProcedure;
        int exec = cmd.ExecuteNonQuery();
        if (exec == 0)
        {
```

```
            return false;
        }
        else
        {
            return true;
        }


    }



  }
}
```

Capa de negocio (manejadorHuesped.cs)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using DATO;
using System.Data;


namespace NEGOCIO
{
 public   class manejadorHuesped
   {



    huespedDALC empD = new huespedDALC();

    public DataSet traerHuesped(string[] dato)
    {

        return empD.traerHuesped(dato);

    }


    public bool GuardarHuesped(string[] dat)
    {
```

```
            return empD.GuardarHuesped(dat);

        }

        public bool EliminarHuesped(string[] dat)
        {

            return empD.EliminarHuesped(dat);
        }

        public DataSet Generar_Codigo()
        {
            return empD.Generar_Codigo();

        }

    }
}
```

Capa de negocio (manejadorReserva.cs)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using DATO;
using System.Data;

namespace NEGOCIO
{
 public  class manejadorReserva
   {

       //instanciamos la clse de datos que corresponde al usuario

       reservaDALC resD = new reservaDALC();

       public DataSet traerReserva(string[] dato)
       {

           return resD.traerReserva(dato);

       }


       public bool GuardarReserva(string[] dat)
       {
```

```
            return resD.GuardarReserva(dat);

        }

        public bool EliminarReserva(string[] dat)
        {

            return resD.EliminarReserva(dat);
        }

        public DataSet Generar_Codigo()
        {
            return resD.Generar_Codigo();

        }


    }
}
```

Capa de negocio (Frm_Huesped.aspx)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using NEGOCIO;
using System.Data;

namespace RESERVACIONES.Formularios
{
    public partial class Frm_Huesped : System.Web.UI.Page
    {

        #region VARIABLES CREADAS POR EL HUESPED

        manejadorHuesped hueM = new manejadorHuesped();


        void Generar_Codigo()
        {
            try
            {
                DataSet dsDataSet = new DataSet();
                dsDataSet = hueM.Generar_Codigo();
                DataTable dtDataTable = null;
```

```
            dtDataTable = dsDataSet.Tables[0];
            TXTCODIGO.Text = dsDataSet.Tables[0].Rows[0][0].ToString();
            if (TXTCODIGO.Text == "")
            {
                TXTCODIGO.Text = "1";
            }
        }
        catch (Exception ex)
        {
            //mensaje(ex.Message);
        }
    }

    public void traerHuesped(string[] dato)
    {
        try
        {

            DataSet dsDataSet = new DataSet();
            dsDataSet = hueM.traerHuesped(dato);
            DataTable dtDataTable = null;
            dtDataTable = dsDataSet.Tables[0];


            if (dato[0] != "n")
            {

                if (dtDataTable != null && dtDataTable.Rows.Count > 0)
                {

                    gvDatos.DataSource = dtDataTable;
                    gvDatos.DataBind();
                }
                else
                {

                    Response.Write("<script languaje = 'JavaScript'" + "> alert('no existe
datos con registro');</script>");
                }
            }
            else
            {
                if (dtDataTable != null && dtDataTable.Rows.Count > 0)
                {
                    foreach (DataRow drDataRow in dtDataTable.Rows)
                    {

                        TXTCODIGO.Text = Convert.ToString(drDataRow[0]);
                        TXTNOMBRE.Text = Convert.ToString(drDataRow[1]);
```

```
                    TXTAPELLIDO.Text = Convert.ToString(drDataRow[2]);
                    TXTDIRECCION.Text = Convert.ToString(drDataRow[3]);
                    TXTCEDULA.Text = Convert.ToString(drDataRow[4]);
                    TXTCORREO.Text = Convert.ToString(drDataRow[5]);
                    CBXTIPO.SelectedItem.Text = Convert.ToString(drDataRow[6]);

                    if (drDataRow[7].ToString() == "A")
                    {

                        RBTESTADO.Checked = true;
                        RBTESTADO.Text = "Activo";
                    }
                    else
                    {

                        RBTESTADO.Checked = false;
                        RBTESTADO.Text = "Inactivo";
                    }
                }
            }
            else
            {
                Response.Write("<script languaje = 'JavaScript'" + ">alert ('no existe
datos con registro');</script>");


            }
        }
    }

    catch (Exception ex)
    {
        Response.Write("<script languaje = 'JavaScript'" + "> alert('" +
ex.Message + "');</script>");
    }
}

#endregion


protected void Page_Load(object sender, EventArgs e)
{

    if (!IsPostBack)
    {
        string[] dato = { "*" };
        traerHuesped(dato);
    }
}
```

```
protected void RBDTODOS_CheckedChanged1(object sender, EventArgs e)
{
   TXTCODIGO.Text = "";
   TXTNOMBRE.Text = "";
   TXTAPELLIDO.Text = "";
   TXTCEDULA.Text = "";
   TXTDIRECCION.Text = "";
   TXTCORREO.Text = "";
   //    CBXTIPO.Text = "";


   RBTESTADO.Checked = false;
   RBTESTADO.Text = "ESTADO";

   ////
   TXTBUSCAR.Text = "";
   TXTFILTRO.Text = "";
   RBDTODOS.Checked = true;

   RBDBUSQUEDA.Checked = false;

   RBDFILTRO.Checked = false;
   TXTFILTRO.Enabled = false;
   TXTBUSCAR.Enabled = false;

   if (RBDTODOS.Checked == true)
   {
      string[] dato = { "*", "A" };
      traerHuesped(dato);
   }
}

protected void RBDFILTRO_CheckedChanged1(object sender, EventArgs e)
{

   if (RBDFILTRO.Checked == true)
   {
      TXTCODIGO.Text = "";
      TXTNOMBRE.Text = "";

      TXTAPELLIDO.Text = "";

      TXTCEDULA.Text = "";
      TXTDIRECCION.Text = "";
      TXTCORREO.Text = "";
      //  CBXTIPO.Text = "";
```

```
            RBTESTADO.Checked = false;
            RBTESTADO.Text = "ESTADO";



            TXTBUSCAR.Text = "";
            RBDTODOS.Checked = false;
            RBDFILTRO.Checked = true;

            RBDBUSQUEDA.Checked = false;
            RBDFILTRO.Text = "BUSQUEDA POR FILTRO ACTIVADO";
            RBDBUSQUEDA.Text = "BUSQUEDA POR NOMBRE
DESACTIVADO";
            TXTBUSCAR.Enabled = false;
            TXTFILTRO.Enabled = true;
            TXTFILTRO.Focus();

        }
        else
        {
            RBDTODOS.Checked = false;
            RBDBUSQUEDA.Checked = true;
            TXTFILTRO.Enabled = false;

            RBDFILTRO.Checked = false;
            RBDFILTRO.Text = "BUSQUEDA POR FILTRO DESACTIVADO";

        }
    }

    protected void RBDBUSQUEDA_CheckedChanged1(object sender, EventArgs
e)
    {

        if (RBDBUSQUEDA.Checked == true)
        {
            TXTFILTRO.Text = "";

            RBDTODOS.Checked = false;

            RBDBUSQUEDA.Checked = true;
            RBDFILTRO.Checked = false;
            TXTFILTRO.Enabled = false;
            RBDBUSQUEDA.Text = "BUSQUEDA POR NOMBRE ACTIVADO";
            RBDFILTRO.Text = "BUSQUEDA POR FILTRO DESACTIVADO";
            TXTBUSCAR.Enabled = true;
            TXTBUSCAR.Focus();
```

```
        }
        else
        {
            RBDTODOS.Checked = false;
            TXTBUSCAR.Enabled = false;

            RBDBUSQUEDA.Checked = false;
            RBDBUSQUEDA.Text = "BUSQUEDA POR FUNCIÓN
DESACTIVADO";

        }
    }

    protected void TXTFILTRO_TextChanged(object sender, EventArgs e)
    {
        try
        {
            string[] nombre = {
                        "f",
                        TXTFILTRO.Text
                    };
            traerHuesped(nombre);
        }
        catch (Exception ex)
        {

            Response.Write("<script languaje = 'JavaScript'" + "> alert('" +
ex.Message + "');</script>");

        }
    }

    protected void TXTBUSCAR_TextChanged1(object sender, EventArgs e)
    {
        try
        {
            string[] nombre = {
                        "n",
                        TXTBUSCAR.Text
                    };
            traerHuesped(nombre);
        }
        catch (Exception ex)
        {

            Response.Write("<script languaje = 'JavaScript'" + "> alert('" +
ex.Message + "');</script>");
```

```
    }
}



protected void gvDatos_SelectedIndexChanged(object sender, EventArgs e)
{

     string estado = "";
    int fila = gvDatos.SelectedIndex;
    TXTCODIGO.Text = gvDatos.Rows[fila].Cells[0].Text;
    TXTNOMBRE.Text = gvDatos.Rows[fila].Cells[1].Text;

    TXTAPELLIDO.Text = gvDatos.Rows[fila].Cells[2].Text;
      TXTDIRECCION.Text = gvDatos.Rows[fila].Cells[3].Text;
    TXTCEDULA.Text = gvDatos.Rows[fila].Cells[4].Text;

    TXTCORREO.Text = gvDatos.Rows[fila].Cells[5].Text;
    CBXTIPO.SelectedItem.Text = gvDatos.Rows[fila].Cells[6].Text;



    estado = gvDatos.Rows[fila].Cells[7].Text;
    if (estado == "A")
    {

        RBTESTADO.Checked = true;
RBTESTADO.Text = "Activo";
    }
    else
    {
        RBTESTADO.Checked = false;
 RBTESTADO.Text = "Inactivo";
    }


    BTNNUEVO.Enabled = true;
    BTNGUARDAR.Enabled = true;
    BTNELIMINAR.Enabled = true;
}


protected void BTNNUEVO_Click(object sender, ImageClickEventArgs e)
{
    TXTCODIGO.Text = "";
    TXTNOMBRE.Text = "";
```

```
        TXTAPELLIDO.Text = "";

        TXTCEDULA.Text = "";
        TXTDIRECCION.Text = "";
        TXTCORREO.Text = "";
        //  CBXTIPO.Text = "";


        RBTESTADO.Checked = false;
        RBTESTADO.Text = "ESTADO";

        BTNNUEVO.Enabled = true;
        BTNGUARDAR.Enabled = true;
        BTNELIMINAR.Enabled = false;
        TXTCODIGO.Focus();
        Generar_Codigo();
    }

    protected void BTNGUARDAR_Click(object sender, ImageClickEventArgs e)
    {
        string estado = "";


        if (RBTESTADO.Checked == true)
        {
            estado = "A";
        }
        else
        {
            estado = "I";
        }

        string[] dato = { TXTCODIGO.Text, TXTNOMBRE.Text,
TXTAPELLIDO.Text,TXTDIRECCION.Text,

                          TXTCEDULA.Text, TXTCORREO.Text,
CBXTIPO.SelectedValue,estado };

        hueM.GuardarHuesped(dato);

        Response.Write("<script language=javascript>alert('Reserva realizada
exitosamente');</script>");

        string[] datos = { "*" };
        traerHuesped(datos);

        TXTCODIGO.Text = "";
        TXTNOMBRE.Text = "";
```

```
        TXTAPELLIDO.Text = "";

        TXTCEDULA.Text = "";
        TXTDIRECCION.Text = "";
        TXTCORREO.Text = "";
        //  CBXTIPO.Text = "";


        RBTESTADO.Checked = false;
        RBTESTADO.Text = "ESTADO";



        BTNNUEVO.Enabled = true;
        BTNGUARDAR.Enabled = false;
        BTNELIMINAR.Enabled = true;
        RBTESTADO.Checked = false;
    }

    protected void BTNELIMINAR_Click(object sender, ImageClickEventArgs e)
    {
        {
            string[] dato = { TXTCODIGO.Text };

            hueM.EliminarHuesped(dato);

            string[] datos = { "*" };

            traerHuesped(datos);



            TXTCODIGO.Text = "";
            TXTNOMBRE.Text = "";

            TXTAPELLIDO.Text = "";

            TXTCEDULA.Text = "";
            TXTDIRECCION.Text = "";
            TXTCORREO.Text = "";
            //  CBXTIPO.Text = "";


            RBTESTADO.Checked = false;
            RBTESTADO.Text = "ESTADO";


            BTNNUEVO.Enabled = true;
```

```
            BTNGUARDAR.Enabled = false;
            BTNELIMINAR.Enabled = true;
            RBTESTADO.Checked = false;
        }
    }

  }
}
```

Capa de negocio (Frm_Reserva.aspx)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using NEGOCIO;
using System.Data;

namespace RESERVACIONES.Formularios
{
  public partial class Frm_Reserva : System.Web.UI.Page
  {


    #region VARIABLES CREADAS POR EL EMPLEADO

    manejadorReserva resM = new manejadorReserva();



    void Generar_Codigo()
    {
      try
      {
        DataSet dsDataSet = new DataSet();
        dsDataSet = resM.Generar_Codigo();
        DataTable dtDataTable = null;
        dtDataTable = dsDataSet.Tables[0];
        TXTCODIGO.Text = dsDataSet.Tables[0].Rows[0][0].ToString();
        if (TXTCODIGO.Text == "")
        {
          TXTCODIGO.Text = "1";
        }
      }
      catch (Exception ex)
      {
        //mensaje(ex.Message);
```

```
        }
    }

    FichaDalc ficha = new FichaDalc();


        private void Reporte_Ficha(string[] datos)
        {
            try
            {
                DataSet dsDataSet = new DataSet();
                dsDataSet = resM.TraerFicha(datos);

                DataTable dtDataTable = null;
                dtDataTable = dsDataSet.Tables[0];
                var reporte = new ReportDocument();
            reporte.Load(Server.MapPath("~/Reportes/Prueba.rpt"));
            reporte.SetDataSource(dtDataTable);
            reporte.DataSourceConnections[0].SetConnection("DAVID", "TESIS",
"sa", "12345");

            CrystalReportViewer1.ReportSource = reporte;


        }
        catch (Exception ex)
        {
            Response.Write("<Script>alert.window('" + ex.Message + "')</Script>");
        }
    }


    public void traerReserva(string[] dato)
    {
        try
        {

            DataSet dsDataSet = new DataSet();
            dsDataSet = resM.traerReserva(dato);
            DataTable dtDataTable = null;
            dtDataTable = dsDataSet.Tables[0];


            if (dato[0] != "n")
            {

                if (dtDataTable != null && dtDataTable.Rows.Count > 0)
                {
```

```
                gvDatos.DataSource = dtDataTable;
                gvDatos.DataBind();
            }
            else
            {

                Response.Write("<script languaje = 'JavaScript'" + "> alert('no existe
datos con registro');</script>");
            }
        }
        else
        {
            if (dtDataTable != null && dtDataTable.Rows.Count > 0)
            {
                foreach (DataRow drDataRow in dtDataTable.Rows)
                {

                    TXTCODIGO.Text = Convert.ToString(drDataRow[0]);
                    CBXNUMERO.SelectedItem.Text =
Convert.ToString(drDataRow[1]);
                    CBXNOMBRE.SelectedItem.Text =
Convert.ToString(drDataRow[2]);
                    CBXTIPO.SelectedItem.Text = Convert.ToString(drDataRow[3]);
                    TXTINGRESO.Text = Convert.ToString(drDataRow[4]);
                    TXTSALIDA.Text = Convert.ToString(drDataRow[5]);
                    TXTCEDULA.Text = Convert.ToString(drDataRow[6]);
                    TXTDIRECCION.Text = Convert.ToString(drDataRow[7]);




                    if (drDataRow[8].ToString() == "A")
                    {

                        RBTESTADO.Checked = true;
                        RBTESTADO.Text = "Activo";
                    }
                    else
                    {

                        RBTESTADO.Checked = false;
                        RBTESTADO.Text = "Inactivo";
                    }
                }
            }
            else
            {
```

```
                Response.Write("<script languaje = 'JavaScript'" + ">alert ('no existe datos con registro');</script>");



                }
            }
        }

        catch (Exception ex)
        {
            Response.Write("<script languaje = 'JavaScript'" + "> alert('" + ex.Message + "');</script>");
        }
    }

    #endregion



    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            string[] dato = { "*" };
            traerReserva(dato);
        }
    }

    protected void Calendar1_SelectionChanged1(object sender, EventArgs e)
    {
        string separar_Nombre = Calendar1.SelectedDate.ToString();
        string[] partes = separar_Nombre.Split(' ');
        TXTINGRESO.Text = partes[0];
        Calendar1.Visible = false;
    }

    protected void BTNCALENDARIO_Click(object sender, EventArgs e)
    {
        Calendar1.Visible = true;
    }

    protected void Calendar2_SelectionChanged(object sender, EventArgs e)
    {
        string separar_Nombre = Calendar2.SelectedDate.ToString();
        string[] partes = separar_Nombre.Split(' ');
        TXTSALIDA.Text = partes[0];
        Calendar2.Visible = false;
```

```
}

protected void BTNCALENDAR_Click(object sender, EventArgs e)
{
   Calendar2.Visible = true;
}

protected void RBDTODOS_CheckedChanged1(object sender, EventArgs e)
{
   TXTCODIGO.Text = "";

   TXTINGRESO.Text = "";
   TXTSALIDA.Text = "";
   TXTCEDULA.Text = "";
   TXTDIRECCION.Text = "";

   //    CBXTIPO.Text = "";


   RBTESTADO.Checked = false;
   RBTESTADO.Text = "ESTADO";

   ////
   TXTBUSCAR.Text = "";
   TXTFILTRO.Text = "";
   RBDTODOS.Checked = true;

   RBDBUSQUEDA.Checked = false;

   RBDFILTRO.Checked = false;
   TXTFILTRO.Enabled = false;
   TXTBUSCAR.Enabled = false;

   if (RBDTODOS.Checked == true)
   {
      string[] dato = { "*", "A" };
      traerReserva(dato);
   }
}

protected void RBDFILTRO_CheckedChanged1(object sender, EventArgs e)
{
   if (RBDFILTRO.Checked == true)
   {
      TXTCODIGO.Text = "";

      TXTINGRESO.Text = "";
      TXTSALIDA.Text = "";
      TXTCEDULA.Text = "";
```

```
            TXTDIRECCION.Text = "";

            //  CBXTIPO.Text = "";


            RBTESTADO.Checked = false;
            RBTESTADO.Text = "ESTADO";




            TXTBUSCAR.Text = "";
            RBDTODOS.Checked = false;
            RBDFILTRO.Checked = true;

            RBDBUSQUEDA.Checked = false;
            RBDFILTRO.Text = "BUSQUEDA POR FILTRO ACTIVADO";
            RBDBUSQUEDA.Text = "BUSQUEDA POR CÓDIGO
DESACTIVADO";
            TXTBUSCAR.Enabled = false;
            TXTFILTRO.Enabled = true;
            TXTFILTRO.Focus();

        }
        else
        {
            RBDTODOS.Checked = false;
            RBDBUSQUEDA.Checked = true;
            TXTFILTRO.Enabled = false;

            RBDFILTRO.Checked = false;
            RBDFILTRO.Text = "BUSQUEDA POR FILTRO DESACTIVADO";

        }
    }

    protected void RBDBUSQUEDA_CheckedChanged1(object sender, EventArgs
e)
    {
        if (RBDBUSQUEDA.Checked == true)
        {
            TXTFILTRO.Text = "";

            RBDTODOS.Checked = false;

            RBDBUSQUEDA.Checked = true;
            RBDFILTRO.Checked = false;
            TXTFILTRO.Enabled = false;
            RBDBUSQUEDA.Text = "BUSQUEDA POR CÓDIGO  ACTIVADO";
```

```
            RBDFILTRO.Text = "BUSQUEDA POR FILTRO DESACTIVADO";
            TXTBUSCAR.Enabled = true;
            TXTBUSCAR.Focus();


        }
        else
        {
            RBDTODOS.Checked = false;
            TXTBUSCAR.Enabled = false;

            RBDBUSQUEDA.Checked = false;
            RBDBUSQUEDA.Text = "BUSQUEDA POR FUNCIÓN
DESACTIVADO";

        }
    }

    protected void TXTFILTRO_TextChanged(object sender, EventArgs e)
    {
        try
        {
            string[] nombre = {
                        "f",
                        TXTFILTRO.Text
                    };
            traerReserva(nombre);
        }
        catch (Exception ex)
        {

            Response.Write("<script languaje = 'JavaScript'" + "> alert('" +
ex.Message + "');</script>");

        }
    }

    protected void TXTBUSCAR_TextChanged1(object sender, EventArgs e)
    {
        try
        {
            string[] nombre = {
                        "n",
                        TXTBUSCAR.Text
                    };
            traerReserva(nombre);
        }
        catch (Exception ex)
        {
```

```
            Response.Write("<script languaje = 'JavaScript'" + "> alert('" +
ex.Message + "');</script>");

        }
    }

    protected void BTNNUEVO_Click(object sender, EventArgs e)
    {
        TXTCODIGO.Text = "";


        TXTINGRESO.Text = "";
        TXTSALIDA.Text = "";
        TXTCEDULA.Text = "";
        TXTDIRECCION.Text = "";

        //  CBXTIPO.Text = "";


        RBTESTADO.Checked = false;
        RBTESTADO.Text = "ESTADO";

        BTNNUEVO.Enabled = true;
        BTNGUARDAR.Enabled = true;
        BTNELIMINAR.Enabled = false;
        TXTCODIGO.Focus();
    }

    protected void BTNGUARDAR_Click(object sender, EventArgs e)
    {


    }

    protected void BTNELIMINAR_Click(object sender, EventArgs e)
    {


    }

    protected void gvDatos_SelectedIndexChanged(object sender, EventArgs e)
    {

        string estado = "";
        int fila = gvDatos.SelectedIndex;
        TXTCODIGO.Text = gvDatos.Rows[fila].Cells[0].Text;
        CBXNUMERO.SelectedItem.Text = gvDatos.Rows[fila].Cells[1].Text;
        CBXNOMBRE.SelectedItem.Text = gvDatos.Rows[fila].Cells[2].Text;
```

```
    CBXTIPO.SelectedItem.Text = gvDatos.Rows[fila].Cells[3].Text;
    TXTINGRESO.Text = gvDatos.Rows[fila].Cells[4].Text;
    TXTSALIDA.Text = gvDatos.Rows[fila].Cells[5].Text;
    TXTCEDULA.Text = gvDatos.Rows[fila].Cells[6].Text;
    TXTDIRECCION.Text = gvDatos.Rows[fila].Cells[7].Text;




    estado = gvDatos.Rows[fila].Cells[8].Text;
    if (estado == "A")
    {

        RBTESTADO.Checked = true;
RBTESTADO.Text = "Activo";
    }
    else
    {
        RBTESTADO.Checked = false;
 RBTESTADO.Text = "Inactivo";
    }


    BTNNUEVO.Enabled = true;
    BTNGUARDAR.Enabled = true;
    BTNELIMINAR.Enabled = true;

 }

 protected void BTNNUEVO_Click1(object sender, ImageClickEventArgs e)
 {

    TXTCODIGO.Text = "";


    TXTINGRESO.Text = "";
    TXTSALIDA.Text = "";
    TXTCEDULA.Text = "";
    TXTDIRECCION.Text = "";

    //  CBXTIPO.Text = "";


    RBTESTADO.Checked = false;
    RBTESTADO.Text = "ESTADO";

    BTNNUEVO.Enabled = true;
```

```
        BTNGUARDAR.Enabled = true;
        BTNELIMINAR.Enabled = false;


        Generar_Codigo();
    }

    protected void BTNGUARDAR_Click1(object sender, ImageClickEventArgs e)
    {

        string estado = "";


        if (RBTESTADO.Checked == true)
        {
            estado = "A";
        }
        else
        {
            estado = "I";
        }

        string[] dato = { TXTCODIGO.Text, CBXNUMERO.SelectedValue,
CBXNOMBRE.SelectedValue, CBXTIPO.SelectedValue, TXTINGRESO.Text,
TXTSALIDA.Text,
                    TXTCEDULA.Text,TXTDIRECCION.Text,estado };

        resM.GuardarReserva(dato);

        Response.Write("<script language=javascript>alert('Registro
Guardado');</script>");


        string[] datos = { "*" };

        traerReserva(datos);

        /*    Application["reserva_Codigo"] = TXTCODIGO.Text;
          Response.Redirect("~/Reportes/Prueba.rpt");

         */



        TXTCODIGO.Text = "";

        TXTINGRESO.Text = "";
        TXTSALIDA.Text = "";
```

```
        TXTCEDULA.Text = "";
        TXTDIRECCION.Text = "";

        //  CBXTIPO.Text = "";


        RBTESTADO.Checked = false;
        RBTESTADO.Text = "ESTADO";



        BTNNUEVO.Enabled = true;
        BTNGUARDAR.Enabled = false;
        BTNELIMINAR.Enabled = true;
        RBTESTADO.Checked = false;



    }

    protected void BTNELIMINAR_Click1(object sender, ImageClickEventArgs e)
    {
        {
            string[] dato = { TXTCODIGO.Text };

            resM.EliminarReserva(dato);

            string[] datos = { "*" };

            traerReserva(datos);



            TXTCODIGO.Text = "";


            TXTINGRESO.Text = "";
            TXTSALIDA.Text = "";
            TXTCEDULA.Text = "";
            TXTDIRECCION.Text = "";

            //  CBXTIPO.Text = "";


            RBTESTADO.Checked = false;
            RBTESTADO.Text = "ESTADO";


            BTNNUEVO.Enabled = true;
```

```
            BTNGUARDAR.Enabled = false;
            BTNELIMINAR.Enabled = true;
            RBTESTADO.Checked = false;
        }
    }


}
}
```