



FACULTAD DE INGENIERÍA

CARRERA DE INGENIERÍA DE SISTEMAS COMPUTACIONALES

SISTEMA DE ADMINISTRACIÓN HOTELERA

Informe académico

Autor(es):

Diego Tasaico
Neftali Zapata
Olenka Lazo
Mauricio Villanueva

Curso:

Modelamiento y Análisis de Software

Docente:

Jorge Alfredo Guevara Jimenez

LIMA – PERÚ

2020-2

ÍNDICE

CAPÍTULO 1. INTRODUCCIÓN	1
1. Identificación del problema.....	1
2. Planteamiento de la solución.....	1
3. Cómo implementar la solución	1
4. Justificación y limitaciones de la investigación.....	1
5. Ventajas de la solución.....	2
6. Desventajas de la solución	2
7. Objetivo general	2
8. Objetivos específicos	2
CAPÍTULO 2. MARCO TEÓRICO	2
9. Marco teórico.....	2
10. Marco metodológico	3
CAPÍTULO 3. DESARROLLO DE LA INVESTIGACIÓN.....	4
11. Implementación de la solución planteada	4
CAPÍTULO 4. CONCLUSIONES Y RECOMENDACIONES.....	45
12. Conclusiones.....	45
13. Recomendaciones	45
14. Referencias Bibliográficas	45
15. Anexos	46

ÍNDICE DE TABLAS

.....	4
Tabla 1: Proceso de Negocio	4
Tabla 2: Trazabilidad de Requisitos	9
Tabla 3: Especificación de Casos de Uso	36

ÍNDICE DE FIGURAS

Ilustración 1: Proceso de Negocio.....	4
Ilustración 2: Proceso de Software.....	5
Ilustración 3: Diagrama de Casos de uso	5
Ilustración 4: Prototipos	6
Ilustración 5: Diagrama de actividades con particiones o calles.....	8
Ilustración 6: Requisitos no funcionales.....	11
Ilustración 7: Repositorio GitHub	18
Ilustración 8: Diagrama de Casos de Uso Relacionado.....	25
Ilustración 9: Funcionamiento del Software	26

CAPÍTULO 1. INTRODUCCIÓN

1. Identificación del problema

Actualmente el problema que presenta el hotel surge en el proceso de alquiler de habitaciones, ya que no cuenta con un sistema de administración y todo se realiza de forma manual, como el cobro por el servicio, el registro de habitaciones, la verificación de habitaciones disponibles y verificación de la cantidad de clientes que se hospedan durante un intervalo de tiempo. Este problema, si bien cumple con su objetivo que es la atención al cliente, demanda mucho tiempo, esfuerzo y malinterpretación de la información con respecto a los datos del cliente y su reserva.

2. Planteamiento de la solución

Para poder solucionar este problema lo que el hotel necesita es un sistema a través del cual se pueda buscar hospedaje, gestar la reserva de éste y realizar su administración, para ello se empezará con un modelo o prototipo que debe tener estas funcionalidades básicas.

3. Cómo implementar la solución

Se va a crear una solución de software que será usada por el usuario de una manera local para mejorar la productividad de la empresa y sus trabajadores. Para esto tomamos en cuenta las necesidades del cliente, el proceso que se lleva durante el registro y el producto final esperado del sistema desarrollado.

4. Justificación y limitaciones de la investigación

Limitaciones:

- Tiempo: Tendremos todo este ciclo académico para desarrollar sistema.
- Espacio: Todo el proyecto será realizado mediante sistemas digitales.
- Recursos: Tendremos que tomar en cuenta las limitaciones para la adquisición de los aparatos electrónicos.

Justificación:

- Se justifica en base a los avances tecnológicos y en su implementación en las grandes, medianas y pequeñas empresas.
- Se justifica en base a la eficacia y productividad de los trabajadores.
- Se justifica en base a la experiencia y servicio brindado a los clientes.

5. Ventajas de la solución

Como ventaja principal tenemos la eficacia que se generará en el registro de nuevos clientes, se podrá visualizar desde un primer momento un cambio significativo en el tiempo requerido para ello.

6. Desventajas de la solución

Como desventaja principal tenemos la adquisición del material necesario para la implementación del sistema, otra de las desventajas sería el gasto en capacitaciones para el personal administrativo del lugar.

7. Objetivo general

Desarrollar un sistema de administración hotelera, que pueda cumplir las necesidades de la empresa, contando con la posibilidad de hacer las reservas de las habitaciones y realizar el cobro respectivo al cliente.

8. Objetivos específicos

- Implementar un mantenimiento de clientes, donde se podrá registrar, consultar, modificar y eliminar a los clientes, incluyendo la entrada y salida del cliente.
- Implementar el mantenimiento de habitaciones y reservas.
- Permitir al administrador o recepcionista poder consultar sobre las habitaciones que fueron reservadas.

CAPÍTULO 2. MARCO TEÓRICO

9. Marco teórico

- Modelo de negocio: Alonso, Martínez y Segovia (2005) señalan que el modelo de negocio es necesario para poder entender los procesos de negocio de la empresa y así poder tener una descripción detallada de los requisitos del sistema.
- Proceso de software: “Un proceso de software es una secuencia de actividades que conducen a la elaboración de un producto de software.” (Sommerville, 2011, p. 9).
- Prototipos: “permiten a los usuarios ver que tan bien el sistema apoya a su trabajo.” (Sommerville, 2011, p. 45).
- Casos de uso: “un caso de uso identifica a los actores implicados en una interacción” (Sommerville, 2011, p. 107).
- Diagrama de casos de uso: “representa todas las interacciones que se describirán en los requerimientos del sistema.” (Sommerville, 2011, p. 107).

- Requerimientos: Alonso, Martínez y Segovia (2005) señalan que los requerimientos se reconocen a partir de las especificaciones del problema que va a mencionar el usuario.
- Diagramas de actividad: "Los diagramas de actividad intentan mostrar las actividades que incluyen un proceso de sistema, así como el flujo de control de una actividad a otra." (Sommerville, 2011, p. 123).
- Proceso unificado: Alonso, Martínez y Segovia (2005) señalan que el proceso unificado modela el sistema como un conjunto de bloques interconectados y se implementan mediante componentes.
- Git hub: Es una plataforma de desarrollo inspirada en tu forma de trabajar. Desde el código abierto hasta el negocio, puede alojar y revisar código, administrar proyectos y crear software junto con 50 millones de desarrolladores.
- Lucidchart: Ofrece una interfaz intuitiva y cientos de plantillas que permiten que tus equipos comuniquen sistemas, procesos e ideas complejos con gráficos. Eleva tu perspectiva en todas las áreas de tu negocio para impulsar ventas, simplificar la gestión del personal, mapear tu infraestructura y más.

10. Marco metodológico

- Paso 1: Formar el equipo de trabajo.
- Paso 2: Identificar una empresa.
- Paso 3: Entrevistar al administrador del negocio.
- Paso 4: Modelar y especificar el proceso de negocio.
- Paso 5: Hacer el modelo de proceso de software.
- Paso 6: Hacer el modelo de casos de uso.
- Paso 7: Creación de prototipos.
- Paso 8: Hacer el diagrama de actividades con particiones.
- Paso 9: Avance de informe

CAPÍTULO 3. DESARROLLO DE LA INVESTIGACIÓN

11. Implementación de la solución planteada

Ilustración 1: Proceso de Negocio

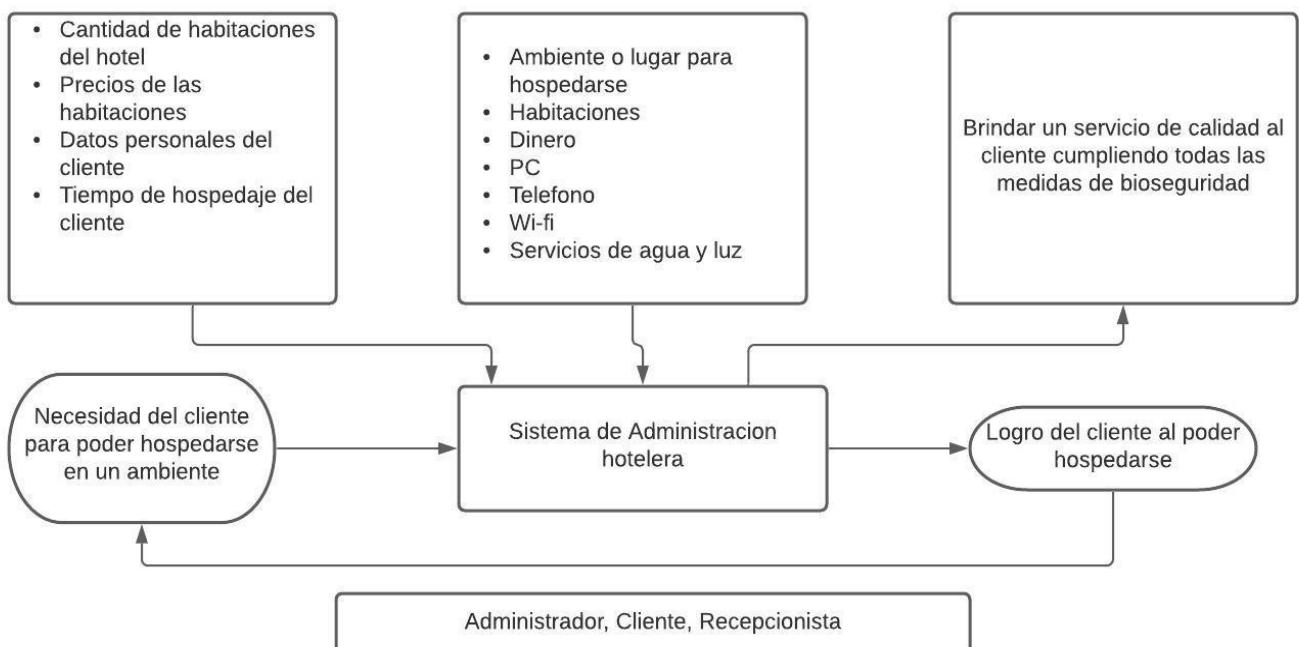


Tabla 1: Proceso de Negocio

Procesos	Entradas	Actividades	Salidas	Valor para el cliente
Sistema de Administración hotelera	Necesidad del cliente para poder hospedarse en un ambiente	<ul style="list-style-type: none"> Registro de cliente. Consulta de precio de habitación. Selección de habitación para el cliente. Verificar que la habitación se encuentre vacía y limpia. Realización del pago del cliente al hotel. Registrar salida del cliente del hotel. 	Registro salida cliente	Brindar servicio de calidad al cliente cumpliendo todas las medidas de bioseguridad

Ilustración 2: Proceso de Software

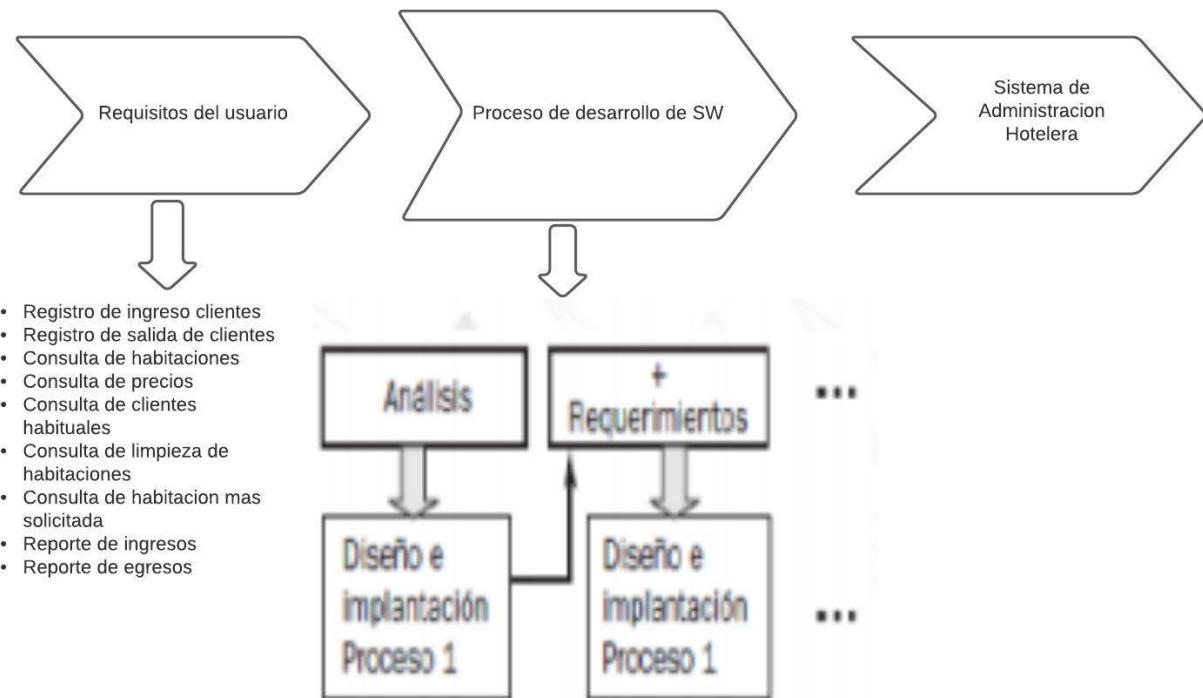


Ilustración 3: Diagrama de Casos de uso

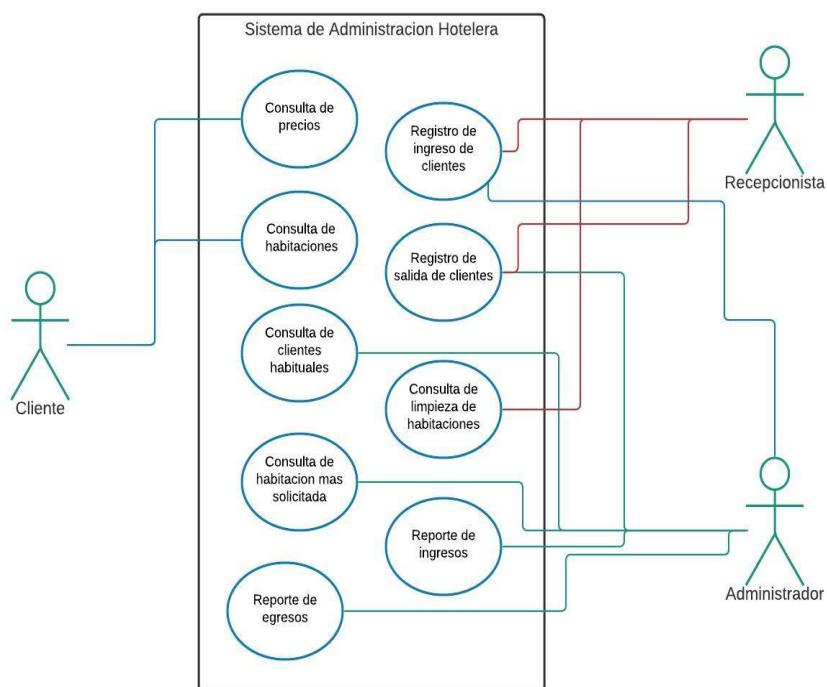


Ilustración 4: Prototipos

Acceso al sistema



Usuario:

Contraseña:

Trabajador

Registro de trabajadores

Código:	<input type="text"/>
Nombres:	<input type="text"/>
Apellido Paterno:	<input type="text"/>
Apellido Materno:	<input type="text"/>
Tipo de Documento:	<input type="text"/>
Número de Documento:	<input type="text"/>
Celular:	<input type="text"/>
Email:	<input type="text"/>
Sueldo:	<input type="text"/>
Acceso:	<input type="text"/>
Login:	<input type="text"/>
Password:	<input type="text"/>
Estado:	<input type="text"/>

Opciones

Detalles

Código	Nombres	Apellido Paterno	Apellido Materno	Tipo de Documento	Nº Documento	Celular	Email	Sueldo	Acceso	Login	Password	Estado
1	Pedro	Azabache	Perez	DNI	48189768	991642315	pedro@correo.com	1178	Recepcionista	pedroazob	azab481	1
2	Julio	Lopez	Martinez	DNI	32382559	992317456	julio@correo.com	2500	Administrador	juliolop	lop323	1
3	Mauricio	Quispe	Leon	DNI	46328748	912118543	mauricio@correo.co	1178	Recepcionista	mouquip	qui463	1
4	Hector	Sanchez	Luna	DNI	72834178	995434515	hector@correo.com	1178	Recepcionista	hectzan	san728	1

Clientes

Registro de clientes

Código:	<input type="text"/>
Nombres:	<input type="text"/>
Apellido Paterno:	<input type="text"/>
Apellido Materno:	<input type="text"/>
Tipo de Documento:	<input type="text"/>
Número de Documento:	<input type="text"/>
Celular:	<input type="text"/>
Email:	<input type="text"/>

Opciones

Detalles

Código	Nombres	Apellido Patern	Apellido Matern	Tipo de Document	Nº Documento	Celular	Email
1	Mario	Aguirre	Pereyra	DNI	48189768	991642315	mario@correo.com
2	Domingo	Oropeza	Lopez	DNI	32382559	992317456	domingo@correo.co
3	Jorge	Molina	Pereyra	DNI	46328748	912118543	jorge@correo.com
4	Pablo	Nuñez	Luna	DNI	72834178	995434515	pablo@correo.com

Reserva

Registro de reserva

Código de Reserva:	<input type="text"/>
Código de Cliente:	<input type="text"/>
Código de Trabajador:	<input type="text"/>
Código de Habitación:	<input type="text"/>
Tipo de Reserva:	<input type="text"/>
Fecha de Reserva:	<input type="text"/> / <input type="text"/> / <input type="button" value="Calendario"/>
Fecha de ingreso:	<input type="text"/> / <input type="text"/> / <input type="button" value="Calendario"/>
Fecha de salida:	<input type="text"/> / <input type="text"/> / <input type="button" value="Calendario"/>
Costo :	<input type="text"/>
Estado:	<input type="text"/>

Opciones

Detalles

Código reserva	Código cliente	Código trabajador	Código habitación	Tipo de reserva	Fecha de reserva	Fecha de ingreso	Fecha de salida	Costo	Estado
1	1	1	1	Reserva	30/09/2020	03/10/2020	04/10/2020	100	1
2	2	2	2	Reserva	30/09/2020	03/10/2020	05/10/2020	250	1
3	3	3	3	Reserva	30/09/2020	04/10/2020	05/10/2020	100	1
4	4	4	4	Reserva	30/09/2020	05/10/2020	06/10/2020	100	1
5	5	5	5	Alquiler	30/09/2020	05/10/2020	07/10/2020	250	1

Ilustración 5: Diagrama de actividades con particiones o calles

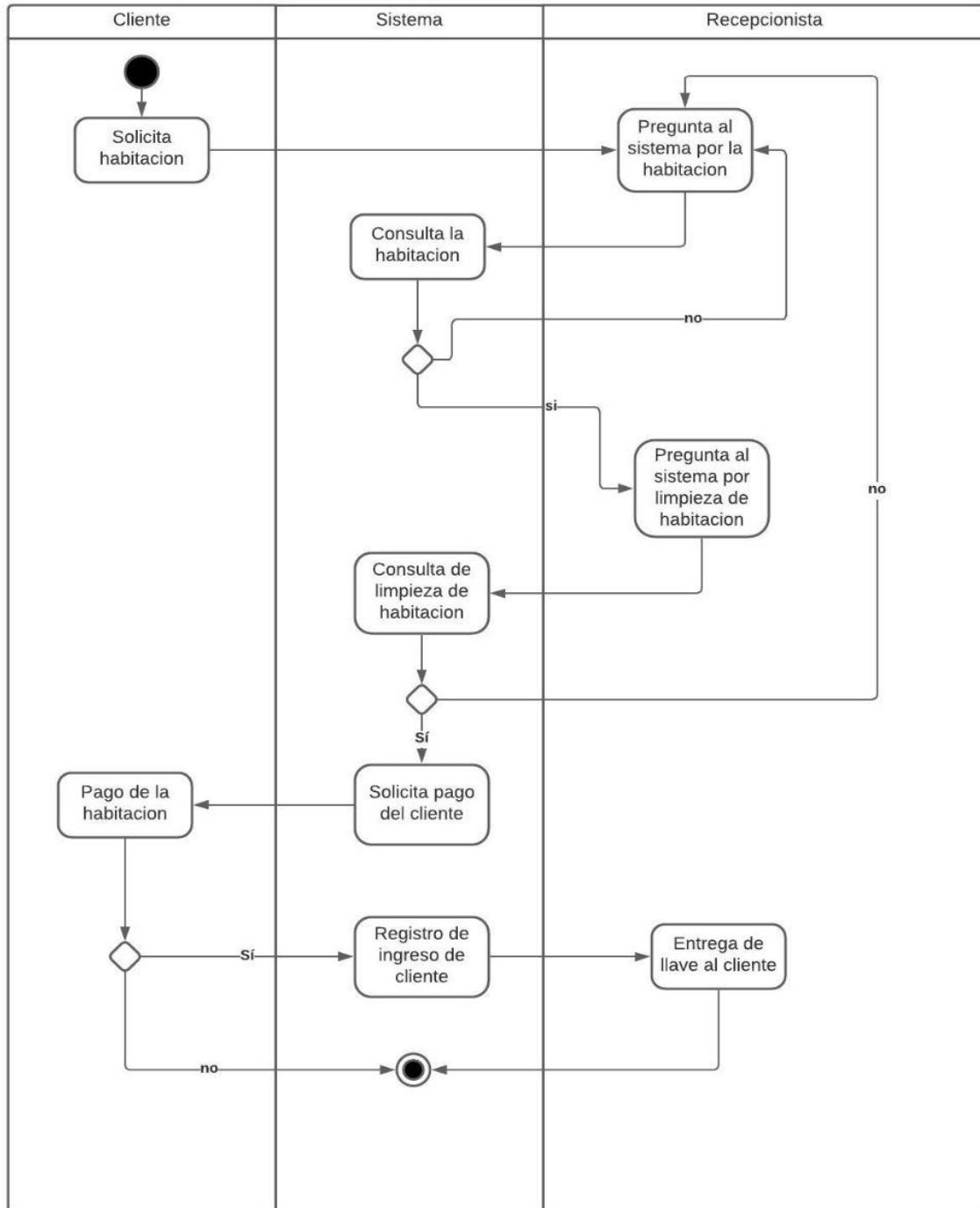


Tabla 2: Trazabilidad de Requisitos

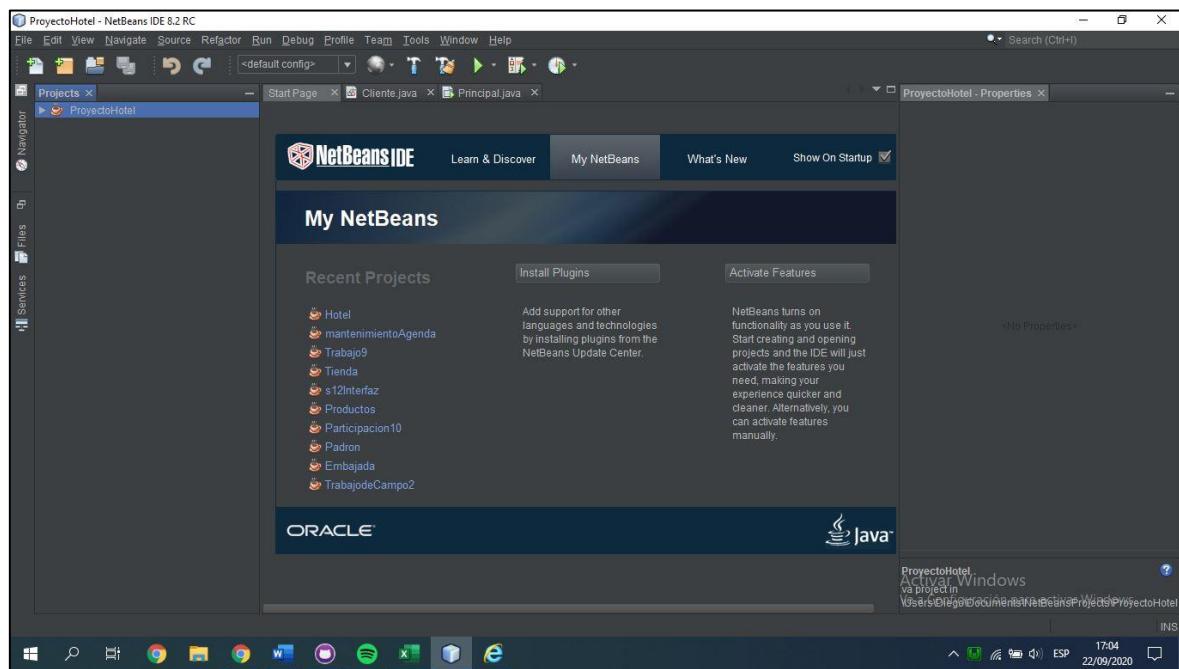
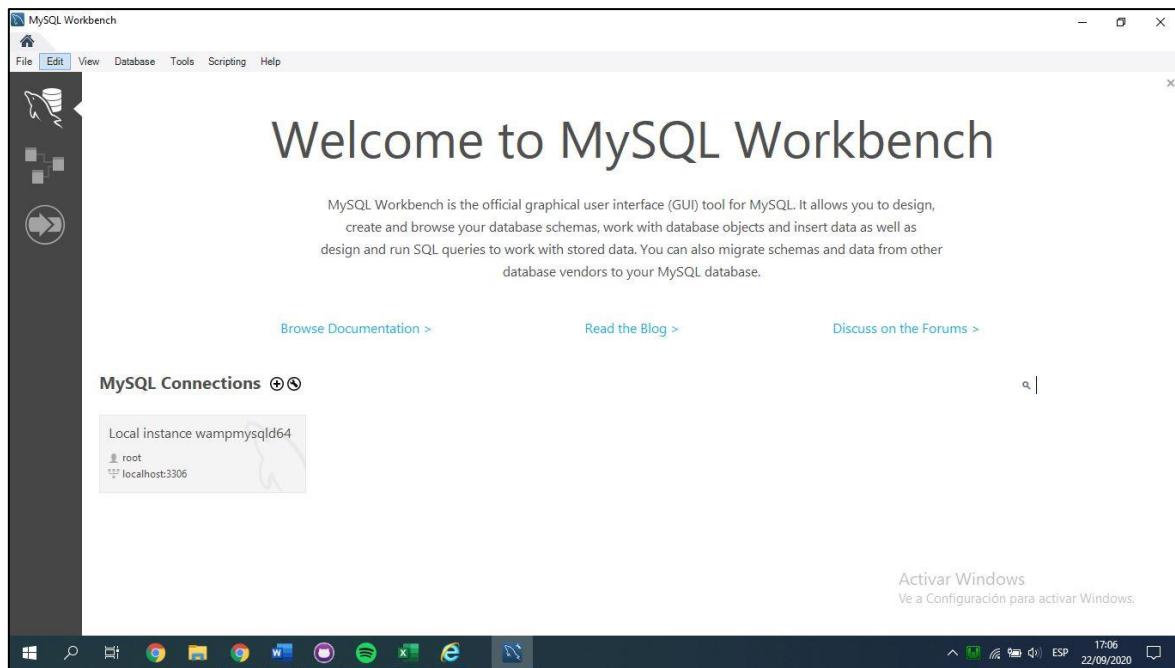
Tipo de Requisito	ID	Requisito	Casos de Uso	Prototipo	Versión	Instalado
Funcional	1	Permite el Registro de trabajadores	Registro de Trabajador	Si	1	/
Funcional	2	Permite hacer un listado de los trabajadores existentes	Consulta de Trabajadores	Si	1	/
Funcional	3	Permite Modificar a los trabajadores registrados	Modificar al Trabajador	Si	1	/
Funcional	4	Permite Eliminar trabajadores registrados	Eliminar Trabajador	Si	1	/
Funcional	5	Verificar el usuario y acceder al sistema	Validar trabajador	Si	1	/
Funcional	6	Permite el Registro de clientes	Registro de Clientes	Si	1	/
Funcional	7	Permite hacer un listado de los clientes existentes	Consulta de Clientes	Si	1	/
Funcional	8	Permite Modificar a los clientes registrados	Modificar al Cliente	Si	1	/
Funcional	9	Permite Eliminar clientes registrados	Eliminar Cliente	Si	1	/
Funcional	10	Permite registrar la fecha de entrada del cliente.	Registro de entrada del cliente	Si	1	/
Funcional	11	Permite registrar la fecha de salida del cliente	Registro de salida del cliente	Si	1	/
Funcional	12	Permite mostrar la fecha de entrada del cliente	Consulta de entrada de los clientes	Si	1	/
Funcional	13	Permite mostrar la fecha de salida del cliente	Consulta de la salida de los clientes	Si	1	/
Funcional	14	Permite modificar la fecha de entrada del cliente	Modificar entrada del cliente	Si	1	/

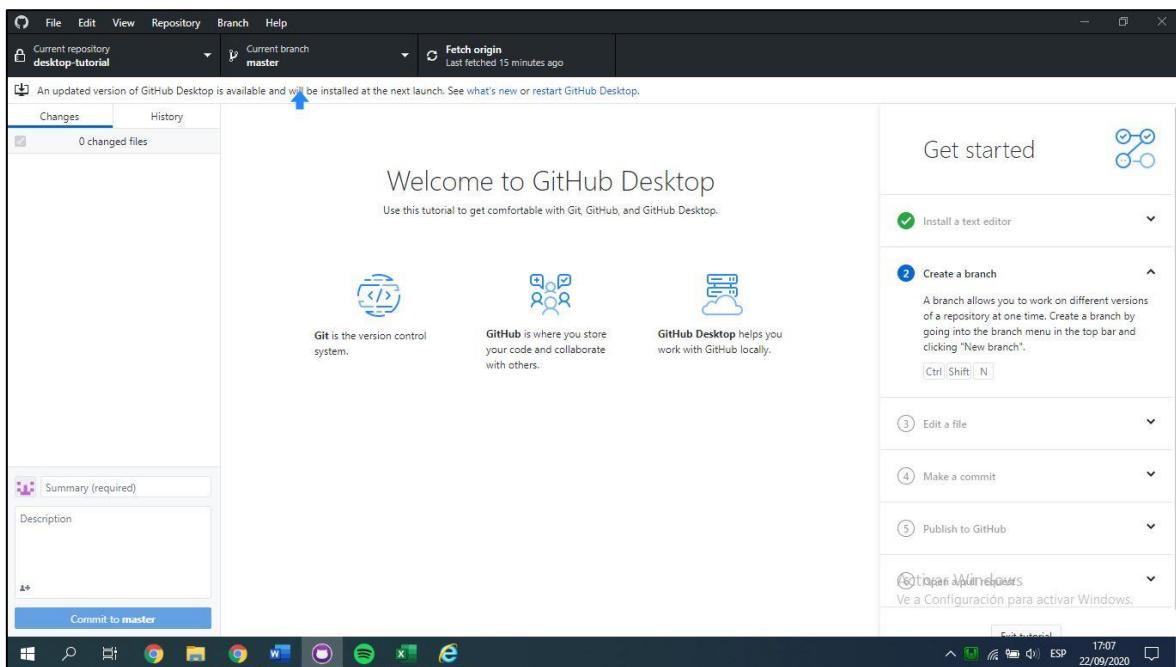


Funcional	15	Permite modificar la fecha de salida del cliente	Modificar salida del cliente	Si	1	/
Funcional	16	Permite eliminar la fecha de entrada del cliente	Eliminar la entrada del cliente	Si	1	/
Funcional	17	Permite eliminar la fecha de salida del cliente	Eliminar la salida del cliente	Si	1	/
Funcional	18	Saber el estado de las habitaciones	Consulta de habitaciones	No	2	/
Funcional	19	Saber los clientes habituales	Consulta de clientes habituales	No	3	/
No funcional	20	NetBeans	/	No	/	Si
No funcional	21	MySQL o WampServer	/	No	/	Si
No funcional	22	MySQL WorkBench	/	No	/	Si

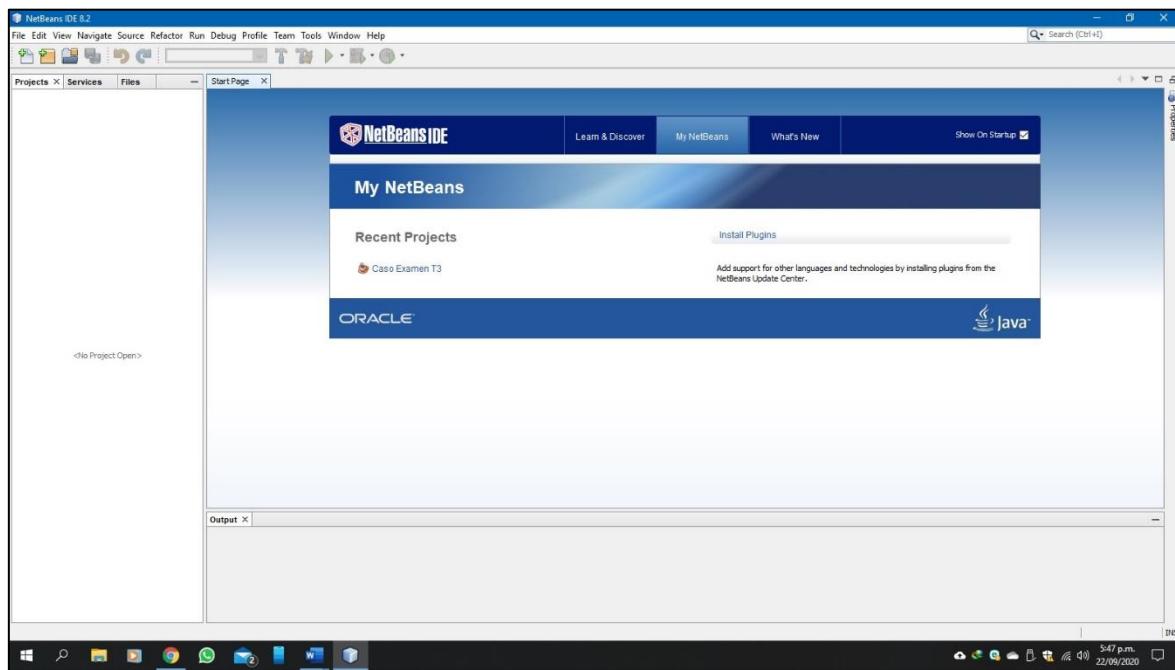
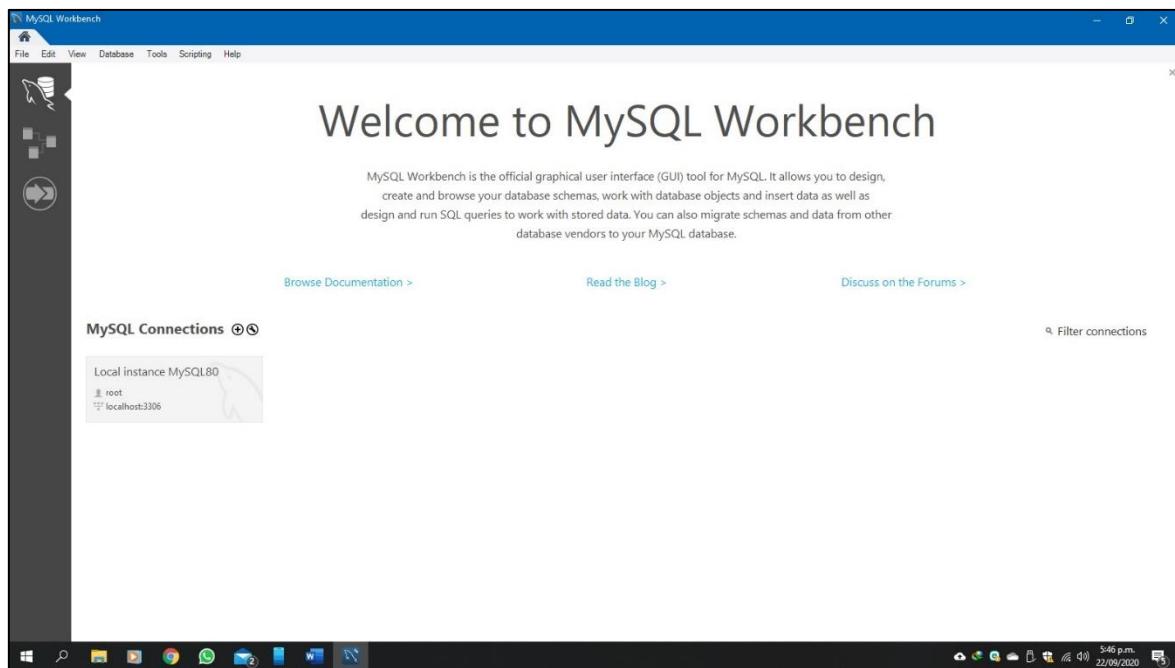
Ilustración 6: Requisitos no funcionales

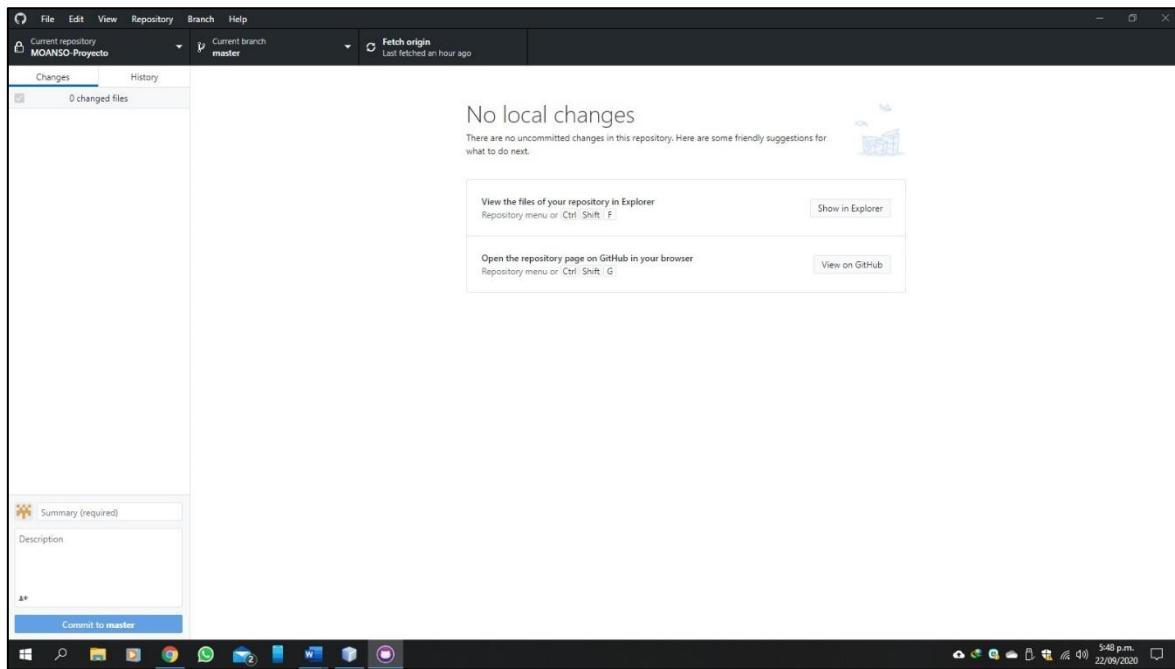
Computadora Diego Tasaico



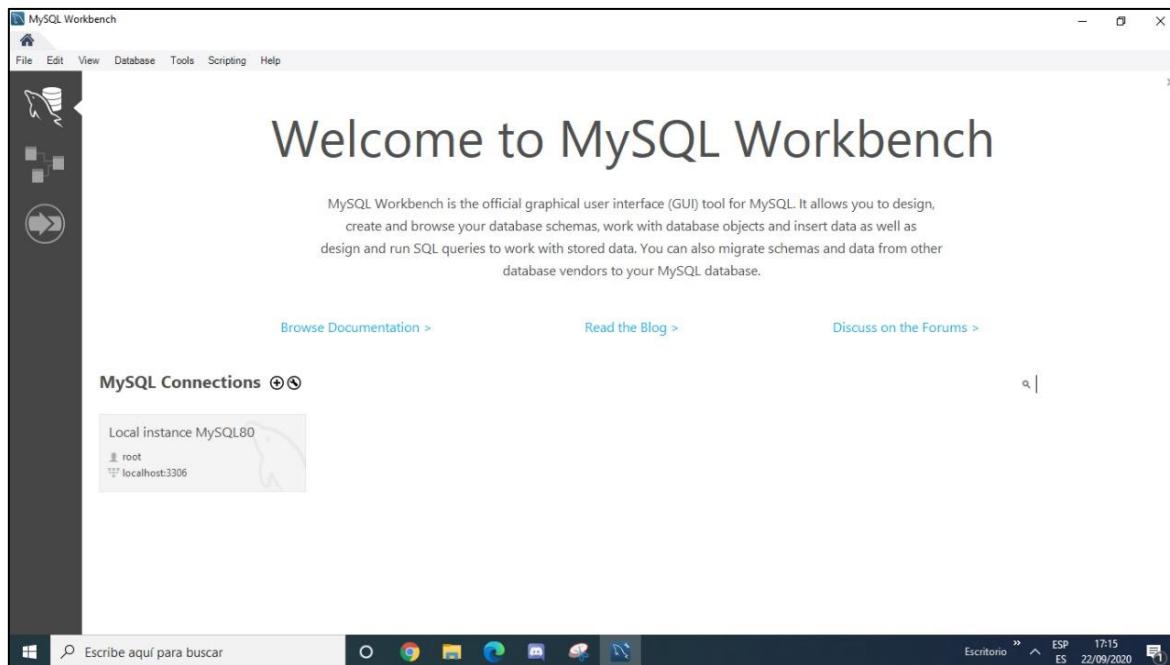


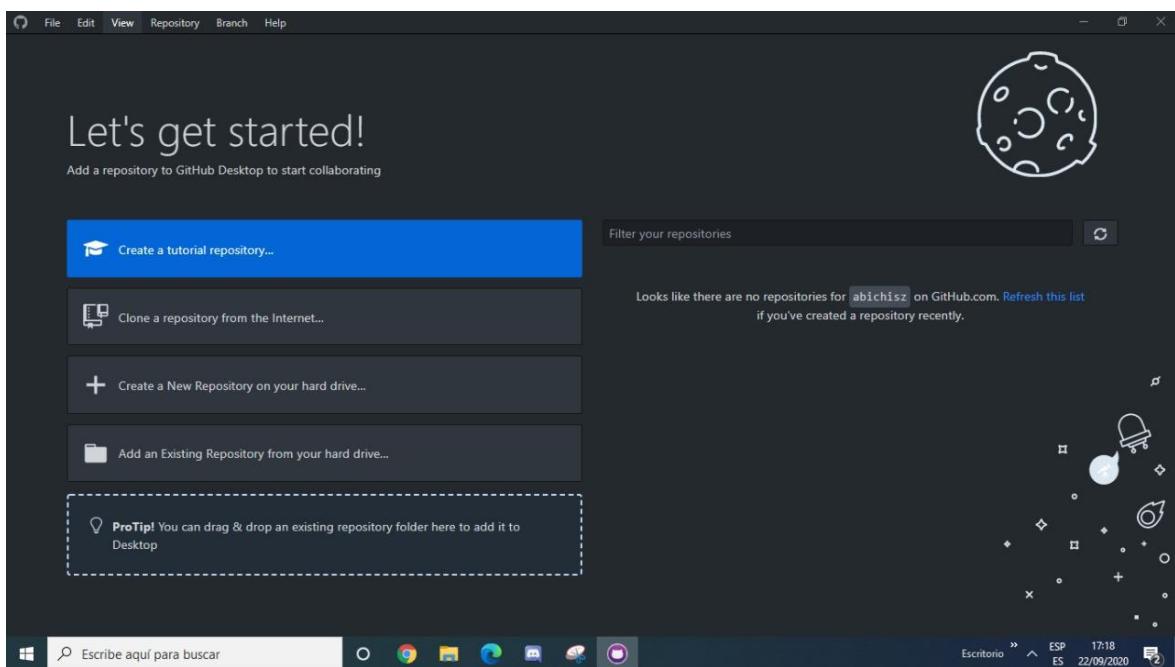
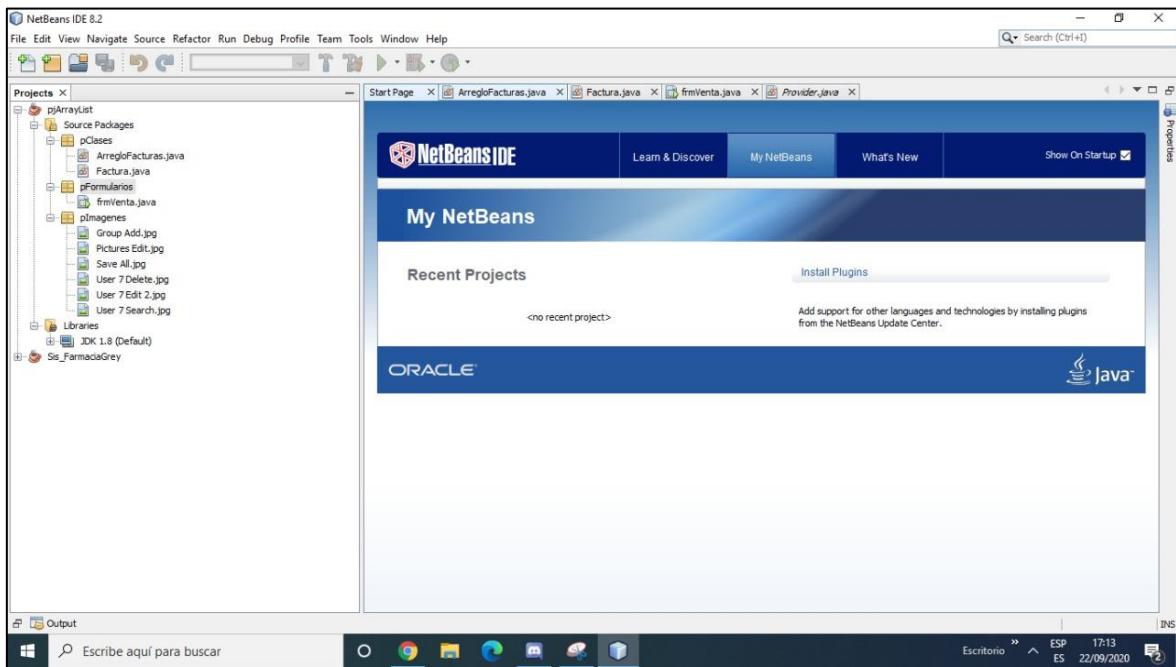
Computadora Mauricio Villanueva

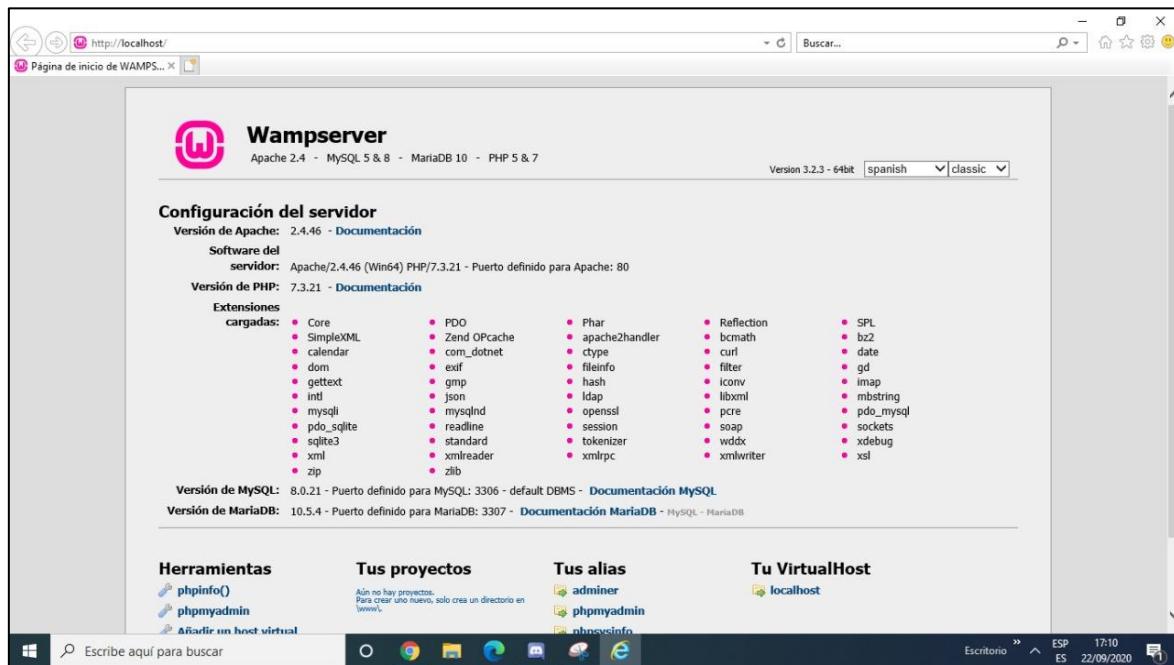




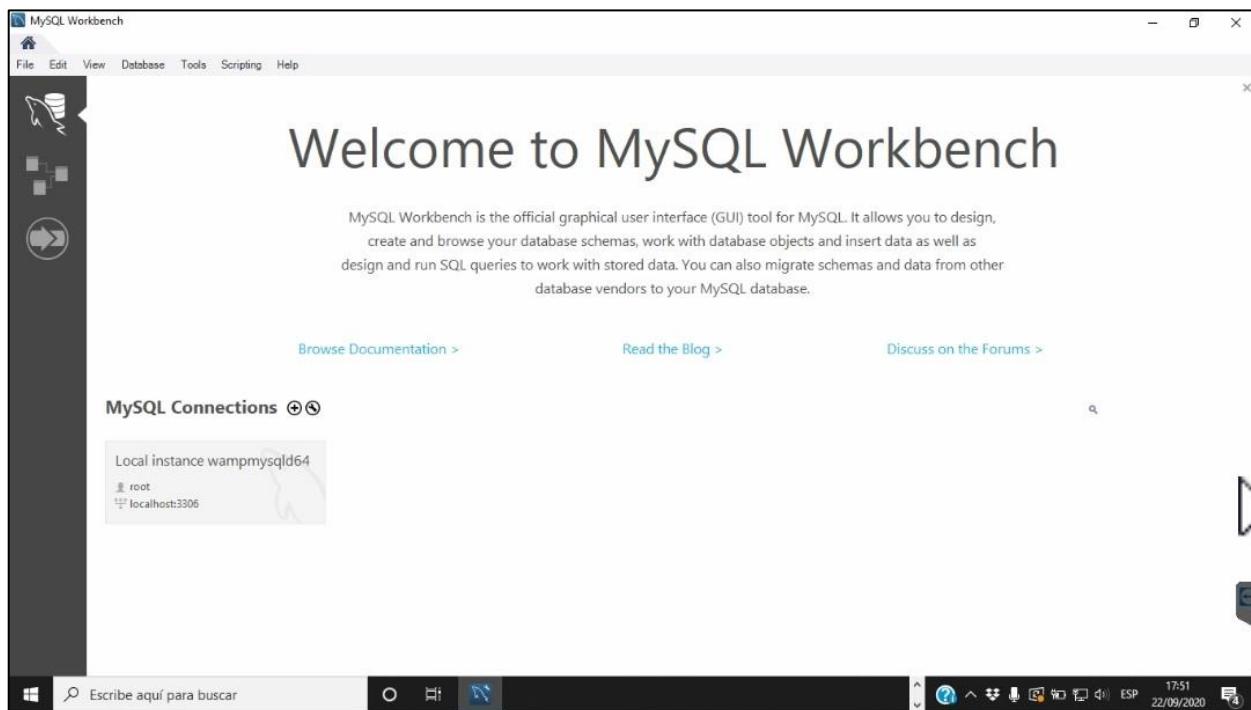
Computadora Abihail Zapata

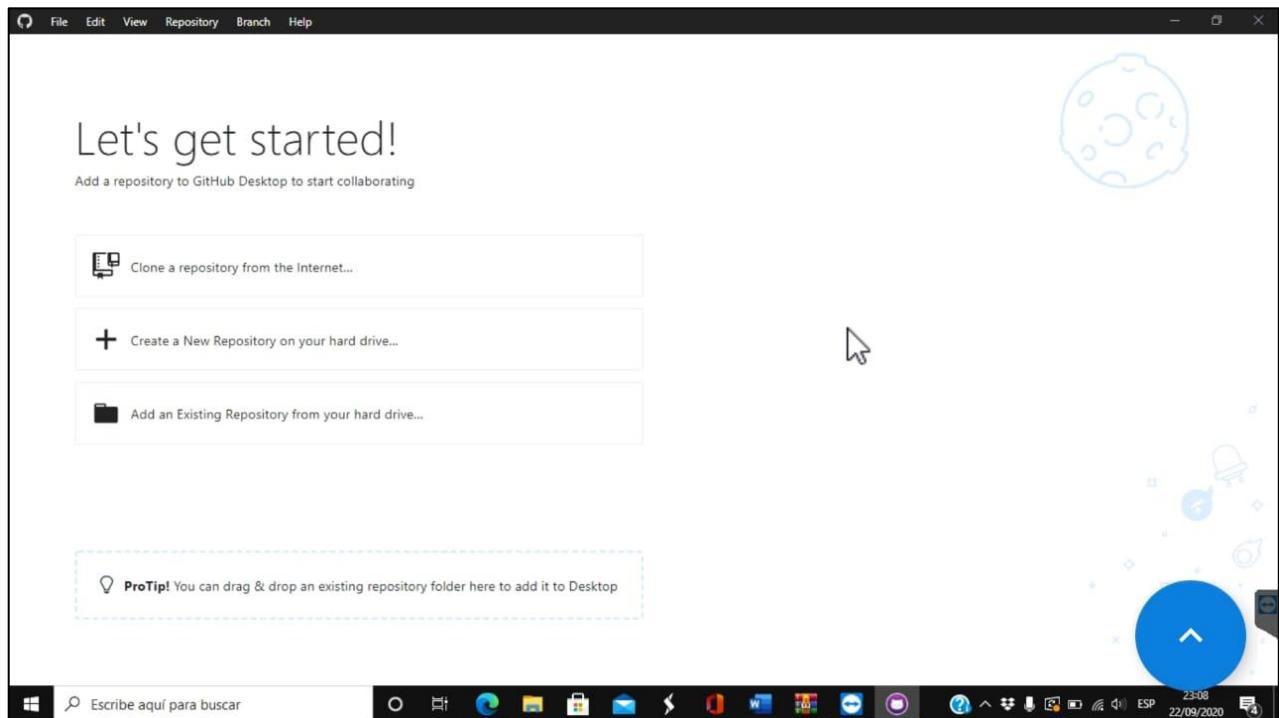
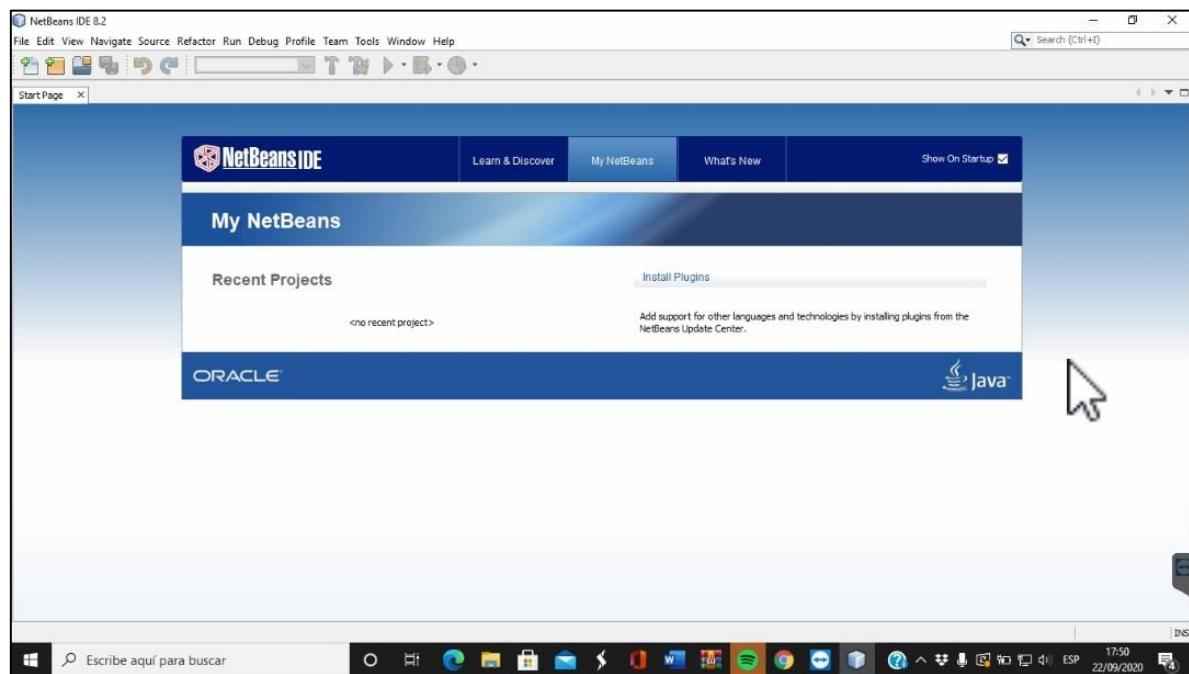






Computadora Olenka Lazo





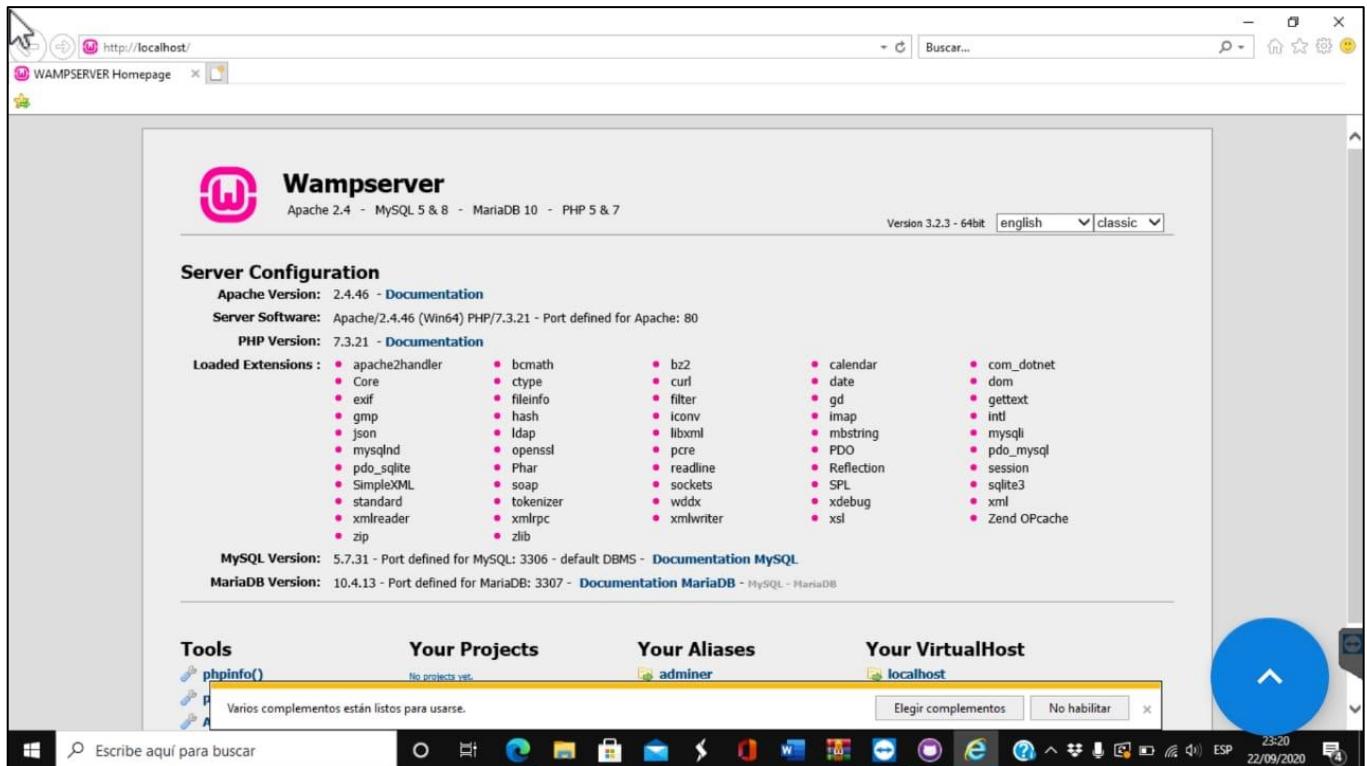
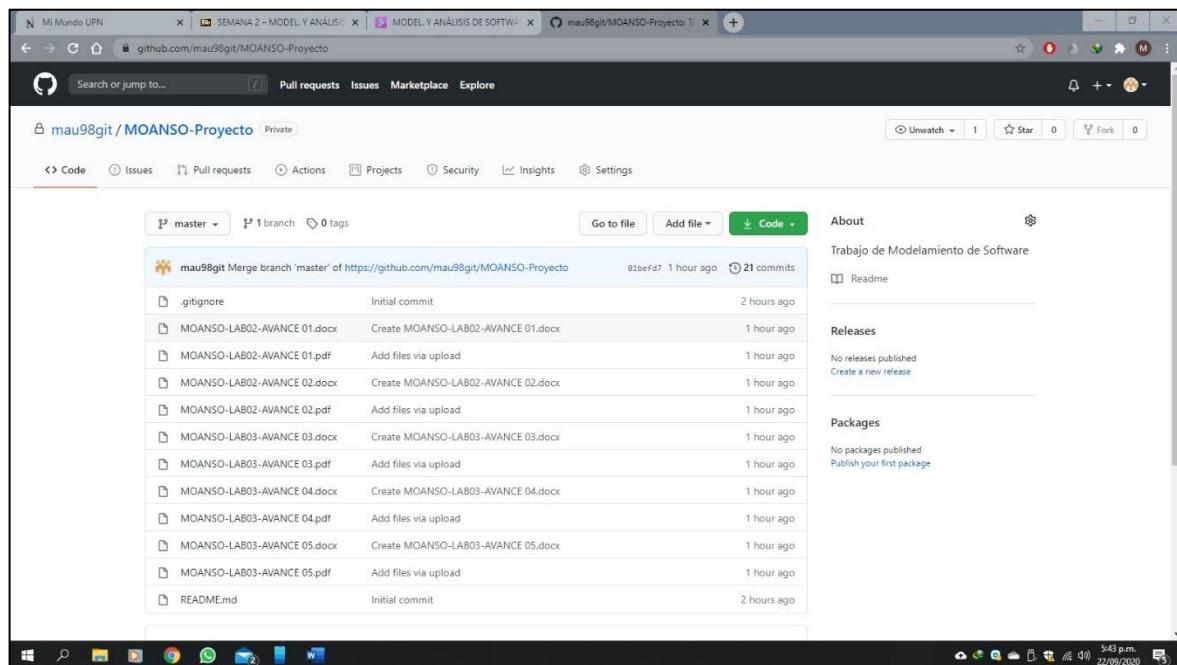


Ilustración 7: Repositorio GitHub

Repositorio GitHub



Script de MySQL

```
-- MySQL Script generated by MySQL Workbench
```

```
-- Wed Sep 30 01:12:07 2020
```

```
-- Model: New Model  Version: 1.0
```

```
-- MySQL Workbench Forward Engineering
```

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
```

```
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,  
FOREIGN_KEY_CHECKS=0;
```

```
SET @OLD_SQL_MODE=@@SQL_MODE,  
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZE  
RO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';
```

```
-----
```

```
-- Schema Proyecto
```

```
-----
```

```
-----
```



```
-- Schema Proyecto
```



```
-----
```

```
CREATE SCHEMA IF NOT EXISTS `Proyecto` DEFAULT CHARACTER SET utf8 ;
```

```
USE `Proyecto` ;
```

```
-----
```

```
-- Table `Proyecto`.`Habitacion`
```

```
-----
```

```
CREATE TABLE IF NOT EXISTS `Proyecto`.`Habitacion` (
```

```
 `idHabitacion` INT NOT NULL AUTO_INCREMENT,
```

```
 `numero` VARCHAR(3) NOT NULL,
```

```
 `piso` VARCHAR(1) NOT NULL,
```

```
`precio` DECIMAL NOT NULL,  
`estado` VARCHAR(25) NOT NULL,  
`tipo_habitacion` VARCHAR(20) NOT NULL,  
PRIMARY KEY (`idHabitacion`))  
ENGINE = InnoDB;
```

```
-- Table `Proyecto`.`Trabajador`
```

```
CREATE TABLE IF NOT EXISTS `Proyecto`.`Trabajador` (  
`idTrabajador` INT NOT NULL AUTO_INCREMENT,  
`nombre` VARCHAR(45) NOT NULL,  
`apellido_paterno` VARCHAR(45) NOT NULL,  
`apellido_materno` VARCHAR(45) NOT NULL,  
`tipo_documento` VARCHAR(45) NOT NULL,  
`num_documento` VARCHAR(45) NOT NULL,  
`celular` VARCHAR(45) NOT NULL,  
`email` VARCHAR(45) NOT NULL,  
`sueldo` DECIMAL NOT NULL,  
`acceso` VARCHAR(20) NOT NULL,  
`login` VARCHAR(20) NOT NULL,  
`password` VARCHAR(20) NOT NULL,  
`estado` VARCHAR(1) NOT NULL,  
UNIQUE INDEX `login_UNIQUE` (`login` ASC) VISIBLE,  
PRIMARY KEY (`idTrabajador`))  
ENGINE = InnoDB;
```

-- Table `Proyecto`.`Cliente`

```
CREATE TABLE IF NOT EXISTS `Proyecto`.`Cliente` (
```

```
    `idCliente` INT NOT NULL AUTO_INCREMENT,  
    `nombres` VARCHAR(45) NOT NULL,  
    `apellido_paterno` VARCHAR(45) NOT NULL,  
    `apellido_materno` VARCHAR(45) NOT NULL,  
    `tipo_documento` VARCHAR(45) NOT NULL,  
    `num_documento` VARCHAR(45) NOT NULL,  
    `celular` VARCHAR(45) NOT NULL,  
    `email` VARCHAR(45) NULL,  
    PRIMARY KEY (`idCliente`))
```

```
ENGINE = InnoDB;
```

-- Table `Proyecto`.`Reserva`

```
CREATE TABLE IF NOT EXISTS `Proyecto`.`Reserva` (
```

```
    `idReserva` INT NOT NULL AUTO_INCREMENT,  
    `idCliente` INT NOT NULL,  
    `idTrabajador` INT NOT NULL,  
    `idHabitacion` INT NOT NULL,  
    `tipo_reserva` VARCHAR(20) NOT NULL,  
    `fecha_reserva` DATE NOT NULL,  
    `fecha_ingreso` DATE NOT NULL,  
    `fecha_salida` DATE NOT NULL,  
    `costo_alojamiento` DECIMAL NOT NULL,
```

```
`estado` VARCHAR(15) NOT NULL,  
PRIMARY KEY (`idReserva`),  
UNIQUE INDEX `tipo_reserva_UNIQUE` (`tipo_reserva` ASC) VISIBLE,  
INDEX `fk_reserva_habitacion_idx` (`idHabitacion` ASC) VISIBLE,  
INDEX `fk_reserva_cliente_idx` (`idCliente` ASC) VISIBLE,  
INDEX `fk_reserva_trabajador_idx` (`idTrabajador` ASC) VISIBLE,  
CONSTRAINT `fk_reserva_habitacion`  
FOREIGN KEY (`idHabitacion`)  
REFERENCES `Proyecto`.`Habitacion`(`idHabitacion`)  
ON DELETE NO ACTION  
ON UPDATE NO ACTION,  
CONSTRAINT `fk_reserva_cliente`  
FOREIGN KEY (`idCliente`)  
REFERENCES `Proyecto`.`Cliente`(`idCliente`)  
ON DELETE NO ACTION  
ON UPDATE NO ACTION,  
CONSTRAINT `fk_reserva_trabajador`  
FOREIGN KEY (`idTrabajador`)  
REFERENCES `Proyecto`.`Trabajador`(`idTrabajador`)  
ON DELETE NO ACTION  
ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

```
-- Table `Proyecto`.`Pago`  
  
-----  
CREATE TABLE IF NOT EXISTS `Proyecto`.`Pago` (  
`idPago` INT NOT NULL AUTO_INCREMENT,
```

```
`idReserva` INT NOT NULL,  
 `tipo_comprobante` VARCHAR(20) NOT NULL,  
 `num_comprobante` VARCHAR(20) NOT NULL,  
 `igv` DECIMAL NOT NULL,  
 `total_pago` DECIMAL NOT NULL,  
 `fecha_emision` DATE NOT NULL,  
 `fecha_pago` DATE NOT NULL,  
 PRIMARY KEY (`idPago`),  
 INDEX `fk_pago_reserva_idx` (`idReserva` ASC) VISIBLE,  
 CONSTRAINT `fk_pago_reserva`  
 FOREIGN KEY (`idReserva`)  
 REFERENCES `Proyecto`.`Reserva` (`idReserva`)  
 ON DELETE NO ACTION  
 ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

```
SET SQL_MODE=@OLD_SQL_MODE;  
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;  
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Ilustración 8: Diagrama de Casos de Uso Relacionado

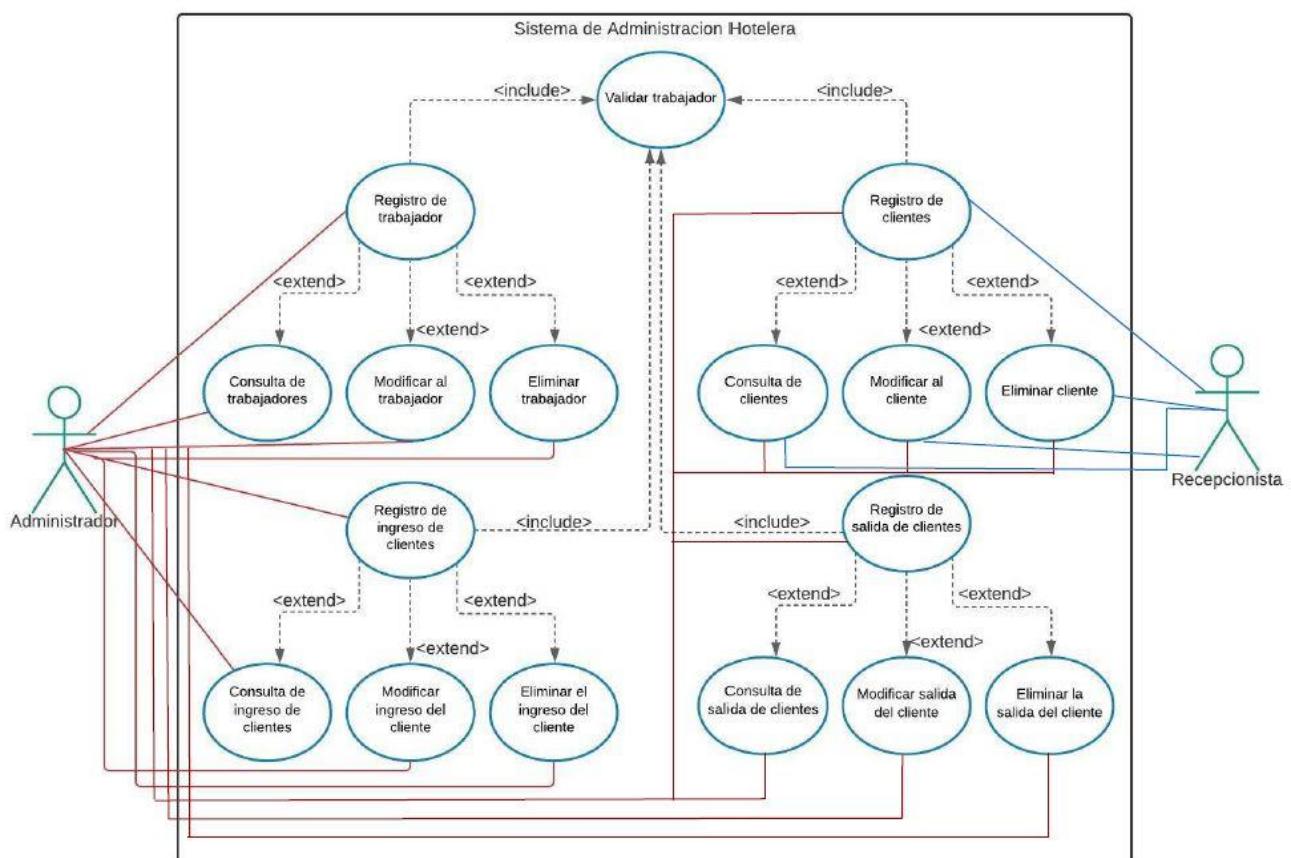
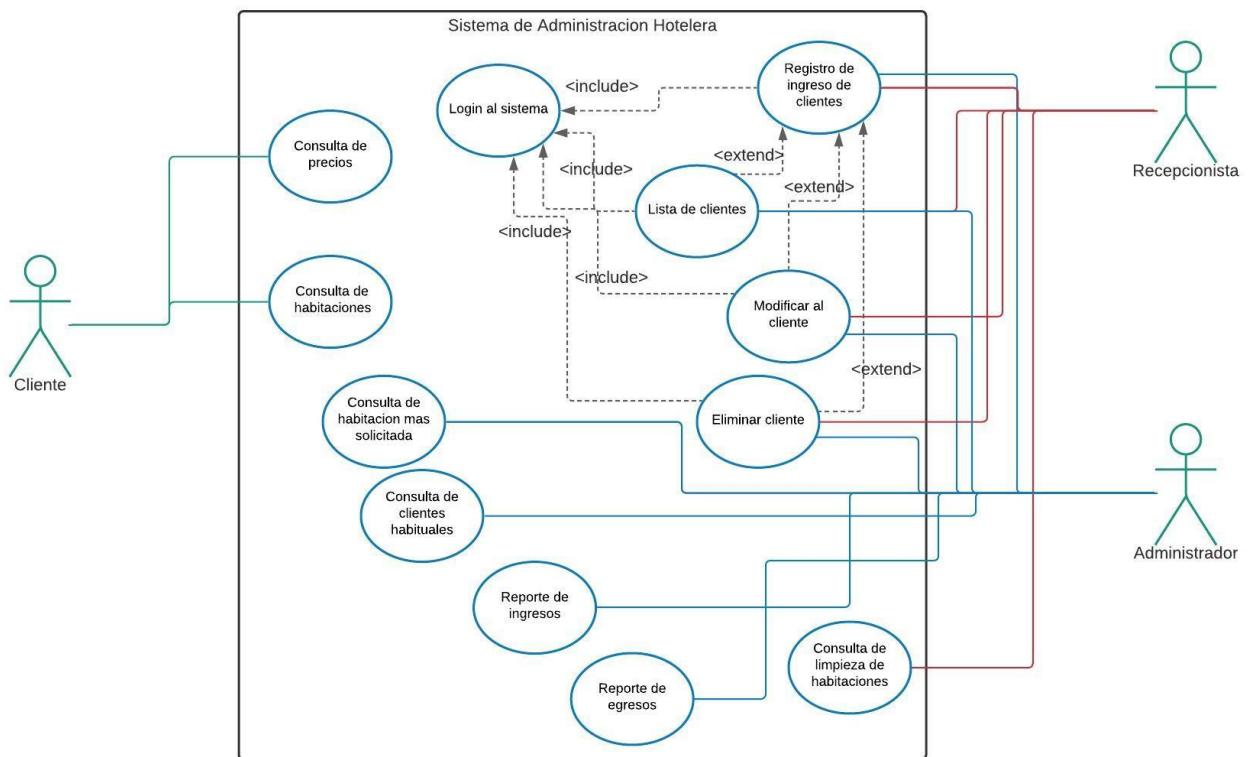
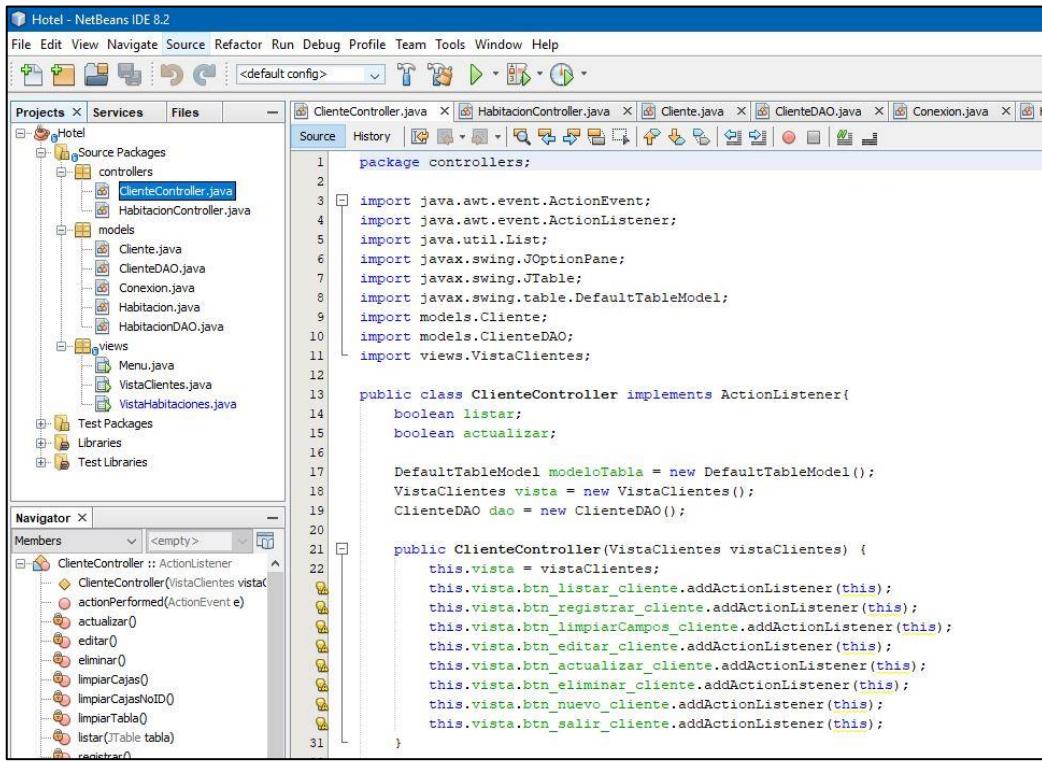


Ilustración 9: Funcionamiento del Software



```

package controllers;

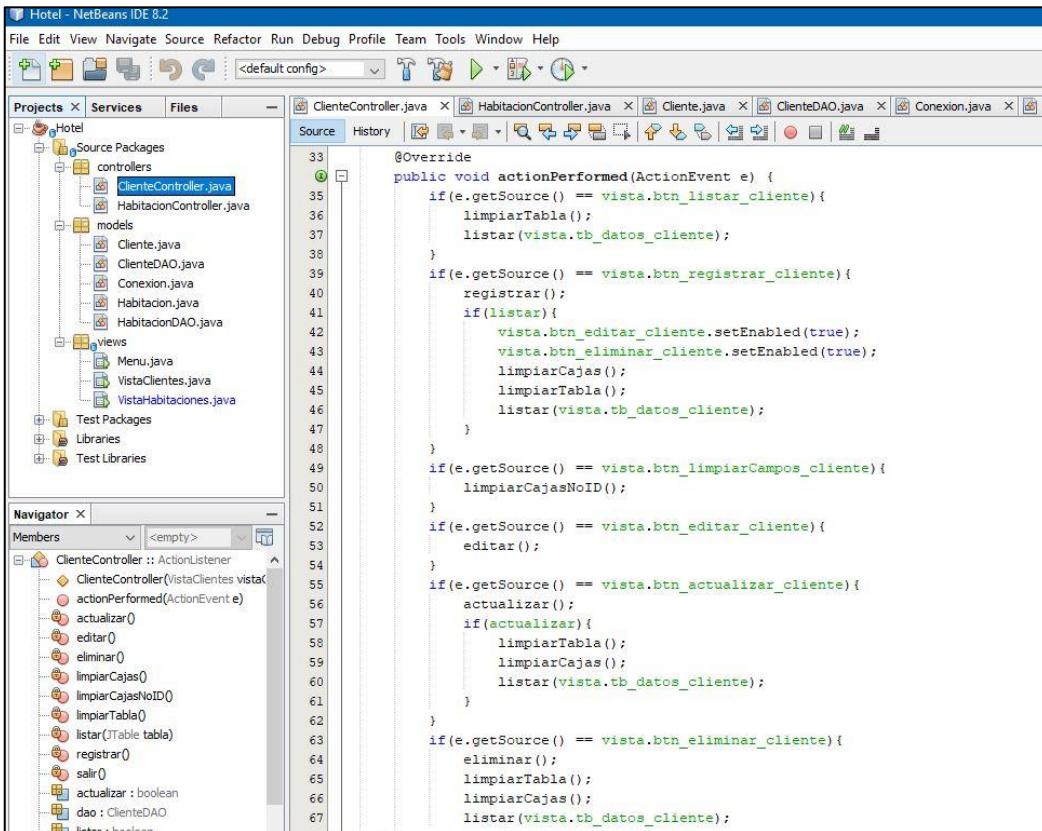
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.List;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;
import models.Cliente;
import models.ClienteDAO;
import views.VistaClientes;

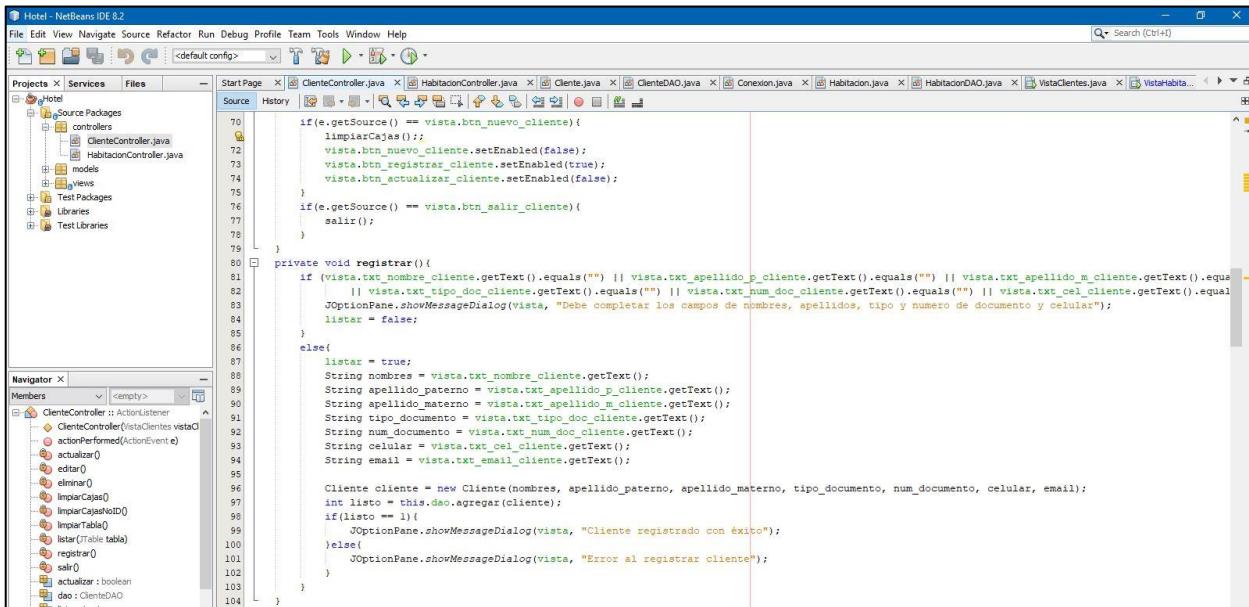
public class ClienteController implements ActionListener{
    boolean listar;
    boolean actualizar;
    DefaultTableModel modeloTabla = new DefaultTableModel();
    VistaClientes vista = new VistaClientes();
    ClienteDAO dao = new ClienteDAO();

    public ClienteController(VistaClientes vistaClientes) {
        this.vista = vistaClientes;
        this.vista.btn_listar_cliente.addActionListener(this);
        this.vista.btn_registrar_cliente.addActionListener(this);
        this.vista.btn_limpiarCampos_cliente.addActionListener(this);
        this.vista.btn_editar_cliente.addActionListener(this);
        this.vista.btn_actualizar_cliente.addActionListener(this);
        this.vista.btn_eliminar_cliente.addActionListener(this);
        this.vista.btn_nuevo_cliente.addActionListener(this);
        this.vista.btn_salir_cliente.addActionListener(this);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if(e.getSource() == vista.btn_listar_cliente){
            limpiarTabla();
            listar(vista.tb_datos_cliente);
        }
        if(e.getSource() == vista.btn_registrar_cliente){
            registrar();
            if(listar){
                vista.btn_editar_cliente.setEnabled(true);
                vista.btn_eliminar_cliente.setEnabled(true);
                limpiarCajas();
                limpiarTabla();
                listar(vista.tb_datos_cliente);
            }
        }
        if(e.getSource() == vista.btn_limpiarCampos_cliente){
            limpiarCajasNoID();
        }
        if(e.getSource() == vista.btn_editar_cliente){
            editar();
        }
        if(e.getSource() == vista.btn_actualizar_cliente){
            actualizar();
            if(actualizar){
                limpiarTabla();
                limpiarCajas();
                listar(vista.tb_datos_cliente);
            }
        }
        if(e.getSource() == vista.btn_eliminar_cliente){
            eliminar();
            limpiarTabla();
            limpiarCajas();
            listar(vista.tb_datos_cliente);
        }
    }
}

```

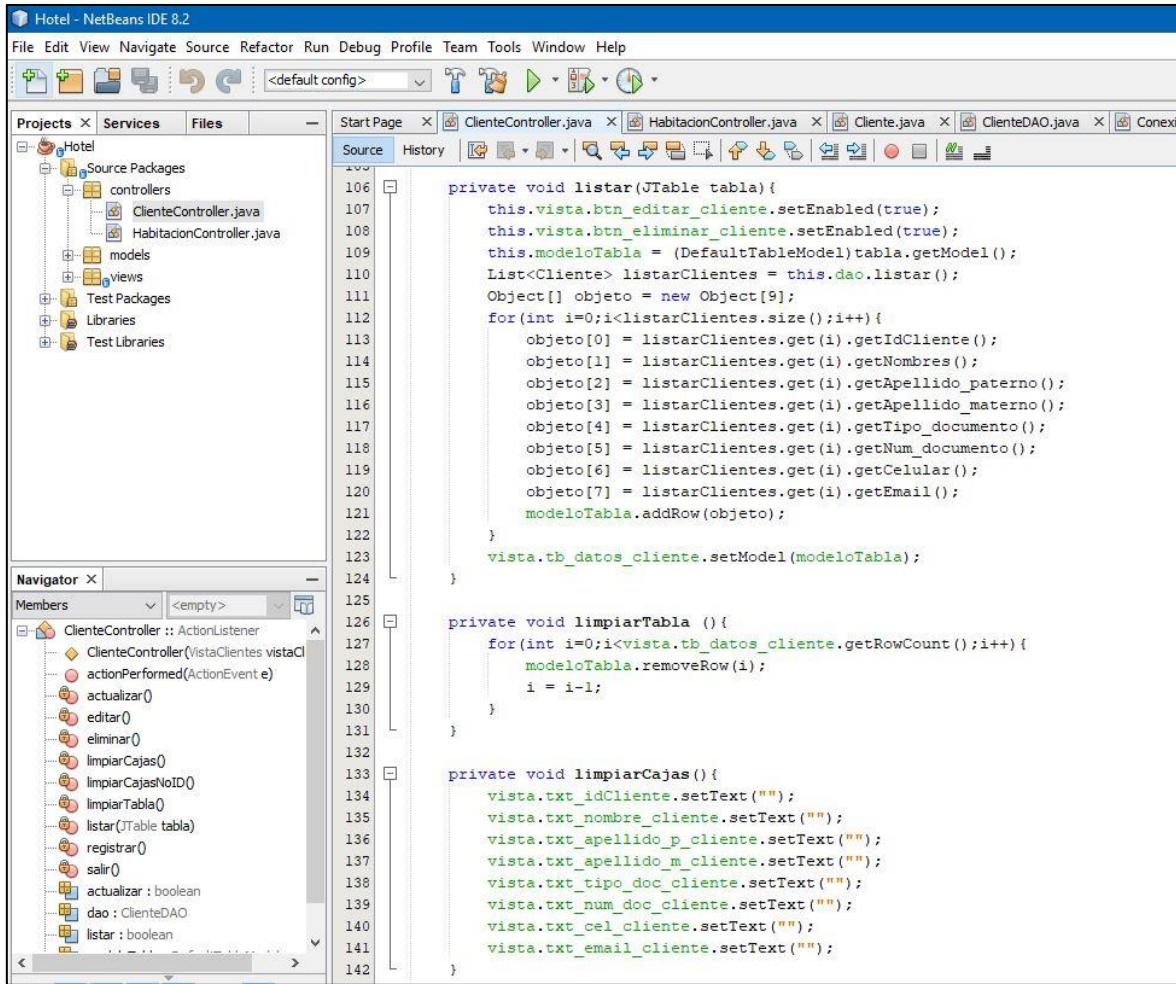




```

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects Services Files Start Page ClienteController.java HabitacionController.java Cliente.java ClienteDAO.java Conexion.java Habitacion.java HabitacionDAO.java VistaClientes.java VistaHabitacion.java
Source History <default config> T T D G S
Projects Services Files Start Page ClienteController.java HabitacionController.java Cliente.java ClienteDAO.java Conexion.java Habitacion.java HabitacionDAO.java VistaClientes.java VistaHabitacion.java
Source History <default config> T T D G S
1 if(e.getSource() == vista.btn_nuevo_cliente){
2     limpiarCajas();
3     vista.btn_nuevo_cliente.setEnabled(false);
4     vista.btn_registrar_cliente.setEnabled(true);
5     vista.btn_actualizar_cliente.setEnabled(false);
6 }
7 if(e.getSource() == vista.btn_salir_cliente){
8     salir();
9 }
10
11 private void registrar(){
12     if (vista.txt_nombre_cliente.getText().equals("") || vista.txt_apellido_p_cliente.getText().equals("") || vista.txt_apellido_m_cliente.getText().equals("") || vista.txt_num_doc_cliente.getText().equals("") || vista.txt_cel_cliente.getText().equals("")){
13         JOptionPane.showMessageDialog(vista, "Debe completar los campos de nombres, apellidos, tipo y numero de documento y celular");
14         lista = false;
15     }
16     else{
17         lista = true;
18         String nombres = Vista.txt_nombre_cliente.getText();
19         String apellido_paterno = Vista.txt_apellido_p_cliente.getText();
20         String apellido_materno = Vista.txt_apellido_m_cliente.getText();
21         String tipo_documento = Vista.txt_tipo_doc_cliente.getText();
22         String num_documento = Vista.txt_num_doc_cliente.getText();
23         String celular = Vista.txt_cel_cliente.getText();
24         String email = Vista.txt_email_cliente.getText();
25
26         Cliente cliente = new Cliente(nombres, apellido_paterno, apellido_materno, tipo_documento, num_documento, celular, email);
27         int lista = this.dao.agregar(cliente);
28         if(lista == 1){
29             JOptionPane.showMessageDialog(vista, "Cliente registrado con éxito");
30         }
31         else{
32             JOptionPane.showMessageDialog(vista, "Error al registrar cliente");
33         }
34     }
35 }
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104

```



```

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects Services Files Start Page ClienteController.java HabitacionController.java Cliente.java ClienteDAO.java Conexion.java
Source History <default config> T T D G S
Projects Services Files Start Page ClienteController.java HabitacionController.java Cliente.java ClienteDAO.java Conexion.java
Source History <default config> T T D G S
106 private void listar(JTable tabla){
107     this.vista.btn_editar_cliente.setEnabled(true);
108     this.vista.btn_eliminar_cliente.setEnabled(true);
109     this.modeloTabla = (DefaultTableModel)tabla.getModel();
110     List<Cliente> listarClientes = this.dao.listar();
111     Object[] objeto = new Object[9];
112     for(int i=0;i<listarClientes.size();i++){
113         objeto[0] = listarClientes.get(i).getIdCliente();
114         objeto[1] = listarClientes.get(i).getNombres();
115         objeto[2] = listarClientes.get(i).getApellido_paterno();
116         objeto[3] = listarClientes.get(i).getApellido_materno();
117         objeto[4] = listarClientes.get(i).getTipo_documento();
118         objeto[5] = listarClientes.get(i).getNum_documento();
119         objeto[6] = listarClientes.get(i).getCelular();
120         objeto[7] = listarClientes.get(i).getEmail();
121         modeloTabla.addRow(objeto);
122     }
123     vista.tb_datos_cliente.setModel(modeloTabla);
124 }
125
126 private void limpiarTabla (){
127     for(int i=0;i<vista.tb_datos_cliente.getRowCount();i++){
128         modeloTabla.removeRow(i);
129         i = i-1;
130     }
131 }
132
133 private void limpiarCajas(){
134     vista.txt_idCliente.setText("");
135     vista.txt_nombre_cliente.setText("");
136     vista.txt_apellido_p_cliente.setText("");
137     vista.txt_apellido_m_cliente.setText("");
138     vista.txt_tipo_doc_cliente.setText("");
139     vista.txt_num_doc_cliente.setText("");
140     vista.txt_cel_cliente.setText("");
141     vista.txt_email_cliente.setText("");
142 }

```

Hotel - NetBeans IDE 8.2

```

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config> Source History
Projects Services Files Start Page ClienteController.java HabitacionController.java Cliente.java ClienteDAO.java Conexion.java Habitacion.java HabitacionDAO.java
Source Packages controllers ClientController.java HabitacionController.java models views Test Packages Libraries Test Libraries
144     private void limpiarCajasNoID(){
145         vista.txt_nombre_cliente.setText("");
146         vista.txt_apellido_p_cliente.setText("");
147         vista.txt_apellido_m_cliente.setText("");
148         vista.txt_tipo_doc_cliente.setText("");
149         vista.txt_num_doc_cliente.setText("");
150         vista.txt_cel_cliente.setText("");
151         vista.txt_email_cliente.setText("");
152     }
153
154     private void editar(){
155         int fila = vista.tb_datos_cliente.getSelectedRow();
156         if(fila == -1){
157             JOptionPane.showMessageDialog(vista, "Debe seleccionar una fila");
158         }else{
159             this.vista.btn_registrar_cliente.setEnabled(false);
160             this.vista.btn_actualizar_cliente.setEnabled(true);
161             this.vista.btn_nuevo_cliente.setEnabled(true);
162             int codigo = Integer.parseInt(String.valueOf(vista.tb_datos_cliente.getValueAt(fila, 0).toString()));
163             String nombres = (String)vista.tb_datos_cliente.getValueAt(fila, 1);
164             String apellido_paterno = (String)vista.tb_datos_cliente.getValueAt(fila, 2);
165             String apellido_materno = (String)vista.tb_datos_cliente.getValueAt(fila, 3);
166             String tipo_Documento = (String)vista.tb_datos_cliente.getValueAt(fila, 4);
167             String num_documento = (String)vista.tb_datos_cliente.getValueAt(fila, 5);
168             String celular = (String)vista.tb_datos_cliente.getValueAt(fila, 6);
169             String email = (String)vista.tb_datos_cliente.getValueAt(fila, 7);
170
171             vista.txt_idCliente.setText(codigo+"");
172             vista.txt_nombre_cliente.setText(nombres);
173             vista.txt_apellido_p_cliente.setText(apellido_paterno);
174             vista.txt_apellido_m_cliente.setText(apellido_materno);
175             vista.txt_tipo_doc_cliente.setText(tipo_Documento);
176             vista.txt_num_doc_cliente.setText(num_documento);
177             vista.txt_cel_cliente.setText(celular);
178             vista.txt_email_cliente.setText(email);
179         }
180     }

```

Hotel - NetBeans IDE 8.2

```

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config> Source History
Projects Services Files Start Page ClienteController.java HabitacionController.java Cliente.java ClienteDAO.java Conexion.java Habitacion.java HabitacionDAO.java VistaClientes.java VistaHabitacion.java
Source Packages controllers ClientController.java HabitacionController.java models views Test Packages Libraries Test Libraries
182     private void actualizar(){
183         if (vista.txt_nombre_cliente.getText().equals("") || vista.txt_apellido_p_cliente.getText().equals("") || vista.txt_apellido_m_cliente.getText().equals("") || vista.txt_tipo_doc_cliente.getText().equals("") || vista.txt_num_doc_cliente.getText().equals("") || vista.txt_cel_cliente.getText().equals("")){
184             listar = false;
185         }else{
186             actualizar = true;
187             vista.btn_registrar_cliente.setEnabled(true);
188             vista.btn_actualizar_cliente.setEnabled(false);
189             vista.btn_nuevo_cliente.setEnabled(false);
190             int idCliente = Integer.parseInt(vista.txt_idCliente.getText());
191             String nombres = vista.txt_nombre_cliente.getText();
192             String apellido_paterno = vista.txt_apellido_p_cliente.getText();
193             String apellido_materno = vista.txt_apellido_m_cliente.getText();
194             String tipo_documento = vista.txt_tipo_doc_cliente.getText();
195             String num_documento = vista.txt_num_doc_cliente.getText();
196             String celular = vista.txt_cel_cliente.getText();
197             String email = vista.txt_email_cliente.getText();
198
199             Cliente cliente = new Cliente(idCliente, nombres, apellido_paterno, apellido_materno, tipo_documento, num_documento, celular, email);
200             int r = this.dao.actualizar(cliente);
201             if(r == 1){
202                 JOptionPane.showMessageDialog(vista, "Cliente actualizado con éxito");
203             }else{
204                 JOptionPane.showMessageDialog(vista, "Error al actualizar cliente");
205             }
206         }
207     }
208
209 }

```

Hotel - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects Services Files Start Page ClienteController.java HabitacionController.java Cliente.java ClienteDAO.java Conexion.java Habitacion.java <default config>

Source History

```

208     }
209 }
210
211     private void eliminar(){
212         int fila = vista.tb_datos_cliente.getSelectedRow();
213         if(fila == -1){
214             JOptionPane.showMessageDialog(vista, "Debe seleccionar una fila");
215         }else{
216             int id = Integer.parseInt((String)vista.tb_datos_cliente.getValueAt(fila, 0).toString());
217             int iden = this.dao.eliminar(id);
218             if (iden == 1){
219                 JOptionPane.showMessageDialog(vista, "Cliente eliminado con éxito");
220                 vista.btn_registrar_cliente.setEnabled(true);
221                 vista.btn_nuevo_cliente.setEnabled(false);
222                 vista.btn_actualizar_cliente.setEnabled(false);
223             }else{
224                 JOptionPane.showMessageDialog(vista, "Error al eliminar cliente");
225             }
226         }
227     }
228     private void salir(){
229         System.exit(0);
230     }
231 }
```

Navigator Members <empty>

- ClienteController :: ActionListener
 - HabitacionController(vistaHabitaciones vista)
 - actionPerformed(ActionEvent e)
 - actualizar()

Hotel - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects Services Files Start Page ClienteController.java HabitacionController.java Cliente.java ClienteDAO.java Cor <default config>

Source History

```

1 package controllers;
2
3 import java.awt.event.ActionEvent;
4 import java.awt.event.ActionListener;
5 import java.util.List;
6 import javax.swing.JOptionPane;
7 import javax.swing.JTable;
8 import javax.swing.table.DefaultTableModel;
9 import models.Habitacion;
10 import models.HabitacionDAO;
11 import views.VistaHabitaciones;
12
13 public class HabitacionController implements ActionListener{
14     boolean listar;
15     boolean actualizar;
16
17     DefaultTableModel modeloTabla = new DefaultTableModel();
18     VistaHabitaciones vista = new VistaHabitaciones();
19     HabitacionDAO dao = new HabitacionDAO();
20
21     public HabitacionController(VistaHabitaciones vistaHabitaciones){
22         this.vista=vistaHabitaciones;
23         this.vista.btnListar_hab.addActionListener(this);
24         this.vista.btnNuevo_hab.addActionListener(this);
25         this.vista.btnGuardar_hab.addActionListener(this);
26         this.vista.btnCancela_hab.addActionListener(this);
27         this.vista.btnBuscar_hab.addActionListener(this);
28         this.vista.btnEliminar_hab.addActionListener(this);
29         this.vista.btnExit_hab.addActionListener(this);
30         this.vista.btnEditar_hab.addActionListener(this);
31     }
32
33     @Override
34     public void actionPerformed(ActionEvent e) {
35
36     }
37 }
```

Navigator Members <empty>

- HabitacionController :: ActionListener
 - HabitacionController(vistaHabitaciones vista)
 - actionPerformed(ActionEvent e)
 - actualizar : boolean
 - dao : HabitacionDAO
 - listar : boolean
 - modeloTabla : DefaultTableModel
 - vista : VistaHabitaciones

Hotel - NetBeans IDE 8.2

```

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Source History <default config> T T W G D P S
Projects Services Files Start Page ClienteController.java HabitacionController.java Cliente.java ClienteDAO.java Conexion.java Habitacion.java HabitacionDAO.java VistaClientes.java VistaHabitacion.java
Source Packages controllers models views Test Packages Libraries Test Libraries
Client Controller.java HabitacionController.java
Client.java ClienteDAO.java Conexion.java Habitacion.java HabitacionDAO.java
Navigator X Members <empty>
Client
  Client()
  Client(int idCliente, String nombres, String apellido_paterno, String apellido_materno, String tipo_documento, String num_documento, String celular, String email)
  Cliente(String nombres, String apellido_paterno, String apellido_materno, String tipo_documento, String num_documento, String celular, String email)
  public int getIdCliente() { return idCliente; }
  public void setIdCliente(int idCliente) { this.idCliente = idCliente; }
  public String getNombres() { return nombres; }
  public void setNombres(String nombres) { this.nombres = nombres; }
  public String getApellido_paterno() { return apellido_paterno; }
  public void setApellido_paterno(String apellido_paterno) { this.apellido_paterno = apellido_paterno; }
  public String getApellido_materno() { return apellido_materno; }
  public void setApellido_materno(String apellido_materno) { this.apellido_materno = apellido_materno; }
  public String getTipo_documento() { return tipo_documento; }
  public void setTipo_documento(String tipo_documento) { this.tipo_documento = tipo_documento; }
  public String getNum_documento() { return num_documento; }
  public void setNum_documento(String num_documento) { this.num_documento = num_documento; }
  public String getCelular() { return celular; }
  public void setCelular(String celular) { this.celular = celular; }
  public String getEmail() { return email; }
  public void setEmail(String email) { this.email = email; }
}

```

Hotel - NetBeans IDE 8.2

```

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Source History <default config> T T W G D P S
Projects Services Files Start Page ClienteController.java HabitacionController.java Cliente.java ClienteDAO.java Conexion.java
Source Packages controllers models views Test Packages Libraries Test Libraries
Client Controller.java HabitacionController.java
Client.java ClienteDAO.java Conexion.java Habitacion.java HabitacionDAO.java
Navigator X Members <empty>
Client
  Client()
  Client(int idCliente, String nombres, String apellido_paterno, String apellido_materno, String tipo_documento, String num_documento, String celular, String email)
  Cliente(String nombres, String apellido_paterno, String apellido_materno, String tipo_documento, String num_documento, String celular, String email)
  public int getIdCliente() { return idCliente; }
  public void setIdCliente(int idCliente) { this.idCliente = idCliente; }
  public String getNombres() { return nombres; }
  public void setNombres(String nombres) { this.nombres = nombres; }
  public String getApellido_paterno() { return apellido_paterno; }
  public void setApellido_paterno(String apellido_paterno) { this.apellido_paterno = apellido_paterno; }
  public String getApellido_materno() { return apellido_materno; }
  public void setApellido_materno(String apellido_materno) { this.apellido_materno = apellido_materno; }
  public String getTipo_documento() { return tipo_documento; }
  public void setTipo_documento(String tipo_documento) { this.tipo_documento = tipo_documento; }
  public String getNum_documento() { return num_documento; }
  public void setNum_documento(String num_documento) { this.num_documento = num_documento; }
  public String getCelular() { return celular; }
  public void setCelular(String celular) { this.celular = celular; }
  public String getEmail() { return email; }
  public void setEmail(String email) { this.email = email; }
}

```

Hotel - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects X Services Files

Source Packages

- Hotel
 - controllers
 - ClienteController.java
 - HabitacionController.java
 - models
 - Cliente.java
 - ClienteDAO.java
 - Conexion.java
 - Habitacion.java
 - HabitacionDAO.java
 - views
 - Test Packages
 - Libraries
 - Test Libraries

Start Page X ClienteController.java X HabitacionController.java X Cliente.java X ClienteDAO.java X

Source History

```

1 package models;
2
3 import java.sql.Connection;
4 import java.sql.PreparedStatement;
5 import java.sql.ResultSet;
6 import java.util.ArrayList;
7 import java.util.List;
8
9
10 public class ClienteDAO {
11     Conexion conectar = new Conexion();
12     Connection con;
13     PreparedStatement ps;
14     ResultSet rs;
15
16     public List<Cliente> listar(){
17         List<Cliente> listarCliente = new ArrayList<>();
18         String sql = "SELECT * FROM cliente";
19         try {
20             con = conectar.conectarServidor();
21             ps = con.prepareStatement(sql);
22             rs = ps.executeQuery();
23             while(rs.next()){
24                 Cliente cli = new Cliente();
25                 cli.setIdCliente(rs.getInt(1));
26                 cli.setNombres(rs.getString(2));
27                 cli.setApellido_paterno(rs.getString(3));
28                 cli.setApellido_materno(rs.getString(4));
29                 cli.setTipo_documento(rs.getString(5));
30                 cli.setNum_documento(rs.getString(6));
31                 cli.setCelular(rs.getString(7));
32                 cli.setEmail(rs.getString(8));
33                 listarCliente.add(cli);
34             }
35         } catch (Exception e) {
36             System.err.println("Error al listar: "+e.getMessage());
37         }
38     }
39
40     return listarCliente;
41 }
```

Navigator X

Members <empty>

ClienteDAO

- actualizar(Cliente cli) : int
- agregar(Cliente cli) : int
- eliminar(int idCliente) : int
- listar() : List<Cliente>
- con : Connection
- conectar : Conexion
- ps : PreparedStatement
- rs : ResultSet

Hotel - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects X Services Files

Start Page X ClienteController.java X HabitacionController.java X Cliente.java X ClienteDAO.java X Conexion.java X Habitacion.java X HabitacionDAO.java X VistaClientes.java X VistaHabit...

Source History

```

41     public int agregar(Cliente cli){
42         String sql = "INSERT INTO cliente VALUES(null, ?, ?, ?, ?, ?, ?)";
43         try{
44             con = conectar.conectarServidor();
45             ps = con.prepareStatement(sql);
46             ps.setString(1, cli.getNombres());
47             ps.setString(2, cli.getApellido_paterno());
48             ps.setString(3, cli.getApellido_materno());
49             ps.setString(4, cli.getTipo_documento());
50             ps.setString(5, cli.getNum_documento());
51             ps.setString(6, cli.getCelular());
52             ps.setString(7, cli.getEmail());
53             ps.executeUpdate();
54         } catch (Exception e) {
55             System.err.println("Error al agregar: "+e.getMessage());
56         }
57     }
58
59     return 1;
60 }
61
62     public int actualizar(Cliente cli){
63         int r = 0;
64         String sql = "UPDATE cliente SET nombres = ?, apellido_paterno = ?, apellido_materno = ?, tipo_documento = ?, num_documento = ?, celular = ?, email = ? WHERE id_cliente = ?";
65         try {
66             con = conectar.conectarServidor();
67             ps = con.prepareStatement(sql);
68             ps.setString(1, cli.getNombres());
69             ps.setString(2, cli.getApellido_paterno());
70             ps.setString(3, cli.getApellido_materno());
71             ps.setString(4, cli.getTipo_documento());
72             ps.setString(5, cli.getNum_documento());
73             ps.setString(6, cli.getCelular());
74             ps.setString(7, cli.getEmail());
75             ps.setInt(8, cli.getIdCliente());
76             r = ps.executeUpdate();
77         }
```

Hotel - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects X Services Files — Start Page X ClienteController.java X HabitacionController.java X Cliente.java X ClienteDAO.java X

Source History

```

76     if(r == 1){
77         return 1;
78     }else{
79         return 0;
80     }
81     } catch (Exception e) {
82         System.err.println("Error al actualizar: "+e.getMessage());
83     }
84     return r;
85 }

86

87     public int eliminar(int idCliente){
88         String sql = "DELETE FROM cliente WHERE idCliente = "+idCliente;
89         try {
90             con = conectar.conectarServidor();
91             ps = con.prepareStatement(sql);
92             ps.executeUpdate();
93             return 1;
94         } catch (Exception e) {
95             System.err.println("Error al eliminar: "+e.getMessage());
96             return 0;
97         }
98     }
99 }
```

Navigator X

Members <empty>

ClienteDAO

- actualizar(Cliente d1) : int
- agregar(Cliente d1) : int

Hotel - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects X Services Files — Start Page X ClienteController.java X HabitacionController.java X Cliente.java X ClienteDAO.java X Conexion.java X

Source History

```

1 package models;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5
6 public class Conexion {
7     Connection con;
8     public Connection conectarServidor(){
9         String url = "jdbc:mysql://localhost:3306/proyecto?serverTimezone=America/Lima";
10        String user = "root";
11        String pass = "";
12        try {
13            Class.forName("com.mysql.cj.jdbc.Driver");
14            con = DriverManager.getConnection(url,user,pass);
15        } catch (Exception e) {
16            System.err.println("No se conectó a la Base de Datos: "+e.getMessage());
17        }
18        return con;
19    }
20 }
```

Navigator X

Hotel - NetBeans IDE 8.2

```

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config> T T D D <empty> Source History > < > <> << >> <<< >>>
Projects Services Files Start Page ClienteController.java HabitacionController.java Cliente.java ClienteDAO.java Conexion.java Habitacion.java HabitacionDAO.java
Source Packages
controllers
models
views
Test Packages
Libraries
Test Libraries
Source
1 package models;
2
3 public class Habitacion {
4     private int idHabitacion;
5     private String numero;
6     private String piso;
7     private Double precio;
8     private String estado;
9     private String tipo_Habitacion;
10
11     public Habitacion(int idHabitacion, String numero, Double precio, String estado, String tipo_Habitacion) {
12         this.idHabitacion = idHabitacion;
13         this.numero = numero;
14         this.piso = piso;
15         this.precio = precio;
16         this.estado = estado;
17         this.tipo_Habitacion = tipo_Habitacion;
18     }
19
20     public Habitacion(String numero, String piso, Double precio, String estado, String tipo_Habitacion) {
21         this.numero = numero;
22         this.piso = piso;
23         this.precio = precio;
24         this.estado = estado;
25         this.tipo_Habitacion = tipo_Habitacion;
26     }
27
28     public Habitacion() {
29     }
30
31     public int getIdHabitacion() {
32         return idHabitacion;
33     }
34
35     public void setIdHabitacion(int idHabitacion) {
36         this.idHabitacion = idHabitacion;
37     }

```

Navigator

Members

Habitacion

- Habitacion(int idHabitacion, String numero, String piso, String tipo_Habitacion)
- Habitacion(String numero, String piso, Double precio, String estado, String tipo_Habitacion)
- Habitacion()
- getIdHabitacion()
- setIdHabitacion(int idHabitacion)
- setNumero(String numero)
- setPiso(String piso)
- setPrecio(Double precio)

Hotel - NetBeans IDE 8.2

```

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config> T T D D <empty> Source History > < > <> << >>>
Projects Services Files Start Page ClienteController.java HabitacionController.java Cliente.java ClienteDAO.java Conexion.java Habitacion.java HabitacionDAO.java
Source Packages
controllers
models
views
Test Packages
Libraries
Test Libraries
Source
1 package models;
2
3 import java.sql.Connection;
4 import java.sql.PreparedStatement;
5 import java.sql.ResultSet;
6 import java.util.ArrayList;
7 import java.util.List;
8
9 public class HabitacionDAO {
10     Conexion conectar = new Conexion();
11     Connection con;
12     PreparedStatement ps;
13     ResultSet rs;
14
15     public List<Habitacion> listar(){
16         List<Habitacion> listarHabitacion = new ArrayList<>();
17         String sql = "SELECT * FROM habitacion";
18         try {
19             con = conectar.conectarServidor();
20             ps = con.prepareStatement(sql);
21             rs = ps.executeQuery();
22             while(rs.next()){
23                 Habitacion hab = new Habitacion();
24                 hab.setIdHabitacion(rs.getInt(1));
25                 hab.setNumero(rs.getString(2));
26                 hab.setPiso(rs.getString(3));
27                 hab.setPrecio(rs.getDouble(4));
28                 hab.setEstado(rs.getString(5));
29                 hab.setTipo_Habitacion(rs.getString(6));
30                 listarHabitacion.add(hab);
31             }
32         } catch (Exception e) {
33             System.err.println("Error al listar: "+e.getMessage());
34         }
35         return listarHabitacion;
36     }

```

Navigator

Members

HabitacionDAO

- actualizar(Habitacion hab) : int
- agregar(Habitacion hab) : int
- eliminar(int idHabitacion) : int
- listar() : List<Habitacion>
- con : Connection
- conectar : Conexion
- ps : PreparedStatement
- rs : ResultSet

HabitacionDAO.java

```

public int agregar(Habitacion hab) {
    String sql = "INSERT INTO habitacion VALUES(null, ?, ?, ?, ?, ?)";
    try{
        con = conectar.conectarServidor();
        ps = con.prepareStatement(sql);
        ps.setString(1, hab.getNumero());
        ps.setString(2, hab.getPiso());
        ps.setDouble(3, hab.getPrecio());
        ps.setString(4, hab.getEstado());
        ps.setString(5, hab.getTipo_Habitacion());
        ps.executeUpdate();
    } catch (Exception e) {
        System.err.println("Error al agregar: "+e.getMessage());
        return 0;
    }
    return 1;
}

public int actualizar(Habitacion hab){
    int r = 0;
    String sql = "UPDATE habitacion SET numero = ?, piso = ?, precio = ?, estado = ?, tipo_habitacion = ? WHERE idHabitacion = ?";
    try {
        con = conectar.conectarServidor();
        ps = con.prepareStatement(sql);
        ps.setString(1, hab.getNumero());
        ps.setString(2, hab.getPiso());
        ps.setDouble(3, hab.getPrecio());
        ps.setString(4, hab.getEstado());
        ps.setString(5, hab.getTipo_Habitacion());
        ps.setInt(6, hab.getIdHabitacion());
        r = ps.executeUpdate();
        if(r == 1){
            return 1;
        }else{
            return 0;
        }
    } catch (Exception e) {
        System.err.println("Error al actualizar: "+e.getMessage());
    }
    return r;
}

```

HabitacionDAO.java

```

r = ps.executeUpdate();
if(r == 1){
    return 1;
}else{
    return 0;
}
} catch (Exception e) {
    System.err.println("Error al actualizar: "+e.getMessage());
}
return r;

}

public int eliminar(int idHabitacion){
    String sql = "DELETE FROM habitacion WHERE idHabitacion = "+idHabitacion;
    try {
        con = conectar.conectarServidor();
        ps = con.prepareStatement(sql);
        ps.executeUpdate();
        return 1;
    } catch (Exception e) {
        System.err.println("Error al eliminar: "+e.getMessage());
        return 0;
    }
}

```

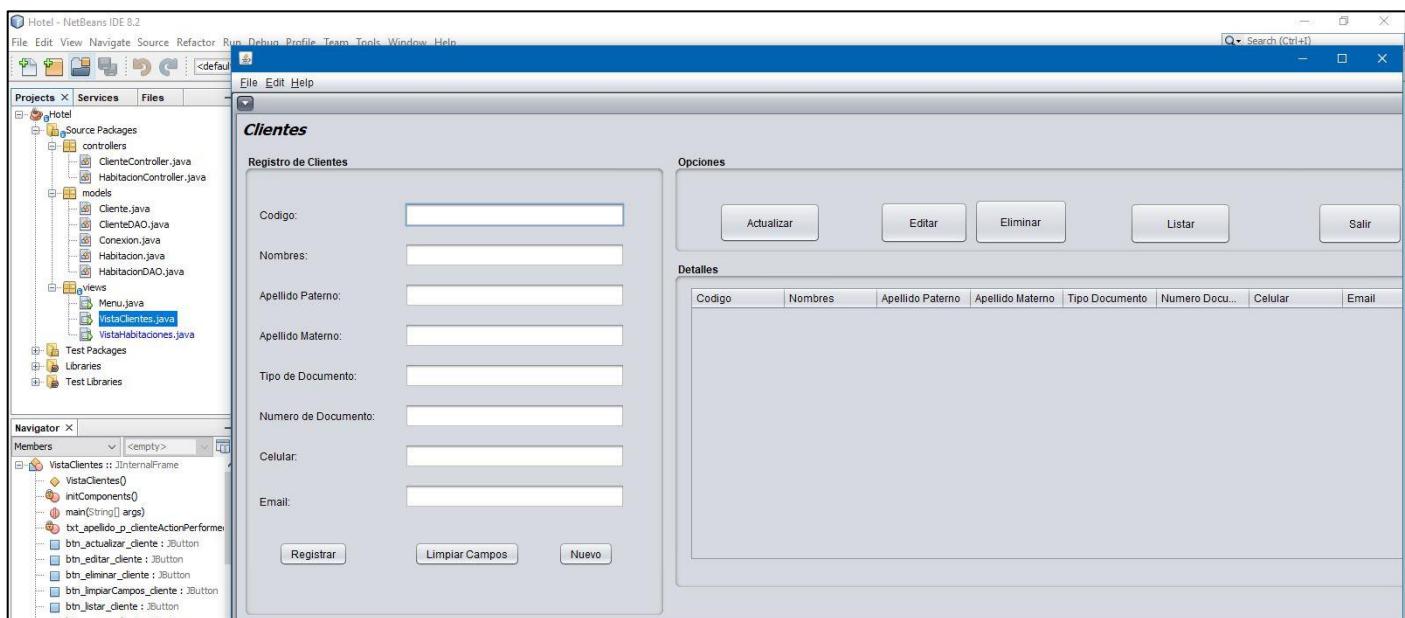


Tabla 3: Especificación de Casos de Uso

ID	ITEM	DESCRIPCIÓN
1	Nombre Corto	Registro de Trabajador
2	Actores	Administrador
3	Objetivo	Ingresar un nuevo trabajador a nuestro sistema.
4	Disparador	El administrador desea registrar un nuevo trabajador en el sistema
5	Pre condiciones	Se debe validar el trabajador mediante el Login
6	Post condiciones	El trabajador será registrado exitosamente.
7	Escenario básico	<p>Escenario 1: Validar trabajador.</p> <ul style="list-style-type: none"> • Administrador ingresa su usuario y su contraseña • El sistema valida el usuario y contraseña <p>Escenario 2: Consulta de trabajadores</p> <ul style="list-style-type: none"> • Administrador solicita la consulta de todos los trabajadores. • El sistema le muestra los datos de todos los trabajadores. <p>Escenario 3: Modificar al trabajador</p> <ul style="list-style-type: none"> • Administrador solicita modificar los datos del trabajador • El sistema le permite modificar los datos del trabajador y se realizan los cambios respectivos <p>Escenario 4: Eliminar al trabajador</p> <ul style="list-style-type: none"> • Administrador solicita eliminar un trabajador • El sistema le permite eliminar al trabajador. <ul style="list-style-type: none"> ➤ El administrador registra todos los datos del trabajador ➤ El sistema valida todos los datos del trabajador ➤ El sistema registra el nombre del trabajador ➤ El sistema registra el apellido paterno del trabajador ➤ El sistema registra el apellido materno del trabajador ➤ El sistema registra el tipo de documento del trabajador ➤ El sistema registra el número de documento del trabajador ➤ El sistema registra el celular del trabajador ➤ El sistema registra el email del trabajador ➤ El sistema registra el sueldo del trabajador

		<ul style="list-style-type: none"> ➤ El sistema registra el acceso del trabajador ➤ El sistema registra el login del trabajador ➤ El sistema registra el password del trabajador ➤ El sistema registra el estado del trabajador
8	Escenario Alternativo	<ul style="list-style-type: none"> ➤ Si no se registra el trabajador, enviar mensaje de ayuda. ➤ Si faltan datos del trabajador, enviar mensaje de ayuda. ➤ Si el usuario no es válido, enviar mensaje de ayuda. ➤ Si la contraseña no es válida, enviar mensaje de ayuda. ➤ Si se consulta datos de un trabajador no seleccionado, enviar mensaje de ayuda. ➤ Si se modifica datos de un trabajador no seleccionado, enviar mensaje de ayuda. ➤ Si se elimina un trabajador no seleccionado, enviar mensaje de ayuda.
9	Prioridad	Versión 1

ID	ITEM	DESCRIPCIÓN
1	Nombre Corto	Registro de Trabajador
2	Actores	Administrador
3	Objetivo	Ingresar un nuevo trabajador a nuestro sistema.
4	Disparador	El administrador desea registrar un nuevo trabajador en el sistema
5	Pre condiciones	Se debe validar el trabajador mediante el Login
6	Post condiciones	El trabajador será registrado exitosamente.
7	Escenario básico	<p>Escenario 1: Validar trabajador.</p> <ul style="list-style-type: none"> • Administrador ingresa su usuario y su contraseña • El sistema valida el usuario y contraseña <p>Escenario 2: Consulta de trabajadores</p> <ul style="list-style-type: none"> • Administrador solicita la consulta de todos los trabajadores. • El sistema le muestra los datos de todos los trabajadores. <p>Escenario 3: Modificar al trabajador</p> <ul style="list-style-type: none"> • Administrador solicita modificar los datos del trabajador • El sistema le permite modificar los datos del trabajador y se realizan los cambios respectivos <p>Escenario 4: Eliminar al trabajador</p> <ul style="list-style-type: none"> • Administrador solicita eliminar un trabajador • El sistema le permite eliminar al trabajador. <ul style="list-style-type: none"> ➤ El administrador registra todos los datos del trabajador ➤ El sistema valida todos los datos del trabajador ➤ El sistema registra el nombre del trabajador ➤ El sistema registra el apellido paterno del trabajador ➤ El sistema registra el apellido materno del trabajador ➤ El sistema registra el tipo de documento del trabajador ➤ El sistema registra el número de documento del trabajador ➤ El sistema registra el celular del trabajador ➤ El sistema registra el email del trabajador

		<ul style="list-style-type: none"> ➤ El sistema registra el sueldo del trabajador ➤ El sistema registra el acceso del trabajador ➤ El sistema registra el login del trabajador ➤ El sistema registra el password del trabajador ➤ El sistema registra el estado del trabajador
8	Escenario Alternativo	<ul style="list-style-type: none"> ➤ Si no se registra el trabajador, enviar mensaje de ayuda. ➤ Si faltan datos del trabajador, enviar mensaje de ayuda. ➤ Si el usuario no es válido, enviar mensaje de ayuda. ➤ Si la contraseña no es válida, enviar mensaje de ayuda. ➤ Si se consulta datos de un trabajador no seleccionado, enviar mensaje de ayuda. ➤ Si se modifica datos de un trabajador no seleccionado, enviar mensaje de ayuda. ➤ Si se elimina un trabajador no seleccionado, enviar mensaje de ayuda.
9	Prioridad	Versión 1

ID	ITEM	DESCRIPCIÓN
1	Nombre Corto	Registro de Clientes
2	Actores	Administrador y Recepcionista
3	Objetivo	Ingresar un nuevo cliente a nuestro sistema.
4	Disparador	El administrador o recepcionista desea registrar un nuevo cliente en el sistema
5	Pre condiciones	Se debe validar el trabajador mediante el Login
6	Post condiciones	El cliente será registrado exitosamente.
7	Escenario básico	<p>Escenario 1: Validar trabajador.</p> <ul style="list-style-type: none"> • Administrador o recepcionista ingresa su usuario y su contraseña • El sistema valida el usuario y contraseña <p>Escenario 2: Consulta de clientes</p> <ul style="list-style-type: none"> • Administrador o recepcionista solicita la consulta de todos los clientes. • El sistema le muestra los datos de todos los clientes. <p>Escenario 3: Modificar al cliente</p> <ul style="list-style-type: none"> • Administrador o recepcionista solicita modificar los datos del cliente • El sistema le permite modificar los datos del cliente y se realizan los cambios respectivos <p>Escenario 4: Eliminar al cliente</p> <ul style="list-style-type: none"> • Administrador o recepcionista solicita eliminar un cliente • El sistema le permite eliminar al cliente. <ul style="list-style-type: none"> ➤ El administrador o recepcionista registra todos los datos del cliente ➤ El sistema valida todos los datos del cliente ➤ El sistema registra el nombre del cliente ➤ El sistema registra el apellido paterno del cliente ➤ El sistema registra el apellido materno del cliente

		<ul style="list-style-type: none"> ➤ El sistema registra el tipo de documento del cliente ➤ El sistema registra el número de documento del cliente ➤ El sistema registra el celular del cliente ➤ El sistema registra el email del cliente
8	Escenario Alternativo	<ul style="list-style-type: none"> ➤ Si no se registra el cliente, enviar mensaje de ayuda. ➤ Si faltan datos del cliente, enviar mensaje de ayuda. ➤ Si el usuario no es válido, enviar mensaje de ayuda. ➤ Si la contraseña no es válida, enviar mensaje de ayuda. ➤ Si se consulta datos de un cliente no seleccionado, enviar mensaje de ayuda. ➤ Si se modifica datos de un cliente no seleccionado, enviar mensaje de ayuda. ➤ Si se elimina un cliente no seleccionado, enviar mensaje de ayuda.
9	Prioridad	Versión 1

ID	ITEM	DESCRIPCIÓN
1	Nombre Corto	Registro de ingreso de cliente
2	Actores	Administrador
3	Objetivo	Ingresar el ingreso de cliente a nuestro sistema.
4	Disparador	El administrador desea registrar el ingreso del cliente en el sistema
5	Pre condiciones	Se debe validar el trabajador mediante el Login
6	Post condiciones	El ingreso de cliente será registrado exitosamente.
7	Escenario básico	<p>Escenario 1: Validar trabajador.</p> <ul style="list-style-type: none"> • Administrador ingresa su usuario y su contraseña • El sistema valida el usuario y contraseña <p>Escenario 2: Consulta de ingreso de clientes</p> <ul style="list-style-type: none"> • Administrador solicita la consulta de ingreso de todos los clientes. • El sistema le muestra el ingreso de todos los clientes. <p>Escenario 3: Modificar el ingreso de cliente</p> <ul style="list-style-type: none"> • Administrador solicita modificar el ingreso del cliente • El sistema le permite modificar el ingreso del cliente y se realizan los cambios respectivos <p>Escenario 4: Eliminar el ingreso del cliente</p> <ul style="list-style-type: none"> • Administrador solicita eliminar el ingreso de cliente • El sistema le permite eliminar el ingreso de cliente. <p>➤ El administrador registra el ingreso del cliente ➤ El sistema valida los datos de ingreso del cliente ➤ El sistema registra el ingreso del cliente</p>
8	Escenario Alternativo	<p>➤ Si no se registra el ingreso del cliente, enviar mensaje de ayuda.</p> <p>➤ Si faltan datos en el ingreso de cliente, enviar mensaje de ayuda.</p> <p>➤ Si el usuario no es válido, enviar mensaje de ayuda.</p> <p>➤ Si la contraseña no es válida, enviar mensaje de ayuda.</p>

		<ul style="list-style-type: none">➤ Si se consulta el ingreso del cliente no seleccionado, enviar mensaje de ayuda.➤ Si se modifica el ingreso del cliente no seleccionado, enviar mensaje de ayuda.➤ Si se elimina el ingreso del cliente no seleccionado, enviar mensaje de ayuda.
9	Prioridad	Versión 1

ID	ITEM	DESCRIPCIÓN
1	Nombre Corto	Registro de salida de cliente
2	Actores	Administrador
3	Objetivo	Ingresar la salida de cliente a nuestro sistema.
4	Disparador	El administrador desea registrar la salida del cliente en el sistema
5	Pre condiciones	Se debe validar el trabajador mediante el Login
6	Post condiciones	La salida de cliente será registrada exitosamente.
7	Escenario básico	<p>Escenario 1: Validar trabajador.</p> <ul style="list-style-type: none"> • Administrador ingresa su usuario y su contraseña • El sistema valida el usuario y contraseña <p>Escenario 2: Consulta de salida de clientes</p> <ul style="list-style-type: none"> • Administrador solicita la consulta de salida de todos los clientes. • El sistema le muestra la salida de todos los clientes. <p>Escenario 3: Modificar la salida de cliente</p> <ul style="list-style-type: none"> • Administrador solicita modificar la salida del cliente • El sistema le permite modificar la salida del cliente y se realizan los cambios respectivos <p>Escenario 4: Eliminar la salida del cliente</p> <ul style="list-style-type: none"> • Administrador solicita eliminar la salida de cliente • El sistema le permite eliminar la salida de cliente. <p>➤ El administrador registra la salida del cliente</p> <p>➤ El sistema valida los datos de salida del cliente</p> <p>➤ El sistema registra la salida del cliente</p>
8	Escenario Alternativo	<p>➤ Si no se registra la salida del cliente, enviar mensaje de ayuda.</p> <p>➤ Si faltan datos en la salida de cliente, enviar mensaje de ayuda.</p> <p>➤ Si el usuario no es válido, enviar mensaje de ayuda.</p> <p>➤ Si la contraseña no es válida, enviar mensaje de ayuda.</p>

		<ul style="list-style-type: none"> ➤ Si se consulta la salida del cliente no seleccionado, enviar mensaje de ayuda. ➤ Si se modifica la salida del cliente no seleccionado, enviar mensaje de ayuda. ➤ Si se elimina la salida del cliente no seleccionado, enviar mensaje de ayuda.
9	Prioridad	Versión 1

CAPÍTULO 4. CONCLUSIONES Y RECOMENDACIONES

12. Conclusiones

- Conclusión 1
- Conclusión 2
- Conclusión 3
- Conclusión 4
- Conclusión 5

13. Recomendaciones

- Recomendación 1
- Recomendación 2
- Recomendación 3
- Recomendación 4
- Recomendación 5

14. Referencias Bibliográficas

- Alonso, F., Martínez, L., & Segovia , J. (2005). *Introducción a la ingeniería de Software*. Zaragoza : Delta Publicaciones.
- Git Hub. (s.f.). Obtenido de <https://github.com/>
- Lucidchart. (s.f.). Obtenido de <https://www.lucidchart.com/pages/es>
- Sommerville, I. (2011). *Ingeniería de software* (9a. ed.). Pearson Educación.

15. Anexos

