



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.....	70
12. Conclusiones.....	70
13. Recomendaciones	70
14. Referencias Bibliográficas	71
15. Anexos	71

ÍNDICE DE TABLAS

Tabla 1: Proceso de Negocio	5
Tabla 2: Trazabilidad de Requisitos	11
Tabla 3: Especificación de Casos de Uso - 1	41
Tabla 4: Especificación de Casos de Uso - 2.....	43
Tabla 5: Especificación de Casos de Uso - 3.....	45
Tabla 6: Especificación de Casos de Uso - 4.....	47
Tabla 7: Plan de pruebas Incremental - 1	49
Tabla 8: Plan de pruebas Incremental - 2.....	51

ÍNDICE DE FIGURAS

Ilustración 1: Proceso de Negocio.....	4
Ilustración 2: Proceso de Software.....	5
Ilustración 3: Diagrama de Casos de uso	6
Ilustración 4: Prototipo - 1.....	6
Ilustración 5: Prototipo - 2.....	7
Ilustración 6: Prototipo - 3.....	7
Ilustración 7: Prototipo - 4.....	8
Ilustración 8: Diagrama de actividades con particiones o calles.....	9
Ilustración 9: Requisitos no funcionales - Diego Tasaico - 1.....	12
Ilustración 10: Requisitos no funcionales - Diego Tasaico - 2.....	12
Ilustración 11: Requisitos no funcionales - Diego Tasaico - 3.....	13
Ilustración 12: Requisitos no funcionales - Diego Tasaico - 4.....	13
Ilustración 13: Requisitos no funcionales - Mauricio Villanueva - 1.....	14
Ilustración 14: Requisitos no funcionales - Mauricio Villanueva - 2.....	14
Ilustración 15: Requisitos no funcionales - Mauricio Villanueva - 3.....	15
Ilustración 16: Requisitos no funcionales - Abihail Zapata - 1.....	15
Ilustración 17: Requisitos no funcionales - Abihail Zapata - 2.....	16
Ilustración 18: Requisitos no funcionales - Abihail Zapata - 3.....	16
Ilustración 19: Requisitos no funcionales - Abihail Zapata - 4.....	17
Ilustración 20: Requisitos no funcionales - Olenka Lazo - 1	17
Ilustración 21: Requisitos no funcionales - Olenka Lazo - 2	18
Ilustración 22: Requisitos no funcionales - Olenka Lazo - 3	18
Ilustración 23: Requisitos no funcionales - Olenka Lazo - 4	19
Ilustración 24: Repositorio GitHub - 1	19
Ilustración 25: Repositorio GitHub - 2	20
Ilustración 26: Repositorio GitHub - 3	20

Ilustración 27: Repositorio GitHub - 4	21
Ilustración 28: Repositorio GitHub - 5	21
Ilustración 29: Repositorio GitHub - 6	22
Ilustración 30: Repositorio GitHub - 7	22
Ilustración 31: Diagrama de Casos de uso Relacionado - General.....	29
Ilustración 32: Diagrama de Casos de uso Relacionado – Versión 1	30
Ilustración 33: Código de programación del software	31
Ilustración 34: Videoconferencia con representante de la empresa	71
Ilustración 35: Integrante - Diego Tasaico	72
Ilustración 36: Integrante - Mauricio Villanueva	73
Ilustración 37: Integrante - Olenka Lazo	73
Ilustración 38: Integrante - Abihail Zapata	74
Ilustración 39: Acta de compromiso.....	75
Ilustración 40: Funcionamiento del Software	76

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.
- NetBeans: Es una herramienta para desarrollar rápida y fácilmente aplicaciones de escritorio, móviles y web con Java, JavaScript, HTML5, PHP, C/C++ y mas.
- WampServer: Es un ambiente de desarrollo web para Windows. Te permite crear aplicaciones con Apache2, PHP y la base de datos MySQL.
- MySQL Workbench: Es una herramienta visual unificada para arquitectos de base de datos, desarrolladores y DBAs. Provee herramientas de modelamiento de datos, desarrollo SQL, herramientas de administración de servidores, administración de usuarios, respaldos y mucho mas.
- MySQL: Es una base de datos de código abierto. Esta 100% desarrollada, gestionada y sostenida por el equipo de MySQL.
- (NetBeans, s.f.) (MySQL, s.f.) (WampServer, s.f.) (MySQL WorkBench, s.f.)

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
- Paso 10: Realizar la captura de los requisitos

- Paso 11: Validar los requisitos
- Paso 12: Realizar la trazabilidad de los requisitos
- Paso 13: Instalar los requisitos no funcionales
- Paso 14: Realizar el diagrama de casos de uso relacionado
- Paso 15: Especificar los casos de uso

CAPÍTULO 3. DESARROLLO DE LA INVESTIGACIÓN

11. Implementación de la solución planteada

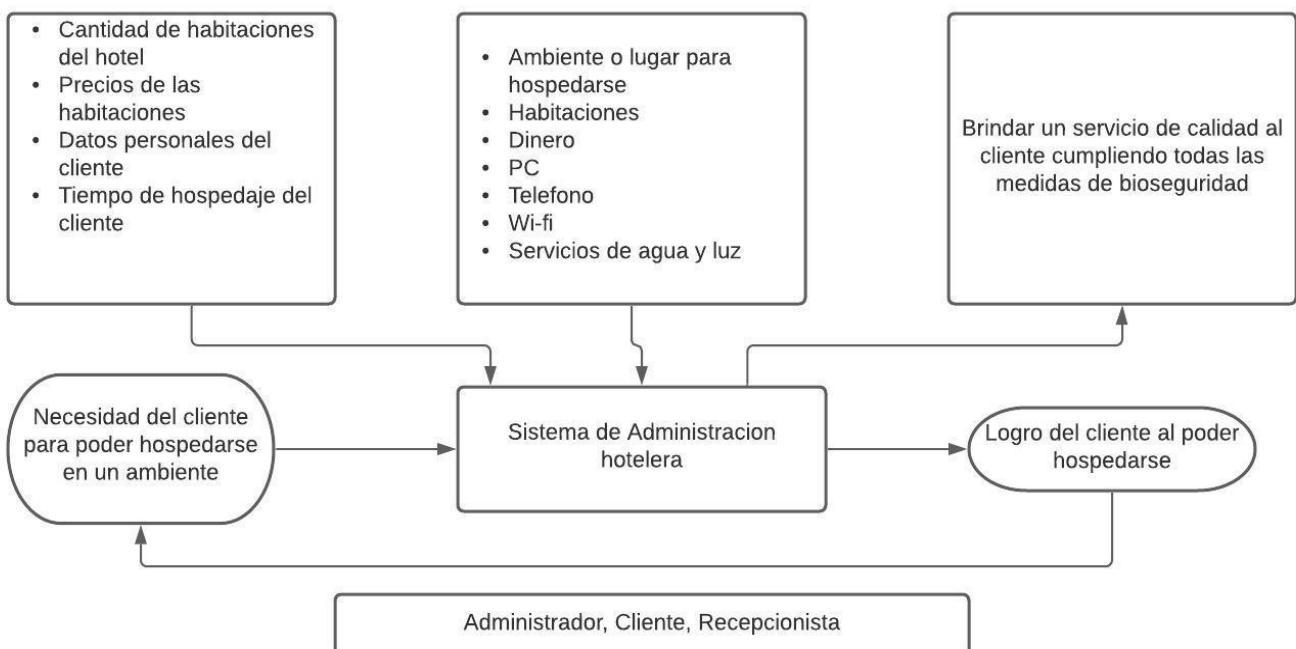


Ilustración 1: Proceso de Negocio

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 de salida del cliente	Brindar un servicio de calidad al cliente cumpliendo todas las medidas de bioseguridad

Tabla 1: Proceso de Negocio

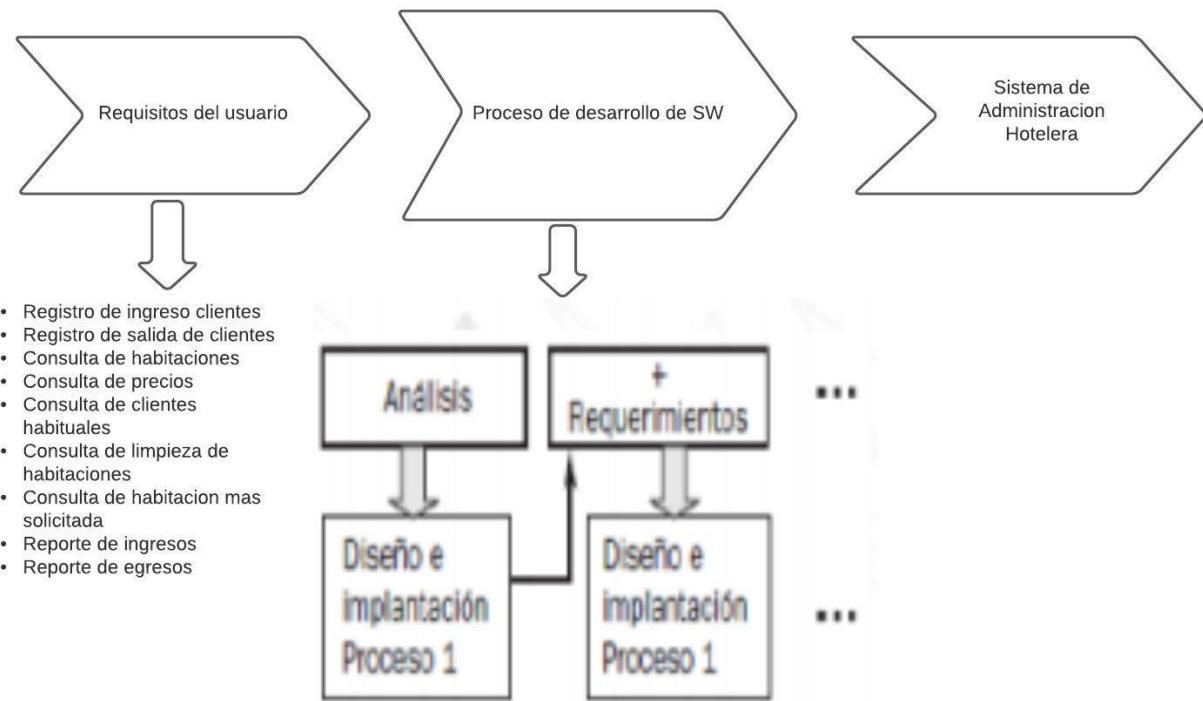


Ilustración 2: Proceso de Software

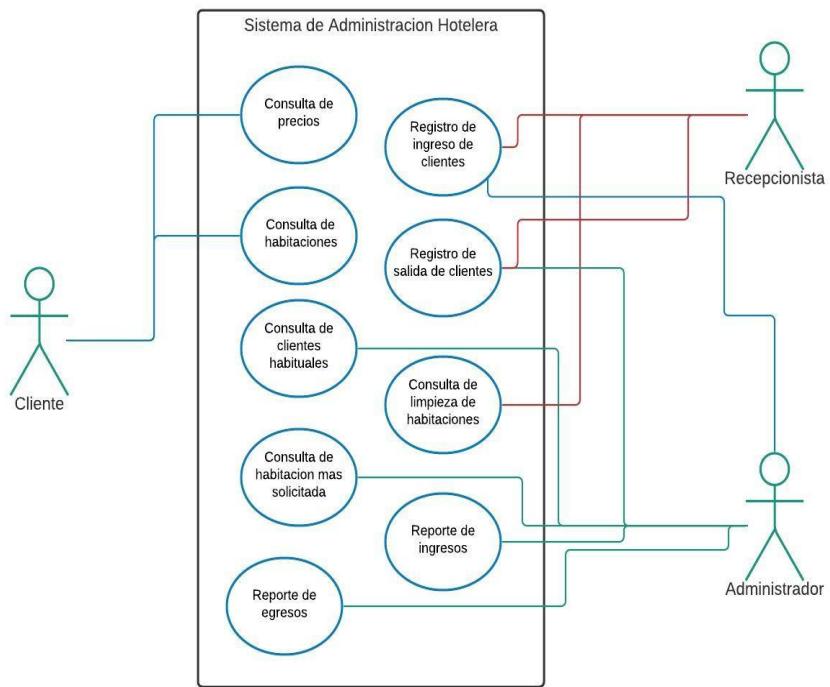


Ilustración 3: Diagrama de Casos de uso

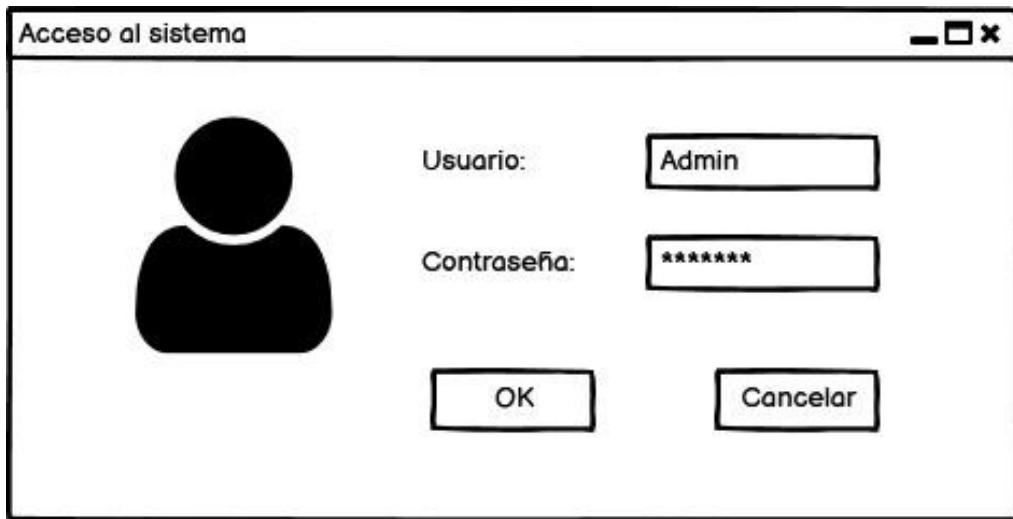


Ilustración 4: Prototipo - 1

Trabajador

Registro de trabajadores

Código:	[Text]
Nombres:	[Text]
Apellido Paterno:	[Text]
Apellido Materno:	[Text]
Tipo de Documento:	[Text]
Número de Documento:	[Text]
Celular:	[Text]
Email:	[Text]
Sueldo:	[Text]
Acceso:	[Text]
Login:	[Text]
Password:	[Text]
Estado:	[Text]

Opciones

Actualizar	Editor	Eliminar	Listar	Salir
------------	--------	----------	--------	-------

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	pedroazab	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

Ilustración 5: Prototipo - 2

Clientes

Registro de clientes

Código:	[Text]
Nombres:	[Text]
Apellido Paterno:	[Text]
Apellido Materno:	[Text]
Tipo de Documento:	[Text]
Número de Documento:	[Text]
Celular:	[Text]
Email:	[Text]

Opciones

Actualizar	Editor	Eliminar	Listar	Salir
------------	--------	----------	--------	-------

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

Ilustración 6: Prototipo - 3

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

<input type="button" value="Actualizar"/>	<input type="button" value="Editar"/>	<input type="button" value="Eliminar"/>	<input type="button" value="Listar"/>	<input type="button" value="Salir"/>
---	---------------------------------------	---	---------------------------------------	--------------------------------------

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 7: Prototipo - 4

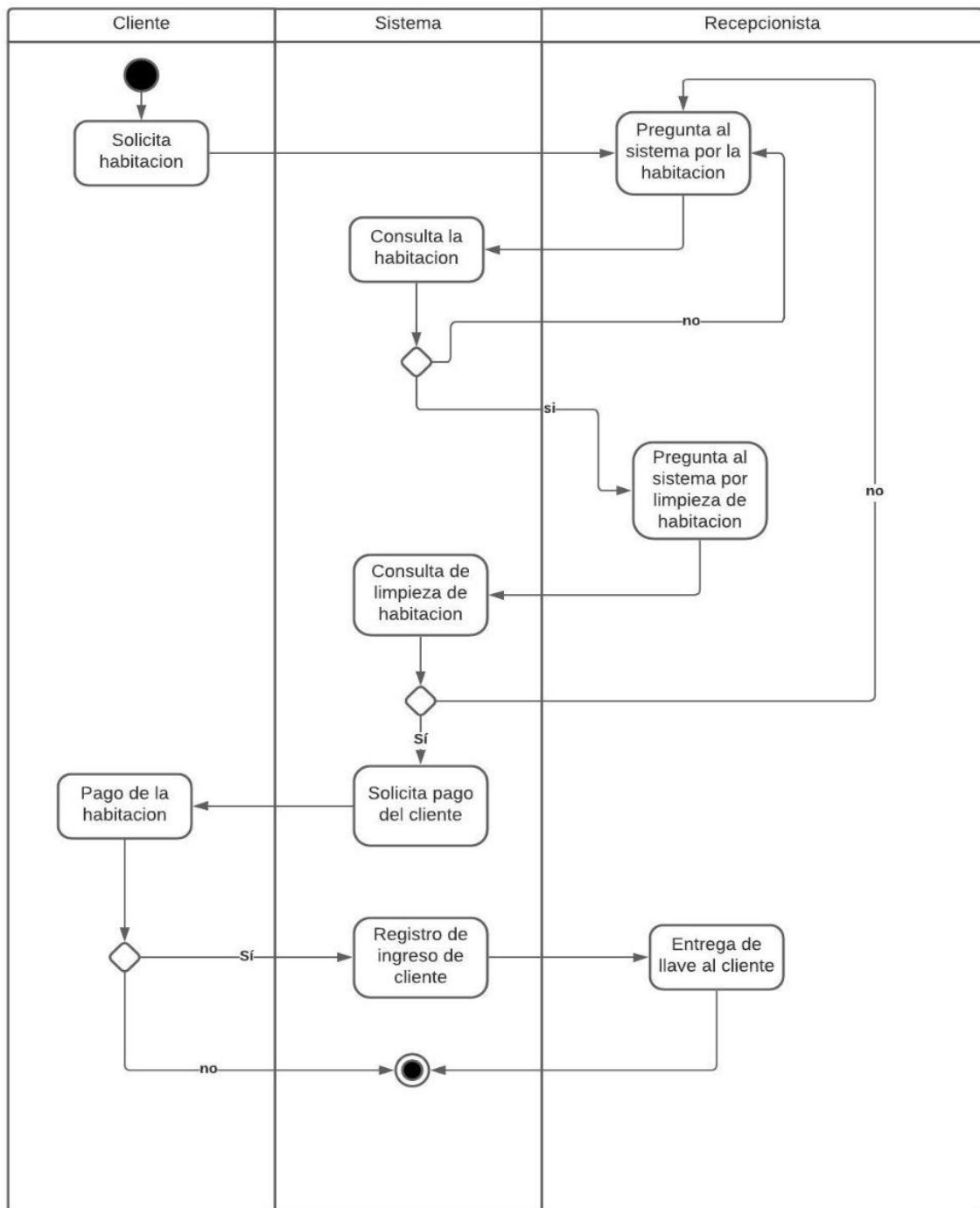


Ilustración 8: Diagrama de actividades con particiones o calles

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

Tabla 2: Trazabilidad de Requisitos

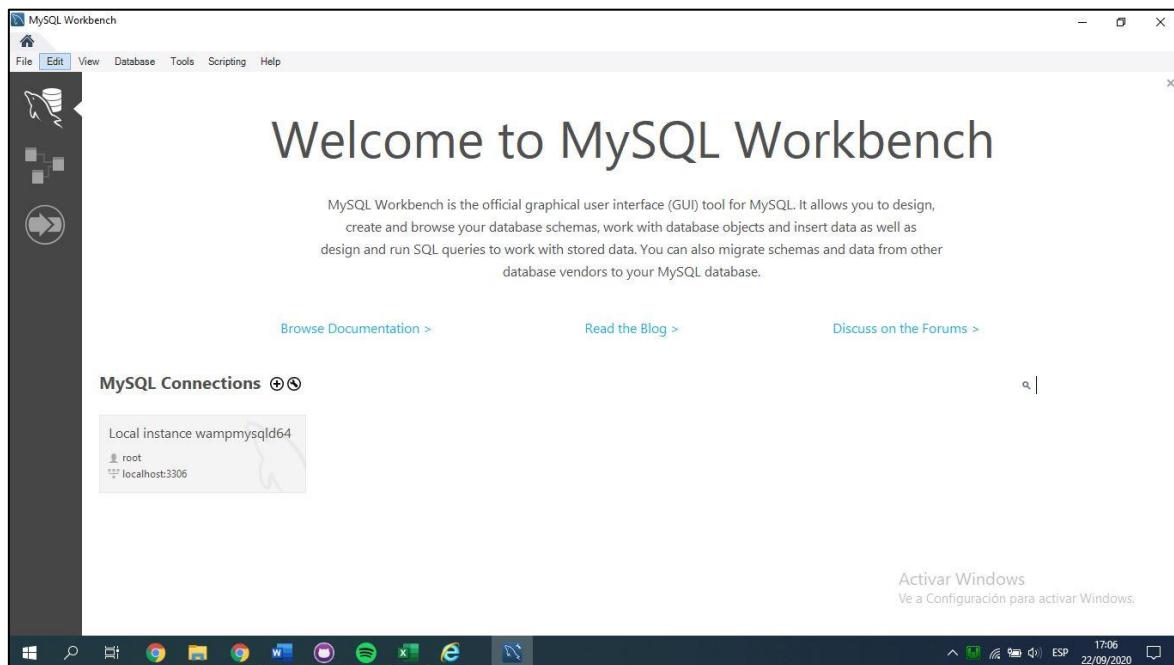


Ilustración 9: Requisitos no funcionales - Diego Tasaico - 1

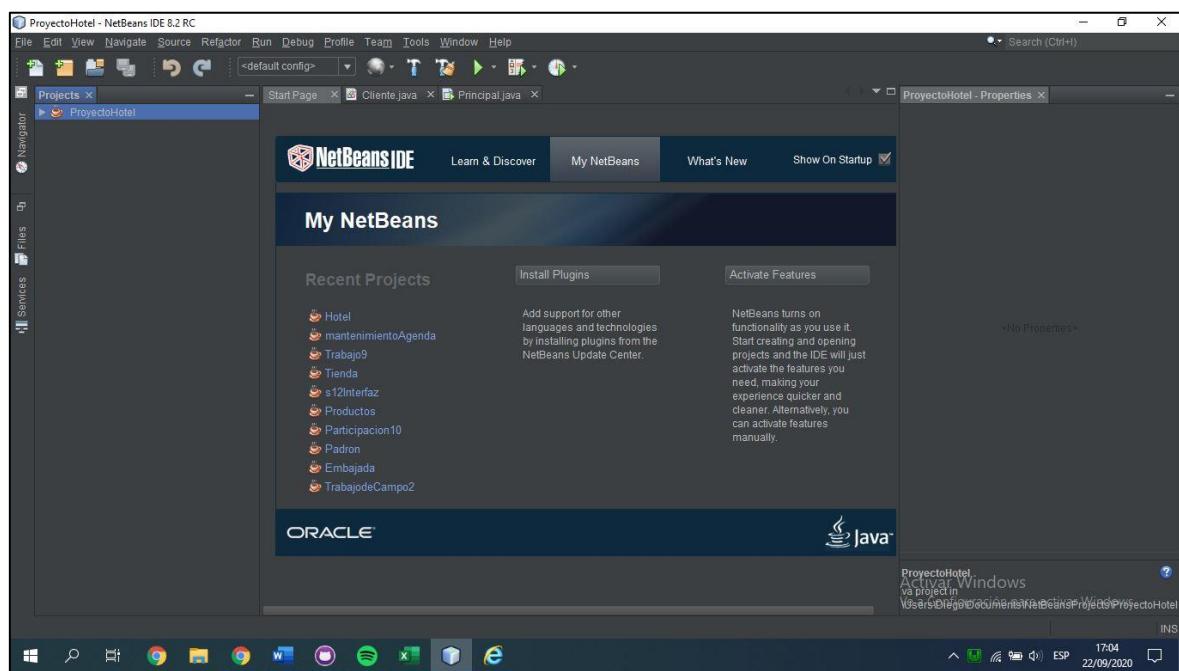


Ilustración 10: Requisitos no funcionales - Diego Tasaico - 2

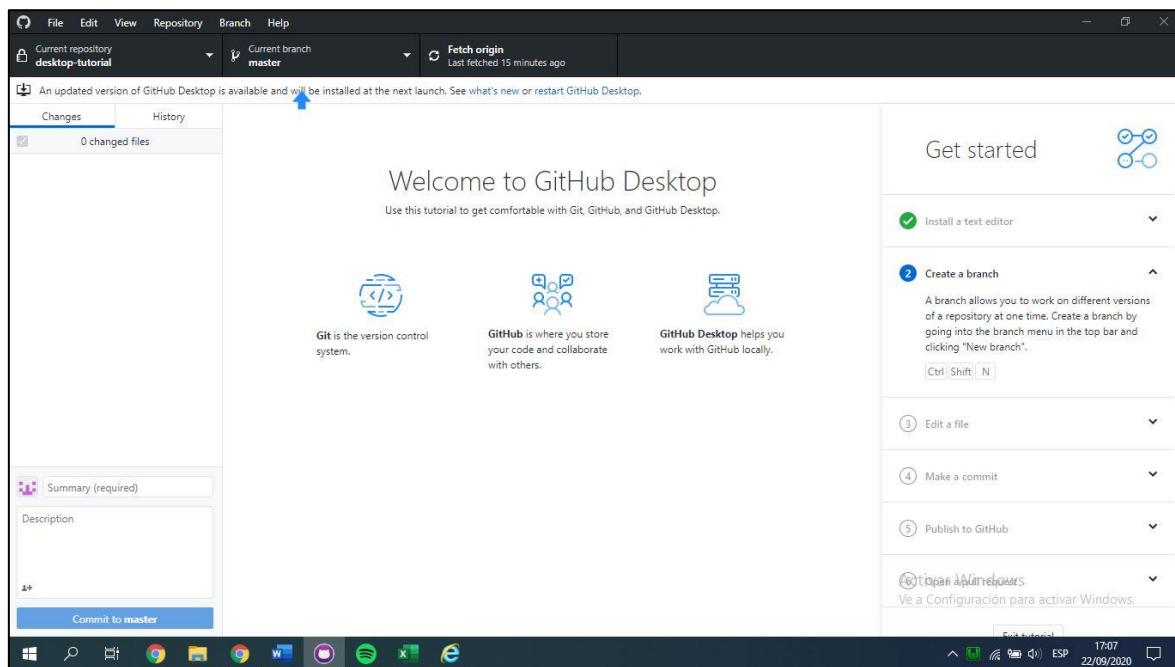


Ilustración 11: Requisitos no funcionales - Diego Tasaico - 3

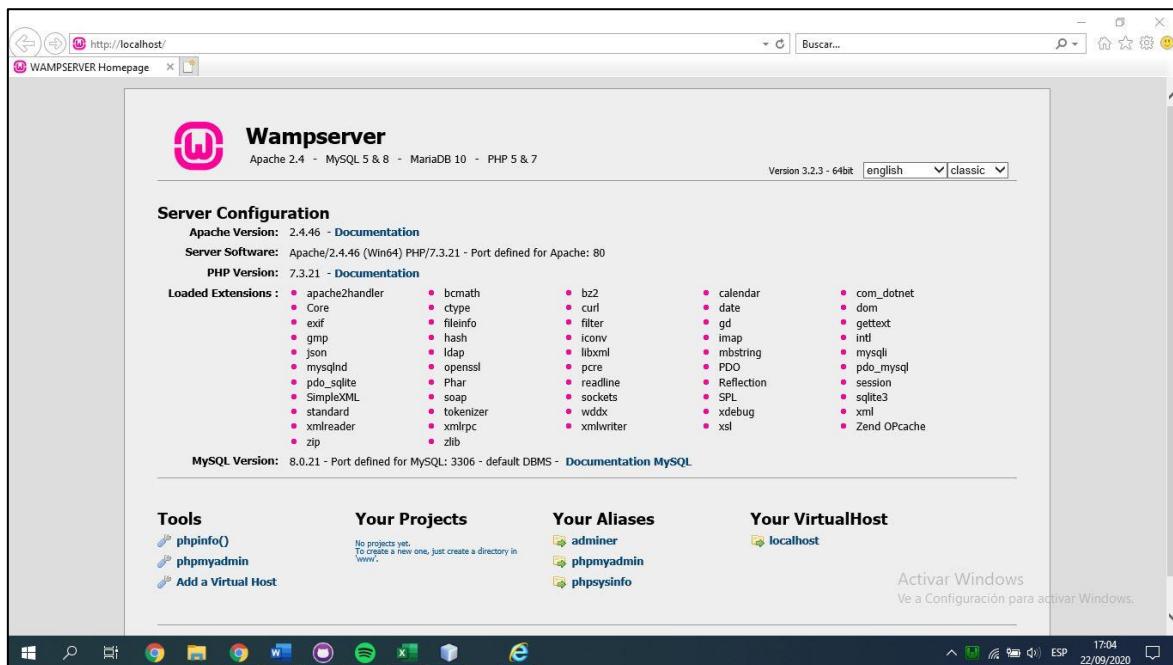


Ilustración 12: Requisitos no funcionales - Diego Tasaico - 4

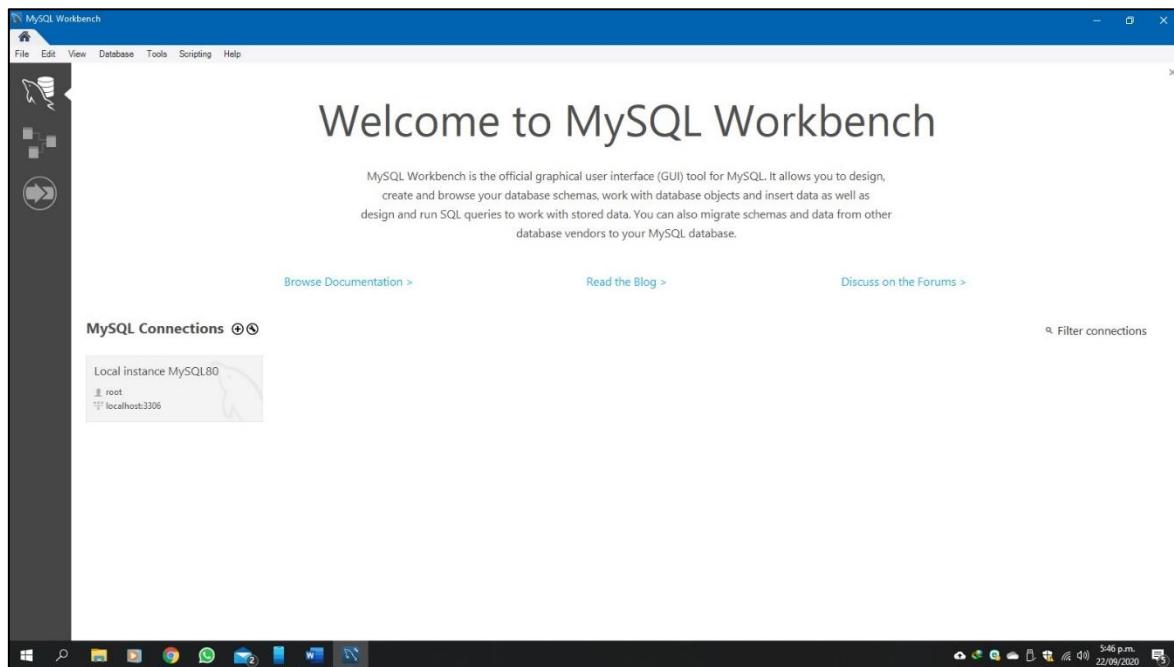


Ilustración 13: Requisitos no funcionales - Mauricio Villanueva - 1

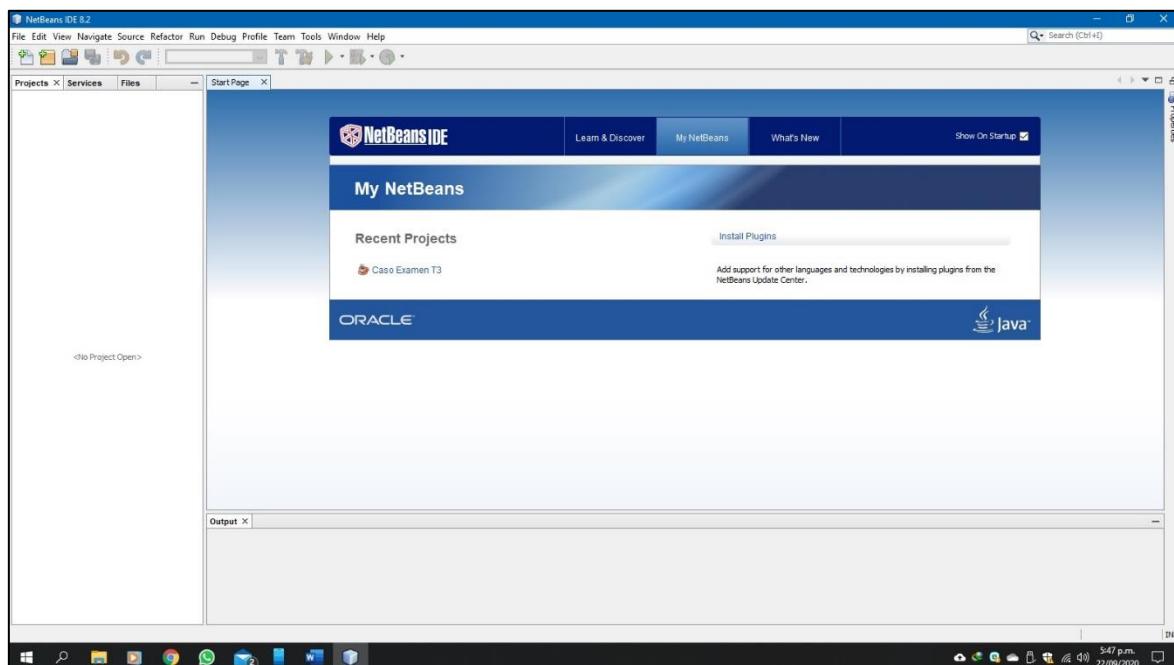


Ilustración 14: Requisitos no funcionales - Mauricio Villanueva - 2

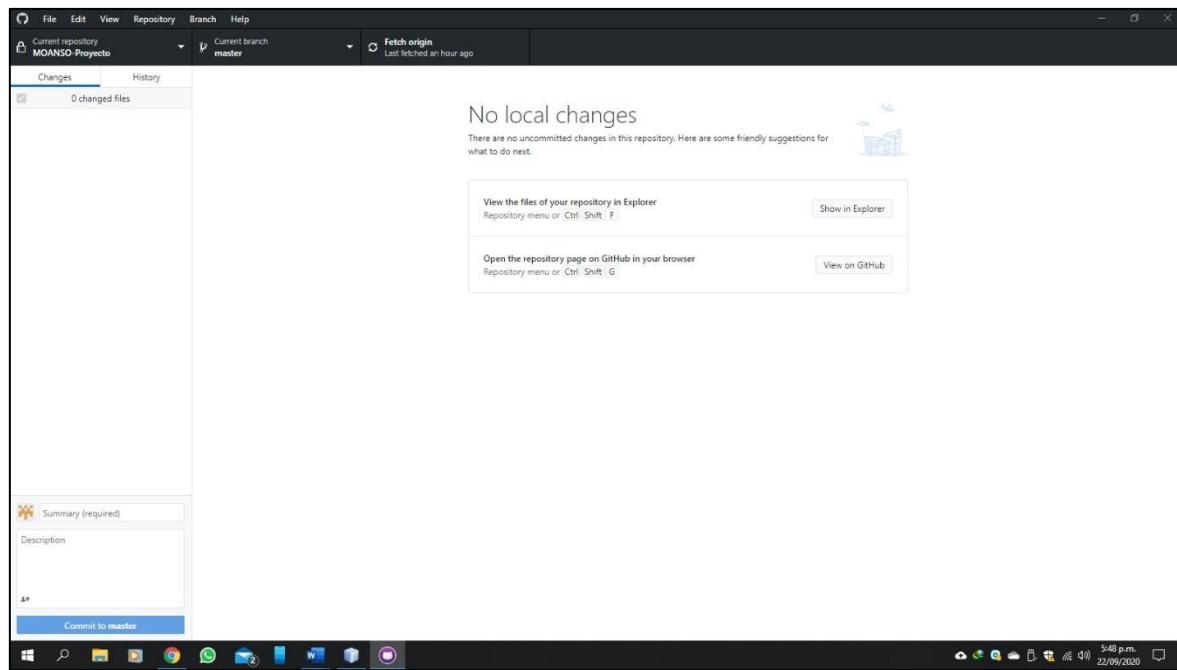


Ilustración 15: Requisitos no funcionales - Mauricio Villanueva - 3

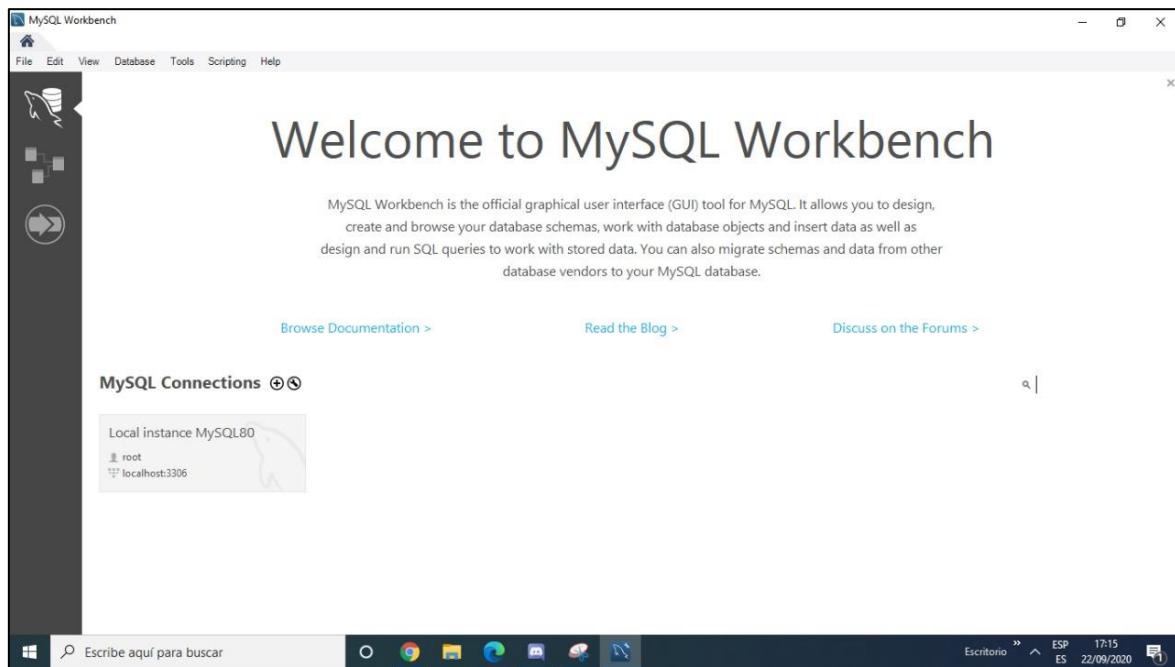


Ilustración 16: Requisitos no funcionales - Abihail Zapata - 1

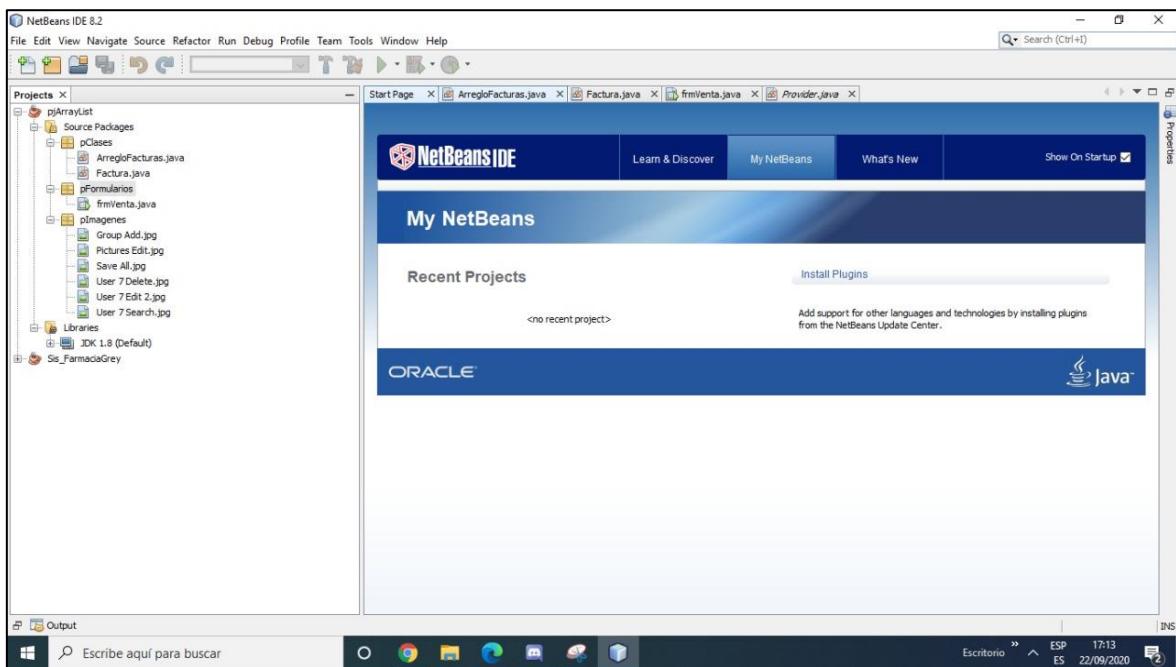


Ilustración 17: Requisitos no funcionales - Abihail Zapata - 2

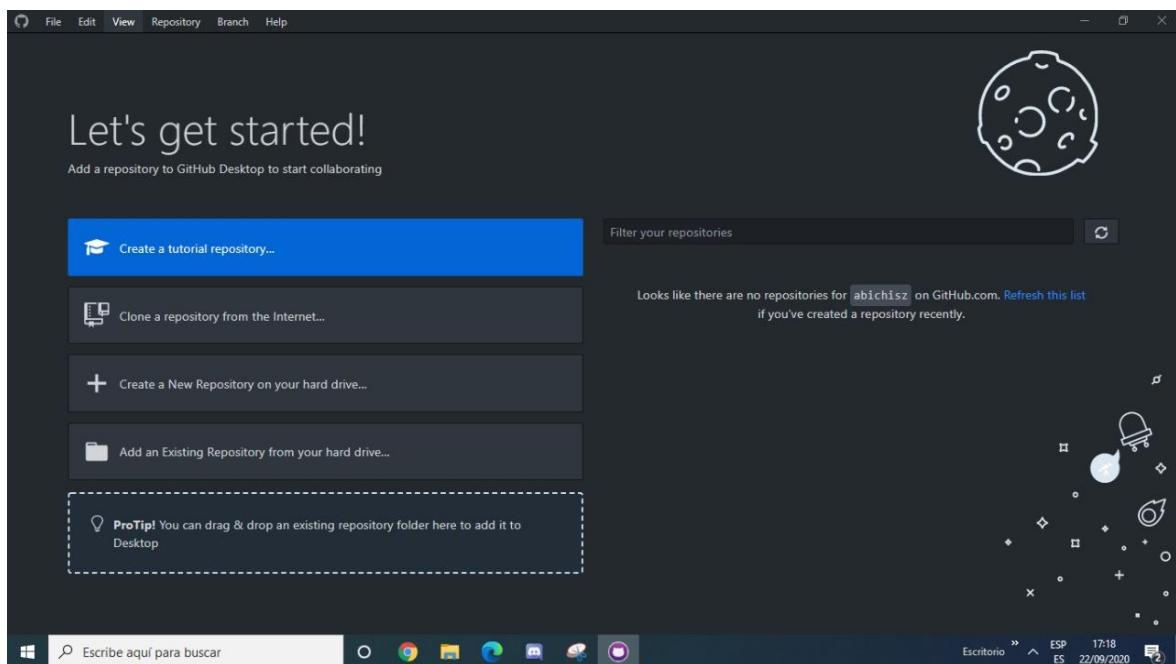


Ilustración 18: Requisitos no funcionales - Abihail Zapata - 3

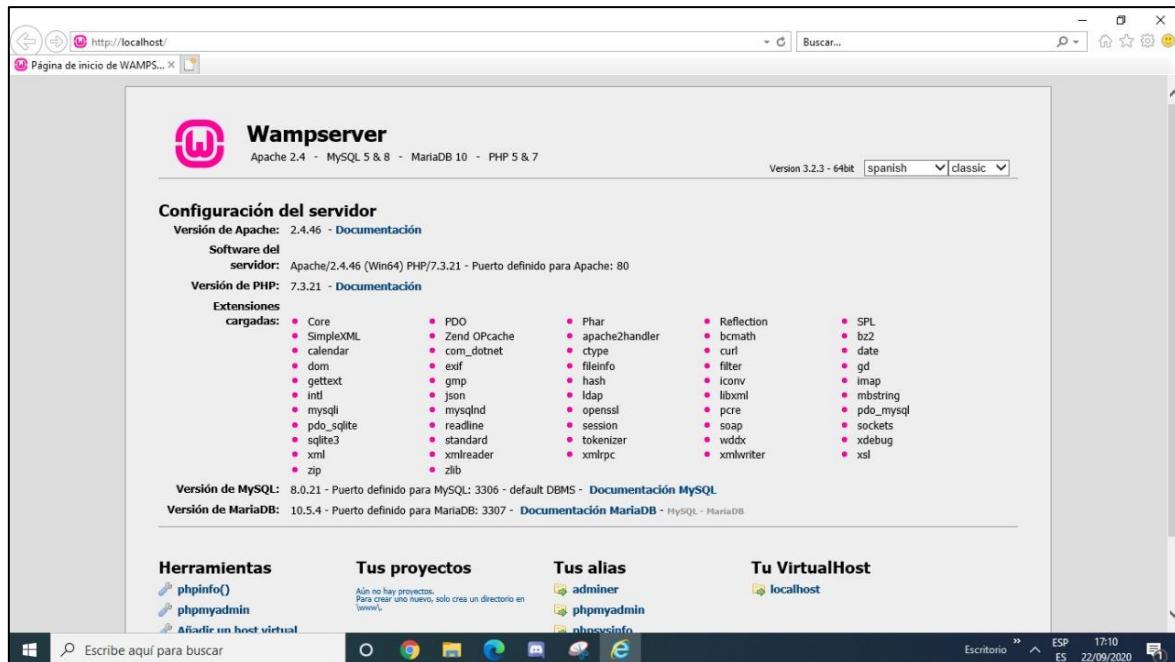


Ilustración 19: Requisitos no funcionales - Abihail Zapata - 4

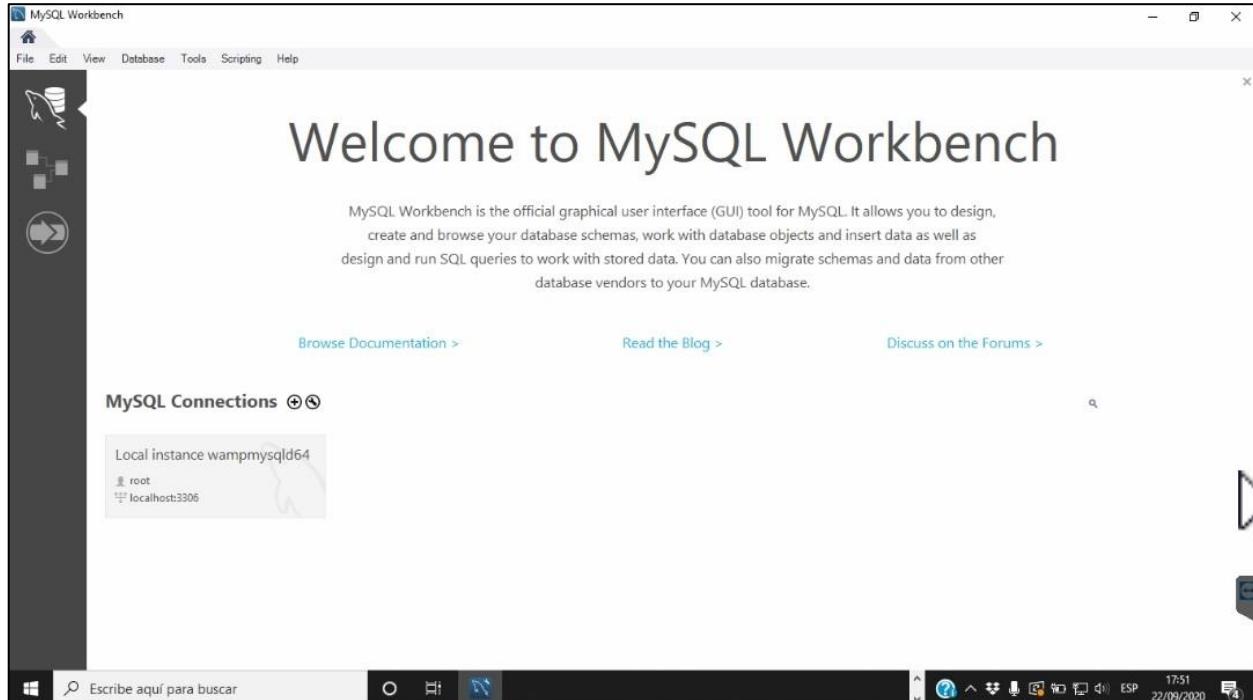


Ilustración 20: Requisitos no funcionales - Olenka Lazo - 1

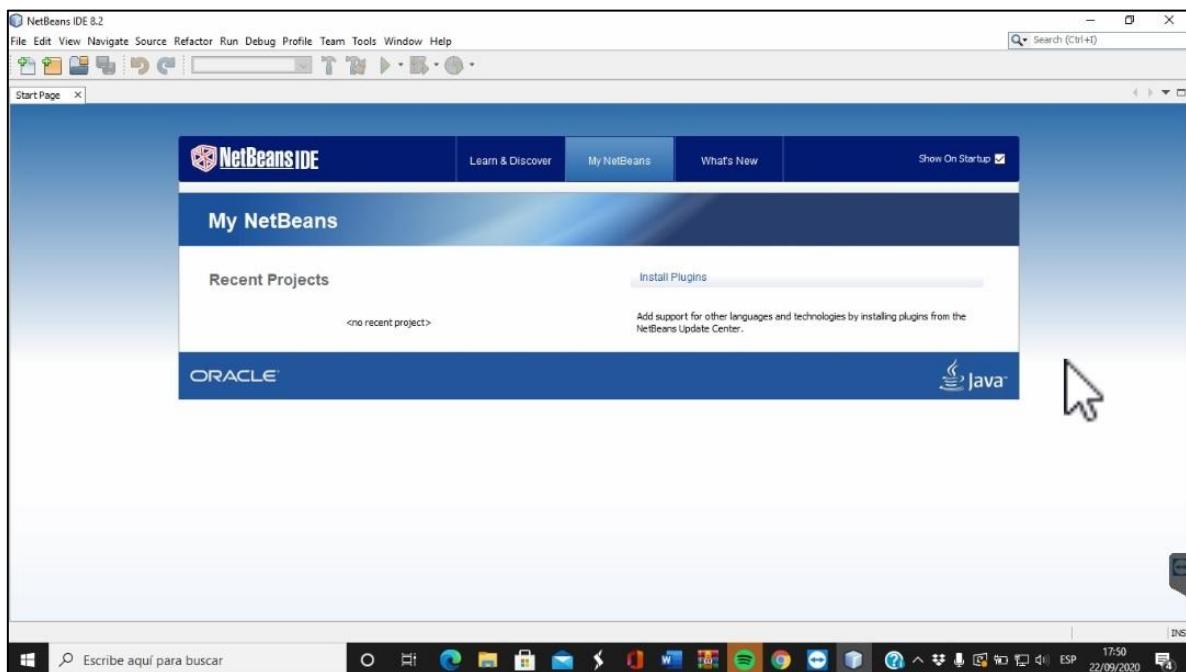


Ilustración 21: Requisitos no funcionales - Olenka Lazo - 2

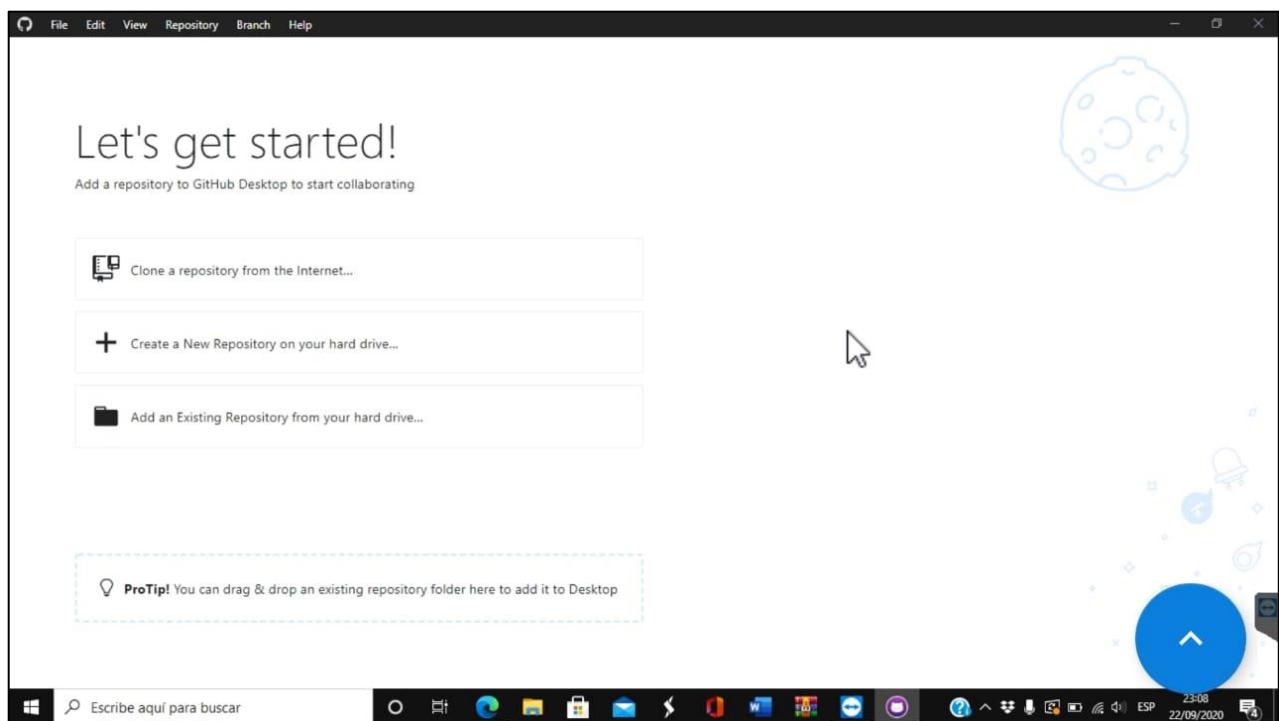


Ilustración 22: Requisitos no funcionales - Olenka Lazo - 3

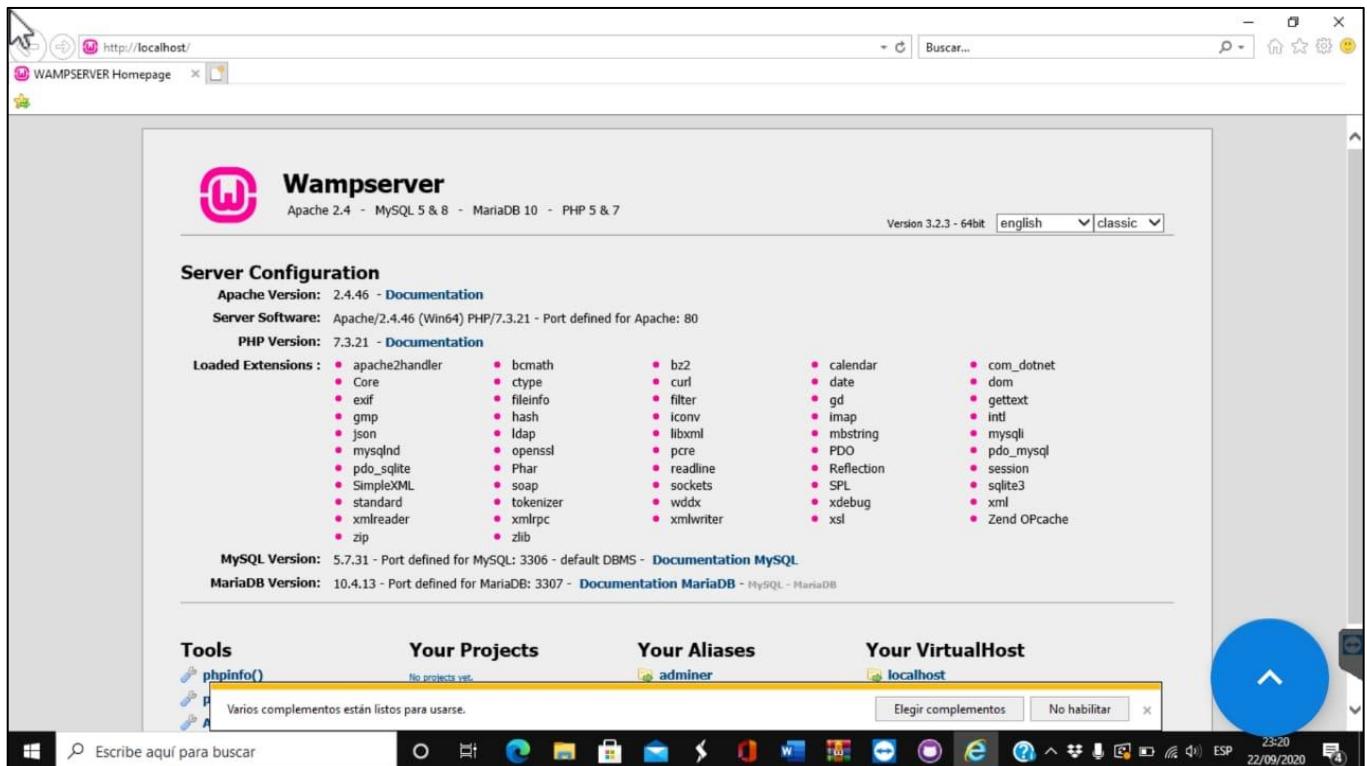


Ilustración 23: Requisitos no funcionales - Olenka Lazo - 4

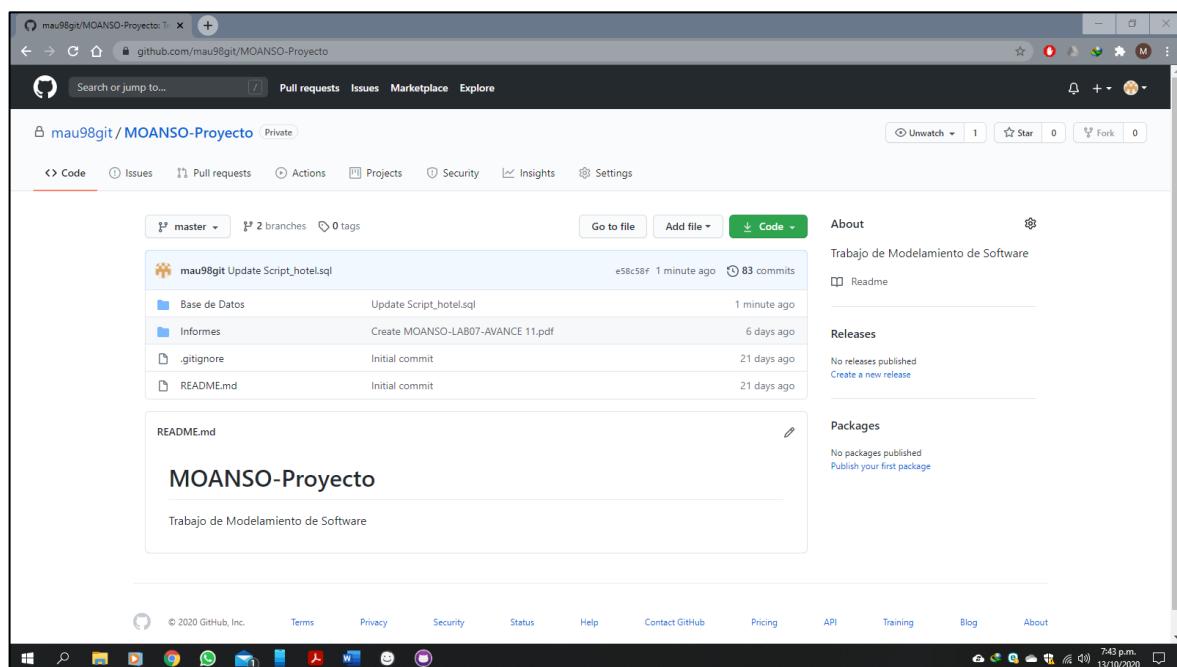


Ilustración 24: Repositorio GitHub - 1

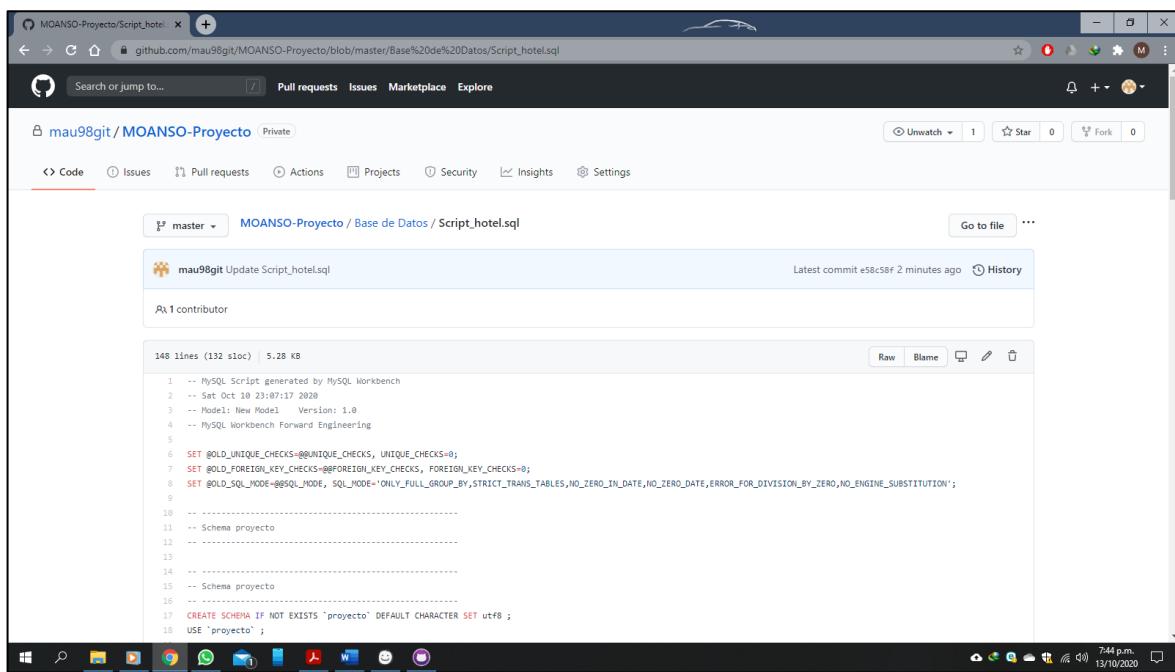


Ilustración 25: Repositorio GitHub - 2

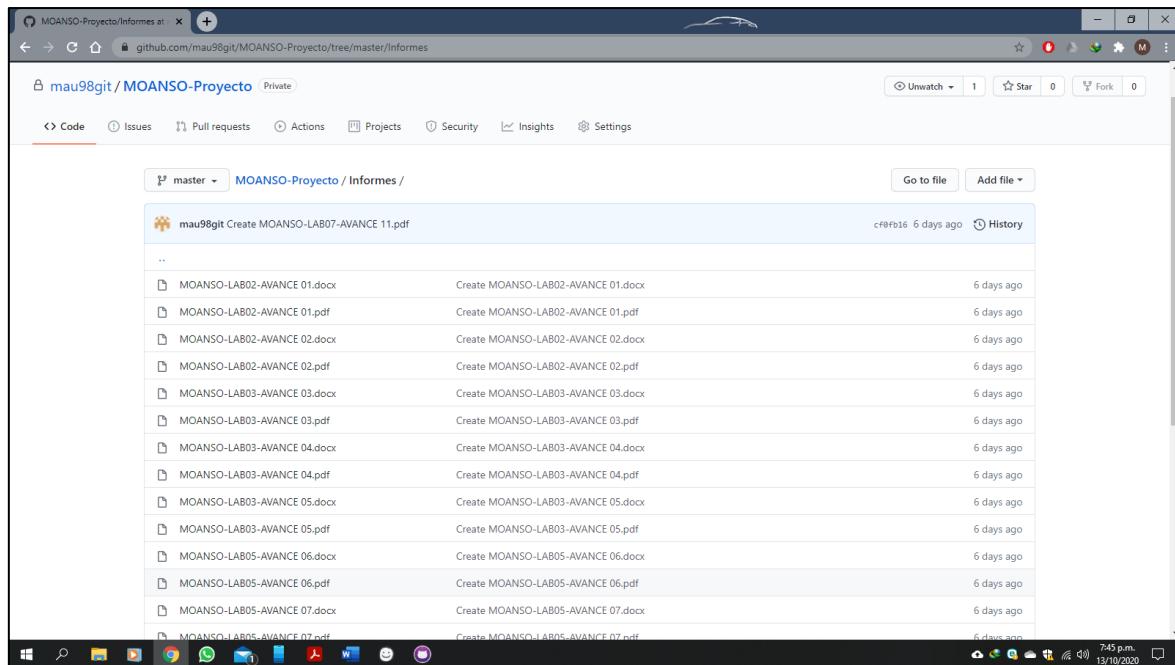


Ilustración 26: Repositorio GitHub - 3

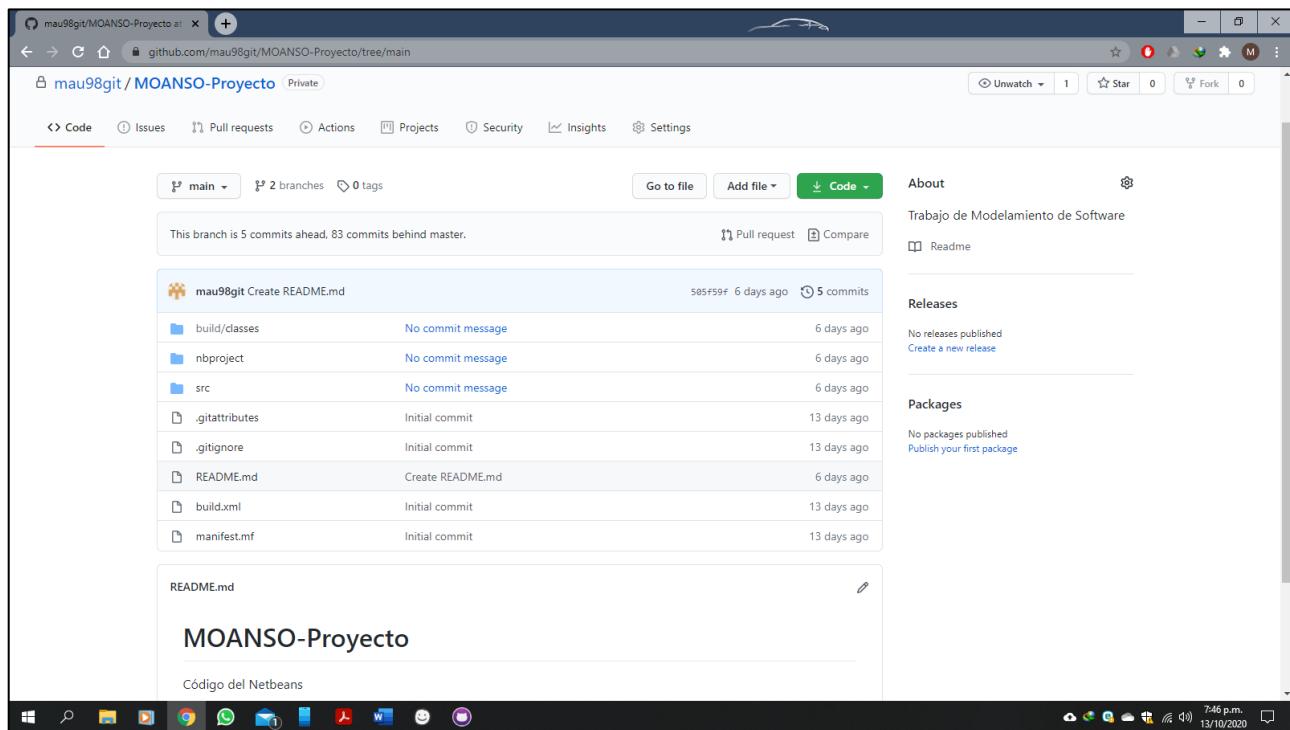


Ilustración 27: Repositorio GitHub - 4

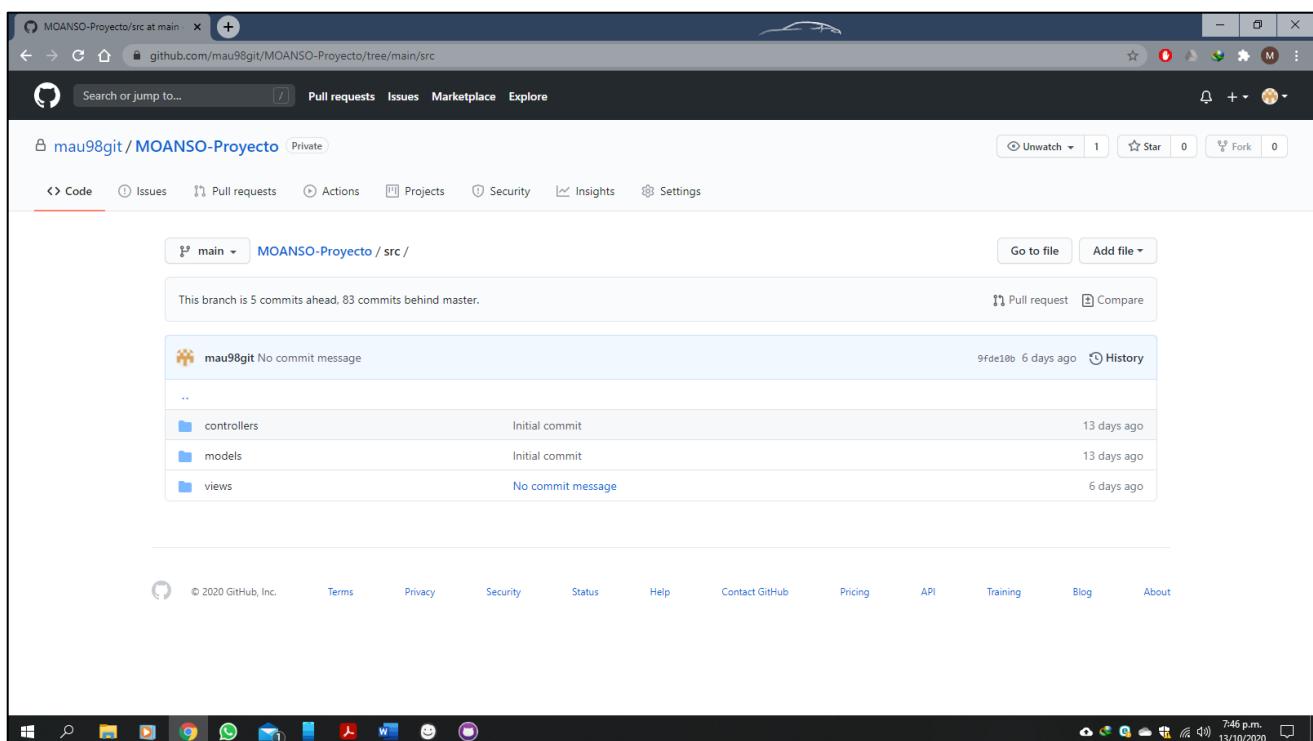


Ilustración 28: Repositorio GitHub - 5

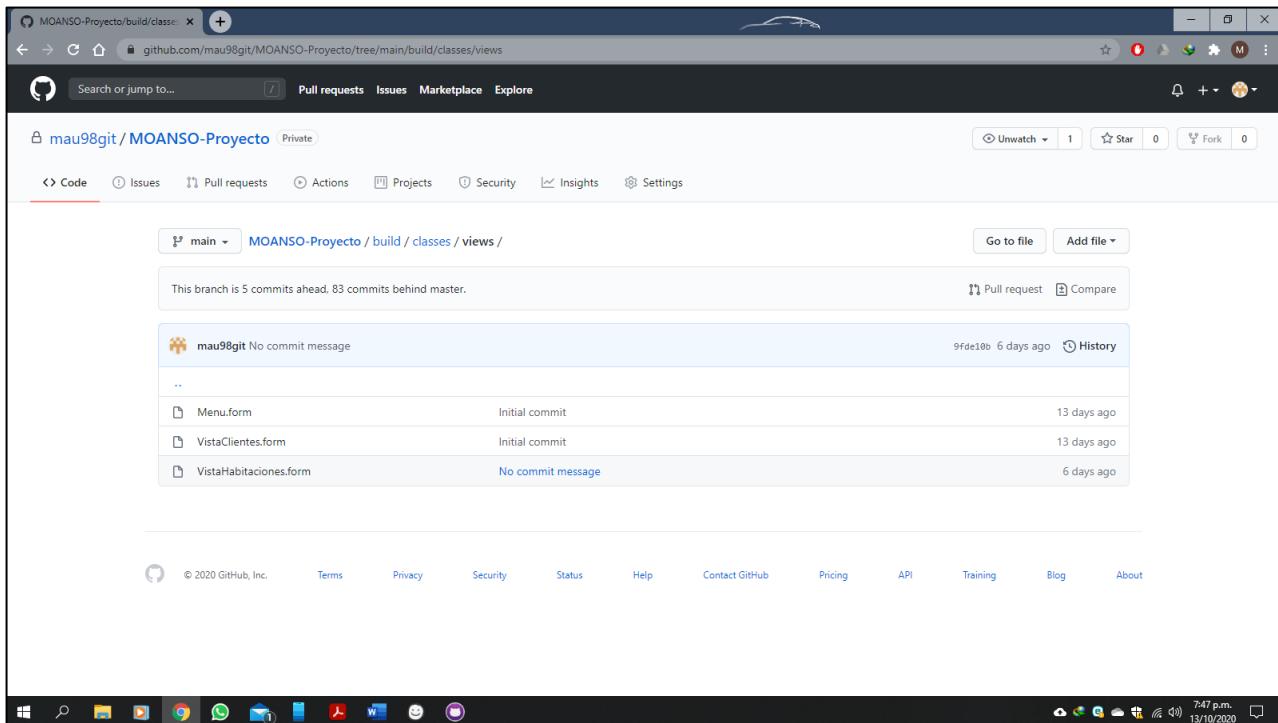


Ilustración 29: Repositorio GitHub - 6

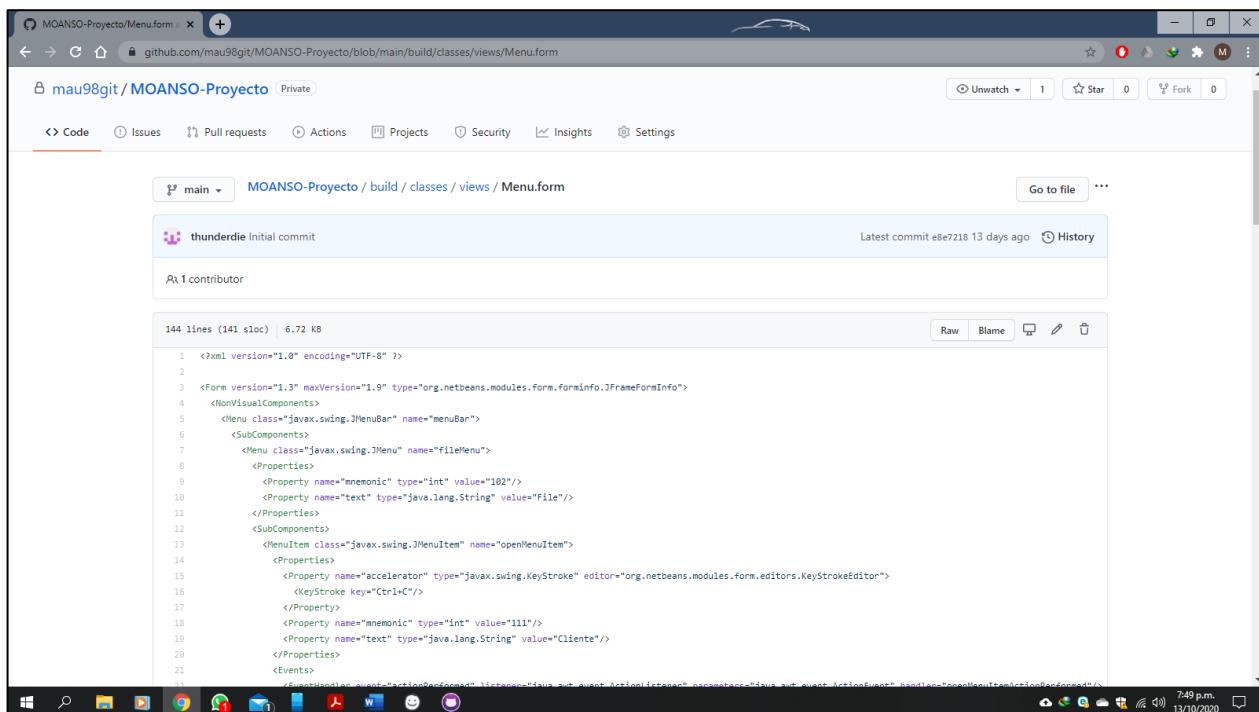


Ilustración 30: Repositorio GitHub - 7

Script de MySQL

```
-- MySQL Script generated by MySQL Workbench
-- 
-- Sat Oct 10 23:07:17 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`.`cliente`
```

```
CREATE TABLE IF NOT EXISTS `proyecto`.`cliente` (
  `idCliente` INT NOT NULL AUTO_INCREMENT,
  `nombres` VARCHAR(50) NOT NULL,
  `apellido_paterno` VARCHAR(45) NOT NULL,
```

```
`apellido_materno` VARCHAR(45) NOT NULL,  
 `tipo_documento` VARCHAR(20) NOT NULL,  
 `num_documento` VARCHAR(15) NOT NULL,  
 `celular` VARCHAR(15) NOT NULL,  
 `email` VARCHAR(45) NULL DEFAULT NULL,  
 PRIMARY KEY (`idCliente`))
```

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

-- 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(10,0) NOT NULL,  
 `estado` VARCHAR(25) NOT NULL,  
 `tipo_habitacion` VARCHAR(20) NOT NULL,  
 PRIMARY KEY (`idHabitacion`))
```

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

-- Table `proyecto`.`trabajador`

```
CREATE TABLE IF NOT EXISTS `proyecto`.`trabajador` (
```

```
`idTrabajador` INT NOT NULL AUTO_INCREMENT,  
`nombre` VARCHAR(60) NOT NULL,  
`apellido_paterno` VARCHAR(45) NOT NULL,  
`apellido_materno` VARCHAR(45) NOT NULL,  
`tipo_documento` VARCHAR(20) NOT NULL,  
`num_documento` VARCHAR(15) NOT NULL,  
`celular` VARCHAR(15) 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` TINYINT(1) NOT NULL,
```

PRIMARY KEY (`idTrabajador`),

UNIQUE INDEX `login_UNIQUE` (`login` ASC) VISIBLE,

UNIQUE INDEX `password_UNIQUE` (`password` ASC) VISIBLE)

ENGINE = InnoDB

AUTO_INCREMENT = 2

DEFAULT CHARACTER SET = utf8;

-- 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(10,0) 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_cliente`  
 FOREIGN KEY (`idCliente`)  
 REFERENCES `proyecto`.`cliente`(`idCliente`),  
 CONSTRAINT `fk_reserva_habitacion`  
 FOREIGN KEY (`idHabitacion`)  
 REFERENCES `proyecto`.`habitacion`(`idHabitacion`),  
 CONSTRAINT `fk_reserva_trabajador`  
 FOREIGN KEY (`idTrabajador`)  
 REFERENCES `proyecto`.`trabajador`(`idTrabajador`))  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8;
```

```
-- -----  
-- 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(10,0) NOT NULL,
`total_pago` DECIMAL(10,0) 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`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

```

-- -----
-- Table `proyecto`.`movimiento`

```

CREATE TABLE IF NOT EXISTS `proyecto`.`movimiento` (
`idMovimiento` INT NOT NULL AUTO_INCREMENT,
`nombreCliente` VARCHAR(45) NULL,
`apellido_p_Cliente` VARCHAR(45) NULL,
`apellido_m_Cliente` VARCHAR(45) NULL,
`habitacionCliente` VARCHAR(45) NULL,
`fecha_hora_ingreso` TIMESTAMP NULL,
`fecha_hora_salida` TIMESTAMP NULL,
`estado` TINYINT(1) NULL,
PRIMARY KEY (`idMovimiento`))
```

ENGINE = InnoDB;

SET SQL_MODE=@OLD_SQL_MODE;

SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;

SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

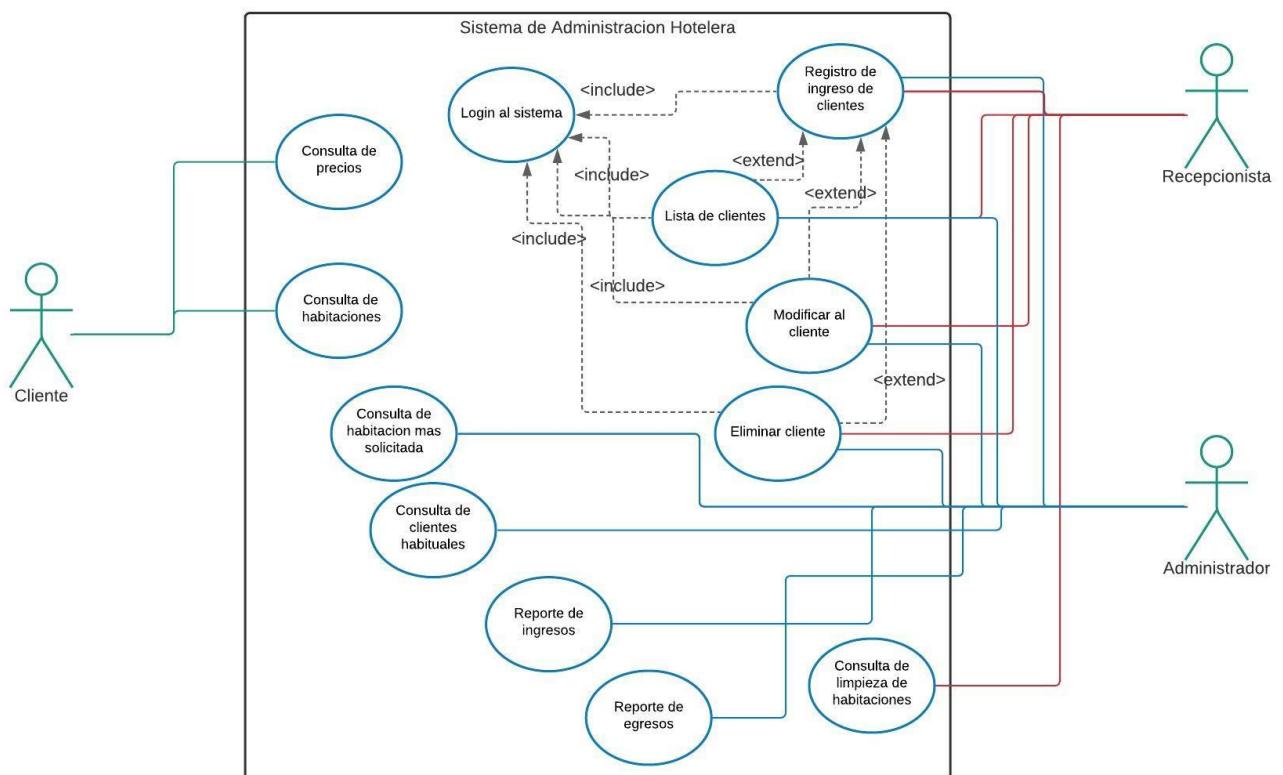


Ilustración 31: Diagrama de Casos de uso Relacionado - General

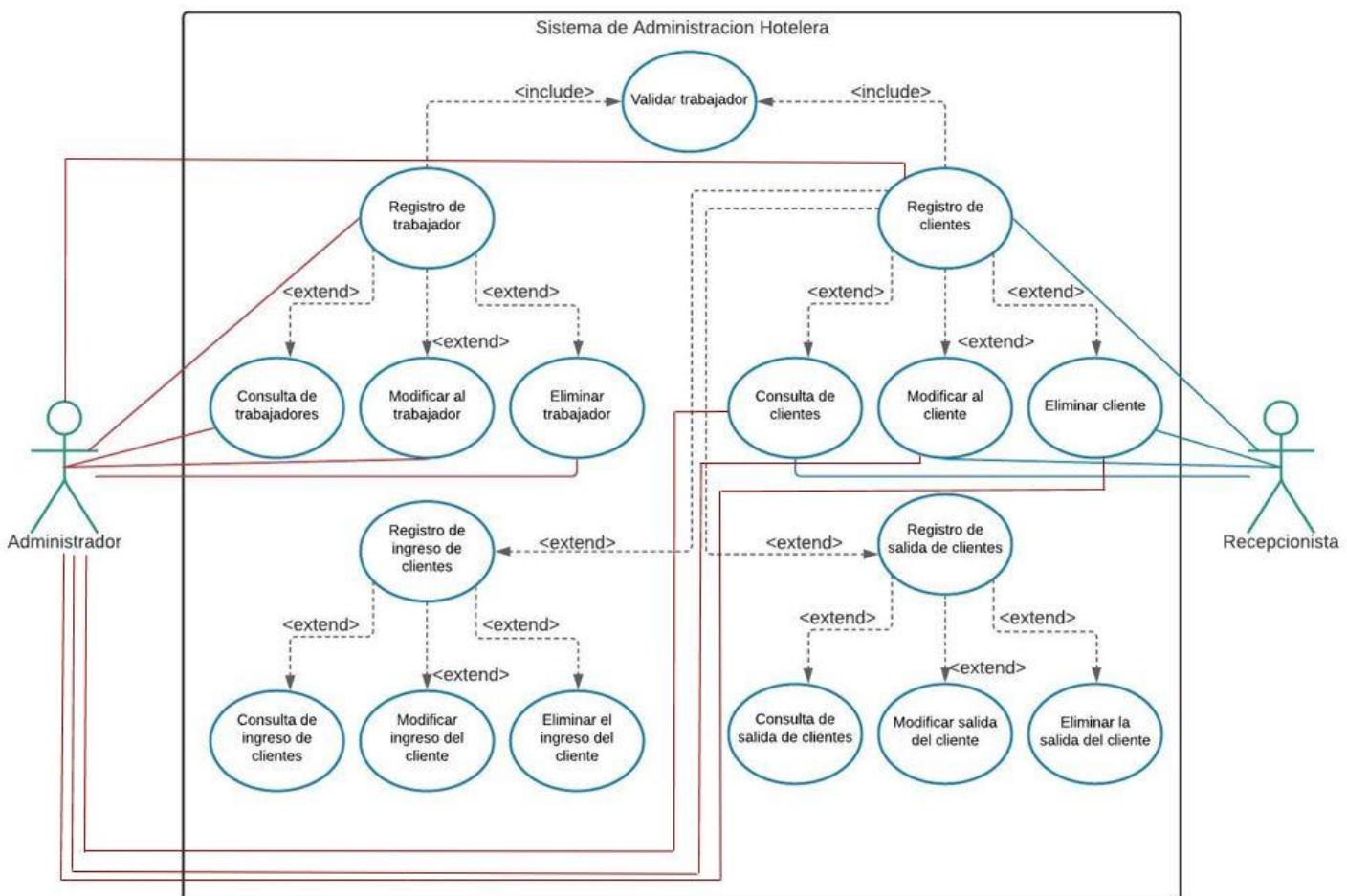
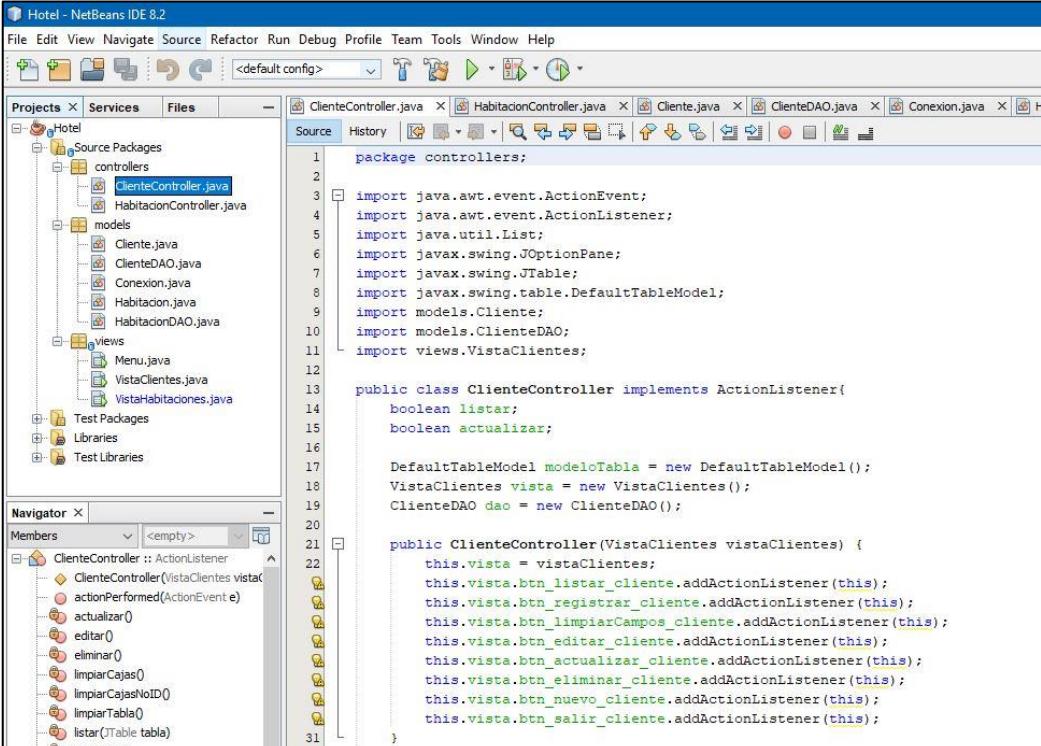


Ilustración 32: Diagrama de Casos de uso Relacionado – Versión 1



```

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;

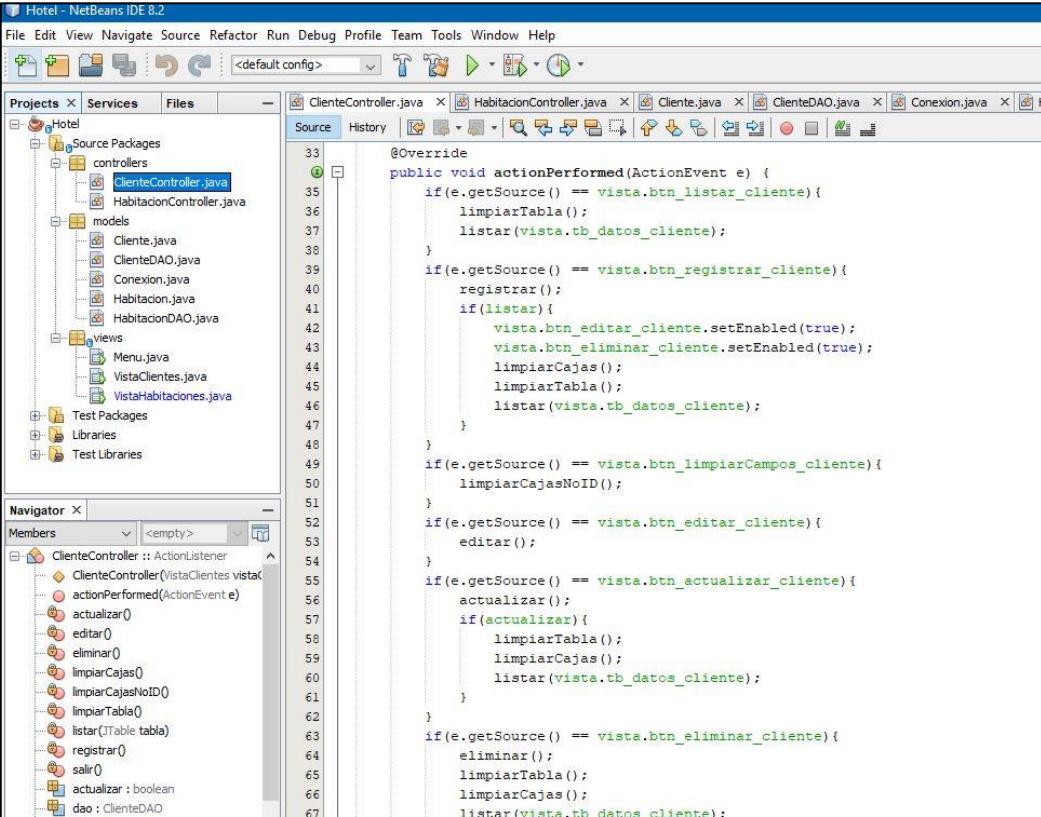
public class ClienteController implements ActionListener{
    boolean listar;
    boolean actualizar;

    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){
            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);
        }
    }
}

```

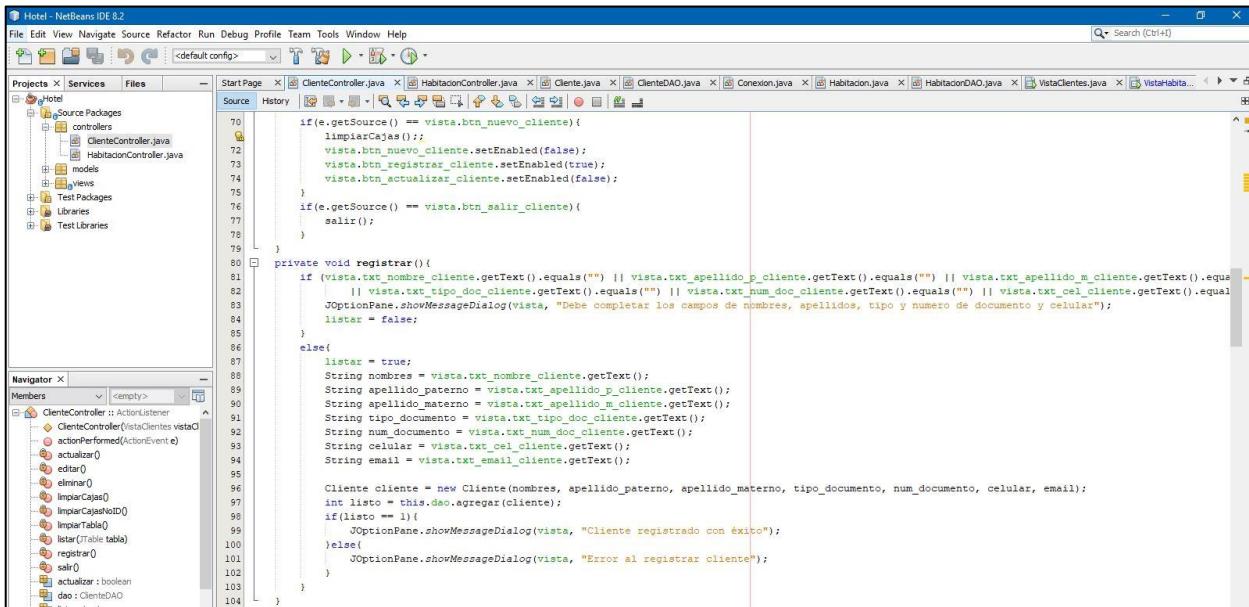


```

@Override
public void actionPerformed(ActionEvent e) {
    if(e.getSource() == vista.btn_listar_cliente){
        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);
    }
}

```

Ilustración 33: Código de programación del software



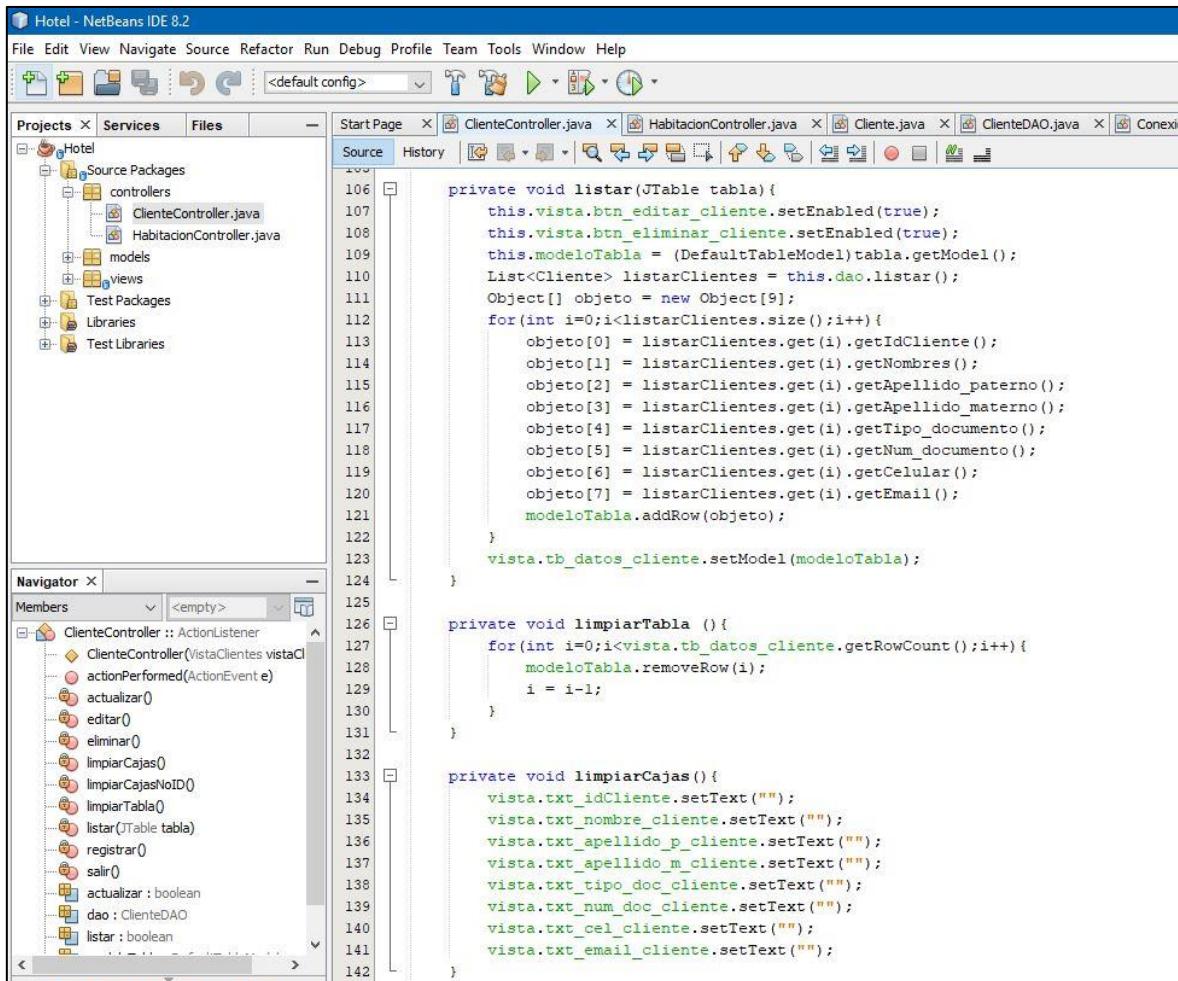
```

Hotel - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Source History Start Page ClienteController.java HabitacionController.java Cliente.java ClienteDAO.java Conexion.java Habitacion.java HabitacionDAO.java VistaClientes.java VistaHabitacion.java
Projects Services Files
Source Packages controllers ClienteController.java HabitacionController.java Cliente.java ClienteDAO.java Conexion.java Habitacion.java HabitacionDAO.java VistaClientes.java VistaHabitacion.java
Test Packages Libraries Test Libraries
if(e.getSource() == vista.btn_nuevo_cliente){
    limpiarCajas();
    vista.btn_nuevo_cliente.setEnabled(false);
    vista.btn_registrar_cliente.setEnabled(true);
    vista.btn_actualizar_cliente.setEnabled(false);
}
if(e.getSource() == vista.btn_salir_cliente){
    salir();
}
}

private void registrar(){
    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("")){
        JOptionPane.showMessageDialog(vista, "Debe completar los campos de nombres, apellidos, tipo y numero de documento y celular");
        lista = false;
    }
} else{
    lista = true;
    String nombres = Vista.txt_nombre_cliente.getText();
    String apellido_paterno = Vista.txt_apellido_p_cliente.getText();
    String apellido_materno = Vista.txt_apellido_m_cliente.getText();
    String tipo_documento = Vista.txt_tipo_doc_cliente.getText();
    String num_documento = Vista.txt_num_doc_cliente.getText();
    String celular = Vista.txt_cel_cliente.getText();
    String email = Vista.txt_email_cliente.getText();

    Cliente cliente = new Cliente(nombres, apellido_paterno, apellido_materno, tipo_documento, num_documento, celular, email);
    int lista = this.dao.agregar(cliente);
    if(lista == 1){
        JOptionPane.showMessageDialog(vista, "Cliente registrado con éxito");
    } else{
        JOptionPane.showMessageDialog(vista, "Error al registrar cliente");
    }
}
}

```



```

Hotel - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Source History Start Page ClienteController.java HabitacionController.java Cliente.java ClienteDAO.java Conexion.java
Projects Services Files
Source Packages controllers ClienteController.java HabitacionController.java Cliente.java ClienteDAO.java Conexion.java
Test Packages Libraries Test Libraries
private void listar(JTable tabla){
    this.vista.btn_editar_cliente.setEnabled(true);
    this.vista.btn_eliminar_cliente.setEnabled(true);
    this.modeloTabla = (DefaultTableModel)tabla.getModel();
    List<Cliente> listarClientes = this.dao.listar();
    Object[] objeto = new Object[9];
    for(int i=0;i<listarClientes.size();i++){
        objeto[0] = listarClientes.get(i).getIdCliente();
        objeto[1] = listarClientes.get(i).getNombres();
        objeto[2] = listarClientes.get(i).getApellido_paterno();
        objeto[3] = listarClientes.get(i).getApellido_materno();
        objeto[4] = listarClientes.get(i).getTipo_documento();
        objeto[5] = listarClientes.get(i).getNum_documento();
        objeto[6] = listarClientes.get(i).getCelular();
        objeto[7] = listarClientes.get(i).getEmail();
        modeloTabla.addRow(objeto);
    }
    vista.tb_datos_cliente.setModel(modeloTabla);
}

private void limpiarTabla (){
    for(int i=0;i<vista.tb_datos_cliente.getRowCount();i++){
        modeloTabla.removeRow(i);
        i = i-1;
    }
}

private void limpiarCajas(){
    vista.txt_idCliente.setText("");
    vista.txt_nombre_cliente.setText("");
    vista.txt_apellido_p_cliente.setText("");
    vista.txt_apellido_m_cliente.setText("");
    vista.txt_tipo_doc_cliente.setText("");
    vista.txt_num_doc_cliente.setText("");
    vista.txt_cel_cliente.setText("");
    vista.txt_email_cliente.setText("");
}

```

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 W G D E 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
Cliente.java
public class Cliente {

    private int idCliente;
    private String nombres;
    private String apellido_paterno;
    private String apellido_materno;
    private String tipo_documento;
    private String num_documento;
    private String celular;
    private String email;

    public Cliente() {
    }

    public Cliente(int idCliente, String nombres, String apellido_paterno, String apellido_materno, String tipo_documento, String num_documento, String celular, String email) {
        this.idCliente = idCliente;
        this.nombres = nombres;
        this.apellido_paterno = apellido_paterno;
        this.apellido_materno = apellido_materno;
        this.tipo_documento = tipo_documento;
        this.num_documento = num_documento;
        this.celular = celular;
        this.email = email;
    }

    public Cliente(String nombres, String apellido_paterno, String apellido_materno, String tipo_documento, String num_documento, String celular, String email) {
        this.nombres = nombres;
        this.apellido_paterno = apellido_paterno;
        this.apellido_materno = apellido_materno;
        this.tipo_documento = tipo_documento;
        this.num_documento = num_documento;
        this.celular = celular;
        this.email = email;
    }
}

```

Navigator X Members <empty>

Cliente

- Cliente()
- Cliente(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)
- getApellido_materno(): String
- getApellido_paterno(): String
- getCellar(): String
- getEmail(): String
- getIdCliente(): int
- getNombres(): String
- getNum_documento(): String
- getTipo_documento(): String
- setApellido_materno(String apellido_materno)
- setApellido_paterno(String apellido_paterno)
- setCellar(String cellar)

Hotel - NetBeans IDE 8.2

```

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Source History <default config> T W G D E 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
Cliente.java
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;
}

```

Navigator X Members <empty>

Cliente

- Cliente()
- Cliente(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)
- getApellido_materno(): String
- getApellido_paterno(): String
- getCellar(): String
- getEmail(): String
- getIdCliente(): int
- getNombres(): String
- getNum_documento(): String
- getTipo_documento(): String
- setApellido_materno(String apellido_materno)
- setApellido_paterno(String apellido_paterno)
- setCellar(String cellar)

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

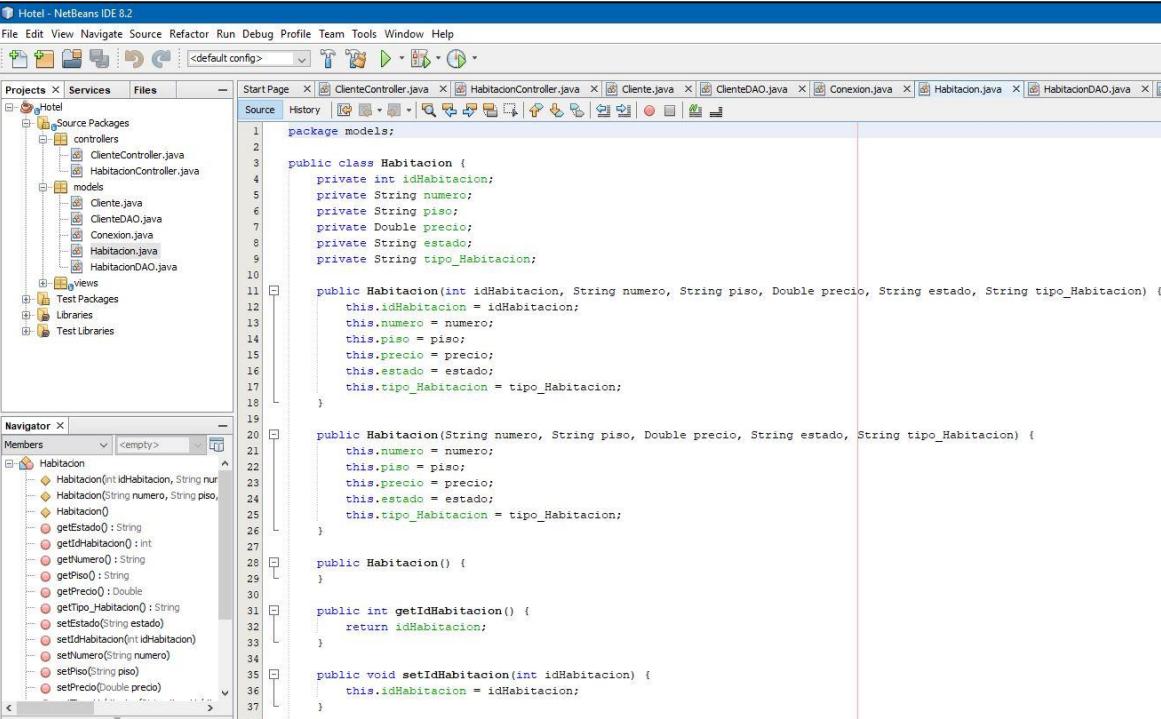
Projects X Services Files — Start Page X ClienteController.java X HabitacionController.java X Cliente.java X ClienteDAO.java X Conexion.java X

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



```

package models;

public class Habitacion {
    private int idHabitacion;
    private String numero;
    private String piso;
    private Double precio;
    private String estado;
    private String tipo_Habitacion;

    public Habitacion(int idHabitacion, String numero, Double precio, String estado, String tipo_Habitacion) {
        this.idHabitacion = idHabitacion;
        this.numero = numero;
        this.piso = piso;
        this.precio = precio;
        this.estado = estado;
        this.tipo_Habitacion = tipo_Habitacion;
    }

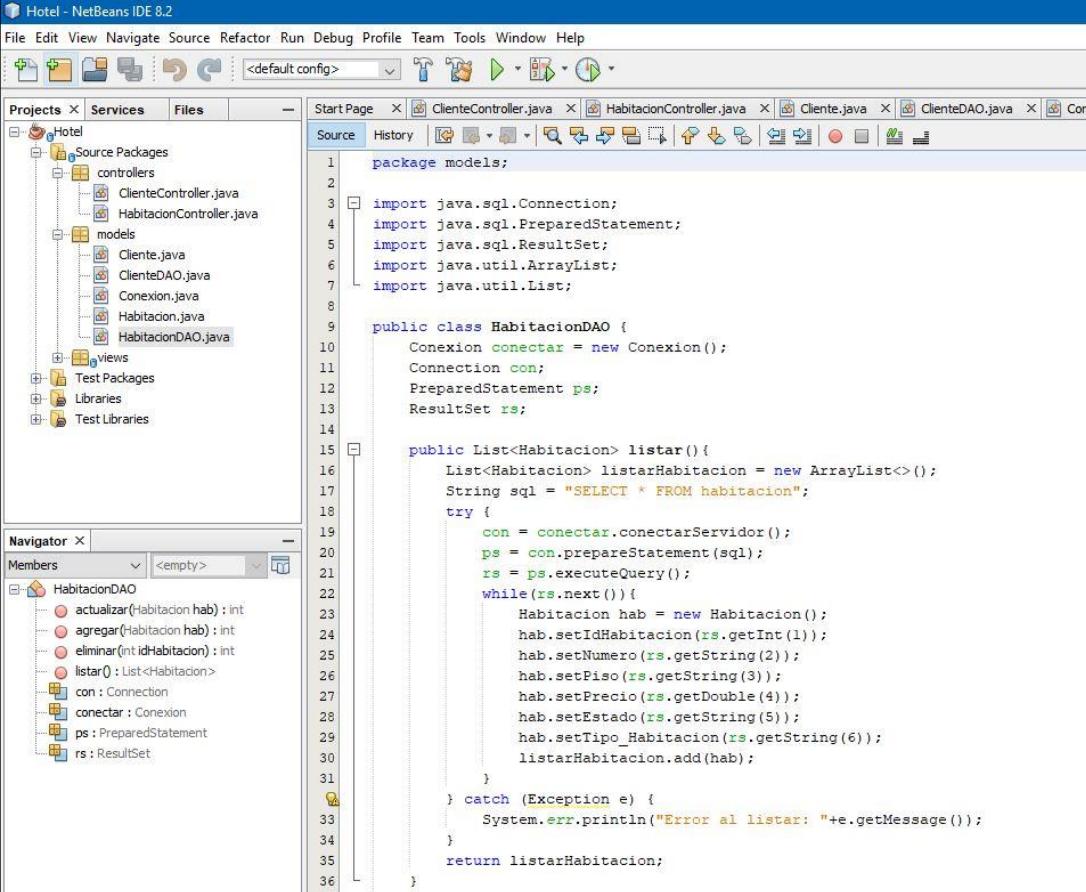
    public Habitacion(String numero, String piso, Double precio, String estado, String tipo_Habitacion) {
        this.numero = numero;
        this.piso = piso;
        this.precio = precio;
        this.estado = estado;
        this.tipo_Habitacion = tipo_Habitacion;
    }

    public Habitacion() {
    }

    public int getIdHabitacion() {
        return idHabitacion;
    }

    public void setIdHabitacion(int idHabitacion) {
        this.idHabitacion = idHabitacion;
    }
}

```



```

package models;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

public class HabitacionDAO {
    Conexion conectar = new Conexion();
    Connection con;
    PreparedStatement ps;
    ResultSet rs;

    public List<Habitacion> listar(){
        List<Habitacion> listarHabitacion = new ArrayList<>();
        String sql = "SELECT * FROM habitacion";
        try {
            con = conectar.conectarServidor();
            ps = con.prepareStatement(sql);
            rs = ps.executeQuery();
            while(rs.next()){
                Habitacion hab = new Habitacion();
                hab.setIdHabitacion(rs.getInt(1));
                hab.setNumero(rs.getString(2));
                hab.setPiso(rs.getString(3));
                hab.setPrecio(rs.getDouble(4));
                hab.setEstado(rs.getString(5));
                hab.setTipo_Habitacion(rs.getString(6));
                listarHabitacion.add(hab);
            }
        } catch (Exception e) {
            System.err.println("Error al listar: "+e.getMessage());
        }
        return listarHabitacion;
    }
}

```

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;
    }
}

```

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 ➤ El sistema registra el acceso del trabajador

		<ul style="list-style-type: none"> ➤ 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

Tabla 3: Especificación de Casos de Uso - 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

Tabla 4: Especificación de Casos de Uso - 2

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

Tabla 5: Especificación de Casos de Uso - 3

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

Tabla 6: Especificación de Casos de Uso - 4

Caso de uso	Datos de Entrada normal	Resultado Esperado normal	Datos de entrada anómalo	Resultado esperado anómalo
Registro de cliente	Nombres Apellido Paterno Apellido Materno Tipo de Documento Número de Documento Celular Email	Cliente registrado con éxito	-Nombre en espacios -Apellido en espacios -Tipo de documento incorrecto -Número de documento inexistente -Falta dígitos	Debe completar los campos de nombres, apellidos, tipo y número de documento y celular.
Eliminar cliente	No tiene datos	Cliente eliminado con éxito		Debe seleccionar una fila
Modificar cliente	Nombres Apellido Paterno Apellido Materno Tipo de Documento Número de Documento Celular Email	Cliente actualizado con éxito	-Nombre en espacios -Apellido en espacios -Tipo de documento incorrecto -Número de documento inexistente -Falta dígitos	Debe completar los campos de nombres, apellidos, tipo y número de documento y celular.
Listar cliente	No tiene datos	Cliente listado con éxito	No tiene datos	No hay clientes para listar
Registrar trabajador	Nombres Apellido Paterno Apellido Materno Tipo de Documento Número de Documento Celular Email Sueldo Acceso Login Password Estado	Trabajador registrado con éxito	-Nombre en espacios -Apellido en espacios -Tipo de documento incorrecto -Número de documento inexistente -Falta dígitos -Login inexistente -Passsword incorrecto	Debe completar todos los campos

Eliminar trabajador		Trabajador eliminado con éxito		Debe seleccionar una fila
Modificar trabajador	Nombres Apellido Paterno Apellido Materno Tipo de Documento Número de Documento Celular Email Sueldo Acceso Login Password Estado	Trabajador actualizado con éxito	-Nombre en espacios -Apellido en espacios -Tipo de documento incorrecto -Número de documento inexistente -Falta dígitos -Login inexistente -Passsword incorrecto	Debe completar todos los campos
Listar trabajador	No tiene datos	Trabajador listado con éxito	No tiene datos	No hay trabajadores para listar
Validar trabajador	Usuario Contraseña	Trabajador validado con éxito	-Usuario inexistente -Contraseña incorrecta	No puede acceder al sistema
Registro Entrada Cliente	Nombres Apellido Paterno Apellido Materno Tipo de Documento Número de Documento Celular Email Habitación Fecha hora ingreso Fecha hora salida	Entrada registrada con éxito	-Nombre en espacios -Apellido en espacios -Tipo de documento incorrecto -Número de documento inexistente -Falta dígitos -Habitación ocupada	Debe completar todos los campos
Registro Salida Cliente	Habitación	Salida registrada con éxito	-Número habitación incorrecto	Debe seleccionar una fila
Modificar Salida Cliente	Habitación	Salida actualizada con éxito	-Número habitación incorrecto	Debe seleccionar una fila

Tabla 7: Plan de pruebas Incremental - 1

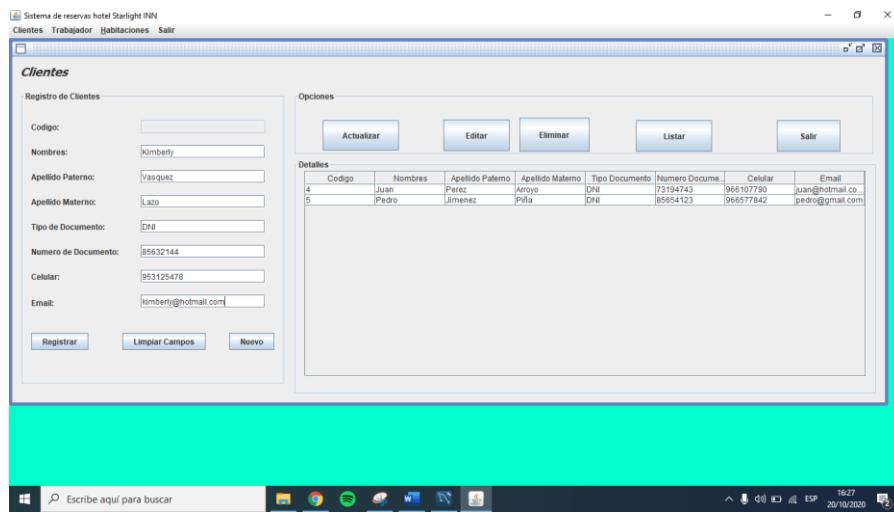
Caso de uso	Datos de Entrada normal	Resultado Esperado normal	Datos de entrada anómalo	Resultado esperado anómalo
Registro de reserva	Tipo de reserva Fecha de reserva Fecha de ingreso Fecha de salida Costo Estado	Reserva realizada con éxito	-Tipo de reserva incorrecta -Fecha de reserva inválida -Fecha de ingreso y salida inválida -Estado inválido	Debe completar todos los campos.
Modificar reserva	Tipo de reserva Fecha de reserva Fecha de ingreso Fecha de salida Costo Estado	Reserva modificada con éxito	-Tipo de reserva incorrecta -Fecha de reserva inválida -Fecha de ingreso y salida inválida -Estado inválido	Debe completar todos los campos.
Eliminar reserva	No tiene datos	Reserva eliminada con éxito	No tiene datos	Debe seleccionar una fila
Consultar reserva	No tiene datos	Consulta realizada con éxito	No tiene datos	No existen reservas
Registro de habitación	Número Piso Descripción Características Precio diario Estado Tipo de habitación	Habitación registrada con éxito	-Número de habitación inválida -Piso inválido -Exceso de límites de caracteres -Estado inválido -Tipo de habitación inválida	Debe completar todos los campos.
Modificar habitación	Número Piso Descripción Características Precio diario Estado Tipo de habitación	Habitación modificada con éxito	-Número de habitación inválida -Piso inválido -Exceso de límites de caracteres -Estado inválido -Tipo de habitación inválida	Debe completar todos los campos.
Eliminar habitación	No tiene datos	Habitación eliminada con éxito	No tiene datos	Debe seleccionar una fila

Consultar habitación	No tiene datos	Consulta realizada con éxito	No tiene datos	No existen habitaciones
----------------------	----------------	------------------------------	----------------	-------------------------

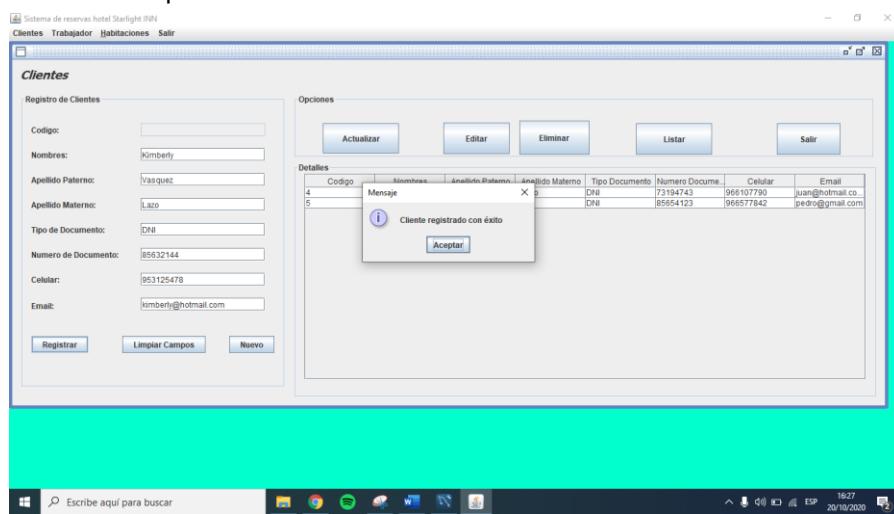
Tabla 8: Plan de pruebas Incremental - 2

Caso de uso Registrar Cliente:

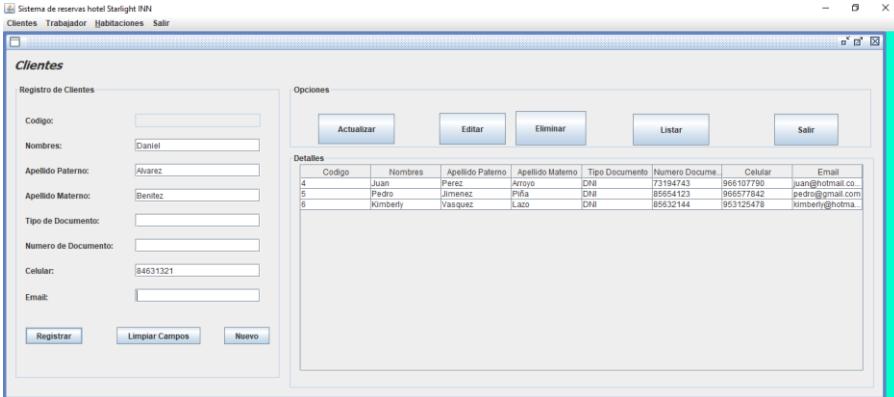
Datos de entrada normal:



Resultado esperado normal:



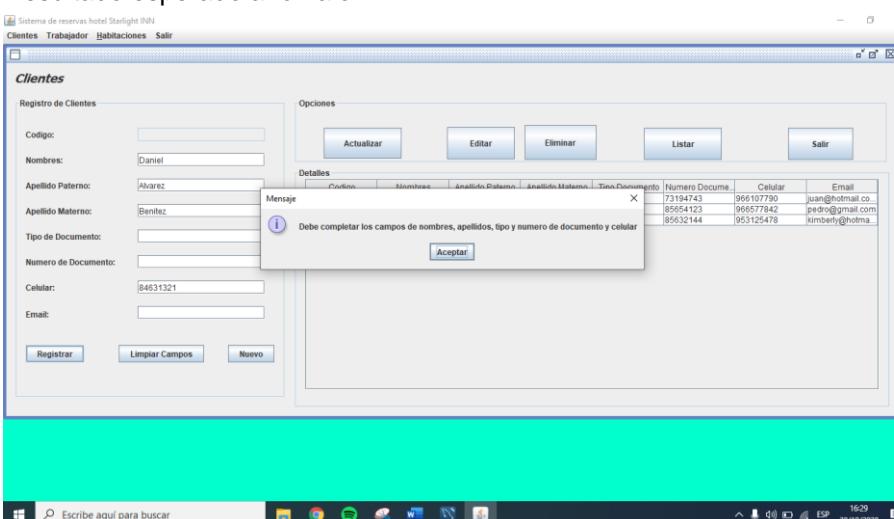
Datos de entrada anómalo:



The screenshot shows the 'Cuentas' (Clients) registration window. On the left, there is a form for entering client information: Código, Nombres, Apellido Paterno, Apellido Materno, Tipo Documento, Número Documento, Celular, and Email. Below the form are buttons for Registrar, Limpiar Campos, and Nuevo. On the right, there is a 'Opciones' (Options) panel with buttons for Actualizar, Editar, Eliminar, Listar, and Salir. Below the options is a 'Detalles' (Details) table showing three client records:

Código	Nombres	Apellido Paterno	Apellido Materno	Tipo Documento	Número Documento	Celular	Email
4	Juan	Perez	Arroyo	DNI	73194743	966107790	juan@hotmail.co
5	Pedro	Jimenez	Piña	DNI	85654123	966677742	pedro@gmail.com
6	Kimberly	Vasquez	Lazo	DNI	85632144	953125478	kimberly@hotmail.com

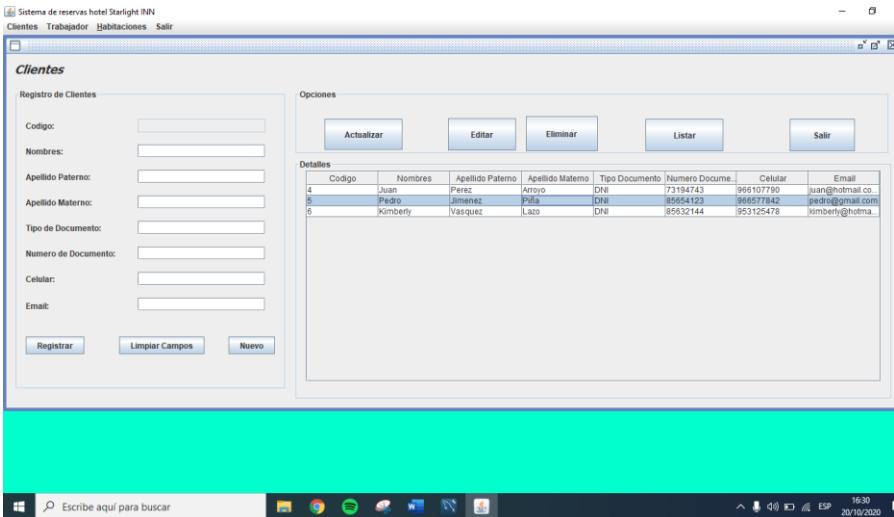
Resultado esperado anómalo:



The screenshot shows the same 'Cuentas' (Clients) registration window. The 'Apellido Paterno' field is highlighted in red, indicating it is required. A modal dialog box titled 'Mensaje' (Message) appears, stating 'Debe completar los campos de nombres, apellidos, tipo y número de documento y celular' (You must complete the fields for names, surnames, type and number of document and cellular). Below the message are 'Aceptar' (Accept) and 'Cancelar' (Cancel) buttons.

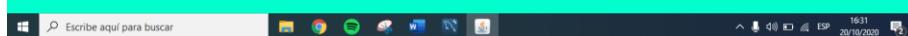
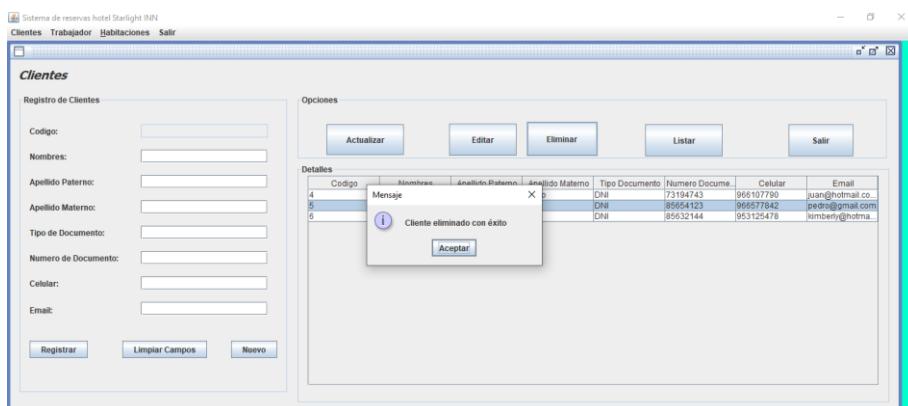
Caso de uso Eliminar Cliente:

Datos de entrada normal:

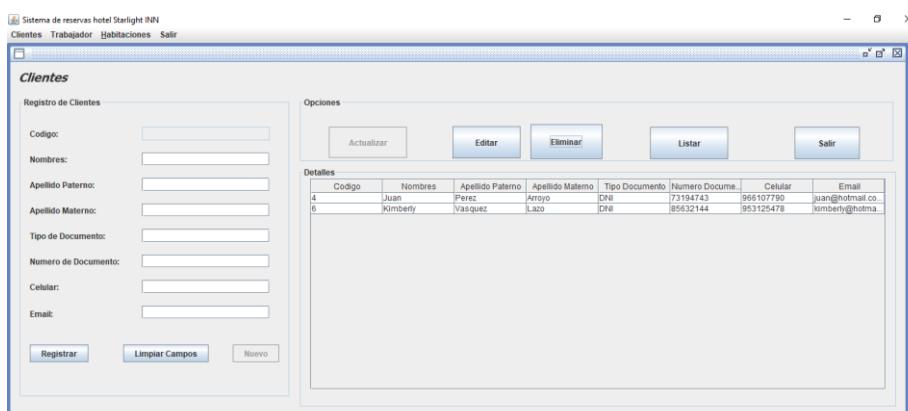


The screenshot shows the 'Cuentas' (Clients) registration window. The 'Apellido Paterno' field is highlighted in red, indicating it is required. The rest of the form and the 'Opciones' (Options) panel are identical to the previous screenshot.

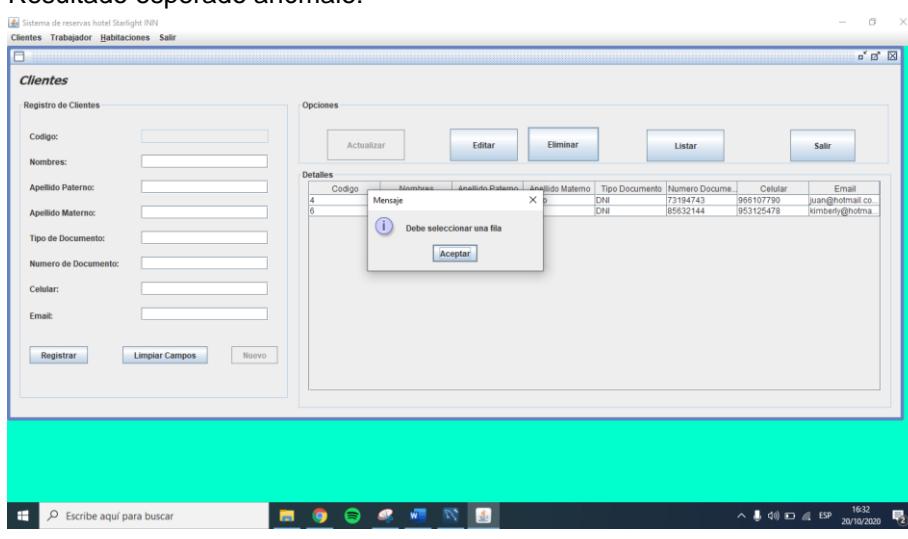
Resultado esperado normal:



Datos de entrada anómalo:

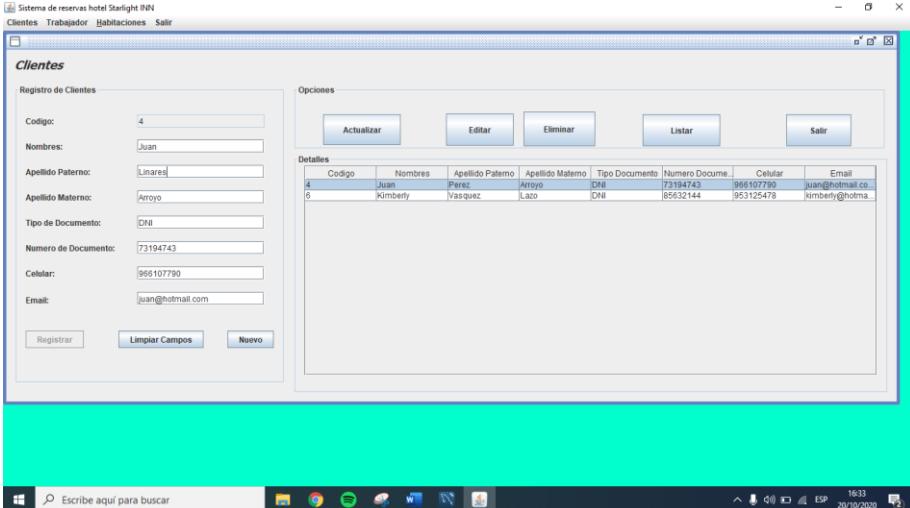


Resultado esperado anómalo:



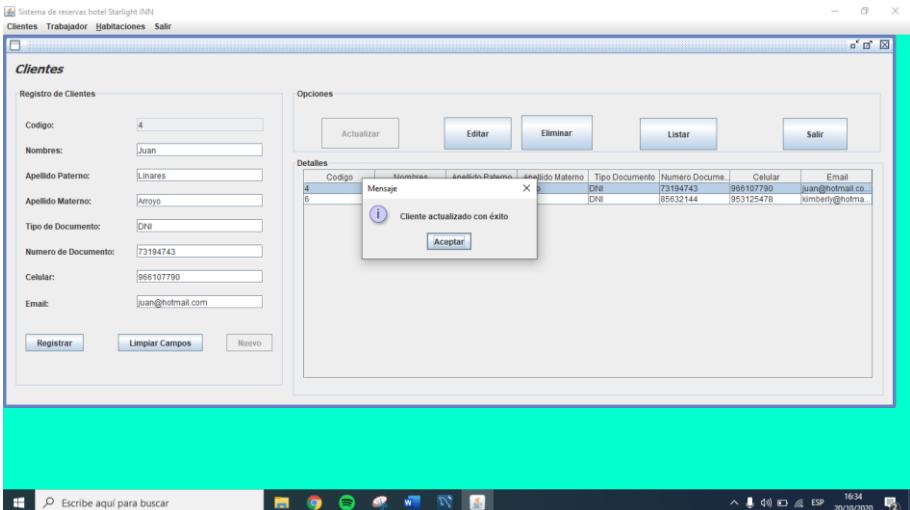
Caso de uso Modificar Cliente:

Datos de entrada normal:



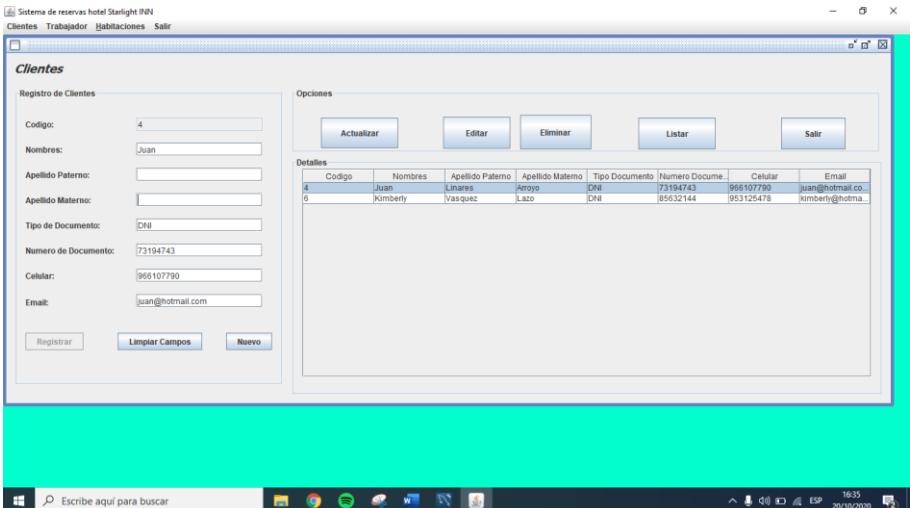
The screenshot shows the 'Cuentas' (Clients) window of the hotel management system. On the left, there's a form for 'Registro de Clientes' (Client Registration) with fields for Código (4), Nombres (Juan), Apellido Paterno (Linares), Apellido Materno (Arroyo), Tipo de Documento (DNI), Número de Documento (73194743), Celular (966107790), and Email (juan@hotmail.com). Below the form are buttons for Registrar, Limpiar Campos, and Nuevo. On the right, there's a 'Opciones' (Options) panel with buttons for Actualizar, Editar, Eliminar, Listar, and Salir. A 'Detalles' (Details) table lists two clients: Juan Linares Arroyo (Código 4) and Kimberly Vasquez Lazo (Código 6). The client Juan Linares Arroyo has been selected. The status bar at the bottom shows the date as 20/10/2020 and the time as 16:33.

Resultado esperado normal:



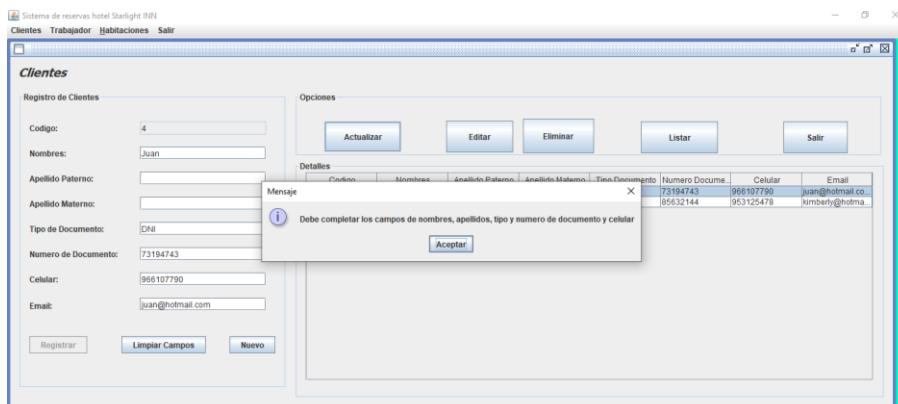
This screenshot is identical to the previous one, but it includes a modal dialog box in the center. The dialog box has a title 'Mensaje' (Message) and a message 'Cliente actualizado con éxito' (Client updated successfully). It contains an 'Aceptar' (Accept) button. The rest of the interface and data are the same as the first screenshot.

Datos de entrada anómalo:



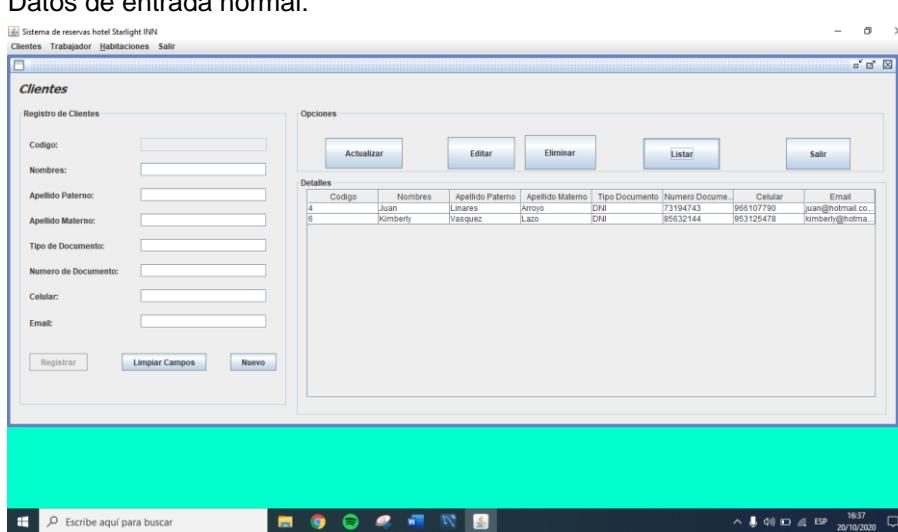
This screenshot shows the same 'Cuentas' window as before, but the client 'Juan Linares Arroyo' has been deleted from the database. The 'Detalles' table now only shows the client 'Kimberly Vasquez Lazo' (Código 6). When the 'Actualizar' (Update) button is clicked, the system fails to find the client with Código 4, resulting in an error message: 'No se ha encontrado el cliente'. The status bar at the bottom shows the date as 20/10/2020 and the time as 16:35.

Resultado esperado anómalo:

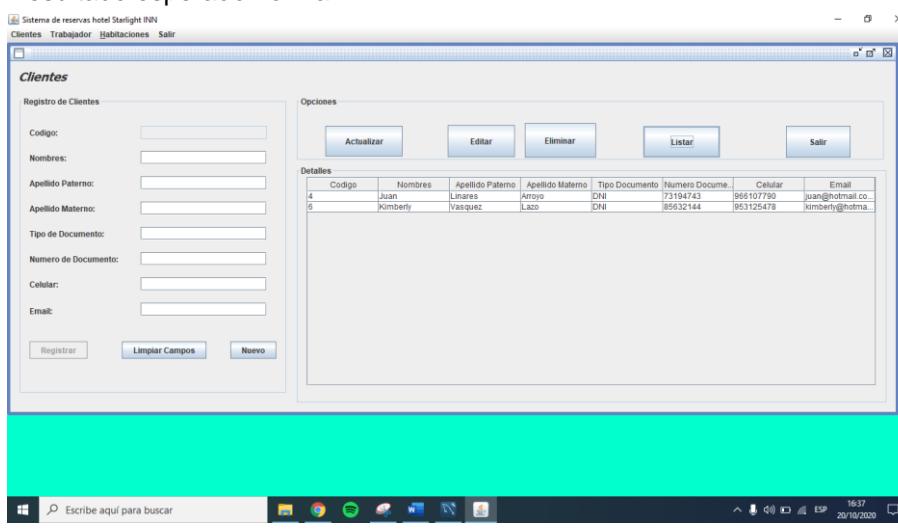


Caso de uso Consultar Cliente:

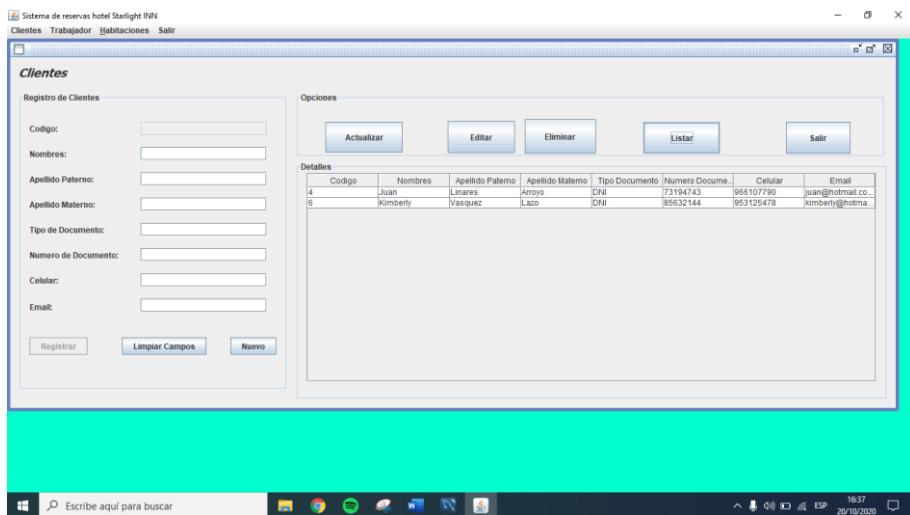
Datos de entrada normal:



Resultado esperado normal:



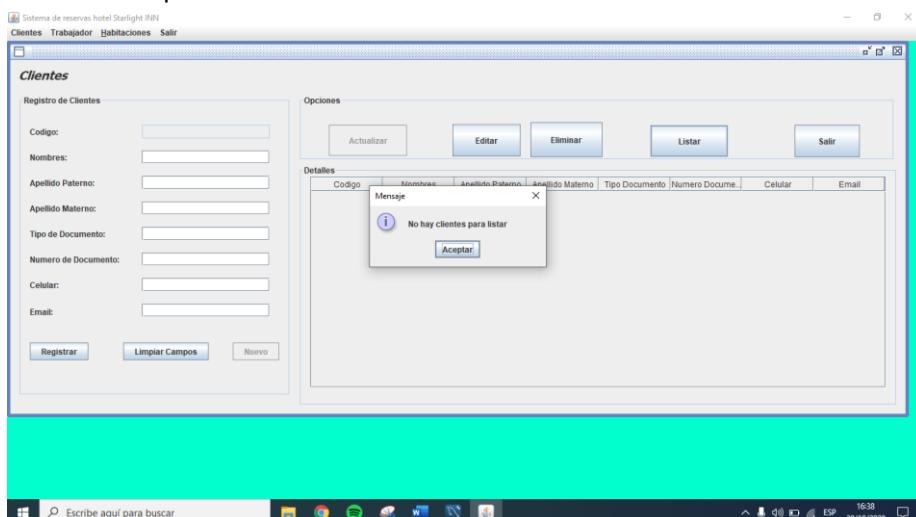
Datos de entrada anómalo:



The screenshot shows the 'Cuentas' registration window. On the left, there is a form for entering client information: Código, Nombres, Apellido Paterno, Apellido Materno, Tipo Documento, Número Documento, Celular, Email, Registrar, Limpiar Campos, and Nuevo. On the right, there is a table titled 'Detalles' with columns: Código, Nombres, Apellido Paterno, Apellido Materno, Tipo Documento, Número Documento, Celular, and Email. Two rows of data are visible:

Código	Nombres	Apellido Paterno	Apellido Materno	Tipo Documento	Número Documento	Celular	Email
4	Juan	Linares	Arroyo	DNI	73194743	965107790	juan@hotmail.co...
6	Kimberly	Vasquez	Lazo	DNI	85632144	953125478	kimberly@hotma...

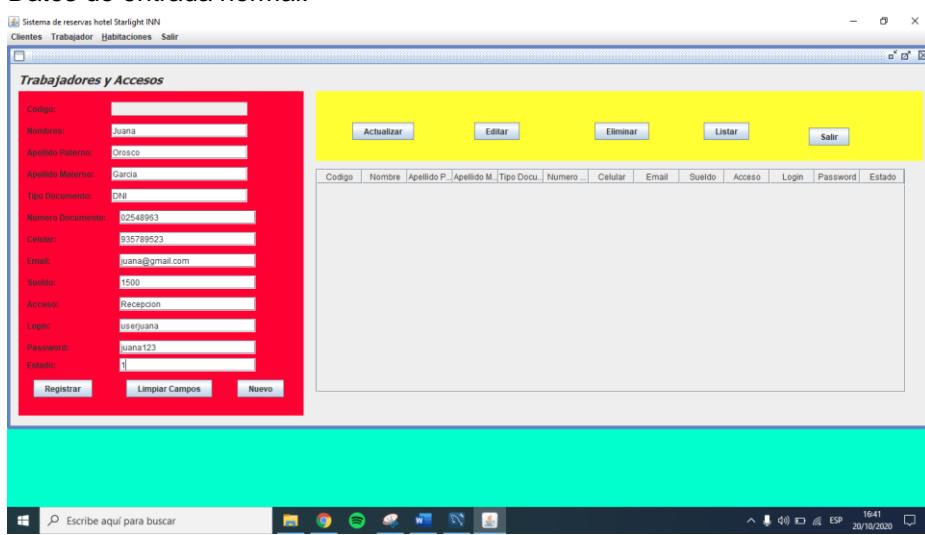
Resultado esperado anómalo:



The screenshot shows the same 'Cuentas' registration window. The 'Detalles' table now has a single row with a message: 'No hay clientes para listar'. A modal dialog box is displayed with the message 'No hay clientes para listar' and an 'Aceptar' button.

Caso de uso Registrar Trabajador:

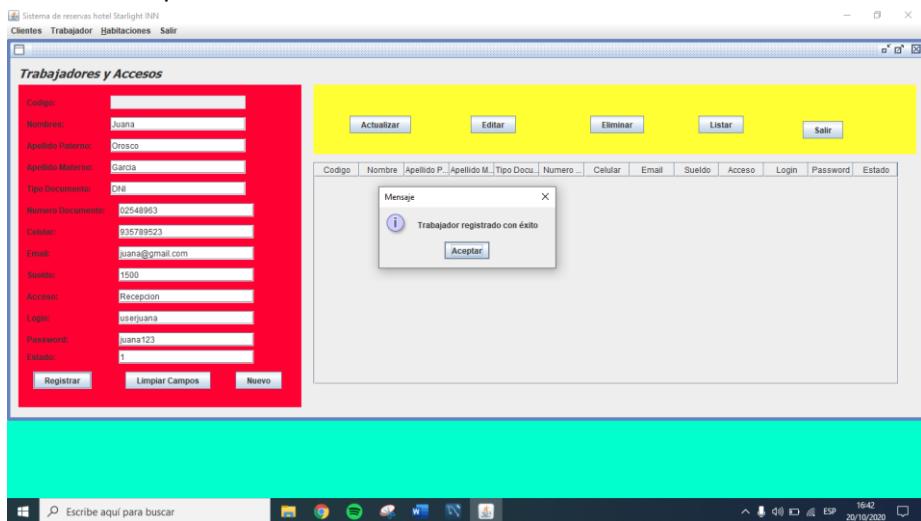
Datos de entrada normal:



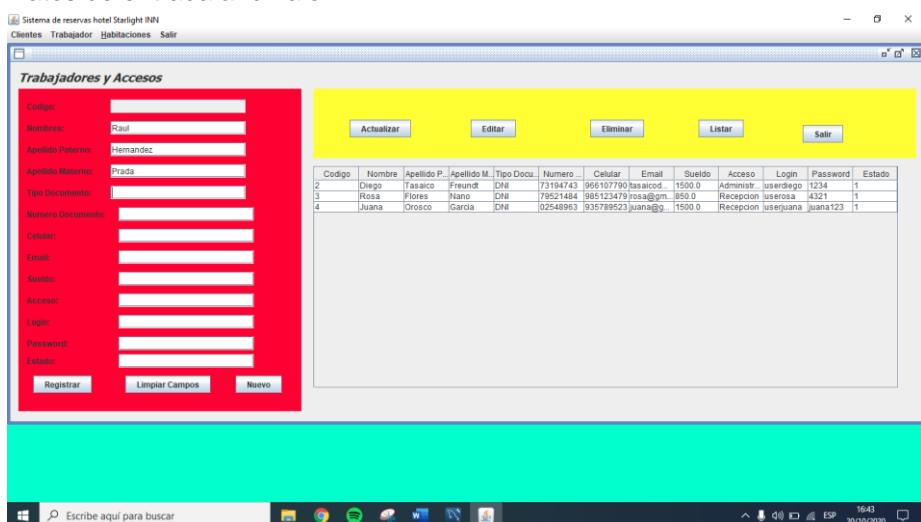
The screenshot shows the 'Trabajadores y Accesos' registration window. On the left, there is a form for entering employee information: Código, Nombres, Apellido Paterno, Apellido Materno, Tipo Documento, Número Documento, Celular, Email, Sueldo, Acceso, Login, Password, Estado, Registrar, Limpiar Campos, and Nuevo. On the right, there is a table titled 'Trabajadores y Accesos' with columns: Código, Nombre, Apellido P., Apellido M., Tipo Docu., Número Doc., Celular, Email, Sueldo, Acceso, Login, Password, and Estado. One row of data is visible:

Código	Nombre	Apellido P.	Apellido M.	Tipo Docu.	Número Doc.	Celular	Email	Sueldo	Acceso	Login	Password	Estado
	Juan	Garcia	Orozco	DNI	02548993	935789523	juana@gmail.com	1500	Recepcion	userjuana	juana123	1

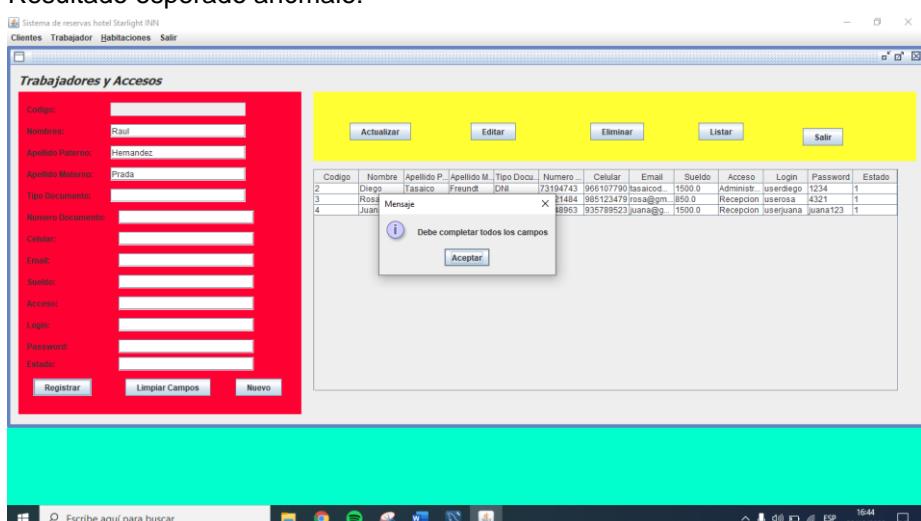
Resultado esperado normal:



Datos de entrada anómalo:

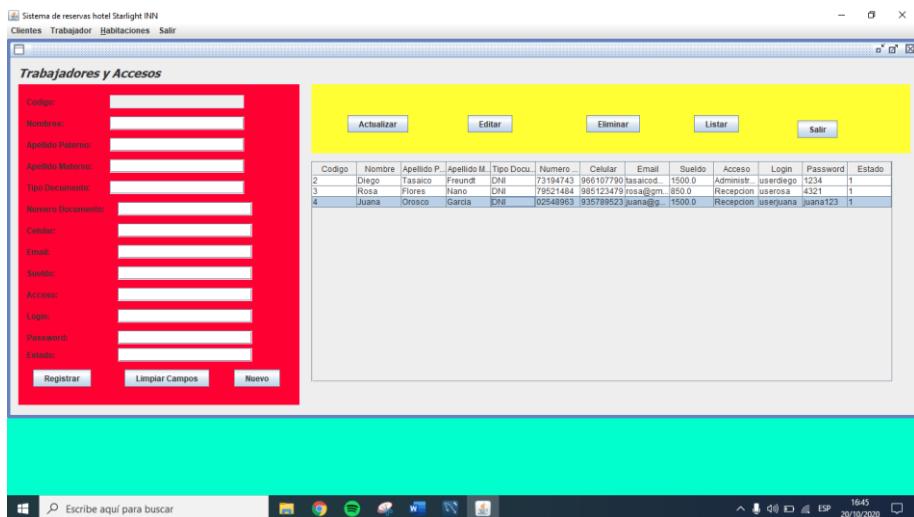


Resultado esperado anómalo:



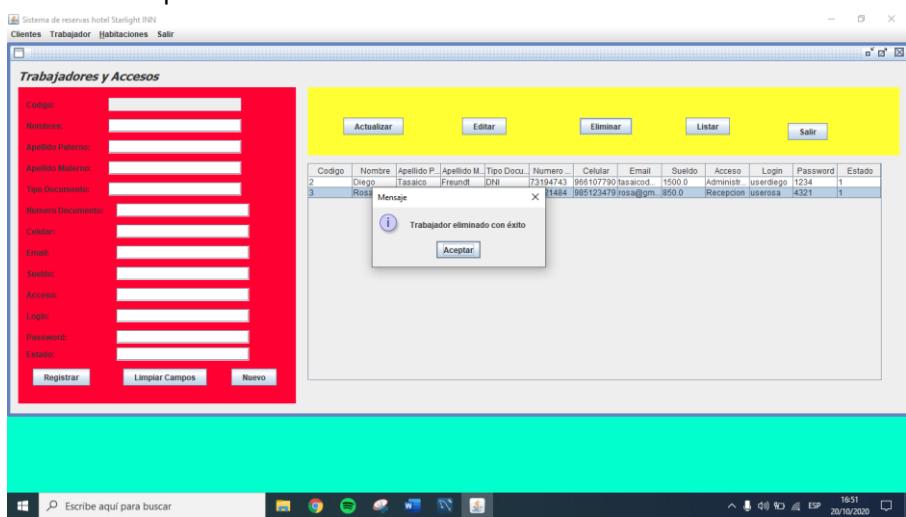
Caso de uso Eliminar Trabajador:

Datos de entrada normal:



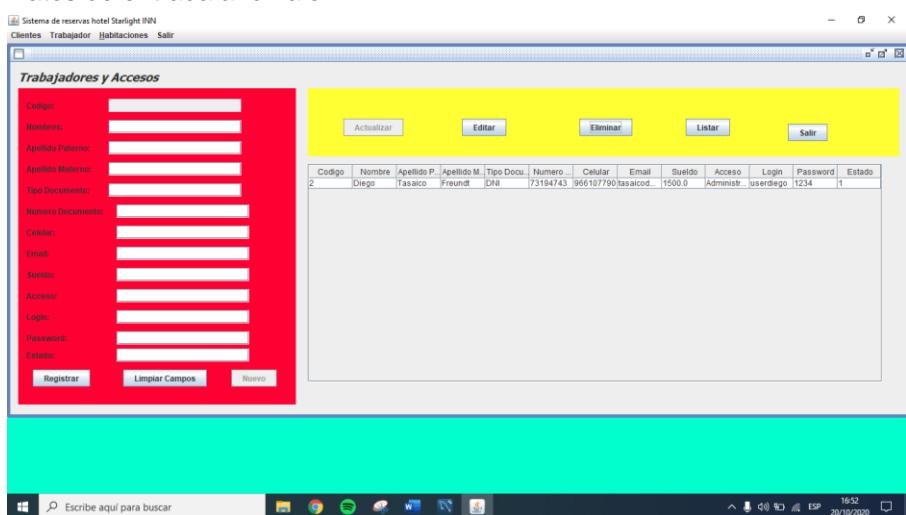
The screenshot shows a Windows desktop environment. The application window title is "Sistema de reservas hotel Starlight INN". The menu bar includes "Clientes", "Trabajador", "Habitaciones", and "Salir". The main window is titled "Trabajadores y Accesos". On the left is a form with fields for Código, Nombre, Apellido Paterno, Apellido Materno, Tipo Documento, Número Documento, Celular, Email, Estado, Acceso, Login, and Password. Below the form are buttons for "Registrar", "Limpiar Campos", and "Nuevo". On the right is a table with columns: Código, Nombre, Apellido P., Apellido M., Tipo Docu., Número..., Celular, Email, Sueldo, Acceso, Login, Password, and Estado. The table contains four rows of data. At the top of the right panel are buttons for "Actualizar", "Editar", "Eliminar", "Listar", and "Salir". A message box at the bottom right of the table area says "Trabajador eliminado con éxito" (Employee deleted successfully) with an "Aceptar" (Accept) button.

Resultado esperado normal:



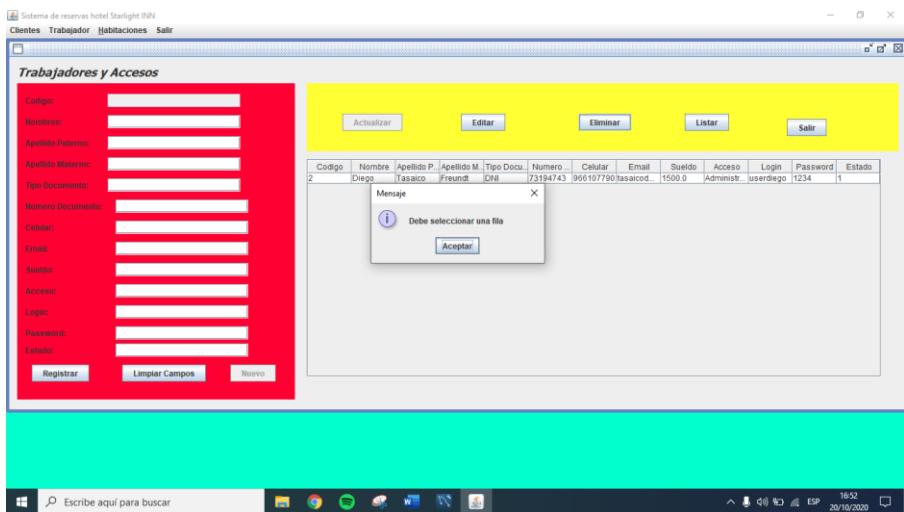
This screenshot is identical to the previous one, showing the same application window and data. The difference is the presence of a message box in the center of the right panel stating "Trabajador eliminado con éxito" (Employee deleted successfully) with an "Aceptar" (Accept) button.

Datos de entrada anómalo:



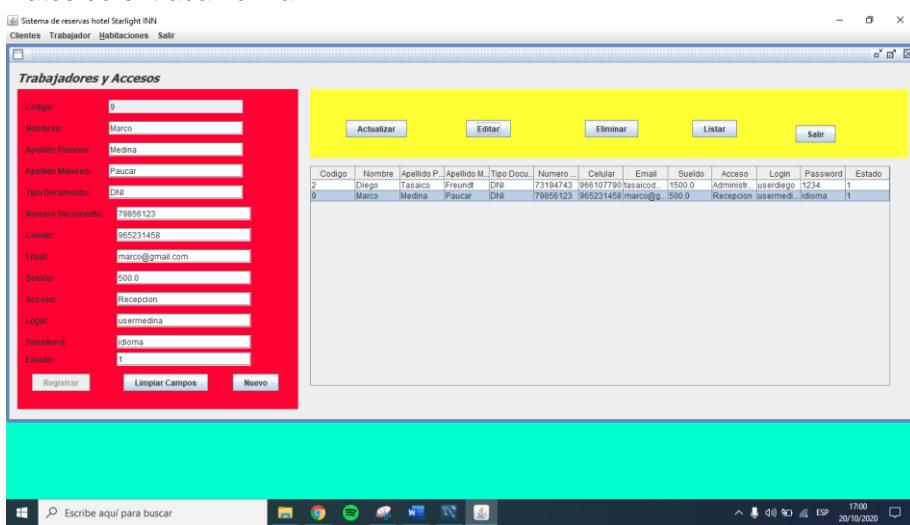
This screenshot shows the same application window and data as the previous ones. However, the table on the right side of the window is now empty, indicating that all data has been deleted.

Resultado esperado anómalo:

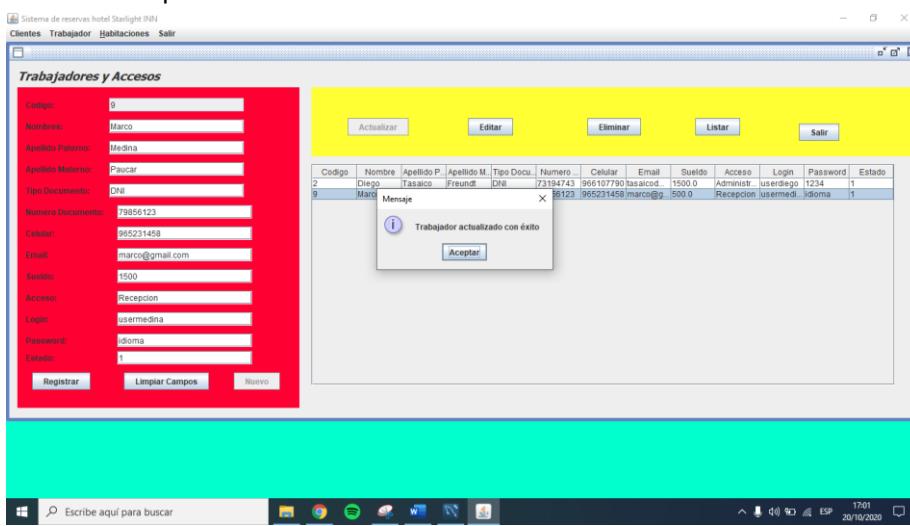


Caso de uso Modificar Trabajador:

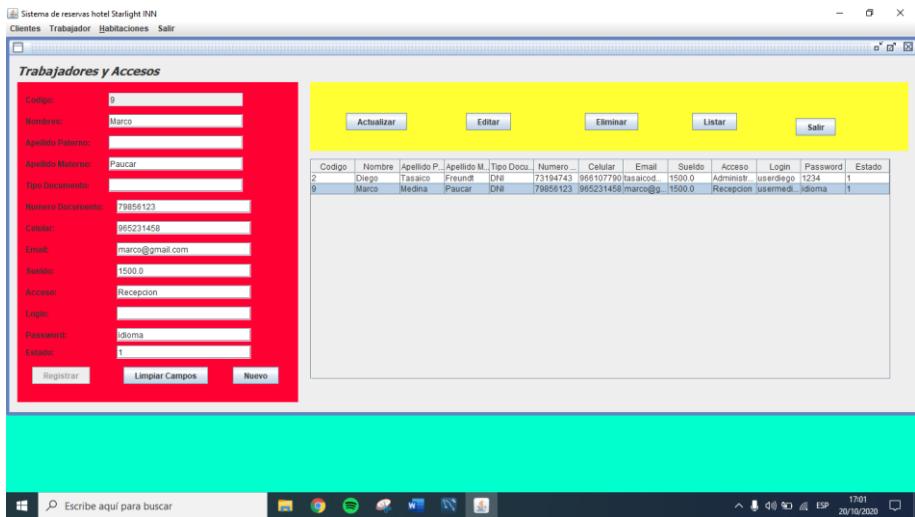
Datos de entrada normal:



Resultado esperado normal:



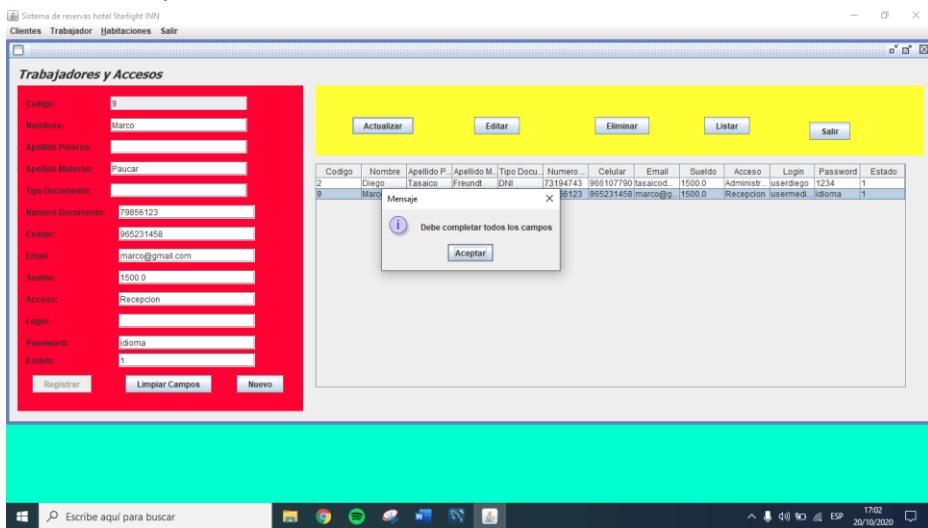
Datos de entrada anómalo:



The screenshot shows a Windows desktop environment with a browser window open to the 'Sistema de reservas hotel Starlight INN' application. The page title is 'Trabajadores y Accesos'. On the left, there is a form with various input fields (Nombre, Apellido Paterno, etc.) filled with placeholder text. On the right, a table displays employee data. A modal dialog box is overlaid on the table, containing a message: 'Debe completar todos los campos' (You must complete all fields) with an 'Aceptar' (Accept) button.

Código	Nombre	Apellido P.	Apellido M.	Tipo Docu.	Número	Celular	Email	Sueldo	Acceso	Login	Password	Estado
2	Diego	Tasaico	Freundt	DNI	73194743	966107790	tasaicod.	1500.0	Administr.	userdiego	1234	1
9	Marco	Medina	Paucar	DNI	79856123	965231458	marco@g...	1500.0	Recepcion	usermedio	idioma	1

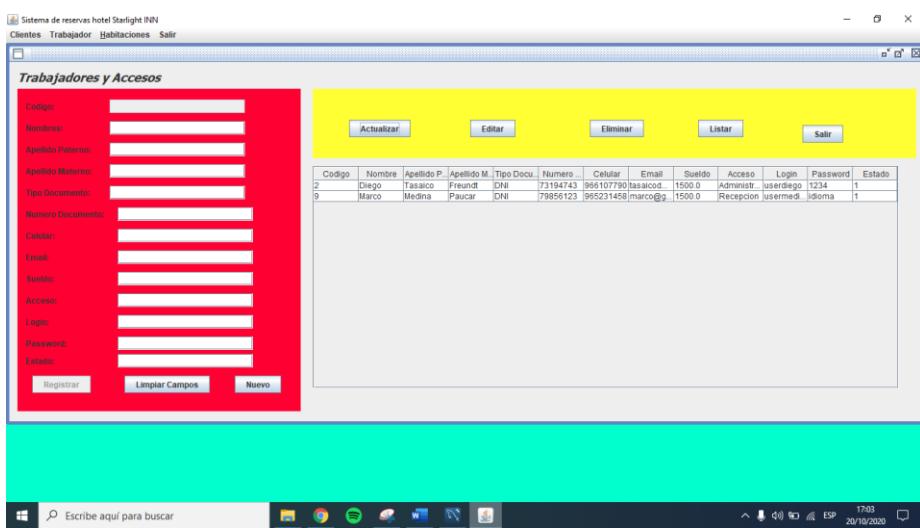
Resultado esperado anómalo:



This screenshot is identical to the previous one, showing the same application interface and data. The difference is that the modal dialog box now contains a different message: 'Debe completar todos los campos' (You must complete all fields) with an 'Aceptar' (Accept) button.

Caso de uso Consultar Trabajador:

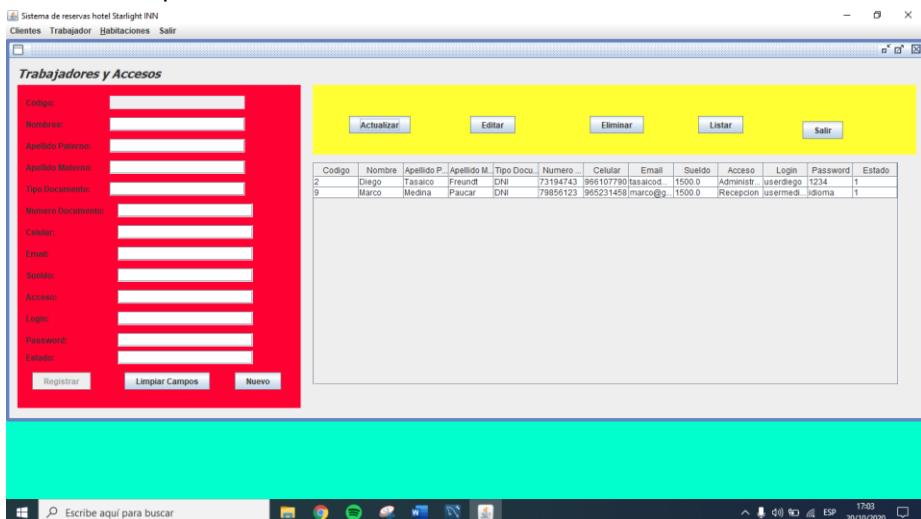
Datos de entrada normal:



This screenshot shows the same application interface with valid data entered into the form fields. The table on the right displays the correct employee information, and the modal dialog box is no longer present.

Código	Nombre	Apellido P.	Apellido M.	Tipo Docu.	Número	Celular	Email	Sueldo	Acceso	Login	Password	Estado
2	Diego	Tasaico	Freundt	DNI	73194743	966107790	tasaicod.	1500.0	Administr.	userdiego	1234	1
9	Marco	Medina	Paucar	DNI	79856123	965231458	marco@g...	1500.0	Recepcion	usermedio	idioma	1

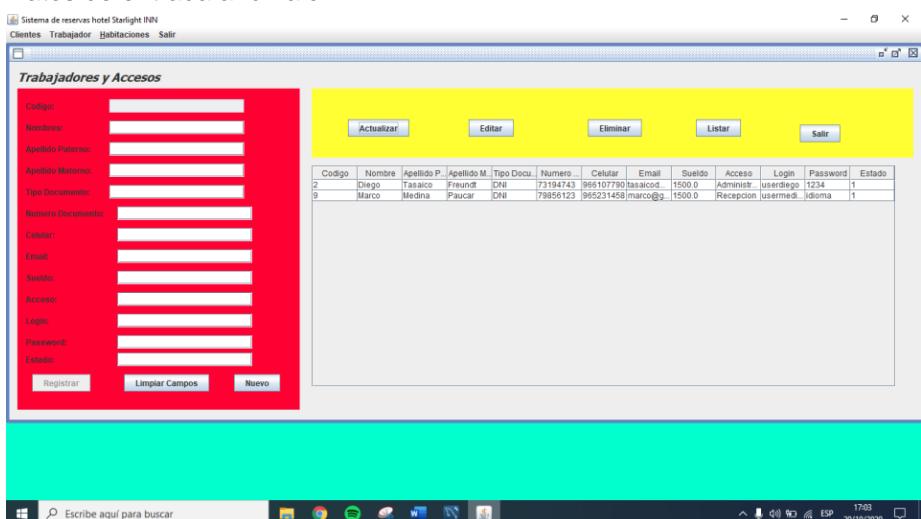
Resultado esperado normal:



The screenshot shows a Windows desktop environment with a taskbar at the bottom. The application window title is "Trabajadores y Accesos". The left panel contains a form with fields for Código, Nombre, Apellido Paterno, Apellido Materno, Tipo Documento, Número Documento, Celular, Email, Sueldo, Acceso, Login, Password, and Estado. Buttons at the bottom include "Registrar", "Limpiar Campos", and "Nuevo". The right panel displays a table with two rows of employee data. The table has columns: Código, Nombre, Apellido P., Apellido M., Tipo Docu., Número, Celular, Email, Sueldo, Acceso, Login, Password, and Estado. The data is as follows:

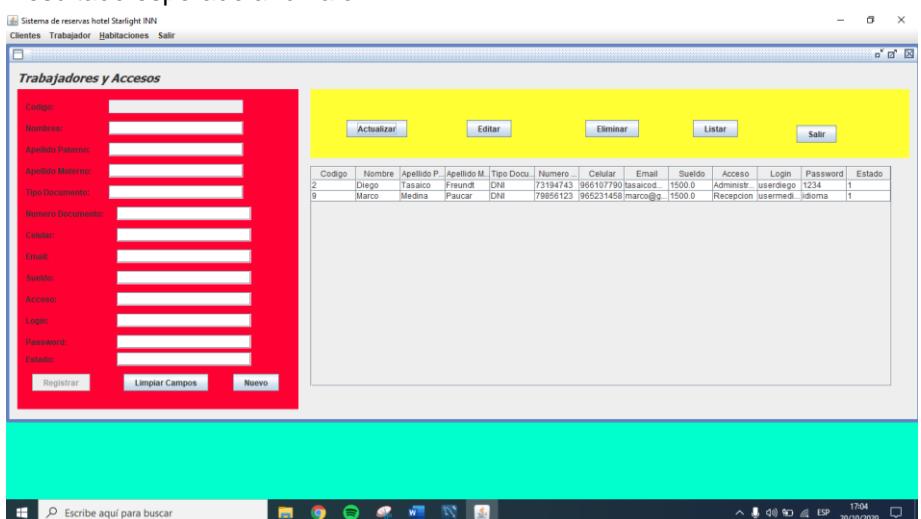
Código	Nombre	Apellido P.	Apellido M.	Tipo Docu.	Número	Celular	Email	Sueldo	Acceso	Login	Password	Estado
2	Diego	Tasaico	Freundt	DNI	73194743	966107790	tasaicod...	1500.0	Administr.	userdiego	1234	1
9	Marco	Medina	Paucar	DNI	79856123	965231458	marco@g...	1500.0	Recepcion	usermed...	idioma	1

Datos de entrada anómalo:



This screenshot is identical to the previous one, showing the same application window and data. However, every field in the left panel's form is highlighted with a red background, indicating validation errors or anomalies in the input data.

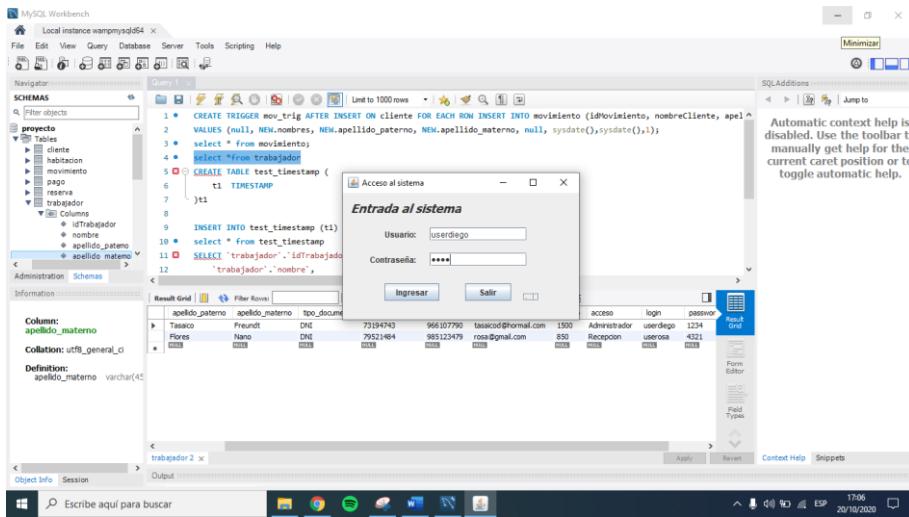
Resultado esperado anómalo:



This screenshot shows the application window with the same layout and data as before. However, the table on the right side is completely empty, with no rows displayed, which is the expected outcome for invalid input.

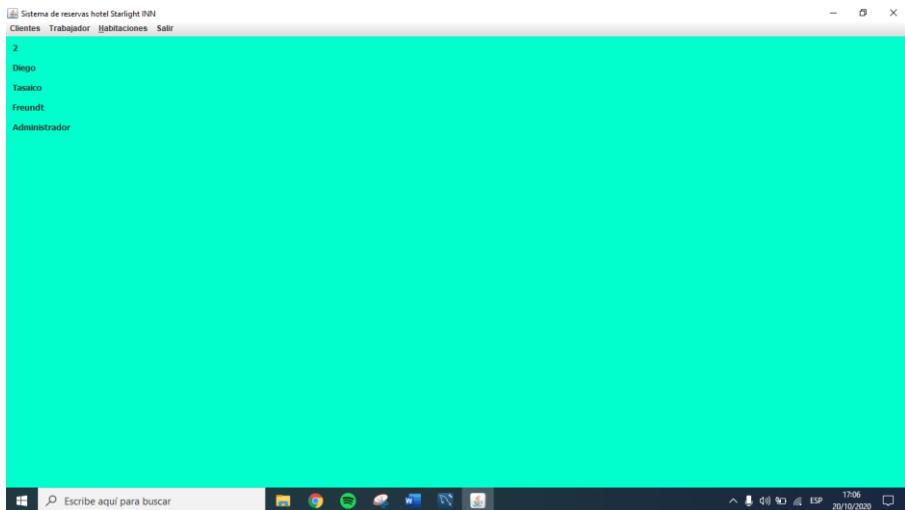
Caso de uso Validar Trabajador:

Datos de entrada normal:



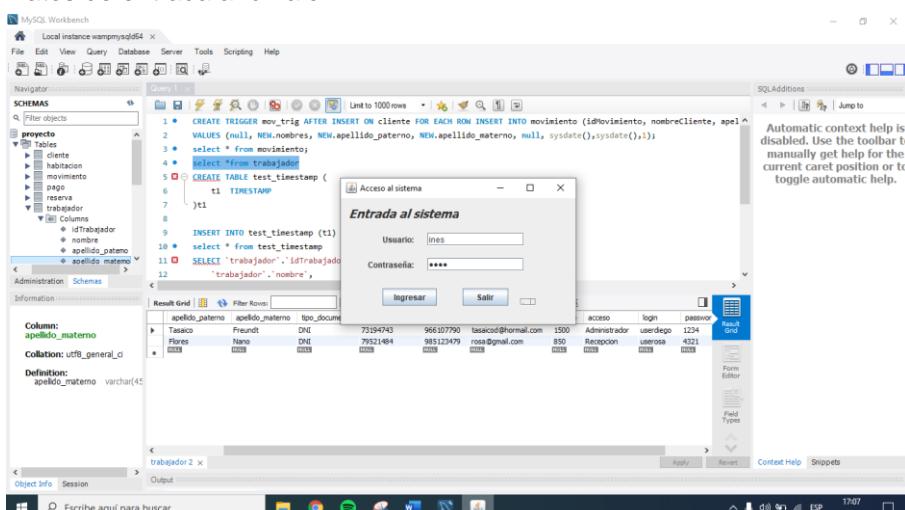
The screenshot shows the MySQL Workbench interface. A query editor window displays a trigger creation script and a select query from the 'trabajador' table. An 'Acceso al sistema' (Access system) dialog box is open, showing a successful login attempt for the user 'userdiego'.

Resultado esperado normal:



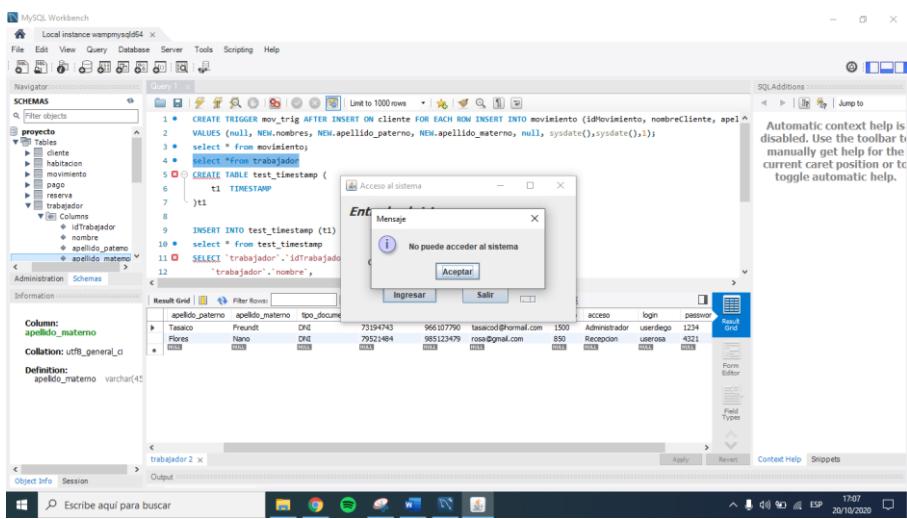
The screenshot shows a web browser window titled 'Sistema de reservas hotel Starlight INN'. The URL bar shows 'Escribe aquí para buscar'. The page content displays a successful login message for the user 'Diego'.

Datos de entrada anómalo:



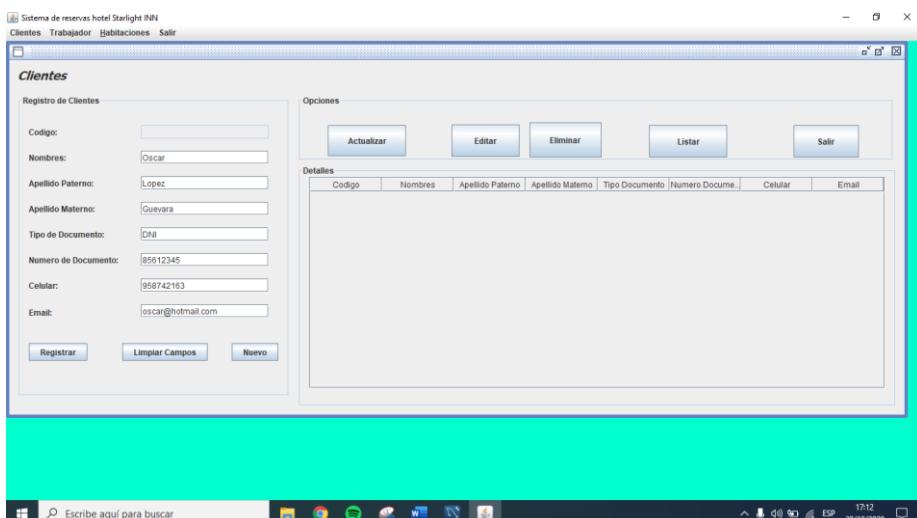
The screenshot shows the MySQL Workbench interface with the same trigger creation and table select queries as the previous screenshot. The 'Acceso al sistema' dialog box shows an unsuccessful login attempt for the user 'ines'.

Resultado esperado anómalo:

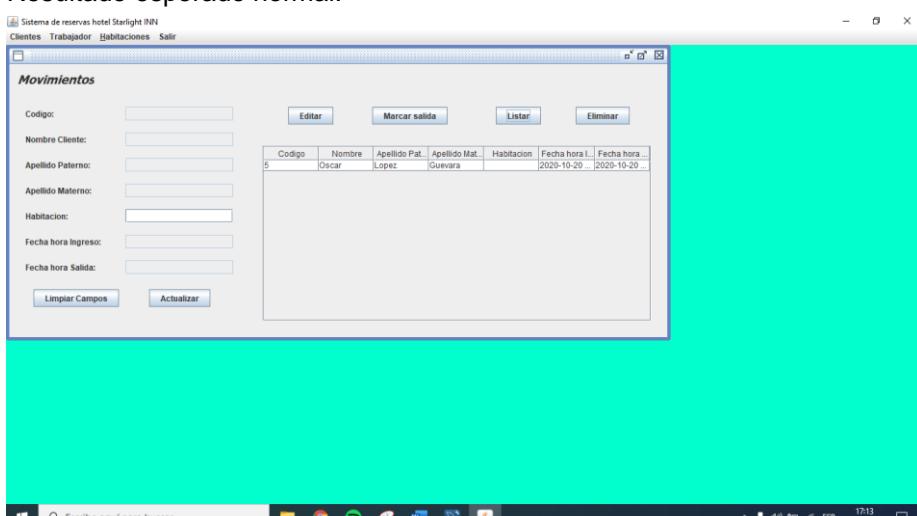


Caso de uso Registro de Entrada Cliente:

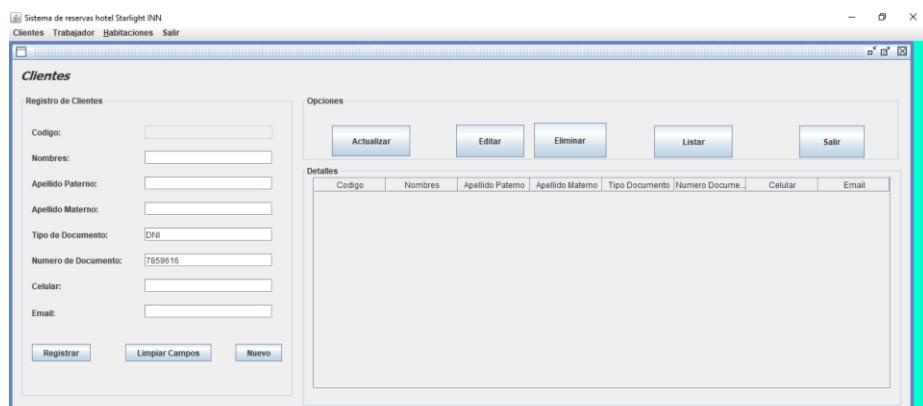
Datos de entrada normal:



Resultado esperado normal:



Datos de entrada anómalo:



Cientes

Registro de Clientes

Código:	Nombre:
Apellido Paterno:	Apellido Materno:
Tipo de Documento:	DNI
Número de Documento:	7859615
Celular:	
Email:	

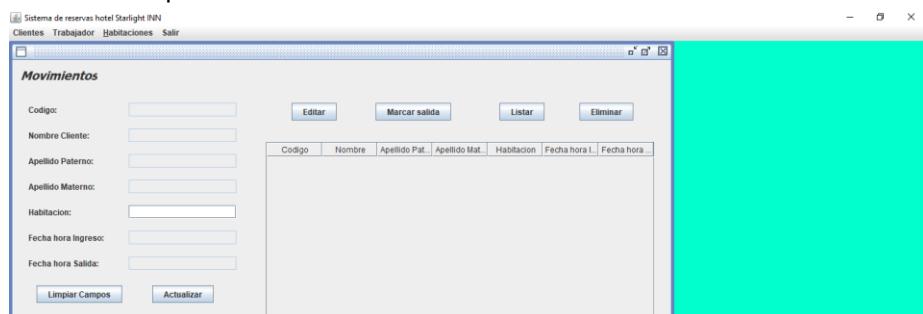
Botones: Registrar, Limpiar Campos, Nuevo

Opciones: Actualizar, Editar, Eliminar, Listar, Salir

Detalles: Código, Nombre, Apellido Pat., Apellido Mat., Tipo Documento, Número Docume..., Celular, Email



Resultado esperado anómalo:



Movimientos

Código:	Nombre:	Apellido Pat.	Apellido Mat.	Habitación:	Fecha hora I.	Fecha hora S.
---------	---------	---------------	---------------	-------------	---------------	---------------

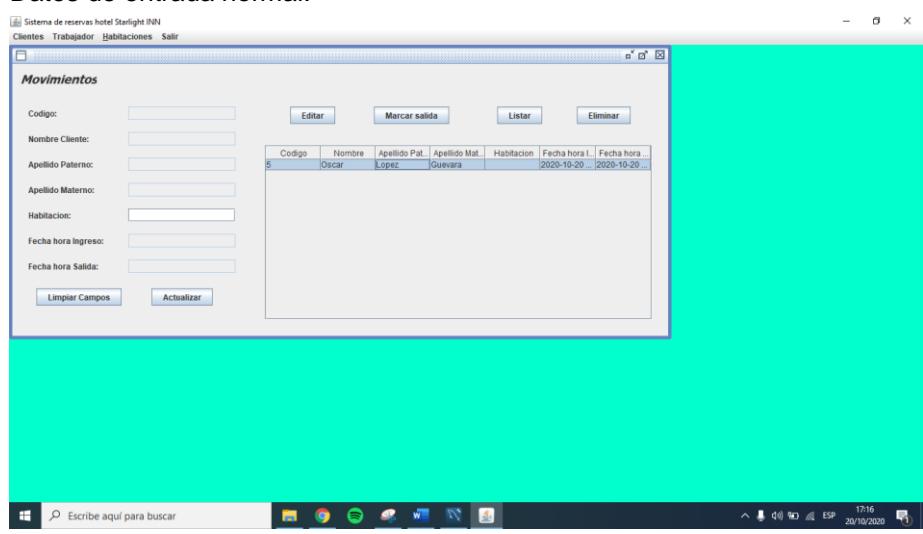
Botones: Editar, Marcar salida, Listar, Eliminar

Botones: Limpiar Campos, Actualizar



Caso de uso Registro de Salida Cliente:

Datos de entrada normal:



Movimientos

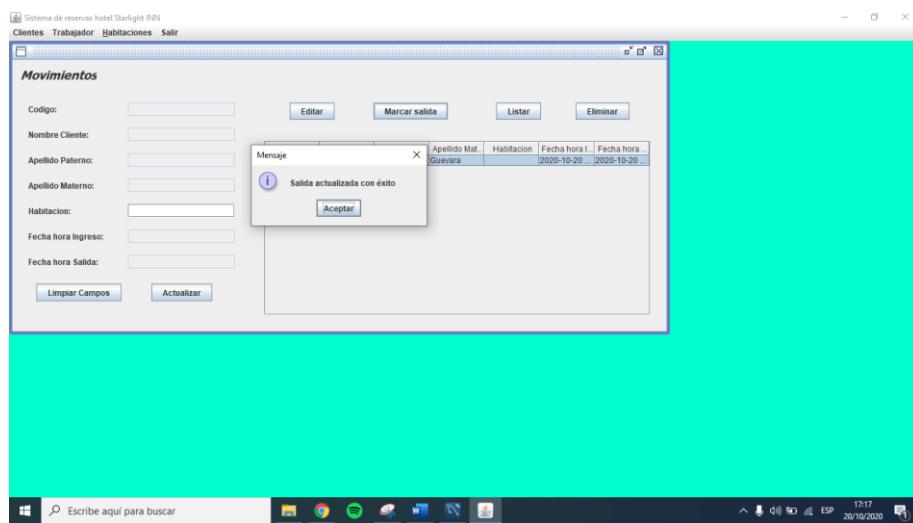
Código:	Nombre:	Apellido Pat.	Apellido Mat.	Habitación:	Fecha hora I.	Fecha hora S.
---------	---------	---------------	---------------	-------------	---------------	---------------

Botones: Editar, Marcar salida, Listar, Eliminar

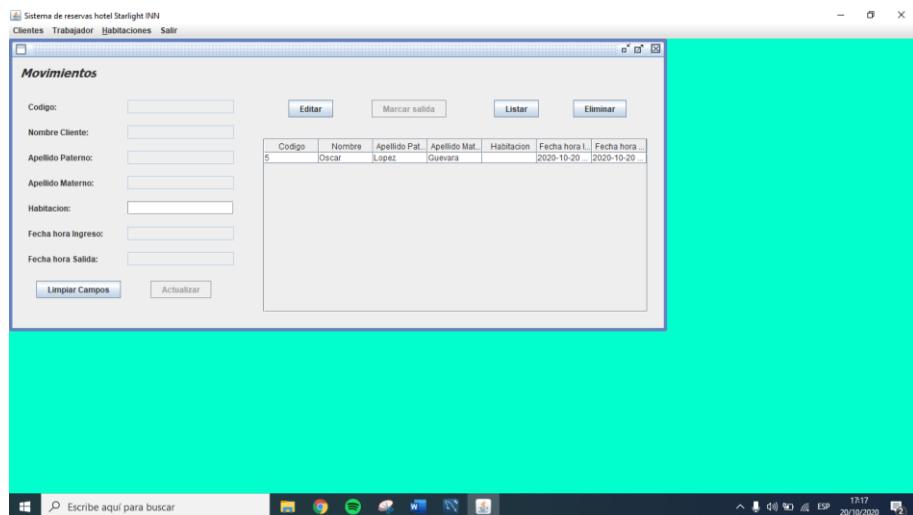
Botones: Limpiar Campos, Actualizar

5	Oscar	Lopez	Guevara		2020-10-20	2020-10-20
---	-------	-------	---------	--	------------	------------

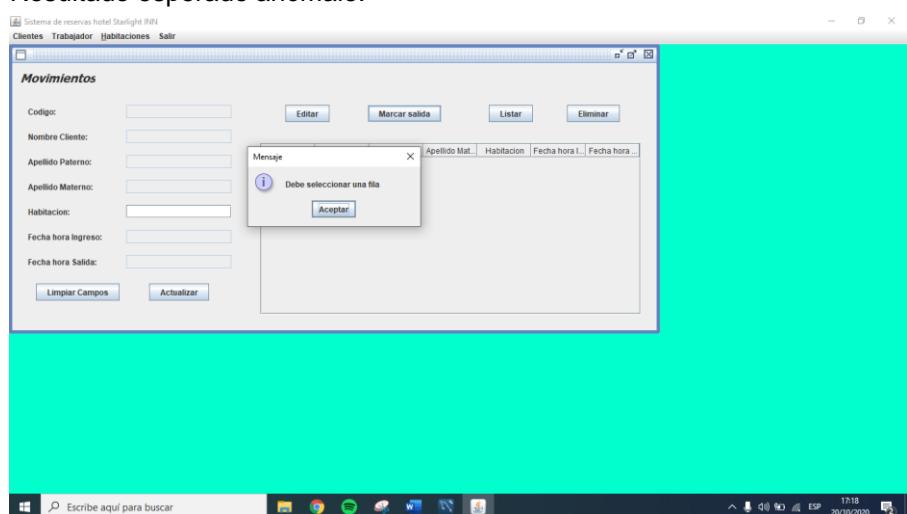
Resultado esperado normal:



Datos de entrada anómalo:

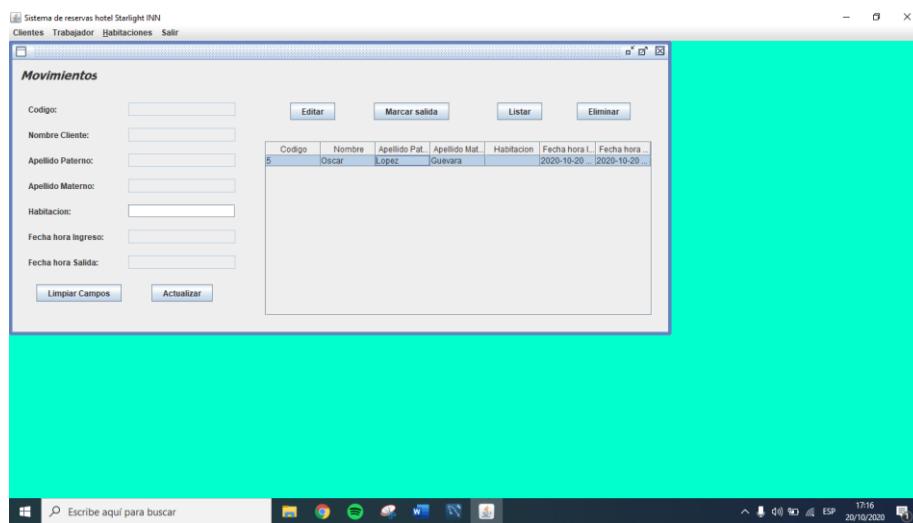


Resultado esperado anómalo:

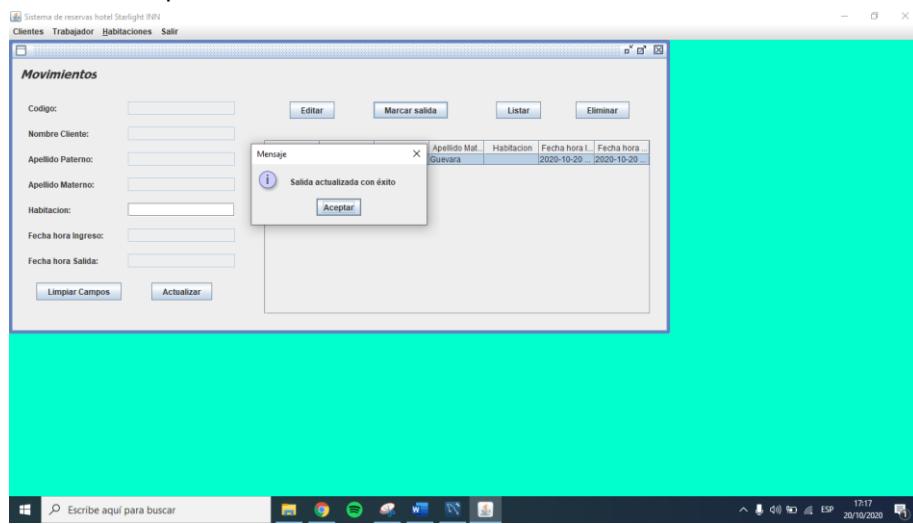


Caso de uso Modificar Salida de Cliente:

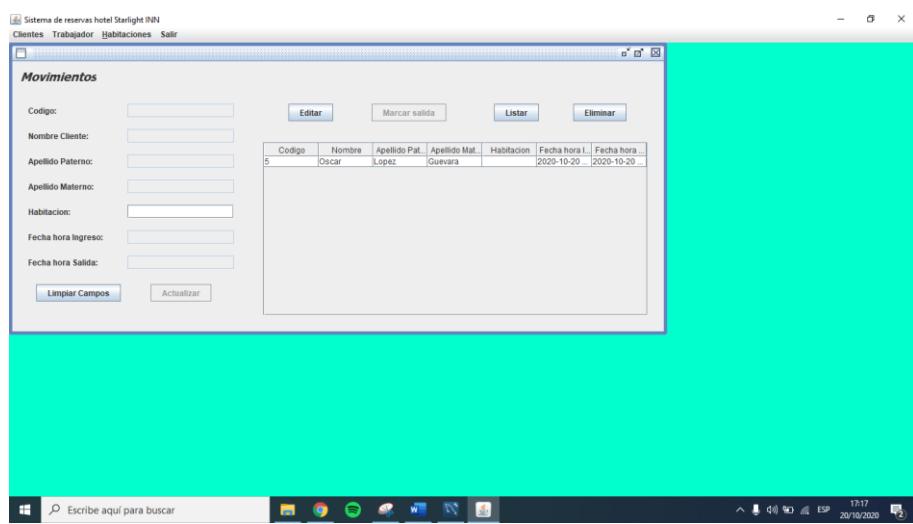
Datos de entrada normal:



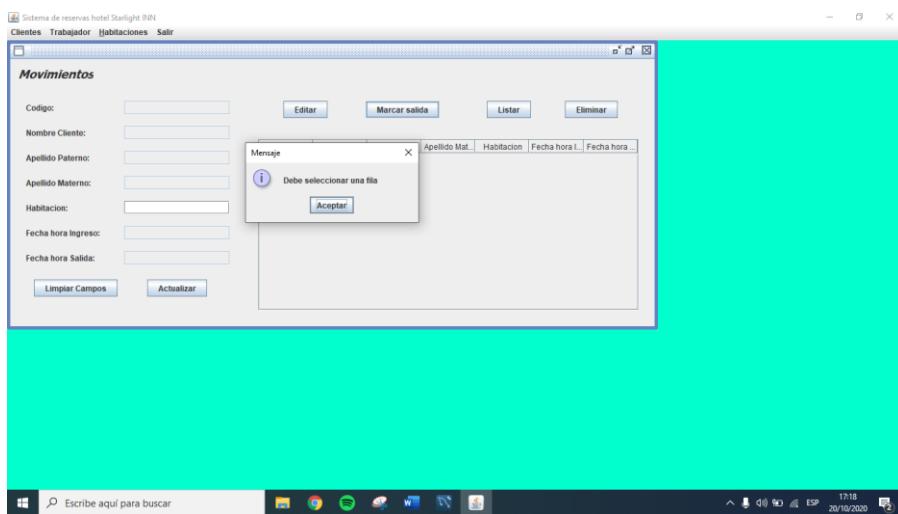
Resultado esperado normal:



Datos de entrada anómalo:



Resultado esperado anómalo:



Hallazgos

Clientes

- Tipo Documento acepta números cuando sólo debe aceptar texto
- Número de Documento acepta letras cuando sólo debe aceptar números
- Celular acepta letras cuando sólo debe aceptar números
- Email se registra en la base de datos aun cuando no se ingresa el @

Trabajador

- Cuando se ingresa letras en el campo Sueldo, no muestra mensaje de ayuda.
- Cuando se ingresa otro tipo de dato en Estado, no muestra mensaje de ayuda.
- Tipo de Documento acepta números cuando sólo debe aceptar texto.
- Número de Documento acepta letras cuando sólo debe aceptar números
- Celular acepta letras cuando sólo debe aceptar números
- Email se registra en la base de datos aun cuando no se ingresa el @
- Password se registra con un sólo valor, cuando debería haber un mínimo de 4 caracteres

MODELO FURPS+ APLICADO AL PROYECTO AL 27/10/2020

FACTOR	NRO	PREGUNTA	ARTEFACTO	SI CUMPLE	NO CUMPLE	ACCION CORRECTIVA	RESPONSABLE	FECHA SOLUCIÓN	EVIDENCIA
Funcionalidad versión o incremento 1	1	¿El software ejecuta el caso de uso Registro de trabajador?	Software funcionando	X					Sección ejecución del plan de pruebas
	2	¿El software ejecuta el caso de uso Modificar trabajador?	Software funcionando	X					Sección ejecución del plan de pruebas
	3	¿El software ejecuta el caso de uso Eliminar trabajador?	Software funcionando	X					Sección ejecución del plan de pruebas
	4	¿El software ejecuta el caso de uso Consulta de trabajadores?	Software funcionando	X					Sección ejecución del plan de pruebas
	5	¿El software ejecuta el caso de uso Validar trabajador?	Software funcionando	X					Sección ejecución del plan de pruebas
	6	¿El software ejecuta el caso de uso Registro de cliente?	Software funcionando	X					Sección ejecución del plan de pruebas
	7	¿El software ejecuta el caso de uso Modificar cliente?	Software funcionando	X					Sección ejecución del plan de pruebas
	8	¿El software ejecuta el caso de uso Eliminar cliente?	Software funcionando	X					Sección ejecución del plan de pruebas
	9	¿El software ejecuta el caso de uso Consulta de clientes?	Software funcionando	X					Sección ejecución del plan de pruebas
	10	¿El software ejecuta el caso de uso Registro de Entrada de cliente?	Software funcionando	X					Sección ejecución del plan de pruebas
	11	¿El software ejecuta el caso de uso Registro de Salida de cliente?	Software funcionando	X					Sección ejecución del plan de pruebas
	12	¿El software ejecuta el caso de uso Modificar Salida de cliente?	Software funcionando	X					Sección ejecución del plan de pruebas

Funcionalidad versión o incremento 2	13	¿El software ejecuta el caso de uso Registro de reserva?	Código fuente		X	Incluir caso de uso	Diego Tasaico	28/10/2020	Sección ejecución del plan de pruebas
	14	¿El software ejecuta el caso de uso Modificar reserva?	Código fuente		X	Incluir caso de uso	Diego Tasaico	28/10/2020	Sección ejecución del plan de pruebas
	15	¿El software ejecuta el caso de uso Eliminar reserva?	Código fuente		X	Incluir caso de uso	Mauricio Villanueva	28/10/2020	Sección ejecución del plan de pruebas
	16	¿El software ejecuta el caso de uso Consultar reserva?	Código fuente		X	Incluir caso de uso	Olenka Lazo	29/10/2020	Sección ejecución del plan de pruebas
	17	¿El software ejecuta el caso de uso Registro de habitación?	Código fuente		X	Incluir caso de uso	Olenka Lazo	29/10/2020	Sección ejecución del plan de pruebas
	18	¿El software ejecuta el caso de uso Modificar habitación?	Código fuente		X	Incluir caso de uso	Mauricio Villanueva	30/10/2020	Sección ejecución del plan de pruebas
	19	¿El software ejecuta el caso de uso Eliminar habitación?	Código fuente		X	Incluir caso de uso	Abihail Zapata	30/10/2020	Sección ejecución del plan de pruebas
	20	¿El software ejecuta el caso de uso Consultar habitación?	Código fuente		X	Incluir caso de uso	Abihail Zapata	30/10/2020	Sección ejecución del plan de pruebas

Usabilidad	21	¿Se muestra las imágenes de la carpeta de software y de informe en el GITHUB?	GITHUB	X					Sección repositorio Github
	22	¿Si los matices de color de texto y de fondo son cómodos para el usuario?	Software funcionando	X					
	23	¿Es de fácil uso para el usuario?	Software funcionando	X					
	24	¿Permite un fácil salida del software?	Software funcionando		X	Corregir la salida del sistema	Mauricio Villanueva	31/10/2020	Sección funcionamiento de software
	25	¿Se adapta al tipo de usuario?	Software funcionando	X					
Confiabilidad	26	¿Si el software controla el acceso no disponible de la Base de datos?	Software funcionando		X	Controlar el acceso a Base de Datos si no se conecta.	Mauricio Villanueva	31/10/2020	Sección funcionamiento de software
	27	Es de uso seguro para los usuarios?	Software funcionando		X	Corregir seguridad del sistema	Mauricio Villanueva	31/10/2020	Sección funcionamiento de software
	28	Es resistente ante fallos?	Software funcionando		X	Corregir la resistencia del sistema	Diego Tasaico	31/10/2020	Sección funcionamiento de software
	29	Todas las partes del sistema se encuentran conectadas entre sí?	Software funcionando	X					
	30	Tiene un comportamiento adecuado ante situaciones inusuales?	Software funcionando		X	Corregir la respuesta ante situaciones inusuales	Neftali Zapata	31/10/2020	Sección funcionamiento de software

Rendimiento	31	¿Si el tiempo de respuesta es menor a 7 segundos?	Software funcionando	X					
	32	¿Si el software ocupa espacio de memoria mínima?	Administrador de tareas del SO		X	Corrección de espacio de memoria	Neftali Zapata	31/10/2020	Sección de funcionamiento de software
	33	¿Disponible las 24 horas del día?	Software funcionando	X					
	34	¿Soluciona errores?	Software funcionando		X	Mejorar la solución de errores	Olenka Lazo	31/10/2020	Sección funcionamiento de software
	35	Detecta errores?	Software funcionando	X					
Soporte	36	¿Si el software cuenta con un procedimiento de compilación e instalación?	Procedimiento		X	Completar procedimiento	Mauricio Villanueva	31/10/2020	Sección procedimientos
	37	Cuenta con un resguardo de datos?	Software funcionando		X	Crear la opción de resguardo de datos	Diego Tasaico	31/10/2020	Sección funcionamiento de software
	38	Puede ejecutarse ante alguna condición extrema?	Software funcionando		X	Corregir al ejecutar	Olenka Lazo	31/10/2020	Sección funcionamiento de software
	39	Detecta errores en el sistema de manera rápida?	Software funcionando		X	Corregir el tiempo de solución	Neftali Zapata	31/10/2020	Sección funcionamiento de software
	40	Cuenta con una opción de ayuda al usuario?	Software funcionando		X	Implementar opción de ayuda	Diego Tasaico	31/10/2020	Sección funcionamiento de software

PLUS(+)	41	¿Se ha completado el marco teórico con citas y referencias?	Informe	X						Sección marco teórico del informe
	42	¿Se ha completado el plan de pruebas de la versión 1 y 2?	Informe	X						
	43	¿Se ha completado la ejecución de pruebas de la versión 1 y 2?	Informe	X						Sección ejecución de plan de pruebas
	44	¿Se ha completado las conclusiones y recomendaciones de la versión 2?	Informe		X	Completar conclusiones y recomendaciones de la versión 2	Olenka Lazo	31/10/2020		Sección conclusiones y recomendaciones
	45	Se ha completado plan de pruebas de la versión 3?	Informe		X	Completar plan de pruebas	Olenka Lazo	31/10/2020		Sección plan de pruebas

CAPÍTULO 4. CONCLUSIONES Y RECOMENDACIONES

12. Conclusiones

- Se presentó a la empresa una solicitud para obtener el permiso respectivo, mencionándole el objetivo del estudio. Una vez aceptado el documento se coordinó con el administrador, para la recolección de datos con el fin de obtener las facilidades pertinentes para llevar a cabo el estudio.
- La recopilación de información permitió comprender mejor el sistema, que después fue representado a través de los diagramas utilizados. De dicho análisis, del cual se pudo extraer los requerimientos del sistema que sirven de base para el desarrollo del mismo.
- Se desarrolló el sistema utilizando NetBeans IDE 8.2 como lenguaje de programación y SQL server para la gestión de la base de datos, logrando la automatización eficiente del sistema.
- Se utilizó adicionalmente, las herramientas informáticas: Lucidchart, MySQL Workbench y Balsamiq que nos permitió realizar los diagramas y prototipos, logrando así un mejor desarrollo y conceptualización básica que requiere el sistema.
- Por último, se cumplió con el objetivo del sistema, permitiendo así la implementación de sus respectivas opciones entre ellas mantenimiento de clientes, donde se podrá registrar, consultar, modificar y eliminar a los clientes, incluyendo la entrada y salida del cliente.

13. Recomendaciones

- Se recomienda continuar la realización de los próximos prototipos en la aplicación Balsamiq ya que nos permite comprender de manera visual el proyecto y explorar más opciones.

- Para seguir mejorando el sistema de administración hotelera se ha recomendado seguir solicitando información a la empresa para así culminar los objetivos proyectados.
- Al ser el presente proyecto, un sistema desarrollado a base del mantenimiento de registro de clientes y trabajadores, se recomienda la implementación del mantenimiento de las habitaciones y reservas para un mejor manejo.
- Se recomienda usar el mensaje de ayuda con el objetivo de resolver algún inconveniente que se presente.
- Y finalmente, se hará un énfasis a la revisión constante de las funciones y procesos, debido a que existirán cambios en cada una de las funcionalidades a fin de así poder mejorar el desempeño del sistema.

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>

MySQL. (s.f.). Obtenido de <https://www.mysql.com/>

MySQL WorkBench. (s.f.). Obtenido de <https://www.mysql.com/products/workbench/>

NetBeans. (s.f.). Obtenido de <https://netbeans.org/>

Sommerville, I. (2011). *Ingeniería de software (9a. ed.)*. Pearson Educación.

WampServer. (s.f.). Obtenido de <https://www.wampserver.com/en/>

15. Anexos



Ilustración 34: Videoconferencia con representante de la empresa



Ilustración 35: Integrante - Diego Tasaico



Ilustración 36: Integrante - Mauricio Villanueva



Ilustración 37: Integrante - Olenka Lazo



Ilustración 38: Integrante - Abihail Zapata



Pisco, 13 octubre del 2020

Srs. Representantes
Universidad Privada del Norte

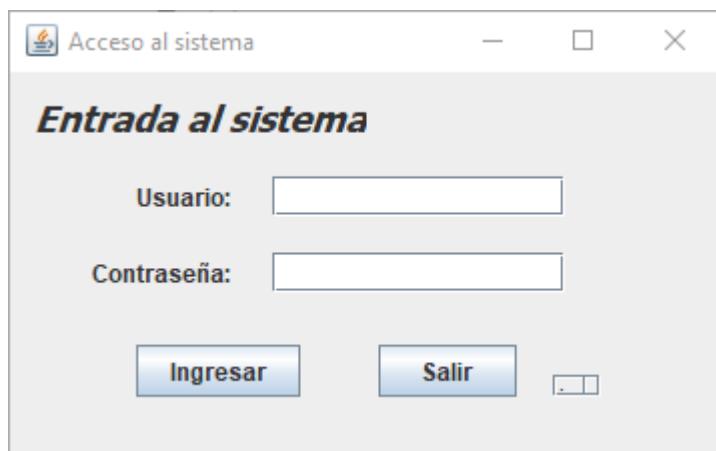
Con la finalidad de completar y modernizar nuestro sistema administrativo, se autoriza para que los jóvenes estudiantes (Diego Tasaico, Mauricio Villanueva, Abihail Zapata y Olenka Lazo), realicen un programa virtual que nos facilite nuestro registro diario que actualmente usamos. De ser necesario, se aprueba la visita a las instalaciones del HOTEL, así como cualquier otra información que ayude en la elaboración de este importante proyecto.

Atte.

Eduardo Enrique Albarracín Trillo
Administrador

Ilustración 39: Acta de compromiso

Vista de entrada al sistema:

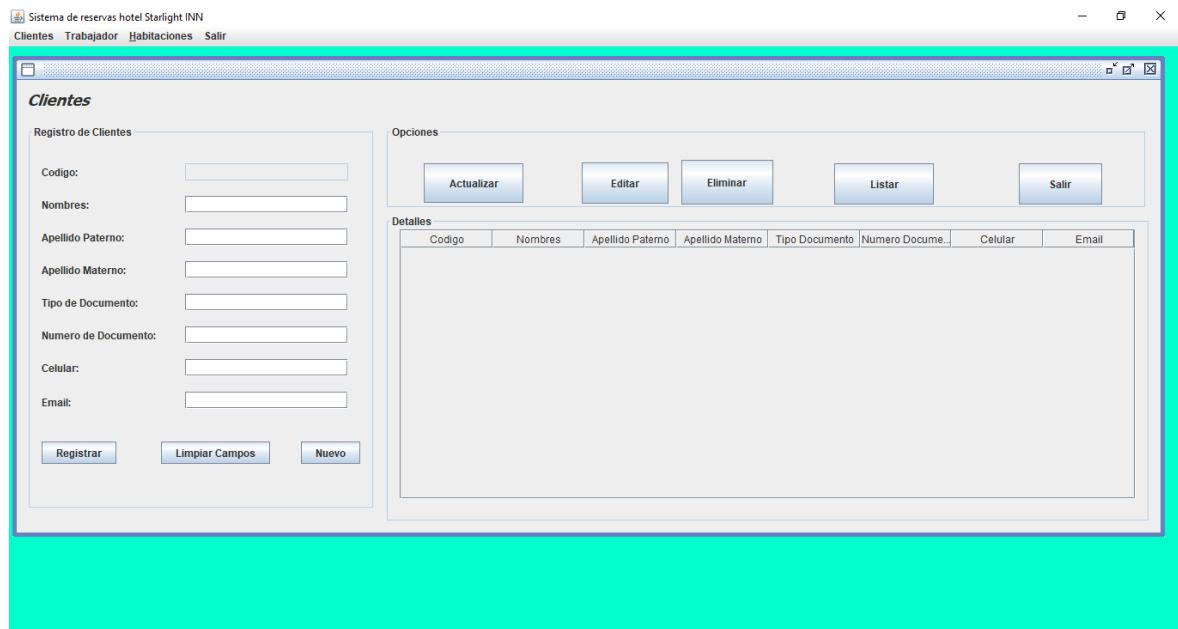


Menú principal:



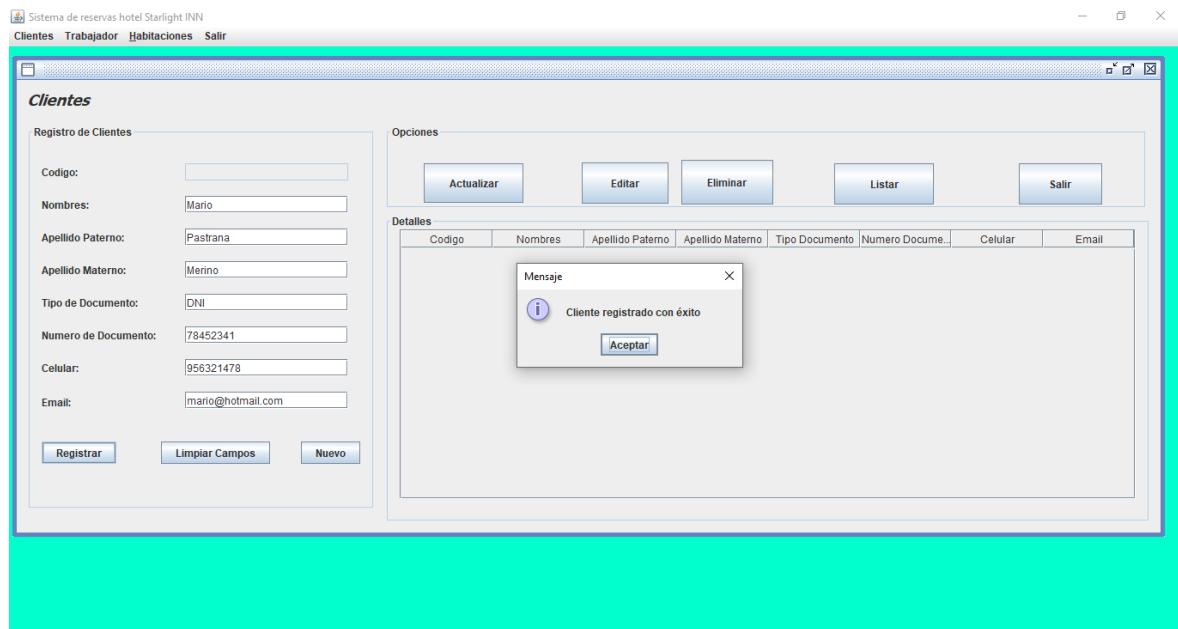
Ilustración 40: Funcionamiento del Software

Vista de los clientes:



The screenshot shows the 'Clientes' (Clients) screen of the hotel management system. On the left, there is a 'Registro de Clientes' (Client Registration) form with fields for Código, Nombres, Apellido Paterno, Apellido Materno, Tipo de Documento, Número de Documento, Celular, and Email. Below the form are three buttons: Registrar, Limpiar Campos, and Nuevo. On the right, there is an 'Opciones' (Options) panel with buttons for Actualizar, Editar, Eliminar, Listar, and Salir. Below the options is a 'Detalles' (Details) grid with columns for Código, Nombres, Apellido Paterno, Apellido Materno, Tipo Documento, Número Docume.., Celular, and Email. The grid currently contains no data.

Registrando cliente:



The screenshot shows the same 'Clientes' screen as the previous one, but with a success message displayed. The registration form has been filled with sample data: Código (empty), Nombres (Mario), Apellido Paterno (Pastrana), Apellido Materno (Merino), Tipo de Documento (DNI), Número de Documento (78452341), Celular (956321478), and Email (mario@hotmail.com). After clicking the 'Registrar' button, a modal message box appears stating 'Mensaje' (Message) with the text 'Cliente registrado con éxito' (Client registered successfully) and an 'Aceptar' (Accept) button.

Sistema de reservas hotel Starlight INN

Ciudades Trabajador Habitaciones Salir

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

Actualizar Editar Eliminar Listar Salir

Detalles

Código	Nombres	Apellido Paterno	Apellido Materno	Tipo Documento	Número Docume.	Celular	Email
2	Mario	Pastrana	Merino	DNI	78452341	956321478	mario@hotmail.c...

Registrar Limpiar Campos Nuevo

Modificar Cliente:

Sistema de reservas hotel Starlight INN

Ciudades Trabajador Habitaciones Salir

Clientes

Registro de Clientes

Código:	<input type="text" value="2"/>
Nombres:	<input type="text" value="Mario"/>
Apellido Paterno:	<input type="text" value="Pastrana"/>
Apellido Materno:	<input type="text" value="Merino"/>
Tipo de Documento:	<input type="text" value="DNI"/>
Número de Documento:	<input type="text" value="78452341"/>
Celular:	<input type="text" value="956321478"/>
Email:	<input type="text" value="mario@hotmail.com"/>

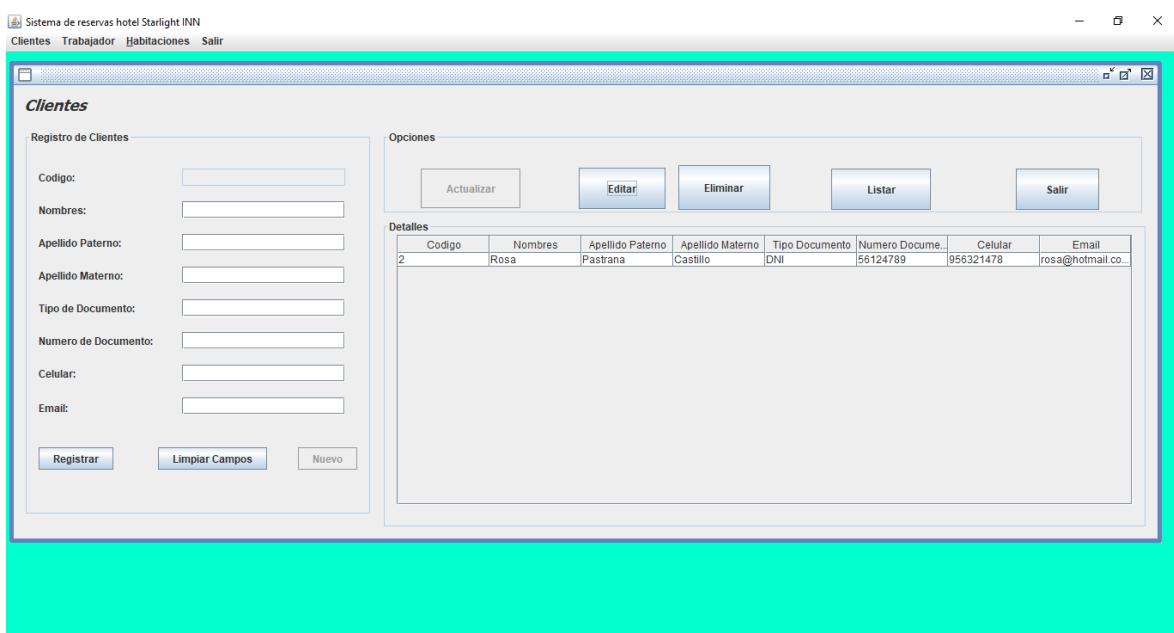
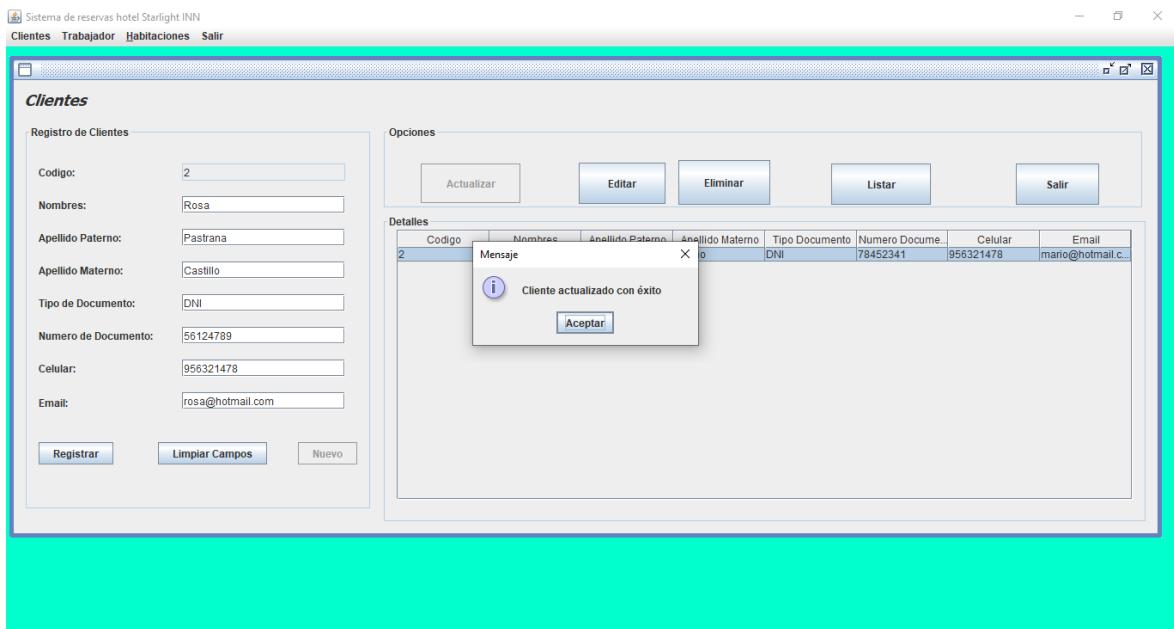
Opciones

Actualizar Editar Eliminar Listar Salir

Detalles

Código	Nombres	Apellido Paterno	Apellido Materno	Tipo Documento	Número Docume.	Celular	Email
2	Mario	Pastrana	Merino	DNI	78452341	956321478	mario@hotmail.c...

Registrar Limpiar Campos Nuevo



Eliminando Cliente:

Sistema de reservas hotel Starflight INN

Ciudades Trabajador Habitaciones Salir

Clientes

Registro de Clientes								
Código:								
Nombres:								
Apellido Paterno:								
Apellido Materno:								
Tipo de Documento:								
Número de Documento:								
Celular:								
Email:								
<input type="button" value="Registrar"/>	<input type="button" value="Limpiar Campos"/>	<input type="button" value="Nuevo"/>						

Opciones

Detalles

Código	Nombres	Apellido Paterno	Apellido Materno	Tipo Documento	Número Docume..	Celular	Email
2	Rosa	Pastrana	Castillo	DNI	56124789	956321478	rosa@hotmail.co...

Sistema de reservas hotel Starflight INN

Ciudades Trabajador Habitaciones Salir

Clientes

Registro de Clientes								
Código:								
Nombres:								
Apellido Paterno:								
Apellido Materno:								
Tipo de Documento:								
Número de Documento:								
Celular:								
Email:								
<input type="button" value="Registrar"/>	<input type="button" value="Limpiar Campos"/>	<input type="button" value="Nuevo"/>						

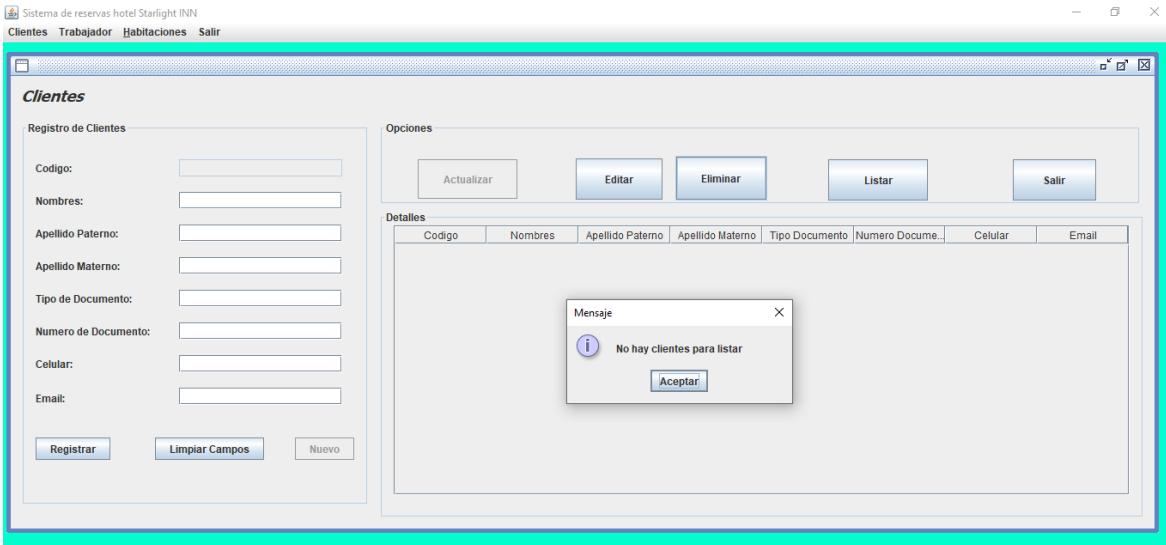
Opciones

Detalles

Código	Nombres	Apellido Paterno	Apellido Materno	Tipo Documento	Número Docume..	Celular	Email
2	Mensaje	X	o	DNI	56124789	956321478	rosa@hotmail.co...

Mensaje

Cliente eliminado con éxito



Clients

Registro de Clientes

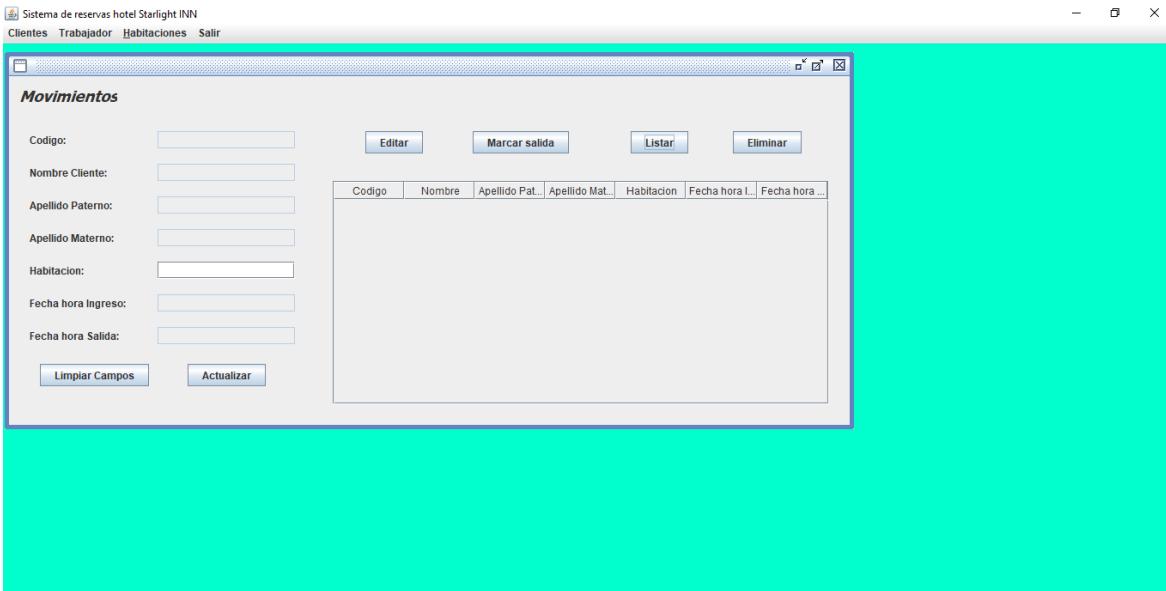
Código:	<input type="text"/>
Nombres:	<input type="text"/>
Apellido Paterno:	<input type="text"/>
Apellido Materno:	<input type="text"/>
Tipo de Documento:	<input type="text"/>
Número de Documento:	<input type="text"/>
Celular:	<input type="text"/>
Email:	<input type="text"/>

Opciones

Detalles

Código	Nombres	Apellido Paterno	Apellido Materno	Tipo Documento	Número Docume..	Celular	Email
Mensaje							
No hay clientes para listar							
<input type="button" value="Aceptar"/>							

Vista Movimientos:



Movimientos

Código:

Nombre Cliente:

Apellido Paterno:

Apellido Materno:

Habitación:

Fecha hora Ingreso:

Fecha hora Salida:

Código	Nombre	Apellido Pat.	Apellido Mat.	Habitación	Fecha hora I.	Fecha hora S.