



# UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

## Facultad de Estudios Superiores Aragón



### Alumnos:

- Ramírez González Jean Brandon
- Rivas Pérez Fernando
- Maldonado Arrieta Johan Uriel
- Tena Rodríguez Edson Alejandro

### Proyecto final

**Materia:** Base de datos 1

**Grupo:** 2410

**Profesor(a):** Miguel Ángel Sánchez Hernández

# Índice

- Objetivos-----	2
- Introducción-----	2
- Desarrollo-----	3
- Diagrama Relacional-----	7
- SQL DDL-----	8
- Manual-----	10
-Abrir-----	10
-Insertar-----	11
-Modificar-----	14
-Eliminar-----	17
- Conclusiones-----	18
- Bibliografía-----	19

## Objetivos

- Utilizar una aplicación que se conecte a una base de datos que pueda hacer CRUD (Crear, Leer, Actualizar y Eliminar).
- Crear un manual que especifique de manera clara la manera en que funciona la aplicación.
- Realizar una función que nos permita la creación de las entidades Hotel, Gerente y Habitaciones, así como la función de eliminación de Hotel.
- La creación de query's para crear hotel, habitación y gerente y eliminar hotel.
- La especificación del diagrama relacional de forma que logremos realizar una descripción breve pero útil de la estructura de nuestra base de datos.
- Aplicar todos los conocimientos aprendidos en el curso de Base de Datos I y reforzar los conocimientos de JAVA.

## Introducción

Para este proyecto se realizará una base de datos funcional la cual trabaje en complemento de una aplicación la cual cubrirá las necesidades de un negocio, es decir, nos ayuda a solucionar un problema. La resolución del problema principal es la creación de una base de datos para un hotel la cual funcione a través de una interfaz, que cualquier empleado sin conocimiento de programación pueda entender, además de brindar la documentación necesaria en caso de ser requerida (para un programador).

Para lograr esto se hará necesario la creación de la aplicación con JAVA donde crearemos métodos y funciones CRUD (Crear, Leer, Actualizar, Eliminar) que nos ayuden a realizar lo que requerimos, también será necesaria la creación de una conexión con nuestra base de datos con Mysql Workbench. Revisaremos y resolveremos todos aquellos problemas que vayan surgiendo durante la implementación de nuestras ideas, nos apoyaremos en algunas funciones ya existentes del código JAVA, así como de sintaxis de SQL. El manejo de consulta SQL es importante para el uso a través de una aplicación, así como la correcta aplicación de

los query's para obtener los mejores resultados posibles en cuanto eficiencia del manejo de datos.

Explicaremos y mostraremos nuestro diagrama relacional para que así se pueda entender y explicar de forma sencilla nuestra base de datos por medio de DDL (lenguaje de definición de datos) así de cómo se creó, también de cómo se crearon sus CONSTRAINTS si son PK, FK, también si son Nullable o No Nullable y su tipo de dato.

La realización de un manual para el entendimiento de la aplicación es decir su interfaz y cómo funciona los procedimientos que hace, los métodos que llama, los query's que ocupa y cómo se implementan estos a la base de datos. Un manual ilustrado para que sea fácil de entender y por supuesto práctico.

Este proyecto aplicará los conocimientos aprendidos a lo largo del curso, será un reto difícil ya que será aplicarlo en una aplicación JAVA y de tal manera hacer las interconexiones y lo demás es decir, conocimientos de programación general, así como bases de datos e interfaces gráficas.

## Desarrollo

Para comenzar con nuestro proyecto requerimos de varios aspectos que harán funcional nuestra base de datos unificada a nuestra aplicación. Como primer punto se creó el paquete **fes.aragon.mysql** en donde creamos la clase: **Conexion.java** la cual tiene la función de crear la interconexión entre la aplicación y la base de datos Mysql, importamos los controladores proporcionados por la misma SQL los cuales nos ayudaran en las conexiones, creamos la conexión gracias a jdbc proporcionando los datos URL, Usuario y clave (de conexión para la base de datos).

Creamos el paquete **fes.aragon.interfaz** que contiene la clase **IBaseDatos** en el cual se agregó los métodos necesarios para las demás clases en este caso fue **Consulta** (y consulta de id), **Buscar**, **Insertar**, **Modificar**, **consulta** (para id), **eliminar** y **eliminarProc**. Al momento de la creación del código nos encontramos con algunos problemas en este caso para aplicar el método Insertar, insertar hoteles, gerentes y habitaciones.

Retomamos el paquete **fes.aragon.mysql** en donde se creó la clase: **HotelesImp** donde se heredaron los métodos de **IBaseDatos** y se crearon los *getter and setters* y en los métodos

de consulta insertar modificar y eliminar se agregaron query's necesarios para realizar consultas desde una base de datos.

En el paquete **fes.aragon.mysql** se creó la clase **habitacionesImpl** heredaron los métodos de **IBaseDatos** del mismo paquete y se crearon los *getter and setters* y en los métodos de consulta insertar modificar y eliminar se agregaron query's necesarios para realizar consultas desde una base de datos.

En el paquete **fes.aragon.mysql** se creó la clase **Tipolmp** que complementa a las clases **habitacionesImpl** del mismo paquete la cual heredó los métodos de **IBaseDatos** y se crearon los *getter and setters* y en los métodos de consulta insertar modificar y eliminar se agregaron query's necesarios para realizar consultas desde una base de datos.

En el paquete **fes.aragon.mysql** se creó la clase **GerentesImpl** del mismo paquete la cual heredó los métodos de **IBaseDatos** y se crearon los *getter and setters* y en los métodos de consulta insertar modificar y eliminar se agregaron query's necesarios para realizar consultas desde una base de datos.

Para la implementación de una aplicación y una base de datos, es necesaria una aplicación la cual ya tenemos realizada antes de comenzar la base de datos, esta aplicación tiene funcionalidades controlar la información para esto se crea un paquete **fes.aragon.controlador** donde tenemos la clase **BaseController** la cual nos ayudara en esto utilizando los métodos: **Expresiones**: Nos permite verificar que los datos sólo puedan utilizar cierto tipo de expresiones; **NuevaVentana**: Abre una nueva ventana donde corre cada escena necesaria; **ventanaEmergente**: Esta funciona para avisar sobre posibles errores de sintaxis o algún dato faltante; **CerrarVentana**: Cierra las ventanas; **verificarEntrada**: verifica las expresiones y evita que existan errores; **recogerDatos**: asigna los valores para cada objeto Hotel; **guardarArchivo**: Guarda un archivo que puede ser leído por java.

En el mismo paquete **fes.aragon.controlador** creamos **gerenteController** que extiende a **BaseController** e implementa **Initializable** la cual utiliza los métodos: **Cancelargerente**: cancela la creación de un nuevo gerente; **nuevoGerente**: crea un nuevo gerente; **Initialize**: llama a todos los datos de inicialización; **Verificar**: Ayuda a evitar que no haya espacios en blanco.

En el mismo paquete **fes.aragon.controlador** creamos **habitacionController** que extiende a **BaseController** e implementa **Initializable** la cual utiliza los métodos:

**Cancelarhabitacion:** cancela la creación de un nuevo gerente; **nuevahabitacion:** crea una nueva habitación; **Initialize:** llama a todos los datos de inicialización; **Verificar:** Ayuda a evitar que no haya espacios en blanco.

En el mismo paquete **fes.aragon.controlador** creamos **hotelController** que extiende a **BaseController** e implementa **Initializable** la cual utiliza los métodos:

**Cancelarhotel:** cancela la creación de un nuevo gerente; **nuevaHabitacion:** abre la ventana de nueva habitación; **nuevoGerente:** abre la ventana de nuevo gerente; **nuevohotel:** crea un nuevo hotel; **Initialize:** llama a todos los datos de inicialización; **Verificar:** Ayuda a evitar que no haya espacios en blanco.

En el mismo paquete **fes.aragon.controlador** creamos **modificarHabitacion** que extiende a **BaseController** e implementa **Initializable** la cual utiliza los métodos:

**modificar:** modifica la habitación; **cancelar:** cancela la modificación de una habitación; **nuevo:** sirve para acomodar los hoteles en la interfaz; **Initialize:** llama a todos los datos de inicialización; En el mismo paquete **fes.aragon.controlador** creamos **inicioController** que extiende a **BaseController** e implementa **Initializable** la cual utiliza los métodos: **Eliminarhotel:** Elimina el hotel de la interfaz; **modificarHotel:** permite seleccionar el hotel y llama el método modificar y trae la instancia correspondiente al hotel; **nuevoHotel:** Llama a las instancias para el hotel correspondiente y aplica los métodos correspondientes; **salir:** Nos permite salir de la aplicación; **Initialize:** llama a todos los datos de inicialización; **abrirArchivo:** realiza la lectura de una copia de la base de datos con ayuda del método **recogerDatos**; **recogerDatos** lee y agrega los datos del objeto hotel; **guardarArchivo:** guarda el archivo en formato java; **Deshabilitar:** una bandera para habilitar o deshabilitar los botones.

Para poder consultar la información debíamos pensar en una manera que no fuera tan complicada, lo que se ocurrió fue hacer un método de consulta en **HotelesImp** el cual nos ayudará a realizar una consulta de todo lo que necesitábamos; **Hotel:** lo que hicimos fue pedir el id nombre dirección correo y teléfono del hotel, **Gerente:** lo que hicimos fue similar pedir su atributos y **Habitación** fue buscar todas las habitaciones que hay en un hotel.

La sintaxis que debíamos ocupar era muy sencilla la simple implementación de un query en cada clase **GerenteImp**, **habitacionesImp**, **HotelesImp** que nos ayudara con esta función. A nivel de interfaz se podían registrar los datos, pero estos en realidad no se quedaban grabados en la base de datos por lo que había que buscar otro método

Nuestra nueva idea para lograr la inserción de los datos fue realizar modificaciones en unas partes del código que nos permitieran usar el método insertar para cada clase Hotel, Gerente y Habitaciones, Los cambios importantes fueron realizados en las clases: **GerenteImpl**, **HabitacionesImpl** y **HotelesImpl**, debido a que todas implementan la interfaz **IBaseDatos**.

Modificación en el método insertar: Cada clase tiene una copia del método insertar por lo que podemos realizar una sola modificación y posteriormente adaptarla para cada instancia de la clase ya que se recibe el objeto, se implementa el query anteriormente creado de **insert into**, agregamos también los campos correspondientes del objeto, y para poderlo actualizar junto con la base de datos se ocupó el método **PreparedStatement** que nos permite realizar una solicitud y ahora en conjunto con el otro método **ReturnGeneratedKeys** el cual nos retorna valores creados automáticamente. Para garantizar que todos los campos se llenaran correctamente, había que asegurarnos que los objetos tuvieran toda la información para ese momento, incluyendo los id, por lo que analizamos la forma en que se crea cada uno y el orden en el que se insertan.

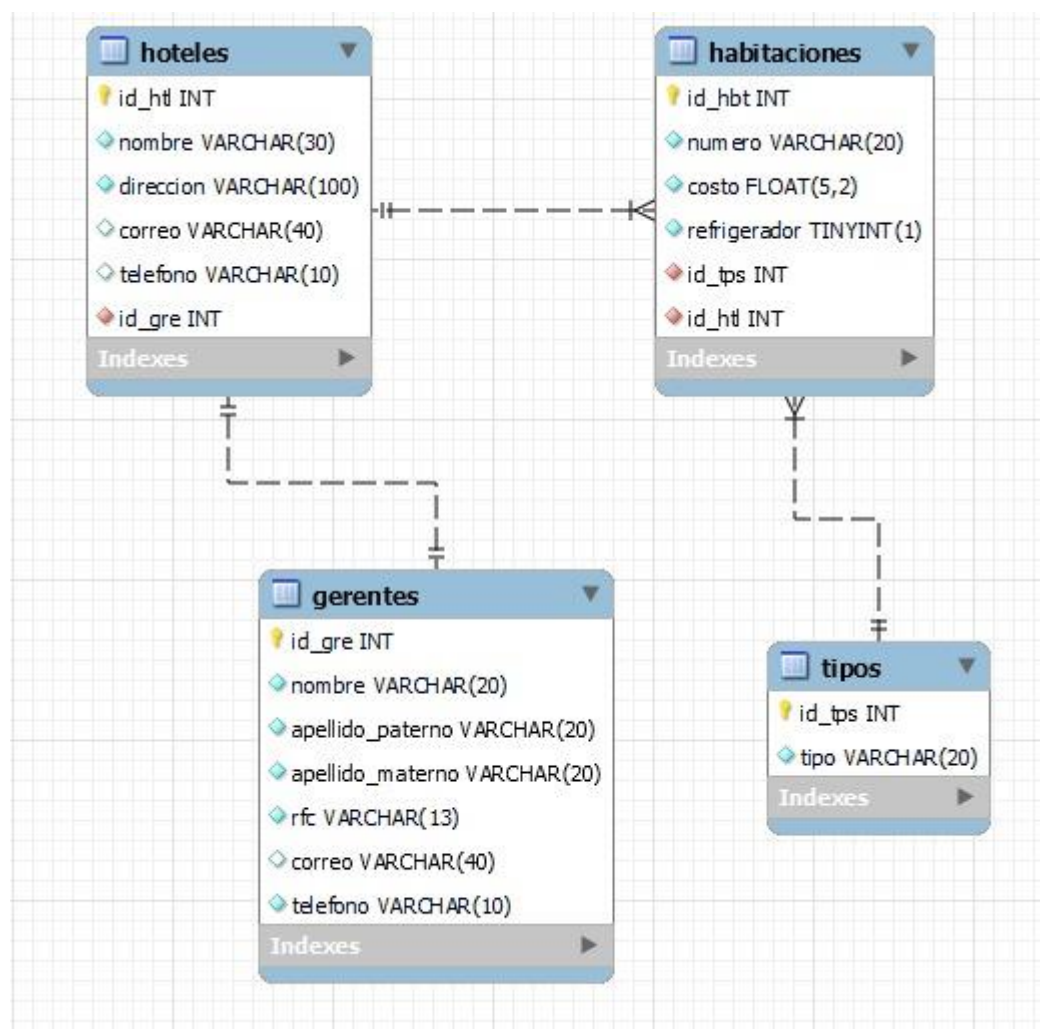
Lo que se realizó en esta parte fue la modificación del paquete **fes.aragon.controlador** cambiando partes de las clases: **GerenteController**, **HabitacionController** y **HotelController**, La mayoría de cambios fueron aplicados para **HotelController**, ya que los insert fueron creados en su método nuevoHotel siempre respetando el orden, La función que realiza este método es verificar con una bandera si se trata de una modificación o de un nuevo hotel, cuando la bandera nos indica que es una modificación se creó un ciclo for que inserta nuevas habitaciones en caso de que se hayan añadido nuevas en la modificación. En el caso alterno que la bandera nos indique que se trata de un nuevo hotel se realizarán todos los **insert** con el orden: **gerente**, **hotel** y un for para **habitaciones**, Fue necesario realizarlo de esta manera ya que requerimos insertar primero las tablas fuertes y que los demás conozcan el id necesario para su inserción.

Para la clase **GerenteController** sólo se incluyó un if para verificar si es un gerente ya existente o no, de una manera similar en **HabitacionController** se incluyó un procedimiento if para saber si la **habitación** ya había sido creada (esta corresponde a un hotel por lo que debería ser parte de la modificación de un hotel, en caso de que sea así, se modificaría directamente en la Base de datos, de lo contrario solo se modificaría en la aplicación de modo que espera a que se cree un hotel para poder ser insertados en la base de datos. Una vez implementados estos cambios y haber llenado todos los datos, la aplicación ya conectada a la base realizaría la subida de todos los datos que se le proporcionaron y estos se insertarán en la base de datos con sus respectivas referencias.

Para poder realizar la actualización de los datos en **hotelesImpl**, **habitacionesImpl** y **GerenteImpl** en cada uno de estos se llamó al método modificar el cual obtiene los datos que el usuario quiere modificar o que ya estaban y posteriormente pide los nuevos datos para así ser reemplazados.

En la parte de eliminar tuvimos una problemática en donde pensábamos que con nada más eliminar Hotel ya sería suficiente para la eliminación del campo pero no fue así, lo que hicimos para eliminar fue algo parecido, esta vez en las clases **GerenteImpl**, **HabitacionesImpl** y **HotelesImpl** específicamente en sus métodos eliminar se hicieron query delete, y en **InicioController** (método **eliminarHotel**) se mandaron a llamar los métodos eliminar en el siguiente orden: Habitaciones, hoteles y gerentes, por las referencias que tenían, pasando como argumento del método el hotel seleccionado por el usuario.

### Diagrama relacional





Vamos a utilizar este diagrama relacional el cual explicaremos en términos de cardinalidad con la que funciona la base de datos. Comenzamos con la creación de tablas fuertes primera “Gerentes”, después “Tipos” y por último las que no son tablas fuertes, son “Hoteles” y “Habitaciones”, Lo sabemos ya que estas tablas tienen llaves foráneas.

Una vez creada las tablas hemos de encontrar la forma en funcionar:

- Gerentes y Hoteles (1:1): Este decidimos relacionarlo de uno a uno, ya que el gerente es parte fundamental de un hotel, ya que un hotel tiene la clave foránea de gerente (el id del gerente), a su vez un gerente está asignado a un hotel, y un hotel solo cuenta con un gerente, es por eso que la relación es 1:1.

- Tipo y Habitaciones (1:N): Esta cardinalidad fue importante tenerla en cuenta al inicio ya que estas dos tablas desde su creación llevan relación, ya que “Tipos” define a las “Habitaciones”, ya que una “Habitación” tiene un “Tipo” y un “Tipo” pertenece a varias “Habitaciones”, un ejemplo sería que una habitación tiene un cuarto matrimonial, pero hay cinco habitaciones matrimoniales en el hotel.

- Hoteles y Habitaciones (1:N): La última cardinalidad, de uno a muchos, ya que hablamos del “Hotel” y sus “Habitaciones”, donde cada “Hotel” tiene muchas “Habitaciones” y varias “Habitaciones” pertenecen a un “Hotel”

## SQL DDL

El SQL DDL que utilizamos para crear la base de datos es el siguiente:

```
SET NAMES 'utf8';
DROP DATABASE IF EXISTS hotel;
CREATE DATABASE IF NOT EXISTS hotel DEFAULT CHARACTER SET utf8;
USE hotel;

CREATE TABLE gerentes(
    id_gre                                INT NOT NULL AUTO_INCREMENT,
    nombre                               VARCHAR(20) NOT NULL,
    apellido_paterno                      VARCHAR(20) NOT NULL,
    apellido_materno                      VARCHAR(20) NOT NULL,
    rfc                                   VARCHAR(13) NOT NULL,
    correo                               VARCHAR(40) BINARY UNIQUE,
    telefono                             VARCHAR(10) NOT NULL,
    PRIMARY KEY(id_gre),
    CONSTRAINT rango_correo_gerente CHECK (correo LIKE '%@%.%'),
    CONSTRAINT apellido_paterno CHECK (REGEXP_LIKE(apellido_paterno, '[A-Z]')),
    CONSTRAINT apellido_materno CHECK (REGEXP_LIKE(apellido_materno, '[A-Z]'))
)DEFAULT CHARACTER SET utf8;
```

```

CREATE TABLE hoteles(
id_htl                                INT NOT NULL AUTO_INCREMENT,
nombre                               VARCHAR(30) NOT NULL,
direccion                             VARCHAR(100) NOT NULL,
correo                               VARCHAR(40) BINARY UNIQUE,
telefono                             VARCHAR(10) UNIQUE,
id_gre                               INT NOT NULL,
PRIMARY KEY(id_htl),
CONSTRAINT rango_correo_hotel CHECK (correo LIKE '%@%.%'),
CONSTRAINT nombre CHECK (REGEXP_LIKE(nombre,'[A-Z]')),
FOREIGN KEY(id_gre) REFERENCES gerentes(id_gre)
)DEFAULT CHARACTER SET utf8;

CREATE TABLE tipos(
id_tps                                INT NOT NULL AUTO_INCREMENT,
tipo                                 VARCHAR(20) NOT NULL,
PRIMARY KEY(id_tps)
)DEFAULT CHARACTER SET utf8;
INSERT INTO tipos(tipo) VALUES('INDIVIDUAL');
INSERT INTO tipos(tipo) VALUES('MATRIMONIAL');
INSERT INTO tipos(tipo) VALUES('MIXTO');

CREATE TABLE habitaciones(
id_hbt                                INT NOT NULL AUTO_INCREMENT,
numero                               VARCHAR(20) NOT NULL,
costo                                FLOAT(5,2) NOT NULL,
refrigerador                         BOOLEAN NOT NULL DEFAULT FALSE,
id_tps                               INT NOT NULL,
id_htl                               INT NOT NULL,
PRIMARY KEY(id_hbt),
FOREIGN KEY(id_tps) REFERENCES tipos(id_tps),
FOREIGN KEY(id_htl) REFERENCES hoteles(id_htl)
)DEFAULT CHARACTER SET utf8;

insert into gerentes(nombre,apellido_paterno,apellido_materno,rfc,correo,telefono)
values('G1','G1','G1','asdfghjklwert','demo.com@demo.com','1234456781');
insert into hoteles(nombre,direccion,correo,telefono,id_gre)
values('H1','Demo','h1@demo.com','3424232525',1);
insert into habitaciones(numero,costo,refrigerador,id_tps,id_htl) values('A1',23.23,true,1,1);

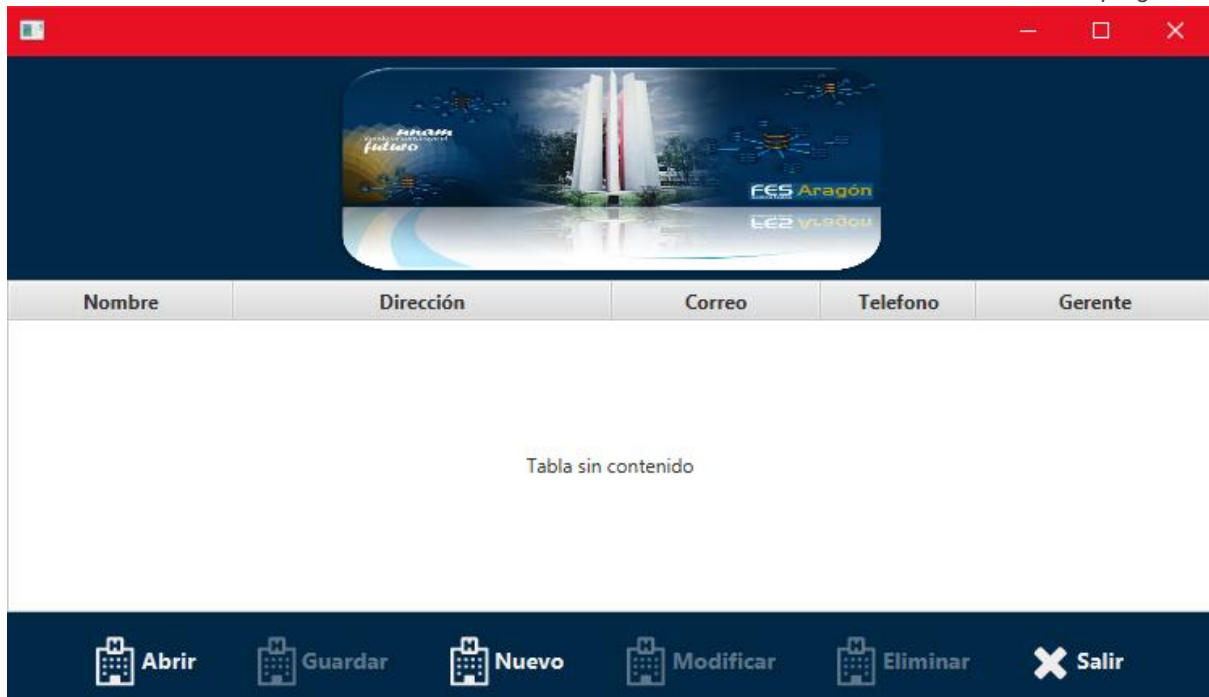
```

## Manual

### Abrir

Primero el usuario debe correr la aplicación Hotel, esperar unos segundos y posteriormente se abrirá la aplicación de Hotel.

*Interfaz del programa.*



Después selecciona el apartado de “Abrir” este apartado invoca el procedimiento de consulta el cual es un query que llama a la base de datos Hotel y específicamente a las tablas de hoteles, gerentes y habitaciones donde realiza la consulta gracias a los siguientes query’s “select \* from hoteles a,gerentes b where a.id\_gre=b.id\_gre” y “select \* from tipos”;


*Interfaz del programa después del botón abrir.*



## Insertar

Para agregar un nuevo hotel se necesita agregar un nuevo gerente y nuevas habitaciones para el hotel. Para esto presionamos el botón “*Nuevo*”. Una vez presionado se abrirá otra ventana que contendrá los campos para la creación de un nuevo hotel.

*Interfaz de Nuevo Hotel*

A screenshot of a web form titled "Hotel". It contains six input fields: "Nombre:", "Dirección:", "Correo:", "Teléfono:", "Gerente:", and "Habitación:". The "Gerente:" and "Habitación:" fields have dropdown menus with the labels "Gerente" and "Habitación" respectively. At the bottom, there are two buttons: "Aceptar" and "Cancelar".

**Hotel**

Nombre:

Dirección:

Correo:

Teléfono:

Gerente:

Habitación:

Para llenar estos campos será necesario hacerlo adecuadamente ya que sino el sistema no te permitirá registrar esos datos. Una vez agregados los campos en blanco debemos presionar el botón “*Aceptar*” lo cual nos permite guardar el Hotel esto llama al método haciendo uso del query: “**insert into hoteles(nombre,dirección,correo,telefono,id\_gr)** **values(?,?,?,?,?)**”;. ¡OJO! El programa no permite agregar el hotel sin haber agregado un gerente nuevo por lo que será necesario crear uno nuevo.

Para agregar un nuevo gerente presionará en el apartado “*Gerente*” Una vez presionado se abrirá otra ventana que contendrá los campos para la creación de un nuevo gerente.

*Interfaz de Nuevo Gerente*

A screenshot of a web form titled "Nuevo Gerente". It contains six input fields: "Nombre:", "Apellido Paterno:", "Apellido Materno:", "RFC:", "Correo:", and "Teléfono:". At the bottom, there are two buttons: "Aceptar" and "Cancelar".

**Nuevo Gerente**

Nombre:

Apellido Paterno:

Apellido Materno:

RFC:

Correo:

Teléfono:

Al igual que en el nuevo Hotel al momento de llenar los campos hay que tener en cuenta los requisitos, de no ser así el sistema no te dejará agregar al gerente. Una vez agregados los datos en los campos correspondientes y sin errores bastará con presionar en el apartado “*Aceptar*” lo cual guarda los datos utilizando el query: **“insert into gerentes(nombre,apellido\_paterno,apellido\_materno,rfc,correo,telefono) values(?,?,?,?,?,?)”**;. Una vez presionado el botón aceptar se regresa a la interfaz de hotel donde podrás agregar una nueva habitación.

*Interfaz de Nuevo Hotel*

A screenshot of a web form titled "Hotel". It contains several input fields: "Nombre:", "Dirección:", "Correo:", and "Teléfono:". Below these are two buttons labeled "Gerente" and "Habitación". At the bottom are "Aceptar" and "Cancelar" buttons. The form is styled with a light gray background and white input fields.

Para agregar un nueva habitación presionará en el apartado “*Habitación*” Una vez presionado se abrirá otra ventana que contendrá los campos para la creación de una nueva habitación.

*Interfaz de Nueva Habitación*

A screenshot of a web form titled "Nueva Habitación". It contains input fields for "Número:", "Costo:", and "Refrigerador:" (with a checkbox). There is also a dropdown menu for "Tipo:" with the text "Selecciona un tipo". At the bottom are "Aceptar" and "Cancelar" buttons. The form has a light gray background.

Al igual que con el nuevo gerente al momento de llenar los campos hay que tener en cuenta los requisitos, de no ser así el sistema no te dejará agregarla habitación en este caso hay

casilla opcional y tipo predeterminados solo basta con seleccionar uno de ellos. Una vez agregados los datos en los campos correspondientes y sin errores bastará con presionar en el apartado “Aceptar” lo cual guarda los datos utilizando el query: **“insert into habitaciones(numero,costo,refrigerador,id\_tps,id\_htl) values(?,?,?,?,?)”;**.

Una vez presionado el botón “*aceptar*” se regresa a la interfaz de hotel donde podrás agregar una nueva habitación o confirmar el hotel. Para completar la creación del nuevo hotel solo queda presionar el apartado aceptar en la interfaz de Hotel.

*Interfaz de Nuevo Hotel*

Ahora puedes observar tu nuevo hotel en el cuadro con sus respectivos atributos ordenados

*Interfaz del programa después del botón aceptar.*

Nombre	Dirección	Correo	Telefono	Gerente
NuevoHotel	DireccionNueva	correo@correo.com	2413355613	Martin

Cargar
 Guardar
 Nuevo
 Modificar
 Eliminar
 Salir

Y si observamos en nuestra base de datos, ya tendremos los datos insertados:

*Datos insertados en la base de datos*

	id_htl	nombre	direccion	correo	telefono	id_gre
▶	1	NuevoHotel	DireccionNueva	correo@correo.com	2413355613	1
*	NULL	NULL	NULL	NULL	NULL	NULL

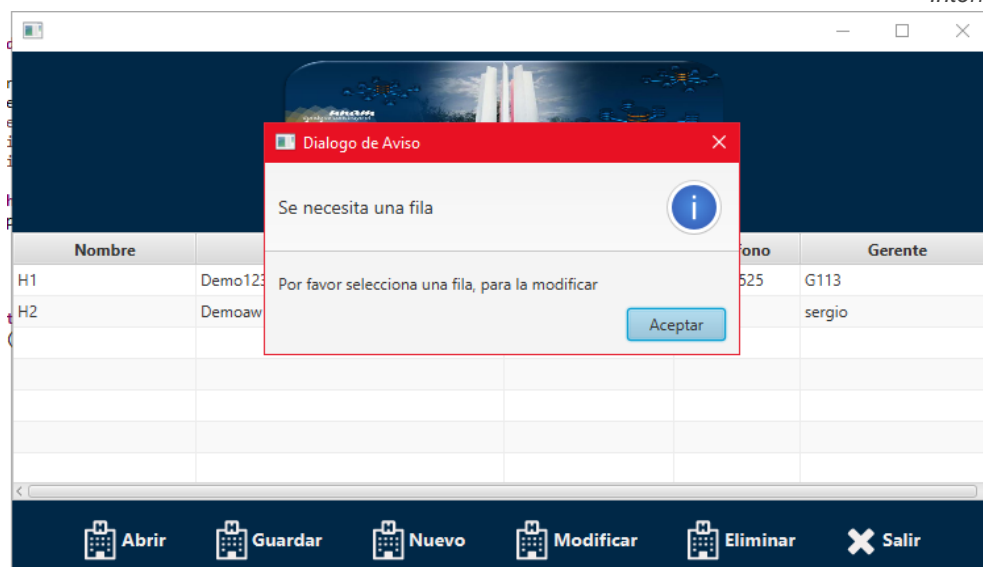
	id_gre	nombre	apellido_paterno	apellido_materno	rfc	correo	telefono
▶	1	Martin	Rodriguez	Perez	KADYRCSKFYWWR	correo@gmail.com	2354566333
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

	id_hbt	numero	costo	refrigerador	id_tps	id_htl
▶	1	HabNueva1	60.00	1	1	1
	2	HabNueva2	70.00	1	2	1
*	NULL	NULL	NULL	NULL	NULL	NULL

## Modificar

Para modificar un Hotel necesitamos seleccionar un hotel en la tabla y presionar el apartado de "Modificar". ¡OJO! De lo contrario marcará error y no te permitirá modificar el Hotel.

*Interfaz de Error*



Después de seleccionar un hotel correctamente podrás presionar el botón "Modificar" unos segundos después se abrirá la interfaz de hotel la cual contendrá los datos agregados los cuales podrás modificar con el simple hecho de seleccionar los campos ahora llenos ¡OJO! tendrás que seguir las restricciones anteriormente mencionadas.

Para aplicar el cambio y verlo reflejado bastará con presionar en el apartado aceptar el cual llamará el método modificar que implementará el siguiente query: **"update hoteles set nombre=?,direccion=? "+ ",correo=?,telefono=? where"+ " id\_htl=?";** .

¡ESTO APLICA EN TODOS LOS CASOS!

Interfaz de Modificar Hotel

## Hotel

Nombre:

Dirección:

Correo:

Teléfono:

Gerente:

Habitación:

Interfaz después de Modificar Hotel

Nombre ▲	Dirección	Correo	Telefono	Gerente
H11	Demo1232	h21@demo.com	3424232995	G113
H2	Demoaw	h2@demo.com	3123131	sergio

**Abrir**
 **Guardar**
 **Nuevo**
 **Modificar**
 **Eliminar**
 **Salir**

Para modificar un gerente debemos seleccionar el hotel con el gerente asociado y presionar la sección modificar. Se abrirá la ventana de modificar hotel, buscaremos la sección gerentes y daremos click se abrirá la ventana modificar gerente ya ahí podremos realizar todos los cambios necesarios siempre respetando las restricciones.

Una vez realizados los cambios bastará con presionar el apartado “Aceptar” el cual llamará el método modificar complementado con el query: **"update gerentes set nombre=?,apellido\_paterno=? " + ",apellido\_materno=?,rfc=?,correo=?,telefono=? where" + " id\_gre=?";**



## Nuevo Gerente

Nombre:   
 Apellido Paterno:   
 Apellido Materno:   
 RFC:   
 Correo:   
 Teléfono:

Para aplicar el cambio y verlo reflejado bastará con presionar en el apartado aceptar el cual llamará el método modificar que implementará.

Nombre ▲	Dirección	Correo	Telefono	Gerente
H11	Demo1232	h21@demo.com	3424232995	G122
H2	Demoaw	h2@demo.com	3123131	sergio

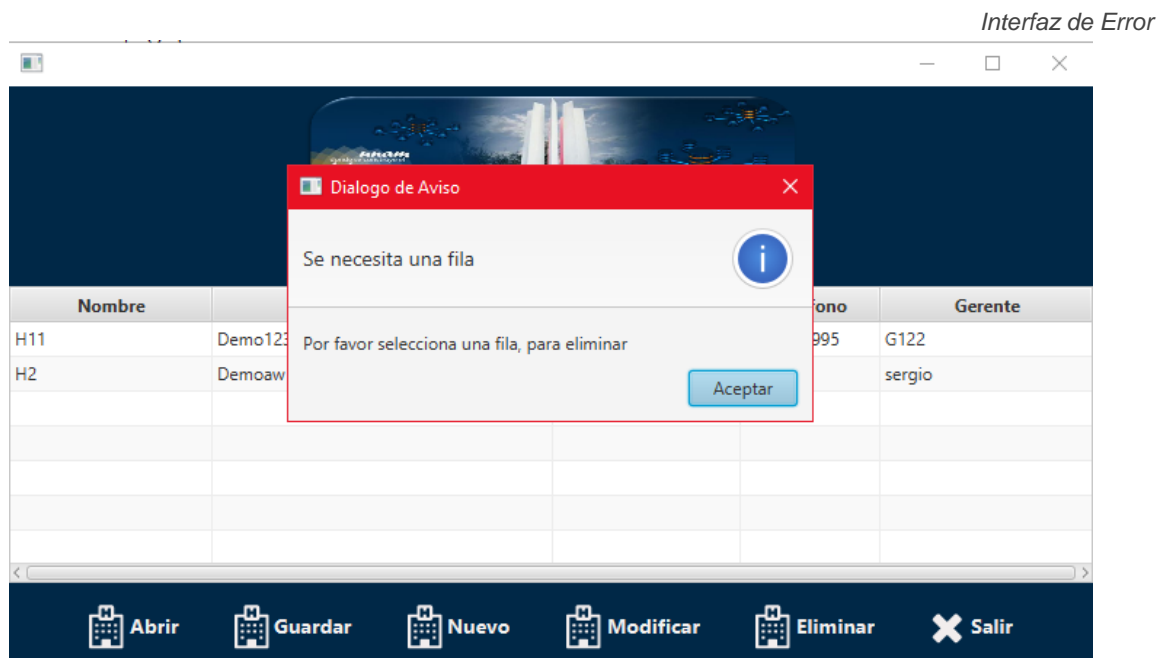
Para modificar una habitación debemos seleccionar el hotel y presionamos la sección “*modificar*”. Se abrirá la ventana de modificar hotel, buscaremos la sección habitaciones y daremos click se abrirá la ventana modificar habitación ya ahí podremos realizar todos los cambios necesarios siempre respetando las restricciones.

Una vez realizados los cambios bastará con presionar el apartado “*Aceptar*” el cual llamará el método modificar complementado con el query: **"update habitaciones set numero=?,costo=? " + ",refrigerador=?,id\_tps=? where " + " id\_htl=? and id\_hbt=?";**

¡OJO! si no quieres modificar gerente y/o habitación este paso puede ser omitido.

## Eliminar

Para eliminar un Hotel necesitamos seleccionar un hotel en la tabla y presionar el apartado de “Eliminar”. De lo contrario te marcará error y no te permitirá eliminar el hotel.



Después de seleccionar un hotel correctamente podrás presionar el botón “Eliminar” unos segundos después se eliminará el hotel. Se llama el método de eliminarHotel en cierto orden específico, primero habitaciones query: “**delete from habitaciones where id\_htl=?**”; luego hoteles query: “**delete from hoteles where id\_htl=?**” y al último gerente, query: “**delete from gerentes where id\_gre=?**”. Eliminando el hotel que se creó en el apartado de “Agregar”:



Se elimina y queda de la siguiente manera:



Y si corroboramos en la base de datos, ya se habrá eliminado:

*Eliminación en la base de datos*

	id_htl	nombre	direccion	correo	telefono	id_gre
*	NULL	NULL	NULL	NULL	NULL	NULL

	id_gre	nombre	apellido_paterno	apellido_materno	rfc	correo	telefono
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

	id_hbt	numero	costo	refrigerador	id_tps	id_htl
*	NULL	NULL	NULL	NULL	NULL	NULL

## Conclusiones

-Para la realización de este trabajo pudimos aplicar todos los conocimientos adquiridos durante este curso desde la simple creación de una base de datos con base a una problemática o necesidad hasta la implementación de un modelo físico (Base de datos) y la creación de la misma aplicación y la interconexión de ambas para así lograr un sistema que nos permitiera la llamada CRUD (crear, leer ,actualizar y eliminar) con solo presionar unos botones y haciéndola útil aun cuando no se sabe la estructura de funcionalidad lo cual fue logrado gracias a algunas técnicas de programación y métodos para bases de datos.

-Pudimos resolver satisfactoriamente las problemáticas que nos surgieron para la utilización y creación de algunos de los métodos como el de insertar, eliminar, actualizar y leer, así como también nos dimos cuenta de algunas de las fallas que teníamos en el código ya sean de sintaxis o de lógica. Tuvimos que recurrir a la investigación, revisar apuntes de clases anteriores y consultar algunos libros que fueron de gran ayuda para hacer este proyecto.

-La creación de los query's no fue tan difícil, ya que estos no demandan tanta complejidad de traer o eliminar o actualizar y agregar datos de otras tablas y gracias a esto fue fácil pasarlo en el código y en la base de datos.

-Al hacer este proyecto podemos observar que la materia base de datos es muy importante, hablando de las bases de datos que hoy en día podemos ver cómo se utilizan fuera del simple hecho de la creación de las mismas en la computación, ya que en cualquier lugar como en este caso un Hotel en donde se almacena datos e información es importante la creación de una base de datos para la recuperación y almacenamiento de aquellos datos necesarios para que funcione el negocio.

## **Bibliografía**

-ERIC GODOC. (2014). SQL - FUNDAMENTOS DEL LENGUAJE. Barcelona: ENI.

-ABRAHAM SILBERSCHATZ. (2006). BASE DE DATOS COMANDOS. En FUNDAMENTOS DE BASES DE DATOS (5ª ED.) (497). ESPAÑA: S.A. MCGRAW-HILL / INTERAMERICANA DE ESPAÑA.

-Bryan Syverson and Joel Murach. (2020). Sql. En Murach' s SQL Server 2019 for Developers (674). Estados Unidos: Murach.

Deitel P.J. 2008 25.11 objetos PreparedStatement JAVA como programar (7ma edición, pp.1115) Editorial Pearson

Mercedes Marqués. (2011). Bases de Datos. Brasil: UNIVERSISAT JAUME.

Abraham Silberschatz. (2002). Fundamentos de Bases de datos. México: Mc Graw Hill.