

# **PROYECTO**

***“Sistema de  
Reservaciones de  
Hotel”***

**17 DE NOV DE 2023**

***MANEJO E IMPLEMENTACION DE  
ARCHIVOS***

***MYNOR BEZALEEL RAMOS  
GONZÁLEZ***

***ING. INDIRA VALDES***

# INDICE

<b>INTRODUCCIÓN .....</b>	<b>1</b>
<b>OBJETIVOS.....</b>	<b>2</b>
<b>MARCO TEÓRICO .....</b>	<b>3</b>
Sistema Operativo:.....	3
Herramientas de Desarrollo: .....	3
Lenguaje de Programación: .....	3
Base de Datos:.....	3
Contenedor: .....	3
Plataforma de Desarrollo Colaborativo:.....	4
<b>DISEÑO DEL SISTEMA .....</b>	<b>5</b>
<b>BASE DE DATOS .....</b>	<b>7</b>
DDL .....	7
DML.....	9
<b>DESARROLLO CODIGO C .....</b>	<b>11</b>
Archivos Utilizados:.....	11
Descripción:.....	11
1.    conexión.h.....	11
2.    main.c .....	11
3.    menuHabitaciones.c.....	13
4.    menuClientes.c .....	17
5.    menuReservaciones.c .....	22
<b>CONCLUSIONES .....</b>	<b>31</b>

# INTRODUCCIÓN

En el dinámico mundo de la industria hotelera, la eficiencia en la gestión de reservaciones es esencial para brindar una experiencia excepcional a los huéspedes. Este proyecto se centra en el desarrollo de un sistema de reservaciones diseñado específicamente para optimizar las operaciones del recepcionista, la figura clave en la interacción directa con los clientes. El objetivo principal es proporcionar una plataforma intuitiva y efectiva que simplifique la gestión de habitaciones, clientes y reservaciones, contribuyendo así a una administración más eficiente y a una experiencia del cliente mejorada.

# OBJETIVOS

- **Gestión Integrada de Habitaciones:**

- Desarrollar un sistema que permita al recepcionista gestionar de manera eficiente la disponibilidad, estado y asignación de habitaciones, proporcionando una interfaz clara para visualizar y modificar la información de las habitaciones.

- **Facilidades en el Manejo de Clientes:**

- Implementar funcionalidades que simplifiquen la gestión de clientes para el recepcionista, incluyendo la capacidad de ver, agregar, actualizar y eliminar información de clientes, con validaciones para evitar duplicaciones y garantizar datos precisos.

- **Optimización de Procesos de Reservación:**

- Mejorar la eficiencia en el manejo de reservaciones, ofreciendo funciones detalladas para consultar, agregar, actualizar y cancelar reservas. Además, proporcionar información histórica y alertas automatizadas para facilitar la administración de las reservaciones por parte del recepcionista.

# MARCO TEÓRICO

## Sistema Operativo:

### 1. macOS : Sonoma 14.0

Es la versión del sistema operativo utilizada en los dispositivos Mac para el desarrollo de este sistema. Ofrece una interfaz gráfica amigable, un entorno Unix robusto, y compatibilidad con herramientas esenciales de desarrollo, como CLion y Git. La versatilidad de macOS facilita el desarrollo en lenguajes como C y la integración con contenedores Docker para una implementación eficiente.

## Herramientas de Desarrollo:

### 2. Entorno de Desarrollo Integrado (IDE):

**CLion:** IDE utilizado para el desarrollo en C. Proporciona un conjunto de herramientas para escribir, depurar y compilar código de manera eficiente.

### 3. Gestión de Versiones:

**Git:** Sistema de control de versiones que permite el seguimiento de cambios, colaboración y gestión del historial de versiones del código.

## Lenguaje de Programación:

### 4. Lenguajes de Programación:

**C:** Seleccionado como el lenguaje principal para el desarrollo del sistema. C es conocido por su eficiencia y capacidad de bajo nivel.

## Base de Datos:

### 5. Gestor de Bases de Datos:

**MySQL (en Docker):** Se utiliza un contenedor Docker para ejecutar una instancia de MySQL, facilitando la gestión y la implementación de la base de datos.

### 6. Gestor de Bases de Datos (IDE):

**DataGrip:** Herramienta de JetBrains utilizada como entorno de desarrollo para la administración de bases de datos. Ofrece funciones avanzadas de gestión y visualización de datos.

## Contenedor:

### 7. Contenedor:

**Docker:** Utilizado para encapsular la aplicación y su entorno, garantizando la consistencia en diferentes entornos y simplificando la implementación.

## Plataforma de Desarrollo Colaborativo:

### 8. Plataforma de desarrollo colaborativo:

**GitHub:** Utilizado como plataforma para alojar y colaborar en el desarrollo del código fuente. Facilita la colaboración entre miembros del equipo y el control de versiones.

### 9. Codesnap.dev: Herramienta en línea utilizada para compartir y revisar fragmentos de código de manera colaborativa.

# DISEÑO DEL SISTEMA

Este programa es un sistema de gestión de reservaciones de hotel que proporciona al recepcionista un menú interactivo para administrar diferentes aspectos relaciones con las habitaciones, clientes y las reservaciones. A continuación se describe el menu principal y los sub menú disponibles en el sistema:

## Menú Principal Reservaciones:

1. Manejo de Habitaciones
2. Manejo de Clientes
3. Manejo de Reservaciones
4. Salir

### - Sub-Menú Manejo de Habitaciones:

#### 1. Ver Habitaciones Disponibles

Muestra una lista de habitaciones disponibles a la fecha actual, incluyendo detalles como Número de habitación, tipo, precio y estado en la que se encuentra la habitación.

#### 2. Ver Habitaciones Ocupadas

Muestra una lista de habitaciones ocupadas a la fecha actual, con detalles similares a la opción de habitaciones disponibles.

#### 3. Ver Habitaciones En Mantenimiento (Limpieza o reparación)

Muestra una lista de habitaciones en mantenimiento en la fecha actual.

#### 4. Modificar Estado de Habitación

Permite cambiar el estado de una habitación, por ejemplo, de disponible a ocupada o viceversa.

#### 5. Volver al Menú Principal

Permite regresar al menú principal para seleccionar otro sub-menú.

### - Sub-Menú Manejo de Clientes:

#### 1. Ver Clientes

Muestra una lista de clientes con información relevante como dpi, nombre, apellido, email y teléfono.

#### 2. Agregar Cliente

Permite agregar un nuevo cliente proporcionando los detalles necesarios.

#### 3. Eliminar Cliente

Permite eliminar un cliente existente, generalmente utilizando el dpi del cliente.

#### 4. Actualizar Cliente

Permite actualizar la información de un cliente existente.

#### 5. Volver al Menú Principal

Permite regresar al menú principal para seleccionar otro sub-menú.

- **Sub-Menú Manejo de Reservaciones:**

**1. Historial de reservaciones por habitación**

Muestra el historial de reservaciones para una habitación específica, ingresando el número de habitación.

**2. Reservaciones para fecha específica**

Muestra las reservaciones para una fecha específica, ingresando la fecha de ingreso.

**3. Reservaciones confirmadas para fecha específica**

Muestra las reservaciones para una fecha específica, solamente si están confirmadas, ingresando la fecha de ingreso.

**4. Reservaciones para cliente específico (DPI)**

Muestra las reservaciones asociadas a un cliente específico utilizando su identificación (DPI).

**5. Agregar una reservación**

Permite al recepcionista agregar una reservación proporcionando la información necesaria, al ingresar muestra los datos de la reservación.

**6. Actualizar datos de una reservación**

Permite actualizar información de una reservación existente y con un estado “Confirmada”.

**7. Cancelar una reservación**

Permite cancelar una reservación, solamente si es confirmada, y pone en cancelada.

**8. Generar factura por número de reservación**

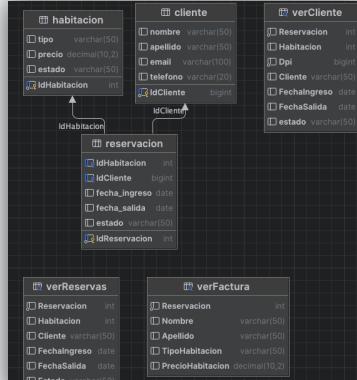
Genera una factura para una habitación específica, mostrando detalles de la reservación.

**9. Volver al Menú Principal**

Permite regresar al menú principal para seleccionar otro sub-menú.

# BASE DE DATOS

- Esquema:



DDL

- Creación Base de Datos “DBReservacion”.

```
CREATE DATABASE ReservacionDB;
```

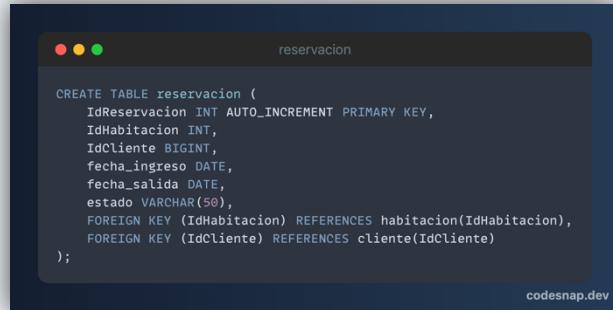
- Tabla “habitación”.

```
CREATE TABLE habitacion (
    IdHabitacion INT NOT NULL PRIMARY KEY,
    tipo VARCHAR(50),
    precio DECIMAL(10, 2),
    estado VARCHAR(50)
);
```

- Tabla “cliente”.

```
CREATE TABLE cliente (
    IdCliente BIGINT NOT NULL PRIMARY KEY,
    nombre VARCHAR(50),
    apellido VARCHAR(50),
    email VARCHAR(100),
    telefono VARCHAR(20)
);
```

- Tabla “reservación”.



```

CREATE TABLE reservacion (
    IdReservacion INT AUTO_INCREMENT PRIMARY KEY,
    IdHabitacion INT,
    IdCliente BIGINT,
    fecha_ingreso DATE,
    fecha_salida DATE,
    estado VARCHAR(50),
    FOREIGN KEY (IdHabitacion) REFERENCES habitacion(IdHabitacion),
    FOREIGN KEY (IdCliente) REFERENCES cliente(IdCliente)
);

```

codesnap.dev

- Creación de vistas:



**verReservas**

```

CREATE VIEW verReservas AS
SELECT
    r.IdReservacion AS Reservacion,
    r.IdHabitacion AS Habitacion,
    c.nombre AS Cliente,
    r.fecha_ingreso AS FechaIngreso,
    r.fecha_salida AS FechaSalida,
    r.estado AS Estado
FROM
    reservacion r
JOIN
    cliente c ON r.IdCliente = c.IdCliente;

```

codesnap.dev

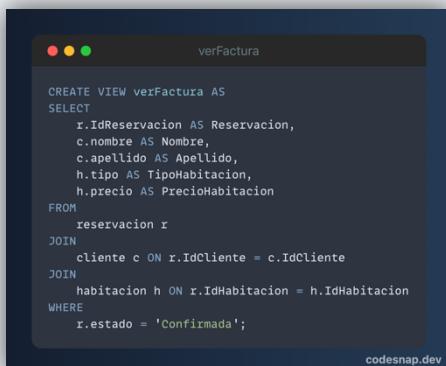
**verCliente**

```

CREATE VIEW verCliente AS
SELECT
    r.IdReservacion AS Reservacion,
    r.IdHabitacion AS Habitacion,
    c.IdCliente AS Dpi,
    c.nombre AS Cliente,
    r.fecha_ingreso AS FechaIngreso,
    r.fecha_salida AS FechaSalida,
    r.estado
FROM
    reservacion r
JOIN
    cliente c ON r.IdCliente = c.IdCliente;

```

codesnap.dev



```

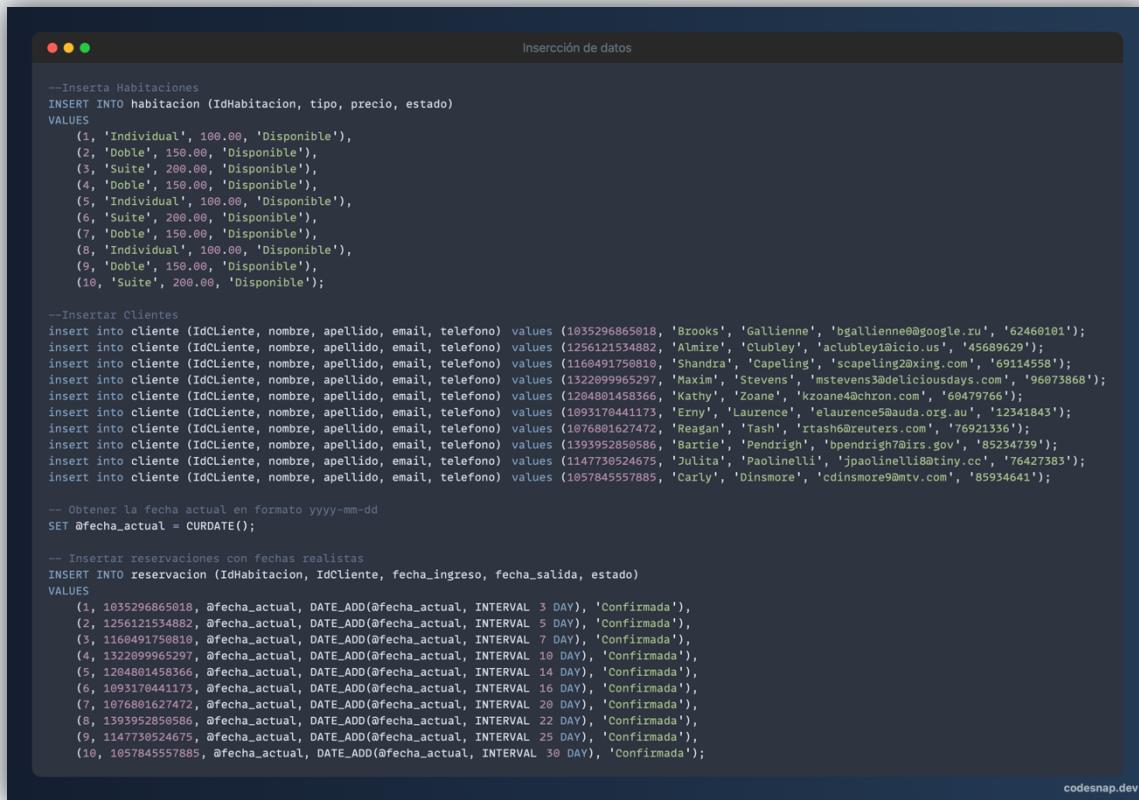
CREATE VIEW verFactura AS
SELECT
    r.IdReservacion AS Reservacion,
    c.nombre AS Nombre,
    c.apellido AS Apellido,
    h.tipo AS TipoHabitacion,
    h.precio AS PrecioHabitacion
FROM
    reservacion r
JOIN
    cliente c ON r.IdCliente = c.IdCliente
JOIN
    habitacion h ON r.IdHabitacion = h.IdHabitacion
WHERE
    r.estado = 'Confirmada';

```

codesnap.dev

# DML

- Inserción de datos:



```
--Inserta Habitaciones
INSERT INTO habitacion (IdHabitacion, tipo, precio, estado)
VALUES
(1, 'Individual', 100.00, 'Disponible'),
(2, 'Doble', 150.00, 'Disponible'),
(3, 'Suite', 200.00, 'Disponible'),
(4, 'Doble', 150.00, 'Disponible'),
(5, 'Individual', 100.00, 'Disponible'),
(6, 'Suite', 200.00, 'Disponible'),
(7, 'Doble', 150.00, 'Disponible'),
(8, 'Individual', 100.00, 'Disponible'),
(9, 'Doble', 150.00, 'Disponible'),
(10, 'Suite', 200.00, 'Disponible');

--Insertar Clientes
insert into cliente (IdCliente, nombre, apellido, email, telefono) values (1035296865018, 'Brooks', 'Gallienne', 'bgallienne0@google.ru', '62460101');
insert into cliente (IdCliente, nombre, apellido, email, telefono) values (1256121534882, 'Almine', 'Clubley', 'aclubley1@icio.us', '45689629');
insert into cliente (IdCliente, nombre, apellido, email, telefono) values (1160491750810, 'Shandra', 'Capeling', 'scapealing2@xing.com', '69114558');
insert into cliente (IdCliente, nombre, apellido, email, telefono) values (1322099965297, 'Maxim', 'Stevens', 'mstevens3@deliciousdays.com', '96073868');
insert into cliente (IdCliente, nombre, apellido, email, telefono) values (1204801458366, 'Kathy', 'Zoane', 'kzane4@chion.com', '68479766');
insert into cliente (IdCliente, nombre, apellido, email, telefono) values (1093170441173, 'Erny', 'Laurence', 'elaurence5@auda.org.au', '12341843');
insert into cliente (IdCliente, nombre, apellido, email, telefono) values (1076801627472, 'Reagan', 'Tash', 'rtash6@reuters.com', '76921356');
insert into cliente (IdCliente, nombre, apellido, email, telefono) values (1393952850586, 'Bartie', 'Pendright', 'bpendright7@irs.gov', '85234739');
insert into cliente (IdCliente, nombre, apellido, email, telefono) values (1147730524675, 'Julita', 'Paolinelli', 'jpaoline@ib8tiny.cc', '76427383');
insert into cliente (IdCliente, nombre, apellido, email, telefono) values (1057845557885, 'Carly', 'Dinsmore', 'cdinsmore9@mtv.com', '85934641');

-- Obtener la fecha actual en formato yyyy-mm-dd
SET @fecha_actual = CURDATE();

-- Insertar reservaciones con fechas realistas
INSERT INTO reservacion (IdHabitacion, IdCliente, fecha_ingreso, fecha_salida, estado)
VALUES
(1, 1035296865018, @fecha_actual, DATE_ADD(@fecha_actual, INTERVAL 3 DAY), 'Confirmada'),
(2, 1256121534882, @fecha_actual, DATE_ADD(@fecha_actual, INTERVAL 5 DAY), 'Confirmada'),
(3, 1160491750810, @fecha_actual, DATE_ADD(@fecha_actual, INTERVAL 7 DAY), 'Confirmada'),
(4, 1322099965297, @fecha_actual, DATE_ADD(@fecha_actual, INTERVAL 10 DAY), 'Confirmada'),
(5, 1204801458366, @fecha_actual, DATE_ADD(@fecha_actual, INTERVAL 14 DAY), 'Confirmada'),
(6, 1093170441173, @fecha_actual, DATE_ADD(@fecha_actual, INTERVAL 16 DAY), 'Confirmada'),
(7, 1076801627472, @fecha_actual, DATE_ADD(@fecha_actual, INTERVAL 20 DAY), 'Confirmada'),
(8, 1393952850586, @fecha_actual, DATE_ADD(@fecha_actual, INTERVAL 22 DAY), 'Confirmada'),
(9, 1147730524675, @fecha_actual, DATE_ADD(@fecha_actual, INTERVAL 25 DAY), 'Confirmada'),
(10, 1057845557885, @fecha_actual, DATE_ADD(@fecha_actual, INTERVAL 30 DAY), 'Confirmada');
```

codesnap.dev

- SQL Consultas para INSERT, UPDATE, DELETE, INSERT de los datos de las tablas utilizadas en el lenguaje c:

SQL Consultas para: Update, Insert, Delete, Update.

```

/*Menú ManejoHabitaciones*/
/*función verHabitacionesDisponibles();*/
    SELECT IdHabitacion, tipo, precio, estado FROM habitacion WHERE estado = 'Disponible';
/*función verHabitacionesOcupadas();*/
    SELECT IdHabitacion, tipo, precio, estado FROM habitacion WHERE estado = 'Ocupado';
/*función verHabitacionesEnMantenimiento();*/
    SELECT IdHabitacion, tipo, precio, estado FROM habitacion WHERE estado = 'En mantenimiento';
/*ModificarEstadoHabitacion();*/
/*Verifica habitación seleccionada:*/
    SELECT * FROM habitacion WHERE IdHabitacion = '';
/*Realiza el cambios*/
    UPDATE habitacion SET estado = '' WHERE IdHabitacion = '';

/*Menú ManejoClientes*/
/*función verClientes();*/
    SELECT IdCliente, nombre, apellido, email, telefono FROM cliente;
/*función agregaCliente();*/
/*verifica el DPI(IdCliente)*/
    SELECT * FROM cliente WHERE IdCliente = '';
/*inserta*/
    INSERT INTO cliente (IdCliente, nombre, apellido, email, telefono) VALUES ('','','','','','');
/*función eliminarCliente();*/
/*verifica el DPI(IdCliente)*/
    SELECT * FROM cliente WHERE IdCliente = '';
/*elimina*/
    DELETE FROM cliente WHERE IdCliente = '';
/*función actualizarCliente();*/
/*verifica el DPI(IdCliente)*/
    SELECT * FROM cliente WHERE IdCliente = '';
/*actualiza*/
    UPDATE cliente SET nombre = '', apellido '', email= '', telefono = '' WHERE IdCliente = '';

/*Menú ManejoReservaciones*/
/*función consultarReservasHabitacion();*/
/*verifica la habitación(IdHabitacion)*/
    SELECT * FROM habitacion WHERE IdHabitacion = '';
/*selecciona*/
    SELECT Reservacion, Habitacion, cliente, FechaIngreso, FechaSalida, Estado FROM verReservas WHERE Habitacion = '';
/*función verReservasFechaEspecificas();*/
    SELECT Reservacion, Habitacion, cliente, FechaIngreso, FechaSalida, Estado FROM verReservas WHERE DATE(FechaIngreso) = '';
/*función verReservasConfirmadasFecha();*/
    SELECT Reservacion, Habitacion, cliente, FechaIngreso, FechaSalida, Estado FROM verReservas WHERE DATE(FechaIngreso) = '' AND Estado = 'Confirmada';
/*función reservacionesClientesPorId();*/
/*verifica el DPI(IdCliente)*/
    SELECT * FROM verCliente WHERE Dpi = '';
/*función agregarReservacion();*/
/*verifica la habitación(IdHabitacion)*/
    SELECT * FROM habitacion WHERE IdHabitacion = '';
/*verifica el dpi(IdCliente)*/
    SELECT * FROM cliente WHERE IdCliente = '';
/*Inserta*/
    INSERT INTO reservacion (IdHabitacion, IdCliente, fecha_ingreso, fecha_salida, estado) VALUES ('','','','','','');
/*función actualizarReservacion();*/
/*verifica la reservacion(IdReservacion)*/
    SELECT * FROM reservacion WHERE IdReservacion = '';
/*verifica si se asigna un valor repetido a fecha de ingreso para la misma habitacion con Estado: "Confirmada"*/
    SELECT * FROM reservacion WHERE IdHabitacion = (SELECT IdHabitacion FROM reservacion WHERE IdReservacion = '') AND fecha_ingreso = '' AND estado = 'Confirmada';
/*actualiza*/
    UPDATE reservacion SET fecha_ingreso = '', fecha_salida = '' WHERE IdReservacion = '';
/*función cancelarReservacion();*/
/*verifica la reservacion(IdReservacion)*/
    SELECT * FROM reservacion WHERE IdReservacion = '';
/*actualiza*/
    UPDATE reservacion SET estado = 'Cancelada' WHERE IdReservacion = '';
/*función generarFactura();*/
/*verifica la reservacion(IdReservacion)*/
    SELECT * FROM reservacion WHERE IdReservacion = '';
/*selección*/
    SELECT * FROM verFactura WHERE Reservacion = '';

```

codesnap.dev

# DESARROLLO CODIGO C

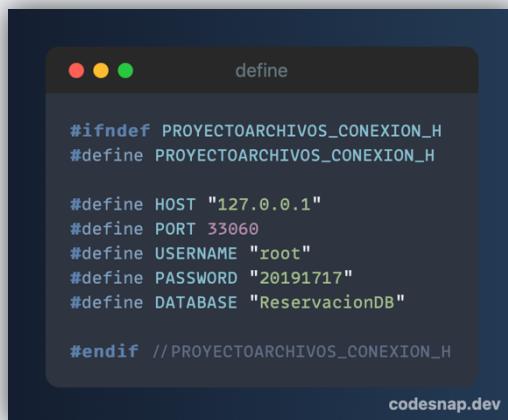
## Archivos Utilizados:

1. conexión.h
2. main.c
3. menuHabitaciones.c
4. menuClientes.c
5. menuReservaciones.c

## Descripción:

### 1. conexión.h

El código que proporcionaste define un conjunto de constantes que representan los parámetros de conexión a la base de datos MySQL.



```
#ifndef PROYECTOARCHIVOS_CONEXION_H
#define PROYECTOARCHIVOS_CONEXION_H

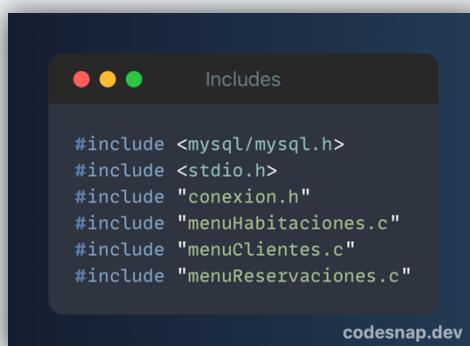
#define HOST "127.0.0.1"
#define PORT 33060
#define USERNAME "root"
#define PASSWORD "20191717"
#define DATABASE "ReservacionDB"

#endif // PROYECTOARCHIVOS_CONEXION_H
```

codesnap.dev

### 2. main.c

- #include es una herramienta para organizar y reutilizar código al permitir dividir el programa en archivos más pequeños manejables, incluidos el archivo conexión.h, y los sub-menú los cuales contienen las funciones necesarias para el funcionamiento adecuado del programa.

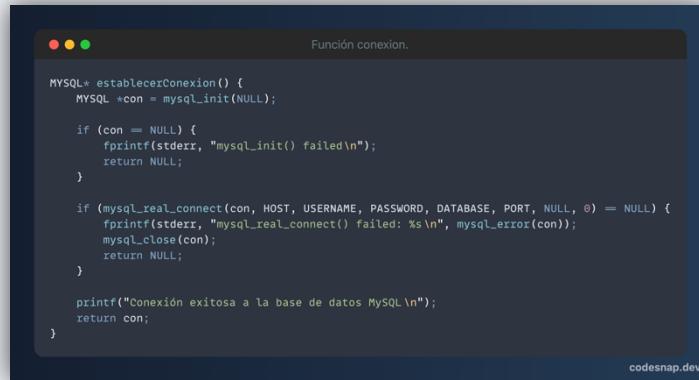


```
#include <mysql/mysql.h>
#include <stdio.h>
#include "conexion.h"
#include "menuHabitaciones.c"
#include "menuClientes.c"
#include "menuReservaciones.c"
```

codesnap.dev

- **Función establecerConexion();**

La función establecer conexión a la base de datos MYSQL utilizando la biblioteca MySQL C API. Si tiene éxito, devuelve un puntero a una estructura MYSQL, que representa la conexión.



```

Función conexion.

MYSQL* establecerConexion() {
    MYSQL *con = mysql_init(NULL);

    if (con == NULL) {
        fprintf(stderr, "mysql_init() failed\n");
        return NULL;
    }

    if (mysql_real_connect(con, HOST, USERNAME, PASSWORD, DATABASE, PORT, NULL, 0) == NULL) {
        fprintf(stderr, "mysql_real_connect() failed: %s\n", mysql_error(con));
        mysql_close(con);
        return NULL;
    }

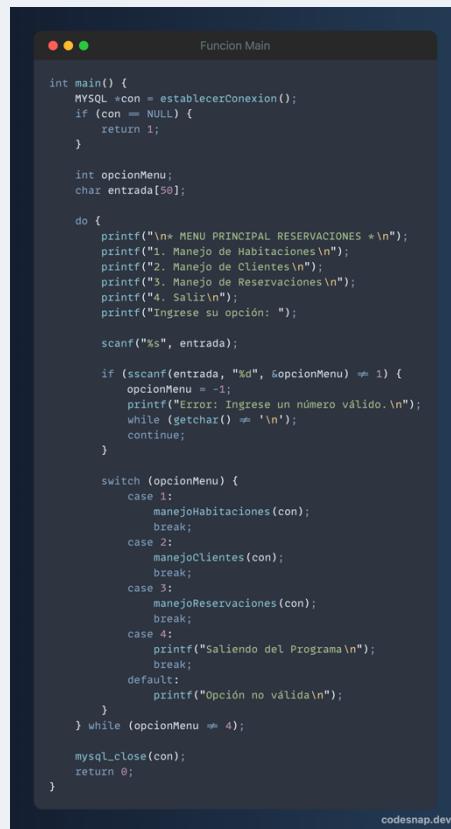
    printf("Conexión exitosa a la base de datos MySQL \n");
    return con;
}

```

codesnap.dev

- **Función Main();**

La función principal del programa que gestiona el menú principal y llama a las funciones correspondientes según la elección del usuario. ¿Qué hace?: Establece conexión, Menú principal con un bucle do-while (verifica si es un número y si esta dentro de las posibilidades, si es cadena de caracteres y si lo es da error), cierra conexión.



```

Función Main

int main() {
    MYSQL *con = establecerConexion();
    if (con == NULL) {
        return 1;
    }

    int opcionMenu;
    char entrada[50];

    do {
        printf("\n* MENU PRINCIPAL RESERVACIONES *\n");
        printf("1. Manejo de Habitaciones\n");
        printf("2. Manejo de Clientes\n");
        printf("3. Manejo de Reservaciones\n");
        printf("4. Salir\n");
        printf("Ingrese su opción: ");

        scanf("%s", entrada);

        if (sscanf(entrada, "%d", &opcionMenu) != 1) {
            opcionMenu = -1;
            printf("Error: Ingrese un número válido.\n");
            while (getchar() != '\n');
            continue;
        }

        switch (opcionMenu) {
            case 1:
                manejoHabitaciones(con);
                break;
            case 2:
                manejoClientes(con);
                break;
            case 3:
                manejoReservaciones(con);
                break;
            case 4:
                printf("Saliendo del Programa\n");
                break;
            default:
                printf("Opción no válida\n");
        }
    } while (opcionMenu != 4);

    mysql_close(con);
    return 0;
}

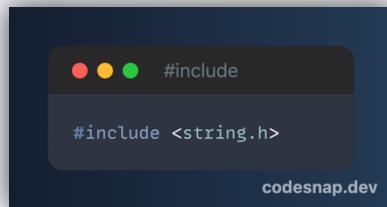
```

codesnap.dev

### 3. menuHabitaciones.c

- **#Include**

Incluyo las librerias <string.h> para su uso en nuestras funciones.



- **verHabitacionesDisponibles();**

Realiza una consulta SQL(SELECT) a la base de datos para obtener la información de habitaciones que tienen el estado ‘Disponible’.

Almacena el resultado de la consulta en una variable “result”.

Recorre las filas del resultado (‘result’) y muestra en la consola la informacion de cada habitacion disponible, como el número de habitación, tipo, precio y estado.

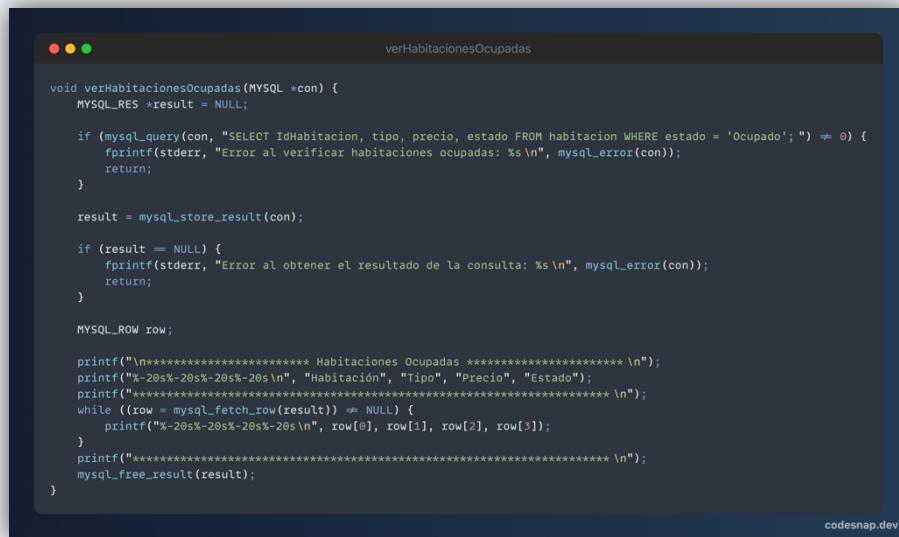
A screenshot of a terminal window titled 'verHabitacionesDisponibles'. It displays C code for querying a MySQL database to find available apartments. The code includes error handling, result storage, and row iteration logic. The window has a dark background with light-colored text. In the bottom right corner, it says 'codesnap.dev'.

- **verHabitacionesOcupadas();**

Realiza una consulta SQL(SELECT) a la base de datos para obtener la información de habitaciones que tienen el estado “Ocupado”.

Almacena el resultado de la consulta en una variable ‘result’.

Recorre las filas del resultado (‘result’) y muestra en la consola la información de cada habitación ocupada, como el número de habitación, tipo, precio y estado.



```

void verHabitacionesOcupadas(MYSQL *con) {
    MYSQL_RES *result = NULL;

    if (mysql_query(con, "SELECT IdHabitacion, tipo, precio, estado FROM habitacion WHERE estado = 'Ocupado';") != 0) {
        fprintf(stderr, "Error al verificar habitaciones ocupadas: %s\n", mysql_error(con));
        return;
    }

    result = mysql_store_result(con);

    if (result == NULL) {
        fprintf(stderr, "Error al obtener el resultado de la consulta: %s\n", mysql_error(con));
        return;
    }

    MYSQL_ROW row;

    printf("\n***** Habitaciones Ocupadas *****\n");
    printf("%-20s%-20s%-20s\n", "Habitación", "Tipo", "Precio");
    printf("*****\n");
    while ((row = mysql_fetch_row(result)) != NULL) {
        printf("%-20s%-20s%-20s\n", row[0], row[1], row[2], row[3]);
    }
    printf("*****\n");
    mysql_free_result(result);
}

```

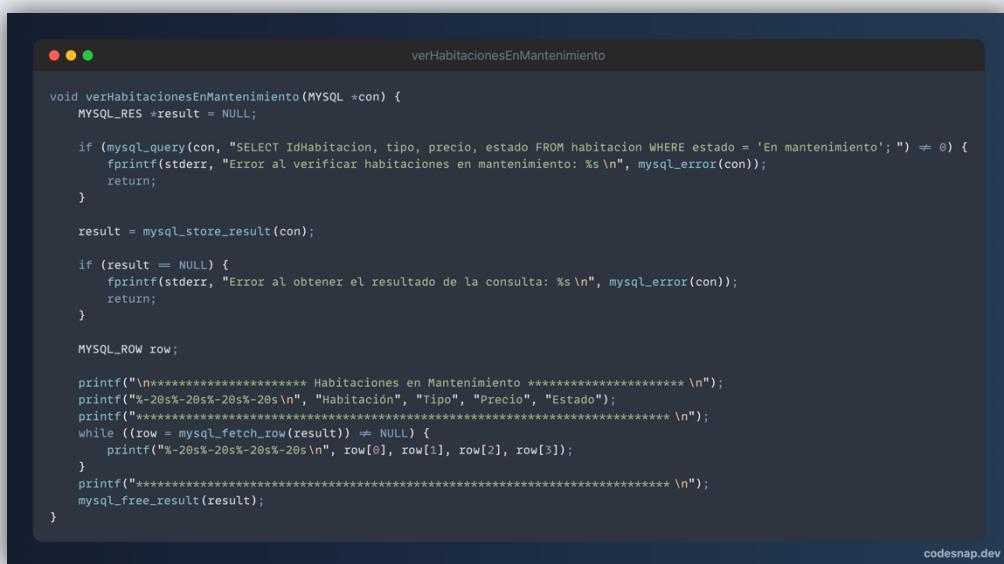
codesnap.dev

- **verHabitacionesEnMantenimiento();**

Realiza una consulta SQL(SELECT) a la base de datos para obtener la información de habitaciones que tienen el estado ‘En mantenimiento’.

Almacena el resultado de la consulta en una variable ‘result’.

Recorre las filas del resultado (‘result’) y muestra en la consola la información de cada habitación en mantenimiento, como el número de habitación, tipo, precio y estado.



```

void verHabitacionesEnMantenimiento(MYSQL *con) {
    MYSQL_RES *result = NULL;

    if (mysql_query(con, "SELECT IdHabitacion, tipo, precio, estado FROM habitacion WHERE estado = 'En mantenimiento';") != 0) {
        fprintf(stderr, "Error al verificar habitaciones en mantenimiento: %s\n", mysql_error(con));
        return;
    }

    result = mysql_store_result(con);

    if (result == NULL) {
        fprintf(stderr, "Error al obtener el resultado de la consulta: %s\n", mysql_error(con));
        return;
    }

    MYSQL_ROW row;

    printf("\n***** Habitaciones en Mantenimiento *****\n");
    printf("%-20s%-20s%-20s\n", "Habitación", "Tipo", "Precio");
    printf("*****\n");
    while ((row = mysql_fetch_row(result)) != NULL) {
        printf("%-20s%-20s%-20s\n", row[0], row[1], row[2], row[3]);
    }
    printf("*****\n");
    mysql_free_result(result);
}

```

codesnap.dev

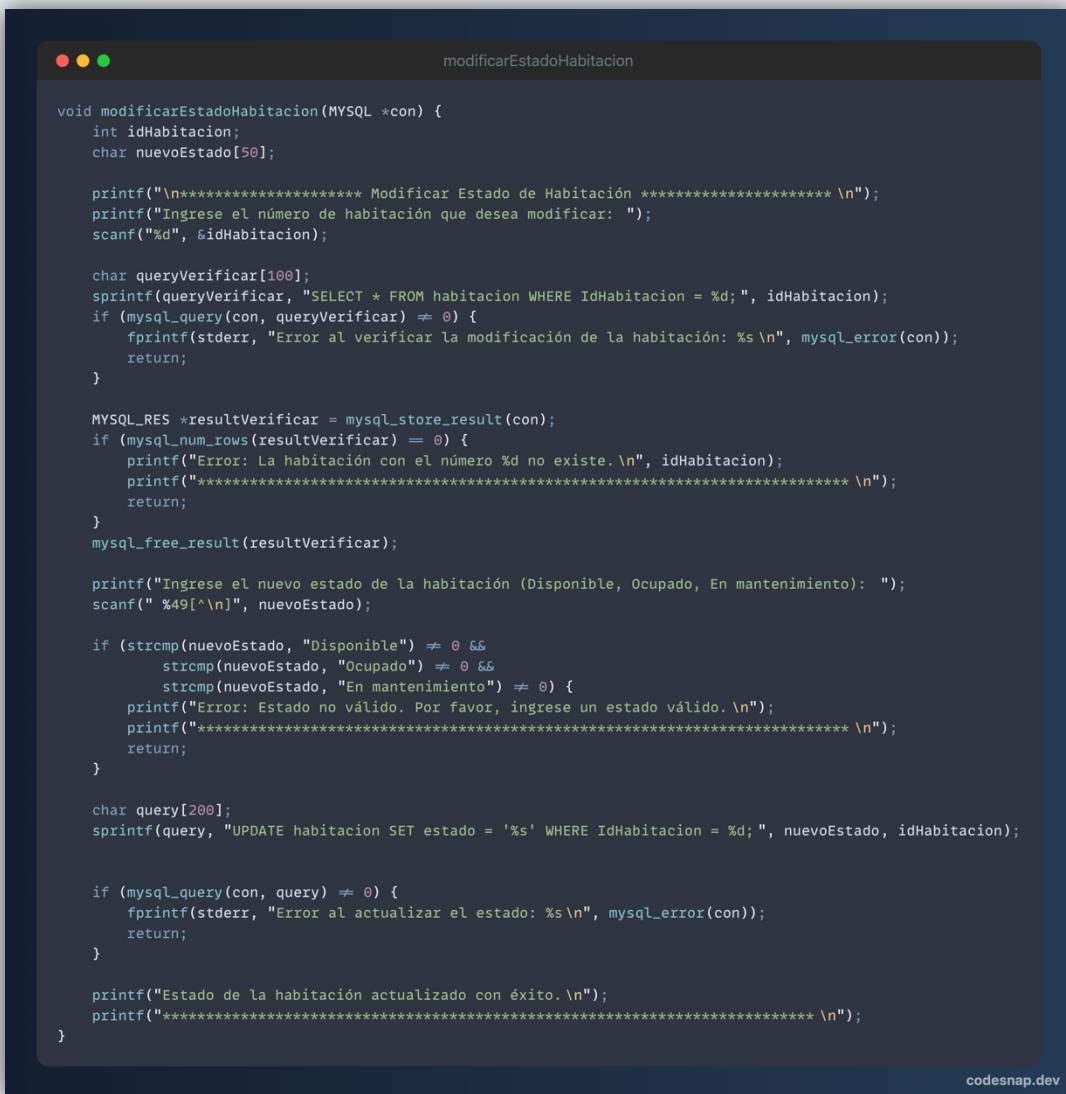
- **ModificarEstadoHabitacion();**

Solicita al usuario ingresar el número de habitación que desea modificar. Y tiene manejo de errores para ver si la habitación existe.

Verifica si la habitación con ese número existe en la base de datos mediante una consulta SQL(SELECT).

Si la habitación existe, solicita al usuario ingresar el nuevo estado de la habitación ('Disponible', 'Ocupado', 'En mantenimiento').

Actualiza el estado de la habitación en la base de datos mediante una consulta SQL (UPDATE).



The screenshot shows a terminal window with a dark background and light-colored text. The title bar reads "modificarEstadoHabitacion". The code inside the window is as follows:

```
void modificarEstadoHabitacion(MYSQL *con) {
    int idHabitacion;
    char nuevoEstado[50];

    printf("\n***** Modificar Estado de Habitación *****\n");
    printf("Ingrese el número de habitación que desea modificar: ");
    scanf("%d", &idHabitacion);

    char queryVerificar[100];
    sprintf(queryVerificar, "SELECT * FROM habitacion WHERE IdHabitacion = %d;", idHabitacion);
    if (mysql_query(con, queryVerificar) != 0) {
        fprintf(stderr, "Error al verificar la modificación de la habitación: %s\n", mysql_error(con));
        return;
    }

    MYSQL_RES *resultVerificar = mysql_store_result(con);
    if (mysql_num_rows(resultVerificar) == 0) {
        printf("Error: La habitación con el número %d no existe.\n", idHabitacion);
        printf("*****\n");
        return;
    }
    mysql_free_result(resultVerificar);

    printf("Ingrese el nuevo estado de la habitación (Disponible, Ocupado, En mantenimiento): ");
    scanf(" %49[^\\n]", nuevoEstado);

    if (strcmp(nuevoEstado, "Disponible") != 0 &&
        strcmp(nuevoEstado, "Ocupado") != 0 &&
        strcmp(nuevoEstado, "En mantenimiento") != 0) {
        printf("Error: Estado no válido. Por favor, ingrese un estado válido.\n");
        printf("*****\n");
        return;
    }

    char query[200];
    sprintf(query, "UPDATE habitacion SET estado = '%s' WHERE IdHabitacion = %d;", nuevoEstado, idHabitacion);

    if (mysql_query(con, query) != 0) {
        fprintf(stderr, "Error al actualizar el estado: %s\n", mysql_error(con));
        return;
    }

    printf("Estado de la habitación actualizado con éxito.\n");
    printf("*****\n");
}
```

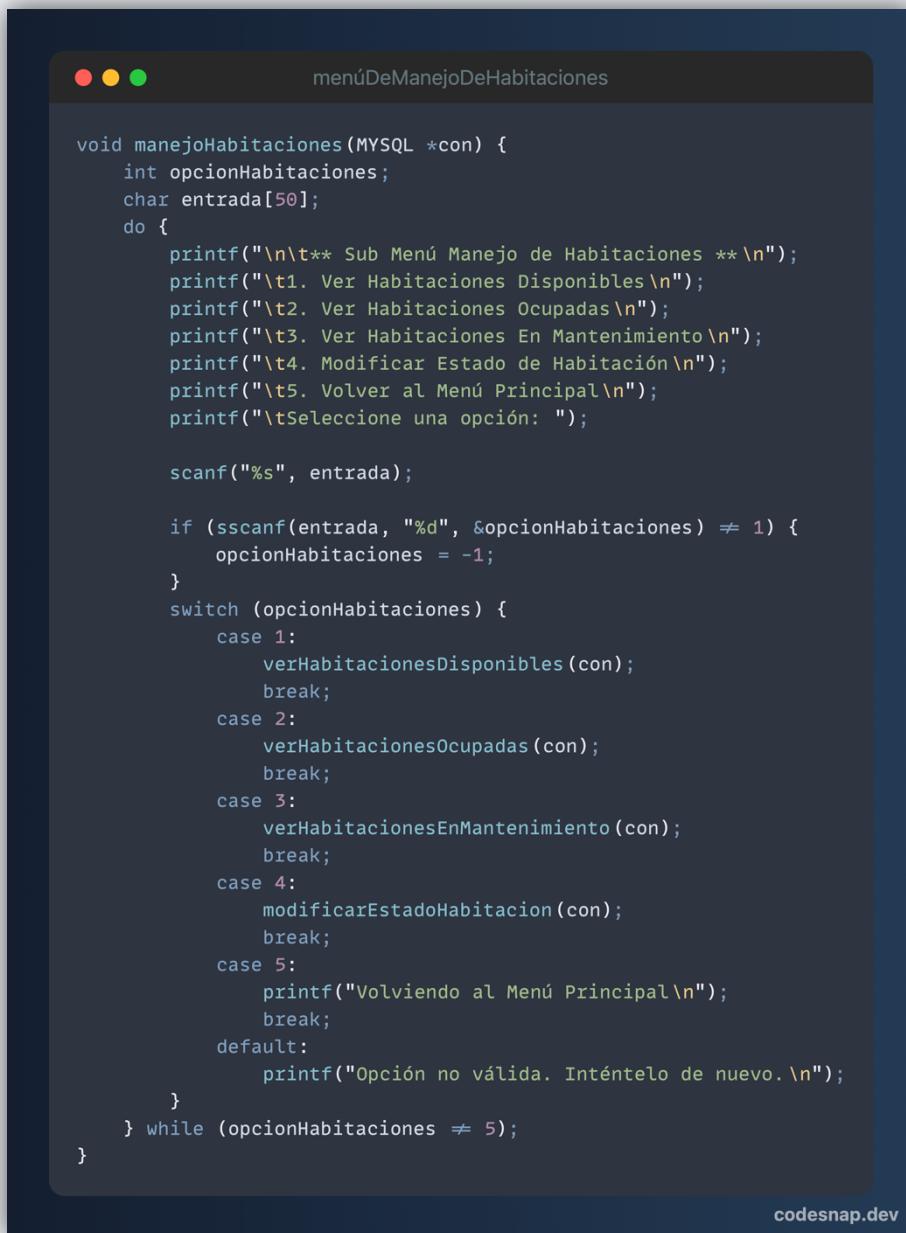
In the bottom right corner of the terminal window, the text "codesnap.dev" is visible.

- **manejoHabitaciones();**

Presenta un menú interactivo con las opciones mostradas en el código.

Solicita al recepcionista seleccionar una opción del menú y si no lo hace de manera correcta muestra mensaje de error y vuelve a solicitar una opción compara números distintos y si se ingresan cadenas de texto.

Llama a la función correspondiente según la opción seleccionada.



The screenshot shows a terminal window with a dark background and light-colored text. The title bar reads "menúDeManejoDeHabitaciones". The code is a C function named "manejoHabitaciones" that takes a MySQL connection pointer as a parameter. It uses a do-while loop to repeatedly present a menu to the user until they select option 5 (exit). The menu options are numbered 1 through 5, corresponding to different functions: Ver Habitaciones Disponibles, Ver Habitaciones Ocupadas, Ver Habitaciones En Mantenimiento, Modificar Estado de Habitación, and Volver al Menú Principal. The code includes printf statements for the menu options and scanf for reading user input. It also includes a check for invalid input and a break statement to exit the switch block when option 5 is selected.

```
void manejoHabitaciones(MYSQL *con) {
    int opcionHabitaciones;
    char entrada[50];
    do {
        printf("\n\t** Sub Menú Manejo de Habitaciones **\n");
        printf("\t1. Ver Habitaciones Disponibles\n");
        printf("\t2. Ver Habitaciones Ocupadas\n");
        printf("\t3. Ver Habitaciones En Mantenimiento\n");
        printf("\t4. Modificar Estado de Habitación\n");
        printf("\t5. Volver al Menú Principal\n");
        printf("\tSeleccione una opción: ");

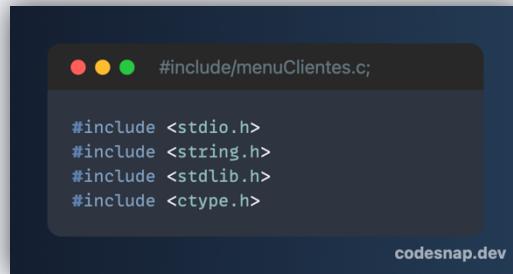
        scanf("%s", entrada);

        if (sscanf(entrada, "%d", &opcionHabitaciones) != 1) {
            opcionHabitaciones = -1;
        }
        switch (opcionHabitaciones) {
            case 1:
                verHabitacionesDisponibles(con);
                break;
            case 2:
                verHabitacionesOcupadas(con);
                break;
            case 3:
                verHabitacionesEnMantenimiento(con);
                break;
            case 4:
                modificarEstadoHabitacion(con);
                break;
            case 5:
                printf("Volviendo al Menú Principal\n");
                break;
            default:
                printf("Opción no válida. Inténtelo de nuevo.\n");
        }
    } while (opcionHabitaciones != 5);
}
```

codesnap.dev

## 4. menuClientes.c

- **#Include**



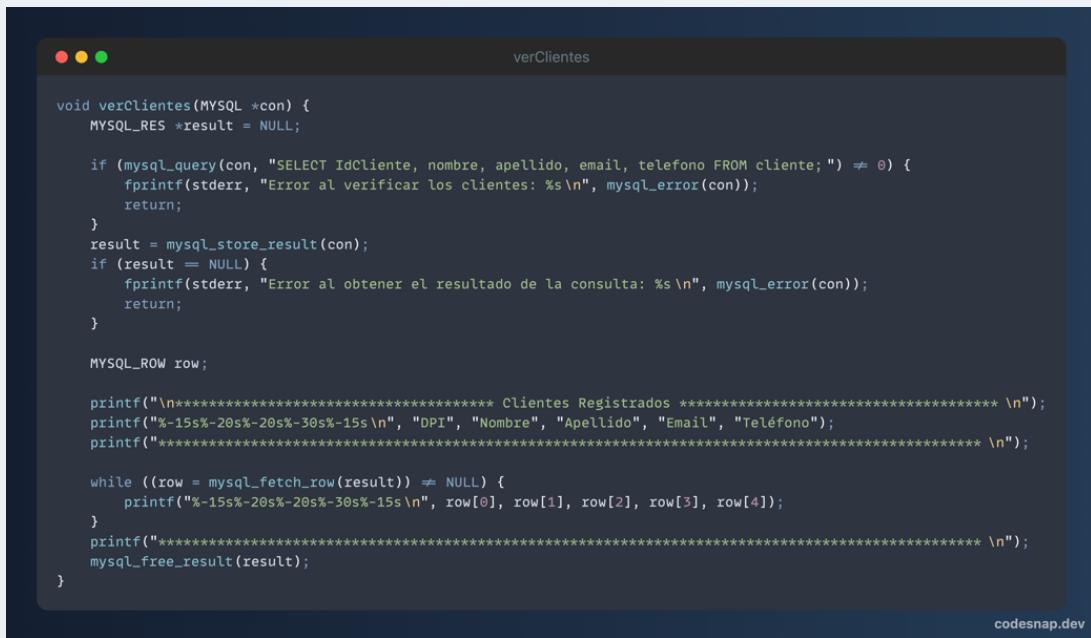
```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
```

codesnap.dev

- **verClientes();**

Realiza una consulta SQL(SELECT) a la base de datos para obtener la información de todos los clientes.

Imprime en la consola la información de los clientes, incluyendo el DPI, nombre, apellido, email, teléfono.



```
verClientes

void verClientes(MYSQL *con) {
    MYSQL_RES *result = NULL;

    if (mysql_query(con, "SELECT IdCliente, nombre, apellido, email, telefono FROM cliente;") != 0) {
        fprintf(stderr, "Error al verificar los clientes: %s\n", mysql_error(con));
        return;
    }
    result = mysql_store_result(con);
    if (result == NULL) {
        fprintf(stderr, "Error al obtener el resultado de la consulta: %s\n", mysql_error(con));
        return;
    }

    MYSQL_ROW row;

    printf("\n***** Clientes Registrados *****\n");
    printf("%-15s%-20s%-20s%-30s%-15s\n", "DPI", "Nombre", "Apellido", "Email", "Teléfono");
    printf("*****\n");

    while ((row = mysql_fetch_row(result)) != NULL) {
        printf("%-15s%-20s%-20s%-30s%-15s\n", row[0], row[1], row[2], row[3], row[4]);
    }
    printf("*****\n");
    mysql_free_result(result);
}
```

codesnap.dev

- **agregaCliente();**

Solicita al recepcionista ingresar información para un nuevo cliente, como DPI(IdCliente), nombre, apellido, email y teléfono.

Verifica si el DPI del cliente ya existe en la base de datos, debe tener 13 números y si tiene mas o menos de 13 o letras genera un error.

Verifica si el número de teléfono debe tener 8 números y si tiene mas o menos de 8 o letras genera un error.

Si el DPI no existe, realiza una consulta SQL(INSERT) para agregar al nuevo cliente a la base de datos.

```

agregarCliente()

void agregarCliente(MYSQL *con) {
    char nombre[50], apellido[50], email[100], telefono[20], idCliente[20];
    printf("\n***** Agregar Cliente *****\n");
    printf("Ingrese el DPI del nuevo cliente: ");
    scanf("%s", idCliente);

    int valido = 1;
    for (int i = 0; i < strlen(idCliente); i++) {
        if (!isdigit(idCliente[i])) {
            valido = 0;
            break;
        }
    }

    while (strlen(idCliente) != 13 || !valido) {
        printf("Error: El DPI debe tener exactamente 13 dígitos y contener solo números.\n");
        printf("Ingrese el DPI del nuevo cliente: ");
        scanf("%s", idCliente);

        valido = 1;
        for (int i = 0; i < strlen(idCliente); i++) {
            if (!isdigit(idCliente[i])) {
                valido = 0;
                break;
            }
        }
    }

    char queryVerificar[100];
    sprintf(queryVerificar, "SELECT * FROM cliente WHERE IdCliente = '%s';", idCliente);
    if (mysql_query(con, queryVerificar) == 0) {
        printf(stderr, "Error al verificar el DPI del cliente: %s\n", mysql_error(con));
        return;
    }

    MYSQL_RES *resultVerificar = mysql_store_result(con);
    if (mysql_num_rows(resultVerificar) > 0) {
        printf("Error: El DPI del cliente ya existe.\n");
        printf("*****\n");
        mysql_free_result(resultVerificar);
        return;
    }
    mysql_free_result(resultVerificar);

    printf("Ingrese el nombre del cliente: ");
    scanf("%s", nombre);

    if (strlen(nombre) > 49) {
        printf("Error: Longitud del nombre excede el límite permitido.\n");
        printf("*****\n");
        return;
    }

    printf("Ingrese el apellido del cliente: ");
    scanf("%s", apellido);

    if (strlen(apellido) > 49) {
        printf("Error: Longitud del apellido excede el límite permitido.\n");
        printf("*****\n");
        return;
    }

    printf("Ingrese el email del cliente: ");
    scanf("%s", email);

    printf("Ingrese el teléfono del cliente, Formato: (XXXXXX): ");
    scanf("%s", telefono);

    int validoTelefono = 1;
    for (int i = 0; i < strlen(telefono); i++) {
        if (!isdigit(telefono[i])) {
            validoTelefono = 0;
            break;
        }
    }

    while (strlen(telefono) != 8 || !validoTelefono) {
        printf("Error: El teléfono debe tener exactamente 8 dígitos y contener solo números.\n");
        printf("Ingrese el teléfono del cliente, Formato: (XXXXXX): ");
        scanf("%s", telefono);

        validoTelefono = 1;
        for (int i = 0; i < strlen(telefono); i++) {
            if (!isdigit(telefono[i])) {
                validoTelefono = 0;
                break;
            }
        }
    }

    char query[200];
    sprintf(query, "INSERT INTO cliente (IdCliente, nombre, apellido, email, telefono) VALUES ('%s', '%s', '%s', '%s', '%s');", idCliente, nombre, apellido, email, telefono);

    if (mysql_query(con, query) == 0) {
        printf(stderr, "Error al ejecutar la consulta: %s\n", mysql_error(con));
        return;
    }

    printf("Cliente agregado con éxito.\n");
    printf("*****\n");
}

```

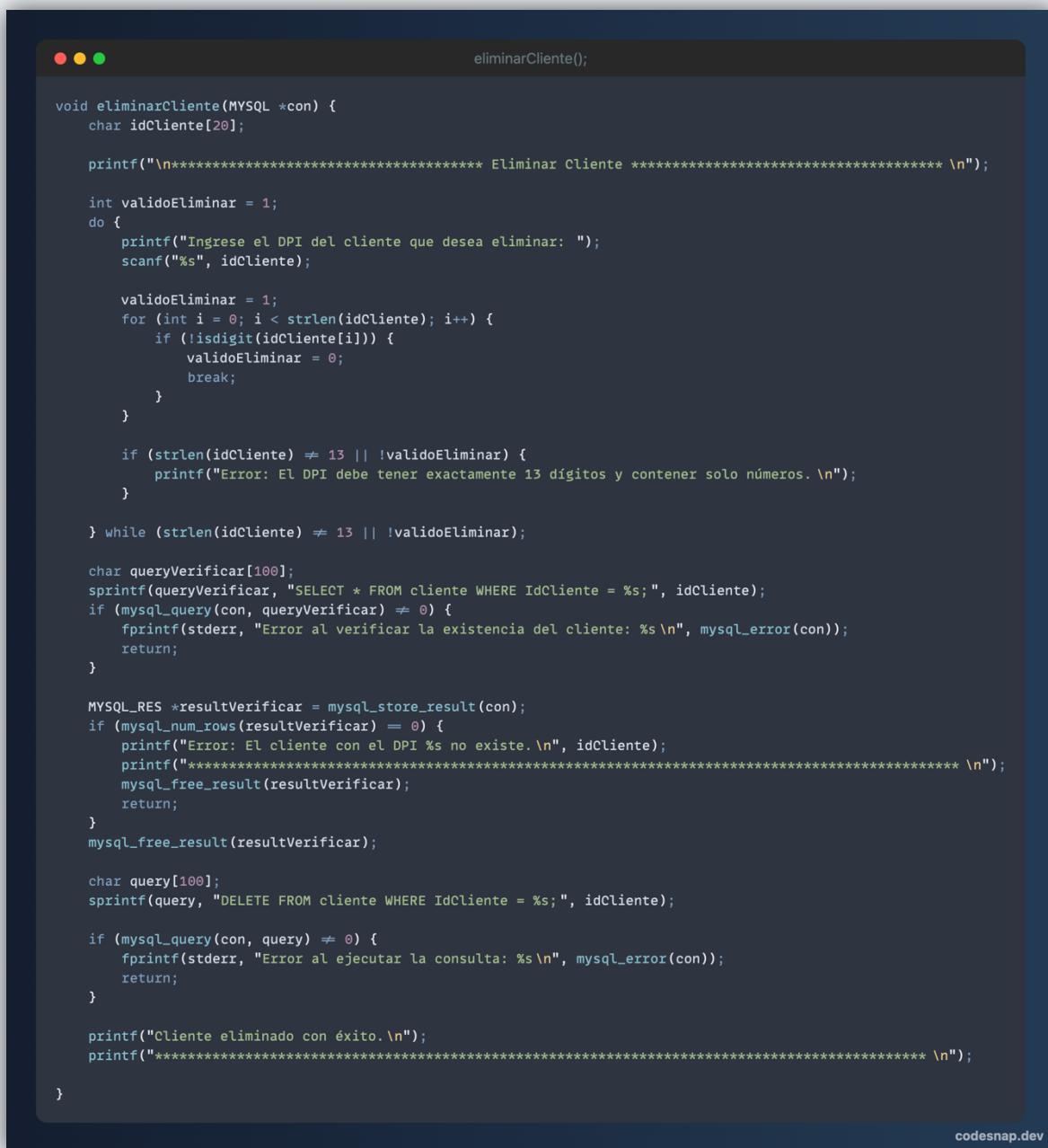
codesnap.dev

- **eliminarCliente();**

Solicita al usuario ingresar el DPI(IdCliente) del cliente que sea actualizar.

Verifica si el DPI del cliente ya existe en la base de datos debe tener 13 números y si tiene mas o menos de 13 o letras genera un error.

Si el cliente existe, realiza una consulta SQL('DELETE') para eliminar al cliente de la base de datos.



```
eliminarciente();

void eliminarcliente(MYSQL *con) {
    char idcliente[20];

    printf("\n***** Eliminar Cliente *****\n");

    int validoeliminar = 1;
    do {
        printf("Ingrese el DPI del cliente que desea eliminar: ");
        scanf("%s", idcliente);

        validoeliminar = 1;
        for (int i = 0; i < strlen(idcliente); i++) {
            if (!isdigit(idcliente[i])) {
                validoeliminar = 0;
                break;
            }
        }

        if (strlen(idcliente) != 13 || !validoeliminar) {
            printf("Error: El DPI debe tener exactamente 13 dígitos y contener solo números.\n");
        }
    } while (strlen(idcliente) != 13 || !validoeliminar);

    char queryverificar[100];
    sprintf(queryverificar, "SELECT * FROM cliente WHERE IdCliente = %s;", idcliente);
    if (mysql_query(con, queryverificar) != 0) {
        fprintf(stderr, "Error al verificar la existencia del cliente: %s\n", mysql_error(con));
        return;
    }

    MYSQL_RES *resultverificar = mysql_store_result(con);
    if (mysql_num_rows(resultverificar) == 0) {
        printf("Error: El cliente con el DPI %s no existe.\n", idcliente);
        printf("*****\n");
        mysql_free_result(resultverificar);
        return;
    }
    mysql_free_result(resultverificar);

    char query[100];
    sprintf(query, "DELETE FROM cliente WHERE IdCliente = %s;", idcliente);

    if (mysql_query(con, query) != 0) {
        fprintf(stderr, "Error al ejecutar la consulta: %s\n", mysql_error(con));
        return;
    }

    printf("Cliente eliminado con éxito.\n");
    printf("*****\n");
}
```

codesnap.dev

- **actualizarCliente();**

Solicita al usuario ingresar el DPI(IdCliente) del cliente que desea actualizar.

Verifica si el DPI del cliente ya existe en la base de datos, debe tener 13 números y si tiene mas o menos de 13 o letras genera un error.

Verifica si el número de teléfono debe tener 8 números y si tiene mas o menos de 8 o letras genera un error.

Si el cliente existe, solicita al usuario ingresar la nueva información del cliente (nombre, apellido, email, teléfono) y realiza una consulta SQL ('UPDATE') para actualizar los datos del cliente.



```

actualizarCliente();

void actualizarCliente(MYSQL *con) {
    char nuevoNombre[50], nuevoApellido[50], nuevoEmail[100], nuevoTelefono[20], idCliente[20];
    printf("\n***** Actualizar Cliente *****\n");

    int validoActualizar = 1;
    do {
        printf("Ingrese el DPI del cliente que desea actualizar: ");
        scanf("%s", idCliente);

        validoActualizar = 1;
        for (int i = 0; i < strlen(idCliente); i++) {
            if (!isdigit(idCliente[i])) {
                validoActualizar = 0;
                break;
            }
        }

        if (strlen(idCliente) != 13 || !validoActualizar) {
            printf("Error: El DPI debe tener exactamente 13 dígitos y contener solo números.\n");
        }
    } while (strlen(idCliente) != 13 || !validoActualizar);

    char queryVerificar[100];
    sprintf(queryVerificar, "SELECT * FROM cliente WHERE IdCliente = %s", idCliente);
    if (mysql_query(con, queryVerificar) == 0) {
        fprintf(stderr, "Error al verificar la existencia del cliente: %s\n", mysql_error(con));
        return;
    }

    MYSQL_RES *resultVerificar = mysql_store_result(con);
    if (mysql_num_rows(resultVerificar) == 0) {
        printf("Error: El cliente con el DPI %s no existe.\n", idCliente);
        printf("*****\n");
        mysql_free_result(resultVerificar);
        return;
    }
    mysql_free_result(resultVerificar);

    printf("Ingrese el nuevo nombre del cliente: ");
    scanf("%s", nuevoNombre);

    if (strlen(nuevoNombre) > 49) {
        printf("Error: Longitud del nombre excede el límite permitido.\n");
        printf("*****\n");
        return;
    }

    printf("Ingrese el nuevo apellido del cliente: ");
    scanf("%s", nuevoApellido);

    if (strlen(nuevoApellido) > 49) {
        printf("Error: Longitud del apellido excede el límite permitido.\n");
        printf("*****\n");
        return;
    }

    printf("Ingrese el nuevo email del cliente: ");
    scanf("%s", nuevoEmail);

    printf("Ingrese el nuevo teléfono del cliente, Formato: (XXXXXX): ");
    scanf("%s", nuevoTelefono);

    int validoTelefono = 1;
    for (int i = 0; i < strlen(nuevoTelefono); i++) {
        if (!isdigit(nuevoTelefono[i])) {
            validoTelefono = 0;
            break;
        }
    }

    while (strlen(nuevoTelefono) != 8 || !validoTelefono) {
        printf("Error: El teléfono debe tener exactamente 8 dígitos y contener solo números.\n");
        printf("Ingrese el nuevo teléfono del cliente, Formato: (XXXXXX): ");
        scanf("%s", nuevoTelefono);

        validoTelefono = 1;
        for (int i = 0; i < strlen(nuevoTelefono); i++) {
            if (!isdigit(nuevoTelefono[i])) {
                validoTelefono = 0;
                break;
            }
        }
    }

    char query[200];
    sprintf(query, "UPDATE cliente SET nombre = '%s', apellido = '%s', email = '%s', telefono = '%s' WHERE IdCliente = %s", nuevoNombre, nuevoApellido, nuevoEmail, nuevoTelefono, idCliente);

    if (mysql_query(con, query) == 0) {
        fprintf(stderr, "Error al ejecutar la consulta: %s\n", mysql_error(con));
        return;
    }

    printf("Datos del cliente actualizados con éxito.\n");
    printf("*****\n");
}

```

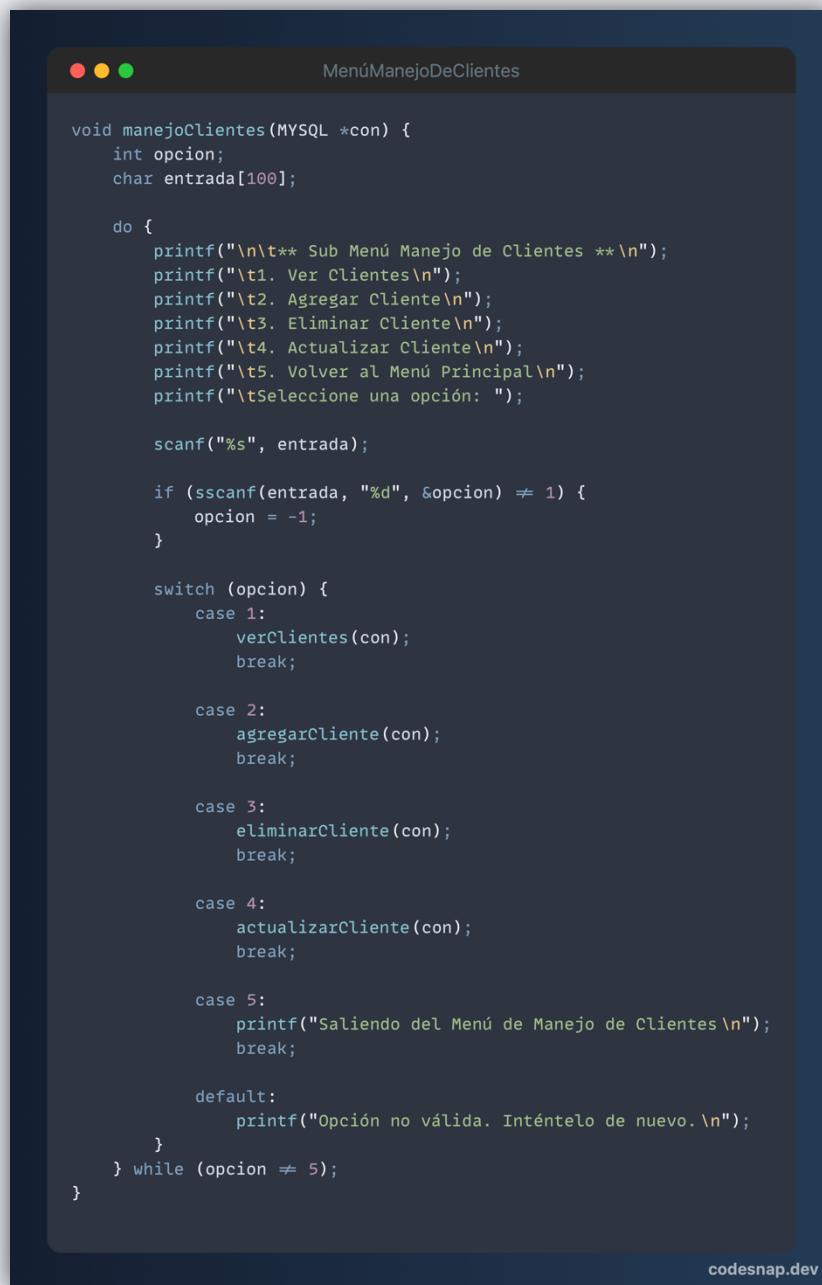
codesnap.dev

- **manejoClientes();**

Presenta un menú interactivo con las opciones mostradas en el código.

Solicita al usuario seleccionar una opción del menú, tiene validaciones que verifica si se ingresa un número valido y si no es una cadena de texto.

Llama a la función correspondiente según la opción seleccionada.



```
MenúManejoDeClientes

void manejoClientes(MYSQL *con) {
    int opcion;
    char entrada[100];

    do {
        printf("\n\t** Sub Menú Manejo de Clientes **\n");
        printf("\t1. Ver Clientes\n");
        printf("\t2. Agregar Cliente\n");
        printf("\t3. Eliminar Cliente\n");
        printf("\t4. Actualizar Cliente\n");
        printf("\t5. Volver al Menú Principal\n");
        printf("\tSeleccione una opción: ");

        scanf("%s", entrada);

        if (sscanf(entrada, "%d", &opcion) != 1) {
            opcion = -1;
        }

        switch (opcion) {
            case 1:
                verClientes(con);
                break;

            case 2:
                agregarCliente(con);
                break;

            case 3:
                eliminarCliente(con);
                break;

            case 4:
                actualizarCliente(con);
                break;

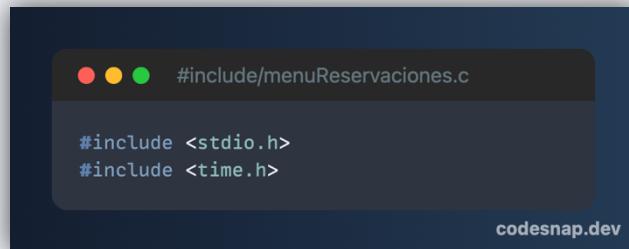
            case 5:
                printf("Saliendo del Menú de Manejo de Clientes\n");
                break;

            default:
                printf("Opción no válida. Inténtelo de nuevo.\n");
        }
    } while (opcion != 5);
}
```

codesnap.dev

## 5. menuReservaciones.c

- **#Include**



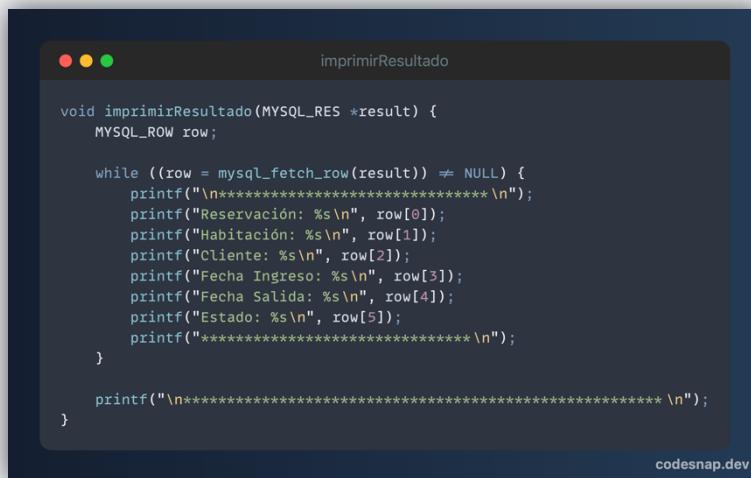
A screenshot of a terminal window titled "#include/menuReservaciones.c". The window contains the following code:

```
#include <stdio.h>
#include <time.h>
```

The terminal has three colored window control buttons (red, yellow, green) at the top left. The URL "codesnap.dev" is visible at the bottom right of the window.

- **imprimirResultado();**

Imprime en la consola la información de las reservaciones, incluyendo detalles como número de reservación (IdReservacion), habitación(IdHabitacion), cliente, fechas de ingreso y salida, y estado.



A screenshot of a terminal window titled "imprimirResultado". The window contains the following C code:

```
void imprimirResultado(MYSQL_RES *result) {
    MYSQL_ROW row;

    while ((row = mysql_fetch_row(result)) != NULL) {
        printf("\n*****\n");
        printf("Reservación: %s\n", row[0]);
        printf("Habitación: %s\n", row[1]);
        printf("Cliente: %s\n", row[2]);
        printf("Fecha Ingreso: %s\n", row[3]);
        printf("Fecha Salida: %s\n", row[4]);
        printf("Estado: %s\n", row[5]);
        printf("*****\n");
    }

    printf("\n*****\n");
}
```

The terminal has three colored window control buttons (red, yellow, green) at the top left. The URL "codesnap.dev" is visible at the bottom right of the window.

- **consultarReservasHabitacion();**  
Solicita al recepcionista ingresar el número de una habitación.

Verifica si la habitación existe.

Realiza una consulta SQL(SELECT) para obtener las reservaciones asociadas a la habitación.

Imprime la información de las reservaciones encontradas usando la función: imprimirResultado();.

```
consultarReservasHabitacion

void consultarReservasHabitacion(MYSQL *con) {
    int idHabitacion;
    char query[200];
    printf("Reservaciones Habitación por Habitación ===== \n");
    printf("Ingrese el numero de la habitación: ");
    scanf("%d", &idHabitacion);

    char queryVerificar[100];
    sprintf(queryVerificar, "SELECT * FROM habitacion WHERE IDHabitacion = %d;", idHabitacion);
    if (mysql_query(con, queryVerificar) == 0) {
        result = mysql_store_result(con);
        if (result != NULL) {
            printf("Existe la habitación con el numero %d no existe.\n", idHabitacion);
            printf("*****\n");
            mysql_free_result(result);
        }
    }
    mysql_free_result(resultVerificar);

    char query[200];
    sprintf(query, "SELECT Reservacion, Habitacion, Cliente, FechaIngreso, FechaSalida, Estado FROM verReservas WHERE Habitacion = %d; ", idHabitacion);

    if (mysql_query(con, query) == 0) {
        result = mysql_store_result(con);
        if (result != NULL) {
            printf("Error al ejecutar la consulta: %s\n", mysql_error(con));
            printf("*****\n");
            mysql_free_result(result);
        }
    }

    MYSQL_RES *result = mysql_store_result(con);
    if (result == NULL) {
        fprintf(stderr, "Error al obtener el resultado de la consulta: %s\n", mysql_error(con));
        printf("*****\n");
        return;
    }

    if (mysql_num_rows(result) == 0) {
        printf("No hay reservaciones para la habitación ingresada.\n");
        printf("*****\n");
    } else {
        imprimirResultado(result);
    }

    mysql_free_result(result);
}

codesnap.dev
```

- **verReservasFechaEspecífica();**  
Solicita al recepcionista ingresa una fecha de ingreso en un formato específico.

Realiza una consulta SQL('SELECT') para obtener las reservaciones en la fecha específica si lo hubiese.

Imprime la información de las reservaciones encontradas usando la función: imprimirResultado();.

```
verReservasFechaEspecífica

void verReservasFechaEspecífica(MYSQL *con) {
    char fecha[20];
    struct tm tm_fecha;

    printf("***** Reservas Fecha de Ingreso *****\n");
    printf("Ingrese la fecha de ingreso (YYYY-MM-DD): ");
    scanf("%s", fecha);

    if (strptime(fecha, "%Y-%m-%d", &tm_fecha) == NULL) {
        printf("Error: El formato de fecha ingresado no es válido. Use el formato YYYY-MM-DD. \n");
        printf("*****\n");
        return;
    }

    char query[200];
    sprintf(query, "SELECT Reservacion, Habitacion, Cliente, FechaIngreso, FechaSalida, Estado FROM verReservas WHERE DATE(FechaIngreso) = '%s'; ", fecha);

    if (mysql_query(con, query) == 0) {
        result = mysql_store_result(con);
        if (result != NULL) {
            printf("Error al ejecutar la consulta: %s\n", mysql_error(con));
            printf("*****\n");
            mysql_free_result(result);
        }
    }

    MYSQL_RES *result = mysql_store_result(con);
    if (result == NULL) {
        fprintf(stderr, "Error al obtener el resultado de la consulta: %s\n", mysql_error(con));
        printf("*****\n");
        return;
    }

    if (mysql_num_rows(result) == 0) {
        printf("No hay reservaciones para la fecha ingresada.\n");
        printf("*****\n");
    } else {
        imprimirResultado(result);
    }

    mysql_free_result(result);
}

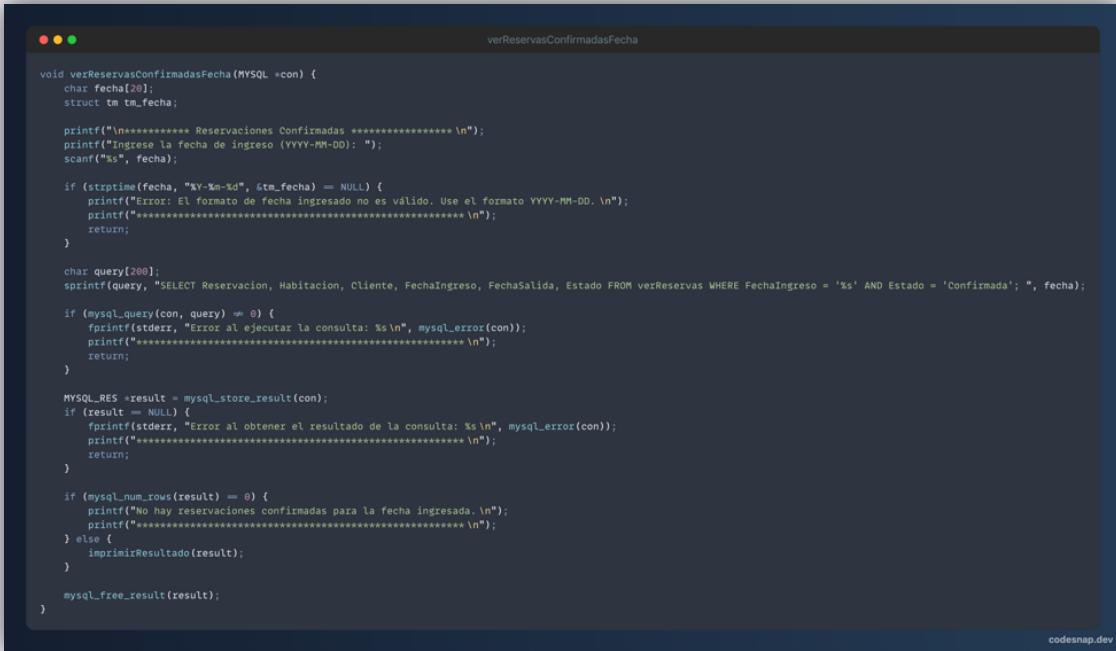
codesnap.dev
```

- **verReservasConfirmadasFecha();**

Solicita al recepcionista ingresar una fecha de ingreso en un formato específico.

Realiza una consulta SQL(SELECT) para obtener las reservaciones confirmadas en la fecha específica.

Imprime la información de las reservaciones confirmadas usando la función: imprimirResultado();



The screenshot shows a terminal window with a dark background and white text. The title bar reads "verReservasConfirmadasFecha". The code inside the window is as follows:

```
void verReservasConfirmadasFecha(MYSQL *con) {
    char fecha[20];
    struct tm tm_fecha;

    printf("\n***** Reservaciones Confirmadas *****\n");
    printf("Ingrese la fecha de ingreso (YYYY-MM-DD): ");
    scanf("%s", fecha);

    if (strptime(fecha, "%Y-%m-%d", &tm_fecha) == NULL) {
        printf("Error: El formato de fecha ingresado no es válido. Use el formato YYYY-MM-DD.\n");
        printf("*****\n");
        return;
    }

    char query[200];
    sprintf(query, "SELECT Reservacion, Habitacion, Cliente, FechaIngreso, FechaSalida, Estado FROM verReservas WHERE FechaIngreso = '%s' AND Estado = 'Confirmada';", fecha);

    if (mysql_query(con, query) != 0) {
        fprintf(stderr, "Error al ejecutar la consulta: %s\n", mysql_error(con));
        printf("*****\n");
        return;
    }

    MYSQL_RES *result = mysql_store_result(con);
    if (result == NULL) {
        fprintf(stderr, "Error al obtener el resultado de la consulta: %s\n", mysql_error(con));
        printf("*****\n");
        return;
    }

    if (mysql_num_rows(result) == 0) {
        printf("No hay reservaciones confirmadas para la fecha ingresada.\n");
        printf("*****\n");
    } else {
        imprimirResultado(result);
    }

    mysql_free_result(result);
}
```

In the bottom right corner of the terminal window, there is a small watermark that says "codesnap.dev".

- **reservacionesClientesPorId();**

Solicita al recepcionista ingresar el DPI(IdCliente) del cliente.

Verifica si el DPI del cliente ya existe en la base de datos, debe tener 13 números y si tiene mas o menos de 13 o letras genera un error.

Realiza una consulta SQL(SELECT) para obtener las reservaciones asociadas al cliente.

Imprime la información de las reservaciones encontradas.



```
reservasionesClientePorId();
```

```
void reservacionesClientePorId(MYSQL *con) {
    char idCliente[20];
    printf("\n***** Reservas de Cliente *****\n");

    int validoVerificar = 1;
    do {
        printf("Ingrese el DPI del cliente: ");
        scanf("%s", idCliente);

        validoVerificar = 1;
        for (int i = 0; i < strlen(idCliente); i++) {
            if (!isdigit(idCliente[i])) {
                validoVerificar = 0;
                break;
            }
        }

        if (strlen(idCliente) != 13 || !validoVerificar) {
            printf("Error: El DPI debe tener exactamente 13 dígitos y contener solo números.\n");
        }
    } while (strlen(idCliente) != 13 || !validoVerificar);

    char query[200];
    sprintf(query, "SELECT * FROM verCliente WHERE Dpi = %s;", idCliente);

    if (mysql_query(con, query) != 0) {
        fprintf(stderr, "Error al ejecutar la consulta: %s\n", mysql_error(con));
        return;
    }

    MYSQL_RES *result = mysql_store_result(con);

    if (result == NULL) {
        fprintf(stderr, "Error al obtener el resultado de la consulta: %s\n", mysql_error(con));
        return;
    }

    if (mysql_num_rows(result) == 0) {
        printf("No hay reservaciones para el cliente con el DPI proporcionado o el DPI no existe.\n");
        printf("*****\n");
    } else {
        MYSQL_ROW row;
        while ((row = mysql_fetch_row(result)) != NULL) {
            printf("\n*****\n");
            printf("Reservación: %s\nHabitación: %s\nDPI: %s\nCliente: %s\nFecha Ingreso: %s\nFecha Salida: %s\nEstado: %s\n",
                  row[0], row[1], row[2], row[3], row[4], row[5], row[6]);
            printf("*****\n");
        }
        printf("\n*****\n");
    }

    mysql_free_result(result);
}
```

codesnap.dev

- **agregarReservacion();**

Solicita al recepcionista la información necesaria para la nueva reservación, como el número de habitación, el DPI del cliente, y las fechas de ingreso y salida.

Realiza una consulta SQL para verificar si la habitación especificada por el recepcionista existe en la base de datos. Si no existe, muestra un mensaje de error y sale de la función.

Valida que el dpi solamente tenga 13 números, y que le número de telefono solamente tenga 8.

Verifica mediante una consulta SQL para verificar si existe una reservación confirmada para la habitacion y fecha de ingreso especificadas. Si existe, muestra un mensaje de error y se sale. Y realiza una consulta SQL para obtener el ID de la ultima fila insertada en la tabla de reservaciones para mostrar el número de reservación. Muestra al recepcionista información ingresada para la nueva reservación, y el estado aparece como: "Confirmada".

```

    agregarReservacion();

void agregarReservacion(MYSQL *con) {
    int idHabitacion;
    char fechaIngreso[20], fechaSalida[20], estado[] = "Confirmada";
    printf("***** Agregar Reservación *****\n");
    printf("Ingrese el numero de habitación: ");
    scanf("%s", idHabitacion);
    char queryVerificarHabitacion[100];
    sprintf(queryVerificarHabitacion, "SELECT * FROM habitacion WHERE IdHabitacion = %d; ", idHabitacion);
    if (mysql_query(con, queryVerificarHabitacion) == 0) {
        fprintf(stderr, "Error al verificar la habitación: %s\n", mysql_error(con));
        return;
    }
    MYSQL_RES resultVerificarHabitacion = mysql_store_result(con);
    if (mysql_num_rows(resultVerificarHabitacion) == 0) {
        printf("Error: La habitación con el número %d no existe.\n", idHabitacion);
        printf("*****\n");
        mysql_free_result(resultVerificarHabitacion);
        return;
    }
    mysql_free_result(resultVerificarHabitacion);

    int validoDPI = 1;
    do {
        printf("Ingrese el DPI del cliente: ");
        scanf("%s", idCliente);
        validoDPI = 1;
        for (int i = 0; i < strlen(idCliente); i++) {
            if (!isdigit(idCliente[i])) {
                validoDPI = 0;
                break;
            }
        }
        if (strlen(idCliente) != 13 || !validoDPI) {
            printf("Error: El DPI debe tener exactamente 13 dígitos y contener solo números.\n");
        } else {
            char queryVerificarCliente[100];
            sprintf(queryVerificarCliente, "SELECT * FROM cliente WHERE IdCliente = %s; ", idCliente);
            if (mysql_query(con, queryVerificarCliente) == 0) {
                fprintf(stderr, "Error al verificar la existencia del cliente: %s\n", mysql_error(con));
                return;
            }
            MYSQL_RES resultVerificarCliente = mysql_store_result(con);
            if (mysql_num_rows(resultVerificarCliente) == 0) {
                printf("Error: El cliente con el DPI %s no existe.\n", idCliente);
                printf("*****\n");
                mysql_free_result(resultVerificarCliente);
                return;
            }
            mysql_free_result(resultVerificarCliente);
        }
    } while (strlen(idCliente) != 13 || !validoDPI);

    printf("Ingrese fecha de ingreso (YYYY-MM-DD): ");
    scanf("%s", fechaIngreso);
    struct tm tmFecha;
    if (strptime(fechaIngreso, "%Y-%m-%d", &tmFecha) == NULL) {
        printf("Error: El formato de la fecha de ingreso no es válido.\n");
        printf("*****\n");
        return;
    }

    char queryVerificarReservacionExiste[200];
    sprintf(queryVerificarReservacionExiste, "SELECT * FROM reservacion WHERE IdHabitacion = %d AND Fecha_Ingreso = '%s' AND Estado = 'Cancelada'; ", idHabitacion, fechaIngreso);
    if (mysql_query(con, queryVerificarReservacionExiste) == 0) {
        fprintf(stderr, "Error al verificar la existencia de reservaciones: %s\n", mysql_error(con));
        return;
    }

    MYSQL_RES resultVerificarReservacionExiste = mysql_store_result(con);
    if (mysql_num_rows(resultVerificarReservacionExiste) > 0) {
        MYSQL_ROW rowReservacionExiste = mysql_fetch_row(resultVerificarReservacionExiste);
        if (strcmp(rowReservacionExiste[4], "Cancelada") == 0) {
            printf("Error: La reserva registrada la nueva reserva ya que la reserva anterior está cancelada.\n");
            mysql_free_result(resultVerificarReservacionExiste);
            return;
        }
    }
    mysql_free_result(resultVerificarReservacionExiste);

    printf("Ingrese fecha de salida (YYYY-MM-DD): ");
    scanf("%s", fechaSalida);
    struct tm tmFecha2;
    if (strptime(fechaSalida, "%Y-%m-%d", &tmFecha2) == NULL) {
        printf("Error: El formato de la fecha de salida no es válido.\n");
        printf("*****\n");
        return;
    }

    char queryInsertar[100];
    sprintf(queryInsertar, "INSERT INTO reservacion (IdHabitacion, IdCliente, Fecha_Ingreso, Fecha_Salida, Estado) VALUES (%d, %s, '%s', '%s', '%s'); ", idHabitacion, idCliente, fechaIngreso, fechaSalida, estado);

    if (mysql_query(con, queryInsertar) == 0) {
        fprintf(stderr, "Error al ejecutar la consulta: %s\n", mysql_error(con));
        return;
    }

    if (mysql_query(con, "SELECT LAST_INSERT_ID();") == 0) {
        fprintf(stderr, "Error al obtener el ID de la última fila insertada: %s\n", mysql_error(con));
        return;
    }

    MYSQL_RES resultID = mysql_store_result(con);
    if (!resultID) {
        fprintf(stderr, "Error al obtener el resultado de la consulta para el ID: %s\n", mysql_error(con));
        return;
    }

    MYSQL_ROW row = mysql_fetch_row(resultID);
    if (!row) {
        fprintf(stderr, "Error: No se pudo obtener el ID de la última fila insertada.\n");
        mysql_free_result(resultID);
        return;
    }

    int idReservacion = atoi(row[0]);
    mysql_free_result(resultID);
    printf("*****\n");
    printf("Reservación agregada exitosamente.\n");
    printf("Número de Reservación: %d\n", idReservacion);
    printf("Habitación: %d\n", idHabitacion);
    printf("Cliente: %s\n", idCliente);
    printf("Fecha de Ingreso: %s\n", fechaIngreso);
    printf("Fecha de Salida: %s\n", fechaSalida);
    printf("Estado: %s\n", estado);
    printf("*****\n");
    return;
}

```

codesnap.dev

- actualizarReservacion();**

Solicita el número de reservación que desea actualizar, Realiza una consulta de SQL para verificar si la reservación especificada por el recepcionista existe en la base de datos, si no existe, muestra un mensaje de error y sale.

Solicita nuevas fecha de ingreso y salida. Y verifica conflicto (si hay alguna otra reservación confirmada para la misma fecha y misma habitación).

Si todas las verificaciones son existosas, ejecuta una consulta SQL para actualizar las fechas de ingreso y salida de la reservación en la base de datos, muestra un mensaje indicando que se actualizó exitosamente.

```

void actualizarReservacion(mysql_conn *con) {
    int idReservacion;
    char nuevaFechaIngreso[20], nuevaFechaSalida[20];
    char (*queryVerify)[100];
    MYSQL_RES *resultVerify;
    MYSQL_ROW rowVerify;
    char (*queryConflict)[100];
    MYSQL_RES *resultConflict;
    MYSQL_ROW rowConflict;
    char (*queryUpdate)[100];
    MYSQL_RES *resultUpdate;
    MYSQL_ROW rowUpdate;
    char (*queryExistencia)[100];
    MYSQL_RES *resultExistencia;
    MYSQL_ROW rowExistencia;

    printf("Ingresar el número de reservación a actualizar: ");
    scanf("%d", &idReservacion);

    char queryVerify[100];
    sprintf(queryVerify, "SELECT * FROM reservacion WHERE IdReservacion = %d; ", idReservacion);
    if (mysql_query(con, queryVerify) == 0) {
        mysql_free_result(resultVerify);
        return;
    }

    resultVerify = mysql_use_result(resultVerify);
    if (mysql_fetch_row(resultVerify) == 0) {
        printf("Error: La reservación con el número %d no existe.\n", idReservacion);
        mysql_free_result(resultVerify);
        return;
    }

    MySQL_ROW row = mysql_use_result(resultVerify);
    char *estadoReservacion = row[6];
    mysql_free_result(resultVerify);

    if (strcmp(estadoReservacion, "Cancelada") == 0) {
        printf("Error: No se puede actualizar la reservación porque está cancelada.\n");
        mysql_free_result(resultConflict);
        return;
    }

    printf("Ingresar la nueva fecha de ingreso (YYYY-MM-DD): ");
    scanf("%s", nuevaFechaIngreso);
    struct tm tmeFecha;
    if (strptime(nuevaFechaIngreso, "%Y-%m-%d", &tmeFecha) == NULL) {
        printf("Error: El formato de la fecha de ingreso no es válido.\n");
        mysql_free_result(resultConflict);
        return;
    }

    char queryConflict[100];
    sprintf(queryConflict, "SELECT * FROM reservacion WHERE IdHabitacion = (SELECT IdHabitacion FROM reservacion WHERE IdReservacion = %d) AND Estado = 'Confirmada' AND ((%s BETWEEN Fecha_Ingreso AND Fecha_Salida) OR (%s BETWEEN Fecha_Ingreso AND Fecha_Salida)); ", idReservacion, idReservacion, nuevaFechaIngreso, nuevaFechaSalida);
    if (mysql_query(con, queryConflict) == 0) {
        mysql_free_result(resultConflict);
        return;
    }

    resultConflict = mysql_use_result(resultConflict);
    if (mysql_num_rows(resultConflict) > 0) {
        printf("Error: La nueva fecha de ingreso coincide con otra reservación confirmada en la misma habitación. \n");
        mysql_free_result(resultConflict);
        return;
    }

    MySQL_ROW rowConflict = mysql_use_result(resultConflict);
    if (rowConflict[6] == "Confirmada") {
        printf("Error: Ya hay una reservación confirmada para la misma habitación en la nueva fecha de ingreso. \n");
        mysql_free_result(resultConflict);
        return;
    }

    mysql_free_result(resultConflict);
    mysql_free_result(resultExistencia);

    printf("Ingresar la nueva fecha de salida (YYYY-MM-DD): ");
    scanf("%s", nuevaFechaSalida);

    struct tm tmeFecha2;
    if (strptime(nuevaFechaSalida, "%Y-%m-%d", &tmeFecha2) == NULL) {
        printf("Error: El formato de la Fecha de salida no es válido.\n");
        mysql_free_result(resultExistencia);
        return;
    }

    char queryUpdate[100];
    sprintf(queryUpdate, "UPDATE reservacion SET Fecha_Ingreso = '%s', Fecha_Salida = '%s' WHERE IdReservacion = %d; ", nuevaFechaIngreso, nuevaFechaSalida, idReservacion);

    if (mysql_query(con, queryUpdate) == 0) {
        mysql_free_result(resultExistencia);
        return;
    }

    printf("Las fechas de la reservación actualizadas exitosamente. \n");
    printf("-----\n");
}

```

codenap.dev

- **cancelarReservacion();**

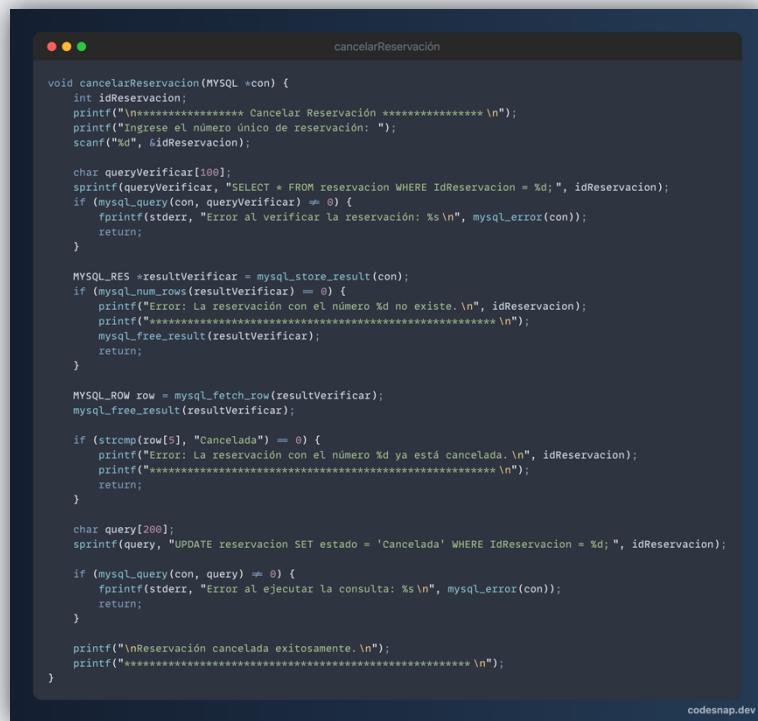
Solicita al recepcionista ingresar el número de reservación(IdReservacion).

Verifica si la reservación existe.

Realiza una consulta SQL(UPDATE) para cambiar el estado de la reservación a ‘Cancelada’.

Verifica si el estado de la reservación es “Cancelada”. Si es así, imprime un mensaje de error y termina la función, regresando al menú principal.

Si la todo es satisfactorio crea la consulta SQL(UPDATE) para actualizar el estado de la reservación a “Cancelada” y regresa al menú principal.



The screenshot shows a terminal window with a dark background and white text. The title bar says "cancelarReservación". The code inside the window is as follows:

```
void cancelarReservacion(MYSQL *con) {
    int idReservacion;
    printf("***** Cancelar Reservación *****\n");
    printf("Ingrese el número único de reservación: ");
    scanf("%d", &idReservacion);

    char queryVerificar[100];
    sprintf(queryVerificar, "SELECT * FROM reservacion WHERE IdReservacion = %d;", idReservacion);
    if (mysql_query(con, queryVerificar) != 0) {
        fprintf(stderr, "Error al verificar la reservación: %s\n", mysql_error(con));
        return;
    }

    MYSQL_RES *resultVerificar = mysql_store_result(con);
    if (mysql_num_rows(resultVerificar) == 0) {
        printf("Error: La reservación con el número %d no existe.\n", idReservacion);
        printf("*****\n");
        mysql_free_result(resultVerificar);
        return;
    }

    MYSQL_ROW row = mysql_fetch_row(resultVerificar);
    mysql_free_result(resultVerificar);

    if (strcmp(row[5], "Cancelada") == 0) {
        printf("Error: La reservación con el número %d ya está cancelada.\n", idReservacion);
        printf("*****\n");
        return;
    }

    char query[200];
    sprintf(query, "UPDATE reservacion SET estado = 'Cancelada' WHERE IdReservacion = %d;", idReservacion);

    if (mysql_query(con, query) != 0) {
        fprintf(stderr, "Error al ejecutar la consulta: %s\n", mysql_error(con));
        return;
    }

    printf("\nReservación cancelada exitosamente.\n");
    printf("*****\n");
}
```

At the bottom right of the terminal window, it says "codesnap.dev".

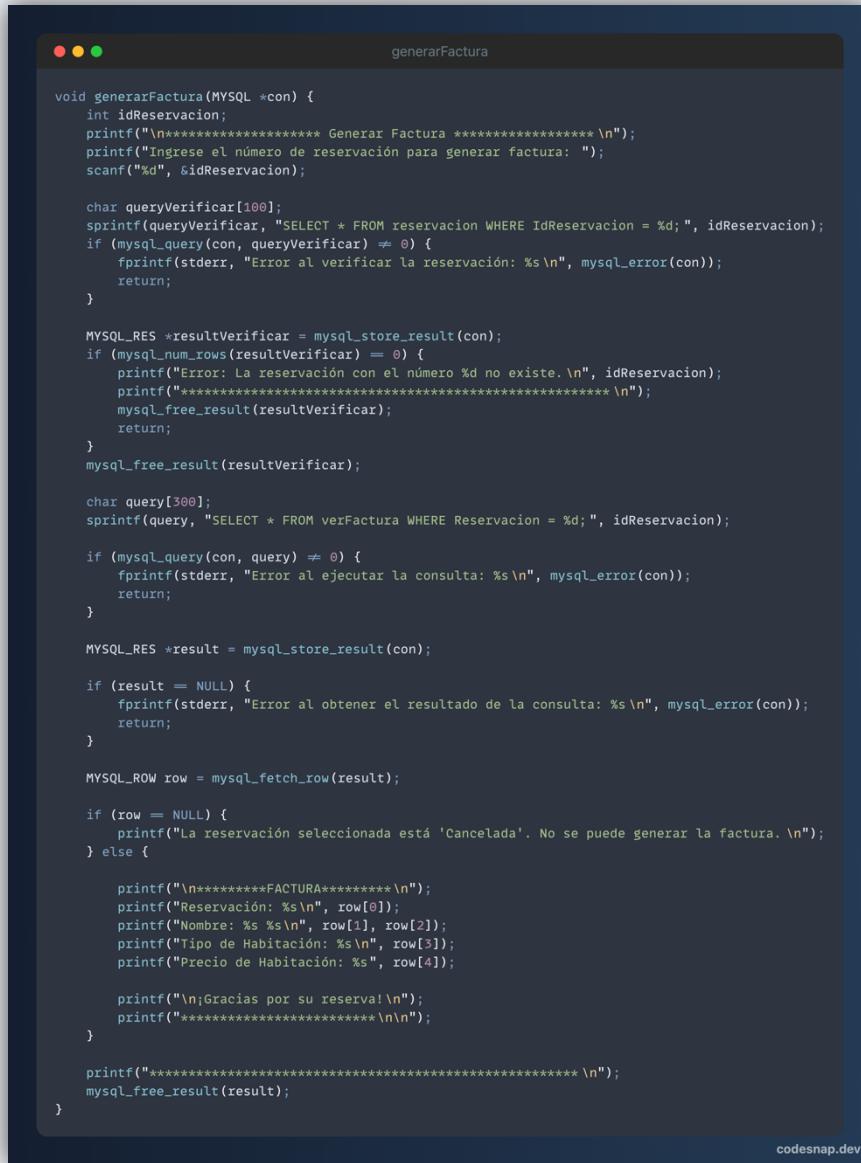
- **generarFactura();**

Solicita al recepcionista ingresar el número de reservación.

Verifica si la reservación existe.

Realiza una consulta SQL(SELECT) para obtener la información necesaria para genera la factura.

Imprime la factura, incluyendo detalles como nombre, tipo de habitación y precio.



```
void generarFactura(MYSQL *con) {
    int idReservacion;
    printf("\n*****Generar Factura *****\n");
    printf("Ingrese el número de reservación para generar factura: ");
    scanf("%d", &idReservacion);

    char queryVerificar[100];
    sprintf(queryVerificar, "SELECT * FROM reservacion WHERE IdReservacion = %d;", idReservacion);
    if (mysql_query(con, queryVerificar) != 0) {
        fprintf(stderr, "Error al verificar la reservación: %s\n", mysql_error(con));
        return;
    }

    MYSQL_RES *resultVerificar = mysql_store_result(con);
    if (mysql_num_rows(resultVerificar) == 0) {
        printf("Error: La reservación con el número %d no existe.\n", idReservacion);
        printf("*****\n");
        mysql_free_result(resultVerificar);
        return;
    }
    mysql_free_result(resultVerificar);

    char query[300];
    sprintf(query, "SELECT * FROM verFactura WHERE Reservacion = %d;", idReservacion);

    if (mysql_query(con, query) != 0) {
        fprintf(stderr, "Error al ejecutar la consulta: %s\n", mysql_error(con));
        return;
    }

    MYSQL_RES *result = mysql_store_result(con);
    if (result == NULL) {
        fprintf(stderr, "Error al obtener el resultado de la consulta: %s\n", mysql_error(con));
        return;
    }

    MYSQL_ROW row = mysql_fetch_row(result);

    if (row == NULL) {
        printf("La reservación seleccionada está 'Cancelada'. No se puede generar la factura.\n");
    } else {

        printf("\n*****FACTURA*****\n");
        printf("Reservación: %s\n", row[0]);
        printf("Nombre: %s\n", row[1], row[2]);
        printf("Tipo de Habitación: %s\n", row[3]);
        printf("Precio de Habitación: %s", row[4]);

        printf("\n;Gracias por su reserva!\n");
        printf("*****\n");
    }

    printf("*****\n");
    mysql_free_result(result);
}
```

codesnap.dev

- **manejoReservaciones();**

Presenta un menú interactivo con las opciones mostradas en el código.

Solicita al recepcionista seleccionar una opción valida del menú en la cual verifica si esta dentro del rango de numeros correcto y si es una cadena de texto.

Llama a la función correspondiente según la opción seleccionada.

The screenshot shows a terminal window with a dark background and light-colored text. The title bar reads "MenúManejoDeReservaciones". The code inside the window is as follows:

```
void manejoReservaciones(MYSQL *con) {
    int opcion;
    char entrada[100];

    do {
        printf("\n\t** Sub Menú Manejo de Reservaciones **\n");
        printf("\t1. Historial de reservaciones por habitación\n");
        printf("\t2. Reservaciones para fecha específica\n");
        printf("\t3. Reservaciones confirmadas para fecha específica\n");
        printf("\t4. Reservaciones para cliente específico (DPI)\n");
        printf("\t5. Agregar una reservación\n");
        printf("\t6. Actualizar datos de una reservación\n");
        printf("\t7. Cancelar una reservación\n");
        printf("\t8. Generar factura por número de reservación\n");
        printf("\t9. Volver al Menú Principal\n");
        printf("\tSeleccione una opción: ");

        scanf("%s", entrada);

        if (sscanf(entrada, "%d", &opcion) != 1) {
            opcion = -1;
        }

        switch (opcion) {
            case 1:
                consultarReservasHabitacion(con);
                break;
            case 2:
                verReservacionesFechaEspecifica(con);
                break;
            case 3:
                verReservasConfirmadasFecha(con);
                break;
            case 4:
                reservacionesClientePorId(con);
                break;
            case 5:
                agregarReservacion(con);
                break;
            case 6:
                actualizarReservacion(con);
                break;
            case 7:
                cancelarReservacion(con);
                break;
            case 8:
                generarFactura(con);
                break;
            case 9:
                printf("Saliendo del Menú de Manejo de Reservaciones\n");
                break;
            default:
                printf("Opción no válida. Inténtelo de nuevo.\n");
        }
    } while (opcion != 9);
}
```

In the bottom right corner of the terminal window, the text "codesnap.dev" is visible.

# CONCLUSIONES

- **Gestión Integrada de Habitaciones:**

- El desarrollo de un sistema eficiente para gestionar la disponibilidad, estado y asignación de habitaciones ha resultado en una mayor organización y agilidad en el manejo de los recursos del hotel.
- La interfaz clara y visual proporciona al recepcionista la capacidad de acceder rápidamente a la información relevante de las habitaciones, permitiéndole realizar modificaciones de manera efectiva.

- **Facilidades en el Manejo de Clientes:**

- La implementación de funcionalidades diseñadas para simplificar la gestión de clientes ha optimizado las operaciones diarias del recepcionista.
- La capacidad de ver, agregar, actualizar y eliminar información de clientes, junto con las validaciones incorporadas, ha mejorado la precisión de los datos y reducido el riesgo de duplicaciones.

- **Optimización de Procesos de Reservación:**

- La mejora en la eficiencia del manejo de reservaciones ha llevado a una mayor productividad en la gestión diaria de reservas.
- Las funciones detalladas para consultar, agregar, actualizar y cancelar reservas han facilitado el trabajo del recepcionista, proporcionándole herramientas más poderosas para administrar las reservaciones. Además, la información histórica y las alertas automatizadas han contribuido a una gestión más proactiva y efectiva de las reservaciones.