**Código del GitHub colaborativo del grupo:** <https://github.com/FcoGomez10/UMH-BD2-2025P1-GRUPO6/tree/main/Proyecto>

**Código del GitHub de Edgardo Gomez:**

**Código del GitHub de Jhon Turcios:**

**Código del GitHub de Edwin Marquina:**

**Código del GitHub de Gustavo Ferrufino:** <https://github.com/Tavhoo/UMH-BD2-2025P1-202300171/tree/main/Base%20de%20datos%20II%20/Proyectos>

**—----------- Script DDL de base de datos.sql —----------**

# PROYECTO FORMULARIO DE HOTEL

-- -----------------------------------------------------

-- Schema proyecto2

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `proyecto2` DEFAULT CHARACTER SET utf8;

USE `proyecto2`;

-- -----------------------------------------------------

-- Table `proyecto2`.`pais`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `proyecto2`.`pais` (

`id\_pais` INT NOT NULL AUTO\_INCREMENT,

`nombre` VARCHAR(45) NULL,

PRIMARY KEY (`id\_pais`)

);

-- -----------------------------------------------------

-- Table `proyecto2`.`ciudad`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `proyecto2`.`ciudad` (

`id\_ciudad` INT NOT NULL AUTO\_INCREMENT,

`nombre` VARCHAR(45) NULL,

`id\_pais` INT NOT NULL,

PRIMARY KEY (`id\_ciudad`),

INDEX `fk\_ciudad\_pais1\_idx` (`id\_pais` ASC) VISIBLE,

CONSTRAINT `fk\_ciudad\_pais1`

FOREIGN KEY (`id\_pais`)

REFERENCES `proyecto2`.`pais` (`id\_pais`)

ON DELETE NO ACTION

ON UPDATE NO ACTION

);

-- -----------------------------------------------------

-- Table `proyecto2`.`domicilio`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `proyecto2`.`domicilio` (

`id\_domicilio` INT NOT NULL AUTO\_INCREMENT,

`direccion` VARCHAR(45) NULL,

`id\_ciudad` INT NOT NULL,

PRIMARY KEY (`id\_domicilio`),

INDEX `fk\_domicilio\_ciudad1\_idx` (`id\_ciudad` ASC) VISIBLE,

CONSTRAINT `fk\_domicilio\_ciudad1`

FOREIGN KEY (`id\_ciudad`)

REFERENCES `proyecto2`.`ciudad` (`id\_ciudad`)

ON DELETE NO ACTION

ON UPDATE NO ACTION

);

-- -----------------------------------------------------

-- Table `proyecto2`.`cliente`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `proyecto2`.`cliente` (

`id\_cliente` INT NOT NULL AUTO\_INCREMENT,

`nombre` VARCHAR(45) NULL,

`apellido` VARCHAR(45) NULL,

`organizacion` VARCHAR(45) NULL,

`cargo` VARCHAR(45) NULL,

`numPasaporte` VARCHAR(45) NULL,

`fechaNacimiento` DATE NULL,

`nacionalidad` VARCHAR(45) NULL,

`telefono` VARCHAR(45) NULL,

`fax` VARCHAR(45) NULL,

`email` VARCHAR(45) NULL,

`id\_domicilio` INT NOT NULL,

PRIMARY KEY (`id\_cliente`),

INDEX `fk\_participante\_domicilio1\_idx` (`id\_domicilio` ASC) VISIBLE,

CONSTRAINT `fk\_participante\_domicilio1`

FOREIGN KEY (`id\_domicilio`)

REFERENCES `proyecto2`.`domicilio` (`id\_domicilio`)

ON DELETE NO ACTION

ON UPDATE NO ACTION

);

-- -----------------------------------------------------

-- Table `proyecto2`.`hotel`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `proyecto2`.`hotel` (

`id\_hotel` INT NOT NULL AUTO\_INCREMENT,

`nombre` VARCHAR(45) NULL,

PRIMARY KEY (`id\_hotel`)

);

-- -----------------------------------------------------

-- Table `proyecto2`.`tipoTarjeta`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `proyecto2`.`tipoTarjeta` (

`id\_tipoTarjeta` INT NOT NULL AUTO\_INCREMENT,

`tipo` VARCHAR(45) NULL,

PRIMARY KEY (`id\_tipoTarjeta`)

);

-- -----------------------------------------------------

-- Table `proyecto2`.`tarjeta`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `proyecto2`.`tarjeta` (

`id\_tarjeta` INT NOT NULL AUTO\_INCREMENT,

`numTarjeta` VARCHAR(45) NULL,

`fechaVencimiento` DATE NULL,

`nombreTitular` VARCHAR(45) NULL,

`id\_cliente` INT NOT NULL,

`id\_tipoTarjeta` INT NOT NULL,

PRIMARY KEY (`id\_tarjeta`),

INDEX `fk\_tarjeta\_participante1\_idx` (`id\_cliente` ASC) VISIBLE,

INDEX `fk\_tarjeta\_tipoTarjeta1\_idx` (`id\_tipoTarjeta` ASC) VISIBLE,

CONSTRAINT `fk\_tarjeta\_participante1`

FOREIGN KEY (`id\_cliente`)

REFERENCES `proyecto2`.`cliente` (`id\_cliente`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_tarjeta\_tipoTarjeta1`

FOREIGN KEY (`id\_tipoTarjeta`)

REFERENCES `proyecto2`.`tipoTarjeta` (`id\_tipoTarjeta`)

ON DELETE NO ACTION

ON UPDATE NO ACTION

);

-- -----------------------------------------------------

-- Table `proyecto2`.`tipoHabitacion`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `proyecto2`.`tipoHabitacion` (

`id\_tipoHabitacion` INT NOT NULL AUTO\_INCREMENT,

`tipoHabitacion` VARCHAR(45) NULL,

PRIMARY KEY (`id\_tipoHabitacion`)

);

-- -----------------------------------------------------

-- Table `proyecto2`.`acomodacion`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `proyecto2`.`acomodacion` (

`id\_acomodacion` INT NOT NULL AUTO\_INCREMENT,

`tipo` VARCHAR(45) NULL,

PRIMARY KEY (`id\_acomodacion`)

);

-- -----------------------------------------------------

-- Table `proyecto2`.`habitacion`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `proyecto2`.`habitacion` (

`id\_habitacion` INT NOT NULL AUTO\_INCREMENT,

`precio` DECIMAL NULL,

`id\_hotel` INT NOT NULL,

`id\_tipoHabitacion` INT NOT NULL,

`id\_acomodacion` INT NOT NULL,

PRIMARY KEY (`id\_habitacion`),

INDEX `fk\_Habitacion\_hotel1\_idx` (`id\_hotel` ASC) VISIBLE,

INDEX `fk\_habitacion\_tipoHabitacion1\_idx` (`id\_tipoHabitacion` ASC) VISIBLE,

INDEX `fk\_habitacion\_acomodacion1\_idx` (`id\_acomodacion` ASC) VISIBLE,

CONSTRAINT `fk\_Habitacion\_hotel1`

FOREIGN KEY (`id\_hotel`)

REFERENCES `proyecto2`.`hotel` (`id\_hotel`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_habitacion\_tipoHabitacion1`

FOREIGN KEY (`id\_tipoHabitacion`)

REFERENCES `proyecto2`.`tipoHabitacion` (`id\_tipoHabitacion`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_habitacion\_acomodacion1`

FOREIGN KEY (`id\_acomodacion`)

REFERENCES `proyecto2`.`acomodacion` (`id\_acomodacion`)

ON DELETE NO ACTION

ON UPDATE NO ACTION

);

-- -----------------------------------------------------

-- Table `proyecto2`.`reserva`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `proyecto2`.`reserva` (

`id\_reserva` INT NOT NULL AUTO\_INCREMENT,

`fechaArribo` DATE NULL,

`fechaSalida` DATE NULL,

`numNoches` INT NULL,

`vueloArribo` DATE NULL,

`vueloSalida` DATE NULL,

`early\_check\_in` VARCHAR(45) NULL,

`id\_cliente` INT NOT NULL,

`id\_tarjeta` INT NOT NULL,

`id\_habitacion` INT NOT NULL,

PRIMARY KEY (`id\_reserva`),

INDEX `fk\_reserva\_participante1\_idx` (`id\_cliente` ASC) VISIBLE,

INDEX `fk\_reserva\_tarjeta1\_idx` (`id\_tarjeta` ASC) VISIBLE,

INDEX `fk\_reserva\_habitacion1\_idx` (`id\_habitacion` ASC) VISIBLE,

CONSTRAINT `fk\_reserva\_participante1`

FOREIGN KEY (`id\_cliente`)

REFERENCES `proyecto2`.`cliente` (`id\_cliente`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_reserva\_tarjeta1`

FOREIGN KEY (`id\_tarjeta`)

REFERENCES `proyecto2`.`tarjeta` (`id\_tarjeta`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_reserva\_habitacion1`

FOREIGN KEY (`id\_habitacion`)

REFERENCES `proyecto2`.`habitacion` (`id\_habitacion`)

ON DELETE NO ACTION

ON UPDATE NO ACTION

);

-- -----------------------------------------------------

-- Table `proyecto2`.`personaContacto`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `proyecto2`.`personaContacto` (

`id\_contacto` INT NOT NULL AUTO\_INCREMENT,

`nombre` VARCHAR(45) NULL,

`telefono` VARCHAR(45) NULL,

`fax` VARCHAR(45) NULL,

`email` VARCHAR(45) NULL,

`id\_hotel` INT NOT NULL,

PRIMARY KEY (`id\_contacto`),

INDEX `fk\_contacto\_hotel\_idx` (`id\_hotel` ASC) VISIBLE,

CONSTRAINT `fk\_contacto\_hotel`

FOREIGN KEY (`id\_hotel`)

REFERENCES `proyecto2`.`hotel` (`id\_hotel`)

ON DELETE NO ACTION

ON UPDATE NO ACTION

);

**—-------------Script DML de datos de prueba --------------**

# -------------- Inserciones Cliente ------------------------

# Inserción de países

INSERT INTO pais (nombre)

VALUES

("Honduras"),

("Mexico"),

("USA");

# Inserción de ciudades

INSERT INTO ciudad (nombre, id\_pais)

VALUES

("Ciudad de Mexico", 1);

# Inserción de domicilios

INSERT INTO domicilio (direccion, id\_ciudad)

VALUES

("AV. Reforma 200", 1);

# Inserción de clientes

INSERT INTO cliente

(nombre, apellido, organizacion, cargo, numPasaporte,

fechaNacimiento, nacionalidad, telefono, fax, email, id\_domicilio)

VALUES

('Carlos',

'González',

'Empresa A',

'Gerente',

'X123456',

'1985-05-10',

'Mexicano',

'555-1234',

'555-5678',

'carlos@email.com',

1);

# Consulta para verificar la inserción de clientes

SELECT \* FROM proyecto2.cliente;

# ------------------------ Inserciones Hotel ----------------------------

# Inserción de hoteles

INSERT INTO hotel (nombre)

VALUES

("Plaza San Francisco"),

("Caesar Business Santiago"),

("Sheraton Santiago"),

("San Cristobal Tower");

# Inserción de tipos de habitación

INSERT INTO tipoHabitacion (tipoHabitacion)

VALUES

("Standard"),

("Classic"),

("Grand Deluxe");

# Inserción de tipos de acomodación

INSERT INTO acomodacion (tipo)

VALUES

("single"),

("doble"),

("twin");

# Inserción de habitaciones

INSERT INTO habitacion (id\_hotel, id\_tipoHabitacion, id\_acomodacion, precio)

VALUES

(1, 1, 1, 125.00);

# Consulta para verificar la inserción de habitaciones

SELECT \* FROM habitacion;

# --------------------------- Inserciones Tarjeta ----------------------------

# Inserción de tipos de tarjeta

INSERT INTO tipoTarjeta (tipo)

VALUES

('AMEX'),

('VISA'),

('MasterCard'),

('Diners');

# Inserción de tarjetas

INSERT INTO tarjeta (numTarjeta, fechaVencimiento, nombreTitular, id\_cliente, id\_tipoTarjeta)

VALUES

('9999-9999-9999-9999', '2025/01/15', 'Carlos Gonzales', 1, 1);

# Consulta para verificar la inserción de tarjetas

SELECT \* FROM proyecto2.tarjeta;

**—------------- Script DDL de procedimientos —-------------**

# ------------- Actualizar tabla Tarjeta ------------------

DELIMITER //

CREATE PROCEDURE proyecto2.sp\_tarjeta\_upd (

IN p\_id\_tarjeta INT,

IN p\_id\_cliente INT,

IN p\_tipoTarjeta INT,

IN p\_numTarjeta VARCHAR(45),

IN p\_fechaVencimiento DATE,

IN p\_nombreTitular VARCHAR(45)

)

BEGIN

DECLARE v\_idExiste INT;

# Verifica si la tarjeta existe

SELECT COUNT(\*) INTO v\_idExiste

FROM tarjeta

WHERE id\_tarjeta = p\_id\_tarjeta;

# Condición para realizar el update de la tabla tarjeta

IF v\_idExiste > 0 THEN

UPDATE tarjeta

SET numTarjeta = p\_numTarjeta,

fechaVencimiento = p\_fechaVencimiento,

nombreTitular = p\_nombreTitular,

id\_cliente = p\_id\_cliente,

id\_tipoTarjeta = p\_tipoTarjeta

WHERE id\_tarjeta = p\_id\_tarjeta;

ELSE

SELECT 'No se puede realizar el proceso' AS Resultado;

END IF;

END // DELIMITER ;

/\* Llamada al procedimiento para actualizar la tarjeta.

Parametros (p\_id\_tarjeta, p\_id\_cliente, p\_tipoTarjeta,

p\_numTarjeta, p\_fechaVencimiento, p\_nombreTitular); \*/

CALL proyecto2.sp\_tarjeta\_upd(1, 1, 1, '8888-8888-8888-8888', '2025/02/24', 'Carlos Gonzales');

# Consulta para verificar la actualización de la tarjeta

SELECT \* FROM tarjeta;

# ----------------- Actualizar tabla de habitacion --------------------------

DELIMITER //

CREATE PROCEDURE proyecto2.sp\_habitacion\_upd (

IN p\_id\_habitacion INT,

IN p\_precio DECIMAL(10, 2),

IN p\_id\_hotel INT,

IN p\_id\_tipoHabitacion INT,

IN p\_id\_acomodacion INT

)

BEGIN

DECLARE v\_idExiste INT;

# Verifica si la habitación existe

SELECT COUNT(\*) INTO v\_idExiste

FROM habitacion

WHERE id\_habitacion = p\_id\_habitacion;

# Condición para realizar el update de la tabla habitacion

IF v\_idExiste > 0 THEN

UPDATE habitacion

SET precio = p\_precio,

id\_hotel = p\_id\_hotel,

id\_tipoHabitacion = p\_id\_tipoHabitacion,

id\_acomodacion = p\_id\_acomodacion

WHERE id\_habitacion = p\_id\_habitacion;

ELSE

SELECT 'No se puede realizar el proceso' AS Resultado;

END IF;

END // DELIMITER ;

/\* Llamada al procedimiento para actualizar la habitación.

Parametros (p\_id\_habitacion, p\_precio, p\_id\_hotel, p\_id\_tipoHabitacion, p\_id\_acomodacion); \*/

CALL sp\_habitacion\_upd(

1,

190.00,

3,

2,

2

);

# Consulta para verificar la actualización de la habitación

SELECT \* FROM habitacion;

# ------------------------- Actualizar tabla de cliente ---------------------------

DELIMITER //

CREATE PROCEDURE proyecto2.sp\_cliente\_upd (

IN p\_id\_cliente INT,

IN p\_nombre VARCHAR(45),

IN p\_apellido VARCHAR(45),

IN p\_organizacion VARCHAR(45),

IN p\_cargo VARCHAR(45),

IN p\_numPasaporte VARCHAR(45),

IN p\_fechaNacimiento DATE,

IN p\_nacionalidad VARCHAR(45),

IN p\_telefono VARCHAR(45),

IN p\_fax VARCHAR(45),

IN p\_email VARCHAR(45),

IN p\_id\_domicilio INT

)

BEGIN

DECLARE v\_idExiste INT;

# Verifica si el cliente existe

SELECT COUNT(\*) INTO v\_idExiste

FROM cliente

WHERE id\_cliente = p\_id\_cliente;

# Condición para realizar el update de la tabla cliente

IF v\_idExiste > 0 THEN

UPDATE cliente

SET id\_cliente = p\_id\_cliente,

nombre = p\_nombre,

apellido = p\_apellido,

organizacion = p\_organizacion,

cargo = p\_cargo,

numPasaporte = p\_numPasaporte,

fechaNacimiento = p\_fechaNacimiento,

nacionalidad = p\_nacionalidad,

telefono = p\_telefono,

fax = p\_fax,

email = p\_email,

id\_domicilio = p\_id\_domicilio

WHERE id\_cliente = p\_id\_cliente;

ELSE

SELECT 'No se puede realizar el proceso' AS Resultado;

END IF;

END // DELIMITER ;

/\* Llamada al procedimiento para actualizar el cliente.

parametros (p\_id\_cliente, p\_nombre, p\_apellido, p\_organizacion, p\_cargo,

p\_numPasaporte, p\_fechaNacimiento, p\_nacionalidad, p\_telefono,

p\_fax, p\_email, p\_id\_domicilio); \*/

CALL proyecto2.sp\_cliente\_upd(

1,

'Gustavo',

'Ferrufino',

'Empresa B',

'Gerente General',

'Z123456',

'1990/05/30',

'Americano', '

9885-2587',

'9885-2588',

'gferrufino@email.com',

1

);

# Consulta para verificar la actualización del cliente

SELECT \* FROM cliente;

**—--------------- Script DDL de funciones ---------------**

# ------------------ Función Total a pagar ----------------------------------

DELIMITER //

CREATE FUNCTION proyecto2.fn\_get\_total (

p\_precio DOUBLE, # Precio por noche de la habitación

p\_numNoches DOUBLE # Número de noches de la reserva

)

RETURNS DOUBLE DETERMINISTIC

BEGIN

DECLARE total DOUBLE; # Variable para almacenar el total a pagar

# Calcular el total a pagar

SET total = p\_precio \* p\_numNoches;

RETURN total;

END // DELIMITER ;

# Llamada a la función para calcular el total a pagar

# Orden para ingresar los parametros (p\_precio, p\_numNoches)

SELECT fn\_get\_total(142, 3) AS Total;

# ---------------------- Función Cantidad de Reservaciones -------------------

DELIMITER //

CREATE FUNCTION proyecto2.fn\_get\_reservaciones (

p\_id\_cliente INT # ID del cliente para contar sus reservas

)

RETURNS INT DETERMINISTIC

BEGIN

DECLARE v\_count INT; # Variable para almacenar el número de reservas

# Contar las reservas del cliente

SELECT COUNT(\*) INTO v\_count

FROM reserva

WHERE id\_cliente = p\_id\_cliente;

RETURN v\_count;

END // DELIMITER ;

# Llamada a la función para obtener la cantidad de reservaciones de un cliente

# Orden para ingresar los parametros (p\_id\_cliente)

SELECT fn\_get\_reservaciones(1) AS Reservaciones;

# ---------------------- Función Confirmación de Reservaciones -------------------

DELIMITER //

CREATE FUNCTION proyecto2.fn\_confirmacion (

id\_cliente INT, # ID del cliente para obtener su nombre

id\_hotel INT # ID del hotel para obtener su nombre

)

RETURNS VARCHAR(200) DETERMINISTIC

BEGIN

DECLARE cliente VARCHAR(45); # Variable para almacenar el nombre del cliente

DECLARE hotel VARCHAR(45); # Variable para almacenar el nombre del hotel

# Obtener el nombre completo del cliente

SET cliente = (

SELECT CONCAT(nombre, " ", apellido)

FROM cliente

WHERE id\_cliente = id\_cliente

);

# Obtener el nombre del hotel

SET hotel = (

SELECT nombre

FROM hotel

WHERE id\_hotel = id\_hotel

LIMIT 1

);

# Verificar si el hotel existe y retornar el mensaje correspondiente

IF hotel IS NOT NULL THEN

RETURN CONCAT(cliente, " gracias por su reservación en el hotel ", hotel);

ELSE

RETURN CONCAT(cliente, " no tiene hotel asignado");

END IF;

END // DELIMITER ;

# Llamada a la función para obtener la confirmación de la reserva

# Orden para ingresar los parametros (id\_cliente, id\_hotel)

SELECT fn\_confirmacion(1, 1) AS Confirmacion;