UNIVERSIDAD METROPOLITANA DE HONDURAS

Innovación, Valores, Liderazgo

Informe final de Proyecto Grupo#5

Integrantes :

Helen Barrientos 202403157

Yasandra Hernández 202302647

Maryi Julissa Pavón 202302648

Nauh Hernández 202102911

Catedrático:

Allan Noel López Cruz

Sección:

1900-IIN-065

Asignatura:

Base de Datos # 2

Fecha:

28/02/25

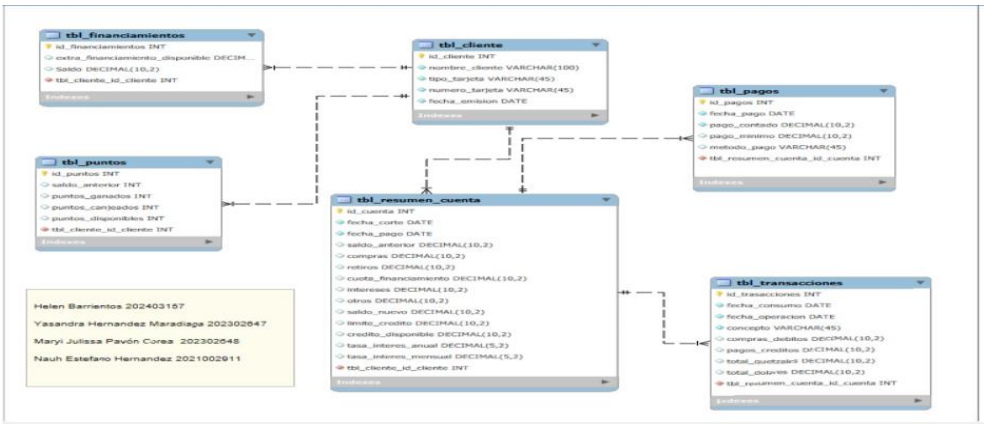
Año:2025

Descripción de problema

Nuestro proyecto se llama Estado de cuenta.

Es un estado de cuenta de una tarjeta de crédito de Banrural. Este documento contiene información detallada sobre los saldos de la tarjeta, los pagos realizados, las compras, los intereses generados, y los puntos acumulados, entre otros datos financieros. Para convertir este estado de cuenta en un modelo relacional, se necesitaría crear tablas que representen las diferentes categorías de datos mencionadas en el documento. Y así vamos definiendo la tablas conforme la información de la imagen.

- ## Modelo relacional



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

```
-- MySQL Workbench Synchronization

-- Generated: 2025-03-23 15:53

-- Model: New Model

-- Version: 1.0

-- Project: Name of the project

-- Author: Helen Barrientos

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

SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,

FOREIGN_KEY_CHECKS=0;

SET @OLD_SQL_MODE=@@SQL_MODE,

SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZE

RO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

ALTER SCHEMA `db_estado_cuenta` DEFAULT CHARACTER SET utf8 DEFAULT COLLATE

utf8_general_ci ;

ALTER TABLE `db_estado_cuenta`.`tbl_transacciones`

DROP FOREIGN KEY `fk_tbl_transacciones_tbl_cuenta1`;

ALTER TABLE `db_estado_cuenta`.`tbl_puntos`

DROP FOREIGN KEY `fk_tbl_puntos_tbl_cliente1`;

ALTER TABLE `db_estado_cuenta`.`tbl_pagos`

DROP FOREIGN KEY `fk_tbl_pagos_tbl_cuenta1`;
```

```
ALTER TABLE `db_estado_cuenta`.`tbl_financiamientos`
DROP FOREIGN KEY `fk_tbl_financiamientos_tbl_cliente1`;
ALTER TABLE `db_estado_cuenta`.`tbl_tarjeta`
DROP FOREIGN KEY `fk_tbl_tarjeta_tbl_cliente1`,
DROP FOREIGN KEY `fk_tbl_tarjeta_tbl_tipo_tarjeta1`;
ALTER TABLE `db_estado_cuenta`.`tbl_cliente`
CHARACTER SET = utf8 , COLLATE = utf8_general_ci ;
CREATE TABLE IF NOT EXISTS `db_estado_cuenta`.`tbl_cuenta` (
 `id_cuenta` INT(11) NOT NULL AUTO_INCREMENT,
 `fecha_corte` DATE NOT NULL,
 `fecha_pago` DATE NOT NULL,
 `saldo_anterior` DECIMAL(10,2) NULL DEFAULT NULL,
 `compras` DECIMAL(10,2) NULL DEFAULT NULL,
 `retiros` DECIMAL(10,2) NULL DEFAULT NULL,
 `cuota_financiamiento` DECIMAL(10,2) NULL DEFAULT NULL,
 `intereses` DECIMAL(10,2) NULL DEFAULT NULL,
 `otros` DECIMAL(10,2) NULL DEFAULT NULL,
 `saldo_nuevo` DECIMAL(10,2) NULL DEFAULT NULL,
 `limite_credito` DECIMAL(10,2) NULL DEFAULT NULL,
 `credito_disponible` DECIMAL(10,2) NULL DEFAULT NULL,
 `tasa_interes_anual` DECIMAL(5,2) NULL DEFAULT NULL,
 `tasa_interes_mensual` DECIMAL(5,2) NULL DEFAULT NULL,
 `tbl_cliente` INT(11) NOT NULL,
 PRIMARY KEY (`id_cuenta`),
```

```
    INDEX `fk_tbl_resumen_cuenta_tbl_cliente_idx` (`tbl_cliente` ASC) VISIBLE,
    CONSTRAINT `fk_tbl_resumen_cuenta_tbl_cliente`
    FOREIGN KEY (`tbl_cliente`)
    REFERENCES `db_estado_cuenta`.`tbl_cliente` (`id_cliente`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
  ENGINE = InnoDB
  DEFAULT CHARACTER SET = utf8;
  ALTER TABLE `db_estado_cuenta`.`tbl_transacciones`
  CHARACTER SET = utf8 , COLLATE = utf8_general_ci ,
  DROP COLUMN `tbl_cuenta`,
  ADD COLUMN `tbl_cuenta` INT(11) NOT NULL AFTER `total_dolares`,
  ADD INDEX `fk_tbl_transacciones_tbl_cuenta1_idx` (`tbl_cuenta` ASC) VISIBLE,
  DROP INDEX `fk_tbl_transacciones_tbl_cuenta1_idx` ;
  ;
  ALTER TABLE `db_estado_cuenta`.`tbl_puntos`
  CHARACTER SET = utf8 , COLLATE = utf8_general_ci ,
  DROP COLUMN `tbl_cliente`,
  ADD COLUMN `tbl_cliente` INT(11) NOT NULL AFTER `puntos_disponibles`,
  ADD INDEX `fk_tbl_puntos_tbl_cliente1_idx` (`tbl_cliente` ASC) VISIBLE,
  DROP INDEX `fk_tbl_puntos_tbl_cliente1_idx` ;
  ;
```

```
ALTER TABLE `db_estado_cuenta`.`tbl_pagos`
CHARACTER SET = utf8 , COLLATE = utf8_general_ci ,
DROP COLUMN `tbl_cuenta`,
DROP COLUMN `tbl_metodos_pago`,
ADD COLUMN `tbl_metodos_pago` INT(11) NOT NULL AFTER `pago_minimo`,
ADD COLUMN `tbl_cuenta` INT(11) NOT NULL AFTER `tbl_metodos_pago`,
ADD INDEX `fk_tbl_pagos_tbl_metodos_pago1_idx` (`tbl_metodos_pago` ASC) VISIBLE,
ADD INDEX `fk_tbl_pagos_tbl_cuenta1_idx` (`tbl_cuenta` ASC) VISIBLE,
DROP INDEX `fk_tbl_pagos_tbl_cuenta1_idx` ,
DROP INDEX `fk_tbl_pagos_tbl_metodos_pago1_idx` ;
;
ALTER TABLE `db_estado_cuenta`.`tbl_financiamientos`
CHARACTER SET = utf8 , COLLATE = utf8_general_ci ,
DROP COLUMN `tbl_cliente`,
ADD COLUMN `tbl_cliente` INT(11) NOT NULL AFTER `Saldo`,
ADD INDEX `fk_tbl_financiamientos_tbl_cliente1_idx` (`tbl_cliente` ASC) VISIBLE,
DROP INDEX `fk_tbl_financiamientos_tbl_cliente1_idx` ;
;
ALTER TABLE `db_estado_cuenta`.`tbl_tarjeta`
CHARACTER SET = utf8 , COLLATE = utf8_general_ci ,
DROP COLUMN `tbl_tipo_tarjeta`,
DROP COLUMN `tbl_cliente`,
```

```
    ADD COLUMN `tbl_cliente` INT(11) NOT NULL AFTER `fecha_emision`,
    ADD COLUMN `tbl_tipo_tarjeta` INT(11) NOT NULL AFTER `tbl_cliente`,
    ADD INDEX `fk_tbl_tarjeta_tbl_cliente1_idx` (`tbl_cliente` ASC) VISIBLE,
    ADD INDEX `fk_tbl_tarjeta_tbl_tipo_tarjeta1_idx` (`tbl_tipo_tarjeta` ASC)
VISIBLE,
    DROP INDEX `fk_tbl_tarjeta_tbl_tipo_tarjeta1_idx` ,
    DROP INDEX `fk_tbl_tarjeta_tbl_cliente1_idx` ;
    ;
    ALTER TABLE `db_estado_cuenta`.`tbl_metodos_pago`
    CHARACTER SET = utf8 , COLLATE = utf8_general_ci ;
    ALTER TABLE `db_estado_cuenta`.`tbl_tipo_tarjeta`
    CHARACTER SET = utf8 , COLLATE = utf8_general_ci ;
    ALTER TABLE `db_estado_cuenta`.`tbl_transacciones`
    ADD CONSTRAINT `fk_tbl_transacciones_tbl_cuenta1`
     FOREIGN KEY (`tbl_cuenta`)
     REFERENCES `db_estado_cuenta`.`tbl_cuenta` (`id_cuenta`)
     ON DELETE NO ACTION
     ON UPDATE NO ACTION;
    ALTER TABLE `db_estado_cuenta`.`tbl_puntos`
    ADD CONSTRAINT `fk_tbl_puntos_tbl_cliente1`
     FOREIGN KEY (`tbl_cliente`)
     REFERENCES `db_estado_cuenta`.`tbl_cliente` (`id_cliente`)
     ON DELETE NO ACTION
     ON UPDATE NO ACTION;
    ALTER TABLE `db_estado_cuenta`.`tbl_pagos`
```

```
DROP FOREIGN KEY `fk_tbl_pagos_tbl_metodos_pago1`;
ALTER TABLE `db_estado_cuenta`.`tbl_pagos` ADD CONSTRAINT
`fk_tbl_pagos_tbl_metodos_pago1`
 FOREIGN KEY (`tbl_metodos_pago`)
 REFERENCES `db_estado_cuenta`.`tbl_metodos_pago` (`id_metodos_pago`)
 ON DELETE NO ACTION
 ON UPDATE NO ACTION,
ADD CONSTRAINT `fk_tbl_pagos_tbl_cuenta1`
 FOREIGN KEY (`tbl_cuenta`)
 REFERENCES `db_estado_cuenta`.`tbl_cuenta` (`id_cuenta`)
 ON DELETE NO ACTION
 ON UPDATE NO ACTION;
ALTER TABLE `db_estado_cuenta`.`tbl_financiamientos`
ADD CONSTRAINT `fk_tbl_financiamientos_tbl_cliente1`
 FOREIGN KEY (`tbl_cliente`)
 REFERENCES `db_estado_cuenta`.`tbl_cliente` (`id_cliente`)
 ON DELETE NO ACTION
 ON UPDATE NO ACTION;
ALTER TABLE `db_estado_cuenta`.`tbl_tarjeta`
ADD CONSTRAINT `fk_tbl_tarjeta_tbl_cliente1`
 FOREIGN KEY (`tbl_cliente`)
 REFERENCES `db_estado_cuenta`.`tbl_cliente` (`id_cliente`)
 ON DELETE NO ACTION
 ON UPDATE NO ACTION,
```

```
ADD CONSTRAINT `fk_tbl_tarjeta_tbl_tipo_tarjeta1`
 FOREIGN KEY (`tbl_tipo_tarjeta`)
 REFERENCES `db_estado_cuenta`.`tbl_tipo_tarjeta` (`idl_tipo_tarjeta`)
 ON DELETE NO ACTION
 ON UPDATE NO ACTION;
SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

• Script DML de datos de prueba

```
SELECT * FROM db_estado_cuenta.tbl_cliente;
/* Insertando datos de prueba en la tabla cliente */
INSERT INTO db_estado_cuenta.tbl_cliente (id_cliente, nombre_cliente,
numero_identidad, telefono, direccion)
VALUE
(1, 'Marta', '0801199400012', '123456789', 'avenida mirafloes'),
(2, 'Jeff', '0801199733333', '1235426589', 'calle 44,miraflores'),
(3, 'Danielle', '0108197800090', '0422753321', 'col colinas');
 SELECT * FROM db_estado_cuenta.tbl_financiamientos;
/* Insertando datos de prueba en la financiamientos */
 INSERT INTO db_estado_cuenta.tbl_financiamientos (tbl_cliente,
extra_financiamiento_disponible, saldo)
 VALUES
(1, 5000.00, 12000.50),
(2, 3000.00, 7500.00),
```

```
(3, 8000.00, 15000.75);


SELECT * FROM db_estado_cuenta.tbl_metodos_pago;

/* Insertando datos de prueba en la tabla medodos pago */

INSERT INTO db_estado_cuenta.tbl_metodos_pago (nombre_metodo)

VALUE

 ('Efectivo'),

('Tarjeta de Crédito'),

('Tarjeta de Débito'),

('Transferencia Bancaria'),

('PayPal'),

('Criptomonedas');

 /* Inserción de datos de prueba para tbl_cuenta */

INSERT INTO db_estado_cuenta.tbl_cuenta (fecha_corte, fecha_pago, saldo_anterior,

compras, retiros, cuota_financiamiento, intereses, otros, saldo_nuevo, limite_credito,

credito_disponible, tasa_interes_anual)

VALUES

('2025-03-01', '2025-03-10', 5000.00, 1500.00, 500.00, 200.00, 50.00, 30.00, 6280.00,

10000.00, 3720.00, 18.50);
```

• Script DDL de procedimientos

```
/*PROCEDIMIENTO #1 */

SELECT * FROM db_estado_cuenta.tbl_cliente;
```

```
DROP PROCEDURE IF EXISTS sp_actualizar_telefono_cliente;
DELIMITER //
/* Procedimiento para actualizar el teléfono del cliente */
CREATE PROCEDURE db_estado_cuenta.sp_actualizar_telefono_cliente(
 IN p_id_cliente INT,
 IN p_nuevo_telefono VARCHAR(15)
)
BEGIN
 UPDATE tbl_cliente SET telefono = p_nuevo_telefono WHERE id_cliente = p_id_cliente;
END //
DELIMITER //
/* ejuctar el procedimiento para probarlo usando call */
Call db_estado_cuenta.sp_actualizar_telefono_cliente(
"1", "93254401"
 );
/*PRODECIMIENTO #2 */
SELECT * FROM db_estado_cuenta.tbl_pagos;
/* Eliminar el procedimiento si ya existe */
DROP PROCEDURE IF EXISTS registrar_pago;
/* Crear el procedimiento para registar pago*/
DELIMITER //
CREATE PROCEDURE db_estado_cuenta.sp_registrar_pago(
 IN p_fecha_pago DATE,
 IN p_pago_contado DECIMAL(10,2),
```

```
 IN p_pago_minimo DECIMAL(10,2),
 IN p_metodo_pago INT,
 IN p_cuenta INT
)
BEGIN
 /* Insertar el pago en la tabla */
INSERT INTO tbl_pagos (fecha_pago, pago_contado, pago_minimo,
tbl_metodos_pago, tbl_cuenta)
 VALUES (p_fecha_pago, p_pago_contado, p_pago_minimo, p_metodo_pago,
p_cuenta);
END //
DELIMITER ;
/* Ejecutar el procedimiento para probarlo */
CALL db_estado_cuenta.sp_registrar_pago('2025-03-23', 500.00, 200.00, 1,
101);
/* PROCEDIMEINTO 3 */
SELECT * FROM db_estado_cuenta.tbl_tarjeta;
DROP PROCEDURE IF EXISTS sp_agregar_numero_tarjeta;
DELIMITER //
CREATE PROCEDURE db_estado_cuenta.sp_agregar_numero_tarjeta(
 IN p_nuevo_numero_tarjeta VARCHAR(19),
 IN p_tbl_cliente INT,
 IN p_tbl_tipo_tarjeta INT -- Agregar parámetro para tipo de tarjeta
)
BEGIN
```

```
    INSERT INTO tbl_tarjeta (numero_tarjeta, tbl_cliente, tbl_tipo_tarjeta)
    VALUES (p_nuevo_numero_tarjeta, p_tbl_cliente, p_tbl_tipo_tarjeta);
END //
DELIMITER //
CALL  db_estado_cuenta.sp_agregar_numero_tarjeta('4446552365687894',  1,
1);
/* PROCEDIMIENTO #4 */
/* Procedimiento para actualizar la dirección del cliente */
SELECT * FROM db_estado_cuenta.tbl_cliente;
DELIMITER //
CREATE PROCEDURE db_estado_cuenta.sp_actualizar_direccion_cliente(
    IN p_id_cliente INT,
    IN p_nueva_direccion VARCHAR(255)
)
BEGIN
    /* Validar que la dirección no sea nula ni vacía y que el ID del cliente sea válido
*/
    IF p_nueva_direccion IS NOT NULL AND p_nueva_direccion <> ''
    AND p_id_cliente IS NOT NULL AND p_id_cliente > 0 THEN

    UPDATE tbl_cliente
    SET direccion = p_nueva_direccion
    WHERE id_cliente = p_id_cliente;

    COMMIT; -- Confirmar la transacción
```

```
 ELSE
 -- Mensaje si no se puede actualizar
 SELECT 'No se puede actualizar la dirección';
 END IF;
END //
DELIMITER //
/* Ejecutar el procedimiento para probarlo */
CALL db_estado_cuenta.sp_actualizar_direccion_cliente("2",
"col.kennedy,Tegucigalpa");
```

• Script DDL de funciones

```
/* FUNCION #1 */
/* Eliminar la función si ya existe */
DROP FUNCTION IF EXISTS fn_calcular_descuento;
/* Cambiar delimitador para evitar problemas con los puntos y comas dentro de
la función
*/
DELIMITER //
/* Crear la función */
CREATE FUNCTION fn_calcular_descuento(
 p_monto DECIMAL(10,2),
 p_descuento DECIMAL(5,2)
) RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
 /* Calcular el nuevo monto con el descuento aplicado */
```

```
 DECLARE v_monto_final DECIMAL(10,2);
 SET v_monto_final = p_monto - (p_monto * p_descuento / 100);
 RETURN v_monto_final;
END //
/* Restaurar el delimitador original */
DELIMITER ;
/* Probar la función */
SELECT fn_calcular_descuento(1000, 10) AS Monto_Con_Descuento;
/* FUNCTION #2 */
/* Eliminar la función si ya existe */
DROP FUNCTION IF EXISTS fn_obtener_nombre_cliente;
/* Cambiar delimitador para evitar problemas con los puntos y comas dentro de
la función
*/
DELIMITER //
/* Crear la función para obtener el nombre completo de un cliente */
CREATE FUNCTION fn_obtener_nombre_cliente(
 p_cliente_id INT
) RETURNS VARCHAR(255)
DETERMINISTIC
BEGIN
 /* Declarar variable para almacenar el nombre del cliente */
 DECLARE v_nombre_cliente VARCHAR(255);

 /* Seleccionar el nombre completo del cliente con el ID proporcionado */
```

```
    SELECT CONCAT(nombre_cliente, ' ', apellido_cliente) INTO
v_nombre_cliente
    FROM tbl_cliente
    WHERE id_cliente = p_cliente_id;

    /* Retornar el nombre completo */
    RETURN v_nombre_cliente;
   END //
   /* Restaurar el delimitador original */
   DELIMITER ;
   SELECT fn_obtener_nombre_cliente(1) AS nombre_completo_cliente;
   /* Funcion para obtener el saldo actual de una cuenta */
   CREATE FUNCTION fn_obtener_saldo_cuenta(p_id_cuenta INT) RETURNS
DECIMAL(10,2)
   DETERMINISTIC
   BEGIN
    DECLARE saldo_actual DECIMAL(10,2);

    SELECT saldo_nuevo INTO saldo_actual
    FROM tbl_cuenta
    WHERE id_cuenta = p_id_cuenta;

    RETURN saldo_actual;
   END //
   /*Uso:
```

Esta función devuelve el saldo actual de una cuenta específica.

Para ejecutarla, utilice la siguiente consulta:

SELECT fn_obtener_saldo_cuenta(1);*/

SELECT fn_obtener_saldo_cuenta(1)

- **Creación de triggers**

```
delimiter //
DROP TRIGGER tgr_log_insercion ;
delimiter //
CREATE TRIGGER tgr_log_insercion BEFORE INSERT ON tbl_cuenta
FOR EACH ROW
BEGIN
    DECLARE v_datos varchar (4000);
  SET v_datos := CONCAT(
  (NEW.id_cuenta, '), ',', COALESCE(NEW.fecha_corte, '), ',', COALESCE(NEW.fecha_pago, '), ',', COALESCE(NEW.saldo_anterior, '), ',',
  COALESCE(NEW.compras, '), ',', COALESCE(NEW.retiros, '), ',', COALESCE(NEW.cuota_financiamiento, '), ',', COALESCE(NEW.intereses, '), ',',
  COALESCE(NEW.otros, '), ',', COALESCE(NEW.saldo_nuevo, '), ',', COALESCE(NEW.limite_credito, '), ',', COALESCE(NEW.credito_disponible, '), ',',
  COALESCE(NEW.tasa_interes_anual, '), ',', COALESCE(NEW.tasa_interes_mensual, '), ',', COALESCE(NEW.tbl_cliente, ' ) );
  insert into t_log_inserciones(fecha, tipo_evento, datos )
    values(now(), 'INSERCION', v_datos ) ;
  END;
  //
```

```
drop table t_log_inserciones ;

CREATE TABLE t_log_inserciones (

id int AUTO_INCREMENT PRIMARY KEY,

fecha datetime, tipo_evento varchar (45) , datos varchar (4000)  );

SELECT * FROM t_log_inserciones ;

INSERT INTO tbl_cuenta (fecha_corte, fecha_pago, saldo_anterior, compras, retiros, cuota_financiamiento, intereses, otros, saldo_nuevo, limite_credito, credito_disponible, tasa_interes_anual, tbl_cliente)

VALUES

('2025-03-03','2025-03-15', 7000.00, 1000.00, 300.00, 160.00, 70.00, 20.00, 7700.00, 10000.00, 4300.00, 18.50, 1);

SELECT * FROM tbl_cuenta;

DELETE FROM tbl_cuenta

WHERE id_cuenta = 5;

drop table t_log_inserciones ;

CREATE TABLE t_log_inserciones (

id int AUTO_INCREMENT PRIMARY KEY,

fecha datetime, tipo_evento varchar (45) , datos varchar (4000)  );

SELECT * FROM t_log_inserciones ;

INSERT INTO tbl_cuenta (fecha_corte, fecha_pago, saldo_anterior, compras, retiros, cuota_financiamiento, intereses, otros, saldo_nuevo, limite_credito, credito_disponible, tasa_interes_anual, tbl_cliente)

VALUES

('2025-03-03','2025-03-15', 7000.00, 1000.00, 300.00, 160.00, 70.00, 20.00, 7700.00, 10000.00, 4300.00, 18.50, 1);

SELECT * FROM tbl_cuenta;
```

```
DELETE FROM tbl_cuenta

WHERE id_cuenta = 5;
```

Conclusiones

- Aprendimos que no solo se trata solo de guardar datos, sino de organizarlos de manera que sean fáciles de consultar, modificar y mantener. Un mal diseño puede generar problemas de rendimiento, redundancia y pérdida de información.
- En este proyecto incluimos inserción de datos de pruebas en varias tabla, lo cual nos permite probar la funcionalidad de la base de datos, por eso es importante la relación entre las tablas, para asegurar que los datos estén correctamente vinculados.
- En la clase de base de datos II aprendimos sobre el uso de procedimientos almacenados, las funciones y triggers para la mejor de una base de datos.
- En la clase de Base de Datos 2, hemos adquirido mucha información avanzada que van más allá de simplemente almacenar información también aprendimos a optimizar consultas, utilizar mejor el MySQL.
- Finalmente, nos educamos sobre cómo diseñar bases de datos que sean fáciles de mantener a largo plazo. Aprender a prevenir los posibles problemas de crecimiento y las soluciones necesarias es crucial para el desarrollo de aplicaciones que manejan grandes volúmenes de datos.