

A dark blue vertical bar runs down the left side of the page. A blue arrow points to the right from this bar, containing the date.

21-8-2023

CODERHOUSE

CURSO SQL

Several thin, curved lines in dark blue and light grey originate from the bottom left and sweep upwards and to the right.

Cesar Gaston Petit Martin

CADENA DE GIMNASIOS
ENTREGA FINAL

Contenido

Definición	2
Objetivo	2
Necesidades	2
Diagrama Entidad-Relación (conceptual)	3
Diagrama Entidad Relación (DER).....	4
.....	4
VISTAS GENERADAS	5
VISTA 1	5
VISTA 2	7
VISTA 3	9
VISTA 4	11
VISTA 5	12
VISTA 6	13
VISTA 7	14
STORES PROCEDURES.....	15
STORES PROCEDURE 1	15
STORES PROCEDURE 2	16
STORES PROCEDURE 3	17
STORES PROCEDURE 4	18
STORES PROCEDURE 5	20
STORES PROCEDURE 6	22
STORES PROCEDURE 7	22
STORES PROCEDURE 8	23
FUNCIONES.....	24
FUNCION 1	24
FUNCION 2	25
FUNCION 3	26
FUNCION 4	27
FUNCION 5	27
FUNCION 6	28
FUNCION 7	30
TRIGGER	31
TRIGGER 1.....	31
TRIGGER 2.....	31
TECNOLOGIAS UTILIZADAS.....	32

Definición

El modelo que se va a utilizar es el de una cadena de gimnasios fitness, conformado por diversas sucursales, y que a su vez cuenta muchas actividades, en los distintos turnos.

Objetivo

Crear un sistema de Base De Datos, que no permita llevar el correcto funcionamiento de la cadena de gimnasios y permitir un control de todos los aspectos correspondientes al modelo elegido, a su vez poder llevar una de control y seguimiento.

Necesidades

Podes administrar todas las cadenas de gimnasios, poder llevar un correcto control contables de los clientes y poder controlar la evolución de los empleados que trabajan.

Diagrama Entidad-Relación (conceptual)

Cadena de Gimnasios Cesar Petit

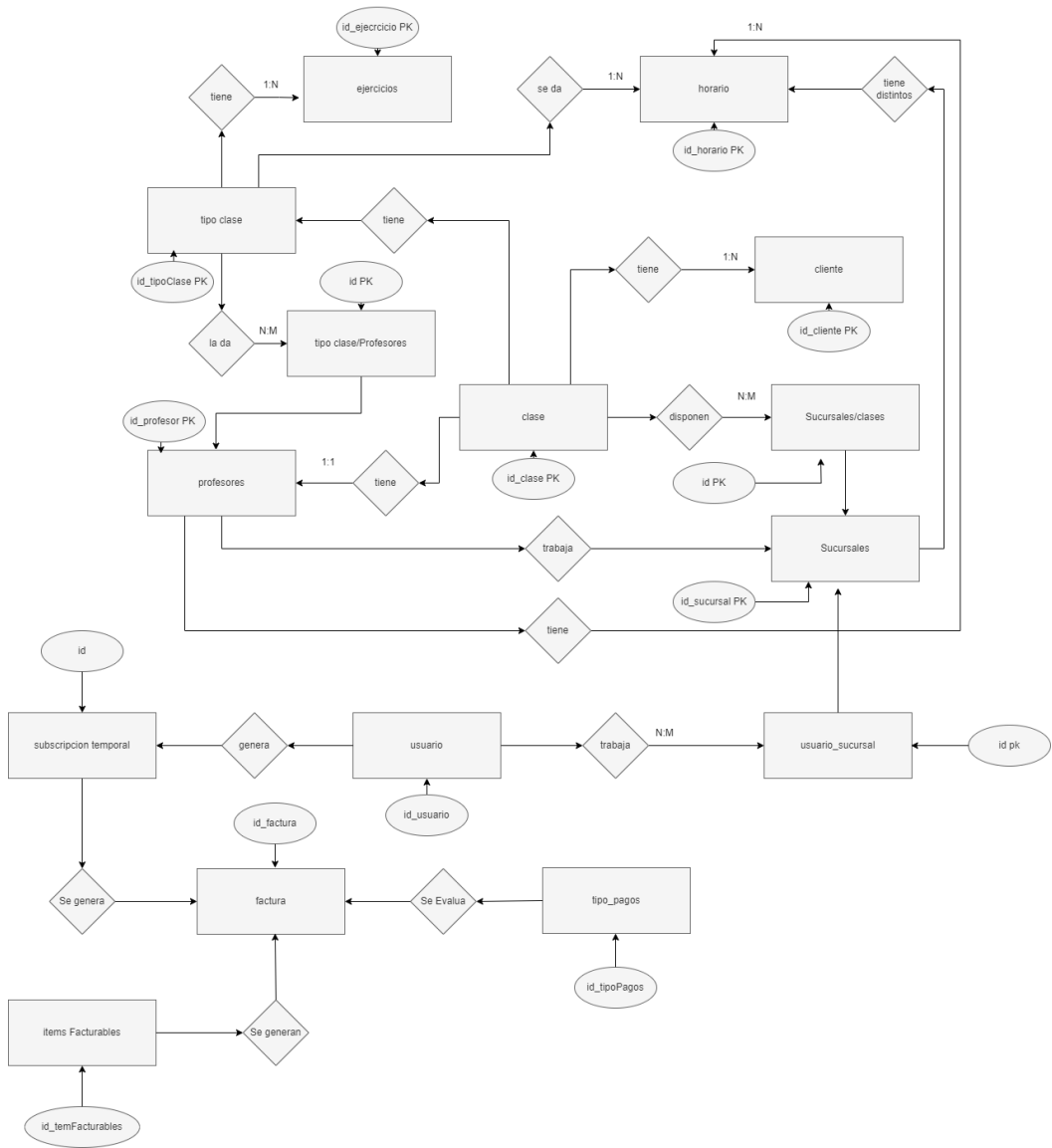
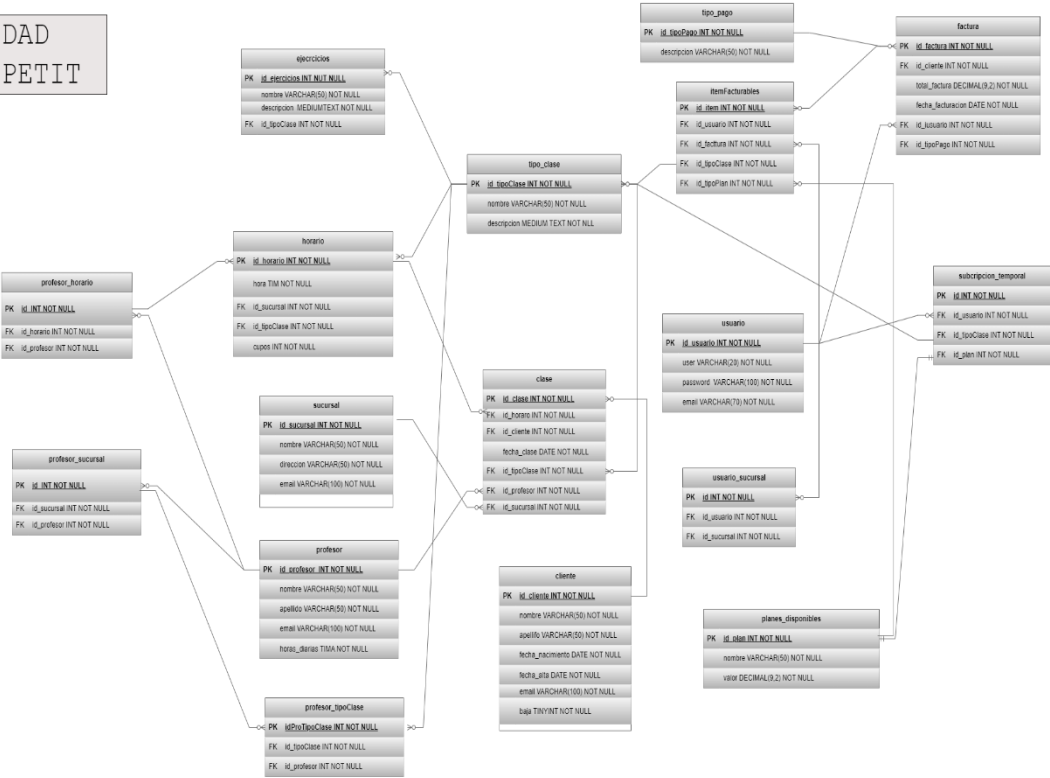


Diagrama Entidad Relación (DER)

DIAGRAMA ENTIDAD RELACION CESAR PETIT

ing_detalle
PK id_detalle INT NOT NULL
id_detalle INT NOT NULL
id_detalle INT NOT NULL
id_detalle INT NOT NULL
id_detalle INT NOT NULL
id_detalle INT NOT NULL
id_detalle INT NOT NULL
id_detalle INT NOT NULL

ing_detalle
PK id_detalle INT NOT NULL
id_detalle INT NOT NULL
id_detalle INT NOT NULL
id_detalle INT NOT NULL



VISTAS GENERADAS

VISTA 1

NOMBRE: v_clases_info

DEFINICION: ESTA VISTA MUESTRA LA INFORMACIÓN DE LAS CLASES.

ESQUEMA:

CREATE

ALGORITHM = UNDEFINED

DEFINER = `root`@`localhost`

SQL SECURITY DEFINER

VIEW `workshop_cesarpetit`.v_clases_info `v` AS

SELECT

`c`.`id_clase` AS `id_clase`,

`h`.`hora` AS `hora`,

`c`.`fecha_clase` AS `fecha_clase`,

`s`.`nombre` AS `nombre_sucursal`,

`p`.`nombre` AS `nombre_profesor`,

`tc`.`nombre` AS `tipo_clase`

FROM

(((`workshop_cesarpetit`.`clase` `c`

JOIN `workshop_cesarpetit`.`horario` `h` ON ((`c`.`id_horario` = `h`.`id_horario`)))

JOIN `workshop_cesarpetit`.`sucursal` `s` ON ((`c`.`id_sucursal` = `s`.`id_sucursal`)))

```
JOIN `workshop_cesarpetit`.`profesor` `p` ON ((`c`.`id_profesor` = `p`.`id_profesor`)))
```

```
JOIN `workshop_cesarpetit`.`tipo_clase` `tc` ON ((`c`.`id_tipoClase` = `tc`.`id_tipoClase`)))
```

RESULTADO OBTENIDO:

	id_clase	hora	fecha_clase	nombre_sucursal	nombre_profesor	tipo_clase
▶	1	08:00:00	2023-02-07	ALMAGRO	Juan	CrossFit
	2	08:00:00	2023-02-07	ALMAGRO	Maria	CrossFit
	3	08:00:00	2023-02-07	BALVANERA	Laura	CrossFit
	4	08:00:00	2023-02-07	BARRACAS	Maria	Funcional
	5	08:00:00	2023-02-07	ALMAGRO	Juan	Funcional
	6	08:00:00	2023-02-07	BARRACAS	Maria	Spinning
	7	08:00:00	2023-02-08	ALMAGRO	Juan	CrossFit
	8	08:00:00	2023-02-08	ALMAGRO	Maria	CrossFit
	9	08:00:00	2023-02-08	BALVANERA	Laura	CrossFit
	10	08:00:00	2023-02-08	BARRACAS	Maria	Funcional
	11	08:00:00	2023-02-08	ALMAGRO	Juan	Funcional
	12	08:00:00	2023-02-08	BARRACAS	Maria	Spinning

VISTA 2

NOMBRE: v_clases_por_cliente :

DEFINICION: ESTA VISTA MUESTRA LAS CLASES QUE HICIERON LOS CLIENTES

ESQUEMA:

CREATE

ALGORITHM = UNDEFINED

DEFINER = `root`@`localhost`

SQL SECURITY DEFINER

VIEW `workshop_cesarpetit`.`v_clases_por_cliente` AS

SELECT

`c`.`id_cliente` AS `id_cliente`,

CONCAT(`c`.`nombre`, ' ', `c`.`apellido`) AS `nombre_completo`,

COUNT(`cl`.`id_clase`) AS `total_clases`

FROM

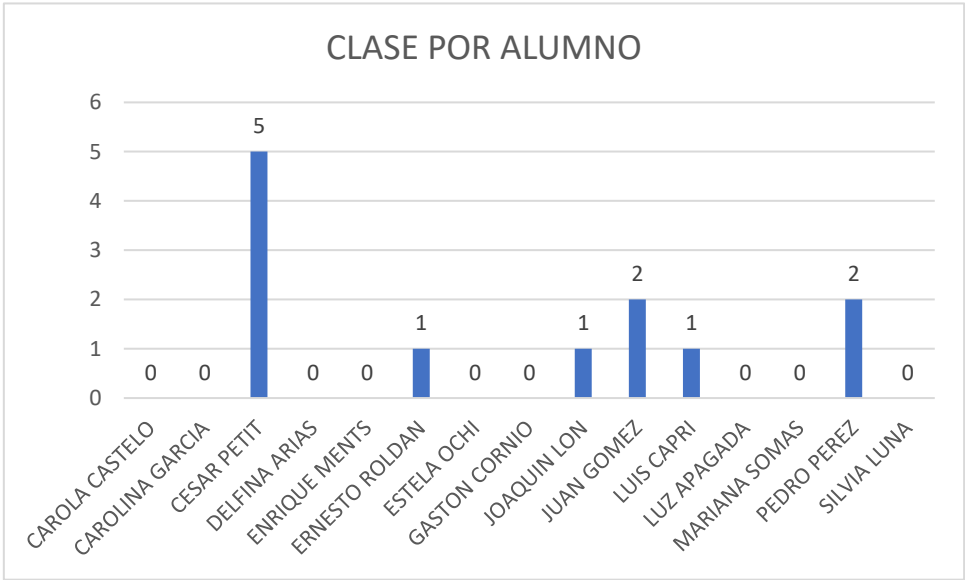
(`workshop_cesarpetit`.`cliente` `c`

LEFT JOIN `workshop_cesarpetit`.`clase` `cl` ON ((`c`.`id_cliente` = `cl`.`id_cliente`)))

GROUP BY `c`.`id_cliente`, `nombre_completo`

RESULTADO OBTENIDO:

	id_cliente	nombre_completo	total_clases
▶	11	CAROLA CASTELO	0
	13	CAROLINA GARCIA	0
	1	CESAR PETIT	5
	12	DELFINA ARIAS	0
	8	ENRIQUE MENTS	0
	3	ERNESTO ROLDAN	1
	10	ESTELA OCHI	0
	7	GASTON CORNIO	0
	5	JOAQUIN LON	1
	2	JUAN GOMEZ	2
	6	LUIS CAPRI	1
	15	LUZ APAGADA	0
	9	MARIANA SOMAS	0
	4	PEDRO PEREZ	2
	14	SILVIA LUNA	0



VISTA 3

NOMBRE: v_cupos_horario_sucursal:

DEFINICION: ESTA VISTA MUESTRA LOS CUPOS LIBRES POR CLASE

ESQUEMA:

CREATE

ALGORITHM = UNDEFINED

DEFINER = `root`@`localhost`

SQL SECURITY DEFINER

VIEW `workshop_cesarpetit`.`v_cupos_horario_sucursal` AS

SELECT

`h`.`id_horario` AS `id_horario`,

`h`.`hora` AS `hora`,

`s`.`nombre` AS `nombre_sucursal`,

`h`.`cupos` AS `total_cupos`,

(`h`.`cupos` - COUNT(`cl`.`id_clase`)) AS `cupos_disponibles`

FROM

((`workshop_cesarpetit`.`horario` `h`

JOIN `workshop_cesarpetit`.`sucursal` `s` ON ((`h`.`id_sucursal` = `s`.`id_sucursal`)))

LEFT JOIN `workshop_cesarpetit`.`clase` `cl` ON ((`h`.`id_horario` = `cl`.`id_horario`)))

GROUP BY `h`.`id_horario`, `h`.`hora`, `s`.`nombre`, `h`.`cupos`

RESULTADO OBTENIDO:

	id_horario	hora	nombre_sucursal	total_cupos	cupos_disponibles
▶	1	08:00:00	ALMAGRO	20	14
	2	08:00:00	ALMAGRO	20	16
	3	08:00:00	BALVANERA	10	8
	4	08:00:00	BALVANERA	10	10
	5	09:00:00	BARRACAS	20	20
	6	10:00:00	RECOLETA	15	15
	7	11:00:00	COLEGIALES	20	20
	8	12:00:00	VILLA ORTUZAR	20	20
	9	13:00:00	RECOLETA	15	15
	10	14:00:00	CABALLITO	15	15
	11	15:00:00	BARRACAS	10	10
	12	16:00:00	BALVANERA	10	10
	13	17:00:00	ALMAGRO	20	20
	14	18:00:00	BARRACAS	20	20
	15	19:00:00	PARQUE CHAC...	20	20

15

VISTA 4

NOMBRE: vistacanttipoclase:

DEFINICION: ESTA VISTA MUESTRA LA CANTIDAD DE CLASES QUE TIENE EL PROFESOR

ESQUEMA:

CREATE

ALGORITHM = UNDEFINED

DEFINER = `root`@`localhost`

SQL SECURITY DEFINER

VIEW `workshop_cesarpetit`.`vistacanttipoclase` AS

SELECT

CONCAT(`p`.`nombre`, ' ', `p`.`apellido`) AS `nombre Profesor`,

`tc`.`nombre` AS `nombre`,

COUNT(0) AS `Cantidad Clase`

FROM

((`workshop_cesarpetit`.`clase` `c`

JOIN `workshop_cesarpetit`.`profesor` `p` ON ((`p`.`id_profesor` = `c`.`id_profesor`)))

JOIN `workshop_cesarpetit`.`tipo_clase` `tc` ON ((`tc`.`id_tipoClase` = `c`.`id_tipoClase`)))

WHERE

(`p`.`id_profesor` = 1)

GROUP BY `tc`.`id_tipoClase`

RESULTADO OBTENIDO:

	nombre Profesor	nombre	Cantidad Clase
▶	Juan Perez	CrossFit	2
	Juan Perez	Funcional	2

VISTA 5

NOMBRE: facturacion_total:

DEFINICION: ESTA VISTA MUESTRA LAS FACTURAS TOTALES CREADAS

ESQUEMA:

CREATE

ALGORITHM = UNDEFINED

DEFINER = `root`@`localhost`

SQL SECURITY DEFINER

VIEW `facturacion_total` AS

SELECT

```
`fc`.`id_factura` AS `id_factura`,
`fc`.`subtotal` AS `subtotal`,
`fc`.`fecha_facturacion` AS `fecha_facturacion`,
`fc`.`total_factura` AS `total_factura`,
COUNT(`ifs`.`id_tipoClase`) AS `Cantidad_Items`,
CONCAT(`c`.`nombre`, ' ', `c`.`apellido`) AS `Cliente`
```

FROM

```
(((`factura` `fc`
JOIN `itemfacturables` `ifs` ON ((`ifs`.`id_factura` = `fc`.`id_factura`)))
JOIN `tipo_clase` `tc` ON ((`tc`.`id_tipoClase` = `ifs`.`id_tipoClase`)))
JOIN `cliente` `c` ON ((`c`.`id_cliente` = `fc`.`id_cliente`)))
JOIN `planes_disponibles` `pd` ON ((`pd`.`id_plan` = `ifs`.`id_tipoPlan`)))
```

GROUP BY `fc`.`id_factura`

RESULTADO OBTENIDO:

	id_factura	subtotal	fecha_facturacion	total_factura	Cantidad_Items	Cliente
▶	3	11300.00	2023-08-21 11:48:53	11622.05	3	CESAR PETIT
	4	7300.00	2023-08-21 11:49:59	8833.00	2	CESAR PETIT

VISTA 6

NOMBRE: facturacion_total_con_items:

DEFINICION: ESTA VISTA MUESTRA LAS FACTURAS TOTALES CREADAS AGREGANDOLE LAS CLASES SELECCIONADAS

ESQUEMA:

CREATE

ALGORITHM = UNDEFINED

DEFINER = `root`@`localhost`

SQL SECURITY DEFINER

VIEW `facturacion_total_con_items` AS

SELECT

```
`fc`.`id_factura` AS `id_factura`,
`fc`.`subtotal` AS `subtotal`,
`fc`.`fecha_facturacion` AS `fecha_facturacion`,
CONCAT(`c`.`nombre`, ' ', `c`.`apellido`) AS `CLIENTE`,
`tc`.`nombre` AS `NOMBRE_CLASE`,
`pd`.`nombre` AS `NOMBRE_PLAN`,
`fc`.`total_factura` AS `total_factura`
```

FROM

```
(((`factura` `fc`
JOIN `itemfacturables` `ifs` ON ((`ifs`.`id_factura` = `fc`.`id_factura`)))
JOIN `tipo_clase` `tc` ON ((`tc`.`id_tipoClase` = `ifs`.`id_tipoClase`)))
JOIN `cliente` `c` ON ((`c`.`id_cliente` = `fc`.`id_cliente`)))
JOIN `planes_disponibles` `pd` ON ((`pd`.`id_plan` = `ifs`.`id_tipoPlan`)))
```

ORDER BY `fc`.`id_factura`

VISTA 7

NOMBRE: conttipoclases:

DEFINICION: muestra la cantidad de tipos de clases que hay en la tabla clases

ESQUEMA:

CREATE

ALGORITHM = UNDEFINED

DEFINER = `root`@`localhost`

SQL SECURITY DEFINER

VIEW `conttipoclases` AS

SELECT

COUNT(`c`.`id_tipoClase`) AS `Cantidad Clases`,

`tc`.`nombre` AS `nombre`

FROM

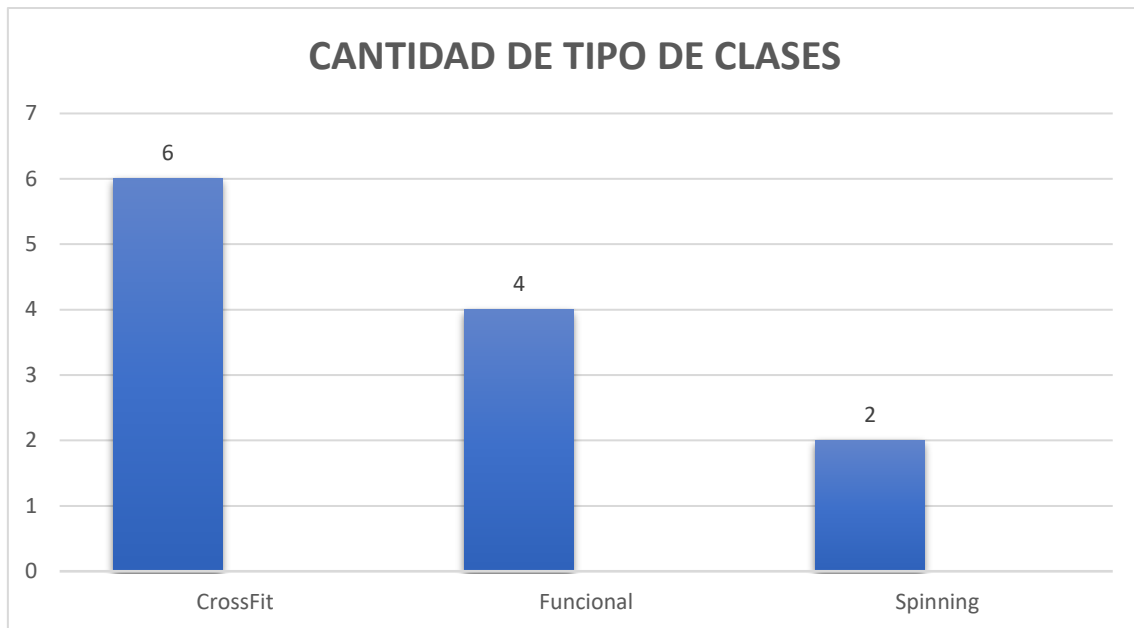
(`clase` `c`

JOIN `tipo_clase` `tc` ON ((`tc`.`id_tipoClase` = `c`.`id_tipoClase`)))

GROUP BY `c`.`id_tipoClase`

RESULTADO OBTENIDO:

	Cantidad Clases	nombre
▶	6	CrossFit
	4	Funcional
	2	Spinning



STORES PROCEDURES

STORES PROCEDURE 1

NOMBRE: sp_agregar_profesor:

PARAMETROS:

P_nombre: NOMBRE DEL PROFESOR

p_apellido : APELLIDO DEL PROFESOR

p_email: EMAIL DEL PROFESOR

p_horasDiarias: HORAS DEL PROFESOR

DEFINICION: AGREGA UN NUEVO PROFESOR AL STAFF

ESQUEMA:

```
CREATE DEFINER='root'@'localhost' PROCEDURE `sp_agregar_profesor`(in p_nombre  
varchar(50), p_apellido varchar(50), in p_email varchar(100), in p_horasDiarias time)
```

```
BEGIN
```

```
    INSERT INTO profesor(nombre,apellido,email,horas_diarias)
```

```
    values(p_nombre,p_apellido,p_email,p_horasDiarias);
```

```
END
```


STORES PROCEDURE 2

NOMBRE: sp_ordenar_clase

PARAMETROS:

Field: NOMBRE DE LA COLUMNA QUE QUIERO FILTRAR,

Orden: ELIGO EL ORDENAMIENTO ASC - DESC

DEFINICION: DEVUELVE LA CLASES

ESQUEMA:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_ordenar_clase`(IN field VARCHAR(20),  
IN orden VARCHAR(20) )
```

```
BEGIN
```

```
    IF field <> '' THEN
```

```
        SET @order_clase = concat('ORDER BY',' ',field ,' ', orden);
```

```
    ELSE
```

```
        set @order_clase = '';
```

```
    END IF;
```

```
    SET @clausula = concat('select * from clase ', @order_clase);
```

```
    PREPARE runSQL FROM @clausula;
```

```
    EXECUTE runSQL;
```

```
    DEALLOCATE PREPARE runSQL;
```

```
END
```

STORES PROCEDURE 3

NOMBRE: sp_cargarProfesor

PARAMETROS:

p_nombre: Nombre nuevo profesor

p_apellido: Apellido Nuevo Profesor

p_email: Email nuevo Profesor

p_horas: Carga de horas

DEFINICION: Carga nuevo Profesores.

ESQUEMA:

DELIMITER \$\$

CREATE PROCEDURE `sp_cargarProfesor`(

IN p_nombre VARCHAR(50),

IN p_apellido VARCHAR(50),

IN p_email VARCHAR(70),

IN p_horas TIME)

BEGIN

INSERT INTO profesor()

VALUES (null,p_nombre,p_apellido,p_email,p_horas);

END\$\$

DELIMITER ;

STORES PROCEDURE 4

NOMBRE: sp_generarFacturacion

PARAMETROS:

p_id_cliente: Nombre nuevo cliente

p_id_usuario: usuario generador factura

p_tipo_pago: que pago efectua

DEFINICION: Store que genera la facturación del periodo

ESQUEMA:

DELIMITER \$\$

CREATE DEFINER='root'@'localhost' PROCEDURE `sp_generarFacturacion` (IN p_id_cliente INT, IN p_id_usuario INT, IN p_tipo_pago INT)

BEGIN

DECLARE v_idFactura INT DEFAULT 0;

DECLARE subtotal, iva, total DECIMAL(11,2);

DECLARE rb BOOL DEFAULT FALSE;

DECLARE msg TEXT DEFAULT 'Error desconocido';

DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET rb := TRUE;

IF p_id_usuario <= 0 OR p_id_cliente <= 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE_TEXT = 'Todos los campos son requeridos';

ELSE

START TRANSACTION;

INSERT INTO factura

VALUES(null,p_id_cliente,NULL,NULL,NULL,NULL,p_id_usuario,NULL);

SET @v_idFactura = LAST_INSERT_ID();

```

____ IF @v_idFactura = 0 THEN
____
____ SET rb := TRUE;
____
____ SET msg := 'No se genero una nueva factura';
____
____ END IF;

____
____ INSERT INTO itemfacturables(id_usuario,id_factura,id_tipoClase,id_tipoPlan)
____
____ SELECT
____
____ id_usuario,
____
____ @v_idFactura,
____
____ id_tipoClase,
____
____ id_Plan
____
____ FROM subcripcion_temporal
____
____ WHERE id_usuario = p_id_usuario;

____
____ -- borro lo registros tabla temporal por usuario y inicializo el increment
____
____ DELETE FROM subcripcion_temporal WHERE id_usuario =
p_id_usuario;
____
____ ALTER TABLE subcripcion_temporal AUTO_INCREMENT = 1;

____
____ # CTE (Common Table Expression) que es una FUNCION VENTANA
____
____ WITH tabla_temporal_1 AS (
____
____ SELECT ifs.id_factura, sum(pd.valor) AS
importe_total FROM itemfacturables ifs
____
____ INNER JOIN planes_disponibles pd on
pd.id_plan = ifs.id_tipoPlan
____
____ WHERE id_factura = @v_idFactura
____
____ GROUP BY id_factura
____
____ )
____
____
____ SELECT importe_total
____
____ INTO @subtotal

```

```

FROM tabla temporal 1;

IF @subtotal = 0 THEN
    SET rb := TRUE;
    SET msg := 'El total es de 0 pesos';
END IF;

UPDATE factura
    SET subtotal = @subtotal, iva = calcular_iva(@subtotal), total_factura
    = calcular_importe(@subtotal,p_tipo_pago), id_tipoPago = p_tipo_pago ,fecha_facturacion =
    now()
    WHERE id_factura = @v_idFactura;

IF rb THEN
    ROLLBACK;
    SELECT CONCAT('Error: ', msg) AS 'Error';
ELSE
    COMMIT;
END IF;
END IF;

END$$
DELIMITER ;

```

STORES PROCEDURE 5

NOMBRE: sp_getFactura

PARAMETROS:

p_idFactura: numero de factura a consultar

DEFINICION: Muestra los datos de la factura consultada

ESQUEMA:

DELIMITER \$\$

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_getFactura`(IN p_idFactura INT)

BEGIN

DECLARE clases_factura varchar(50) DEFAULT NULL;

DECLARE planes_factura varchar(50) DEFAULT NULL;

SET @clases_factura = tipoClaseConcatenados(p_idFactura);

SET @planes_factura = tipoPlanesConcatenados(p_idFactura);

SELECT fc.id_factura,

CONCAT(c.nombre,' ',c.apellido) nombre_cliente,

fc.subtotal,fc.fecha_facturacion,

fc.total_factura,

count(ifs.id_tipoClase) AS cantidad_items,

@clases_factura AS clases_facturadas ,

@planes_factura AS planes_seleccionados

FROM workshop_cesarpetit.factura fc

INNER JOIN itemfacturables ifs ON ifs.id_factura = fc.id_factura

INNER JOIN tipo_clase tc ON tc.id_tipoClase = ifs.id_tipoClase

INNER JOIN cliente c ON c.id_cliente = fc.id_cliente

INNER JOIN planes_disponibles pd ON pd.id_plan = ifs.id_tipoPlan

WHERE fc.id_factura = p_idFactura

GROUP BY (fc.id_factura);

END\$\$

DELIMITER ;

STORES PROCEDURE 6

NOMBRE: sp_quitarProfesor:

PARAMETROS:

P_id: id del profesor que deseo dar de baja

DEFINICION: Quito Profesor del staff

ESQUEMA:

DELIMITER \$\$

CREATE DEFINER='root'@'localhost' PROCEDURE `sp_quitarProfesor` (IN p_id INT)

BEGIN

DELETE FROM profesor WHERE id_profesor = p_id;

END\$\$

DELIMITER;

STORES PROCEDURE 7

NOMBRE: sp_transaccion:

DEFINICION: Agrego o quito profesor evaluando si existen

ESQUEMA:

DELIMITER \$\$

CREATE DEFINER='root'@'localhost' PROCEDURE `sp_transaccion`()

BEGIN

DECLARE registros INT DEFAULT 0;

START TRANSACTION;

SET @registros = (SELECT count(*) FROM profesor);

```

        IF (@registros > 0) THEN
            CALL sp_quitarProfesor(1);
        ELSE
            CALL sp_cargarProfesor('Carlos','Sand','carlos@gmail.com','9:00');
        END IF ;

COMMIT;

-- ROLLBACK;

END$$
DELIMITER;

```

STORES PROCEDURE 8

NOMBRE: sp_ventaSubcripcion:

PARAMETROS:

p_idUser: id del usuario generador
p_id_tipoClase: Clase que deseo facturar
p_id_tipoPlan : plan seleccionado

DEFINICION: genero ítems temporales para aplicar en la facturación

ESQUEMA:

DELIMITER \$\$

```

CREATE PROCEDURE `sp_ventaSubcripcion`(  in p_idUser int,
                                           in p_id_tipoClase int,
                                           in p_id_tipoPlan int )

BEGIN

    DECLARE v_item INT DEFAULT NULL;

    IF p_idUser <= 0 OR p_id_tipoClase <= 0 OR p_id_tipoPlan <= 0 THEN

        SIGNAL SQLSTATE '45000'

        SET MESSAGE_TEXT = 'Todos los campos son requeridos';
    
```



```

ELSE

        -- VALIDO SI YA CARGUE EL TIPO DE CLASE, SI ES ASI ACTUALIZAO SOLO EL
TIPO DE PLAN

        SET v_item = (SELECT COUNT(*) FROM subcripcion_temporal WHERE
p_idUser = id_usuario AND id_tipoClase = p_id_tipoClase );

IF      v_item > 0 THEN

        UPDATE subcripcion_temporal SET id_plan = p_id_tipoPlan WHERE p_idUser =
id_usuario AND id_tipoClase = p_id_tipoClase;

ELSE

        insert into subcripcion_temporal(id_usuario,id_tipoClase,id_Plan)

        VALUES(p_idUser,p_id_tipoClase,p_id_tipoPlan);

        END IF;

    END IF;

END$$

DELIMITER;

```

FUNCIONES

FUNCION 1

NOMBRE: profesoresCantidadHoras

PARAMETROS :

p_horas: PARAMETRO QUE DEFINE LOS PROFESORES QUE TIENENE MAS DE ESAS HORAS.

DEFINICION: DEVUELVE TODOS LO PROFESORES QUE HACEN LAS MISMAS O MAYORES HORAS QUE EL PARAMETRO DE ENTRADA.

ESQUEMA:

```

CREATE DEFINER=`root`@`localhost` FUNCTION `profesoresCantidadHoras`( p_horas time)
RETURNS int

```

```

    READS SQL DATA

```

```

BEGIN

```

```

declare profesoresHoras int ;

set profesoresHoras = (select count(*) cantidadProfesores from profesor where
horas_diarias >=p_horas);

return profesoresHoras;

END

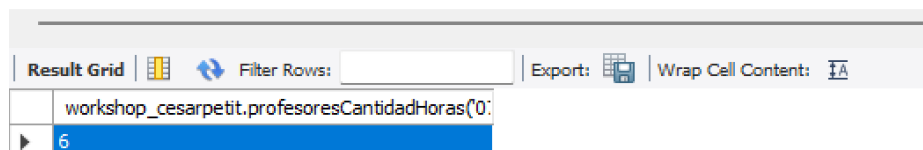
```

RESULTADO OBTENIDO:

```

1 • select workshop_cesarpetit.profesoresCantidadHoras('07:00');
2

```



	workshop_cesarpetit.profesoresCantidadHoras('07:00')
6	

FUNCION 2

NOMBRE: mayor_numero

PARAMETROS:

num1: NUMERO 1 PARA COMPARAR

num2: NUMERO 2 PARA COMPARAS

DEFINICION: FUNCION QUE DEVUELVE CUAL ES EL NUMERO MAS GRANDE EN CASO DE IGUALDAD DEVUELVE IGUALES

ESQUEMA:

delimiter \$\$

CREATE FUNCTION mayor_numero(num1 int,num2 int) RETURNS varchar(50)

NO SQL

BEGIN

declare mayor varchar(50);

if (num1 > num2) then

set mayor = concat('el mayor numero es:',',',CONVERT(num1, CHAR(20)));

end if;

if (num2 > num1) then

```

        set mayor = concat('el mayor numero es:',',',CONVERT(num2,
CHAR(20)));

    end if;

    if (num1 = num2) then

        set mayor = concat("Los numeros son iguales!!!!");

    end if;

    return mayor;

END$$

delimiter ;

```

FUNCION 3

NOMBRE: calcular_descuento

PARAMETROS:

Importe : importe a aplicar descuento

DEFINICION: Función que devuelve el 15 % de descuento al pagar en efectivo

ESQUEMA:

DELIMITER \$\$

```

CREATE DEFINER=`root`@`localhost` FUNCTION `calcular_descuento`(importe DECIMAL(11,2)
) RETURNS decimal(11,2)

```

NO SQL

BEGIN

```

    DECLARE resultado DECIMAL(9,2);

```

```

        DECLARE descuento DECIMAL(11,2) DEFAULT 15.00;

```

```

        -- *****si se modificar el descuento utilizar esta
linea*****

```

```

        -- set descuento = -- porcentaje que quiero descontar

```

```

        SET resultado = importe *(descuento/100);

```

```
    RETURN resultado;

END$$

DELIMITER ;
```

FUNCION 4

NOMBRE: calcular_iva

PARAMETROS:

Importe : importe a aplicar descuento

DEFINICION: Función que calculo el iva del importe ingresado

ESQUEMA:

DELIMITER \$\$

```
CREATE DEFINER=`root`@`localhost` FUNCTION `calcular_iva`(importe DECIMAL(11,2) )
RETURNS decimal(11,2)
```

```
NO SQL
```

```
BEGIN
```

```
    DECLARE impuestoActual DECIMAL(9,2) DEFAULT 21.00;
```

```
    DECLARE resultado DECIMAL(9,2);
```

```
    -- *****si se modificar el impuesto utilizar esta
linea*****
```

```
        -- set ImpuestoActual = -- porcentaje que quiero incrementar
```

```
    SET resultado = importe *(impuestoActual/100);
```

```
    RETURN resultado;
```

```
END$$
```

```
DELIMITER ;
```

FUNCION 5

NOMBRE: calcular_importe

PARAMETROS:

Importe : importe a aplicar descuento

Id_tipoPago : carga el tipo de pago

DEFINICION: Función que calculo el importe aplicando descuento e IVA

ESQUEMA:

DELIMITER \$\$

```
CREATE DEFINER=`root`@`localhost` FUNCTION `calcular_importe` (importe DECIMAL(11,2),  
id_tipoPago int) RETURNS decimal(11,2)
```

```
NO SQL
```

```
BEGIN
```

```
    DECLARE resultado DECIMAL(9,2) default 0.0;
```

```
    DECLARE importeInicial DECIMAL(9,2) default 0.0;
```

```
    # si tipo de pago es 1 Efectivo le aplico un descuento
```

```
    SET @importeInicial = importe + calcular_iva(importe);
```

```
    IF (id_tipoPago = 1) THEN
```

```
        SET resultado = @importeInicial - calcular_descuento(@importeInicial);
```

```
    ELSE
```

```
        SET resultado = @importeInicial;
```

```
    END IF;
```

```
    RETURN resultado;
```

```
END$$
```

```
DELIMITER;
```

FUNCION 6

NOMBRE: tipoClaseConcatenados

PARAMETROS:

p_idFactura: id de Factura

DEFINICION: Función que devuelve concatenado las clases a partir del nro. de facturación

ESQUEMA:

DELIMITER \$\$

**CREATE DEFINER=`root`@`localhost` FUNCTION `tipoClaseConcatenados` (p_idFactura int)
RETURNS varchar(100) CHARSET utf8mb4**

NO SQL

BEGIN

DECLARE tipoClasesConcat VARCHAR(100);

**SET tipoClasesConcat = (SELECT GROUP_CONCAT(upper(tc.nombre)
SEPARATOR ' - ')**

FROM

((('workshop_cesarpetit`.`factura` `fc`

JOIN

`workshop_cesarpetit`.`itemfacturables` `ifs` ON ((`ifs`.`id_factura` = `fc`.`id_factura`)))

JOIN

`workshop_cesarpetit`.`tipo_clase` `tc` ON ((`tc`.`id_tipoClase` = `ifs`.`id_tipoClase`)))

JOIN

`workshop_cesarpetit`.`cliente` `c` ON ((`c`.`id_cliente` = `fc`.`id_cliente`)))

JOIN

`workshop_cesarpetit`.`planes_disponibles` `pd` ON ((`pd`.`id_plan` = `ifs`.`id_tipoPlan`)))

WHERE fc.id_factura =

p_idFactura);

RETURN tipoClasesConcat;

END\$\$

DELIMITER;

FUNCION 7

NOMBRE: tipoPlanesConcatenados

PARAMETROS:

p_idFactura: id de Factura

DEFINICION: Función que devuelve concatenado los planes a partir del nro. de facturación

ESQUEMA:

DELIMITER \$\$

```
CREATE DEFINER=`root`@`localhost` FUNCTION `tipoPlanesConcatenados`(p_idFactura int)
RETURNS varchar(100) CHARSET utf8mb4
```

```
NO SQL
```

```
BEGIN
```

```
DECLARE tipoPlanConcat VARCHAR(100);
```

```
SET tipoPlanConcat = (SELECT GROUP_CONCAT(upper(pd.nombre) SEPARATOR ' - ')
```

```
FROM
```

```
        (((`workshop_cesarpetit`.`factura` `fc`
```

```
        JOIN
```

```
`workshop_cesarpetit`.`itemfacturables` `ifs` ON ((`ifs`.`id_factura` = `fc`.`id_factura`)))
```

```
        JOIN
```

```
`workshop_cesarpetit`.`tipo_clase` `tc` ON ((`tc`.`id_tipoClase` = `ifs`.`id_tipoClase`)))
```

```
        JOIN `workshop_cesarpetit`.`cliente` `c`
```

```
ON ((`c`.`id_cliente` = `fc`.`id_cliente`)))
```

```
        JOIN
```

```
`workshop_cesarpetit`.`planes_disponibles` `pd` ON ((`pd`.`id_plan` = `ifs`.`id_tipoPlan`)))
```

```
WHERE fc.id_factura = p_idFactura);
```

```
RETURN tipoPlanConcat
```

END\$\$

DELIMITER;

TRIGGER

TRIGGER 1

NOMBRE: tr_bajaUser

DEFINICION: SE CREO TRIGGER QUE CUANDO SE DA DE BAJA EL USUARIO DE MANERA LOGICA LO GUARDA EN EL LOG

TABLAS QUE IMPACTA: CLIENTE – LOG_CLIENTE

ESQUEMA:

delimiter \$\$

use workshop_CesarPetit \$\$

CREATE TRIGGER tr_bajaUser

AFTER UPDATE

ON cliente FOR EACH ROW

BEGIN

IF (OLD.baja <> NEW.baja AND NEW.BAJA = 1) THEN

INSERT INTO log_bajaCliente (id_cliente, fechaBaja)

VALUES (NEW.id_cliente, NOW());

END IF;

END \$\$

TRIGGER 2

NOMBRE: clase_BEFORE_DELETE

DEFINICION: SE CREO UN TRIGGER QUE AL ELIMINAR LA CLASE DEL CLIENTE LO GUARDA EN EL LOG

TABLAS QUE IMPACTA: CLASE – LOG_BAJACLASE

ESQUEMA:


```
use workshop_CesarPetit $$
```

```
CREATE TRIGGER `workshop_CesarPetit`.`clase_BEFORE_DELETE` BEFORE DELETE ON `clase`  
FOR EACH ROW
```

```
BEGIN
```

```
INSERT INTO log_bajaClase
```

```
VALUES(OLD.id_clase,OLD.id_horario,OLD.id_cliente,OLD.fecha_clase,OLD.id_tipoClase,OLD.i  
d_profesor,OLD.id_sucursal);
```

```
END$$
```

```
delimiter ;
```

TECNOLOGIAS UTILIZADAS

- Desarrollo del Documento: Microsoft Office 365
- Gráficos Estadístico: Microsoft Excel 365
- Diagrama Entidad Relación y Contexto: Draw.io
- Gestor BASE DE DATOS: MySQL WorkBench 8.0
- Motor de Base de Datos: MySQL 8.0