

A dark blue vertical bar on the left side of the page. A blue arrow points to the right from the 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

## 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

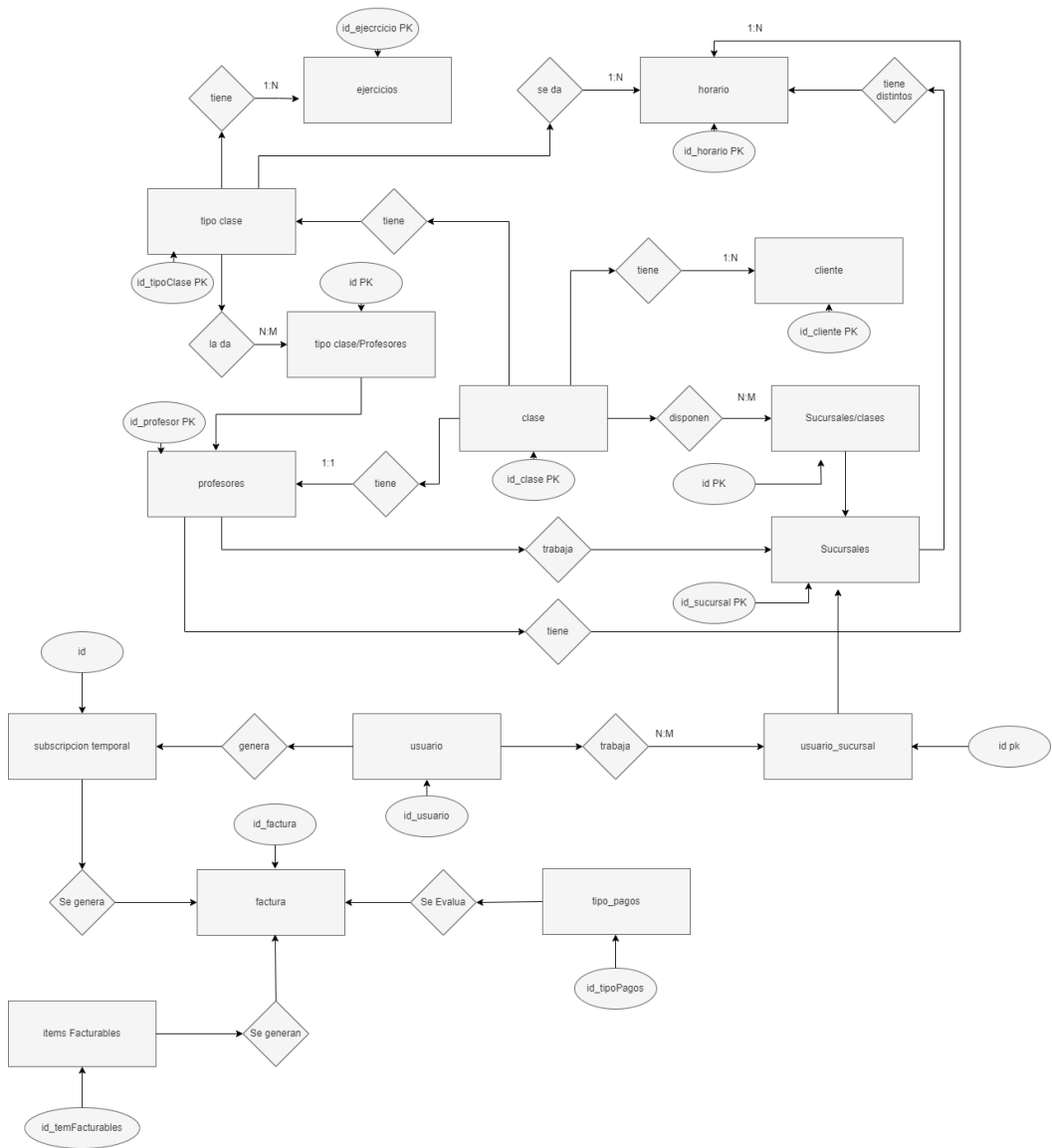
Crear un sistema de Base De Datos, que no permita llevar el correcto funcionamiento de la cadena de gimnasios y permitir un control de todos los aspectos correspondientes al modelo elegido, a su vez poder llevar una de control y seguimiento.

## 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

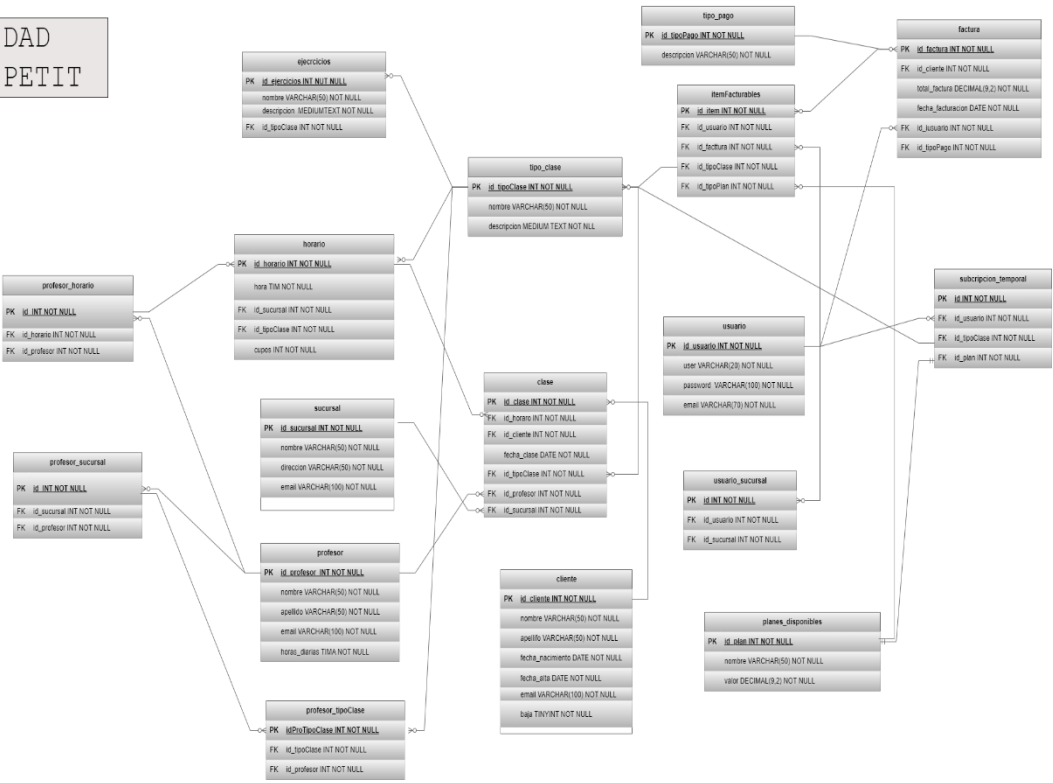


# Diagrama Entidad Relación (DER)

## DIAGRAMA ENTIDAD RELACION CESAR PETIT

log_tipoClase
PK id INT NOT NULL
id_clase INT NOT NULL
id_horario INT NOT NULL
id_cliente INT NOT NULL
fecha_clase DATE NOT NULL
id_tipoClase INT NOT NULL
id_profesor INT NOT NULL
id_secursal INT NOT NULL

log_tipoCliente
PK id INT NOT NULL
id_cliente INT NOT NULL
nombre 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`

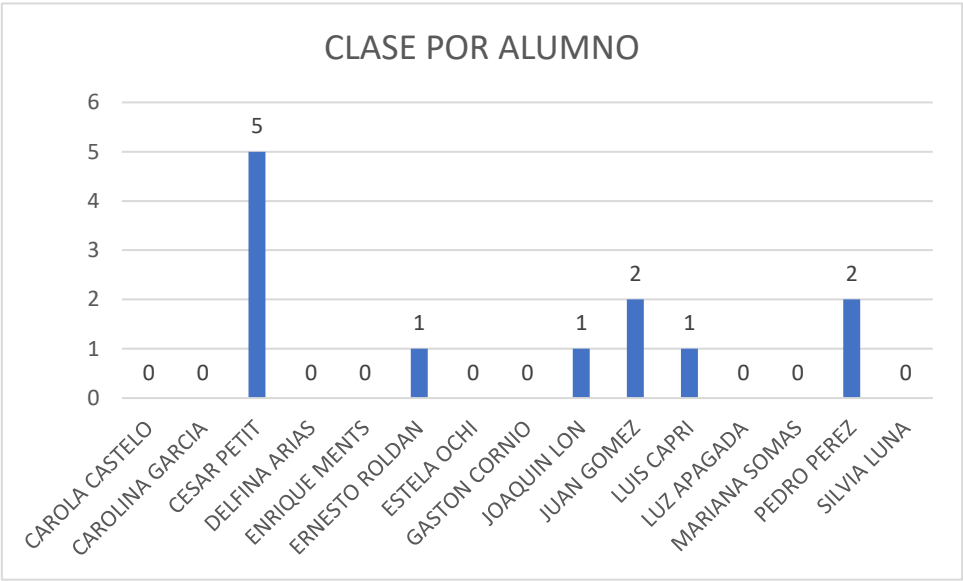
    LEFT JOIN `workshop\_cesarpetit`.`clase` `cl` ON ((`c`.`id\_cliente` = `cl`.`id\_cliente`)))

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

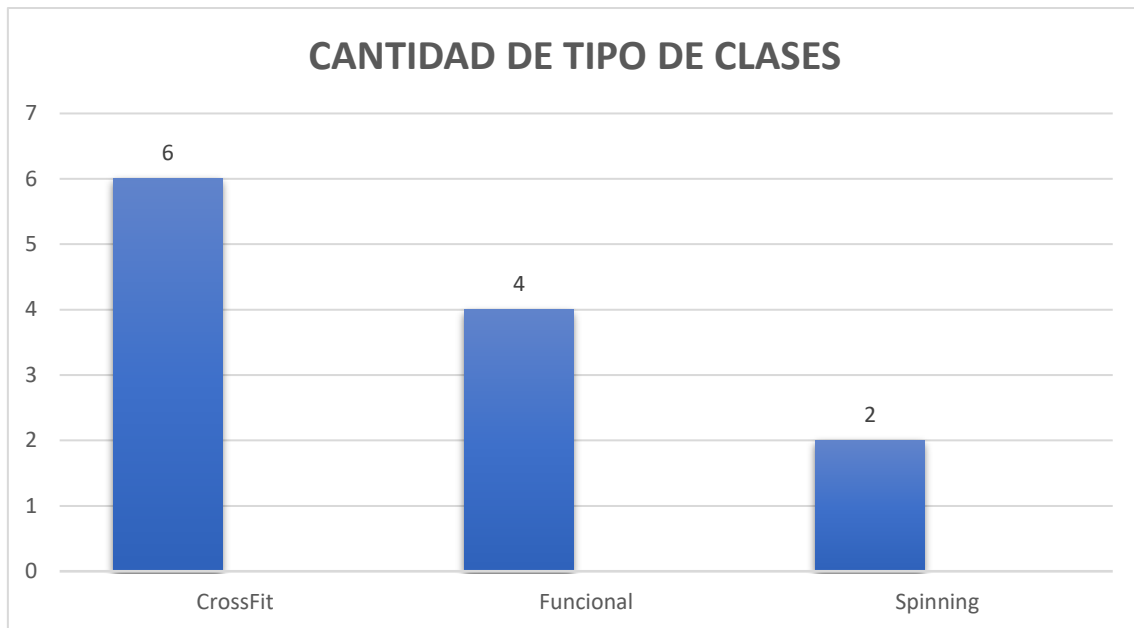
(`clase` `c`

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

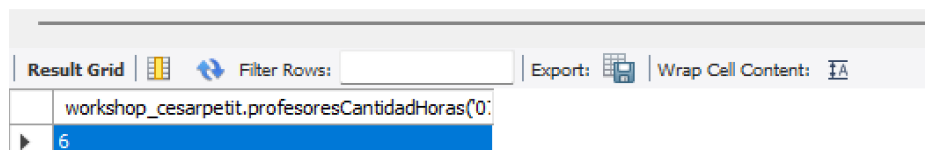
```

#### 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 ;
```