21-8-2023

Cesar Gaston Petit Martin

CADENA DE GIMNASIOS

ENTREGA FINAL

CODERHOUSE

CURSO SQL

Contenido

[Definición 2](#_Toc143625988)

[Objetivo 2](#_Toc143625989)

[Necesidades 2](#_Toc143625990)

[Diagrama Entidad-Relación (conceptual) 3](#_Toc143625991)

[Diagrama Entidad Relación (DER) 4](#_Toc143625992)

[4](#_Toc143625993)

[VISTAS GENERADAS 5](#_Toc143625994)

[VISTA 1 5](#_Toc143625995)

[VISTA 2 7](#_Toc143625996)

[VISTA 3 9](#_Toc143625997)

[VISTA 4 11](#_Toc143625998)

[VISTA 5 12](#_Toc143625999)

[VISTA 6 13](#_Toc143626000)

[VISTA 7 14](#_Toc143626001)

[STORES PROCEDURES 15](#_Toc143626002)

[STORES PROCEDURE 1 15](#_Toc143626003)

[STORES PROCEDURE 2 16](#_Toc143626004)

[STORES PROCEDURE 3 17](#_Toc143626005)

[STORES PROCEDURE 4 18](#_Toc143626006)

[STORES PROCEDURE 5 20](#_Toc143626007)

[STORES PROCEDURE 6 22](#_Toc143626008)

[STORES PROCEDURE 7 22](#_Toc143626009)

[STORES PROCEDURE 8 23](#_Toc143626010)

[FUNCIONES 24](#_Toc143626011)

[FUNCION 1 24](#_Toc143626012)

[FUNCION 2 25](#_Toc143626013)

[FUNCION 3 26](#_Toc143626014)

[FUNCION 4 27](#_Toc143626015)

[FUNCION 5 27](#_Toc143626016)

[FUNCION 6 28](#_Toc143626017)

[FUNCION 7 30](#_Toc143626018)

[TRIGGER 31](#_Toc143626019)

[TRIGGER 1 31](#_Toc143626020)

[TRIGGER 2 31](#_Toc143626021)

[TECNOLOGIAS UTILIZADAS32](#_Toc143626022)

# 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.

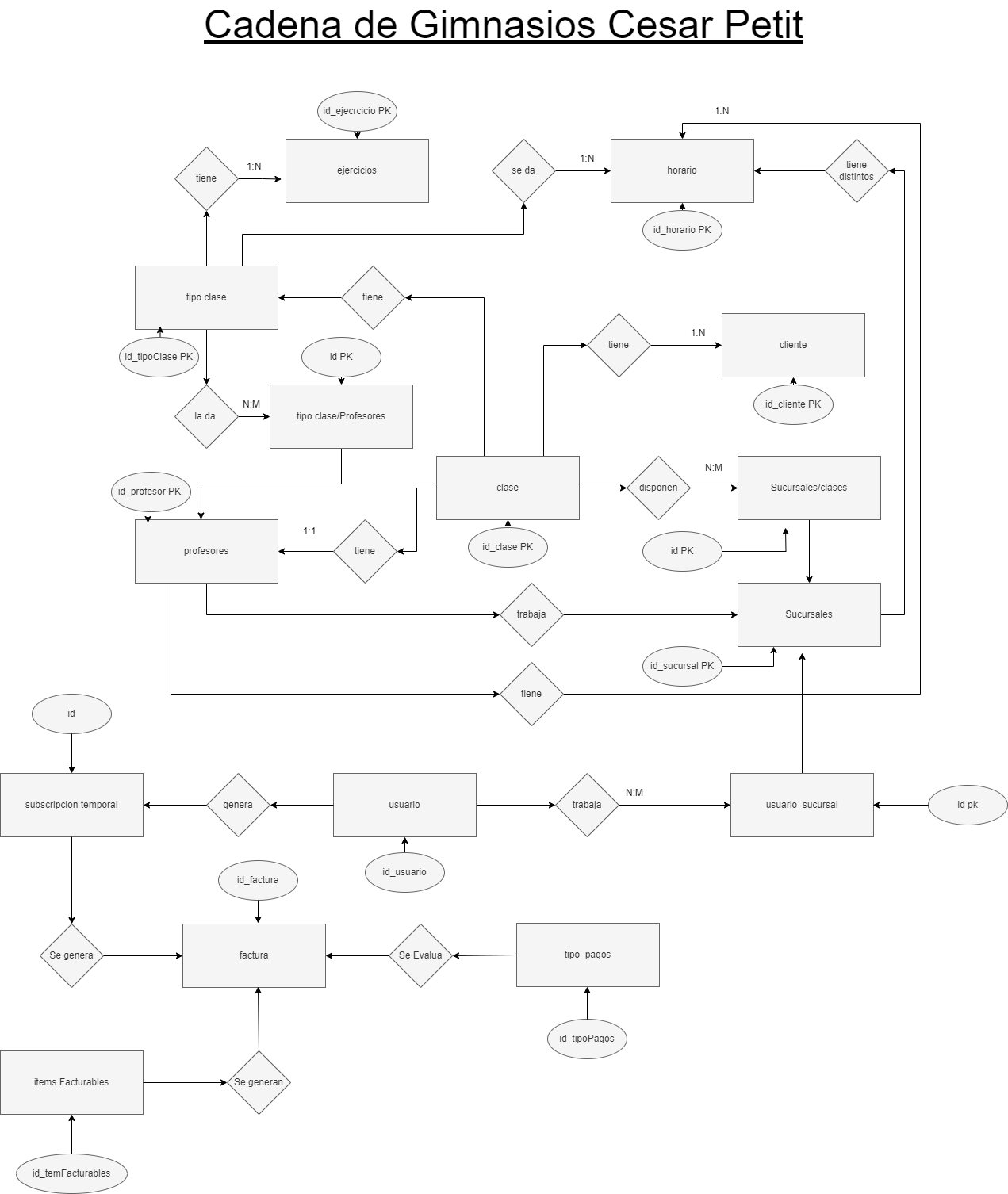
# Objetivo

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)



# Diagrama Entidad Relación (DER)

# 

# 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:**

Tabla

Descripción generada automáticamente

## 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:**

**Tabla

Descripción generada automáticamente**

## 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:**

**Tabla

Descripción generada automáticamente**

## 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:**

**Imagen que contiene Aplicación

Descripción generada automáticamente**

## 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`)))

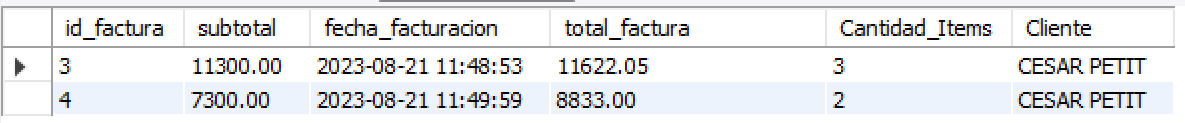
JOIN `tipo\_clase` `tc` ON ((`tc`.`id\_tipoClase` = `ifs`.`id\_tipoClase`)))

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:**



## 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:**

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

# 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:**

**Interfaz de usuario gráfica, Texto, Aplicación, Word

Descripción generada automáticamente**

## 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.id\_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