

CRUD 1: N



CONTENIDO

CRUD 1: N	1
1. Introducción.....	2
2. Base de datos.....	2
2.1. Tablas de la base de datos.....	2
2.1.2 Tabla clientes	2
2.1.2. Tabla facturas	3
2.2. Código SQL.....	3
3. Clases	4
3.1. Conexión.....	4
3.2. Clientes.....	5
Funciones de tipo CRUD:.....	5
Otras funciones	6
3.3. Facturas.....	10
Funciones de tipo CRUD.....	10
Otras Funciones.....	11

1. INTRODUCCIÓN

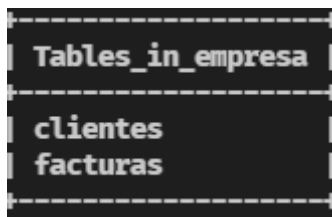
Vamos a diseñar una Web con dos apartados principales, uno para **Cientes** y otro para **Facturas**.

Para los clientes mostraremos su id, numero identificación fiscal (único), nombre (único), teléfono, código postal, provincia, población, país. Además, en este apartado podremos borrar clientes, editarlos, o añadir nuevos.

Para las facturas mostraremos un apartado a los detalles de dicha factura, el id que la identifica, la fecha de su creación, el importe neto, el IVA, impuestos (calculado en base al neto y porcentaje), el total (suma de neto + impuestos), el cliente al que está asociado, además de un enlace al documento de la factura (que podremos haber subido durante la creación). Además, al igual con los clientes podremos borrar, editar, y añadir nuevas facturas.

2. BASE DE DATOS

2.1. TABLAS DE LA BASE DE DATOS



Tables_in_empresa
clientes
facturas

2.1.2 TABLA CLIENTES

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
vat	varchar(12)	NO	UNI	NULL	
nombre	varchar(150)	NO	UNI	NULL	
telefono	int(9)	NO		NULL	
codigo_postal	int(5)	NO		NULL	
provincia	varchar(50)	YES		NULL	
poblacion	varchar(50)	YES		NULL	
pais	varchar(50)	NO		España	

2.1.2. TABLA FACTURAS

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
fecha	date	NO		NULL	
neto	decimal(10,2)	NO		NULL	
porcentaje	int(2)	NO		NULL	
impuestos	float(10,2)	NO		NULL	
total	float(10,2)	NO		NULL	
archivo	varchar(400)	YES		NULL	
id_cliente	int(11)	NO	MUL	NULL	

2.2. CÓDIGO SQL

► Run on active connection | ≡ Select block | ► Run SQL

```
CREATE database empresa ;
```

► Run SQL

```
USE empresa;
```

► Run SQL

```
CREATE TABLE clientes (  
  id INT AUTO_INCREMENT PRIMARY KEY,  
  vat VARCHAR(12) NOT NULL UNIQUE,  
  nombre VARCHAR(150) NOT NULL UNIQUE,  
  telefono INT(9) NOT NULL,  
  codigo_postal INT(5) NOT NULL,  
  provincia VARCHAR(50) DEFAULT NULL,  
  poblacion VARCHAR(50) DEFAULT NULL,  
  pais VARCHAR(50) DEFAULT 'España' NOT NULL  
);
```

► Run SQL

```
CREATE table facturas(  
  id INT AUTO_INCREMENT PRIMARY KEY,  
  fecha DATE NOT NULL,  
  neto DECIMAL(10,2) NOT NULL,  
  porcentaje INT(2) NOT NULL,  
  impuestos FLOAT(10,2) NOT NULL,  
  total FLOAT(10,2) NOT NULL,  
  archivo VARCHAR (400),  
  id_cliente INT NOT NULL,  
  CONSTRAINT fk_facturas FOREIGN KEY (id_cliente) references clientes(id) ON DELETE CASCADE ON UPDATE CASCADE  
);
```

3. CLASES

3.1. CONEXIÓN

Con esta clase creamos un objeto conexión, a partir del cuál podremos crear una conexión a la base de datos.

```
src > ConexionBD.php > ...
1  <?php
2  namespace Empresa;
3
4  use PDO;
5  use PDOException;
6
7  class ConexionBD{
8      protected static $conexion;
9
10     public function __construct(){
11         if(self::$conexion==null){
12             self::crearConexion();
13         }
14     }
15
16     static function cerrar(){
17         self::$conexion=null;
18     }
19
20
21     public static function crearConexion(){
22
23         $ruta_conf=dirname(__DIR__,1).'/configuracion.ini';
24         $configuracion=parse_ini_file($ruta_conf);
25         $usuario=$configuracion['usuario'];
26         $pass=$configuracion['pass'];
27         $servidor=$configuracion['servidor'];
28         $bd=$configuracion['bd'];
29         $dns = "mysql:host=$servidor;dbname=$bd;charset=utf8mb4";
30
31         try {
32             self::$conexion = new PDO($dns, $usuario, $pass);
33             self::$conexion->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
34         } catch (PDOException $ex) {
35             die("Error en la conexion!!! :". $ex->getMessage());
36         }
37     }
38 }
39
40
```

Depende de un archivo de configuración al cuál se le pasan los parámetros de conexión a la BD

“Configuración.ini”

```
estilos.css  configuracion.ini X
configuracion.ini
1  usuario=root
2  pass=
3  servidor=localhost
4  bd=empresa
```

3.2. CLIENTES

FUNCIONES DE TIPO CRUD:

3.2.1.1 Crear un elemento

```
public function create()
{
    if (isset($this->pais)) {
        $sql = "INSERT INTO clientes (
            vat,
            nombre,
            telefono,
            codigo_postal,
            provincia,
            poblacion,
            pais
        ) VALUES (:v,:n,:t,:c,:p,:pob,'$this->pais')";
    } else {
        $sql = "INSERT INTO clientes (vat, nombre, telefono,codigo_postal,provincia,poblacion) VALUES (:v,:n,:t,:c,:p,:pob)";
    }

    $statement = parent::$conexion->prepare($sql);
    echo $sql."<br>";
    try {
        $statement->execute([
            ':v' => $this->vat,
            ':n' => $this->nombre,
            ':t' => $this->telefono,
            ':c' => $this->codigo_postal,
            ':p' => $this->provincia,
            ':pob' => $this->poblacion
        ]);
    } catch (PDOException $e) {
        die($e->getMessage());
    }
    parent::$conexion = null;
}
```

3.2.1.2. Actualizar elemento

```
public function update()
{
    $sql = "UPDATE clientes SET
        vat=:v,
        nombre=:n,
        telefono=:t,
        codigo_postal=:c,
        provincia=:p,
        poblacion=:pob,
        pais=:pa
    WHERE id = '$this->id'";
    $statement = parent::$conexion->prepare($sql);
    try {
        $statement->execute([
            ':v' => $this->vat,
            ':n' => $this->nombre,
            ':t' => $this->telefono,
            ':c' => $this->codigo_postal,
            ':p' => $this->provincia,
            ':pob' => $this->poblacion,
            ':pa' => $this->pais
        ]);
    } catch (PDOException $e) {
        die($e->getMessage());
    }
    parent::$conexion = null;
}
```

3.2.1.3. Borrar elemento

```
public function delete($id)
{
    $sql = "DELETE FROM clientes WHERE id = '$id'";
    $statement = parent::$conexion->prepare($sql);
    try {
        $statement->execute();
    } catch (PDOException $e) {
        die($e->getMessage());
    }
    parent::$conexion = null;
}
```

3.2.1.4. Leer elementos

```
public function read()
{
    if (isset($this->id)) {
        $sql = "SELECT * FROM clientes WHERE id = '$this->id'";
    } else {
        $sql = "SELECT * FROM clientes";
    }
    $statement = parent::$conexion->prepare($sql);
    try {
        $statement->execute();
    } catch (PDOException $e) {
        die($e->getMessage());
    }
    parent::$conexion = null;
    return $statement;
}
```

OTRAS FUNCIONES

3.2.1.1. Función que devuelve solo la fila id

```
public function readID()
{
    $sql = "SELECT id FROM clientes";

    $statement = parent::$conexion->prepare($sql);
    try {
        $statement->execute();
    } catch (PDOException $e) {
        die($e->getMessage());
    }
    parent::$conexion = null;
    return $statement;
}
```

3.2.1.2. Función que devuelve id y nombre

```
public function devuelveClientes()
{
    $sql = "SELECT id,nombre FROM clientes";

    $statement = parent::$conexion->prepare($sql);
    try {
        $statement->execute();
    } catch (PDOException $e) {
        die($e->getMessage());
    }
    parent::$conexion = null;
    return $statement;
}
```

3.2.1.3. Función que comprueba si existe un id

```
public function existeId($id)
{
    $sql = "SELECT * FROM clientes WHERE id=$id";
    $statement = parent::$conexion->prepare($sql);

    try {
        $statement->execute();
    } catch (PDOException $e) {
        die($e->getMessage());
    }
    parent::$conexion = null;
    return ($statement->rowCount() != 0);
}
```

3.2.1.4. Función que genera Clientes

```
public function generarClientes($cant)
{
    if (!$this->hayClientes()) {
        $faker = Faker\Factory::create('es_ES');
        //Array Asociativo con algunas provincias Españolas y 3 municipios para cada provincia
        $localidades = array(
            "Alava" => array("Vitoria", "Llodio", "Salvatierra"),
            "Albacete" => array("Veste", "Hellin", "Almansa"),
            "Alicante" => array("Jijona", "Elche", "Orihuela"),
            "Almeria" => array("El Ejido", "Viator", "Huercal de Almeria"),
            "Asturias" => array("Gijon", "Oviedo", "Aviles"),
            "Avila" => array("El Miron", "Arevalo", "Arenas de San Pedro"),
            "Badajoz" => array("Don Benito", "Zafra", "Almendralejo"),
            "Barcelona" => array("Montcada I Reixac", "Badalona", "Parets del Valles"),
            "Burgos" => array("Aranda de Duero", "Lerma", "Miranda de Ebro"),
            "Cuenca" => array("Tarancon", "Iniesta", "Las Mesas"),
            "Gerona" => array("Vidreres", "Celra", "Caldes de Malavella"),
            "Granada" => array("Guadix", "Baza", "Fuente Vaqueros"),
            "Guadalajara" => array("Azuqueca de Henares", "Siguenza", "Cabanillas del Campo"),
            "Guipuzcoa" => array("San Sebastian", "Eibar", "Hondarribia"),
            "Huelva" => array("Matalascañas", "Almonte", "Cartaya"),
            "Huesca" => array("Alcala del Obispo", "Jaca", "Barbastro"),
            "Jaen" => array("Ubeda", "Linares", "Baeza")
        );
        //Array para generar una letra aleatoria en el numero de identificacion fiscal
        $abc = array("A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "N", "O", "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z");
        //Razones sociales para concatenar despues del texto generado por faker
        $razonessociales = array(" S.A.", " S.L.", " S.L.U.", " S.A.D ", " S. Coop.", " S.C.", " S.A.L.");

        for ($i = 0; $i < $cant; $i++) {
            $vat = $abc[array_rand($abc, 1)] . random_int(111111111, 999999999);
            //Concatenamos a la palabra generada por faker alguna de las razones sociales del array superior
            $nombre = ucfirst($faker->words(3, true)) . $razonessociales[array_rand($razonessociales, 1)];
            $telefono = random_int(111111111, 999999999);
            $codigo_postal = $faker->randomNumber(5, true);
            //Generamos un numero (0,1,2) para elegir un municipio de los 3 disponibles
            $municipio_aleatorio = $faker->numberBetween(0, 2);
            $provincia_aleatoria = array_rand($localidades, 1);
            //Cargamos una poblacion en base a la clave aleatoria (Provincia) y el 0,1 o 2 de la poblACION (POSICION EN EL ARRAY)
            $poblacion = $localidades[$provincia_aleatoria][$municipio_aleatorio];
            $provincia = $provincia_aleatoria;

            (new Clientes)
                ->setVat($vat)
                ->setNombre($nombre)
                ->setTelefono($telefono)
                ->setCodigo_postal($codigo_postal)
                ->setProvincia($provincia)
                ->setPoblacion($poblacion)
                ->create();
        }
    }
}
```


3.2.1.5. Función que comprueba si la BD está vacía

```
public function hayClientes()
{
    $sql = "SELECT * FROM clientes";

    $statement = parent::$conexion->prepare($sql);
    try {
        $statement->execute();
    } catch (PDOException $e) {
        die($e->getMessage());
    }
    parent::$conexion = null;
    return ($statement->rowCount() != 0);
}
```

3.2.1.6. Función que comprueba la unicidad de un campo

```
public function valorUnico($campo,$valor){
    $sql="SELECT * FROM clientes WHERE $campo='$valor'";
    $stmt=parent::$conexion->prepare($sql);

    try{
        $stmt->execute();
    }catch(PDOException $e){
        die("Error al comprobar unicidad ".$e->getMessage());
    }
    parent::$conexion = null;
    return ($stmt->rowCount()==0);
}
```

3.3. FACTURAS

FUNCIONES DE TIPO CRUD

3.3.1.1. Crear Factura

```
public function create()
{
    $sql = "INSERT INTO facturas (fecha,neto, porcentaje,impuestos,archivo,total,id_cliente)
    VALUES (NOW(),:n,:p,:i,:a,:t,:id)";
    $statement = parent::$conexion->prepare($sql);
    try {
        $statement->execute([
            ':n' => $this->neto,
            ':p' => $this->porcentaje,
            ':i' => $this->impuestos,
            ':t' => $this->total,
            ':id' => $this->id_cliente,
            ':a' => $this->archivo
        ]);
    } catch (PDOException $e) {
        die($e->getMessage());
    }
    parent::$conexion = null;
}
```

3.3.1.2. Actualizar Factura

```
public function update()
{
    $sql = "UPDATE facturas SET
    neto=:n,
    porcentaje=:p,
    impuestos=:i,
    total=:t,
    id_cliente=:d,
    archivo=:a
    WHERE id = $this->id";
    $statement = parent::$conexion->prepare($sql);
    try {
        $statement->execute([
            ':n' => $this->neto,
            ':p' => $this->porcentaje,
            ':i' => $this->impuestos,
            ':t' => $this->total,
            ':d' => $this->id_cliente,
            ':a' => $this->archivo
        ]);
    } catch (PDOException $e) {
        die($e->getMessage());
    }
    parent::$conexion = null;
}
```

3.3.1.3. Borrar Factura

```
public function delete($id)
{
    $sql = "DELETE FROM facturas WHERE id = '$id'";
    $statement = parent::$conexion->prepare($sql);
    try {
        $statement->execute();
    } catch (PDOException $e) {
        die($e->getMessage());
    }
    parent::$conexion = null;
}
```

3.3.1.4. Leer Factura

```
public function read()
{
    if (isset($this->id)) {
        $sql = "SELECT * FROM facturas WHERE id = '$this->id'";
    } else {
        $sql = "SELECT * FROM facturas";
    }
    $statement = parent::$conexion->prepare($sql);
    try {
        $statement->execute();
    } catch (PDOException $e) {
        die($e->getMessage());
    }
    parent::$conexion = null;
    return $statement;
}
```

OTRAS FUNCIONES

3.3.2.1. Función que comprueba si hay Facturas

```
public function hayFacturas()
{
    $sql = "SELECT * FROM facturas";

    $statement = parent::$conexion->prepare($sql);
    try {
        $statement->execute();
    } catch (PDOException $e) {
        die($e->getMessage());
    }
    parent::$conexion = null;
    return ($statement->rowCount() ≠ 0);
}
```

3.3.2.2. Función que comprueba si existe el ID de una Factura

```
public function existeId($id)
{
    $sql = "SELECT * FROM facturas WHERE id=$id";
    $statement = parent::$conexion->prepare($sql);

    try {
        $statement->execute();
    } catch (PDOException $e) {
        die($e->getMessage());
    }

    parent::$conexion = null;
    return ($statement->rowCount() > 0);
}
```

3.3.2.3. Función que para un ID, devuelve los detalles del mismo

```
public function detallefactura($id)
{
    $sql = "SELECT facturas.*, clientes.nombre, clientes.vat
    FROM facturas, clientes WHERE clientes.id=facturas.id_cliente AND facturas.id=$id";

    $statement = parent::$conexion->prepare($sql);

    try {
        $statement->execute();
    } catch (PDOException $e) {
        die($e->getMessage());
    }

    parent::$conexion = null;
    return $statement->fetch(PDO::FETCH_OBJ);
}
```

3.3.2.4. Función que muestra los clientes para un ID

```
public function filtrar($valor){
    $sql="SELECT * FROM facturas WHERE id_cliente=$valor";
    $statement = parent::$conexion->prepare($sql);

    try {
        $statement->execute();
    } catch (PDOException $e) {
        die($e->getMessage());
    }

    parent::$conexion = null;
    return $statement;
}
```

3.3.2.5. Función que genera Facturas

```
public function generarFacturas($cant)
{
    if (!$this->hayFacturas()) {
        $faker = Faker\Factory::create('es_ES');
        $clientes = (new Clientes)->readID();
        $array = array();
        while ($fila = $clientes->fetch(PDO::FETCH_OBJ)) {
            array_push($array, $fila->id);
        }

        for ($i = 0; $i < $cant; $i++) {
            $neto = $faker->randomFloat(2, 50, 99999);
            $porcentaje = $faker->randomElement($iva = array(4, 10, 21));
            $impuestos = $neto * ($porcentaje / 100);
            $total = $neto + $impuestos;
            $id_cliente = $array[array_rand($array, 1)];
            $archivo = "http://localhost/pdo/empresa/public/subidas/default.pdf";
            (new Facturas)
                ->setNeto($neto)
                ->setPorcentaje($porcentaje)
                ->setImpuestos($impuestos)
                ->setArchivo($archivo)
                ->setTotal($total)
                ->setId_cliente($id_cliente)
                ->create();
        }
    }
}
```