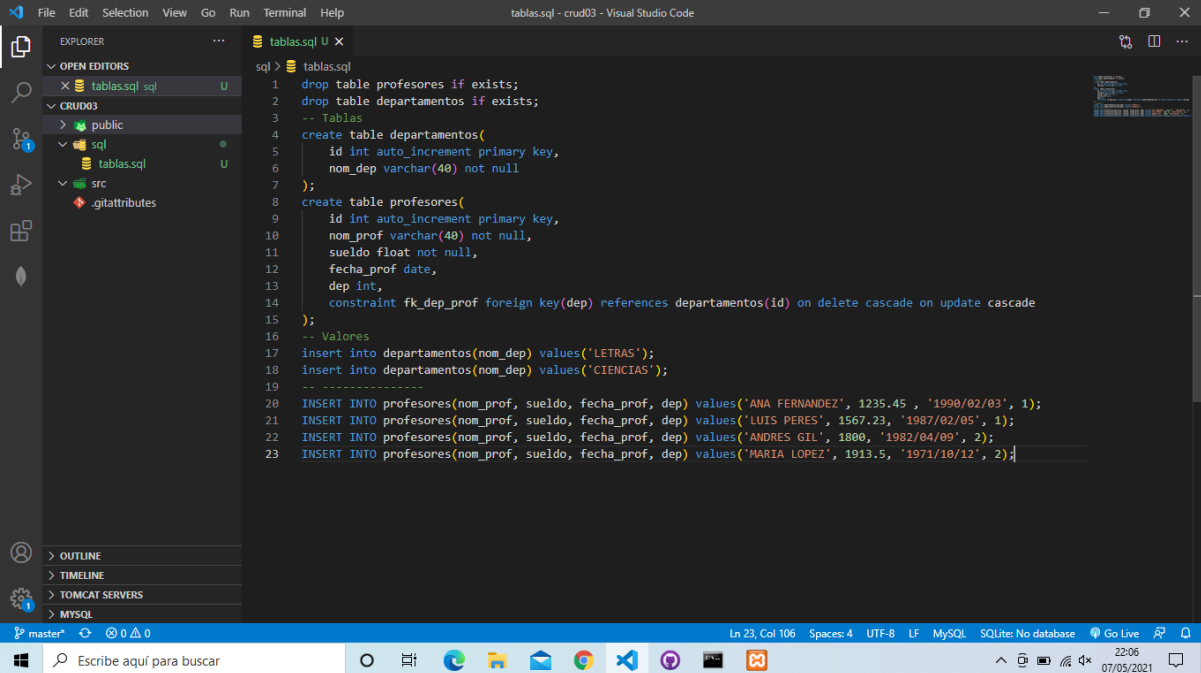


# 1.- Crear la estructura del proyecto

Crear una estructura de proyecto con las carpetas usuales (public, src y sql). Con dos tablas interrelacionadas en mysql esta vez



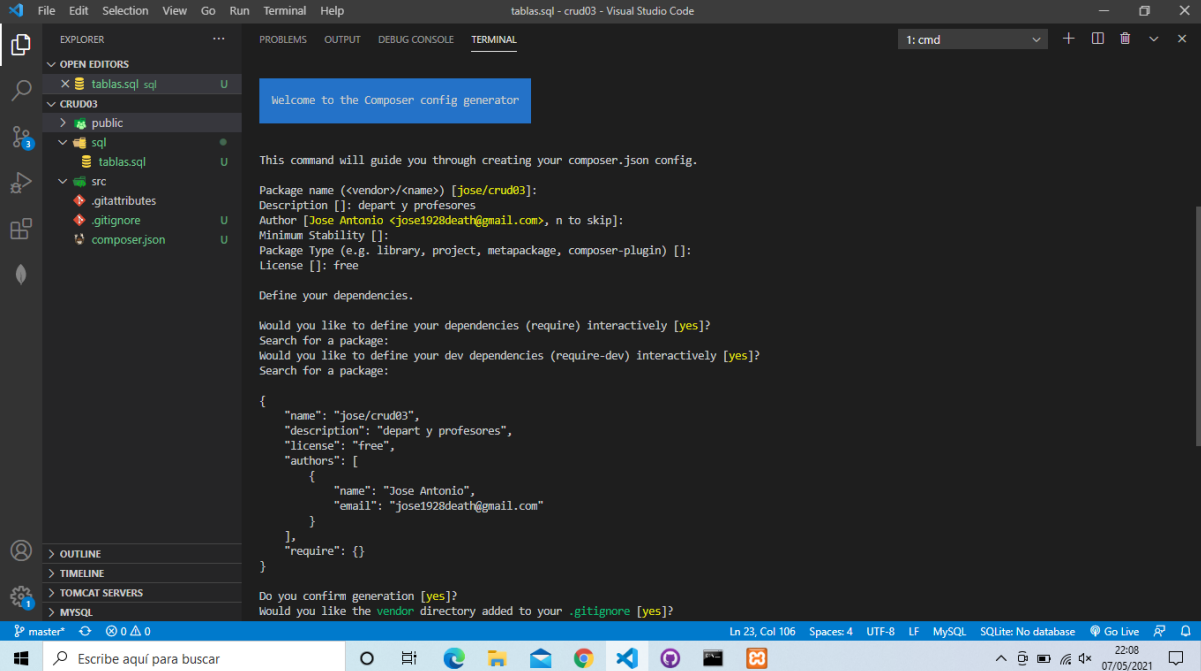
The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left. The project structure is as follows:

- public
  - sql
    - tablas.sql
- src
- .gitattributes

The main editor displays the content of `tablas.sql`:

```
sql > tablas.sql
1 drop table profesores if exists;
2 drop table departamentos if exists;
3 -- Tablas
4 create table departamentos(
5     id int auto_increment primary key,
6     nom_dep varchar(40) not null
7 );
8 create table profesores(
9     id int auto_increment primary key,
10    nom_prof varchar(40) not null,
11    sueldo float not null,
12    fecha_prof date,
13    dep int,
14    constraint fk_dep_prof foreign key(dep) references departamentos(id) on delete cascade on update cascade
15 );
16 -- Valores
17 insert into departamentos(nom_dep) values('LETRAS');
18 insert into departamentos(nom_dep) values('CIENCIAS');
19 -----
20 INSERT INTO profesores(nom_prof, sueldo, fecha_prof, dep) values('ANA FERNANDEZ', 1235.45, '1990/02/03', 1);
21 INSERT INTO profesores(nom_prof, sueldo, fecha_prof, dep) values('LUIS PERES', 1567.23, '1987/02/05', 1);
22 INSERT INTO profesores(nom_prof, sueldo, fecha_prof, dep) values('ANDRES GIL', 1800, '1982/04/09', 2);
23 INSERT INTO profesores(nom_prof, sueldo, fecha_prof, dep) values('MARIA LOPEZ', 1913.5, '1971/10/12', 2);
```

Ejecutamos el comando `composer init` y rellenamos los campos adecuados, también acepto la proposición de incluir `.gitignore` para no tener redundancia con vendor



The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left. The project structure is as follows:

- public
  - sql
    - tablas.sql
- src
- .gitattributes
- .gitignore
- composer.json

The main editor displays the output of the `composer init` command:

```
1: cmd
Welcome to the Composer config generator

This command will guide you through creating your composer.json config.

Package name (vendor/name) [jose/crud03]:
Description []: depart y profesores
Author [Jose Antonio <jose1928death@gmail.com>, n to skip]:
Minimum Stability []:
Package Type (e.g. library, project, metapackage, composer-plugin) []:
License []: free

Define your dependencies.

Would you like to define your dependencies (require) interactively [yes]?
Search for a package:
Would you like to define your dev dependencies (require-dev) interactively [yes]?
Search for a package:

{
    "name": "jose/crud03",
    "description": "depart y profesores",
    "license": "free",
    "authors": [
        {
            "name": "Jose Antonio",
            "email": "jose1928death@gmail.com"
        }
    ],
    "require": {}
}

Do you confirm generation [yes]?
Would you like the vendor directory added to your .gitignore [yes]?
```

En el archivo composer.json añadimos config y autoload, para que cargue los objetos (profesores, departamentos) y manipularlos en el public para que se manipulen en esta carpeta, es decir establecemos la comunicación del src a cualquier carpeta del proyecto. Con esto actualizamos con **composer update** para que nos añada el vendor y sus correspondientes librerías (require)

The screenshot shows the Visual Studio Code interface. The Explorer panel on the left shows the project structure with files like `tablas.sql`, `composer.json`, `public`, `src`, `vendor`, `.gitattributes`, `.gitignore`, `composer.lock`, and `composer.lock`. The main editor shows the `composer.json` file with the following content:

```

{
    "name": "jose/crud03",
    "description": "depart y profesores",
    "license": "free",
    "authors": [
        {
            "name": "Jose Antonio",
            "email": "jose1928death@gmail.com"
        }
    ],
    "config": {
        "optimize-autoloader": true
    },
    "autoload": {
        "psr-4": {
            "Classes\\": "src"
        }
    },
    "require": {}
}

```

The terminal at the bottom shows the output of the `composer update` command:

```

C:\Users\Jose\Documents\GitHub\crud03>composer update
Loading composer repositories with package information
Updating dependencies
Nothing to modify in lock file
Writing lock file
Installing dependencies from lock file (including require-dev)
Nothing to install, update or remove
Generating optimized autoload files

```

Accedemos a nuestro cmd o terminal desde linux y escribiremos los siguientes comandos para que la base de datos funcione.

Entrar en mysql

Crear usuario

**mysql -u root**

**create user user@localhost identified by secret0;**

**sudo mysql -u root**

Crear base de datos

Otorgar privilegios al usuario en la base de datos

**create database crud03;**

**grant all privileges on crud03.\* to user@localhost;**

Usar la base de datos

**use crud03**

The screenshot shows a terminal window with the following commands and output:

```

MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| crud01   |
| crud02   |
| information_schema |
| instituto |
| laravel |
| laravel |
| laravel |
| mysql    |
| performance_schema |
| phpmyadmin |
| test     |
+-----+
11 rows in set (0.064 sec)

MariaDB [(none)]> create database crud03;
Query OK, 1 row affected (0.003 sec)

MariaDB [(none)]> grant all privileges on crud03.* to user@localhost;
Query OK, 0 rows affected (0.013 sec)

MariaDB [(none)]> use crud03;
Database changed
MariaDB [crud03]> drop table profesores if exists;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'if exists' at line 1
MariaDB [crud03]> drop table departamentos if exists;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'if exists' at line 1
MariaDB [crud03]> -- Tables
MariaDB [crud03]> create table departamentos(
-> id int auto_increment primary key,
-> nom_dep varchar(40) not null
-> );
Query OK, 0 rows affected (0.061 sec)

MariaDB [crud03]> create table profesores(
-> id int auto_increment primary key,
-> nom_prof varchar(40) not null,
-> sueldo float not null,
-> fecha_prof date,

```

Insertamos la creación de las dos tablas con sus correspondientes insertos y relación de tablas.

```
Simbolo del sistema - mysql -u root
Database changed
MariaDB [crud03]> drop table profesores if exists;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'if exists'
at line 1
MariaDB [crud03]> drop table departamentos if exists;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'if exists'
at line 1
MariaDB [crud03]> -- Tablas
MariaDB [crud03]> create table departamentos(
-> id int auto increment primary key,
-> nom_dep varchar(40) not null
-> );
Query OK, 0 rows affected (0.061 sec)

MariaDB [crud03]> create table profesores(
-> id int auto increment primary key,
-> nom_prof varchar(40) not null,
-> sueldo float not null,
-> fecha_prof date,
-> dep int,
-> constraint fk_dep_prof foreign key(dep) references departamentos(id) on delete cascade on update cascade
-> );
Query OK, 0 rows affected (0.049 sec)

MariaDB [crud03]> -- Valores
MariaDB [crud03]> insert into departamentos(nom_dep) values('LETRAS');
Query OK, 1 row affected (0.096 sec)

MariaDB [crud03]> insert into departamentos(nom_dep) values('CIENCIAS');
Query OK, 1 row affected (0.003 sec)

MariaDB [crud03]> -- -----
MariaDB [crud03]> INSERT INTO profesores(nom_prof, sueldo, fecha_prof, dep) values('ANA FERNANDEZ', 1235.45 , '1998/02/03', 1);
Query OK, 1 row affected (0.005 sec)

MariaDB [crud03]> INSERT INTO profesores(nom_prof, sueldo, fecha_prof, dep) values('LUIS PERES', 1567.23, '1987/02/05', 1);
Query OK, 1 row affected (0.001 sec)

MariaDB [crud03]> INSERT INTO profesores(nom_prof, sueldo, fecha_prof, dep) values('ANDRES GIL', 1800, '1982/04/09', 2);
Query OK, 1 row affected (0.002 sec)

MariaDB [crud03]> INSERT INTO profesores(nom_prof, sueldo, fecha_prof, dep) values('MARIA LOPEZ', 1913.5, '1971/10/12', 2);
Query OK, 1 row affected (0.004 sec)
```

Creamos el archivo config para que estos atributos están bien guardados por seguridad y llamarlos en la clase Conexion para establecer la conexión entre mysql y el programa php.

```
File Edit Selection View Go Run Terminal Help
.config - crud03 - Visual Studio Code

EXPLORER
OPEN EDITORS
  tablas.sql sql U
  .config U
  composer.json U
CRUD03
  public
  sql
    tablas.sql U
  src
  vendor
  .config U
  .gitattributes
  .gitignore U
  composer.json U
  composer.lock U

OUTLINE
  TIMELINE
  TOMCAT SERVERS
  MYSQL

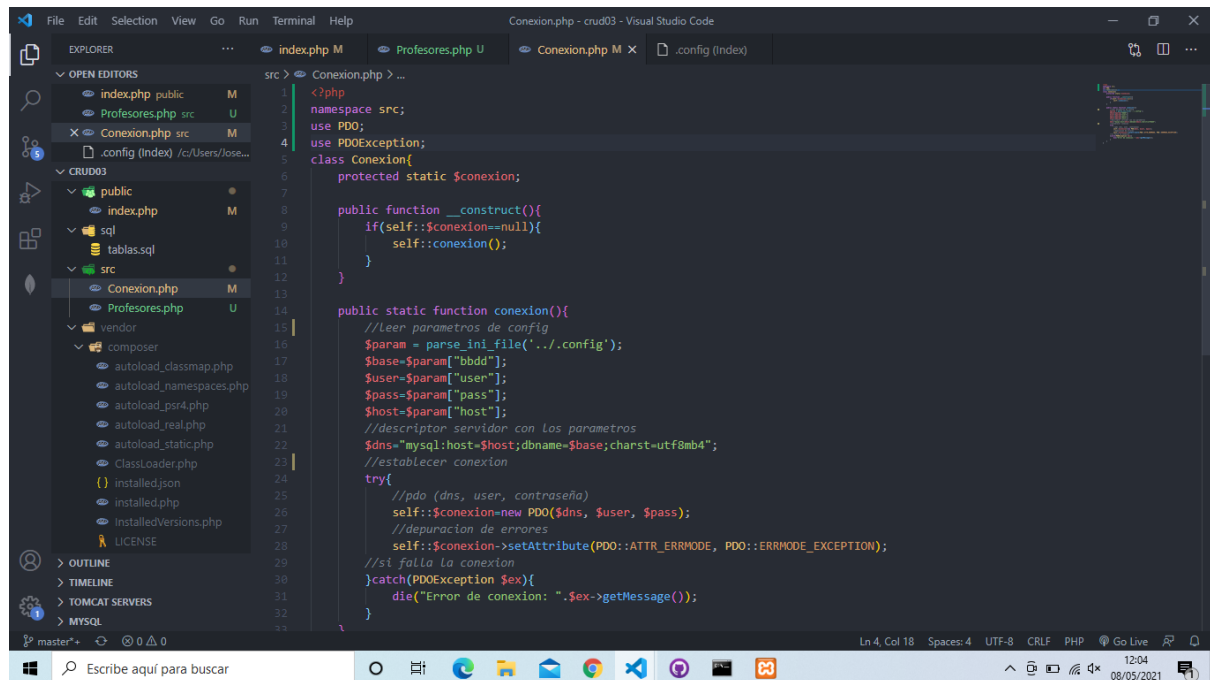
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: cmd
Writing lock file
Installing dependencies from lock file (including require-dev)
Nothing to install, update or remove
```

## 2.- Establecer conexión

Añadimos la clase conexion en src y establecemos los parámetros en sus correspondientes sitios.

En el primer constructor podemos observar que se ejecutará en tiempo de ejecución. Si vemos que no hay conexión intentamos ejecutar el método conexión de la misma clase.

En el método conexión llamamos los parámetros de .config y lo asignamos a las variables, entonces añadimos estos parámetros a la nomenclatura de dns y ejecutamos la conexión y si falla la conexión lanzará el error por defecto.



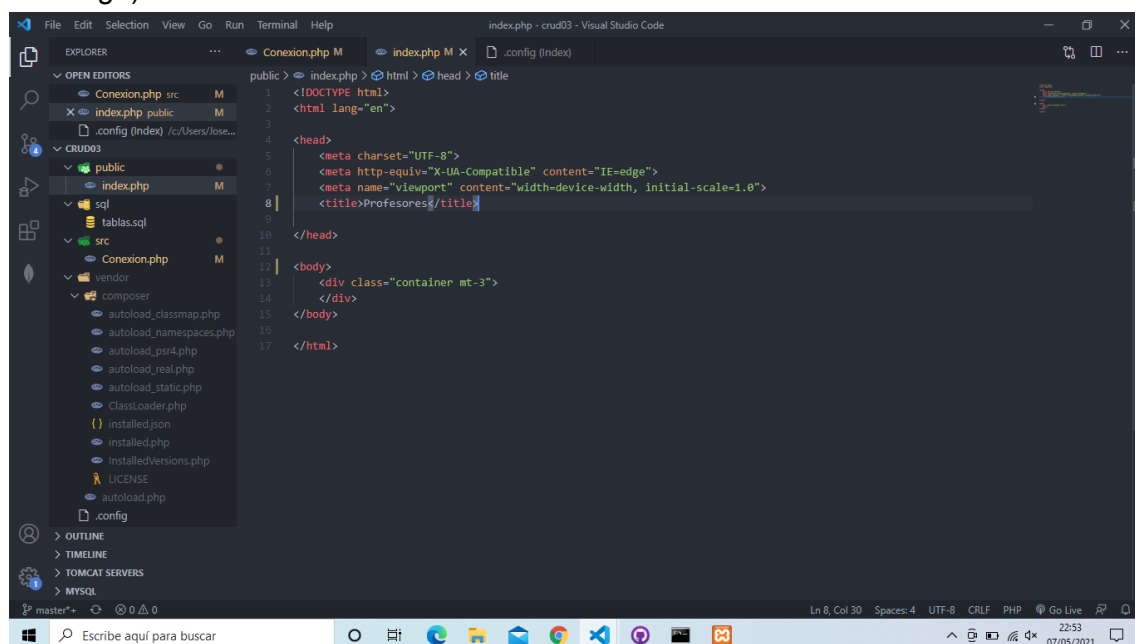
```
<?php
namespace src;
use PDO;
use PDOException;

class Conexion{
    protected static $conexion;

    public function __construct(){
        if(self::$conexion==null){
            self::conexion();
        }
    }

    public static function conexion(){
        //Leer parametros de config
        $param = parse_ini_file("../.config");
        $base=$param["bbdd"];
        $user=$param["user"];
        $pass=$param["pass"];
        $host=$param["host"];
        //descriptor servidor con los parametros
        $dns="mysql:host=$host;dbname=$base;charset=utf8mb4";
        //establecer conexion
        try{
            //pdo (dns, user, contraseña)
            self::$conexion=new PDO($dns, $user, $pass);
            //depuracion de errores
            self::$conexion->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        }catch(PDOException $ex){
            //si falla la conexion
            die("Error de conexion: ".$ex->getMessage());
        }
    }
}
```

Comprobamos que todo funcione y esté bien conectado con un index.php sencillo, si responde bien se visualizará algo (lo que haya escrito en el index.php responde conforme a su código) si no lanzará un error.



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Profesores</title>
</head>
<body>
    <div class="container mt-3">
    </div>
</body>
</html>
```

[Conexion.php](#)



### 3.- Objetos departamento y profesor

Añadimos los atributos correspondientes que nos dictan las tablas y creamos los atributos.

```
class Profesores extends Conexion
{
    private $id;
    private $nombre;
    private $sueldo;
    private $fecha;
    private $idDep;

    public function __construct()
    {
        parent::__construct();
    }
}
```

```
class Departamentos extends Conexion
{
    private $id;
    private $nom_dep;

    public function __construct()
    {
        parent::__construct();
    }
}
```

## 4.- Tabla profesor

### 4.1- Mostrar tabla

Creamos una tabla vacía cuyas filas son generadas a partir del método `mostrarProfesores()` de la clase `profesores`. Con un `while` que generará las filas necesarias hasta el final. Representará en cada fila las columnas siguientes (nombre del profesor, sueldo, fecha del añadido del tiempo real, el departamento que imparte y las acciones).

```
24 <body style="background-color: CadetBlue;">
25 <div class="container mt-3">
26 </div>
27 <a href="crearProfesor.php" class="btn btn-primary my-3">Añadir profesor</a>
28 <table class="table table-success table-striped">
29 <thead>
30 <tr>
31 <th scope="col">Nombre</th>
32 <th scope="col">Sueldo</th>
33 <th scope="col">Fecha</th>
34 <th scope="col">Departamento</th>
35 <th scope="col">Acciones</th>
36 </tr>
37 </thead>
38 <tbody>
39 <?php
40 //Repetir el proceso mientras haya profesores
41 while ($profesor = $profesorMostrar->fetch(PDO::FETCH_OBJ)) {
42     echo "<tr>";
43     echo "<td>{$profesor->nom_prof}</td>";
44     echo "<td>{$profesor->sueldo} €</td>";
45     echo "<td>{$profesor->fecha_prof}</td>";
46     echo "<td>{$profesor->nom_dep}</td>";
47     echo "<td>";
48     echo "<form name='borrar' method='POST' class='form-inline' action='borrarProfesor.php'>";
49     echo "<a href='editarProfesor.php?id={$profesor->id}' class='btn btn-primary m-auto'><i>Editar</i></a>";
50     echo "<input type='hidden' name='id' value='{$profesor->id}'>";
51     echo "<button type='submit' class='ml-2 btn btn-danger'>Borrar</button></form>";
52     echo "</td>\n";
53     echo "</tr>\n";
54 }
55 ?>
56 </tbody>
57 </table>
58 </table>
59 <div class="col-md-12 text-center">
60 <a href="index.php" class="btn btn-primary my-3">Volver</a>
61 </div>
62 </body>
```

Crear el objeto `Profesores` y llamar el método `mostrarProfesores()` para mostrar sus filas.

```
<?php
use Clases\Profesores;

require '../vendor/autoload.php';
$profesor = new Profesores();
$profesorMostrar = $profesor->mostrarProfesores();
$profesor = null;
?>
```

[profesores.php](#)

Este método lo usaremos para mostrar todas las filas de la tabla profesor, con la siguiente consulta que recogerá la información de la tabla profesores y departamentos con la clave foránea de dep procedente de profesores y con el id de departamentos, esta consulta sirve para que visualice el nombre del departamento en lugar del id del departamento. Preparamos la consulta y la ejecutamos, devolvemos el resultado para que muestre el contenido en profesores.php

```
public function mostrarProfesores()
{
    //muestro todos los profesores de la tabla
    $consulta = "select p.id, p.nom_prof, p.sueldo, p.fecha_prof, d.nom_dep
    from profesores p inner join departamentos d
    on p.dep = d.id";
    //preparar consulta
    $stmt = parent::$conexion->prepare($consulta);
    try {
        //ejecutar consulta
        $stmt->execute();
    } catch (PDOException $ex) {
        //error si el intento falla
        die("Error no carga tabla: " . $ex->getMessage());
    }
    return $stmt;
}
```

[mostrarProfesores\(\) line 115](#)

## 4.2- Añadir profesor

Realizamos la consulta para insertar una fila a la tabla profesores con parámetros excepto la fecha ya que el añadido de fechas se realiza a la hora del añadido y no una fecha inventada. Ejecutamos la consulta con los parámetros correspondientes con su catch.

```
public function create()
{
    //consulta de insercion de profesores
    $consulta = "insert into profesores(nom_prof, sueldo, fecha_prof, dep) values(:n, :s, now(), :d)";
    //preparar consulta con el codigo
    $stmt = parent::$conexion->prepare($consulta);
    try {
        //ejecutar consulta con parametros
        $stmt->execute([
            ':n' => $this->nombre,
            ':s' => $this->sueldo,
            ':d' => $this->idDep
        ]);
    } catch (PDOException $ex) {
        //error si el intento falla
        die("Error al insertar profesor: " . $ex->getMessage());
    }
}
```

[create\(\) Profesores.php](#)

Añadimos 2 campos (nombre y sueldo) con un select que muestra los departamentos (no sus ids, sus nombres). A partir del método campo nombre, sueldo y recoger las opciones y leerlas en el select.

```
28 <!DOCTYPE html>
29 <html lang="en">
30
31 <head>
32     <meta charset="UTF-8">
33     <meta http-equiv="X-UA-Compatible" content="IE=edge">
34     <meta name="viewport" content="width=device-width, initial-scale=1.0">
35     <title>Añadir profesor</title>
36     <!-- CSS only -->
37     <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-6V7wQGN663wczY63oAC1IHJ8ZtSzwYFxh3EO2upgRN59Zy3-wq/XB1eztX3os" crossorigin="anonymous">
38     <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@fortawesome/fontawesome-free@5.15.3/css/all.min.css" integrity="sha384-Si2vOq57LRHjW78F+V8I58t+XU6kY5f2PFDqYUwIyN/O6Iu3Q886YGvD066M4" crossorigin="anonymous">
39     <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.bundle.min.js" integrity="sha384-3p4gw2s1f6oWf7Afu6fOysa7cPD1QwqHv0wQfzLwfHvUwixOI5tBz069p1zL4" crossorigin="anonymous">
40     </script>
41 </head>
42
43 <body style="background-color: CadetBlue;">
44     <h3 class="text-center mt-3">Crear profesor</h3>
45     <div class="container mt-3">
46         <form name="n" action="<?php echo $_SERVER['PHP_SELF']; ?>" method="POST">
47             <div class="mt-3">
48                 <input type="text" name="nombre" placeholder="Nombre" required class="form-control" />
49             </div>
50             <div class="mt-3">
51                 <input type="number" step="0.01" name="sueldo" placeholder="Sueldo" required class="form-control" />
52             </div>
53             <div class="mt-3">
54                 <select name="departamentos" class="form-select" aria-label="Default select example">
55                     <?php
56                     while ($item = $todas->fetch(PDO::FETCH_OBJ)) {
57                         echo "<option value='{$item->id}'>{$item->nom_dep}</option>";
58                     }
59                     ?>
60                 </select>
61             </div>
62             <div class="mt-2">
63                 <input type="submit" name="crear" value="Crear" class="btn btn-success mr-2" />
64                 <input type="reset" value="Limpiar" class="btn btn-warning mr-2" />
65                 <a href="profesores.php" class="btn btn-primary mr-2">Volver</a>
66             </div>
67         </form>
68     </div>
69 </body>
70
71 </html>
```



Al pulsar el botón crear le añadimos las condiciones que hace falta para que no haya errores y guardamos los campos si pasan la condición, el selector recoge la opción indicada. Para recoger los datos llamamos devolverTodos().

```
<?php
require '../vendor/autoload.php';

use Clases\Departamentos;
use Clases\Profesores;

$profesor = new Departamentos();
$todas = $profesor->devolverTodos();
$profesor = null;

if (isset($_POST['crear'])) {
    $nombre = trim($_POST['nombre']);
    $sueldo = trim($_POST['sueldo']);
    if (strlen($nombre) == 0 || strlen($sueldo) == 0) {
        $_SESSION['mensaje'] = "Rellene el campo";
        header("Location:{"$_SERVER['PHP_SELF']}");
        die();
    }
    $profesor = new Profesores;
    $profesor->setNombre(ucwords($nombre));
    $profesor->setSueldo(ucwords($sueldo));
    $profesor->setIdDep($_POST['departamentos']);
    $profesor->create();
    $profesor = null;
    header(("Location:profesores.php"));
} else {
    ?>
```

[crearProfesor.php](#)

## 4.3- Borrar profesor

```
98      //Para borrar profesores desde borrarProfesor
99      public function delete()
100     {
101         //consulta para borrar desde tabla profesores que posea el id correspondiente
102         $consulta = "delete from profesores where id=:i";
103         $stmt = parent::$conexion->prepare($consulta);
104         try {
105             //ejecutar consulta
106             $stmt->execute([
107                 ':i' => $this->id
108             ]);
109         } catch (PDOException $ex) {
110             //error si el intento falla
111             die("Error al borrar profesor: " . $ex->getMessage());
112         }
113     }
```

[delete\(\) Profesor.php](#)

Borramos aquel profesor que posea el id parametrizado que coincida con la fila seleccionada, hacemos el borrado de aquella fila que coincida con el id.

Y será llamado en la clase borrarProfesor.php con el id recogido desde \$\_POST['id'] de la fila seleccionada. Y se realiza el método con el id parametrizado

```
<?php
require "../vendor/autoload.php";
use Clases\Profesores;

if(!isset($_POST['id'])){
    header("Location:profesores.php");
    die();
}

$profesor=new Profesores();
$profesor->setId($_POST['id']);
$profesor->delete();
$profesor=null;
header(("Location:profesores.php"));
```

[borrarProfesor.php](#)

```
echo "<form name='borrar' method='POST' class='form-inline' action='borrarProfesor.php'>";
```

[borrarProfesor profesores.php](#)

## 4.4- Editar profesor

Tendremos dos campos con los valores leídos, procedentes del método read().

```
<body style="background-color: CadetBlue;">
<h3 class="text-center mt-3">Editar profesor</h3>
<div class="container mt-3">
    <form name="nt" action="<?php echo $_SERVER['PHP_SELF'] . "?id=$id"; ?>" method="POST">
        <div class="mt-3">
            <input type="text" name="nombre" value="<?php echo $profesorLeido->nom_prof; ?>" required class="form-control" />
        </div>
        <div class="mt-3">
            <input type="text" name="sueldo" value="<?php echo $profesorLeido->sueldo; ?>" required class="form-control" />
        </div>
        <div class="mt-3">
            <select name='departamentos' class="form-select" aria-label="Default select example">
                <!--<option selected><?php echo $profesorLeido->nom_dep; ?></option>; -->
                <?php
                while ($item = $todas->fetch(PDO::FETCH_OBJ)) {
                    echo "<option value='{ $item->id }'{ $item->nom_dep}</option>";
                }
            <?>
            </select>
        </div>
        <div class="mt-2">
            <input type="submit" name="editar" value="Editar" class="btn btn-success mr-2" />
            <a href="profesores.php" class="btn btn-primary mr-2">Volver</a>
        </div>
    </form>
</div>
</body>
</html>
```

Obtenemos el id de la fila seleccionada y obtendremos su información a partir del id seleccionado, si no existe el id devuelto se devuelve a la página anterior.

Al obtenemos los valores de los campos y la opción del select con POST y llamamos el método update para realizar los cambios

```
<?php
require '../vendor/autoload.php';

use Clases\Profesores;
use Clases\Departamentos;

$id=$_GET['id'];
$profesor=new Profesores;
$profesor->setId($id);

$departamento = new Departamentos();
$todas = $departamento->devolverTodos();
$departamento = null;

$profesorLeido=$profesor->readSelected();
if(!isset($_GET['id'])){
    header("Location:index.php");
    die();
}
//Al pulsar el boton editar recoger toda la información y contemplar que no
if(isset($_POST['editar'])){
    $nombre=trim($_POST['nombre']);
    $sueldo=trim($_POST['sueldo']);

    if(strlen($nombre)==0 || strlen($sueldo)==0){
        $_SESSION['mensaje']="Rellene el campo";
        header("Location:{"$_SERVER['PHP_SELF']}?id=$id");
        die();
    }

    //Obtener las variables de cada campo
    if($profesorLeido==ucwords($nombre) || $profesorLeido==ucwords($sueldo)){
        $profesor=null;
        header(("Location:profesores.php"));
        die();
    }

    //realizar el cambio a partir del id
    $profesor->setNombre(ucwords($nombre));
    $profesor->setSueldo(ucwords($sueldo));
    $profesor->setIdDep($_POST['departamentos']);
    $profesor->update();
    $profesor=null;
    header(("Location:profesores.php"));
}else{
```

[editarProfesor.php](#)

### [readSelected\(\) Profesor.php](#)

La consulta sería la obtención de toda la información con las dos tablas unidas gracias al departamento de profesor y el id de departamento de aquél id parametrizado de profesor. Preparamos la consulta a partir de conexión y ejecutamos la sesión con id parametrizado.

```
//para obtener el index seleccionado del select
public function readSelected()
{
    //recuperar aquel profesor con el id leído (uniendo las dos tablas por el foreign key)
    $consulta = "select * from profesores p inner join departamentos d
    on p.dep = d.id where P.id=:i";
    $stmt = parent::$conexion->prepare($consulta);
    try {
        //ejecutar consulta con parametros :atributo
        $stmt->execute([
            ':i' => $this->id
        ]);
    } catch (PDOException $ex) {
        //error si el intento falla
        die("Error al leer profesor: " . $ex->getMessage());
    }
    return $stmt->fetch(PDO::FETCH_OBJ);
}
```

### [update\(\) Profesor.php](#)

Realizamos la actualización con el nombre, sueldo y departamento desde el id parametrizado del profesor, ejecutamos la conexión y la consulta con aquellos atributos parametrizados.

```
//actualizar los profesores con los parametros nombre, stock, pvp e id
public function update(){
    //realizar la actualizacion de las variables a partir del id deseado
    $consulta="update profesores set nom_prof=:n, sueldo=:s, dep=:d where id=:i";
    $stmt=parent::$conexion->prepare($consulta);
    try{
        //ejecucion de la consulta con estos parametros
        $stmt->execute([
            ':i'=>$this->id,
            ':n'=>$this->nombre,
            ':s'=>$this->sueldo,
            ':d'=>$this->idDep
        ]);
    }catch(PDOException $ex){
        //error si falla el intento
        die("Error al actualizar profesor: ".$ex->getMessage());
    }
}
```

## 5.- Tabla departamento

### 5.1- Mostrar tabla

Una tabla vacía con las filas generadas a partir del método `mostrarDepartamentos()` de la clase `departamentos`. Con un `while` que generará las filas necesarias hasta el final. Representará en cada fila el atributo nombre como atributo a trabajar (el id está escondido porque no hace falta verlo desde la vista del usuario).

```
<body style="background-color: CadetBlue;">
  <div class="container mt-3">
    </div>
    <a href="crearDepartamento.php" class='btn btn-primary my-3'>Añadir departamento</a>
    <table class="table table-success table-striped">
      <thead>
        <tr>
          <th scope="col">Nombre departamento</th>
          <th scope="col">Acciones</th>
        </tr>
      </thead>
      <tbody>
        <?php
          //Repetir el proceso mientras haya departamentos
          while ($departamento = $departamentoMostrar->fetch(PDO::FETCH_OBJ)) {
            echo "<tr>";
            echo "<td>{$departamento->nom_dep}</td>";
            echo "<td>";
            echo "<form name='borrar' method='POST' class='form-inline' action='borrarDepartamento.php'>";
            echo "<a href='editarDepartamento.php?id={$departamento->id}' class='btn btn-primary m-auto'></i>Editar</a>";
            echo "<input type='hidden' name='id' value='{$departamento->id}'>";
            echo "<button type='submit' class='ml-2 btn btn-danger'>Borrar</button></form>";
            echo "</td>\n";
            echo "</tr>\n";
          }
        <?>
      </tbody>
    </table>
  </table>
  <div class="col-md-12 text-center">
    <a href="index.php" class='btn btn-primary my-3'>Volver</a>
  </div>
```

El método `mostrarDepartamentos()` para que muestre toda la información de los departamentos y posicionarnos en el `while`.

```
<?php

use Clases\Departamentos;

require '../vendor/autoload.php';
$departamento = new Departamentos();
$departamentoMostrar = $departamento->mostrarDepartamentos();
$departamento = null;
?>
```

[departamentos.php](#)

```

//devolver la información de los departamentos
public function mostrarDepartamentos()
{
    //muestro todos los departamentos de la tabla
    $consulta = "select * from departamentos";
    //preparar consulta
    $stmt = parent::$conexion->prepare($consulta);
    try {
        //ejecutar consulta
        $stmt->execute();
    } catch (PDOException $ex) {
        //error si el intento falla
        die("Error no carga tabla: " . $ex->getMessage());
    }
    return $stmt;
}

```

[mostrarDepartamentos\(\)](#)

## 5.2- Añadir departamento

La consulta para insertar una fila a la tabla departamentos con el parámetro nombre. Ejecutamos la consulta con los parámetros correspondientes con su catch.

```

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Nuevo profesor</title>
    <!-- CSS only -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css" rel="stylesheet">
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/font-awesome@5.15.3/css/all.min.css">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.bundle.min.js" integrity="sha384-wonkru" ></script>
</head>

<body style="background-color: CadetBlue;">
    <h3 class="text-center mt-3">Añadir departamento</h3>
    <div class="container mt-3">
        <form name="n" action="<?php echo $_SERVER['PHP_SELF']; ?>" method="POST">
            <div class="mt-3">
                <input type="text" name="nombre" placeholder="Departamento" required class="form-control" />
            </div>
            <div class="mt-2">
                <input type="submit" name="crear" value="Crear" class="btn btn-success mr-2" />
                <input type="reset" value="Limpiar" class="btn btn-warning mr-2" />
                <a href="departamentos.php" class="btn btn-primary mr-2">Volver</a>
            </div>
        </form>
    </div>
</body>

</html>

```

Al pulsar el botón crear obtenemos el valor del campo nombre y llamar al método crear.

```
<?php
require '../vendor/autoload.php';

use Clases\Departamentos;

if (isset($_POST['crear'])) {
    $nombre = trim($_POST['nombre']);
    if (strlen($nombre) == 0) {
        $_SESSION['mensaje'] = "Rellene el campo";
        header("Location:{"$_SERVER['PHP_SELF']}");
        die();
    }
    $profesor = new Departamentos;
    $profesor->setNom_dep(ucwords($nombre));
    $profesor->create();
    $profesor = null;
    header(("Location:departamentos.php"));
} else {
    ?>
```

[crearDepartamento.php](#)

```
public function create()
{
    //consulta de insercion de departamentos
    $consulta = "insert into departamentos(nom_dep) values(:n)";
    //preparar consulta con el codigo
    $stmt = parent::$conexion->prepare($consulta);
    try {
        //ejecutar consulta con parametros
        $stmt->execute([
            ':n' => $this->nom_dep
        ]);
    } catch (PDOException $ex) {
        //error si el intento falla
        die("Error al insertar departamento: " . $ex->getMessage());
    }
}
```

[create departamentos.php](#)

## 5.3- Borrar departamento

Obtener el id de aquel departamento con POST y borrar aquella fila con el id obtenido.

```
<?php
require "../vendor/autoload.php";
use Clases\Departamentos;

if(!isset($_POST['id'])){
    header("Location:departamentos.php");
    die();
}

$departamento=new Departamentos();
$departamento->setId($_POST['id']);
$departamento->delete();
$departamento=null;
header(("Location:departamentos.php"));
```

[borrarDepartamento.php](#)

```
echo "<form name='borrar' method='POST' class='form-inline' action='borrarDepartamento.php'>";
```

```
//Para borrar departamentos desde borrarDepartamento
public function delete()
{
    //consulta para borrar desde tabla departamentos que posea el id correspondiente
    $consulta = "delete from departamentos where id=:i";
    $stmt = parent::$conexion->prepare($consulta);
    try {
        //ejecutar consulta
        $stmt->execute([
            ':i' => $this->id
        ]);
    } catch (PDOException $ex) {
        //error si el intento falla
        die("Error al borrar profesor: " . $ex->getMessage());
    }
}
```

[delete departamentos.php](#)



#### 4.4- Editor departamento

Editar departamentos con el id seleccionado de la fila tabla de departamentos.php. Una vez hecho, leemos el departamento con el id parametrizado para obtener el valor de nombre desde el método read().

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Editor departamentos</title>
  <!-- CSS only -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-zv6p/0R7aNCj0a/tHf67o/hkh/iB170A9f+tH9ma/yUwvTv+kZO0u6l3oVll3uhp+6O" crossorigin="anonymous">
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@fortawesome/fontawesome-free@5.15.3/css/all.min.css" integrity="sha384-QW9ZmhuUlQ1l1G7Un8zJf7noOcFh3893jUWdP3iMk30JXfWqP3fVq36LJf32" crossorigin="anonymous">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.bundle.min.js" integrity="sha384-JmNKwg1G13B17D7BCgCOupdr1VG6Ib720lq3BCW3x9/RWc7qWwV7ZUjYDq32P" crossorigin="anonymous"></script>

</head>

<body style="background-color: CadetBlue;">
  <h3 class="text-center mt-3">Editor departamento</h3>
  <div class="container mt-3">
    <form name="nt" action="<?php echo $_SERVER['PHP_SELF'] . '?id=$id'; ?>" method="POST">
      <div class="mt-3">
        <input type="text" name="nombre" value="<?php echo $departamentoLeido->nom_dep; ?>" required class="form-control">
      </div>
      <div class="mt-2">
        <input type="submit" name="editar" value="Editar" class="btn btn-success mr-2" />
        <a href="departamentos.php" class="btn btn-primary mr-2">Volver</a>
      </div>
    </form>
  </div>
</body>

</html>
```

Obtenemos el id para parametrizar en el método read y así obtener aquella información gracias al GET id.

Al pulsar el botón editar se llevará a cabo las comprobaciones necesarias para poder realizar el método update sin problemas con el nombre valor solamente (ya que se trabajara con el único valor que posee departamento).

```
<?php
require '../vendor/autoload.php';

use Clases\Departamentos;
$id=$_GET['id'];
$departamento=new Departamentos;
//obtener id
$departamento->setId($id);
$departamentoLeido=$departamento->read();
if(!isset($_GET['id'])){
    header("Location:departamentos.php");
    die();
}
//Al pulsar el boton editar recoger toda la información
if(isset($_POST['editar'])){
    $nombre=trim($_POST['nombre']);
    if(strlen($nombre)==0){
        $_SESSION['mensaje']="Rellene el campo";
        header("Location:{$_SERVER['PHP_SELF']}?id=$id");
        die();
    }

    //Obtener las variables de cada campo
    if($departamentoLeido==ucwords($nombre)){
        $departamento=null;
        header(("Location:departamentos.php"));
        die();
    }

    //realizar el cambio a partir del id
    $departamento->setNom_dep(ucwords($nombre));
    $departamento->update();
    $departamento=null;
    header(("Location:departamentos.php"));
}else{
    ?>
```

[editarDepartamento.php](#)

Leemos el id proporcionado por departamentos desde editarDepartamentos y devolver toda la información procedente de la consulta

```
//lectura del departamento a partir del id de departamentos
public function read()
{
    //recuperar aquel departamento con el id leído
    $consulta = "select * from departamentos where id=:i";
    $stmt = parent::$conexion->prepare($consulta);
    try {
        //ejecutar consulta con parametros :atributo
        $stmt->execute([
            ':i' => $this->id
        ]);
    } catch (PDOException $ex) {
        //error si el intento falla
        die("Error al leer departamento: " . $ex->getMessage());
    }
    return $stmt->fetch(PDO::FETCH_OBJ);
}
```

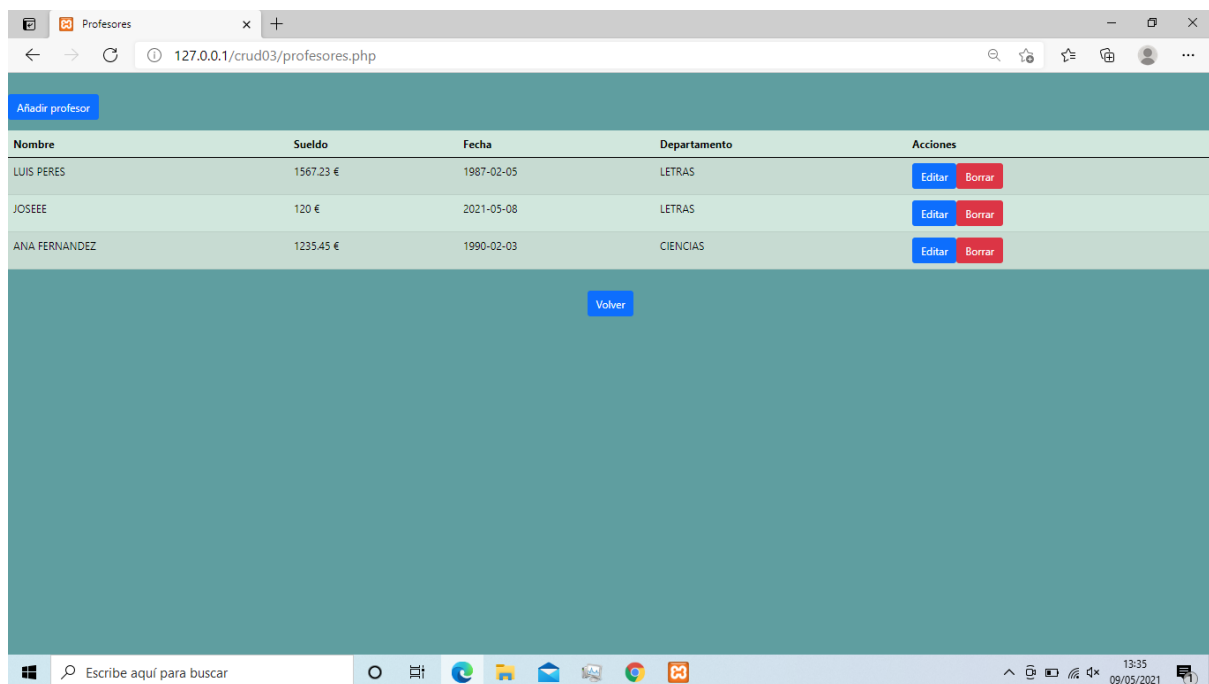
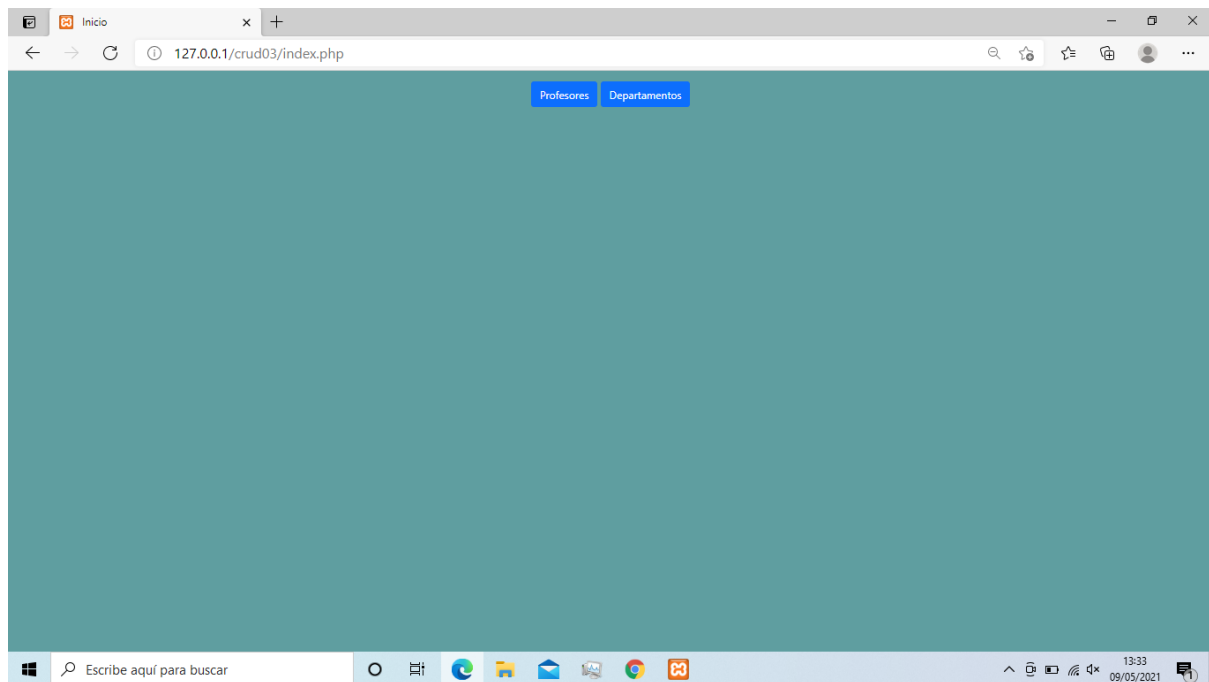
[read\(\) departamentos](#)

Se realizará el cambio necesario con el cambio de valor de nombre (con el valor procesado por las condiciones) y realizar la actualización de aquel id obtenido.

```
//actualizar el departamento con los parametros nombre, stock, pvp e id
public function update(){
    //realizar la actualizacion de las variables a partir del id deseado
    $consulta="update departamentos set nom_dep=:n where id=:i";
    $stmt=parent::$conexion->prepare($consulta);
    try{
        //ejecucion de la consulta con estos parametros
        $stmt->execute([
            ':i'=>$this->id,
            ':n'=>$this->nom_dep
        ]);
    }catch(PDOException $ex){
        //error si falla el intento
        die("Error al actualizar departamento: ".$ex->getMessage());
    }
}
```

[update\(\) departamentos](#)

## 6.- Vista general del proyecto desde el usuario



Añadir profesor

127.0.0.1/crud03/crearProfesor.php

### Crear profesor

ADMIN

3000

CONOCIMIENTO

Crear Limpiar Volver

Escribe aquí para buscar

13:35 09/05/2021

Profesores

127.0.0.1/crud03/profesores.php

Añadir profesor

Nombre	Sueldo	Fecha	Departamento	Acciones
ANA FERNANDEZ	1235.45 €	1990-02-03	CIENCIAS	<a href="#">Editar</a> <a href="#">Borrar</a>
LUIS PERES	1567.23 €	1987-02-05	LETRAS	<a href="#">Editar</a> <a href="#">Borrar</a>
JOSEEE	120 €	2021-05-08	LETRAS	<a href="#">Editar</a> <a href="#">Borrar</a>
ADMIN	3000 €	2021-05-09	CONOCIMIENTO	<a href="#">Editar</a> <a href="#">Borrar</a>

Volver

Escribe aquí para buscar

13:35 09/05/2021

