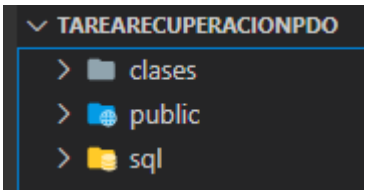
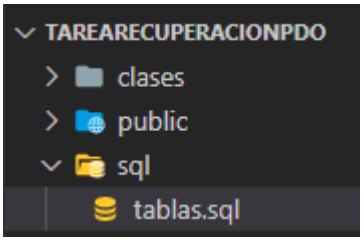
NOMBRE: Quero Gómez, Juan Manuel.

Documentación Tarea Recuperación PDO+CRUD 1.

1. Crearemos una estructura de proyecto con las carpetas usuales(public, clases, sql).



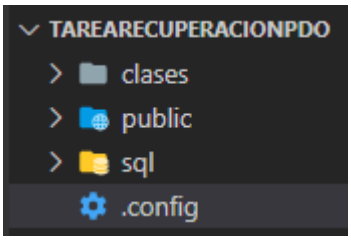
2. Dentro de la carpeta sql que hemos creado anteriormente, creamos un fichero nuevo en el que vamos a poner las tablas de nuestra base de datos.

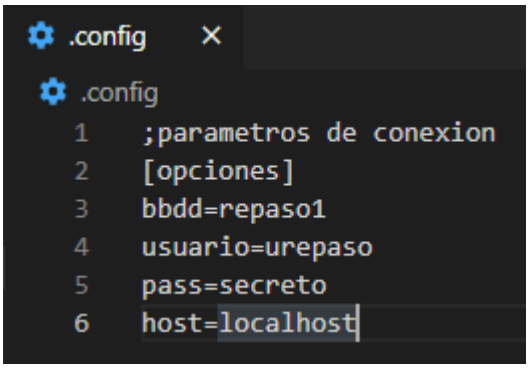



```sql
drop table if exists poststemas;
drop table if exists posts;
drop table if exists users;
drop table if exists tags;
-- ------------------------------------------------------------
create table tags(
    id int auto_increment primary key,
    categoria varchar(120) unique not null
);
create table users(
    id int auto_increment primary key,
    nombre varchar(40) not null,
    apellidos varchar(100) not null,
    username varchar(40) unique not null,
    mail varchar(60) unique not null,
    pass varchar(256) not null
);
create table posts(
    id int auto_increment primary key,
    titulo varchar(80) not null,
    cuerpo text not null,
```

```
    idUser int,
    fecha timestamp default CURRENT_TIMESTAMP,
    constraint postsUser foreign key(idUser) references users(id) on
delete cascade on update cascade
);
create table poststemas(
    id int auto_increment primary key,
    idTag int default -1,
    idPost int,
    constraint reltag foreign key(idTag) references tags(id) on delete
cascade on update cascade,
    constraint relPost foreign key(idPost) references posts(id) on
delete cascade on update cascade
);
```

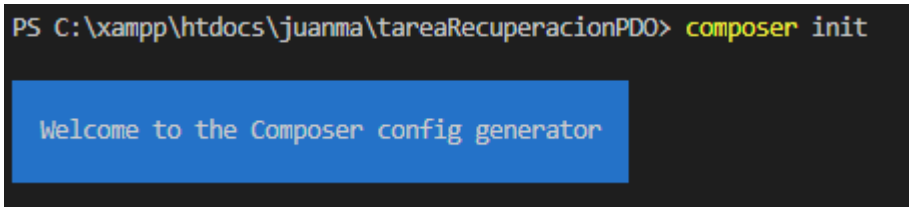
3. Crearemos un archivo de configuración con los parámetros de conexión.





4. Para hacer uso de la autocarga de clases y datos aleatorios, tenemos que iniciar composer en nuestro proyecto para ello haremos lo siguiente:

    a. Iniciamos composer.



b. Se nos creará un fichero .json con los datos que hemos introducido a la hora de iniciar composer en nuestro proyecto.

```
{
    "name": "juanm/tarea-recuperacion-pdo",
    "description": "Proyecto CRUD",
    "type": "project",
    "license": "GNU",
    "authors": [
        {
            "name": "juanma",
            "email": "juanma@gmail.com"
        }
    ],
    "require": {}
}
```

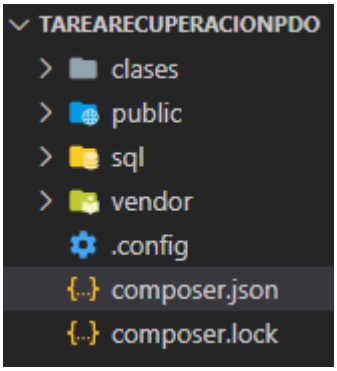
c. A este .json tenemos que añadirle la opción de autoload de la siguiente manera:

```
{
    "name": "juanm/tarea-recuperacion-pdo",
    "description": "Proyecto CRUD",
    "type": "project",
    "license": "GNU",
    "authors": [
        {
            "name": "juanma",
            "email": "juanma@gmail.com"
        }
    ],
    "config": {
        "optimize-autoloader": true
    },
    "autoload": {
        "psr-4": {
            "Clases\\": "clases"
        }
    },
    "require": {}
}
```

d. Una vez tengamos configurado nuestro .json, hacemos un composer update para actualizar.

```
PS C:\xampp\htdocs\juanma\tareaRecuperacionPDO> 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
```

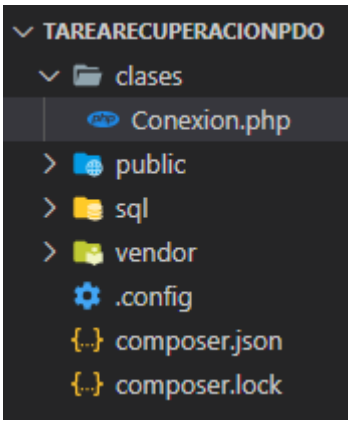
e. Al hacer esto, se nos creará carpeta vendor dentro de nuestro proyecto lo que nos permitirá hacer la autocarga de nuestras clases.



f. Ahora, para poder añadir datos aleatorios a nuestra base de datos, podemos hacer uso de la librería faker. Para ello la instalamos en nuestro proyecto por medio de composer:

```
PS C:\xampp\htdocs\juanma\tareaRecuperacionPDO> composer require fakerphp/faker
Using version ^1.14 for fakerphp/faker
./composer.json has been updated
Running composer update fakerphp/faker
Loading composer repositories with package information
Updating dependencies
Lock file operations: 3 installs, 0 updates, 0 removals
  - Locking fakerphp/faker (v1.14.1)
  - Locking psr/container (1.1.1)
  - Locking symfony/deprecation-contracts (v2.2.0)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 3 installs, 0 updates, 0 removals
  - Installing symfony/deprecation-contracts (v2.2.0): Extracting archive
  - Installing psr/container (1.1.1): Extracting archive
  - Installing fakerphp/faker (v1.14.1): Extracting archive
Generating optimized autoload files
1 package you are using is looking for funding.
Use the `composer fund` command to find out more!
```

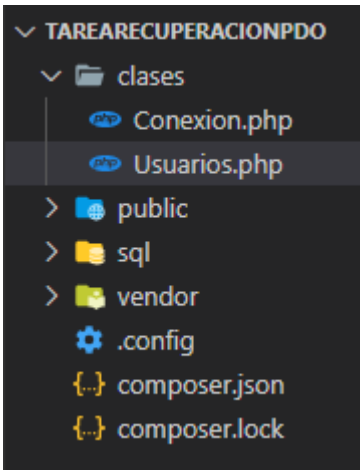
5. Nos creamos una clase “Conexion” para establecer como su nombre indica una conexión a la base de datos.




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

class Conexion {
    protected static $conexion;

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

    public static function crearConexion() {
        $opciones = parse_ini_file("../.config");
        $base = $opciones["bbdd"];
        $usuario = $opciones["usuario"];
        $pass = $opciones["pass"];
        $host = $opciones["host"];

        $dns = "mysql:host=$host;dbname=$base;charset=utf8mb4";

        try {
            self::$conexion = new PDO($dns, $usuario, $pass);
            self::$conexion->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
        }catch(PDOException $ex) {
```

```
            die("Error al intentar conectar a la base de datos: "
.$ex->getMessage());
        }
    }
}
```

6. Creamos otra clase con el nombre “Usuarios”, donde vamos a crear los métodos para hacer nuestro CRUD y demás métodos según necesitemos.




```php
<?php
namespace Clases;
use Clases\Conexion;
use PDO;
use PDOException;

class Usuarios extends Conexion {
    private $id;
    private $nombre;
    private $apellidos;
    private $username;
    private $mail;
    private $pass;

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

    //### CRUD ###
    public function create() {
```

```
        $c = "insert into users(nombre, apellidos, username, mail,
pass) values(:n, :a, :u, :m, :p)";
        $stmt = parent::$conexion->prepare($c);
        try {
            $stmt->execute([
                ':n'=>$this->nombre,
                ':a'=>$this->apellidos,
                ':u'=>$this->username,
                ':m'=>$this->mail,
                ':p'=>$this->pass
            ]);
        }catch(PDOException $ex) {
            die("Error al añadir los usuarios: " .$ex->getMessage());
        }
    }

    public function read() {

    }

    public function update() {

    }

    public function delete() {

    }
    //#########

    //### Otros Métodos ###
    public function usuarioCorrecto($username, $pass): bool {
        $c = "select * from users where username=:u and pass=:p";
        $stmt = parent::$conexion->prepare($c);
        try {
            $stmt->execute([
                ':u'=>$username,
                ':p'=>$pass
            ]);
        }catch(PDOException $ex) {
            die("Error al comprobar los datos del usuario: "
.$ex->getMessage());
        }
        $file = $stmt->fetch(PDO::FETCH_OBJ);
```

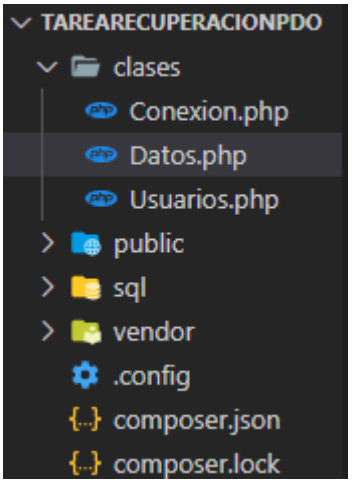
```php
        return ($file != null) ? true : false;
    }

    public function borrarDatos() {
        $c = "delete from users";
        $stmt = parent::$conexion->prepare($c);
        try {
            $stmt->execute();
        }catch(PDOException $ex) {
            die("Error al borrar los datos: " .$ex->getMessage());
        }
    }
    //#########

    //Métodos Set y Get
    /**
     * Get the value of id
     */
    public function getId()
    {
        return $this->id;
    }

    /**
     * Set the value of id
     *
     * @return  self
     */
    public function setId($id)
    {
        $this->id = $id;

        return $this;
    }

    /**
     * Get the value of nombre
     */
    public function getNombre()
    {
        return $this->nombre;
    }
```

```php
    /**
     * Set the value of nombre
     *
     * @return  self
     */
    public function setNombre($nombre)
    {
        $this->nombre = $nombre;

        return $this;
    }

    /**
     * Get the value of apellidos
     */
    public function getApellidos()
    {
        return $this->apellidos;
    }

    /**
     * Set the value of apellidos
     *
     * @return  self
     */
    public function setApellidos($apellidos)
    {
        $this->apellidos = $apellidos;

        return $this;
    }

    /**
     * Get the value of username
     */
    public function getUsername()
    {
        return $this->username;
    }

    /**
     * Set the value of username
     *
```

```
     * @return  self
     */
    public function setUsername($username)
    {
        $this->username = $username;

        return $this;
    }

    /**
     * Get the value of mail
     */
    public function getMail()
    {
        return $this->mail;
    }

    /**
     * Set the value of mail
     *
     * @return  self
     */
    public function setMail($mail)
    {
        $this->mail = $mail;

        return $this;
    }

    /**
     * Get the value of pass
     */
    public function getPass()
    {
        return $this->pass;
    }

    /**
     * Set the value of pass
     *
     * @return  self
     */
    public function setPass($pass)
```

```
    {
        $this->pass = $pass;

        return $this;
    }
}
```

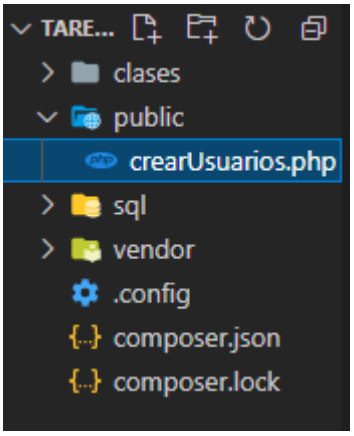
7. Creamos otra clase con el nombre “Datos”. En esta clase, nos crearemos un usuario para poder acceder con él y luego haciendo uso de la librería Faker nos crearemos otros usuarios de prueba.




```php
<?php
namespace Clases;
use Faker\Factory;
use Clases\Usuarios;
require "../vendor/autoload.php";

class Datos {
    public $faker;

    public function __construct($tabla, $cantidad)
    {
        $this->faker = Factory::create('es_ES');
        switch($tabla) {
            case "users":
                $this->miUsuario($cantidad);
                break;
        }
    }
```

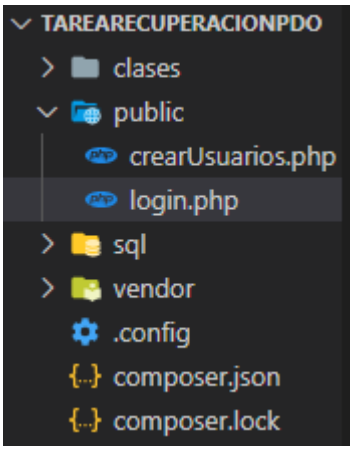
```php
    public function miUsuario($n) {
        $miUsuario = New Usuarios();
        $miUsuario->borrarDatos();
        $miUsuario->setNombre("Juan");
        $miUsuario->setApellidos("Quero Gomez");
        $miUsuario->setUsername("Admin");
        $miUsuario->setMail("juan@gmail.com");
        $password = hash('sha256', 'secret0');
        $miUsuario->setPass($password);
        $miUsuario->create();

        for($i=0;$i<$n-1;$i++) {
            $miUsuario->setNombre($this->faker->firstName());
            $miUsuario->setApellidos($this->faker->lastName() ." ".
$this->faker->lastName());
            $miUsuario->setUsername($this->faker->unique()->userName);
            $miUsuario->setMail($this->faker->unique()->email);
            $miUsuario->setPass($this->faker->sha256);
            $miUsuario->create();
        }

        $miUsuario = null;
    }
}
```

8. Para poder crear nuestros usuarios de prueba, dentro de la carpeta public de nuestro proyecto, crearemos un fichero .php donde llamaremos a la clase Datos y elegiremos la tabla que queremos rellenar con datos y cuantos usuarios de prueba queremos crear.



```php
<?php
require "../vendor/autoload.php";
use Clases\Datos;
$usuarios = new Datos('users', 50);
echo "<h2>Usuarios Creados Correctamente</h2>";
```

9. Por último, crearemos otro fichero pero en este caso vamos hacer un login para comprobar el uso de sesiones.




```php
<?php
session_start();
require "../vendor/autoload.php";

use Clases\Usuarios;

//Creamos una funcion para mostrar un error
function mostrarError($texto)
{
    $_SESSION['error'] = $texto;
    header("Location:{$_SERVER['PHP_SELF']}");
    die();
}

//Comprobamos si hemos dado al boton de login
if (isset($_POST['login'])) {
    //Si existe procesamos nuestro formulario
    $nombre = trim($_POST['nombre']);
    $pass = trim($_POST['pass']);
    //Comprobamos si los campos estan vacios
    if (strlen($nombre) == 0 || strlen($pass) == 0) {
```

```
        mostrarError("Rellene los campos");
    }
    $contrasenaValida = hash('sha256', $pass);
    $usuario = new Usuarios();
    //Comprobamos si los datos introducidos son validos
    if ($usuario->usuarioCorrecto($nombre, $contrasenaValida)) {
        $usuario = null;
        die("Usuario validado con exito");
    } else {
        $usuario = null;
        mostrarError("El nombre o la contraseña son incorrectos");
    }
} else {
?>
    <!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">
        <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/boots
trap.min.css" rel="stylesheet"
integrity="sha384-eOJMYsd53ii+scO/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP48ck
xlpbzKgwra6" crossorigin="anonymous">
        <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/al
l.min.css"
integrity="sha512-iBBXm8fW90+nuLcSKlbmrPcLa0OT92xO1BIsZ+ywDWZCvqsWgccV3
gFoRBv0z+8dLJgyAHIhR35VZc2oM/gI1w==" crossorigin="anonymous" />
        <title>Login</title>
    </head>

    <body style="background-color: bisque;">
        <div class="container mt-3 mb-4">
            <?php
            if (isset($_SESSION['error'])) {
                echo "<p class='my-3 p-3 bg-dark text-danger
font-weight-bold'>{$_SESSION['error']}</p>";
                unset($_SESSION['error']);
            }
```

```
            ?>
            <h3 class="text-center mt-3">Posts Almería S.A.</h3>
            <div class="container mt-3 mb-4">
                <div class="card text-white bg-secondary mb-3 m-auto
mt-5" style="max-width: 48rem;">
                    <div class="card-header text-center">Login</div>
                    <div class="card-body">
                        <form name="login" action="<?php echo
$_SERVER['PHP_SELF'] ?>" method="POST">
                            <div class="form-group">
                                <label for="nu">Nombre de
Usuario</label>
                                <input type="text" class="form-control"
id="nu" placeholder="Ingrese su nombre" name="nombre" required>

                            </div>
                            <div class="form-group mt-3">
                                <label for="np">Password</label>
                                <input type="password"
class="form-control" id="np" placeholder="Password" name="pass"
required>
                            </div>
                            <div class="mt-3">
                                <button type='submit' class='btn
btn-primary' name='login'><i class='fas fa-sign-in-alt mr-2'></i>
Login</button>
                            </div>
                        </form>
                    </div>
                </div>
            </div>

    </body>

    </html>
<?php } ?>
```