

Tareas Unidad 5

Desarrollo Web en Entorno Servidor

Autor: Derimán Tejera Fumero

Fecha: 25/10/2024

Índice

Enunciado de la práctica	3
Explicación del código	6
Descarga e instalación de Composer.....	6
Creación de la base de datos para el ejercicio	6
Estructura de directorios del proyecto.....	10
¿Qué es Composer?	11
¿Qué dependencias hemos utilizado y cuales son sus funciones?	11
crearDatos.php.....	11
Fcrear.php.....	13
generarCode.php.....	15
Guardar_Jugador.php	17
Index.php.....	19
Instalacion.php	20
Jugadores.php	21
Conexión.php	23
Jugador.php	25
Styles.css	27
Views/plantillas/plantilla1.blade.php	29
vCrear.blade.php	29
vInstalacion.blade.php	30
Vjugadores.blade.php	31
Mostrando el funcionamiento.....	31

Enunciado de la práctica

Deseamos diseñar una aplicación para gestionar los jugadores de un equipo de fútbol. Para dicha aplicación trabajaremos con la tabla jugadores, para ella nos crearemos una base de datos nueva y un usuario con permiso en ella. Podemos reutilizar el usuario gestor, de otros ejercicios.

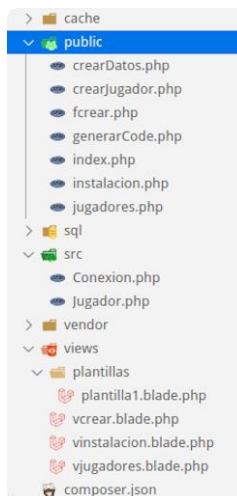
Se deja el enlace a "tablas.sql" con lo necesario para crear la base de datos, dar a gestor permiso en ella y crear la tabla jugadores. Archivo SQL. (sql - 1,15 KB)

De los jugadores guardaremos nombre y apellidos, una posición, un número de dorsal (único) y un código de barras obligatorio único para cada uno.

Dependencias obligatorias a instalar con Composer.

- philo/laravel-blade como motor de plantillas.
- milon/barcode Para generar los códigos de barra (Se utilizará el formato EAN-13, que utiliza 13 números).
- fzhaninotto/faker Para genera datos aleatorios de prueba.
- autoload con optimizador.

En la siguiente imagen podrás ver una estructura del proyecto ya terminada, lógicamente puedes usar los nombre de carpetas y ficheros que quieras:



1.- "cache" : Esta carpeta es necesaria crearla, y darle permiso "777" si estamos en Linux, para Blade y Barcode.

2.- "public": Aquí crearemos todas las páginas que visualizaremos desde el navegador o páginas ".php" para procesar algo. Las páginas que tendrá serán:

- "crearDatos.php" : Está página me genera los datos de ejemplo.
- "crearJugador.php" : esta página es el "action" del formulario para crear un jugador. Controlaremos errores para no introducir un dorsal que ya existe, que nombre y apellidos no estén vacíos.
- "fcrear.php": Es el formulario para crear el jugador. Llama a la vista: "vcrear.blade.php". A parte de los botones normales pondremos un botón para generar un código de

- barras válido ("href" a la página "generarCode.php"). El campo para el código de barra lo pondremos de sólo lectura (atributo "readonly")
- "generarCode.php": Está página me genera un código de barras EAN-13 válido y que además no exista en la base de datos.
 - "index.php": Es la página de inicio, si la tabla jugadores no tiene datos llamará a "instalacion.php" para crearnos unos datos de ejemplo, si los tiene cargará la página "jugadores.php"
 - "instalacion.php": Carga la vista "vinstalacion.php" básicamente un botón para ir a "crearDatos.php" y crearnos datos de ejemplo.
 - "jugadores.php": Llama a la vista "vjugadores.php" Muestra en una tabla los datos de los jugadores. Tiene un botón crear que llama al formulario para crear un jugador nuevo. Si un jugador NO tiene dorsal mostraremos "Sin asignar".

3.- "sql" : Tiene el archivo ".sql", no es necesaria realmente.

4.- "src": Contiene las Clases para gestionar la base de datos:

- "Conexion.php": Para crear la conexión.
- "Jugador.php": Para gestionar la tabla "jugadores". Aquí estarán los métodos para comprobar el "barcode", el dorsal, devolver los jugadores, insertarlos...En fin todos los que sean necesarios.

5.- "vendor": La genera Composer.

6.- "views": Guardaremos todas las vistas ya mencionadas y la plantillas que utilizaremos.

A parte de todo esto está el archivo: "composer.json".

Recursos Necesarios:

- Ordenador con un entorno AMP instalado y correctamente configurado, acceso a internet y un navegador.
- Xdebug instalado y configurado para PHP.
- Composer instalado
- Visual Studio Code instalado con las extensiones que hemos ido viendo, incluida la extensión para Xdebug.

Recomendaciones:

- Además del manual online de PHP, se recomienda dar libre acceso a Internet para la búsqueda de información.
- Infórmate de las versiones que instalas con Composer, instalar la última versión de las dependencias puede no ser una buena idea. En este proyecto, y a la fecha de elaborar esta documentación, la última versión de "milon/barcode", no es compatible con la versión mas reciente de "philo/laravel-blade". Se ha instalado para no tener problemas la versión: "For Laravel 5.0 and 5.1". En la documentación de ambas librerías puedes ver que la versión de "philo/laravel-blade" última es para el Blade de la versión 5.1 de Laravel.

The image displays two screenshots of a web application interface.

Screenshot 1: Crear Jugador (Create Player) Form

This screenshot shows a form titled "Crear Jugador" (Create Player). The form fields include:

- Nombre (Name):** A text input field labeled "Nombre".
- Apellidos (Last Name):** A text input field labeled "Apellidos".
- Dorsal (Back Number):** A dropdown menu labeled "Dorsal" containing the value "Dorsal".
- Posición (Position):** A dropdown menu labeled "Portero" containing the value "Portero".
- Código de Barras (Barcode):** A text input field labeled "Código de Barras".

Below the form are four buttons:

- Crear (Create)**
- Limpiar (Clear)**
- Volver (Back)**
- Generar Barcode (Generate Barcode)**

Screenshot 2: Instalación (Installation) Page

This screenshot shows a page titled "Instalación" (Installation). It features a single button:

- Instalar Datos de Ejemplo (Install Example Data)**

Listado de Jugadores			
Jugador Creado con éxito			
Nuevo Jugador			
Nombre Completo	Posicion	Dorsal	Código de Barras
Candelaria Betancourt, Alberto	Portero	5	
Noriega Valadez, Diego	Portero	55	
Padrón Reyna, Nalara	Portero	20	
Pons Chavarria, Maria	Portero	3	
Ruiz Ruiz, Esteban	Portero	Sin Asignar	
Espino De la Torre, Alex	Defensa	34	
Leyva Mufiz, Ángeles	Defensa	51	
Nava Nevárez, Miguel Angel	Defensa	4	
Tovar Guardado, Francisca	Defensa	44	
Ceballos Montez, Erik	Lateral Izquierdo	12	
Garrido Gallego, Claudia	Lateral Izquierdo	23	
Téllez Cortez, Helena	Lateral Izquierdo	21	
Farias Armendariz, Maria Carmen	Central	14	

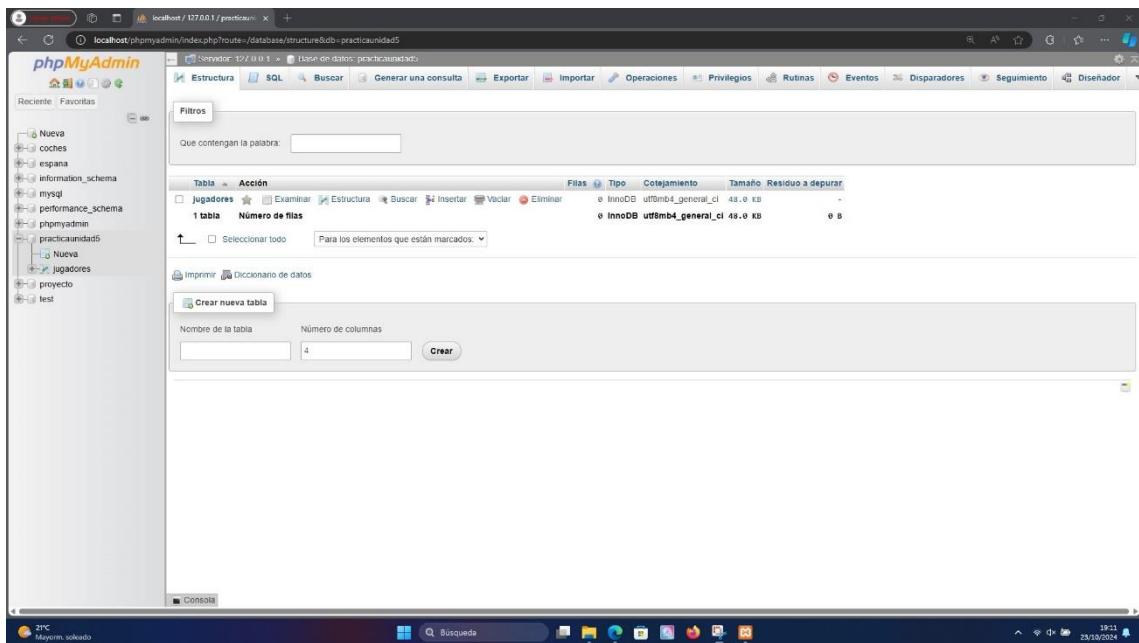
Explicación del código

Descarga e instalación de Composer

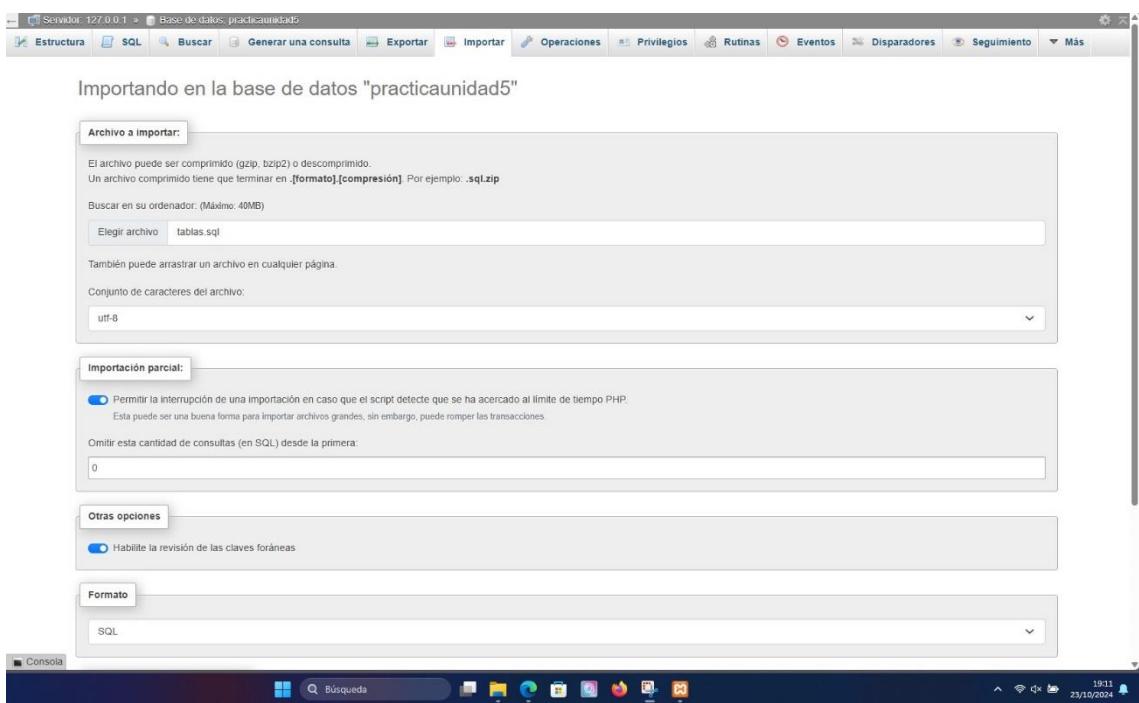
En primer lugar descargamos composer desde este [enlace](#). Se siguen los pasos indicados por la instalación.

Creación de la base de datos para el ejercicio

Desde el explorador Firefox accedemos a “localhost/phpmyadmin”, en el menú lateral izquierdo, pulsamos Nueva y colocamos el nombre de la base de datos que será: “practicaunidad5”.



Luego vamos a la pestaña superior de “Importar” para cargar el archivo “tablas.sql” y le damos a “Aceptar en la parte inferior”:



El contenido del archivo “tablas.sql” es el siguiente:

```
1 -- La seleccionamos
2 use practicaUnidad5;
3 -- Reutilizamos el usuario gestor que ya teníamos (Podemos crear otro)
4 grant all on practicaUnidad5.* to gestor@'localhost';
5 -- Creamos las Tablas --
6 create table jugadores(
7     id int auto_increment primary key,
8     nombre varchar(40) not null,
9     apellidos varchar(60) not null,
10    dorsal int unique,
11    posicion enum('Portero', 'Defensa', 'Lateral Izquierdo', 'Lateral Derecho', 'Central', 'Delantero'),
12    barcode varchar(13) unique not null
13 );
14
15 -- ## Insertamos Algunos datos, descomentar si no te ha funcionado fazinotto/faker ##
16 -- insert into jugadores(nombre, apellidos, dorsal, posicion, barcode) values('Antonio','Gil Gil', 1, 1,'0952945303398');
17 -- insert into jugadores(nombre, apellidos, dorsal, posicion, barcode) values('Ana','Hernandez Perez', 2, 2,'2406603743234');
18 -- insert into jugadores(nombre, apellidos, dorsal, posicion, barcode) values('Juan','Valdemoro Gil', 3, 2, '2829114057100');
19 -- insert into jugadores(nombre, apellidos, dorsal, posicion, barcode) values('Maria','Ruano Perez', 4, 2, '9745708466710');
```

Nota: Si quieres que inserte los datos en la tabla (esto puede ser útil inicialmente para poder crear y visualizar el correcto funcionamiento al mostrar la tabla de jugadores), deberás descomentar las líneas 16 a 19 eliminando el “--”.

Dentro de la carpeta de XAMPP, en la subcarpeta “htdocs”, creamos una nueva carpeta llamada “futbolApp” y abrimos esta carpeta con VS Code.

En VS Code accedemos al terminal de comandos: View > Terminal. En el terminal escribimos el comando **composer init** y vamos respondiendo a las preguntas acerca de nuestro nuevo proyecto para esta tarea número 5, con esto se generará el archivo “composer.json”, después de responder a todas las preguntas, este será (en mi caso) el contenido de composer.json:

```
1 {
2     "name": "futbolapp/gestorjugadores",
3     "description": "Un programa relacionado con la práctica del tema 5 de la asignatura de 3º de CFGS DAW, DWES.",
4     "type": "project",
5     "require": {
6         "fakerphp/faker": "^1.9",
7         "picqer/php-barcode-generator": "^2.2",
8         "jenssegers/blade": "^1.4"
9     },
10    "autoload": {
11        "psr-4": {
12            "Futbolapp\\Gestorjugadores\\": "src/"
13        }
14    },
15    "authors": [
16        {
17            "name": "TuNombre",
18            "email": "tucorreogmail.com"
19        }
20    ],
21    "config": {
22        "platform": {
23            "php": "8.2.4"
24        }
25    }
26 }
```

Ahora desde el terminal de VS Code escribimos el comando **composer install**:

Luego, si hacemos otras modificaciones posteriores, debemos recordar el comando `composer update` para hacerlas efectivas.

Nota importante. Debo señalar que las versiones de las dependencias pedidas por el enunciado para la realización de este ejercicio están desactualizadas, y generaron muchos problemas de incompatibilidades (con la versión de PHP actual) en la instalación, por ello ha sido necesario hacer varias modificaciones.



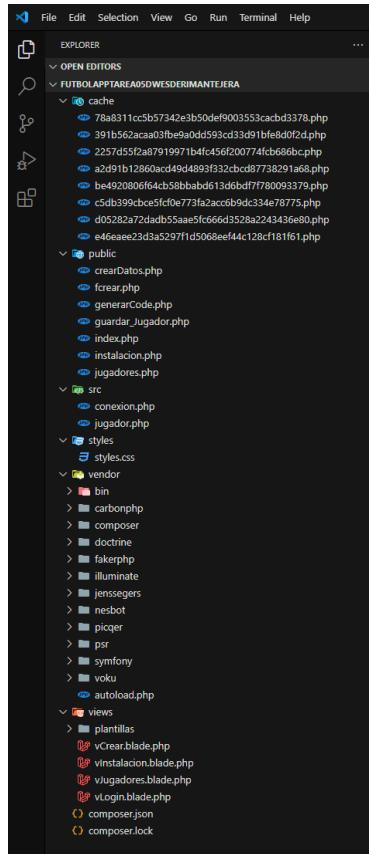
```
1 "require": {  
2     "fakerphp/faker": "^1.9",  
3     "picqer/php-barcode-generator": "^2.2",  
4     "jenssegers/blade": "^1.4"  
5 },
```

Se reemplazó la librería fzaninotto/faker, incompatible con PHP 8, por su fork fakerphp/faker, que mantiene la misma funcionalidad para generar datos de prueba de forma aleatoria. También se sustituyó milon/barcode por picqer/php-barcode-generator, debido a problemas de compatibilidad con PHP 8.1 en la versión requerida de milon/barcode; la nueva librería permite generar códigos de barras EAN-13 sin inconvenientes.

Además, se optó por jenssegers/blade en lugar de philo/laravel-blade, al ser una opción más actual y compatible para manejar plantillas Blade en PHP.

Por último, se añadieron versiones específicas de las librerías de Illuminate para evitar conflictos y asegurar su correcta integración en el proyecto, permitiendo el uso adecuado de las vistas y funcionalidades requeridas.

Estructura de directorios del proyecto



¿Qué es Composer?

Composer es un gestor de dependencias para proyectos en PHP. Facilita la instalación y gestión de bibliotecas y paquetes que necesitamos para realizar este proyecto de **tarea 5 DWES**. Con Composer, podemos especificar qué librerías externas usar en el proyecto en un archivo llamado **composer.json**, y luego **Composer** las descarga y gestiona automáticamente por nosotros.

¿Qué dependencias hemos utilizado y cuales son sus funciones?

fakerphp/faker: Genera los datos ficticios que necesitaremos, es decir, datos para llenar la base de datos de jugadores sin tener que crearlos manualmente.

jenssegers/blade: Implementa el motor de plantillas Blade, así podremos crear las vistas.

illuminate/database (incluido con Blade): No se usa como tal en el proyecto, pero forma parte de Blade, sirve para el manejo de las plantillas y la interacción con bases de datos.

crearDatos.php

```
● ● ●
1 <?php
2
3 require '../vendor/autoload.php';
4
5 use Faker\Factory as Faker;
6 use Futbolapp\Gestorjugadores\Jugador;
7
8 $faker = Faker::create();
9
10 $jugador = new Jugador();
11
12 for ($i = 0; $i < 10; $i++) {
13     $nombre = $faker->firstName;
14     $apellidos = $faker->lastName;
15     $posicion = $faker->randomElement(['Portero', 'Defensa', 'Lateral Izquierdo', 'Lateral Derecho', 'Central', 'Delantero']);
16     $barcode = $faker->unique()->ean13;
17
18     do {
19         $dorsal = $faker->numberBetween(1, 99);
20     } while (!$jugador->esDorsalUnico($dorsal));
21
22     $jugador->crearJugador($nombre, $apellidos, $dorsal, $posicion, $barcode);
23 }
24
25 header('Location: jugadores.php');
26 exit;
```

```
● ● ●
1 require '../vendor/autoload.php';
```

Este fragmento permite que todas las clases y dependencias gestionadas por Composer se carguen automáticamente sin necesidad de incluir una por una cada una de ellas.

```
1 use Faker\Factory as Faker;
2 use Futbolapp\Gestorjugadores\Jugador;
```

Importamos las clases que utilizaremos Faker\Factory y GestorJugadores.

```
1 $faker = Faker::create();
2
3 $jugador = new Jugador();
```

Creamos una instancia de las clases **faker** y otra para **jugador**. La primera para generar los datos aleatorios y la segunda para interactuar con la base de datos.

```
1 for ($i = 0; $i < 10; $i++) {
2     $nombre = $faker->firstName;
3     $apellidos = $faker->lastName;
4     $posicion = $faker->randomElement(['Portero', 'Defensa', 'Lateral Izquierdo', 'Lateral Derecho', 'Central', 'Delantero']);
5     $barcode = $faker->unique()->ean13;
6
7     do {
8         $dorsal = $faker->numberBetween(1, 99);
9     } while (!$jugador->esDorsalUnico($dorsal));
10
11     $jugador->crearJugador($nombre, $apellidos, $dorsal, $posicion, $barcode);
12 }
```

Mediante un bucle **for**, creamos los datos para 10 jugadores. En el interior del bucle hay otro bucle **do while** que genera el dorsal con números entre el 1 y el 99, en el caso de que ya exista un dorsal con ese número en uso el método esDorsalUnico lo comprobará, y si es cierto, generará otro número aleatorio, volviendo a verificarlo usando ese mismo método, y así hasta lograr un dorsal único.

```
1 header('Location: jugadores.php');
2 exit;
```

Cuando ya se han insertado los 10 jugadores, se redirigirá a la página jugadores.php para que el usuario pueda ver la lista.

Fcrear.php

```
● ● ●
1 <?php
2 require '../vendor/autoload.php';
3 session_start();
4
5 use Futbolapp\Gestorjugadores\Jugador;
6 use Faker\Factory as Faker;
7 use Jenssegers\Blade\Blade;
8
9 $views = '../views';
10 $cache = '../cache';
11
12 $blade = new Blade($views, $cache);
13
14 $faker = Faker::create();
15 $jugador = new Jugador();
16
17 if (!isset($_GET['barcode'])) {
18     do {
19         $codigo_barras = $faker->ean13;
20     } while (!$jugador->esCodigoBarrasUnico($codigo_barras));
21
22     header("Location: fcrear.php?nombre=&apellidos=&dorsal=&posicion=&barcode={$codigo_barras}");
23     exit;
24 }
25
26 $nombre = isset($_GET['nombre']) ? $_GET['nombre'] : '';
27 $apellidos = isset($_GET['apellidos']) ? $_GET['apellidos'] : '';
28 $dorsal = isset($_GET['dorsal']) ? $_GET['dorsal'] : '';
29 $posicion = isset($_GET['posicion']) ? $_GET['posicion'] : '';
30 $codigo_barras = isset($_GET['barcode']) ? $_GET['barcode'] : '';
31
32 echo $blade->render('vcrear', [
33     'error' => $_SESSION['error'] ?? null,
34     'mensaje_exito' => $_SESSION['mensaje_exito'] ?? null,
35     'codigo_barras' => $codigo_barras,
36     'nombre' => $nombre,
37     'apellidos' => $apellidos,
38     'dorsal' => $dorsal,
39     'posicion' => $posicion
40 ]);
41
42 unset($_SESSION['error'], $_SESSION['mensaje_exito']);
```

```
● ● ●
1 session_start();
```

Se inicia una sesión para almacenar y recuperar los datos que guardemos como los mensajes de error o éxito en la inserción de datos de jugadores.

```
● ● ●
1 $views = '../views';
2 $cache = '../cache';
3
4 $blade = new Blade($views, $cache);
```

Configuramos Blade indicando las rutas de las vistas y el directorio donde se guardarán los archivos compilados en caché.

```
 1 $faker = Faker::create();
 2 $jugador = new Jugador();
 3
 4 if (!isset($_GET['barcode'])) {
 5     do {
 6         $codigo_barras = $faker->ean13;
 7     } while (!$jugador->esCodigoBarrasUnico($codigo_barras));
 8
 9     header("Location: fcrear.php?nombre=$apellidos&dorsal=$posicion&barcode={$codigo_barras}");
10     exit;
11 }
12
```

Usamos **Faker** para crear una instancia para generar el código de barras **EAN-13**.

Se buscará el código de barras generado previamente, en la URL, si no lo encuentra, entonces generaremos uno y verificaremos con el método **esCodigoBarrasUnico** si ya existe otro igual en la base de datos, si es así, se genera sucesivamente hasta que sea único.

Finalmente se pasa el código de barras junto con el resto de datos del jugador en vacío por la URL.

```
 1 $nombre = isset($_GET['nombre']) ? $_GET['nombre'] : '';
 2 $apellidos = isset($_GET['apellidos']) ? $_GET['apellidos'] : '';
 3 $dorsal = isset($_GET['dorsal']) ? $_GET['dorsal'] : '';
 4 $posicion = isset($_GET['posicion']) ? $_GET['posicion'] : '';
 5 $codigo_barras = isset($_GET['barcode']) ? $_GET['barcode'] : '';
```

Aquí recuperaremos los valores indicados desde la URL y se almacenan en las respectivas variables, si no existen, entonces se almacenará una cadena vacía “”.

```
● ● ●
1 echo $blade->render('vcrear', [
2     'error' => $_SESSION['error'] ?? null,
3     'mensaje_exito' => $_SESSION['mensaje_exito'] ?? null,
4     'codigo_barras' => $codigo_barras,
5     'nombre' => $nombre,
6     'apellidos' => $apellidos,
7     'dorsal' => $dorsal,
8     'posicion' => $posicion
9 ]);
```

Utilizaremos Blade para renderizar la plantilla **vcrear.blade.php** pasándole las variables a la vista.

```
● ● ●
1 unset($_SESSION['error'], $_SESSION['mensaje_exito']);
```

Finalizamos eliminando las variables almacenadas en la sesión para evitar errores cuando se cargue de nuevo.

generarCode.php

```
● ● ●
1 <?php
2
3 require '../vendor/autoload.php';
4
5 use Futbolapp\Gestorjugadores\Jugador;
6 use Faker\Factory as Faker;
7
8 $faker = Faker::create();
9
10 $jugador = new Jugador();
11
12 do {
13     $codigo_barras = $faker->ean13;
14 } while (!$jugador->esCodigoBarrasUnico($codigo_barras));
15
16 $nombre = isset($_GET['nombre']) ? $_GET['nombre'] : '';
17 $apellidos = isset($_GET['apellidos']) ? $_GET['apellidos'] : '';
18 $dorsal = isset($_GET['dorsal']) ? $_GET['dorsal'] : '';
19 $posicion = isset($_GET['posicion']) ? $_GET['posicion'] : '';
20
21 header("Location: fcrear.php?nombre={$nombre}&apellidos={$apellidos}&dorsal={$dorsal}&posicion={$posicion}&barcode={$codigo_barras}");
22 exit;
```

```
● ● ●
1 do {
2     $codigo_barras = $faker->ean13;
3 } while (!$jugador->esCodigoBarrasUnico($codigo_barras));
```

Utilizamos un bucle **do while** para generar un código de barras único.

```
● ● ●
1 $nombre = isset($_GET['nombre']) ? $_GET['nombre'] : '';
2 $apellidos = isset($_GET['apellidos']) ? $_GET['apellidos'] : '';
3 $dorsal = isset($_GET['dorsal']) ? $_GET['dorsal'] : '';
4 $posicion = isset($_GET['posicion']) ? $_GET['posicion'] : '';
```

Obtenemos los valores de los campos del formulario que están presentes en la URL, así mantenemos los datos del usuario si se produce una redirección.

```
● ● ●
1 header("Location: fcrear.php?nombre={$nombre}&apellidos={$apellidos}&dorsal={$dorsal}&posicion={$posicion}&barcode={$codigo_barras}");
2 exit;
```

Una vez hemos generado el código de barras, se redirige al usuario a fcrear.php, al pasarle los datos por la URL se logra mantener el formulario prerrellenado, sin que el usuario deba volver a introducirlos al producirse la carga de la página.

Guardar_Jugador.php

```
● ● ●
1 <?php
2
3 require ' ../vendor/autoload.php';
4
5 use Futbolapp\Gestorjugadores\Jugador;
6
7 session_start();
8
9 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
10     $nombre = trim($_POST['nombre']);
11     $apellidos = trim($_POST['apellidos']);
12     $dorsal = trim($_POST['dorsal']);
13     $posicion = $_POST['posicion'];
14     $barcode = trim($_POST['codigo_barras']);
15
16     $jugador = new Jugador();
17
18     if (!$jugador->esDorsalUnico($dorsal)) {
19         $_SESSION['error'] = "El dorsal ya está en uso. Por favor, elige otro.";
20         header('Location: fcrear.php');
21         exit;
22     }
23
24     if (empty($barcode)) {
25         $_SESSION['error'] = "El código de barras no puede estar vacío.";
26         header('Location: fcrear.php');
27         exit;
28     }
29     if (!$jugador->esCodigoBarrasUnico($barcode)) {
30         $_SESSION['error'] = "El código de barras ya está en uso. Por favor, genera uno nuevo.";
31         header('Location: fcrear.php');
32         exit;
33     }
34
35     $jugador->crearJugador($nombre, $apellidos, $dorsal, $posicion, $barcode);
36
37     $_SESSION['mensaje_exito'] = "Jugador creado con éxito.";
38
39     header('Location: jugadores.php');
40     exit;
41 }
```

```
● ● ●
1 use Futbolapp\Gestorjugadores\Jugador;
```

Importamos la clase Jugador para poderla usar en este archivo, la clase la necesitamos porque contiene los métodos necesarios para interactuar con la base de datos.

```
● ● ●
1 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
2     $nombre = trim($_POST['nombre']);
3     $apellidos = trim($_POST['apellidos']);
4     $dorsal = trim($_POST['dorsal']);
5     $posicion = $_POST['posicion'];
6     $barcode = trim($_POST['codigo_barras']);
```

Verificamos si la solicitud es de tipo POST lo que indicaría que proviene del formulario, si no es así, no se ejecutará el resto del código.

Luego recuperamos los datos del formulario y aplicamos trim para eliminar espacios en blanco.

```
● ● ●
1 $jugador = new Jugador();
2
3 if (!$jugador->esDorsalUnico($dorsal)) {
4     $_SESSION['error'] = "El dorsal ya está en uso. Por favor, elige otro.";
5     header('Location: fcrear.php');
6     exit;
7 }
8
9 if (empty($barcode)) {
10    $_SESSION['error'] = "El código de barras no puede estar vacío.";
11    header('Location: fcrear.php');
12    exit;
13 }
14 if (!$jugador->esCodigoBarrasUnico($barcode)) {
15    $_SESSION['error'] = "El código de barras ya está en uso. Por favor, genera uno nuevo.";
16    header('Location: fcrear.php');
17    exit;
18 }
```

Creamos una instancia para jugador.

En los subsiguientes condicionales if, verificamos si el dorsal ya está en uso, si el código de barras está vacío o si el código de barras ya existe en la base de datos respectivamente. Para cada caso se almacenará en la variable `$_SESSION` un mensaje para mostrar al usuario con esta información para que comprenda el error y actúe en consecuencia.

```
● ● ●
1 $jugador->crearJugador($nombre, $apellidos, $dorsal, $posicion, $barcode);
```

En este punto, ya habrá superado todas las verificaciones anteriores y por tanto se procederá a insertar en la base de datos utilizando el método `crearJugador()` de la clase `jugador`.

```
● ● ●
1 $_SESSION['mensaje_exito'] = "Jugador creado con éxito.";
2
3 header('Location: jugadores.php');
4 exit;
```

Después de crear el jugador, se almacenará un mensaje indicando el éxito de la operación en `$_SESSION` para luego mostrárselo al usuario y se redirigirá a `jugadores.php`.

Index.php

```
● ● ●  
1 <?php  
2  
3 require ' ../vendor/autoload.php';  
4  
5 session_start();  
6  
7 use Futbolapp\Gestorjugadores\Jugador;  
8  
9 $jugadorModel = new Jugador();  
10  
11 $jugadores = $jugadorModel→obtenerJugadores();  
12  
13 if (empty($jugadores)) {  
14     header('Location: instalacion.php');  
15     exit;  
16 } else {  
17     header('Location: jugadores.php');  
18     exit;  
19 }
```

El propósito de **index.php** es comprobar si ya hay usuarios en la base de datos, si no los hay, entonces redirigirá a **instalacion.php** para crear los datos aleatorios de 10 jugadores.

```
● ● ●  
1 $jugadorModel = new Jugador();  
2  
3 $jugadores = $jugadorModel→obtenerJugadores();
```

Creamos un objeto de la clase jugador para interactuar con la base de datos. Y luego obtenemos los jugadores que existan.

```
● ● ●
1 if (empty($jugadores)) {
2     header('Location: instalacion.php');
3     exit;
4 } else {
5     header('Location: jugadores.php');
6     exit;
7 }
```

Comprobaremos si la tabla de jugadores está vacía, si está vacía entonces redirigiremos a **instalacion.php** para generar los 10 jugadores, si no está vacía, entonces redirigiremos a **jugadores.php** para mostrar la tabla con los jugadores.

Instalacion.php

```
● ● ●
1 <?php
2
3 require '../vendor/autoload.php';
4
5 use Jenssegers\Blade\Blade;
6
7 $views = '../views';
8 $cache = '../cache';
9 $blade = new Blade($views, $cache);
10
11 echo $blade->render('vinstalacion');
```

```
● ● ●
1 use Jenssegers\Blade\Blade;
```

Importamos la clase Blade.

```
● ● ●
1 $views = '../views';
2 $cache = '../cache';
```

Definimos la ubicación de las vistas y la caché.

```
● ● ●
1 $blade = new Blade($views, $cache);
2
3 echo $blade->render('vinstalacion');
```

Creamos una instancia de Blade y renderizamos la vista de **vinstalacion.blade.php**.

Jugadores.php

```
● ● ●
1 <?php
2 require '../vendor/autoload.php';
3
4 use Futbolapp\Gestorjugadores\Jugador;
5 use Jenssegers\Blade\Blade;
6
7 session_start();
8
9 $views = '../views';
10 $cache = '../cache';
11
12 $blade = new Blade($views, $cache);
13
14 $jugadorModel = new Jugador();
15 $jugadores = $jugadorModel->obtenerJugadores();
16
17 $mensaje_exito = $_SESSION['mensaje_exito'] ?? null;
18
19 echo $blade->render('vjugadores', [
20     'jugadores' => $jugadores,
21     'mensaje_exito' => $mensaje_exito
22 ]);
23
24 unset($_SESSION['mensaje_exito']);
```

El objetivo de este archivo es mostrar la lista de jugadores obtenidos de la base de datos y también el de mostrar un mensaje de éxito si se ha introducido un usuario nuevo individual de forma correcta.

```
● ● ●
1 $views = '../views';
2 $cache = '../cache';
3
4 $blade = new Blade($views, $cache);
```

Esto es la configuración de Blade, indicamos la ubicación de las vistas y el caché, y creamos una nueva instancia de Blade.

```
● ● ●
1 $jugadorModel = new Jugador();
2 $jugadores = $jugadorModel->obtenerJugadores();
```

Creamos una instancia de Jugador y obtenemos los datos de los jugadores almacenados mediante el método **obtenerJugadores()**;

```
● ● ●
1 $mensaje_exito = $_SESSION['mensaje_exito'] ?? null;
```

Almacenamos el mensaje de éxito en `$_SESSION`, si no existe se le asigna null.

```
● ● ●
1 echo $blade->render('vjugadores', [
2     'jugadores' => $jugadores,
3     'mensaje_exito' => $mensaje_exito
4 ]);
5
6 unset($_SESSION['mensaje_exito']);
```

Renderizamos la vista `vjugadores.blade.php` pasandole los datos de los jugadores como parámetros.

Finalmente limpiamos el mensaje almacenado en `$_SESSION`, ya se ha mostrado, así que es eliminado para evitar errores.

Conexión.php

```
1 <?php
2
3 namespace Futbolapp\Gestorjugadores;
4
5 use PDO;
6 use PDOException;
7
8 class Conexion
9 {
10     private $host;
11     private $db;
12     private $user;
13     private $pass;
14     private $dsn;
15     protected $conexion;
16
17     public function __construct()
18     {
19         $this->host = "localhost";
20         $this->db = "practicaUnidad5";
21         $this->user = "gestor";
22         $this->pass = "admin";
23         $this->dsn = "mysql:host={$this->host};dbname={$this->db};charset=utf8mb4";
24         $this->crearConexion();
25     }
26
27     public function crearConexion()
28     {
29         try {
30             $this->conexion = new PDO($this->dsn, $this->user, $this->pass);
31             $this->conexion->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
32         } catch (PDOException $ex) {
33             die("Error en la conexión: " . $ex->getMessage());
34         }
35     }
36 }
```

Este es el archivo encargado de establecer la conexión con la base de datos utilizando PDO.

```
1 namespace Futbolapp\Gestorjugadores;
2
3 use PDO;
4 use PDOException;
```

Declaramos el namespace e importamos las clases necesarias PDO y PDOException.

```
● ● ●
1 private $host;
2 private $db;
3 private $user;
4 private $pass;
5 private $dsn;
6 protected $conexion;
```

Dentro de la clase Conexión, definimos las propiedades de la clase

```
● ● ●
1 public function __construct()
2 {
3     $this->host = "localhost";
4     $this->db = "practicaUnidad5";
5     $this->user = "gestor";
6     $this->pass = "admin";
7     $this->dsn = "mysql:host={$this->host};dbname={$this->db};charset=utf8mb4";
8     $this->crearConexion();
9 }
```

El constructor de la clase se ejecuta automáticamente y se le asignan los valores de conexión, se construye el DSN y para terminar se llama al método crearConexion() para establecer la conexión con la base de datos.

```
● ● ●
1 public function crearConexion()
2 {
3     try {
4         $this->conexion = new PDO($this->dsn, $this->user, $this->pass);
5         $this->conexion->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
6     } catch (PDOException $ex) {
7         die("Error en la conexión: " . $ex->getMessage());
8     }
9 }
```

Aquí crearConexion() intenta crear una nueva instancia utilizando los datos proporcionados por el constructor. Luego se utiliza una estructura try catch para si ocurren errores durante la ejecución.

Jugador.php

```
● ● ●
1 <?php
2
3 namespace Futholapp\Gestorjugadores;
4
5 use PDO;
6
7 class Jugador extends Conexion {
8
9     public function obtenerJugadores() {
10         $consulta = "SELECT * FROM jugadores ORDER BY dorsal ASC";
11         $stmt = $this->conexion->prepare($consulta);
12         $stmt->execute();
13         return $stmt->fetchAll(PDO::FETCH_OBJ);
14     }
15
16     public function crearJugador($nombre, $apellidos, $dorsal, $posicion, $barcode) {
17         $consulta = "INSERT INTO jugadores (nombre, apellidos, dorsal, posicion, barcode) VALUES (:nombre, :apellidos, :dorsal, :posicion, :barcode)";
18         $stmt = $this->conexion->prepare($consulta);
19         $stmt->execute([
20             ':nombre' => $nombre,
21             ':apellidos' => $apellidos,
22             ':dorsal' => $dorsal,
23             ':posicion' => $posicion,
24             ':barcode' => $barcode
25         ]);
26     }
27
28     public function esDorsalUnico($dorsal) {
29         $consulta = "SELECT COUNT(*) as total FROM jugadores WHERE dorsal = :dorsal";
30         $stmt = $this->conexion->prepare($consulta);
31         $stmt->execute([':dorsal' => $dorsal]);
32         $resultado = $stmt->fetch(PDO::FETCH_OBJ);
33         return $resultado->total == 0;
34     }
35
36     public function esCodigoBarrasUnico($barcode) {
37         $consulta = "SELECT COUNT(*) as total FROM jugadores WHERE barcode = :barcode";
38         $stmt = $this->conexion->prepare($consulta);
39         $stmt->execute([':barcode' => $barcode]);
40         $resultado = $stmt->fetch(PDO::FETCH_OBJ);
41         return $resultado->total == 0;
42     }
43 }
```

```
● ● ●
1 class Jugador extends Conexion {
```

La clase jugador hereda de la clase conexión lo que le permite utilizar la conexión a la base de datos.

```
● ● ●
1 public function obtenerJugadores() {
2     $consulta = "SELECT * FROM jugadores ORDER BY dorsal ASC";
3     $stmt = $this->conexion->prepare($consulta);
4     $stmt->execute();
5     return $stmt->fetchAll(PDO::FETCH_OBJ);
6 }
```

Este método permite hacer una consulta a la base de datos para obtener todos los jugadores almacenados en la tabla y ordenados de menor a mayor dorsal.

```
● ● ●  
1 return $stmt→fetchAll(PDO::FETCH_OBJ);
```

Cada fila es recuperada como un objeto.

```
● ● ●  
1 public function crearJugador($nombre, $apellidos, $dorsal, $posicion, $barcode) {  
2     $consulta = "INSERT INTO jugadores (nombre, apellidos, dorsal, posicion, barcode) VALUES (:nombre, :apellidos, :dorsal, :posicion, :barcode)";  
3     $stmt = $this→conexion→prepare($consulta);  
4     $stmt→execute([  
5         ':nombre' => $nombre,  
6         ':apellidos' => $apellidos,  
7         ':dorsal' => $dorsal,  
8         ':posicion' => $posicion,  
9         ':barcode' => $barcode  
10    ]);  
11 }
```

Este método inserta un nuevo jugador en la base de datos, se le pasan los datos del jugador y se ejecuta la consulta.

```
● ● ●  
1 public function esDorsalUnico($dorsal) {  
2     $consulta = "SELECT COUNT(*) as total FROM jugadores WHERE dorsal = :dorsal";  
3     $stmt = $this→conexion→prepare($consulta);  
4     $stmt→execute([':dorsal' => $dorsal]);  
5     $resultado = $stmt→fetch(PDO::FETCH_OBJ);  
6     return $resultado→total == 0;  
7 }
```

Este método comprueba si ya existe un dorsal con el número indicado en la base de datos.

```
● ● ●  
1 public function esCodigoBarrasUnico($barcode) {  
2     $consulta = "SELECT COUNT(*) as total FROM jugadores WHERE barcode = :barcode";  
3     $stmt = $this→conexion→prepare($consulta);  
4     $stmt→execute([':barcode' => $barcode]);  
5     $resultado = $stmt→fetch(PDO::FETCH_OBJ);  
6     return $resultado→total == 0;  
7 }
```

Este método comprueba si ya existe un código de barras con el número indicado en la base de datos.

Styles.css

```
● ● ●
1 body {
2     background-color: #FF8A65 !important;
3 }
4
5 /*----- Estilos del archivo: fcrear.php -----*/
6 #fcrear .form-control {
7     margin-bottom: 15px;
8 }
9
10 #fcrear .btn {
11     margin-right: 10px;
12 }
13
14 #fcrear .button-group {
15     display: flex;
16     justify-content: flex-start;
17 }
18
19 /*----- Estilos del archivo: listaJugadores.php -----*/
20 #listaJugadores .table {
21     border-collapse: collapse;
22     width: 100%;
23     border-spacing: 0;
24     border-spacing: 0;
25 }
26
27 #listaJugadores .table th,
28 #listaJugadores .table td {
29     border: none;
30     padding: 8px;
31     text-align: center;
32     vertical-align: middle;
33     border-collapse: collapse;
34 }
35
36 #listaJugadores .table thead {
37     background-color: #343A40;
38     color: white;
39     text-align: center;
40 }
41
42 #listaJugadores .table tbody tr:nth-child(odd) {
43     background-color: #3E444A;
44     color: white;
45     text-align: center;
46 }
47
48 #listaJugadores .table tbody tr:nth-child(even) {
49     background-color: #343A40;
50     color: white;
51     text-align: center;
52 }
53
54 #listaJugadores .table tbody tr:hover {
55     background-color: white;
56     color: black;
57 }
58
59 #listaJugadores .table .barcode {
60     font-family: "Libre Barcode 128", cursive;
61     font-size: 50px;
62     display: flex;
63     justify-content: center;
64     align-items: center;
65     height: 100%;
66 }
67
68 /*----- Estilos del archivo: instalacion.php -----*/
69 #instalacion .container {
70     margin-top: 50px;
71     text-align: center;
72 }
73
74 #instalacion .install-btn {
75     font-size: 1.5rem;
76     padding: 15px 30px;
77 }
```

Estilos personalizados utilizados en las páginas, estos se apoyan en los utilizados por Bootstrap ya que estos son más específicos.

Views/plantillas/plantilla1.blade.php

```
● ● ●
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>@yield('titulo')</title>
7     <link href="..//styles/styles.css" rel="stylesheet"> <!-- Ruta absoluta -->
8     <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
9     <link href="https://fonts.googleapis.com/css2?family=Libre+Barcode+128&display=swap" rel="stylesheet">
10 </head>
11 <body id="@yield('body_id')">
12     <div class="container mt-5">
13         <h1 class="text-center">@yield('encabezado')</h1>
14         @yield('contenido')
15     </div>
16     <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js"></script>
17 </body>
18 </html>
```

Estructura base que se empleará en las demás vistas, aquí se declara la estructura HTML básica, archivos de estilo y lo necesario para el correcto funcionamiento de Bootstrap.

vCrear.blade.php

```
● ● ●
1 @extends('plantillas.plantilla1')
2
3 @section('titulo', 'Crear Jugador')
4
5 @section('contenido')
6 <div class="container mt-5">
7     <h1 class="text-center">Crear Jugador</h1>
8
9     @if ($error)
10         <div class="alert alert-danger">
11             {{ $error }}
12         </div>
13     @endif
14
15     <?php if (isset($_SESSION['mensaje_exito'])): ?>
16         <div class="alert alert-success">
17             <?= $_SESSION['mensaje_exito'] ?>
18         </div>
19     <?php unset($_SESSION['mensaje_exito']); ?>
20     <?php endif; ?>
21
22
23     <form action="guardar_jugador.php" method="POST">
24         <div class="row">
25             <div class="col-md-6">
26                 <label for="nombre">Nombre</label>
27                 <input type="text" class="form-control" id="nombre" name="nombre" placeholder="Nombre" value="{{ $nombre }}" required>
28             </div>
29             <div class="col-md-6">
30                 <label for="apellidos">Apellidos</label>
31                 <input type="text" class="form-control" id="apellidos" name="apellidos" placeholder="Apellidos" value="{{ $apellidos }}" required>
32             </div>
33         </div>
34
35         <div class="row">
36             <div class="col-md-4">
37                 <label for="dorsal">Dorsal</label>
38                 <input type="number" class="form-control" id="dorsal" name="dorsal" placeholder="Dorsal" value="{{ $dorsal }}" required>
39             </div>
40             <div class="col-md-4">
41                 <label for="posicion">Posición</label>
42                 <select class="form-control" id="posicion" name="posicion" required>
43                     <option value="Portero" {{ $posicion == 'Portero' ? 'selected' : '' }}>Portero</option>
44                     <option value="Defensa" {{ $posicion == 'Defensa' ? 'selected' : '' }}>Defensa</option>
45                     <option value="Centrocampista" {{ $posicion == 'Centrocampista' ? 'selected' : '' }}>Centrocampista</option>
46                     <option value="Delantero" {{ $posicion == 'Delantero' ? 'selected' : '' }}>Delantero</option>
47                 </select>
48             </div>
49             <div class="col-md-4">
50                 <label for="codigo_barras">Código de Barras</label>
51                 <input type="text" class="form-control" id="codigo_barras" name="codigo_barras" placeholder="Código de Barras" value="{{ $codigo_barras }}" readonly>
52             </div>
53         </div>
54
55         <div class="row mt-4">
56             <div class="col-md-12">
57                 <div class="button-group">
58                     <button type="submit" class="btn btn-primary">Crear</button>
59                     <button type="reset" class="btn btn-success">Limpiar</button>
60                     <a href="jugadores.php" class="btn btn-info">Volver</a>
61                     <a href="generarBarcode.php?nombre={{ $nombre }}&apellidos={{ $apellidos }}&dorsal={{ $dorsal }}&posicion={{ $posicion }}>Generar Barcode</a>
62                 </div>
63             </div>
64         </div>
65     </div>
66 </div>
67 @endsection
```

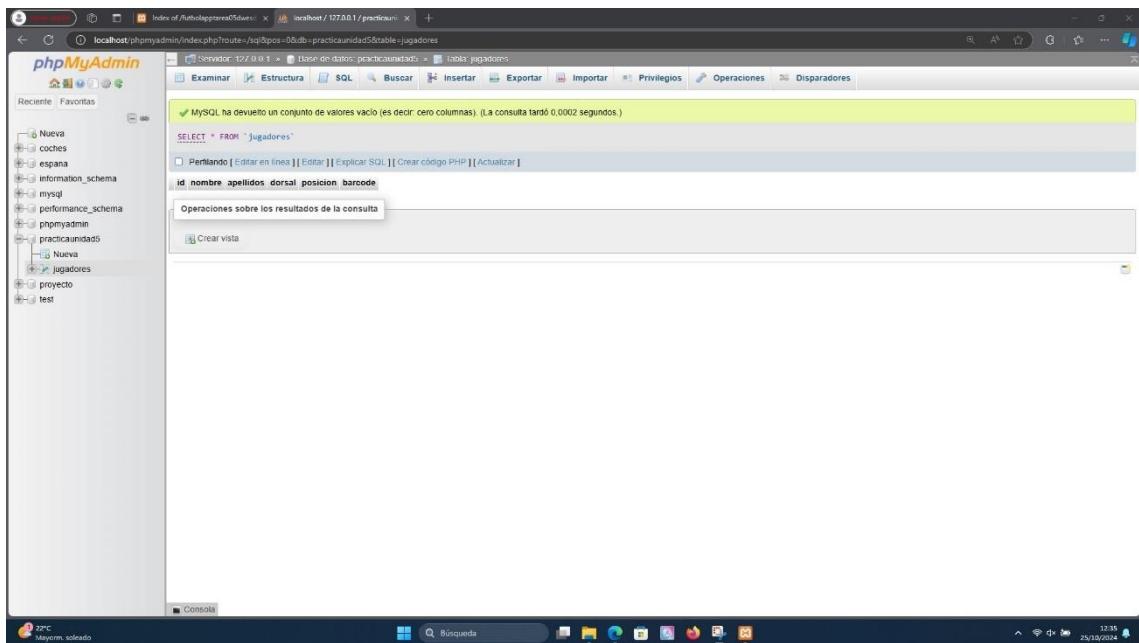
v\Instalacion.blade.php

```
● ● ●
1 @extends('plantillas.plantilla1')
2
3 @section('titulo', 'Instalación de Datos')
4
5 @section('contenido')
6 <div class="container mt-5">
7     <h1 class="text-center">Instalación de Datos</h1>
8     <p class="text-center">Presiona el botón para generar datos de ejemplo.</p>
9     <div class="text-center">
10         <a href="crearDatos.php" class="btn btn-success install-btn">
11             <i class="bi bi-database"></i> Instalar Datos de Ejemplo
12         </a>
13     </div>
14 </div>
15 @endsection
```

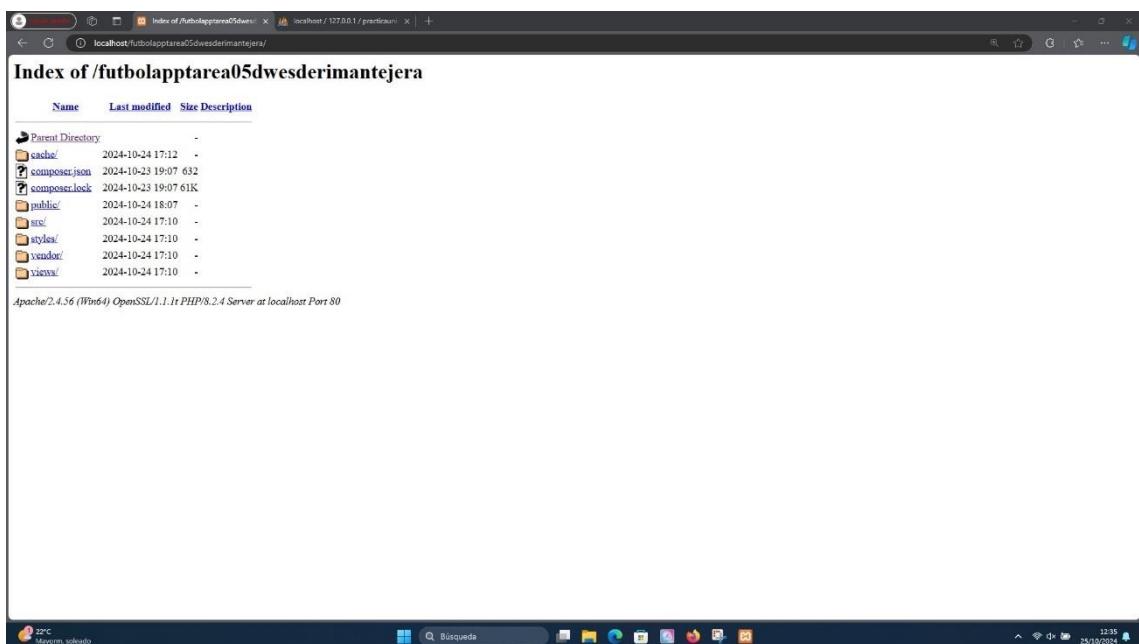
vJugadores.blade.php

```
● ● ●
1 @extends('plantillas.plantilla1')
2
3 @section('titulo', 'Jugadores')
4 @section('body_id', 'listaJugadores')
5
6 @section('contenido')
7 <div class="container mt-5">
8     <h1 class="text-center">Listado de Jugadores</h1>
9
10    @if ($mensaje_exito)
11        <div class="alert alert-success">
12            {{ $mensaje_exito }}
13        </div>
14    @endif
15
16    <a href="fcrear.php" class="btn btn-success mb-3">+ Nuevo Jugador</a>
17
18    <table class="table">
19        <thead>
20            <tr>
21                <th>Nombre Completo</th>
22                <th>Posición</th>
23                <th>Dorsal</th>
24                <th>Código de Barras</th>
25            </tr>
26        </thead>
27        <tbody>
28            @forelse ($jugadores as $jugador)
29                <tr>
30                    <td>{{ $jugador->nombre }} {{ $jugador->apellidos }}</td>
31                    <td>{{ $jugador->posicion }}</td>
32                    <td>{{ $jugador->dorsal ?? 'Sin Asignar' }}</td>
33                    <td class="barcode">{{ $jugador->barcode }}</td>
34                </tr>
35            @empty
36                <tr>
37                    <td colspan="4" class="text-center">No hay jugadores registrados</td>
38                </tr>
39            @endforelse
40        </tbody>
41    </table>
42 </div>
43 @endsection
```

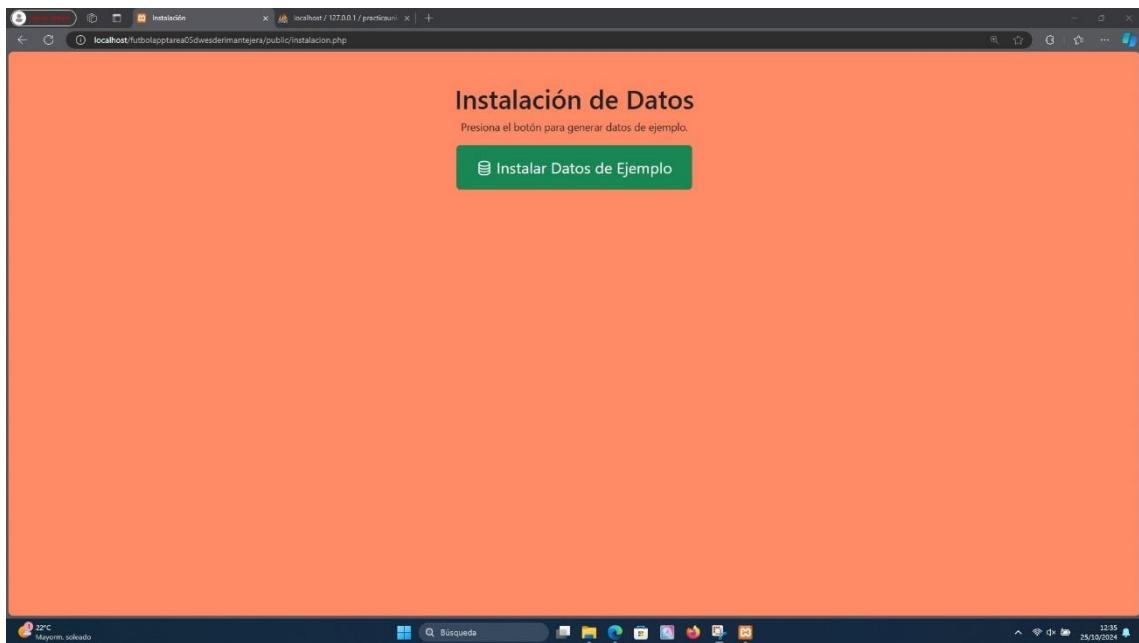
Mostrando el funcionamiento



Hemos creado la base de datos, están las tablas, pero están vacías, no hay jugadores.



Hacemos clic en public/ para comenzar.



Como no hay jugadores, redirige a instalacion.php. Pulsamos el botón para añadir los datos aleatorios de 10 jugadores.

A screenshot of a web browser window titled 'Jugadores'. The page has an orange background and displays a table titled 'Listado de Jugadores'. The table has four columns: 'Nombre Completo', 'Posición', 'Dorsal', and 'Código de Barras'. There are 10 rows of data, each representing a randomly generated player. The 'Código de Barras' column contains a barcode for each player. A green button labeled '+ Nuevo Jugador' is located at the top left of the table.

Alysha Dare	Central	21	
Chaya Schultz	Defensa	22	
Sabryna Boehm	Defensa	49	
Weston Daugherty	Lateral Derecho	54	
Kaela Cummerata	Lateral Izquierdo	63	
Scotty O'Reilly	Lateral Derecho	67	
Dorothy Mueller	Defensa	79	
Marcus Wisoky	Portero	85	
Schuyler Padberg	Lateral Izquierdo	87	

Somos redirigidos pero esta vez a jugadores.php para poder ver la lista de datos almacenados en la base de datos. Y a continuación pulsamos el botón “+ Nuevo Jugador”.

Crear Jugador

Nombre Nombre	Apellidos Apellidos
Dorsal Dorsal	Posición Portero
Código de Barras 9698298957680	

Crear **Limpiar** **Volver** **Generar Barcode**

Ahora podemos rellenar los datos del nuevo jugador que queremos introducir, como puede verse, nada mas entrar ya se ha asignado un código de barras, puede modificarse pulsando tantas veces como se quiera el botón “Generar Barcode”.

Screenshot of a web browser showing the "Crear Jugador" (Create Player) form. The URL is `localhost/futbolapptarea05wesderimantejera/public/c/create.php?nombre=&apellidos=&dorsal=&posicion=&barcode=9698298957680`.

The form fields are:

Nombre	Apellidos	
Papa	Noel	
Dorsal	Posición	Código de Barras
10	Portero	9698298957680

Buttons at the bottom: Crear, Limpiar, Volver, Generar Barcode.

Añadimos a Papá Noel como jugador. Y posteriormente somos redirigidos a jugadores.php.

Screenshot of a web browser showing the "Listado de Jugadores" (Player List) page. The URL is `localhost/futbolapptarea05wesderimantejera/public/c/jugadores.php`.

A message at the top says: "Jugador creado con éxito."

Table showing the list of players:

Nombre Completo	Posición	Dorsal	Código de Barras
Annette Barton	Central	5	
Papa Noel	Portero	10	
Alysha Dare	Central	21	
Chaya Schultz	Defensa	22	
Sabryna Boehm	Defensa	49	
Weston Daugherty	Lateral Derecho	54	
Kaela Cummerata	Lateral Izquierdo	63	

Volvemos a crear un nuevo jugador, pero esta vez vamos a asignar el dorsal de Papá Noel que era el 10, al pulsar el botón “Crear” mostrará la pantalla anterior.

	id	nombre	apellidos	dorsal	posición	barcode
<input type="checkbox"/>	6	Annette	Barton	5	Central	7101182431446
<input checked="" type="checkbox"/>	10	Papa	Noel	10	Portero	969879957830
<input type="checkbox"/>	11	Paterquito	Pérez	11	Portero	2917975326313
<input type="checkbox"/>	21	Alysha	Dere	21	Central	5765369582655
<input type="checkbox"/>	22	Chaya	Schultz	22	Defensa	7538993716798
<input type="checkbox"/>	49	Sabrina	Boehm	49	Defensa	2222826905316
<input type="checkbox"/>	54	Weston	Daugherty	54	Lateral Derecho	9017316493786
<input type="checkbox"/>	63	Kaela	Cumerata	63	Lateral Izquierdo	5431568939795
<input type="checkbox"/>	67	Scotty	O'Reilly	67	Lateral Derecho	1579441076329
<input type="checkbox"/>	79	Dorothy	Mueler	79	Defensa	9700459630018
<input type="checkbox"/>	85	Marcus	Wlosky	85	Portero	1181075406167
<input type="checkbox"/>	87	Schuyler	Padberg	87	Lateral Izquierdo	817759046258

Esta es la vista desde la base de datos.

The screenshot shows the phpMyAdmin interface for the 'practicuidad5' database. The 'jugadores' table is selected. A new row is being inserted with the following values:

Columna	Tipo	Función	Nulo	Valor
id	int(11)			137
nombre	varchar(40)			Annette
apellidos	varchar(60)			Barton
dorsal	int(11)	<input checked="" type="checkbox"/>		
posición	enum	<input type="checkbox"/>		Central
barcode	varchar(10)			7101102131446

At the bottom, there are buttons for 'Guardar y luego Volver', 'Previsualizar SQL', 'Reiniciar', and 'Continuar'. The status bar at the bottom right shows '22°C' and the date '25/10/2024'.

Entramos en el primer jugador y eliminamos el dorsal, haciendo clic en Nulo.

The screenshot shows the 'Listado de Jugadores' page. The table displays the following data:

Nombre Completo	Posición	Dorsal	Código de Barras
Annette Barton	Central	Sin Asignar	
Papa Noel	Portero	10	
Paterquito Pérez	Portero	11	
Alysha Dare	Central	21	
Chaya Schultz	Defensa	22	
Sabryna Boehm	Defensa	49	
Weston Daugherty	Lateral Derecho	54	

The status bar at the bottom right shows '22°C' and the date '25/10/2024'.

Ahora aparece con el texto: Sin Asignar.

FIN