DESARROLLO DE APLICACIONES WEB

12-2-2024

ROBERTO RODRÍGUEZ JIMÉNEZ

roberto.rodjim.1@educa.jcyl.es

Desarrollo Web en Entorno Servidor

Programación Orientada a Objetos en PHP

Tarea 05

# 

Contenido

[Tarea online DWES03 2](#_Toc159151763)

[1.- Descripción de la tarea 2](#_Toc159151764)

[Enunciado 2](#_Toc159151765)

[**Caso práctico** 2](#_Toc159151766)

[2.- Información de interés 5](#_Toc159151767)

[Recursos necesarios 5](#_Toc159151768)

[Consejos y recomendaciones 5](#_Toc159151769)

[3.- Evaluación de la tarea 5](#_Toc159151770)

[Criterios de evaluación implicados 5](#_Toc159151771)

[RESOLUCIÓN 7](#_Toc159151772)

[Dependencias obligatorias para instalar con Composer. 7](#_Toc159151773)

[Estructura del proyecto 8](#_Toc159151774)

[Código 9](#_Toc159151775)

[SQL 9](#_Toc159151776)

# Tarea online DWES03

**Título de la tarea**: Jugadores de fútbol.  
**Unidad 5**: Desarrollo de aplicaciones web PHP.  
**Ciclo formativo y módulo**: Desarrollo de Aplicaciones Web. Módulo DWES.  
**Curso académico**: 2023-24

## 1.- Descripción de la tarea

### Enunciado

### **Caso práctico**

Juan se va a encargar de desarrollar un proyecto para el equipo de futbol local. Para el proyecto va a tener que generar códigos de barras para identificar a cada jugador y hacer pruebas con datos ficticios para comprobar que todo funciona antes de entregar el proyecto.

Juan le propone a Carlos que le ayude.

-Carlos ¿Podríamos buscar en Packagist alguna aplicación los códigos de barras?

-Juan Efectivamente. Trabajaremos también con Blade, para ahorrar código y buscaremos alguna aplicación para generar los datos de prueba.

Deseamos diseñar una aplicación para gestionar los jugadores de un equipo de futbol. 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.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente con confianza mediaArchivo 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 utilizara el formato EAN-13, que utiliza 13 números).
* fzaninotto/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 carpetas 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 seran:

* "crearDatos.php" : Está página me genera los datos de ejemplo.
* "crearJugador.php" : está 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 esté vacios.
* "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")

Imagen que contiene Escala de tiempo

Descripción generada automáticamente

* "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.

Interfaz de usuario gráfica

Descripción generada automáticamente

* "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".

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

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".

## 2.- Información de interés

### 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.

### Consejos y 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 más 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.

## 

## 3.- Evaluación de la tarea

### Criterios de evaluación implicados

* Se han identificado las ventajas de separar la lógica de negocio de los aspectos de presentación de la aplicación.
* Se han analizado tecnologías y mecanismos que permiten realizar esta separación y sus características principales.
* Se han utilizado objetos y controles en el servidor para generar el aspecto visual de la aplicación Web en el cliente.
* Se han utilizado formularios generados de forma dinámica para responder a los eventos de la aplicación Web.
* Se han identificado y aplicado los parámetros relativos a la configuración de la aplicación Web.
* Se han escrito aplicaciones Web con mantenimiento de estado y separación de la lógica de negocio.
* Se han aplicado los principios de la programación orientada a objetos.
* Se ha probado y documentado el código.

|  |  |
| --- | --- |
| **Rúbrica de la tarea** | |
| Crea la estructura correcta del proyecto y se instalan correctamente las dependencias con **Composer**. | 2.5 Puntos. |
| Se crean correctamente, con los métodos necesarios, las clases "Conexion" y "Jugador" y se hace uso de "namespaces" y herencia. | 2 Puntos. |
| Crea correctamente los datos datos de Ejemplo con "fzaninotto/faker". | 1 Punto. |
| Las páginas de la carpeta "public" cargan las vistas apropiadamente y les pasan los parámetros necesarios. | 1 Punto. |
| El formulario para crear un jugador funciona correctamente, controlando posibles errores. | 1 Punto. |
| Se muestran y generan correctamente los códigos de barra. | 1 Punto. |
| Las vistas hacen uso de una plantilla común. | 0.5 Puntos. |
| La página de inicio comprueba si existen datos o no en la tabla "jugadores" y hace las redirecciones adecuadas. | 0.5 Puntos. |
| Se hace uso correcto de las sesiones para mostrar mensajes. | 0.25 Puntos. |
| El diseño es adecuado y el código está comentado. | 0.25 Puntos. |

## RESOLUCIÓN

### Dependencias obligatorias para 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).
* **fzaninotto/faker** Para genera datos aleatorios de prueba.
* **autoload** con optimizador.

En lugar de **fzaninotto/faker** se instala **fakerphp/faker**.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

fzaninotto/faker no se encuentra activo

{

    "name": "roberto/jugador",

    "description": "DWES Tarea05",

    "type": "project",

    "license": "GPL",

    "config": {

        "optimize-autoloader": true,

        "allow-plugins": {

            "kylekatarnls/update-helper": true

        }

    },

    "autoload": {

        "psr-4": { "Clases\\": "src/" }

    },

    "authors": [

        {

            "name": "Roberto Rodriguez",

            "email": "roberto.rodjim.1@educa.jcyl.es"

        }

    ],

    "require": {

        "philo/laravel-blade": "^3.1",

        "fakerphp/faker": "^1.23",

        "milon/barcode": "^6.0"

    }

}

Composer.json

### Estructura del proyecto

Texto

Descripción generada automáticamente

Estructura del proyecto

### 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"*

Para redirigir a la página de ***instalación*** o de ***listado*** se crea una instancia del Jugador y se llama a su método isEmptyJugadores que devolverá ***true*** o ***false*** en caso de que existan o no, jugadores registrados.

$j = new Jugador();

$load = ($j->isEmptyJugadores())? 'instalacion' : 'jugadores';

header("location:{$load}.php");

Parte de código de index.php

### Instalación

Controlador: instalacion.php

Vistas: vinstalacion.blade.php extends plantillas/plantilla1.blade.php

En instalacion.php se crean la instancia de Blade y se especifican los directorios que se van a usar.  
También se crean las variables con datos que van a pasarse a las vistas.

Finalmente se muestra en pantalla el renderizado de las vistas cargadas.

$views = '../views';

$cache = '../cache';

$titulo = 'DWES05 - Tarea';

$h1 = "Instalación";

$blade = new Blade([$views],$cache);

echo $blade->view()->make('vinstalacion',compact('titulo','h1'))->render();

Fragmento de código de instalacion.php

@extends('plantillas.plantilla1')

@section('titulo')

    {{ $titulo }}

@endsection

@section('h1')

    {{$h1}}

@endsection

@section('contenido')

<section class="container text-center">

    <a href="crearDatos.php"  class="btn btn-success">

        <i class="fa-solid fa-database me-2"></i> Instalar Datos de Ejemplo

    </a>

</section>

@endsection

Fragmento de código de vinstalacion.php

### jugadores.php

Esta página se carga desde *index.php* si existen jugadores en la base de datos o desde *crearjugador.php* al crear un jugador nuevo.

// Dependiendo de si hay contenido, se carga una página u otra

$load = ($j->isEmptyJugadores())? 'instalacion' : 'jugadores';

header("location:{$load}.php");

Redirección a jugadores.php desde index.php

// Redireccionamos al listado de jugadores

$url = 'jugadores.php?';

$url .= (empty($err)) ? 'create=ok' : 'err=' . $err;

header('location:' . $url);

Redirección a jugadores.php desde crearJugador.php después de recibir los datos del formulario

Jugadores va a cargar la vista *vjugadores* y esta, a su vez, extiende de *plantillas.plantilla1*.

jugadores.php comienza cargando el ***autoload***, a continuación, carga las clases necesarias haciendo referencia a los namespaces.

Se crea una instancia de Barcode y se establece la ruta a la caché.

require '../vendor/autoload.php';

use Philo\Blade\Blade;

use Clases\Jugador;

use Milon\Barcode\DNS1D;

$barcode = new DNS1D();

$barcode->setStorPath(\_\_DIR\_\_ . '/cache/');

$views = '../views';

$cache = '../cache';

$titulo = 'DWES05 - Tarea';

$h1 = 'Listado de Jugadores';

## Código

### SQL

-- Database: `practicaunidad5`

CREATE DATABASE IF NOT EXISTS `practicaunidad5` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci;

USE `practicaunidad5`;

-- --------------------------------------------------------

-- Table structure for table `jugadores`

DROP TABLE IF EXISTS `jugadores`;

CREATE TABLE `jugadores` (

  `id` int(11) NOT NULL,

  `nombre` varchar(40) NOT NULL,

  `apellidos` varchar(60) NOT NULL,

  `dorsal` int(11) DEFAULT NULL,

  `posicion` enum('Portero','Defensa','Lateral Izquierdo','Lateral Derecho','Central','Delantero') DEFAULT NULL,

  `barcode` varchar(13) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- Indexes for table `jugadores`

ALTER TABLE `jugadores`

  ADD PRIMARY KEY (`id`),

  ADD UNIQUE KEY `barcode` (`barcode`),

  ADD UNIQUE KEY `dorsal` (`dorsal`);

-- AUTO\_INCREMENT for table `jugadores`

ALTER TABLE `jugadores`

  MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT;

COMMIT;