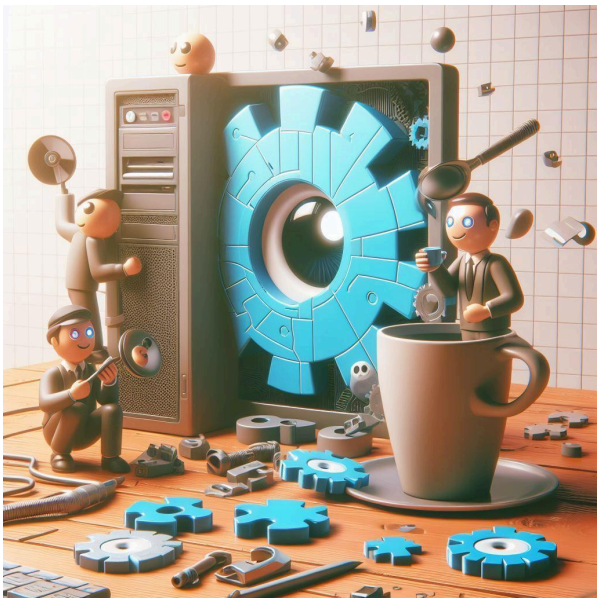

PRODUCTO 3

FP.064 - DESARROLLO BACK-END CON PHP, FRAMEWORK MVC Y GESTOR DE CONTENIDOS

Grupo 02 – bits & PHPieces

Consultor:

Borja Mulleras

Integrantes:

Carlos Espigol Flores
Esteban Toledo Barrios
Felix Ryan Pasadas Ellison

ÍNDICE

Contenido

1. Instalar en el hosting AWS proporcionado por el consultor si no se ha hecho ya en el Producto 1, el entorno Laravel.

Aunque hemos usado la misma imagen en Docker para crear el contenedor se han tenido que corregir las versiones con el fin de que funcione correctamente Laravel, que está en su versión 10 en AWS.

Para acceder al AWS tendremos que usar este comando:

ssh uocx1@fp064.techlab.uoc.edu -p 55000

pass: GMRFPtpR

Desde el terminal entramos en la carpeta public_html en AWS o web en el Docker y escribimos el siguiente comando:

composer create-project laravel/laravel:^10 producto3

Luego escribimos el comando:

composer install

Ahora corresponde dar permisos a las diferentes carpetas para que funcione:

sudo chmod -R 775 /var/www/html/producto3/storage

sudo chmod -R 775 /var/www/html/producto3/bootstrap/cache

sudo chmod -R 755 /var/www/html/producto3

Habilitamos el rewrite en apache

sudo a2enmod rewrite

En el archivo .env contiene las variables de entorno. Es donde hemos incluido las conexión a la BBDD y el acceso al servidor de correo, es mejor utilizar este archivo ya que en caso de usar el GITHUB este archivo no se sube y mantiene las contraseñas a salvo

.env
… DB_CONNECTION=mysql DB_HOST=mysql-db-prdto3 DB_PORT=3306 DB_DATABASE=wordpress1 DB_USERNAME=root DB_PASSWORD=root … MAIL_MAILER=smtp MAIL_HOST=mailpit MAIL_PORT=1025

```
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
MAIL_FROM_ADDRESS="hello@example.com"
MAIL_FROM_NAME="${APP_NAME}"
…
```

La configuración del acceso de database también se podría realizar en el fichero database.php

producto3/config/database.php

```
…
        'mysql' => [
            'driver' => 'mysql',
            'url' => env('DATABASE_URL'),
            'host' => env('DB_HOST', 'mysql-db-prdto3'),
            'port' => env('DB_PORT', '3306'),
            'database' => env('DB_DATABASE', 'wordpress1'),
            'username' => env('DB_USERNAME', 'root'),
            'password' => env('DB_PASSWORD', 'root'),
            'unix_socket' => env('DB_SOCKET', ''),
            'charset' => 'utf8mb4',
            'collation' => 'utf8mb4_unicode_ci',
            'prefix' => '',
            'prefix_indexes' => true,
            'strict' => true,
            'engine' => null,
            'options' => extension_loaded('pdo_mysql') ? array_filter([
                PDO::MYSQL_ATTR_SSL_CA => env('MYSQL_ATTR_SSL_CA'),
            ]) : [],
        ],
…
```

Los controles y modelos se pueden crear usando instrucciones en el terminal:

php artisan make:controller FormularioController : Crear un controlador

php artisan make:migration nombre_de_migracion : Crear una migración

php artisan make:middleware Authenticate : Crear un fichero middleware

php artisan make:component HotelSelect : Crear un componente

php artisan view:clear : Limpiar la cache de vistas en caso de hacer cambios y que no se reflejan.

php artisan route:clear : Limpiar la cache de Route en caso de hacer cambios y que no se reflejan.

2. Adaptar el producto 2 al Framework Laravel mejorando el aspecto gráfico.

En el producto anterior los archivos se colocaban en unas carpetas que simulaban el entorno MVC. Este sistema daba muchos problemas en las rutas de archivos y también en el volumen de ficheros generados.

Para Laravel 10, que es la versión instalada, el sistema es totalmente diferente. Los principales puntos son:

- Seguridad:
  - El código está separado de la carpeta desde donde accede el usuario final. dentro de “producto3” existe una carpeta llamada “public” que sólo almacena las carpetas de las librerías (“assets”) y el css. Además de estas carpetas está el fichero index.php que enlazará con Laravel y el archivo .htaccess que indica la seguridad de la carpeta “public”.
  - Session: Ya no hace falta poner session_start(); en todas las páginas ya que está incluido en Laravel.
    Se pueden crear utilizando el siguiente comando:

```
Session::put('id', $user->id_hotel);
Session::put('mail', $user->email);
Session::put('usertype', $user->Id_tipo_usuario);
Session::put('userroute', "hotel");
Session::put('loggin', 'on');
```

    Para recuperar la información podemos usar el siguiente comando:

```
$variable2 = Session::get('usertype');
$variable = Session::get('loggin');
```

  - La gestión del “loggin” se realiza a través de middleware, que está en la carpeta “producto3/Http/Middleware” donde se ubican los ficheros CheckAdmin,php, CheckHotel.php, CheckVehiculo.php y CheckViajero.php que controlarán los accesos a la aplicación. Por ejemplo:

producto3/app/Http/Middleware/CheckHotel.php

```
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Symfony\Component\HttpFoundation\Response;
use Illuminate\Support\Facades\Log;
class CheckHotel
{
    /**
     * Handle an incoming request.
```

```
     *
     * @param  \Closure(\Illuminate\Http\Request):
(\Symfony\Component\HttpFoundation\Response)  $next
     */
    public function handle(Request $request, Closure $next): Response
    {
        log::channel('mylog')->info('Pasando por CheckHotel');
        // Verifica si el usuario está autenticado
        if (!$request->session()->has('user')) {
            return redirect('/');
        }
        // Obtiene el usuario autenticado
        $user = $request->session()->get('user');
        log::channel('mylog')->info('Usuario autenticado: ' . $user->Id_tipo_usuario);
        // Verifica si el usuario es un hotel
        if ($user->Id_tipo_usuario != 5) {
            // El usuario no es un hotel, redirige a la página de inicio
            return redirect('/');
        }
        return $next($request);
    }
}
```

- Por otro lado el archivo “web.php” que está dentro de “producto3/routes” contiene todas las rutas a los archivos y la relación con el controlador.

  En este caso podemos encontrar varías líneas importantes:

  - Esta línea indica donde está el fichero del controlador, de esta forma está referenciado a lo largo del fichero.
    use App\Http\Controllers\AdminPanelController;
  - De esta forma se define la ruta de entrada a la aplicación.
    Route::get('/', HomeController::class)->name('home');
  - De esta forma se define un grupo que contendrá todas las rutas a ficheros/controladores
    Route::middleware(['checkAdmin'])->group(function () {
    …
    }

  Este sería un ejemplo de la parte de vehículo:

```
Route::middleware(['checkVehiculo'])->group(function () {
    Route::get('vehiculo', [VehiculoPanelController::class, 'index']);
    // ----------------------------------------------------------
    // Agrega aquí otras rutas que devuelvan vistas dentro de la carpeta 'vehiculo/'
    Route::get('vehiculo/listar', [VehiculoPanelController::class, 'index'])->name('vehiculo.listar');
    Route::get('vehiculo/itinerario/{tramo}/{fecha}/{conductor}',
[transfer_listarreservaController::class,
'listarReservasConductor'])->name('vehiculo.itinerario');
    Route::get('vehiculo/cambiar-datos', [VehiculoPanelController::class,
'cambiarDatos'])->name('vehiculo.cambio-datos');
    Route::post('vehiculo/cambiar-datos', [VehiculoPanelController::class,
'cambiarDatos'])->name('vehiculo.cambio-datos');
    Route::get('vehiculo/cambiar-contraseña', [VehiculoPanelController::class,
'cambiarContraseña'])->name('vehiculo.cambio-contraseña');
```

```
    Route::post('vehiculo/cambiar-contraseña', [VehiculoPanelController::class,
'cambiarContraseña'])->name('vehiculo.cambio-contraseña');
    // Agrega aquí otras rutas que devuelvan vistas dentro de la carpeta 'vehiculo/'
});
```

- Simplicidad: Podemos generar una plantilla para garantizar que todas nuestras páginas se verán de la misma forma. Esto se hace dentro de la carpeta "/producto3/resources/views/layouts/plantilla.blade.php".

  Para usar esta plantilla todos nuestros ficheros de la parte de "view" deben contener el texto ".blade.php" con el fin de que se active la plantilla.

  La plantilla contendrá unos "tags" que indicarán la parte que se ha de sustituir, de forma que en las siguientes páginas sólo marcaremos la parte del tag que se ha de actualizar con su información. Por ejemplo:

Plantilla.blade.php	index.blade.php

<!DOCTYPE html>
<html lang="en">
 	
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>**@yield('title')**</title>
  <link rel="stylesheet" href="{{ asset('assets/libraries/bootstrap.min.css') }}" id="bootstrap-css">
  <link rel="stylesheet" href="{{ asset('css/style.css') }}">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.7.2/font/bootstrap-icons.css" rel="stylesheet">
  {{-- No se encuentra --}}
  {{-- <link rel="stylesheet" href="{{ asset('css/style2.css') }}"> --}}
  <link rel="stylesheet" href="{{ asset('css/wizard.css') }}">
</head>
 	
<body>
…
  @yield('content')
 	
  bits & PHPieces - Producto 2 - FP064
  <script lang="javascript" src="{{ asset('assets/libraries/jquery-3.7.1.min.js') }}"></script>
  <script lang="javascript" src="{{ asset('assets/libraries/bootstrap.bundle.min.js')
	@extends('layouts.plantilla')
	@section('title', 'Menú de conductores / vehículos')
	
	@section('content')
	
	 Vehículo
	
	…
	
	@endsection

```
}}"></script>
    <script lang="javascript" src="{{
asset('assets/libraries/popper.min.js')
}}"></script>
    <script lang="javascript" src="{{
asset('assets/libraries/wizard.js') }}"></script>

</body>

</html>
```

- Componentes. Los componentes nos permiten agregar funcionalidades como si fueran html sin necesidad de anexar llamadas al controlador o líneas de código.

  El componente se define en “producto3/resources/views/components”. Por ejemplo este sería el de un combo para seleccionar el hotel:

```
<select class="form-select" name="{{ $name }}">
   @foreach ($hoteles as $hotel)
      <option value="{{ $hotel->id_hotel }}" {{ $selected == $hotel->id_hotel ? 'selected' : '' }}>
         {{ $hotel->NombreHotel }}
      </option>
   @endforeach
</select>
```

  En la carpeta “producto3/app/View/Components” se incluiría el código para que se muestre la información:

```
<?php
namespace App\View\Components;

use Closure;
use Illuminate\Contracts\View\View;
use Illuminate\View\Component;
use App\Models\TransferHotel;

class HotelSelect extends Component{
   public $selected;
   public $name;

   public function __construct($selected = null, $name = 'id_hotel')
   {
      $this->selected = $selected;
      $this->name = $name;
   }

   public function render()
   {
      $hoteles = TransferHotel::all(); // Obtener todos los hoteles desde la base de datos

      return view('components.hotel-select', [
         'hoteles' => $hoteles
      ]);
   }
}
```

En el código de la aplicación se puede utilizar de la siguiente forma:

producto3/resources/views/reservas/aeropuerto/crear_reservaaeropuerto.blade.php

```
…
<x-hotel-select :selected="0" name="Hotel_Destino" />
…
```

- Mejora en el acceso a la BBDD. Dentro de la carpeta “producto3/app/Models” se indican todos los modelos de datos para que cuando accedemos a la base de datos se pueda usar el **ORM Eloquent** pueda acceder rápidamente a ella. Por ejemplo:

```
<?php
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class TransferUsuariosTipo extends Model{
    use HasFactory;

    protected $table = 'transfer_usuarios_tipo';
    public $timestamps = false;

    protected $fillable = [
        'Descripcion',
        'Id_tipo_usuario',
    ];
}
```

Estos son unos ejemplos de cómo se haría la función de

Insertar

```
    public function store(Request $request)
    {
        try {
            log::channel('mylog')->info(json_encode($request->all()));
            $administrador = new TransferAdministrador();
            $administrador->Id_tipo_usuario = $request->tipoUsuario;
            $administrador->Username = $request->username;
            $administrador->nombre = $request->nombre;
            $administrador->Password = $request->password;
            $administrador->email = $request->email;
            $administrador->Apellido1 = $request->primerApellido;
            $administrador->Apellido2 = $request->segundoApellido;
            $administrador->Id_tipo_usuario = $request->tipoUsuario;
            $administrador->save();
            $administradores = TransferAdministrador::all();
            return redirect()->route('administrador.listaAdministradores')->with('success', 'Usuario
creado con éxito.');
        } catch (\Exception $e) {
```

```
            log::channel('mylog')->error('Error al crear el usuario: ' . $e->getMessage());
            return redirect()->route('administrador.listaAdministradores')->with('error', 'Error al crear
el usuario.');
        }
    }
```

Actualizar

```
    public function update(Request $request, $Id_usuario)
    {
        $administrador = TransferAdministrador::find($Id_usuario);
        if ($administrador) {
            $administrador->Username = $request->username;
            $administrador->nombre = $request->nombre;
            $administrador->Password = $request->password;
            $administrador->email = $request->email;
            $administrador->Apellido1 = $request->primerApellido;
            $administrador->Apellido2 = $request->segundoApellido;
            // $administrador->Id_tipo_usuario = $request->Id_tipo_usuario;
            $administrador->update();
            $administradores = TransferAdministrador::all();
            return redirect()->route('administrador.listaAdministradores')->with('success', 'Usuario
actualizado con éxito.');
        } else {
            return redirect()->route('administrador.listaAdministradores')->with('error', 'Usuario no
encontrado.');
        }
    }
```

Listar

```
    public function listaVehiculos()
    {
        log::channel('mylog')->info('Pasando por lista Vehículos');

        $vehiculos = TransferVehiculo::all();
        // return $administradores;
        return view('administrador.listaVehiculos', compact('vehiculos'));
    }
```

Eliminar

```
    public function delete($Id_usuario)
    {
        //Busca el administrador por el Id_usuario
        $administrador = TransferAdministrador::find($Id_usuario);
        //Verifica si el administrador existe
        if ($administrador) {
            //Elimina el administrador
            $administrador->delete();
            // Redirige al administrador a la lista de usuarios con un mensaje de éxito
            return redirect()->route('administrador.listaAdministradores')->with('success', 'Usuario
eliminado con éxito.');
        } else {
            // Redirige al administrador a la lista de usuarios con un mensaje de error
```

```
            return redirect()->route('administrador.listaAdministradores')->with('error', 'Usuario no
encontrado.');
        }
    }
```

- Limpieza en el código: Podemos usar instrucciones de Laravel 10 dentro del código lo que nos permite hacer bucles o estructuras condicionales. Por ejemplo:

Estructura condicional

```
@if (Session::get('usertype')!="6")
      <x-viajero-select :selected="0" name="emailreserva" />
@else
      <input name="emailreserva" id="emailreserva" type="mail" value="{{ Session::get('mail')
}}">
@endif
```

bucles

```
@foreach ($matrizresultado as $info)
    <tr>
        <td>{{ $info["localizador"] }}</td>
        <td>{{ $info["Descripción"] }}</td>
        <td>{{ $info["email_cliente"] }}</td>
        <td>{{ $info["fecha_reserva"] }}</td>
        …
@endforeach
```

3. Acceso Web a la aplicación

Aplicación Web: https://fp064.techlab.uoc.edu/~uocx1/producto3/public/

PHPMyadmin: http://fp064.techlab.uoc.edu/bbdd/

usuario BBDD: wordpress1

Password BBDD: DWD8Ds3l4dvXpjZH

Configuración FileZilla: La configuración que nos entregó el consultor es:

El servidor es fp064.techlab.uoc.edu

usuario: uocx1

Password: DWD8Ds3l4dvXpjZH

4. Link de Youtube con el video

Este es el link de Youtube mostrando el proceso de instalación y comprobación del funcionamiento.

https://youtu.be/_yebh5Ui4tY

5. GITHUB

El link de acceso a nuestro GITHub para el producto 3 es el siguiente:

https://github.com/CarlosEF2023/bits-PHPieces_Producto3.git

BIBLIOGRAFÍA

CSS Tutorial (w3schools.com): Tutorial de CSS para personalizar algunas partes de la aplicación.

PHP Tutorial (w3schools.com): Resolución de dudas con PHP.

jQuery Tutorial (w3schools.com): Envío de formularios por ajax y pulsar botones.

Bootstrap 5 Tutorial (w3schools.com): Realización del menú responsive y aplicar Bootstrap a la aplicación.

https://www.youtube.com/watch?v=KhIDv-ViHBU&list=PLZ2ovOgdI-kWShYbJSN5RiLzpQEm0nEVx : Patrón MVC

https://www.youtube.com/playlist?list=PLZ2ovOgdI-kWWS9aq8mfUDkJRfYib-SvF : Curso Laravel