

JUNIO DE 2021



# Proyecto fin de ciclo

REALIZADO POR

José Manuel Arrieta Soto

# **Índice**

Introducción .....	3
Descripción.....	4
Instalación .....	5
Guía de estilos .....	7
Diseño de base de datos .....	10
Desarrollo .....	13
Desarrollo servidor.....	13
Modelo .....	13
Vista .....	15
Controladores.....	17
Blade .....	18
Jetstream .....	18
Desarrollo cliente .....	19
Tailwindcss .....	19
Sweetalert2 .....	20
Livewire .....	20
Despliegue.....	27
Manual.....	29
Conclusiones .....	32
Listado de imágenes.....	33
Bibliografía y referencias.....	34

# Introducción

La idea principal surgió de una renovación de la página web del grupo puesto que llevaba varios años sin actualizarse y a día de hoy no era útil para nada.

Por ello hace 2 años surgió la idea de renovarla con WordPress a través de mi proyecto de grado de ASIR, este año he querido hacerlo diferente ya que al había estudiado nuevas tecnologías.

La necesidad era de pasar los datos de los socios a un sitio que fuera más accesible para cualquier monitor ya que estos datos están en papel y de ahí surgió la idea de que además de ser una página web simple se añadiera un espacio para los monitores donde pudieran tener guardados ciertos datos necesarios, como los datos de los socios, los campamentos que hayamos ido, los materiales que tenemos, etc.

Todo esto se iba a realizar con Laravel ya que era un framework que he estado trabajando y viendo cómo funciona, pero a la vez añadirle nuevas tecnologías que me hicieran más fácil el escribir el código.

## Descripción

La página web consiste en una parte de un rediseño de la misma la cual tendrá un menú con las mismas opciones que tenía hasta ahora excepto por un nuevo enlace que es el login de la aplicación.

Una vez hayamos iniciado sesión entraremos en la parte de administración de los monitores el cual tendrá un menú diferente hasta ahora visto para poder ver los datos que se encuentren en la base de datos en forma de tabla, cada tabla es asíncrona lo que al cambiar de numero de visualizaciones de la tabla, buscar algún elemento, editar, borrar o incluso crear, no necesitará que la página se recargue.

# Instalación

La instalación se puede hacer de diversas formas, yo voy a explicar la que he utilizado que es la más simple y rápida de hacer.

Lo primero para poder instalarla aplicación es que será necesario de un servidor de bases de datos mysql o mariadb, es indiferente.

Una vez tengamos el servidor deberemos de crear una base de datos que le daremos el nombre que queramos en mi caso la llamare proyecto y en el proyecto tendremos que ir al archivo .env que se encuentra en la carpeta principal.

En el deberemos de irnos a las líneas 10 a 15 en el cual tendremos que poner los datos de la base de datos, en mi caso el usuario y contraseña de la B.D. es root y el puerto de dicho servicio que es 3307 (por defecto es 3306).

**DB\_CONNECTION=mysql**

**DB\_HOST=127.0.0.1**

**DB\_PORT=3307**

**DB\_DATABASE=proyecto**

**DB\_USERNAME=root**

**DB\_PASSWORD=root**

Una vez hecho esto necesitaríamos un servidor web como apache o por defecto laravel ya viene con un comando que te crear un servidor. Antes de iniciar el servidor web tendremos que escribir el siguiente comando dentro de la ruta principal del proyecto

**php artisan migrate:fresh --seed**

Este comando dropea todas las tablas de la base de datos y las crea de nuevo, además con --seed activamos los seeder de laravel y le añadimos datos ficticios.

Después de esto si estamos en un servidor apache entraremos en localhost y el nombre de la carpeta y luego public.

<http://localhost/proyecto/public>

Si no queremos un servidor web con escribir el siguiente comando en la ruta principal del proyecto, creamos un servidor.

**php artisan serve**

y en <http://localhost:8000/> se abrirá un servidor web.

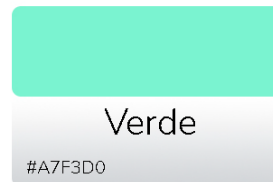
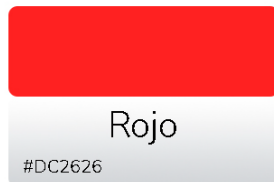
Para poder registrarse deberemos de escribir en la url register ya que por defecto no debe de registrarse nadie, pero para poder probarlo se ha dejado deshabilitado dicha opción.

# Guía de estilos

La guía de estilos y el wireframe vienen incluido en el proyecto.

La guía de estilos es la siguiente

## Colores



## Tipografía

Tamaño de fuente

Cuerpo del texto Nunito

### Heading 6xl - 60px

Párrafo xl - 24 x

Párrafo predeterminado - 16px

Texto pequeño - 12px

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

**Bold** *Italic* [Enlace](#)

1. Lista desordenada
2. Lista desordenada

- Lista ordenada
- Lista ordenada

## Botones



Imagen 1.1

Los colores elegidos son básicamente por los colores del grupo ya que son el púrpura y verde fuerte, por cual he decidido usar una gama de ellos en el que el verde es muy claro y el púrpura es un poco más clarito.

El color rojo lo he usado para botones que destaquen por su peligrosidad, como puede ser el botón de borrar.

Las letras son principalmente Negras, grises y púrpura y la tipografía es nunito, un estilo que se basa en su sencillez, fácil de leer y toque diferente a los estilos más predominantes.

Los tamaños de los textos son grandes para que cualquier persona pueda verlo sin necesidad de forzar la vista o tener que aumentar el tamaño del navegador.

El mock-up de la aplicación viene incluida en un archivo .xd que se puede abrir con Adobe XD, pero para una mejor visualización adjunto captura del inicio y la preinscripción en ordenador y la tabla en móvil.



Imagen 1.2.1

Esta es una captura de la página de preinscripción de la web del grupo scout Chiclana. El encabezado es idéntico al de la página de inicio. El título principal de la sección es 'PREINSCRIPCIÓN'. A la izquierda, hay un formulario con cuatro campos de entrada etiquetados: 'Nombre:', 'Apellidos:', 'Teléfono de contacto:' y 'Fecha de nacimiento:'. A la derecha del formulario, hay un icono de un documento con una pluma. Debajo del formulario, hay un texto que dice: 'Si quieres formar parte de alguna de las Secciones del Grupo Scout Chiclana ¡¡ Completa los datos que se piden a continuación !!'. Seguido de un texto que indica que se debe proporcionar información adicional de interés para el equipo de educadores. Debajo de este texto, hay un campo de entrada para la información adicional. A la izquierda de este campo, hay un logo del grupo scout. Debajo del campo de entrada, hay un texto que indica los condicionantes a tener en cuenta: 'CONDICIONANTES A TENER EN CUENTA: Por motivos organizativos el Grupo Scout Chiclana '77 tiene limitado el número de socios por sección, por lo que nos vemos obligados a atender a los criterios de prioridad que se detallan a continuación y según el orden indicado:'. Se listan cuatro criterios: 1- Año de nacimiento, 2- Familiares inscritos, 3- Paridad, 4- Fecha de Inscripción. Debajo de la lista, hay un botón verde que dice 'Enviar'. En la parte inferior, hay una barra de pie de página verde que contiene los enlaces 'Política de privacidad' y 'Términos de uso', el logo del grupo, y los datos de contacto: 'Grupo scout Chiclana 1977 700', '123 456 789' y 'gschiclana@scout.com'.

Imagen 1.2.2





Imagen 1.3.1

Imagen 1.3.2

Imagen 1.3.3

# Diseño de base de datos

En el proyecto viene incluido un archivo .mwb el cual se abre con mysql Workbench y nos muestra las tablas de la base de datos con sus relaciones.

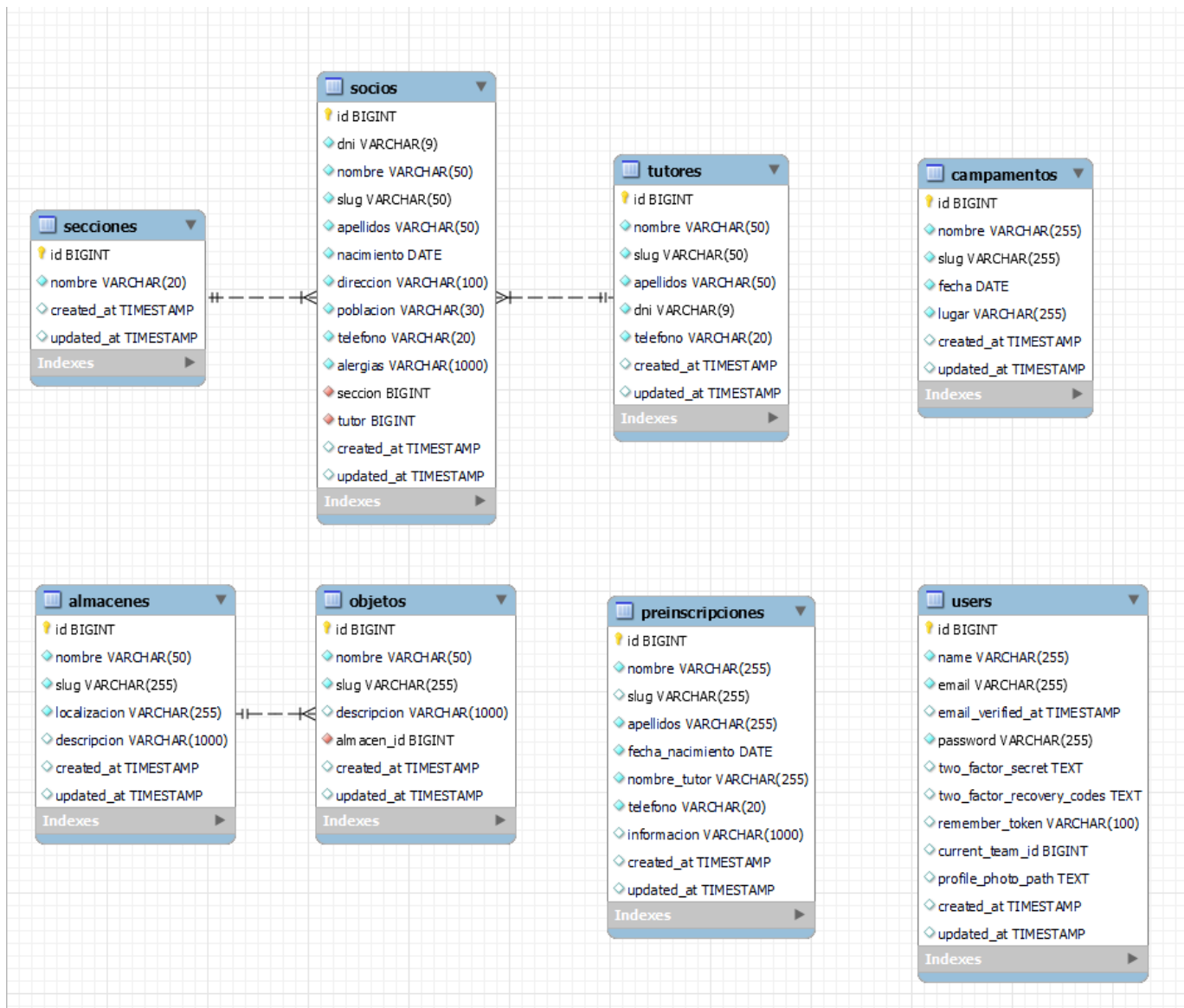


Imagen 2.1

El diseño principal consiste en 8 tablas secciones, socios y tutores las cuales están unidas en una relación 1 a muchos hacia socios, almacenes y objetos con otra de 1 a muchos y 3 tablas independientes para los campamentos, preinscripciones y el usuario.

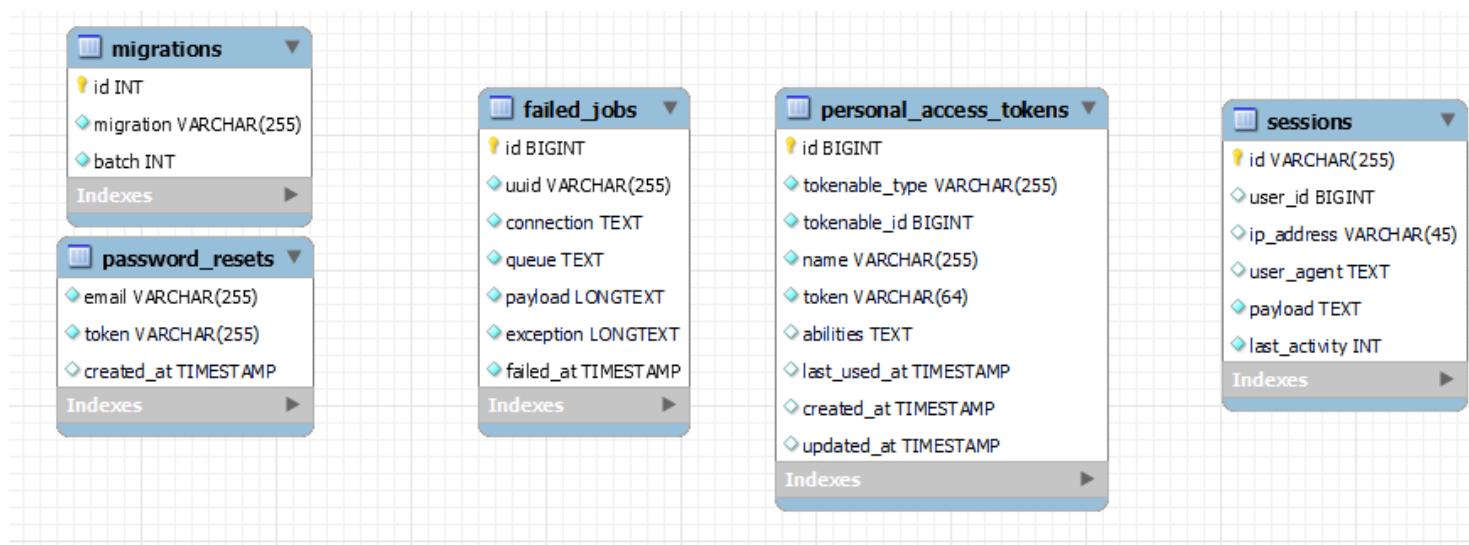


Imagen 2.2

Además, al utilizar jetstream, el cual explicaremos en la parte de desarrollo servidor, nos crea unas tablas adicionales para dicho funcionamiento.

Para ver como se crean estas tablas y sus datos deberemos de ir al proyecto y entrar en la carpeta database.

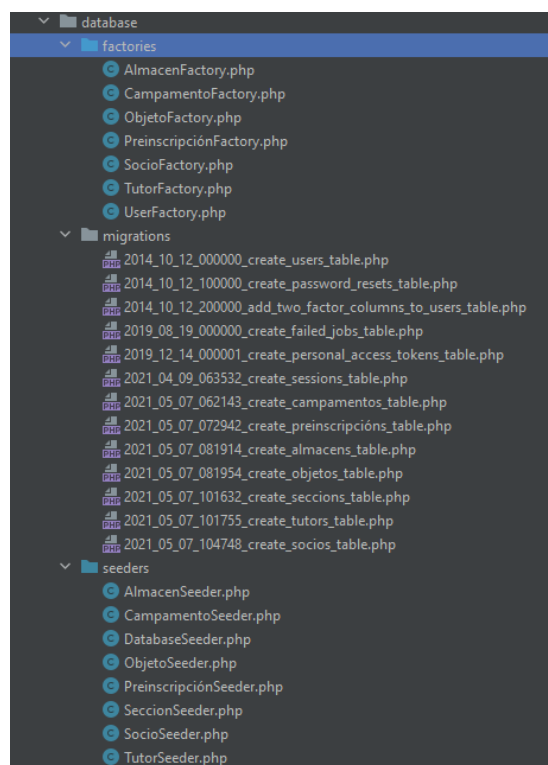


Imagen 2.3

En dicha carpeta encontramos 3 subcarpetas, migrations contiene la creación de las tablas, factories contiene los archivos para crear datos falsos de las diferentes tablas y seeders contiene los activadores para crear dichos datos falsos.

# Desarrollo

## Desarrollo servidor

Para el desarrollo en el servidor he utilizado Laravel, Blade y jetstream.

Laravel es uno de los frameworks de código abierto más fáciles de asimilar para PHP. Es simple, muy potente y tiene una interfaz elegante y fácil de usar.

Laravel usa el patrón mvc el cual separa la aplicación en 3 capas diferentes (modelo, vista, controlador). A continuación, explicaré donde se encuentra cada capa y su funcionalidad.

## Modelo

El modelo se encuentra en la ruta `App\Models`

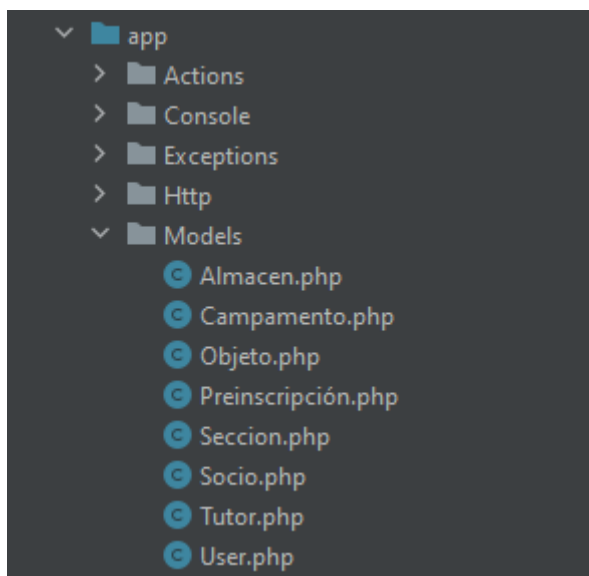


Imagen 3.1

Dichos archivos se encargan de todo lo relacionado con la base de datos. Consultas, actualizaciones, borrados, etc.

```
class Tutor extends Model
{
    use HasFactory;

    protected $guarded = ['status'];

    protected $table = 'tutores';

    public function socios(){
        return $this->hasMany('App\Models\Socio','tutor');
    }
}
```

Imagen 3.2

Este ejemplo de modelo de tutor vemos que extiende de la clase Model,

Para que podamos usar datos ficticios debemos de indicar en el modelo que usaremos HasFactory

Las variables \$guarded y \$table están en la clase Model, con guarded hacemos que el campo que tenga ese nombre en la tabla no pueda ser cambia externamente, mientras que con table hacemos que la tabla de la base de datos cambie de nombre.

Cuando un modelo tiene una relación con otro , se relacionan mediante un id en la tabla y un método que indica el tipo de relación con el otro modelo, en este caso tenemos una relación de muchos son socios y la clave foránea con la que se relaciona es tutor.

## Vista

La vista se encuentra en resources

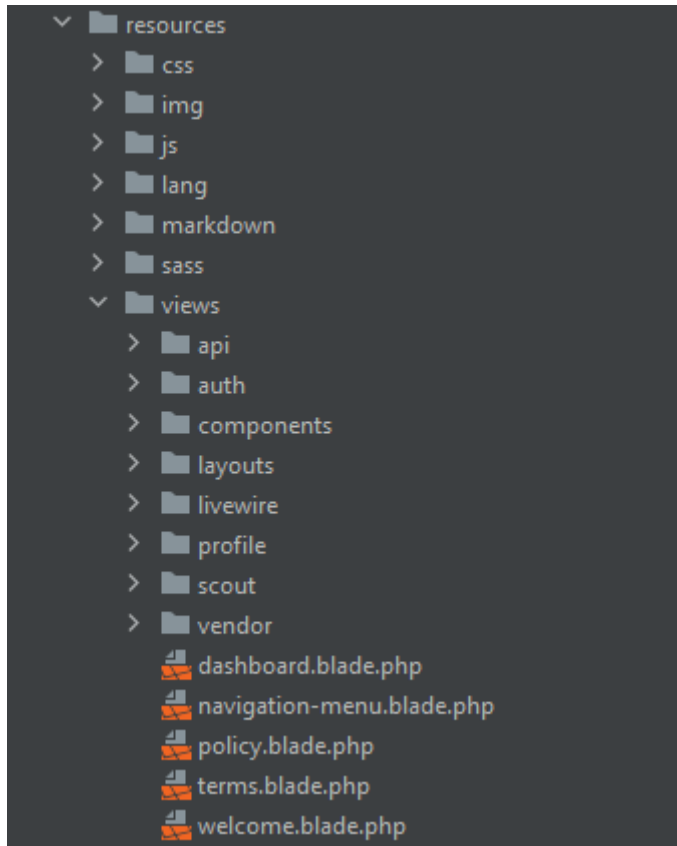


Imagen 3.3

En dicha carpeta se encuentra los archivos css, imágenes, javascript, sass y la carpeta views. En la vista encontramos los archivos necesarios para que veamos la página web.

```

<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="csrf-token" content="{{ csrf_token() }}">

    <title>Administración</title>

    <!-- Fonts -->
    <link rel="stylesheet"
href="https://fonts.googleapis.com/css2?family=Nunito:wght@400;600;700&display=swap">

    <!-- Styles -->
    <link rel="stylesheet" href="{{ mix('css/app.css') }}">
    <link rel="stylesheet" href="{{ asset('css/app.css') }}">

@livewireStyles

<!-- Scripts -->
    <script src="{{ mix('js/app.js') }}" defer></script>
    <script src="{{ asset('js/app.js') }}" defer></script>
</head>
<body>

@livewire('navigation-menu')

@if (isset($header))
    <header class="bg-white shadow">
        <div class="max-w-7xl mx-auto py-6 px-4 sm:px-6 lg:px-8">
            {{ $header }}
        </div>
    </header>
@endif

<main class="container text-center py-4">
    <h1 class="text-6xl text-purple-800">Bienvenido al panel de administración</h1>
    <p class="text-xl">Desde el menu que tienes arriba puedes acceder a las diferentes tablas de la base de
datos, las cuales puedes añadir, modificar y eliminar los datos que sean necesarios.</p>
</main>

@stack('modals')
</body>

```

Imagen 3.4

La vista contiene partes de html y sintaxis Blade que se visualizará para el usuario, esta vista es el panel de administración principal el cual contiene el menú de administración incluido con livewire, el cual se explicará en cliente y un header y texto.



## Controladores

Se encuentran en la carpeta App\Http\Controllers

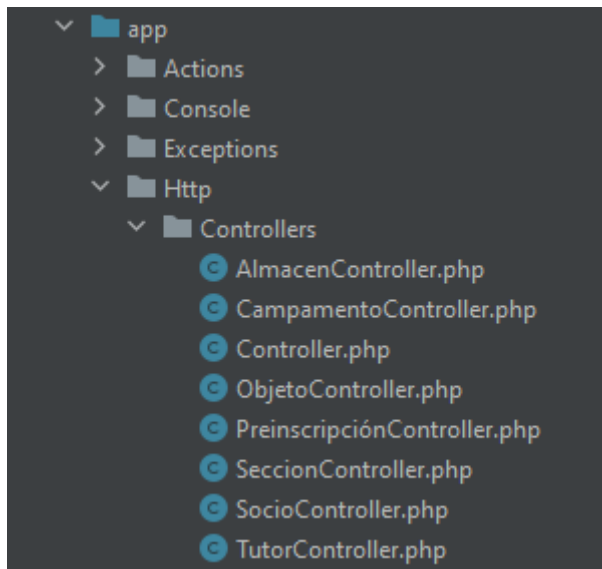


Imagen 3.5

Dichos archivos se encargan de conectar la vista con el modelo.

```
class PreinscripciónController extends Controller
{
    public function store(StorePreinscripcion $request)
    {
        $preinscripcion = Preinscripción::create($request->all());
        $slug = Str::slug($preinscripcion->nombre);
        $preinscripcion->update(['slug' => $slug]);
        return redirect()->route('index');
    }
}
```

Imagen 3.6

Los controladores extienden de la clase Controller.

Este controlador pertenece a las preinscripciones y cuando una vista llame al método store lo que hará es crear una preinscripción con el modelo y después redirigir a otra vista.

## Blade

Blade es un sistema de plantillas de laravel, el cual nos hace más fácil escribir código php, ya que contiene directivas las cuales no hacen más engorroso el programa y para poner código php simplemente con escribirlo entre `{{ }}` bastaría.

## Jetstream

Laravel Jetstream es un kit de inicio de aplicación bellamente diseñado para Laravel y proporciona el punto de partida perfecto para su próxima aplicación de Laravel. Jetstream proporciona la implementación para el inicio de sesión, registro, verificación de correo electrónico, autenticación de dos factores, administración de sesiones, API a través de Laravel Sanctum y funciones opcionales de administración de equipos.

La parte del Crud se realiza con componentes de livewire con lo que para ver dicha sección hay que ir hasta la parte de livewire en desarrollo cliente.

## Desarrollo cliente

Para el desarrollo en el lado del cliente he usado tailwindcss, jetstream, sweetalert2 y livewire.

Como ya mencionamos antes en la parte de servidor jetstream nos proporciona muchas utilidades y una de ellas son plantillas Blade ya creadas para diferentes cosas como botones, alertas, menús, registros, logins, etc.

Gracias a estos podemos editar dichas plantillas Blade y rediseñarlas a nuestro gusto.

### Tailwindcss

Tailwind CSS es un framework CSS que permite un desarrollo ágil, basado en clases de utilidad que se pueden aplicar con facilidad en el código HTML y unos flujos de desarrollo que permiten optimizar mucho el peso del código CSS.

Por eso en vez de escribir nuestro propio código CSS, con las clases de tailwind iremos desarrollando el CSS para nuestra página.

Para usar tailwindcss tan solo basta con llamar a la clase que necesitamos ya que al instalar jetstream se instala también tailwindcss

```
<main class="container text-center py-4">
  <h1 class="text-6xl text-purple-800">Bienvenido al panel de administración</h1>
  <p class="text-xl">Desde el menu que tienes arriba puedes acceder a las diferentes tablas de la base de
  datos, las cuales puedes añadir, modificar y eliminar los datos que sean necesarios.</p>
</main>
```

Imagen 4.1

Este ejemplo le da la clase contenedor al main, lo cual hace que el contenido esté centrado y no ocupe toda la pantalla. En el h1 le damos las clases text-6xl que sirve para aumentar el tamaño del texto un 3.25 rem y text-purple-800 que le da un color púrpura determinado.

## Sweetalert2

Sweetalert2 es una librería javascript que te permite crear ventanas emergentes con un diseño profesional y fácil de personalizar e implementar. La cual usaremos para indicar a los usuarios si ha hecho bien las acciones.

```
<script>
  Livewire.on('alerta', function (message) {
    Swal.fire(
      '¡Bien hecho!',
      message,
      'success'
    )
  })
</script>
```

Imagen 4.2

Este script de sweetalert2 hace que se active una ventana emergente cuando se emita el evento alerta y este hará que salga una alerta con el mensaje pasado.

## Livewire

Livewire es un framework de pila completa para Laravel que simplifica la construcción de interfaces dinámicas, sin dejar la comodidad de Laravel.

Para crear u componente de livewire solamente basta con escribir el siguiente comando dentro de la ruta del proyecto

**php artisan make:livewire NombreDelComponente**

Con esto habremos creado dos archivos, uno es el controlador y el otro la vista del componente.

Los controladores se crean en la siguiente ruta App\Http\Livewire y las vistas en resources\view\livewire

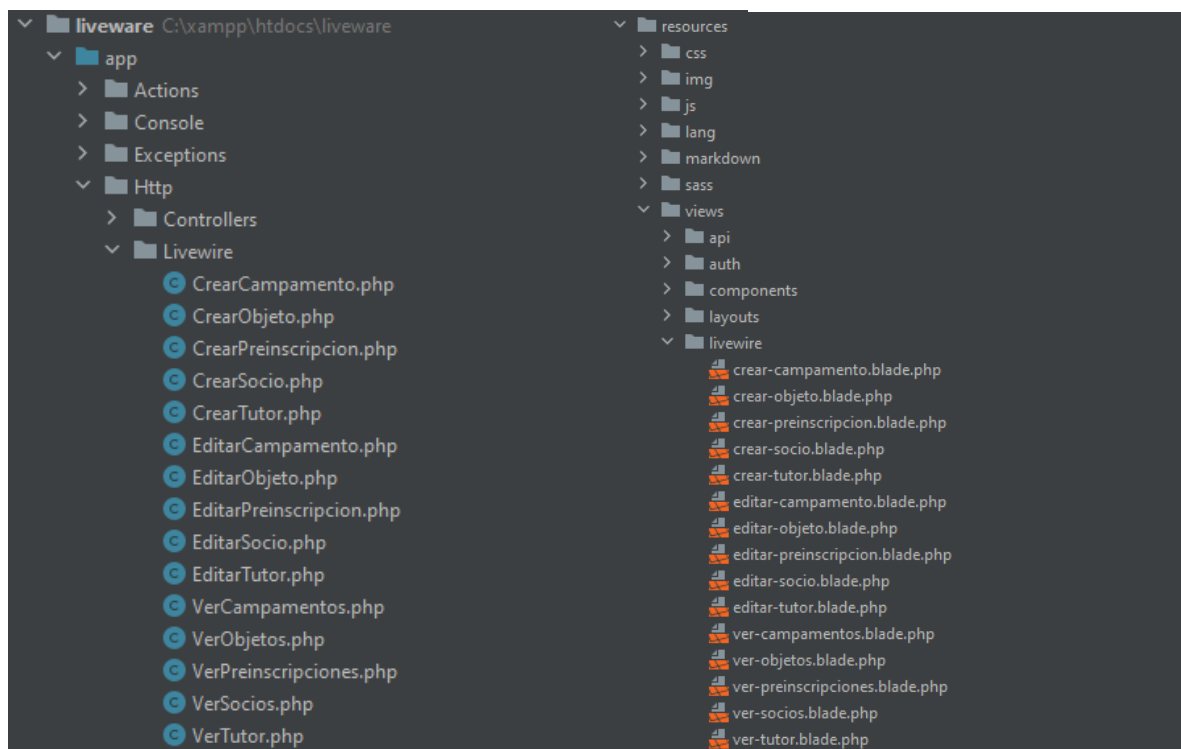


Imagen 4.3

Para poder ver cómo funciona un componente se va a mostrar un ejemplo de 1 de ellos.

```

class VerCampamentos extends Component
{
    use WithPagination;

    protected $listeners = ['renderAñadirCampamento' => 'render',
        'renderEditarCampamento' => 'render',
        'borrar'];

    public $buscador;
    public $ordenar = 'id';
    public $direccion = "asc";
    public $cantidad = 10;

    public function updatingBuscador() {
        $this->resetPage();
    }

    public function render()
    {
        $campamentos = Campamento::where('nombre', 'like', '%'.$this->buscador.'%')
            ->orWhere('lugar', 'like', '%'.$this->buscador.'%')
            ->orWhere('fecha', 'like', '%'.$this->buscador.'%')
            ->orderBy($this->ordenar,$this->direccion)
            ->paginate($this->cantidad);

        return view('livewire.ver-campamentos', compact('campamentos'));
    }

    public function ordenar($orden)
    {
        if ($this->ordenar == $orden){
            if ($this->direccion == 'desc'){
                $this->direccion = 'asc';
            } else {
                $this->direccion = 'desc';
            }
        }
        else {
            $this->ordenar = $orden;
        }
    }

    public function borrar(Campamento $campamento){
        $campamento->delete();
    }
}

```

Imagen 4.4

Este ejemplo se trata del controlador de la visualización de los campamentos, el cual contiene 5 atributos y 4 métodos, el método principal es render el cual se ejecuta cada vez que cambiemos algún atributo de la clase.

Dicho método selecciona todos los campamentos o con lo que hayamos puesto en el buscador, ordenados por el id, de forma descendente y con una paginación de 10 elementos por defecto.

Para poder hacer una paginación necesitaremos que use WithPagination

Los atributos son \$buscador, que guarda lo que vayamos escribiendo en el buscador, ordenar que se utiliza para ordenar por el campo que seleccionemos, \$direccion para indicar su ordenándolos de forma ascendente o descendente y \$cantidad que será el número de elementos que se muestre por página.

Aparte está el atributo protegido \$listener que es un array asociativo, el cual como clave tiene el nombre del evento que recibe y como valor el método que ejecutará cuando lo escuche.

Después tenemos 3 métodos.

El primero updatingBuscador, sirve para actualizar la paginación para que en caso de estar en la última página y se borre o busque un elemento y ya no tenga tantas páginas como antes, se reseteen

El segundo es ordenar, dicho método ordena según el parámetro que le hayamos pasado, y si hubiéramos seleccionado el mismo orden lo que cambiaríamos en la dirección, es decir ascendente o descendente

El último de ellos es el método borrar que se le pasa un elemento y este se borra de la base de datos.

```
<div class="mt-10">  
<div class="flex flex-col">
```

```

<div class="-my-2 overflow-x-auto sm:mx-6 lg:mx-8">
  <div class="py-2 align-middle inline-block min-w-full sm:px-6 lg:px-8">
    <div class="shadow overflow-hidden border-b border-gray-200 sm:rounded-lg">
      <div class="px-6 py-4 flex items-center">
        <div class="flex items-center">
          <span>Mostrar</span>
          <select wire:model="cantidad" class="mx-2 form-control">
            <option value="10">10</option>
            <option value="25">25</option>
            <option value="50">50</option>
            <option value="100">100</option>
          </select>
          <span>Entradas</span>
        </div>
        <x-jet-input class="flex-1 mx-4" placeholder="Escriba el campamento que esté buscando"
          wire:model="buscador" type="text"/>
        @livewire('crear-campamento')
      </div>
      @if($campamentos->count())
        <table class="min-w-full divide-y divide-gray-200">
          <thead class="bg-green-100">
            <tr>
              <th scope="col"
                class="cursor-pointer px-6 py-3 text-left text-xs font-medium text-purple-600
uppercase"
                wire:click="ordenar('id')">
                Id
              </th>
              <th scope="col"
                class="cursor-pointer px-6 py-3 text-left text-xs font-medium text-purple-600
uppercase"
                wire:click="ordenar('nombre')">
                Nombre
              </th>
              <th scope="col"
                class="cursor-pointer px-6 py-3 text-left text-xs font-medium text-purple-600
uppercase"
                wire:click="ordenar('lugar')">
                Lugar
              </th>
              <th scope="col"
                class="cursor-pointer px-6 py-3 text-left text-xs font-medium text-purple-600
uppercase"
                wire:click="ordenar('fecha')">
                Fecha
              </th>
              <th scope="col"
                class="cursor-pointer px-6 py-3 text-left text-xs font-medium text-purple-600
uppercase">
              </th>
            </tr>
          </thead>
          <tbody class="bg-white divide-y divide-purple-600">
            @foreach($campamentos as $campamento)
              <tr>
                <td class="px-6 py-4 ">
                  {{ $campamento->id }}
                </td>
                <td class="px-6 py-4 ">
                  {{ $campamento->nombre }}
            </tr>
          </tbody>
        </table>
      </if>
    </div>
  </div>
</div>

```



```

        </td>
        <td class="px-6 py-4 ">
            {{ $campamento->lugar }}
        </td>
        <td class="px-6 py-4 ">
            {{ $campamento->fecha }}
        </td>
        <td class="px-6 py-4 flex float-right">
            @livewire('editar-campamento', ['campamento' => $campamento ],
key($campamento->id))
            <x-jet-danger-button wire:click="$emit('borrarCampamento', {{ $campamento->id }})"
                                class="ml-2"><i class="far fa-trash-alt"></i>
            </x-jet-danger-button>
        </td>
    </tr>
</tbody>
</table>
@else
    <div class="px-6 py-4">
        No existe ningún campamento
    </div>
@endif

@if($campamentos->hasPages())
    <div class="px-6 py-4">
        {{ $campamentos->links() }}
    </div>
@endif
</div>
</div>
</div>
</div>

@push('js')
<script src="sweetalert2.all.min.js"></script>
<script>
    Livewire.on('borrarCampamento', idCampamento => {
        Swal.fire({
            title: '¿Estás seguro?',
            text: "¡No podrás revertir este cambio!",
            icon: 'warning',
            showCancelButton: true,
            confirmButtonColor: '#3085d6',
            cancelButtonColor: '#d33',
            confirmButtonText: 'Si, borrarlo'
        }).then((result) => {
            if (result.isConfirmed) {
                Livewire.emitTo('ver-campamentos', 'borrar', idCampamento);
                Swal.fire(
                    '¡Borrado!',
                    'El campamento ha sido eliminado.',
                    'success'
                )
            }
        })
    })
</script>

```

```
@endpush  
</div>
```

Imagen 4.5

Ahora veremos su vista, en dicha vista podemos apreciar que todo está contenido en un div que dentro contiene principalmente 1 div con el buscador, el componente de crear un campamento y un select con el número de elementos a mostrar y después la tabla en la que se muestra el elemento, en este caso los campamentos

Dentro del div podemos ver como se llaman a los componentes de livewire, para ellos con la directiva de blade `@livewire` llamaremos a la vista de dicho componente

Además, cuando queremos enlazar un atributo de la clase del componente con un input, select o textarea, etc usamos `wire:model=""` y entre comillas el nombre del atributo así cada vez que seleccionemos o añadamos algo dicha variable se actualiza.

Dentro de la tabla tenemos la cabecera que en cada celda tenemos una acción de clicar `wire:click=""` y dentro de las comillas llamaremos al método correspondiente junto con sus parámetros.

En el cuerpo de la tabla con un bucle `foreach` recorreremos la variable que hemos pasado a la vista y creamos las celdas de cada fila de la tabla, la última celda contiene un botón de borrar que al hacer click emite un evento llamado borrar campamento que será recogido por el script de `sweetalert2` el cual al seleccionar que queremos borrar el campamento emitirá un evento hacia el propio componente y este actualizará en render después de haber borrado el campamento. También tenemos otro componente que se usa para editar el campamento el cual le indicamos la vista y luego le pasamos los datos del campamento, para que estos puedan ser editados.

Los componentes para crear y editar campamentos abren un modal (con los datos rellenos en el caso de editar) el cual contiene un formulario con los campos necesarios, en dichos controladores tienen las reglas para que el formulario se envíe y para que en caso de no estar correcto salte una plantilla de blade con el error.

Dentro del controlador de crear hay una función que crea el campamento en este caso y en el controlador de editar otra función que actualiza todos los campos pasados.

# Despliegue

Como ya comentamos en la instalación nuestro despliegue se hará con una base de datos mysql o mariadb y con los comandos artisan de laravel o un servidor web

Pero también se puede desplegar con Docker, cabe destacar que docker-compose no funciona muy bien debido a factores externos a nosotros ya que el propio contenedor se intenta actualizar y puede llegar a fallos o tarda mucho en recargar los archivos que modifiquemos.

```
version: '3.7'

services:
  servidor:
    image: fjortegan/dwes:laravel
    stdin_open: true # docker run -i
    tty: true        # docker run -t
    ports:
      - "80:80"
    # development
    volumes:
      - ./var/www/html/

  db:
    image: mariadb
    ports:
      - "3308:3306"
    volumes:
      - ./db-data:/var/lib/mysql/
    environment:
      MYSQL_DATABASE: laravel
      MYSQL_USER: admin
      MYSQL_PASSWORD: laravel
      MYSQL_ROOT_PASSWORD: pestillo

  phpmyadmin:
    image: phpmyadmin
    ports:
      - "8080:80"
    environment:
      - PMA_ARBITRARY=1
```

Imagen 5.1

Este docker-compose se compone de 2 servicios, un servidor mysql y un servidor web.

Para el servidor web creamos un volumen con todos los archivos de laravel y los pasamos al servidor a la ruta /var/www/html

Luego en la base de datos pondremos los puertos del cliente en 3306 ya que está cambiado por tener un servidor mysql en el propio ordenador, si se diera el mismo caso no haría falta cambiar los puertos, luego le daremos los datos de mysql que están por defecto

Además, tenemos una imagen de phpmyadmin por si queremos entrar en la base de datos.

Para levantar el proyecto con escribir **docker-compose up** en la ruta inicial de laravel bastaría. Y luego seguir los pasos de la instalación para migrar los datos desde el servidor web.

Una vez migrado los datos accedemos a localhost y veríamos el proyecto

# Manual

Cuando entramos en la página principal veremos un menú con las diferentes opciones que tiene un usuario normal, las cuales son 4 páginas estáticas sobre información del grupo y 1 formulario para la preinscripción de los niñ@s.

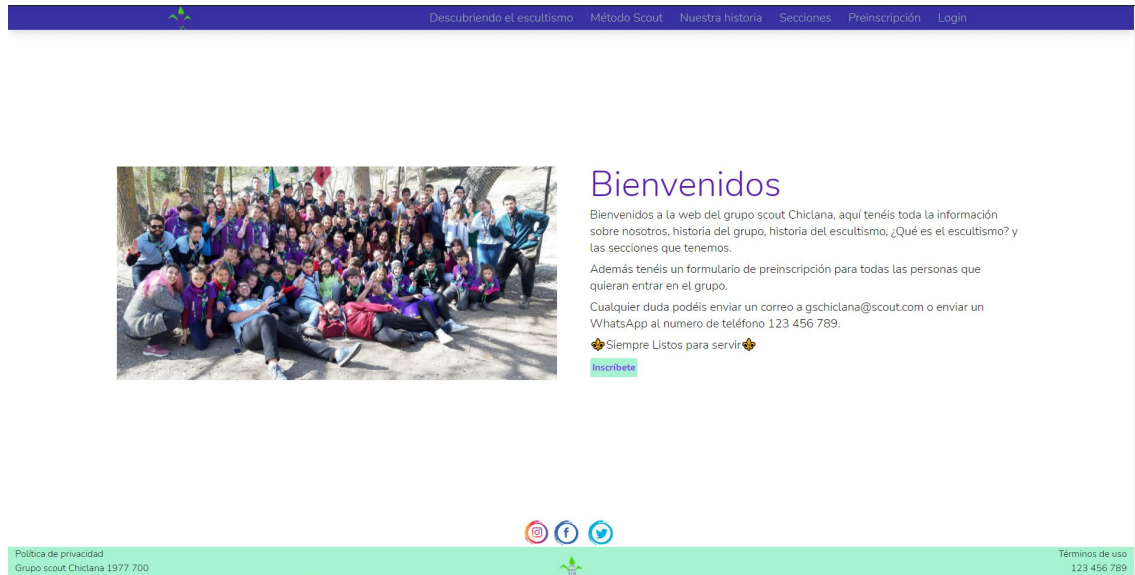



Imagen 6.1

The screenshot displays the 'Preinscripción' form. The navigation bar is identical to the previous page. The form title 'Preinscripción' is centered. Below it, a note states: 'Si quieres formar parte de alguna de las Secciones del Grupo Scout Chiciana (Completa los datos que se piden a continuación!)'. The form is divided into two sections: 'Información Personal' and 'Información adicional'. The personal section includes input fields for 'Nombre:', 'Apellidos:', 'Fecha de nacimiento:' (with a date picker), 'Teléfono de contacto:', and 'Nombre del tutor/a:'. The additional information section has a large text area for 'Cualquier información que pueda ser del interés para el equipo de educadores, así como aquella que deba ser tenida en cuenta de forma especial según los criterios de selección'. Below this is a section titled 'CONDICIONANTES A TENER EN CUENTA' with a list of criteria: 'Año de nacimiento', 'Familiares inscritos', 'Pertenencia', and 'Fecha de inscripción'. A green 'Enviar' button is at the bottom of the form. The footer is the same as in the previous image.

Imagen 6.2

La última opción del menú es un inicio de sesión para los monitores que una vez se logueen entraran en el apartado de administración.

Email

Password

☐ Recuerdame


[ENTRAR](#)

Imagen 6.3



Imagen 6.4

Dicho apartado contiene un nuevo menú en el cual saldrán las diferentes tablas de la base de datos.


[Inicio](#)
[Campamentos](#)
[Preinscripciones](#)
[Materiales](#)
[Socios](#)
[Tutores](#)
[Registro](#)

Grupo Scout Chiciana

Mostrar 10 Entradas

Escriba el campamento que esté buscando

AÑADIR NUEVO CAMPAMENTO

ID	NOMBRE	LUGAR	FECHA	
1	Bette Wilkinson	Elmville	1986-08-18	<div>EDITAR</div> <div></div>
2	Ms. Thora Gibson V	Greenchester	2008-02-17	<div>EDITAR</div> <div></div>
3	Bruce Pfannerstill	East Cristina	1973-12-14	<div>EDITAR</div> <div></div>
4	Dr. Jennyfer Kuhn Jr.	Lake Cleta	1998-01-23	<div>EDITAR</div> <div></div>
5	Andy Bartoletti	Buddyport	1980-12-20	<div>EDITAR</div> <div></div>
6	Zetta Jast DVM	Zulaland	1975-06-17	<div>EDITAR</div> <div></div>
7	Mrs. Courtney Fadel III	South Lawsontown	2003-10-25	<div>EDITAR</div> <div></div>
8	Kiarra Will	Reillyland	1981-09-11	<div>EDITAR</div> <div></div>
9	Dr. Florine Kihn	Lillianaport	2002-07-31	<div>EDITAR</div> <div></div>
10	Mr. Dennis Zieme	South Noemieborough	1976-12-18	<div>EDITAR</div> <div></div>

Showing 1 to 10 of 50 results

<

1

2

3

4

5

>

Imagen 6.5

Cada tabla tiene la opción de mostrar diferentes números de entradas y poder navegar por las diferentes páginas de la tabla, poder buscar algo en concreto, crear un nuevo elemento y borrar o editar un elemento seleccionado.

# Conclusiones

Comparado con la idea inicial, la parte de administración está mucho más completa y mejor diseñada ya que para cada tabla de la base de datos hacía falta varias páginas, una para ver, crear, editar y con livewire he desarrollado todo en la misma página.

Además, los menús han sido modificados en cuanto a estilo ya que en menú de administración era muy parecido al de un usuario normal, pero para diferenciarlo del otro decidí darle otra apariencia.

Aparte la gran mayoría de aplicaciones que se han usado son totalmente nuevas con respecto a lo dado en clases pues que solo se ha usado laravel.

Jetstream, Blade, tailwindcss, sweetalert2 y livewire con aplicaciones que he ido aprendiendo por mí mismo y se han utilizado con el propósito de mostrar que se pueden hacer más cosas aparte de lo que se nos enseñe.

Como mejoras para un futuro sería añadirle más tipos de datos que necesitemos, crear varios roles para que solo hubiera 1 administrador que controle todo y los demás solo puedan utilizar las tablas que están en el menú.



# Listado de imágenes

Imagen 1.1 Guía de estilos

Imagen 1.2.1 Mock-up ordenador inicio de web

Imagen 1.2.2 Mock-up ordenador preinscripción

Imagen 1.3.1 Mock-up móvil crear elemento

Imagen 1.3.2 Mock-up Móvil buscador

Imagen 1.3.3 Mock-up Móvil tabla

Imagen 2.1 Diagrama UML Parte 1

Imagen 2.2 Diagrama UML Parte 2

Imagen 2.3 Ruta las migraciones, factories y seeder de Laravel

Imagen 3.1 Ruta modelos de Laravel

Imagen 3.2 Ejemplo de modelo

Imagen 3.3 Ruta vistas de laravel

Imagen 3.4 Ejemplo de vista

Imagen 3.5 Ruta controladores de Laravel

Imagen 3.6 Ejemplo de controlador

Imagen 4.1 Ejemplo de Tailwindcss

Imagen 4.2 Ejemplo de sweetalert2

Imagen 4.3 Ruta de vistas y controladores de Livewire.

Imagen 4.4 Ejemplo de controlador de Livewire

Imagen 4.5 Ejemplo de vista de Livewire

Imagen 5.1 Archivo docker-compose

Imagen 6.1 Página de inicio

Imagen 6.2 Formulario de Preinscripción

Imagen 6.3 Inicio de sesión

Imagen 6.4 Panel de administración

Imagen 6.5 Página de campamentos

# Bibliografía y referencias

*Codersfree.* (s. f.). Codersfree. <https://codersfree.com/>

*Font Awesome.* (s. f.). Font Awesome. <https://fontawesome.com/>

*Installation - Laravel - The PHP Framework For Web Artisans.* (s. f.). Laravel.

<https://laravel.com/docs/8.x>

*Introduction | Laravel Jetstream.* (s. f.). Jetstream.

<https://jetstream.laravel.com/2.x/introduction.html>

*Laracast.* (s. f.). Laracast. <https://laracasts.com/>

*Livewire.* (s. f.). Livewire. <https://laravel-livewire.com/>

*Stack Overflow - Where Developers Learn, Share, & Build Careers.* (s. f.). Stack

Overflow. <https://stackoverflow.com/>

*SweetAlert2.* (s. f.). Sweetalert2. <https://sweetalert2.github.io/>

*Videos Codersfree.* (s. f.). Youtube Codersfree.

<https://www.youtube.com/channel/UCWuyqD6Pm70GwjWtENF5XJA>