# Documentación del Proyecto Laravel

## Índice

# Resumen del Proyecto

Este es un proyecto web desarrollado con **Laravel** para la gestión de eventos en una comunidad tipo **caravana**. La aplicación permite a los usuarios:

- **Registrarse e iniciar sesión** de forma segura utilizando **Laravel Sanctum**.
- Crear, ver, editar y eliminar **eventos** mediante una interfaz CRUD completa.
- Cada evento incluye información como nombre, descripción, fecha y ubicación.

El objetivo principal es facilitar la organización de actividades dentro de una comunidad móvil o viajera, brindando una plataforma simple y segura para gestionar eventos.

# Estructura MVC

A continuación se describe cómo está estructurado el proyecto según el patrón **Modelo - Vista - Controlador (MVC)**.

---

## Modelos (Models)

- **User.php**
  - Representa a los usuarios registrados.
  - Campos principales:

```
protected $fillable = [
    'name',
    'email',
    'email_verified_at',
    'password',
    'remember_token'
  ];
```

- **Evento.php**
  - Representa los eventos creados por los usuarios.
  - Campos principales:

```
protected $fillable = [
    'nombre',
    'marca',
    'fecha',
    'hora',
    'lugar_evento',
    'tipo_evento',
    'coste_evento',
    'aforo',
    'coste_unitario',
    'enlace',
    'documentacion',
    'texto_invitacion',
    'imagen'
  ];
```

## Relaciones en el modelo Evento

- **invitados()**
  - Tipo: `belongsToMany`
  - Un evento puede tener **muchos conductores** como invitados.
  - Relación con el modelo `Conductor`.
  - Usa la tabla pivote `evento_conductor` con las claves: `evento_id` y `conductor_id`.

```
public function invitados()
{
    return $this->belongsToMany(Conductor::class, 'evento_conductor', 'evento_id',
'conductor_id');
```

```
}

public function marcas()
{
    return $this->belongsToMany(Marca::class, 'eventos_marca', 'evento_id',
'marca_id');
}
```

- **Conductor.php**
- Representa los invitados creados por los usuarios.
    - Campos principales:

```
protected $fillable = [
    'cif',
    'nombre',
    'apellido',
    'email',
    'telefono',
    'empresa',
    'vehiculo_prop',
    'vehiculo_emp',
    'intolerancia',
    'preferencia',
    'carnet',
    'etiqueta',
    'kam',
    'asiste'
  ];
```

### Relaciones en el modelo Conductor

- **eventos()**
    - Tipo: `belongsToMany`
    - Un conductor puede estar **en varios** eventos.
    - Relación con el modelo `Evento`.
    - Usa la tabla pivote `evento_conductor` con las claves: `conductor_id` y `evento_id`.

```
public function eventos()
    {
        return $this->belongsToMany(Evento::class, 'evento_conductor',
'conductor_id', 'evento_id');
    }
```

- **EventoConductor.php**
- Representa tabla intermedia entre modelos Evento y Conductor.
    - Campos principales:

```
protected $fillable = [
    'evento_id', //Apunta al id del evento.
    'conductor_id' //Apunta al id del conductor.
  ];
```

## Relaciones en el modelo EventoConductor

- **invitados()**
  - Tipo: `hasMany`
  - Un evento puede tener varios conductores asociados.
  - Relación con el modelo `Conductor`.

```
public function invitados(){
        return $this->hasMany(Conductor::class);
    }
```

- **EventosMarca.php**
- Representa tabla intermedia entre modelos Evento y Marcas.
  - Campos principales:

```
protected $fillable = [
    'evento_id', //Apunta al id del evento.
    'marca_id' //Apunta al id de la marca.
  ];
```

- **Marca.php**
- Representa los nombres de las marcas que tengo guardados en la BD.
  - Campos principales:

```
protected $fillable = [
    'nombre'
  ];
```

## Relaciones en el modelo Evento

- **eventos()**
  - Tipo: `belongsToMany`

    - Una marca puede estar asociada cuando se crea el evento.
    - Relación con el modelo `Evento`.
    - Usa la tabla pivote `eventos_marca` con las claves: `marca_id` y `evento_id`.

```
public function eventos(){
        return $this->belongsToMany(Evento::class , 'eventos_marca' , 'marca_id' , 'evento_id');
    }
```

- **TipoEvento.php**
- Representa tabla del tipo de evento que tengo en mi BD
    - Campos principales:

```
protected $fillable = [
    'nombre'
  ];
```

## Vistas (Views)

Esta sección describe la estructura y organización de las vistas dentro del proyecto.

**VISTAS EVENTOS**

`dashboard.blade.php`

- **Propósito**: Pagina de inicio al entrar.
- **Datos requeridos**: For each con distintos campos creados para mostrar todo el contenido (se pasan en el controlador).
- **Acciones**: La vista dashboard esta comentada para que puedas leer el contenido de cada una de sus partes , a continuacion te explico lo mas importante.
- **Extras**: La vista cuenta con comentarios donde se especifica concretamente cada parte del codigo , de tal manera que se vea mas legible.

### Vista con código JS

A continuación, se muestra un ejemplo de cómo se puede incluir una funcion en JavaScript dentro de una vista Blade para limpiar y refrescar la pagina

**Código Blade + JavaScript**

```
<!-- CONTENEDOR DE BOTONES (FORMULARIO) -->
    <form class="d-block flex-wrap col-xl-10" action="{{ route('eventos.show') }}" method="GET">
```

```
        <div class="d-flex flex-wrap col-lg-12 align-items-center">
            <div class="col-12 col-lg-6">
            //Campo de busqueda ( se usa para el bsucador).
                <input class="form-control mb-3" id="nombre" name="buscador"
type="text" placeholder="Introducir nombre y otro campos">
            </div>
            <div class="col-12 col-lg-6 d-flex gap-2 justify-content-between">
                <button type="submit" class="btn_color txt col-4 col-sm-4 col-lg-4
mb-3 mx-lg-2">Buscar</button>
                <button type="submit" class="btn_color txt col-3 btn_secundario
col-sm-4 col-lg-4 mb-3" id="reset">Limpiar</button>
                <a href="{{ route('eventos.create') }}" class="btn_color col-4
col-sm-4 col-lg-4 mb-3">Crear evento</a>
            </div>
        </div>
    </form>

<!--LIMPIAR CAMPO-->
    <script>
    //Campos de busqueda (DOM)
        const input_nombre = document.getElementById('nombre');
        const btn_reset = document.getElementById('reset');

    //Funcion limpiar.
        function reset() {
            input_nombre.value = "";
            window.location = "/";
        }

        btn_reset.addEventListener('click', (e) => {
            e.preventDefault();
            reset();
        })
    </script>
```

`create.blade.php`

- **Propósito**: Pagina para crear evento.
- **Datos requeridos**: For each con distintos campos creados para mostrar todo el contenido (se pasan en el controlador).
- **Extras**: La vista cuenta con comentarios donde se especifica concretamente cada parte del codigo , de tal manera que se vea mas legible.

## Vista con código JS

A continuación, se muestra un ejemplo de cómo se puede incluir una funcion en JavaScript dentro de una vista Blade para calcular la media y obtener resultado del COSTE UNITARIO..

```


//CAMPOS DE CADA INPUT.
```

```
<!-- COSTE EVENTO -->
<input type="text" id="coste_evento" class="form-control" name="coste_evento"
placeholder="Coste evento" oninput="calcular_media()">

<!-- AFORO MÁXIMO -->
<input type="text" id="aforo_maximo" class="form-control" name="aforo"
placeholder="Aforo máximo" oninput="calcular_media()">

<!-- COSTE UNITARIO -->
<input type="text" id="coste_unitario" class="form-control" name="coste_unitario"
placeholder="Coste unitario">

<!---COSTE UNITARIO (FUNCION JS)-->
    <script>
        const coste_evento = document.getElementById('coste_evento');
        const aforo_maximo = document.getElementById('aforo_maximo');
        const coste_unitario = document.getElementById('coste_unitario');

        function calcular_media() {
            let coste = parseFloat(coste_evento.value);
            let aforo = parseFloat(aforo_maximo.value);
            let media = coste / aforo;

            if (!isNaN(media) && isFinite(media)) {
                coste_unitario.value = media.toFixed(2);
            } else {
                coste_unitario.value = '';
            }
        }
    </script>
```

`edit.blade.php`

- **Propósito**: Pagina para editar evento.
- **Datos requeridos**: For each con distintos campos creados para mostrar todo el contenido (se pasan en el controlador).
- **Extras**: La vista cuenta con comentarios donde se especifica concretamente cada parte del codigo , de tal manera que se vea mas legible. Suele ser igual la vista de create.blade.php pero en este caso recogemos valores que se mostraran en el formulario (GET).

```
//En el formulario se pasa el "enctype" debido a que subiremos archivos e imagenes
(se explican y se pasan luego en el controlador).
<form method="POST" action="{{route('eventos.update' , $evento->id)}}" class="m-
auto mt-5 mb-5" style="width: 70%;" enctype="multipartform-data">


//Tener en cuenta esta seccion ya que depende del modelo Eloquent de MARCA y
EVENTO.
<!--MARCA-->
```

```
    <div class="mb-3">
        <label for="marca">Marca* (Para seleccionar mas de una opcion, pulsa la
tecla Control)</label>
        //Pasamos marca como array ya que en la tabla intermedia se recogen los
valores.
        <select class="form-select" name="marca[]" multiple>
            @if(isset($marcas))
            //sortBy ordena los nombres en orden alfabetico.
            @foreach($marcas->sortBy('nombre') as $marca)
            //Recoge el valor del id de la tabla marcas.
            <option value="{{$marca->id}}"
                //Luego lo busca y lo selecciona en el option.
                {{ $evento->marcas->pluck('id')->contains($marca->id) ? 'selected'
: '' }}>
                {{$marca->nombre}}
            </option>
            @endforeach
            @endif
        </select>
    </div>
```

**VISTAS INVITADOS**

`index.blade.php`

- **Propósito**: Pagina principal de invitados.
- **Datos requeridos**: For each con distintos campos creados para mostrar todo el contenido (se pasan en el controlador).
- **Extras**: La vista cuenta con comentarios donde se especifica concretamente cada parte del codigo , de tal manera que se vea mas legible , esta vista cuenta con dos maneras de ver el contenido de los invitados. , una CARD para tamaño SM Y MD y otra que es la tabla que se mostrara para tamaño LG Y XL (rsponsive).

## Vista con código JS

A continuación, se muestra un ejemplo de cómo se puede seleccionar en un CHECKBOX si el invitado ASISTE o NO ASISTE , tambien te explico el JS de como poder eliminar un invitado de ese mismo evento.

```
    <!--BOTON ELIMINAR--->
    <form action="{{route('invitados.delete', $invitado->id)}}" method="POST">
        @csrf
        @method('DELETE')
        //El boton recoge informacion del id(invitado) mediante un nombre que le
ponemos.
        <button name="delete" id_delete="{{$invitado->id}}" type="submit"
style="background:none;border:none;color:#05072e;">
            <i class="bi bi-trash3-fill"></i>
        </button>
    </form>
```

```
    //FUNCION PARA ELIMINAR
    document.addEventListener('DOMContentLoaded', function() {
        //Pasamos al DOM el boton para cada invitado mediante su nombre , con
querySelectorAll seleccionamos todos.
        const btn_delete = document.querySelectorAll('[name="delete"]');

        btn_delete.forEach((element) => {
            element.addEventListener('click', (e) => {
                e.preventDefault();
                Swal.fire({
                    title: "¿Seguro que quieres eliminar?",
                    icon: "warning",
                    iconColor: "#05072e",
                    showDenyButton: true,
                    denyButtonColor: "#05072e",
                    showCancelButton: false,
                    showCancelColor: "#05072e",
                    confirmButtonText: "Sí",
                    confirmButtonColor: "#05072e",
                    denyButtonText: `No`
                }).then((result) => {
                    if (result.isConfirmed) {
                        //Recogemos el campo id_delete mediante funcion
getAttribute.
                        let id = element.getAttribute('id_delete');
                        //Pasamos en el fetch la ruta y se reemplaza el:id por id
para que me reciba la peticion.
                        fetch(`{{route('invitados.delete',
':id')}}`.replace(':id', id), {
                                method: 'DELETE',
                                //Token de seguridad.
                                headers: {
                                    'X-CSRF-TOKEN':
document.querySelector('meta[name="csrf-token"]').getAttribute('content')
                                }
                            })
                            .then(response => response.json())
                            .then(data => {
                                //Si es exitoso.
                                if (data.success) {
                                    Swal.fire({
                                            title: "¡Eliminado!",
                                            icon: "success",
                                            iconColor: "#05072e",
                                            showConfirmButton: true,
                                            confirmButtonColor: "#05072e"
                                        })
                                        .then(() => {
                                            window.location.href = "
{{route('invitados.index' , $evento->id)}}";
                                        })
                                //Si falla.
                                } else {
```

```
                                    Swal.fire("Error al eliminar.", "", "error");
                                }
                            })
                            //Error
                            .catch(error => {
                                console.log("Respuesta");
                                Swal.fire("Error al eliminar.", "", "error");
                            });
                    //Si damos al no los cambios no se guardan.
                    } else if (result.isDenied) {
                        Swal.fire({
                            title: "Cambios no guardados.",
                            icon: "warning",
                            iconColor: "#05072e",
                            confirmButtonColor: "#05072e"
                        });
                    }
                });
            });
        });
    });


    //FUNCION CHECKBOX ASISTENCIA
    function actualizarAsistencia(id, valor) { //Pasaremos id como valor y
recogeremos en la llamada un dato en este caso valor.
        fetch(`/invitados/${id}/asistencia`, { //Ruta para la solicitud fetch.
                //Pasamos el metodo.
                method: 'POST',
                //Recogemos datos y el token csrf del formulario.
                headers: {
                    'Content-Type': 'application/json',
                    'X-CSRF-TOKEN': document.querySelector('meta[name="csrf-
token"]').getAttribute('content')
                },
                //Cuerpo , pasamos el valor a string.
                body: JSON.stringify({
                    asiste: valor
                })
            })
            .then(response => response.json())
            .then(data => {
                //Si no entra en la paticion.
                if (!data.success) {
                    Swal.fire({
                        title: "Error al actualizar asistencia.",
                        icon: "warning",
                        iconColor: "#05072e",
                        confirmButtonColor: "#05072e"
                    })
                }
            })
            //Error
            .catch(() => {
```

```
                Swal.fire({
                    title: "Error de conexion.",
                    icon: "warning",
                    iconColor: "#05072e",
                    confirmButtonColor: "#05072e"
                })
            })
    }
```

`create.blade.php`

- **Propósito**: Pagina para crear invitados.
- **Datos requeridos**: For each con distintos campos creados para mostrar todo el contenido (se pasan en el controlador).
- **Extras**: La vista cuenta con comentarios donde se especifica concretamente cada parte del codigo , de tal manera que se vea mas legible.

```
    <!---VEHICULO PROPIO-->
    <label for="vehiculo_prop" class="form-label fw-bold" style="margin-right:
2%">¿Cuenta con un vehiculo propio?</label>
     <input id="btn_si" class="form-check-input" type="radio" value="si"
name="vehiculo_prop">
     <input id="btn_no" class="form-check-input" type="radio" value="no"
name="vehiculo_prop">

    <!----ETIQUETA (MOSTRAR)----->
    <div id="etiqueta-container" style="display: none;" class="mx-md-2">

    <!---VEHICULO EMPRESA--->
    <label for="vehiculo_emp" class="form-label fw-bold">¿Cuenta con un vehiculo
de empresa?</label>
    <input id="btn_si_emp" class="form-check-input" type="radio" value="si"
name="vehiculo_emp">
    <input id="btn_no_emp" class="form-check-input" type="radio" value="no"
name="vehiculo_emp">
```

## Vista con código JS

A continuación, se muestra un ejemplo de cómo se puede seleccionar en un CHECKBOX si el invitado cuenta con coche de vehiculo empresa o coche propio , actualmente esta parte esta hecha para que solo pueda ser si por una de sus partes ya que un invitado solo podra tener un solo coche.

```
    <!---JS ETIQUETA--->
    <script>
        const boton_si = document.getElementById('btn_si');//Vehiculo propio
        const boton_no = document.getElementById('btn_no');//Vehiculo propio
        const boton_si_emp = document.getElementById('btn_si_emp');//Vehiculo
empresa
```

```
        const boton_no_emp = document.getElementById('btn_no_emp');//Vehiculo
empresa
        const etiquetaContainer = document.getElementById('etiqueta-
container');//Etiqueta contenedor

        //Funcion para mostrar y ocultar contenedor etiqueta.
        function mostrar() {
            if(boton_si.checked || boton_si_emp.checked){
                etiquetaContainer.style.display = "block";
            }else{
                etiquetaContainer.style.display = "none";
            }
        }

        boton_si.addEventListener('click' , () =>{
            boton_no_emp.checked = true;
            mostrar();
        })

        boton_si_emp.addEventListener('click', () =>{
            boton_no.checked = true;
            mostrar();
        })

        boton_no.addEventListener('click' , () =>{
            mostrar();
        });

        boton_no_emp.addEventListener('click' , () =>{
            mostrar();
        })
    </script>
```

`edit.blade.php`

- **Propósito**: Pagina para editar invitados.
- **Datos requeridos**: For each con distintos campos creados para mostrar todo el contenido (se pasan en el controlador).
- **Extras**: La vista cuenta con comentarios donde se especifica concretamente cada parte del codigo , de tal manera que se vea mas legible. Suele ser igual la vista de create.blade.php pero en este caso recogemos valores que se mostraran en el formulario (GET).

**VISTAS ERRORES**

Estructura de carpetas

Vistas comunes

| Carpeta | Vista | Descripción |
|---|---|---|
| `errors/401` | `401.blade.php` | Acceso no autorizado |

| Carpeta | Vista | Descripción |
|---|---|---|
| errors/404 | 404.blade.php | URL no existe. |

- **Propósito**: Paginas principales que mostraran errores que puedan surgir.
- **Extras**: La vista cuenta con comentarios donde se especifica concretamente cada parte del codigo.

**VISTAS AUTH**

## Estructura de carpetas

## Vistas comunes

| Carpeta | Vista | Descripción |
|---|---|---|
| auth/ | confirm-password.blade.php | Confirma contraseña |
| auth | forgot.blade.php | Olvida contraseña |
| auth/ | login.blade.php | Inicio sesion |
| auth/ | register.blade.php | Registro |
| auth | reset-password.blade.php | Cambiar contraseña |
| auth/ | verify-email.blade.php | Verificar email |

- **Propósito**: Paginas principales de autenticacion que serviran para acceder a la aplicacion como administrador entre otras.
- **Extras**: La vista cuenta con comentarios donde se especifica concretamente cada parte del codigo.

## Vistas destacadas

**auth/login.blade.php**

- **Propósito**: Poder loguearte al entrar en la pagina del dashboard.
- **Datos requeridos**: Marcar la casilla de politica de privacidad y normas de uso.
- **Acciones**: Permitir validacion antes de entrar.

## Vista con código JS

A continuación, se muestra un ejemplo de cómo se puede incluir una validación en JavaScript dentro de una vista Blade para asegurarse de que el usuario acepte los términos antes de iniciar sesión.

### Objetivo

Evitar que el formulario se envíe si el checkbox de "He leído y acepto las normas de uso" no está marcado y funcion tamnbien para mostrar y ocultar.

### Código Blade + JavaScript

```
<!-- Remember Me -->
<div class="mb-3">
    <label for="remember_me" class="inline-flex items-center">
        <input id="remember_me" type="checkbox" class="rounded border-gray-300
text-indigo-600 shadow-sm focus:ring-indigo-500" name="remember">
        <span class="ms-2 text-sm text-gray-600">{{ __('He leído y acepto las
normas de uso y la política de privacidad.') }}</span>
    </label>
</div>

<!-- VALIDACIÓN JS PARA PODER LOGUEARTE -->
<script>
    const remember = document.getElementById('remember_me');
    const form = document.getElementById('form');

    function validarCheckbox() {
        if (!remember.checked) {
            Swal.fire({
                title: "Marque casilla para continuar.",
                icon: "warning",
                iconColor: "#05072e",
                confirmButtonText: "Aceptar",
                confirmButtonColor: "#05072e"
            });
            return false;
        }
        return true;
    }

    form.addEventListener('submit', (e) => {
        if (!validarCheckbox()) {
            e.preventDefault();
        }
    });
</script>


<!----FUNCION PARA MOSTRAR/OCULTAR CONTRASEÑA(JS)---->

<div class="input-group">
    <span id="toggle-password" class="input-group-text"><i class="bi bi-eye-fill">
</i></span>
    <x-text-input id="password" type="password" name="password" required
autocomplete="current-password" class="form-control" />
</div>

<script>
    const inputPassword = document.getElementById('password'); //Campo texto de
contraseña.
    const togglePassword = document.getElementById('toggle-password'); //Icono
imagen contraseña para mostrar y ocultar.
    const icon = togglePassword.querySelector('i');//Icono
```

```
    function mostrar() {
        //Si el campo de texto es tipo password que me cambie el icono y me
muestre la contraseña(mostrar).
        if (inputPassword.type === 'password') {
            inputPassword.type = 'text';
            icon.classList.remove('bi-eye-fill');
            icon.classList.add('bi-eye-slash-fill');

        } else {
            //Sino que me muestre todo lo contrario y lo deje como tipo
password(ocultar)
            inputPassword.type = 'password';
            icon.classList.remove('bi-eye-slash-fill');
            icon.classList.add('bi-eye-fill');
        }
    }

    //Cuando le de click que me realize la funcion.
    togglePassword.addEventListener('click', () => {
        mostrar();
    })
</script>
```

# Controlador (Controllers)

Descripción general

Los controladores son responsables de recibir las solicitudes del usuario, procesar la lógica correspondiente (generalmente delegando en modelos o servicios), y retornar una respuesta (normalmente una vista o JSON).

En este proyecto, los controladores siguen el patrón MVC proporcionado por Laravel y se encuentran en la ruta:

```
    app/Http/Controllers/
```

Estructura de controladores

```
EventosController
Encargado de manejar toda la lógica relacionada con la gestión de eventos:
mostrar, crear, editar y eliminar eventos.

InvitadosController
Encargado de manejar toda la lógica relacionada con la gestión de invitados:
mostrar, crear, editar y eliminar invitados.

AuthController
Controla los procesos de autenticación: login, logout, registro, etc.
```

### EventosController

#### Métodos principales

| Método | Descripción |
|---|---|
| `index()` | Muestra una vista con la lista de eventos paginados, tipos de evento y marcas. |
| `create()` | Muestra el formulario. |
| `store()` | Guarda un nuevo evento en la base de datos. |
| `edit($id)` | Muestra el formulario para editar un evento existente. |
| `update(Request $request, $id)` | Actualiza la información de un evento. |
| `delete($id)` | Elimina un evento. |
| `show(Request $request)` | Busca por todos los campos en el buscador. |
| `filtrarFecha(Request $request)` | Filtra por fecha de inicio y fehca de fin (rango). |
| `exportarInvitado($evento_id)` | Exporta en documento Excel los invitados de cada evento. |

**RUTAS EventosController**

```
/**CRUD EVENTOS */
    Route::get('/eventos/create', [EventosController::class, 'create'])->name('eventos.create');
    Route::post('/eventos/store', [EventosController::class, 'store'])->name('eventos.store');
    Route::delete('/eventos/delete/{id}', [EventosController::class, 'delete'])->name('eventos.delete');
    Route::get('/eventos/edit/{id}', [EventosController::class, 'edit'])->name('eventos.edit');
    Route::put('/eventos/update/{id}', [EventosController::class, 'update'])->name('eventos.update');

    Route::get('/eventos/show', [EventosController::class, 'show'])->name('eventos.show');
    Route::get('/eventos/filtrar' , [EventosController::class , 'filtrarFecha'])->name('eventos.filtrar');

    Route::get('/eventos/{evento_id}/exportar-invitados', [EventosController::class, 'exportarInvitados'])->name('eventos.exportarInvitados');
```

`index()`

```
    //MOSTRAR EVENTOS
    public function index()
    {
        if (Auth::check()) {
            //Pagina todos los eventos a 3 por pagina.
            $eventos = Evento::paginate(3);
            //Cuenta el total de eventos que se tiene.
            $total = $eventos->count();
            //Recoge todos los tipos de evento de la BD.
            $tipo_evento = TipoEvento::all();
            //Recoge todas las marcas de ls BD.
            $marcas = Marca::all();

            // Obtiene las marcas para cada evento
            foreach ($eventos as $evento) {
                $selectString = $evento->marca;
                $selectIds = explode(',', $selectString); // Convierte la cadena
en un array
                $evento->marcasSeleccionadas = Marca::whereIn('id', $selectIds)-
>get(); // Filtrar las marcas por idy las asocia.
            }

            //Retorna a la vista dashboard con los datos que se han recopilado.
            return view('dashboard', compact('eventos', 'tipo_evento', 'total',
'marcas'));
        } else {
            //Sino redirige al login.
            return view('./auth/login');
        }
    }
```

create()

```
    //MOSTRAR VENTANA CREAR EVENTO
    public function create()
    {
        //Recoge las marcas de la BD tabla marcas.
        $marcas = Marca::all();
        //Recoge los tipos d eevento de la BD tabla tipo_evento.
        $tipo_evento = TipoEvento::all();

        return view('eventos.create', compact('marcas', 'tipo_evento'));
    }
```

store()

```
    public function store(Request $request)
    {
```

```
        // VALIDAR CAMPOS
        $request->validate([
            'nombre' => 'required',
            'marca' => 'required|array',
            'fecha' => 'required',
            'hora' => 'required',
            'lugar_evento' => 'required',
            'tipo_evento' => 'required',
            'coste_evento' => 'required',
            'aforo' => 'required',
            'coste_unitario' => 'required',
            'enlace' => 'nullable',
            'documentacion' => 'nullable',
            'texto_invitacion' => 'required',
            'imagen' => 'required|image|mimes:jpeg,png,jpg,gif|max:2048'
        ]);

        try {
            $evento = new Evento();
            $evento->nombre = $request->nombre;
            //Une el array de marcas colocandole una , entre medias.
            $evento->marca = implode(' , ', $request->marca);
            $evento->fecha = $request->fecha;
            $evento->hora = $request->hora;
            $evento->lugar_evento = $request->lugar_evento;
            $evento->tipo_evento = $request->tipo_evento;
            $evento->coste_evento = $request->coste_evento;
            $evento->aforo = $request->aforo;
            $evento->coste_unitario = $request->coste_unitario;
            $evento->texto_invitacion = $request->texto_invitacion;
            $evento->imagen = $request->imagen;

            //Verificacion de la imagen , para poder subirla.
            if ($request->hasFile('imagen')) {
                $path = $request->file('imagen')->store('images/eventos',
'public');
                $evento->imagen = $path;
            }

            //Verificacion de la documentacion , para poder subirla.
            if ($request->hasFile('documentacion')) {
                $path = $request->file('documentacion')-
>store('images/eventos_documentacion', 'public');
                $evento->documentacion = $path;
            }

            //Guarda nuevo evento.
            $evento->save();

            //Recoge con un foreach todas las marcas que tenemos en la BD para
poder seleccionarla.
            foreach ($request->marca as $marca_id) {
                $eventos_marca = new EventosMarca();
                $eventos_marca->evento_id = $evento->id;
```

```
                $eventos_marca->marca_id = $marca_id;
                $eventos_marca->save();
            }

            return redirect()->route('eventos.index')->with('success', 'Evento
creado con éxito');
        } catch (Exception $e) {
            return redirect()->route('eventos.index')->with('error', 'Error al
crear evento');
        }
    }
```

delete()

```
//ELIMINAR EVENTO
    public function delete(int $id)
    {
        //Busca el evento por el id.
        $evento = Evento::find($id);

        if (!$evento) {
            /**Estamos usando SweetAlert por lo cual no debemos confundirnos ya
que este espera
             * un respuesta en formato JSON.
             */
            return response()->json([
                'success' => false,
                'message' => 'Error al eliminar'
            ], 404);
        } else {
            $evento->delete();
            return response()->json([
                'success' => true,
                'message' => 'Evento eliminado'
            ], 200);
        }
    }
```

edit()

```
//MOSTRAR VENTANA EDITAR.
    public function edit(int $id)
    {
        $evento = Evento::with('marcas')->findOrFail($id); // Carga el evento con
sus marcas asociadas
        $marcas = Marca::all(); //Obtiene todas las marcas.
        $tipo_evento = TipoEvento::all(); //Obtiene todos los tipos de eventos.
```

```
        if (!$evento) {
            return redirect()->route('eventos.index')->with('error', 'Evento no
encontrado');
        } else {
            return view('eventos.edit', compact('evento', 'marcas',
'tipo_evento'));
        }
    }
```

update()

```
//ACTUALIZAR EVENTO
    public function update(Request $request, int $id)
    {
        //Busca evento por id.
        $evento = Evento::find($id);

        if (!$evento) {
            return redirect()->route('eventos.index')->with('error', 'No se
encontró el evento');
        }

        // VALIDAR CAMPOS
        $request->validate([
            'nombre' => 'required',
            'marca' => 'required|array',
            'fecha' => 'required',
            'hora' => 'required',
            'lugar_evento' => 'required',
            'tipo_evento' => 'required',
            'coste_evento' => 'required',
            'aforo' => 'required',
            'coste_unitario' => 'required',
            'enlace' => 'nullable',
            'documentacion' => 'nullable',
            'texto_invitacion' => 'required',
        ]);

        try {
            // Actualizar los campos del evento
            $evento->nombre = $request->nombre;
            $evento->fecha = $request->fecha;
            $evento->hora = $request->hora;
            $evento->lugar_evento = $request->lugar_evento;
            $evento->tipo_evento = $request->tipo_evento;
            $evento->coste_evento = $request->coste_evento;
            $evento->aforo = $request->aforo;
            $evento->coste_unitario = $request->coste_unitario;
            $evento->texto_invitacion = $request->texto_invitacion;
```

```
            // Manejamos la documentación con la siguiente condicion.
            if ($request->hasFile('documentacion')) {
                // Elimina la documentación anterior si existe
                if ($evento->documentacion && Storage::exists('public/' . $evento->documentacion)) {
                    Storage::delete('public/' . $evento->documentacion);
                }

                // Guardar la nueva documentación en "images/eventos_documentacion"
                $pathDocumentacion = $request->file('documentacion')->store('images/eventos_documentacion', 'public');
                $evento->documentacion = $pathDocumentacion; // Guarda la ruta relativa en la base de datos
            }

            $evento->save();

            // Sincroniza las marcas seleccionadas
            $evento->marcas()->sync($request->marca);

            return redirect()->route('eventos.index')->with('success', 'Evento actualizado con éxito');
        } catch (Exception $e) {
            return redirect()->route('eventos.index')->with('error', 'Error al actualizar el evento');
        }
    }
```

show()

```
public function show(Request $request)
    {
        //Input llamado buscador.
        $query = $request->input('buscador');
        //Recogemos el array de marcas con una variable.
        $marcaId = $request->input('marca');

        //Carga los eventos con su marca relacionada.
        $eventos = Evento::with('marcas')
        //Si existe el buscador $query se realiza el filtrado.
            ->when($query, function ($q) use ($query) {
                $q->where(function ($subQ) use ($query) {
                    $subQ->where('nombre', 'like', "%{$query}%")
                        ->orWhere('tipo_evento', 'like', "%{$query}%")
                        ->orWhere('lugar_evento', 'like', "%{$query}%")
                        ->orWhereHas('marcas', function ($q3) use ($query) {
                            $q3->where('nombre', 'like', "%{$query}%");
                        });
                });
            })
```

```
            //Si se proporciona marcaId se filtran los eventos con esa marca.
            ->when($marcaId, function ($q) use ($marcaId) {
                $q->whereHas('marcas', function ($q2) use ($marcaId) {
                    $q2->where('id', $marcaId);
                });
            })
            //Paginacion  de 3 maximo en pantalla.
            ->paginate(3)
            //Mantiene filtros en la paginacion.
            ->appends(['buscador' => $query, 'marca' => $marcaId]);
        //Total de eventos.
        $total = $eventos->total();

        return view('dashboard', compact('eventos', 'total'));
    }
```

### filtrarFecha()

```
    //FUNCION PARA FILTRAR FECHA
    public function filtrarFecha(Request $request)
    {
        //Input de fecha inicio.
        $fecha_inicio = $request->fecha_inicio;
        //Input de fecha fin.
        $fecha_fin = $request->fecha_fin;

        if ($fecha_inicio && $fecha_fin) {
            //Filtrar los eventos por la fecha inicio y fecha fin.
            $eventos = Evento::where('fecha', '>=', $fecha_inicio)
                ->whereDate('fecha', '<=', $fecha_fin)
                //Paginacion total 3.
                ->paginate(3)
                //Mantiene filtro en la paginacion.
                ->appends(['fecha_inicio' => $fecha_inicio, 'fecha_fin' =>
$fecha_fin]);

            //Cuenta el total de eventos.
            $total = $eventos->count();
        } else {
            //Mantiene paginacion.
            $eventos = Evento::paginate(3);
            //Mantiene numero de eventos.
            $total = $eventos->count();
        }

        return view('dashboard', compact('eventos', 'total'));
    }
```

### exportarInvitados()

```
    public function exportarInvitados($evento_id)
    {
        //Busca el evento por el id.
        $evento = Evento::find($evento_id);
        //Devuelve un enlace de descarga en formato Excel para poder ver los
invitados de ese evento.
        return Excel::download(new InvitadosExport($evento), 'invitados_evento_' .
$evento_id . '.xlsx');
    }
```

**Datos extras funcion exportarInvitados():**

📌 Descripción general

Funciones para poder exportar el archivo en Excel:

```
    class InvitadosExport implements FromCollection, WithHeadings
{
    protected $evento; //Protege la propiedad del evento.

    public function headings(): array
    {
        //Nombre del campos de las casillas.
        return [
            'Empresa',
            'CIF',
            'Nombre',
            'Apellidos',
            'Telefono',
            'Dni',
            'Email',
            'KAM',
            'Asistio'
        ];
    }

    public function __construct(Evento $evento)
    {
        $this->evento = $evento; //Guarda el evento para usarlo despues.
    }

    public function collection()
    {
        /***Recoge de cada evento el invitado y recorre con un bucle para que
         * me pinte todos los que hay ahi , ya que espera un array. */
        return $this->evento->invitados->map(function ($invitado) {
            return [
                'empresa'    => $invitado->empresa,
                'cif'        => $invitado->cif,
                'nombre'     => $invitado->nombre,
```

```
                'apellidos'  => $invitado->apellido,
                'telefono'   =>  "\t" . $invitado->telefono,//Se coloca "\t" para
que se veo el formato texto (numeros).
                'dni'        => "\t" . $invitado->dni,//Se coloca "\t" para que se
veo el formato texto (numeros).
                'email'      => $invitado->email,
                'kam'        => $invitado->kam,
                'asistio'    => $invitado->asiste ? 'Si' : 'No' //Condicion si el
invitado asiste o no asiste.
            ];
        });
    }
}
```

## InvitadosController

### Métodos principales

| Método | Descripción |
|---|---|
| `index()` | Muestra una vista con la lista de invitadosa paginados. |
| `create()` | Muestra el formulario. |
| `store()` | Guarda un nuevo invitado en la base de datos. |
| `edit($id)` | Muestra el formulario para editar un invitado existente. |
| `update(Request $request, $id)` | Actualiza la información de un invitado. |
| `delete($id)` | Elimina un invitado. |
| `show(Request $request)` | Busca por todos los campos en el buscador. |
| `actualizarAsistencia($id)` | Marca en el chekbox y guarda si el invitado asiste o no. |

**RUTAS InvitadosController**

```
/***CRUD INVITADOS(CONDUCTORES) */
    Route::get('/invitados/{id}' , [InvitadosController::class , 'index'])-
>name('invitados.index');
    Route::get('/invitados/create/{id}' , [InvitadosController::class ,
'create'])->name('invitados.create');
    Route::post('/invitados/store/{id}' , [InvitadosController::class , 'store'])-
>name('invitados.store');
    Route::delete('/invitados/delete/{id}' , [InvitadosController::class ,
'delete'])->name('invitados.delete');
    Route::get('/invitados/edit/{id}' , [InvitadosController::class , 'edit'])-
>name('invitados.edit');
    Route::put('/invitados/update/{id}' , [InvitadosController::class ,
'update'])->name('invitados.update');

    Route::get('/invitados/show/{id}' , [InvitadosController::class , 'show'])-
```

```
>name('invitados.show');
    Route::post('/invitados/{id}/asistencia' , [InvitadosController::class ,
'actualizarAsistencia']);
```

index()

```
    //FUNCION MUESTRA INVITADOS TABLA.
    public function index($id)
    {
        //Pasamos el id para buscar y recoger la informacion de cada evento.
        $evento = Evento::find($id);
        if (!$evento) {
            return redirect()->back()->with('error', 'Evento no encontrado');
        }
        //Paginacion de 50.
        $invitados = $evento->invitados()->paginate(50);
        //Me cuenta el total de invitados dentro de cada evento.
        $total = $invitados->count();
        //Pasamos en el compact todo lo que vamos a mostrar.
        return view('invitados.index', ['id' => $evento], compact('evento',
'invitados', 'total'));
    }
```

create()

```
    //FUNCION MUESTRA FORMULARIO CREAR.
    public function create($id)
    {
        //Muestra formulario de creacion para cada evento (crea un nuevo
invitado).
        $evento = Evento::find($id);
        return view('invitados.create', compact('evento'));
    }
```

store()

```
    //FUNCION PARA CREAR INVITADO
    public function store(Request $request, $id)
    {
        //VALIDACION
        $request->validate([
            'cif' => 'nullable',
            'nombre' => 'required',
            'apellido' => 'required',
            'email' => 'required',
            'telefono' => 'required',
```

```
            'empresa' => 'required',
            'vehiculo_prop' => 'required',
            'vehiculo_emp' => 'required',
            'intolerancia' => 'required',
            'preferencia' => 'required',
            'carnet' => 'nullable',
            'kam' => 'required',
            'dni' => 'nullable',
            'etiqueta' => 'required_if:vehiculo_prop,si'
        ]);

        try {
            //Busca por cada evento y creara invitado dentro de ese evento.
            $evento = Evento::find($id);
            //Creamos un  nuevo invitado(conductor).
            $invitado = new Conductor();
            //Recogemos el id del invitado y lo pasamos al evento para poder
crearlo.
            $invitado->id = $request->evento;

            $invitado->cif = $request->cif;
            $invitado->nombre = $request->nombre;
            $invitado->apellido = $request->apellido;
            $invitado->email = $request->email;
            $invitado->telefono = $request->telefono;
            $invitado->empresa = $request->empresa;
            $invitado->vehiculo_prop = $request->vehiculo_prop;
            $invitado->vehiculo_emp = $request->vehiculo_emp;
            $invitado->intolerancia = $request->intolerancia;
            $invitado->preferencia = $request->preferencia;
            $invitado->carnet = $request->carnet;
            $invitado->kam = $request->kam;
            $invitado->dni = $request->dni;

            //Guarda el valor de uno o del otro por si esta marcado.
            if ($request->vehiculo_prop === 'si' || $request->vehiculo_emp ===
'si') {
                $invitado->etiqueta = $request->etiqueta;
            }

            $invitado->save();

            //Creare un nuevo invitado dentro del evento seleccionado.
            $evento_conductor = new EventoConductor();

            //Cogeremos la tabla del evento_conductor y le pasaremos el id para
cada uno (conductor_id).
            $evento_conductor->conductor_id = $invitado->id;
            //Cogeremos la tabla del evento_conductor y le pasaremos el id para
cada uno (evento_id).
            $evento_conductor->evento_id = $evento->id;
            $evento_conductor->save();

            return redirect()->route('invitados.index', compact('evento',
```

```
'invitado'))->with('success', 'Invitado creado con éxito.');
        } catch (Exception $e) {
            return redirect()->route('invitados.index', $evento->id)-
>with('error', 'Error al crear invitado.');
        }
    }
```

delete()

```
    //FUNCION PARA ELIMINAR INVITADO.
    public function delete(int $id)
    {
        //Busca invitado por id.
        $invitados = Conductor::find($id);

        //Recoge la respuesta en formato JSON (SweetAlert)
        if (!$invitados) {
            return response()->json([
                'success' => false,
                'message' => 'Error al eliminar invitado'
            ], 404);
        } else {
            $invitados->delete();
            return response()->json([
                'success' => true,
                'message' => 'Invitado eliminado'
            ], 200);
        }
    }
```

edit()

```
    //FUNCION PARA MOSTRAR EL FORMULARIO DE EDITAR.
    public function edit(int $id)
    {
        //Busca el invitado dentro de cada evento creado.
        $invitados = Conductor::find($id);
        //Obtiene el evento asociado en la tabla evento conductor.
        $eventoConductor = EventoConductor::where('conductor_id', $invitados->id)-
>first();
        //Obtiene el id del evento si existe.
        $eventoId = $eventoConductor ? $eventoConductor->evento_id : null;

        if (!$invitados) {
            return redirect()->route('invitados.index')->with('error', 'No se
encontro el invitado');
        }
```

```
        return view('invitados.edit', compact('invitados', 'eventoId'));
    }
```

update()

```
//FUNCION PARA EDITAR
    public function update(Request $request, int $id)
    {
        //Busca el invitado mediante su id
        $invitados = Conductor::find($id);

        if (!$invitados) {
            return redirect()->route('invitados.index', ['id' => $id])-
>with('error', 'No se encontró el invitado.');
        }

        //Asocia el id del conductor al evento y lo recoge.
        $eventoConductor = EventoConductor::where('conductor_id', $invitados->id)-
>first();

        if (!$eventoConductor) {
            return redirect()->route('invitados.index', ['id' => $id])-
>with('error', 'No se encontró el evento asociado al invitado.');
        }

        $eventoId = $eventoConductor->evento_id;

        //Validacion
        $request->validate([
            'cif' => 'nullable',
            'nombre' => 'required',
            'apellido' => 'required',
            'email' => 'required',
            'telefono' => 'required',
            'empresa' => 'required',
            'vehiculo_prop' => 'required',
            'vehiculo_emp' => 'required',
            'intolerancia' => 'required',
            'preferencia' => 'required',
            'carnet' => 'nullable',
            'kam' => 'required',
            'dni' => 'nullable',
            'etiqueta' => 'required_if:vehiculo_prop,si'
        ]);

        try {
            $invitados->cif = $request->cif;
            $invitados->nombre = $request->nombre;
            $invitados->apellido = $request->apellido;
            $invitados->email = $request->email;
            $invitados->telefono = $request->telefono;
```

```
            $invitados->empresa = $request->empresa;
            $invitados->vehiculo_prop = $request->vehiculo_prop;
            $invitados->vehiculo_emp = $request->vehiculo_emp;
            $invitados->intolerancia = $request->intolerancia;
            $invitados->preferencia = $request->preferencia;
            $invitados->carnet = $request->carnet;
            $invitados->kam = $request->kam;
            $invitados->dni = $request->dni;

            //Guarda el valor de uno o del otro por si esta marcado.
            if ($request->vehiculo_prop === 'si' || $request->vehiculo_emp ===
'si') {
                $invitados->etiqueta = $request->etiqueta;
            }

            $invitados->save();

            return redirect()->route('invitados.index', ['id' => $eventoId])-
>with('success', 'Invitado actualizado con éxito.');
        } catch (Exception $e) {
            return redirect()->route('invitados.index', ['id' => $eventoId])-
>with('error', 'Error al actualizar el invitado.');
        }
    }
```

show()

```
    //FUNCION PARA FILTRAR BUSQUEDA
    public function show(Request $request, int $id)
    {
        //Recoge informacion del input
        $query = $request->input('buscador');
        //Buscamos y recogemos datos del evento.
        $evento = Evento::find($id);

        $invitados = $evento->invitados()->where(function ($q) use ($query) {
            $q->where('empresa', 'like', "%{$query}%")
                ->orWhere('cif', 'like', "%{$query}%")
                ->orWhere('nombre', 'like', "%{$query}%")
                ->orWhere('apellido', 'like', "%{$query}%")
                ->orWhere('email', 'like', "%{$query}%")
                ->orWhere('telefono', 'like', "%{$query}%")
                ->orWhere('kam', 'like', "%{$query}%");
        })->paginate(6)//Paginacion de 6 por pantalla (busqueda)
        //Mantiene la paginacion aun realizando el filtrado.
            ->appends(['buscador' => $query]);

        $total = $invitados->count();//Total de invitados por evento.
        return view('invitados.index', compact('invitados', 'total', 'evento'));
    }
```

`actualizarAsistencia()`

```
    //FUNCION PARA MARCAR LA ASISTENCIA
    public function actualizarAsistencia(Request $request, $id)
    {
        //Busca el invitado por su id y recoge la asitencia.
        $invitado = Conductor::find($id);
        //Si se marca estara en la posicion 1 (true) o 0 (false)
        $invitado->asiste = $request->asiste ? 1 : 0;
        //Guarda el invitado y su asistencia.
        $invitado->save();

        return response()->json([
            'success'  => true
        ]);
    }
```

## CochesController

### Métodos principales

| Método | Descripción |
|---|---|
| `index()` | Muestra una vista con la lista de coches paginados. |
| `create()` | Muestra el formulario. |
| `store()` | Guarda un nuevo coche en la base de datos. |
| `edit($id)` | Muestra el formulario para editar un invitado existente. |
| `update(Request $request, $id)` | Actualiza la información de un invitado. |
| `delete($id)` | Elimina un invitado. |
| `show(Request $request)` | Busca por todos los campos en el buscador. |
| `actualizarAsistencia($id)` | Marca en el chekbox y guarda si el coche tiene llave o no |
| `exportarCoches($evento_id)` | Exporta en Excel. |
| `importarCoches($id)` | Importa en Excel. |

**RUTAS CochesController**

```
    //CRUD COCHES
    Route::get('/coches/{id}', [CochesController::class, 'index'])-
>name('coches.index');
    Route::get('/coches/create/{id}', [CochesController::class, 'create'])-
>name('coches.create');
    Route::post('/coches/{id}', [CochesController::class, 'store'])-
>name('coches.store');
```

```
    Route::delete('/coches/delete/{id}', [CochesController::class, 'delete'])-
>name('coches.delete');
    Route::get('/coches/edit/{id}', [CochesController::class, 'edit'])-
>name('coches.edit');
    Route::put('/coches/update/{id}', [CochesController::class, 'update'])-
>name('coches.update');

    Route::get('/coches/show/{id}', [CochesController::class, 'show'])-
>name('coches.show');
    Route::post('/coches/{id}/actualizar', [CochesController::class,
'actualizarAsistencia']);

    //EXPORTAR EXCEL
    Route::get('/coches/{evento_id}/exportar-coches', [CochesController::class,
'exportarCoches'])->name('coches.exportarCoches');
    //IMPORTAR EXCEL
    Route::post('/coches/importar/{id}', [CochesController::class,
'importarCoches'])->name('coches.importarCoches');
```

index()

```
    public function index($id)
    {
        //Busca el evento por el id para mostrarlo.
        $evento = Evento::find($id);
        if (!$evento) {
            return redirect()->back()->with('error', 'Coche no encontrado');
        }
        // Filtra los coches por evento_id
        $coches = Coch::where('evento_id', $evento->id)->paginate(10);
        //Realiza el conteo de los coches en la misma pagina.
        $total = $coches->count();

        return view('coches.index', compact('coches', 'evento', 'total'));
    }
```

create()

```
    public function create($id)
    {
        //Busca el evento por el id y luego mostrara el formulario.
        $evento = Evento::find($id);
        return view('coches.create', compact('evento'));
    }
```

store()

```
    public function store(Request $request, $id)
    {
        //Validaciones para poder crewar formulario.
        $request->validate([
            'marca' => 'required',
            'modelo' => 'required',
            'version' => 'required',
            'matricula' => 'required',
            'kam' => 'required',
            'asiste' => 'nullable|boolean'
        ]);

        try {
            //Busca el evento y crea el cohe de ese evento en especifico.
            $evento = Evento::find($id);
            if (!$evento) {
                return redirect()->back()->with('error', 'Evento no encontrado');
            }

            //Se crea un nuevo coche.
            $coche = new Coch();
            $coche->marca = $request->marca;
            $coche->modelo = $request->modelo;
            $coche->version = $request->version;
            $coche->matricula = $request->matricula;
            $coche->kam = $request->kam;
            $coche->asiste = $request->input('asiste', 0);
            $coche->evento_id = $evento->id;
            $coche->save();

            return redirect()->route('coches.index', $evento->id)->with('success',
'Coche creado con éxito.');
        } catch (Exception $e) {
            return redirect()->route('coches.index', $id)->with('error', 'Error al
crear coche.');
        }
    }
```

delete()

```
    public function delete(int $id)
    {
        //Busca el coche creado por el id que tiene.
        $coches = Coch::find($id);

        //Manejamos con una condicion para un JSON para el sweetAlert , sino no
elimina.
        if (!$coches) {
            return response()->json([
                'success' => false,
```

```
                'message' => 'Error al eliminar coche'
            ], 404);
        } else {
            $coches->delete();
            return response()->json([
                'success' => true,
                'message' => 'Coche eliminado'
            ], 200);
        }
    }
```

edit()

```
    public function edit(int $id)
    {
        //Busca el coche poor el id y muestra el formulario de editar.
        $coches = Coch::find($id);

        if (!$coches) {
            return redirect()->route('coches.index')->with('error', 'No se
encontro el invitado');
        }
        //Maneja la relacion eloquent para que solo traiga y pinte los coches de
ese evento.
        $evento = $coches->evento;
        return view('coches.edit', compact('coches', 'evento'));
    }
```

update()

```
    public function update(Request $request, int $id)
    {
        $coche = Coch::find($id);

        if (!$coche) {
            return redirect()->route('coches.index')->with('error', 'No se
encontró el coche');
        }

        $evento = $coche->evento; //Obtiene evento desde la relacion.

        $request->validate([
            'marca' => 'required',
            'modelo' => 'required',
            'version' => 'required',
            'matricula' => 'required',
            'kam' => 'required',
            'asiste' => 'nullable|in:0,1'
```

```
        ]);

        try {
            $coche->marca = $request->marca;
            $coche->modelo = $request->modelo;
            $coche->version = $request->version;
            $coche->matricula = $request->matricula;
            $coche->kam = $request->kam;
            $coche->asiste = $request->input('asiste', 0);
            $coche->save();

            //Realizamos la condicion ternaria para que no nos salga el error de
null al actualizar (muy importante).
            return redirect()->route('coches.index', $evento ? $evento->id :
null)->with('success', 'Coche actualizado correctamente');
        } catch (Exception $e) {
            return redirect()->back()->with('error', 'Error al actualizar el
coche: ');
        }
    }
```

**actualizarAsistencia()**

```
    public function actualizarAsistencia(Request $request, $id)
    {
        //Busca el coche por su id.
        $coche = Coch::find($id);
        //Si se marca el checkbox mediante el JS creado esto recoge la informacion
, por si es true (boolean).
        $coche->asiste = $request->asiste ? 1 : 0;
        //Guarda y actualiza el registro.
        $coche->save();

        return response()->json([
            'success' => true
        ]);
    }
```

**show()**

```
public function show(Request $request, $id)
    {
        //Busca el coche por su id.
        $evento = Evento::find($id);
        //Recoge el valor del input llamado buscador.
        $buscador = $request->input('buscador');

        $coches = Coch::where('evento_id', $evento->id)
```

```
            ->where(function ($query) use ($buscador) {
                $query->where('marca', 'like', '%' . $buscador . '%')
                    ->orWhere('modelo', 'like', '%' . $buscador . '%')
                    ->orWhere('version', 'like', '%' . $buscador . '%')
                    ->orWhere('matricula', 'like', '%' . $buscador . '%')
                    ->orWhere('kam', 'like', '%' . $buscador . '%');
            })
            ->paginate(6);

        //Realiza el conteo de los coches en la vista.
        $total = Coch::where('evento_id', $evento->id)->count();
        return view('coches.index', compact('coches', 'total', 'evento'));
    }
```

exportarCoches()

```
    public function exportarCoches($evento_id)
    {
        //Busca el evento por el id.
        $evento = Evento::find($evento_id);

        //Devuelve la decarga con el excel creado.
        return Excel::download(
            new CochExport($evento),
            'coches' . $evento_id . '.xlsx'
        );
    }
```

importarCoches()

```
     public function importarCoches(Request $request, $id)
    {
        //Busca el evento pro el id.
        $evento = Evento::find($id);

        try {
            //Genera una importacion de ese mismo evento(por eso se le apsa el
id).
            Excel::import(new CochImport($evento->id), $request->file('file'));

            // Si la importación es exitosa, redirige a la vista que muestra los
coches del evento
            // y muestra un mensaje de éxito.
            return redirect()
                ->route('coches.show', $evento->id)
                ->with('success', 'Coches importados correctamente');
        } catch (\Exception $e) {
            // Si ocurre algún error durante la importación,
```

```
            // redirige a la página anterior y muestra un mensaje de error con la
causa.
            return redirect()
                ->back()
                ->withErrors(['error' => 'Hubo un error al importar los coches: '
. $e->getMessage()]);
        }
    }
```

## 📌 Conclusión

Este proyecto ha sido desarrollado utilizando **Laravel 12** como framework principal del lado del servidor y **JavaScript** para las interacciones en el cliente. La estructura basada en el patrón **Modelo-Vista-Controlador (MVC)** permitió una clara separación de responsabilidades, facilitando tanto el desarrollo como el mantenimiento del sistema.

Gracias a Laravel, se logró una integración eficiente entre la lógica de negocio, la gestión de rutas y el acceso a la base de datos, mientras que JavaScript aportó dinamismo y una mejor experiencia para el usuario en la interfaz.

### ☑ Logros principales:

- Implementación exitosa del patrón MVC con Laravel 12.
- Código organizado, reutilizable y fácil de escalar.
- Interfaz funcional y dinámica con JavaScript.

### 🚀 Recomendaciones para futuras mejoras:

- Incorporar autenticación basada en roles (admin, usuario, etc.).
- Agregar validaciones asíncronas del lado del cliente.
- Implementar pruebas automatizadas para asegurar la calidad del código.

Este proyecto representa una base sólida para continuar creciendo en complejidad y funcionalidad en futuras versiones.