APLICACIÓN WEB DE SISTEMA GESTIÓN DE PARTES PARA EL CENTRO IES SAN SEBASTIÁN

Sergio Guerrero Borrero

Alejandro Cabot Caparrós

Junta de Andalucía

IES SAN SEBASTIÁN

S.G.P S.S

IES SAN SEBASTIÁN 2024

CICLO FORMATIVO DE GRADO SUPERIOR DESARROLLO DE APLICACIONES WEB

HOJA RESUMEN-PROYECTO

<table>
<tr><td colspan="2">Título del proyecto: Sistema de Gestión de Partes para el centro IES San Sebastián (S.G.P S.S)</td></tr>
<tr><td>Autores:
Sergio Guerrero Borrero
Alejandro Cabot Caparrós</td><td>Fecha:
/06/2024
Curso 2023-2024</td></tr>
<tr><td colspan="2">Tutores: Manuel Alfonso Romero Caro / Manuel Jesús Iglesias Espina</td></tr>
<tr><td colspan="2">Titulación: C.F.G.S Desarrollo de Aplicaciones Web</td></tr>
<tr><td colspan="2">Palabras clave: Laravel, PHP, Javascript, MySQL, …</td></tr>
<tr><td colspan="2">Resumen del Proyecto (Español):

Resumen del Proyecto (Inglés):</td></tr>
</table>

Índice

1. INTRODUCCIÓN

1.1 Introducción a la memoria

En esta memoria se tratarán todos los aspectos relacionados con el proyecto a realizar, tales como la descripción general del mismo, estudios sobre su viabilidad, su presupuesto necesario, sus aspectos visuales, diseño, implementación, así como ejemplos de su funcionamiento mediante diferentes pruebas y demostraciones.

Como indica el nombre del proyecto, tratará de la creación de una aplicación web para digitalizar todo el proceso de realización de sanciones a los alumnos que anteriormente era gestionado físicamente por el personal del centro.

1.2 Descripción

El alcance tratará de alcanzar las expectativas del cliente, desplegando así una aplicación web robusta y funcional que cumpla sus funciones en los años venideros. Su funcionamiento a rasgos generales tratará de la descarga y exportación de datos de los alumnos para así asociar sus sanciones de forma rápida y efectiva, notificando automáticamente a todos los involucrados en el proceso mediante los correos electrónicos asociados.

Para ello se utilizarán Laravel y diversos paquetes que ayudarán a implementar las funcionalidades anteriormente explicadas, así como MySQL como sistema gestor de base de datos.

1.3 Objetivos generales

Como se ha tratado en el punto anterior, los objetivos principales serán cumplir las necesidades del cliente, adoptando las mismas al funcionamiento de la aplicación y añadiendo otras que los desarrolladores de este proyecto hayan considerado oportunas y útiles.

La intención de los mismos es desplegar una aplicación que cumpla sus funciones con una alta disponibilidad y tolerancia y control de errores, otorgando mantenimiento si fuese necesario.

1.4 Motivaciones personales

Debido al encargo de esta aplicación, ambos autores se han sentido muy alabados al confiárseles el desarrollo de la misma, viendo una utilidad real en la misma ya que será llevada a producción y usada por el instituto IES San Sebastián en un futuro inmediato.

Además de las razones expuestas, el desarrollo en Laravel se ha tratado con mucha profundidad a lo largo del curso, siendo este un gran aliciente para poner a prueba los conocimientos adquiridos.

En su módulo de FCT Alejandro ha tratado el desarrollo en Laravel, por lo que el desarrollo simultáneo con este proyecto se complementa con sus funciones realizadas en su empresa. Sergio en menor medida al centrarse en tecnologías de Microsoft, pero también ha tratado Laravel con el tratamiento de APIs y creación páginas con tablas de datos.

1.5 Estructura de la memoria

La estructura de esta memoria contará con varios apartados. Comenzará con el Estudio de viabilidad del proyecto, tratando una explicación general del mismo, sus requerimientos, objetivos, planificación etc.

Una vez finalizada el estudio de viabilidad del proyecto, comenzará el apartado de su desarrollo, es decir, tanto la implantación como las diferentes pruebas que se realizarán para comprobar el correcto funcionamiento de la misma.

Para finalizar encontraremos el apartado de conclusiones, en el que como su nombre indica se realizarán las últimas reflexiones respecto al proyecto con una valoración personal, y tras esto encontraremos el apartado de bibliografía, en los que se mostrarán las diferentes fuentes de información que han sido utilizadas y consultadas para la consulta de datos y ayudas en el proceso de configuración.

2. ESTUDIO DE LA VIABILIDAD

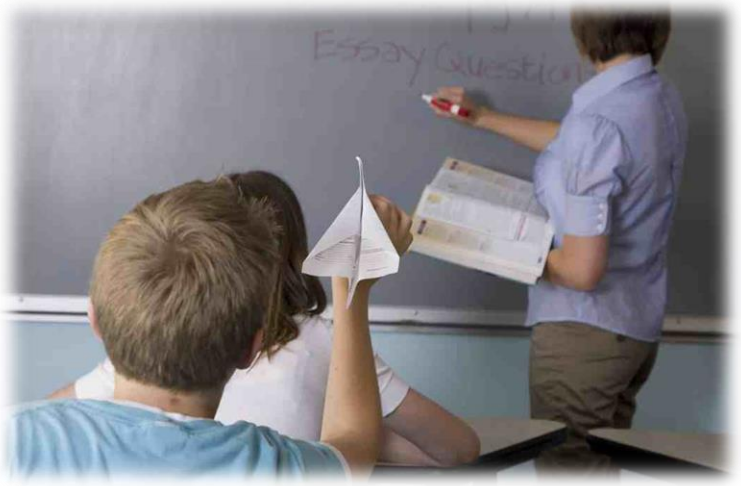
2.1 Introducción

En este apartado se tratará el estudio de la viabilidad del proyecto, en otras palabras, la valoración de la situación actual y la razón de por qué es necesario. También se expondrá la planificación de la misma en cuanto a todas las funciones implementadas y el tiempo aproximado que se ha empleado en ellas.

Se tratará también el presupuesto aproximado incluyendo el coste de algunos recursos, así como un supuesto en el que se hubiese facturado el tiempo y los recursos empleados, tratándose en mayor profundidad en su respectivo subapartado.

2.2 Estudio de la situación actual

La idea de esta aplicación surgió debido al gran aumento de sanciones de los alumnos en el instituto IES San Sebastián que deben ser tramitado, un caso que parece darse a nivel general en la educación, especialmente en educación secundaria (Especialmente en los primeros cursos según un estudio del periódico "*La Voz de Galicia*").



Debido a esto, la gestión a papel de las mismas está quedando obsoleta y resultando poco eficiente, razón por la cual comenzó el planteamiento de esta solución. Este sistema facilitará la preservación y realización de dichas sanciones, incluso garantizando una comunicación directa con todos los implicados, siendo de especial utilidad para la familia del alumno causante de la misma.

2.3 Planificación del proyecto

Este subapartado tratará la gestión del y planificación del proyecto, siguiendo las que suelen ser las etapas comunes en el ciclo de desarrollo de una aplicación de Software, comenzando desde el planteamiento de los requisitos hasta el despliegue de las mismas y las operaciones posteriores.

A continuación se planteará la planificación del tiempo estimado empleado en cada una de las funciones implementadas y su investigación correspondiente:

Requisitos	
Diseño	
Desarrollo	
Pruebas	
Despliegue	
Operaciones / Pruebas con la aplicación desplegada	

Este subapartado se complementará con el siguiente, utilizando el tiempo empleado para hacer la estimación de los costes.

2.4 Presupuesto

Todas las herramientas usadas para el desarrollo del proyecto son gratuitas, por lo que el empleo de las mismas no tiene coste. Sin embargo el despliegue del mismo requerirá del uso de un servidor de dominio, así como para la base de datos a utilizar.

El coste de estos recursos sería el siguiente:

Además de esto, suponiendo un caso de desarrollo real, se podría hacer la siguiente estimación por el coste del tiempo y esfuerzos empleados:

3. ANÁLISIS DEL PROYECTO

3.1 Introducción

La tercera sección de la documentación tratará sobre el análisis del proyecto en cuanto a los conceptos y elementos fundamentales para el funcionamiento del mismo. Estas serán las bases del proyecto tanto teóricamente como funcionalmente.

En primer lugar se tratarán los conceptos clave como la Alta Disponibilidad y la escalabilidad, para después explicar las tecnologías existentes y los recursos empleados, con un apartado extra que explicará a rasgos generales el funcionamiento de los paquetes de Composer utilizados, para posteriormente enfocarlos a las necesidades del desarrollo.

3.2 Conceptos clave

3.2.1 Alta Disponibilidad

La Alta Disponibilidad es un concepto propio de las nuevas tecnologías y es fundamental cuando se desean ofrecer servicios en la actualidad, esto se debe a que la Alta Disponibilidad consiste en la capacidad de una arquitectura que proporciona servicios para continuar proporcionándolos de forma ininterrumpida de cara a los usuarios o a la propia empresa.

Para cumplir con los principios fundamentales de la Alta Disponibilidad se deben crear medidas para que en caso de que un componente de la estructura falle, ya sea por fallos en el software, en el hardware, etc., sea posible seguir proporcionando los servicios.

Debido a la situación del sector en la actualidad, también se tiene que tener en cuenta la posibilidad de constantes ataques cibernéticos que puedan atacar a los fallos en la seguridad e incluso intentar obtener información de la base de datos o de incluso los usuarios en ciertas situaciones, ya que es posible que también se tenga como objetivo el bloqueo del acceso a esa información o la eliminación.

Pese a lo anterior, es inevitable realizar interrupciones para el mantenimiento de los equipos, por lo que el objetivo para cumplir la alta disponibilidad en estos casos hace referencia a que dicho mantenimiento sea lo más breve posible.

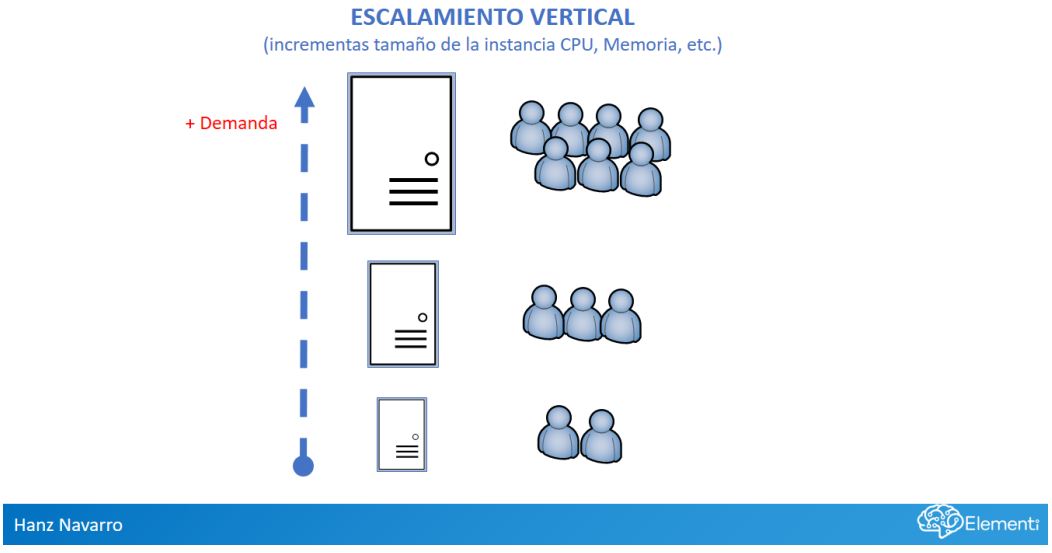
3.3.2 Escalabilidad

El concepto de escalabilidad en la informática se refiere a la capacidad de adaptación y respuesta de un sistema formado por diferentes equipos, como es el caso de un Cluster, ante su expansión debido a las necesidades del mismo. Esto puede ocurrir en el caso de que, por ejemplo, haya un volumen demasiado amplio de usuarios y se requieran de más equipos para poder soportar esa carga.

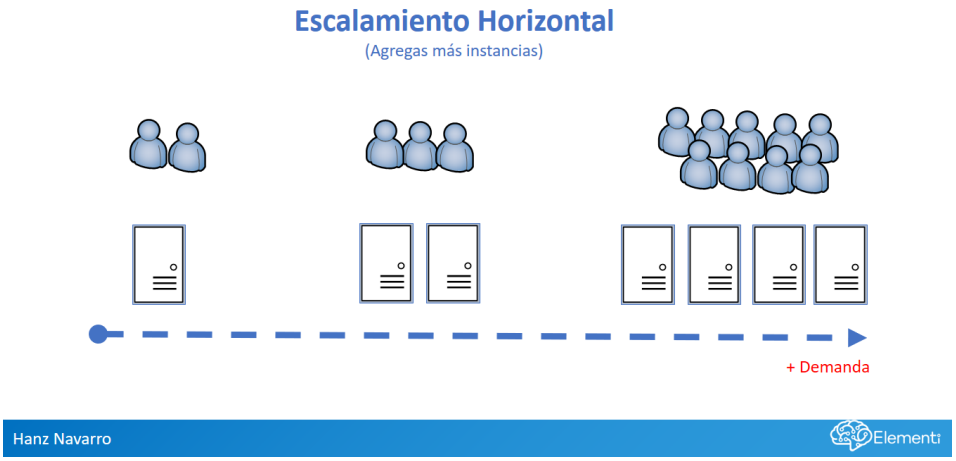
Esta escalabilidad puede darse en dos tipos:

- Escalabilidad vertical: escalar verticalmente se refiere a la capacidad de mejora de un nodo concreto en cuanto a su hardware, mejorando su disco duro, procesador, etc. Esto requiere que el resto de los equipos, pese a tener un Hardware con menores capacidades, puedan seguir funcionando en conjunto con el equipo con las nuevas características, produciendo el menor impacto posible.

  En el siguiente esquema se puede observar un ejemplo visual, suponiendo que el tamaño es equivalente a sus capacidades, vemos como un equipo es capaz de soportar una mayor carga al tener mayores capacidades, pero formando parte de una estructura conjunta.



- Escalabilidad horizontal: escalar horizontal por el contrario se refiere a la mejora general de la arquitectura, no de un único equipo. No se refiere a que se mejoran todos los equipos simultáneamente, se refiere al poder añadir nuevos equipos o nodos al sistema, para así hacer frente al aumento de carga u otras situaciones. Hay que tener en cuenta que, si bien añadir un nuevo equipo aumentará la Alta Disponibilidad, también puede requerir un mayor coste dependiendo de la situación.

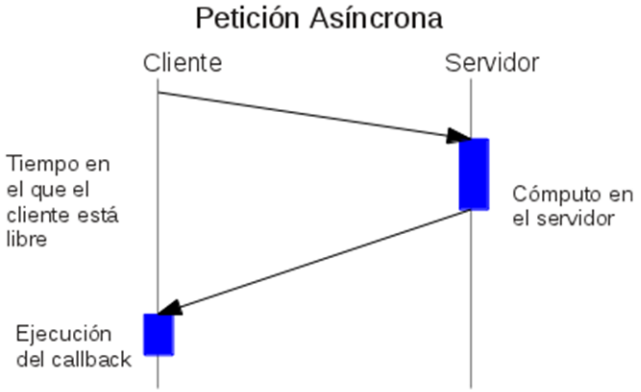


3.2.3 Asincronía

La asincronía es un concepto fundamental cuando se trata con arquitecturas cliente servidor. Las peticiones de este tipo nos permiten enviar una solicitud al servidor y recibir su respuesta y tratarla sin que la página tenga que ser refrescada.

Bajo este concepto suelen funcionar las peticiones de información a base de datos en todo tipo de páginas, las API RESTful, el tratamiento de datos en el servidor para devolverlos al cliente etc.

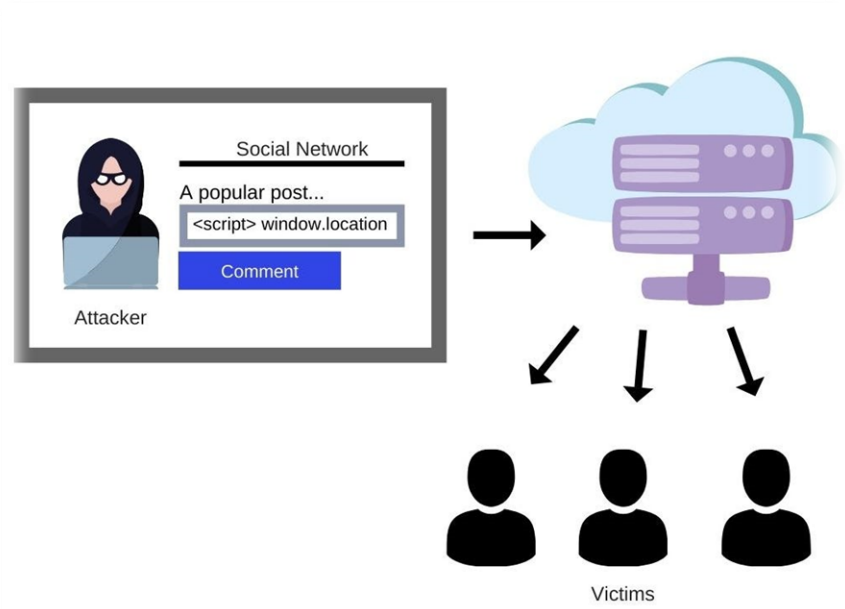
Una de las tecnologías más usada en Javascript para este tipo de peticiones, aunque actualmente está cayendo en desuso, es AJAX (Asynchronous JavaScript And XML), que será usado en este proyecto.



3.2.4 Inyecciones de código

Las inyecciones de código son unas de las vulnerabilidades más explotadas en las aplicaciones web. Suelen darse principalmente como inyecciones de JavaScript o SQL.

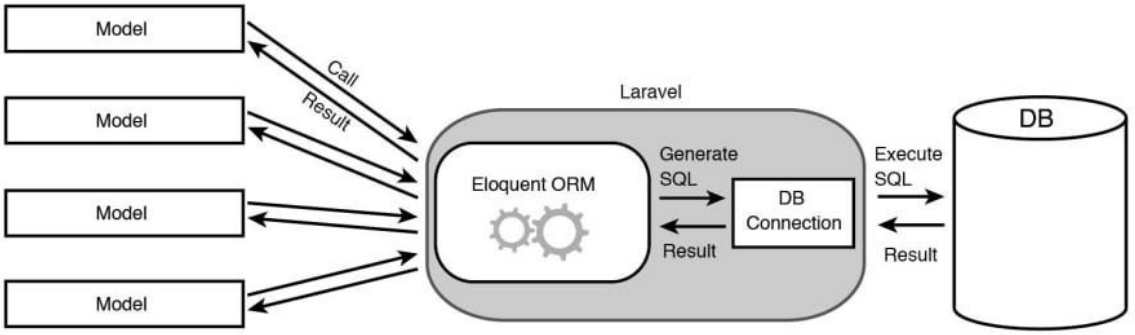
El método más utilizado para hacer inyecciones de JavaScript es el **Cross-site Scripting (XSS),** que consiste en la inyección de código JavaScript mediante inputs de formularios que no han sido validados, de forma que dicho código JavaScript es ejecutado, afectando a la víctima y en casos más graves, al servidor.



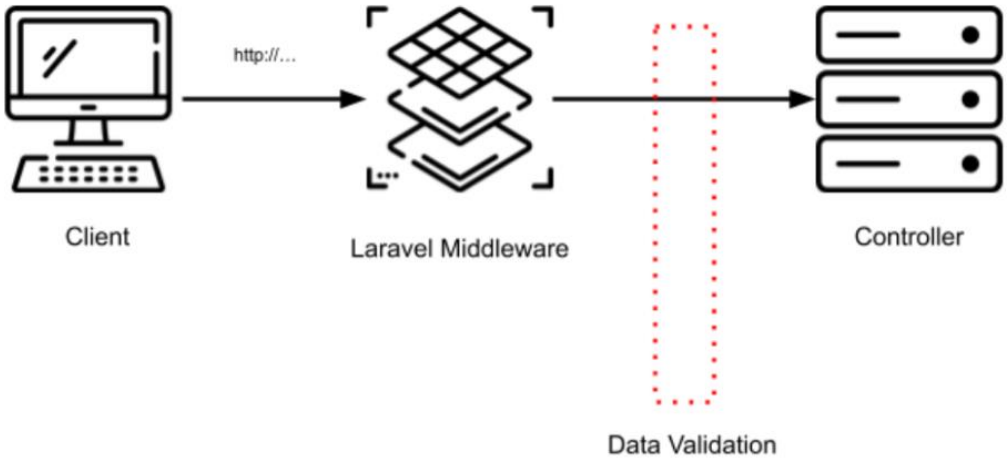
Las inyecciones SQL siguen un procedimiento similar, pero con el objetivo de atacar bases de datos, obteniendo, insertando o eliminando datos para perjudicar a los usuarios y los proveedores de la aplicación. Las consecuencias de no comprobar los datos introducidos pueden ser muy graves debido a lo anteriormente expuesto.

Para evitar estos ataques es fundamental realizar el filtrado y validación de todos los datos recibidos por el usuario, usando métodos que refuercen aún más esta seguridad como Laravel Validation para los datos introducidos por usuarios y los ficheros adjuntados, y Database: Query Builder y Eloquent para las inyecciones SQL.

- Eloquent y Query Builder:



- Validation:



3.3 Tecnologías y recursos empleados

3.3.1 Visual Studio Code

Visual Studio Code es uno de los editores de código creado por Microsoft con gran compatibilidad con todos los sistemas operáticos actuales.

Puede adaptarse a una amplia gama de lenguajes de programación gracias a la implementación de extensiones y paquetes que añaden funcionalidades, sugerencias de código, autocompletado, estilización del código, y un largo etc. de añadidos.

Este entorno ha sido empleado por Sergio Guerrero en el desarrollo de este proyecto.

3.3.2 PHPStorm

PHPStorm es, de forma similar a Visual Studio, un IDE de desarrollo creado por IntelliJ pero enfocado a PHP, como su propio nombre indica, añadiendo diversas funcionalidades muy útiles de base, como la compatibilidad con Docker, la implementación de un visor y editor de base de datos en el propio editor, etc.

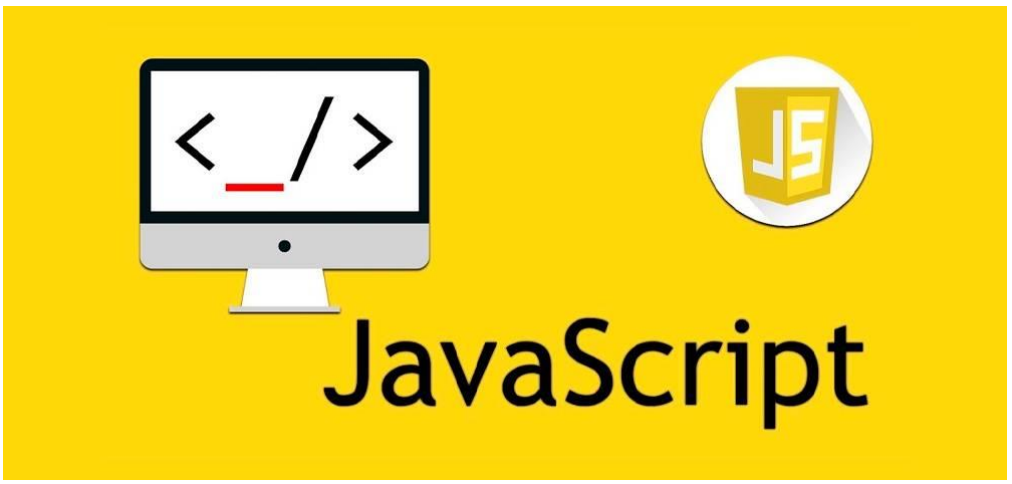
Este entorno ha sido empleado por Alejandro Cabot.

3.3.3 MySQL

MySQL es un sistema de gestión de base de datos relacionales de código abierto, basado en el modelo cliente-servidor y que permite la creación y manipulación de bases de datos. Es compatible en gran parte de los sistemas operativos, desde macOS a Windows y distribuciones de Linux, y es usado por muchas de las aplicaciones que tienen un mayor uso en la actualidad como son Google, Facebook, Twitter y YouTube entre otros, almacenando nombres de usuarios, descripciones, mensajes, etc.

Su lenguaje principal es SQL mediante el cual se pueden realizar diferentes operaciones de CRUD como la consulta y manipulación de datos (SELECT, INSERT, etc.) la definición del tipo de los datos y su cambio y el control de acceso a los datos mediante la creación de usuarios específicos

3.3.4 JavaScript



JavaScript es un lenguaje de programación orientado a objetos enfocado al entorno cliente. Es un lenguaje interpretado, débilmente tipado y creado con el enfoque de hacer a las páginas web interactivas.

Mediante JavaScript pueden realizarse peticiones asíncronas gracias a AJAX, así como complementarse con las etiquetas HTML mediante eventos que alteran el comportamiento y las características visuales de la web sin tener que cargarla nuevamente.

3.3.5 PHP

PHP es un lenguaje de programación orientado a objetos, usado principalmente en entorno de servidor. Sus siglas provienen de Hypertext Preprocessor, y está orientado al desarrollo web.

Permite la conexión a base de datos como MySQL y sus diversos frameworks facilitan el desarrollo web, como Laravel, que permite complementarse con vistas de lenguaje HTML y con código JavaScript.

3.3.6 Laravel

Laravel es un framework de código abierto enfocado en PHP. Está basado en la arquitectura Modelo Vista Controlador (MVC), dividendo el código dependiendo de su enfoque. Siendo los modelos enfocados a las conexiones con base de datos, las vistas como el aspecto visual de la web divido en diferentes componentes y los controladores siendo las clases que tratan los datos antes de enviarlos a las vistas, entre otras funciones de procesamiento.

El eje principal de este proyecto es dicho framework, en combinación con diferentes paquetes creados por los usuarios gracias al sistema de gestión de paquetes llamado Composer.

3.3.7 Canva

Canva es una aplicación de diseño gráfico, enfocado al diseño mediante plantillas para crear amplia variedad de elementos.

Su uso principal ha sido la creación del logotipo de la web, usando recursos de libre uso que proporciona la web.

3.3.8 AJAX

AJAX (Asynchronous JavaScript And XML) es otra de las bases del proyecto, ya que gracias a esta tecnología nos es posible realizar peticiones asíncronas, obteniendo información del lado de servidor para actualizar las vistas en tiempo real.

Uno de los usos principales de esta tecnología es la implementación de la misma en conjunto con DataTables, tablas de datos que reciben datos de la base de datos en tiempo real, cambiando la consulta según lo necesitado.

3.3.8 Laravel Blade

Laravel Blade es un motor de plantillas que puede implementarse junto a Laravel.

Permite incorporar “código” de PHP en las vistas, que usualmente solo tratan con código HTML y JavaScript mediante sencillas sentencias.

Es posible realizar la obtención de valores pasados a la vista desde el controlador, así como modificar visualmente las mismas con sentencias condicionales o con bucles.

3.3.9 Git y GitHub

Git es un sistema de control de versiones que permite la gestión de archivos y sus diferentes cambios, abarcando proyectos enteros. Estos cambios pueden dividirse en ramas, unirse comparando las modificaciones y un largo etcétera de funciones similares.

GitHub por su parte es una plataforma web que puede alojar de forma remota repositorios de Git, siendo una de las mayormente empleadas para ello. Para este proyecto se ha creado un repositorio de Git bajo el siguiente enlace, teniendo una rama principal y otra para cada uno de los integrantes del mismo:

https://github.com/SergioGB0702/ProyectoSergioAlejandro

Han sido fundamentales en este proyecto al ser un proyecto conjunto, ya que nos ha permitido ir realizando las actualizaciones de código por separado y después compararlas y unirlas (merge), cada uno con nuestra rama de desarrollo, pudiendo tomar fragmentos de las otras ramas y adaptándolos a medida que se desarrolla.

3.3.10 Composer

Composer es un sistema de gestión de paquetes que pueden ser implementados para PHP, en este caso con Laravel. Estas dependencias incluyen una gran cantidad de nuevas funciones preparadas por diferentes usuarios, así como mejoras de calidad de vida entre otros.

Los paquetes implementados en un proyecto de Laravel se guardan en un directorio llamado vendor suele ser ignorado a la hora de compartir el mismo, ya que pueden volver a implementarse gracias al contenido de los ficheros composer.json y composer.lock.

Debido a ello, si se necesitase modificar el código de algún paquete, debe publicarse primero, de forma que pase a formar parte del propio proyecto, como es el caso de la paginación.

3.4 Paquetes Composer implementados

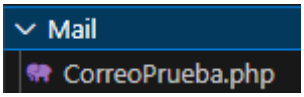
3.4.1 Sympfony Mailgun Mailer

Paquete implementado para el envio de correos desde la aplicación web, permite crear estructuras de correo y ser invocadas de forma similar a otras estructuras de datos de Laravel para.

Es necesario implementar en .env una cuenta de Google desde la que se realizarán los envíos, pudiendo ocultarla posteriormente personalizando el remitente. Esta cuenta de Google requerirá de cierta preparación, permitiendo el uso de la misma para terceros y usando una clave personalizada.

```
MAIL_MAILER=smtp
MAIL_HOST=smtp.gmail.com
MAIL_PORT=465
MAIL_USERNAME="sergioggbb02@gmail.com"
MAIL_PASSWORD=znzctlbgdpyfdsrq
MAIL_ENCRYPTION=ssl
MAIL_FROM_ADDRESS="sergioggbb02@gmail.com"
MAIL_FROM_NAME="${APP_NAME}"
```

Presenta una amplia variedad de formas de crear y enviar los correos, ya sea usando los métodos de la clase o usando vistas para personalizar el cuerpo de los mismos en formato HTML.



```php
public function envelope(): Envelope
{
    return new Envelope(
        from: new Address('sergio_gb02@hotmail.com', 'Sergio'),
        replyTo: [
            new Address('sergioggbb02@gmail.com', 'Sergio 2'),
        ],
        subject: 'Prueba correo',


    );
}
```

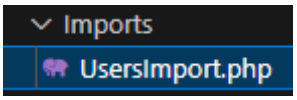
```php
/**
 * Get the message content definition.
 */
public function content(): Content
{
    return new Content(
        view: 'users.correo',
    );
}
```

```
ralumno.blade.php | correo.blade.php | index.blade.php | UsersController.php | Alumn
ProyectoSergioAlejandro > resources > views > users > correo.blade.php > ...
1   @extends('layouts.app')
2
3   @section('content')
4       <div class="container">
5           <div class="row justify-content-center">
6               <div class="col-md-8">
7                   <div class="card">
8                       <div class="card-header">{{ __('Dashboard') }}</div>
9
10                      <div class="card-body">
11                          @if (session('status'))
12                              <div class="alert alert-success" role="alert">
13                                  {{ session('status') }}
14                              </div>
15                          @endif
16
17                          {{ __('You are logged in!') }}
18
19
20                      </div>
21
22                  </div>
23                  <form action="{{route('user.enviarcorreo')}}" method="get">
24                      <div class="form-group">
25                          <input value="enviar correo" type="submit">
26                      </div>
27                  </form>
28              </div>
29          </div>
30      </div>
31  @endsection
```

3.4.2 Maatwebsite Excel

Paquete para importaciones y exportaciones con Excel.

Se pueden crear diferentes clases de importación, que se almacenan en el directorio Imports:



Si heredamos "WithHeadingRow" indicamos que el Excel recibido tendrá como primera fila títulos descriptivos de cada columna, pudiendo así asociar los datos.

```php
class UsersImport implements ToModel, WithHeadingRow
{
    /**
    * @param array $row
    *
    * @return \Illuminate\Database\Eloquent\Model|null
    */
    public function model(array $row)
    {
        return new User([
            'name'     => $row['name'],
            'email'    => $row['email'],
            'id'       => $row['id'],
            'password' => $row['password'],
        ]);
    }
}
```

En este método cada vez que se lee una fila del archivo Excel, este método se llama y hace lo siguiente:

- Recibe una fila del Excel como un array.
- Crea una nueva instancia del Modelo con los datos de la fila
- Devuelve esta nueva instancia del modelo, que luego se guardará en la base de datos.

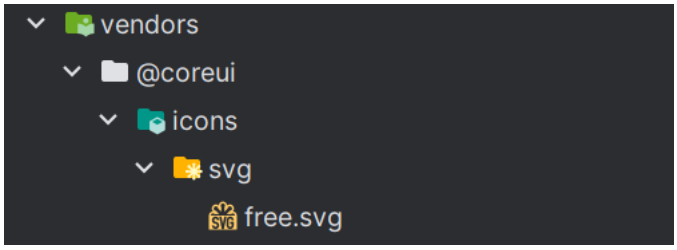
Esta clase facilita la tarea de importar un montón de usuarios desde un archivo Excel.

```php
public function import(Request $request)
{

    User::truncate();
    $file = $request->file( key: 'import_file');

    Excel::import(new UsersImport, $file);

    return redirect()->route( route: 'users.index')
        ->with('Satisfactorio', 'Usuario Creado correctamente.');
}
```

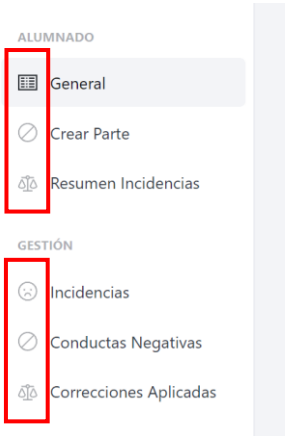
Después lo que haremos será crear un método en nuestro controlador que haga un truncate de la tabla para que elimine todos los datos y así se eviten duplicados.

Por lo otra parte llamamos a la Clase Excel y le pasamos como parámetro tanto la Clase de la Importación de Excel como el archivo Excel que deseemos cargar y con eso ya se insertaría en la base de datos

3.4.3 SVG



En el tema de los SVG de nuestra pagina usamos un paquete de coreui que nos trae una variedad de iconos bastante extensa:




```
<use xlink:href="{{asset('vendors/@coreui/icons/svg/free.svg#cil-ban')}}"></use>
```

Por ejemplo, en nuestro menú de navegación a través de una ruta relativa elegimos el svg que queramos llamando al free.svg y con un # eligiendo el nombre del svg que deseemos.

Estos SVG están adaptados para el modo oscuro que tenemos implementado en nuestra pagina ya que cambian de color para que se visualicen mejor.

3.5 Elementos Bootstrap

3.5.1 Switch

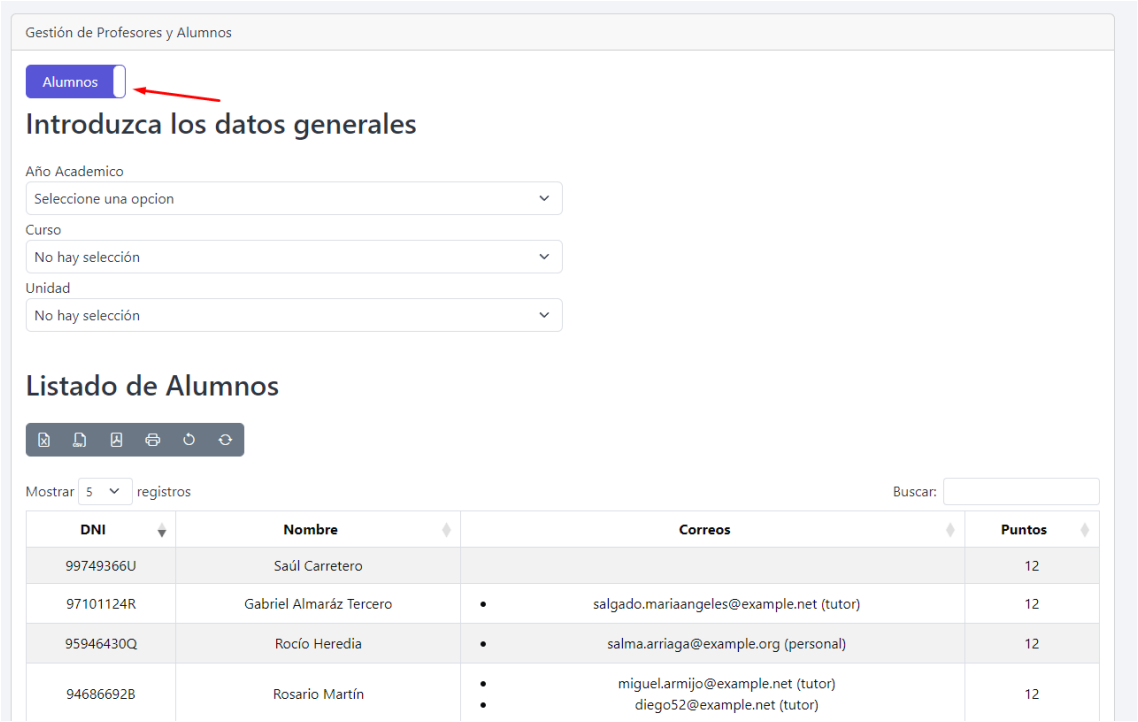
En las vistas planteadas, únicamente profesoralumno utiliza un Switch, esto es debido a que la distribución de la página depende de dicho Switch:

```
<input id="switchProfeAlumno" onchange="cambiarDiv();" type="checkbox"
data-on="Alumnos" data-off="Profesores"
checked data-toggle="toggle" data-onstyle="primary" data-offstyle="secondary">
```

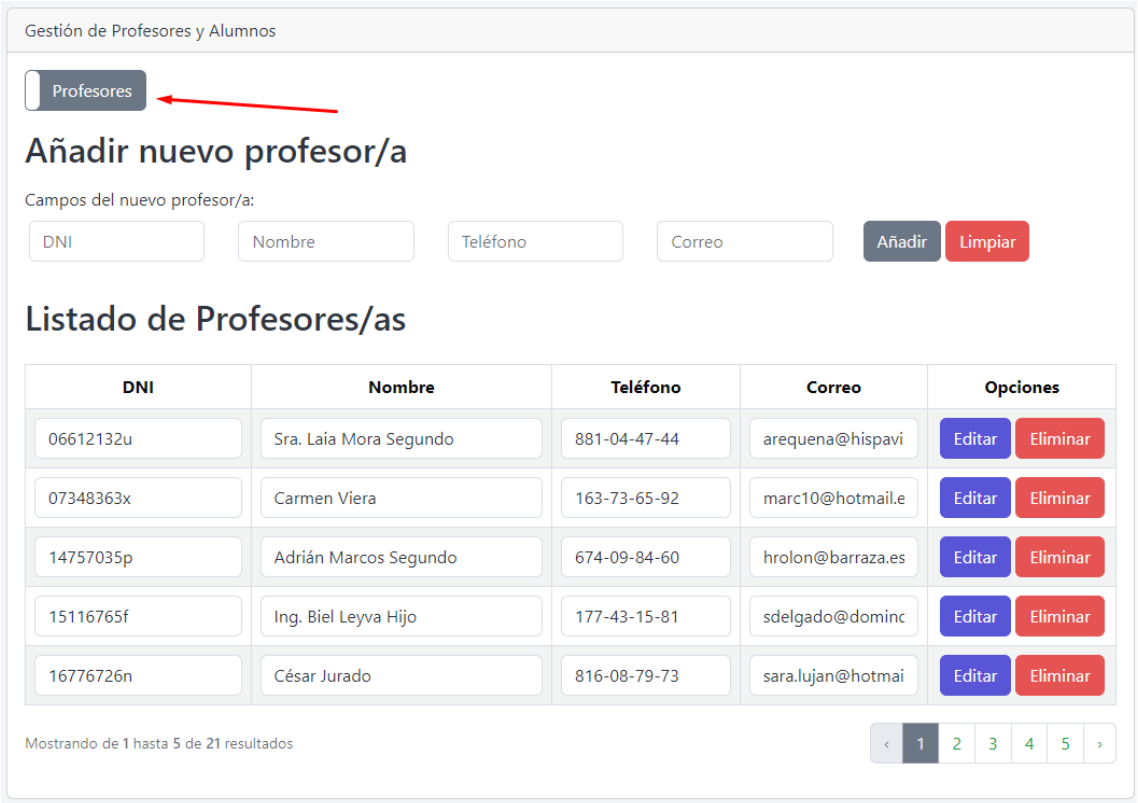
Al cambiar el valor del Switch, cambian el div que se muestra y aquel que se oculta. Alternándose entre el div centrado en los profesores y sus datos y aquel centrado en alumnos y sus datos:

```
function cambiarDiv() {
    const divProfesor = document.getElementById('divProfesor');
    const divAlumno = document.getElementById('divAlumno');
    const checked = document.getElementById('switchProfeAlumno').checked;

    if (checked) {
        divAlumno.style.display = 'block';
        divProfesor.style.display = 'none';
    } else {
        divAlumno.style.display = 'none';
        divProfesor.style.display = 'block';
    }
}
```

Al estar activado dicho input Switch, se oculta el div de profesor y se muestra el div de alumno, será el valor por defecto ya que serán los datos que más requerirán consultarse y editarse. Esta "parte" de la página es relativamente compleja al usar DataTable, que será explicado posteriormente:



Al estar desactivado se muestra el bloque de contenido del profesor, de menor complejidad:



4 Planteamiento e implementación

4.1 Relaciones

En este apartado se establecerán las relaciones entre los diferentes modelos que se han creado anteriormente mediante Eloquent. Para establecer estas relaciones se crean funciones en dichos modelos, haciendo referencia al otro modelo con el que se establece la relación, cambiando ligeramente según.

- Años académico 1:N Cursos:

AniosAcademico

```php
public function cursos () {
    return $this->hasMany(Curso::class);
}
```

1:N

Curso

```php
public function anioAcademico () {
    return $this->belongsTo(AnioAcademico::class);
}
```

- Curso 1:N Unidades:

Curso

```php
public function unidades () {
    return $this->hasMany(Unidad::class);
}
```

1:N

Unidad

```php
public function curso () {
    return $this->belongsTo(Curso::class);
}
```

- Unidad 1:N Alumnos:

Unidad

```php
public function alumnos () {
    return $this->hasMany(Alumno::class);
}
```

1:N

Alumno

```php
public function unidad () {
    return $this->belongsTo(Unidad::class);
}
```

- Alumno 1:N Correos:

Alumno

```php
public function correos () {
    return $this->hasMany(Correo::class);
}
```

1:N

Correo

```php
public function alumno () {
    return $this->belongsTo(Alumno::class);
}
```

- Alumnos N:M Partes (Tabla intermedia alumno_partes):

Alumno

```php
public function partes()
{
    return $this->belongsToMany(Parte::class,
    'alumno_partes', 'alumno_dni', 'dni');
}
```

N:M

Parte

```php
public function alumnos(): BelongsToMany
{
    return $this->belongsToMany(Alumno::class, 'alumno_partes');
}
```

- Profesor 1:N Partes:

Profesor

```php
public function partes () {
    return $this->hasMany(Parte::class);
}
```

1:N

Parte

```php
public function profesors () {
    return $this->belongsTo(Profesor::class);
}
```

- Partes N:M Incidencias (Tabla intermedia parte_incidencias):

Parte

```php
public function incidencias(): BelongsToMany
{
    return $this->belongsToMany(Incidencia::class,
    'parte_incidencias', 'parte_id', 'incidencia_id');
}
```

N:M

Incidencia

```php
public function partes()
{
    return $this->belongsToMany(Parte::class,
    'parte_incidencias');
}
```

- Partes N:M Correcciones aplicadas (Tabla intermedia parte_correccionaplicadas):

Partes

```php
public function correccionesaplicadas()
{
    return $this->belongsToMany(Correccionaplicada::class,
    'parte_correccionaplicadas');
}
```

N:M

Correccionaplicada

```php
public function partes()
{
    return $this->belongsToMany(Parte::class,
    'parte_correccionaplicadas');
}
```

- Partes N:M Conductas Negativas (Tabla intermedia parte_conductanegativas):

Partes

```php
public function conductanegativas()
{
    return $this->belongsToMany(Conductanegativa::class,
    'parte_conductanegativas');
}
```

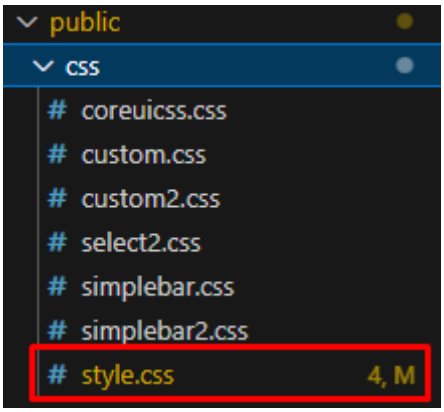
N:M

Conductanegativa

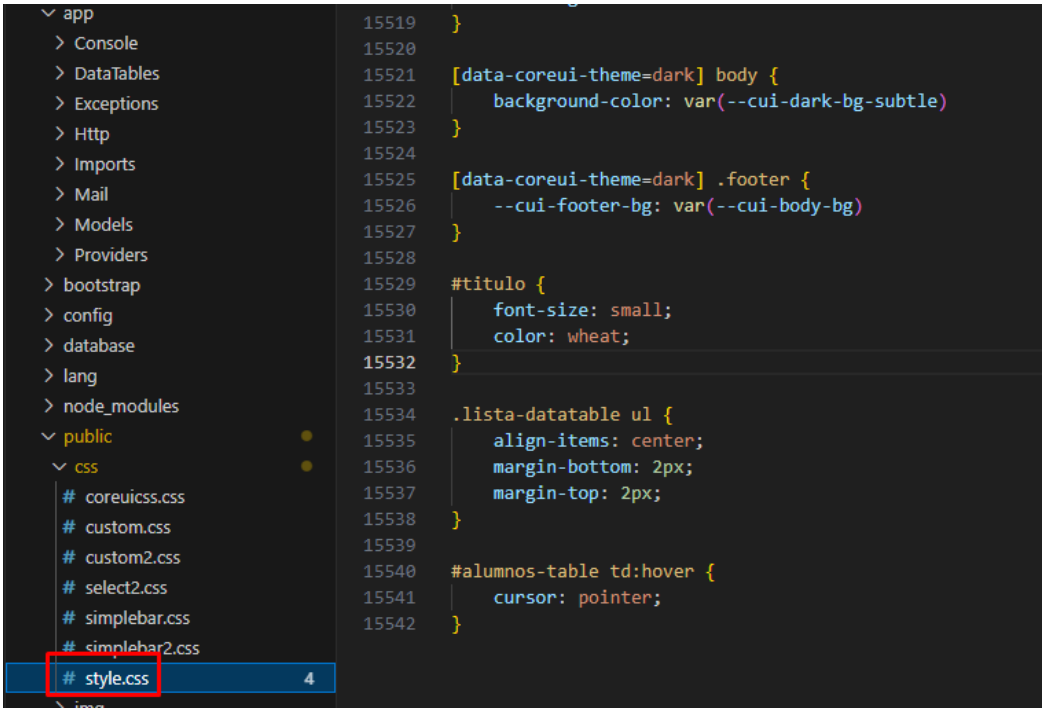
```php
public function partes()
{
    return $this->belongsToMany(Parte::class,
    'parte_conductanegativas');
}
```

4.2 Estilos y Bootstrap

Las clases de estilos de Bootstrap se han importado en el archivo style.css de public:



Siendo además donde añadiremos clases e identificadores personalizados, aunque cuando sea podrán editar las clases propias de Bootstrap:

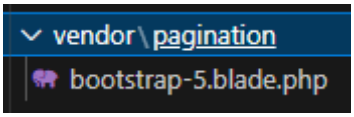


4.3 Paginator

Para editar la paginación automática de Laravel para los elementos consultados en la base de datos, se deberá publicar el respectivo contenido para que sea fácilmente editable y no se pierda al volver a implementar los paquetes del vendor, haciéndolo formar parte del propio proyecto:

```
php artisan vendor:publish --tag=laravel-pagination
```

Esto nos permitirá acceder a las vistas de paginación, en este caso utilizando la paginación de Bootstrap 5:

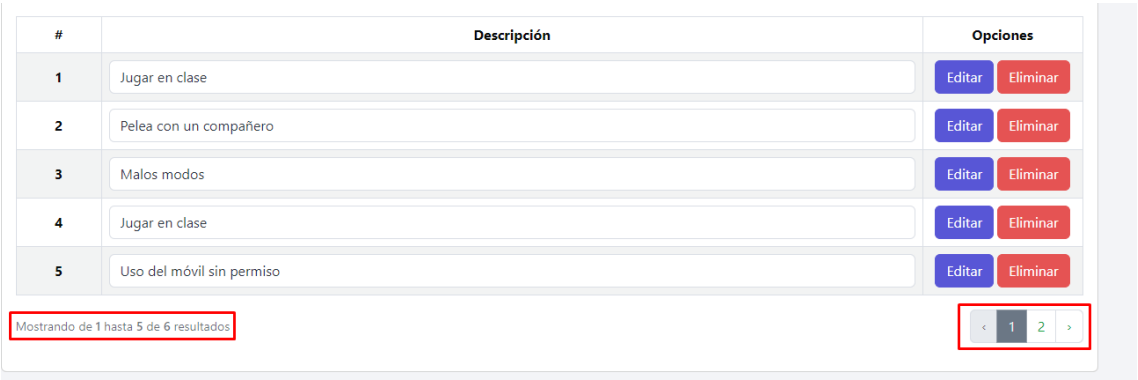


Los cambios principales serán la traducción de los elementos y los cambios en el estilo para que concuerden visualmente con el resto de la web:

```
        </li>
    @else
        <li class="page-item">
            <a class="page-link text-success" href="{{ $paginator->previousPageUrl() }}" rel="prev">« Anterior</a>
        </li>
    @endif

    {{-- Next Page Link --}}
    @if ($paginator->hasMorePages())
        <li class="page-item">
            <a class="page-link text-success" href="{{ $paginator->nextPageUrl() }}" rel="next">Siguiente »</a>
        </li>
    @else
```

Al ser un cambio concreto, se realiza directamente en la etiqueta HTML, siendo fundamental indicar con "!important" que el estilo definido debe tomar prioridad sobre los estilos de Bootstrap:

```
    @foreach ($element as $page => $url)
        @if ($page == $paginator->currentPage())
            <li class="page-item active" aria-current="page"><span class="page-link" style="background-color: #6b7785 !important; border-color: #979fa9 !important;">{{ $page
        @else
            <li class="page-item"><a class="page-link text-success" href="{{ $url }}">{{ $page }}</a></li>
        @endif
    @endforeach
endif
oreach

lext Page Link --}}
$paginator->hasMorePages())
i class="page-item">
  <a class="page-link text-success" href="{{ $paginator->nextPageUrl() }}" rel="next" aria-label="@lang('pagination.next')">&rsaquo;</a>
li>

i class="page-item disabled" aria-disabled="true" aria-label="@lang('pagination.next')">
```



4.4 Datatables

Como se ha explicado en el apartado sobre los paquetes de Composer, los DataTable son estructuras de datos creadas mediante JavaScript y que permiten rápidas consultas a la base de datos para obtener valores según sea conveniente, siendo el paquete utilizado únicamente un método de centralizar la declaración de los mismos.

Debido a esto, en este apartado se profundizará en los diferentes DataTables creados y sus especificaciones, uso y particularidades, dividiéndose según la explicación según la vista en la que han sido creados.

Para instanciar una DataTable, en primer lugar deberá pasarse a través del controlador, pasándose por parámetros pero realmente se instancia por si sola en ese momento. Para llamar a la vista pasando dicha instancia se usa la variable de los parámetros junto al método render, incluido por defecto:

```php
class ProfesorAlumnoController extends Controller
{
    public function index(AlumnoDataTable $dataTable, Request $request) {
        $anoAcademico = AnioAcademico::all();
        $profesores = Profesor::select("*");
        $tandaProfesores = $profesores->paginate(5);
        $paginaActual = $request->page;
        if ($paginaActual != null) {
            return $dataTable->render('gestion.profesoralumno', ['anoAcademico' => $anoAcademico,
        } else return $dataTable->render('gestion.profesoralumno', ['anoAcademico' => $anoAcademic
    }
```

Una vez en la propia vista, la tabla puede “desplegarse” en cualquier lugar del código HTML con la siguiente sentencia, pudiéndose adjuntar clases extras a la declaración:

```
<h2 class="mt-2 mb-4">Listado de Alumnos</h2>
<div class="">
    {{$dataTable->table(['class'=>'w-100 mb-2' ])}}
</div>
iv>
v id="divProfesor">
```

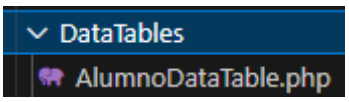
- En General:

- En Resumen Incidencias:

- En Alumnos / Profesores:

La vista de profesoresalumnos se divide en 2 partes, el contenedor centrado en los profesores que consta de una tabla normal, al no requerir demasiada complejidad, y por otra parte el contenedor de alumnos, cuyos datos están esparcidos por varias tablas y se ha implementado un DataTable para gestionar, filtrar y posteriormente editar dichos registros.

El primer aspecto que se explicará es la clase del Datatable AlumnosDataTable:



Al igual que las anteriores, consta de cuatro métodos principales, query, html, getColumns y dataTable.

- El método query se llama al instanciar la tabla, y se irá explicando su procedimiento poco a poco:

```php
public function query(Alumno $model): QueryBuilder
{
    $unidad = $this->request()->get('unidad');
    $searchTerm = $this->request()->get('search')['value'] ?? null;

    $query = $model->newQuery()

    ->leftJoin('correos', 'correos.alumno_dni', '=', 'alumnos.dni')
    ->select('alumnos.*', DB::raw("CONCAT('<ul><li>',
    GROUP_CONCAT(CONCAT(correos.correo, ' (', correos.tipo, ')') SEPARATOR '</li><li>'), '</li></ul>') as Correos"))
    ->groupBy('alumnos.dni');


    if (!empty($unidad)) {

        $query->where('alumnos.id_unidad', '=', $unidad);
    }

    if (!empty($searchTerm)) {
        $model->query()->where('alumnos.dni', 'like', "%{$searchTerm}%")
              ->orWhere('alumnos.nombre', 'like', "%{$searchTerm}%")
              ->orWhere('alumnos.dni', 'like', "%{$searchTerm}%");
    }

    return $query;
```

En primer lugar se obtienen los valores del campo de búsqueda incluido por defecto en DataTable, así como la unidad si se ha rellenado en el select de la vista:

```php
public function query(Alumno $model): QueryBuilder
{
    $unidad = $this->request()->get('unidad');
    $searchTerm = $this->request()->get('search')['value'] ?? null;
```

Tras esto se establece la consulta base de la base de datos, siendo principalmente hacia la tabla de alumnos siendo el modelo base utilizado. También se establece un leftJoin hacia la tabla correos con el DNI, creando así una columna que agrupa todos los correos (y su tipo) asociados a ese DNI de alumno a modo de lista con etiquetas HTML:

```php
$query = $model->newQuery()

->leftJoin('correos', 'correos.alumno_dni', '=', 'alumnos.dni')
->select('alumnos.*', DB::raw("CONCAT('<ul><li>', GROUP_CONCAT(CONCAT
(correos.correo, ' (', correos.tipo, ')') SEPARATOR '</li><li>'), '</li></ul>') as Correos"))
->groupBy('alumnos.dni');
```

Tras esto se comprueba si existe valor introducido en el filtro y si existe valor introducido en el Input Select de unidad. En el caso de la unidad, es una condición “where” simple, siendo el filtro algo más complejo, al buscar en las columnas de DNI y nombre la cadena introducida independientemente de su posición con “like” y una expresión regular:

```php
if (!empty($unidad)) {

    $query->where('alumnos.id_unidad', '=', $unidad);
}

if (!empty($searchTerm)) {
    $model->query()->where('alumnos.dni', 'like', "%{$searchTerm}%")
          ->orWhere('alumnos.nombre', 'like', "%{$searchTerm}%");
}
```

- En el método dataTable se más particularidades de la tabla que deseamos crear, obteniendo la consulta (query) del método anterior:

```php
public function dataTable(QueryBuilder $query): EloquentDataTable
{
    return (new EloquentDataTable($query))
        ->setRowId('dni')
        ->rawColumns(['Correos'])
        ->filterColumn('nombred', function($query, $keyword) {
            $query->whereRaw("alumnos.nombre like ?", ["%{$keyword}%"]);
        })
        //Para que pueda buscar por los correos también
        ->filterColumn('Correos', function($query, $keyword) {
            $query->where(function($query) use ($keyword) {
                $query->where('correos.correo', 'like', "%{$keyword}%");
            });
        });
}
```

En primer lugar establecemos el id de la tabla, que será el DNI al ser la clave primaria de alumno. También se establece la columna Correos anteriormente explicada como rawColum, lo que significa que su contenido será retratado explícitamente en el código HTML, gracias a esto se mostrará a modo de lista, al haber incluido las etiquetas en la concatenación anterior:

```php
return (new EloquentDataTable($query))
    ->setRowId('dni')
    ->rawColumns(['Correos'])
```

• salgado.mariaangeles@example.net (tutor)
• salma.arriaga@example.org (personal)
• miguel.armijo@example.net (tutor) • diego52@example.net (tutor)
• vera.guerrero@example.com (personal) • lola.quezada@example.org (personal) • gael16@example.org (tutor)

También se establecen algunos filtros especiales, siendo fundamental para la columna de Correos al tener un formato tan especial, ya que de lo contrario el filtro generaría problemas:

```
->filterColumn('nombred', function($query, $keyword) {
    $query->whereRaw("alumnos.nombre like ?", ["%{$keyword}%"]);
})
//Para que pueda buscar por los correos también
->filterColumn('Correos', function($query, $keyword) {
    $query->where(function($query) use ($keyword) {
        $query->where('correos.correo', 'like', "%{$keyword}%");
    });
});
```

- getColumns establece las columnas del DataTable y permite asociar clases, crear el título de la misma y asociar el contenido obtenido de la consulta anterior con cada columna mediante Data:

```
public function getColumns(): array
{
    return [
        Column::make('dni')->title('DNI')->data('dni')->className('align-middle text-center'),
        Column::make('nombre')->name('alumnos.nombre')->title('Nombre')->data('nombre')->className('align-middle text-center'),
        Column::make('Correos')->name('Correos')->title('Correos')->data('Correos')->className('align-middle text-center lista-datatable'),
        Column::make('puntos')->title('Puntos')->data('puntos')->data('puntos')->className('align-middle text-center')->searchable(false),
    ];
}
```

- El método html establece otras propiedades a la hora de construir la tabla:

```php
public function html(): \Yajra\DataTables\Html\Builder
{
    return $this->builder()
        ->setTableId('alumnos-table')
        ->columns($this->getColumns())
        ->minifiedAjax()

        ->orderBy(0)
        ->scrollX(true)
        ->language(['url' => '/js/spanish.json'])

        ->parameters([
            'pageLength' => 5, // Limitar los registros a 5 por página
            'lengthMenu' => [[5, 10, 25, 50], [5, 10, 25, 50]],
        ])
        ->buttons([
            Button::make('excel')->titleAttr('Exportar a Excel'),
            Button::make('csv')->titleAttr('Exportar a CSV'),
            Button::make('pdf')->titleAttr('Exportar a PDF'),
            Button::make('print')->titleAttr('Imprimir'),
            Button::make('reset')->titleAttr('Restablecer'),
            Button::make('reload')->titleAttr('recargar'),
        ]);
```

En primer lugar se establece el id de la tabla html, fundamental para posteriores operaciones. Se establecen también las columnas mediante el método anterior y se indica el uso de AJAX con minifiedAjax:

```php
return $this->builder()
    ->setTableId('alumnos-table')
    ->columns($this->getColumns())
    ->minifiedAjax()
```

Se establecen características de estilo como la ordenación, el lenguaje y la posibilidad de moverse por el contenido si la pantalla es demasiado pequeña:

```php
->orderBy(0)
->scrollX(true)
->language(['url' => '/js/spanish.json'])
```

Y se añaden parámetros especiales, en este caso la paginación de 5 en 5, y la instancia de botones con funciones añadidas.

```
->parameters([
    'pageLength' => 5, // Limitar los registros a 5 por página
    'lengthMenu' => [[5, 10, 25, 50], [5, 10, 25, 50]],
])
->buttons([
    Button::make('excel')->titleAttr('Exportar a Excel'),
    Button::make('csv')->titleAttr('Exportar a CSV'),
    Button::make('pdf')->titleAttr('Exportar a PDF'),
    Button::make('print')->titleAttr('Imprimir'),
    Button::make('reset')->titleAttr('Restablecer'),
    Button::make('reload')->titleAttr('recargar'),
]);
```

Bibliografía

Sympfony Mailgun Mailer:

https://laravel.com/docs/11.x/mail#configuration

https://www.youtube.com/watch?v=oq1Qmfj3ADo

Maatwebsite Excel:

https://laravel-excel.com/

https://www.youtube.com/watch?v=znyFsVydALY&pp=ygURbWFhdHdlYnNpdGUvZXhjZWw%3D

Inyecciones de código:

Inyección de código JavaScript | KeepCoding Bootcamps

¿Qué es la inyección de SQL? | Explicación y protección | Avast

Laravel Validator:

Validation - Laravel 11.x - The PHP Framework For Web Artisans

Laravel Query Buider:

Database: Query Builder - Laravel 11.x - The PHP Framework For Web Artisans

Eloquent:

https://laravel.com/docs/11.x/eloquent#generating-model-classes

https://jelledev.com/laravel-how-to-update-all-rows-in-table/

API RESTFul:

https://aws.amazon.com/es/what-is/restful-api/

https://www.neoguias.com/api-rest/

Laravel Datatable:

https://yajrabox.com/docs/laravel-datatables/master/engine-eloquent

https://laracasts.com/discuss/channels/laravel/change-laravel-datatable-text

Boostrap:

https://getbootstrap.com/

https://codersfree.com/posts/pasos-instalar-bootstrap-en-laravel

Bootstrap Toogle:

https://gitbrent.github.io/bootstrap4-toggle/

Documentación del curso