

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

Sergio Guerrero Borrero

Alejandro Cabot Caparrós



IES SAN SEBASTIÁN 2024
CICLO FORMATIVO DE GRADO SUPERIOR DESARROLLO DE APLICACIONES WEB

HOJA RESUMEN-PROYECTO

Título del proyecto: Sistema de Gestión de Partes para el centro IES San Sebastián (S.G.P S.S)	
Autores: Sergio Guerrero Borrero Alejandro Cabot Caparrós	Fecha: 10/06/2024 Curso 2023-2024
Tutores: Manuel Alfonso Romero Caro / Manuel Jesús Iglesias Espina	
Titulación: C.F.G.S Desarrollo de Aplicaciones Web	
Palabras clave: Laravel, PHP, JavaScript, MySQL, Composer, Bootstrap, AJAX, Diseño Web, asincronía, educación, programación, bases de datos	
<p>Resumen del Proyecto (Español): La memoria de este Proyecto está constituida por varias partes, comenzando por una introducción general, un estudio general sobre la situación y el porqué de la necesidad de la realización de este proyecto, así como aspectos de su planificación, tratándose de una aplicación web para llevar toda la gestión de los partes educativos de forma digital.</p> <p>Tras esto se expondrá la parte teórica del proyecto, en cuanto a los diferentes estudios realizados sobre requisitos, casos de uso, etc. así como las tecnologías, recursos y componentes empleados. Tras esto se tratará en profundidad la implementación de todo lo anteriormente mencionado en la propia aplicación, así como una explicación detallada de cada vista y su controlador principalmente asociado.</p> <p>Por último se realizarán las conclusiones finales, reflexionando sobre posibles mejoras que podrían haber sido implementadas y nuevas funcionalidades interesantes que no han llegado a implementarse.</p> <p>Resumen del Proyecto (Inglés): This document is formed by different sections, the first of which being a general introduction to this Application, its objectives and the main motivating factors for both developers. After that, we focus on the reasons why this App has become necessary for the current situation in the educational enviroment, and other topics such as the planning for the devolpment and its cost.</p> <p>We then focus on the study of the necessities that need to be supplied by the system, the used technologies, components and assets used, etc. The next main topic is the implementation of the assets previously mentioned, focusing on Views and their main Controllers.</p> <p>This is followed by the ending appraisals, the possibilities for improving the App and new functionalities that could be included in the future.</p>	

Índice

HOJA RESUMEN-PROYECTO	1
1. INTRODUCCIÓN.....	1
1.1 Introducción a la memoria	1
1.2 Descripción.....	1
1.3 Objetivos generales.....	1
1.4 Motivaciones personales	2
1.5 Estructura de la memoria.....	2
1.6 Repositorio de GitHub.....	3
2. ESTUDIO DE LA VIABILIDAD.....	3
2.1 Introducción	3
2.2 Estudio de la situación actual.....	4
2.3 Objetivos del Proyecto	4
2.4 Planificación del Proyecto	5
2.5 Presupuesto	7
3. ANÁLISIS DEL PROYECTO.....	8
3.1 Introducción	8
3.2 Conceptos clave	8
3.2.1 Alta Disponibilidad	8
3.2.2 Escalabilidad.....	9
3.2.3 Asincronía.....	11
3.2.4 Inyecciones de código	11
3.2.5 API Rest	14
3.3 Tecnologías y recursos empleados.....	15
3.3.1 Visual Studio Code.....	15
3.3.2 PHPStorm	16
3.3.3 MySQL	16
3.3.4 JavaScript.....	17

3.3.5 PHP	18
3.3.6 Laravel	18
3.3.7 Canva	19
3.3.8 AJAX.....	20
3.3.8 Laravel Blade	20
3.3.9 Git y GitHub	21
3.3.10 Sanctum.....	22
3.3.11 Composer	22
3.4 Paquetes Composer implementados	23
3.4.1 Symfony Mailgun Mailer	23
3.4.2 Maatwebsite Excel	29
3.4.3 SVG	31
3.4.4 Spatie/laravel-permission	32
3.5 Elementos Bootstrap y CKEditor	33
3.5.1 Switch	33
3.5.2 CKEditor	35
3.6 Casos de uso	38
3.7 Diagramas de secuencia	39
3.7.1 Rol profesorado (Acceso general con logueo)	40
3.7.2 Rol jefatura	43
4. Planteamiento e implementación.....	47
4.1 Base de datos y relaciones	47
4.2 Estilos y Bootstrap.....	49
4.3 Paginator	50
4.4 Datatables	52
4.5 Preparación de roles e implementación de Middleware.....	61
4.6 Mailer	65
4.7 Excel importaciones	68

4.8 Vistas, estructura y comportamiento general	70
4.8.1 Inicio de sesión	70
4.8.2 Alumnado: Partes	71
4.8.3 Alumnado: Resumen Incidencias	73
4.8.4 Gestión: Incidencias	73
4.8.5 Gestión: Conductas negativas	75
4.8.6 Gestión: Correcciones	76
4.8.7 Gestión: Alumnos / Profesores	77
4.8.8 Gestión: Reiniciar puntos	81
4.8.9 Gestión: Importaciones	83
5. Pruebas y métodos por controlador	83
5.1 Introducción	83
5.2 Alumnado	84
5.2.1 UsersController	84
5.3 Gestión	89
5.3.1 IncidenciasController	89
5.3.2 ConductasNegativasController	92
5.3.3 CorreccionesAplicadasController	95
5.3.4 ProfesorAlumnoController	98
5.3.5 ApiController	102
6. Conclusiones generales	108
6.1 Conclusiones finales	108
6.2 Posibles ampliaciones y modificaciones	108
6.3 Valoraciones personales	109
Bibliografía	110



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 hipotético 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, Javascript y AJAX para el desarrollo web en la parte cliente y la comunicación con el servidor y la base de datos mediante peticiones asíncronas (Con API Rest). Así como otros aspectos generales de diseño que hemos tratado a lo largo del curso, como son la implementación de CSS y Bootstrap con el objetivo de hacer una web usable, educativa y visual, haciendo a su vez que sea responsive para su uso en todo tipo de pantallas.

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. Las funciones generales a conseguir son la realización de partes, la consulta y modificación de sus elementos, y el envío de la información a los correos asociados al alumno afectado.



Relacionado con esto, la información de cada parte deberá poder exportarse a PDF e implementar los datos del alumno en la web con un fichero Excel.

1.4 Motivaciones personales

Debido al encargo de esta aplicación, ambos autores se han sentido muy alabados al confiarles 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 casi 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, tratando además muchas otras cuestiones vistas a lo largo del grado.

En su módulo de FCT ambos integrantes de este proyecto han tratado el desarrollo en Laravel en sus respectivas empresas, por lo que el desarrollo simultáneo con este proyecto se complementa con sus funciones realizadas en ellas. Gracias a esto se han podido implementar diversas funcionalidades con mayor eficacia, como el envío de correos en segundo plano o la implementación de las tablas de datos llamadas Datatables, que se actualizan en tiempo real con peticiones asíncronas.

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.

Tras esto comenzará el apartado del Análisis del proyecto, en el que expondremos todos los elementos usados, conceptos necesarios y tecnologías empleadas, así como exponiendo de forma general los elementos añadidos de fuentes externas, como los paquetes de Composer importados o los elementos de Bootstrap algo más complejos que requieran una explicación más profunda.

Una vez finalizado el análisis del proyecto, comenzará el apartado de su desarrollo, es decir, tanto la implantación como una explicación más detallada de todos los elementos usados, las vistas preparadas, etc. Tras el análisis general de las vistas, se tratará en profundidad cada controlador y para qué se utiliza cada método, junto con las correspondientes pruebas.

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 de ambos



integrantes, y tras esto encontraremos el apartado de bibliografía, en los que se mostrarán las diferentes fuentes de información que han sido utilizadas y consultadas en el desarrollo.

1.6 Repositorio de GitHub

GitHub ha sido el sistema de control de versiones elegido para el desarrollo de este proyecto, gracias a su compatibilidad con ambos entornos de desarrollo empleados. Ha sido especialmente útil al ser un proyecto desarrollado por dos integrantes, pudiendo desarrollar diferentes ramas simultáneamente y unir los cambios cuando fuese necesario.

El repositorio utilizado para el proyecto ha sido el siguiente, teniendo una rama principal y una para cada integrante:

<https://github.com/SergioGB0702/ProyectoSergioAlejandro.git>

2. ESTUDIO DE LA VIABILIDAD

2.1 Introducción

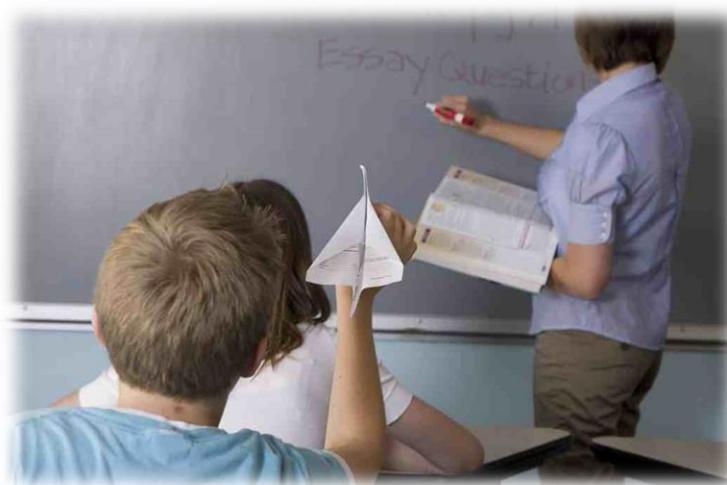
En este apartado se tratará el estudio de la viabilidad del proyecto, en otras palabras, la valoración de la situación actual y la razón de por qué es necesario. También se expondrá la planificación de la misma en cuanto a todas las funciones implementadas y el tiempo aproximado que se ha empleado en 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, lo que ha ido incrementando la dificultad de realizar estas gestiones a papel y de tener que depender de un servicio externo de la Junta de Andalucía. Estos casos se están dando a nivel general en el sector educativo, enfocándose sobre todo 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 Objetivos del Proyecto

Tras las primeras especificaciones dadas por el profesorado, se han concluido que los principales objetivos del proyecto son los siguientes:

- Desarrollo de una aplicación web para la gestión de partes de forma sencilla y eficaz.
- Poder realizar exportaciones de la información de dichos partes a voluntad.
- Permitir la importación y exportación de los datos de los alumnos en archivos Excel.
- Gestión de los alumnos y sus datos de contacto, así como de sus puntos del sistema llevado por el centro para regular su comportamiento.
- El envío de correos a los contactos asociados al alumno tras la realización de un parte de forma automática y en otras situaciones como en la pérdida total de puntos.



- La realización de una aplicación totalmente responsive, ya que gran parte de su uso será en teléfonos móviles.

2.4 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 (16 horas)	Planteamiento de necesidades Investigación de recursos para implementarlas
Diseño (30 horas)	Generación de la estructura general de la aplicación Implementación de los diferentes componentes y recursos Planteamiento de responsividad y funcionamiento de las peticiones asíncronas (AJAX/API Rest)



Desarrollo (120 horas)	Desarrollo de las consultas necesarias para toda la aplicación Creación de las vistas y su código JavaScript necesario, así como sus estilos con Bootstrap y CSS Desarrollo de métodos de controladores y API RestFul Creación de autentificación de usuarios mediante Spatie para web y Sanctum para API
Pruebas (40 horas)	Validaciones de datos Revisión de operaciones CRUD Comprobación de correcto funcionamiento de Middleware y datos de autentificación Correcciones de funcionamiento
Despliegue (8 horas)	Preparación de lanzamiento con Node simulando entorno de producción Corrección de errores debido a la variedad en el entorno
Operaciones / Pruebas con la aplicación desplegada (12 horas)	Mismas comprobaciones que en Pruebas, pero en entorno de producción Corrección de fallos mínimos

Realmente el tiempo empleado en cada una de las partes es mayor, al haber sido dos integrantes en el desarrollo, y haber aliviado la carga que podría haber llevado una persona en mayor tiempo. Además, no se ha tenido en cuenta el tiempo empleado para la documentación, que sumaría alrededor de 30 horas más.

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



2.5 Presupuesto

Todas las herramientas usadas para el desarrollo del proyecto son gratuitas, por lo que el empleo de las mismas no tiene coste. Pudiendo incluso desplegar la aplicación en un servidor que se prepare en el propio instituto pero únicamente accesible desde la red local, a elección del profesorado.

El coste estimado en caso de emplear un servidor de dominio, que sería lo más adecuado, son de 7€ mensuales, aunque depende mucho del proveedor y de los recursos contratados.

Además de esto, suponiendo un caso de desarrollo real, se podría hacer la siguiente estimación por el coste del tiempo y esfuerzos empleados, teniendo en cuenta que el pago mínimo por hora en España es actualmente 7,88€:

Requisitos (16 horas)	126,08€
Diseño (30 horas)	236,4€
Desarrollo (120 horas)	945,6€
Pruebas (40 horas)	315,2€
Despliegue (8 horas)	63,04€
Operaciones / Pruebas con la aplicación desplegada (12 horas)	94,56€
Total de coste estimado (226 horas)	1.780,88€

Sumando el tiempo invertido en documentación, si fuese necesario tomarlo en cuenta, serían 256 horas y por lo tanto el desarrollo pasaría a tener un coste estimado total de 2.017,28€.



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 asincronía y las API Rest, para después explicar las tecnologías existentes y los recursos empleados, así como los elementos de Bootstrap más complejos utilizados y contando con un apartado más extenso 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 ciberneticos 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

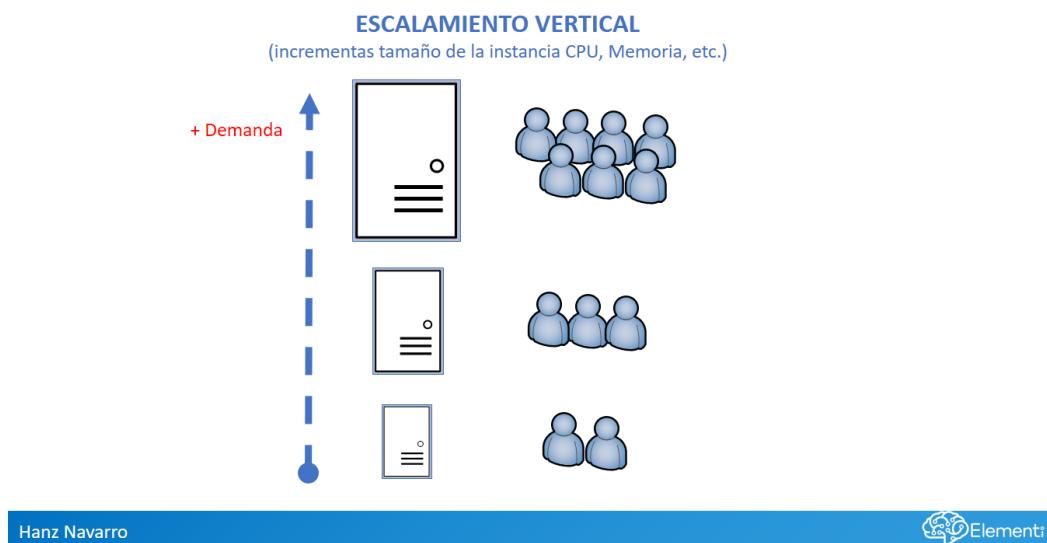
El concepto de escalabilidad en la informática se refiere a la capacidad de adaptación y respuesta de un sistema formado por diferentes equipos ante su expansión debido a las necesidades del mismo. Esto puede ocurrir en el caso de que, por ejemplo, haya un volumen demasiado amplio de usuarios y se requieran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

Esta escalabilidad puede darse en dos tipos:

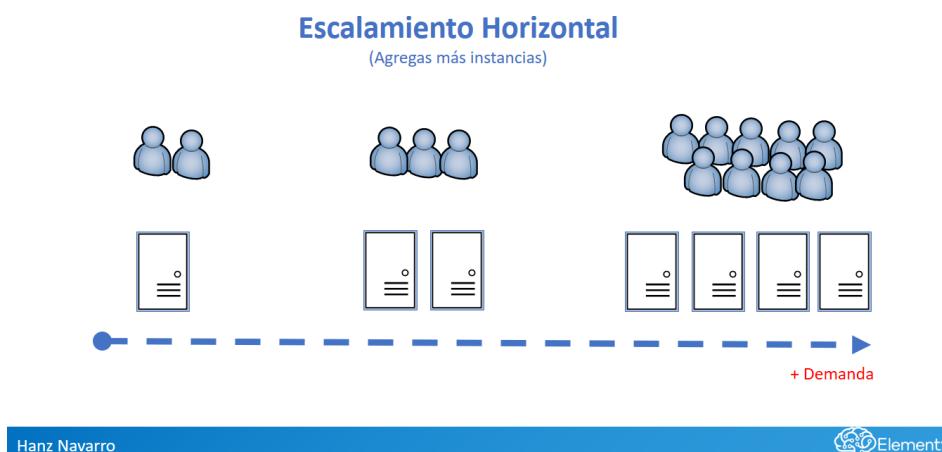
- Escalabilidad vertical: escalar verticalmente se refiere a la capacidad de mejora de un nodo concreto en cuanto a su hardware, mejorando su disco duro, procesador, etc. Esto requiere que el resto de los equipos, pese a tener un Hardware con menores capacidades, puedan seguir funcionando en conjunto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

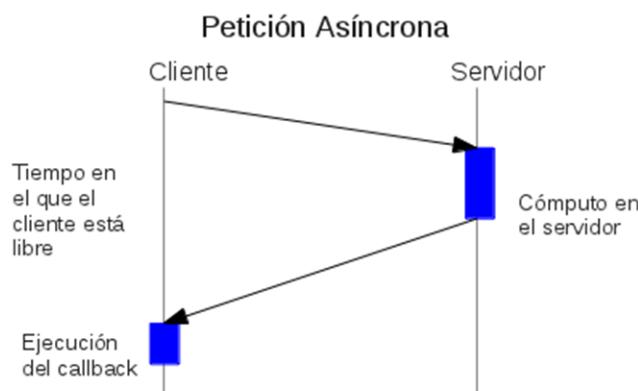
La asincronía es un concepto fundamental cuando se trata con arquitecturas cliente-servidor.

Las peticiones de este tipo nos permiten enviar una solicitud al servidor desde el equipo del cliente y recibir su respuesta y tratarla sin que la página tenga que ser refrescada, realizando las actualizaciones mediante, por ejemplo JQuery o Javascript con DOM.

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 usadas 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.

AJAX nos permite realizar estas peticiones desde código JavaScript, enviando datos al servidor y obteniendo una respuesta acorde al tratamiento que se le ha dado, dependiendo de si ha sido exitosa o no.

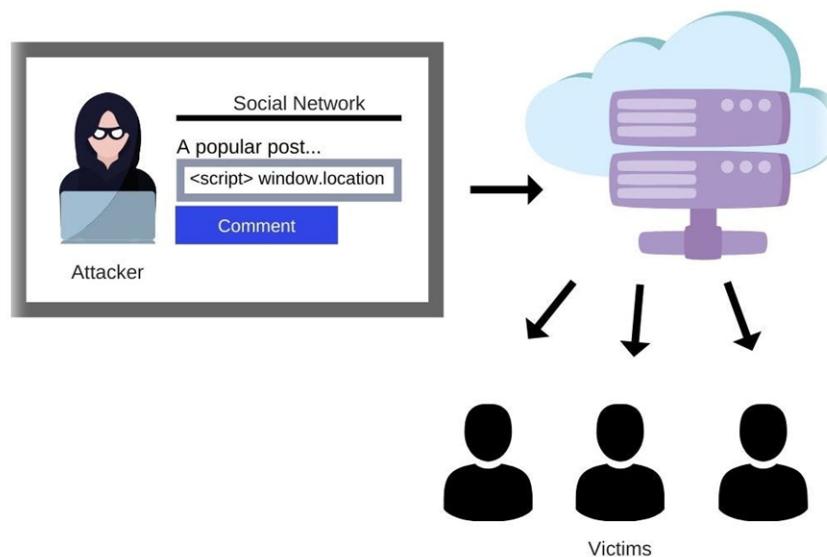


3.2.4 Inyecciones de código

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.

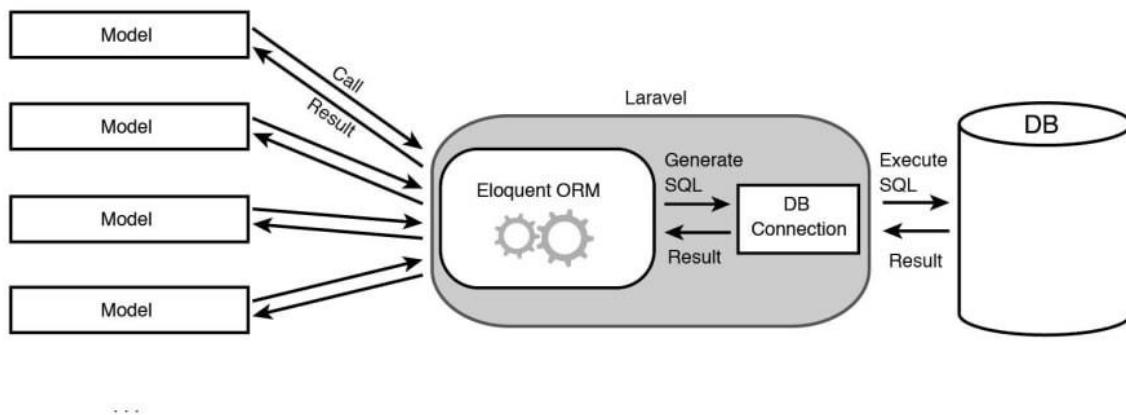




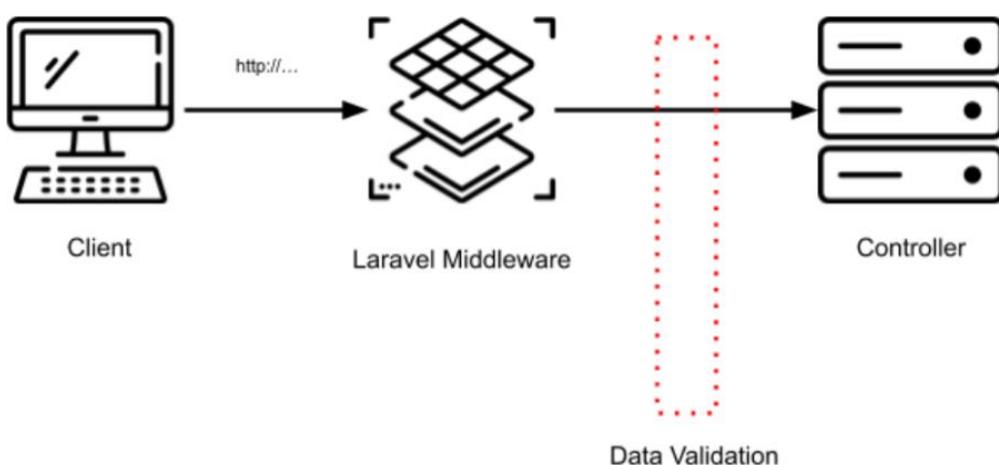
SISTEMA DE GESTIÓN DE PARTES

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:

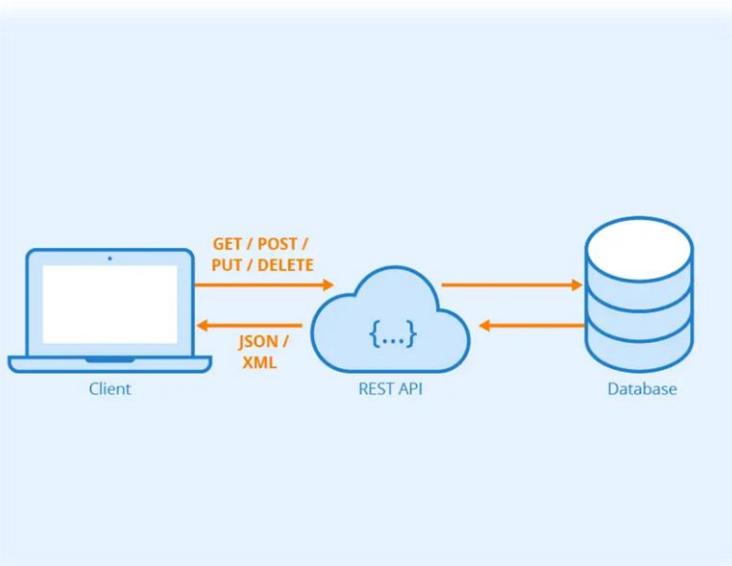


- Validation:





3.2.5 API Rest



Una API es una interfaz de programación de aplicaciones (del inglés Application Programming Interface), es decir, es un conjunto de reglas y protocolos que permiten la comunicación entre dos sistemas diferentes que se complementan entre sí, como es el caso de una base de datos y el equipo de un cliente que esté usando nuestra aplicación.

REST por su parte se refiere a que dicha interfaz utiliza para las peticiones HTTP para sus comunicaciones, pudiendo enviar datos en casi cualquier formato, ya sea JSON, HTML, etc. Es independiente de ambos entornos, necesitando un sistema de autentificación independiente, en este caso utilizando Sanctum.

Las diferentes rutas cumplirán con las convenciones de nombres de las API Rest, utilizando los diferentes métodos para las diferentes funciones a realizar. Estas convenciones se pueden comprobar en la documentación aportada en el [siguiente enlace](#). Siguiendo dichas convenciones, la estructura de nombre y métodos de envío será la siguiente:

```
> Route::middleware('auth:sanctum')->group(function() {
    Route::prefix('gestion/v2')->group(function() {
        Route::post('alumno/crear', [App\Http\Controllers\ApiController::class, 'crearAlumno'])->name('gestion.crearAlumno');
        Route::patch('alumno/editar', [App\Http\Controllers\ApiController::class, 'editarAlumno'])->name('gestion.editarAlumno');
        Route::delete('alumno/eliminar', [App\Http\Controllers\ApiController::class, 'eliminarAlumno'])->name('gestion.eliminarAlumno');
        Route::get('alumno/correos', [App\Http\Controllers\ApiController::class, 'obtenerCorreos'])->name('gestion.obtenerCorreos');
    });
});
```

Uno de los elementos más importantes de este proyecto son los llamados DataTables, tablas de datos relacionales que obtienen datos en tiempo real de la base de datos mediante peticiones asíncronas. La lógica de obtención de datos este tipo de elementos suele desarrollarse mediante



una API Rest, pero esto conlleva código separar el código en varios métodos a lo largo del lado cliente y el lado servidor de la aplicación, quedando poco estructurado y muy disperso, además de que la declaración de un elemento DataTable con sus opciones específicas conlleva una gran cantidad de código JavaScript.

Este era el planteamiento inicial hasta que Alejandro descubrió el paquete anteriormente mencionado, que permite centrar toda la lógica de la declaración de los mismos en un fichero, incluyendo las opciones y detalles de las peticiones asíncronas, es decir, esta funcionalidad nos permite delegar la obtención de ciertos datos que suele realizarse mediante peticiones GET al propio paquete.

3.3 Tecnologías y recursos empleados

3.3.1 Visual Studio Code



Visual Studio Code

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



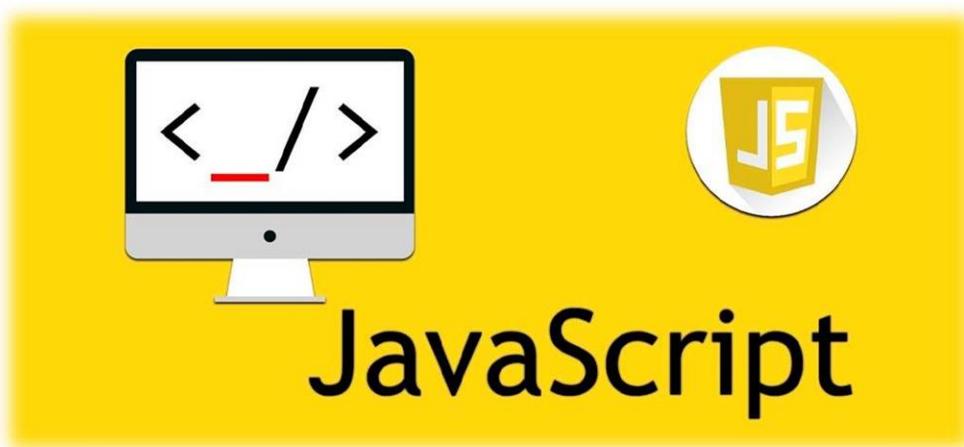
MySQL es un sistema de gestión de base de datos relacionales de código abierto, basado en el modelo cliente-servidor y que permite la creación y manipulación de bases de datos. Es compatible en gran parte de los sistemas operativos, desde macOS a Windows y distribuciones de Linux, y es usado por muchas 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), dividiendo el código dependiendo de su enfoque y funcionalidad. Siendo los modelos enfocados a las conexiones con base de datos, las vistas como el aspecto visual de la web dividido 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

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.

En concreto se ha diseñado un Imagenotipo formado por una imagen y las siglas del proyecto, Sistema de Gestión de Partes del IES San Sebastián (S.G.P.S.S).





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.

Es el método de comunicación principal empleado para la conexión con la API del proyecto y siendo uno de los usos principales de esta tecnología 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 necesario.

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.

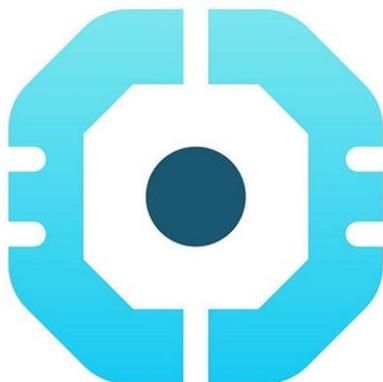
GitHub por su parte es una plataforma web que puede alojar de forma remota repositorios de Git, siendo una de las mayormente empleadas para ello. Para este proyecto se ha creado un repositorio de Git bajo el siguiente enlace, teniendo una rama principal y otra para cada uno de los integrantes del mismo:

<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 Sanctum



Laravel Sanctum

Sanctum es una herramienta de autentificación implementada que puede ser añadida a Laravel de forma rápida y sencilla, orientada a aplicaciones de una única página (SPA o Single Page Application) y por medio de tokens.

Al utilizar una API, no es posible depender de la autentificación del propio usuario en la web, ya que es una arquitectura independiente, por lo que deberán utilizarse los tokens creados por esta herramienta y que quedan asociados al usuario, realizando la identificación de los mismos mediante el correspondiente Middleware.

3.3.11 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.

3.4 Paquetes Composer implementados

3.4.1 Symfony Mailgun Mailer



Paquete implementado para el envío 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 cambiar la estructura de los mismos. Se implementa mediante el siguiente comando:

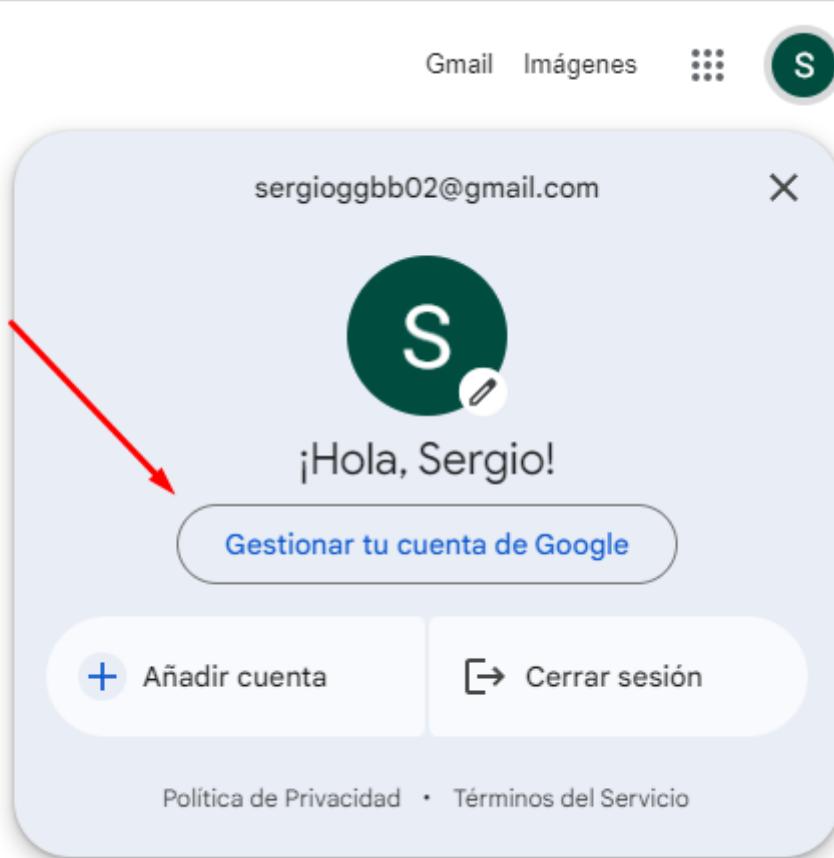
```
composer require symfony/mailgun-mailer symfony/http-client
```

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}"
```



Es importante configurar correctamente la cuenta de Google para permitir el envío desde la misma ya que necesitamos una contraseña de aplicación, el proceso de obtención se detalla a continuación. En primer lugar accedemos a la configuración de la cuenta de Google (Deberá prepararse una cuenta para el centro):



Tras esto accedemos al apartado de Contraseñas de aplicación a través del buscador:

contras

3 RESULTADOS

- Gestor de contraseñas
Seguridad
- Contraseñas de aplicación
Seguridad
- Contraseña
Información personal, Seguridad

Buscar "contras" en el Centro de Ayuda



Nos deberemos autentificar de nuevo:



En este apartado indicamos del nombre de la aplicación en la que usaremos la contraseña y pulsamos crear:

← Contraseñas de aplicación

Las contraseñas de aplicación te ayudan a iniciar sesión en tu cuenta de Google en aplicaciones y servicios antiguos que no son compatibles con los estándares de seguridad modernos.

Las contraseñas de aplicación son menos seguras que usar aplicaciones y servicios actualizados que utilicen estándares de seguridad modernos. Antes de crear una contraseña de aplicación, debes comprobar si tu aplicación la necesita para iniciar sesión.

[Más información](#)

Tus contraseñas de aplicación

Laravel	Creada: 6 abr; utilizada por última vez: 21 abr	
---------	---	--

Para crear una contraseña específica de la aplicación, escribe el nombre de la aplicación a continuación...

Nombre de la aplicación
Demostracion-Proyecto

Crear



Esto nos devolverá la contraseña que deberemos adjuntar en el fichero .env, es importante copiarla inmediatamente ya que no puede volver a ser visualizada:

seguridad modernos.

de
zac
co
par

ña

Contraseña de aplicación generada

Tu contraseña de aplicación para el dispositivo

ipun fxau uutl ftfi

Cómo utilizarla

Accede a la sección de configuración de tu cuenta de Google en la aplicación o el dispositivo que estés intentando configurar. Sustituye tu contraseña por la contraseña de 16 caracteres que se muestra arriba.

Al igual que la contraseña normal, esta contraseña de aplicación ofrece acceso completo a tu cuenta de Google. No tendrás que recordarla, así que no la escribas ni la compartas con nadie.

Hecho

Tras esto lo añadimos al fichero .env, indicando en MAIL_USERNAME la cuenta y en MAIL_PASSWORD la contraseña de aplicación:

```
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}"
```

Otros dos campos muy interesantes de cara a la configuración son:

- MAIL_FROM_ADDRESS: el correo remitente que se muestra, ocultando el real.
- MAIL_FROM_NAME: el nombre del remitente que se muestra a los usuarios.

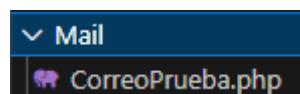


Enfocándonos ya en el propio funcionamiento del paquete, este presenta una amplia variedad de formas de crear y enviar los correos, ya sea usando los métodos de la clase o usando vistas para personalizar el cuerpo de los mismos en formato HTML.

Este es un ejemplo, creamos un correo (de forma similar a un modelo o controlador) con el siguiente comando:

```
php artisan make:mail CorreoPrueba
```

El resultado es el siguiente archivo php:



Las funciones principales a modificar son envelope y content. En envelope indicamos el correo desde el que se envía y el nombre del remitente, siendo posible omitirlo si lo hemos configurado en el fichero .env. Y pudiendo indicar en un array la lista de destinatarios a los que se enviará, así como el Asunto principal en subject:

```
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',
    );
}
```

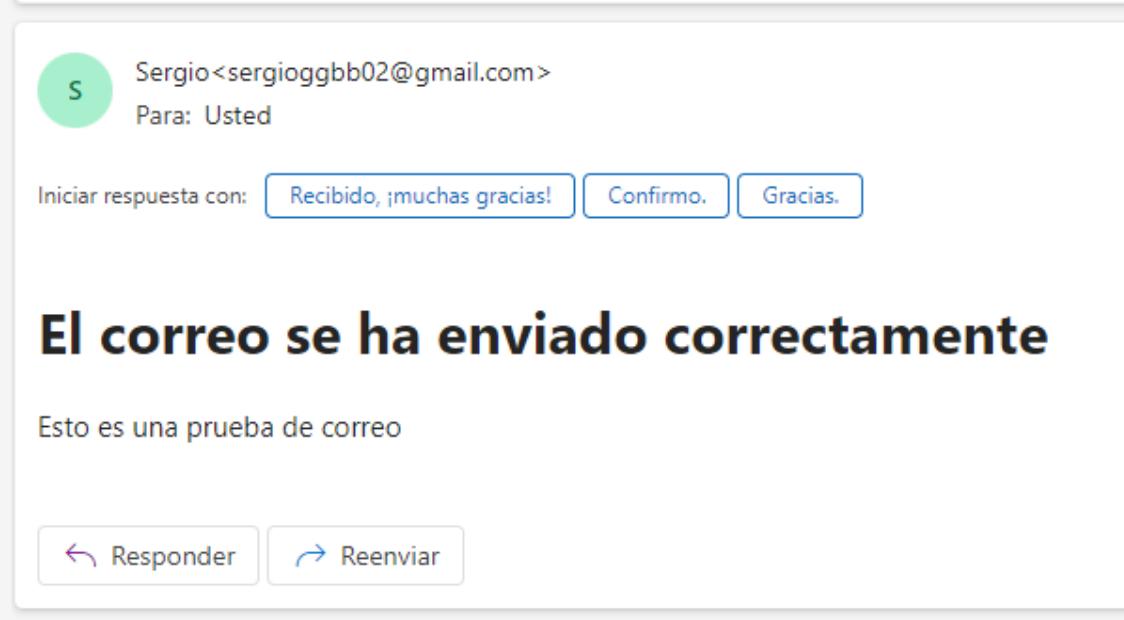
En content podemos indicar su contenido, pudiendo indicar una vista con el formato del mismo:

```
''
public function content(): Content
{
    return new Content(
        view: 'users.correoPrueba',
    );
}
```



```
src > views > users > correoPrueba.blade.php > p  
<h1>El correo se ha enviado correctamente</h1>  
  
<p>Esto es una prueba de correo</p>
```

Al enviar el correo, se recibe con el siguiente aspecto:



Sergio<sergioggbb02@gmail.com>
Para: Usted

Iniciar respuesta con:

El correo se ha enviado correctamente

Esto es una prueba de correo

Hay más variedad de formas de enviar un correo, pudiendo añadir especificaciones desde fuera de la “maqueta” en el método de un controlador:

```
public function pruebaCorreo()  
{  
    Mail::to('alejandrocbt@hotmail.com')->send(new CorreoPrueba());  
    return redirect()->route('user.correo');  
}
```

Si quisiésemos añadir datos para mostrar se deberán añadir en el constructor de la clase, importante indicar que sea público para poder acceder desde la vista:

```
public function __construct(public $nombreAlumno)  
{  
    //  
}
```



Tras esto añadimos el valor de la variable gracias a Blade y pasamos el nombre por parámetros:

```
resources > views > users > correoPrueba.blade.php > p  
1   <h1>El correo se ha enviado correctamente</h1>  
2  
3   <p>Esto es una prueba de correo para {{ $nombreAlumno }}</p>  
  
->send(new CorreoPrueba("Sergio Guerrero"));
```

Y como podemos comprobar, se añade el nombre, lo que nos permitirá personalizar los correos con los datos del Alumno:

Sergio<sergioggbb02@gmail.com>
Para: Usted
Iniciar respuesta con: Recibido, ¡muchas gracias! Confirmo. Gracias.

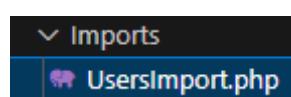
El correo se ha enviado correctamente
Esto es una prueba de correo para **Sergio Guerrero**

3.4.2 Maatwebsite Excel

Paquete para importaciones y exportaciones con Excel. Gracias a este podremos importar la información de los alumnos y otros datos de la Aplicación.

Se pueden crear diferentes clases de importación, que se almacenan en el directorio Imports, asociándolo directamente a un modelo:

```
php artisan make:import UsersImport --model=User
```





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

```
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'],
        ]);
    }
}
```

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 una gran cantidad de usuarios desde un archivo Excel.

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 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:

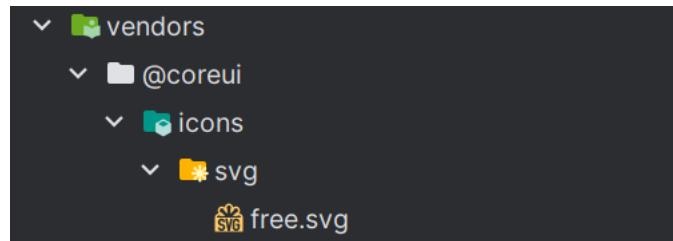
```
public function import(Request $request)
{
    User::truncate();
    $file = $request->file('key: import_file');

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

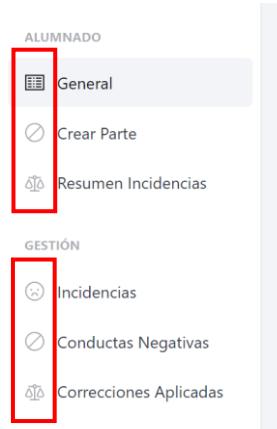
    return redirect()->route('users.index')
        ->with('Satisfactorio', 'Usuario Creado correctamente.');
}
```



3.4.3 SVG



Para el empleo de SVGs en la aplicación se ha usado un paquete de CoreUI que incorpora una amplia gama de diferentes iconos entre los que elegir, siendo usado por ejemplo, en el menú de navegación:



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

La ruta de los mismos se indica con la ruta del archivo seguido de # y eligiendo el nombre del SVG que deseemos.

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



3.4.4 Spatie/laravel-permission

El paquete Spatie es un paquete de autenticación que se ha usado a lo largo del curso. Este paquete añade migraciones que crea modelos de roles y permisos que permiten asociarlos a los modelos, para así restringir ciertas rutas a ciertos roles o permisos. También personalizar ciertos elementos visuales dependiendo del usuario autenticado gracias a su compatibilidad con Laravel Blade.

```
composer require spatie/laravel-permission
```

Una vez descargado el paquete, publicamos sus migraciones para poder añadir los roles y permisos deseados:

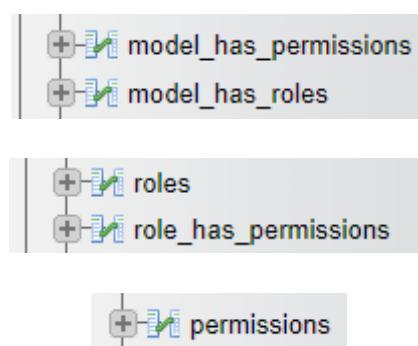
```
PS C:\Users\sergi\OneDrive\Escritorio\Proyecto\ProyectoSergioAlejandro> php artisan vendor:publish --provider="Spatie\Permission\PermissionServiceProvider"
INFO Publishing assets.

Copying file [C:\Users\sergi\OneDrive\Escritorio\Proyecto\ProyectoSergioAlejandro\vendor\spatie\laravel-permission\config\permission.php] to [C:\Users\sergi\OneDrive\Escritorio\Proyecto\ProyectoSergioAlejandro\config\permission.php] DONE
Copying file [C:\Users\sergi\OneDrive\Escritorio\Proyecto\ProyectoSergioAlejandro\vendor\spatie\laravel-permission\database\migrations\create_permission_table.php.stub] to [C:\Users\sergi\OneDrive\Escritorio\Proyecto\ProyectoSergioAlejandro\database\migrations\2024_05_31_193742_create_permission_tables.php] DONE
```

Se nos habrá añadido a las migraciones el siguiente archivo:

```
2020_05_31_193742_create_permission_tables.php
```

Y al refrescar las migraciones, vemos que se nos han asignado las siguientes tablas, en el caso de este proyecto, únicamente será utilizada model_has_roles y roles, ya que tendremos un usuario general para los profesores, y otro específico para jefatura con opciones de gestión:



La implantación de los roles será tratada en el apartado correspondiente en el punto 4.



3.5 Elementos Bootstrap y CKEditor

3.5.1 Switch

En las vistas planteadas, únicamente profesor/alumno 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">
```

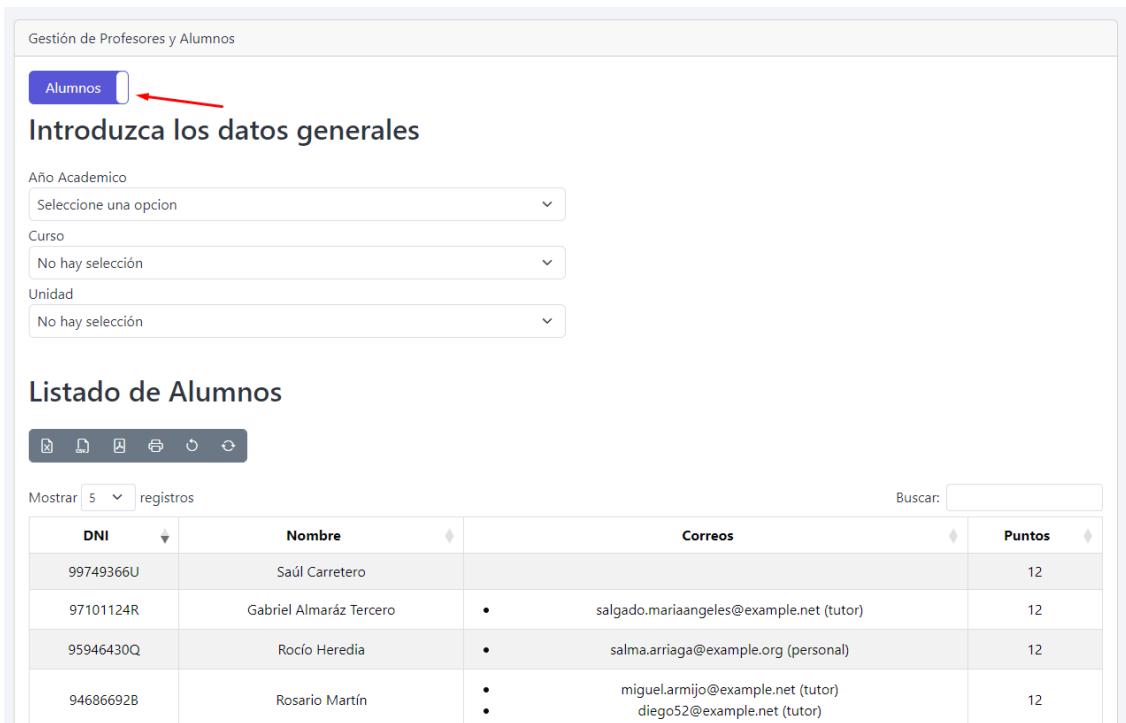
Al cambiar el valor del Switch, cambian el div que se muestra y aquel que se oculta. Alternándose entre el div centrado en los profesores y sus datos y aquel centrado en alumnos y sus datos:

```
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:



The screenshot shows a web application interface for managing teachers and students. At the top, there's a header bar with the title "Gestión de Profesores y Alumnos". Below the header, a blue button labeled "Alumnos" is highlighted with a red arrow pointing to it. The main content area has a heading "Introduzca los datos generales" followed by three dropdown menus: "Año Académico" (with placeholder "Seleccione una opción"), "Curso" (with placeholder "No hay selección"), and "Unidad" (with placeholder "No hay selección"). Below this, another heading "Listado de Alumnos" is shown, followed by a table with the following data:

DNI	Nombre	Correos	Puntos
99749366U	Saúl Carretero		12
97101124R	Gabriel Almaráz Tercero	• salgado.maríaangeles@example.net (tutor)	12
95946430Q	Rocío Heredia	• salma.arriaga@example.org (personal)	12
94686692B	Rosario Martín	• miguel.armijo@example.net (tutor) diego52@example.net (tutor)	12

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



Gestión de Profesores y Alumnos

Profesores

Añadir nuevo profesor/a

Campos del nuevo profesor/a:

DNI	Nombre	Teléfono	Correo	Añadir	Limpiar
-----	--------	----------	--------	--------	---------

Listado de Profesores/as

DNI	Nombre	Teléfono	Correo	Opciones
06612132u	Sra. Laia Mora Segundo	881-04-47-44	arequena@hispavi	<button>Editar</button> <button>Eliminar</button>
07348363x	Carmen Viera	163-73-65-92	marc10@hotmail.e	<button>Editar</button> <button>Eliminar</button>
14757035p	Adrián Marcos Segundo	674-09-84-60	hrolon@barraza.es	<button>Editar</button> <button>Eliminar</button>
15116765f	Ing. Biel Leyva Hijo	177-43-15-81	sdelgado@dominc	<button>Editar</button> <button>Eliminar</button>
16776726n	César Jurado	816-08-79-73	sara.lujan@hotmai	<button>Editar</button> <button>Eliminar</button>

Mostrando de 1 hasta 5 de 21 resultados

3.5.2 CKEditor

El elemento CKEditor es un textarea modificado mediante Javascript para permitir una mayor complejidad en su contenido y que a su vez permite añadir otras funcionalidades como poder cambiar el aspecto del texto en tiempo real, simulando un editor de texto como Word. El principal uso de este elemento es añadir imágenes a dichos campos de texto y poder enviarlos junto al formulario asociado.

En el caso de este proyecto será añadido como un campo del formulario de crear parte, en el que los profesores podrán detallar todo lo ocurrido y adjuntar imágenes si fuese necesario:

```
<div class="col-12 mb-1">
    <label for="inputDescripcionDetallada" class="d-block">Descripción Detallada</label>
    <div class="col-6">
        <div class="form-group">
            <textarea name="inputDescripcionDetallada" class="form-control" id="inputDescripcionDetallada"></textarea>
        </div>
    </div>
</div>
```



Para crear este elemento se establece el siguiente código JavaScript, indicando el lenguaje y que se va a añadir el plugin para adjuntar ficheros:

```
ClassicEditor  
    .create(document.querySelector('#inputDescripcionDetallada'), {  
        language: 'es',  
        extraPlugins: [MyCustomUploadAdapterPlugin],  
    })  
    .catch(error => {  
        console.error(error);  
    });  
  
function MyCustomUploadAdapterPlugin(editor) {  
    editor.plugins.get('FileRepository').createUploadAdapter = (loader) => {  
        return new MyUploadAdapter(loader);  
    };  
}
```

Y con la siguiente clase se permite adjuntar a los datos del formulario los ficheros adjuntados, con una llamada asíncrona independiente:

```
class MyUploadAdapter {  
    constructor(loader) {  
        this.loader = loader;  
    }  
  
    upload() {  
        return this.loader.file  
            .then(file => new Promise((resolve, reject) => {  
                const data = new FormData();  
                data.append('upload', file);  
  
                axios.post('/upload', data)  
                    .then(response => {  
                        resolve({default: response.data.url});  
                    })  
                    .catch(error => {  
                        reject('Upload failed');  
                        console.error(error);  
                    });  
            })  
    }  
  
    abort() {  
    }  
}
```



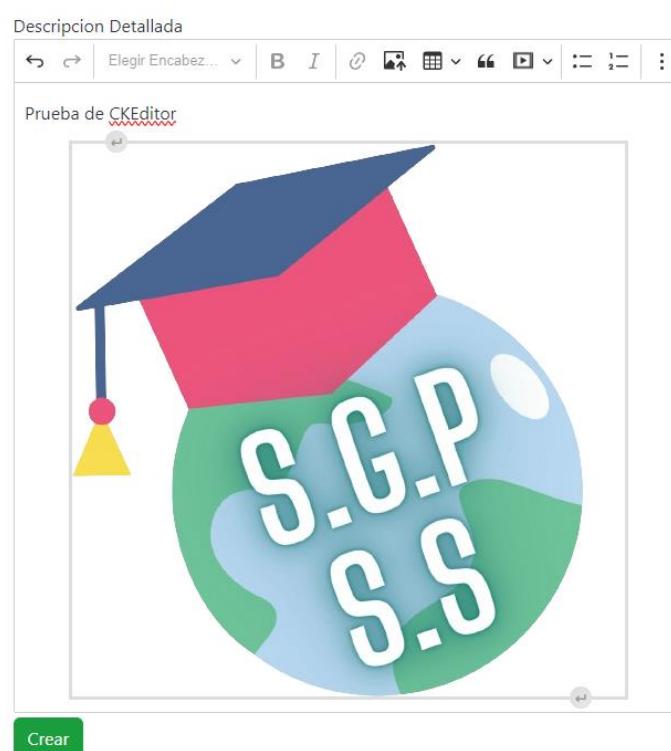
```
public function upload(Request $request)
{
    $file = $request->file('upload');
    $fileName = uniqid() . '_' . trim($file->getClientOriginalName());

    $file->move(public_path('uploads'), $fileName);

    return response()->json([
        'uploaded' => 1,
        'fileName' => $fileName,
        'url' => '/uploads/' . $fileName
    ]);
}
```

La descripción por su parte se añade insertándose en base de datos, incluyendo la referencia a los archivos de imagen, como se puede observar en el siguiente ejemplo de creación de parte:

```
$parte = Parte::create([
    'profesor_dni' => request('inputProfesor'),
    'tramo_horario_id' => request('inputTramoHorario'),
    'alumno_dni' => request('inputAlumno'),
    'colectivo' => 'No',
    'fecha' => $fecha,
    'descripcion_detallada' => request('inputDescripcionDetallada'),
]);
```





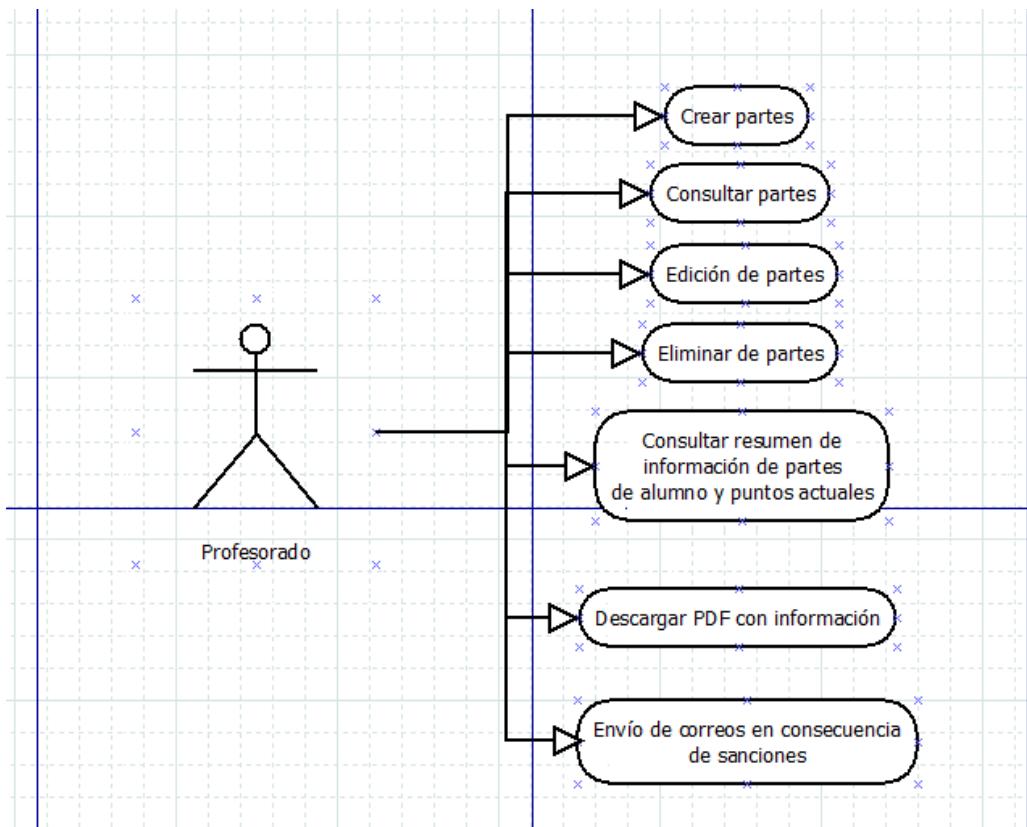
descripcion_detallada

<p>Prueba de CKEditor</p><figure class="image"></figure>

3.6 Casos de uso

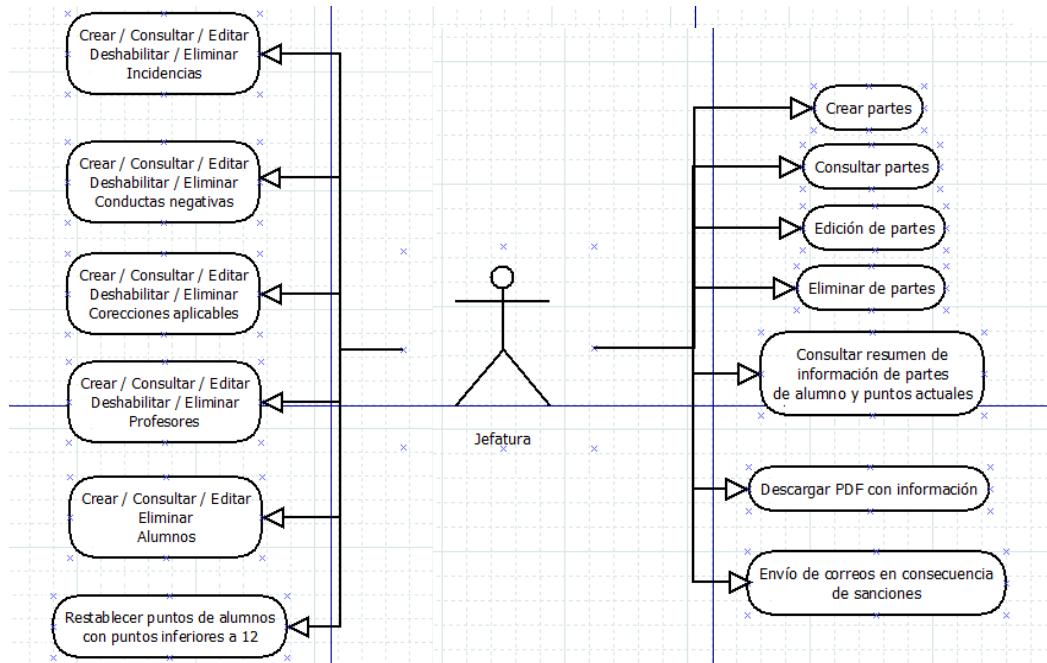
En este apartado se expondrán las diferentes funciones que cada rol podrá realizar, como ya se ha mencionado, al ser una web de datos educativos, únicamente podrán acceder el profesorado y jefatura, tal y como nos lo ha trasladado nuestro tutor de proyecto. Para hecho se utilizará la herramienta Dia, de creación de Diagramas.

- Rol de profesorado, solo acceso a gestión de partes:





- Rol de jefatura, acceso a gestiones totales del alumno, profesorado y campos del parte:



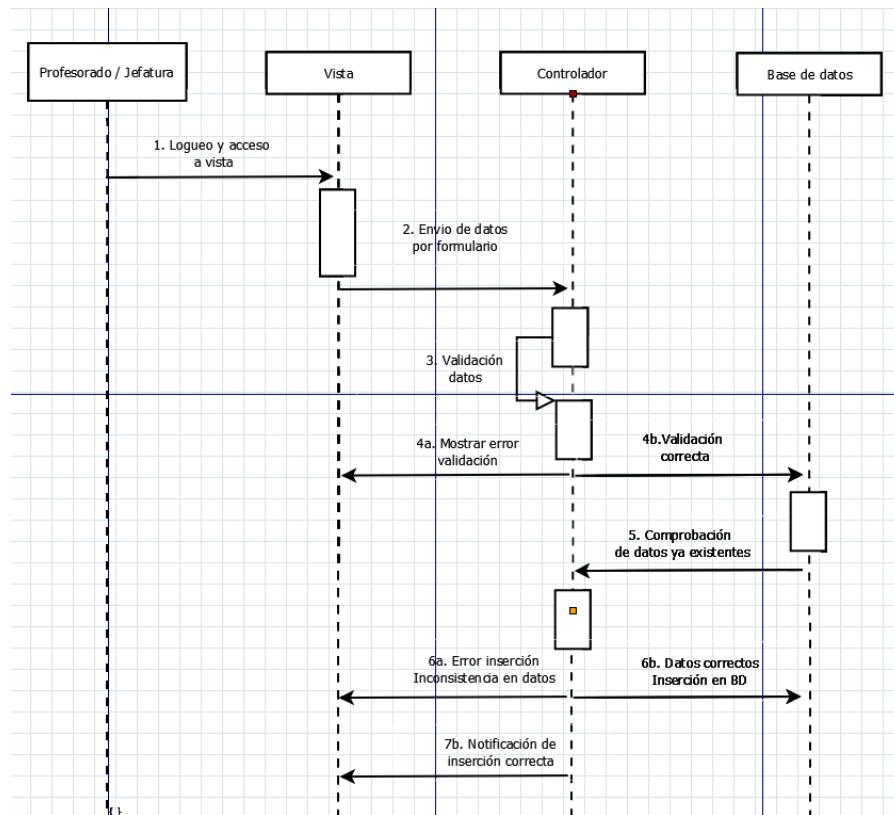
3.7 Diagramas de secuencia

En este apartado se añadirán los diagramas de secuencia relacionados a los casos de uso anteriores, agrupándolos según el procedimiento si siguen una lógica similar.

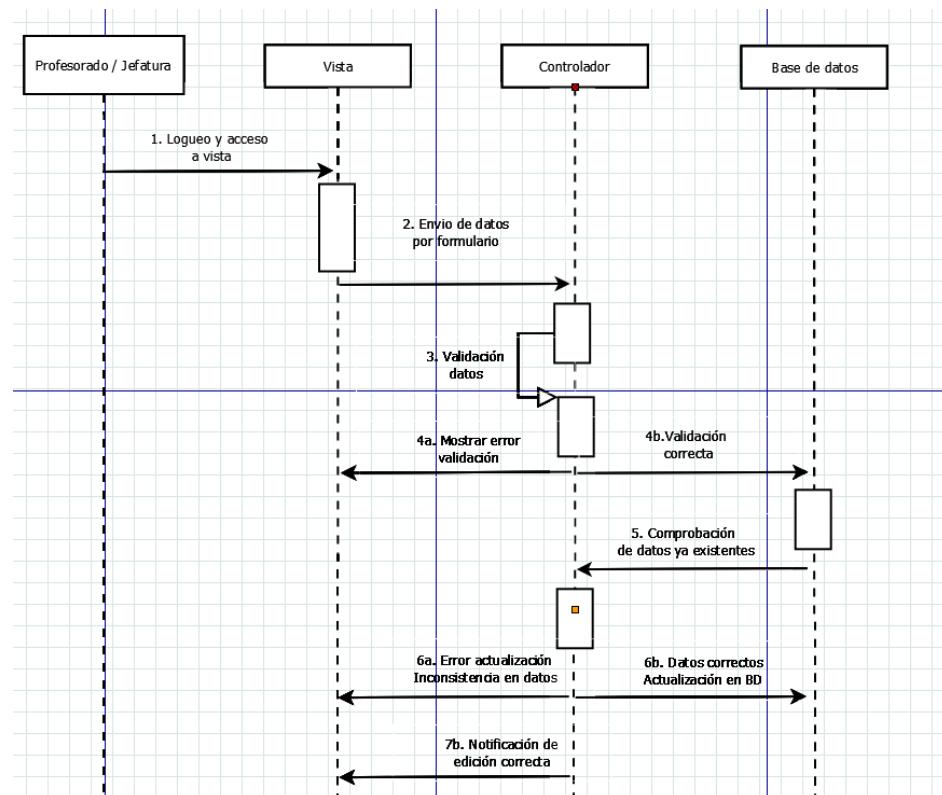


3.7.1 Rol profesorado (Acceso general con logueo)

- Crear partes:

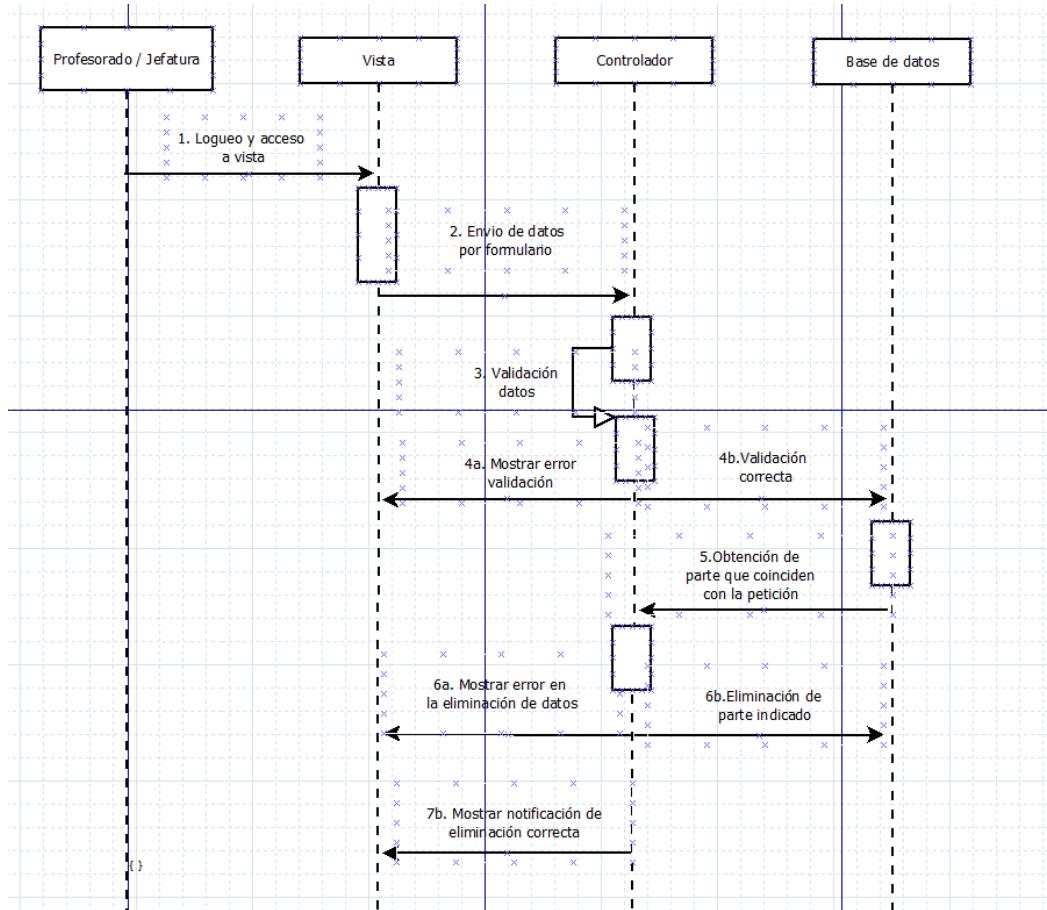


- Editar partes:

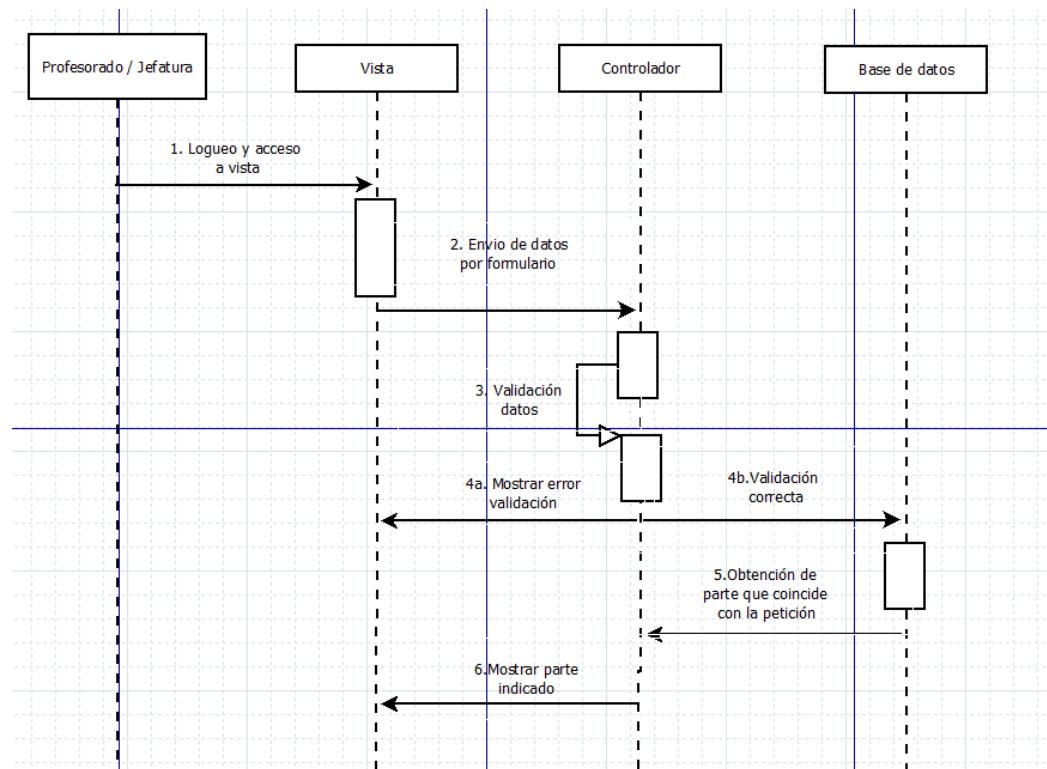


SISTEMA DE GESTIÓN DE PARTES

- Eliminar partes:

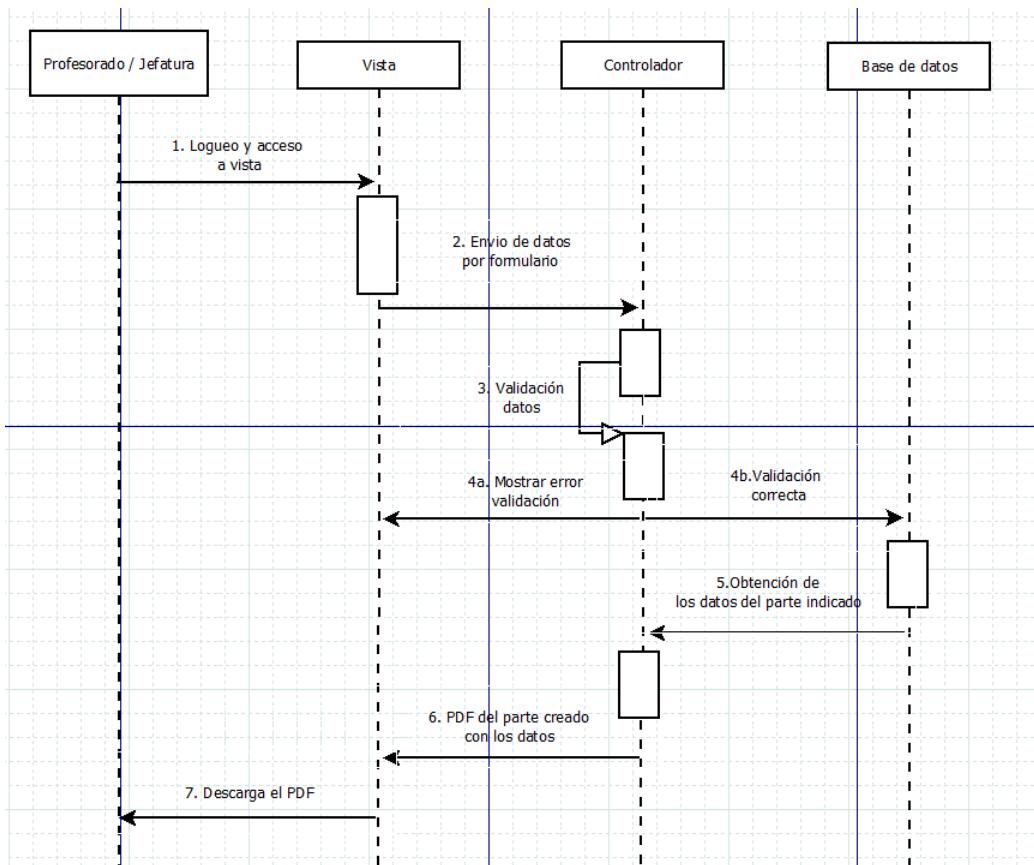


- Consulta de datos de partes:

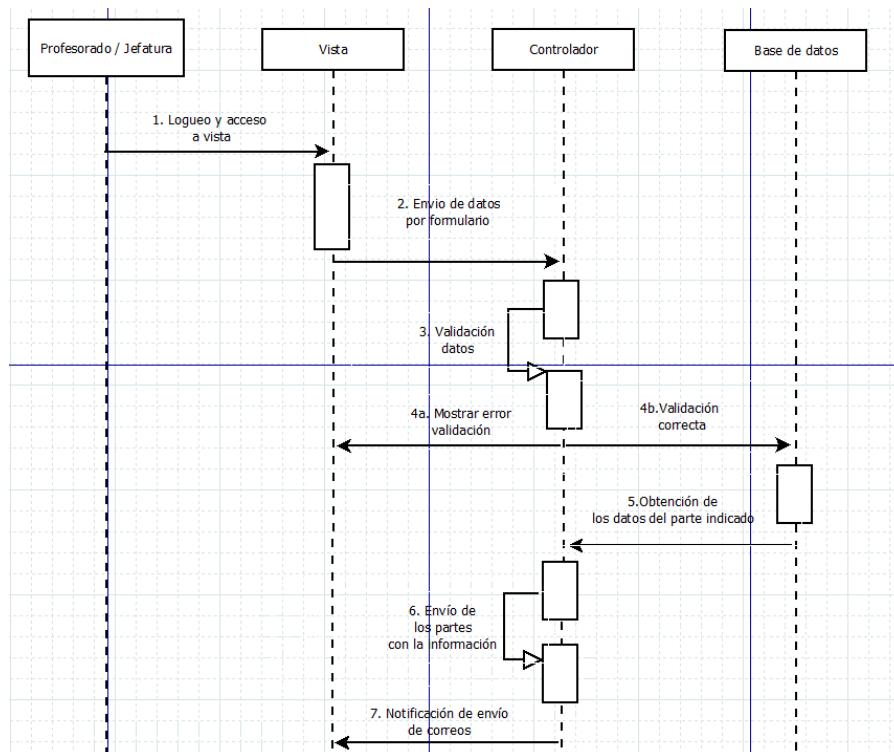


SISTEMA DE GESTIÓN DE PARTES

- Descargar PDF con información del parte:



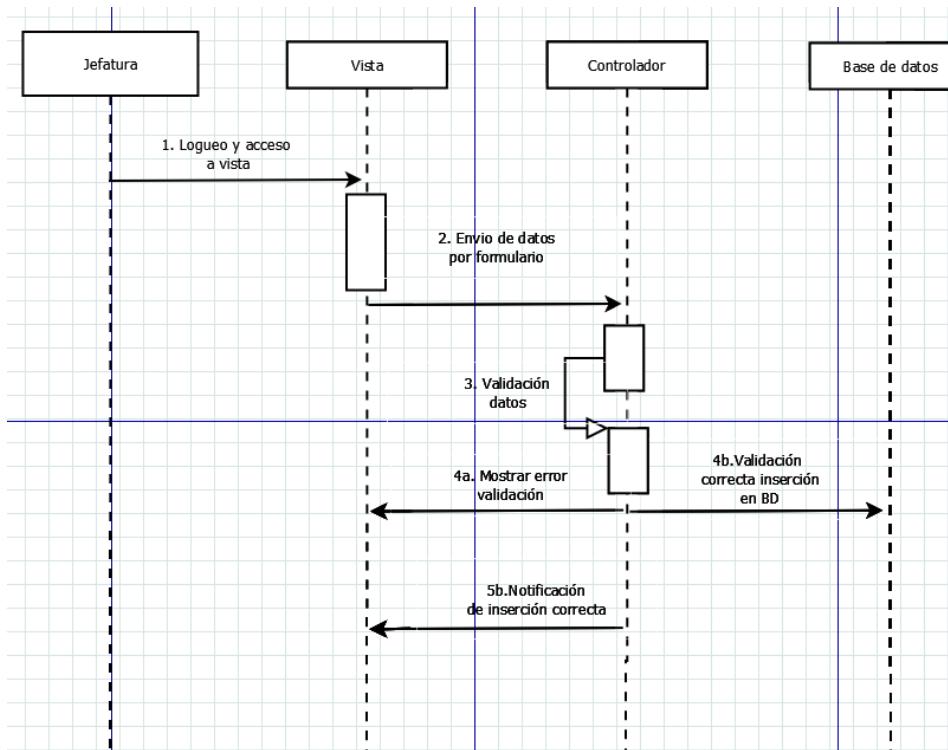
- Envío de correos:



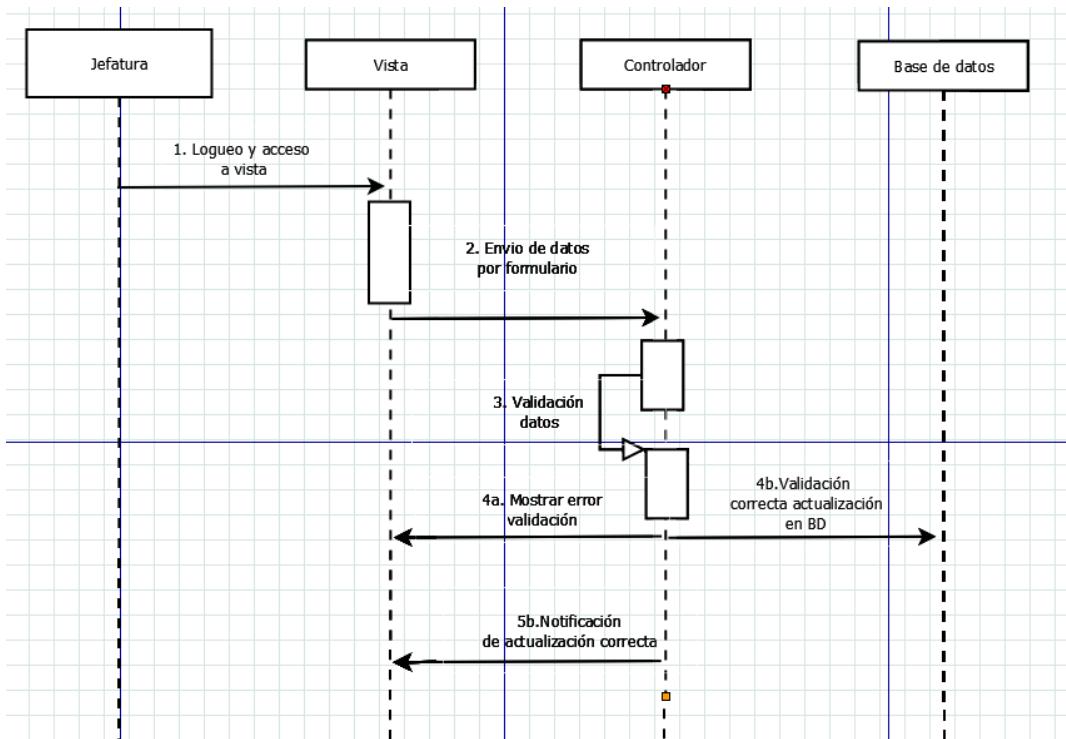
SISTEMA DE GESTIÓN DE PARTES

3.7.2 Rol jefatura

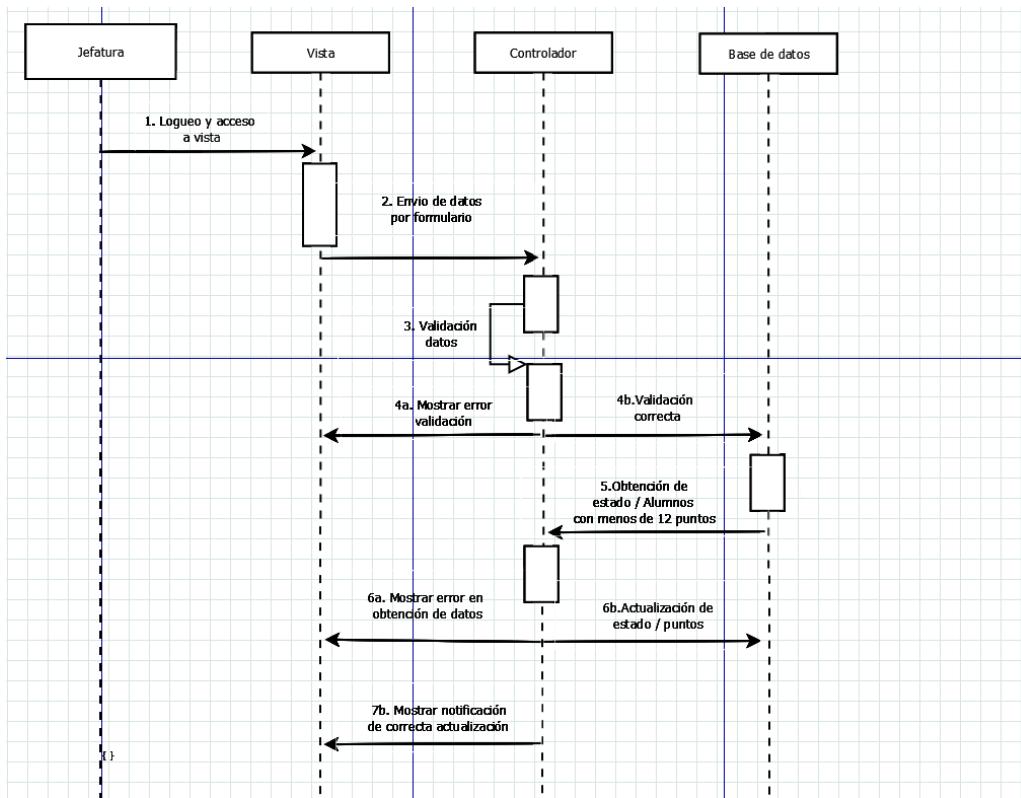
- Crear incidencias, conductas negativas, correcciones aplicables:



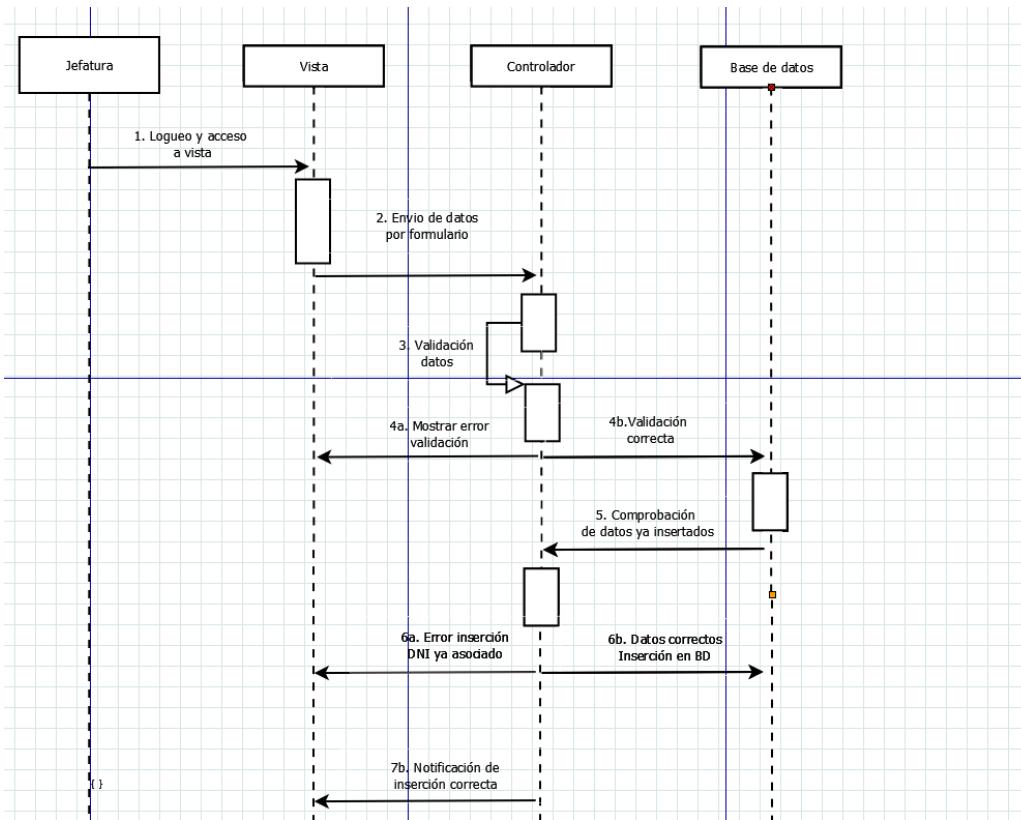
- Editar incidencias, conductas negativas, correcciones aplicables:



- Deshabilitar/Habilitar incidencias, conductas negativas, correcciones aplicables y profesores y cambio de puntos de alumnos a 12:

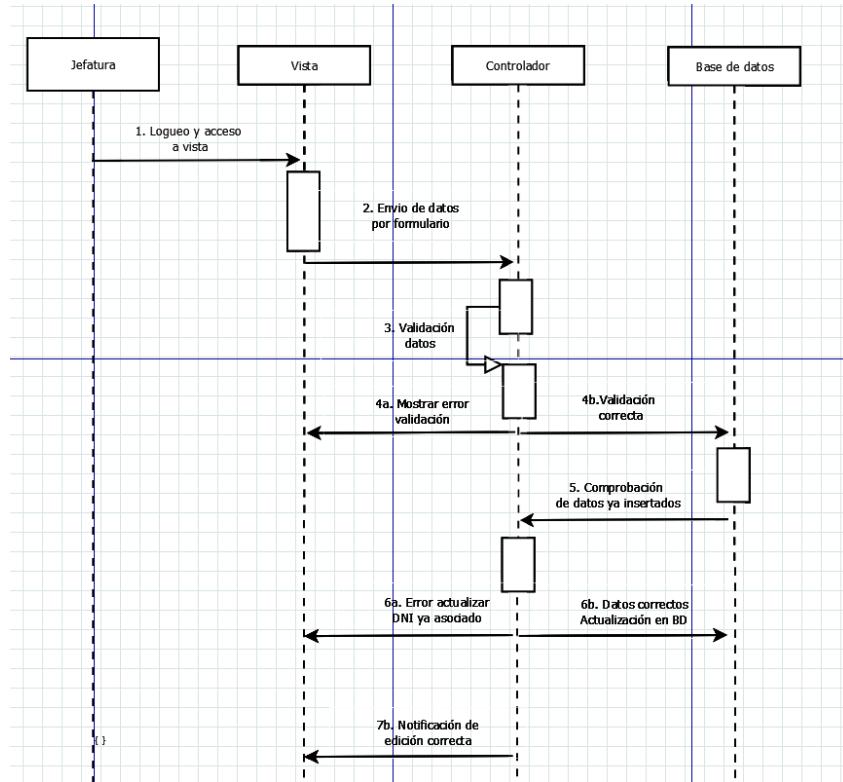


- Crear profesores y alumnos:

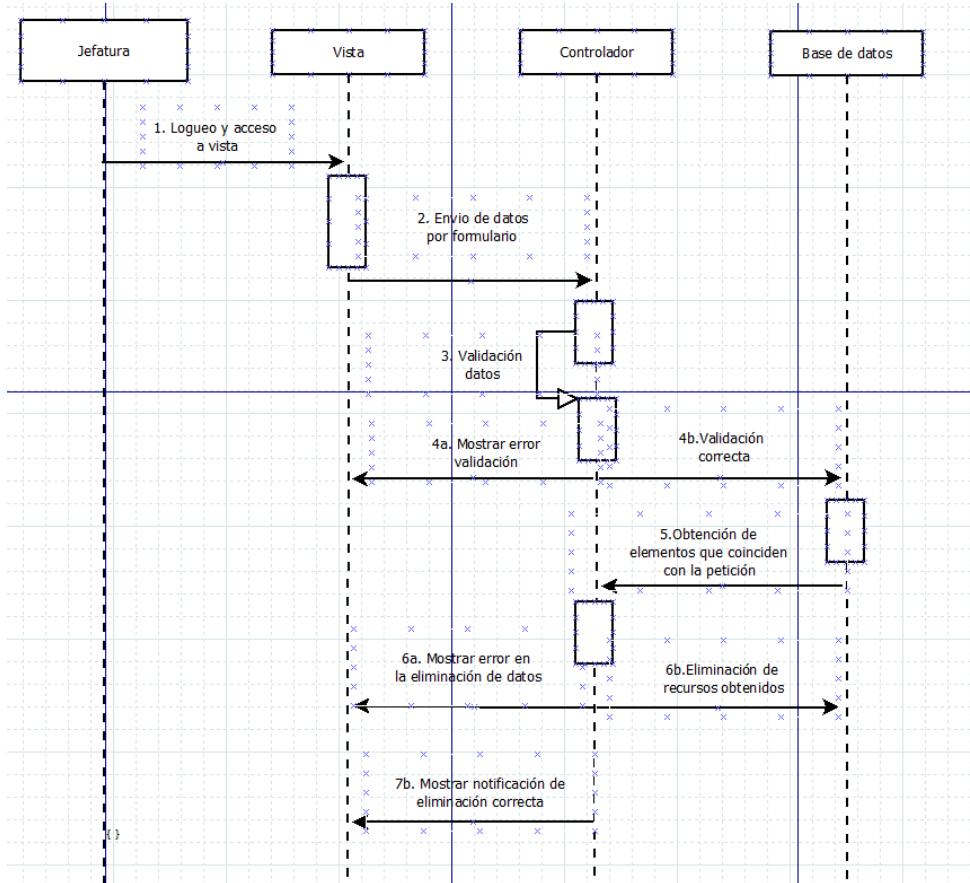


SISTEMA DE GESTIÓN DE PARTES

- Editar profesores y alumnos:

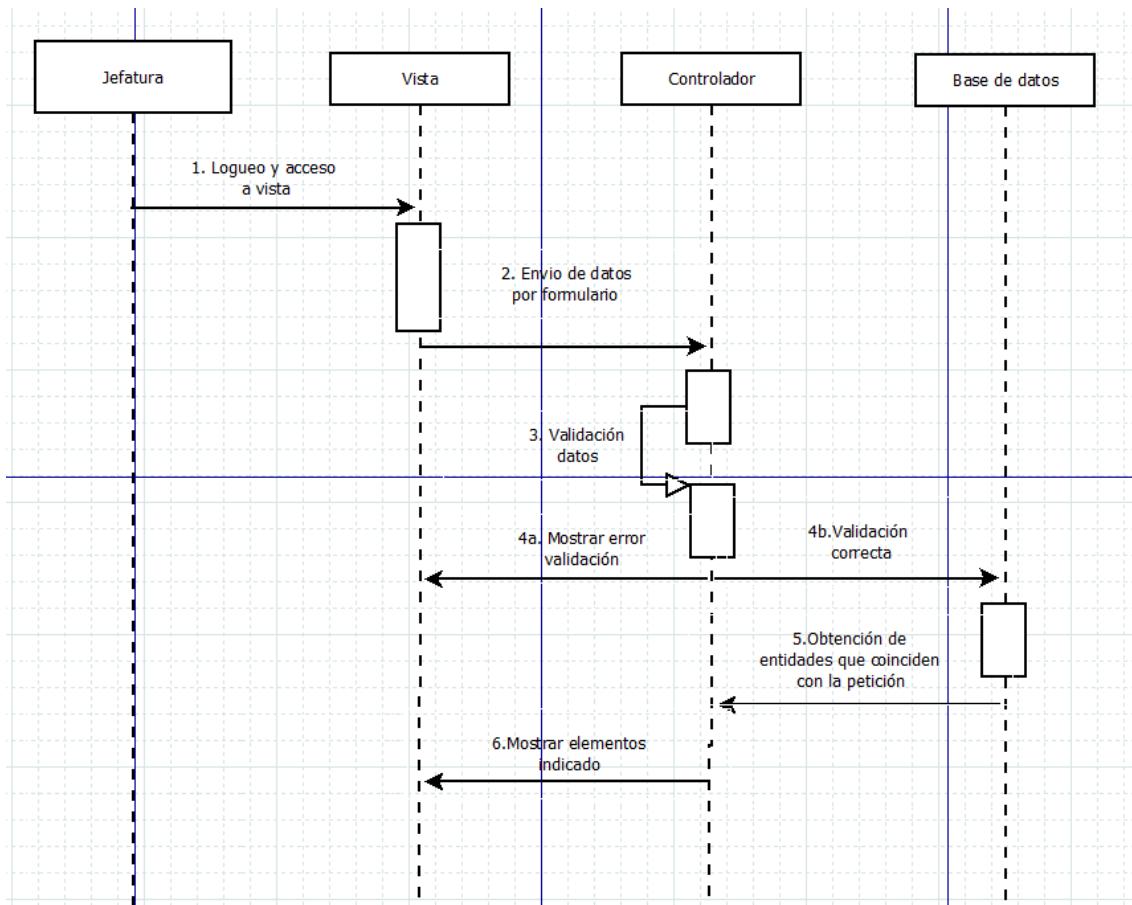


- Eliminar todos los campos:



SISTEMA DE GESTIÓN DE PARTES

- Consulta de todos los datos:



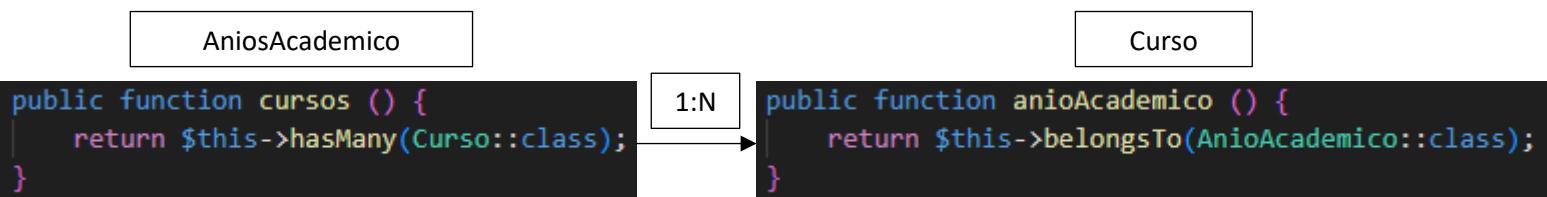


4. Planteamiento e implementación

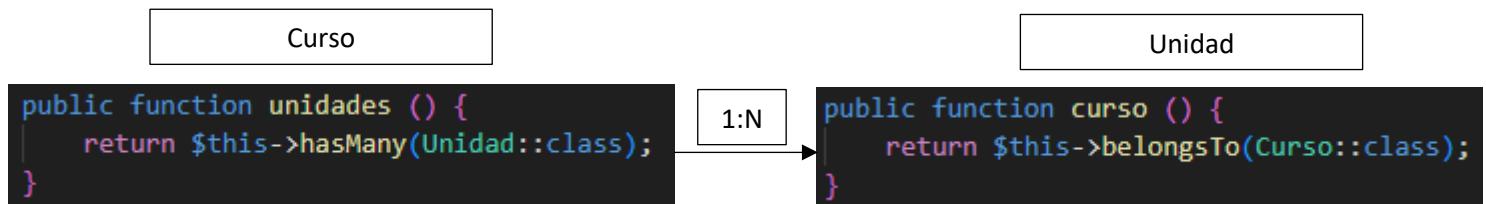
4.1 Base de datos y relaciones

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.

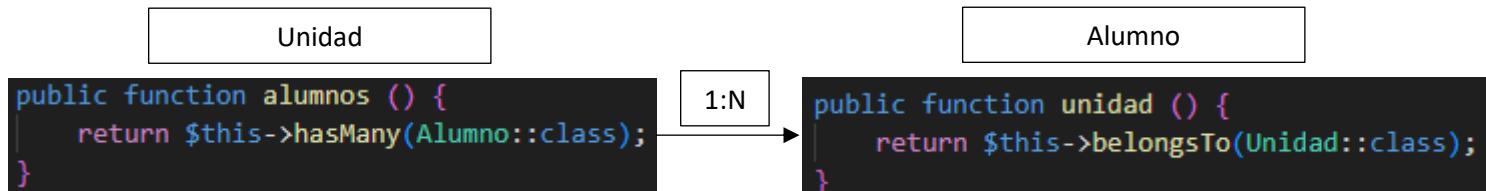
- Años académico 1:N Cursos:



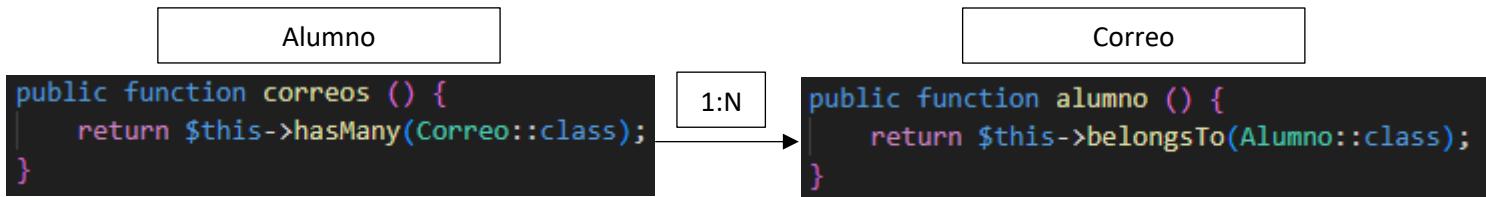
- Curso 1:N Unidades:



- Unidad 1:N Alumnos:

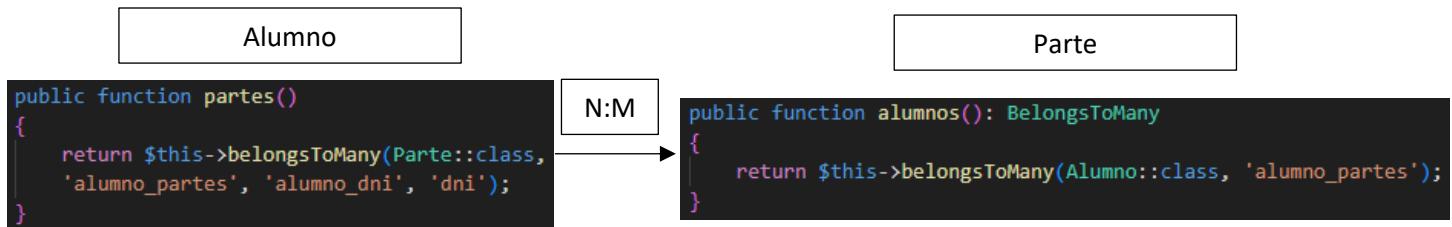


- Alumno 1:N Correos:

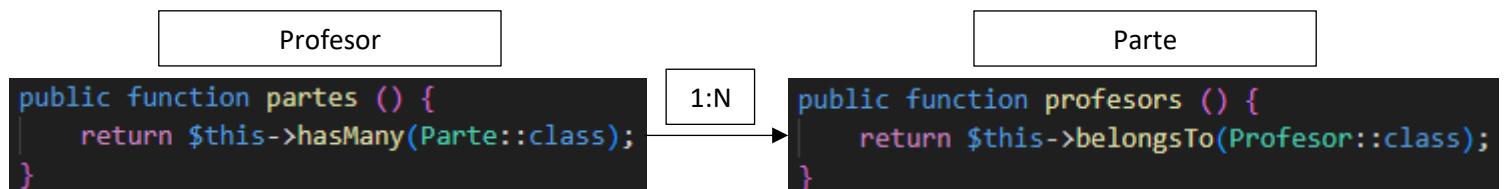




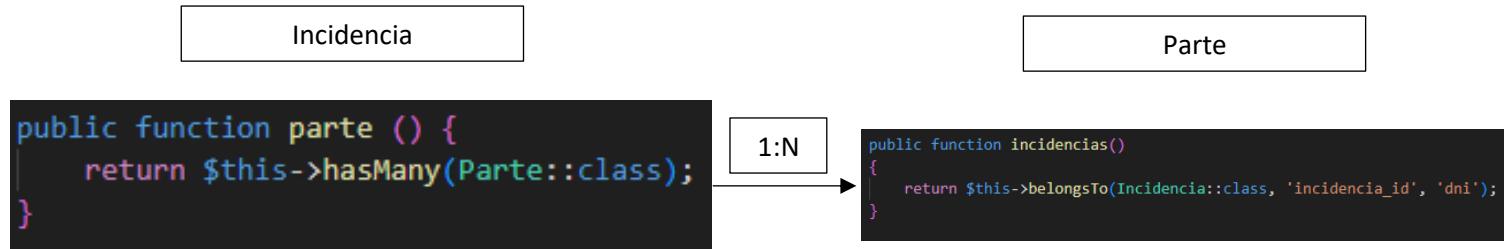
- Alumnos N:M Partes (Tabla intermedia alumno_partes):



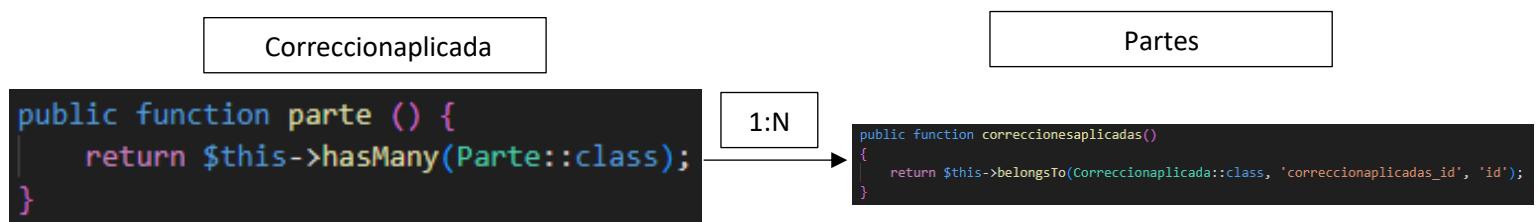
- Profesor 1:N Partes:



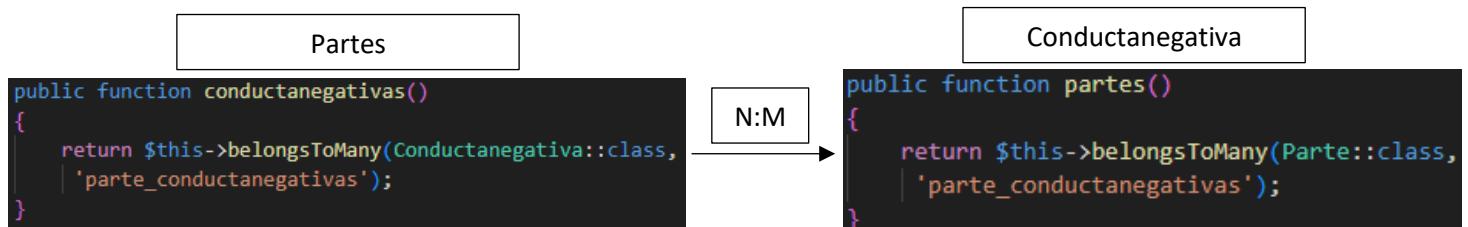
- Incidencia 1:N Partes;



- Correcciones aplicadas 1:N Partes:

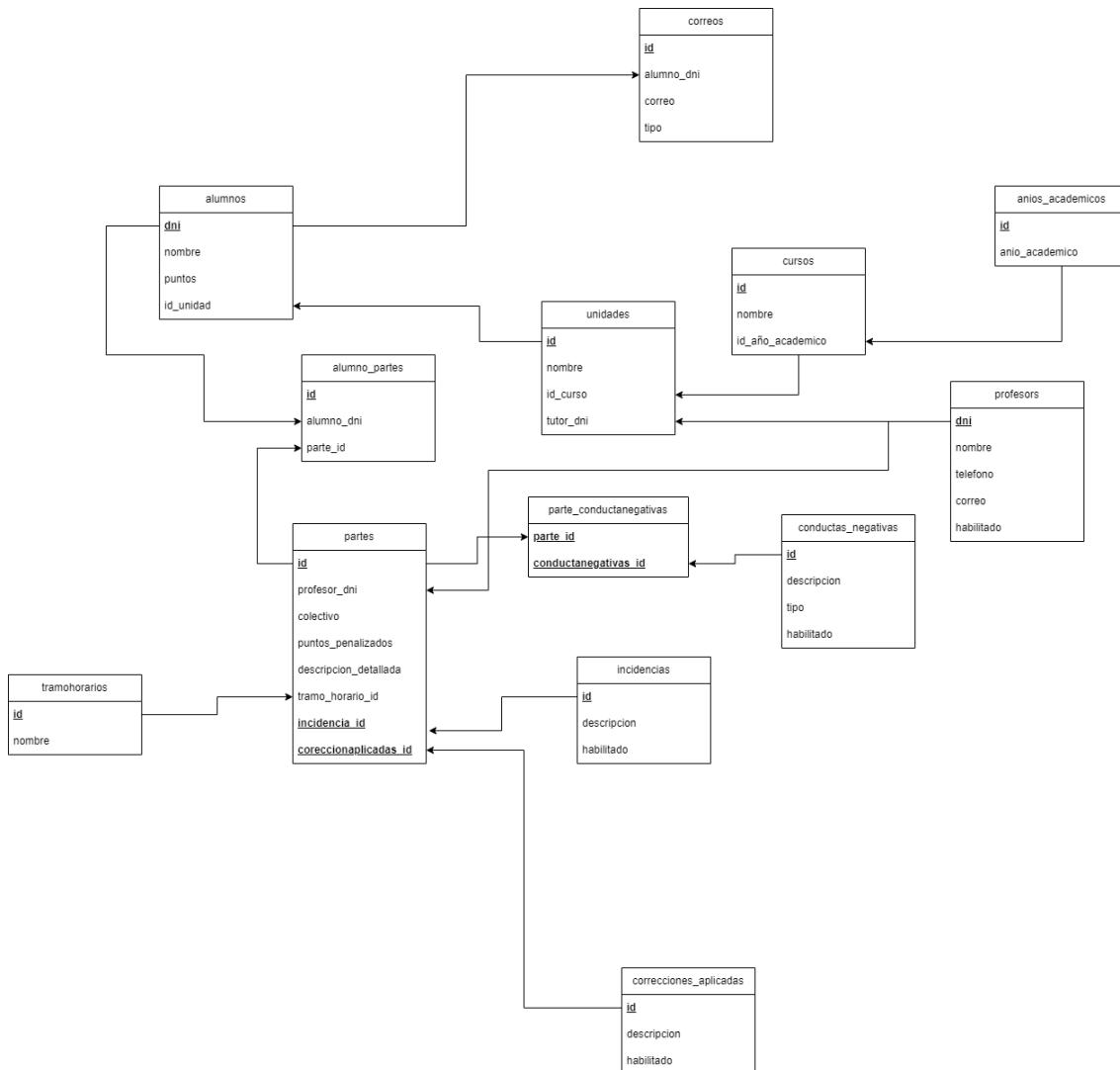


- Partes N:M Conductas Negativas (Tabla intermedia parte_conductanegativas):



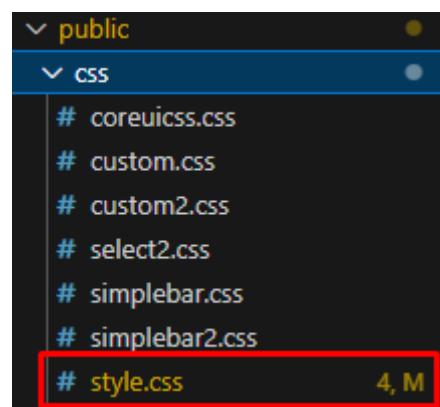


Quedando la estructura general de la base de datos de la siguiente forma:



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:

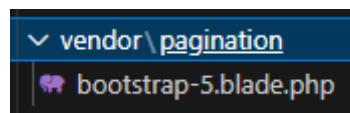
```
15519 }  
15520  
15521 [data-coreui-theme=dark] body {  
15522 | background-color: var(--cui-dark-bg-subtle)  
15523 }  
15524  
15525 [data-coreui-theme=dark] .list-group-item {  
15526 | --cui-footer-bg: var(--cui-body-bg)  
15527 }  
15528  
15529 #list-group-item {  
15530 | font-size: small;  
15531 | color: wheat;  
15532 }  
15533  
15534 .list-group-item {  
15535 | align-items: center;  
15536 | margin-bottom: 2px;  
15537 | margin-top: 2px;  
15538 }  
15539  
15540 #list-group-item-action {  
15541 | cursor: pointer;  
15542 }
```

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>
@endif
<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>
@endif

```

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:

```
if ($paginator->hasMorePages())
    foreach ($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 }}</span>
        @else
            <li class="page-item"><a class="page-link text-success" href="{{ $url }}>{{ $page }}</a></li>
        @endif
    endforeach
endif
foreach

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

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

#	Descripción	Opciones
1	Jugar en clase	<button>Editar</button> <button>Eliminar</button>
2	Pelea con un compañero	<button>Editar</button> <button>Eliminar</button>
3	Malos modos	<button>Editar</button> <button>Eliminar</button>
4	Jugar en clase	<button>Editar</button> <button>Eliminar</button>
5	Uso del móvil sin permiso	<button>Editar</button> <button>Eliminar</button>

Mostrando de 1 hasta 5 de 6 resultados

< 1 2 >



4.4 Datatables

Los elementos 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.

Si no se hubiese implementado el paquete yajra/laravel-datatables, la lógica de obtención de datos de consultas de base de datos debería realizarse a través de peticiones asíncronas con AJAX y la implementación de métodos GET con API. Gracias a este paquete puede enfocarse toda esa lógica en un único archivo .php.

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 un DataTable, en primer lugar deberá pasarse a través del controlador por parámetros a la vista, pero realmente se instancia por si sola en ese momento. Para indicar que debe crearse este elemento, llamamos a la variable con su método render:

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

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>
<div>
|v id="divProfesor">
```

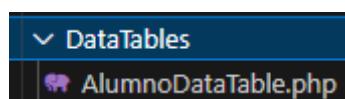


Se utilizará el DataTable de alumnos para la explicación general, mostrando tras esto el resto de DataTables usados en la web.

- Alumnos / Profesores:

La vista de profesores/alumnos 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:



Las clases de un DataTable constan 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:

```
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:

```
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:

```
$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:

```
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:

```
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:

```
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}%");
    });
});
```

Nombre	Correos	Puntos
Ing. Juana Valles	• saulcabrera@example.com (tutor)	12
Saúl Solano Hijo	• ayala.gael@example.com (personal)	12
Saúl Carretero		12

- 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-datable'),
        Column::make('puntos')->title('Puntos')->data('puntos')->data('puntos')->className('align-middle text-center')->searchable(false),
    ];
}
```



El resto de Datatables funcionan de forma similar, pero con los modelos correspondientes.

- En Partes:

Fecha	Nombre	¿Colectivo?	Descripción	Conducta Negativa	Puntos Penalizados	Acciones
09/06/2024 20:26	Dña Ainara Guajardo Hijo	Si		<ul style="list-style-type: none">Agresión física a un miembro de la comunidad educativa.	1	
09/06/2024 20:26	Dña Ainara Guajardo Hijo	No	Jugar en clase	<ul style="list-style-type: none">Impedir o dificultar el estudio a sus compañeros.	1	
				<ul style="list-style-type: none">Actuaciones incorrectas hacia algún miembro de la comunidad educativa.Falta de colaboración sistemática en la realización		

Partes muestra un conjunto de información de los campos principales de la sanción asociada al alumno, pudiendo gracias a esto obtenerse además sus datos para realizar operaciones de edición, como veremos posteriormente.

En cuanto a su consulta principal, al requerir datos de alumno, partes, incidencias y conductas, su complejidad es relativamente alta, especialmente en cuanto a la cantidad de Joins, además de hacer una consulta que concatena las diferentes conductas realizadas, de forma similar a los correos en alumnos:

```
$query = $model->newQuery()
->leftJoin('alumno_partes', 'partes.id', '=', 'alumno_partes.parte_id')
->leftJoin('alumnos', 'alumno_partes.alumno_dni', '=', 'alumnos.dni')
->leftJoin('incidencias', 'partes.incidencia_id', '=', 'incidencias.id')
->leftJoin('parte_conductanegativas', 'partes.id', '=', 'parte_conductanegativas.parte_id')
->leftJoin('conductanegativas', 'parte_conductanegativas.conductanegativas_id', '=', 'conductanegativas.id')
->select(
    'partes.*',
    'alumnos.*',
    'incidencias.descripcion',
    DB::raw(' CONCAT("<ul><li>', GROUP_CONCAT(DISTINCT conductanegativas.descripcion SEPARATOR "</li><li>"), "</li></ul>" ) as descripcion_conducta_negativa')
)
->groupBy('partes.id');

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

return $query;
```

**- Resumen Incidencias:**

Incidencias Alumnos

Año Académico
2023-2024

Curso
3º de ESO

Unidad
3º de ESO B

Mostrar 10 registros Buscar:

Nombre	Incidentes	Conductas graves	Conductas contrarias	Puntos restantes
Alejandro Cadena	1	0	3	8
Dr. Óscar Betancourt	2	2	4	4

Mostrando registros del 1 al 2 de un total de 2 registros

Anterior 1 Siguiente

La complejidad de su consulta recae de nuevo en los Joins, y en evitar la repetición de filas con DISTINCT:

```
$query = $model->newQuery()
    ->leftJoin('alumno_partes', 'alumnos.dni', '=', 'alumno_partes.alumno_dni')
    ->leftJoin('partes', 'alumno_partes.parte_id', '=', 'partes.id')
    ->leftJoin('incidencias', 'partes.incidencia_id', '=', 'incidencias.id')
    ->leftJoin('parte_conductanegativas', 'partes.id', '=', 'parte_conductanegativas.parte_id')
    ->leftJoin('conductanegativas', 'parte_conductanegativas.conductanegativas_id', '=', 'conductanegativas.id')
    ->select(
        'alumnos.*',
        DB::raw("COUNT(DISTINCT partes.incidencia_id) as count_incidencia"),
        DB::raw("COUNT(DISTINCT parte_conductanegativas.conductanegativas_id) as count_conducta_negativa"),
        DB::raw("COUNT(CASE WHEN conductanegativas.tipo = 'grave' THEN parte_conductanegativas.conductanegativas_id END) as count_conducta_grave"),
        DB::raw("COUNT( CASE WHEN conductanegativas.tipo = 'contraria' THEN parte_conductanegativas.conductanegativas_id END) as count_conducta_negativa_contraria"),
    )
    ->groupBy('alumnos.dni');

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

return $query;
```



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

```
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:

```
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:

```
->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'),
]);

```

Un problema a solucionar ha sido que Datatable ordena por defecto todas las columnas como String, por lo que la ordenación de los números era incorrecta:

10
12
2

Para que tome la columna como numérica, hay que indicar en la definición de las mismas su tipo, solución hallada gracias al [foro del paquete](#):

```
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 list-group-item'),
        Column::make('puntos')->title('Puntos')->data('puntos')->type('num')
            ->className('align-middle text-center')->searchable(false),
    ];
}
```



4.5 Preparación de roles e implementación de Middleware

Continuamos con el planteamiento de los roles del apartado de análisis. En primer lugar prepararemos la tabla users para asociarla a los roles.

Añadimos al modelo del usuario el trait de HasRoles, para indicar poder usar las asignaciones de roles de forma rápida y efectiva:

```
use Laravel\Sanctum\HasApiTokens;
use Spatie\Permission\Traits\HasRoles;

class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable, HasRoles;

    /**
     * ...
     */
```

Añadimos la creación de los roles al seeder:

```
use Spatie\Permission\Models\Role;

class DatabaseSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        Role::create(['name' => 'jefatura']);
        Role::create(['name' => 'profesor']);
```

Y los asignamos a los usuarios creador por defecto, correspondientes a jefatura y profesor:

```
$jefatura->assignRole('jefatura');
$profesor->assignRole('profesor');
```

Tras lanzar el seeder podemos comprobar que se han asignado los roles a cada usuario en las tablas:

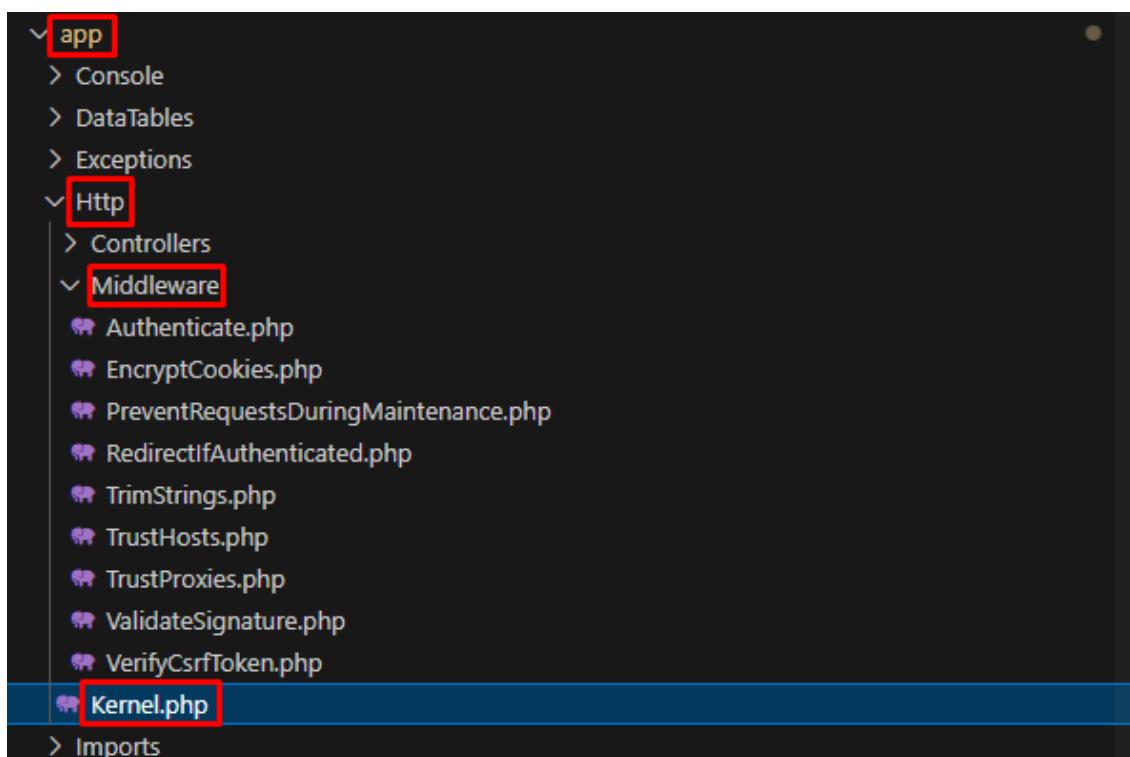
```
1 SELECT U.name AS 'Nombre Usuario', MR.model_type AS 'tipo', R.name AS 'Nombre Rol'
2 FROM `model_has_roles` AS MR
3 INNER JOIN users AS U on U.id = MR.model_id
4 INNER JOIN roles AS R on R.id = MR.role_id
```



Nombre Usuario	tipo	Nombre Rol
jefatura	App\Models\User	jefatura
profesor	App\Models\User	profesor

El objetivo de implementar Spatie ha sido que únicamente jefatura pueda acceder a las rutas de gestión de la web, por lo que para ello deberemos utilizar el Middleware que controla los roles.

Para implementarlo se debe modificar el archivo Kernel.php en app\Http\Middleware:



Al estar usando Laravel 10, únicamente hay que indicar el alias en `MiddlewareAliases`:

```
/*
protected $middlewareAliases = [
    'auth' => \App\Http\Middleware\Authenticate::class,
    'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
    'auth.session' => \Illuminate\Session\Middleware\AuthenticateSession::class,
    'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,
    'can' => \Illuminate\Auth\Middleware\Authorize::class,
    'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
    'role' => \Spatie\Permission\Middleware\RoleMiddleware::class,
    'password.confirm' => \Illuminate\Auth\Middleware\RequirePassword::class,
    'signed' => \App\Http\Middleware\ValidateSignature::class,
    'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,
    'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,
];

```



Y ya podremos utilizar dicho Middleware para agrupar las rutas de gestión y que solo sean accesibles por jefatura:

```
Route::get('/pruebaaz', [App\Http\Controllers\HomeController::class, 'pruebaaz'])->name('pruebaaz');

Route::group(['middleware' => ['role:jefatura']], function () {
    Route::get('/gestion/incidencias', [App\Http\Controllers\IncidentesController::class, 'index'])->name('gestion_incidencias');
    Route::post('/gestion/incidencias', [App\Http\Controllers\IncidentesController::class, 'store'])->name('gestion_incidencias_store');
    Route::get('/gestion/incidencias/editar/{id}', [App\Http\Controllers\IncidentesController::class, 'edit'])->name('gestion_incidencias_editar');
    Route::get('/gestion/incidencias/habilitar/{id}', [App\Http\Controllers\IncidentesController::class, 'habilitar'])->name('gestion_incidencias_habilitar');
    Route::get('/gestion/incidencias/eliminar/{id}', [App\Http\Controllers\IncidentesController::class, 'destroy'])->name('gestion_incidencias_eliminar');

    Route::get('/gestion/correccionesaplicadas', [App\Http\Controllers\CorreccionesAplicadasController::class, 'index'])->name('gestion_correccionesaplicadas');
    Route::post('/gestion/correccionesaplicadas', [App\Http\Controllers\CorreccionesAplicadasController::class, 'store'])->name('gestion_correccionesaplicadas_store');
    Route::get('/gestion/correccionesaplicadas/editar/{id}', [App\Http\Controllers\CorreccionesAplicadasController::class, 'edit'])->name('gestion_correccionesaplicadas_editar');
    Route::get('/gestion/correccionesaplicadas/habilitar/{id}', [App\Http\Controllers\CorreccionesAplicadasController::class, 'habilitar'])->name('gestion_correccionesaplicadas_habilitar');
});
```

Podemos comprobarlo, al intentar acceder con profesor a la ruta se notifica que no tiene los roles adecuados:

The screenshot shows a web browser window with the following details:

- Address bar: 127.0.0.1:8006/gestion/conductasnegativas
- Toolbar: Gmail, YouTube, Maps
- Content area:
 - A red box highlights the URL in the address bar.
 - An error message box at the bottom contains the text "403 | USER DOES NOT HAVE THE RIGHT ROLES." with a red border.



Para reflejar esto también en el menú de navegación, usaremos una sentencia de Blade llamada @role, que hará que solo se muestre el contenido si el usuario usado es el indicado entre paréntesis:

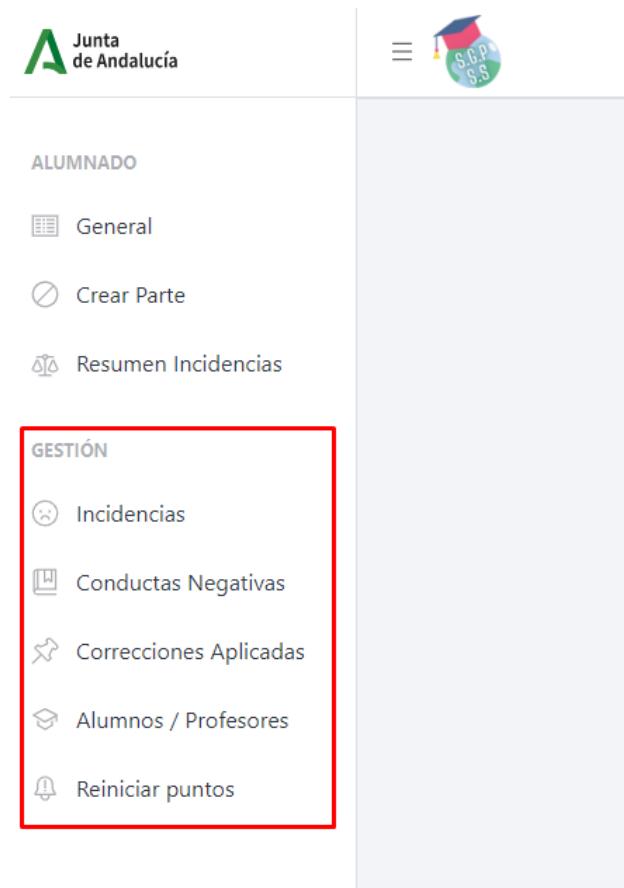
```
<use xlink:href="{{asset('/vendors/@coreui/icons/svg/free.svg#cil-balance-scale')}}">
</svg>
<span data-coreui-i18n="dashboard">Resumen Incidencias</span></a></li>
<!-- Gestión para jefatura -->
@role('jefatura')
<li class="nav-title" data-coreui-i18n="theme">Gestión</li>
<li class="nav-item"><a class="nav-link" href="{{route('gestion.incidencias')}}">
    <svg class="nav-icon">
        <use xlink:href="{{asset('/vendors/@coreui/icons/svg/free.svg#cil-sad')}}"></use>
    </svg>
    <span data-coreui-i18n="dashboard">Incidencias</span></a></li>
<li class="nav-item"><a class="nav-link" href="{{route('gestion.negativas')}}">
    <svg class="nav-icon">
        <use xlink:href="{{asset('/vendors/@coreui/icons/svg/free.svg#cil-book')}}"></use>
    </svg>
    <span data-coreui-i18n="dashboard">Conductas Negativas</span></a></li>
<li class="nav-item"><a class="nav-link" href="{{route('gestion.correcciones')}}">
    <svg class="nav-icon">
```

Como resultado, el menú de navegación se mostraría de la siguiente forma:

The screenshot shows a mobile-style navigation menu. At the top left is the logo of the Junta de Andalucía. To its right is a circular icon containing a graduation cap and the letters 'S.G.P. S.S.'. A three-line menu icon is also present. The main content area has a light gray background. It displays the word 'ALUMNADO' in bold capital letters. Below it are four menu items, each with an icon and text: 'General' (document icon), 'Crear Parte' (cross-out document icon), and 'Resumen Incidencias' (bar chart icon). The 'Gestión' item is expanded, showing its sub-items: 'Incidencias' (document icon) and 'Conductas Negativas' (book icon).



Y el de jefatura mostraría además las rutas de gestión:

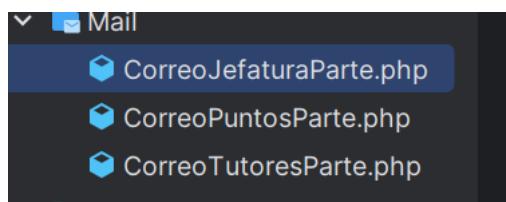


The screenshot shows a sidebar menu with the following items:

- ALUMNADO
 - General
 - Crear Parte
 - Resumen Incidencias
- GESTIÓN
 - Incidencias
 - Conductas Negativas
 - Correcciones Aplicadas
 - Alumnos / Profesores
 - Reiniciar puntos

4.6 Mailer

Tenemos configurados tres tipos correos, en este caso declaramos la estructura de los mismos en el propio content y no en una vista:



- **Correo de Jefatura:** en este correo se enviará el parte colectivo o individual que se le haya puesto en ese momento con toda la información de todos los alumnos implicados y con la información del parte este correo solo lo podrá ver jefatura. En estos momentos en el código está indicado que se envíe al correo de Sergio o Alejandro, por lo que habrá que modificarlo.

**Estructura:****Notificación de Parte**

Estimado miembro de la jefatura,

Le informamos se ha eliminado un parte. A continuación, encontrará los detalles:

Detalles del Parte:

Curso: 3º de ESO

Unidad: 3º de ESO B

Profesor: Francisca Guerrero

Tipo de parte: Individual

Puntos de penalización: 3

Fecha del Parte: 2024-06-09 18:35:04

Alumnos Implicados:

Nombre: Nuria Barela

Tipo De Parte:

Individual

Incidencias:

- Jugar en clase
- Uso del móvil sin permiso
- Uso del PC

Conductas Negativas:

- Actuaciones incorrectas hacia algún miembro de la comunidad educativa.
- Reiteración en un mismo curso de conductas contrarias a las normas de convivencia.
- No asistencia a una actividad en la que se había comprometido.

Correcciones Aplicadas:

- Suspender el derecho de asistencia al centro entre 1 y 3 días.
- Realizar tareas fuera del horario lectivo del Centro.
- Permanecimiento en clase durante el tiempo de recreo

Descripción detallada del Parte:

Un saludo

Este mensaje es una notificación. No responda directamente a este correo. Pulse en el asunto para acceder al correo de la plataforma



- **Correo Tutores:** en este tipo de parte se enviará a los correos que tenga asociado el alumno con solo dicha información del alumno y del parte, ya que no queremos mostrar datos de otros alumnos. Este tipo de correo se enviará por ejemplo a los padres del alumno o a si mismo:

Notificación de Parte

Estimado tutor,

Le informamos de que el parte del alumno Nuria Barela Se ha eliminado. A continuación, encontrará los detalles:

Detalles del Parte:

Curso: 3º de ESO

Unidad: 3º de ESO B

Profesor: Francisca Guerrero

Tipo de parte: Individual

Puntos de penalización: 3

Puntos Actuales del Alumno: 10

Fecha del Parte: 2024-06-09 18:35:04

Incidencias:

- Jugar en clase
- Uso del móvil sin permiso
- Uso del PC

Conductas Negativas:

- Actuaciones incorrectas hacia algún miembro de la comunidad educativa.
- Reiteración en un mismo curso de conductas contrarias a las normas de convivencia.
- No asistencia a una actividad en la que se había comprometido.

Correcciones Aplicadas:

- Suspender el derecho de asistencia al centro entre 1 y 3 días.
- Realizar tareas fuera del horario lectivo del Centro.
- Permanecimiento en clase durante el tiempo de recreo

Descripción detallada del Parte:

Un saludo

Este mensaje es una notificación. No responda directamente a este correo. Pulse en el asunto para acceder al correo de la plataforma

- **Correo de Puntos:** este correo se enviará tanto a jefatura como a los tutores de los alumnos que se han quedado sin puntos en el carnet.



Sergio <sergioggbb02@gmail.com>

para mí ▾

Aviso Importante

El alumno D. Isaac Cordero se ha quedado sin puntos en su carnet.

Un saludo

Este mensaje es una notificación. No responda directamente a este correo. Pulse en el asunto para acceder al correo de la plataforma



4.7 Excel importaciones

Para las importaciones, este es el código php que se encarga de importar todos los alumnos y sus clases. Primero, calcula el año académico actual y luego busca en la base de datos un registro de ese año académico. Si no existe, lo crea. Luego, busca un curso con el nombre proporcionado en la fila de datos que se está procesando. Si no existe un curso o unidad con ese nombre para el año académico actual, lo crea:

```
$year = date('Y');
$nextYear = date('Y', strtotime('datetime: +1 year'));

$anoacademico = AnioAcademico::firstOrCreate(['anio_academico' => $year . '-' . $nextYear]);

$curso = Curso::firstOrCreate(['nombre' => $row['nombre_curso'], 'id_anio_academico' => $anoacademico->id]);

// Crear o encontrar la unidad
$unidad = Unidad::firstOrCreate(['nombre' => $row['nombre_unidad'], 'id_curso' => $curso->id]);
```

Finalmente, crea un nuevo alumno con los datos proporcionados en la fila:

```
// Crear el alumno
return new Alumno([
    'dni' => $row['dni'],
    'nombre' => $row['nombre'],
    'correo' => $row['correo'],
    'puntos' => 12,
    'id_unidad' => $unidad->id,
]);
```

- **DNI**
- **nombre**
- **correo electrónico** (de los tutores legales y el personal del alumno)
- **puntos** (se establecen siempre a 12)
-

Este nuevo alumno no se guarda en la base de datos en este punto, simplemente se crea y se devuelve como resultado de la función.



Aquí en base a lo que nos devuelve la importación, primero eliminamos todos los datos de las tablas para vaciarlas para el nuevo curso y una vez vaciadas, se van creando los alumnos y sus clases:

```
// Comprueba si se ha pasado un archivo
if (!$request->hasFile( key: 'import_file')) {
    return redirect()->route( route: 'users.import')
        ->with('error', 'No se ha subido ningún archivo.');
}

DB::statement( query: 'SET FOREIGN_KEY_CHECKS=0;');

Correos::truncate();
Alumno::truncate();
Curso::truncate();
Unidad::truncate();
AñoAcademico::truncate();

DB::statement( query: 'SET FOREIGN_KEY_CHECKS=1;');

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

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

return redirect()->route( route: 'users.import')
```

En esta vista cargamos la importación de los alumnos y las clases:

Importar Alumnos y Clases

Seleccionar archivo Ningún archivo seleccionado

Importar

4.8 Vistas, estructura y comportamiento general

4.8.1 Inicio de sesión

No se requerirá registro ya que los datos de inicio de sesión han sido indicados por el centro.

Como todas las vistas de la aplicación, será responsive, y usará una autenticación automática de Laravel Breeze:





4.8.2 Alumnado: Partes

Esta vista muestra todos los partes de una unidad, y permite crearlos, editarlos, eliminarlos y descargar su PDF correspondiente. Los datos de los partes se muestran en un DataTable. El Datatable permite desplazarse por su contenido cuando el tamaño de visualización se reduce:

Incidencias Alumnos

Año Académico
2023-2024

Curso
2º de ESO

Unidad
2º de ESO B

Crear

Mostrar 10 registros Buscar:

Fecha	Nombre	¿Colectivo?	Descripción	Conducta Negativa	Puntos Penalizados	Acciones
10/06/2024 21:21	Úrsula Alcántar	No	Malos modos	<ul style="list-style-type: none">Falta de colaboración sistemática en la realización de las actividades.	4	
				<ul style="list-style-type: none">Actuaciones incorrectas hacia algún miembro de la comunidad educativa.Falta de colaboración sistemática en la realización de las actividades.Impedir o dificultar el		
10/06/2024 20:10	Oliver De Ando	Si	Malos modos		1	

Incidencias Alumnos

Año Académico
2023-2024

Curso
2º de ESO

Unidad
2º de ESO B

Crear

Mostrar 10 registros Buscar:

Fecha	Nombre	¿Colectivo?	Descripción	CN
10/06/2024 21:21	Úrsula Alcántar	No	Malos modos	.
				.



En este modal creará el parte con el Alumno o alumnos de esa clase el profesor del parte, la fecha y hora del parte, incidencias, tramos horarios, correcciones aplicadas, conductas negativas, y los puntos de penalización, además con CKEditor que nos permitirá subir imágenes, tablas y texto formateado, se comparte su estructura para editar:

Crear nuevo parte

Fecha:

Profesor:

Tramo Horario:

Alumno Implicados:

Incidencia:

Conductas Negativas:

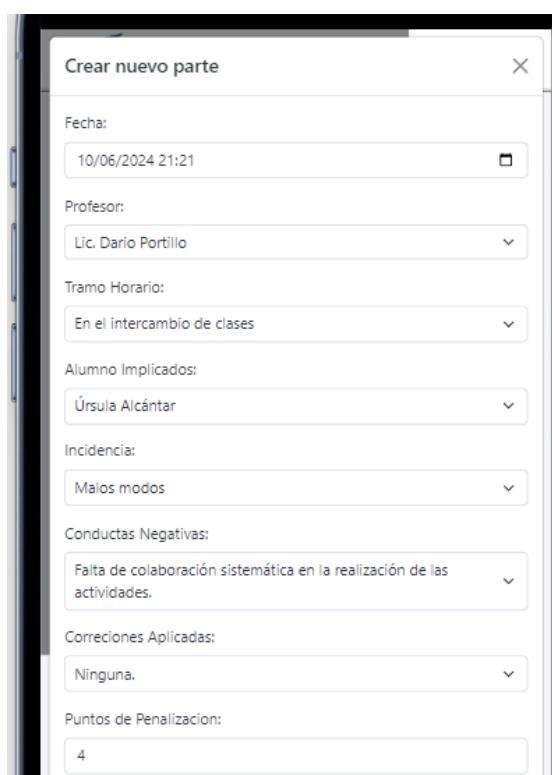
Correcciones Aplicadas:

Puntos de Penalizacion:

Descripcion Detallada:

Párrafo | **A[±]** **A_±** **A_±** **B** **I** **Ø** **:=** **:=** **≡** **≡** | **¶** **¶** **“** **”** **█** **█** **□** **□** **↔** **↔**

Cerrar **Guardar cambios**





4.8.3 Alumnado: Resumen Incidencias

En esta vista se podrá filtrar por los datos de cada unidad de curso para ver un resumen de cada alumno, sus incidencias, conductas y puntos restantes:

Incidencias Alumnos

Año Académico: 2023-2024

Curso: 2º de ESO

Unidad: 2º de ESO B

Mostrar 10 registros Buscar: _____

Nombre	Incidencias	Conductas grave	Conductas contrarias	Puntos restantes
Oliver De Andra Tercero	4	1	6	5
Úrsula Alcántar	2	2	7	6

Mostrando registros del 1 al 2 de un total de 2 registros

Anterior 1 Siguiente

4.8.4 Gestión: Incidencias

La vista de incidencias recoge los accidentes generales por los que puede darse un parte, pudiendo el rol de jefatura añadir más, editar los existentes o deshabilitarlas:

Junta de Andalucía Jefatura

ALUMNADO

- Partes
- Resumen Incidencias

GESTIÓN

- Incidencias
- Conductas Negativas
- Correcciones
- Alumnos / Profesores
- Reiniciar puntos
- Importaciones

Gestión de Incidencias

Añadir nueva Incidencia

Descripción de la incidencia a añadir:

Descripción incidencia Añadir Limpiar

Listado de Incidencias

#	Descripción	Opciones
1	Jugar en clase	Editar Deshabilitar
2	Pelea con un compañero	Editar Deshabilitar
3	Malos modos	Editar Deshabilitar
4	Jugar en clase	Editar Deshabilitar
5	Uso del móvil sin permiso	Editar Deshabilitar

Mostrando de 1 hasta 5 de 6 resultados

Ver deshabilitadas

Los usuarios pueden desplazarse en la tabla con el dedo:



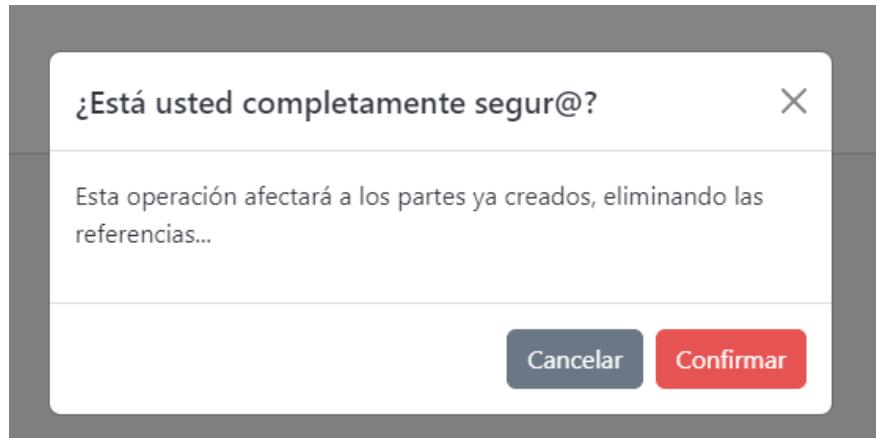
Si el botón de ver deshabilitadas es pulsado, se llevará a otra vista que mostrará las incidencias deshabilitadas, pudiendo habilitarlas de nuevo o eliminarlas completamente, dejando a nulas las referencias en los partes si es el caso:

#	Descripción	Opciones
7	Uso del PC	<button>Habilitar</button> <button>Eliminar</button>

[Ver habilitadas](#)



Si se pulsa eliminar se mostrará un modal indicando que la acción es irreversible:



4.8.5 Gestión: Conductas negativas

La vista de Conductas negativas sigue la misma lógica, con el añadido del tipo de incidencia, pudiéndose también habilitarse y deshabilitarse a voluntad:



De la misma forma, tendrá una vista especial para volver a habilitar las deshabilitadas o eliminarlas completamente:

Gestión de Conductas Negativas deshabilitadas

Listado de Conductas Negativas deshabilitadas

#	Descripción	Tipo	Opciones
7	No asistencia a una actividad en la que se había comprometido.	Grave	<button>Habilitar</button> <button>Eliminar</button>

Ver habilitadas

4.8.6 Gestión: Correcciones

Vista de estructura similar a las anteriores, pero referida a las posibles correcciones que se le pueden aplicar a un alumno tras un parte:

Gestión de Correcciones

Añadir nueva Corrección

Descripción de la Corrección a añadir:

Añadir Limpiar

Listado de Correcciones

#	Descripción	Opciones
1	Suspender el derecho de asistencia al centro entre 1 y 3 días.	<button>Editar</button> <button>Deshabilitar</button>
2	Suspender el derecho de asistencia al centro entre 4 y 30 días.	<button>Editar</button> <button>Deshabilitar</button>
3	Realizar tareas fuera del horario lectivo del Centro.	<button>Editar</button> <button>Deshabilitar</button>

Ver deshabilitadas

También con su apartado para las correcciones deshabilitadas:

Gestión de Correcciones deshabilitadas

Listado de Correcciones deshabilitadas

#	Descripción	Opciones
4	Permanecimiento en clase durante el tiempo de recreo	<button>Habilitar</button> <button>Eliminar</button>

Ver habilitadas



4.8.7 Gestión: Alumnos / Profesores

La vista profesoralumno es la más compleja de este apartado de la web. Esto es debido a que realmente son dos vistas en una, controlado por un Switch que cambia el div mostrado.

En primer lugar nos centraremos en el apartado de alumno. Alumno tiene un Datatable que permite la gestión y búsqueda de todos los alumnos así como filtros por año, curso y unidades que van filtrando progresivamente:

DNI	Nombre	Correos	Puntaje
98460546F	Daniela Contreras Tercero	elena77@example.net (tutor)	6
97591267J	Dr. Víctor Pabón		8



Si seleccionamos todos los filtros, indicando una unidad, podremos crear un nuevo alumno mediante el modal correspondiente:

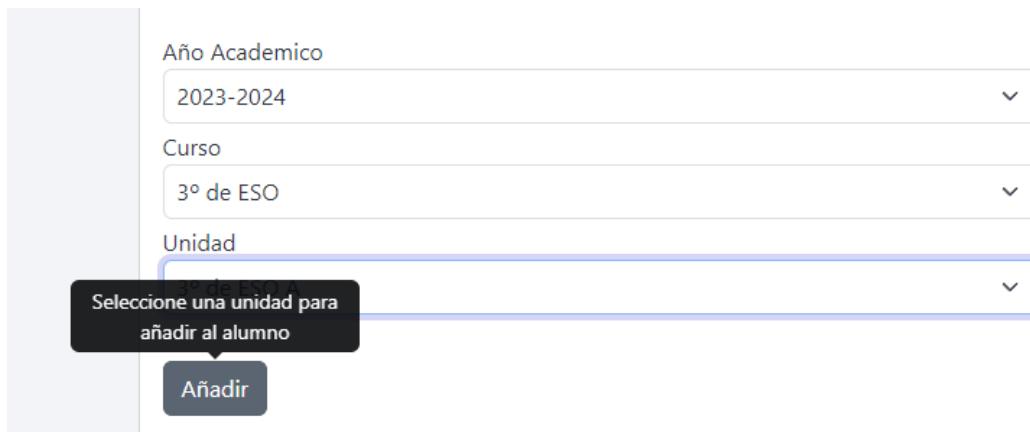
Año Académico
2023-2024

Curso
3º de ESO

Unidad

Selección de unidad para añadir al alumno

Añadir



Formulario

DNI:

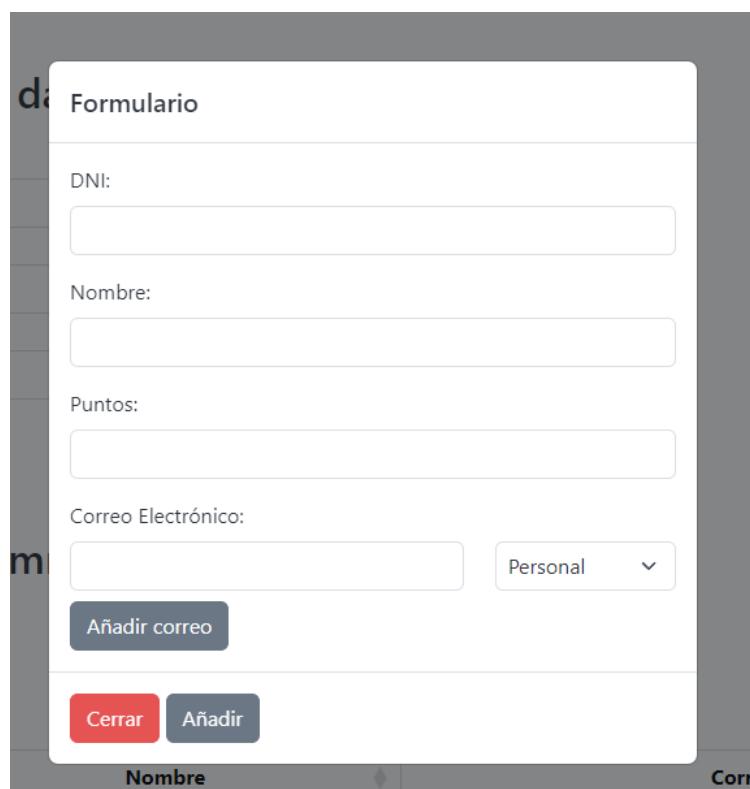
Nombre:

Puntos:

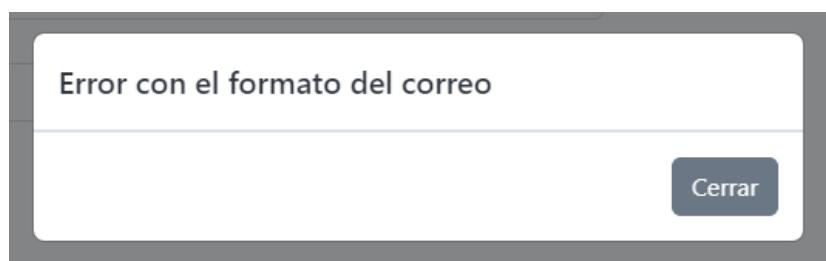
Correo Electrónico:
 Personal

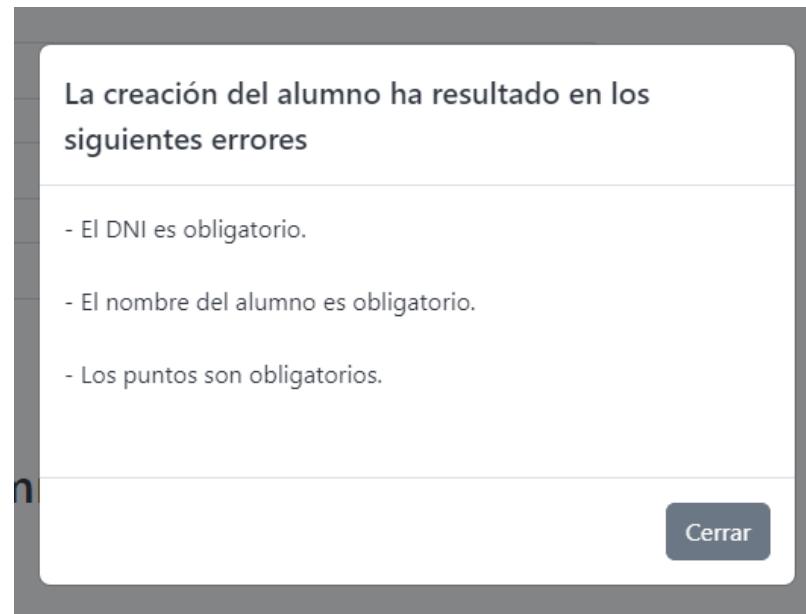
Añadir correo

Cerrar Añadir



Las excepciones y otros fallos se mostrarán en otros modales, para así no perder los datos insertados:





Si queremos editar un alumno, únicamente tendremos que clicarlo o pulsarlo en la tabla, de forma que se nos aparecerá un modal que además de generar cambios, nos permitirá eliminarlo:

Formulario

DNI:

Nombre:

Puntos:

Correo Electrónico: Personal ▾

Añadir correo

Lista de correos registrados:

cristina.rangel@example.org (personal)	Borrar
toro.marcos@example.com (tutor)	Borrar
oabreu@example.org (personal)	Borrar

Eliminar **Cerrar** **Guardar**



SISTEMA DE GESTIÓN DE PARTES

Si cambiamos el Switch accederemos al apartado de profesores, que sigue un formato más similar al resto de vistas de esta sección pero con más campos con sus respectivas validaciones:

Gestión de Profesores y Alumnos

Profesores

Añadir nuevo profesor/a

Campos del nuevo profesor/a:

DNI	Nombre	Teléfono	Correo	Añadir	Limpiar
-----	--------	----------	--------	------------------------	-------------------------

Listado de Profesores/as

DNI	Nombre	Teléfono	Correo	Opciones
68984667P	Ariadna Oliver Tercero	611039369	paula.benitez@aponte.es	Editar Deshabilitar
62197506D	Cristina Olivo	445192990	veronica50@sanchez.com	Editar Deshabilitar
30068351Z	Dr. Cristina Bahena Segundo	651365489	tmarti@altamirano.com	Editar Deshabilitar
92552514P	Ing. Noa Llamas Segundo	569162863	soriano.lucia@hispavista.com	Editar Deshabilitar
86261895G	Iván Aponte Tercero	524613704	rcano@gmail.com	Editar Deshabilitar

Mostrando de 1 hasta 5 de 12 resultados

[Ver deshabilitad@s](#)

También profesores tendrán un apartado para los deshabilitados, en el que únicamente se mostrarán estos, siendo independientes a alumnos:

Gestión de Profesores deshabilitad@s

Listado de Profesores deshabilitad@s

DNI	Nombre	Teléfono	Correo	Opciones
55940701D	Andrea Segura	556146741	alexandra.izquierdo@hotmail	Habilitar Eliminar
49427436E	Carolina Sáenz	262788526	jfont@hotmail.com	Habilitar Eliminar
62190085V	Guillermo Peña	753533851	bruno24@avalos.es	Habilitar Eliminar
89436378G	Héctor Prieto	906845641	jimena.villar@gmail.com	Habilitar Eliminar
53936673V	Jorge Oliva	450695440	imiranda@ojeda.es	Habilitar Eliminar

Mostrando de 1 hasta 5 de 9 resultados

[Ver habilitad@s](#)



4.8.8 Gestión: Reiniciar puntos

El principal objetivo de esta vista es reiniciar los puntos de todos los alumnos una vez acaba un trimestre, devolviéndolos a 12. Se tiene en cuenta que los alumnos que tienen más de 12 puntos no se les penalice. Al ser una operación sin retorno, se indica con varios elementos que es irreversible y una operación importante:

Alumnos / Profesores

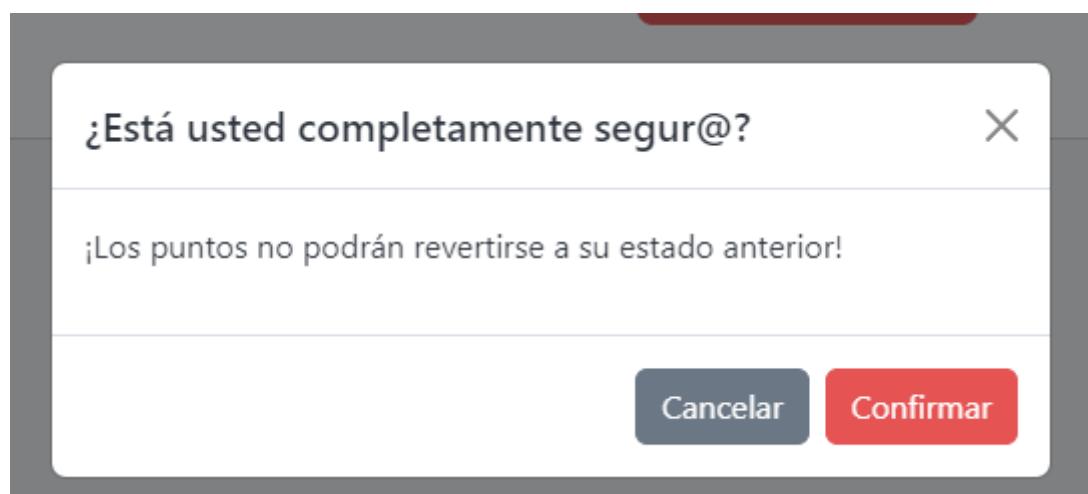
Reiniciar puntos

Importaciones

Reinicio de puntos del alumnado

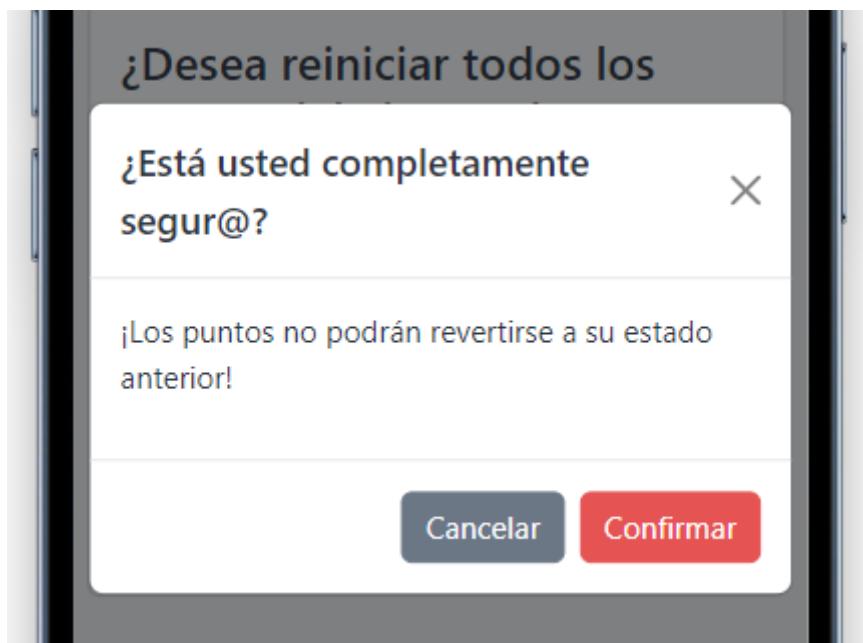
¿Desea reiniciar todos los puntos del alumnado?
Es recomendable realizar la operación únicamente al inicio de cada trimestre.

[Reinic平 puntos](#)





A continuación se muestra la versión adaptada a móvil:





4.8.9 Gestión: Importaciones

Vista cuyo único objetivo es importar los datos de los alumnos al inicio del curso, junto a sus cursos y unidades correspondientes, realizándose las operaciones correspondiente con el paquete barryvdh/laravel-dompdf:

The screenshot shows a user interface titled "Importar Alumnos y Clases" (Import Students and Classes). It features a "Seleccionar archivo" (Select file) button with the message "Ningún archivo seleccionado" (No file selected) displayed next to it. Below this is a prominent blue "Importar" (Import) button. The background of the interface is white, and the overall design is clean and modern.

5. Pruebas y métodos por controlador

5.1 Introducción

En este apartado se tratará la función general de cada uno de los métodos de cada controlador, dividiendo la web en sus dos apartados principales. Además de la realización de pruebas y los resultados que puede ser devuelto en cada situación.



5.2 Alumnado

5.2.1 UsersController

- Método index, pinta los datos de los partes en su DataTable:

```
aleja +1
public function index(ParteDataTable $dataTable)
{
    $anoAcademico = AnioAcademico::all();
    $profesores = Profesor::all()->where( key: 'habilitado', operator: '=', value: true);

    $tramos = Tramohorario::all();

    $cursos = Curso::all();

    $incidencias = Incidencia::all()->where( key: 'habilitado', operator: '=', value: true);

    $conductasNegativas = Conductanegativa::all()->where( key: 'habilitado', operator: '=', value: true);

    $correccionesAplicadas = Correccionaplicada::all()->where( key: 'habilitado', operator: '=', value: true);
    return $dataTable->render([ view: 'users.index', ['anoAcademico' => $anoAcademico, 'profesores' => $profesores, 'tramos' => $tramos, 'cursos' => $cursos, 'incidencias' => $incidencias, 'conductasNegativas' => $conductasNegativas, 'correccionesAplicadas' => $correccionesAplicadas ] );
}
```

Control de errores	Resultado del método
Alguna consulta no devuelve nada	Excepción campos nulos controlada en vista
Si el DataTable no se renderiza	Se lanza un alert con el problema pertinente
Consultas correctas	Se imprime la vista con DataTable

- Validaciones generales al hacer consultas asíncronas con valores predeterminados:

```
$validator = Validator::make($request->all(), [
    'Profesor' => 'required|in:' . implode( separator: ',', Profesor::all()->pluck( value: 'dni')->toArray()),
    'TramoHorario' => 'required|in:' . implode( separator: ',', Tramohorario::all()->pluck( value: 'id')->toArray()),
    'Alumno' => 'required|in:' . implode( separator: ',', Alumno::all()->pluck( value: 'dni')->toArray()),
    'Puntos' => 'required|numeric',
    'Fecha' => 'required|date',
    'Incidencia' => 'required|in:' . implode( separator: ',', Incidencia::all()->pluck( value: 'id')->toArray()),
    'ConductasNegativa' => 'required|in:' . implode( separator: ',', Conductanegativa::all()->pluck( value: 'id')->toArray()),
    'CorreccionesAplicadas' => 'required|in:' . implode( separator: ',', Correccionaplicada::all()->pluck( value: 'id')->toArray())
]);
if ($validator->fails()) {
    return response()->json(['errors' => $validator->errors()], status: 422);
}
```



Control de errores	Resultado del método
Profesor, TramoHorario, Alumno, Conductas Negativas, Correcciones Aplicadas no están en las tablas de la BD o son null	JSON con errores de validación
Puntos no es numérico o esta vacío	JSON con error de que el campo no es numérico
Fecha no es un tipo date	JSON con error de que el campo no es tipo date

- Método eliminarParte, comprueba que existe y elimina las referencias para todos los alumnos:

```
usage: ./aleja
public function eliminarParte($id)
{
    $parte = Parte::find($id);

    $alumnos = AlumnoParte::where('parte_id', $parte->id)->get();

    foreach ($alumnos as $alumno) {
        $alumnoModel = Alumno::where('dni', $alumno->alumno_dni)->first();
        $alumnoModel->increment('puntos', $parte->puntos_penalizados);
        $alumnoModel->save();
        foreach ($alumnoModel->correos as $correo) {
            Mail::to($correo->correo)->queue(new CorreoTutoresParte($alumnoModel, $parte));
            Mail::to('users:alejandrocbt@hotmail.com')->send(new CorreoTutoresParte($alumnoModel, $parte));
        }
    }
    Mail::to('users:alejandrocbt@hotmail.com')->send(new CorreoJefaturaParte($parte, $parte->delete()));

    return redirect()->route('users.index')
        ->with('success', 'Parte eliminado correctamente.');
}
```

Control de errores	Resultado del método
Parte sin referencias	Se elimina directamente mandando una copia de seguridad a jefatura
Si tiene alumnos asociados	Se restablecen sus puntos y se envía un correo notificando



SISTEMA DE GESTIÓN DE PARTES

- Método getParte, obtiene el parte mediante su id, obteniendo todos los datos en consecuencia:

```
Usage: ▶ aleja*
public function getParte($id)
{
    $parteId = $id;
    $parte = Parte::find($parteId);
    $profesorAll = Profesor::all()->where( key: 'habilitado', operator: '=', value: true);
    $profesorAll->push($parte->profesores);
    $alumnos = AlumnoParte::where( column: 'parte_id', $parteId)->get();
    // $profesor = Profesor::where('dni', $parte->profesor_dni)->first()->get();
    $conductasNegativas = ParteConductanegativa::where( column: 'parte_id', $parteId)->get();

    return response()->json([
        'id' => $parte->id,
        'fecha' => Carbon::parse($parte->created_at)->format( format: 'Y-m-d H:i'), // Formato de fecha
        'alumnos' => $alumnos,
        'profesor' => $parte->profesor_dni,
        'profesorAll' => $profesorAll,
        'incidencia' => $parte->incidencias->first()->id,
        'conductasNegativas' => $conductasNegativas,
        'correccionesAplicadas' => $parte->correccionesaplicadas->first()->id,
        'tramoHorario' => $parte->tramo_horario_id,
        'puntos' => $parte->puntos_penalizados,
        'descripcionDetallada' => $parte->descripcion_detallada,
    ]);
}
```

Control de errores	Resultado del método
Consultas nulas o erróneas	JSON con mensaje de error
Consultas correctas	Devuelve el parte con sus datos para mostrarlos



- descargarPartePDF, crea un parte con su id para obtener los datos y que se descargue en el cliente:

```
  Usage: ▾ aleja
  public function descargarPartePDF($id)
  {
      // Obtén el parte basado en el ID
      $parte = Parte::find($id);

      // Comprueba si la descripción detallada está vacía
      if (!empty($parte->descripcion_detallada)) {
          // Crea una nueva instancia de DOMDocument
          $dom = new \DOMDocument();

          // Carga el contenido HTML en DOMDocument
          @$dom->loadHTML($parte->descripcion_detallada, [options: LIBXML_HTML_NOIMPLIED | LIBXML_HTML_NODEFDTD]);

          // Encuentra todas las imágenes en el contenido HTML
          $images = $dom->getElementsByTagName(qualifiedName: 'img');

          // Itera sobre todas las imágenes
          foreach ($images as $image) {
              // Guarda el contenido HTML con las rutas de las imágenes actualizadas
              $parte->descripcion_detallada = $dom->saveHTML();
          }

          // Genera el PDF a partir de una vista (reemplaza 'pdf' con el nombre de tu vista)
          $pdf = PDF::loadView('users.partePDF', ['parte' => $parte]);
      }

      // Devuelve el PDF como una respuesta
      return response()->make($pdf->output(), 200, [
          'Content-Type' => 'application/pdf',
          'Content-Disposition' => 'inline; filename="parte.pdf"',
      ]);
  }
```

Control de errores	Resultado del método
Error al obtener datos	Devuelve JSON con error 500
Datos obtenidos y PDF creado	Se descarga el PDF en el PC del cliente

- Método upload, se encarga de subir las imágenes del CKEditor al servidor y asociarla con el campo de dicho editor:

```
  ↵ Alejandro
  public function upload(Request $request)
  {
      $file = $request->file(key: 'upload');
      $fileName = uniqid() . '_' . trim($file->getClientOriginalName());

      $file->move(public_path('uploads'), $fileName);

      return response()->json([
          'uploaded' => 1,
          'fileName' => $fileName,
          'url' => '/uploads/' . $fileName
      ]);
  }
```



Control de errores	Resultado del método
Error al subir imagen	Se devuelve error en JSON con mensaje de que la imagen no se ha podido subir
Imagen cargada correctamente	Se guarda en el servidor y se asocia en BD

- Método import, importa los datos del archivo Excel añadido y los vuelca en la Base de Datos:

```
  ↵ aleja +1
  public function import(Request $request)
  {
    // Comprueba si se ha pasado un archivo
    if (!$request->hasFile('import_file')) {
      return redirect()->route('users.import')
        ->with('error', 'No se ha subido ningún archivo.');
    }

    DB::statement('SET FOREIGN_KEY_CHECKS=0');

    Correo::truncate();
    Alumno::truncate();
    Curso::truncate();
    Unidad::truncate();
    AnioAcademico::truncate();

    DB::statement('SET FOREIGN_KEY_CHECKS=1');

    $file = $request->file('import_file');

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

    return redirect()->route('users.import')
      ->with('Satisfactorio', 'Datos importados correctamente.');
  }
```

Control de errores	Resultado del método
Creación errónea	Devuelve error
Datos correctos	Se guardan los datos del Excel en Base de Datos



5.3 Gestión

5.3.1 IncidenciasController

- Método index, devuelve las incidencias habilitadas paginadas para mostrarlas, sin margen de error:

```
public function index() {  
    $incidencias = Incidencia::select("*")->where('habilitado', '=', true);  
    $tandaIncidencias = $incidencias->paginate(5);  
    return view('gestion.incidencias', ["incidencias" => $tandaIncidencias]);  
}
```

- Método deshabilitadas, mismo procedimiento pero devolviendo las deshabilitadas, llevando a la vista correspondiente, sin margen de error:

```
public function deshabilitadas() {  
    $incidencias = Incidencia::select("*")->where('habilitado', '=', false);  
    $tandaIncidencias = $incidencias->paginate(5);  
    return view('gestion.deshabilitados.incidencias', ["incidencias" => $tandaIncidencias]);  
}
```

- Método crear, valida los datos y crea la incidencia:

```
public function crear(Request $request) {  
    $request->validate([  
        'nuevaIncidencia' => ['required', 'string', 'min:3', 'max:80'],  
    ], [  
        'nuevaIncidencia.requires' => 'La descripción de la incidencia es obligatoria.',  
        'nuevaIncidencia.string' => 'La descripción de la incidencia debe ser una cadena.',  
        'nuevaIncidencia.min' => 'La descripción de la incidencia debe tener al menos 3 caracteres.',  
        'nuevaIncidencia.max' => 'La descripción de la incidencia debe tener menos de 80 caracteres.',  
    ]);  
    $incidencia = Incidencia::create([  
        'descripcion' => $request['nuevaIncidencia'],  
        'habilitado' => true,  
    ]);  
    return back()->with('success', 'Incidencia creada con éxito');  
}
```

Control de errores	Resultado del método
Error en validación	Incidencia no creada, mensaje en vista de campos incorrectos
Validación correcta	Incidencia creada y habilitada por defecto



- Método editar, obtiene el id mediante la ruta de edición y valida los datos para editar una incidencia:

```
public function editar(Request $request, $id) {  
    $request->validate([  
        'cambioIncidencia' => ['required', 'string', 'min:3', 'max:80'],  
    ], [  
        'cambioIncidencia.requires' => 'La descripción de la incidencia es obligatoria.',  
        'cambioIncidencia.string' => 'La descripción de la incidencia debe ser una cadena.',  
        'cambioIncidencia.min' => 'La descripción de la incidencia debe tener al menos 3 caracteres.',  
        'cambioIncidencia.max' => 'La descripción de la incidencia debe tener menos de 80 caracteres.',  
    ]);  
    $nuevaDescripcion = $request->cambioIncidencia;  
    if ($id > 0 && $id < 1000) {  
        if (Incidencia::find($id) != null) {  
            $incidenciaEditar = Incidencia::find($id);  
            $incidenciaEditar->descripcion = $nuevaDescripcion;  
            $incidenciaEditar->save();  
            return back()->with('success', 'Incidencia editada con éxito');  
        }  
    }  
    return back()->with('success', 'Error al editar la incidencia');  
}
```

Control de errores	Resultado del método
Error en validación	Incidencia no editada, mensaje en vista de campos incorrectos
Id con formato incorrecto	Incidencia no editada, mensaje en vista de error al editar
Incidencia no encontrada con id	Incidencia no editada, mensaje en vista de error al editar
Validación correcta	Incidencia editada

- Método eliminar, obtiene el id de la ruta y comprueba que existe y es válido para eliminar la incidencia correspondiente:

```
public function eliminar($id) {  
    if ($id > 0 && $id < 1000) {  
        if (Incidencia::find($id) != null) {  
            Incidencia::destroy($id);  
            return redirect('/gestion/incidencias')->with('success', 'La incidencia se ha eliminado con éxito');  
        }  
        return redirect('/gestion/incidencias')->with('success', 'La incidencia a eliminar no existe');  
    }  
    return redirect('/gestion/incidencias')->with('success', 'Error al eliminar la incidencia');  
}
```



Control de errores	Resultado del método
Id con formato incorrecto	Incidencia no borrada, mensaje en vista de error al eliminar
Incidencia no encontrada con id	Incidencia no borrada, mensaje en vista de error al eliminar
Validación correcta	Incidencia eliminada

- Método habilitar, recibe el id por Request y lo valida, para cambiar el estado de la Incidencia:

```
public function habilitar(Request $request) {
    $request->validate([
        'id' => ['numeric', 'min:1', 'max:500'],
    ], [
        'id' => 'El identificador no es un valor numérico correcto. ',
    ]);
    $id = $request->id;
    if (count(Incidencia::select('*')->where('id','=',$id)->get()) == 0) return back()->with('success', 'La incidencia no existe');
    $incidencia = Incidencia::select('*')->where('id','=',$id)->get()[0];
    Incidencia::where('id','=',$id)->update(['habilitado' => !($incidencia->habilitado)]);

    if ($incidencia->habilitado) return back()->with('success', 'Incidencia deshabilitada');
    else return back()->with('success', 'Incidencia habilitada');
}
```

Control de errores	Resultado del método
Error en validación	Incidencia no cambiada, mensaje en vista de campos incorrectos
Incidencia no encontrada con id	Incidencia no cambiada, mensaje en vista de que incidencia no existe
Validación correcta e incidencia deshabilitada	Incidencia habilitada
Validación correcta e incidencia habilitada	Incidencia deshabilitada



5.3.2 ConductasNegativasController

- Método index, devuelve las conductas negativas habilitadas paginadas para mostrarlas, sin margen de error:

```
public function index() {  
    $conductas = Conductanegativa::select("*")->where('habilitado','=',true);  
    $tandaConductas = $conductas->paginate(5);  
    return view('gestion.negativas', ["conductas" => $tandaConductas]);  
}
```

- Método deshabilitadas, mismo procedimiento pero devolviendo las deshabilitadas, llevando a la vista correspondiente, sin margen de error:

```
public function deshabilitadas() {  
    $conductas = Conductanegativa::select("*")->where('habilitado','=',false);  
    $tandaConductas = $conductas->paginate(5);  
    return view('gestion.deshabilitados.negativas', ["conductas" => $tandaConductas]);  
}
```

- Método crear, valida los datos y crea la conducta negativa:

```
public function crear(Request $request) {  
    $request->validate([  
        'nuevaConducta' => ['required', 'string', 'min:3', 'max:80'],  
        'nuevaConductaTipo' => ['required', 'string', 'in:Contraria,Grave'],  
    ], [  
        'nuevaConducta.requires' => 'La descripción de la conducta es obligatoria.',  
        'nuevaConducta.string' => 'La descripción de la conducta debe ser una cadena.',  
        'nuevaConducta.min' => 'La descripción de la conducta debe tener al menos 3 caracteres.',  
        'nuevaConducta.max' => 'La descripción de la conducta debe tener menos de 80 caracteres.',  
        'nuevaConductaTipo.requires' => 'El tipo de la conducta es obligatorio.',  
        'nuevaConductaTipo.string' => 'El tipo de la conducta debe ser una cadena.',  
        'nuevaConductaTipo.in' => 'El tipo de la conducta debe ser Contraria o Grave.',  
    ]);  
    $conducta = Conductanegativa::create([  
        'descripcion' => $request['nuevaConducta'],  
        'tipo' => $request['nuevaConductaTipo'],  
        'habilitado' => true,  
    ]);  
    return back()->with('success', 'Conducta Negativa creada con éxito');  
}
```

Control de errores	Resultado del método
Error en validación	Conducta negativa no creada, mensaje en vista de campos incorrectos
Validación correcta	Conducta negativa creada y habilitada por defecto



- Método editar, obtiene el id mediante la ruta de edición y valida los datos para editar una conducta negativa:

```
public function editar(Request $request, $id) {  
    $request->validate([  
        'cambioConducta' => ['required', 'string', 'min:3', 'max:80'],  
        'cambioConductaTipo' => ['required', 'string', 'in:Contraria,Grave'],  
    ], [  
        'cambioConducta.requires' => 'La descripción de la conducta es obligatoria.',  
        'cambioConducta.string' => 'La descripción de la conducta debe ser una cadena.',  
        'cambioConducta.min' => 'La descripción de la conducta debe tener al menos 3 caracteres.',  
        'cambioConducta.max' => 'La descripción de la conducta debe tener menos de 80 caracteres.',  
        'cambioConductaTipo.requires' => 'El tipo de la conducta es obligatorio.',  
        'cambioConductaTipo.string' => 'El tipo de la conducta debe ser una cadena.',  
        'cambioConductaTipo.in' => 'El tipo de la conducta debe ser Contraria o Grave.',  
    ]);  
    $nuevaDescripcion = $request->cambioConducta;  
    $nuevaDescripcionTipo = $request->cambioConductaTipo;  
    if ($id > 0 && $id < 1000) {  
        if (Conductanegativa::find($id) != null) {  
            $conductaEditar = Conductanegativa::find($id);  
            $conductaEditar->descripcion = $nuevaDescripcion;  
            $conductaEditar->tipo = $nuevaDescripcionTipo;  
            $conductaEditar->save();  
            return back()->with('success', 'Conducta negativa editada con éxito');  
        }  
    }  
    return back()->with('success', 'Error al editar la conducta');  
}
```

Control de errores	Resultado del método
Error en validación	Conducta negativa no editada, mensaje en vista de campos incorrectos
Id con formato incorrecto	Conducta negativa no editada, mensaje en vista de error al editar
Conducta negativa no encontrada con id	Conducta negativa no editada, mensaje en vista de error al editar
Validación correcta	Conducta negativa editada



- Método eliminar, obtiene el id de la ruta y comprueba que existe y es válido para eliminar la conducta negativa correspondiente:

```
public function eliminar($id) {
    if ($id > 0 && $id < 1000) {
        if (Conductanegativa::find($id) != null) {
            Conductanegativa::destroy($id);
            return redirect('/gestion/conductasnegativas')->with('success', 'La conducta negativa se ha eliminado con éxito');
        }
        return redirect('/gestion/conductasnegativas')->with('success', 'La conducta negativa a eliminar no existe');
    }
    return redirect('/gestion/conductasnegativas')->with('success', 'Error al eliminar la conducta negativa');
}
```

Control de errores	Resultado del método
Id con formato incorrecto	Conducta negativa no borrada, mensaje en vista de error al eliminar
Conducta negativa no encontrada con id	Conducta negativa no borrada, mensaje en vista de error al eliminar
Validación correcta	Conducta negativa eliminada

- Método habilitar, recibe el id por Request y lo valida, para cambiar el estado de la Conducta negativa:

```
public function habilitar(Request $request) {
    $request->validate([
        'id' => ['numeric', 'min:1', 'max:500'],
    ], [
        'id' => 'El identificador no es un valor numérico correcto.',
    ]);
    $id = $request->id;
    if (count(Conductanegativa::select('*')->where('id', '=', $id)->get()) == 0) return back()->with('success', 'La conducta no existe');
    $conducta = Conductanegativa::select('*')->where('id', '=', $id)->get()[0];
    Conductanegativa::where('id', '=', $id)->update(['habilitado' => !($conducta->habilitado)]);
    if ($conducta->habilitado) return back()->with('success', 'Conducta Negativa deshabilitada');
    else return back()->with('success', 'Conducta Negativa habilitada');
}
```

Control de errores	Resultado del método
Error en validación	Conducta negativa no cambiada, mensaje en vista de campos incorrectos
Conducta negativa no encontrada con id	Conducta negativa no cambiada, mensaje en vista de que la conducta negativa no existe
Validación correcta y conducta deshabilitada	Conducta negativa habilitada



Validación correcta y conducta habilitada

Conducta negativa deshabilitada

5.3.3 CorreccionesAplicadasController

- Método index, devuelve las correcciones aplicables habilitadas paginadas para mostrarlas, sin margen de error:

```
public function index() {  
    $correcciones = Correccionaplicada::select("*")->where('habilitado','=',true);  
    $tandaCorrecciones = $correcciones->paginate(5);  
    return view('gestion.correcciones', ["correcciones" => $tandaCorrecciones]);  
}
```

- Método deshabilitadas, mismo procedimiento pero devolviendo las deshabilitadas, llevando a la vista correspondiente, sin margen de error:

```
public function deshabilitadas() {  
    $correcciones = Correccionaplicada::select("*")->where('habilitado','=',false);  
    $tandaCorrecciones = $correcciones->paginate(5);  
    return view('gestion.deshabilitados.correcciones', ["correcciones" => $tandaCorrecciones]);  
}
```

- Método crear, valida los datos y crea la corrección:

```
public function crear(Request $request) {  
    $request->validate([  
        'nuevaCorreccion' => ['required', 'string', 'min:3', 'max:80'],  
    ], [  
        'nuevaCorreccion.requires' => 'La descripción de la corrección es obligatoria.',  
        'nuevaCorreccion.string' => 'La descripción de la corrección debe ser una cadena.',  
        'nuevaCorreccion.min' => 'La descripción de la corrección debe tener al menos 3 caracteres.',  
        'nuevaCorreccion.max' => 'La descripción de la corrección debe tener menos de 80 caracteres.',  
    ]);  
    $correccion = Correccionaplicada::create([  
        'descripcion' => $request['nuevaCorreccion'],  
        'habilitado' => true,  
    ]);  
    return back()->with('success', 'Corrección creada con éxito');  
}
```

Control de errores	Resultado del método
Error en validación	Corrección no creada, mensaje en vista de campos incorrectos
Validación correcta	Corrección creada y habilitada por defecto



- Método editar, obtiene el id mediante la ruta de edición y valida los datos para editar una corrección:

```
public function editar(Request $request, $id) {  
    $request->validate([  
        'cambioCorreccion' => ['required', 'string', 'min:3', 'max:80'],  
    ], [  
        'cambioCorreccion.requires' => 'La descripción de la corrección es obligatoria.',  
        'cambioCorreccion.string' => 'La descripción de la corrección debe ser una cadena.',  
        'cambioCorreccion.min' => 'La descripción de la corrección debe tener al menos 3 caracteres.',  
        'cambioCorreccion.max' => 'La descripción de la corrección debe tener menos de 80 caracteres.',  
    ]);  
    $nuevaDescripcion = $request->cambioCorreccion;  
    if ($id > 0 && $id < 1000) {  
        if (Correccionaplicada::find($id) != null) {  
            $correccionEditar = Correccionaplicada::find($id);  
            $correccionEditar->descripcion = $nuevaDescripcion;  
            $correccionEditar->save();  
            return back()->with('success', 'Corrección editada con éxito');  
        }  
    }  
    return back()->with('success', 'Error al editar la corrección');  
}
```

Control de errores	Resultado del método
Error en validación	Corrección no editada, mensaje en vista de campos incorrectos
Id con formato incorrecto	Corrección no editada, mensaje en vista de error al editar
Corrección no encontrada con id	Corrección no editada, mensaje en vista de error al editar
Validación correcta	Corrección editada

- Método eliminar, obtiene el id de la ruta y comprueba que existe y es válido para eliminar la corrección correspondiente:

```
public function eliminar($id) {  
    if ($id > 0 && $id < 1000) {  
        if (Correccionaplicada::find($id) != null) {  
            Correccionaplicada::destroy($id);  
            return redirect('/gestion/correccionesaplicadas')->with('success', 'La corrección se ha eliminado con éxito');  
        }  
        return redirect('/gestion/correccionesaplicadas')->with('success', 'La corrección a eliminar no existe');  
    }  
    return redirect('/gestion/correccionesaplicadas')->with('success', 'Error al eliminar la corrección');  
}
```



Control de errores	Resultado del método
Id con formato incorrecto	Corrección no borrada, mensaje en vista de error al eliminar
Corrección no encontrada con id	Corrección no borrada, mensaje en vista de error al eliminar
Validación correcta	Corrección eliminada

- Método habilitar, recibe el id por Request y lo valida, para cambiar el estado de la corrección:

```
public function habilitar(Request $request) {  
    $request->validate([  
        'id' => ['numeric', 'min:1', 'max:500'],  
    ], [  
        'id' => 'El identificador no es un valor numérico correcto.',  
    ]);  
    $id = $request->id;  
    if (count(Correccionaplicada::select('*')->where('id','=',$id)->get()) == 0) return back()->with('success', 'La corrección no existe');  
    $correccion = Correccionaplicada::select('*')->where('id','=',$id)->get()[0];  
    Correccionaplicada::where('id','=',$id)->update(['habilitado' => !($correccion->habilitado)]);  
    if ($correccion->habilitado) return back()->with('success', 'Corrección deshabilitada');  
    else return back()->with('success', 'Corrección habilitada');  
}
```

Control de errores	Resultado del método
Error en validación	Corrección no cambiada, mensaje en vista de campos incorrectos
Corrección no encontrada con id	Corrección no cambiada, mensaje en vista de que la corrección no existe
Validación correcta y corrección deshabilitada	Corrección habilitada
Validación correcta y corrección habilitada	Corrección deshabilitada



5.3.4 ProfesorAlumnoController

Todas las vistas de este controlador renuevan el token de Sanctum, para poder realizar peticiones a la API. Para ello se utiliza el método renovarSesion, ya que dicho token se guardará en una sesión hasta eliminarse:

```
public function renovarSesion() {
    $usuarioActual = Auth::user();
    $rol = $usuarioActual->getRoleNames()[0];
    if ($rol == "jefatura" && Session::get('TokenApi') == "") {
        $tokenPre = $usuarioActual->currentAccessToken();
        if ($tokenPre) $tokenPre->delete();
        $tokenAuth = $usuarioActual->createToken('ApiToken')->plainTextToken;
        Session::put("TokenApi", $tokenAuth);
    }
}
```

- Método index, si recibe el parámetro de página o page, lleva a la vista de Profesor Alumno activando el Switch, ya que eso indica que se está paginando por la tabla de profesores, valida dicho parámetro y obtiene los datos de los profesores y alumnos para mostrarlos:

```
public function index(AlumnoDataTable $dataTable, Request $request) {
    $request->validate([
        'page' => ['numeric', 'min:1', 'max:100'],
    ], [
        'page' => 'Error de paginación, vuelva a cargar la página. ',
    ]);
    $this->renovarSesion();
    $anoAcademico = AnioAcademico::all();
    $profesores = Profesor::select("*")->where('habilitado','=',true)->orderBy('nombre');
    $tandaProfesores = $profesores->paginate(5);
    $paginaActual = $request->page;
    if ($paginaActual != null) {
        return $dataTable->render('gestion.profesoralumno', ['anoAcademico' => $anoAcademico, "profesores" => $tandaProfesores, "paginaProfesor" => $paginaActual]);
    } else return $dataTable->render('gestion.profesoralumno', ['anoAcademico' => $anoAcademico, "profesores" => $tandaProfesores]);
}
```

Control de errores	Resultado del método
No se recibe página como parámetro	Vista normal mostrando el div de alumno primero
Se recibe page como parámetro, pero tiene error en validación	Se retorna a la vista anterior con un mensaje de error
Se recibe page y es validado correctamente	Vista comienza mostrando el div de profesor



- Método crearProfesor, valida todos los datos necesarios y comprueba que el DNI no existe ya antes de insertarlo en base de datos:

```
public function crearProfesor(Request $request) {  
    $request->validate([  
        'dniProfesor' => ['required', 'string', 'size:9', 'regex:/^0-9){8}[TRWAGMYFPDXBNJZSQVHLCKE]$/i'],  
        'nombreProfesor' => ['required', 'string', 'min:10', 'max:80'],  
        'telefonoProfesor' => ['required', 'numeric', 'between:000000001,999999999'],  
        'emailProfesor' => ['required', 'email:filter'],  
    ]);  
    $dniNuevoProfesor = $request['dniProfesor'];  
    if (!$this->validarDNI($dniNuevoProfesor)) [  
        return back()->with('success', 'DNI con formato incorrecto');  
    ]  
    $existeDNI = Profesor::select('*')->where('dni', '=', $dniNuevoProfesor)->get();  
    if (count($existeDNI) > 0) {  
        return back()->with('success', 'No creado, DNI ya existente');  
    }  
    $profesor = Profesor::create([  
        'dni' => $dniNuevoProfesor,  
        'nombre' => $request['nombreProfesor'],  
        'telefono' => $request['telefonoProfesor'],  
        'correo' => $request['emailProfesor'],  
        'habilitado' => true,  
    ]);  
    $this->renovarSesion();  
    return back()->with('success', 'Profesor creado con éxito');  
}
```

Control de errores	Resultado del método
Error de validación	Se retorna a la vista con mensajes de errores de validación
Error de validación posterior con DNI	Se retorna a la vista anterior con un mensaje de que el DNI es inválido
Si el DNI ya existe en la base de datos	No se crea, y se retorna a la vista indicando que ese DNI ya se ha empleado
Datos validados y DNI inexistente	Se crea el profesor



- Método editarProfesor, valida todos los datos necesarios y comprueba que el DNI no existe ya antes de insertarlo en base de datos, únicamente en caso de que el usuario haya cambiado el DNI al editar:

```

public function editarProfesor(Request $request) {
    $request->validate([
        'editarProfesorDniOriginal' => ['required', 'string', 'size:9', 'regex:/^([0-9]{8}[TRWAGMYFPDXBNJZSQVHLCKE])$/i'],
        'editarProfesorDni' => ['required', 'string', 'size:9', 'regex:/^([0-9]{8}[TRWAGMYFPDXBNJZSQVHLCKE])$/i'],
        'editarProfesorNombre' => ['required', 'string', 'min:6', 'max:80'],
        'editarProfesorTelefono' => ['required', 'numeric', 'between:000000001,999999999'],
        'editarProfesorCorreo' => ['required', 'email:filter'],
    ], [
        'editarProfesorDniOriginal.required' => 'El DNI es obligatorio. ',
        'editarProfesorDniOriginal.string' => 'El DNI debe ser una cadena de texto. ',
        'editarProfesorDniOriginal.size' => 'El DNI debe tener una longitud de 9. ',
        'editarProfesorDniOriginal.regex' => 'El DNI tiene un formato incorrecto. ',
        'editarProfesorDni.required' => 'El DNI es obligatorio. ',
        'editarProfesorDni.string' => 'El DNI debe ser una cadena de texto. ',
        'editarProfesorDni.size' => 'El DNI debe tener una longitud de 9. ',
        'editarProfesorDni.regex' => 'El DNI tiene un formato incorrecto. ',
        'editarProfesorNombre.required' => 'El nombre del profesor es obligatorio. ',
        'editarProfesorNombre.string' => 'El nombre debe ser una cadena de texto. ',
        'editarProfesorNombre.min' => 'El nombre debe tener 10 o más caracteres, se deben incluir apellidos. ',
        'editarProfesorNombre.max' => 'El nombre es demasiado largo, no debe superar los 80 caracteres. ',
        'editarProfesorTelefono.required' => 'El teléfono es obligatorio. ',
        'editarProfesorTelefono.numeric' => 'El teléfono debe ser numérico (9 dígitos). ',
        'editarProfesorTelefono.between' => 'El teléfono debe ser una serie de 9 dígitos consecutivos. ',
        'editarProfesorCorreo.required' => 'El correo es obligatorio. ',
        'editarProfesorCorreo.email' => 'El formato del correo no es correcto. ',
    ]);
    $dniEditar = $request->editarProfesorDni;
    if (!$this->validarDNI($dniEditar)) {
        return back()->with('success', 'DNI con formato incorrecto');
    }
    $dniProfesorOriginal = $request->editarProfesorDniOriginal;
    if (count(Profesor::select('*')->where('dni', '=', $dniProfesorOriginal)->get()) == 0) {
        return back()->with('success', 'Error al obtener DNI, prueba a refrescar la página');
    }
    $profesorEditar = Profesor::select('*')->where('dni', '=', $dniProfesorOriginal)->get()[0];
    if ($dniEditar != $dniProfesorOriginal) {
        $existeDNI = Profesor::select('*')->where('dni', '=', $dniEditar)->get();
        if (count($existeDNI) > 0) {
            return back()->with('success', 'No editado, DNI ya registrado');
        } else {
            $profesorEditar->dni = $dniEditar;
        }
    }
    $profesorEditar->nombre = $request->editarProfesorNombre;
    $profesorEditar->telefono = $request->editarProfesorTelefono;
    $profesorEditar->correo = $request->editarProfesorCorreo;
    $profesorEditar->save();
    $this->renovarSesion();
    return back()->with('success', 'Profesor editado con éxito');
}

```

Control de errores	Resultado del método
Error de validación	Se retorna a la vista con mensajes de errores de validación
Error de validación posterior con DNI	Se retorna a la vista anterior con un mensaje de que el DNI es inválido
No existe profesor con el DNI original	Se retorna a la vista anterior con un mensaje de que el profesor no existe
Profesor existe, con datos validados y sin cambiar el DNI	Se edita el profesor
Caso anterior, pero el DNI ha cambiado y ya existe en la base de datos	No se edita, y se retorna a la vista indicando que ese DNI ya se ha empleado



DNI cambiado pero sin coincidencias en base de datos

Se edita el profesor

- Método eliminarProfesor, obtiene el DNI de la ruta y comprueba que existe y es válido para eliminar al profesor correspondiente:

```
public function eliminarProfesor($dni) {  
    if (!$this->validarDNI($dni)) {  
        return back()->with('success', 'DNI con formato incorrecto');  
    }  
    if (count(Profesor::where('dni', '=', $dni)->get()) == 0) {  
        return back()->with('success', 'Error al eliminar profesor');  
    }  
    Profesor::where('dni', '=', $dni)->delete();  
    $this->renovarSesion();  
    return back()->with('success', 'Profesor eliminado');  
}
```

Control de errores	Resultado del método
DNI con formato incorrecto	Profesor no borrado, mensaje en vista de error al eliminar
Profesor no encontrado con DNI	Profesor no borrado, mensaje en vista de error al eliminar
Validación correcta y profesor existente	Profesor eliminado

- Método reiniciarPuntos, obtiene todos los alumnos con puntos menores a 12 y los reinicia a dicha puntuación. Sin margen de error:

```
public function reiniciarPuntos() {  
    // Update general mediante Eloquent para todos los puntos  
    Alumno::query()->where('puntos', '<', 12)->update([  
        'puntos' => 12,  
    ]);  
    return back()->with('success', 'Los puntos de todos los alumnos inferiores a 12 se han restaurado');  
}
```



- Método habilitar, recibe el DNI por ruta y lo valida, para cambiar el estado del profesor:

```
public function habilitar($dni) {
    if (!$this->validarDNI($dni)) return back()->with('success', 'DNI incorrecto');
    $dniProfesor = $dni;
    if (count(Profesor::select('*')->where('dni', '=', $dniProfesor)->get()) == 0) return back()->with('success', 'El profesor no existe');
    $profesor = Profesor::select('*')->where('dni', '=', $dniProfesor)->get()[0];
    Profesor::where('dni', '=', $dniProfesor)->update(['habilitado' => !($profesor->habilitado)]);
    $this->renovarSesion();
    if ($profesor->habilitado) return back()->with('success', 'Profesor deshabilitado');
    else return back()->with('success', 'Profesor habilitado');
}
```

Control de errores	Resultado del método
Error en validación de DNI	Profesor no cambiado, mensaje en vista de DNI inválido
Profesor no encontrado con DNI	Profesor no cambiado, mensaje en la vista de que el profesor no existe
Validación correcta y profesor deshabilitado	Profesor habilitado
Validación correcta y profesor habilitado	Profesor deshabilitado

5.3.5 ApiController

Los métodos de ApiController están todos relacionados con la API creada y cuyas rutas están protegidas por Sanctum. Son métodos relacionados con el DataTable de alumnos, que se va actualizando en tiempo real gracias a las peticiones de la API. Debido a esto siempre se devuelven datos JSON.



- Método obtenerCorreos, recibe un DNI como parámetros de la petición, lo valida y devuelve la lista de correos asociados a él (Si no hay correos devuelve un JSON vacío):

```
public function obtenerCorreos(Request $request) {
    $validator = Validator::make($request->all(), [
        'dni' => ['required', 'string', 'size:9', 'regex:^[0-9]{8}[TRWAGMYFPDXBNJZSQVHLCKE]$/i'],
    ], [
        'dni.required' => 'El DNI es obligatorio. <br><br>',
        'dni.string' => 'El DNI debe ser una cadena de texto. <br><br>',
        'dni.size' => 'El DNI debe tener una longitud de 9. <br><br>',
        'dni.regex' => 'El DNI tiene un formato incorrecto. <br><br>',
    ]);

    // Se devuelve la información de los errores si ha fallado
    if ($validator->fails()) {
        return response()->json([
            'success' => false,
            'errors' => $validator->errors()
        ], 422);
    }

    $dniAlumno = $request->dni;

    $existeDNI = Alumno::select('*')->where('dni', '=', $dniAlumno)->get();
    if (count($existeDNI) == 0) {
        return response()->json([
            'success' => false,
            'errors' => ["- DNI no registrado. <br><br>"]
        ], 422);
    }

    $listaCorreos = Correo::select("id", "correo", "tipo")->where("alumno_dni", "=", $dniAlumno)->get();
    return $listaCorreos;
}
```

Control de errores	Resultado del método
Error en validación de DNI	Devuelve JSON con errores de validación
DNI no registrado	Devuelve JSON con mensaje de que el DNI no está registrado
Validación correcta y alumno sin correos	JSON vacío sin correos y Success
Validación correcta y alumno con correos	JSON con correos del alumno y Success



- Método crearAlumno, método de API que valida los datos y crea el alumno, después asociando los correos introducidos al DNI tras comprobar que su formato es correcto:

```
public function crearAlumno(Request $request) {
    $validator = Validator::make($request->all(), [
        ...
    ]);

    // Se devuelve la información de los errores si ha fallado
    if ($validator->fails()) {
        return response()->json([
            'success' => false,
            'errors' => $validator->errors()
        ], 422);
    }

    // Asignamos los datos recibidos por Request a variables
    $dniAlumnoCrear = $request->dniCrear;
    if (!$this->validarDNI($dniAlumnoCrear)) {
        return response()->json([
            'success' => false,
            'errors' => ["- DNI con formato incorrecto. <br><br>"]
        ], 422);
    }

    $listaCorreosAnadir = $request->correosAnadirCrear;
    $nombreCrear = $request->nombreCrear;
    $puntosCrear = $request->puntosCrear;
    $idUnidadCrear = $request->idUnidadCrear;
    $alumnoCrear = new Alumno();

    $existeDNI = Alumno::select('*')->where('dni', '=', $dniAlumnoCrear)->get();
    if (count($existeDNI) > 0) {
        return response()->json([
            'success' => false,
            'errors' => ["- No creado, DNI ya registrado. <br><br>"]
        ], 422);
    }
}
```

```
// Tras comprobar el dni asignamos el resto de valores antes de guardar
$alumnoCrear->dni = $dniAlumnoCrear;
$alumnoCrear->nombre = $nombreCrear;
$alumnoCrear->puntos = $puntosCrear;
$alumnoCrear->id_unidad = $idUnidadCrear;
$alumnoCrear->save();

// Validamos los correos enviados antes de añadirlo, formateando el string recibido como array
if ($listaCorreosAnadir != null) {
    for ($i = 0; $i < count($listaCorreosAnadir); $i++) {
        if (!filter_var($listaCorreosAnadir[$i][0], FILTER_VALIDATE_EMAIL)
            && ($listaCorreosAnadir[$i][1] != "personal" || $listaCorreosAnadir[$i][1] != "tutor")) {
            return response()->json([
                'success' => false,
                'errors' => ["- Datos editados, pero algunos de los correos no se han añadido por formato incorrecto. <br><br>"]
            ], 200);
        }
        $nuevoCorreo = new Correo();
        $nuevoCorreo->correo = $listaCorreosAnadir[$i][0];
        $nuevoCorreo->tipo = $listaCorreosAnadir[$i][1];
        $nuevoCorreo->alumno_dni = $dniAlumnoCrear;
        $nuevoCorreo->save();
    }
}

return response()->json([
    'success' => true,
    'user' => $alumnoCrear
], 200);
}
```

Control de errores	Resultado del método
Error en validación de campos	Devuelve JSON con errores de validación
DNI con formato incorrecto	Devuelve JSON con mensaje de que el DNI es inválido



DNI ya existente	Devuelve JSON indicando que el DNI ya está registrado por otro usuario
DNI no existente, pero error de correos	Se crea el alumno y se le asignan los correos hasta llegar al erróneo
Validación correcta y alumno con correos sin errores	JSON con success para actualizar con AJAX

- Método editarAlumno, método de API que valida los datos y edita el alumno, tras esto intenta eliminar los correos indicados, y si no ha ocurrido ningún error asocia los nuevos correos introducidos al DNI tras comprobar que su formato es correcto. Si se establecen los puntos a 0, se notifica mediante correo (Método demasiado extenso para adjuntar imágenes):

```
public function editarAlumno(Request $request) {
    $validator = Validator::make($request->all(), [ ...
    ], [...]);
    // Se devuelve la información de los errores si ha fallado
    if ($validator->fails()) {
        return response()->json([
            'success' => false,
            'errors' => $validator->errors()
        ], 422);
    }
    // Asignamos los datos recibidos por Request a variables
    $dniAlumnoOriginal = $request->dniOriginal;
    $listaCorreosAniadir = $request->correosAniadir;
    $listaCorreosEliminar = json_decode($request->correosEliminar);
    $dniEditar = $request->dniEditar;
    $nombreEditar = $request->nombreEditar;
    $puntosEditar = $request->puntosEditar;
    $alumnoEditar = Alumno::find($dniAlumnoOriginal);
    if (!$this->validarDNI($dniEditar)) {
        return response()->json([
            'success' => false,
            'errors' => [" - DNI con formato incorrecto. <br><br>"]
        ], 422);
    }
}
```



Control de errores	Resultado del método
Error en validación de campos	Devuelve JSON con errores de validación
DNI con formato incorrecto	Devuelve JSON con mensaje de que el DNI es inválido
El DNI original a editar no existe	Devuelve JSON indicando que ha habido un error al obtener el DNI
El DNI original existe pero el nuevo DNI coincide con uno existente	Devuelve JSON indicando que el DNI introducido ya existe y no se realiza el cambio
DNI correcto, pero error en correos a eliminar	Se edita el alumno pero paran de eliminarse los correos en el erróneo
DNI correcto, correos eliminados o no indicados a eliminar, error en correos a añadir	Se edita el alumno, se borran los correos si se han indicado, pero se deja de asociar los nuevos correos al llegar al error
DNI correcto, proceso de eliminación de correos correcto y nuevos añadidos o sin tener que añadir	Alumno editado, si es pertinente con correos eliminados y correos nuevos añadidos
Proceso anterior, pero se han reducido los puntos a 0	Mismo procedimiento, enviando un correo notificando de que el alumno ha llegado a 0 puntos



- Método eliminarAlumno, valida el DNI, comprueba que existe y procede a eliminar todas las referencias si las hubiese, así como todos los partes asociados al alumno, comprueba que si el parte se queda sin alumnos asociados, este también se elimina:

```
public function eliminarAlumno(Request $request) {
    $validator = Validator::make($request->all(), [
        ...
    ]);

    // Se devuelve la información de los errores si ha fallado
    if ($validator->fails()) {
        return response()->json([
            'success' => false,
            'errors' => $validator->errors()
        ], 422);
    }

    $dniAlumno = $request->dniEliminar;

    $existeDNI = Alumno::select('*')->where('dni', '=', $dniAlumno)->get();
    if (count($existeDNI) == 0) {
        return response()->json([
            'success' => false,
            'errors' => ["- DNI no registrado. <br><br>"]
        ], 422);
    }

    // Nos aseguramos de eliminar las referencias antes de eliminar el alumno
    $correos = Correo::where('alumno_dni', '=', $dniAlumno)->delete();
    $partes = AlumnoParte::select('*')->where('alumno_dni', '=', $dniAlumno)->get();
    $partesEliminados = AlumnoParte::where('alumno_dni', '=', $dniAlumno)->delete();
    $alumno = Alumno::where('dni', '=', $dniAlumno)->delete();
    // Debemos asegurarnos de que los partes no sean colectivos para que no se eliminan
    if (count($partes) > 0) {
        foreach ($partes as $parte) {
            // Por cada parte comprobamos que queden relaciones en la tabla intermedia
            $parteId = $parte->parte_id;
            $comprobarNoMasAlumnos = AlumnoParte::select('*')->where('parte_id', '=', $parteId)->get();
            // Si no quedan es que el parte no tiene más alumnos asociados a él, por lo que podemos eliminar
            // todas sus referencias, y a él mismo
            if (count($comprobarNoMasAlumnos) == 0) {
                ParteConductaNegativa::where('parte_id', '=', $parteId)->delete();
                Parte::where('id', '=', $parteId)->delete();
            }
        }
    }
    return response()->json([
        'success' => true,
    ], 200);
}
```

Control de errores	Resultado del método
Error en validación de DNI	Devuelve JSON con errores de validación
DNI con formato incorrecto	Devuelve JSON con mensaje de que el DNI es inválido
El DNI a eliminar no existe	Devuelve JSON indicando que ha habido un error al obtener el DNI
El DNI a eliminar existe	Se elimina el alumno y todas sus referencias así como sus partes si no tienen referencias a otros alumnos



6. Conclusiones generales

6.1 Conclusiones finales

En este apartado se realizarán las últimas conclusiones tras el desarrollo del proyecto, reflexionando sobre los resultados obtenidos, posibles mejoras y ampliaciones, así como una última valoración de parte de los dos integrantes de este desarrollo.

De forma resumida, se han cumplido gran parte de los objetivos planteados en un inicio, y se han ido realizando las mejoras planteadas en cada una de las reuniones intermedias de control del progreso del mismo. Se ha podido desarrollar una aplicación que cumple las necesidades planteadas y que esperamos que sea de utilidad en el uso planteado.

6.2 Posibles ampliaciones y modificaciones

Ambos integrantes del proyecto concordamos en que podrían ampliarse y modificar diversos aspectos del proyecto, para perfeccionar aún más el resultado final.

Algunas de estas cuestiones son funcionalidades que no se han añadido principalmente por falta de tiempo, al requerir tanto investigación como prueba y error hasta conseguir el resultado deseado. Una de estas ampliaciones sería la implementación de mensajería, que a pesar de que la necesidad que alivia está cubierta principalmente por el envío de correos, fue una funcionalidad planteada y que hubiese sido interesante de implementar.

Otra ampliación sería desarrollar una guía de estilos independiente de esta documentación, ya que, a pesar de haberse tratado bastantes aspectos del diseño en este documento, dividir dicho contenido sería muy interesante de cara a modificaciones futuras.

Un ejemplo de una modificación a realizar, podría ser una mejora interesante poder añadir y modificar los cursos y las unidades, aunque de cara a nuestro planteamiento, esta información se exportará en un Excel, ya que tanto los años académicos, como los cursos como sus unidades no deberían variar durante un propio año escolar.



6.3 Valoraciones personales

- Valoración de Sergio Guerrero Borrero:

Primero de todo decir que ha sido un gran honor que se nos haya confiado este proyecto como un encargo directo, de cara a ser empleado en un futuro en el centro.

Tanto Alejandro como yo hemos aprendido mucho desarrollando este proyecto en conjunto durante nuestras FCT, y hemos podido aplicar mucho de lo que hemos aprendido en cada empresa, reforzando los conocimientos adquiridos y poniendo en práctica nuestras habilidades.

Hemos afianzado aún más prácticamente todos los temas que hemos tratado a lo largo del curso. En cuanto a Diseño de Interfaces Web hemos reforzado los conocimientos en Bootstrap, CSS y otras cuestiones generales como la

Considero que el resultado es bastante sólido y estoy bastante satisfecho, me habría gustado mejorarla incluso más añadiendo las funciones anteriormente explicadas de no ser por el tiempo disponible.

- Valoración de Alejandro Cabot Caparrós:

Durante el curso del desarrollo del proyecto hemos tenido que aprender a organizarnos y a trabajar en equipo asignándonos tareas a cada uno y buscando soluciones para cada problema que se nos proponía.

Ha sido una tarea ardua analizar lo que nos proponía el profesorado e intentar adaptarlo al código, pero utilizando herramientas que hemos aprendido durante el curso y las prácticas, hemos ido solucionándolo.

Como conclusión ha sido un reto pero hemos aprendido bastante a acerca de herramientas bastante útiles que nos servirán a futuro para otros proyectos.



Bibliografía

Convenciones de API Rest:

[Qué es una API REST | Neoguias](#)

<https://blog.hubspot.es/website/que-es-api-rest>

Laravel validación API:

[The Best Way to Use Request Validation in Laravel REST API \(avenuecode.com\)](#)

Symfony 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=ygURbWFhdHdIYnNpdGUvZXhjZWw%3D>

D

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-datable-text>

Solucionar problema de ordenación de números como string:

<https://datatables.net/forums/discussion/61724/i-have-to-order-a-table-but-the-numbers-are-coming-in-string-and-not-int>

Bootstrap:

<https://getbootstrap.com/>

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

Bootstrap Toggle:

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

Documentación del curso