# Introducción

## Objetivos

A través de esta memoria se tiene como propósito mostrar de una forma clara y más concisa la estructura del proyecto de fin de curso para el módulo de Desarrollo de Aplicaciones Web.

La página tratará sobre la gestión de autoescuelas, que servirá como un punto para centralizar la información necesaria para llevar el negocio, desde los alumnos que están suscritos al servicio, a los profesores que allí trabajan y los coches que se utilizan, pudiendo visualizar información necesaria y modificable sobre cada uno de ellos y estableciendo relaciones entre estos.

El objetivo final es crear una aplicación funcional que permita gestionar las autoescuelas con comodidad y facilidad, con interfaces intuitivas y agradables para la vista y que permita guardar información de una forma más segura y precisa.

## Contexto

Hace algo menos de dos años entré a una autoescuela en mi villa para sacarme el carnet de conducir de tipo B. Estuve todo el verano y varios meses más hasta mayo del siguiente año en aquella autoescuela entre los estudios para el examen teórico y el examen práctico.

La gestión del examen teórico estaba bien, principalmente porque era casi todo online con aplicaciones que estaban a disposición de los conductores de España para realizar test y aprender de forma progresiva, pero una vez que empecé a dar las clases prácticas ya entré en conocimiento de cómo se gestionaban las cosas en una autoescuela.

Por lo general, las operaciones se escribían en papel, tanto para las clases de los alumnos, como las notas, información sobre los coches… etc.

No era extraño ver a mi profesor recibiendo llamadas o llamando a su secretaria para informarse de la clase que tenía después, o de si un alumno estaba disponible para cualquier cosa, incluso preguntando sobre revisiones de los coches.; o a su secretaria tardando algo más de lo usual al buscar las clases que tenían programadas los alumnos de la autoescuela para encontrar huecos. En ese momento no veía nada mejorable.

Una vez que en el segundo curso del módulo se nos propuso crear un proyecto, al pensar en una buena idea se me vino a la cabeza todo aquello y de esta forma propuse crear una aplicación que mejorase todas esas situaciones para que todo se realice de forma rápida, y de esta forma el profesor puede estar más centrado en sus clases y la información quedará mejor guardada.

## Planteamiento del proyecto

El proyecto se divide en dos partes perfectamente distinguibles, el front-end, lo que el usuario ve; y el back-end, lo que el usuario ignora.

### Front-end

Para la parte de front-end del proyecto, se ha empleado HTML, Thymeleaf, CSS, JavaScript, jQuery y Bootstrap, con el objetivo de crear interfaces de usuario bonitas, intuitivas y eficientes, mediante las cuales se pueda informar al usuario de lo que realiza y los errores que puedan ocurrir, además por supuesto de mostrarle toda la información, que es la principal función de la aplicación.

### Back-end

En el back-end nos encontramos con MySQL como sistema gestor de Bases de Datos en la que almacenar la información que será mostrada por el front-end, Java como lenguaje de programación para construir todos los métodos, clases, controladores… en el IDE Spring Tool Suite.

También JavaScript, que se puede utilizar tanto en front-end como en back-end, para el manejo de algunas solicitudes en el lado del servidor.

Por último, Docker, para permitir la mayor portabilidad posible del programa a casi cualquier equipo informático.

# Requisitos de la aplicación

## Introducción

Este apartado constará de una descripción detallada del funcionamiento del sistema, con requisitos que especifican qué debe hacer la aplicación en determinadas situaciones.

## Requisitos

### Requisitos funcionales

La aplicación tendrá tres tipos de usuarios, distinguidos entre sí por la asignación de roles a cada uno de ellos:

Administrador

Obtiene sus privilegios del rol “ROLE\_ADMIN”. Es el usuario con mayores capacidades del sistema, pudiendo ejecutar absolutamente todas las funciones disponibles en él. Es capaz de borrar, editar, insertar y ver todas las entidades, desde alumnos y coches, hasta los propios usuarios. Este usuario está pensado para que sea el dueño o la persona que gestione la autoescuela.

Tiene un panel exclusivo para su uso como una forma de centralizar las operaciones de todas las entidades en una sola página, para facilitar la gestión de su autoescuela, también puede crear usuarios y asignarles roles, y ver todas las clases en el calendario, por lo que su capacidad es total.

Profesor

Distinguido de los demás por el rol “ROLE\_TEACHER”, está en el segundo lugar en la jerarquía del sistema. No tendrá ninguna capacidad de creación, eliminación y edición de registros, no obstante, podrá verlos para informarse sobre los alumnos a los que imparte clase y los vehículos que utiliza.

El profesor estará obligatoriamente vinculado a uno de los profesores presentes en la base de datos mediante los identificadores.

En el calendario, la visualización de las clases se limitará a aquellas en las que el profesor al que el usuario está vinculado participe, pudiendo ver la ahora y el alumno al que imparte clase.

También tendrá una ventana exclusiva para él llamada “Mis alumnos”, en la que aparecerán solo los alumnos a los que imparta clase, de esta manera podrá acceder a la información que necesite con rapidez, especialmente útil al añadir notas al alumno al que esté impartiendo.

Alumno

El tercero en la jerarquía del sistema posee el rol “ROLE\_USER”. Es el usuario registrado con mayores limitaciones de la aplicación. Al igual que el profesor, no podrá crear, eliminar ni editar registros, ni tampoco podrá ver la gran mayoría de ellos, ya que la información estará limitada sólo a aquella con la que él se encuentre relacionado, como las clases, o su perfil.

De esta forma, tendrá acceso al calendario para poder ver sus clases, pero no las de los demás ni tampoco podrá editarla.

Usuario no registrado

No tiene ningún privilegio y solo podrá ver la pantalla de inicio. Tendrá la opción de registrarse e iniciar sesión solamente como un usuario normal.

### Requisitos no funcionales

Para mejorar la carga de datos, se implementarán DTOs donde sea posible, evitando así la sobrecarga de información innecesaria. Esta optimización no solo incrementará la eficiencia, velocidad y rendimiento del sistema, sino que también mejorará la experiencia del usuario al acceder a los datos de una forma más rápida.

Se realizará una carga inicial de datos, que incluirá profesores, alumnos, coches, clases, usuarios y roles para el correcto funcionamiento del sistema sin inicializar. Esta se producirá al visitar la sección de inicio.

La aplicación podrá ser ejecutada en cualquier ordenador, teniendo como únicos requisitos indispensables la conexión a Internet y un navegador desde el que acceder a ella, si bien el comportamiento podría variar en los navegadores menos utilizados.

### Requisitos de Interfaces Externas

#### Interfaces de los usuarios

Los usuarios compartirán las mismas interfaces, la única diferencia que presentarán será las limitaciones en la visualización, por ejemplo, el administrador tendrá un panel exclusivo o el usuario no registrado no podrá ver ningún perfil o calendario.

El header de la aplicación será un elemento común para todos los usuarios, desde el administrador hasta el usuario no registrado.

#### Interfaces hardware

Para el correcto uso de la aplicación se necesitarán ciertos elementos hardware. Para introducir la información será necesario contar con un teclado y para navegación y visualización de datos, un ratón y una pantalla respectivamente.

#### Interfaces software

Como ya fue mencionado en el punto [2.2.2](#_Requisitos_no_funcionales), a pesar de que la aplicación se ejecute de forma local para su presentación, será necesario contar con un navegador y una conexión a Internet para acceder a la aplicación cuando se despliegue de forma pública.

#### Interfaces de comunicaciones

En caso de ser desplegada de forma pública, la aplicación utilizará el protocolo HTTPS para mantener la seguridad de los datos junto con una conexión TCP/IP.

### Requisitos de rendimiento

Para garantizar un buen rendimiento en la aplicación, se procurará utilizar un código limpio y sencillo, que no realice instrucciones innecesarias, además de realizar la carga solo de los datos que sean necesarios.

La base de datos debe estar optimizada para poder responder con rapidez, para lo cual se han empleado prácticas tales como guardar en ellas las rutas de las imágenes, y que las propias imágenes se encuentren en el servidor, de esta forma la carga se realiza más velozmente.

Las imágenes también cuentan con limitaciones de tamaño, como una medida adicional para optimizar el rendimiento.

### Atributos

#### Seguridad

Para mantener la seguridad de los datos, es esencial mantener la jerarquía en el sistema, y limitar las acciones de acuerdo con ella, además de asegurarse de que los datos se entregan de la forma correcta, para que no surjan errores a la hora de manipularlos.

Para el primer caso, se realizan comprobaciones a nivel de código sobre la autoridad de un usuario al realizar una determinada acción, una vez comprobada, dependiendo de si se cumple o no esa autoridad, ejecuta una u otra acción. Además, se incluye una clase de Seguridad que limita el acceso a determinadas URL en función de los roles que tenga el usuario autenticado actual.

En el segundo caso, se realizan validaciones en el lado de cliente, como los vistos en Desarrollo Web Entorno Cliente, mediante el uso de clases de validación de Bootstrap y JavaScript, con los cuales no se permitirá enviar información que no cumpla el formato establecido.

En el lado del servidor, visto en Desarrollo Web Entorno Servidor, se realizan también validaciones de formatos, conversiones (véase la matrícula o el DNI, se pueden introducir con letras minúsculas, pero serán convertidas a mayúsculas) y comprobaciones de campos vacíos.

De esta manera, se facilita la interacción al usuario, a la vez que se evitan una gran cantidad de errores poco importantes que impedirían al usuario realizar las acciones que necesita.

#### Facilidades de mantenimiento

Para asegurar las correctas relaciones entre datos y su persistencia, se ha creado un panel para el administrador, donde se centralizan casi todas las operaciones en un solo lugar, facilitándole la gestión y actualización de todos los datos.

El panel del administrador es una parte importante del sistema al proporcionar una interfaz intuitiva y de fácil uso, diseñado específicamente para satisfacer las necesidades diarias del administrador. Desde este punto de control centralizado, el administrador puede realizar diversas tareas, como ingresar, visualizar y eliminar datos.

Sin embargo, carece de la posibilidad de editarlos, esto es así porque esta función se encuentra en los apartados de alumnos, coches, profesores… etc. Por lo que si el administrador quisiera editar una de estas entidades, muy probablemente acudirá a su sección para buscar su identificador y en esa misma sección se encuentra la función de edición, liberando carga de elementos visuales en el panel del administrador.

#### Portabilidad

La aplicación será ejecutable en cualquier ordenador que cumpla con los requisitos no funcionales especificados en el apartado [2.2.2](#_Requisitos_no_funcionales). Sin embargo, la portabilidad entre estos dispositivos sería tediosa de hacerlo simplemente transfiriendo los archivos, ya que de esta manera habría que cambiar muchas configuraciones de la comunicación entre la base de datos y la aplicación puesto que esta configuración puede variar mucho entre ordenadores, o incluso puede no existir.

Para solucionar esto está Docker, mediante el cual podremos crear dos contenedores, uno para la aplicación y otro para la base de datos, utilizando Docker Compose para orquestar estos servicios.

Así, el único requisito para que la aplicación funcione en cualquier ordenador será contar con la presencia de Docker y Docker Compose en el ordenador anfitrión.

# Análisis

## Introducción

En el apartado de análisis se presentarán distintos diagramas que muestren el funcionamiento y el flujo de información del sistema desde diferentes perspectivas

## Diagrama de clases

En el siguiente diagrama de clases se mostrarán las relaciones entre las clases más esenciales, es decir, las clases modelo, para una mejor comprensión de cómo se relacionan unas con otras:

Diagrama

Descripción generada automáticamente

## Diagrama Descripción generada automáticamenteDiagrama de Casos de Uso

En este apartado se presenta un diagrama de casos de uso, mostrando las principales situaciones que se van a producir a la hora de utilizar la aplicación.

Para el usuario no registrado:

Diagrama

Descripción generada automáticamente

Para el usuario alumno:

Diagrama

Descripción generada automáticamente

Para el usuario profesor:

Diagrama

Descripción generada automáticamente

Para el usuario administrador:

Diagrama

Descripción generada automáticamente

# DISEÑO

## Introducción

El diseño de la aplicación está formado por tres capas: la capa de presentación, la capa de negocio y la capa de persistencia de datos. Cada una de ellas está diferenciada de las demás, asumiendo sus propias responsabilidades y se coordinan para dar forma a todo el flujo de datos.

Mantener las tres capas independientes entre sí ayuda a realizar un correcto mantenimiento, a la escalabilidad de la aplicación y a la reutilización de código.

## Capa de presentación

La capa de presentación es aquella que el usuario puede visualizar, formada por las interfaces y páginas mediante las que interactúa con el sistema.

Esta capa está formada por los archivos HTML, CSS y en parte, JavaScript para algunas tareas de generación de contenido dinámico.

JavaScript se emplea, por ejemplo, en la página inicial, mandando una solicitud GET a la URL de una API, la cual devuelve resultados que se encarga de convertir en elementos del DOM. También para mostrar mensajes de alerta o incluso configurar todo el calendario y su envío de solicitudes al controlador.

En el HTML se utiliza, además de JavaScript, la sintaxis de Thymeleaf, que permite que el código realice ciertas acciones o no dependiendo de lo que se evalúa, tales como la creación de determinados elementos por cada elemento de una lista, mostrar u ocultar elementos según el valor de una variable, o el almacenamiento de componentes de código para reutilizarlos fácilmente.

## Capa de negocio

La capa de negocio es la capa que el usuario no ve, es decir, el código Java en este caso.

Contiene todas las clases de la aplicación, tanto las de modelo como las de los controladores, pasando por los DTO, los servicios… etc. Estas clases están formadas por una serie de atributos y métodos, coordinándose unas con otras y estableciendo relaciones entre sí.

Se encuentra entre la capa de presentación y la capa de persistencia, interactuando con ambas y ejerciendo de mediador.

Con la capa de presentación se comunica recibiendo la información que se le envía, ya sea a través de formularios o de las URL. También se encarga de enviarle información a través de los modelAttributes, los cuales serán utilizados por la sintaxis de Thymeleaf; o de las imágenes, las cuales se envían a través de controladores.

También interactúa con la capa de persistencia, entregándole instrucciones para insertar, borrar, editar o recuperar registros y convirtiéndolos en datos interpretables por la capa de presentación cuando es necesario.

## Capa de persistencia

Constituida por la base de Datos, almacena toda la información de la aplicación, usuarios, coches, alumnos, clases… Esto lo consigue a través de un ORM, Hibernate, el cual mapea los objetos de Java en tablas SQL, ahorrando el tener que ejecutar las consultas SQL manualmente en la gran mayoría de los casos.