

Proyecto Final



**Tutor proyecto: Joan Gerard Camarena**

**Tutor grupo: Jose Alfredo Murcia**

**Valentin Pankov Fichev**

**09/04/2022**

Proyecto Final

Contenido

[Datos del alumno 5](#_Toc105534047)

[Datos del proyecto 5](#_Toc105534048)

[1. Presentación 6](#_Toc105534049)

[2. Contenido 6](#_Toc105534050)

[3. Análisis/herramientas 7](#_Toc105534051)

[3.1 Base de datos 7](#_Toc105534052)

[3.2 Backend 7](#_Toc105534053)

[3.3 Frontend 9](#_Toc105534054)

[4.01 Desarrollo - Back 11](#_Toc105534055)

[4.1 Estructura general 11](#_Toc105534056)

[4.2 Model 13](#_Toc105534057)

[4.2.1 Alumno 14](#_Toc105534058)

[4.2.2 Asignatura 15](#_Toc105534059)

[4.2.3 Curso 15](#_Toc105534060)

[4.2.4 Examen 16](#_Toc105534061)

[4.2.5 Modulo 16](#_Toc105534062)

[4.2.6 NotaExamen 17](#_Toc105534063)

[4.2.7 NotaExamenPK 17](#_Toc105534064)

[4.2.8 Profesor 18](#_Toc105534065)

[4.2.9 ProfesorAsignatura 18](#_Toc105534066)

[4.2.10 ProfesorAsignaturaPK 19](#_Toc105534067)

[4.2.11 Rol 19](#_Toc105534068)

[4.2.12 Usuario 20](#_Toc105534069)

[4.3 Controller 21](#_Toc105534070)

[4.3.1 AlumnoController 21](#_Toc105534071)

[4.3.2 AsignaturaController 22](#_Toc105534072)

[4.3.3 CursoController 22](#_Toc105534073)

[4.3.4 ExamenController 22](#_Toc105534074)

[4.3.5 ProfesorController 23](#_Toc105534075)

[4.3.6 RolController 23](#_Toc105534076)

[4.3.7 UsuarioController 23](#_Toc105534077)

[4.4 Service 24](#_Toc105534078)

[4.5 Repository 25](#_Toc105534079)

[4.5.1 AlumnoRepository 25](#_Toc105534080)

[4.5.2 AsignaturaRepositoy 26](#_Toc105534081)

[4.5.3 CursoRepository 26](#_Toc105534082)

[4.5.4 ExamenRepository 26](#_Toc105534083)

[4.5.5 ModuloRepository 26](#_Toc105534084)

[4.5.6 ProfesorAsignaturaRepository 26](#_Toc105534085)

[4.5.7 ProfesorRepository 27](#_Toc105534086)

[4.5.8 RolRepository 27](#_Toc105534087)

[4.5.9 UsuarioRepository 27](#_Toc105534088)

[4.6 Configuration 28](#_Toc105534089)

[4.7 Security 29](#_Toc105534090)

[4.7.1 AuthenticationRestController 29](#_Toc105534091)

[4.7.2 JwtAuthenticationEntryPoint 30](#_Toc105534092)

[4.7.3 JwtToken 30](#_Toc105534093)

[4.7.4 JwtTokenUtil 31](#_Toc105534094)

[4.7.5 JwtTokenFilter 32](#_Toc105534095)

[4.7.6 SecurityConfiguration 33](#_Toc105534096)

[4.7.7 UsuarioCredencials 35](#_Toc105534097)

[4.7.8 UsuarioDetails 35](#_Toc105534098)

[4.7.9 UsuarioDetailsService 36](#_Toc105534099)

[4.8 Application.properties 37](#_Toc105534100)

[4.02 Desarrollo – Base de datos 38](#_Toc105534101)

[4.03 Desarrollo – Front 41](#_Toc105534102)

[4.1 Login 44](#_Toc105534103)

[4.2 Dashboard 47](#_Toc105534104)

[4.3 Curso 50](#_Toc105534105)

[4.4 Alumno 52](#_Toc105534106)

# Datos del alumno

|  |  |
| --- | --- |
| **Nom i cognoms** | Valentin Pankov Fichev |
| **NIF/NIE** | 18508056W |
| **Curs i CF** | 2n DAM-SP |

# Datos del proyecto

|  |  |
| --- | --- |
| **Títol del projecte** |  |
| **Nom del tutor individual** | Joan Gerard Camarena Estruch |
| **Nom del tutor del grup** | Jose Alfredo Murcia Andrés |
| **Resum** | Aplicación web para visualizar de forma clara los datos principales de un alumno y sus notas. Backend hecho en Java (Spring Boot) y el frontend con Vue. |
| **Abstract** |  |
| **Mòduls implicats** | 1. Acceso a datos 2. Desarrollo de interfaces 3. Programación 4. Base de datos |
| **Data de presentació** | Viernes 10 de Junio de 2022 |

# Presentación

La idea de este proyecto es visualizar y acceder de una forma simple y fácil a los datos de un alumno y a sus calificaciones. Todo esto con el objetivo de poder llevar un seguimiento fácil de cada alumno y de un vistazo poder ver en que asignaturas puede ir mas flojo, en cuales más preparado.

El **backend** se ha hecho en **Java**, ya que es el lenguaje con el que hemos trabajado principalmente en el superior y en mi caso, recibí una formación en la empresa de practicas de **Spring boot**, y es por eso que decidí utilizar este framework.

En la parte de **frontend** he utilizado **Vue**, ya que consideré que era un buen framework para empezar a experimentar en el mundo del front, ya que lo máximo que he hecho ha sido html5 y css, por lo cual ha sido un gran reto hacer esto.

En cuanto a la **base de datos**, he utilizado JPA para autogenerar la base de datos en mysql, ya que me ha parecido una herramienta super útil que te ahorra muchísima faena.

# Contenido

Como resumen, tenemos:

* Base de datos: Mysql generada con JPA desde el backend
* Backend: Java (Spring boot)
* Frontend: Vue

# Análisis/herramientas

En este apartado quiero entrar un poco más en detalle con todas las herramientas que se han utilizado y como se ha estructurado todo, antes de pasar al siguiente apartado de desarrollo que ahí ya se veremos el código con muchísimo más detalle.

## Base de datos

Vamos a empezar primero con la base de datos.

La idea de la base de datos es, por una parte, que tenga una tabla módulos, donde vamos a registrar los diferentes módulos de FP, DAM, DAW, por ejemplo, y la ESO

Cada uno de estos módulos va a tener “cursos” o años, los cuales van a contener asignaturas, por ejemplo, el curso 1 de DAM tiene, base de datos, programación… ponemos siempre la foreign key en la parte de los “muchos”, en estos casos la relación es uno a muchos.

Luego, tenemos una relación ternaria para decir que profesor imparte qué asignaturas y a que curso pertenecen estas. Tenemos como clave primaria la asignatura y el curso, que al mismo tiempo son foreign keys, para poder asignar varias asignaturas a un curso y que una asignatura pueda pertenecer a diferentes cursos, como en el caso de DAM y DAW, el primer año. Faltaría solamente hacer una constraint para indicar que cuando curso vale “x” y asignatura “y” solo pueda haber un profesor dando clase exactamente esa asignatura de ese curso.

Cada asignatura va a tener unos exámenes que la tabla simplemente va a ser una “convocatoria”, vamos a registrar el examen en si. Por otra parte tenemos a los alumnos que tienen una relación muchos a muchos con exámenes, por eso surge la tabla del medio que son las notas. Clave primaria va a ser alumno y examen, de esta manera podemos tener a muchos alumnos para el mismo examen y muchos exámenes para el mismo alumno y luego a parte la nota.

Para acabar tenemos la relación de profesor con usuario que es de uno a uno, lo que significa que un profesor puede tener un usuario y este solo puede pertenecer a un profesor y este mismo usuario puede tener un rol.

## Backend

Como hemos mencionado anteriormente, el backend está hecho con Spring, que ha sido muy cómodo y nos ha facilitado muchísimo la faena. Hemos utilizado la librería que tiene Spring, Lombok, que simplemente con anotar una clase con las etiquetas @AllArgsConstructor, @NoArgsConstructor y @Data, nos crea el constructor con todos los atributos, un constructor vacío y todos los setter y getters, pero estos no se ven en el código. El código se ve muchísimo mas limpio y como justo la parte de los setter y getters es siempre la misma, con saber cuales son los atributos de la clase, ya sabemos que cada uno de ellos tiene get y set.

Texto

Descripción generada automáticamente

Otra librería muy útil es la de seguridad de Spring y la de jsonwebtoken que nos ha permitido llevar todo el tema de seguridad a través de un usuario y contraseña y a parte de un JWT, que mas adelante veremos con detalle.

## Frontend

En la parte del frontend tenemos Vue, que tiene una parte de vistas, que es donde crearemos cada una de nuestras pantallas, que estas se componente por una parte HTML, otra que es JavaScript y otra, opcional, que es el CSS, por si queremos dar algún diseño.

Texto

Descripción generada automáticamente

Por otra está el router que es el encargado de gestionar todo el tema de rutas dentro de lo que es nuestro front. Asigna un componente a la ruta que le indiquemos, entonces nosotros al pulsar un botón, le indicamos que queremos que nos redireccione a X ruta y de esta manera el router sabe que componente/vista cargar.

Texto

Descripción generada automáticamente

# 4. Desarrollo - Back

## 4.1 Estructura general

Vamos a empezar viendo el backend y a la par la base de datos.

Texto

Descripción generada automáticamente

Esta es la estructura principal del proyecto.

* Primero tenemos la carpeta configuración, que es donde esta el fichero de configuración web, que habilitamos el CORS, para poder realizar la comunicación entre el back y el front, que por defecto viene bloqueado.
* A continuación están los controller, que es donde definimos el endpoints, las consultas a nuestra API REST.
* Luego, los modelos, que son las clases principales de nuestro proyecto y al mismo tiempo las entidades que se crean en la base de datos.
* Los repositorios, que es desde donde accedemos a la base de datos
* La carpeta de seguridad, que es donde esta todo lo relacionado con la seguridad y la configuración del JWT.
* Por último los servicios, que es la parte lógica de nuestra API REST.

Diagrama

Descripción generada automáticamente

En esta imagen podemos ver de forma muy simple y resumida como esta estructurada una API REST.

En el primer bloque tenemos el controller, al que se le pueden hacer peticiones GET, POST, PUT, DELETE. La primera para obtener datos de la base de datos, la segunda para insertar datos, actualizar datos y borrar datos, respectivamente.

El segundo bloque que vemos es el service, es donde tendremos toda la lógica de nuestra aplicación. Si antes de devolver un dato, tenemos que comprobar que cumple X requisitos, es en esta capa donde lo haremos.

Por último tenemos el repositorio, que es la capa que comunica directamente con la base de datos.

El orden siempre es el mismo. El usuario hace una petición al controller, este llama al servicio, el servicio llama al repositorio, el repositorio pide los datos a la base de datos y estos datos siguen el mismo camino que el de ida, de base de datos pasa a repositorio, a servicio, controller y cliente respectivamente.

## Model

Vamos a empezar por lo modelos

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Intentaré ser lo mas breve posible ya que es algo repetitivo.

### Alumno

Texto

Descripción generada automáticamente

Tenemos al alumno con los siguientes atributos y abajo del todo vemos que tiene un curso y como se marca la relación que tiene con dicho curso. Es una relación de muchos a uno, ya que un curso tiene muchos alumnos y un alumno pertenece solo a un curso al mismo tiempo. Al estar en la parte de los muchos, que son los alumnos, se pone el @ManyToOne. El JoinColumn indica la columna en la base de datos. Por último tenemos la anotación JsonManagedReference que lo que evita es una recursión infinita al tener una relación bidireccional entre clases. Veremos mas adelante que las relaciones marcadas con JsonManagedReference son las que se serializan y aparecen en la base de datos, las otras que van con la anotación JsonBackReference, aparecen en la clase en java pero no aparecerán en la base de datos ya que no son serializables.

### Asignatura

Texto

Descripción generada automáticamente

Por aquí tenemos asignatura, que tiene una relación @OneToMany con Exámenes ya que una asignatura tiene muchos exámenes, pero un examen en concreto solo puede pertenecer a una asignatura. El resto de anotaciones hasta ahora ya las hemos visto.

### Curso

Texto

Descripción generada automáticamente

### Examen

Texto

Descripción generada automáticamente

### Modulo

Texto

Descripción generada automáticamente

### NotaExamen

Texto

Descripción generada automáticamente

Aquí podemos ver que tenemos una EmbeddedId que es una clave compuesta, que se indica en la clase que veremos a continuación.

### NotaExamenPK

Texto

Descripción generada automáticamente

Aquí es donde definimos que dos atributos van a ser las ID’s de la clase anterior, NotaExamen e indicamos el nombre de la columna que van a tener en la base de datos.

### Profesor

Texto

Descripción generada automáticamente

Aquí lo único que tenemos de diferente es la anotación Exclude que la tuve que poner porque daba un error al crear el hashCode con el lombok automáticamente y entraba en bucle, entonces tuve que excluirlo en una de las dos clases, de Profesor o Usuario.

### ProfesorAsignatura

Texto

Descripción generada automáticamente

Esta es la tabla que se genera de la relación ternaria que hay entre profesor curso y asignatura, en la que tenemos una clave compuesta entre asignatura y curso

### ProfesorAsignaturaPK

Texto

Descripción generada automáticamente

### Rol

Texto

Descripción generada automáticamente

### Usuario

Texto

Descripción generada automáticamente

## Controller

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

### AlumnoController

Texto

Descripción generada automáticamente

Aquí tenemos los endpoints de la API REST relacionados con Alumno

Primero indicamos que después de localhost:8080 tenemos /api y a continuación va el la petición que queremos hacer, por ejemplo el GET de alumno, que seria localhost:8080/api/alumno?idAlumno=1, el ?idAlumno indica que es un parámetro, que es requerido para esta petición.

Al principio podemos ver el alumnoService que es una interfaz la cual se inyecta, que es lo que indica el @Autowired, con lo cual “une” al controller con el service, de esta manera podemos llamarlo desde aquí.

### AsignaturaController

Texto

Descripción generada automáticamente

### CursoController

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

### ExamenController

Texto

Descripción generada automáticamente

### ProfesorController

Texto

Descripción generada automáticamente

### RolController

Texto

Descripción generada automáticamente

### UsuarioController

Texto

Descripción generada automáticamente

## Service

Texto

Descripción generada automáticamente con confianza baja

Por una parte tenemos la interfaz

Texto

Descripción generada automáticamente

Y por otra, el servicio con la interfaz implementada donde llamamos al repositorio y hacemos la consulta.

Texto

Descripción generada automáticamente

No voy a poner la captura de todos los servicios ya que es bastante repetitivo y se puede ver en la primera captura de este apartado, son muchas interfaces y clases. No hay nada relevante respecto a esta captura.

## Repository

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

### 4.5.1 AlumnoRepository

Texto

Descripción generada automáticamente

Tenemos el repositorio de Alumno que extiende de CRUD, que viene de Create, Read, Update y Delete que hace referencia a la capa de persistencia. Esto permite que podamos guardar en nuestra base de datos objetos, pasar un objeto a datos mediante la serialización y viceversa. En el CRUD tenemos las consultas básica a la base de datos, en caso de querer hacer algo diferente a lo que este no proporciona tenemos que hacerlo una custom query como en el caso de alumno. Como se puede ver no es una query común de mysql sino que utiliza los objetos.

En este caso seleccionamos los alumnos con un curso en especifico

### 4.5.2 AsignaturaRepositoy

Texto

Descripción generada automáticamente

Se seleccionan las asignaturas de un curso

### 4.5.3 CursoRepository

Texto

Descripción generada automáticamente

Seleccionamos los cursos que pertenecen a un modulo

### 4.5.4 ExamenRepository

Texto

Descripción generada automáticamente

Seleccionamos los exámenes de una asignatura

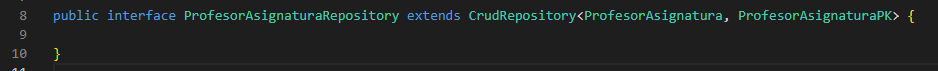
### 4.5.5 ModuloRepository

Imagen que contiene Texto

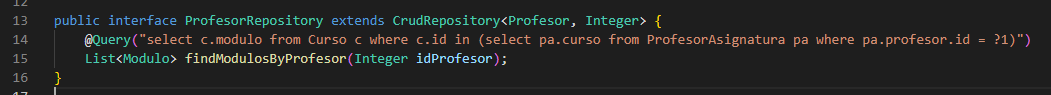
Descripción generada automáticamente

En este caso con el CRUD tenemos suficiente por lo cual, no tenemos que implementar ninguna búsqueda adicional

### 4.5.6 ProfesorAsignaturaRepository



### 4.5.7 ProfesorRepository



Aquí seleccionamos los módulos a los que imparte un profesor, por lo que tenemos que hacer dos selects ya que no tenemos toda la información en una única tabla

### 4.5.8 RolRepository

Texto

Descripción generada automáticamente

### 4.5.9 UsuarioRepository

Texto

Descripción generada automáticamente

## Configuration

Texto

Descripción generada automáticamente

Esto es la configuración Web para habilitar el CORS, que es el intercambio de recursos de origen cruzado, que lo que viene a decir es que restringe las solicitudes HTTP desde un dominio diferente, como es en nuestro caso, peticiones desde el front al back y viene deshabilitado por defecto. Aquí habilitamos el CORS y los métodos que se ven a continuación, de cualquier origen.

## Security

Texto

Descripción generada automáticamente

### AuthenticationRestController

Texto

Descripción generada automáticamente

Esto es el endpoint al que enviamos el usuario y contraseña, verifica si estos son correctos y le asigna al usuario un token.

### JwtAuthenticationEntryPoint

Texto

Descripción generada automáticamente

Es es la clase que se encarga de verificar si el usuario tiene token y sino devuelve un 401 unauthorized.

### JwtToken

Texto

Descripción generada automáticamente

Esta es la clase que vamos a utilizar para almacenar el token que nos asignen y el usuario

### JwtTokenUtil

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Esta es la clase que genera el token y valida que este bien formado y no haya expirado.

### JwtTokenFilter

Texto

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamente

Es la clase que utiliza el JwtTokenUtil para comprobar que el token es valido y permitir el acceso al usuario.

### SecurityConfiguration

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente con confianza media

En esta clase es donde tenemos la configuración de seguridad que extiende de WebSecurityConfigurerAdapter y este ya trae los métodos necesarios que tendremos que implementar en nuestra clase, como por ejemplo permitir solo la entrada a la aplicación por /api/login a los usuarios sin token, añadir el filtro que mencionamos anteriormente antes de la autenticación, de esta manera, si tienen token entran directamente, sin necesidad de volver a validar el usuario y contraseña.

### UsuarioCredencials

Texto

Descripción generada automáticamente

Esta es la clase para almacenar los datos del usuario

### UsuarioDetails

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Esta clase es la que enlaza a nuestro usuario con el userDetails que nos proporciona los datos necesarios que nos pide spring que tenga un usuario.

### Texto Descripción generada automáticamenteUsuarioDetailsService

UsuarioDetailsService es la clase que se encarga de buscar al usuario con el nombre de usuario.

## Application.properties

Texto

Descripción generada automáticamente

En este archivo es donde se indican todas las propiedades que hacen referencia al funcionamiento de la aplicación, como el puerto en el que va a funcionar, la configuración de la base de datos, el driver…

# Desarrollo – Base de datos

Vamos a ver como ha quedado la base de datos después de haber visto las anotaciones y las relaciones en la parte del back

Diagrama

Descripción generada automáticamente

Para cargar los datos mínimos para poder probar el funcionamiento de la base de datos he creado un archivo sql que tenemos que ejecutar en el DBeaver en mi caso o en un workbench por ejemplo.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Texto, Tabla

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

El campo password es la contraseña encriptada con bcrypt con una fuerza de 12 rondas.

# Desarrollo – Front

Hemos visto una parte del front en apartados anteriores, pero vamos a ver lo que nos queda.

Para empezar, tenemos la página de Login

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

Al darle a Iniciar, hace una petición al back

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Devuelve un 200 OK y la respuesta:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Patrón de fondo

Descripción generada automáticamente con confianza media

El usuario admin es el usuario del profesor 1, el cual imparte a DAM y DAW

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

En el desplegable vemos los 2 años de DAM, si entramos en el año 1:

Tabla

Descripción generada automáticamente

Nos carga todos los alumnos de este curso, si entramos a ver al usuario 1 por ejemplo:

Vemos los datos detallados

Gráfico, Gráfico radial

Descripción generada automáticamente

La idea es que aquí aparezcan todas las asignaturas del alumno con sus respectivas medias

## 6.1 Login

Imagen que contiene Escala de tiempo

Descripción generada automáticamente

Imagen que contiene Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

En esta parte, cogemos los datos introducidos en usuario y contraseña y hacemos una petición al back a /api/login y si la respuesta es 200 OK, guardamos el usuario y el token en localStorage del navegador.

## 6.2 Dashboard

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

En la primera ventana, una vez entramos a la aplicación, se hace una petición al back con el nombre de usuario para obtener el objeto entero del usuario, a partir de este obtenemos el profesor y hacemos una petición para sacar los módulos a los que da clase el profesor y los cursos.

## 6.3 Curso

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

En esta ventana recibimos por parámetros del router link, el nombre del modulo y el id del curso y su año

Con el curso obtenemos todos los alumnos y con un for rellenamos toda la tabla con sus datos.

## 6.4 Alumno

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

En esta última parte del proyecto ya damos información mas detallada del alumno junto con las asignaturas y las notas medias de estas.

No me ha dado tiempo a hacer esta parte dinámica como el resto, he añadido la grafica fija para que se pudiera ver cual era realmente la idea.

# Swagger

He utilizado Swagger para tener bien documentada la API REST, ya que puedes ver todos los endpoints, detalles de que parámetros esperan, que tipo de datos, la ruta…

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Se puede ver como indicándole el nombre de un rol, nos devuelve un 200 y el rol

# Conclusiones

Como he explicado en una de las partes del trabajo, no he podido acabar la parte del front por falta de tiempo ya que todo el tema del back, la seguridad y la base de datos me ha llevado bastante y he tenido algún que otro problema. Por ejemplo, el tema del bucle en la serialización me ha costado encontrar el error.

He disfrutado mucho haciendo todo de cero ya que por primera vez he llegado a ver el “puzzle” entero. Como circulan los datos desde la base de datos al front y todos los problemas con los que te vas encontrando.

El tema del front si que ha sido un reto muy grande ya que soy mucho mas fan del back y a parte, nunca había trabajado anteriormente con un framework en front, como Vue. He aprendido muchísimo y ha sido una experiencia muy buena.