Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Desarrollo back-end en .NET de una aplicación web para el reconocimiento de DNIs en exámenes manuscritos y su clasificación para su posterior distribución personalizada

Trabajo Fin de Máster

Máster Universitario en Ingeniería Informática

Autor: Josep Ferrandis Jorge

Tutor: Antonio Martí Campoy

2020-2021

Resumen

Este trabajo se centra en diseñar e implementar la parte servidor o *back-end* de una aplicación web que ofrecerá soporte haciendo uso de un catálogo de servicios para la clasificación automática de exámenes evaluados a partir del DNI del alumno.

Para realizar dicha clasificación se utilizarán tecnologías de inteligencia artificial de reconocimiento de caracteres manuscritos, las cuales hacen uso de redes neuronales.

La aplicación permitirá mejorar el proceso de aprendizaje del estudiante, ya que facilitará a los profesores la tarea de mostrar los exámenes evaluados, y de esta forma los estudiantes podrán aprender de sus propios errores a la vez que comprobar que su corrección ha sido justa.

El sistema incluye la gestión de asignaturas, exámenes y profesores, para que se pueda trabajar de forma organizada. Además, proporcionará herramientas para resolver los posibles conflictos que pueda producir el predictor de caracteres manuscritos a la hora de reconocer el DNI del alumno.

A pesar de que este trabajo se centra en el diseño e implementación de la parte servidor, también se ha desarrollado una aplicación cliente básica para poder demostrar el funcionamiento y poder hacer uso de todas las funcionalidades.

Para llevar a cabo el desarrollo de este proyecto se ha hecho uso principalmente de las tecnologías y lenguajes .NET Core, SQL, C#, Docker. También se ha hecho uso de la metodología TDD a la hora de implementar la aplicación.

Palabras clave: .NET Core, C#, SQL, Docker, TDD, back-end, API-REST

Resum

Aquest treball se centra a dissenyar i implementar la part servidor o *back-end* d'una aplicació web que oferirà suport fent ús d'un catàleg de serveis per a la classificació automàtica d'exàmens avaluats a partir del DNI de l'alumne.

Per a realitzar la dita classificació s'utilitzaran tecnologies d'intel·ligència artificial de reconeixement de caràcters manuscrits, les quals fan ús de xarxes neuronals.

L'aplicació permetrà millorar el procés d'aprenentatge de l'estudiant, ja que facilitarà la tasca als professors de mostrar els exàmens avaluats, i d'aquesta manera els estudiants podran aprendre dels seus propis errors al mateix temps que comprovar que la seua correcció ha sigut justa.

El sistema inclou una gestió d'assignatures, exàmens i professors, perquè es puga treballar de forma organitzada. A més, proporcionarà eines per a resoldre els possibles conflictes que puga produir el predictor de caràcters manuscrits a l'hora de reconéixer el DNI de l'alumne.

Malgrat que aquest treball se centra en el disseny i implementació de la part servidor, també s'ha desenvolupat una aplicació client bàsica per a poder demostrar el funcionament i poder fer ús de totes les funcionalitats.

Per a dur a terme el desenvolupament d'aquest projecte s'ha fet ús principalment de les tecnologies i llenguatges .NET Core, SQL, C#, Docker. També s'ha fet ús de la metodologia TDD a l'hora d'implementar l'aplicació.

Paraules clau: .NET Core, C#, SQL, Docker, TDD, back-end, API-REST

Abstract

This project focuses on designing and implementing the back-end of a web application that will provide support using a catalogue of services for the automatic classification of evaluated exams based on the student's DNI.

To perform this classification, artificial intelligence technologies for handwritten characters recognition will be used, which make use of neural networks.

The application will improve the student's learning process by making it easier for teachers to display the exams, so that students can learn from their own mistakes and check that they have been corrected fairly.

The system includes a management of subjects, exams, and teachers, so that they can work in an organized way. In addition, it will provide tools to resolve possible conflicts that the handwritten characters predictor may produce when recognizing the student's DNI.

Although this work focuses on the design and implementation of the server side, a basic client application has also been developed to demonstrate how it works and to be able to make use of all the functionalities.

To carry out the development of this project we have made use mainly of the technologies and languages .NET Core, SQL, C#, Docker. The TDD methodology has also been used to implement the application.

Key words: .NET Core, C#, SQL, Docker, TDD, back-end, API-REST

Glosario

- **Front-end:** la parte de un sistema informático o de una aplicación con la que el usuario interactúa directamente.
- **Back-end:** la parte de la aplicación a la que no accede directamente el usuario, normalmente responsable de almacenar y manipular datos.
- **Framework:** es una estructura que ofrece soporte tecnológico, principalmente a través de módulos y bibliotecas. Puede servir como estructura base para el desarrollo de nuevas aplicaciones de software.
- **Arquitectura software:** es la estructura de un sistema informático en la que se define cada una de las relaciones entre todos los componentes que lo componen.
- **Interfaz:** es un método de intercambio de información entre dos o más componentes y/o usuarios de un sistema informático.
- **API:** es una interfaz de programación de aplicaciones (Application Programming Interface). Es un intermediario software que ofrece a un software la posibilidad de utilizar los servicios de otro software.
- **Test unitario:** son pruebas automatizadas escritas y ejecutadas por los desarrolladores de software para garantizar que una parte pequeña de la lógica de una aplicación satisface el comportamiento esperado.
- **Arquitectura de Capas:** es una forma de estructurar una aplicación donde las estructuras fundamentales de un sistema de software se separan en capas o niveles.
- **Dto (Data transfer object):** se trata de un objeto que se utiliza para encapsular datos y enviarlos de un subsistema de una aplicación a otro.
- **Entidades:** se trata de un objeto perteneciente al dominio de la aplicación. Estas entidades representan las tablas que se dispondrán en la base de datos relacional de la aplicación.
- **Http (*Hypertext Transfer Protocol*):** se trata de un protocolo que pertenece a la capa de aplicación. Este protocolo está diseñado para transferir información entre dispositivos en red y se ejecuta sobre otras capas de la pila de protocolos de red.
- **Diagrama entidad-relación:** muestra las relaciones de los conjuntos de entidades almacenados en una base de datos.
- **Endpoint:** Es la URL de la API o un *back-end* que responde a una petición.

Tabla de contenidos

Tabla de figuras

Tablas

1. Introducción

Este Trabajo final de máster se desarrolla en base a una propuesta del tutor. Este trabajo viene motivado por una mejora en el proceso de aprendizaje del alumnado durante sus estudios en la universidad. Para ello se plantea una herramienta que facilite a los profesores la tarea de organizar y clasificar los exámenes evaluados de su alumnado. Esta clasificación se compone de dos acciones. La primera, identificar las hojas de examen que corresponden a un alumno, extrayéndolas del total de hojas de todos los alumnos. La segunda, identificar al alumno por medio del DNI manuscrito en la primera hoja del examen. De esta forma los profesores podrán compartir estos exámenes corregidos, digitalizados y clasificados con sus alumnos, por ejemplo, utilizando el correo electrónico o el sistema de recursos compartido de PoliformaT.

Proporcionar este recurso al alumnado es beneficioso por dos motivos. En primer lugar, el estudiante podrá revisar que su examen ha sido evaluado y corregido de una forma justa y en segundo lugar es una gran fuente de aprendizaje, ya que el estudiante podrá revisar sus fallos y aprender de ellos. Además, utilizando esta herramienta se gestionará mejor el tiempo, tanto del alumno como del profesor, puesto que no serán necesarias tutorías entre ellos en caso de que el alumno esté conforme con el resultado de la corrección del examen o no necesite aclaraciones sobre los errores cometidos. A su vez se mejorarán las atenciones cuando el alumno requiera una revisión con la presencia, física o virtual, del profesor.

Para la realización de este trabajo, se desarrollará una aplicación web que se dividirá en dos partes, la parte *front-end* y la parte *back-end*. La primera parte, se encargará de todas las interfaces gráficas del usuario, es decir toda la parte visual de la aplicación. La segunda parte, el *back-end* o servidor, se encargará de procesar y almacenar todos los datos de la aplicación web. Este TFM se centrará en la segunda parte mencionada, la parte *back-end*, de la que se realizará el diseño, la implementación y despliegue. Es importante mencionar que se utilizará una metodología llamada TDD o *Test Driven Development* para la implementación del servidor. Esta metodología se explicará más adelante.

La aplicación estará dirigida a un solo tipo de usuario, un usuario profesor el cual dispone de un grupo de alumnos.

La aplicación permitirá clasificar exámenes evaluados por un profesor a través del reconocimiento del DNI escrito por los estudiantes que lo realizan. Este reconocimiento de texto se realizará a través de la utilización de un motor de reconocimiento basado en de redes neuronales, las cuales ofrecen una serie de características interesantes para este tipo de tarea.

Esta aplicación proporcionará a los profesores que la utilicen una herramienta que no solo les permitirá clasificar los exámenes evaluados de sus estudiantes, sino que además podrán gestionarlos en base a sus asignaturas.

Como la aplicación utilizará un sistema de reconocimiento de caracteres manuscritos y estos reconocimientos podrán no ser exactos, el sistema ofrecerá una gestión para estos casos mediante la cual se proporcionará soporte a los profesores ayudándoles a realizar la asignación del DNI al examen de una forma cómoda.

Además, la aplicación también ofrecerá una funcionalidad en la que el profesor marcará un examen como revisado, tras comprobar que la predicción o la asignación del DNI es correcta. Esto ofrecerá dos ventajas, por un lado, el profesor sabrá que exámenes están pendientes de revisar y, por otro lado, podrá dejarlos para revisar más tarde, ya que la aplicación guardará esta información, por lo que no es necesario revisar todos los exámenes en una única sesión de trabajo con la aplicación.

1.1 Objetivos

El objetivo del trabajo es desarrollar una aplicación web que reconozca el DNI manuscrito en exámenes escaneados, y asocie las páginas del examen a cada alumno, creando un fichero ZIP descargable con las hojas de los exámenes de cada alumno en formato PDF. Además, esta aplicación incluye gestión de asignaturas, exámenes y profesores, para que diferentes asignaturas y profesores puedan trabajar de forma organizada.

Para poder alcanzar el objetivo principal del proyecto, será necesario cumplir con ciertos objetivos específicos que se detallan a continuación:

- Definir el motivo de la aplicación.
- Plantear y analizar los requisitos que serán necesarios para el desarrollo de la aplicación.

- Examinar los posibles casos de uso de la aplicación web.
- Comprender, utilizar y dominar la metodología TDD.
- Diseñar la arquitectura y el modelo de datos a implementar en la aplicación.

El desarrollo del *front-end* o parte cliente de la aplicación donde reside toda la parte visual de la aplicación, no son objetivo del proyecto. No obstante, se ha implementado una interfaz básica pero completamente operativa con la finalidad de que el tutor y otros profesores puedan hacer pruebas para comprobar la usabilidad y aplicabilidad de la aplicación.

1.2 Estructura

La memoria está organizada en doce apartados. En el primer apartado se encuentra la introducción que engloba los objetivos generales y la estructura de la memoria.

En lo referido al contenido del segundo punto, se plantea una estimación de tiempo necesaria para el desarrollo de las distintas tareas en las que se encuentra dividido el proyecto.

En tercer lugar, se encuentra el estado del arte, en el que se detallan las aplicaciones similares que existen actualmente y la investigación realizada sobre las posibles tecnologías de reconocimiento de textos.

Por otro lado, en el cuarto punto, se definen y plasman los requisitos software de la aplicación utilizando el estándar IEEE 830/1998 [11].

En cuanto al quinto punto, se concretan los distintos casos de uso a través de diagramas de secuencia y de diagramas de caso de uso. Además, se plasman las definiciones referentes a cada caso mediante una tabla.

En el siguiente apartado, diseño, se desarrolla la explicación sobre el diseño de la propia aplicación, tanto en lo referido a la arquitectura del servidor como al diseño de la base de datos con el diagrama de entidad-relación correspondiente.

Tras haber finalizado la etapa de diseño, se llevará a cabo la fase de implementación en la cual se detallarán las tecnologías y las herramientas utilizadas para el desarrollo de la aplicación.

Una vez haya concluido la etapa de implementación, se realizará una descripción de todas las pruebas llevadas a cabo que garantizan el correcto funcionamiento de los requisitos definidos con anterioridad.

Tras la comprobación de la correcta funcionalidad y cumplimiento de todos los casos de uso de la aplicación desarrollada, se llevará a cabo el despliegue de la misma.

A modo de cierre, se incluye un apartado de conclusiones en el cual se reflexionará sobre los posibles problemas surgidos durante el desarrollo del proyecto y distintos aspectos a tener en cuenta. Asimismo, se encuentran los apéndices y la bibliografía que se ha utilizado para la obtención de la información necesaria para la realización de este proyecto.

2. Estimación de tiempos

Este apartado detalla la estimación de tiempos que se debería cumplir para la realización de cada una de las etapas o fases del proyecto. Se va a utilizar la metodología de desarrollo llamada en cascada, que divide el trabajo en etapas que se ejecutan de forma secuencial o con poco solape entre ellas. A continuación, se describen las etapas y el tiempo establecido para cada una de ellas. Finalmente, la Tabla 1 muestra la duración en tiempo del proyecto, establecido en 12 semanas.

En primer lugar, está la fase de análisis, en la cual se identifican tanto los requisitos funcionales y no funcionales del proyecto, esta fase se ha estimado en un período de dos semanas.

En segundo lugar, se realizará la fase de especificación de casos de uso. Se necesitará una semana para realizar todos los estudios pertinentes a los casos de uso de la aplicación, asimismo se diseñarán los diferentes diagramas de secuencia que muestren el flujo de la aplicación.

A continuación, se encuentra la fase de diseño, en la cual se realizará un pequeño estudio sobre las tecnologías que se necesitarán, cual es el tipo de arquitectura que se ajusta más a las necesidades del proyecto, etc. Debido a la gran importancia de esta fase en el desarrollo del proyecto se plantea utilizar un total de dos semanas para la realización de todo el diseño.

Una vez finalizada la fase de diseño de la aplicación, se empezará la fase de implementación, en la cual se llevará a cabo todo el desarrollo de la aplicación web. Esta fase será la más costosa de todo el proyecto y se ha estimado un total de cinco semanas para su realización. Esta fase se iniciará juntamente con la fase de pruebas de la aplicación, ya que se va a hacer uso de la metodología TDD, la cual consiste principalmente en desarrollar todos los casos de prueba antes que la propia implementación. Es importante mencionar que esta fase de pruebas dispondrá de una semana adicional al finalizar la fase de implementación con el fin de revisar todos los casos de prueba junto con la funcionalidad de la aplicación.

Por último, se realizará la fase de despliegue donde se desplegará y configurará la aplicación en el equipo proporcionado por el tutor. Para la realización de esta fase se estima un total de dos semanas.

				Semanas											
Tarea	Inicio	Fin	Duración (dias)	1	2	3	4	5	6	7	8	9	10	11	12
Requisitos del proyecto	01/02/2021	14/02/2021	14												
Casos de uso	15/02/2021	21/02/2021	7												
Diseño	22/02/2021	07/03/2021	14												
Implementación	08/03/2021	11/04/2021	35												
Pruebas	08/03/2021	18/04/2021	42												
Despliegue	12/04/2021	26/04/2021	14												

Tabla 1 Estimación de tiempos

3. Estado del arte

En esta sección del trabajo se hablará sobre las investigaciones llevadas a cabo en las cuales se ha recogido información acerca de la existencia de tecnologías, herramientas y aplicaciones disponibles en el mercado que se puedan asemejar a la funcionalidad perseguida en este proyecto. Esta sección se divide en dos apartados, en primer lugar, se hablará acerca de las aplicaciones similares que existen en el mercado, con sus respectivas características y limitaciones. En segundo lugar, se detallarán las diferentes tecnologías que se han investigado para hacer posible el reconocimiento de texto manuscrito en la aplicación que se quiere desarrollar.

3.1 Aplicaciones similares

La primera máquina de corrección automática fue la *IBM 805 Test Scoring Machine* [1]. Esta máquina fue creada por el inventor norteamericano Reynold B. Johnson en el año 1934. Mediante esta máquina se podía puntuar las hojas de respuesta marcadas con lápices especiales. Su primer uso a gran escala fue en 1936, para la corrección del examen Regents de Nueva York.

Esta máquina no identificaba al examinado, sino que el operador de esta registraba la nota obtenida a partir de la salida de la máquina, por lo que se trataba de una máquina de corrección más que de clasificación. No obstante, en desarrollos posteriores se ideó la forma de identificar al alumno. La forma más habitual en España era mediante la codificación del número de DNI en una matriz de marcas donde la fila representaba el dígito y la columna la posición en el DNI.

Como se ha comentado, esta máquina abrió la puerta a la corrección automática y clasificación de pruebas académicas. Estas nuevas implementaciones fueron utilizadas en un gran número de universidades, incluida la Universitat Politècnica de València la cual proporcionó el servicio de corrección automatizada de exámenes tipo test hasta el año 2014.

Posteriormente desde el departamento de Informática de Sistemas y Computadores (DISCA), los profesores Alberto Gonzáles Téllez y José Miguel Valiente González desarrollaron ALCE [2, 3]. Esta herramienta utilizaba una hoja de respuestas mecanizada

que permitía al alumno codificar su DNI y sus respuestas. El uso de ALCE fue posible gracias a la instalación de máquinas fotocopiadoras que incluían escáneres con alimentadores automáticos. En la ilustración 1 se puede observar la hoja de respuestas utilizada por ALCE, la cual debía ser codificada por los alumnos.

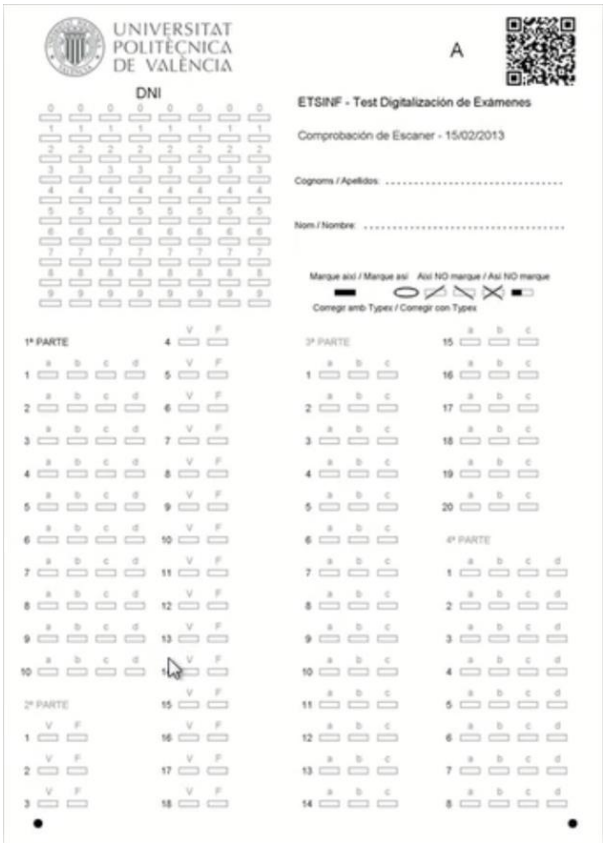


Ilustración 1 Hoja de respuestas ALCE

Además, ALCE proporcionó dos grandes ventajas a la hora de clasificar y corregir exámenes de una forma automática.

La primera de ellas es que era una aplicación a la que se accedía a través de la web, en concreto a través de PoliformaT, facilitando su uso por parte de los profesores, pues se evitaba la petición de cita previa y el desplazamiento físico al ICE para el escaneo de los exámenes.

La segunda ventaja, mucho más relevante para este trabajo, era la posibilidad de publicar, de forma automática, la hoja del examen de cada uno de los alumnos. La aplicación generaba un fichero con un identificador por alumno y lo publicaba automáticamente en su espacio compartido en PoliformaT. De esta forma el alumno recibía toda la

realimentación posible en este tipo de examen, sin perjuicio de que éste quisiera revisar su examen con el profesor.

ALCE podía utilizarse también en exámenes de respuesta abierta, o cualquier otro tipo de pregunta. En este caso no se realizaba la corrección automática, pero añadiendo como primera hoja del examen la hoja mecanizada, y a continuación las hojas de respuesta de los alumnos, era posible escanear los exámenes corregidos y publicarlos en el espacio compartido de PoliformaT. En algunos casos, incluso el profesor podía codificar la nota del alumno en la hoja mecanizada y esta se introducía automáticamente en la lista de notas. Esta forma de trabajar se utilizó durante algunos años en asignaturas del DISCA, como FCO (Fundamentos de computadores) del grado GITST (Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación) y FSO (Fundamentos de sistemas operativos) y FCO (Fundamentos de computadores) del grado GII (Grado en Ingeniería Informática).

Una nueva evolución de ALCE prescindía de la hoja mecanizada. En este caso se utilizaba una hoja de examen normal con una cabecera donde el alumno introducía su nombre, apellidos y DNI. Una vez corregidos los exámenes, estos se escaneaban y con una aplicación de escritorio, el profesor visualizaba la parte superior del examen y seleccionaba de la lista de alumnos de la asignatura, el alumno al que le correspondía el examen. Una vez asignado el alumno, se genera el fichero con las hojas de examen del alumno. Uno de los inconvenientes que planteaba esta aplicación era que, una vez realizada la asignación, esta no se podía deshacer, por lo que un error por parte del profesor requería una gestión que podía resultar compleja. Una vez asignados todos los exámenes, otra aplicación de escritorio permitía subirlos al espacio compartido de cada alumno en PoliformaT.

Como alternativa a estas dos aplicaciones de escritorio, algunos profesores han optado por hacer uso de los servicios de su sistema operativo, como la vista previa y la edición de ficheros PDF integrada en el explorador de archivos para realizar la clasificación de forma manual.

En todos los casos, ALCE y sus evoluciones presentaban una limitación importante. Esta era que el número de hojas que el alumno podía entregar era fijo y establecido a priori. En exámenes donde los enunciados incluyen los espacios para las respuestas, no es un

problema importante, pero en exámenes donde se utiliza el papel de examen de la UPV y el alumno tiene libertad para escribir tanto como considere necesario, sí es una restricción inasumible.

Además de la herramienta ALCE previamente descrita, existen múltiples aplicaciones, principalmente para móvil, que permiten escanear y/o corregir exámenes de tipo test, como por ejemplo ZipGrade [4]. Aunque no todas son idénticas, comparten características que no las hacen adecuadas para el propósito de este trabajo. Muchas de ellas están basadas en OMR (Optical mark recognitztion) por lo que sólo aceptan exámenes de tipo test. También la mayoría necesitan que en la hoja de examen se impriman marcas para ayudar al reconocimiento. Algunas son capaces de identificar al alumno si éste codifica su identificador en la hoja de respuestas.

Una propuesta más próxima al objetivo de este trabajo es el póster presentado en el pasado INRED 2021[1] celebrado en la UPV. En esta propuesta, la identificación del alumno se realiza mediante el uso de un código QR con la información del alumno. Este código QR se imprime en etiquetas autoadhesivas por el profesor antes del examen, y se entrega al alumno para que la pegue en la primera hoja del examen durante la realización de éste. Es una opción interesante cuando el número de alumnos es muy reducido, pero imposible de gestionar en asignaturas con varios centenares de alumnos.

GradeScop [5] es otra aplicación que representa una funcionalidad similar a la que se persigue en este trabajo. Los dos principales inconvenientes de esta herramienta son su precio (la versión más básica es de 1 dólar por alumno y año), y que los exámenes saldrían de los servidores de la UPV, lo que puede comprometer los requisitos de custodia y de protección de datos.

3.2 Tecnologías de reconocimiento de texto

Una de las funcionalidades que se desea para este trabajo es el reconocimiento de letras de imprenta y dígitos manuscritos, para asistir al profesor en la asignación de exámenes a alumnos. Esta tecnología está en un estado de madurez suficiente para su uso, y existen una gran variedad de opciones disponibles, que se muestran a continuación:

[1] En la fecha de redacción de esta memoria todavía no se han publicado las actas del INRED.

3.2.1 Cloud Vision API – Google Cloud

Cloud Visión [6] se trata de un servicio en la nube de Google el cual ofrece la posibilidad de integrar de una forma sencilla una gran variedad de funciones de reconocimiento de imágenes, en las cuales se incluyen detección de etiquetas, caras y puntos referencia, pero lo más interesante para este proyecto es la funcionalidad que ofrece para el reconocimiento de texto. Más concretamente ofrece una API sencilla, la cual ofrecerá todas estas funcionalidades de reconocimiento de imágenes.

3.2.2 Amazon Textract:

Amazon Textract [7] se trata de un servicio en la nube desarrollado por Amazon Web Services, este servicio ofrece la posibilidad de extraer de una forma totalmente automática texto, escritura y datos de imágenes escaneadas con funciones que van más allá de un simple OCR (reconocimiento óptico de caracteres).

3.2.3 Computer Vision – Microsoft Azure

Azure Computer Vision [8] es un servicio en la nube gestionado por la empresa Microsoft que proporciona acceso a un conjunto de algoritmos avanzados para el procesamiento de imágenes.

Esta herramienta, a partir de una imagen proporcionada, es capaz de devolver información relacionada con diversas características visuales de interés, como podría ser reconocimiento de imágenes, colores, objetos y sobre todo lo que más interesa para este proyecto, reconocimiento de texto escrito a máquina y también texto manuscrito.

Esta herramienta ofrece la posibilidad de ser utilizada a través de una API, mediante la cual se podrá utilizar todas las funcionalidades de que dispone este servicio.

3.2.4 Recogniform ICR SDK

Recogniform ICR SDK [37], desarrollado por Recogniform Technologies S.p.A, es un kit para el desarrollo de aplicaciones con reconocimiento de caracteres alfanuméricos manuscritos. Utiliza redes neuronales y lógica difusa para reconocer tanto la escritura europea como americana. El motor de reconocimiento se distribuye como una DLL para el sistema operativo Windows.

3.2.5 Lipi Toolkit

Lipi Toolkit [38] es un kit para el desarrollo de aplicaciones con reconocimiento de texto manuscrito (ICR) de código abierto escrito en C, disponible para Linux y Windows. Está desarrollado por HP Labs en India.

3.2.6 Tesseract OCR

Tesseract OCR [9] se trata de un motor de reconocimiento óptico de caracteres u OCR, el cual es de uso libre. Este motor permite utilizar más de cien lenguajes diferentes con lo que consigue que sus predicciones sean precisas. Además de estos lenguajes predefinidos, ofrece la posibilidad de crear nuevos lenguajes en los que se podrán añadir palabras personalizadas.

Tesseract ofrece la posibilidad de integración en una gran variedad de tecnologías, como son Python, NodeJs y .Net entre muchas otras. Además, también ofrece la posibilidad de ser instalado en un equipo y de esta forma no depender de terceros.

3.2.7 EasyOCR

EasyOCR [10] se trata de un motor OCR (Optical Character Recognition) de uso libre, desarrollado por Jaided AI. Mediante este motor es posible extraer textos de imagen y documentos, utilizando un procesamiento de lenguaje natural y herramientas de inteligencia artificial para conseguirlo. EasyOCR hace uso de más de 80 lenguajes para que los reconocimientos de textos sean precisos.

Al tratarse de un motor de uso libre, permite la posibilidad de instalarlo en cualquier sistema que se desee. Cabe destacar que ofrece la posibilidad de realizar dicha instalación tanto en Windows como en Linux.

Tras haber investigado cada una de las tecnologías detalladas anteriormente, se han obtenido las siguientes conclusiones.

El principal inconveniente de Recogniform ICR SDK, y de otros SDK similares de otras compañías, es que no es gratuito y algunas licencias incluyen royalties. La utilización de este kit requeriría conversaciones entre la UPV y la empresa, para establecer las condiciones en que se puede utilizar en un entorno web.

Aunque Lipi Toolkit aparenta, en un principio, ser una opción válida, rápidamente se encontró en su documentación que está diseñado para reconocer caracteres de diferentes idiomas utilizados en la India, por lo que no es útil para este trabajo.

Los tres servicios en la nube presentan principalmente dos desventajas, las cuales obligan a descartarlas por requerimientos del proyecto. Una es el coste, ya que no se trata de herramientas gratuitas, y la segunda y más importante es que los exámenes deben salir de la UPV y pasar por sus servidores.

Los dos motores OCR, Tesseract OCR y EasyOCR, son muy similares, pero con algunas pequeñas diferencias las cuales van a ser claves para la elección. EasyOCR funciona con menor precisión que Tesseract OCR a la hora de reconocer texto manuscrito, funcionalidad que va a ser de gran importancia para este proyecto. Además, Tesseract OCR ofrece una integración con la tecnología que se utilizará para el desarrollo de la aplicación.

Por los puntos mostrados anteriormente la tecnología que se va a utilizar para el reconocimiento del DNI escrito por los alumnos en los exámenes será Tesseract OCR.

4. Requisitos

En este capítulo se definirán los requisitos que la aplicación web que se pretende desarrollar deberá cumplir. Para ello, se utilizará el estándar IEEE 830/1998 [11], estándar que define una forma correcta y completa de presentar la especificación de requisitos.

4.1 Introducción

Se va a hacer uso del estándar IEEE 830/1998 [11] como se ha comentado, el cual permitirá una correcta y completa definición de los requisitos de la aplicación que se pretende desarrollar. Este estándar define las distintas secciones y estructura que debe seguir el análisis de requisitos. De esta forma se seguirá la estructura que se presenta a continuación.

4.1.1 Propósito

El principal objetivo de un análisis de especificaciones de requisitos es la correcta y completa definición de todas las funcionalidades que la aplicación en cuestión deberá satisfacer. Asimismo, se especificarán funcionalidades que la aplicación no realizará y por tanto quedarán fuera del alcance del proyecto.

4.1.2 Ámbito del sistema

Este proyecto tiene como objetivo desarrollar una API que dará soporte a una aplicación que permitirá clasificar por DNI, de forma automática, los exámenes corregidos por los profesores y de esta forma facilitarles el trabajo a la hora de organizar la información y proporcionar *feedback* a sus alumnos.

El usuario que utilice esta aplicación será capaz de gestionar sus asignaturas, pudiendo crear, borrar y obtener información de ellas. Cabe destacar que para la creación de nuevas asignaturas solo será necesario definir el nombre de éstas.

De la misma forma que las asignaturas, el usuario será capaz de gestionar los exámenes de cada asignatura, pudiendo crear, eliminar y obtener información sobre cada uno de estos exámenes.

Además de estas gestiones mencionadas, el usuario será capaz de asignar estudiantes a cada una de sus asignaturas, esta asignación será desde una exportación de la lista de alumnos que ofrece la herramienta PoliformaT, herramienta que pertenece a la Universitat Politècnica de València.

En base a estas funcionalidades, el usuario será capaz de subir los exámenes corregidos, estos exámenes deben estar digitalizados en formato PDF, y podrán estar de forma continuada en un mismo documento o en distintos documentos.

El sistema clasificará de forma automática estos exámenes subidos por el usuario, pero como se trata de reconocimiento manuscrito, puede pasar que el sistema no encuentre una única solución al reconocer el DNI, a este caso concreto se le llamará "exámenes con conflicto", y será necesaria una acción manual por parte del usuario para resolverlo, y podrá realizarlo de dos formas distintas. En primer lugar, el usuario podrá elegir entre tres opciones que proporcionará el sistema, basadas en las tres mejores predicciones realizadas por el predictor. La segunda opción se trata de escribir de forma manual el DNI del alumno. Con la finalidad de evitar errores tipográficos en este segundo caso, el sistema comprobará que el DNI proporcionado por el usuario se encuentra entre los estudiantes que están cursando la asignatura.

Una vez se han clasificado los distintos exámenes, el usuario podrá marcar con un tic aquellos exámenes que ya ha revisado y el DNI clasificado corresponde exactamente con los datos del propio examen. Esta funcionalidad ayudará a los usuarios a tener el control de qué exámenes han sido revisados y cuáles no.

Por último, el usuario podrá descargar un archivo ZIP con todos y cada uno de los exámenes en formato PDF. Cada archivo PDF recibirá como nombre la concatenación del nombre del examen con el DNI del alumno correspondiente.

En ningún caso el sistema corregirá los exámenes o extraerá las notas incluidas en ellos.

4.1.3 Referencias

IEEE Recommended Practice for Software Requirements Specification. ANSI/IEEE 830/1998.

4.1.4 Visión general del documento

Este apartado de la memoria consta de tres partes: la introducción, descripción general y requisitos específicos. La introducción pretende proporcionar una visión general este trabajo. La descripción general pretende proporcionar la información referente a todos los aspectos relevantes de la aplicación. Por último, los requisitos específicos pretenden definir de una forma detallada todos los requisitos del proyecto. El nivel de detalle deberá ser suficiente para poder realizar la implementación del mismo en la siguiente fase del proyecto. Mediante estos requisitos será posible la planificación de las pruebas necesarias que indicarán en un futuro el cumplimiento de cada uno de los requisitos presentados.

4.2 Descripción general

En este apartado se pretende describir todos los aspectos relevantes a los requisitos de la aplicación. En primer lugar, se describirá el contexto en el que se desarrollará el proyecto y a continuación se definirán cada uno de los requisitos de una forma más detallada.

4.2.1 Perspectiva del producto

Se va a desarrollar una aplicación web *back-end* la cual ofrecerá unos servicios o funcionalidades a través de una API, que podrán ser consumidos por un *front-end*. Este *front-end* podrá ser diseñado e implementado en cualquier tecnología que permita realizar peticiones HTTP REST.

4.2.2 Funciones del producto

Este apartado definirá todas las principales funcionalidades que la aplicación va a ser capaz de realizar. Sin embargo, antes de empezar con la descripción de cada una de las funcionalidades del producto, es importante destacar la distinción de los dos roles de usuarios que existen en esta aplicación. Estos dos roles se pueden apreciar en la ilustración 2. Por una parte, existe el rol de usuario no registrado, el cual tiene como única funcionalidad registrarse en el sistema. Por otra parte, existe el rol de usuario profesor, el cual pueden utilizar todas las funcionalidades de la aplicación.

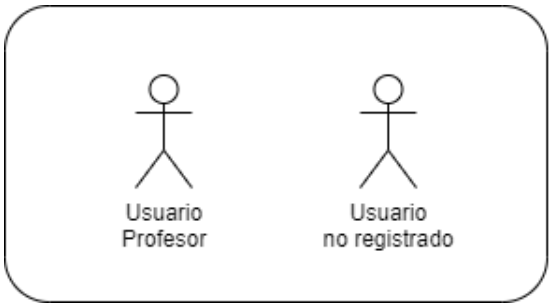


Ilustración 2 Roles de la aplicación

A continuación, se van a describir las principales funciones que la aplicación web realizará:

- Registro de usuario: Cualquier usuario con acceso a la aplicación va a poder registrarse en ella para su posterior uso.
- Inicio de sesión: Todo usuario registrado podrá iniciar sesión en la aplicación.
- Crear, obtener y eliminar asignaturas: El usuario tendrá la posibilidad de crear nuevas asignaturas o eliminar las asignaturas de los que ya disponía. Para la creación de nuevas asignaturas será necesario proporcionar un nombre de asignatura. Es importante destacar que, a la hora de borrar una asignatura, también se borrarán todos los exámenes con sus correspondientes clasificaciones. Además, el usuario también podrá recuperar la información perteneciente a sus asignaturas.
- Crear, obtener y eliminar exámenes: El usuario tendrá la posibilidad de crear nuevos exámenes o eliminar los exámenes que previamente había creado. Para la creación de nuevos exámenes será necesario proporcionar un nombre de examen. Cuando se borre un examen, también se borrarán todos los exámenes evaluados y clasificados de los estudiantes. Además, el usuario también podrá recuperar la información perteneciente a sus exámenes.
- Asignar lista de alumnos a una asignatura: El usuario podrá asignar una lista de alumnos a una asignatura, este proceso será una importación de la lista de alumnos que los profesores pueden descargar a través de PoliformaT en formato CSV.

- Subir exámenes para ser clasificados: Tras haber subido el examen a la aplicación, esta realizará las siguientes acciones:
    - Conversión de cada página de los exámenes al formato de imagen PNG.
    - Separación de cada uno de los exámenes corregidos.
    - Predicción del DNI del examen.
    - Almacenamiento del examen en un repositorio de documentos.
    - Almacenamiento de los metadatos del examen a la base de datos.
- Listar exámenes clasificados: El usuario podrá visualizar los exámenes clasificados por DNI, a través de dos listas. La primera contendrá los exámenes clasificados sin conflicto, es decir los exámenes que se ha podido predecir el DNI de forma correcta. La segunda contendrá los exámenes clasificados con conflicto, que son los exámenes que no se ha podido predecir el DNI de forma inequívoca.
- Resolver conflictos de predicción: El usuario podrá resolver el DNI de aquellos exámenes donde el predictor no haya encontrado una única solución para el DNI. El usuario podrá elegir entre tres opciones proporcionadas por la aplicación o escribir de forma manual el DNI.
- Marcar o desmarcar un examen como revisado: Para asegurar la exactitud en la revisión de cada examen, existirá un botón en el cual el docente podrá marcar o desmarcar un examen como revisado o no.
- Descargar exámenes clasificados: Una vez los exámenes hayan sido clasificados, el usuario tendrá la opción de descargar todos los exámenes en formato PDF comprimidos en un archivo con formato Zip.

4.2.3 Características de los usuarios

Esta aplicación está pensada para facilitar al profesorado y al alumnado la comunicación en cuanto a la corrección de exámenes. A través de esta aplicación, será mucho más sencillo poder clasificar y comunicar los resultados de los exámenes realizados por los estudiantes en las distintas asignaturas cursadas. El usuario dispondrá de su propio perfil en el que iniciará sesión y podrá acceder a los exámenes realizados por los estudiantes que cursan su asignatura.

Puesto que la aplicación dará soporte a una gestión de usuarios, cada profesor/a podrá establecer y organizar sus asignaturas y sus exámenes.

4.2.4 Restricciones

La aplicación presenta tres restricciones relacionadas con el reconocimiento automático del DNI y la asignación de hojas de examen a un alumno. La primera restricción es la necesaria aparición conjunta de las palabras impresas EXAMEN, DNI y NOMBRE en la primera página del examen. La segunda restricción es que en las siguientes páginas podrán aparecer las palabras DNI y NOMBRE, pero no EXAMEN. Y la tercera restricción es la calidad del texto manuscrito, que afectará directamente al éxito del reconocimiento automático.

4.2.5 Suposiciones y Dependencias

Como se ha comentado, esta aplicación está diseñada para profesores que tengan exámenes para clasificar.

La solución *back-end* está preparada para poder trabajar juntamente con cualquier cliente que sea capaz de enviar y recibir peticiones HTTP.

El rendimiento de la aplicación web puede verse afectado por el tipo de servidor en el que se aloje dicha aplicación.

4.2.6 Requisitos Futuros

Podría ser de interés poner en marcha los siguientes requisitos en un futuro:

- Integrar y automatizar el envío de los exámenes clasificados al espacio compartido de PoliformaT.
- Crear flexibilidad en cuanto a la disposición de la lectura del campo DNI.
- Ordenar los exámenes alfabéticamente, por nombres a la hora de ser descargados.

4.3 Requisitos específicos

En esta sección se definirá cada uno de los requisitos de una forma mucho más detallada y completa tal y como dicta el estándar IEEE 830/1998.

4.3.1 Interfaces Externas

El objetivo de este proyecto es el desarrollo del *back-end*, pero para permitir a los profesores evaluar su funcionamiento se desarrollará una interfaz básica de usuario.

4.3.2 Funciones

En esta sección se van a detallar todas las tareas que serán necesarias para la realización y cumplimiento de todas las funcionalidades descritas anteriormente. Para ello, se va a presentar una serie de tablas las cuales detallarán todo lo necesario para poder realizar cada una de las tareas.

Número	1
Requisito	Registrar usuario
Prioridad	Media
Descripción	El usuario se registra en la aplicación proporcionando un nombre de usuario único y una contraseña.

Tabla 2 Requisito: Registrar usuario

Número	2
Requisito	Iniciar sesión
Prioridad	Alta
Descripción	El usuario inicia sesión en la aplicación proporcionando el usuario y la contraseña con el que se ha registrado.

Tabla 3 Requisito: Iniciar sesión

Número	3
Requisito	Cerrar sesión
Prioridad	Media
Descripción	El usuario cierra la sesión de la aplicación

Tabla 4 Requisito: Cerrar sesión

Número	4
Requisito	Crear asignatura
Prioridad	Alta
Descripción	El usuario crea una nueva asignatura proporcionando el nombre de esta.

Tabla 5 Requisito: Crear asignatura

Número	5
Requisito	Eliminar asignatura
Prioridad	Baja
Descripción	El usuario elimina una asignatura con lo que se eliminará toda la información referente a ella incluida la información de los exámenes de la asignatura.

Tabla 6 Requisito: Eliminar asignatura

Número	6
Requisito	Obtener lista de asignaturas
Prioridad	Alta
Descripción	El usuario lista todas las asignaturas que ha creado.

Tabla 7 Requisito: Obtener lista de asignaturas

Número	7
Requisito	Obtener la información de una sola asignatura
Prioridad	Alta
Descripción	El usuario recibe información sobre los nombres de los exámenes creados dentro de la asignatura que previamente ha creado.

Tabla 8 Requisito: Obtener la información de una sola asignatura

Número	8
Requisito	Asignar estudiantes a la asignatura
Prioridad	Alta
Descripción	El usuario asigna los estudiantes que cursan la asignatura creada proporcionando un archivo CSV exportado directamente de PoliformaT.

Tabla 9 Requisito: Asignar estudiantes a la asignatura

Número	9
Requisito	Crear examen
Prioridad	Alta
Descripción	El usuario crea un nuevo examen proporcionando el nombre de éste.

Tabla 10 Requisito: Crear examen

Número	10
Requisito	Eliminar examen
Prioridad	Baja
Descripción	El usuario elimina un examen, eliminando a su vez toda la información.

Tabla 11 Requisito: Eliminar examen

Número	11
Requisito	Obtener todos los exámenes de una asignatura.
Prioridad	Alta
Descripción	El usuario lista todos los exámenes que ha creado para una asignatura.

Tabla 12 Requisito: Obtener todos los exámenes

Número	12
Requisito	Obtener la información de un solo examen
Prioridad	Alta
Descripción	El usuario recibe la información perteneciente a la clasificación de los exámenes evaluados de sus alumnos. Mas concretamente se recupera el nombre del estudiante, el DNI reconocido, una pequeña imagen del campo donde el estudiante ha escrito el DNI y una caja que mostrará si el examen ha sido revisado o no.

Tabla 13 Requisito: Obtener la información de un solo examen

Número	13
Requisito	Subir exámenes de los alumnos corregidos
Prioridad	Alta
Descripción	El usuario sube los exámenes corregidos de los alumnos, con la finalidad de que el

	sistema los procese y los clasifique reconociendo el DNI de cada uno de ellos.

Tabla 14 Requisito: Subir exámenes de los alumnos corregidos

Número	14
Requisito	Descargar los exámenes clasificados
Prioridad	Alta
Descripción	El usuario descarga en un fichero ZIP todos los exámenes clasificados por la aplicación. Los exámenes sin conflicto de reconocimiento de DNI recibirán nombre de examen más DNI, mientras que los exámenes con conflicto recibirán el nombre del examen más un nombre arbitrario.

Tabla 15 Requisito: Descargar los exámenes clasificados

Número	15
Requisito	Mostrar los exámenes clasificados
Prioridad	Alta
Descripción	El usuario lista todos los exámenes clasificados en dos listas distintas, la primera lista los exámenes sin conflicto, es decir donde el predictor de textos ha encontrado una sola solución. La segunda lista muestra los exámenes donde el predictor no ha sido capaz de encontrar una única solución al reconocer el DNI manuscrito.

Tabla 16 Requisito: Mostrar los exámenes clasificados por la aplicación

Número	16
Requisito	Marcar el examen como revisado
Prioridad	Media
Descripción	El usuario marca un examen como revisado tras comprobar que el DNI asignado corresponde exactamente con el DNI del examen en cuestión.

Tabla 17 Requisito: Marcar el examen como revisado

Número	17
Requisito	Mostrar la primera página del examen
Prioridad	Alta
Descripción	El usuario visualizará la primera página del examen.

Tabla 18 Requisito: Mostrar la primera página del examen

Número	18
Requisito	Mostrar los tres DNI que más se parezca al DNI predicho en caso de conflicto.
Prioridad	Alta
Descripción	El usuario lista los tres DNI con más semejanza según el predictor de textos para facilitar la tarea de asignar el DNI correcto al examen.

Tabla 19 Requisito: Mostrar los tres DNI que más se parezca al DNI predicho en caso de conflicto.

Número	19
Requisito	Asignar DNI a examen con conflicto eligiendo entre tres posibles opciones
Prioridad	Alta
Descripción	El usuario asigna el DNI al examen en cuestión eligiendo una de las tres posibles opciones que proporciona el sistema.

Tabla 20 Requisito: Asignar DNI a examen con conflicto eligiendo entre tres posibles opciones

Número	20
Requisito	Asignar DNI a examen con conflicto escribiéndolo
Prioridad	Alta
Descripción	El usuario asigna el DNI al examen en cuestión escribiendo en un formulario el DNI del alumno. El sistema comprobará que dicho DNI exista entre la lista de alumnos de la asignatura. Además, el sistema proporciona una funcionalidad donde aparecerá una lista con los posibles DNIs según el usuario vaya tecleándolo.

Tabla 21 Requisito: Asignar DNI a examen con conflicto escribiéndolo

4.3.3 Requisitos de rendimiento

Se debe cuidar la complejidad de los algoritmos para cumplir con los requisitos de rendimiento, ya que las predicciones de texto basados en imagen son tareas costosas. Además, el rendimiento de la aplicación estará sujeto a la capacidad de cómputo del servidor que aloje la aplicación.

4.3.4 Atributos del sistema

En esta sección se detallan las diferentes cualidades que serán necesarias para garantizar la calidad de la aplicación que se pretende desarrollar. Estas cualidades son: seguridad, fiabilidad, escalabilidad y portabilidad.

En primer lugar, el sistema debe ser seguro, debe garantizar persistencia de los datos y protegerse de accesos no autorizados.

En segundo lugar, la aplicación debe ser fiable, debe ofrecer todas y cada una de las funcionalidades descritas en los puntos anteriores. Además, todas las funcionalidades deben cumplir todos los requisitos descritos.

Por último, la aplicación debe ser portable, esto significa que los mecanismos de despliegue e instalación de la aplicación en otros servidores deben ser sencillos.

5. Casos de uso

Los casos de uso proporcionan información a alto nivel sobre los requisitos funcionales de la aplicación que se desarrollará. Haciendo uso de estos, se define y especifica el comportamiento de la aplicación que se pretende desarrollar. [12]

El objetivo de los casos de uso es establecer las funcionalidades del sistema, desde la perspectiva de los distintos roles de la aplicación. Los casos de uso pretenden establecer una guía para todo el proceso de desarrollo de la aplicación web que se pretende implementar.

Para la creación de estos casos de uso, se utilizarán los “diagramas de caso de uso”. En ellos, se encuentran los distintos actores y las diferentes funciones que el sistema llevará a cabo.

Se asigna un rol de usuario a cada actor, los cuales son los encargados de intercambiar información con la aplicación. Cada uno de los actores dispondrá de diversas actividades o acciones que corresponderán a la especificación de sus casos de uso.

A continuación, se presenta el diagrama de casos de uso de la aplicación y posteriormente se describe en detalle cada uno de los casos.

5.1 Diagrama de casos de uso

La ilustración 3, muestra el diagrama de casos de uso [12] de la aplicación.

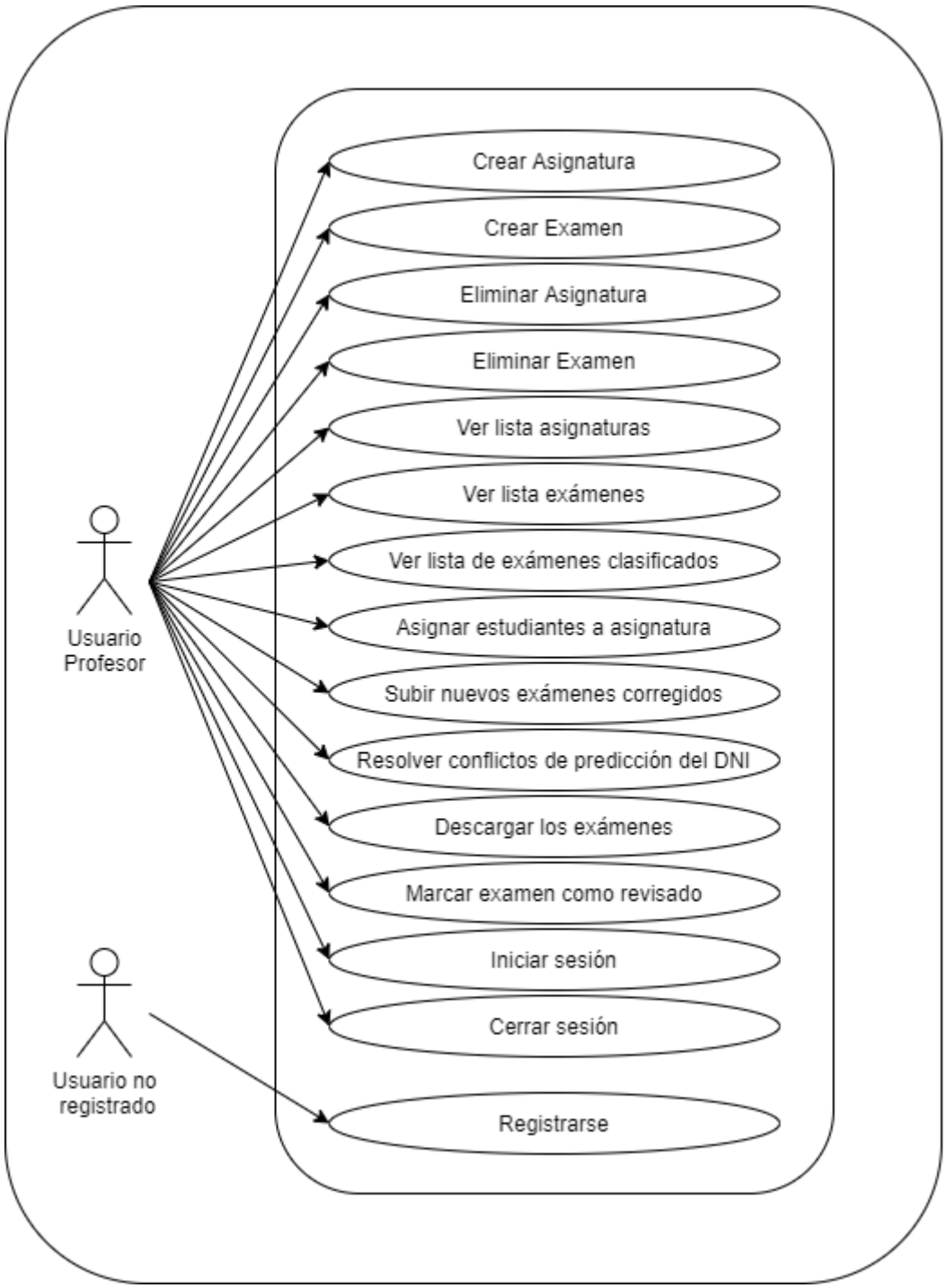


Ilustración 3 Casos de uso

5.2 Descripción casos de uso

En este apartado, las tablas 22 a 36 describen los casos de uso para posteriormente realizar la implementación de cada uno de ellos.

Caso de uso	Crear Asignatura
Actores	Usuario Profesor
Propósito	Añadir una nueva asignatura al sistema
Resumen	El usuario profesor añade una nueva asignatura a la que posteriormente se le añadirán exámenes.
Precondiciones	El profesor debe haber iniciado sesión en la aplicación.

Tabla 22 Caso de uso: Crear Asignatura

Caso de uso	Crear Examen
Actores	Usuario Profesor
Propósito	Añadir un nuevo examen al sistema
Resumen	El usuario profesor creará un nuevo examen perteneciente a una asignatura.
Precondiciones	El profesor debe haber iniciado sesión en la aplicación. Debe existir la asignatura.

Tabla 23 Caso de uso: Crear Examen

Caso de uso	Eliminar Asignatura
Actores	Usuario Profesor
Propósito	Eliminar una asignatura del sistema
Resumen	El usuario profesor elimina una asignatura del sistema eliminando a su vez todos los exámenes con sus correspondientes clasificaciones.
Precondiciones	El profesor debe haber iniciado sesión en la aplicación.

Tabla 24 Caso de uso: Eliminar Asignatura

Caso de uso	Eliminar Examen
Actores	Usuario Profesor
Propósito	Eliminar un examen del sistema
Resumen	El usuario profesor elimina un examen del sistema eliminando a su vez todas las clasificaciones realizadas de dicho examen.
Precondiciones	El profesor debe haber iniciado sesión en la aplicación.

Tabla 25 Caso de uso: Eliminar Examen

Caso de uso	Ver lista de asignaturas
Actores	Usuario Profesor
Propósito	Listar todas las asignaturas creadas por un profesor
Resumen	El profesor lista todas las asignaturas que hayan sido creadas por este propio usuario.
Precondiciones	El profesor debe haber iniciado sesión en la aplicación.

Tabla 26 Caso de uso: Ver lista de asignaturas

Caso de uso	Ver lista de exámenes
Actores	Usuario Profesor
Propósito	Listar todos los exámenes de una asignatura.
Resumen	El profesor lista todos los exámenes que hayan sido creados en una asignatura.
Precondiciones	El profesor debe haber iniciado sesión en la aplicación. Debe existir la asignatura de la cual se pretende listar los exámenes.

Tabla 27 Caso de uso: Ver lista de exámenes

Caso de uso	Ver lista de exámenes clasificados
Actores	Usuario Profesor
Propósito	Listar todos los exámenes de los alumnos los cuales han sido clasificados a partir del DNI por el sistema
Resumen	El usuario profesor lista los exámenes de los alumnos subidos a un examen una vez procesados y clasificados por el sistema, mostrándolos en dos listas diferentes.

	La primera lista muestra los exámenes donde el sistema ha encontrado una única solución al reconocimiento del DNI. Esta lista muestra el nombre del estudiante, el DNI reconocido, una pequeña imagen del campo donde el estudiante ha escrito el DNI y una caja que muestra si el examen ha sido revisado o no. La segunda lista muestra los exámenes donde el sistema ha encontrado múltiples soluciones a la hora de reconocer el DNI del examen. Esta lista muestra los tres DNIs que más se asemejan a la solución encontrada junto con una imagen del campo donde el estudiante ha escrito el DNI.
Precondiciones	El profesor debe haber iniciado sesión en la aplicación. Deben haberse creado con anterioridad la asignatura y el examen, y haberse subido exámenes de los alumnos al examen.

Tabla 28 Caso de uso: Ver lista de exámenes clasificados

Caso de uso	Asignar estudiantes a asignatura
Actores	Usuario Profesor
Propósito	Asignar lista de estudiantes a una asignatura
Resumen	El usuario profesor asigna estudiantes a la asignatura creada, esta asignación se realiza a través de la subida de un archivo en formato CSV que previamente ha sido descargado desde PoliformaT.
Precondiciones	El profesor debe haber iniciado sesión en la aplicación. Debe existir la asignatura a la cual se pretenden añadir estudiantes.

Tabla 29 Caso de uso: Asignar estudiantes a asignatura

Caso de uso	Subir nuevos exámenes corregidos
Actores	Usuario Profesor
Propósito	Subir nuevos exámenes corregidos al sistema

Resumen	El usuario profesor sube los exámenes escaneados en formato PDF de forma continuada, es decir uno o varios exámenes en un mismo PDF. El sistema separará cada uno de los exámenes de forma automática convirtiendo cada una de las páginas escaneadas en imágenes de formato PNG. Utilizando el predictor de textos, el sistema reconocerá el DNI de cada uno de los exámenes. En caso de no encontrar una única solución, el sistema guardará los tres DNI que más se asemejen a la solución encontrada. Una vez el sistema ha realizado el reconocimiento del DNI, éste subirá los exámenes a un repositorio de documentos con la finalidad de poder descargarlos en el futuro.
Precondiciones	El profesor debe haber iniciado sesión en la aplicación. Debe existir una asignatura y un examen creados previamente.

Tabla 30 Caso de uso: Subir nuevos exámenes corregidos

Caso de uso	Resolver conflictos de predicción del DNI
Actores	Usuario Profesor
Propósito	Resolver los conflictos generados por el sistema a la hora de predecir el DNI del examen.
Resumen	El usuario profesor resuelve el conflicto generado por el predictor de textos encargado de reconocer el DNI de cada examen, de dos formas distintas. La primera opción de resolución es la elección de DNI de entre tres posibles DNI que el sistema sugiere por ser los que más se han acercado al DNI reconocido. En la segunda opción de resolución de conflictos el profesor escribe el DNI del examen, el sistema comprobará que el DNI asignado existe entre la lista de estudiantes de la asignatura. Para ambos casos, el sistema muestra la primera página del examen donde se apreciará el DNI escrito por el alumno.

Precondiciones	El profesor debe haber iniciado sesión en la aplicación. Debe existir una asignatura, un examen y haberse subido exámenes de los alumnos previamente que tengan conflicto en el reconocimiento del DNI

Tabla 31 Caso de uso: Resolver conflictos de predicción del DNI

Caso de uso	Descargar los exámenes
Actores	Usuario Profesor
Propósito	Descargar los exámenes de los alumnos
Resumen	El usuario profesor descarga los exámenes que el sistema ha clasificado. A la hora de descargar los exámenes, el sistema proporciona al profesor un archivo Zip dentro del cual están cada uno de los exámenes de los estudiantes en formato PDF. Respecto a los exámenes que se hayan resuelto los conflictos de predictor recibirán el nombre del examen concatenado con el DNI del alumno, mientras que los exámenes que no se haya resuelto los conflictos recibirán el nombre del examen junto con "Anónimo" y un número consecutivo.
Precondiciones	El profesor debe haber iniciado sesión en la aplicación. Debe existir una asignatura, examen, y haberse subido exámenes de los alumnos previamente.

Tabla 32 Caso de uso: Descargar los exámenes

Caso de uso	Marcar examen como revisado
Actores	Usuario Profesor
Propósito	Indicar que se ha revisado la predicción del DNI del examen
Resumen	El usuario profesor revisa la predicción realizada por el sistema o resolución del conflicto que él mismo ha hecho, una vez se revisa que el DNI asignado en el sistema corresponde exactamente con el DNI del alumno, el profesor puede marcar dicho examen como revisado.
Precondiciones	El profesor debe haber iniciado sesión en la aplicación.

	Debe existir una asignatura, examen y haberse subido exámenes de los alumnos previamente.

Tabla 33 Caso de uso: Marcar examen como revisado

Caso de uso	Iniciar sesión
Actores	Usuario Profesor
Propósito	Iniciar sesión en el sistema
Resumen	El usuario profesor puede iniciar sesión en el sistema proporcionando su usuario y contraseña.
Precondiciones	El profesor debe estar registrado en la aplicación.

Tabla 34 Caso de uso: Iniciar sesión

Caso de uso	Cerrar sesión
Actores	Usuario Profesor
Propósito	Cerrar sesión en el sistema
Resumen	El usuario profesor cierra la sesión en la aplicación.
Precondiciones	El usuario debe haber iniciado sesión en la aplicación.

Tabla 35 Caso de uso: Cerrar sesión

Caso de uso	Registrarse
Actores	Usuario no registrado
Propósito	Registrar un nuevo profesor en el sistema
Resumen	El usuario no registrado puede crear una cuenta estableciendo un usuario y una contraseña.
Precondiciones	El nombre de usuario debe ser único.

Tabla 36 Caso de uso: Registrarse

5.3 Diagramas de secuencia

Seguidamente, se mostrará mediante la ilustración 4, el diagrama de secuencia con el objetivo final de mostrar el flujo de la aplicación general, tanto desde la perspectiva del usuario como desde la perspectiva del servidor.

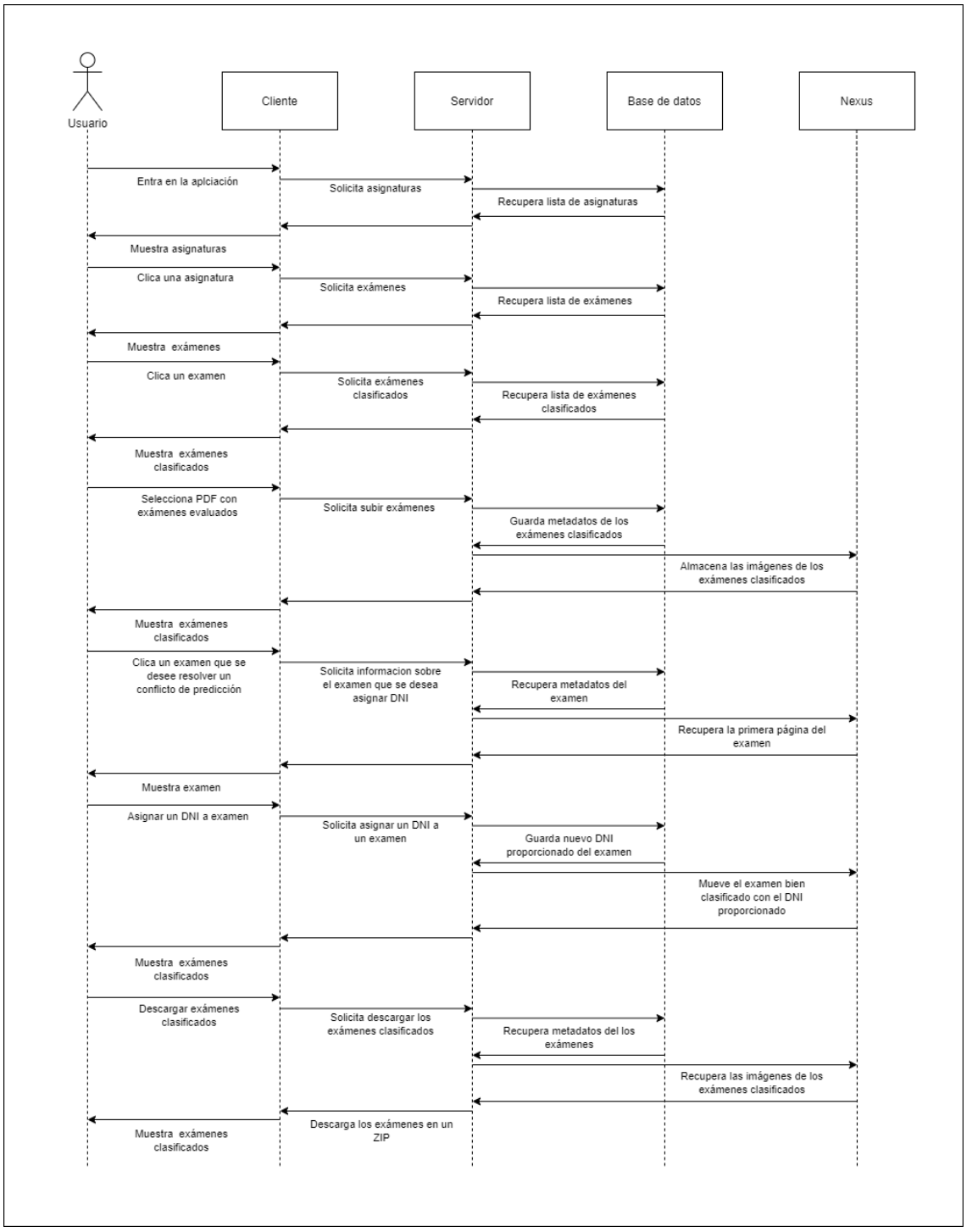


Ilustración 4 Diagrama de secuencia

6. Diseño

Tal y como se ha mencionado en la introducción, la aplicación que se plantea implementar está dividida en dos partes, una de ellas el *front-end* o cliente, la cual queda fuera de los objetivos de este proyecto. Esta es la parte visual de la aplicación y es la responsable de implementar la interfaz gráfica de usuario junto con algunas operaciones de poca carga computacional. La otra parte es el *back-end* o servidor, este es responsable de implementar todas las operaciones de mayor carga computacional y exponer cada una de las funcionalidades a través de una API, y su desarrollo es el objetivo de este trabajo.

De este modo se observa que esta aplicación utilizará un modelo cliente-servidor, en el que el *back-end* ofrecerá una serie de servicios o funcionalidades los cuales serán consumidos por un *front-end* a través de peticiones HTTP. Como se ha comentado con anterioridad el objetivo de este TFM es implementar el *back-end* de la aplicación, no obstante, se implementará un *front-end* básico, pero completamente funcional con la finalidad de que los profesores puedan hacer uso y evaluar la aplicación.

Entrando en más detalle sobre la estructura del *back-end*, destacar que se utilizará una arquitectura por capas, donde cada una de ellas dispondrá de una funcionalidad y responsabilidad diferente. Esta arquitectura viene definida por el *framework* que se va a utilizar, Devonfw que se describe en la sección de tecnologías del capítulo siguiente.

Respecto al almacenamiento de datos, se van a utilizar dos tipos diferentes de almacenamiento. Por un lado, se dispondrá de una base de datos relacional, la cual gestionará toda la información de los usuarios y metadatos de la aplicación. Por otro lado, se dispondrá de un repositorio de documentos donde se almacenarán los exámenes de los alumnos.

6.1 Modelo Cliente Servidor

Como se ha comentado, se utilizará una arquitectura cliente/servidor [13], la cual permite realizar una división de tareas. Mediante esta arquitectura, el procesamiento de la información se distribuye de una forma cooperativa, en la que el cliente o demandante de recursos, solicita al servidor la información que se necesita.

Al hacer uso de este modelo, se consigue que estas dos partes trabajen de forma totalmente desacoplada, lo que simplifica su desarrollo y mantenimiento.

En cuanto al cliente, es el responsable de efectuar las llamadas al servidor con la finalidad de recoger la información necesaria para mostrar correctamente la información en las interfaces gráficas de usuario. Además, también es el responsable de construir y gestionar dichas interfaces gráficas, garantizando una buena experiencia de uso para el usuario final.

En cuanto al servidor, ofrecerá servicios y funcionalidades que podrán ser localizadas y consumidas mediante peticiones del cliente. El servidor se encargará de procesar todas aquellas operaciones que sean complejas o representen una gran carga computacional. Este comportamiento es debido a que el administrador de la aplicación puede elegir la potencia de cómputo del servidor . Incluso puede ser alojado en sistemas distribuidos donde la potencia y capacidad puede escalar de forma rápida y sencilla.

Todo aquel sistema que emplee esta arquitectura seguirá el esquema que se lista a continuación:

1- El cliente envía al servidor, mediante el protocolo HTTP, la petición junto con la información necesaria.
2- La solicitud es procesada por el servidor.
3- La respuesta sobre la solicitud es enviada de vuelta mediante el protocolo HTTP.
4- La información de respuesta es procesada por el cliente.

En la ilustración 5 se puede observar una representación del esquema descrito:

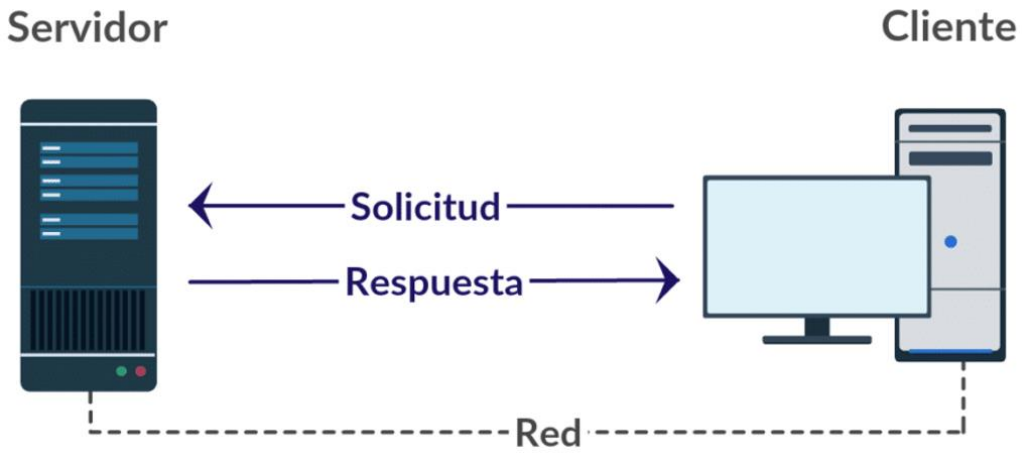


Ilustración 5 Modelo Cliente Servidor

Haciendo uso de este modelo, se dispone de un control centralizado de los datos y el procesamiento de la información, ya que todos los accesos a los recursos son controlados por el servidor. Además, hace mucho más fácil el mantenimiento e integración de nuevas tecnologías y funcionalidades, pudiendo de esta forma actualizar y escalar con mayor facilidad.

6.2 Arquitectura por capas

En esta sección se detallará la arquitectura del servidor de la aplicación y se explicará por qué se ha elegido esta arquitectura.

En primer lugar, se ha hecho uso del patrón de desarrollo de n-capas, o también conocida como *Oninon architecture* [14]. Esta arquitectura va a permitir la división de la aplicación en distintas capas, las cuales van a facilitar y simplificar la separación de responsabilidades. Además, también ayudará a la reutilización de código y de esta forma evitar su repetición innecesaria.

Esta arquitectura usa interfaces para la comunicación entre capas, definiendo un contrato entre capas. Esto evita el acceso a implementaciones específicas de capas interiores protegiendo de esta forma el código de cada una de ellas.

Cada capa dispone de una serie de funcionalidades independientes, dichas funcionalidades se clasificarán en base a qué tratamiento de los datos se está realizando. Cuanto más directo es el trato con los datos pertenecientes a las entidades de la aplicación, más interna será la capa en la que residan sus funcionalidades.

Este tipo de arquitectura proporciona una gran flexibilidad a la hora de implementar proyectos, y a su vez ayuda a los desarrolladores a crear código limpio y mantenible en el tiempo.

6.2.1 Capas de la arquitectura

La ilustración 6 muestra una representación gráfica de las distintas capas las cuales, se van a detallar con profundidad.

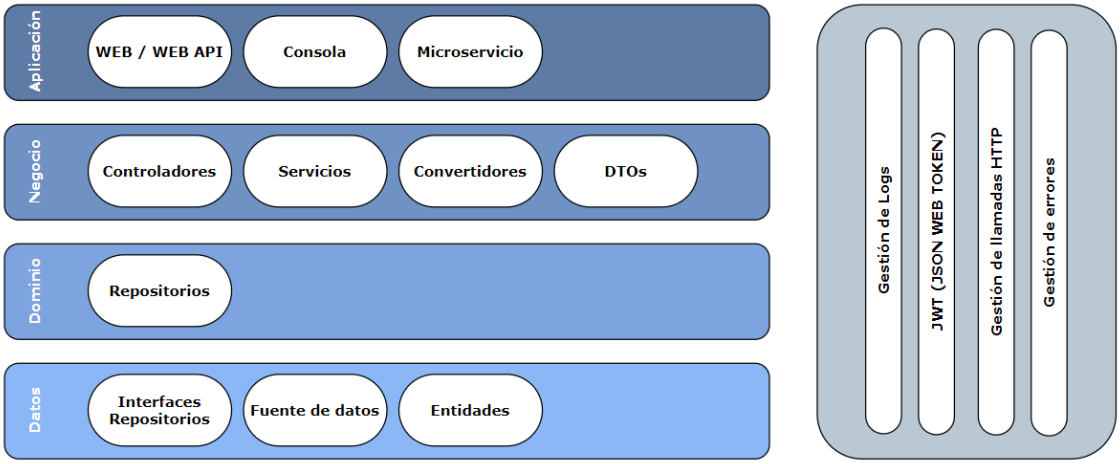


Ilustración 6 Arquitectura de N capas

Capa de aplicación:

Esta capa es la responsable de gestionar la interacción directamente con el cliente, esta recibirá sus peticiones y se comunicará con la capa inmediatamente inferior. También es la responsable de exponer todos y cada uno de los servicios que ofrece el servidor con la finalidad de que sean utilizados por el cliente.

Con esta capa de aplicación se comunican otras entidades como pueden ser: el repositorio Nexus, los reportes, las aplicaciones móviles, cualquier otra aplicación que quiera hacer uso de los servicios proporcionados por el *back-end*.

Capa de negocio:

Esta capa es la responsable de alojar la funcionalidad central de la aplicación, es decir, las reglas necesarias para poder abordar el problema para el que se ha construido la aplicación. Esta capa además de gestionar la lógica de negocio transforma los datos de DTO a entidad y viceversa para poder realizar las llamadas necesarias a sus capas contiguas.

Capa de dominio:

Esta capa es la responsable de gestionar las operaciones y la lógica de los datos de la aplicación. De esta forma, esta capa implementará todos los métodos necesarios para poder satisfacer todas las necesidades de su capa superior (capa de negocios) y poder cumplir con todas las reglas de negocio.

Capa de datos:

Esta capa es la responsable de gestionar el acceso a los datos de la aplicación. La principal fuente de datos de las aplicaciones suelen ser las bases de datos, pero podrían existir otro tipo de fuentes.

6.2.1 Componentes transversales

La arquitectura que se utilizará además de la organización de capas previamente descrita dispone de una serie de componentes transversales, es decir que se aplican en todos los niveles de la aplicación. Algunos de ellos son:

Gestión Logs:

Se ha incluido una gestión personalizable de los logs de la aplicación, haciendo sencillo el uso de estos a cualquier nivel. Mediante este componente los desarrolladores pueden escribir logs en cuatro niveles diferentes: depuración, información, error, fatal. Además, se podrá configurar cual será el nivel que se desea mostrar a la hora de presentar estos logs. En cuanto a la forma de almacenar los logs, este componente está preparado para almacenarlos en distintos lugares como documentos de texto, base de datos, o enviarlos a servicios externos como podría ser Graylog, uno de los gestores de logs más usados actualmente.

JWT:

El componente de JWT o Jason Web Token permite a los desarrolladores configurar y trabajar con esta herramienta de forma sencilla. Algunos de los parámetros configurables son: tiempo de validez del token, secreto del token, tipo de encriptación, entre otros. Además, también dispone de un manejador del token haciendo fácil la creación de nuevos tokens o desencriptar los tokens recibidos y extraer la información necesaria.

Manejador de llamadas HTTP:

Se dispone de un manejador de llamadas HTTP mediante el cual los desarrolladores pueden lanzar peticiones HTTP a distintos servicios. Este componente soporta todos los métodos de HTTP, entre los cuales se encuentran GET, POST, DELETE, PUT. Se pretende a través de este componente que se unifique la forma de realizar las llamadas HTTP, creando de esta forma un código mucho más limpio y reutilizable.

Manejador de errores:

El componente de manejo de errores permite a los desarrolladores gestionar de una forma eficiente los errores que se produzcan durante la ejecución de las aplicaciones. Dispone de una gestión de errores imprevistos, esto significa que protegerá a las aplicaciones de cierres inesperados o errores que detengan la ejecución de la aplicación.

Además, este componente, ofrece la posibilidad de crear excepciones personalizadas en las que se define el código de respuesta de la petición junto con un mensaje. El componente creará un log con dicho mensaje y enviará al cliente un mensaje con el código de respuesta establecido.

Este manejador de errores favorece la creación de un código limpio y con buenas prácticas ya que apenas será necesario crear bloques de *try catch* [36], ya que este manejador se encargará de gestionar los errores.

6.3 Base de datos relacional

La aplicación que se pretende implementar utilizará de una base de datos relacional para almacenar información referente a los usuarios y los metadatos de los exámenes. En esta sección se detallarán las ventajas de este tipo de datos y su estructura.

En cuanto al origen de este modelo de base de datos, el modelo relacional, fue creado por IBM [15]. Este modelo almacena los datos en tablas, las cuales están compuestas por filas y columnas y disponen de una lógica de predicados los cuales establecen relaciones entre los datos almacenados.

Respecto de las ventajas ofrecidas por este modelo de datos, se puede decir que facilita la comprensión de la información almacenada por los desarrolladores inexpertos o personas

no técnicas involucradas en el proyecto. Además, este modelo también dispone de herramientas que permiten evitar duplicidad de los registros. Asimismo, haciendo uso de este modelo, se garantiza la integridad referencial, ya que cuando se elimina un registro, se eliminan todos los registros dependientes que estén relacionados con éste.

Se ha utilizado un diagrama de entidad-relación para el diseño de todas y cada una de las tablas de la base de datos. Este diagrama reúne tanto las entidades como las distintas relaciones que existen en el sistema que se va a implementar. Permitirá comprender de una forma cómoda y sencilla el modelo de datos del sistema, ya que este esquema representará al completo la base de datos de la aplicación, representando por tanto todos sus datos y las distintas relaciones que existen entre ellos.

En la ilustración 7 se muestra el diagrama entidad relación que se ha empleado en el diseño de la aplicación:

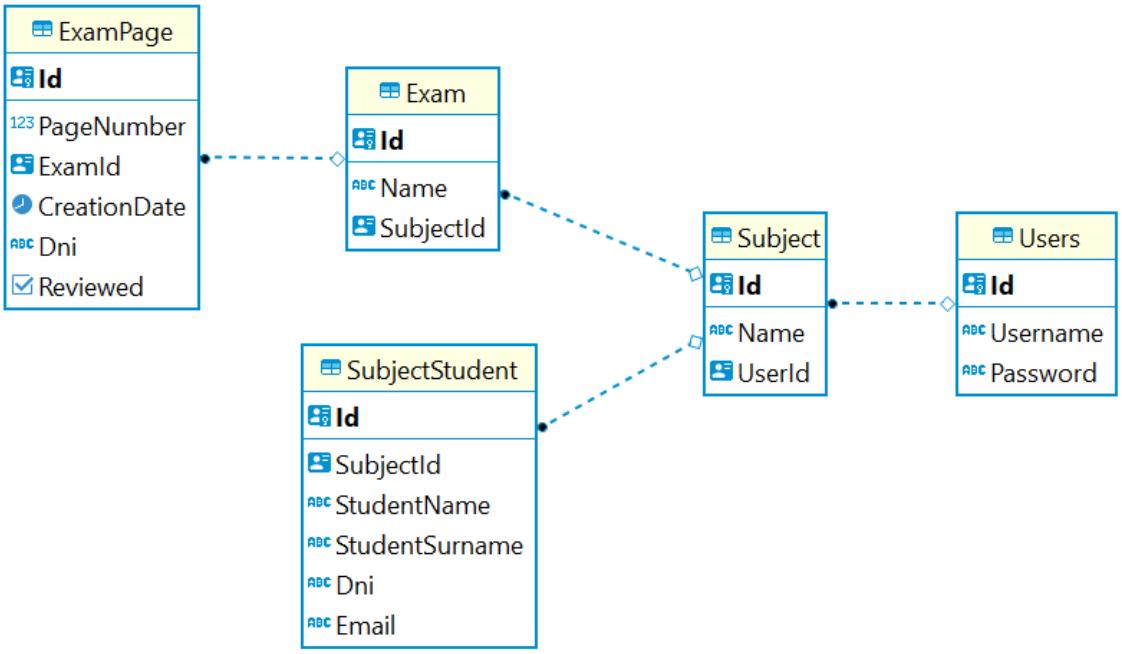


Ilustración 7 Diagrama entidad relación

Como se puede observar en el diagrama de la figura 7, este modelo proporcionará soporte para la asignación de asignaturas para cada uno de los usuarios. A su vez, a estas asignaturas se podrán asignar exámenes y de la misma forma se podrán asignar alumnos. Por último, a cada examen se podrán asignar páginas de exámenes.

Con la finalidad de establecer todas estas relaciones entre las distintas tablas, se han definido como claves primarias todos los identificadores de cada una de las tablas y a su vez se han definido como claves ajenas aquellos atributos con la terminación Id de las tablas. A modo de ejemplo para facilitar la comprensión de estas relaciones, el atributo de la tabla Subject, UserId, es clave ajena y se encuentra relacionada directamente con la clave primaria de la tabla Users la cual su atributo es Id.

A continuación, se detalla la funcionalidad de cada una de las tablas mostradas en la ilustración 7:

- Users: Esta tabla almacenará la información referente a los usuarios que se registren en la aplicación.
- Subject: Esta tabla almacenará la información perteneciente a las asignaturas creadas por los usuarios de la aplicación.
- SubjectStudent: Esta tabla almacenará la información de los estudiantes asignados a las asignaturas.
- Exam: Esta tabla almacenará la información referente a los exámenes creados por los usuarios.
- ExamPage: Esta tabla almacenará la información perteneciente a los metadatos de cada una de las páginas de los exámenes evaluados que han sido subidos por el usuario.

6.4 Almacenamiento de documentos

Como se ha comentado en anteriores puntos de esta memoria, no solo se utilizará una base de datos, sino que además se empleará un repositorio de documentos para almacenar las páginas de los exámenes de los alumnos Para ello se hará uso de Nexus [16], que permite almacenar de una forma ordenada todos estos documentos. Esta herramienta se describe con más profundidad en el siguiente capítulo.

La gestión de estos documentos o exámenes en el repositorio Nexus será realizada al completo por el servidor de la aplicación, siendo totalmente transparente para el usuario final.

Nexus hará uso de los identificadores únicos creados y gestionados por la base de datos para organizar la información de todos y cada uno de los exámenes de la aplicación en el repositorio.

La conexión entre este repositorio y el servidor se realizará a través de llamadas HTTP, ya que Nexus expone una serie de funcionalidades y servicios a través de una API.

En cuanto a la seguridad, almacenar estos exámenes en este repositorio es completamente seguro, ya que es necesario un usuario y contraseña para acceder a su contenido. El servicio hará uso de una combinación de usuario y contraseña creados específicamente para este uso. Es importante destacar que la gestión de estas credenciales es realizada completamente por el servidor, siendo un sistema de almacenaje totalmente transparente para el usuario final.

6.5 Formatos de documentos

Como se ha visto en puntos anteriores, esta aplicación va a ser capaz de importar una serie de documentos, los cuales deben presentar un formato específico para un correcto procesamiento de los datos. En los siguientes puntos se detalla el formato especifico de cada uno de los tipos de documentos que utilizará la aplicación.

6.5.1 Asignación de alumnos a una asignatura

La aplicación permitirá importar un documento que contenga el listado de los alumnos y de esta forma poder asignarlos a una asignatura concreta. El formato de este documento va a ser de tipo CSV [17].

Un archivo CSV (valores separados por comas) es un archivo de texto plano que contiene una lista de datos. Es un formato de archivo muy utilizado a la hora de intercambiar información entre diferentes aplicaciones. Respecto a la estructura de este tipo de archivos es bastante simple, ya que se trata de una especie de tabla donde los datos de cada columna están separados por comas. Este tipo de archivo puede contener en la primera fila unos encabezamientos o títulos de cada una de las columnas, pero cabe destacar que es una característica opcional.

Para ser más concretos en la estructura del archivo CSV que se espera recibir a la hora de asignar estudiantes a una asignatura, no se deberá especificar ninguna cabecera, y los datos deberán ser escritos en el siguiente orden:

- Apellidos
- Nombre
- DNI
- Email

Este formato es el que genera PoliformaT con la herramienta Gestión, obtención de orlas y listas, opción Fichero (CSV). Un ejemplo válido de dos alumnos sería:

- Martí Campoy, Antonio,111111111,example1@upv.es
- Ferrandis Jorge, Josep,222222222,example2@upv.es

6.5.2 Exámenes evaluados y escaneados por el profesor

La aplicación permitirá subir los exámenes escaneados de las evaluaciones realizadas por el profesor con la finalidad de que el sistema separe cada uno de los exámenes y lo clasifique reconociendo el DNI escrito por el alumno. Estos exámenes escaneados tienen que ser necesariamente de tipo PDF. El sistema es capaz de añadir nuevos exámenes en el caso de subir un examen por PDF, o de separar cada uno de los exámenes si se suben múltiples exámenes en un mismo PDF. También es posible subir múltiples ficheros con todo o parte de uno o varios exámenes, siempre y cuando las hojas escaneadas y los ficheros estén ordenados.

6.5.3 Plantilla exámenes

Para que la aplicación pueda identificar la primera y última hoja de un examen, así como intentar reconocer el DNI del alumno, las cabeceras de las hojas de examen deben cumplir unas condiciones.

El funcionamiento de la aplicación a la hora de separar los distintos exámenes se basa en el reconocimiento en la cabecera de la primera página de cada examen de la combinación de las palabras “Examen” y “DNI” o “Examen” y “Nombre”. De esta forma estas combinaciones de dos palabras deberán aparecer en la cabecera de la primera página de todos los exámenes. Es importante que en las demás páginas del examen no aparezcan

ninguna de estas combinaciones de dos palabras. Además, es recomendable que el campo del DNI a rellenar por el alumno esté en la parte derecha de la cabecera del examen. A modo de ejemplo se puede ver una plantilla bien formada en el anexo 2 de la memoria.

7. Implementación

A lo largo de este capítulo, se detallará toda la información referente a la implementación del proyecto. Para ello, se empezará hablando de las tecnologías y herramientas empleadas y posteriormente se detallará información referente a metodologías y otros aspectos.

7.1 Tecnologías y herramientas

En este apartado, como se ha mencionado anteriormente, se van a detallar las tecnologías y herramientas utilizadas durante el desarrollo del proyecto.

7.1.1 .NET y C#

.NET [18], es una plataforma que ha sido desarrollada por Microsoft, la cual permite a los desarrolladores de software crear distintos tipos de aplicaciones multiplataforma y de código abierto proporcionándoles un conjunto de herramientas, tecnologías y servicios.

.NET ofrece soporte a diferentes lenguajes de programación. Algunos de ellos son Visual Basic, C++, F#, C#, entre otros.

Para el desarrollo del *back-end* de este proyecto se ha utilizado .NET Core [19], puesto que ofrece las siguientes características:

- Multiplataforma: Permite a los desarrolladores implementar aplicaciones que van a poder ser ejecutadas en los principales sistemas operativos de la actualidad como son: Windows, Linux y MacOS.
- Código abierto: .NET Core proporciona una gran transparencia durante el proceso de implementación de la aplicación.
- Modular: .NET Core permite un modularidad total gracias a su gestión de paquetes *NuGet* el cual permite un gran control sobre las funcionalidades que se necesiten de terceros.

.NET CORE utiliza como lenguaje de programación C# por lo que este es el principal lenguaje que se ha utilizado en el desarrollo del *back-end* del proyecto.

7.1.2 Visual Studio 2019

Visual Studio 2019 [20] es un producto desarrollado por Microsoft, el cual ofrece un entorno integrado de desarrollo (IDE) de software. Este IDE ofrece soporte para diferentes lenguajes de programación, no obstante, ofrece funcionalidades especiales para los lenguajes que se utilizan para trabajar con .NET, explicado anteriormente.

Se ha tomado la decisión de utilizar este entorno de desarrollo para la parte *back-end* de la aplicación ya que como se ha comentado la tecnología empleada es .NET Core y este IDE ofrece funcionalidades y herramientas especiales para .NET. Se ha utilizado la última versión estable de esta herramienta, Visual Studio 2019.

7.1.3 Devonfw

Devonfw [21] se trata de una plataforma con un enfoque de industrialización con el fin de acelerar y agilizar los procesos de desarrollo de proyectos software, más concretamente aplicaciones web. Para conseguir este objetivo, esta plataforma cuenta con soporte para las principales tecnologías de desarrollo de aplicaciones web como pueden ser: java, node, typescript, .NET, entre otras. Esta plataforma está constituida completamente de productos de código abierto y que sigue los estándares definidos de cada tecnología, ayudando de esta forma al desarrollador a implementar aplicaciones de forma correcta y con buenas prácticas de programación.

7.1.4 REST (*REpresentational State Transfer*)

La tecnología REST [22] permite intercomunicar distintas aplicaciones utilizando el protocolo HTTP. Sus principales características son:

- Utiliza una interfaz uniforme, por lo que es necesario utilizar una URL juntamente con un método HTTP, algunos de estos métodos son:
    - GET: recoge información del servidor.
    - PUT: modifica o actualiza el estado de un recurso del servidor.
    - POST: crea un nuevo recurso en el servidor.
    - DELETE: elimina un recurso del servidor.

- Los formatos más habituales para enviar y recibir los parámetros y datos son JSON y XML, aunque también soporta otro tipo de datos como son CSV.
- Ofrece una gestión de códigos de respuesta, la cual ayuda a identificar el estado de la petición. Algunos ejemplos de códigos de respuesta podrían ser 201 (*Created*), 404 (*Not found*), 500 (*Internal Server Error*).

7.1.5 Swagger

Swagger [23] implementa el estándar OpenAPI [24] y ofrece una amplia variedad de herramientas a los desarrolladores que ayuda a diseñar, documentar, consumir y construir aplicaciones web REST mediante la generación del contrato de los servicios. Dicho contrato puede ser compartido entre los distintos desarrolladores de diferentes equipos y permitirá el desarrollo en paralelo de las partes *front-end* y *back-end* de aplicaciones. Estas herramientas ofrecen soporte a la generación automática de documentación y casos de prueba del proyecto.

Mediante Swagger, ha sido posible generar un cliente HTML de forma dinámica cada vez que se compilaba y ejecutaba la parte servidor del proyecto permitiendo la publicación de todos los servicios ofertados o puntos de acceso disponibles con el fin de que los consumidores dispongan de toda la información para realizar una conexión exitosa con cada una de las funcionalidades del servidor.

De esta forma, Swagger, ha permitido acelerar el desarrollo de la parte *front-end*, que como se ha comentado anteriormente, no era un objetivo del proyecto, pero se ha realizado una interfaz básica y completamente funcional. Esta aceleración ha tenido lugar a la hora de seguir el contrato definido y diseñado con anterioridad, esta herramienta ha proporcionado un visionado claro y rápido de todo lo necesario para comunicarse con el *back-end* del proyecto.

7.1.6 JSON WEB TOKEN

JSON Web Token (JWT) [25] es un estándar de seguridad que define una forma compacta y autónoma de transmitir información de forma segura entre cliente y servidor mediante la creación de un *token*. JWT se basa en objetos de tipo JSON. Se puede confiar en esta información ya que puede ser validada pues está firmada digitalmente.

De la misma forma que la herramienta Swagger, JSON Web Token está integrado en Devonfw. Mediante esta herramienta se gestiona el acceso no autorizado a la aplicación y protege la información de cada usuario, ya que dispondrá de un token único cada usuario.

7.1.7 Angular

Angular [26] es una plataforma desarrollada por Google de código abierto que facilita el diseño e implementación de aplicaciones web, de mediana y alta complejidad.

Angular utiliza el lenguaje *Typescript*, el cual es compatible con todos los sistemas operativos y navegadores web de hoy en día. Además, ofrece la posibilidad de validar y corregir el código desarrollado por los programadores de una forma eficiente ahorrando tiempo y trabajo al programador.

Angular ayuda al desarrollador proporcionando herramientas que inspiran a realizar código de alta calidad permitiéndole centrarse únicamente en la lógica y el comportamiento de la aplicación y evitar la repetición de código.

7.1.8 Visual Studio Code

Visual Studio Code [27] (conocido como VS Code) es un editor de texto gratuito de código abierto de Microsoft. VS Code está disponible para Windows, Linux y macOS. Aunque el editor es relativamente ligero, incluye algunas características potentes que han hecho de VS Code una de las herramientas de entorno de desarrollo más populares de los últimos tiempos.

Se ha utilizado este editor de texto para el desarrollo del *front-end*, ya que ofrece una serie de funcionalidades que agilizan y aceleran el proceso de desarrollo de este tipo de aplicaciones.

7.1.9 PostgreSQL

PostgreSQL [28] es un potente sistema de base de datos relacional de código abierto que utiliza y amplía el lenguaje SQL combinado con muchas características que almacenan y escalan con seguridad. PostgreSQL permite el manejo de cargas de trabajo que abarcan

desde pequeñas aplicaciones de una sola máquina hasta aplicaciones empresariales donde existen multitud de máquinas y muchos usuarios que se conectan de forma simultánea.

7.1.10 SQL

SQL [29] es un lenguaje de programación que permite una gestión total sobre las bases de datos relacionales. Mediante este lenguaje se realiza el acceso a los datos, permitiendo la creación, borrado y modificación de tablas, junto con la inserción, consumición, borrado y actualizado de los datos de las propias tablas y entidades.

7.1.11 DBeaver

DBeaver [30] es una herramienta gratuita de gestión de bases de datos multiplataforma para desarrolladores, administradores de bases de datos, analistas y cualquier usuario que necesite trabajar con bases de datos. Soporta todas las bases de datos populares, entre ellas PostgreSQL, motor de base de datos que se ha elegido para esta aplicación.

7.1.12 Nexus Repository

Nexus Repository [16] es una herramienta de código abierto que ofrece la posibilidad de almacenar y gestionar una gran variedad de archivos de diferentes formatos. En este trabajose utiliza para el almacenamiento de las imágenes de cada página de los exámenes de los alumnos. Nexus ofrece una forma sencilla de conectar con sus funcionalidades de gestión de archivos mediante su API. Además, Nexus ofrece soporte para poder ser ejecutado en contenedores Docker, forma en la que se ha decidido utilizar este repositorio.

7.1.13 Tesseract Open-Source OCR Engine

Tesseract [9] es un motor de reconocimiento óptico de caracteres o OCR para varios sistemas operativos. Es un software libre, publicado bajo la licencia Apache, desarrollado originalmente por Hewlett-Packard.

Gracias a su popularidad, existen implementaciones en una gran variedad de tecnologías, entre ellas .NET, tecnología que se ha utilizado para la implementación del servicio de la aplicación.

7.1.14 Postman

La herramienta Postman [31] permite a los desarrolladores crear peticiones a distintas APIs o servidores, ya sean implementados por ellos mismos o de terceros. La finalidad de utilizar esta herramienta es poder gestionar y probar las funcionalidades de dichas APIs.

Más concretamente, se ha utilizado Postman para realizar las pruebas de uso de la aplicación. También se ha utilizado para entender y comprender el funcionamiento de la API de Nexus que como se ha explicado anteriormente es el repositorio de documentos que se ha elegido para almacenar las páginas de los exámenes.

Postman ha permitido ejecutar todas y cada una de las llamadas al servidor y de esta forma probar todas las funcionalidades que se establecieron durante la fase de análisis de los requisitos y casos de uso.

7.1.15 Docker

Docker [32] es una plataforma de código abierto que es utilizada para desarrollar, enviar y ejecutar aplicaciones. Docker permite separar las aplicaciones de su infraestructura para que pueda entregar el software rápidamente, esto significa que se va a poder ejecutar la misma aplicación en distintas máquinas garantizando siempre el mismo comportamiento. Esto se consigue gracias a la virtualización, ya que Docker crea una capa por encima del sistema operativo la cual permite ejecutar las aplicaciones sin depender de éste.

Mediante Docker se generarán paquetes o como son llamados, imágenes, por cada una de las partes de la aplicación, de esta forma podrán ser ejecutadas en cualquier sistema operativo que disponga de Docker instalado en su máquina. Esto ofrece una gran versatilidad ya que permite una rápida instalación y puesta en marcha de las aplicaciones desarrolladas.

7.1.16 Firefox y Chrome

Mozilla Firefox [33] y Google Chrome [34] son dos de los principales navegadores en la actualidad. A lo largo de este proyecto se han utilizado ambos tanto para mostrar los contratos generados por la herramienta Swagger como para la visualización del *front-end* de la aplicación.

7.2 TDD (*Test-driven-development*)

La práctica de programación *Test-driven-development* o como es comúnmente conocida, TDD, se centra en la programación dirigida por casos de prueba. Para llevar a cabo esta práctica, se ha hecho uso del libro *The Art of Unit Testing* [35], del cual se ha obtenido todo el conocimiento necesario para utilizar esta metodología. Con el fin de utilizar TDD se han diseñado todos los casos de prueba junto con su correspondiente implementación antes de la realización del código fuente. Esta práctica se divide en tres partes, las cuales se detallan a continuación.

La primera de ellas es el diseño e implementación del caso de prueba que se pretenda abordar, para ello se centrará el foco en el requisito o funcionalidad que se quiera cumplir. En esta parte se va a hacer uso de las pruebas unitarias, esto significará que todos y cada una de las pruebas deberán cumplir los siguientes aspectos: deberán ser automáticas y repetibles, en cuanto a su implementación debe ser simple, fácil y rápida, en cuanto a su ejecución, debe ser rápida y en cuanto a resultados de respuesta deben devolver siempre el mismo valor a partir de la misma entrada. Todos los test deben ser autónomos y aislados de las demás pruebas y todos los casos de prueba deben ser válidos tanto para el presente del proyecto como para el futuro.

Una vez se dispone de todos los casos de prueba diseñados e implementados se procederá con la implementación del código fuente con el único objetivo de ir cumpliendo todos y cada uno de los casos de prueba. Pruebas que previamente han sido diseñadas para cumplir todos y cada uno de los requisitos establecidos. En esta parte se consigue implementar la funcionalidad estrictamente definida por el caso de prueba, de esta forma es muy importante destacar y recordar la importancia del diseño de estos casos de prueba.

Como última etapa o fase de esta práctica, se encuentra la refactorización, esto significa que será necesario realizar una revisión del código escrito que supera el caso de prueba de forma satisfactoria con el objetivo de optimizar su funcionamiento y limpiar el código. Mediante este proceso, se centra el foco en una unidad pequeña de código, ya que solo se revisará el caso pertinente a una prueba unitaria. De esta forma la refactorización y optimización del código es mucho más sencilla y liviana puesto que reduce la cantidad de código a revisar. No obstante, a esta última parte también es necesaria la adición de un análisis del código haciendo uso de preguntas como: ¿Se podrá entender este código

dentro de un año? o ¿Esta función se podría implementar de una forma más simple o eficiente?

A continuación, la ilustración 8, muestra el diagrama de flujo que se ha utilizado a la hora de aplicar TDD en el proyecto.

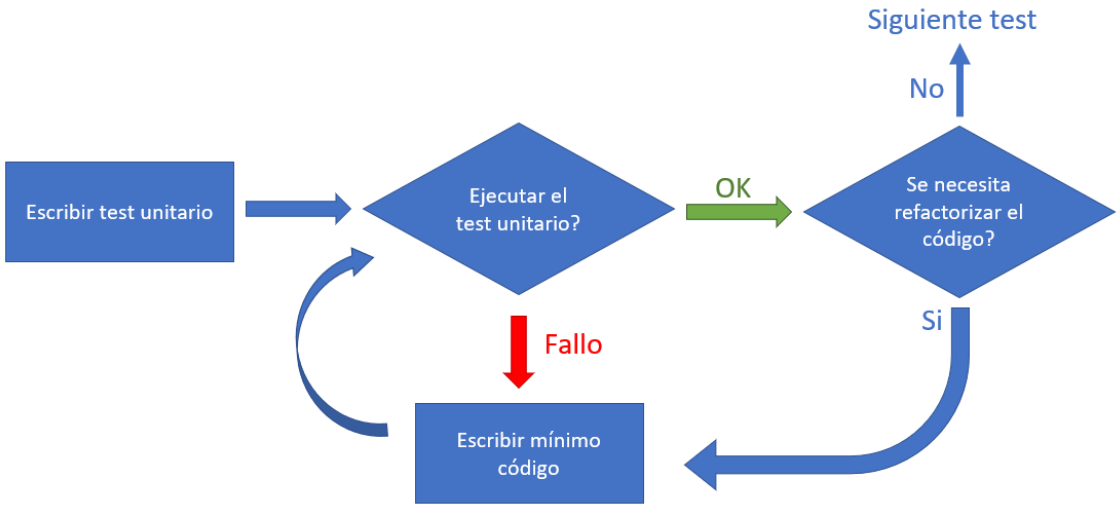


Ilustración 8 Esquema metodología TDD

Aplicando esta práctica, el desarrollador asegura que el código fuente implementado funciona de una forma determinada ante ciertos casos, ya que se respalda de unos resultados satisfactorios en los casos de prueba diseñados y así mismo garantiza que cumple con los requisitos y funcionalidades requeridas.

Como se ha comentado anteriormente este proyecto ha sido desarrollado en su totalidad utilizando la práctica de TDD, con ello se ha conseguido una lista con todos los casos de prueba que comprobarán todas las funcionalidades que se han concretado durante el análisis de requisitos. A su vez, se ha conseguido obtener un código óptimo y limpio ya que tras haber superado cada uno de los casos de prueba, se ha empleado tiempo en el análisis del código haciéndolo óptimo.

La ilustración 9 muestra a modo de ejemplo uno de los casos de prueba implementados con el fin de detallar la estructura y funcionalidad de este.

```
[Fact]
| 0 references | 0 changes | 0 authors, 0 changes
public async Task CreateNewExamWithCorrectDataShouldCreateANewExam()
{
    //Arrange
    var subjectId = new Guid("0f9283e6-5da1-4044-b699-7923cd004cd6");
    var inputDataExam = new ExamDto { Name = "Example", SubjectId = subjectId };
    var examExpected = new ExamDto { Name = "Example", SubjectId = subjectId };

    //Act
    var examResult = await _examService.CreateExam(inputDataExam).ConfigureAwait(false);

    //Assert
    Assert.Equal(examExpected, examResult);
}
```

Ilustración 9 Ejemplo de test implementado

El caso de prueba que se aprecia en la ilustración 9 consta de tres partes, *Arrange*, *Act* y *Assert*. La primera de ellas es donde se crean tanto los datos de entrada de la prueba como otros datos que serán necesarios para el cumplimiento de ésta. En la segunda sección se realiza la llamada al servicio proporcionando los datos creados en la sección anterior. En la última sección es donde se validan los datos obtenidos de la segunda parte, aquí pueden existir distintas validaciones, tantas como sean necesarias para verificar el cumplimiento del caso de uso que se desee probar.

Es importante destacar que se ha seguido una nomenclatura para obtener un código legible y de fácil mantenimiento en el tiempo. Se puede observar en la ilustración 9 que para el objeto esperado se utiliza la terminación *Expected* con la finalidad de diferenciarlo del objeto obtenido por el servicio. De la misma forma el objeto resultante del servicio recibirá la terminación de *Result*. Además, el nombre de cabecera del test también sigue la siguiente formulación, *{Función bajo prueba}With{condiciones de la prueba}Should{Resultado esperado de la prueba}*. Mediante esta nomenclatura se garantiza la unificación de los nombres en todas y cada una de las pruebas.

7.3 Seguridad

En esta sección se detallarán los distintos aspectos referentes a la seguridad que han sido implementados en la aplicación. Como ya se ha comentado anteriormente, la aplicación hace uso de JWT [25] a la hora de comunicar datos sensibles entre el cliente y el servidor. JWT es una herramienta que se encuentra en una gran variedad de tecnologías, esta es una de las principales razones por la que se ha utilizado, ya que ofrece versatilidad en el

tipo de cliente que quiera conectarse con el servidor desarrollado. Mediante esta herramienta el servidor podrá crear tokens únicos y de un tiempo de validez finito en el que se recopilará información sensible de la aplicación.

Con el fin de ser un poco más concretos en cuanto al uso de esta herramienta en la aplicación desarrollada, es importante comentar que se crea un token por cada inicio de sesión con el cual se almacena el Id del usuario en cuestión. Este Id es necesario para que la aplicación recupere solamente la información perteneciente a dicho usuario. A su vez este token tendrá una validez de una hora por lo que pasado este tiempo el usuario deberá iniciar sesión nuevamente. El token viajará en la cabecera de la petición al servidor.

Además, se ha creado una gestión de roles con la finalidad de aumentar la seguridad y proporcionar opción de ampliar funcionalidades futuras. Actualmente sólo existe el rol *Professor*.

En los controladores de la aplicación residen las comprobaciones necesarias de seguridad, este es el punto donde se evalúa si el token es válido y si el usuario dispone del rol correcto para acceder a la funcionalidad. La ilustración 10 muestra un ejemplo de esta comprobación:

```
[HttpGet]
[Authorize(AuthenticationSchemes = AuthConst.AuthenticationScheme, Roles = AuthConst.Professor)]
[ProducesResponseType(typeof(List<SubjectDto>), StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
[ProducesResponseType(StatusCodes.Status404NotFound)]
[ProducesResponseType(StatusCodes.Status500InternalServerError)]
0 references | JosepFe, 48 days ago | 1 author, 2 changes
public async Task<ActionResult> GetSubject()
```

Ilustración 10 Ejemplo comprobación rol de usuario

Mediante la línea enmarcada en rojo, la aplicación es capaz de identificar el rol del usuario, ya que descifra el token y obtiene la información necesaria para realizar dicha comprobación.

Para poder descifrar el token y obtener la información almacenada dentro de él se ha utilizado la función que se muestra en la ilustración 11, que recupera el Id del usuario que realiza la petición.

```
private Guid GetUserId(HttpRequest httpRequest)
{
    var token = httpRequest.Headers["Authorization"].ToString().Replace($"{AuthConst.AuthenticationScheme} ", string.Empty);
    var userClaims = JwtHandler.GetUserClaims(token).ToList();
    return new Guid(userClaims.FirstOrDefault(x => x.Type == "UserId").Value);
}
```

Ilustración 11 Método para obtención id del usuario del token JWT

De esta manera se garantiza el acceso autorizado de todos los usuarios ya que cada uno de ellos dispondrá de un token único con un tiempo de vida limitado el cual contendrá la información sensible de cada uno de ellos. La aplicación denegará el acceso a la funcionalidad cuando una de las diferentes cláusulas de seguridad no se cumpla.

7.4 Estructura de la aplicación

Este apartado detalla la organización y estructura de carpetas que se ha utilizado durante la implementación del proyecto. En primer lugar, destacar que se ha utilizado una arquitectura por capas como se ha explicado anteriormente, esto ha tenido un impacto en cuanto a la organización ya que se ha dividido en cuatro capas bien diferenciadas, estas capas son: capa de aplicación, capa de negocio, capa de dominio y capa de datos.

La capa de aplicación dispone de toda la configuración del servidor, es el lugar en el que se definen y se establecen cada una de las opciones que ofrece. Entre las configuraciones más importantes se pueden encontrar la configuración de Swagger, JWT, el manejador de errores, los logs de la aplicación, la base de datos, entre otras muchas más. Como se ha explicado anteriormente es el punto más exterior de la aplicación por lo que es la capa encargada de realizar la redirección de las peticiones que lleguen al servidor al controlador que corresponda de la capa de negocio.

Respecto a la capa de negocio, se ha implementado haciendo uso de carpetas de gestiones, estas contienen todos aquellos elementos necesarios para la correcta gestión de cada entidad. Se ha creado una carpeta de gestión por cada entidad de la aplicación.

Cada una de estas carpetas de gestión de entidades dispone de tres carpetas, la primera de ellas la carpeta *Controller* que aloja el controlador de la entidad que se pretende gestionar, éste es el encargado de recibir y responder las distintas peticiones que el cliente r sobre el servidor. Además, el controlador se comunicará con la capa de servicio, la cual se encuentra ubicada en otra subcarpeta que recibe el nombre de *Services*. Estos servicios

disponen de los métodos CRUD (*Create, Read, Update, Delete*), los cuales permiten realizar las operaciones más básicas sobre estas entidades. Estas operaciones son: crear nuevas entidades, modificar entidades existentes, recuperarlas o borrarlas. No obstante, también se encuentran las funcionalidades específicas que se requieran de cada entidad. Otra de las subcarpetas que se encuentra en esta carpeta de gestión es la de *Converter*, que realiza la conversión de objetos entre las entidades de la aplicación y los objetos de transferencia de datos o DTOs. Estos últimos son los objetos que se utilizan para comunicarse con el cliente. Por último, se encuentra la carpeta de DTO, que es donde se alojarán estos objetos mencionados anteriormente, En la ilustración 12 se muestra uno de los gestores de los que dispone la aplicación.

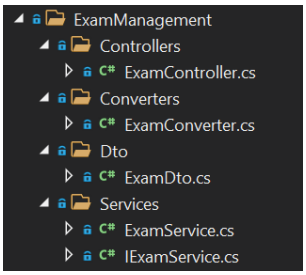


Ilustración 12 Estructura carpetas de gestión

La siguiente capa que se va a detallar es la capa de dominio la cual aloja los repositorios de cada una de las entidades, en ellas se encuentran todas las operaciones de bases de datos que se realizarán en la aplicación. De una forma similar a la anterior gestión, se ha creado un archivo por cada entidad de la base de datos donde cada uno aloja sus propias operaciones. La ilustración 13 muestra dicha estructura:

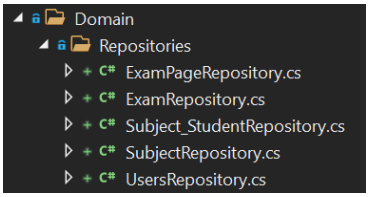


Ilustración 13 Estructura carpetas de Dominio

Por último, está la capa de datos, en la cual se encuentran ubicadas las entidades de la aplicación juntamente con las interfaces de los repositorios y el contexto de la base de datos. Este último contexto es la representación de la base de datos dentro de la aplicación web. Se puede observar dicha estructura en la ilustración 14:

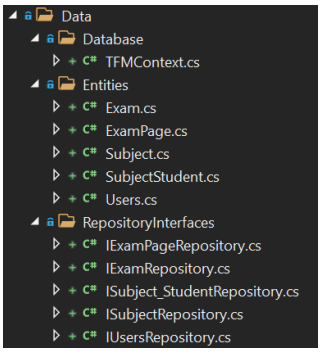


Ilustración 14 Estructura carpetas de datos

7.5 Servicios implementados

En esta sección se describen todos los servicios que se han implementado durante el desarrollo de la aplicación con la finalidad de poder utilizar todas y cada una de las funcionalidades requeridas. Para ello se van a mostrar una serie de ilustraciones en las cuales se podrá observar el *endpoint* del servicio junto con su contrato. Cabe mencionar que todas las ilustraciones que se van a mostrar a continuación han sido obtenidas haciendo uso de la generación de documentación que ofrece la herramienta Postman, mediante la cual se ha confeccionado una colección de todas las peticiones HTTP que ofrece el *back-end* de la aplicación que se ha desarrollado. Esta colección de peticiones es la que se ha utilizado en el capítulo siguiente donde se abordarán las pruebas realizadas a la aplicación.

La ilustración 15 muestra el *endpoint* y el contrato para registrar nuevos usuarios en el sistema:

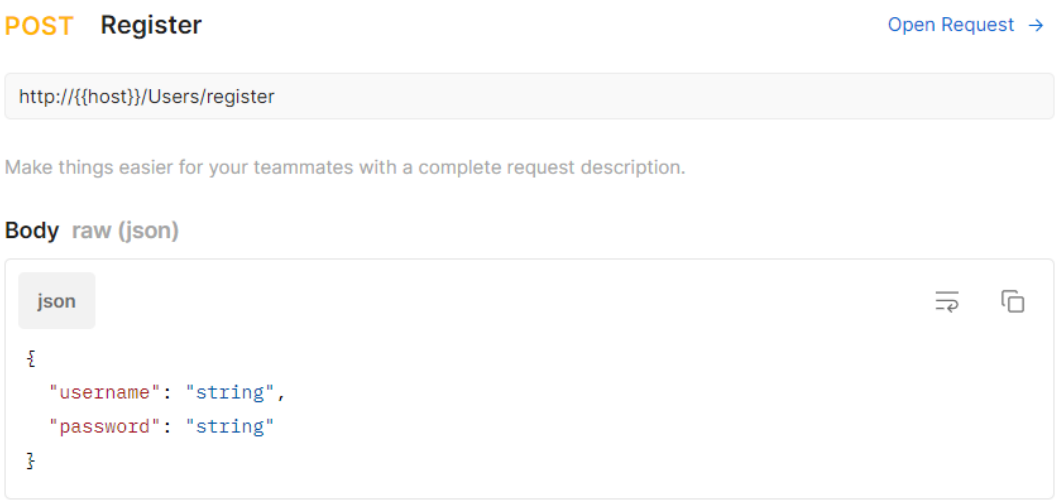


Ilustración 15 Registrar nuevos usuarios

La ilustración 16 muestra el *endpoint* y el contrato para iniciar sesión en el sistema:

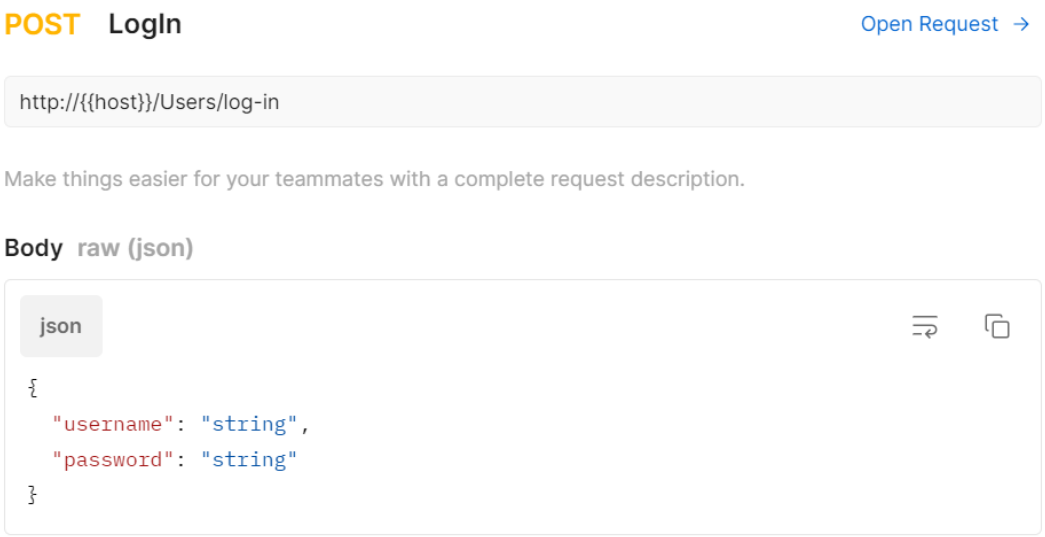


Ilustración 16 Iniciar sesión

La ilustración 17 muestra el *endpoint* y el contrato para la creación de una nueva asignatura:

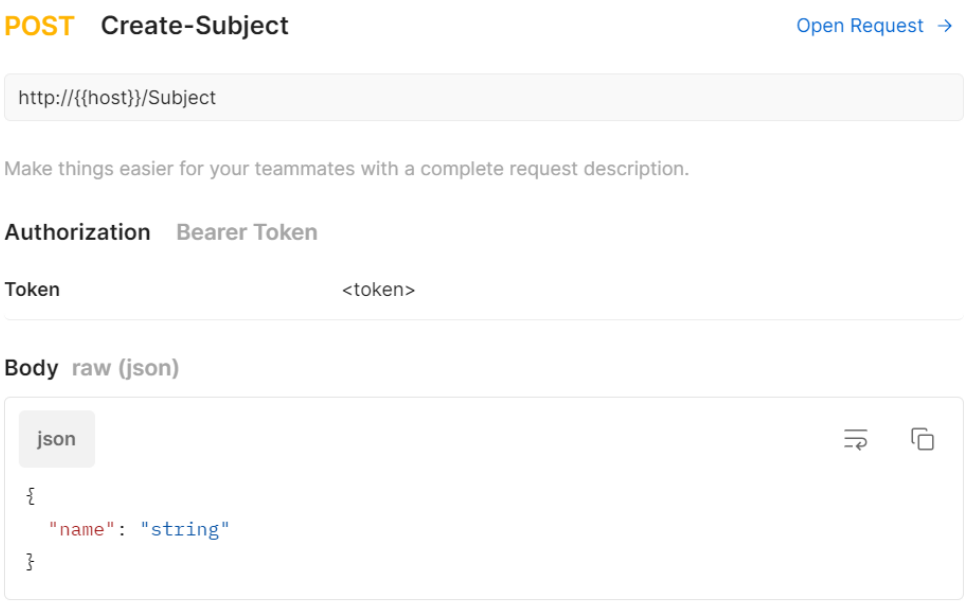


Ilustración 17 Creación de una nueva asignatura

La ilustración 18 muestra el *endpoint* y el contrato para listar todas las asignaturas disponibles para un usuario determinado:

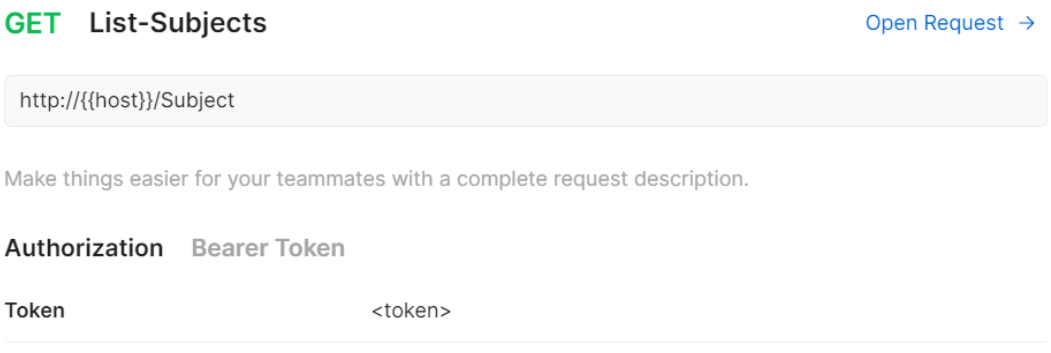


Ilustración 18 Listar todas las asignaturas disponibles

La ilustración 19 muestra el *endpoint* y el contrato para la creación de un nuevo examen en el sistema:

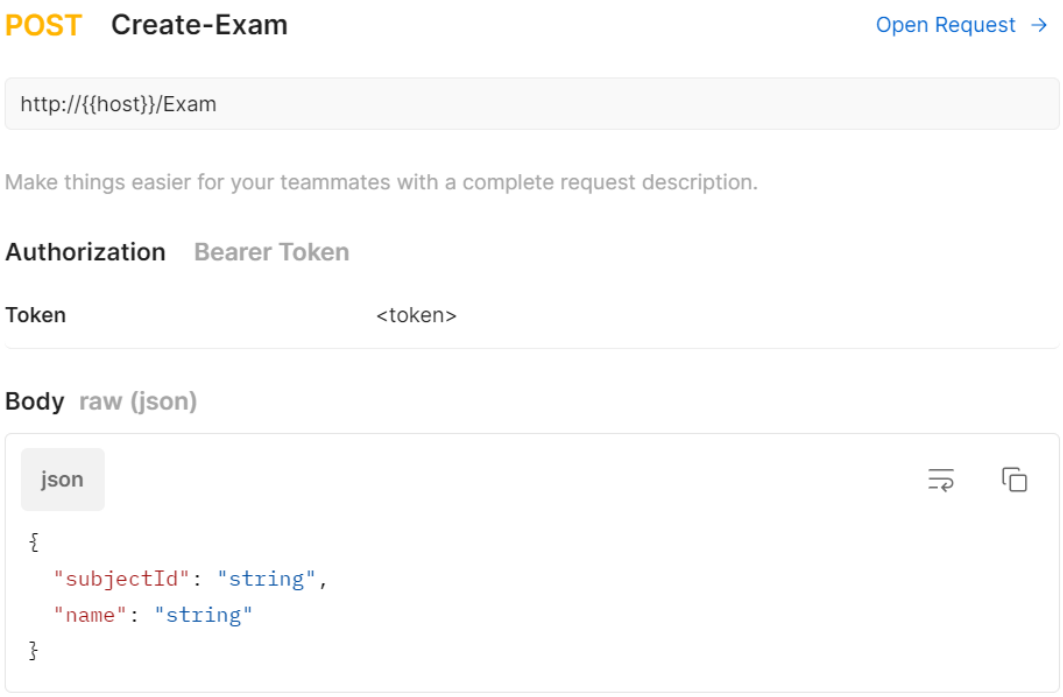


Ilustración 19 Crear un nuevo examen

La ilustración 20 muestra el *endpoint* y el contrato para listar todos los exámenes de una asignatura:

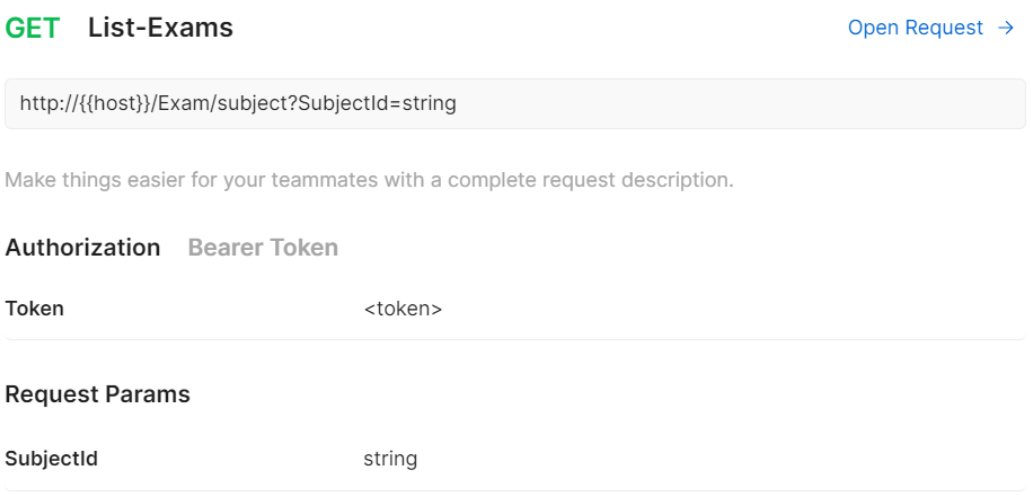


Ilustración 20 Listar todos los exámenes de una asignatura

La ilustración 21 muestra el *endpoint* y el contrato para asignar una lista de estudiantes a una asignatura:

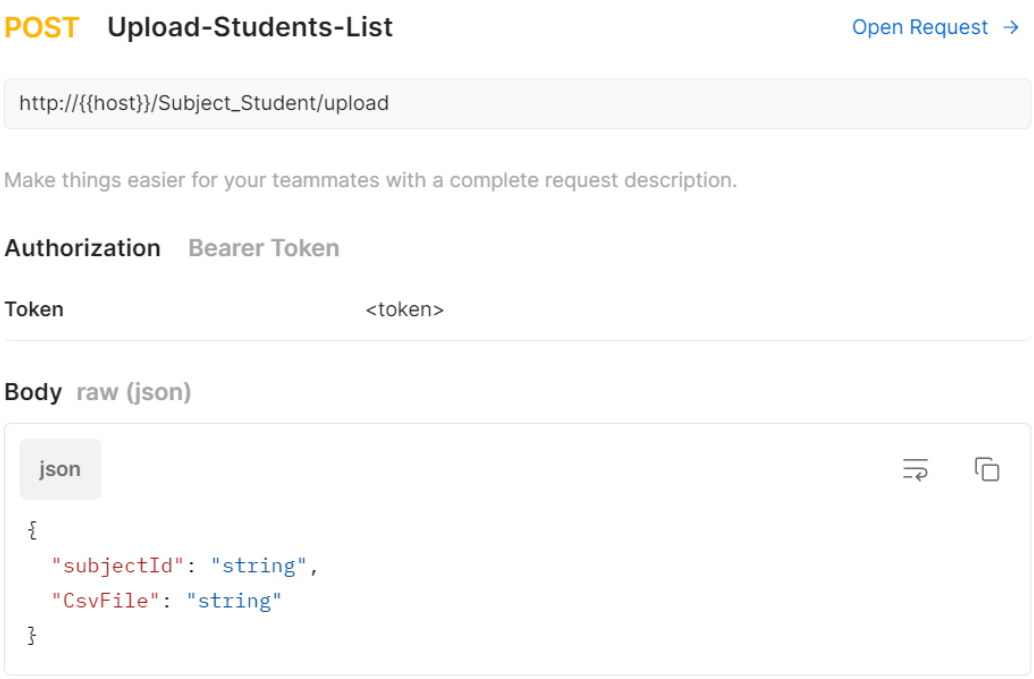


Ilustración 21 Asignar una lista de estudiantes a una asignatura

La ilustración 22 muestra el *endpoint* y el contrato para subir los exámenes evaluados para su clasificación:

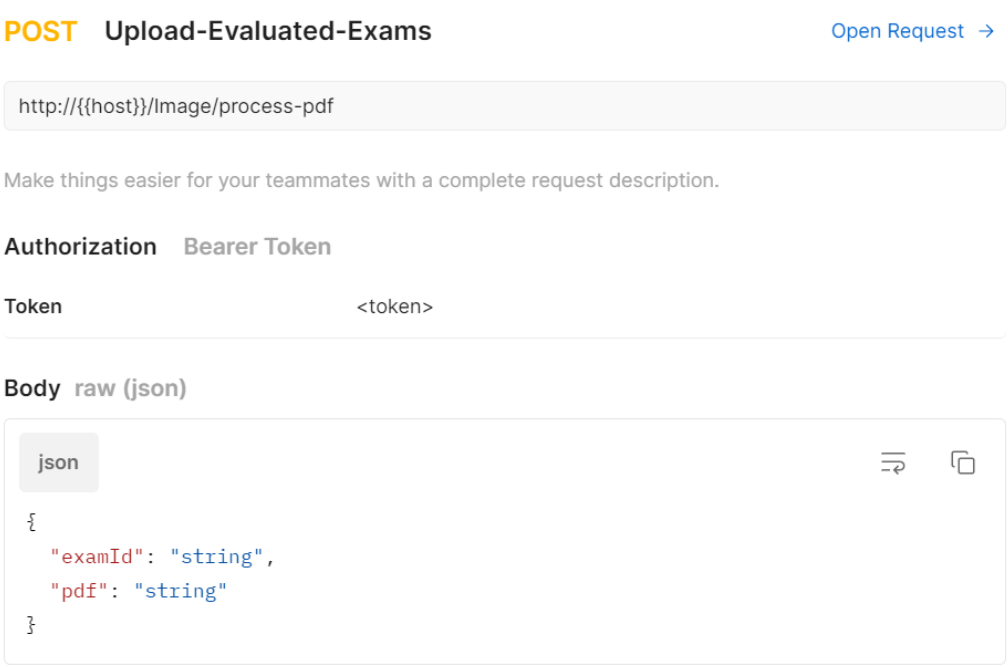


Ilustración 22 Subir los exámenes evaluados para su clasificación

La ilustración 23 muestra el *endpoint* y el contrato para listar los exámenes evaluados y clasificados:

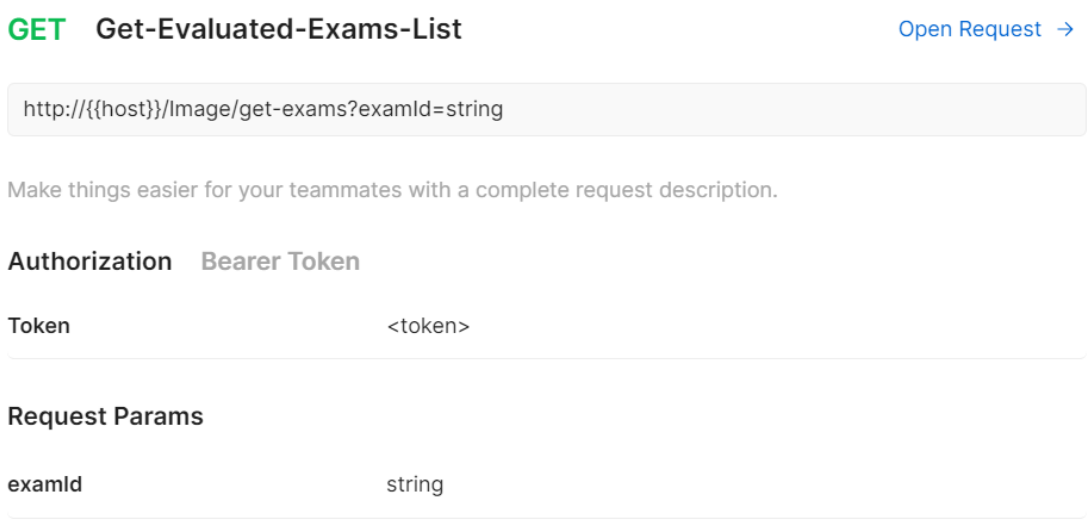


Ilustración 23 Listar los exámenes evaluados y clasificados

La ilustración 24 muestra el *endpoint* y el contrato para obtener la información del examen del que se quiere resolver el conflicto:

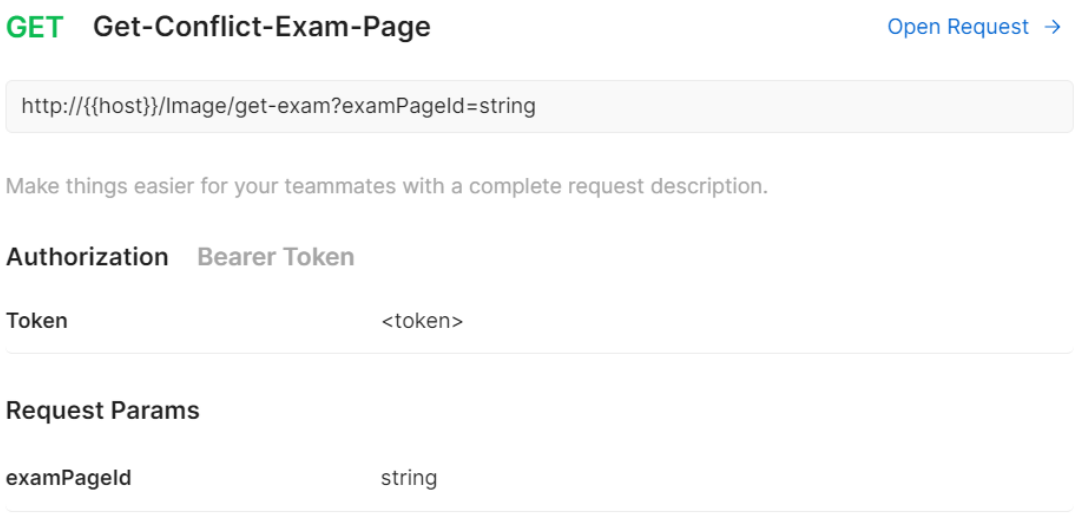


Ilustración 24 Obtener información examen para resolver el conflicto

La ilustración 25 muestra el *endpoint* y el contrato para resolver el conflicto asignando el DNI correcto al examen:

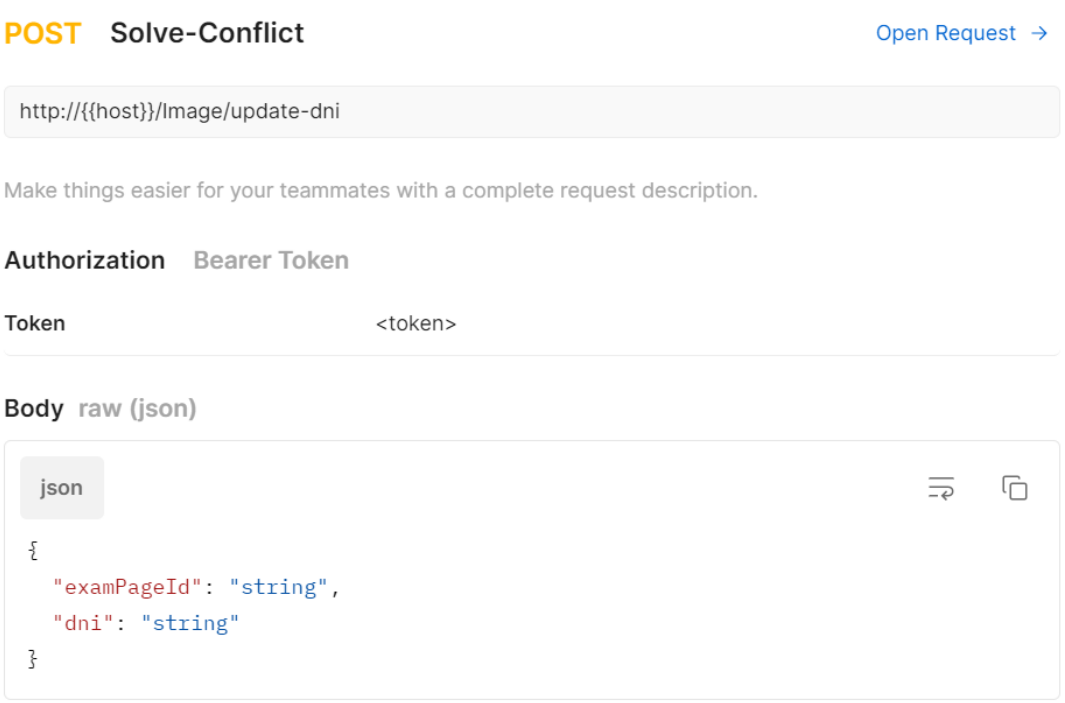


Ilustración 25 Resolver el conflicto asignando el DNI correcto al examen

La ilustración 26 muestra el *endpoint* y el contrato para descargar los exámenes evaluados:

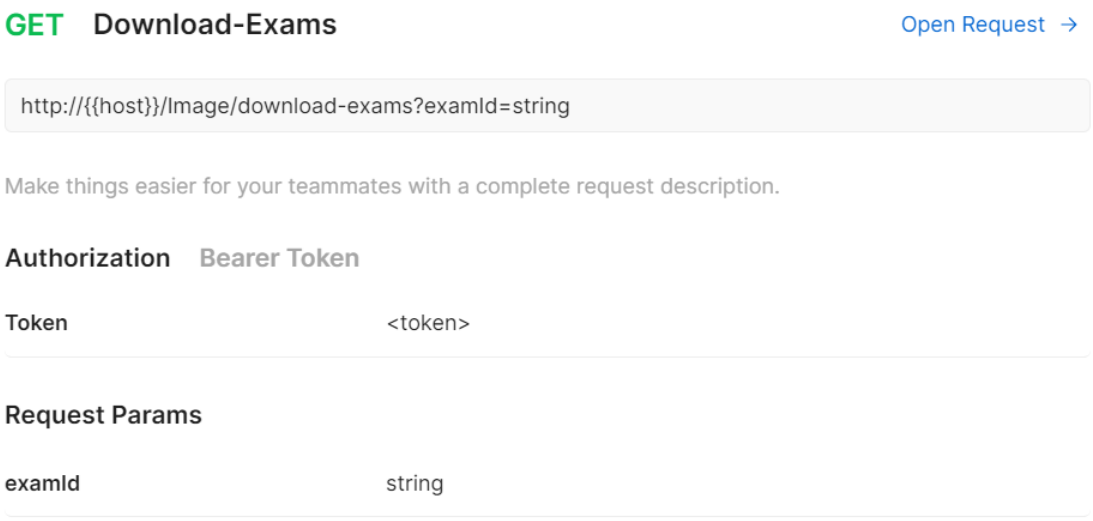


Ilustración 26 Descargar los exámenes evaluados

La ilustración 27 muestra el *endpoint* y el contrato para eliminar un examen del sistema:

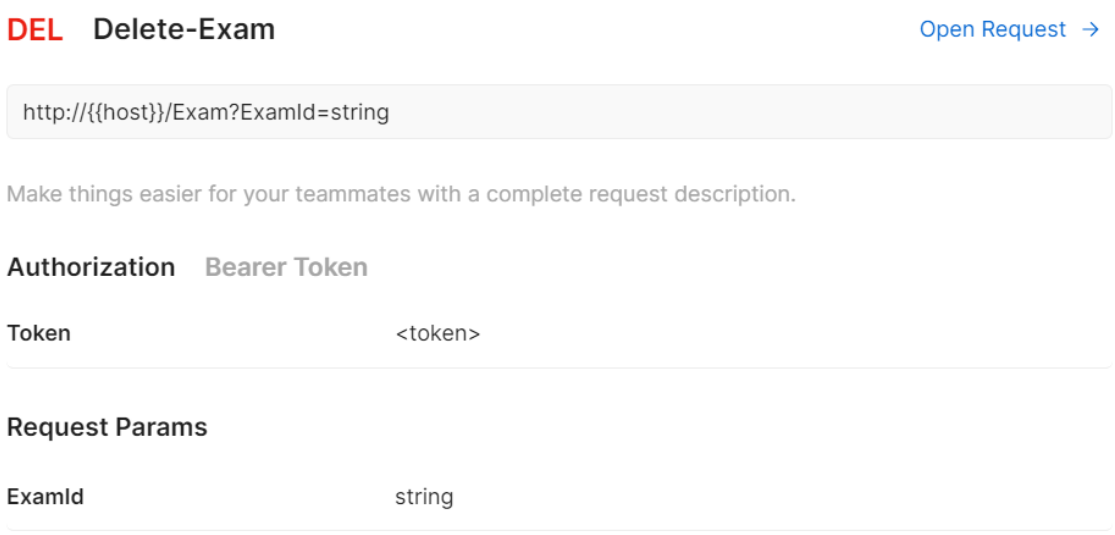


Ilustración 27 Eliminar un examen

La ilustración 28 muestra el *endpoint* y el contrato para eliminar una asignatura del sistema:

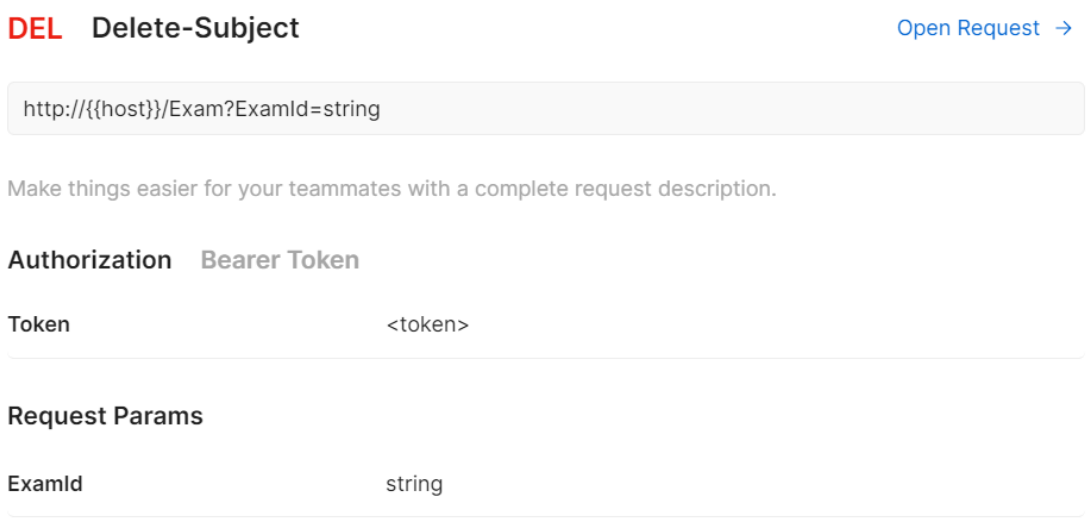


Ilustración 28 Eliminar una asignatura

7.6 Aplicación cliente

En esta sección se describe la parte cliente o *front-end* que se ha implementado. Es importante recordar que la implementación de esta parte queda fuera de los objetivos del proyecto, pero se ha decidido crear una aplicación simple y sencilla con la finalidad de que los profesores puedan utilizar y evaluar todas las funciones que se han desarrollado en el *back-end*.

Para el desarrollo de este cliente se ha hecho uso de la tecnología angular, como se ha descrito anteriormente en la sección 7.1 Tecnologías y herramientas.

Respecto a la estructura de la aplicación, se han seguido las prácticas dictadas por Angular [39], en las cuales se han contemplado aspectos como la nomenclatura de cada uno de los elementos del código, la estructura de carpetas y ficheros y las responsabilidades de cada sección de la aplicación entre otros.

Se han utilizado de referencia las entidades y manejadores implementados en el servidor o *back-end* para la generación de los componentes de aplicación cliente o *front-end*. De esta forma se han implementado los siguientes componentes:

- Login: Gestiona el acceso a la aplicación, permitiendo acceder o registrarse.
- Asignatura: Gestiona las asignaturas del usuario ofreciendo la posibilidad de listarlas, de crear nuevas asignaturas y el borrado de ellas.
- Examen: Gestiona los exámenes del usuario ofreciendo la posibilidad de listarlos, de crear nuevos exámenes y el borrado de ellos.
- Exámenes clasificados: Gestiona la visualización de los exámenes evaluados subidos por el profesor y clasificados por la aplicación y su descarga en un fichero ZIP.
- Resolución de conflictos: Gestiona la resolución de los posibles conflictos ocasionados por el predictor de textos.

Cada uno de estos componentes está dividido en tres ficheros mediante los cuales se consigue gestionar la lógica del componente, el esqueleto de la vista de usuario y los estilos de este esqueleto visual.

A continuación, las ilustraciones 29 a 33 muestran la apariencia de cada uno de los componentes descritos anteriormente:

La ilustración 29 muestra el componente de Login:

Ilustración 29 Componente de login

La ilustración 30 muestra el componente de Asignatura:

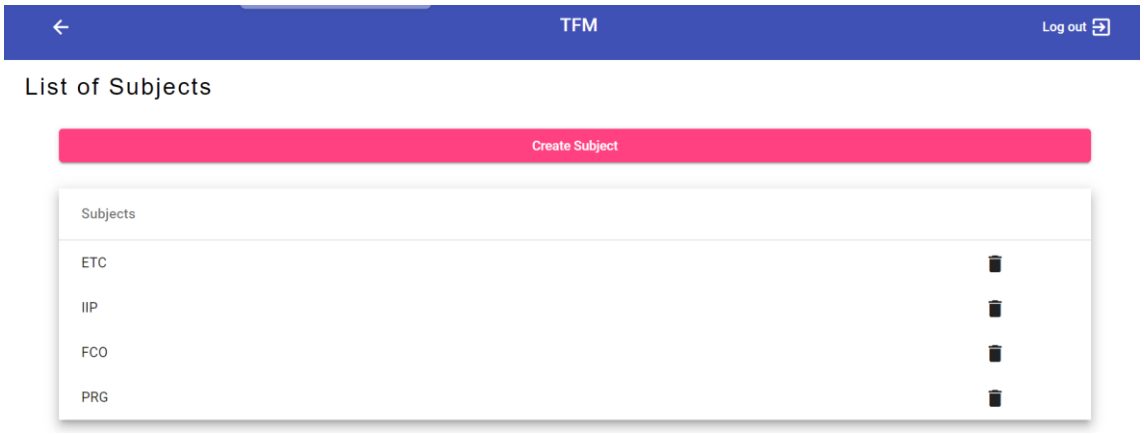


Ilustración 30 Componente de asignatura

La ilustración 31 muestra el componente de Examen:

Ilustración 31 Componente de examen

La ilustración 32 muestra el componente Exámenes clasificados:

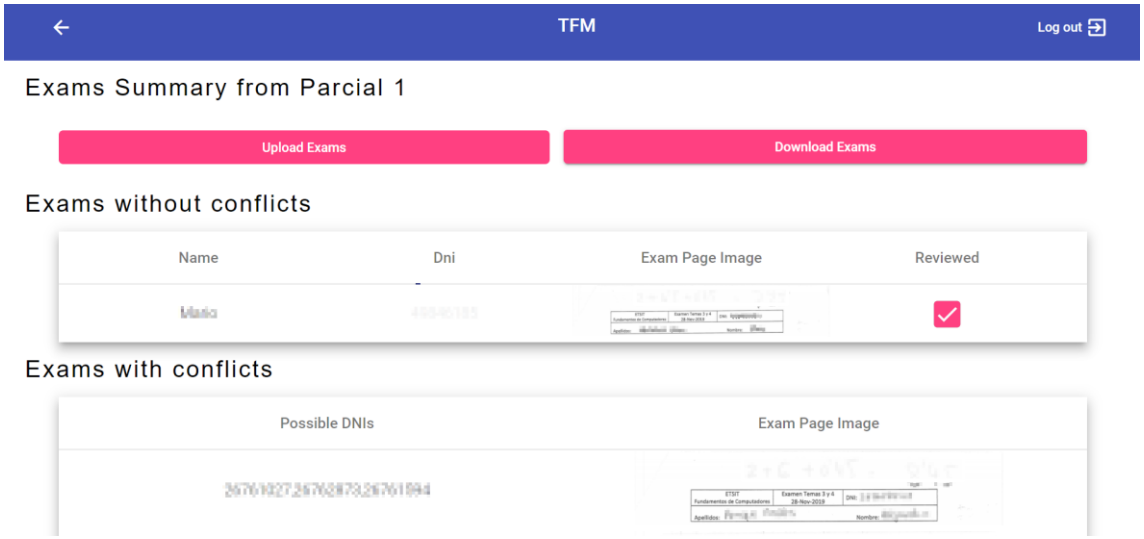


Ilustración 32 Componente exámenes clasificados

La ilustración 33 muestra el componente de Resolución de conflictos:

Ilustración 33 Componente de resolución de conflictos

8. Pruebas y Resultados

En este capítulo se aborda el estudio del comportamiento de la aplicación, en él se comprobará que todas y cada una de las funcionalidades implementadas cumplen con los requisitos definidos durante la fase de toma de requisitos y especificación de casos de uso. Para ello se dividirá este capítulo en dos secciones. En primer lugar, se mostrarán los resultados de las pruebas unitarias desarrolladas durante la fase de implementación. Y en segundo lugar las pruebas de uso, las cuales se apoyarán en la herramienta Postman y en el propio *front-end* desarrollado. Este tipo de pruebas garantizarán el correcto funcionamiento los casos de uso diseñados en los capítulos anteriores.

8.1 Pruebas unitarias

Como se ha visto en el capítulo anterior, se ha hecho uso de la metodología TDD para conseguir cumplir con todos los requisitos y casos de uso definidos en su correspondiente análisis. Por ello se ha conseguido un buen número de pruebas unitarias, las cuales garantizan el correcto funcionamiento de la aplicación y que esta cuenta con todos y cada uno de los requisitos establecidos. La ilustración 34 muestra a modo de resumen las pruebas unitarias implementadas donde además se aprecia que todas las pruebas disponen de un estado satisfactorio al haber sido ejecutadas.

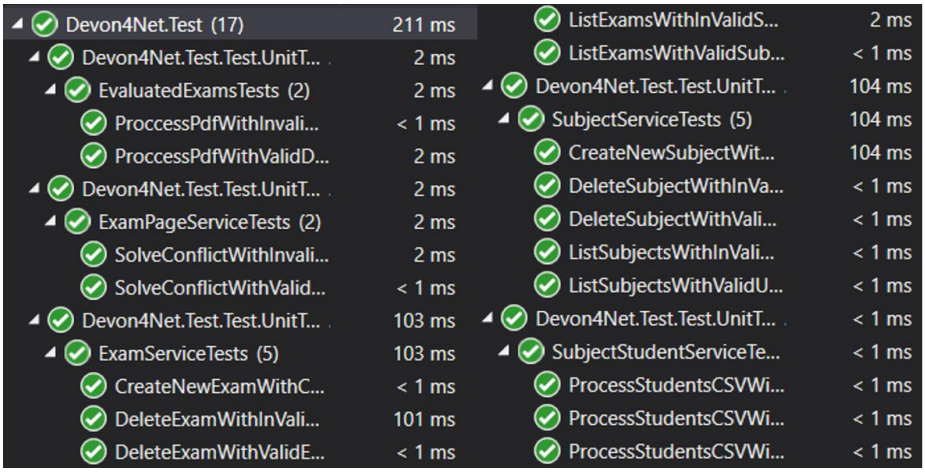


Ilustración 34 Pruebas unitarias satisfactorias

8.2 Pruebas de uso

Una vez realizadas las pruebas unitarias, en las cuales se ha comprobado en primera instancia que el comportamiento de la aplicación es el correcto y esperado, se han realizado las pruebas de caso de uso mediante la herramienta Postman. Esta herramienta ha permitido realizar llamadas al servidor y poder analizar cada una de las respuestas de estas llamadas.

Esta herramienta se ha detallado en el capítulo anterior, no obstante, se van a mostrar una serie de imágenes de la herramienta y se describirán sus principales funciones. En primer lugar, se encuentra la ilustración 35 la cual muestra la interfaz proporcionada por Postman a la hora de preparar una petición o llamada al servidor. Se han añadido cuadros de colores para facilitar la explicación que se realiza a continuación de la ilustración.

Ilustración 35 Petición HHTP Postman

En primer lugar, se observa el cuadro de color rojo, el cual define el tipo de petición que se realizará, las posibilidades para esta aplicación son GET, POST y DELETE.

En segundo lugar, se observa el cuadro de color amarillo, en el cual se especificará la ruta o URL del servicio que se desee llamar. Esta ruta o URL está compuesta por el protocolo de la llamada, la dirección del host donde reside el servidor y el *endpoint* que se desea acceder. A continuación de este campo se encuentra el botón de enviar, botón que se accionará para enviar la petición al servidor.

Justo debajo de este estos dos campos se encuentra el campo coloreado en azul, en esta sección Postman ofrece las diferentes opciones para modificar las distintas autorizaciones que puedan ser necesarias, las cabeceras de la llamada y el propio cuerpo de la petición. Además, Postman también ofrece la posibilidad de crear pruebas automáticas las cuales se definirán entrando en el apartado de test de este propio submenú.

El siguiente campo es el de color rosa, en el cual se escribirá el contenido del cuerpo del mensaje, en el caso de la aplicación desarrollada estará en formato JSON, pero es importante destacar que podría estar escrito en otros formatos. Solo será necesario rellenar el cuerpo del mensaje en las llamadas de tipo POST.

A continuación de este campo se encuentra el cuadro de color naranja, en este campo se podrá cambiar las vistas entre el cuerpo del mensaje, sus cabeceras, y la respuesta a la petición, entre otras opciones. A la derecha de este submenú se encuentra el cuadro verde, el cual mostrará información referente a la respuesta de la llamada, mostrando el código de estado, el tiempo de respuesta del servidor y el tamaño de la respuesta.

Por último, se encuentra el campo de color negro en el que se podrá leer el contenido del cuerpo de la respuesta.

Una vez explicada la interfaz general de Postman, se mostrará a través de la ilustración 36, la ventana de autorización. Esta ventana corresponde con la pestaña de autorización de la sección de color azul de la ilustración 35. Este apartado define las autorizaciones que serán necesarias para realizar correctamente las llamadas. Mediante la sección de color verde se puede seleccionar el tipo de autorización, que en el caso de esta aplicación se seleccionará *Bearer Token*. Y el token se escribirá en la sección de color azul en la que se aprecia la palabra *Token.*

Se hace especial atención a este apartado debido a que como se ha explicado en puntos anteriores se ha hecho uso de JWT para garantizar la seguridad de la aplicación y este es el lugar donde Postman permite configurar dicho token en cada una de las llamadas.

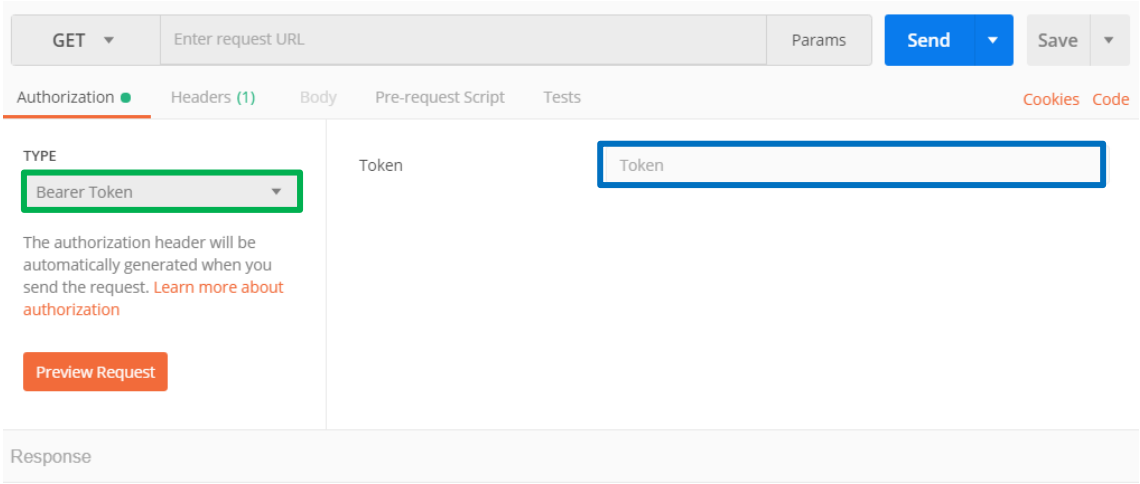


Ilustración 36 Autorización petición HTTP Postman

En el Anexo 1 se encuentran todas las llamadas realizadas a través de la herramienta Postman a los servicios implementados en las cuales se observa el correcto y esperado funcionamiento de todas las funcionalidades que se requerían.

8.3 Pruebas de reconocimiento de DNI

Además de las pruebas realizadas para comprobar el correcto funcionamiento de la aplicación, se han realizado distintas pruebas sobre el predictor de caracteres manuscritos mediante el cual la aplicación consigue reconocer y clasificar los DNIs de los alumnos.

Para llevar a cabo estas pruebas, se han utilizado un total de 55 exámenes de alumnos reales los cuales han sido proporcionados por el tutor. Así mismo el tutor también proporcionó la lista con la información de dichos alumnos. Es importante destacar que estos exámenes se realizaron con anterioridad a la propuesta de este TFM, por lo que los alumnos no fueron advertidos de la importancia y conveniencia de escribir su DNI con una caligrafía esmerada.

Estos exámenes utilizaban la plantilla explicada en el capítulo de diseño y que se muestra en el Anexo 2. Es decir, estos exámenes disponían en su primera página de una cabecera en la cual aparecía tanto la palabra “examen” como “DNI”.

Para preparar los casos de prueba, se escanearon todos los exámenes en un mismo documento PDF, que es el formato requerido por la aplicación para subir los exámenes evaluados de los alumnos y de esta forma poder clasificarlos.

De esta forma se subieron los 55 exámenes a la aplicación y se obtuvieron los resultados que se muestran en la tabla 1:

Reconocimiento	Predicciones	Porcentaje
Correcto	32	59%
Erróneo	3	5%
Conflicto	20	36%
Total	55	100%

Tabla 37 Resultados prueba reconocimiento DNI

Como se aprecia en la tabla anterior, tras la ejecución de las pruebas se obtuvo que de los 55 exámenes un total de 32 se habían predicho de forma correcta, lo que representa un 59% del total de la muestra. En 3 de los exámenes subidos, es decir el 5%, el predictor ha encontrado una única solución, pero no la correcta, esto se define como un falso positivo, ya que a primera vista parece que ha predicho de forma correcta el DNI dado que ha encontrado una única solución, pero tras analizar el DNI predicho se observa que no es el correcto. Por último, en 20 de los 55 exámenes que representan el 36%, se encontró un conflicto, es decir que el predictor no encontró una única solución.

En los exámenes anteriores el número de páginas por examen era constante, aunque esta información no se proporciona a la aplicación y por tanto debe detectar el número de páginas que conforma cada examen. Para verificar esta funcionalidad, se crearon 10 exámenes ficticios formados por una primera hoja con la cabecera descrita en el Anexo 2 y un número variable, entre 0 y 5, de hojas de examen de las que son proporcionadas por la UPV. Estos exámenes se escanearon y procesaron con la aplicación, que creó, de forma correcta, los ficheros de salida correspondientes a cada alumno con sus hojas de examen.

9. Despliegue

A lo largo de este capítulo se explicará el proceso de despliegue que se ha definido y seguido a la hora de instalar y configurar la aplicación en la máquina proporcionada por el tutor de este proyecto. El equipo es un computador personal con procesador Intel i5, 8GB de memoria principal, 128GB de disco SSD, y ejecuta el sistema operativo Kubuntu 18.04 LTS. Además, en este capítulo se detallarán todas las configuraciones necesarias para la instalación y ejecución de la aplicación.

En primer lugar, la máquina proporcionada por el tutor se encuentra en la Universitat Politècnica de València y para acceder a ella es necesario estar en la misma red o utilizar una VPN. La dirección para acceder a la aplicación desplegada es: http://pfc02.disca.upv.es:4200/login.

En segundo lugar, mencionar que la tecnología que se ha escogido para trabajar y hacer posible todo el despliegue ha sido Docker, esta tecnología ha sido explicada previamente en puntos anteriores, pero no obstante se profundizará en las razones que han motivado la elección de esta tecnología.

Docker ha sido la tecnología elegida para abordar el despliegue porque ofrece las siguientes características: una estandarización en la estructura de la aplicación, esto significa que permitirá replicarla y ampliarla de una forma sencilla. Por lo tanto, se trata de una tecnología con buena escalabilidad.

Además, Docker ofrece portabilidad sobre las aplicaciones, esto significa que proporciona un proceso sencillo para instanciar las aplicaciones desarrolladas en distintas máquinas. De esta forma se podrá disponer exactamente de la misma aplicación, con el mismo comportamiento y con todas las dependencias instaladas en cualquier ordenador.

Docker consigue esta portabilidad y escalabilidad gracias a su sistema de virtualización, el cual, a diferencia de una máquina virtual tradicional, no necesita contener todo el sistema operativo, ya que aprovecha el propio sistema operativo sobre el cual se está ejecutando, compartiendo el *kernel* del sistema anfitrión e incluso parte de sus bibliotecas.

Respecto al tamaño en disco, Docker ofrece una gran ventaja ya que solo se va a almacenar el contenido de aquello que lo diferencia del sistema operativo que lo ejecuta. Este mismo comportamiento se aplica al uso de CPU y RAM, ya que comparten este tipo de recursos con el sistema anfitrión.

A continuación, mediante la ilustración 37, se muestra la diferencia de enfoque de usar máquinas virtuales convencionales y Docker:

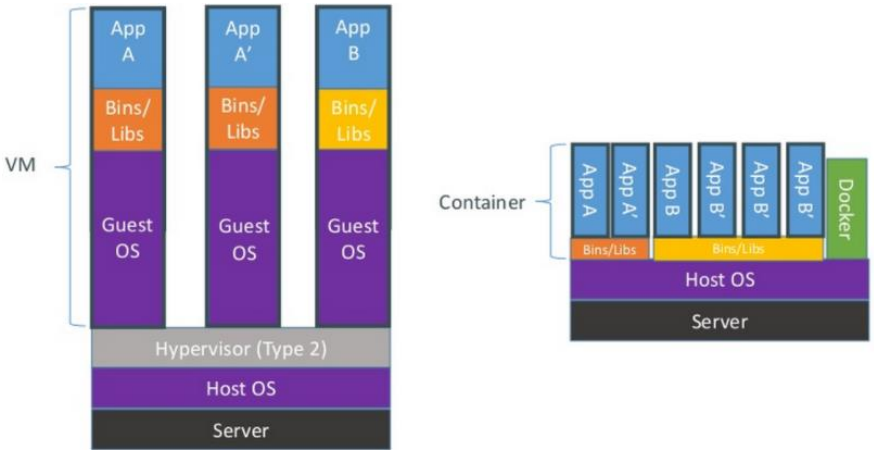


Ilustración 37 Máquinas virtuales / Docker

En el universo de Docker es necesario conocer dos términos, los cuales son “imágenes” y “contenedores”. Las imágenes en Docker podrían verse como componentes estáticos, ya que son los sistemas operativos base con la combinación de las aplicaciones empaquetadas. En cambio, los contenedores, corresponden con la instalación y/o ejecución de las imágenes, donde se pueden ejecutar varios contenedores partiendo de una misma imagen.

Además de estos términos, es importante conocer los siguientes tipos de archivos, los cuales permiten a los desarrolladores crear estas imágenes y a su vez los contenedores.

Por una parte, existe el fichero “Dockerfile”, el cual es un fichero de texto plano que permite definir todas las instrucciones que Docker debe seguir para construir una imagen, en otras palabras, es como escribir una receta con todos los pasos necesarios mediante la cual Docker creará las imágenes que servirán posteriormente para ejecutar los contenedores.

Por otra parte, existe el fichero “Docker Compose” el cual define los diferentes servicios que componen la aplicación. Hay que destacar que cada uno de los servicios que se

definen en este archivo se ejecutará en un contenedor diferente, ya que cada uno de estos dispondrá de una imagen distinta. Mediante este fichero se consigue una orquestación de todos los servicios que van a componer la aplicación, haciendo realmente simple la gestión de estos.

Entrando con más detalle en la aplicación desarrollada, esta consta de un total de cuatro servicios que se ejecutarán en cuatro contenedores diferentes, los cuales son:

- Servidor o *back-end* de la aplicación: Este servicio corresponde al servidor diseñado e implementado a lo largo de esta memoria. Para hacer posible su ejecución en Docker, se ha implementado una imagen personalizada, la cual está basada en un sistema operativo Ubuntu.
- Cliente o *front-end* de la aplicación: Este cliente corresponde a la interfaz básica que se ha creado con la finalidad de poder utilizar el servidor implementado. De la misma forma que con el servidor, se ha implementado una imagen personalizada, pero esta vez basada en el sistema operativo Alpine Linux.
- Base de datos PostgreSQL: Para la ejecución de este servicio se ha hecho uso de la imagen oficial de PostgreSQL para Docker, disponible en los repositorios de Docker.
- Repositorio Nexus: De la misma forma que la base de datos se ha hecho uso de la imagen oficial de Nexus disponible en los repositorios de Docker.

Estos cuatro servicios han sido configurados y orquestados mediante un archivo “Docker Compose”.

Para desplegar la aplicación, se han creado dos archivos “Dockerfile”, uno para la aplicación servidor y otro para la aplicación cliente, y un archivo “Docker Compose”, el cual gestiona todos estos servicios. No ha sido necesaria la creación de este tipo de archivos para Nexus y la base de datos PostgreSQL, ya que se ha utilizado la imagen oficial publicada en Docker.

10. Conclusiones

El proyecto realizado tenía como objetivo principal el diseño y desarrollo de una aplicación web que permita a sus usuarios clasificar, en el sentido indicado en la introducción, los exámenes evaluados a partir del DNI escrito por los estudiantes.

Como se ha mostrado en el capítulo de pruebas y resultados, la aplicación cumple con todos y cada uno de los requisitos descritos al inicio del proyecto, y por tanto se puede decir que se ha alcanzado de forma satisfactoria el objetivo principal. Es cierto que, aunque la efectividad de la herramienta de reconocimiento de caracteres ha sido menor a la deseada, sigue representando una ayuda importante al trabajo del profesor. Como se ha comentado, los alumnos no estaban advertidos de la necesidad de escribir con claridad los dígitos del DNI. También sería conveniente, en un futuro, estudiar otras alternativas para el reconocimiento de texto manuscrito.

Los objetivos secundarios descritos en el punto 1.1 se han cumplido todos y cada uno de ellos. No obstante, no todos los objetivos se superaron con la misma dificultad. Uno de los objetivos más difíciles de alcanzar ha sido la utilización de la metodología TDD, ya que ha sido necesario consultar una gran cantidad de documentación, artículos y ejemplos con la finalidad de entenderla de una forma correcta y completa. Pese a ello, emplear TDD ha sido una experiencia muy gratificante, puesto que en todo momento del desarrollo se tiene la certeza de que todo lo que se ha implementado funciona correctamente, por lo que se podía saber el estado del proyecto en todo momento. Una vez superado el periodo de adaptación a esta metodología, el desarrollo del proyecto se ha realizado de una forma ágil.

El haber superado satisfactoriamente todos los objetivos y requisitos me ha proporcionado una gran satisfacción.

Asimismo, el desarrollo de la aplicación se ha realizado en el tiempo estimado al principio del proyecto. Sin embargo, es importante resaltar que el módulo de reconocimiento de texto manuscrito llevó un poco más de tiempo del esperado, ya que disponía de una alta complejidad a la hora de su integración. En cambio, la fase de análisis y toma de requisitos fue mucho más rápida de lo esperado, debido a la fluida comunicación con el tutor.

Respecto a la implementación de la aplicación, me gustaría señalar que ha sido la primera vez que he trabajado con reconocimiento de textos, lo cual ha supuesto un gran reto que considero he superado satisfactoriamente.

En cuanto a la relación que existe entre el TFM desarrollado y los estudios cursados a lo largo del máster, han sido un gran número de asignaturas las que han contribuido a adquirir el conocimiento necesario para la realización del proyecto. No obstante, se van a mencionar las que han sido más relevantes. Una de las más útiles ha sido "Planificación y dirección de proyectos de TI", ya que ha proporcionado conocimientos útiles a la hora de planificar el proyecto de una forma ágil y eficiente. Otra de las asignaturas que han sido muy provechosas ha sido "Auditoría, calidad y gestión de sistemas de información", la cual ha proporcionado conocimientos profundos para el diseño de las distintas pruebas realizadas a lo largo de este trabajo. Además, también ha permitido establecer una estrategia correcta para garantizar la calidad del producto desarrollado. Otra de las asignaturas ha sido "Servicios y aplicaciones distribuidas" la cual ha proporcionado los conocimientos necesarios para realizar el diseño y desarrollo de aplicaciones web.

Para finalizar este TFM, se va a comentar el aprendizaje que ha supuesto la realización de este trabajo. En primer lugar, quiero destacar que ha sido una gran oportunidad el trabajar con muchas tecnologías nuevas para mí, como ha sido la tecnología de reconocimiento de textos, Tesseract, o la tecnología Docker, que ha permitido realizar el despliegue de la aplicación de una forma realmente simple y efectiva. En segundo lugar, se ha adquirido un mayor dominio a la hora de plantear proyectos y llevarlos a cabo, destacando sobre todo la realización de la fase de diseño. Esta es una de las fases más importantes de la vida de un proyecto, y gracias a este TFM y con la colaboración del tutor he conseguido mejorar mi metodología de trabajo.

Por último, mirando a mi futuro profesional, la realización de este trabajo ha consolidado la idea de continuar por este camino y seguir mejorando cada una de mis habilidades en esta área de la ingeniería informática.

11. Referencias

[1] IBM 805 Test Scoring Machine [En línea] [fecha consulta: 20 de abril de 2021]. Disponible en https://www.ibm.com/ibm/history/ibm100/us/en/icons/testscore/

[2] Alberto Gonzáles Téllez [En línea] [fecha consulta: 20 de abril de 2021]. Disponible en http://www.upv.es/visor/media/91b9a290-1270-11e9-a450-cd0c800984c1/c

[3] José Miguel Valiente González [En línea] [fecha consulta: 20 de abril de 2021]. Disponible en http://www.upv.es/visor/media/1980fc5f-d816-cc43-a0c9-d8d8083989ca/c

[4] Zipgrade [En línea] [fecha consulta: 25 de abril de 2021]. Disponible en https://www.zipgrade.com/

[5] GradeScope [En línea] [fecha consulta: 25 de abril de 2021]. Disponible en https://www.gradescope.com/

[6] Cloud vision [En línea] [fecha consulta: 1 de marzo de 2021]. Disponible en https://cloud.google.com/vision

[7] Amazon Textract [En línea] [fecha consulta: 1 de marzo de 2021]. Disponible en https://aws.amazon.com/es/textract/

[8] Computer Vision [En línea] [fecha consulta: 1 de marzo de 2021]. Disponible en https://azure.microsoft.com/es-es/services/cognitive-services/computer-vision/

[9] Tesseract OCR [En línea] [fecha consulta: 1 de marzo de 2021]. Disponible en https://tesseract-ocr.github.io/

[10] EasyOCR [En línea] [fecha consulta: 1 de marzo de 2021]. Disponible en https://www.jaided.ai/easyocr/

[11] Especificación de Requisitos según el estándar de IEEE 830 (22 de octubre de 2008) [En línea] [fecha de consulta: 28 de abril de 2021]. Disponible en https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf

[12] Diagrama de Casos de uso [En línea]. Polimedia, realizado por Penadés Gramage, María Carmen. 28 de junio de 2016 [fecha consulta 3 de mayo de 2021]. Disponible en: http://hdl.handle.net/10251/66596

[13] CIUBOTARU, B. MUNTEAN, G. *Advanced Network Programming-Principles and TechniquesTítulo*. Londres: Springer, 2013, pp. 89-91. ISBN 978-1-4471-5291-0.

[14] Code Maze, *Onion Architecture in ASP.NET Core* [en línea] [fecha de consulta: 8 de mayo de 2021]. Disponible en https://code-maze.com/onion-architecture-in-aspnetcore/

[15] Relational Database [en línea] [fecha de consulta: 8 de mayo de 2021]. Disponible en https://www.ibm.com/ibm/history/ibm100/us/en/icons/reldb/

[16] Nexus respository [en línea] [fecha de consulta: 5 de junio de 2021]. Disponible en https://help.sonatype.com/repomanager3

[17] Formato CSV [en línea] [fecha de consulta: 12 de julio de 2021]. Disponible en https://docs.fileformat.com/spreadsheet/csv/

[18] Documentación de .NET [en línea] [fecha de consulta: 12 de agosto de 2021]. Disponible en https://docs.microsoft.com/es-es/dotnet/

[19] Documentación .NET Core [en línea] [fecha de consulta: 12 de agosto de 2021]. Disponible en https://docs.microsoft.com/es-es/aspnet/core/?view=aspnetcore-3.1

[20] Visual Studio 2019 [en línea] [fecha de consulta: 15 de agosto de 2021]. Disponible en https://visualstudio.microsoft.com/es/vs/

[21] devonfw [en línea] [fecha de consulta: 2 de agosto de 2021]. Disponible en https://devonfw.com/website/pages/welcome/welcome.html

[22] REST [en línea] [fecha de consulta: 5 de agosto de 2021]. Disponible en https://www.redhat.com/es/topics/api/what-is-a-rest-api

[23] Swagger [en línea] [fecha de consulta: 11 de agosto de 2021]. Disponible en https://swagger.io/

[24] OpenAPI [en línea] [fecha de consulta: 11 de agosto de 2021]. Disponible en https://www.openapis.org/

[25] JSON Web Token [en línea] [fecha de consulta: 17 de agosto de 2021]. Disponible en https://jwt.io/introduction

[26] Angular [en línea] [fecha de consulta: 20 de agosto de 2021]. Disponible en https://angular.io/

[27] Visual Studio Code [en línea] [fecha de consulta: 20 de agosto de 2021]. Disponible en https://code.visualstudio.com/

[28] PostgreSQL [en línea] [fecha de consulta: 20 de agosto de 2021]. Disponible en https://www.postgresql.org/

[29] SQL [en línea] [fecha de consulta: 20 de agosto de 2021]. Disponible en https://datademia.es/blog/que-es-sql

[30] DBeaver [en línea] [fecha de consulta: 22 de agosto de 2021]. Disponible en https://dbeaver.io/

[31] Postman [en línea] [fecha de consulta: 22 de agosto de 2021]. Disponible en https://www.postman.com

[32] Docker [en línea] [fecha de consulta: 25 de agosto de 2021]. Disponible en https://www.docker.com/

[33] Mozilla Firefox [en línea] [fecha de consulta: 25 de agosto de 2021]. Disponible en https://www.mozilla.org

[34] Google Chrome [en línea] [fecha de consulta: 25 de agosto de 2021]. Disponible en https://www.google.com/intl/es/chrome/

[35] FEATHERS, M. MARTIN, R. *The art of Unit Testing*. Segunda edición. Shelter Island: Manning, 2014. ISBN 9781617290893.

[36] MERRILL, B. ALBAHARI, B. DRAYTON, P. *C# Essentials*. O'Reilly Media, Inc. febrero 2001, capitulo "*try Statements and Exceptions*". ISBN 9781449390839.

[37] Recogniform ICR SDK [En línea] [fecha consulta: 25 de abril de 2021]. Disponible en http://www.recogniform.net/eng/icr-sdk.html

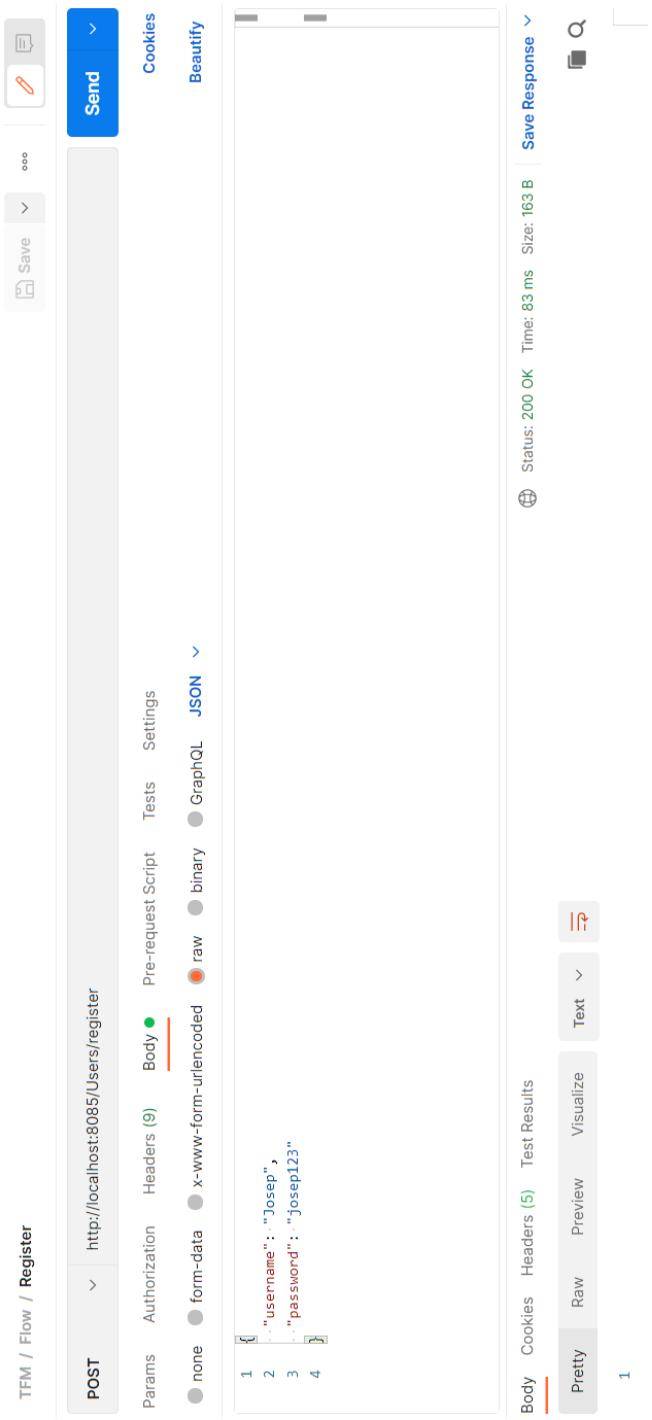
[38] Lipi Toolkit [En línea] [fecha consulta: 25 de abril de 2021]. Disponible en http://lipitk.sourceforge.net/index.htm

[39] Angular guía de estilos [En línea] [fecha consulta: 10 julio de 2021]. Disponible en https://angular.io/guide/styleguide

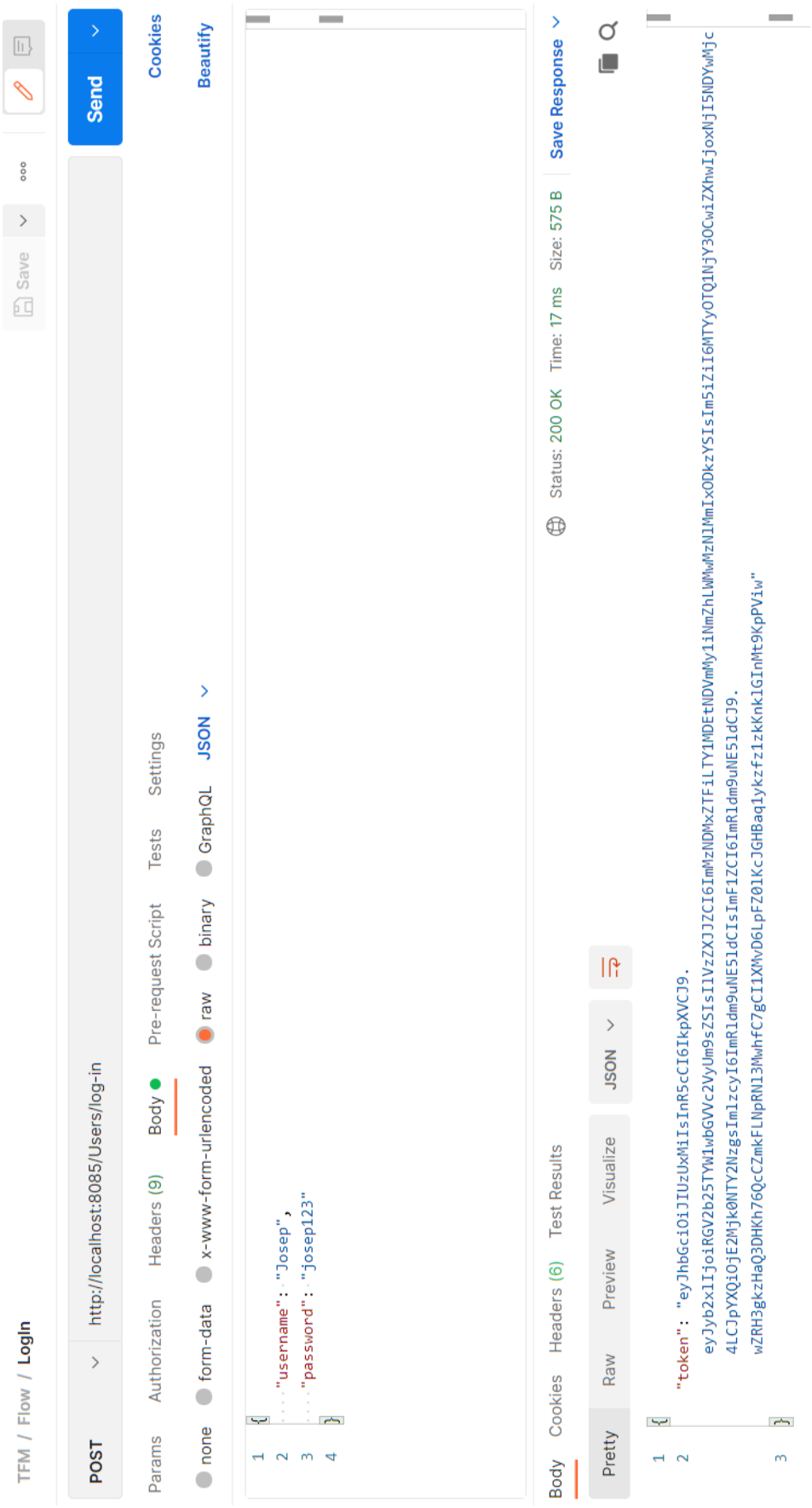
12. Anexos

12.1 Anexo 1. Pruebas Casos de uso Postman

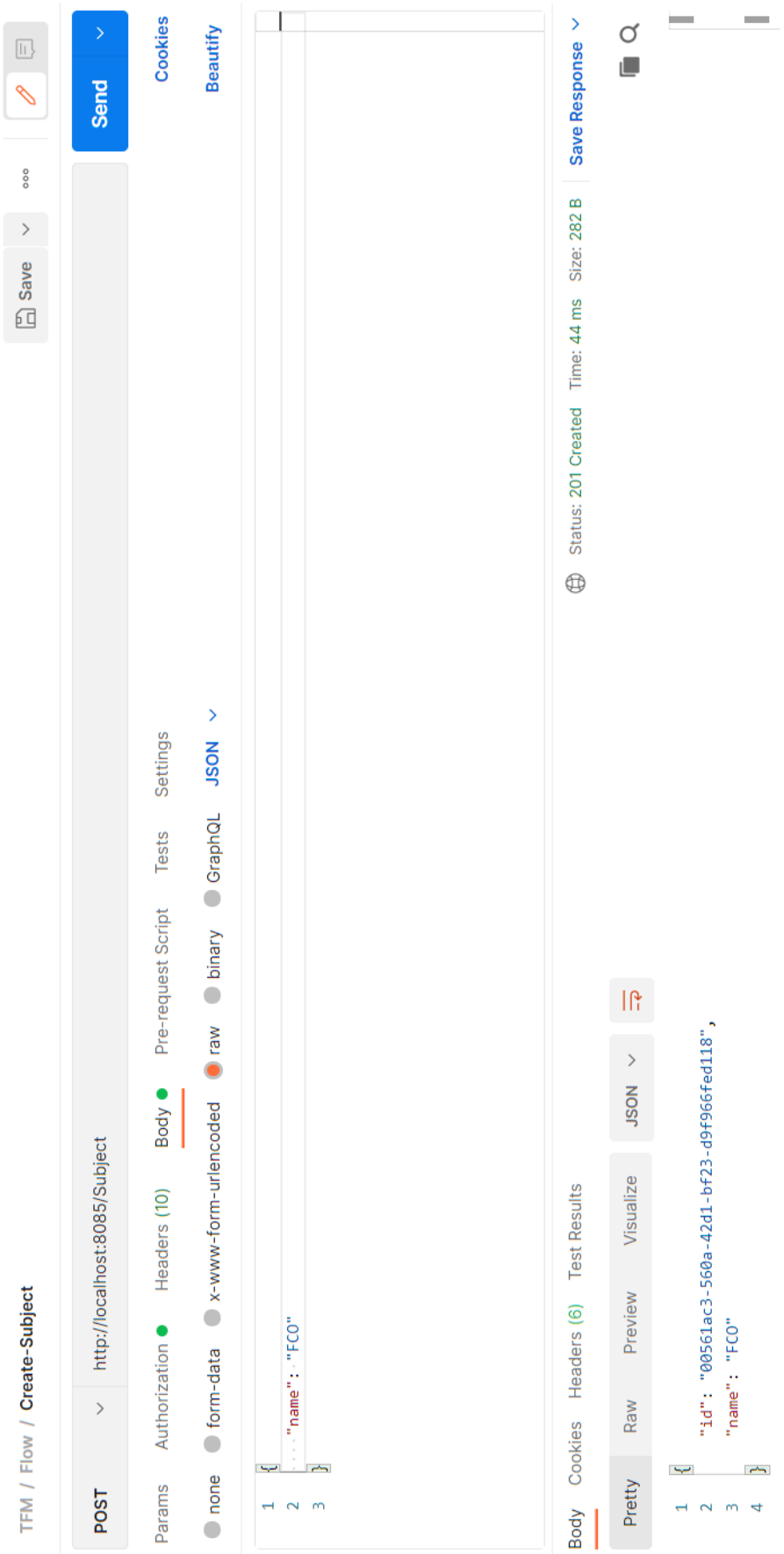
Registro nuevo usuario:



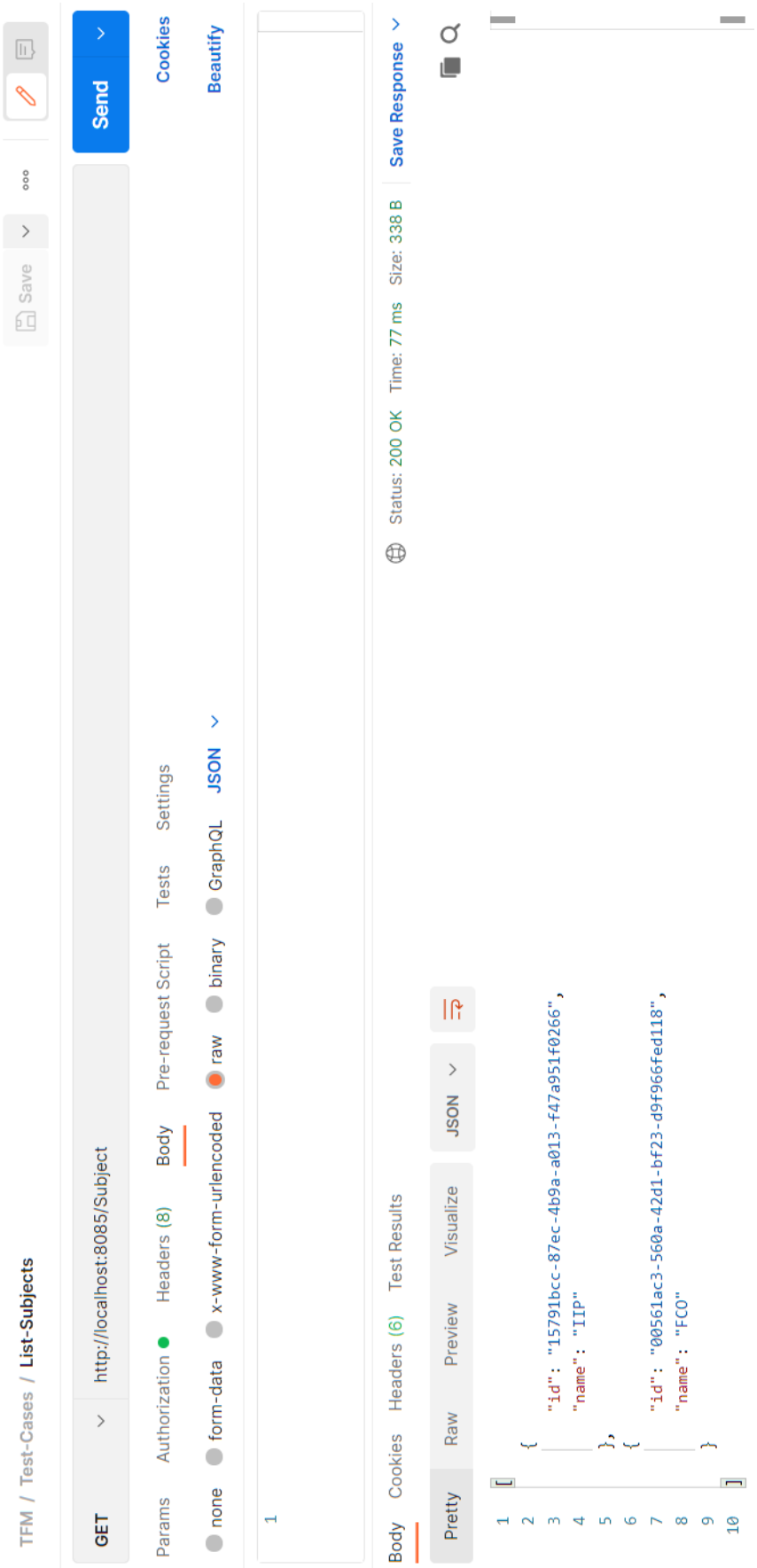
Inicio sesión del usuario:



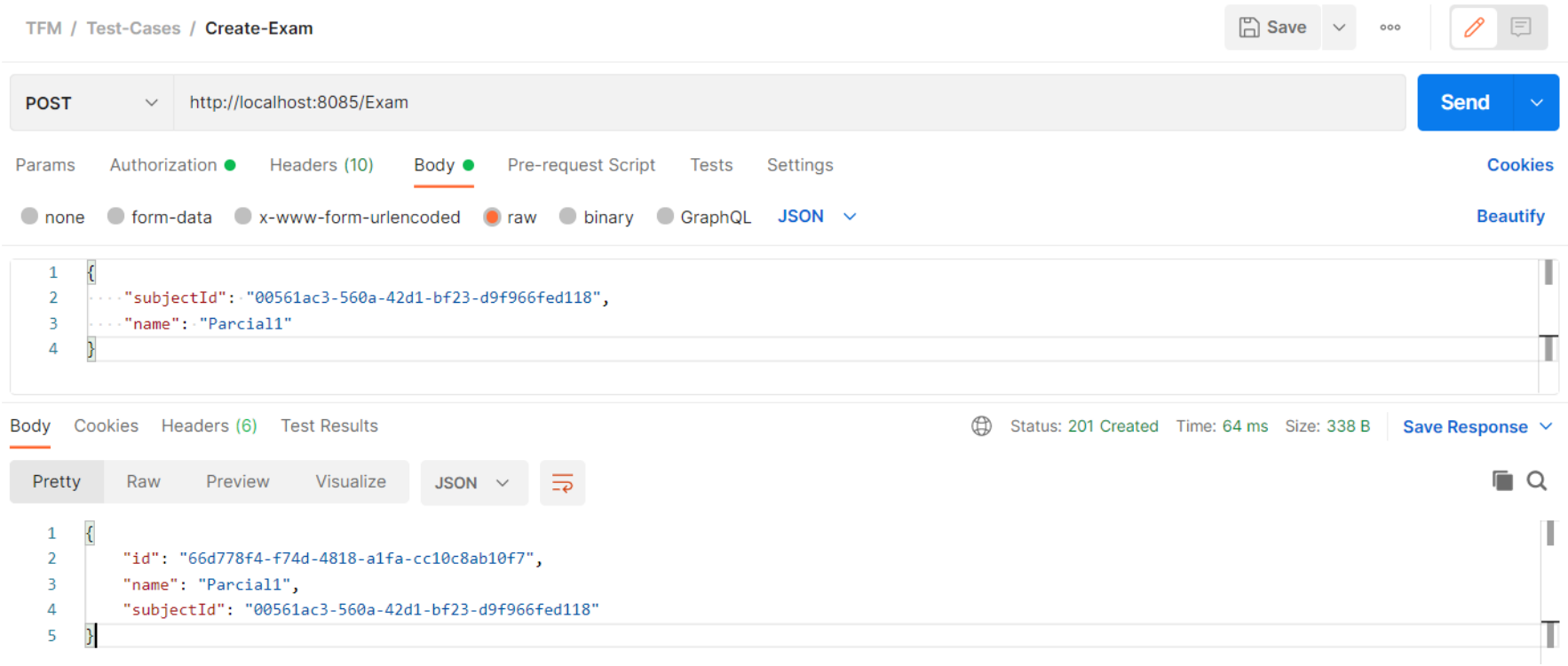
Creación de nueva asignatura:



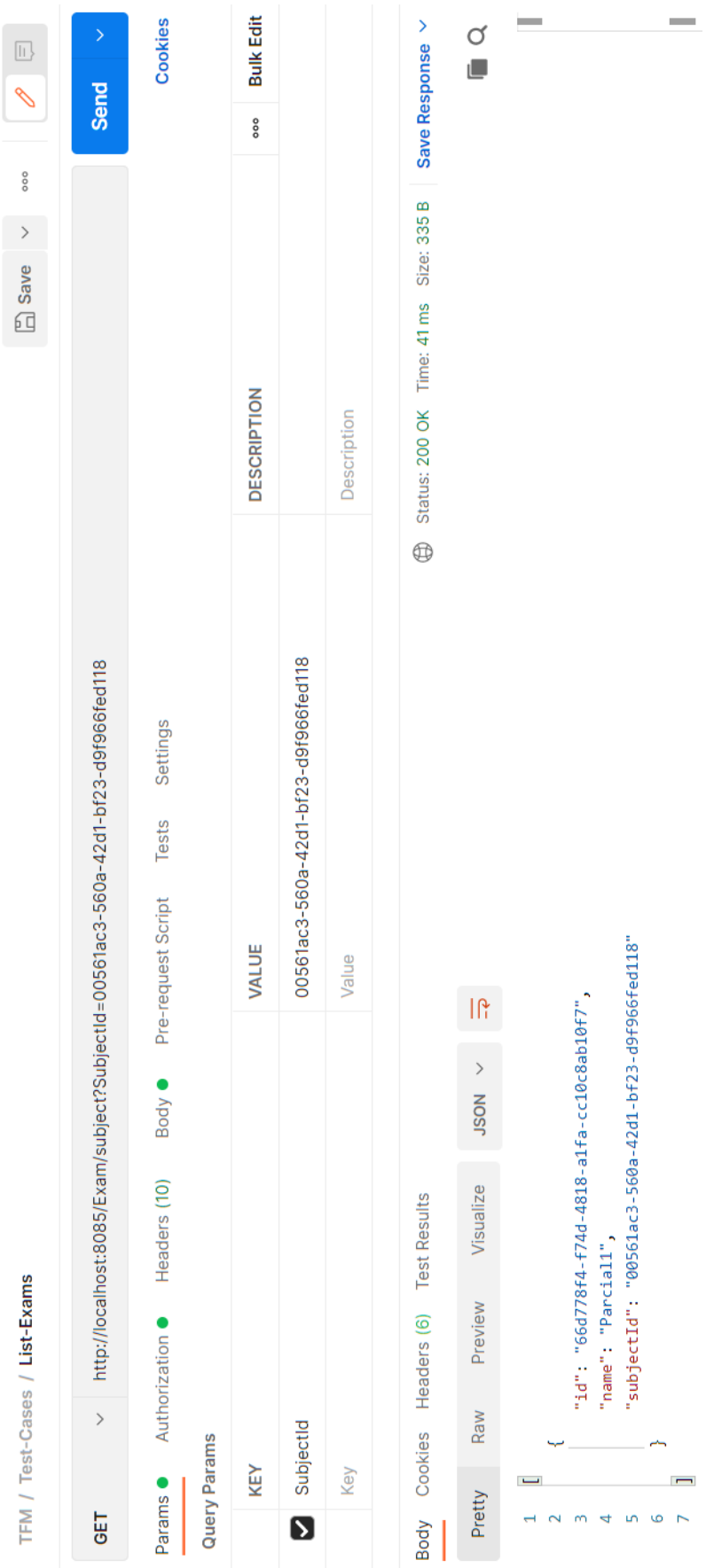
Obtención Lista de asignaturas:



Creación nuevo examen:

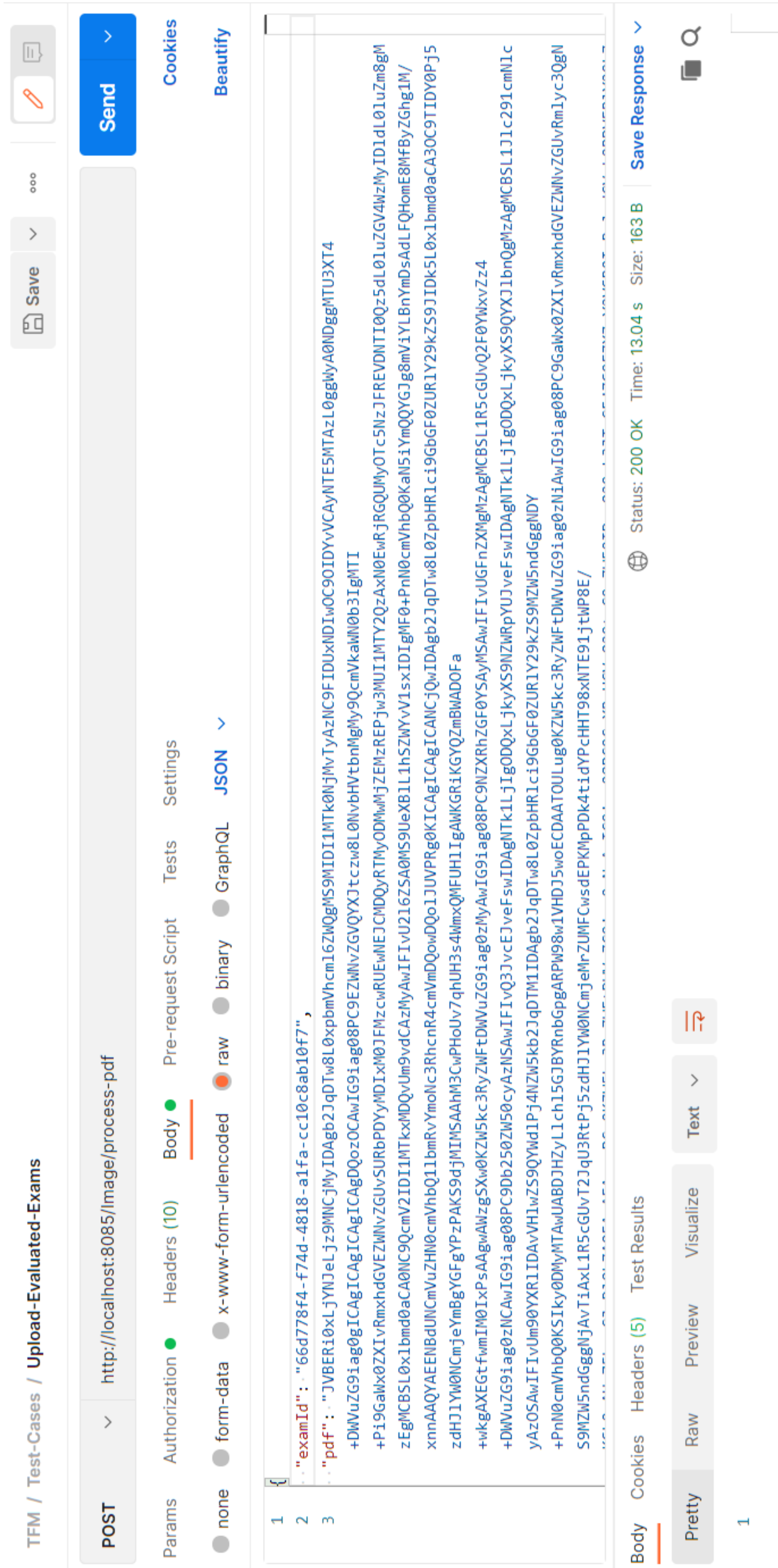


Obtención Lista de exámenes:

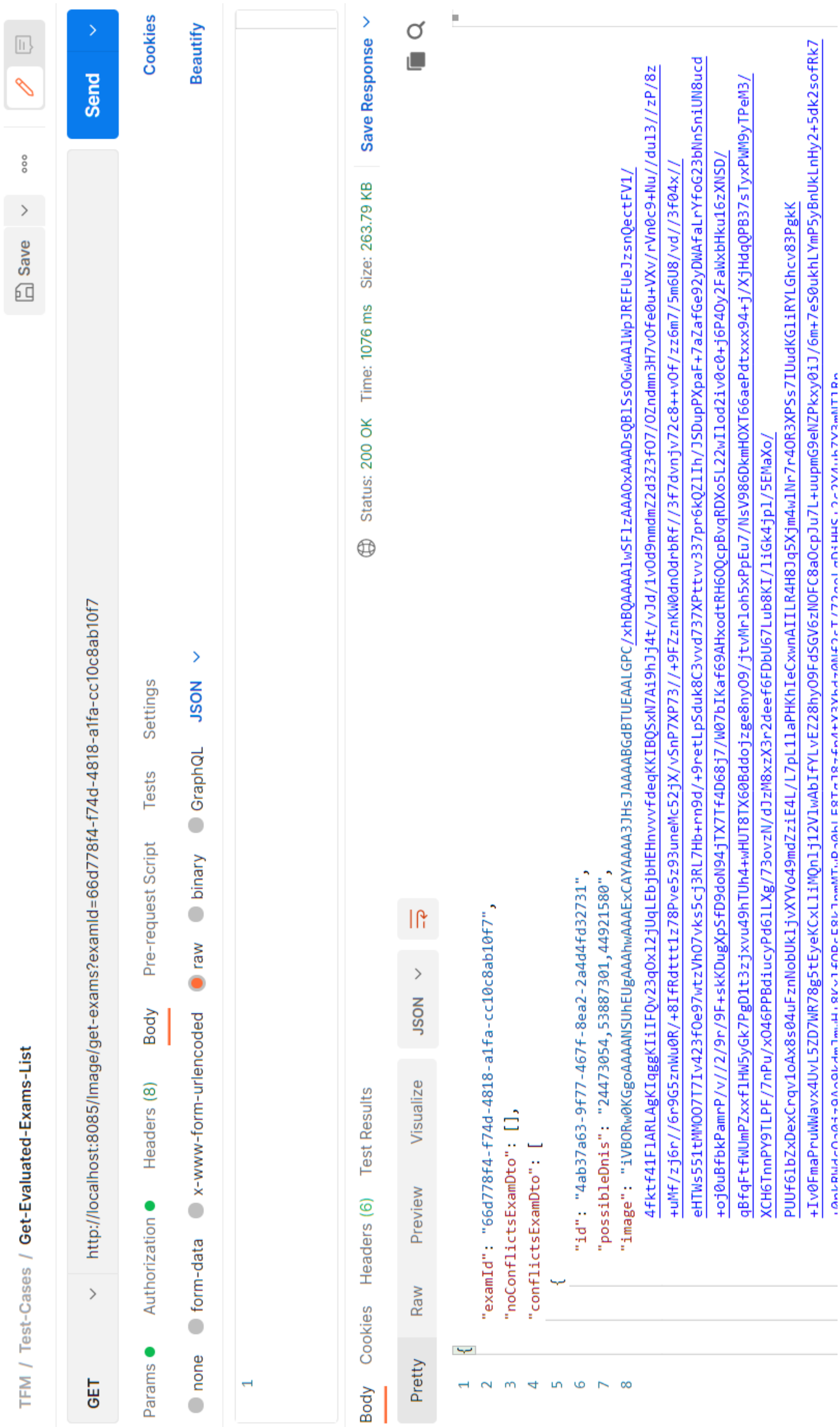


Asignar estudiantes a una asignatura:

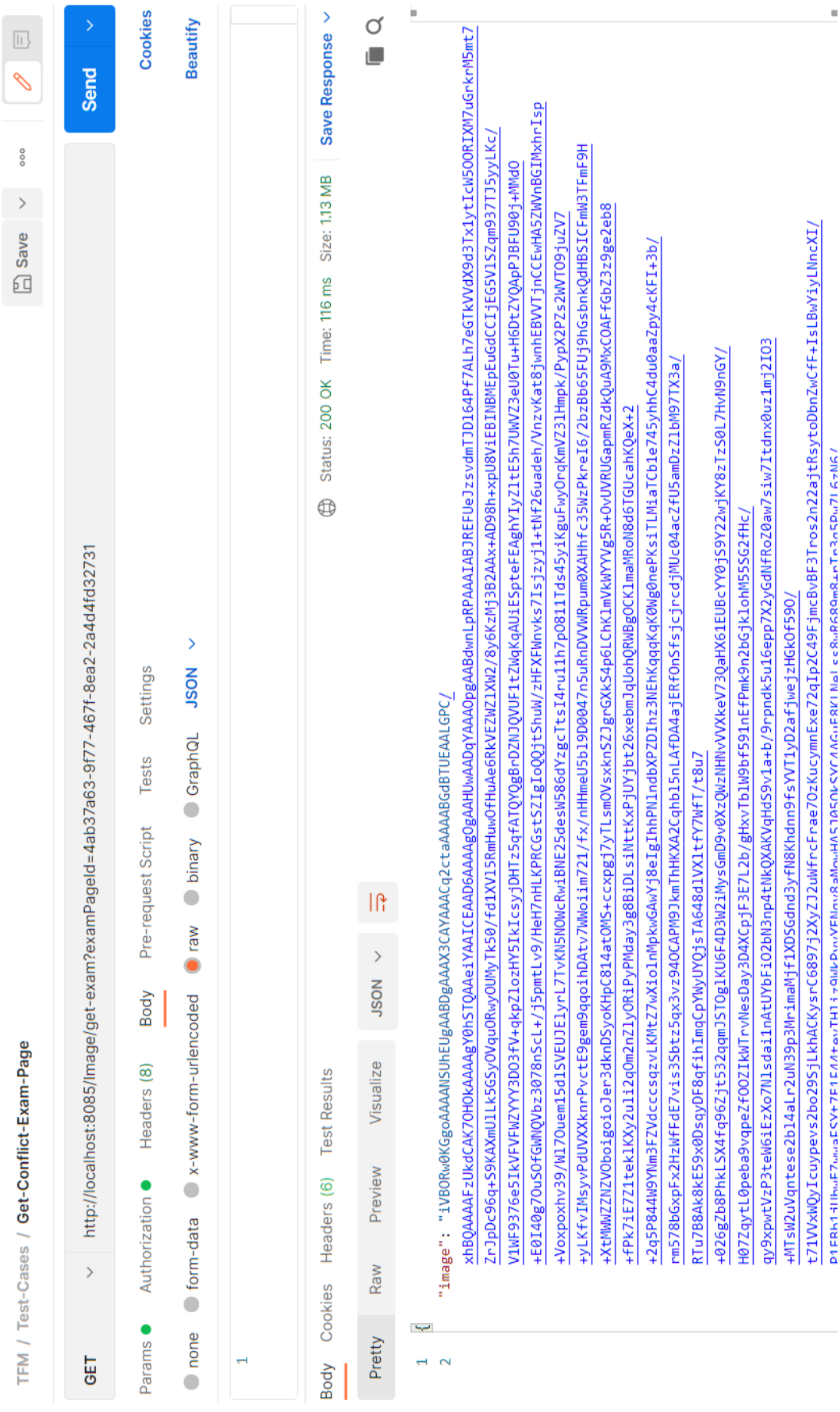
Subir exámenes evaluados para su clasificación:



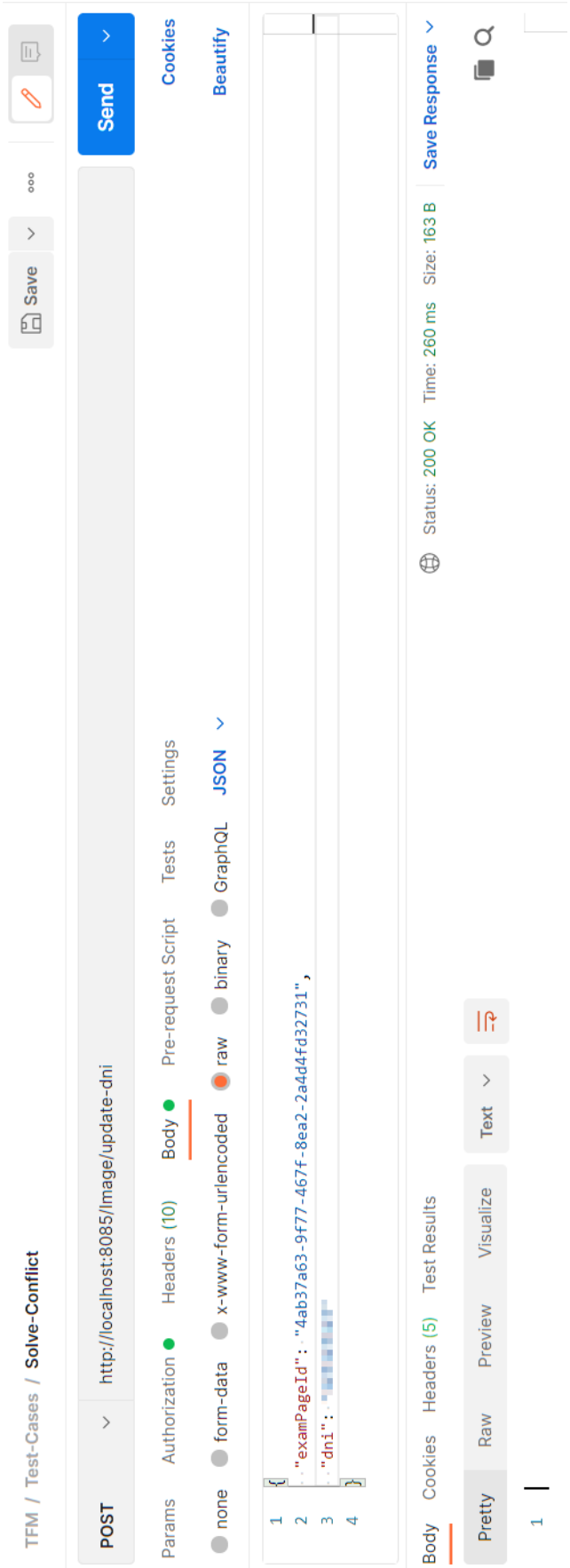
Obtener exámenes evaluados y clasificados:



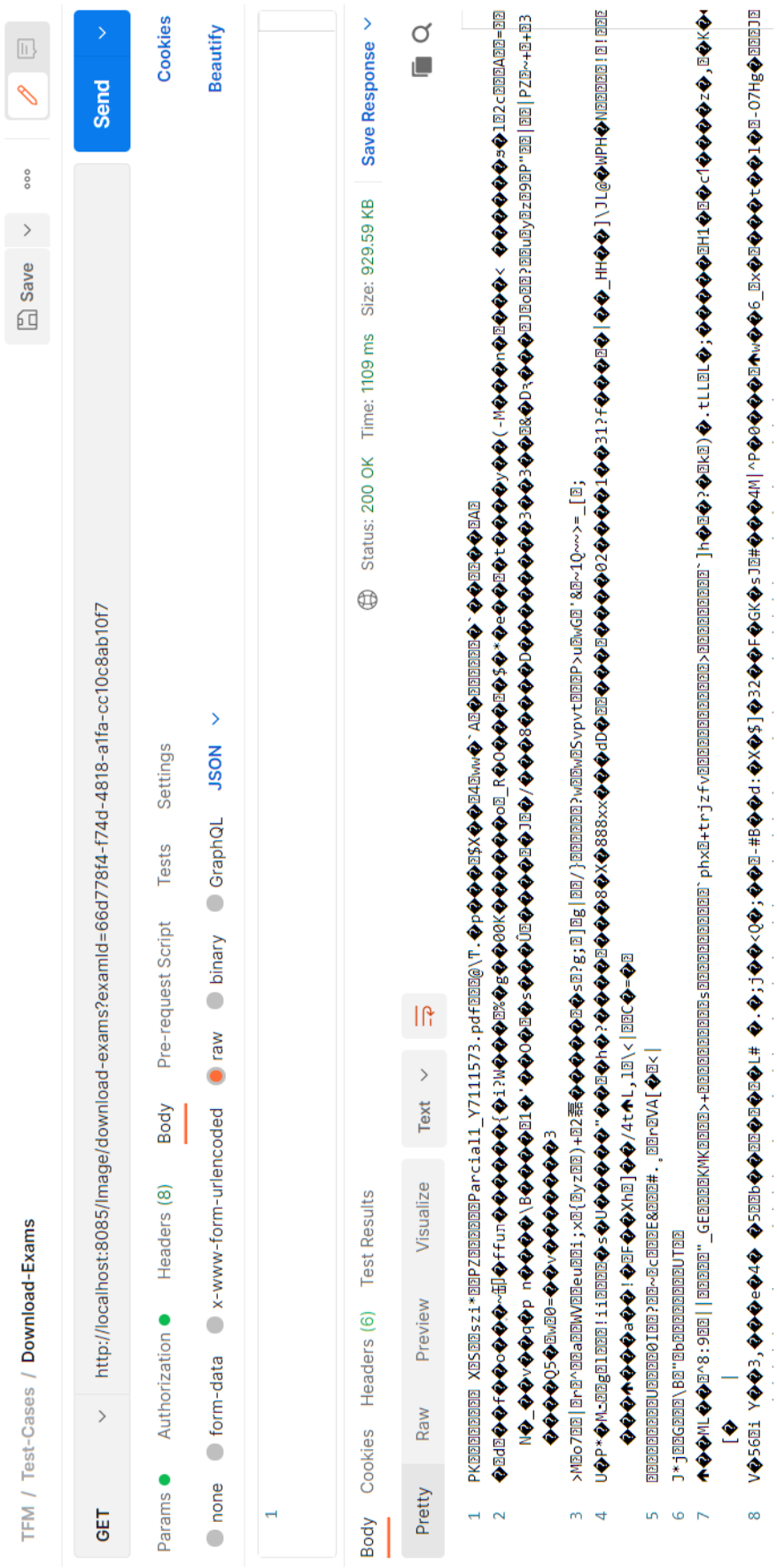
Obtener página de examen para resolver el conflicto de clasificación:



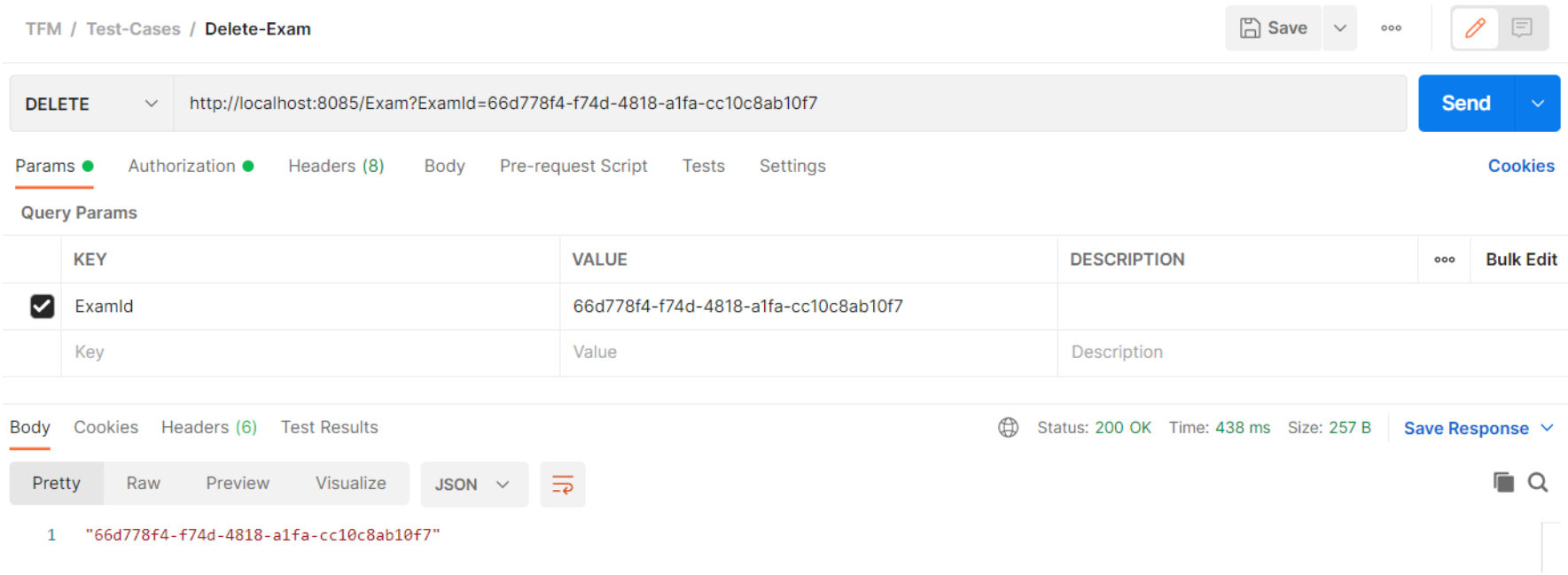
Resolver conflicto de clasificación:



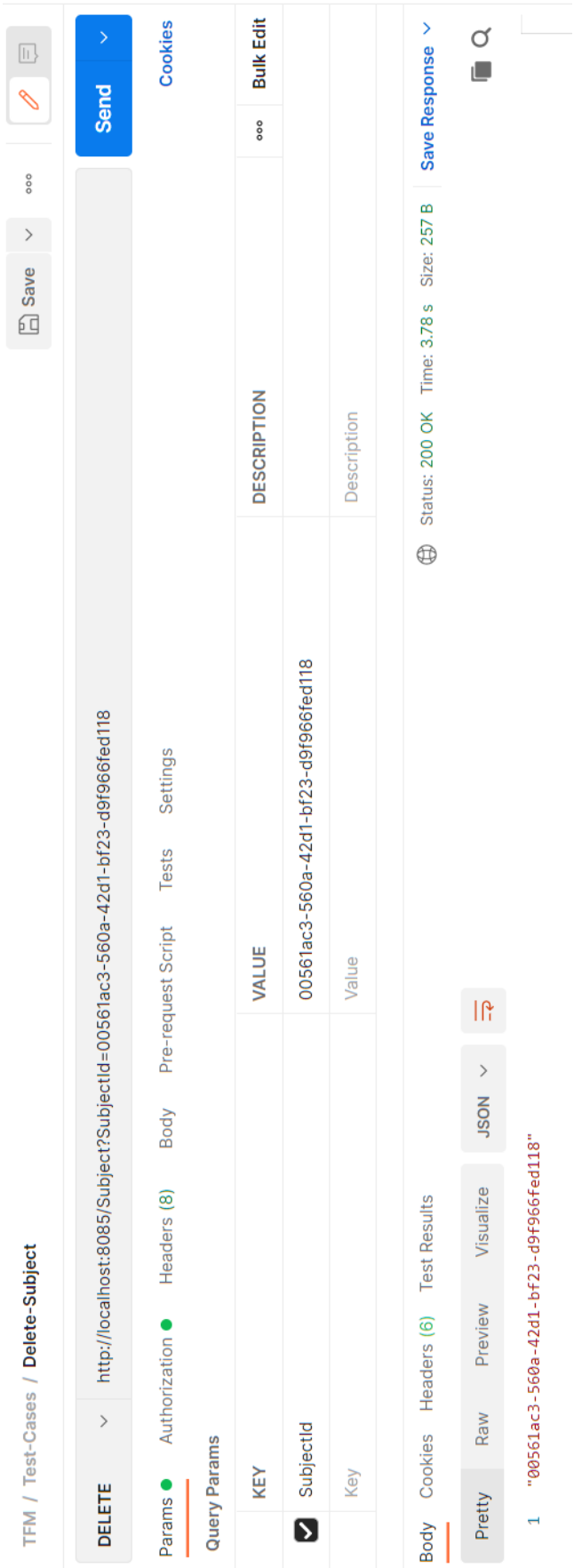
Descargar los exámenes clasificados:



Borrar Examen del sistema:



Borrar Asignatura del sistema:



12.2 Anexo 2. Plantilla examen

Para que la aplicación reconozca correctamente las páginas correspondientes al examen de un alumno, y pueda localizar el DNI manuscrito para reconocerlo, la cabecera de la primera hoja del examen debe seguir el formato que se presenta a continuación, y para las siguientes páginas es obligatorio que no aparezcan las palabras “Examen” ni “DNI”.

Nombre asignatura y demás información.	Examen parcial X dia-mes-año	DNI:
Apellidos:		Nombre:

Instrucciones:

- Escriba los apellidos y el nombre en todas las hojas. Hágalo ahora.