UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

# Asistente Web para la publicación de exámenes corregidos y calificaciones

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Caiza Jami, Jefferson Paul

Tutor/a: Martí Campoy, Antonio

CURSO ACADÉMICO: 2023/2024

# Resumen

Madrina es un asistente web que facilita a los profesores de la Universitat Politècnica de València (UPV) la devolución de exámenes manuscritos corregidos a través del campus virtual. En su versión actual, el asistente permite la clasificación manual de exámenes, proporcionando funcionalidades para simplificar el proceso. Este proceso manual supone que los profesores deben asignar cada examen a su respectivo alumno uno por uno, completando meticulosamente cada campo requerido. Este proceso, aunque simplificado, supone aún una actividad que puede resultar laboriosa y requerir una gran cantidad de tiempo.

Este proyecto se centra en la mejora y despliegue de una nueva versión del asistente web Madrina, cuyo objetivo principal es reducir el tiempo necesario para entregar al alumnado los exámenes y las calificaciones de los alumnos. Esta nueva versión se enfocará principalmente en automatizar la tarea de asignación de exámenes, que actualmente se realiza de forma manual.

La automatización se logrará mediante la integración de técnicas avanzadas de reconocimiento de imágenes, las cuales permitirán identificar la escritura a mano del DNI y la calificación, y asociar cada examen a su respectivo alumno de manera automática y eficiente. Con esta mejora, el profesorado solo necesitará verificar la correcta asociación, eliminando la necesidad de realizar esta tarea manualmente.

Además de la automatización, se implementarán otro tipo de mejoras en el asistente, que incluyen mejoras en la seguridad para proteger los datos, optimización en el rendimiento y ajustes basados en sugerencias del profesorado, que mejoren la funcionalidad y usabilidad del asistente.

**Palabras clave:** asistente, automatizado, correcciones, examen, calificaciones, Madrina.

# Abstract

Madrina is a web assistant designed to simplify the process of returning graded handwritten exams to students at the Universitat Politècnica de València (UPV) through the virtual campus. In its current version, the assistant facilitates manual exam classification by offering functionalities to simplify the process. This manual process requires professors to individually assign each exam to its respective student, completing each required field. While simplified, this task can still be laborious and time-consuming.

This project focuses on enhancing and deploying a new version of the Madrina web assistant. The primary objective is to reduce the time needed to return graded exams to students. This new version will primarily focus on automating the exam assignment process, which is currently performed manually.

Automation will be achieved by integrating advanced image recognition techniques. These techniques will enable the identification of handwritten ID and grades, automatically and efficiently associating each exam with its corresponding student. With this enhancement, professors will only need to verify the accuracy of the automatic assignments, eliminating the need for manual input.

Beyond automation, the project will implement other improvements, including enhanced security measures to protect data, performance optimization, and adjustments based on suggestions from professors to improve the functionality and usability of the assistant.

**Keywords:** assistant, automated, corrections, exam, grades, Madrina.

# Índice de contenidos

Índice de contenido

# Índice de ilustraciones

# Índice de tablas

# Glosario

**API (Application Programming Interface):** Interfaz de programación de aplicaciones que consiste en un conjunto de reglas y herramientas que permite a diferentes aplicaciones comunicarse entre sí.

**API RESTful:** API ajustada a los principios *Representational State Transfer* (REST) que, empleando métodos HTTP estándar (GET, POST, PUT, DELETE), proporciona una interfaz uniforme y sin estado para la comunicación entre sistemas.

**Back-end:** Componente de una aplicación que controla el procesamiento de datos y la lógica de negocio. Este componente no es visible para los usuarios.

**Clasificación automatizada:** En el contexto del proyecto, clasificación automatizada hace referencia al nuevo proceso de clasificación de exámenes, donde la clasificación será realizada por el propio asistente, dejando al usuario la única tarea de validar la información.

**Clasificación manual:** En el contexto del proyecto, clasificación manual hace referencia al modelo de clasificación actual en el asistente, donde la clasificación se realiza introduciendo los datos uno a uno manualmente.

**Cloud:** Modelo de computación que proporciona recursos a través de internet, permitiendo el acceso a servicios sin necesidad de gestionar infraestructuras propias.

**CNN (Convolutional Neural Network):** Red neuronal especializada en el análisis de datos con estructura de cuadrícula, como imágenes. Es empleada comúnmente en tareas como el reconocimiento de imágenes o la detección de objetos.

**CSV (Comma-Separated Values):** Tipo de documento de texto empleado para almacenar datos, en el que cada línea representa un registro y los valores están separados por comas.

**Framework:** Conjunto de herramientas y bibliotecas que proporcionan una estructura estandarizada para el desarrollo de software.

**Front-end:** Componente de una aplicación que gestiona la interfaz de usuario. Incluye el diseño y las funcionalidades visibles para los usuarios.

**Hilo:** Subproceso dentro de un proceso que permite ejecutar tareas de forma concurrente con otros hilos del mismo proceso, compartiendo recursos.

**Inteligencia artificial:** Tecnología que permite a los sistemas de software imitar funciones cognitivas humanas, como el aprendizaje, la toma de decisiones o la resolución de problemas.

**Madrina:** Asistente web destinado a la clasificación de exámenes por parte del profesorado para su posterior distribución a los estudiantes.

**OCR (Optical Character Recognition):** Conocido también como reconocimiento óptico de caracteres, es una tecnología capaz de convertir el texto de imágenes en texto editable.

**Padrino:** Herramienta de la UPV destinada a facilitar la gestión de alumnos de profesores de la UPV, permite la gestión de calificaciones y otros datos de interés.

**Perfilador de rendimiento:** Herramienta empleada para medir y analizar el rendimiento de una aplicación, identificando aquellos fragmentos que más tiempo de ejecución o llamadas requiera.

**PoliformaT:** Plataforma en línea de la UPV que permite, entre otras cosas, crear, gestionar y distribuir contenido educativo a estudiantes.

**Proceso:** Programa en ejecución que realiza tareas específicas dentro de un sistema operativo, empleando recursos como memoria y CPU.

**Red neuronal:** Modelo computacional compuesto por nodos o neuronas interconectados, empleado para reconocer patrones y resolver problemas complejos.

**Servicio:** Componente software que proporciona una funcionalidad especifica a otras aplicaciones o sistemas. Estos servicios no están diseñados para proporcionar resultados procesables por usuarios finales, por lo que son otros sistemas o aplicaciones los que los invocan.

**TCP (Transmission Control Protocol):** Protocolo de control de transmisión que asegura la entrega confiable y ordenada de datos en una red.

# 1 INTRODUCCIÓN

## 1.1 MOTIVACIÓN

Desde los inicios de la informática, la finalidad perseguida ha sido siempre reducir el esfuerzo de realizar tareas rutinarias que puedan ser programadas de forma precisa, así como minimizar aquellos errores derivados de la negligencia humana. En el caso de la clasificación y distribución de exámenes no podía ser distinto. Es bien sabido que los tiempos de corrección de exámenes pueden llegar a ser elevados, esto es en parte debido a que, además de la propia corrección manual del examen, se deben realizar otras tareas como la clasificación de exámenes, la suma de notas, la ordenación según criterios específicos o la revisión con los alumnos.

La revisión de exámenes es un desafío tanto para profesores como para alumnos, especialmente para asignaturas de primer curso, donde el número de alumnos es elevado. De ahí surge la necesidad de reducir estos números con estrategias como la entrega en línea de exámenes a alumnos. Según [1] la entrega de los exámenes a alumnos de forma individualizada y en línea es valorada positivamente por los alumnos, quienes además consideran que es útil para adquirir más conocimientos. El mismo artículo concluye que el profesorado considera que es una metodología enriquecedora, que además supone una reducción considerable de revisiones de examen presenciales, sin afectar de manera negativa a ninguna de las partes.

Es por eso por lo que la creación de herramientas que ayuden en la clasificación y distribución de exámenes a alumnos es indispensable, y beneficiaria tanto a profesores como a alumnos, reduciendo los tiempos que tardan en recibir la corrección de exámenes y añadiendo las ventajas implícitas de informatizar el proceso como recibir copias digitales de forma automatizada o consultar sus calificaciones en línea. Por otra parte, el profesorado también vería reducidos los tiempos invertidos en estas actividades, pudiendo dedicarlos a otras tareas como la creación de nuevos contenidos educativos.

## 1.2 OBJETIVOS

La aplicación, en su versión actual, se encuentra en uso actualmente de forma interna entre el profesorado de la UPV, mayoritariamente de la ETSINF, por lo que uno de los objetivos principales de este proyecto será mantener la funcionalidad actual intacta con el fin de que aquellos profesores que no quiera adaptarse al nuevo modelo puedan seguir utilizándola como han venido haciendo hasta ahora.

Los objetivos planteados son los siguientes:

- Mejorar los tiempos de clasificación de exámenes.
- Mejorar la seguridad del asistente de clasificación.
- Utilizar técnicas de inteligencia artificial para automatizar el proceso.
- Proporcionar un asistente intuitivo y respetuoso con el modelo de clasificación anterior.
- Reducir la carga de trabajo del profesorado.

## 1.3 IMPACTO

El proyecto se destinará al entorno académico, por lo que afectará principalmente a sus dos grandes grupos, profesorado y alumnado.

Cuando se trata de corrección, clasificación y entrega de exámenes, el profesorado tiene que invertir una gran cantidad de horas y esfuerzo fuera del horario de clases para poder entregar los exámenes corregidos. Estos profesores verían reducido el tiempo de clasificación y subida de

notas al hacerse de forma automática. Como consecuencia de dicha automatización, los profesores podrán enviar a cada alumno su respectivo examen con mucha más facilidad, evitando así largas sesiones de revisiones de exámenes y reduciéndolo a aquellos casos donde el alumno, comparando con la corrección oficial, detecte algún fallo.

Además, al ofrecer mejoras que pueden resultar útiles y atractivas, se fomenta el uso de la plataforma, abriendo su uso a más profesores de otros departamentos y escuelas, reduciendo globalmente el tiempo invertido en estas tareas.

El alumnado, a su vez, también se vería afectado positivamente, reduciendo los tiempos de entrega de exámenes de forma generalizada o haciendo más cómodas las revisiones, pudiendo realizarse de forma autónoma desde cualquier parte. Esto contribuye a que los exámenes sean valorados siempre de forma justa y objetiva, puesto que muchas veces, por cuestiones de planificación y horarios, resulta complicado acudir a revisiones presenciales, pudiendo omitir así posibles fallos de corrección, suma de puntuaciones, etc. Asimismo, aquellos alumnos que tengan que presentarse a exámenes de recuperación o que quieran aprender de sus errores, podrán conocer los fallos de sus exámenes, pudiendo centrarse en reforzar aquellos puntos débiles y fomentando así una educación de mayor calidad.

## 1.4 ESTRUCTURA

**Capítulo 1: Introducción**

En este capítulo se presenta el contexto general del proyecto, definiendo el problema a resolver, los objetivos que se pretenden alcanzar y el impacto que tendrá el proyecto.

**Capítulo 2: Estado del arte**

En este capítulo se realiza una revisión en profundidad de las tecnologías y herramientas existentes que puedan ser relevantes para el proyecto. Se analizarán soluciones similares disponibles en el mercado y sus características, ventajas y desventajas.

**Capítulo 3: Análisis**

Este capítulo está dedicado al análisis detallado de las necesidades del proyecto, con un enfoque en los distintos módulos que lo conformarán. Cada uno de estos módulos se analizan, identificando su funcionalidad y desafíos potenciales para posteriormente elaborar una propuesta de desarrollo. Además, se discuten las distintas metodologías de trabajo que podrían aplicarse, para finalmente justificar la elección de la metodología más adecuada y presentar un plan de trabajo que se adecue a ella.

**Capítulo 4: Requisitos del sistema**

En este capítulo se detallan los requisitos funcionales y no funcionales del sistema, sobre los cuales se basará el desarrollo. Se describen las funcionalidades clave que debe ofrecer el asistente, así como otras características como rendimiento o seguridad. También se incluyen los casos de uso, que ilustran como se relacionan los usuarios con el sistema y definen las interacciones y flujos de trabajo básicos para el funcionamiento del asistente.

**Capítulo 5: Diseño**

Este capítulo aborda el diseño del sistema, describiendo tanto la arquitectura general como el diseño detallado de cada módulo. También se presentan diagramas que ayuden a la comprensión

del diseño técnico, así como bocetos del diseño de interfaces, asegurando que todos los aspectos del sistema estén bien definidos.

**Capítulo 6: Desarrollo**

En el capítulo de desarrollo se documenta el proceso de implementación del proyecto, describiendo cómo se llevó a cabo la programación, las tecnologías empleadas y los posibles desafíos encontrados en cada unidad de trabajo definida. Al final de cada *sprint*, también se incluye una pequeña reflexión sobre el cierre de este, comentando el estado tras su finalización, retroalimentación obtenida y posibles aspectos a mejorar.

**Capítulo 7: Pruebas**

En este capítulo se llevan a cabo pruebas para verificar el correcto funcionamiento del sistema y si se alcanzan o no los objetivos planteados. Además, se presentan conclusiones y se proponen posibles mejoras basadas en los resultados obtenidos.

**Capítulo 8: Despliegue**

En el capítulo de despliegue se describe el proceso de instalación y puesta en marcha del proyecto en el entorno de producción, para ello, además, se proporcionan materiales adicionales que faciliten este proceso.

**Capítulo 9: Conclusiones**

Este capítulo ofrece una reflexión sobre los resultados del proyecto, evaluando si se han alcanzado los objetivos planteados. Además, se comenta el aprendizaje obtenido a lo largo del desarrollo del proyecto, destacando las habilidades adquiridas y los desafíos superados.

**Capítulo 10: Trabajos futuros**

En este capítulo se exploran las posibles direcciones para la evolución y mejora del proyecto. Se presentarán diversas opciones, así como sus posibles ventajas e inconvenientes.

**Capítulo 11: Bibliografía**

En este capítulo se recopilan todas las fuentes bibliográficas y documentos consultados a lo largo del proyecto y que sustentan el desarrollo de este.

**Capítulo 12: Anexo**

En el anexo se incluyen documentos adicionales que complementan el contenido principal, como instrucciones, archivos o información técnica relevante. Estos materiales no han sido incluidos en capítulos anteriores debido a su gran nivel de detalles o extensión.

# 2 ESTADO DEL ARTE

En esta sección se tratarán las alternativas existentes en el mercado para la entrega de exámenes a los alumnos, así como las que se encuentran ya en uso entre el profesorado, con el fin de poder analizarlas y ofrecer ventajas competitivas respecto a ellas.

Para este apartado, primeramente, es importante conocer el contexto actual de las herramientas en uso entre los profesores.

Por una parte, el asistente web Madrina, desarrollado en el trabajo de fin de grado *Asistente Web para la extracción y clasificación de exámenes desde ficheros PDF* [2].

Este asistente web se encarga de facilitar al profesorado la clasificación de exámenes y su posterior distribución de manera individualizada, tanto a la plataforma de calificaciones, conocida como Padrino, de la universidad, como a los alumnos, mediante el espacio compartido de PoliformaT[1]. Ofrece facilidades como lectura de archivos CSV con la lista de alumnos, subida de documentos con los exámenes corregidos, posibilidad de separar un documento con exámenes en varios documentos que estén asociados a su respectivo alumno, etc.

En las Ilustraciones 1 a 4, se muestran las pantallas más relevantes del asistente web, abarcando desde la creación de asignaturas hasta la clasificación manual del examen. Estas ilustraciones proporcionan una visión completa de las principales funcionalidades del asistente.

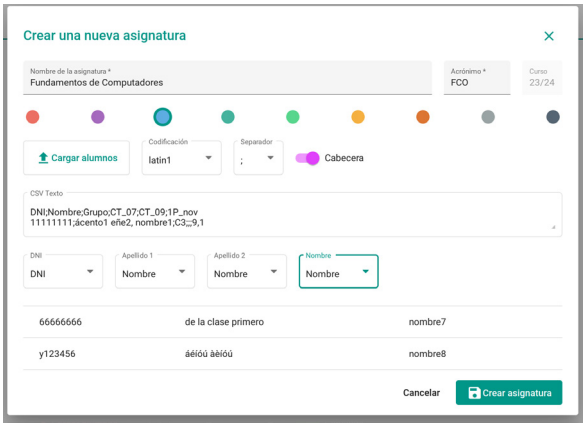



*Ilustración 1.* Creación de asignatura en Madrina

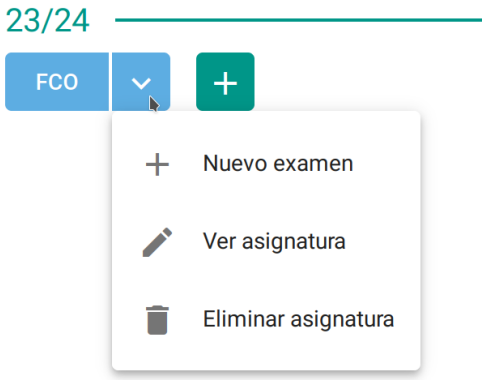



*Ilustración 2.* Menú de opciones en asignatura

[1] https://poliformat.upv.es/

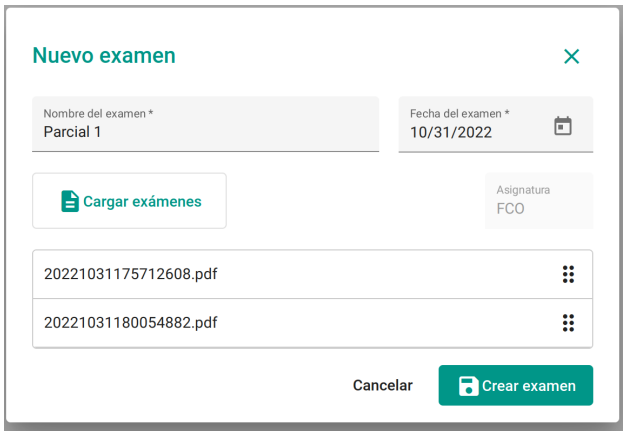



*Ilustración 3.* Creación de examen en Madrina

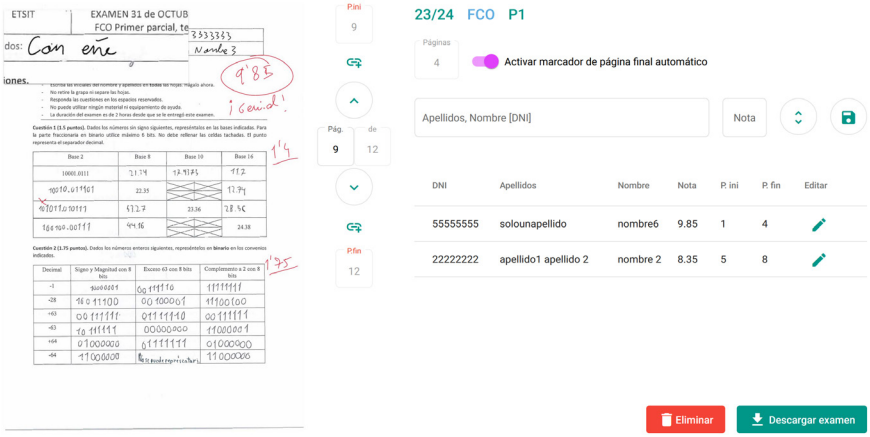



*Ilustración 4.* Interfaz principal del asistente web

Pero sin duda la pantalla más relevante es la mostrada en la Ilustración 4, lugar donde se realiza la clasificación manual. Su diseño es sencillo y está centrado en el usuario, ofreciendo una interfaz clara y concisa, separada en dos grandes bloques, por un lado, se muestra la imagen de la página del documento en el que se está posicionado, permitiendo leer datos como nombre, apellidos, DNI y nota del alumno. Por otro lado, a su derecha, se encuentran las herramientas propias del asistente de clasificación, ofreciendo una barra de búsqueda en la que se encuentran los alumnos que previamente hayan sido cargados, botones para seleccionar las páginas de inicio y fin del examen y una casilla para anotar la nota del alumno. Debajo de las herramientas se muestra una lista donde se irán añadiendo las clasificaciones de alumnos que se vayan realizando. Esta interfaz, además, ofrece atajos de teclado para cada acción, permitiendo al usuario no tener que utilizar el ratón y completar la clasificación tan solo empleando el teclado. Para asociar un alumno a un examen se tendrán que completar todos los campos de la sección de herramientas, datos del alumno, páginas y nota.

A este proceso de clasificación del asistente web en el que se tienen que añadir los alumnos uno por uno y campo a campo se le conocerá de ahora en adelante como clasificación manual.

En cuanto a la tecnología empleada para su desarrollo, para la parte visual o *front-end* se utiliza como tecnología principal Angular, un *Framework* de desarrollo web que aúna e integra tecnologías como *TypeScript*, *HTML* y *CSS*. Este se encargará de mostrar la información con la que interactuará directamente el usuario y de hacer llegar al servidor las peticiones que sean necesarias en función de las acciones del usuario. A su vez, la parte del servidor o *back-end,* aquella que se encarga de servir y procesar las peticiones que lleguen desde el *front-end,* está desarrollado utilizando Node.js, un entorno de ejecución de JavaScript caracterizado por su alto rendimiento y su funcionamiento asíncrono y no bloqueante.

Por otra parte, también es importante conocer el reconocedor de DNI manuscritos desarrollado en *Python application to process images of handwritten digits* [3]. Este proyecto desarrollado en Python[2] consiste en un *script* el cual, dada una colección de exámenes en formato PDF, una lista de alumnos en formato CSV y un archivo CSV donde guardar los datos, es capaz de asociar las páginas y DNI de cada examen a un alumno de la lista, guardando en el archivo CSV de salida el listado completo de asociaciones.

La Ilustración 5 muestra el fichero CSV obtenido tras la ejecución del *script* con archivos de prueba.

```
DNI;Name;Surname;First page;Last page
5632896;name1;ape1;1;2
19981998;name3;ape3;3;5
2348567;name5;ape5;6;9
2348567;name5;ape5;10;11
812579;name9;ape9;12;15
759248310;name11;ape11;16;18
54693116;name13;ape13;19;20
2373521;name15;ape15;21;25
1247958;name17;ape17;26;27
731983219;name19;ape19;28;33
237654396;name21;ape21;34;35
65484546G;name2;ape2;36;42
202329063;name4;ape4;43;45
-75210075;name6;ape6;46;51
Y9085409V;name8;ape8;52;52
481635029;name10;ape10;53;54
0124568;name12;ape12;55;56
7527291;name14;ape14;57;61
493952;name16;ape16;62;66
057821325;name18;ape18;67;68
750799457;name20;ape20;69;72
```

*Ilustración 5.* Ejemplo de resultados obtenidos en el fichero CSV

El *script* funciona de la siguiente manera; primero se transforma el documento PDF a imágenes PNG individuales de la parte superior derecha de la página, donde está contenido el DNI del alumno. Posteriormente, se procesa la imagen para poder detectar mejor la estructura definida para anotar el DNI y se hacen cálculos hasta encontrar los bordes de las casillas del DNI.

Una vez se sabe en qué páginas se encuentran las casillas del DNI, se recorta individualmente cada casilla de cada dígito del DNI y se realiza una predicción usando una red neuronal CNN, previamente entrenada. Identificar la estructura que contiene el DNI implica también conocer los límites de los exámenes, pues la página anterior será la última página del examen anterior, por lo que también se pueden inferir los datos de la longitud de todos los exámenes. Conociendo estos datos, solo queda juntarlos y volcarlos sobre el archivo CSV de salida.

[2] https://www.python.org/

Este proyecto está desarrollado completamente en Python y usa algunas librerías de apoyo como *OpenCV* [3]para el procesamiento de imágenes o *Torch*[4] para el desarrollo de inteligencia artificial.

Aunque ambas partes, asistente y reconocedor de textos, deban estar integrados en una sola aplicación, se analizarán de forma separada con el fin de poder compararlo con otras alternativas existentes que se ajusten mejor a los requisitos de cada parte. De esta manera, se podrá obtener un análisis más riguroso y se asegurará el cumplimiento de los objetivos del proyecto.

## 2.1 ASISTENTE DE CLASIFICACIÓN

Existen múltiples aplicaciones comerciales que podrían ser útiles para el propósito de este trabajo. De todas ellas, la más completa es Gradescope[5], una plataforma de gestión de exámenes en línea que permite escanear, enviar, calificar y analizar documentos en papel de forma rápida. Ofrece utilidades como reconocimiento de texto, automatizaciones, reconocimiento de patrones o entregas individualizadas. También admite distintos modelos de examen, pudiendo detectar textos matemáticos o incluso lenguajes de programación. Es utilizada por más de 2600 universidades en todo el mundo, demostrando su éxito en la clasificación y corrección de exámenes. Una de sus principales ventajas es la posibilidad de usarlo en cualquier dispositivo con un navegador y acceso a internet, lo que evita la instalación de software y mantiene la aplicación siempre actualizada. Además, ofrece una gran libertad al profesor, pues podría corregir y clasificar exámenes desde cualquier parte. Por otra parte, presenta el inconveniente de no ser una tecnología propia de la universidad, por lo que los exámenes tendrían que abandonar la custodia de la universidad. Además, se requiere pagar por los planes necesarios para poder utilizar la plataforma.

Por otro lado, en el contexto de la UPV, en las primeras experiencias de devolución de exámenes, el proceso se realizaba de forma manual. El profesor introducía las calificaciones y los exámenes en la plataforma de la universidad de forma manual, escaneando examen por examen y asignando el alumno y su calificación uno por uno. Era un proceso lento, útil únicamente cuando el número de exámenes era reducido.

Posteriormente, se desarrollaron herramientas informáticas, como *scripts* y hojas de Excel, que facilitaban el trabajo, aunque no eran suficientemente satisfactorias.

Con el fin de evitar costes económicos de aplicaciones comerciales y la distribución de exámenes de alumnos fuera del ámbito de la universidad, se desarrollaron herramientas propias en diferentes iteraciones como las desarrolladas [2] [4], hasta llegar a la última versión del asistente web Madrina[6], utilizada en la actualidad.

Sus principales ventajas son la considerable reducción de tiempo empleado en clasificar los exámenes, las facilidades que ofrece para separar exámenes individualmente y su funcionalidad para subir calificaciones y exámenes a las plataformas oficiales de la universidad. Además, se trata de una aplicación desarrollada y alojada dentro del entorno académico de la UPV, por lo que se garantiza su seguridad y custodia y no supone la contratación de ningún servicio externo, pudiendo mantenerlo en un servidor de la universidad. Algunas desventajas de esta alternativa son que, si bien han visto reducido el tiempo empleado, el profesorado ha observado que es posible reducirlo más utilizando, por ejemplo, técnicas de reconocimiento de textos. Por otra parte, presenta algunas carencias como la falta de soporte a determinados sistemas operativos o pequeños errores que no han llegado a corregirse al tratarse de un proyecto cuyo desarrollo fue

[3] https://pypi.org/project/opencv-python/
[4] https://pypi.org/project/torch/
[5] https://www.gradescope.com/
[6] https://madrina.webs.upv.es/

finalizado. Además, presenta algunos defectos en la seguridad del asistente, los cuales deberán ser corregidos.

En este punto es importante mencionar que la UPV dispone de una herramienta llamada ALCE[7], completamente integrada en PoliformaT, que permite corregir y entregar a los alumnos sus exámenes. Sin embargo, esta herramienta está destinada únicamente a exámenes de tipo test realizados en hojas mecanizadas, por lo que no es relevante para el proyecto.

## 2.2 RECONOCEDOR DE TEXTOS

Una de las tareas que más tiempo requieren en Madrina es la clasificación o asignación de un examen a su alumno, teniendo que realizarlo del profesor de forma manual seleccionando el número de páginas de inicio y fin, los datos identificativos del alumno y la calificación. Esta tarea podría automatizarse mediante el reconocimiento automático del nombre o DNI del alumno, así como la calificación del alumno. Algunas opciones para realizar dicho reconocimiento son:

**Google Cloud Vision API OCR**[8]

Se trata de un servicio avanzado de análisis de imágenes proporcionado por Google Cloud[9] que permite una fácil integración en todo tipo de aplicaciones. Utiliza modelos de aprendizaje automático para extraer información a partir de imágenes y ofrece características como etiquetado de imágenes, reconocimiento de texto, reconocimiento facial, entre otros. Esta alternativa resulta particularmente interesante, teniendo en cuenta su característica de reconocimiento de texto avanzado, sería posible ceder la parte de análisis de imágenes a este servicio y dejando como única tarea el integrar dicho servicio en la aplicación.

Por otra parte, su uso supondría el envío de imágenes de exámenes fuera del entorno académico de la universidad, poniendo en riesgo su seguridad. Además, hay que tener en cuenta el coste económico que supone su uso, que en caso de ser intensivo podría llegar a aumentar considerablemente. Finalmente, la utilización de cualquier servicio externo supone un riesgo potencial en cuanto a la disponibilidad de la aplicación, ya que en caso de que el proveedor tuviese cualquier tipo de problema de disponibilidad afectaría directamente a la utilización de la aplicación web, dejándola inoperativa.

**ABBYY FineReader**[10]

Consiste en un software de reconocimiento óptico de caracteres que permite convertir documentos escaneados en textos editables. Utiliza tecnologías avanzadas para identificar y digitalizar texto en múltiples idiomas, conservando el formato original del documento. Es especialmente útil para digitalizar documentos en papel o extraer datos de ellos. Su principal desventaja es el coste que supone su utilización. Además, aunque se obtiene el texto del documento, aún habría que procesar todos esos datos para poder obtener unos resultados adecuados que puedan ser cargados a la plataforma de calificaciones de la universidad.

**Adobe Acrobat Pro**[11]

Adobe Acrobat Pro es una herramienta avanzada para la creación, edición y gestión de documentos PDF. Entre sus numerosas funciones, destaca su capacidad de reconocimiento óptico de caracteres, que permitiría convertir los documentos con exámenes en archivos PDF editables,

[7] https://wiki.upv.es/confluence/display/MANUALES/Alce
[8] https://cloud.google.com/vision?hl=es_419
[9] https://cloud.google.com/?hl=es
[10] https://pdf.abbyy.com/es/
[11] https://www.adobe.com/es/acrobat.html

de los que se pueden extraer datos como el DNI o la calificación. Por otra parte, esta funcionalidad de reconocimiento de caracteres puede no ser demasiado fiable, ya que esta herramienta no está dedicada íntegramente al reconocimiento de textos. A esto hay que sumarle que, al igual que en la alternativa anterior, los resultados obtenidos deberán ser procesados para poder extraer la información estrictamente necesaria.

**Reconocedor automático desarrollado**

Otra alternativa es el reconocedor de DNI, desarrollado en varias iteraciones de trabajo [3] [5]. Este reconocedor utiliza la plantilla de examen definida en el Anexo 1, en la cual estará contenido el DNI del alumno. La plantilla se emplea para identificar las páginas iniciales de los exámenes e identificar al alumno, obteniendo como resultado los datos de la Ilustración 5. Sus principales ventajas son su utilización gratuita y la concisión de su funcionalidad, obteniendo únicamente la información necesaria del documento. Por otra parte, los datos obtenidos pueden contener errores al no presentar tecnologías tan avanzadas, por lo que será necesario verificarlos y, posteriormente, procesarlos para que se adecuen al formato de la plataforma de calificaciones de la universidad.

# 3 ANÁLISIS

El análisis realizado en este trabajo tiene dos objetivos. El primer análisis se centrará en plantear diferentes soluciones para añadir las nuevas funcionalidades definidas. El segundo análisis, dado que se parte de una aplicación existente, y aunque su funcionamiento es satisfactorio, tendrá como propósito detectar mejoras que puedan ser realizadas en aspectos de prestaciones y seguridad.

Debido a que existen varias alternativas para cada aspecto o componente que compondrá el proyecto, se mostrarán las soluciones propuestas separadas por módulos, de esta forma las alternativas se adaptarán mejor a los requisitos y funcionalidades individuales de cada componente. Las soluciones se dividirán entre asistente web, reconocedor de DNI y reconocedor de calificaciones.

## 3.1 ANÁLISIS Y SOLUCIONES PROPUESTAS

### 3.1.1 Asistente web

En una fase inicial y en comunicación directa con el profesor y tutor Antonio Martí Campoy, responsable del mantenimiento y resolución de problemas de la aplicación web Madrina, se barajan posibles soluciones para resolver el problema de los tiempos de clasificación de exámenes. Una de las primeras ideas fue desarrollar un nuevo asistente web construido desde cero. Esta idea, si bien puede ser interesante para evitar problemas de integración y coherencia respecto a los cambios a aplicar, quedó descartada debido al elevado coste temporal que tendría realizarlo. Por otra parte, el asistente actual se encuentra en uso, por lo que la creación de uno nuevo carecería de sentido si como mínimo no se puede llegar a cumplir con los requisitos de la versión actual.

Partiendo de eso, se concluye que para poder resolver el problema lo más conveniente será mantener como base la versión actual de la aplicación web Madrina y realizar sobre ella las modificaciones necesarias para cumplir con los nuevos objetivos establecidos.

A continuación, se detallan las distintas soluciones planteadas.

**Sustituir la forma actual de clasificación**

Esta alternativa consiste en modificar el funcionamiento actual de la aplicación web, sustituyendo el modelo manual por el modelo automatizado. Esto evitaría añadir una complejidad excesiva a su mantenimiento y desarrollo, pues no habría que preocuparse por afectar a algún componente que ya se encuentre en funcionamiento. También se reduciría la complejidad para los usuarios porque no tendrían que convivir con dos modelos distintos de trabajo.

Por otra parte, esta alternativa supondría la eliminación del flujo de trabajo de clasificación manual, afectando a aquellos usuarios que no vean necesario utilizar el nuevo modelo automatizado. Esto obligaría a los usuarios a adoptar al nuevo modelo, el cual podría no cumplir con sus expectativas como lo hacía el modelo anterior. También, para el correcto funcionamiento del modelo automatizado, será necesario el uso de una plantilla de examen que contenga los datos del DNI en la página inicial para facilitar su detección, lo cual obligaría a todos los usuarios actuales de la aplicación a adaptarse al nuevo modelo, sin posibilidad de seguir utilizando un modelo que funcionaba adecuadamente.

**Desarrollar un flujo de trabajo mixto**

La idea de un flujo de trabajo mixto implica la coexistencia de ambos modelos, siendo compatible el uno con el otro y quedando a elección del usuario el modelo a utilizar y la decisión de cambiar

de uno a otro. Esta alternativa ofrecería la máxima flexibilidad posible, ya que el usuario podría elegir cualquier modelo y cambiar al otro siempre que quiera. Un ejemplo claro sería que el usuario empezara con el modelo automatizado y tras recibir la asociación de exámenes se diera cuenta de que no había utilizado el formato de examen requerido y, por tanto, el reconocimiento de alumnos no fuera correcto. En este caso tan solo tendría que retroceder y abrirlo con la vista disponible actualmente, para clasificarlos de forma manual. Esta solución sería muy apropiada para ese tipo de errores de formato o estructura, puesto que no se tendría que borrar el examen y crear otro desde cero, sino que bastaría con alternar al otro modelo, eliminando los tiempos de procesado que suponen la creación de uno nuevo.

Esta alternativa, si bien puede resultar muy cómoda para el usuario, puede aumentar de forma considerable la complejidad del proyecto y su desarrollo. No solo habría que crear el nuevo modelo automatizado, sino que también habría que adaptar el modelo actual para que soporte nuevas características, añadiendo nuevos elementos en la interfaz, nueva comunicación entre *front-end* y *back-end* o nuevos campos en la base de datos. Esto podría causar conflictos e incoherencias de datos, poniendo en peligro no solo el nuevo desarrollo automatizado, sino también el actual modelo manual, pudiendo provocar pérdida de datos, bloqueos y reinicios inesperados.

**Desarrollar un flujo de trabajo independiente**

Otra alternativa sería desarrollar un nuevo flujo de trabajo que coexista con el modelo manual, pero totalmente independiente, sin posibilidad de cambiar de modelo una vez creado el examen. Para ello se modificaría el proceso de creación del examen, incorporando una opción que permita elegir entre el modelo manual y automatizado. En función de la elección, al acceder a dicho examen, se abrirá con su respectivo modelo, manual o automatizado.

Esta opción resulta muy interesante, ya que no se alteraría el modelo manual y, por tanto, no afectaría al comportamiento de este, asegurando que los usuarios puedan seguir usándolo en caso de no querer adaptarse al nuevo. A esto hay que sumarle que, al tratarse de modelos completamente independientes y que en ningún momento llegan a intercambiar o usar los mismos datos, resulta imposible que se produzcan inconsistencias de datos o incompatibilidades, aportando seguridad y robustez a la aplicación.

Por otro lado, considerando la posibilidad de que la clasificación automatizada pueda cometer fallos en sus respuestas, es necesario proporcionar al usuario un procedimiento sencillo para verificar las clasificaciones automatizadas, y en caso necesario, corregirlas. Para ello, será necesario proporcionar nuevas herramientas en la interfaz, que únicamente se podrían añadir sin afectar al modelo manual empleando un proceso completamente independiente.

La principal desventaja de esta alternativa es su falta de flexibilidad hacia el usuario, obligándolo a elegir el modelo a utilizar durante su creación y teniendo que mantenerlo hasta la eliminación del examen. Esto implica que, en caso de querer utilizar otro modelo, deberán crear un examen nuevo.

### 3.1.2 Reconocedor DNI

**Desarrollar un reconocedor de textos manuscritos**

El desarrollo de un reconocedor de textos desde cero es un proyecto de gran complejidad que implica no solo su implementación, sino también un análisis previo de tecnologías existentes, comparativas de rendimiento, estudio del contexto de uso, entrenamiento del modelo, entre otros. Esta alternativa resulta interesante ya que ofrecería la mayor adaptabilidad posible, pudiendo crear

un reconocedor de textos específicamente para el contexto de uso en el que se desarrollará el proyecto.

Las ventajas de esta alternativa son múltiples, siendo las principales, un control total sobre el proyecto, asegurando una correcta integración y un comportamiento limitado a las necesidades del proyecto. Por otra parte, la ausencia de dependencias a servicios externos garantizaría su disponibilidad y aseguraría que datos confidenciales, como exámenes o identificaciones, no salgan del ámbito de la universidad.

La implementación de esta alternativa implica, además de la propia implementación, un análisis y una formación previa que requeriría una gran cantidad de tiempo, llegando a poner en peligro la entrega final del proyecto o bien afectando a la calidad de las asociaciones de exámenes por haber tomado decisiones de forma apresurada. Esto supone un gran riesgo para el proyecto y, por tanto, deberá tenerse en cuenta para la propuesta de implementación final.

**Integrar servicios *cloud* externos**

Para esta alternativa habría que emplear algún servicio de reconocimiento de textos como Google Cloud Vision API, mencionado anteriormente. Para ello se necesitaría contratar alguno de los planes de Google Cloud y posteriormente integrar la API en el servidor *back-end*. Su utilización sería relativamente sencilla, se tendría que hacer un recorte de la imagen donde se encuentre contenida la información del DNI, se enviaría la petición al servidor mediante la API y se esperaría la devolución de la lectura del DNI. Esta opción ofrece ventajas como un reconocimiento de imágenes avanzado, utilizando técnicas de aprendizaje automático, y una gran capacidad de procesamiento, capaz de soportar casi cualquier carga de trabajo. Además, eliminaría por completo la implementación de toda la lógica detrás del reconocimiento de imágenes, por lo que sería una opción muy rápida de implementar.

Por otra parte, al tratarse de un servicio externo, habría que enviar información confidencial fuera del entorno de la universidad, rompiendo la custodia de los exámenes y poniendo en riesgo la confidencialidad de exámenes y datos de alumnos. A esto hay que sumar que el modelo automatizado dependería de un servicio de terceros, pudiendo no estar disponible en algún momento y dejando inoperativo el modelo automatizado hasta su restablecimiento. Por último, otro punto importante es el coste económico que supone su utilización, este tipo de servicios tienen un coste variable en función de las consultas realizadas, en caso de un uso intensivo este coste podría ser inasumible.

**Adaptar e integrar el reconocedor de DNI**

Otra opción es adaptar el *script* reconocedor de dígitos desarrollado por [3]. Este *script*, desarrollado en Python, es capaz de reconocer los dígitos del DNI de una plantilla predefinida en la que se anotan los dígitos del DNI en casillas. Para esta alternativa será necesario integrar el *script* en el servidor *back-end* para que puedan comunicarse correctamente y transmitir la información, de manera que ambos puedan comprenderla y trabajar adecuadamente. Su integración supondrá un cambio en la estructura general del *script*, adaptándolo para que utilice las mismas estructuras de datos que el servidor *back-end*. Además, habrá que crear un canal de comunicación entre ambos para poder enviar información como el estado de ejecución, la detección de errores y finalmente los resultados obtenidos.

La Ilustración 6 muestra la salida por consola del *script* reconocedor de DNI en su versión actual.


*Ilustración 6.* Salida por consola de la ejecución del script

Como se puede observar para un documento PDF de 72 páginas y 21 alumnos detectados tarda aproximadamente tres minutos y medio en un caso de ejecución individual y disponiendo de todos los recursos del sistema para su ejecución, a lo que también habría que añadir el tiempo que tarda el *script* en importar todas sus dependencias y la conversión de PDF a PNG, que no se incluye en la salida por consola. En el contexto de uso dentro de la aplicación Madrina, las condiciones son bastante diferentes, siendo necesarias varias ejecuciones del *script* de forma simultánea según la cantidad de usuarios que se encuentren utilizando la aplicación. A esto se suma que, para asignaturas con grupos de alumnos especialmente elevados, como en asignaturas de primer curso del grado, el número, tanto de páginas como de alumnos, aumentará considerablemente. Esto supondrá un aumento drástico en los tiempos de respuesta del *script*, pudiendo llegar a incomodar al usuario y acabar utilizando el modelo manual. Es por esto por lo que será necesario utilizar técnicas de optimización sobre el *script,* tales como la paralelización o la optimización de la complejidad de algoritmos.

Utilizar el *script* de Python para reconocer los DNI manuscritos, presenta múltiples ventajas. Entre ellas destaca la protección de información confidencial, ya que la ejecución del *script* reconocedor se podrá alojar en el mismo servidor en el que se aloja el servidor *back-end*. Como consecuencia, su utilización será gratuita, a expensas del costo que supone el servidor y su mantenimiento. También se contempla que el reconocedor de textos ya esté implementado, reduciendo el tiempo de desarrollo considerablemente y limitándolo a su optimización e integración en el proyecto.

Por otra parte, la integración de un proyecto ya realizado conlleva ciertas desventajas, como un aumento de la complejidad global del proyecto debido a la utilización de nuevas tecnologías. También hay que tener en cuenta que comprender el código de otra persona siempre representa un desafío y puede requerir un tiempo considerable para comprenderlo en su totalidad, especialmente cuando se trata de librerías avanzadas como las destinadas al reconocimiento de imágenes. Finalmente, la optimización del *script* puede resultar sencilla en un primer intento con técnicas como la paralelización, que no requieren modificar su estructura. Aunque, en caso de no ser suficiente, se tendrán que utilizar técnicas más avanzadas para localizar aquellos puntos del algoritmo que más recursos consumen y modificarlos. Estas técnicas avanzadas, además de tomar una cantidad de tiempo moderada para detectar aquellos puntos débiles, sí que requieren modificar la estructura del algoritmo, pudiendo afectar a su correcto funcionamiento y empeorando la calidad de los resultados.

### 3.1.3 Reconocedor calificación

En este apartado se detallan las alternativas destinadas al reconocimiento de la calificación del examen, el propósito en sí es el mismo que el del reconocedor de DNI, detectar dígitos de un texto manuscrito, por lo que podrán compartir algunas alternativas.

**Desarrollar un reconocedor de dígitos**

Esta alternativa es similar a la del desarrollo del reconocedor para DNI e incluso podría usarse el mismo desarrollo para el mismo propósito con adaptaciones para reconocer los dígitos de la calificación del alumno, por tanto, tendrá las mismas ventajas y desventajas que las mencionadas anteriormente.

**Desarrollar un nuevo reconocedor a partir del reconocedor de DNI**

El desarrollo de un reconocedor de dígitos desde cero requiere una gran inversión de tiempo tanto en formación del desarrollador, como en análisis e implementación. Por ello, una alternativa bastante prometedora es la implementación del reconocedor de dígitos, pero a partir de la base de otro. Para ello se usará como base el *script* reconocedor de dígitos de DNI [3], dejando únicamente la red neuronal capaz de detectar predicciones a dígitos individuales y el modelo entrenado de la misma, puesto que el entrenamiento es una de las fases que más tiempo tomaría. Sobre esta base se implementará un nuevo módulo reconocedor que será capaz de identificar una nueva estructura de exámenes predefinida, con un campo especial para anotar la calificación, para posteriormente reconocer los dígitos y realizar una predicción que pueda ser enviada al usuario.

Según [6], algunos de los límites del aprendizaje automático son las grandes cantidades de datos de entrenamiento que se necesitan y el tiempo que conlleva el etiquetado de estos. Con esta alternativa se evita tener que realizar de nuevo este proceso de recolección y entrenamiento, que suponen una gran cantidad de tiempo y ponen en riesgo la entrega final de proyecto. El núcleo sobre el que está construido el *script* reconocedor de DNI se trata de una red neuronal CNN que reconoce dígitos de forma individual a partir de una imagen procesada que contenga un único digito manuscrito, por lo que se reutilizará esa parte y se construirá sobre él un nuevo módulo capaz de reconocer calificaciones.

Para esto, primero será necesario obtener los dígitos individuales de la calificación, pero será imposible usar la anterior implementación que realizaba esta tarea, pues se trata de una estructura nueva con características completamente distintas, el anterior modelo utilizaba nueve dígitos y el nuevo necesitará cuatro dígitos separados por una coma central. Por ello, será necesario implementar un nuevo algoritmo que sea capaz de reconocer el patrón de la estructura que contiene la nota para poder detectar y recortar la imagen con la calificación para posteriormente hacer el reconocimiento con la red neuronal.

Al analizar el código del *script* original, se observa que la detección de los bordes de la estructura contenedora de los dígitos emplea métodos de manejo de imágenes que pueden consumir una cantidad elevada de recursos, por lo que a la hora de implementar el nuevo algoritmo que reconozca la estructura se tendrá en cuenta para evitar ese tipo de funciones, empleando otro tipo de librerías o accediendo a las imágenes de formas distintas.

La ventaja principal de esta alternativa es la reducción considerable de tiempo de desarrollo, pues se evitaría una de las etapas que más tiempo consumen, que es la recolección de datos y entrenamiento de la red neuronal. Además, al tener el código fuente, puede ser ejecutado en un servidor de la universidad, con las ventajas que supone, como el reducido coste de mantenimiento o la confidencialidad de exámenes y datos de alumnos, ya que esta información no saldría de la universidad.

Por otra parte, esta alternativa puede suponer que se agrave alguno de los problemas que presenta el reconocedor de DNI, como un posible bajo rendimiento debido a que no fue probado en el mismo contexto de uso que en el que se encuentra ahora, por lo que una mayor carga de trabajo puede hacer que los tiempos de procesamiento sean considerablemente elevados. Que los tiempos de procesado sean mayores que lo que tarda un usuario haciendo una clasificación manual supone un inconveniente, pues el usuario podría preferir el modelo manual. Si esto llegase a ocurrir, sería necesario optimizar su funcionamiento para que el tiempo dedicado sea igual o menor al tiempo que se tarda en clasificar un examen usando el modelo manual.

La optimización del *script* puede resultar relativamente sencilla, pues en todo momento se trabaja con datos distintos y algunas técnicas como la paralelización no tendrían problemas para ser implementadas.

Si las técnicas iniciales no resultasen suficientes, la complejidad de la optimización aumentaría, ya que sería necesario identificar aquellos fragmentos que producen mayor pérdida de rendimiento e implementarlo de manera que consuman menos recursos.

**Integrar servicios *cloud* externos**

Esta alternativa es similar a la misma ofrecida para el reconocedor de DNI, por lo que se puede asumir que tendrá las mismas ventajas y desventajas mencionadas en él.

## 3.2 PROPUESTA DE DESARROLLO

Tras analizar las posibles soluciones definidas anteriormente para cumplir los objetivos planteados, se hace una propuesta de desarrollo teniendo en cuenta las ventajas y desventajas de cada alternativa para cada módulo.

Respecto al asistente web, se concluye que la mejor alternativa es desarrollar un flujo de trabajo nuevo e independiente sobre el asistente web Madrina.

Esta alternativa es la única que no penaliza a aquellos usuarios que quieran seguir utilizando el modelo manual, lo cual es un punto importante, ya que habrá profesores que no quieran utilizar el modelo automatizado, bien por desconfianza o bien por no querer adaptarse a la nueva estructura de examen, que incluye espacios reservados para DNI y calificación. Si bien puede ser una alternativa poco flexible hacia el usuario en cuanto a la posibilidad de alternar entre modelos, no tendría por qué ser un factor decisivo, ya que una vez se usen las plantillas y formatos de examen adecuados solo se podrían producir pequeños errores puntuales de asociación de exámenes. En el peor de los casos se obtendría una clasificación automatizada vacía, que obligaría al usuario a realizar las asignaciones manualmente. Dado que la interfaz tendría elementos y funcionalidades similares a la clasificación manual, el usuario se encontraría en una situación equivalente a cuando se hace una clasificación utilizando el modelo manual, por lo que no vería reducidas las funcionalidades básicas, pudiendo clasificar el examen con normalidad de la misma manera que se hacía con el modelo manual.

También es importante destacar que, al tratarse de una implementación añadida y completamente independiente de la implementación actual, se evitarán por completo errores de incompatibilidades o inconsistencias de datos, que sí que podrían producirse en las otras alternativas. De esta manera se deja intacto el modelo manual, ya que, por muy pequeños que puedan ser los cambios, pueden alterar considerablemente su funcionamiento.

Por otra parte, es importante mencionar que, aunque se vaya a desarrollar la automatización sobre el asistente web existente, y este esté en uso en la universidad, se han detectado algunos puntos de mejora relacionados con la seguridad e integridad de datos. Entre ellos están, la codificación de las credenciales de acceso en el código del proyecto o los accesos poco seguros a la base de datos, donde están contenidos los datos de los alumnos. Todos estos fallos, así como otros posibles defectos que se vayan encontrando, serán corregidos para garantizar una experiencia segura para los usuarios.

Respecto al reconocedor de DNI, la alternativa de creación de un nuevo reconocedor fue descartada por suponer una cantidad de tiempo elevada y por poner en riesgo la entrega final del proyecto, siendo este uno de los puntos más importantes en cualquier tipo de desarrollo. La integración de servicios *cloud* para el reconocimiento de imágenes también fue descartada, principalmente por el coste económico, pero también por suponer un riesgo potencial para la seguridad e integridad de datos confidenciales de alumnos, así como agregar una dependencia de un tercero que puede afectar a la disponibilidad del asistente web. Por tanto, la alternativa a implementar será la adaptación e integración del script reconocedor de DNI. Esta alternativa,

aunque pueda suponer algunos inconvenientes como el tener que adaptar el código o que sean necesarias varias mejoras de rendimiento, sigue siendo la mejor de las opciones al no tener que realizar aquellas fases que más tiempo suelen tomar durante el desarrollo de inteligencias artificiales como son la recolección de datos o el entrenamiento de esta.

Para el caso del reconocedor de calificaciones, las alternativas de un desarrollo nuevo y la integración de un servicio *cloud* quedan descartadas por los mismos motivos que en el caso del reconocedor de DNI. Esto nos deja únicamente con la opción de la implementación del reconocedor de calificaciones a partir del reconocedor de DNI. Esta opción supone la reutilización únicamente de la red neuronal entrenada, lo que, por una parte, puede suponer una carga de trabajo elevada al tener que implementar la mayor parte del código, pero que, por otra parte, abre una nueva posibilidad de construirlo teniendo en cuenta los fallos detectados en el *script* original, eliminándolos así por completo. Contando con esos cambios, se trataría de una solución muy sólida en la que no se dependería de servicios de terceros y que ya contaría con un rendimiento optimizado al conocer previamente qué fragmentos del *script* original producían mayor pérdida de rendimiento.

## 3.3 METODOLOGÍA

Para definir las distintas metodologías consideradas para su utilización durante el desarrollo del proyecto es importante primero conocer el contexto de las metodologías existentes y diferenciarlas entre metodologías ágiles y clásicas.

En una metodología clásica normalmente los requisitos y la planificación del proyecto están bien definidos desde el inicio, proporcionando una estructura lineal que se deberá seguir rigurosamente durante el ciclo de vida del proyecto. Por otra parte, las metodologías ágiles se centran en el desarrollo iterativo e incremental, realizando planificaciones centradas en entregas en periodos cortos de tiempo que aporten valor al producto, de esta manera, al no tener una planificación estricta, permite adaptarse rápidamente ante cualquier cambio.

Para el desarrollo de este trabajo se plantean las siguientes metodologías estandarizadas, de las cuales se elegirá una, que se aplicará durante el desarrollo del proyecto.

### 3.3.1 SCRUM

SCRUM [7] es un marco de trabajo destinado al desarrollo de software y perteneciente a las llamadas metodologías ágiles. Esta metodología se centra en hacer entregas continuas e incrementales, permitiendo recibir una rápida retroalimentación y una detección temprana de errores.

Sus orígenes se remontan a principios de los años 80, cuando fue identificado y definido en [8], pero fue con [9] que se definirían las reglas para el desarrollo de *software* basado en los principios SCRUM.

Una de sus características principales es la división del tiempo en *sprints* de una duración determinada y constante, en la que previamente se habrá elegido de una lista las tareas a realizar en cada *sprint*. Esta lista de tareas, también conocida como *backlog,* es una lista formada por todas las unidades de trabajo pendientes de desarrollar y ordenada en función de su prioridad, que se decidirá en función del valor que suponga para el cliente.

Esta metodología tiene tres roles bien definidos: *Product Owner*, *SCRUM Master* y desarrollador.

El *Product Owner* (PO) será aquel responsable de maximizar el valor del producto en desarrollo, haciendo que se cumplan y respeten los intereses de los clientes desde un punto de vista de negocio. Este rol apoyará en el desarrollo y la comunicación de los objetivos del producto, así

como en la gestión del *backlog* del proyecto, bajo la supervisión y alineación con la dirección general del proyecto. El *Product Owner* será una única persona, aunque puede delegar responsabilidades en otras personas.

Este rol es de mucha utilidad, ya que permite reducir ambigüedades o diferencias respecto al producto final deseado, pues este rol representa al conjunto de las partes interesadas en el desarrollo del proyecto.

El *Scrum Máster* será el encargado de hacer que se comprendan y adopten las pautas y principios propios de la metodología SCRUM. Para ello, facilitará el desarrollo en la medida que sea posible, capacitando a los miembros del equipo en autogestión, eliminando posibles bloqueos, haciendo de intermediario entre miembros del equipo y garantizando que se lleven a cabo los eventos propios de la metodología.

Finalmente, los desarrolladores, serán aquellas personas encargadas de crear un incremento útil en cada *sprint.* Desempeñan funciones como crear un plan para el *sprint* y el *backlog* o aportar calidad al producto durante su desarrollo, respetando los principios de la metodología.

#### *3.3.1.1 Ventajas*

Una de las principales ventajas de esta metodología es su gran flexibilidad, al tener una estimación inicial de tiempo para cada unidad de trabajo y ordenando la lista de tareas por prioridad, nos aseguramos de realizar una entrega que suponga un incremento en la funcionalidad del producto, pudiendo incrementar o reducir la prioridad de cada unidad de trabajo, pero siempre manteniendo el total de horas cerca de la capacidad teórica de trabajo del equipo, ayudando así a que se cumplan los plazos de entrega.

Otra ventaja implícita en la metodología y asociada al rol de *Scrum Master* es el alto grado de transparencia y la comunicación continua. Esto es debido a sus aportes como reuniones o intermediaciones, que ayudan a conocer los posibles problemas que vayan surgiendo y así identificarlos en una fase temprana y ofrecer una rápida respuesta. Además, el rol de *Product Owner,* en representación de todos los interesados en el desarrollo del proyecto, garantizará que se respeten todos los requisitos necesarios para satisfacer a las distintas partes interesadas, evitando así posibles disconformidades en las entregas.

#### *3.3.1.2 Desventajas*

SCRUM es una metodología que requiere compromiso y dedicación por parte del equipo, pero, sobre todo, disciplina de trabajo, especialmente en las etapas iniciales de su implantación. Si no se ha utilizado previamente la metodología puede resultar complicado adaptarse a actividades como el registro de horas de trabajo, la estimación de horas, la especificación de requisitos al principio de cada *sprint*, las reuniones diarias o la comunicación constante con otros desarrolladores y con el *Product Owner*.

Por otra parte, incluso habiendo puesto en práctica la metodología previamente, puede haber un periodo de adaptación elevado, pues cada proyecto es distinto y hay habilidades, como la capacidad para estimar horas de trabajo, que únicamente se pueden llegar a perfeccionar basándose en experiencias previas dentro de ese proyecto y comprobando si se cumplen los plazos de entrega.

### 3.3.2 *Extreme Programing* (XP)

La metodología *Extreme Programing* [10] es una metodología ágil enfocada en mejorar la calidad del software y la capacidad de respuesta a cambios en los requisitos de clientes. Es especialmente adecuada para proyectos donde los requisitos cambian con frecuencia o no pueden ser definidos al principio.

Fue desarrollada por Kent Beck y finalmente publicada en [11]. Según este libro, la metodología se rige por cinco valores, entre los que se encuentran la comunicación continua entre todos los implicados en el proyecto, la simplicidad en el desarrollo para evitar funciones innecesarias, la retroalimentación rápida mediante entregas frecuentes, la valentía para estar dispuesto a efectuar cambios que puedan producir complicaciones y el respeto hacia el equipo y su toma de decisiones.

Comparte características con *SCRUM*, como sus entregas constantes o su flexibilidad ante cambios. Algunas de sus características más distintivas son la programación en parejas, donde dos desarrolladores trabajan en conjunto sobre una misma tarea, permitiendo que validen sus desarrollos mutuamente, o el desarrollo dirigido por pruebas, que prioriza la creación de pruebas automatizadas antes del desarrollo del código, permitiendo que las tareas puedan ser validadas de forma objetiva con mayor facilidad desde el inicio del desarrollo.

Otra característica importante es su refactorización del código implícita, que establece que siempre que se pueda se reescribirán partes del código para aumentar su legibilidad, pero sin afectar a su comportamiento.

#### *3.3.2.1 Ventajas*

Una de las ventajas de esta metodología es la flexibilidad y adaptabilidad ante posibles cambios, pudiendo actuar rápidamente ante cualquier cambio de requisitos. Por otro lado, el desarrollo dirigido por pruebas asegura que cada desarrollo sea probado adecuadamente desde el principio y permitiendo que se realicen periódicamente pruebas para comprobar que todo sigue funcionando correctamente.

#### *3.3.2.2 Desventajas*

Algunas características de esta metodología no pueden adoptarse en grupos muy reducidos como la programación en parejas. Por otra parte, en el caso de proyectos heredados o cuyo desarrollo ya haya sido iniciado en una iteración anterior, donde no se hayan aplicado estos principios, se puede requerir mucho tiempo para comprender el código y adaptarlo a los principios de la metodología.

Otro aspecto a considerar es el impacto del desarrollo dirigido por pruebas. Aunque proporciona una manera fácil de verificar que todo funciona correctamente y permite tener una manera de probar que no se hayan introducido errores en otras funcionalidades a medida que avanza el desarrollo, también representa una inversión de tiempo considerable. Esto, en proyectos con tiempos de desarrollo muy limitados, puede resultar un inconveniente y afectar a las fechas de entrega.

Finalmente, la adopción de la metodología supone un grado alto de conocimiento y experiencia en al área del desarrollo, la falta de planificación excesiva, implícita en la metodología, supone que el desarrollador tenga que tomar decisiones rápidas y autónomas en el momento del desarrollo. En caso contrario, se podrían tomar decisiones equivocadas y perjudicar al desarrollo gravemente.

### 3.3.3 Desarrollo en cascada

El desarrollo en cascada [12] es una metodología clásica, donde el proceso se divide en fases secuenciales. Estas fases serán todas las etapas del ciclo de vida del desarrollo de software y se deberán ejecutar siempre en orden, habiendo finalizado previamente la etapa anterior.

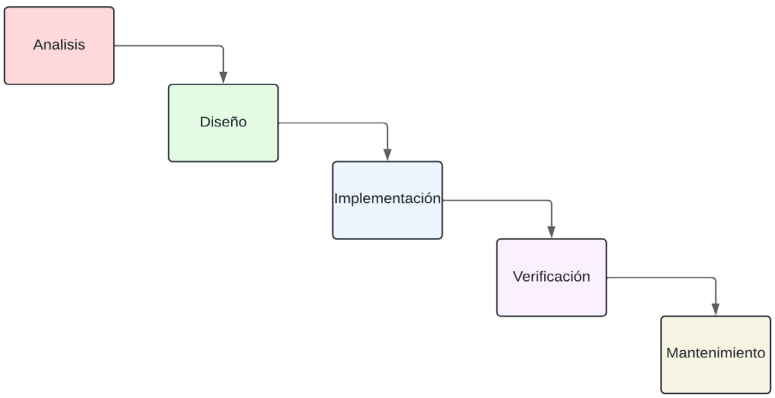



*Ilustración 7.* Fases del desarrollo en cascada

La Ilustración 7 muestra las etapas típicas de un desarrollo en cascada, aunque pueden variar en función del proyecto y su contexto.

Esta metodología destaca por su claridad y estructura, donde la planificación del proyecto se definirá antes del desarrollo, permitiendo planificar con precisión las distintas etapas y pudiendo hacer un seguimiento preciso en cada punto del desarrollo. Como resultado de esta planificación, cada fase estará completamente documentada, facilitando el desarrollo y reduciendo la complejidad para las personas encargadas de implementarlo. Estas características lo hacen adecuado para proyectos en los que se conocen bien los requisitos desde el principio y que no son susceptibles a cambios abruptos.

#### *3.3.3.1 Ventajas*

Su principal ventaja es la planificación temprana, que permite tener cada etapa del proyecto bien definida. Esto facilita su implementación, pues los encargados de implementarla pueden seguir la documentación paso a paso. También resulta muy fácil de gestionar debido a su estructura secuencial y su alta planificación.

#### *3.3.3.2 Desventajas*

Gran parte de las ventajas de la metodología vienen de su alto grado de planificación. Sin embargo, en proyectos donde pueden surgir cambios o bien los requisitos no se pueden definir al principio puede resultar inconveniente, ya que si se realiza la planificación y posteriormente se necesita hacer un cambio se tendrá que repetir de nuevo este proceso, pudiendo afectar a la estructura global y a otras etapas ya planificadas. Esto, además de resultar un proceso complejo y requerir grandes cantidades de tiempo, también supone un posible incremento en el coste del proyecto y retrasos en las entregas.

También, la planificación permite obtener una fecha de finalización concreta, pero en el caso de producirse cualquier modificación o bien una estimación incorrecta, se producirán retrasos en las entregas.

Finalmente, cualquier error o malentendido, sobre todo en las primeras fases, puede tener un impacto significativo en el éxito del proyecto, la documentación puede ser ambigua o no ser lo suficientemente detallada y conducir a errores que pueden ser detectados en las etapas finales del desarrollo, suponiendo un gran coste revertirlos.

### 3.3.4 Desarrollo iterativo

El desarrollo iterativo o incremental es un proceso de desarrollo de software creado para reforzar las debilidades del modelo en cascada. Esta metodología se caracteriza por su desarrollo a través

de ciclos repetitivos e incrementales. La idea de un desarrollo incremental permite al desarrollador aprovechar lo aprendido a lo largo del desarrollo, pudiendo aplicarse en futuras entregas.

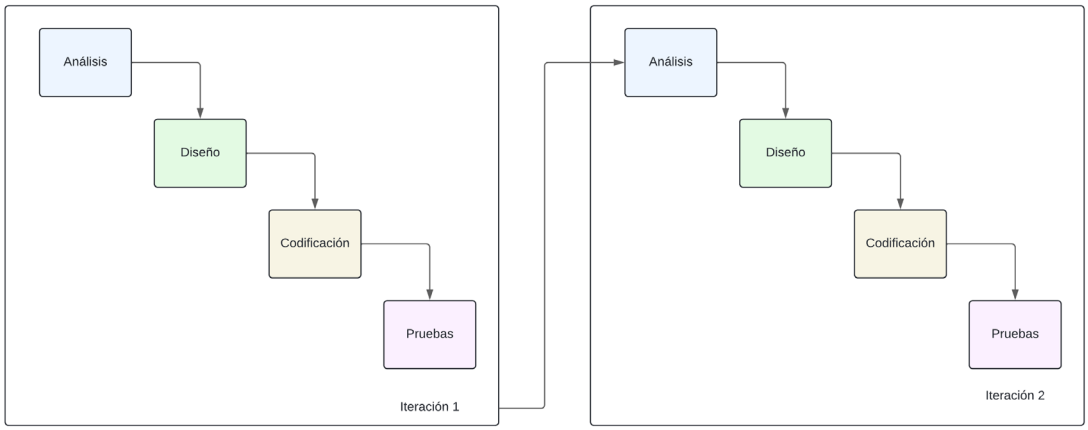



*Ilustración 8.* Fases del desarrollo iterativo

La Ilustración 8 muestra las etapas de cada ciclo y cómo la salida de un ciclo supone la base del siguiente ciclo, consiguiendo así el desarrollo incremental.

Cada ciclo o iteración supone realizar nuevamente las actividades o fases definidas. Al finalizar un ciclo, el resultado debería ser un incremento en el producto, sobre el cual se recopilará retroalimentación, tanto de usuarios como desarrolladores, para efectuar ajustes y mejoras antes del siguiente ciclo.

Este enfoque permite detectar errores y problemas en fases tempranas, dando la posibilidad de solucionarlo rápidamente, a diferencia de otras alternativas como el desarrollo en cascada. También se mejoran aspectos como la excesiva planificación, que puede ser susceptible a cambios en función de nuevos requisitos que puedan surgir, limitando la planificación a lo que se pretenda abordar en cada ciclo.

#### *3.3.4.1 Ventajas*

La nueva estructura iterativa permite efectuar ajustes en los requisitos y adaptarse con mayor facilidad que en otros modelos como el desarrollo en cascada. Además, estas iteraciones siempre incluyen una etapa de pruebas, permitiendo validar que todo funciona correctamente y detectando errores en fases tempranas.

Finalmente, la característica más relevante de esta metodología es la entrega periódica de nuevas versiones incrementales, permitiendo detectar errores en la especificación de requisitos o ambigüedades, y contribuyendo a mejorar futuras iteraciones mediante los comentarios obtenidos.

#### *3.3.4.2 Desventajas*

Uno de los mayores inconvenientes de esta metodología es su compleja gestión, donde será necesario tener una gran coordinación con el equipo para asegurar que cada iteración aporte valor al producto. Otro punto negativo es que al no tener una planificación tan detallada resulta difícil estimar los tiempos y costes del desarrollo, esto se debe a la falta de una visión completa del proyecto, que permite que existan requisitos que aún no han sido identificados, dando lugar a estimaciones demasiado optimistas y afectando a los tiempos de entrega. Por último, aunque se obtiene retroalimentación al final de cada ciclo, puede no ser suficiente al no establecer una comunicación continua con los usuarios durante la duración del ciclo, pudiendo no ajustarse a sus expectativas y no ser conscientes hasta la entrega de la versión.

## 3.4 METODOLOGÍA ELEGIDA

Para analizar las ventajas y desventajas de cada metodología se muestra a continuación, en la Tabla 1, una comparativa con las características más relevantes observadas.

*Tabla 1.* Comparativa de metodologías

| | SCRUM | XP | Cascada | Iterativo |
|---|---|---|---|---|
| **Flexibilidad** | ✓ | ✓ | ✗ | ✓ |
| **Entregas Constantes** | ✓ | ✓ | ✗ | ✓ |
| **Alta Comunicación** | ✓ | ✓ | ✗ | ✗ |
| **Respuesta rápida a cambios** | ✓ | ✓ | ✗ | ✓ |
| **Identificación temprana de problemas** | ✓ | ✓ | ✗ | ✓ |
| **Visibilidad del progreso** | ✓ | ✗ | ✗ | ✓ |
| **Excesiva planificación inicial** | ✗ | ✗ | ✓ | ✗ |
| **Gestión definida** | ✓ | ✗ | ✓ | ✗ |
| **Estimación de tiempo efectiva** | ✓* | ✗ | ✓** | ✗ |
| **Alto conocimiento técnico** | ✗ | ✓ | ✗ | ✗ |

*En SCRUM las estimaciones de tiempo serán efectivas cuando se tenga la experiencia suficiente para poder estimar adecuadamente.
** Únicamente si no surgen nuevos requisitos o modificaciones

Tras analizar las metodologías expuestas anteriormente, así como sus ventajas y desventajas en relación con el proyecto a desarrollar, se concluye que la metodología más adecuada será la metodología ágil SCRUM.

La metodología en cascada queda descartada por ser demasiado rígida con la planificación, en este proyecto los requisitos podrán variar en función del tiempo o de las necesidades que puedan surgir durante su desarrollo, provocando un grado de incertidumbre para el que la metodología en cascada no está preparada. El desarrollo iterativo se descarta debido a su falta de comunicación, aunque se obtiene retroalimentación al final de cada ciclo, no será suficiente para este proyecto donde los requisitos no están bien definidos al inicio y es necesario tener una comunicación constante con los usuarios. La metodología *extreme programming* se descarta por no poder aprovecharla completamente, tanto por el desarrollo en parejas, al ser un desarrollo individual, como por el desarrollo guiado por pruebas, que resulta inasumible en un proyecto heredado. A esto hay que sumar que requiere un alto conocimiento técnico en el área y tecnologías del desarrollo para la correcta toma de decisiones.

Esto deja únicamente la metodología ágil SCRUM, la cual se adapta adecuadamente al contexto del proyecto, donde será necesaria una alta comunicación con los usuarios y una gran flexibilidad para adaptarse rápidamente a los requisitos que puedan surgir durante el desarrollo. Si bien es una metodología pensada para equipos de varias personas, resulta fácil de adaptar a un desarrollo individual, sin perder sus principios fundamentales.

Una vez elegida la metodología a emplear, se procederá a su implantación y adaptación al proyecto.

Respecto a los roles principales, se adoptará el siguiente plan:

**Product Owner:** En este caso será el tutor del trabajo de fin de grado, tanto por su amplio conocimiento sobre el proyecto en cuestión y sobre iteraciones anteriores de su desarrollo, como por la estrecha relación con el profesorado que utiliza el asistente actualmente, la cual aporta una perspectiva valiosa sobre los requisitos y expectativas del proyecto. Por tanto, será consultado para aportar una perspectiva que garantice la satisfacción de los usuarios, en lo que respecta a decisiones funcionales que puedan afectar a su experiencia de usuario.

**SCRUM Master:** Dado que el proyecto se desarrolla de manera individual, resulta imposible contar con una persona encargada únicamente de este rol. Sin embargo, el desarrollador tomará esta responsabilidad, al asumir que está familiarizado con las técnicas y principios propios de la metodología SCRUM.

**Desarrollador:** Este rol lo asumirá el alumno autor de este Trabajo de Fin de Grado, siendo su único integrante.

## 3.5 PLAN DE TRABAJO

El desarrollo de este proyecto tendrá una duración estimada de tres meses, dividido en tres *sprints* de un mes de duración cada uno. La duración en horas de un Trabajo de Fin de Grado está estimada en 300 horas, por lo que para cada *sprint* se dedicarán 100 horas de trabajo, de las cuales se dedicarán 60 horas al desarrollo del proyecto y 40 horas al desarrollo de la memoria.

Teniendo en cuenta la cantidad de horas de trabajo disponibles para dedicar al desarrollo del proyecto, se realizará una selección de unidades de trabajo, definidas en el *backlog*, que se desarrollarán durante el *sprint* y cuya estimación global de horas de trabajo deberá estar cercana a la capacidad de trabajo establecida para cada *sprint*.

Una de las características de la metodología SCRUM es su ausencia de planificación a largo plazo con el fin adaptarse rápidamente a posibles cambios, por lo que no es posible tener una planificación completa y detallada en una fase inicial.

Para la generación del *backlog* se llevará a cabo una reunión inicial con el *Product Owner* con el fin de definir las unidades de trabajo necesarias, así como ordenarlas en función de su prioridad o importancia para la primera entrega. Por parte del desarrollador y posterior a la definición del *backlog* se realizarán las estimaciones correspondientes a cada unidad de trabajo definida, así como su refinamiento para que puedan ser entendidas con mayor claridad. Para la gestión del proyecto se utilizará la herramienta Trello[12], una herramienta de gestión de proyectos que permite de forma sencilla y visual crear tableros, listas y tarjetas con las que pueden ser representadas fácilmente etapas del flujo de trabajo y unidades de trabajo.

En la Ilustración 9 se puede apreciar la estructura que se utilizará para la gestión del proyecto.

[12] https://trello.com/es

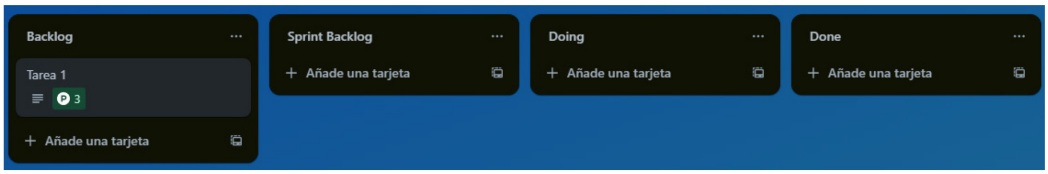



*Ilustración 9.* Estructura para gestión del proyecto en Trello

Para representar cada unidad de trabajo se utilizarán las tarjetas de la Ilustración 10, las cuales deberán tener siempre un título corto e intuitivo y una estimación de horas de trabajo. Opcionalmente, se podrá añadir una descripción para aquellas unidades de trabajo que puedan resultar más complejas de entender.

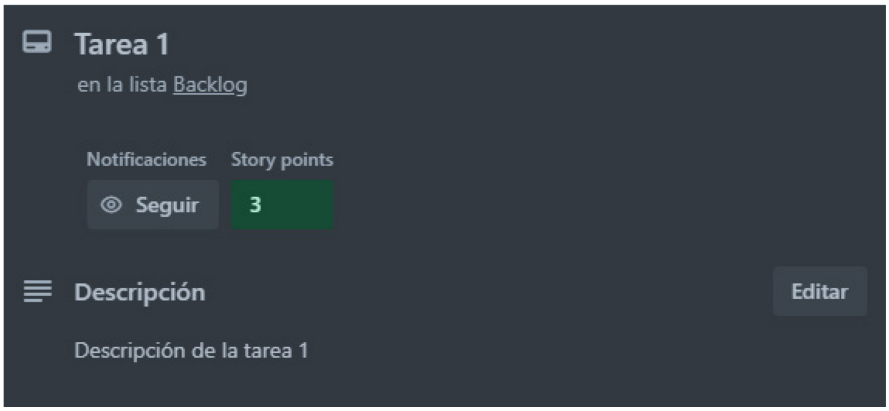



*Ilustración 10.* Ejemplo de tarea en Trello

Para la separación en etapas del flujo de trabajo se usarán las listas de tarjetas, siendo la lista *backlog* el backlog del proyecto, la lista *sprint backlog* las unidades de trabajo a desarrollar en el *sprint* actual, la lista *doing* las unidades de trabajo en desarrollo y la lista *done* las unidades de trabajo ya finalizadas.

Al inicio del primer *sprint* se elegirán todas las tareas del principio del *backlog*, ordenado por prioridad, cuya suma de estimaciones de tiempo esté cercana a la capacidad en horas dedicada al desarrollo del proyecto para cada *sprint*. Durante la duración del sprint se desarrollarán las tareas elegidas y en caso de prever que se pueda finalizar antes de tiempo cualquier unidad de trabajo será posible realizar una nueva estimación de tiempo, pudiendo, en caso de tratarse de una cantidad de tiempo considerable, añadir más unidades de trabajo al *sprint* hasta alcanzar nuevamente la estimación inicial de horas.

Tras la finalización de cada *sprint*, tendrá lugar una reunión con el *Product Owner* para realizar la entrega de la nueva versión del producto, haciendo una pequeña demostración de las novedades y anotando impresiones y cosas a mejorar que puedan surgir como retroalimentación y que podrán ser añadidas como nuevas unidades de trabajo. Además, se revisará el orden de prioridades del backlog, priorizando aquellos aspectos que puedan ser más necesarios para la próxima entrega.

Para gestionar de manera eficiente el código fuente del proyecto desarrollado, se empleará la herramienta GitHub[13], la cual se empleará como control de versiones y repositorio contenedor del proyecto. Esto garantizará un registro detallado de todos los cambios realizados y protegerá el código frente a posibles perdidas de datos. Además, facilitará su posterior distribución cuando sea requerido para su despliegue, siendo posible añadir como colaboradores a aquellas personas que lo soliciten.

[13] https://github.com/

# 4 REQUISITOS DEL SISTEMA

Para la definición de los requisitos del sistema se utilizará la familia de normas ISO/IEC 25000[14] como base. Este estándar ha sido desarrollado por ISO [15] *(International Organization for Standardization*), organización internacional no gubernamental e independiente que desarrolla y publica estándares internacionales, fundada en 1947 y con sede en Ginebra, Suiza. Dicha organización, en colaboración con organizaciones de normalización nacionales de varios países, se encarga de ofrecer guías o procedimientos para elaborar soluciones a problemas del mundo real.

La ISO/IEC 25000, también conocida como *SQuaRE* (*System and Software Quality Requirements and Evaluation*) es una familia de normas internacionales basadas en los estándares ISO/IEC 9126 e ISO/IEC 14598, cuyo objetivo es facilitar el proceso de especificación de requisitos y evaluación de características de calidad mediante el uso de un marco común que permita clasificar las características del *software* en varias categorías predefinidas.

La ISO/IEC 25010[16] define nueve características de calidad del producto, las cuales se dividen en subcaracterísticas genéricas que se pueden adaptar y especializar a casi cualquier tipo de desarrollo *software*. El Anexo 3 muestra las características y subcaracterísticas definidas en la ISO/IEC 25010.

La utilización de este estándar no implica que se deban abordar todas las características definidas, sino que se deberá realizar un análisis previo para definir qué características se deben priorizar en función de las necesidades del proyecto. Por ejemplo, para un producto *software* destinado a un sector financiero serán prioritarias características como fiabilidad o seguridad y no tanto otras como capacidad de interacción, pues se puede preferir una sólida integridad de datos financieros antes que una experiencia de usuario fluida y amigable. Por tanto, una mayor cantidad de características abordadas no implica que se vayan a detallar mejores requisitos o que el producto vaya a tener más calidad, al contrario, puede dar como resultado una cantidad de requisitos imposible de abordar o cumplir.

La aplicación de este estándar en el proyecto se justifica en el hecho de ser un marco estandarizado y con gran relevancia, lo que garantiza que los requisitos definidos sean, como mínimo, coherentes y completos. Por otra parte, al tratarse de un estándar internacional, facilita su entendimiento a otras personas que puedan estar interesadas en conocer el proyecto y asegura la consistencia de los requisitos durante todo el desarrollo, reduciendo errores y omisiones de requisitos.

Debido a la naturaleza de este proyecto, se han seleccionado las siguientes características sobre las cuales posteriormente se definirán los requisitos del sistema:

**Compatibilidad**

La propia naturaleza del proyecto, realizado en varias iteraciones de diferentes trabajos a lo largo del tiempo, supone que el resultado final incluya tanto diferentes tecnologías como módulos independientes, por lo que la compatibilidad resulta un aspecto fundamental para que todos estos componentes sean capaces de coexistir en un mismo entorno y comunicarse adecuadamente.

[14] https://iso25000.com/
[15] https://www.iso.org/home.html
[16] https://iso25000.com/index.php/normas-iso-25000/iso-25010

**Seguridad**

Cuando se trabaja con información confidencial, como es el caso de los datos personales de alumnos, es necesario custodiarlos de manera adecuada para prevenir que terceras personas puedan tener acceso a ellos, o bien para defenderse de posibles ataques malintencionados. Por todo esto, la seguridad se convierte en una característica imprescindible en el contexto de desarrollo de este proyecto.

**Adecuación funcional**

La adecuación funcional se define como la capacidad del producto para proporcionar funciones que satisfagan las necesidades declaradas e implícitas de los usuarios cuando se usan en unas condiciones específicas. En el caso del asistente web es importante proveer al usuario de todas las funciones necesarias, como añadir, modificar o eliminar alumnos, para poder realizar la clasificación. En caso de no poder realizar alguna acción se podría bloquear su finalización e impedir al usuario que pueda trabajar con normalidad.

**Capacidad de interacción**

La capacidad de interacción se define como la capacidad del producto para que un usuario intercambie información mediante su interfaz para completar una determinada tarea.

Especialmente en el uso de herramientas como un asistente de clasificación web es muy importante que el usuario pueda interactuar correctamente con la interfaz, pues esto supondrá una mayor autonomía para el usuario y una reducción del tiempo empleado para realizar la misma tarea. Por muy bien que funcione una aplicación internamente, si resulta confusa o ineficaz cuando es utilizada por un usuario, este podría decidir no utilizarla y buscar otras alternativas.

### 4.1.1 Requisitos funcionales

Los requisitos funcionales definen las capacidades y comportamientos específicos que debe tener el sistema para satisfacer las necesidades de los usuarios y cumplir con los objetivos marcados. En esta sección se detallan las funciones que el sistema debe ofrecer para asegurar que se realiza correctamente cada tarea en función de la característica del estándar al que pertenezca.

**Compatibilidad**

*Tabla 2.* RF-1 Coexistencia entre modelos de trabajo

| | |
|---|---|
| Identificador | RF-1 |
| Nombre | Coexistencia entre modelos de trabajo |
| Descripción | El usuario podrá usar tanto el modelo manual como el modelo automatizado |
| Prioridad | Alta |
| Criterio de aceptación | El usuario podrá tener exámenes con el modelo manual y con el modelo automatizado |
| Subcaracterística | Coexistencia |

### Seguridad

*Tabla 3.* RF-2 Confidencialidad en exámenes

| | |
|---|---|
| Identificador | RF-2 |
| Nombre | Confidencialidad en exámenes |
| Descripción | Los exámenes y sus respectivas clasificaciones podrán ser consultadas únicamente por el usuario creador |
| Prioridad | Alta |
| Criterio de aceptación | Solo el usuario creador podrá acceder al examen |
| Subcaracterística | Confidencialidad |

*Tabla 4.* RF-3 Integridad de datos

| | |
|---|---|
| Identificador | RF-3 |
| Nombre | Integridad de datos |
| Descripción | El sistema garantizará que sus datos están protegidos frente a modificaciones o eliminaciones no autorizadas. El sistema únicamente autorizará modificaciones o eliminaciones de datos al usuario creador |
| Prioridad | Alta |
| Criterio de aceptación | Solo el usuario podrá modificar o eliminar datos de la aplicación |
| Subcaracterística | Integridad |

### Adecuación funcional

*Tabla 5 RF-4 Crear examen automatizado*

| | |
|---|---|
| Identificador | RF-4 |
| Nombre | Crear examen automatizado |
| Descripción | El usuario podrá crear un examen con el modelo de clasificación automatizada |
| Prioridad | Alta |
| Criterio de aceptación | Se crea un examen automatizado correctamente |
| Subcaracterística | Completitud funcional |

*Tabla 6 RF-5 Descargar examen automatizado*

| | |
|---|---|
| Identificador | RF-5 |
| Nombre | Descargar examen automatizado |
| Descripción | El usuario podrá descargar el examen automatizado, junto con todos los archivos que conlleve la descarga |
| Prioridad | Alta |
| Criterio de aceptación | El usuario puede descargar el examen automatizado |
| Subcaracterística | Completitud funcional |

*Tabla 7 RF-6 Validar asociación alumno*

| | |
|---|---|
| Identificador | RF-6 |
| Nombre | Validar asociación alumno |
| Descripción | Las asociaciones automatizadas por defecto están marcadas como no verificadas, siendo el usuario el que debe verificarlas como señal de que los datos son correctos |
| Prioridad | Alta |
| Criterio de aceptación | El usuario puede validar una asociación de un alumno |
| Subcaracterística | Completitud funcional |

*Tabla 8 RF-7 Acceder al examen automatizado*

| | |
|---|---|
| Identificador | RF-7 |
| Nombre | Acceder al examen automatizado |
| Descripción | El usuario creador del examen automatizado podrá acceder al examen automatizado. En caso de no haberse realizado aún, el procedimiento de reconocimiento automatizado se iniciará, mostrando una pantalla de espera. En caso de haberse realizado, se mostrará la pantalla principal con las asociaciones. Al obtener los resultados de la asociación automatizada, todas estarán marcadas como no verificadas, pues deberán ser validadas por el usuario |
| Prioridad | Alta |
| Criterio de aceptación | El usuario puede acceder al examen automatizado |
| Subcaracterística | Completitud funcional |

*Tabla 9 RF-8 Crear asociación alumno*

| | |
|---|---|
| Identificador | RF-8 |
| Nombre | Crear asociación alumno |
| Descripción | El usuario podrá crear una nueva asociación de un alumno a un examen, para lo cual tendrá que introducir los datos de identificación, páginas y calificación |
| Prioridad | Alta |
| Criterio de aceptación | El usuario podrá crear una asociación a un alumno |
| Subcaracterística | Completitud funcional |

*Tabla 10 RF-9 Editar asociación alumno*

| | |
|---|---|
| Identificador | RF-9 |
| Nombre | Editar asociación alumno. |
| Descripción | El usuario podrá editar un alumno asociado siempre que lo considere, pudiendo modificar cualquiera de los campos de la asociación |
| Prioridad | Alta |
| Criterio de aceptación | El usuario podrá editar la asociación de un alumno |
| Subcaracterística | Completitud funcional |

*Tabla 11 RF-10 Seleccionar asociación alumno*

| | |
|---|---|
| Identificador | RF-10 |
| Nombre | Seleccionar asociación alumno |
| Descripción | El usuario podrá seleccionar cualquier asociación de un alumno a un examen de la lista, mostrando los datos del alumno en los campos de datos |
| Prioridad | Alta |
| Criterio de aceptación | El usuario podrá seleccionar cualquier asociación a un alumno |
| Subcaracterística | Completitud funcional |

*Tabla 12 RF-11 Guardar asociación alumno*

| | |
|---|---|
| Identificador | RF-11 |
| Nombre | Guardar asociación alumno |
| Descripción | Habiendo editado alguno de los campos de la asociación de un alumno, el usuario podrá guardar la asociación con los nuevos datos, sustituyendo los valores actuales por los nuevos en los datos de la aplicación |
| Prioridad | Alta |
| Criterio de aceptación | El usuario podrá guardar una asociación de alumno en caso de haber editado alguno de sus campos |
| Subcaracterística | Completitud funcional |

*Tabla 13 RF-12 Ordenar lista de alumnos asociados*

| | |
|---|---|
| Identificador | RF-12 |
| Nombre | Ordenar lista de alumnos asociados |
| Descripción | El usuario podrá ordenar la clasificación de alumnos utilizando las propiedades del nuevo modelo, por ejemplo, según si los alumnos han sido verificados o no |
| Prioridad | Alta |
| Criterio de aceptación | La lista de alumnos clasificados será ordenable utilizando los nuevos campos del alumno asociado |
| Subcaracterística | Completitud funcional |

*Tabla 14 RF-13 Máxima información en asociaciones*

| | |
|---|---|
| Identificador | RF-13 |
| Nombre | Máxima información en asociaciones |
| Descripción | El sistema garantizará que el usuario reciba la mayor cantidad posible de información disponible, aunque no sea posible identificar algún dato como el número de páginas, nombre o nota, dejando una indicación clara en aquellos campos que no se hayan podido identificar |
| Prioridad | Alta |
| Criterio de aceptación | El usuario siempre recibirá información de una clasificación, aunque no se pueda identificar algún dato o campo concreto |
| Subcaracterística | Completitud funcional |

**Capacidad de interacción**

*Tabla 15 RF-14 Clasificación automatizada*

| | |
|---|---|
| Identificador | RF-14 |
| Nombre | Clasificación automatizada |
| Descripción | El sistema devolverá una propuesta de clasificación de forma automática |
| Prioridad | Alta |
| Criterio de aceptación | El usuario recibe una clasificación automatizada al acceder al examen |
| Subcaracterística | Asistencia al usuario |

*Tabla 16 RF-15 Detección de errores*

| | |
|---|---|
| Identificador | RF-15 |
| Nombre | Detección de errores |
| Descripción | El usuario recibirá una advertencia visual en caso de que se detecte un posible fallo como una posible mala clasificación, algún alumno repetido, textos imposibles de reconocer… |
| Prioridad | Media/Alta |
| Criterio de aceptación | El usuario recibe una advertencia en caso de detectar un posible error |
| Subcaracterística | Asistencia al usuario |

*Tabla 17 RF-16 Interfaz intuitiva*

| | |
|---|---|
| Identificador | RF-16 |
| Nombre | Interfaz intuitiva |
| Descripción | La interfaz debe ser intuitiva y fácil de aprender, permitiendo a los usuarios familiarizarse rápidamente |
| Prioridad | Media |
| Criterio de aceptación | Los usuarios, en el primer uso, necesitarán menos de 10 minutos para conocer todas las acciones que se pueden realizar en el asistente |
| Subcaracterística | Aprendizabilidad |

### 4.1.2 Requisitos no funcionales

**Compatibilidad**

*Tabla 18 RNF-1 Mantener el modelo manual inalterable*

| | |
|---|---|
| Identificador | RNF-1 |
| Nombre | Mantener el modelo manual inalterable |
| Descripción | La aplicación debe mantener el modelo manual con sus funcionalidades intactas e idénticas, sin afectar a sus características, hasta el final del desarrollo |
| Prioridad | Alta |
| Criterio de aceptación | Se podrá realizar una clasificación manual con normalidad |
| Subcaracterística | Coexistencia |

*Tabla 19 RNF-2 Interoperabilidad de datos*

| | |
|---|---|
| Identificador | RNF-2 |
| Nombre | Interoperabilidad de datos |
| Descripción | El paso de mensajes y los resultados ofrecidos al usuario se realizarán empleando formatos estandarizados, lo que permitirá a los diferentes sistemas intercambiar información entre sí de manera efectiva, incluyendo la comunicación con la plataforma externa de calificaciones de la universidad, conocida como Padrino |
| Prioridad | Alta |
| Criterio de aceptación | El paso de mensajes y el envío del resultado al usuario se realizarán utilizando formatos estandarizados como PDF, CSV, JSON… |
| Subcaracterística | Interoperabilidad |

### Capacidad de interacción

*Tabla 20 RNF-3 Facilidad de operación*

| | |
|---|---|
| Identificador | RNF-3 |
| Nombre | Facilidad de operación |
| Descripción | El sistema automatizado proporcionará una clasificación de manera eficiente, reduciendo el tiempo necesario para completar el proceso de clasificación, mejorando la operabilidad global, ya que el trabajo del usuario se verá reducido considerablemente |
| Prioridad | Alta |
| Criterio de aceptación | El tiempo total dedicado a realizar una clasificación será menor al del modelo manual |
| Subcaracterística | Operabilidad |

*Tabla 21 RNF-4 Soporte a atajos de teclado*

| | |
|---|---|
| Identificador | RNF-4 |
| Nombre | Soporte a atajos de teclado |
| Descripción | El sistema permitirá que todas las acciones que puedan realizarse en la interfaz sean accesibles mediante atajos de teclado |
| Prioridad | Alta |
| Criterio de aceptación | Se podrá clasificar un examen automatizado utilizando únicamente atajos de teclado |
| Subcaracterística | Operabilidad |

### Seguridad

*Tabla 22 RNF-5 Confidencialidad en información sensible*

| | |
|---|---|
| Identificador | RNF-5 |
| Nombre | Confidencialidad en información sensible |
| Descripción | Los datos de profesores y alumnos deben permanecer siempre dentro del entorno universitario, garantizando su máxima protección y confidencialidad |
| Prioridad | Alta |
| Criterio de aceptación | Los datos confidenciales no saldrán del entorno universitario |
| Subcaracterística | Confidencialidad |

*Tabla 23 RNF-6 Seguridad en acceso y manipulación de datos*

| | |
|---|---|
| Identificador | RNF-6 |
| Nombre | Seguridad en acceso y manipulación de datos |
| Descripción | El sistema debe proveer una capa de acceso a la base de datos que actúe como punto único de acceso. Esta capa debe implementar medidas de protección contra posibles ataques, como ataques de inyección SQL, que puedan poner en peligro la información contenida |
| Prioridad | Alta/Media |
| Criterio de aceptación | Todas las nuevas consultas deben ser implementadas a través de esta capa de acceso, garantizando que sus entradas son sanitizadas y validadas antes de ser enviadas |
| Subcaracterística | Confidencialidad |

*Tabla 24 RNF-7 Gestión segura de credenciales de base de datos*

| | |
|---|---|
| Identificador | RNF-7 |
| Nombre | Gestión segura de credenciales de base de datos |
| Descripción | Las credenciales de acceso a la base de datos serán almacenadas de forma segura, en lugar de codificadas sobre el código fuente |
| Prioridad | Alta |
| Criterio de aceptación | Las credenciales se almacenarán de forma segura |
| Subcaracterística | Confidencialidad |

### 4.1.3 Casos de uso

El *Unified Modeling Language* [13] (UML) o lenguaje unificado de modelado es un lenguaje de modelado de sistemas software que permite visualizar, especificar, construir y documentar un sistema. Uno de los modelos que contiene es el diagrama de casos de usos UML, el cual es una representación que describe cómo los usuarios o actores interactúan con el sistema para lograr sus objetivos. Este diagrama tiene cuatro componentes, actores, casos de uso, sistema y relaciones.

**Actores**

Representa la entidad externa que interactúa con él. Estos pueden ser usuarios, otros sistemas o dispositivos externos. Se representa como la figura de una persona.

**Casos de uso**

Un caso de uso representa una funcionalidad del sistema que proporciona un resultado hacia el actor y, por tanto, un comportamiento que el sistema debe implementar. Se representa mediante un óvalo con el nombre del caso de uso dentro.

**Sistema**

El sistema delimita el conjunto de funcionalidades o casos de uso que estarán contenidos en él. Se representa como un rectángulo que contiene todos los casos de uso.

**Relaciones**

Las relaciones representan como el resto de los componentes se relacionan entre sí.

Existen cuatro tipos de relaciones, asociación, para representación de interacciones entre actor y caso de uso, inclusión, que indica que un caso de uso incluye siempre otro, extensión, que indica

que un caso de uso puede incluir el comportamiento de otro y generalización, para herencia entre actores o casos de uso.

La utilización de un diagrama de casos de uso UML aporta ventajas como una visión clara y concisa, así como una notación simple y universal. Además, al ser un estándar ampliamente utilizado, facilita la compresión por parte de los lectores.

Para la representación de los nuevos casos de uso del asistente web se utilizará un diagrama de casos de uso UML, obteniendo el resultado mostrado en la Ilustración 11.


*Ilustración 11.* Casos de uso

Estos casos de uso coincidirán con los requisitos funcionales pertenecientes a la característica de adecuación funcional, ya que representan acciones directas del usuario hacia el sistema. Este diagrama, además de mostrar con mayor claridad las acciones que podrá realizar el usuario, representa como se relaciona el usuario con el sistema y como se relacionarán las distintas acciones.

# 5 DISEÑO

## 5.1 ARQUITECTURA

En este apartado se describirá la arquitectura de la aplicación de forma general, haciendo una diferenciación por bloques o módulos del proyecto.

La Ilustración 12 representa la arquitectura general del asistente web.

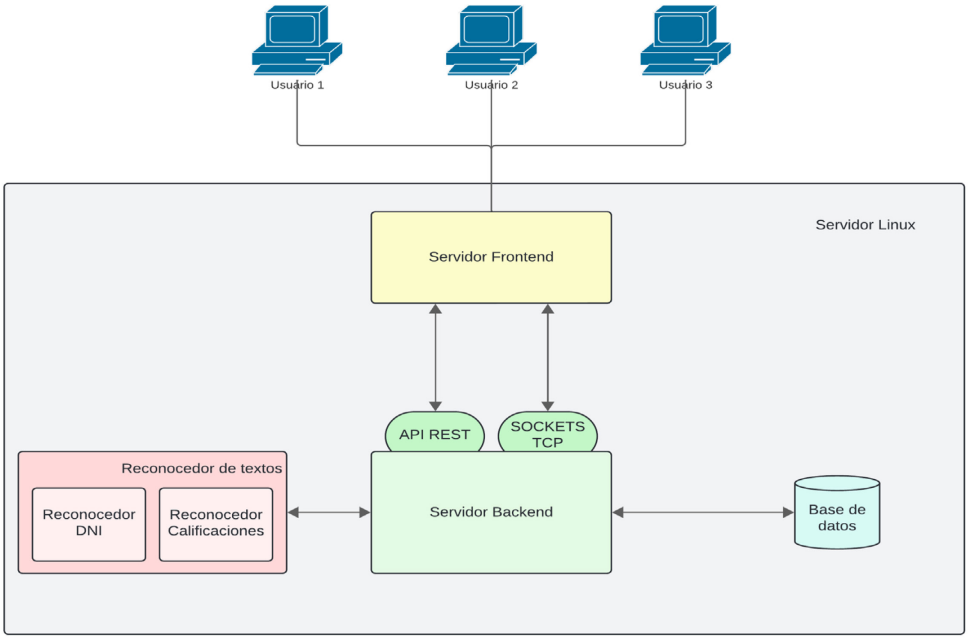



*Ilustración 12.* Arquitectura general del asistente web

Como se puede apreciar, los módulos principales serán el servidor *front-end*, el servidor *back-end*, el reconocedor de textos y la base de datos.

Cada uno de estos módulos deberá comunicarse con los otros de la manera en la que se describe en la imagen. En el caso de la comunicación entre *front-end* y *back-end* los canales de comunicación ya están implementados en la versión actual de Madrina, los cuales utilizan una API RESTful para aquellas comunicaciones que impliquen una única respuesta, y un socket TCP para aquellas comunicaciones que puedan requerir varios mensajes para completar una tarea.

La comunicación entre el servidor *back-end* y la base de datos también está desarrollada, pero es necesario renovarla y crear una nueva forma de acceder a ella, ya que actualmente se hace mediante sentencias SQL en aquellos lugares donde sea necesario. Esto no solo es poco seguro, pues es más susceptible a ataques maliciosos por inyección SQL, sino que además elimina la capacidad de reutilización en otros puntos del proyecto, donde puedan ser necesarias las mismas consultas a la base de datos, obligando a añadir el mismo código potencialmente peligroso en más lugares. Para solucionar este inconveniente se creará una *Data Access Layer* [14] (DAL) o Capa de Acceso a Datos. Una DAL es un componente de la arquitectura de software, esta se encarga de la interacción entre la aplicación y la base de datos. Su propósito principal es proporcionar una abstracción sobre la base de datos y sus accesos, encargándose de los detalles específicos de cómo se almacena o accede a los datos, eliminando esa responsabilidad del resto del proyecto. Esto implica que la forma de acceder a los datos es transparente para el resto del proyecto, únicamente teniendo acceso a aquellas funcionalidades que ofrezca esta nueva capa. Por otra parte, al estar toda la lógica de acceso a datos encapsulada en un único lugar, es más sencillo controlar y

gestionar las consultas, pudiendo establecer filtros y otras medidas de protección que permitan aumentar la seguridad del proyecto.

Finalmente, la comunicación entre *back-end* y reconocedor de textos está sin desarrollar, pues forma parte de la adaptación e integración del módulo reconocedor de textos. Por ello, primero se deberá elegir la forma de ejecutar el *script* reconocedor de textos.

Para ello existen dos opciones, la ejecución del *script* mediante una llamada al sistema operativo o la creación de un servidor dedicado exclusivamente a su ejecución. El servidor dedicado plantea una ventaja estratégica respecto a la escalabilidad de la aplicación, ya que, en caso de ser necesario, se podrían desplegar varios servidores y se podría realizar un balance de cargas para repartir las tareas entre los servidores disponibles, agilizando la respuesta en caso de haber una alta demanda de trabajo de forma simultánea. Por otra parte, esta opción aumentaría considerablemente la complejidad de la integración y puede no llegar a ser eficaz, en caso de no existir tanta demanda de trabajo, o incluso perjudicar al rendimiento, ya que habría que establecer un nuevo canal de comunicación entre servidores. En comunicación con el *Product Owner* se concluye que la cantidad de usuarios simultáneos no es tan elevada como para requerir estrategias avanzadas como el balance de cargas, debido a esto, se decide que la mejor alternativa será ejecutar el *script* como un proceso haciendo una llamada al sistema operativo.

Al ejecutar el *script* mediante una llamada al sistema operativo, los canales de comunicación serán la entrada estándar (*stdin*) y la salida estándar (*stdout*) del proceso. Como datos de entrada *stdin,* se enviarán los datos necesarios para la ejecución del *script*, estos son, los datos del examen y el listado de alumnos, recibiendo los resultados en la salida estándar *stdout.* La salida puede emitir información de forma recurrente, pudiendo enviar mensajes periódicamente con el estado de ejecución y otros datos como el tiempo estimado o el porcentaje restante y, en última instancia, la asociación de exámenes a alumnos.

## 5.2 DISEÑO DE MÓDULOS

A continuación, se expone con detalle el diseño de cada módulo del proyecto, haciendo hincapié en detalles técnicos y justificando la toma de decisiones.

### 5.2.1 Back-end

El servidor *back-end* forma parte del asistente web Madrina, por lo que ya se encuentra implementado. Este utiliza NodeJS, un entorno de ejecución de JavaScript, asíncrono, no bloqueante y con una arquitectura orientada a eventos. Esta capacidad asíncrona y no bloqueante permitirá manejar múltiples solicitudes de forma concurrente, como es el caso cuando existen canales de comunicación como una API.

La Ilustración 13 describe la arquitectura del servidor *back-end.*

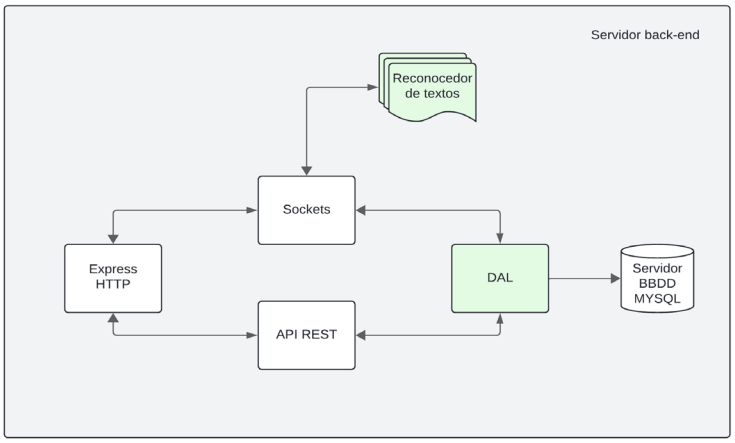



*Ilustración 13.* Diseño del módulo back-end

En la ilustración se marcan en verde aquellos componentes nuevos, siendo estos el módulo reconocedor de textos y la nueva capa de acceso a datos o *Data Access Layer* (DAL).

La DAL se implementará como un módulo con capacidad de conectarse a la base de datos y contendrá todas las funciones que permitan consultar o modificar la base de datos. Cada consulta nueva deberá ser añadida en este módulo para posteriormente poder usarse en cualquier otro punto del código. Toda consulta a la base de datos tendrá que pasar ahora por la capa de acceso a datos, teniendo que sustituir las sentencias SQL inseguras por llamadas a una función nueva que deberá ser añadida. Este proceso se realizará de forma progresiva, ya que durante el cambio se puede modificar la consulta de forma accidental y recibir respuestas inesperadas. Las consultas nuevas se implementarán directamente en DAL. Además, como parte de las mejoras de seguridad, se guardará el usuario y la contraseña de la base de datos en una variable de entorno segura, en lugar de estar contenida directamente en el código fuente. Todos estos cambios ayudarán al cumplimiento de los requisitos RF-2, RF-3, RNF-6 y RNF-7.

La ejecución del reconocedor de textos, como se comentó anteriormente, se hará mediante una llamada al sistema operativo que ejecutará el script. Se comunicarán utilizando la entrada y salida estándar, de tal forma que permita recibir respuestas de forma periódica para conocer el estado de ejecución. Dado que el *script* enviará información periódicamente con el estado o el tiempo de espera, será necesario utilizar un canal de comunicación con el *front-end* que permita el envío de múltiples respuestas, en este caso el socket TCP. Es por esto por lo que la llamada que iniciará el proceso de reconocimiento de textos se tendrá que hacer mediante una entrada en el socket que, posteriormente, enviará la información obtenida al *front-end* según le llegue del *script*.

Un aspecto importante con respecto al requisito RF-15 es la detección de errores. En el caso de las asociaciones será de gran importancia, ya que el usuario puede cometer algún error al modificar los números de páginas de inicio o fin de un examen. Para ello se necesitará analizar la lista de alumnos clasificados para poder inferir si existe algún error. Esta detección se hará sobre la lista de alumnos clasificados cada vez que se modifique o añada una nueva entrada, mediante

la búsqueda de estados inconsistentes como exámenes con los mismos números de páginas o páginas sin asociar a ningún alumno.

La Ilustración 14 muestra los posibles estados inconsistentes en una asociación.

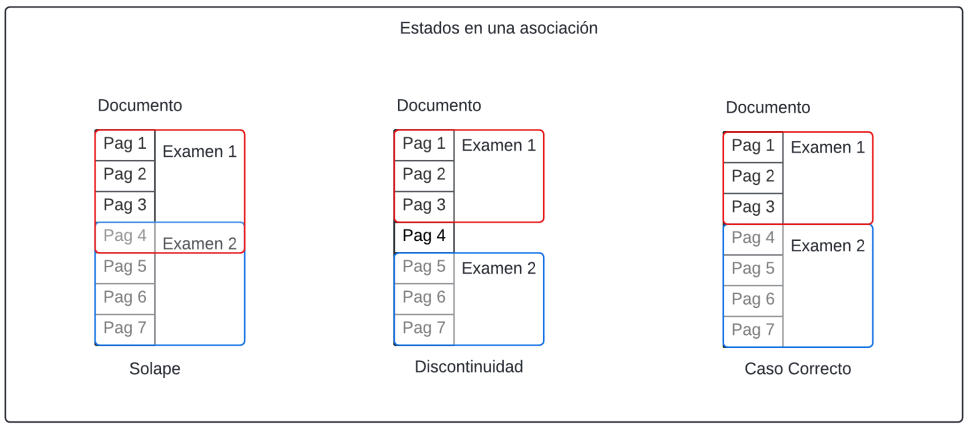



*Ilustración 14.* Estados inconsistentes en asociaciones

La naturaleza del *script* reconocedor impide que pueda emitir estados inconsistentes, pues se asume que justo después de la última página del examen de un alumno se encontrará la primera página del siguiente, resultando imposible producir solapes o discontinuidades, por lo que este tipo de inconsistencias serán producidas únicamente por acciones directas del usuario. En el supuesto de que una asociación automatizada sea correcta, pero el usuario modifique las páginas de inicio o fin de un examen podría producir un solape o discontinuidad con respecto al siguiente examen de la lista. Esta acción, al ser realizada directamente por el usuario, se supone correcta y aparecerá como validada. En ese caso, la detección de errores toma un papel de gran importancia, enviando un aviso en caso de detectar alguna inconsistencia.

Para la implementación de esta detección de errores se necesitará únicamente un mensaje que contenga los errores detectados, en caso de haberlos, por lo que puede ser implementada en un canal de comunicaciones que envíe un único mensaje, en nuestro caso la API. Para ello se creará una nueva entrada en la API que devuelva una lista con las posibles inconsistencias detectadas en los datos almacenados de la asociación, en caso de no detectar ninguna devolverá una lista vacía.

### 5.2.2 Front-end

El *front-end*, en su versión actual, se encuentra desarrollado en Angular. Este *framework* de desarrollo web destaca por su compromiso con el Modelo Vista Controlador (MVC). El MVC es un patrón de arquitectura que hace una separación en estructura de datos (modelo), interacción del usuario o interfaz (vista) y lógica de negocio (controlador).

La Ilustración 15 detalla la estructura interna del servidor *front-end,* divida según la arquitectura MVC.

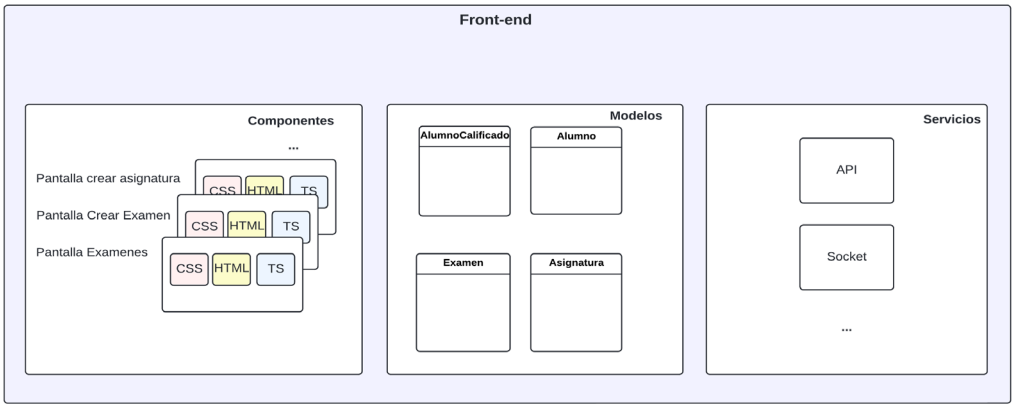



*Ilustración 15.* Diseño del módulo front-end

En el caso de Madrina, estos módulos reciben el nombre de componentes, modelos y servicios.

Los componentes son todas aquellas entidades que presentan información al usuario mediante la interfaz, estos utilizan HTML y CSS para la parte propiamente visual y TypeScript para su lógica interna. Los modelos serán todas aquellas clases o estructuras de datos que requiera la aplicación, estos organizan los datos de forma coherente y permiten definir las propiedades de una entidad específica, algunos ejemplos son las clases *alumno* o *asignatura*. Finalmente, los servicios serán todas aquellas comunicaciones con el servidor *back-end* que aportan la lógica avanzada que no puede realizarse en el *front-end*.

El primer paso será diseñar la lógica del ciclo de vida de un examen automatizado, que se representará usando un diagrama de actividades UML.

La Ilustración 16 muestra las actividades disponibles para un examen automatizado, desde su creación hasta su eliminación.

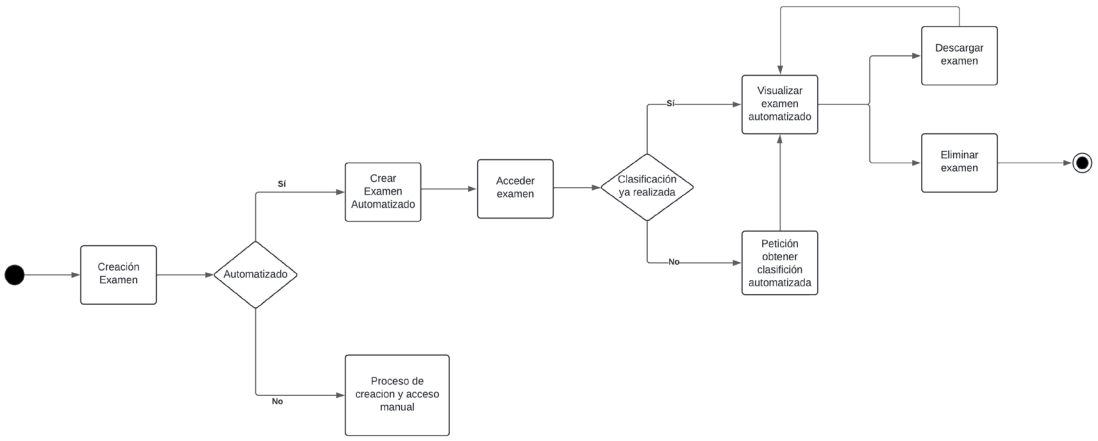



*Ilustración 16.* Ciclo de vida de un examen automatizado

Los cambios en la lógica del proyecto implican un cambio en los modelos o estructura de clases del *front-end*. Como se pretende no afectar al modelo manual y los cambios en la estructura de clases pueden de alguna manera afectar a su correcto funcionamiento, se añadirán las nuevas propiedades necesarias como extensión o herencia de las clases originales. La herencia de clases es un concepto en programación orientada a objetos (OOP) que permite que una clase derive o extienda las propiedades y comportamientos de una clase padre. Esto permite que la clase original siga funcionando con normalidad, pues su estructura no se ve afectada, al mismo tiempo que permite crear una clase nueva con las mismas propiedades que la clase de la que hereda, añadiendo

las nuevas propiedades requeridas. Las clases afectadas por esta modificación serán todas aquellas que necesiten nuevas propiedades para definir correctamente el modelo automatizado.

Ya que se quiere desarrollar un flujo de trabajo completamente separado del modelo de clasificación manual, será necesario un nuevo componente que contendrá la interfaz del nuevo modelo automatizado. Para esto se utilizará como base el componente del modelo manual, al cual se le eliminarán aquellas funcionalidades que no sean necesarias en el nuevo modelo y se reutilizarán aquellas que tengan en común, como el visor de imágenes o las opciones para añadir datos del alumno. Como la interfaz resultará familiar a la anterior, ayudará en el cumplimiento del requisito RF-16. También será necesario modificar el componente de creación de exámenes para añadir una propiedad que represente si es un examen manual o automatizado.

**Lógica de creación examen automatizado**

Para poder diferenciar entre si se trata de un examen manual o uno automatizado se definirá esta propiedad al inicio del ciclo de vida del examen. Para ello será necesario modificar el proceso de creación de un examen, común para ambos modelos.

La Ilustración 3 muestra el componente de creación de exámenes que será necesario modificar para añadir la propiedad. En este componente será necesario añadir algún elemento con el que el usuario pueda interactuar para elegir entre el modelo manual o el modelo automatizado, este elemento podrá ser un botón o un interruptor. En función de lo que el usuario haya seleccionado se creará un examen manual o un examen automatizado. Esta propiedad será inalterable durante todo el ciclo de vida del examen. Con esta nueva propiedad definida, una vez se acceda al examen se enviará al usuario al componente de la clasificación manual o automatizada en función de su elección. Esto garantizará que puedan coexistir ambos modelos en el mismo asistente, cumpliendo los requisitos RF-1 y RNF-1.

La Ilustración 17 muestra un boceto del nuevo proceso de creación de exámenes con un botón con el que interactuará el usuario para elegir entre el modelo manual o automatizado.

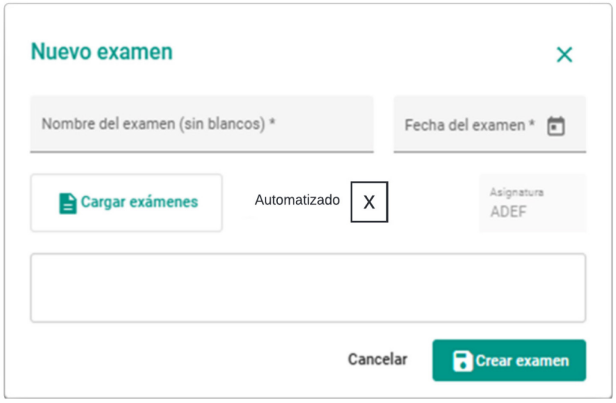



*Ilustración 17.* Boceto nuevo proceso de creación de exámenes

Respecto a la parte controladora o lógica del nuevo componente del examen automatizado, será necesario hacer desarrollos en dos lugares, la lógica del propio componente o interfaz y la lógica del módulo de servicios a través del cual se comunica con el servidor *back-end.*

**Lógica del componente examen automatizado**

Puesto que la clasificación automatizada tomará un tiempo en realizarse, será necesario mostrar una pantalla de carga con el progreso y el estado actual de la clasificación, al mismo tiempo que

se bloquean los controles para que el usuario no pueda interactuar con la pantalla y provocar algún error. Estos datos se recibirán de la conexión con el *back-end* mediante el *socket* TCP.

Una vez recibida la asociación automatizada o en caso de que ya se haya realizado anteriormente, se mostrará la interfaz del nuevo componente del examen automatizado. La respuesta de la asociación automatizada contendrá todas las asociaciones a alumnos marcadas por defecto como "no validada" implicando que es una asociación que no ha tenido una confirmación explicita de su corrección por parte del usuario. Esta propiedad se representará visualmente mostrando una marca de verificación verde en caso de haber sido validada o con una cruz roja en caso contrario.

La Ilustración 18 muestra un boceto de cómo se vería expresada la propiedad de validación.

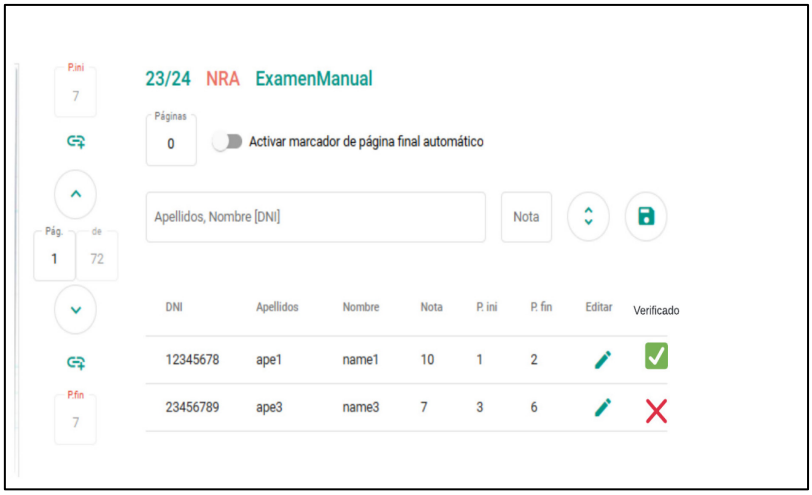



*Ilustración 18.* Boceto de la interfaz de clasificación con nuevas propiedades

Para agilizar más el proceso de clasificación automática, al validar cualquier entrada de la lista se moverá el cursor o foco al siguiente alumno de forma automática. Esto provocará que no se puedan crear nuevos alumnos calificados, pues para ello es necesario que internamente se cree un nuevo alumno calificado y se establezcan sus propiedades a sus valores por defecto. Como al verificar un alumno automáticamente se continuará con el siguiente, jamás se dará esa situación, pues los datos siempre pertenecerán a la siguiente entrada en la lista. Esto no resultará un problema, pues, al tratarse del modelo automatizado, todas las asociaciones de alumnos, o la gran mayoría, deberían estar creadas por el reconocedor, por lo que no será necesario añadir nuevas entradas en la mayoría de los casos.

Independientemente de la frecuencia con la que sea necesario añadir un nuevo alumno calificado, será necesario implementar un nuevo mecanismo para poder añadirlos en caso de ser necesario. Para ello se añadirá un nuevo botón en la interfaz que al pulsarlo creará un nuevo alumno calificado y vaciará los textos que pueda haber en los campos de datos de nombre, calificación y páginas.

La Ilustración 19 muestra un boceto con el nuevo botón de la interfaz para añadir nuevos alumnos.

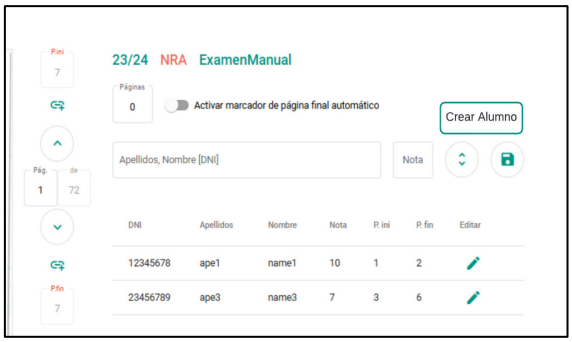



*Ilustración 19.* Boceto de nueva creación de alumnos calificados

Un aspecto fundamental del modelo de clasificación manual es su posibilidad de trabajar utilizando accesos directos del teclado. Para respetar la misma dinámica de trabajo hacia los usuarios y cumplir con el requisito RNF-4, se desarrollará un nuevo flujo de trabajo mediante atajos de teclado.

El diagrama de estados y transiciones de la Ilustración 20 definirá el comportamiento del flujo de trabajo mediante atajos de teclado, siendo los estados los puntos donde se pueda encontrar el cursor o foco y las transiciones las distintas combinaciones de teclas permitidas en dicho estado.

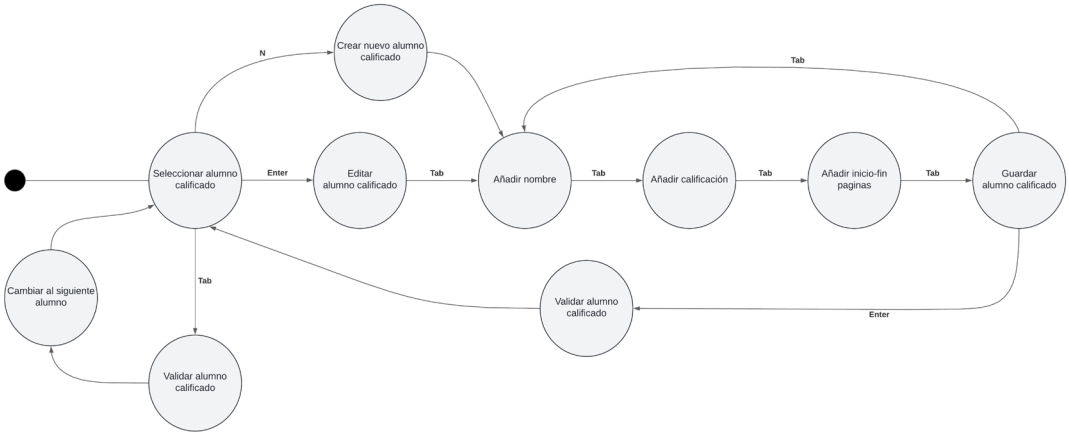



*Ilustración 20.* Diagrama de estados para atajos de teclado en la interfaz

El nuevo flujo de trabajo mediante atajos de teclado respetará el modelo utilizado en el modelo manual, por lo que compartirán los mismos atajos para aquellas acciones comunes entre ambos modelos y favoreciendo así la velocidad de aprendizaje, pues el usuario no tendrá tantas acciones que aprender. Esto permitirá el cumplimiento del requisito RF-16 sobre facilidad de aprendizaje. En este diagrama están contenidas todas las acciones necesarias para finalizar una clasificación automatizada completamente, por lo que el usuario podría llegar a finalizar la tarea tan solo usando el teclado.

Otro aspecto importante que resolver es la posibilidad de fallos en la clasificación automatizada. Dentro de estos fallos existen dos tipos, aquellos que el sistema puede identificar y aquellos que no.

Los identificables, aquellos definidos en la Ilustración 14, producidos por una acción del usuario, son fácilmente detectables por el sistema, evaluando los números de las páginas de los exámenes de la lista. Cada vez que se modifique o guarde una asociación a un alumno, se deberá hacer una consulta al servidor *back-end*, en la que se recibirá como respuesta una lista de inconsistencias detectadas. En caso de no detectarse ninguna se devolverá vacía. Esta consulta se realizará usando

como comunicación la entrada en la API definida en el punto 5.2.1. En caso de que esta consulta devuelva alguna inconsistencia, se deberá informar al usuario mediante una advertencia.

La Ilustración 21 muestra un boceto de las advertencias que recibirá el usuario al producirse alguna inconsistencia

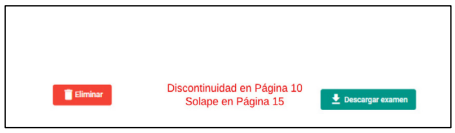



*Ilustración 21.* Boceto de advertencias de errores en interfaz de clasificación

Estas advertencias, en caso de haberlas, aparecerán al final de la lista de alumnos asociados, junto al botón de descargar, para que el usuario sea consciente de los errores detectados antes de descargar el examen.

Por otro lado, los fallos no identificables por el sistema pueden ser producidos tanto por acciones del usuario como por el módulo reconocedor, aunque este último en menor medida. Estos fallos no son detectables por el sistema, pues las páginas no contienen ni solapes ni discontinuidades, sino que es el número de páginas asociadas al examen lo que está mal. Por tanto, el fallo se encuentra en que dos asociaciones contiguas tengan un número distinto de páginas de las que deberían, pero las páginas de fin e inicio de la otra coincidan.

Como se puede ver en la Ilustración 22, el final de la primera asociación y el principio de la segunda están bien definidos, porque son consecutivos, pero el contenido de cada uno no es el del examen real.

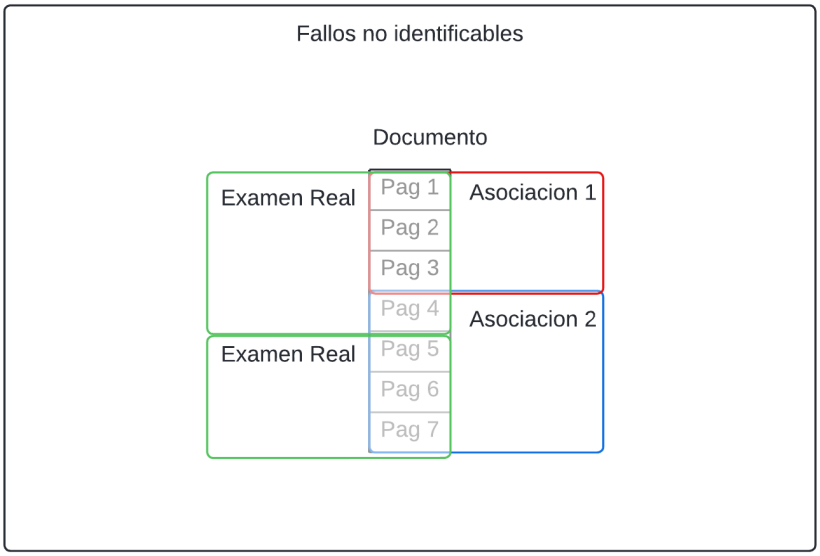



*Ilustración 22.* Fallos no identificables en asociaciones automatizadas

Para que el usuario sea capaz de identificar rápidamente estos casos, se mostrará una previsualización de la parte superior derecha del siguiente examen clasificado de la lista. En caso de darse un fallo no identificable, el usuario podrá apreciar fácilmente que la previsualización no muestra la estructura definida para la identificación de la primera página del siguiente examen, la cual deberá contener el DNI y la calificación.

En caso de que el reconocedor no detecte un inicio de examen, se asignarán dos exámenes a un mismo alumno, al asociar el final del examen actual como el inicio del examen siguiente al que no pudo ser reconocido, avanzando dos exámenes hacia delante. Para ayudar a detectar esta

situación, poco probable, pero no imposible, se mostrará junto a cada alumno asociado la cantidad de páginas que tiene asociado, permitiendo detectar con facilidad si existe algún examen con una cantidad anormal de páginas.

La Ilustración 23 muestra un boceto de la previsualización del siguiente examen y el nuevo campo que indica el número de páginas de cada asociación, añadido sobre la interfaz del modelo manual.

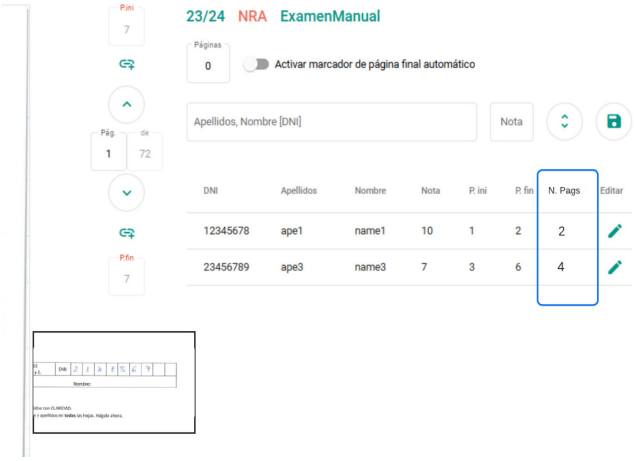



*Ilustración 23.* Boceto de previsualización del siguiente examen en Madrina

Para aprovechar el mayor espacio posible y no reducir el tamaño de la lista o la imagen principal, se añadirá la previsualización debajo de los controles de avance o retroceso de páginas, ya que era un espacio que no tenía ninguna utilidad.

Como parte de las herramientas del modelo automatizado, ahora la lista de alumnos clasificados será una lista seleccionable, sobre cuyos elementos se podrá hacer *click* y actualizará la página para que se muestren los datos asociados a la fila o alumno de la lista. Estos datos incluyen identificación, páginas de inicio y fin, imagen del examen e imagen del siguiente examen. Una vez se seleccione un elemento de la lista se mostrará con un fondo de un color distinto al resto, indicando que ese es el elemento seleccionado. Como se mostrarán los datos de esa fila en los campos de datos, el usuario podrá editarlos fácilmente, desplazándose al campo que desee modificar utilizando el ratón, para finalmente guardar los datos y así verificar dicha entrada. En caso de no hacer cambios y presionar el botón de guardar, se asume que han sido verificados y se guardarán como tal en la base de datos. En caso de presionar el botón de guardar sobre una asociación de un alumno existente, se avanzará al siguiente alumno de la lista. Esto permitirá que se puedan validar todos los alumnos únicamente presionando el botón de guardar para validar las entradas, en caso de ser correctas.

**Lógica del módulo de servicios**

También será necesario diseñar las comunicaciones que tendrá el nuevo componente automatizado con el servidor *back-end*. Para estas comunicaciones tenemos dos opciones, la API, para una respuesta, o el socket, para múltiples respuestas.

Para iniciar la clasificación automatizada será necesario enviar una señal de inicio al servidor *back-end*, que posteriormente emitirá mensajes de forma recurrente con el estado de la ejecución, hasta que finalmente devuelva los resultados. Puesto que para conocer el estado de ejecución será necesario la emisión de múltiples mensajes, la petición de iniciar la clasificación se realizará a través del socket TCP, esperando por más mensajes hasta que se reciba un mensaje de finalización.

Además, serán necesarias otras comunicaciones para recibir información del estado del examen, alumnos asociados, detección de errores, datos de alumnos o modificaciones de asociaciones. Todas estas comunicaciones, tanto de consulta como de modificación, requieren una única respuesta o incluso ninguna, por lo que serán implementadas sobre la API. Algunas de estas consultas, como recibir el estado de un examen, se encuentran implementadas para el modelo manual, pero, ya que el nuevo modelo utilizará nuevas propiedades o estructuras y se pretende mantener el modelo manual intacto, se crearán nuevas entradas para todos los casos donde se modifique la estructura de los mensajes intercambiados.

### 5.2.3 Base de datos

La base de datos del sistema está implementada usando MySQL, un sistema de gestión de bases de datos relacional. El esquema de la base de datos está implementado para el modelo de clasificación manual, por lo que será necesario modificar algunas tablas para poder expresar correctamente las nuevas propiedades del nuevo modelo automatizado.

La Ilustración 24 muestra el esquema de la base de datos representada mediante un diagrama de clases UML.

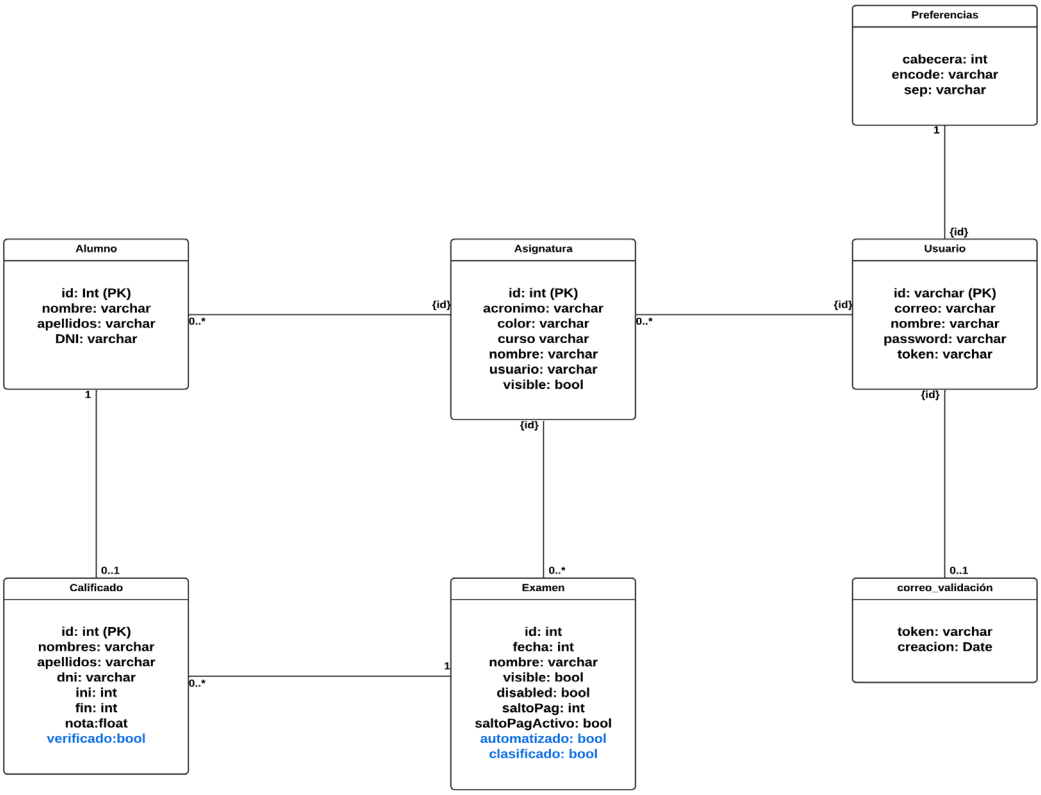



*Ilustración 24.* Diagrama de clases UML del esquema de la base de datos

En la ilustración se muestran en azul las nuevas columnas necesarias para el correcto funcionamiento del modelo automatizado. Para la clase *Examen* serán necesarias la columna *automatizado*, que representa que se trata de un examen automatizado, y la columna *clasificado*, que representa que ya se ha obtenido una clasificación automatizada por parte del módulo reconocedor. En la clase *Clasificado* se añadirá la columna *verificado*, que indica si el usuario ha confirmado explícitamente que se trata de un resultado correcto. Para que estas nuevas columnas no afecten al modelo manual se añadirán valores por defecto para que en caso de que las estructuras de datos recibidas pertenezcan al modelo manual, se inserten los valores por defecto, que expresarán que se trata de una clasificación manual.

### 5.2.4 Reconocedor de DNI

Para el reconocimiento de textos manuscritos, tanto DNI como calificaciones, se utilizará el módulo reconocedor de textos. Este contendrá dos submódulos, el reconocedor de DNI y el reconocedor de calificaciones.

El reconocedor de DNI se encuentra ya desarrollado [3] como parte de una estancia de prácticas dentro del programa Erasmus+. Este está desarrollado utilizando Python y contiene una red neuronal CNN para realizar lecturas de dígitos manuscritos. A grandes rasgos, funciona de la siguiente manera: el *script* toma como datos de entrada un PDF con exámenes, un fichero CSV con los alumnos a clasificar y un fichero CSV de salida donde se almacenan los resultados. Las imágenes del PDF se separan individualmente y se procesan para que puedan ser interpretadas por la red neuronal. Posteriormente, se buscan los límites de la estructura de casillas que contiene el DNI para recortar los dígitos individualmente casilla a casilla. Estos dígitos recortados ya pueden ser interpretados, por lo que solo queda evaluarlos con la red neuronal y obtener los resultados. Una vez obtenido el resultado, se busca al alumno en la lista y se añade el resultado al fichero CSV, considerando la página donde se ha identificado el DNI como inicio de un examen y como fin la página anterior al inicio del siguiente examen. Para realizar este proceso es necesario que se emplee la plantilla del Anexo 1.

Este *script* estaba pensado originalmente para ser utilizado en una terminal ejecutándolo junto con los argumentos de entrada, por lo que el primer paso será adaptarlo para poder ejecutarse desde el servidor *back-end*, como se definió en el punto 5.1.

Para ello, lo primero será que el *script* pueda recibir los datos necesarios para su ejecución.

El argumento del fichero PDF que contiene las imágenes de los exámenes no será necesario modificarlo, pues en el servidor *back-end* existe una copia en PDF del examen, por lo que únicamente será necesario enviar la ruta del *back-end* correspondiente a donde se encuentra el fichero PDF del examen. Originalmente, se planteó la posibilidad de utilizar las imágenes PNG del examen, que también existen en el servidor *back-end*, para eliminar el proceso de conversión de PDF a PNG en el *script*, pero estas se almacenan con una calidad inferior a la necesaria para que el *script* reconocedor pueda trabajar adecuadamente, por lo que se descartó y se optó por utilizar el documento PDF, que tiene la máxima calidad posible.

El segundo argumento, el CSV con la lista de alumnos, habrá que modificarlo, pues los datos de los alumnos no se guardan en ficheros CSV, sino en la base de datos MySQL. Para ahorrarnos los costes de creación y eliminación de ficheros CSV, se decide que la mejor opción será enviar la lista de alumnos del examen en un mensaje JSON desde el servidor *back-end*. El uso del formato JSON facilita la comunicación, ya que es un estándar ampliamente utilizado y que permite su lectura en casi cualquier lenguaje de programación, siendo incluso nativo en NodeJS. Además, garantiza el cumplimiento del requisito RNF-2. Los datos de los alumnos serán obtenidos por el servidor *back-end*, que previamente realizará una consulta a la base de datos. Estos datos se obtienen en el *back-end*, y no en el propio *script*, porque para ello sería necesario hacer una consulta a través de una conexión propia con la base de datos, esto no solo aumentaría su complejidad, sino que podría reducir el rendimiento de la base de datos al tener que mantener más conexiones a la vez.

El argumento del fichero CSV, en el que se volcaban los resultados, ya no será necesario, ya que el retorno de los resultados se hará utilizando la salida estándar *stdout*. Continuar devolviendo los resultados en un fichero CSV solo penalizaría el rendimiento, al tener que crear, leer y eliminarlo cada vez que se reciba uno. En lugar de eso, los resultados del *script* reconocedor se enviarán como un mensaje en formato JSON.

Por último, será necesario un argumento adicional que contendrá el nombre del examen, debido a que cada asignatura puede tener varios exámenes, será necesario el nombre para poder asociarlo correctamente cuando se devuelvan los resultados.

Una vez definidos los nuevos argumentos de entrada, se adaptará el *script* a los nuevos datos y al nuevo funcionamiento esperado.

El *script* ahora podrá tener varias instancias de ejecución simultáneas en función de las peticiones que se reciban en el *back-end*, por lo que es necesario adaptar cada instrucción donde se guarden o eliminen archivos, de lo contrario se guardarían todos los exámenes en el mismo sitio y podrían borrarse archivos de otros exámenes durante su ejecución. Para almacenar los datos de cada ejecución sin afectar a otras se utilizará la carpeta "*.img-dump*" en la cual se creará una carpeta nueva por cada petición de clasificación que le llegue usando, usando como nombre la estructura "*idUsuario-idAsignatura-idExamen*".

Debido a la posibilidad de múltiples instancias del *script* ejecutándose simultáneamente, es importante que tenga un rendimiento adecuado, en caso contrario podría ralentizar todo el sistema o incluso llegar a bloquearse. Además, supondría un mayor tiempo de espera para el usuario, pudiendo llegar a ser incómodo.

En la Ilustración 6 se aprecia que para un documento PDF de 72 páginas y 21 alumnos se tarda aproximadamente tres minutos y medio, teniendo todos los recursos del sistema a su disposición. Este tiempo, si bien no es óptimo, puede ser asumible, ya que, aunque tardase lo mismo que un usuario en realizar la misma tarea empleando el modelo manual, aún tendría el valor añadido de realizarse de forma automatizada, sin acciones del usuario. El problema radica en que en el entorno real al que estará destinado, no solo se tendrán que ejecutar múltiples instancias del *script* simultáneamente, sino que también tendrán que compartir los recursos del sistema con la ejecución de los otros módulos del proyecto, *front-end*, *back-end* y base de datos. Por esto, será necesario realizar optimizaciones sobre el *script* para mejorar los tiempos de respuesta y asegurar la integridad del sistema en caso de cargas elevadas de trabajo.

La Ilustración 25 muestra la carga de trabajo sobre los núcleos del procesador durante la ejecución del script reconocedor de DNI.

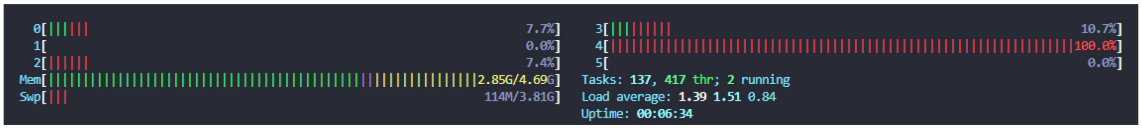



*Ilustración 25.* Utilización de CPU en ejecución del script reconocedor

Se observa claramente que toda la carga de trabajo cae sobre un único núcleo del procesador, dejando al resto infrautilizado y dejando claro que una de las mejoras de optimización más relevantes será la paralelización, que permitirá que esa carga de trabajo se distribuya entre todos los núcleos, mejorando considerablemente los tiempos de procesamiento.

Se empleará la Ley de Amdahl, que permite cuantificar la aceleración o mejora de rendimiento, para calcular el grado de mejora tras la paralelización.

$$S(N) = \frac{1}{(1-P) + \frac{P}{N}}$$

*Donde*
*S(N) es la aceleración o mejora de rendimiento*

*P es la proporción del programa que es paralelizable*

*N es el número de núcleos o cores del procesador*

En un caso ideal donde todo el código del *script* sea paralelizable se obtendría el siguiente resultado.

$$S(N) = \frac{1}{(1-1)+\frac{1}{N}} = N$$

Por tanto, podríamos llegar a obtener una mejora de hasta N veces en el rendimiento, siendo N el número de núcleos del procesador.

Para lograr una paralelización exitosa y evitar problemas como condiciones de carrera o estados inconsistentes, es crucial realizar un estudio en profundidad del flujo de ejecución del *script* para identificar qué secciones del código pueden ser paralelizadas y cuáles no.

Según [15], una sección del código se podrá paralelizar si se dan las dos siguientes condiciones. La primera, independencia de memoria o independencia de datos, esto supone que los datos utilizados por sus operaciones no deben depender entre sí, es decir, la ejecución de uno no debe afectar a otro. La segunda, independencia de control, se cumplirá si el orden de ejecución no es relevante, pudiendo ejecutarse en cualquier orden sin afectar al resultado final.

Adicionalmente, se tendrá en cuenta otro factor conocido como *overhead*. Este factor es la carga que se añade al sistema debido a la gestión y coordinación de las tareas paralelizadas, por lo que, para que la paralelización tenga un impacto positivo, el tamaño del bloque paralelizado debe ser lo suficientemente grande como para que su paralelización compense el coste del *overhead*.

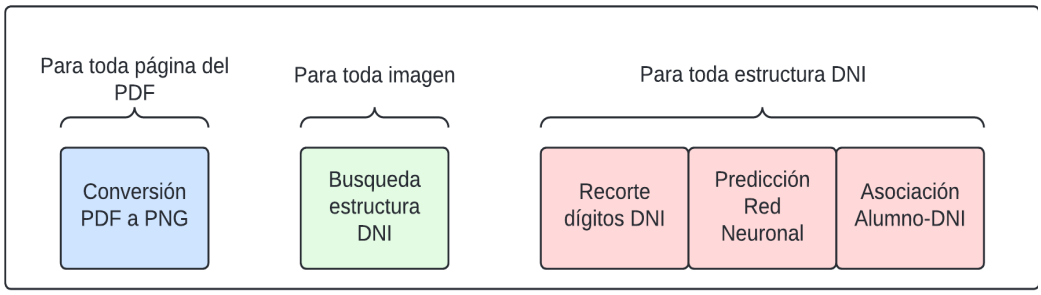



*Ilustración 26.* Separación del script en bloques de trabajo

La Ilustración 26 muestra los tres bloques principales de código que conforman el *script* reconocedor de DNI. El primer bloque, en azul, realiza una conversión a imagen PNG de cada página de un fichero PDF. El segundo bloque, en verde, busca de forma secuencial en todas las imágenes obtenidas en el bloque azul la estructura de DNI, contenida en la primera página de cada examen. El tercer bloque, en rojo, realiza el recorte, la predicción y la asociación, de forma secuencial, para cada imagen en la que previamente ha sido detectada la estructura del DNI. Como el bloque rojo tiene que ser ejecutado después del bloque verde y el bloque verde tiene que ser ejecutado después del bloque azul, existen dependencias de control, por lo que no pueden ser paralelizados en conjunto. En su lugar se podrán paralelizar de forma independiente.

La Tabla 25 muestra los tiempos que toma la ejecución de cada bloque para el caso de la Ilustración 6.

*Tabla 25.* Desglose por bloques del tiempo de ejecución del reconocedor de DNI

| | Azul | Verde | Rojo | T. Total |
|---|---|---|---|---|
| Tiempo | 25’’ (11 %) * | 3’ 14’’ (87 %) | 4’’ (2 %) | 3’ 43’’ |
| *El tiempo del bloque azul ha sido medido manualmente, ya que la salida por consola no mostraba el tiempo que tomaba | | | | |

La conversión de PDF a PNG, bloque azul, cumple con todos los requisitos definidos anteriormente, pues, no utilizan los mismos datos, ya que cada imagen distinta pertenece a una página distinta, ni existen dependencias de control, pues el orden en que se realice la conversión no afecta a las imágenes obtenidas. Además, su ejecución consume una cantidad de tiempo considerable, por lo que su mejora permitirá asumir los costes del *overhead.*

El bloque verde también cumple todos los requisitos, pues la estructura del DNI se busca en imágenes distintas y el orden en que se encuentra la estructura del DNI o primera página de cada examen no es relevante. Este bloque es el que más tiempo toma con diferencia, por lo que su paralelización permite asumir los costes del *overhead* sin ningún problema.

El bloque rojo cumple los requisitos básicos, ya que las actividades se realizan sobre imágenes distintas y el orden no es relevante, ya que el orden final de los alumnos asociados es indiferente, solo son importantes sus datos. Sin embargo, no cumple el requisito adicional del *overhead*. El tiempo que toma para realizar su actividad no es lo suficientemente significativo como para asumir los costes del *overhead*, por lo que este bloque no se paralelizará.

Por tanto, se realizará la paralelización para las actividades que están contenidas en los bloques azul y verde de la Ilustración 26.

Otro aspecto fundamental a la hora de implementar paralelización es elegir el tipo. Para el desarrollo de este proyecto se tendrán en cuenta dos tipos, multihilo y multiproceso.

La Ilustración 27 muestra a grandes rasgos, como la diferencia principal entre paralelización multiproceso y multihilo es que, en el caso de multiproceso, cada proceso tendrá su propia CPU y memoria, mientras que en multihilo estas serán compartidas por cada hilo.

| Multiproceso | | |
|---|---|---|
| CPU/Memoria | CPU/Memoria | CPU/Memoria |
| Proceso 1 | Proceso 2 | Proceso 3 |
| ... | ... | .. |

| Multihilo | | |
|---|---|---|
| CPU/Memoria | | |
| Hilo 1 | Hilo 2 | Hilo 3 |
| ... | ... | ... |

*Ilustración 27.* Multiproceso vs. Multihilo

La paralelización multihilo, por tanto, consumirá menos recursos, pues se comparten entre ellos. Además, permite una comunicación implícita entre los hilos, ya que pueden escribir sobre la misma memoria para transmitirse información. Esto, por otra parte, también puede suponer un problema, ya que si dos hilos escriben sobre la misma memoria a la vez pueden causar estados inconsistentes o condiciones de carrera, obligando a implementar otros mecanismos avanzados de protección como los semáforos.

En comparación, la paralelización multiproceso consumirá más recursos, pero es más segura de utilizar, ya que cada proceso tendrá su propia memoria y, por tanto, sus propias variables.

El factor decisivo para la elección del tipo de paralelización será el *Global Interpreter Lock* [16] (GIL) de Python. Este consiste en un mecanismo interno de control en Python que garantiza que solo un hilo tenga el control del intérprete de Python, esto es útil, pues no se tienen que implementar mecanismos de protección extra para evitar condiciones de carrera, pero, por otra parte, penaliza el rendimiento incluso pudiendo llegar a empeorarlo. Como solo un hilo puede tener el control de la ejecución a la vez, acabarán ejecutándose uno a uno, siendo equivalente a la ejecución sin paralelización, a lo que aún habría que añadirle el *overhead* que supone la gestión de la paralelización. Por todo esto, se decide que se realizará utilizando paralelización

multiproceso, que además de ser más segura, es la única que garantiza una paralelización concurrente real en este contexto.

Para la reducción de la complejidad del algoritmo será necesario el uso de herramientas de perfilado de rendimiento como *cProfile*, en caso de utilizar Python, con las que detectar aquellos fragmentos de código que más tiempo de ejecución consumen.

La Ilustración 28 muestra la salida ordenada por tiempo acumulado (*cumtime*) del perfilador de rendimiento *cProfile* para el *script* reconocedor de DNI.

```
ncalls  | tottime | percall | cumtime  | percall | filename:lineno(function)
--------------------------------------------------------------------------------------------------------------
7       | 0.001   | 0.0     | 1178.094 | 1178.094 | ['{built-in', 'method', 'builtins.exec}']
1       | 0.0     | 0.0     | 1178.093 | 1178.093 | ['<string>:1(<module>)']
1       | 122.691 | 122.691 | 1178.093 | 1178.093 | ['FirstApp3.py:159(main)']
8030633 | 233.519 | 0.0     | 580.292  | 0.0      | ['ImageFile.py:145(load)']
4015310 | 175.044 | 0.0     | 524.277  | 0.0      | ['Image.py:1759(putpixel)']
4016671 | 173.603 | 0.0     | 521.514  | 0.0      | ['Image.py:1398(getpixel)']
8032006 | 232.09  | 0.0     | 346.714  | 0.0      | ['Image.py:788(load)']
8031997 | 114.624 | 0.0     | 114.624  | 0.0      | ['{method', "'pixel_access'", 'of', "'ImagingCore'", 'objects}']
4015310 | 58.835  | 0.0     | 58.835   | 0.0      | ['{method', "'putpixel'", 'of', "'ImagingCore'", 'objects}']
4016670 | 58.015  | 0.0     | 58.015   | 0.0      | ['{method', "'getpixel'", 'of', "'ImagingCore'", 'objects}']
10347   | 0.228   | 0.0     | 8.211    | 0.041    | ['{built-in', 'method', 'builtins.len}']
1       | 0.0     | 0.0     | 8.206    | 8.206    | ['_page.py:2062(__len__)']
1       | 0.0     | 0.0     | 8.206    | 8.206    | ['_reader.py:433(_get_num_pages)']
89      | 0.02    | 0.0     | 8.205    | 8.205    | ['_reader.py:1084(_flatten)']
179     | 0.014   | 0.0     | 8.162    | 0.091    | ['_base.py:258(get_object)']
179     | 0.018   | 0.0     | 8.153    | 0.091    | ['_reader.py:1191(get_object)']
89      | 0.779   | 0.009   | 8.118    | 0.091    | ['_reader.py:1130(_get_object_from_stream)']
8499    | 0.788   | 0.0     | 4.118    | 0.0      | ['_base.py:403(read_from_stream)']
9276    | 1.493   | 0.0     | 2.747    | 0.0      | ['_utils.py:144(read_until_regex)']
1953    | 0.271   | 0.0     | 1.983    | 0.022    | ['_data_structures.py:1030(read_object)']
125     | 0.129   | 0.001   | 1.961    | 0.021    | ['_data_structures.py:329(read_from_stream)']
14113   | 0.828   | 0.0     | 1.324    | 0.0      | ['_utils.py:115(read_non_whitespace)']
41227   | 0.919   | 0.0     | 0.919    | 0.0      | ['{method', "'read'", 'of', "'_io.BytesIO'", 'objects}']
1       | 0.0     | 0.0     | 0.809    | 0.809    | ['_reader.py:1413(read)']
1       | 0.0     | 0.0     | 0.809    | 0.809    | ['_reader.py:297(__init__)']
1       | 0.0     | 0.0     | 0.806    | 0.806    | ['_reader.py:1600(_read_xref_tables_and_trailers)']
1       | 0.001   | 0.001   | 0.803    | 0.803    | ['_reader.py:1715(_read_pdf15_xref_stream)']
158     | 0.13    | 0.001   | 0.774    | 0.005    | ['_data_structures.py:128(read_from_stream)']
27005   | 0.612   | 0.0     | 0.612    | 0.0      | ['{method', "'seek'", 'of', "'_io.BytesIO'", 'objects}']
```

*Ilustración 28.* Salida del perfilador de rendimiento cProfile para el reconocedor de DNI

La mayoría de los procesos que generan esas entradas son propios del lenguaje de programación, por lo que en principio no podrían optimizarse. En cambio, se aprecia un número anormalmente elevado de llamadas y de tiempo de ejecución destinado a funciones que manejan imágenes como "*getpixel*" o "*putpixel*", por lo que para mejorar el rendimiento serán necesarias otras alternativas que demanden menos recursos, o bien realizar menos llamadas a esas funciones.

Este tipo de funciones, aunque son prácticas, pueden resultar ineficientes cuando se realizan en grandes cantidades, pues cada llamada a una función supone una nueva entrada en la pila de llamadas de la ejecución. La pila de llamadas es una estructura que gestiona las llamadas a funciones, así como la gestión de su flujo, una nueva entrada supone que esta estructura tenga que realizar todas las tareas necesarias para almacenar y manejar el contexto de la función, suponiendo un incremento en el coste temporal. En su lugar, una alternativa equivalente, es acceder a las imágenes como posiciones de una matriz. Una imagen puede ser representada como una matriz de píxeles mediante el uso de librerías como *NumPy*[17], por lo que el acceso a cada píxel podrá realizarse de forma directa accediendo a la posición de la matriz, evitando así llamadas a funciones que puedan perjudicar el rendimiento.

### 5.2.5 Reconocedor de calificaciones

El reconocedor de calificaciones se tendrá que diseñar casi desde cero, conservando únicamente la red neuronal y sus datos de entrenamiento. Este módulo se añadirá junto al reconocedor de DNI y trabajarán en conjunto para ofrecer una única respuesta que contenga toda la información.

[17] https://pypi.org/project/numpy/

Puesto que la búsqueda de la estructura del DNI ya se hace en el *script* reconocedor de DNI, no será necesario efectuar la búsqueda de la estructura de la calificación, ya que se asume que donde haya un DNI también habrá una calificación. En su lugar, se realizará directamente el recorte y la predicción de la calificación en aquellas páginas que haya detectado el reconocedor de DNI.

El Anexo 2 muestra la nueva plantilla que se empleará para poder detectar tanto el DNI como calificaciones. Esta nueva estructura para añadir la calificación cumple una característica fundamental para el diseño del nuevo reconocedor, todos los dígitos están contenidos en cuadrados, mientras que el resto son rectángulos, la coma y la propia estructura contenedora. Esto es de gran utilizad, ya que existen librerías como *OpenCV* que permiten, además de procesar eficientemente imágenes, la detección de estructuras en imágenes. Como las únicas estructuras cuadradas que hay en ese espacio son los dígitos de la calificación, bastará con hacer uso de la librería para obtener todos los cuadrados y descartar cualquier otra figura. Una vez obtenidos los dígitos individualmente, se procesarán nuevamente las imágenes para asegurar su detección y se usará la red neuronal para obtener una predicción que posteriormente será añadida a la información obtenida del reconocedor de DNI.

## 5.3 TECNOLOGÍAS Y HERRAMIENTAS

En esta sección se presentan las tecnologías y herramientas empleadas en el desarrollo del proyecto. Para cada una se proporciona una breve descripción junto con el punto de aplicación específico dentro del proyecto.

**Angular**[18] [17]

Framework de desarrollo web de código abierto basado en TypeScript, que facilita la creación de aplicaciones web dinámicas y robustas mediante la arquitectura de componentes. Este framework se empleará para desarrollar el servidor *front-end* y, por tanto, la interfaz de usuario.

**HTML (HyperText Markup Language)**

Lenguaje estándar de marcado utilizado para crear la estructura y contenido de páginas web, permitiendo la inclusión de textos, imágenes y otros elementos. En este proyecto, se utilizará para estructurar la interfaz de usuario dentro de los componentes de Angular.

**CSS (Cascading Style Sheets)**

Lenguaje de estilos utilizado para describir la presentación visual de documentos HTML, incluyendo diseño, colores, fuentes y disposición. Se empleará para aplicar estilos y mejorar la apariencia de la interfaz de usuario en los componentes de Angular.

**JavaScript**

Lenguaje de programación interpretado, ampliamente utilizado en el desarrollo web tanto para el lado del cliente como en el servidor. Se empleará en el proyecto para el desarrollo del servidor *back-end.*

**TypeScript**

Lenguaje de programación de código abierto, creado como un superconjunto o extensión de JavaScript, destinado tanto a ejecutarse del lado del cliente como del servidor. Se utilizará para implementar la lógica detrás de los componentes de Angular.

[18] https://angular.dev/

**NodeJS**[19]

Entorno de ejecución de JavaScript que permite construir aplicaciones escalables y de alto rendimiento en servidores. Se empleará para desarrollar el servidor *back-end* del proyecto, gestionando las solicitudes y respuestas del servidor

**JSON (*JavaScript Object Notation*)**

Formato de intercambio de datos ligero y fácil de leer. Se utilizará en la comunicación entre módulos del proyecto, permitiendo el intercambio de datos de forma estandarizada.

**Python**[20] [18]

Lenguaje de programación interpretado de alto nivel empleado en múltiples contextos como el desarrollo web, la ciencia de datos o la inteligencia artificial. En este proyecto se utilizará para el desarrollo de los módulos reconocedores de DNI y calificaciones.

**MySQL**[21]

Sistema de gestión de base de datos relacional, de código abierto, utilizado para almacenar, organizar y gestionar grandes volúmenes de datos de manera eficiente. Se empleará para la persistencia y gestión de los datos que sean necesarios para el funcionamiento del asistente.

**MySQL Workbench**

Herramienta visual unificada para la administración de bases de datos MySQL, que facilita la creación, modelado, administración y gestión de bases de datos. Aunque se puede modificar la base de datos utilizando sentencias SQL se empleará esta herramienta para poder gestionar la base de datos de manera visual.

**Ubuntu**[22]

Distribución de Linux basada en Debian, de código abierto, conocida por su facilidad de uso y estabilidad. Se empleará como sistema operativo para el desarrollo y ejecución del asistente.

**Oracle VirtualBox**

Software de virtualización de código abierto que permite ejecutar múltiples sistemas operativos en un solo dispositivo. Ya que no se dispone de un dispositivo con un sistema operativo Linux instalado, se empleará esta herramienta para crear una máquina virtual utilizando el sistema operativo Ubuntu para desarrollar el proyecto.

**Visual Studio Code**[23]

Editor de código fuente ligero y extensible que soporta múltiples lenguajes de programación y proporciona herramientas de depuración, control de versiones y otras funcionalidades útiles para el desarrollo de software. Se empleará para el desarrollo del código de todo el proyecto.

[19] https://nodejs.org/en
[20] https://www.python.org/
[21] https://www.mysql.com/
[22] https://ubuntu.com/download
[23] https://code.visualstudio.com/

**GitHub**[24]

Plataforma de desarrollo colaborativo que permite a desarrolladores almacenar, gestionar y controlar versiones del código fuente en repositorios. Se utilizará para almacenar y gestionar el código fuente del proyecto, así como su posterior distribución en caso de ser necesario.

[24] https://github.com/

# 6 DESARROLLO

Debido a la naturaleza de la metodología de trabajo SCRUM y a un posible cambio de requisitos en función de su prioridad para la entrega, se dividirá este apartado en tres secciones, una por cada *sprint* de trabajo. Esto ayudará no solo a que queden más claras cuáles eran las unidades de trabajo que tenían más prioridad en cada momento, sino que permitirá exponer con más claridad las conclusiones obtenidas al final de cada *sprint* en función de la retroalimentación obtenida del *Product Owner*.

## 6.1 PRIMER SPRINT

Para la realización del primer *sprint* se eligieron las unidades de trabajo en función de la prioridad definida inicialmente. Este *sprint* se centrará en la integración del modelo reconocedor de DNI en el proyecto madrina para que funcionen usando los mismos datos y se puedan hacer las primeras asociaciones de exámenes a alumnos automatizadas.

Las unidades de trabajo seleccionadas para el *sprint* 1 se detallan en el Anexo 4. En este, además de las unidades de trabajo que suponen un desarrollo en el proyecto, se incluyen las unidades de trabajo de la propia redacción de esta memoria, estas unidades de trabajo se pueden diferenciar porque comienzan con la palabra “Memoria” y no se comentarán en este apartado. Estas unidades de trabajo, asociadas a la redacción, se omitirán en el futuro por repetirse en la mayoría de los casos.

### 6.1.1 Unidades de trabajo

**Creación Página Examen Automatizado**

Esta unidad de trabajo se centra en la creación del nuevo flujo de trabajo automatizado, garantizando que se creen las propiedades necesarias para el nuevo modelo sin afectar al modelo manual. Aún no presentará cambios visuales en la interfaz, pues su fin, por ahora, es el de probar el comportamiento y las comunicaciones con el resto de los módulos del sistema.

Para el desarrollo de la nueva página para el examen automatizado en el *front-end*, se emplea una copia del modelo de examen manual, al cual se le eliminan todas aquellas funcionalidades que no sean compatibles o no sean necesarias en el nuevo modelo. Aquí se desarrollan nuevos métodos para la obtención del examen automatizado, así como su correcta gestión. También se crean los nuevos modelos o clases necesarios para el modelo automatizado, para lo cual se empleará herencia de clases respecto a las clases originales sobre las que haya que añadir las nuevas propiedades, asegurando así que no se modifica el comportamiento del modelo manual.

Finalmente, se integran las nuevas comunicaciones con el servidor *back-end*, asegurándonos de su correcta utilización, pues en algunos casos puede variar respecto al modelo manual y puede ser necesario ejecutarlas en un orden concreto.

**Visor del siguiente examen**

Esta unidad de trabajo consiste en el desarrollo de un nuevo componente visual con el cual se podrá visualizar una página elegida por el sistema en función de las acciones del usuario.

Para ello será necesario crear un nuevo componente en Angular, el cual mostrará en la interfaz la imagen enviada desde el componente principal, en este caso, la primera página del siguiente examen. Para este desarrollo se consideran dos alternativas principales, la recepción de la imagen a mostrar como un argumento de entrada a la creación del componente o bien como parámetro de una función que ofrecerá el componente. Las imágenes a mostrar tendrán el formato de una hoja

de tamaño completo, ya que se emplearán las mismas que se muestran en el visor a tamaño completo, para no aumentar la memoria empleada. Sin embargo, en este caso será necesario mostrar únicamente la parte superior derecha, donde estarán contenidos el DNI y la calificación, información suficiente para que el profesor pueda comprobar si corresponde a la primera página del examen o no. Para poder hacer el recorte de la imagen de forma efectiva, resulta más útil el enfoque de recibir la imagen mediante un método de una función, pues permitirá que se recorte nada más recibir la imagen. Por esto, se desarrolla un método dentro del componente llamado "*changeImage*" al cual se le puede enviar una imagen y se recortará de manera adecuada para mostrar la información relevante. Una vez creado el componente se añade a la nueva interfaz de acuerdo con la Ilustración 23.

**Comunicación Front-Back-Script**

Durante la duración de esta unidad de trabajo se desarrolla la comunicación entre *front-end*, *back-end* y el módulo reconocedor. Esto incluye desde la llamada que realiza el *front-end* al servidor *back-end*, hasta que el *script* devuelve la clasificación. De acuerdo con el diseño definido en el punto 5.2.1, esta comunicación se realiza utilizando el socket TCP, para permitir el envío de múltiples respuestas. Para ello se crea una nueva entrada en el socket, la cual será accesible desde el servidor *front-end*, la cual podrá iniciar la clasificación automatizada. Esta entrada recibirá como entrada los datos del usuario, la asignatura y el examen. Empleando estos datos, se obtendrá el listado de alumnos contenido en la base de datos que, posteriormente, se enviarán al módulo reconocedor.

En NodeJS existen dos formas principales de ejecutar comandos de forma nativa, las instrucciones *spawn* y *exec*. La diferencia principal entre estas alternativas es la forma de comunicación. La instrucción *spawn* establece canales de comunicación en tiempo real conocidos como *stdin* y *stdout*, mientras que la instrucción *exec* solo obtiene la respuesta al final de la ejecución de la instrucción. Para el caso del reconocedor es necesario poder enviar respuestas en tiempo real con el estado y el progreso, por lo que se emplea la instrucción *spawn*. A esta instrucción, junto a la ruta al *script*, se le envían los datos definidos en el diseño, la ruta al PDF, la lista de alumnos en formato JSON y el nombre de la asignatura.

Una vez ejecutado, se espera a obtener respuestas del *script* en la salida estándar, para lo cual se emplea un *listener*, que se activa al recibir datos desde el *script* en la salida estándar. Una vez recibidos los datos, se procesan en función del tipo de mensaje. Los mensajes de progreso se reenvían al *front-end* para actualizar el progreso en la pantalla de carga, los alumnos asociados se acumulan en una lista que se insertará en la base de datos al finalizar la ejecución y el mensaje de finalización enviará un mensaje de finalización al servidor *front-end*, indicando que puede eliminar la pantalla de carga y obtener el examen y su clasificación.

Para cada llamada al *socket*, se guarda en una lista el proceso creado para ejecutar el *script*, permitiendo conocer si ya existe un proceso en marcha para un examen y eliminarlo en caso de abandonar la ventana del examen.

**Integración del reconocedor DNI**

La integración del *script* reconocedor de DNI consiste en su modificación de tal forma que pueda realizar las mismas actividades con los nuevos datos disponibles en el sistema. Para ello, el primer paso será la correcta lectura de los valores de entrada definidos en el apartado 5.2.4, para lo cual se procesarán y guardarán en variables globales, accesibles desde cualquier punto del *script*. Con esta información además se construyen las rutas y carpetas necesarias para almacenar los datos del *script* y trabajar de manera concurrente sin afectar a otras ejecuciones del *script* de otros usuarios.

Para poder realizar la integración adecuadamente se sustituyen todas las rutas donde se guarden o eliminen archivos por las nuevas rutas definidas en función de los argumentos de entrada, en caso contrario se podrían modificar archivos de otras ejecuciones que se estén realizando al mismo tiempo.

Durante la integración y sus pruebas se encontraron dos errores a resolver, el primero, la consideración de que todos los DNI tienen el mismo número de dígitos y el segundo, que no manejaban adecuadamente las excepciones que pudieran ocurrir en tiempo de ejecución.

El DNI reconocido por la red neuronal, anteriormente, se asociaba a un alumno real comparando dígito a dígito las coincidencias con todos los alumnos que tuviera ese examen, el problema surge cuando existen alumnos con un número distinto de dígitos en el DNI, bien por presentar ceros al principio, o bien por ser estudiantes de intercambio y tener otro tipo de documento de identificación. Al llegar a algún alumno con un menor número de dígitos, el *script* intentaba acceder a una posición de memoria que no existe, provocando que la ejecución se detuviera. Para solucionar esto, la comparación con los DNI de la lista real ahora se realiza como un único bloque y no dígito a dígito, para ello se utiliza la Distancia de Levenshtein [19].

La distancia de Levenshtein o distancia entre palabras define el número mínimo de operaciones requeridas para transformar una cadena de caracteres en otra. Este número de operaciones permite calcular con qué palabra de la lista guarda más similitud, pues aquella que tenga el mínimo de operaciones será aquella que más similar sea. La Ilustración 29 muestra un ejemplo para tres posibles palabras, siendo la que menos transformaciones necesita la palabra más similar a la palabra objetivo, habiendo asociado la palabra correctamente.

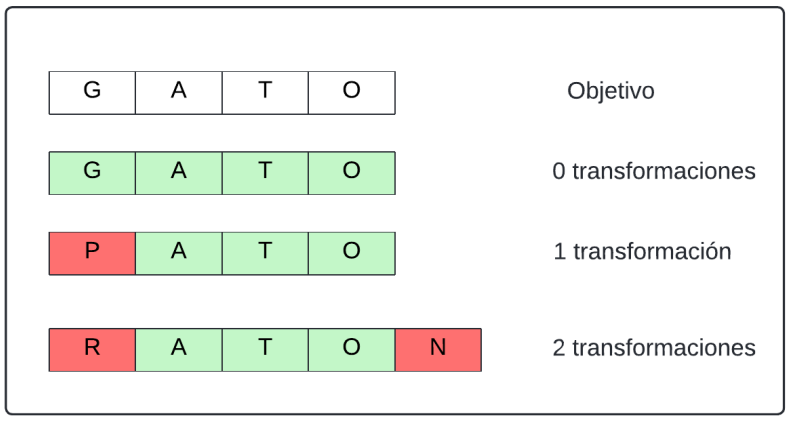



*Ilustración 29.* Distancia de Levenshtein

El otro problema detectado es la falta del manejo de excepciones durante la ejecución. En caso de que se produzca algún error crítico durante la ejecución del *script*, al no haber un correcto manejo de excepciones, la ejecución se detenía por completo, desechando todos los resultados obtenidos hasta ese momento. Para resolver este problema se envuelven los bloques donde se pueden producir errores en una estructura *try…catch*, para, en caso de producirse alguna excepción, ser capaces de capturarla y acumular toda la información que se haya detectado hasta el momento. Posteriormente, se continuará con la búsqueda del siguiente examen, en lugar de detener todo el proceso. Estos errores son producidos por el manejo de las imágenes, que pueden tener tamaños variables en función de cómo se haya escaneado la página del examen, provocando que en algunos casos se acceda a una posición de la imagen que no existe. Aunque se produzca este error es importante saber que hasta ese momento se habría detectado la primera página del examen, por lo que, en caso de error, se minimizará el impacto al únicamente no tener el valor del DNI, pero sí las páginas de inicio y fin, aportando así siempre el máximo posible de información y cumpliendo el requisito RF-13.

### Emisión de mensajes desde el *script*

Para poder enviar información desde el *script* con mayor facilidad y con un formato común en todo el código, se desarrolla una clase encargada de emitir las comunicaciones hacia el servidor *back-end*. Esta clase se encarga de escribir en la salida estándar *stdout* los tres posibles mensajes a enviar: progreso, finalización, y nuevo estudiante. Cada mensaje añadido en la salida estándar tiene una cabecera o preámbulo que los identifica, pudiendo detectarse en el *back-end* leyendo el primer fragmento del mensaje para poder procesarlo adecuadamente en función del tipo de mensaje.

La Ilustración 30 muestra los distintos mensajes que podrán ser enviados, estas funciones ahora podrán ser llamadas en cualquier punto del código en función de las necesidades del momento.

```
class Emitter:
    def emit_progress(self, progress, state):
        sys.stdout.write(f"progress {progress} {state}\n")

    def emit_done(self, time_elapsed):
        sys.stdout.write(f"done {time_elapsed}\n")

    def emit_student(self, alumno : Alumno):
        sys.stdout.write(f"new-calificado {alumno.to_json()}\n")
```

*Ilustración 30.* Clase Emitter para emisión de mensajes desde el script reconocedor

### Modificación en la base de datos

En esta unidad de trabajo se añaden las nuevas columnas en la base de datos, necesarias para el correcto funcionamiento del modelo automatizado. Estas se añaden basándose en el diseño definido en el punto 5.2.3. Para mayor comodidad se añadirán usando MySQL Workbench, que permite gestionar la base de datos de manera visual con una interfaz intuitiva, como se muestra en la Ilustración 31.

Name: calificados Schema:

| Column Name | Datatype |
|---|---|
| id | INT |
| apellidos | VARCHAR(256) |
| dni | VARCHAR(64) |
| examen | INT |
| fin | INT |
| ini | INT |
| nombre | VARCHAR(256) |
| nota | FLOAT |
| verificado | TINYINT |

*Ilustración 31.* Ejemplo de modificación en MySQL Workbench

### Integración nueva estructura base de datos

Para la correcta integración de los nuevos campos de la base de datos se desarrollan nuevas entradas en la API que permitan realizar operaciones comunes como obtener los datos de un examen u obtener alumnos, añadiendo estos campos a la consulta. Estas entradas de la API posteriormente serán utilizadas por el *front-end*, sustituyendo a sus antiguas entradas.

### 6.1.2 Finalización del primer sprint

El primer *sprint*, dedicado principalmente a la integración del *script* reconocedor de DNI, concluye habiendo completado correctamente todas las unidades de trabajo propuestas sin complicaciones mayores.

A la entrega del primer *sprint* se tiene en cuenta la retroalimentación obtenida del *Product Owner* en una demostración de la versión incremental, de lo cual se obtienen el siguiente punto a mejorar.

La orden de detener el *script* reconocedor al abandonar la ventana del modelo automatizado debe evitarse. Por una cuestión de simplicidad y ahorro de procesamiento de CPU, al abandonar la ventana automatizada, en caso de estarse ejecutando el script reconocedor, se enviaba una señal para que se detuviera y que no consumiera más recursos. Se concluye junto al *Product Owner* que es mejor que el proceso de asociación continúe en segundo plano al abandonar la pantalla, permitiendo además que el usuario pueda realizar varias asociaciones a la vez. Se creará una unidad de trabajo para corregir este punto, la cual se añadirá al siguiente *sprint*.

Se obtiene además una priorización de los requisitos para el siguiente *sprint*, el cual se centrará en completar el ciclo de creación del examen, además de desarrollar las herramientas necesarias en la interfaz. La optimización del rendimiento pasa a un segundo plano, siendo el último *sprint* donde se desarrollarán todas las optimizaciones necesarias.

Por parte del desarrollador se hace una propuesta de atajos de teclado para controlar la interfaz en el siguiente *sprint*, así como una mejora con la que se definirá el modelo del examen en el momento de su creación y no justo después, en la primera vez que se abra el examen. Ambas propuestas son validadas y aceptadas, por lo que se añadirán en el siguiente *sprint*.

En conclusión, la metodología se ha aplicado correctamente y ha permitido obtener una retroalimentación útil que será aplicada en siguientes *sprints*, así como la detección de errores de manera temprana, facilitando su corrección. También se han finalizado todas las unidades de trabajo planificadas, poniendo de manifiesto la eficacia de la estimación de horas de trabajo y ayudando a cumplir los objetivos de la entrega.

## 6.2 Segundo Sprint

Las prioridades de trabajo para el segundo *sprint* estarán sujetas a la retroalimentación obtenida del primer *sprint*, siendo estas el arreglo del error encontrado y completar el ciclo de vida del examen, aunque no tenga aún todas las funcionalidades desarrolladas. Además, por orden de prioridad en el *backlog* se desarrollará el reconocedor de calificaciones y las herramientas para la interfaz del examen automatizado.

Las unidades de trabajo para el segundo *sprint* están presentes en el Anexo 5.

### 6.2.1 Unidades de trabajo

#### Creación DAL para base de datos

En esta tarea se desarrolla la capa de acceso a datos o DAL definida en la arquitectura del proyecto y concretamente en la comunicación entre servidor *back-end* y la base de datos en el punto 5.1. Esta tarea se desarrollará antes que cualquier otra del *sprint* por tener una prioridad superior basada en dos puntos, la necesidad de incrementar la seguridad en el proyecto y la necesidad de nuevos accesos a la base de datos. Muchas de las unidades de trabajo de este *sprint* implican hacer nuevas consultas a la base de datos, por lo que, para evitar el tener que crearlas para luego sustituirlas por las nuevas consultas de la DAL, se prioriza la creación de la DAL, permitiendo, además de reutilizar las consultas, evitar un cambio a futuro que tomaría tiempo realizar.

Para desarrollar esta nueva capa se crea el fichero "*DAL.js*". En este fichero están contenidos, además de la conexión con la base de datos MySQL, todas las consultas útiles para el desarrollo del proyecto. Cada una de estas consultas se exportará como se muestra en la Ilustración 32, permitiendo que este módulo sea importado desde cualquier otra parte del proyecto y teniendo acceso a todas sus consultas.

```
module.exports = {
  getAlumnosFromAsignatura,
  insertAlumnoCalificado,
  setExamenIsAutomatizado,
  getCalificadosOrderByIni,
  getAlumnosNoCalificados,
  updateCalificadoAutomatizado
};
```

*Ilustración 32.* Consultas accesibles desde la capa DAL

Además, para mejorar la seguridad de las consultas, se realizará una limpieza a los datos de entrada antes de enviar la consulta a la base de datos. Esto incluye la comprobación de los tipos de datos de entrada, para que se adecuen a los tipos de la base de datos, y la eliminación de caracteres potencialmente peligrosos como <, >, &, ' o " mediante el uso de la librería *validator.js* de NodeJS. Con esto aumentará considerablemente la seguridad del proyecto, pues se evitarán ataques como los de inyección SQL.

En esta tarea, además, al ser necesaria una conexión con la base de datos, se modifica la forma de guardar el usuario y la contraseña para acceder a la base de datos. Para ello se usará la librería *dotenv* de NodeJS, que permite guardar las credenciales en una variable de entorno segura, para posteriormente cargarlas en el proyecto. Para realizarlo se crea el fichero "*.env"*, donde se añaden las credenciales de la base de datos. Es importante que, posterior a su creación, se modifiquen los permisos del fichero para únicamente tengan acceso aquellos usuarios autorizados. Una vez creado se podrá acceder a las variables dentro del proyecto haciendo referencia a "*process.env.VARIABLE*".

**Nuevo proceso de creación de exámenes**

Esta unidad de trabajo está destinada al desarrollo del nuevo proceso de creación de exámenes definido el punto 5.2.2 en el que, al momento de crear el examen, se definirá el modelo del examen en función de la elección del usuario. Para ello se utilizará la Ilustración 17 como referencia para el desarrollo de la interfaz.

Para que el usuario sea capaz de elegir la opción, se añade a la interfaz un componente de tipo *switch* o interruptor, al pulsar sobre el botón *crear examen* se envía, además de los datos del examen, el valor del interruptor, el cual en el servidor *back-end* se añade a la base de datos usando la capa de acceso a datos o DAL.

La Ilustración 33 muestra el resultado final del desarrollo de la nueva interfaz destinada a la creación de exámenes.

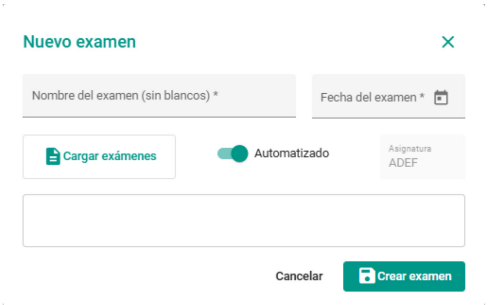



*Ilustración 33.* Nueva interfaz de creación de exámenes

### Herramientas para la clasificación automatizada

Otro punto importante para el *sprint* es la necesidad de poder completar una clasificación automatizada, para lo que serán necesarias las herramientas para poder desplazarse por los alumnos y comprobar que las asociaciones hechas por el *script* reconocedor son correctas. Estas herramientas, indispensables para una correcta adecuación funcional, garantizarán el cumplimiento de los requisitos RF-5, RF-6 y los requisitos RF del 8 al 12.

Para ello se transforma la lista de alumnos clasificados, obtenida del *script*, en una lista seleccionable, para que el usuario pueda avanzar rápidamente a aquellas entradas de la lista que considere. Al finalizar la creación de la lista de alumnos clasificados, una vez se ha recibido del servidor *back-end*, se guardarán los elementos de la lista como componentes HTML. Cuando se hace *click* sobre cualquiera de las filas de la tabla, se guardará el elemento seleccionado en una variable, para poder marcarla y diferenciarla del resto. Además, en caso de que el usuario presione sobre cualquiera de los campos de datos para editar la asociación, el foco se situará sobre él, por lo que una vez se guarde la edición de la asociación, se devolverá el foco a la fila seleccionada, previamente guardada, empleando la función *focus*. Esta función permite que el foco se posicione sobre cualquiera de los elementos de la interfaz visual, permitiendo que se siga el correcto flujo de acciones. Una vez guardados los datos de la asociación, se asume que se han introducido los datos correctos y se devuelve el control a la lista para poder seguir navegando por ella. Además, al tratarse de una acción directa del usuario, el alumno asociado se enviará con la propiedad *verificado* como cierta.

Como parte del desarrollo de las herramientas para la clasificación automatizada, también se incluirá el nuevo botón para crear nuevos alumnos o entradas en la lista, definido en la Ilustración 19. Para ello, se añade un botón encima del resto de campos de datos, que, al pulsarlo, creará un nuevo alumno calificado con todos sus datos ajustados a los valores por defecto y se limpiarán los datos contenidos en los campos de datos. Esto es equivalente a cuando en el modelo manual se añade un nuevo alumno, por lo que no será necesario realizar más cambios, al ser una característica base del asistente.

### Verificación de alumnos calificados

Para que el usuario sea capaz de conocer qué alumnos asociados han sido verificados y cuáles no, se necesita modificar la interfaz de usuario. Esta unidad de trabajo consiste en el desarrollo de las modificaciones en la interfaz del modelo automatizado para que se pueda diferenciar entre asociaciones verificadas y no verificadas. Para ello se utilizará la Ilustración 18 como referencia para desarrollar la nueva interfaz.

Para realizar el desarrollo se modifica la lista de alumnos asociados para que contenga una columna más, llamada *verificado*. Esta nueva columna expresará un valor en función de la propiedad *verificado* de cada asociación. Si es verdadero contendrá una marca de verificación verde y si es falso una cruz roja.

Cuando un usuario actualiza alguna asociación, después de guardarla, se hace una consulta a la API para obtener los alumnos asociados nuevamente, esto se hace para evitar posibles desfases entre los datos contenidos en el *front-end* y el *back-end*. Si la propiedad *verificado* se actualizase de forma independiente en el *front-end*, sin esperar a que se actualice en el *back-end*, podría darse el caso en que, por problemas de conexión o un error en la inserción, no se modificara en la base de datos, provocando que se encuentre verificado en el *front-end* pero no en el *back-end*, pudiendo causar pérdida de información.

**Nuevo flujo de atajos de teclado**

Aunque ahora ya se pueda finalizar una clasificación automatizada mediante el uso de la interfaz y sus herramientas, es importante que se pueda realizar mediante atajos de teclado para reducir al máximo el tiempo invertido en la clasificación. Para ello, se implementan los atajos de teclado necesarios para finalizar la clasificación automatizada y se conservarán aquellos que se podían utilizar en el modelo manual. Los atajos de teclado se desarrollarán utilizando como referencia la Ilustración 20, que contiene el diseño de todos los estados y transiciones disponibles en el nuevo modelo automatizado.

En el modelo manual existían atajos de teclado para los campos como nombre, páginas o calificación, pero para el modelo automatizado no serán suficientes. Estos atajos de teclado se conservarán para que los usuarios puedan adaptarse rápidamente al nuevo flujo de atajos.

Dado que además de la lista, existen otros elementos seleccionables en la interfaz y que se emplearán las mismas teclas para realizar acciones similares, será necesario interceptar todas las teclas pulsadas sobre la interfaz para poder decidir qué acción realizar. Si el último elemento seleccionado es un elemento de la lista de alumnos clasificados, se iterará sobre la lista. En caso contrario, si el último elemento seleccionado es uno de los campos para añadir información, se conservará el ciclo de atajos del modelo manual. Para conocer qué acción realizar, cada vez que se interactúe con algún componente de la interfaz se guardará y se tendrá en cuenta para realizar la acción correspondiente.

Para el caso de la tecla *Tab*, si el último elemento es una fila de la lista, se avanza y selecciona el siguiente alumno asociado. Al asumir que se han revisado los datos del alumno anterior, se marcarán como validados, pudiendo validar las asociaciones pulsando una única tecla. Por otra parte, si el último elemento seleccionado es cualquiera de los campos para añadir o editar información, se realizará el flujo del modelo manual, iterando sobre los campos de datos.

Para la tecla *Enter*, en caso de que el elemento seleccionado sea una fila de la lista, se permite editar la información de la asociación y editar los campos de datos. En caso de que el elemento seleccionado sea el botón de guardar, se guarda la información en la base de datos y devuelve el control a la última fila seleccionada de la lista.

Para la tecla *N*, y su único caso, la creación de un alumno nuevo, se asume que es equivalente a la edición de un alumno calificado, pero creando un alumno calificado nuevo y vacío, por lo que se reenvía al flujo en el que se editan los datos del alumno, pero con todos los campos vacíos.

Con todos estos atajos, se cierra el ciclo de acciones realizables por el usuario, pudiendo utilizar todos los elementos necesarios para completar una clasificación automatizada, empleando únicamente el teclado.

Uno de los problemas detectados durante el desarrollo de la tarea fue que la tecla *Tab* es un atajo de teclado utilizado comúnmente, por lo que, si el foco no estaba situado sobre uno de los elementos definidos para los atajos, el navegador asumía que esos atajos iban destinados a él, desplazándose sobre otros elementos no deseados o incluso enviando el foco fuera de la página.

Para solucionarlo se interceptaron todas las teclas pulsadas sobre la página y en caso de no coincidir con los atajos definidos se ignoran para evitar errores.

**Creación del reconocedor de calificaciones**

El nuevo reconocedor de calificaciones se desarrollará basándose en el diseño establecido en el punto 5.2.5. Para ello se utilizará la red neuronal CNN del *script* reconocedor de DNI como base, sobre la que se construirá el reconocedor de calificaciones. Para poder detectar la calificación se utilizará la plantilla del Anexo 2, siendo esta una adaptación de la plantilla original para el reconocedor de DNI con una nueva estructura preparada para contener la calificación.

La conversión del fichero PDF a imágenes PNG, se realizará empleando el comando *pdftoppm*, utilizado también en el reconocedor de DNI, que permite convertir ficheros PDF a imágenes especificando formatos, proporciones y recortes. Para ello, como argumentos se enviarán el fichero PDF y las proporciones y posiciones para el recorte inicial de la imagen. Esto nos devuelve una lista de imágenes recortadas que contendrán el fragmento de la página donde deben estar las calificaciones.

Se aprovechará que el reconocedor de DNI ya identifica las páginas de inicio para conocer en qué páginas estarán también las calificaciones. Cuando el reconocedor de DNI encuentre una página de inicio, se procesa la imagen recortada de la calificación para transformarla a escala de grises y se difumina para que se minimicen los posibles desperfectos causados por los dobleces del papel o por el escaneo de los exámenes.

Para evitar problemas de rendimiento en la búsqueda y recorte de la estructura, como los producidos en el reconocedor de DNI, se utilizará la librería *OpenCV* de Python para ello. Esta librería permite, entre otras cosas, la búsqueda de formas en imágenes de forma eficiente. Esto es de gran utilidad, pues permite detectar las figuras cuadradas en las que estarán contenidos los dígitos de la calificación. Para recortar los fragmentos de la imagen que contienen los dígitos, primero se buscan todos los contornos cerrados dentro de una imagen mediante la función *findContours*. Esta nos devuelve una lista de todos los contornos cerrados detectados.

Puesto que además de los cuadrados con los dígitos pueden existir otros contornos, será necesario filtrarlos para quedarnos únicamente con aquellos que sean cuadrados. Para decidir qué contornos son cuadrados y cuáles no, se comprueban las medidas del contorno, comprobando que la altura y el ancho sean proporcionales. En caso de no ser proporcionales, automáticamente serán descartados, pues todos los cuadrados deben tener la misma altura que base. La siguiente comprobación se hace utilizando la función *approxPolyDP*, que dado un contorno lo aproxima al polígono más cercano. Si el polígono al que se aproxima tiene cuatro lados, será un cuadrado y, por tanto, un dígito de la calificación. Adicionalmente, se comprueba que el contorno tenga un área considerable, para descartar posibles contornos cuadrados que puedan surgir de la colisión de los dígitos con los bordes del cuadro. Al no existir más cuadrados en el lugar donde debe estar la calificación, se asume que cada cuadrado pertenece a un dígito. Estos cuadrados se guardan en una lista para posteriormente ser procesados y adaptados al formato reconocido por la red neuronal. Este proceso de adaptación se conocerá como *normalización de imágenes*, en el cual se dejará un único canal de color en la imagen y se realizarán otras operaciones como el difuminado o el rellenado de espacios vacíos.

Con las imágenes ya normalizadas, podrán ser procesadas por la red neuronal para obtener una predicción de calificación que será añadida a la respuesta del reconocedor de textos.

Tras realizar pruebas con exámenes reales se observa que las predicciones aún no son muy precisas, esto es debido a que el procesamiento de las imágenes ha cambiado respecto al *script* reconocedor de DNI, por lo que será necesario invertir más tiempo para poder conseguir un mejor

procesado y una imagen que se adapte más a las características que requiere la red neuronal. El tiempo invertido para la unidad de trabajo alcanzó el máximo estimado, por lo que se creará otra unidad de trabajo como continuación de esta en el siguiente *sprint*.

**Integración del reconocedor de calificaciones**

La integración del reconocedor de calificaciones supone unificar el *script* reconocedor de DNI y el reconocedor de calificaciones para que trabajen conjuntamente y emitan una única respuesta que contenga todos los datos. Para ello, en la respuesta del reconocedor de DNI se añadirá un nuevo campo que contendrá la calificación del alumno.

Se unificará la respuesta utilizando el nuevo reconocedor de calificaciones como un módulo que puede ser llamado desde el reconocedor de DNI. Una vez se detecte una página de inicio en el reconocedor de DNI, se llamará al módulo reconocedor de calificaciones para obtener la calificación y añadirla a la respuesta. De esta forma se garantiza que solo se busquen calificaciones en aquellas páginas donde exista un DNI y, además, se mejora el rendimiento, ya que no se tiene que realizar otra vez la búsqueda de la página inicial donde está contenida la calificación.

**Comunicación API para detección de errores**

Para la detección de errores o inconsistencias detectables, mostradas en la Ilustración 14, se realiza una nueva entrada en la API, la cual es invocada por el *front-end* cada vez que efectúe un cambio o guarde un nuevo alumno. Esta entrada leerá la lista de alumnos calificados de la base de datos y buscará posibles solapes o discontinuidades, comparando los números de páginas de inicio y fin de la lista. En caso de encontrar algún estado inconsistente, se añade a una lista que es enviada como respuesta a la llamada de la API. Si no se encuentra ninguna inconsistencia, se devuelve la lista vacía, indicando que no se han detectado errores.

### 6.2.2 Finalización del segundo sprint

El segundo *sprint*, centrado principalmente en la corrección de errores, la finalización del ciclo de clasificación automatizada y la creación del reconocedor de calificaciones, concluye con normalidad a excepción de la unidad de trabajo relacionada con el reconocedor de calificaciones. Esta unidad de trabajo se finaliza, pero las predicciones aún no son lo suficientemente precisas como para concluir el reconocedor de calificaciones, por lo que se creará otra unidad de trabajo en el siguiente *sprint* como continuación.

A la entrega de la versión al *Product Owner*, se vuelve a tener en cuenta la retroalimentación obtenida. No se menciona ningún aspecto concreto a mejorar o modificar, por lo que se concluye que se alcanzan las expectativas definidas para este *sprint*.

Se comunica al *Product Owner* que el reconocedor de calificaciones, si bien está integrado y funcionando, aún no realiza predicciones con la máxima precisión, dejando claro que en el siguiente *sprint* se trabajará en ello en una unidad de trabajo de continuación.

No existen más tareas que tengan una prioridad tan elevada, por lo que el siguiente *sprint* será dedicado en su mayoría a la optimización y corrección de posibles errores que se detecten.

Durante la entrega de la nueva versión, no se identificaron comentarios negativos sobre el producto, por lo que se concluye que la metodología empleada ha funcionado adecuadamente, cumpliendo con los requisitos esperados. Además, la apertura de una nueva unidad de trabajo como continuación para mejorar la precisión del reconocedor de DNI demuestra la gran flexibilidad y adaptabilidad ofrecida por la metodología, permitiendo entregar una nueva versión incremental y obtener retroalimentación de los cambios que sí han podido ser desarrollados, en lugar de retrasar la entrega y aumentar el tiempo total de desarrollo.

## 6.3 TERCER SPRINT

El tercer *sprint* tendrá algunas consideraciones especiales al tratarse del último *sprint* del proyecto. Debido a la redacción de la memoria del trabajo de fin de grado será necesario dedicar más tiempo a ella, en lugar de a desarrollo. Por otra parte, tampoco existen unidades de trabajo prioritarias o esenciales para la entrega final, a excepción de algún error detectado y las optimizaciones de los reconocedores de DNI y calificaciones. Por ello, de las 100 horas de trabajo estimadas para el tercer *sprint*, se destinarán 65 horas a la redacción de la memoria y 35 horas al desarrollo de los últimos requisitos del proyecto. Las unidades de trabajo del tercer *sprint* se encuentran en el Anexo 6.

### 6.3.1 Unidades de trabajo

**Paralelización del reconocedor de DNI en conversión PDF a PNG**

La paralelización de la conversión de PDF a PNG se realizará respetando las pautas definidas en el punto 5.2.4. Para ello se emplea la librería *concurrent.futures* de *Python* y concretamente la clase *ProcessPoolExecutor*. Esta clase permite no solo la creación de procesos asíncronos, sino también la gestión de ellos para operaciones como detección de finalización o la lectura de las respuestas de los procesos. La clase *ProcessPoolExecutor* ofrece la función *submit*, que, dados una tarea o función y sus argumentos de entrada, ejecutará dicha tarea en nuevo proceso de forma concurrente. La función *submit* nos devuelve un objeto *Future*, que representa la ejecución asincrónica de la tarea. Una vez finalice la ejecución del proceso, este objeto contendrá el resultado que devuelva el proceso. Cada objeto *Future* se almacena en una lista y se esperará hasta que todos los procesos de la lista hayan finalizado su ejecución. Una vez todos los procesos hayan finalizado, se permitirá continuar al siguiente paso de la ejecución según la Ilustración 26.

Puesto que cada proceso tiene su propia copia de memoria, a mayor número de procesos mayor será el consumo de recursos, por lo que se minimizará el número de procesos en ejecución. Para ello se divide el número de páginas entre el número de núcleos del procesador. Esto nos devuelve la proporción óptima teórica de imágenes que puede tomar cada proceso, pues en un caso ideal todos los procesos se ejecutarían a la vez y finalizarían a la vez. Si, por el contrario, se crease un proceso por cada página del PDF, además de existir un gran consumo de memoria, al acabar de convertir cada imagen se tendría que realizar un cambio de contexto, eliminando el proceso finalizado y alternando a la ejecución de otro, con el correspondiente coste de tiempo y procesamiento.

Por tanto, la función que ejecutará cada proceso, en lugar de convertir una sola imagen, convertirá la proporción de imágenes resultante de la división del total de las imágenes entre el número de núcleos del procesador. En caso de que esta proporción sea una fracción, para simplificar el algoritmo, se podrá ejecutar la conversión de la misma imagen en dos procesos distintos. Esto, aunque pueda ser redundante, ya que se podrá convertir una misma imagen dos veces, es la solución más eficiente, ya que otros algoritmos para hacer un reparto equitativo de tareas como *Round Robin* aumentarían la complejidad y perjudicarían el rendimiento debido a su propia gestión. Si se diera el caso en que dos procesos convirtieran la misma imagen no causaría ningún problema, pues en el sistema de ficheros solo puede existir una imagen con el mismo nombre, por tanto, se conservaría la imagen del último proceso en guardarla.

**Paralelización del reconocedor de DNI en la búsqueda de estructura contenedora del DNI**

La paralelización de la búsqueda de la estructura contenedora del DNI se realizará de manera similar a la conversión de imágenes, con la diferencia de que en este caso el número de páginas o

los datos de posición obtenidos no se pueden repetir, pues podría provocar alumnos duplicados. Por tanto, se tendrá que hacer un reparto, aunque no tenga la misma carga para cada proceso, que no tenga páginas duplicadas, como en el caso de la conversión PDF. Para ello se reparte el total de las páginas en orden lineal, una a una, entre una lista con tantas entradas como núcleos del procesador existan, de esta forma nos aseguramos de que no existan páginas repetidas en varios procesos. La diferencia con el reparto realizado en la optimización anterior es que en ese caso el reparto se realizaba en bloques, teniendo siempre imágenes o páginas consecuentes, mientras que en este caso las imágenes se pueden intercalar y realizar saltos entre ellas.

Tras finalizar el reparto, tendremos una lista formada por listas de páginas. Cada una de estas listas de páginas se pasarán como argumento a la función *submit*, junto a la función encargada de buscar la estructura del DNI.

A medida que se ejecutan los procesos se guardarán los objetos *Future* en una lista y, una vez finalizados todos los procesos, se añadirán los resultados a las estructuras correspondientes para continuar con el siguiente paso de la ejecución.

**Paralelización del reconocedor de calificaciones en conversión PDF a PNG**

El proceso de paralelización para la conversión de PDF a PNG en el reconocedor de calificaciones es idéntico al realizado para el reconocedor de DNI, con la única diferencia de que el fragmento a recortar se encontrará en una localización distinta a la del DNI en la página. El resto del desarrollo es idéntico, por lo que no se detallará más.

**Corrección de un error en la descarga de exámenes**

Un error detectado durante la realización de pruebas de funcionamiento es el caso en que dos o más asociaciones no hayan podido identificar correctamente al alumno. Al presionar el botón de *Descargar Examen* se exportarán varios ficheros PDF con el mismo nombre, sobrescribiendo todos salvo el último, perdiendo información en el proceso.

La Ilustración 34 muestra el caso concreto donde se produce el error. Al no poder completarse los campos con un alumno real se rellenaban con la palabra por defecto "*Err*".


*Ilustración 34.* Error causado por entradas repetidas

Para solucionarlo, cuando no se pueda identificar correctamente al alumno, pero sí los números de las páginas, se añadirá la entrada con un nombre distinto e incremental que seguirá la estructura "Err + [Número de error]".

**Advertencias por errores detectados**

En esta unidad de trabajo se desarrolla la modificación en la interfaz para la detección de errores evitables definida en el punto 5.2.2 y concretamente en la Ilustración 21.

Para ello, en el *sprint* anterior, se desarrolló una entrada en la API que devolviese una lista con los errores detectados, dejando la única tarea pendiente de hacer la llamada a la API y mostrar las respuestas obtenidas.

Dado que cada modificación por parte del usuario puede causar errores, cada vez que se modifique la lista de alumnos calificados se realizará una consulta a la API para detectar nuevos posibles

fallos. Estos fallos se almacenarán en una lista en el *front-end* y, en caso de haberlos, se mostrarán en la interfaz junto al botón de descargar. Estos errores se separarán en tres categorías: solapes, discontinuidades y errores de predicción. Estos últimos existirán si alguno de los campos de cualquiera de los alumnos de la lista no está definido o contiene la estructura "Err + [Número de error]".

La Ilustración 35 muestra el resultado final de la interfaz con algunos errores de ejemplo.

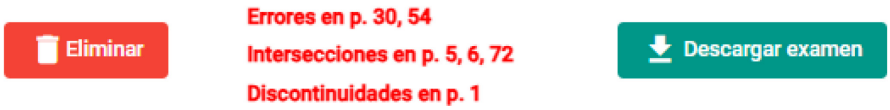



*Ilustración 35.* Interfaz de detección de errores

**Continuación del desarrollo del reconocedor de calificaciones**

Aunque la anterior unidad de trabajo dedicada al reconocedor de calificaciones concluyó en el segundo *sprint*, se detectó que la precisión de las predicciones aún no era lo suficientemente buena, por lo que se abrió una unidad de trabajo como continuación en este tercer *sprint.*

El problema se encontraba en el procesado de las imágenes, previo al uso de la red neuronal para realizar las predicciones. La Ilustración 36 muestra a la izquierda la imagen tras el nuevo procesado y a la derecha un ejemplo de procesado en el reconocedor de DNI, que tiene la máxima precisión.

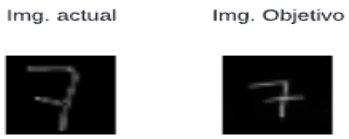



*Ilustración 36.* Comparación del procesado de imágenes

Se aprecia que hay una clara diferencia entre ellas, siendo las más destacables, el difuminado excesivo y la presencia de espacios vacíos en el contorno del dígito.

Para solucionar esta falta de precisión se hacen pruebas cambiando diferentes parámetros del procesado hasta lograr una imagen con más similitud a la imagen objetivo. No existe una forma eficiente, por lo que se realiza mediante prueba y error hasta obtener mejoras en la predicción de la red neuronal.

### 6.3.2 Finalización del tercer sprint

El tercer *sprint*, centrado principalmente en la corrección de algún error detectado y la optimización de los reconocedores, tanto de DNI como de calificaciones, concluye habiendo finalizado todas las unidades de trabajo planteadas. A la entrega del producto y en una demostración al *Product Owner* no se identifican nuevos problemas ni se mencionan aspectos negativos, concluyendo que el producto satisface las expectativas con éxito.

No existen más unidades de trabajo en el *backlog* ni requisitos por cumplir, por lo que, habiendo concluido el tercer *sprint*, se da por finalizado el desarrollo del proyecto.

# 7 PRUEBAS

Las pruebas desempeñan un papel crucial en el desarrollo y validación de cualquier sistema, especialmente cuando está diseñado para tareas tan delicadas como la detección automática del DNI o la calificación de exámenes, donde la precisión y la fiabilidad de los resultados no solo impactan directamente en la experiencia de usuario, sino que son fundamentales para asegurar la confianza y la utilidad real del asistente. Para el caso concreto del proyecto, no solo será importante la exactitud de los resultados, sino que se tendrá también en cuenta el tiempo de procesamiento hasta la obtención de los resultados y el tiempo necesario para finalizar la clasificación, una vez obtenidos los resultados.

Como conjunto de datos se usaron dos ficheros PDF con las estructuras necesarias para la detección en las páginas iniciales de cada examen, correspondientes a las estructuras definidas en la plantilla del Anexo 2.

La primera evaluación corresponde a un conjunto de exámenes reales de alumnos, formados por 16 exámenes diferentes escritos por personas diferentes, con una extensión de 6 páginas por examen, dando un total de 96 páginas.

La segunda evaluación corresponde a un conjunto de exámenes creados por el desarrollador del proyecto, formados por 20 exámenes, con DNI y calificación manuscritos, con una extensión de entre 4 y 7 páginas por examen, dando un total de 109 páginas.

## 7.1 PRECISIÓN

En la Tabla 26 se muestran los resultados de precisión obtenidos a partir de las pruebas realizadas.

El número máximo de aciertos posibles, considerando algunas particularidades, estará representado en la columna *objetivo*. Por otra parte, los aciertos obtenidos por los reconocedores se verán reflejados en la columna *resultado*. Estos resultados reflejan, en general, una capacidad correcta de identificación de los datos contenidos en los exámenes. Destaca sobre todo la capacidad para acertar los números de páginas, logrando en ambos casos acertar en todos los conjuntos de exámenes. Por otra parte, la cantidad de aciertos de la detección de calificaciones, si bien es aceptable, aún es algo mejorable, dando paso a una posible futura mejora de ese módulo en concreto.

*Tabla 26.* Pruebas de precisión

| | **Evaluación 1** | | | **Evaluación 2** | | |
|---|---|---|---|---|---|---|
| | Objetivo | Resultado | Proporción de Acierto | Objetivo | Resultado | Proporción de Acierto |
| **Aciertos en páginas** | 16 | 16 | 100 % | 20 | 20 | 100 % |
| **Aciertos en identificación** | 14* | 10 | 71.42 % | 20 | 18 | 90 % |
| **Aciertos en calificación** | 15** | 9 | 60 % | 20 | 13 | 65 % |
| * Se descartaron dos exámenes por no seguir el formato de calificación establecido.<br>** Un examen fue descartado por haber añadido caracteres alfabéticos en los recuadros de calificación | | | | | | |

Durante la realización de las pruebas se detectó que cuando se producían la mayoría de los errores, tanto en identificación de DNI como en calificaciones, era debido a que los dígitos manuscritos llegaban a tocar los bordes de las casillas que los contenían, afectando considerablemente a la predicción. Esto afecta en mayor medida a la cantidad de aciertos en calificaciones, ya que un

único error en cualquier dígito invalida automáticamente el dato completo, en cambio, un error en el DNI puede no tener repercusiones al utilizarse algoritmos como la Distancia de Levenshtein, que son adecuados para soportar una cierta cantidad de errores.

## 7.2 RENDIMIENTO

La Tabla 27 muestra el resultado de las pruebas de rendimiento. La descripción de cada columna se encuentra a continuación en orden de aparición.

*Tabla 27.* Pruebas de rendimiento

| | T. total proceso manual (s) | T. procesado (s) | T. validación (s) | T. total automatizado (s) | Aceleración global | Aceleración sobre acciones |
|---|---|---|---|---|---|---|
| Evaluación 1 | 7’17’’ | 58’’ | 1’ 03’’ | 2’ 01’’ | 261 % | 593 % |
| Evaluación 2 | 8’10’’ | 1’ 07’’ | 40’’ | 1’ 33’’ | 426 % | 1125 % |

**T. total proceso manual (s):** Tiempo total requerido para clasificar manualmente los exámenes

**T. procesado automatizado (s):** Tiempo desde que el usuario inicia la clasificación automatizada del examen hasta que se obtienen los resultados, se puede entender como el tiempo de respuesta.

**T. validación (s):** Tiempo invertido en comprobar y realizar modificaciones a los resultados automatizados obtenidos hasta validar todas las entradas.

**T. total automatizado (s):** Suma del tiempo de procesado automatizado y el tiempo de validación, dando lugar al tiempo total hasta obtener una clasificación totalmente correcta.

**Aceleración global**: Porcentaje de aceleración entre el tiempo total invertido en la clasificación automatizada y el tiempo total del proceso manual.

$$A.Global = \left(\frac{T.Total\ proceso\ Manual}{T.Total\ Automatizado} - 1\right) \times 100$$

**Aceleración sobre acciones:** Porcentaje de aceleración respecto al tiempo invertido en realizar acciones o modificaciones en la interfaz entre el tiempo invertido en el modelo automatizado y el tiempo total del proceso manual.

$$A.\ Acciones = \left(\frac{T.\ Total\ proceso\ manual}{T.\ Validaciones} - 1\right) \times 100$$

Como se puede apreciar, todas las aceleraciones calculadas demuestran haber conseguido una mejora en los tiempos de clasificación, pero destacan sobre todo los valores para la aceleración por acciones, indicando que el tiempo que los usuarios tienen que invertir en realizar acciones manuales se ha reducido drásticamente.

## 7.3 CONCLUSIONES

Los resultados obtenidos en las pruebas realizadas demuestran que el sistema desarrollado, en general, es efectivo para la automatización de la clasificación de exámenes, logrando una alta precisión y mejora en el tiempo invertido para la clasificación, en la que destaca sobre todo la eficiencia del nuevo proceso, logrando una aceleración global de hasta el 426 %.

Por otra parte, ha permitido identificar algún área de mejora como la identificación de dígitos, en la que, cuando los dígitos colisionan con el recuadro que los contiene, presenta dificultades para reconocerlo, dando paso a una posible mejora futura donde se revise la estructura donde están contenidos los dígitos para facilitar su detección.

En conclusión, las pruebas realizadas confirman que el asistente ha logrado mejorar de manera significativa el proceso de clasificación de exámenes, reduciendo considerablemente el tiempo dedicado a esta tarea y obteniendo una gran precisión en los resultados.

# 8 DESPLIEGUE

## 8.1 ENTORNO

Para el desarrollo y pruebas del proyecto se empleó el sistema operativo Ubuntu en su versión 22.04.4 LTS, aunque puede ser ejecutado en cualquier sistema basado en Linux.

Las especificaciones del sistema final variarán, ya que la aplicación está pensaba para ser alojada en un servidor de la universidad, aunque para el desarrollo y las pruebas se utilizó una máquina virtual funcionando en Oracle VM VirtualBox, empleando 6 núcleos de un procesador Intel Core i5-11400H, 5 GB de memoria RAM y una partición de memoria de 80 GB de un disco duro SSD NVMe M.2 de 1 TB de capacidad.

## 8.2 INSTALACIÓN

El primer paso para instalar el proyecto será obtener los archivos del proyecto. Estos se pueden solicitar a través del correo jpcaijam@upv.edu.es, al encargado del desarrollo, o bien a anmarti@upv.edu.es, responsable del mantenimiento de la aplicación y tutor de este trabajo.

Para facilitar la instalación del proyecto, se ha creado un fichero, disponible en el Anexo 7, que detalla los pasos necesarios y los comandos específicos a ejecutar. Este fichero contiene las instrucciones necesarias para instalar las dependencias y crear la base de datos.

El proceso de creación de la base de datos dentro de las instrucciones anteriores tendrá dos variantes, pues se puede dar el caso en que se esté actualizando el proyecto desde la versión anterior de Madrina o bien que se esté realizando una instalación desde cero.

En caso de una instalación desde cero, se pueden seguir las instrucciones con normalidad, ya que el fichero *BBDD.sql*, incluido en el Anexo 8, es el fichero por defecto. Para el caso de una actualización de Madrina a la última versión, se crea el fichero *updateBBDD.sql*, disponible en el Anexo 9, el cual se tendrá que utilizar en lugar del fichero *BBDD.sql* en el paso correspondiente. Independientemente del caso, se necesitará tener cualquiera de los dos ficheros descargado en el sistema.

## 8.3 ARRANQUE

Para iniciar el proyecto será necesario arrancar tanto el servidor *front-end* como el servidor *back-end*.

Para el caso del *front-end*, nos situaremos en la carpeta del proyecto del *front-end* y en una terminal ejecutaremos el comando *ng build*. Esto creará en la carpeta "*dist/"* los ficheros compilados del *front-end*, que posteriormente podrán ser ejecutados en un servidor, en nuestro caso, añadiéndolos en la carpeta "*var/www/html*".

Para el *back-end*, en otra terminal, nos situamos en el directorio padre del proyecto *back-end* y utilizaremos el archivo *start_backend.sh*, incluido en el Anexo 10, bien ejecutándolo de forma directa de la forma *"./start_backend.sh"* o bien mediante un gestor de procesos como PM2, utilizando el comando *"pm2 start start_backend.sh"*.

En ambos casos el *script* deberá tener permisos de ejecución, los cuales se pueden añadir utilizando el comando *"chmod +x start_backend.sh"*.

# 9 CONCLUSIONES

En este Trabajo de Fin de Grado se ha diseñado, desarrollado e implementado la automatización de un asistente web para la publicación de exámenes corregidos y calificaciones, completando con éxito los objetivos propuestos al inicio del proyecto. El asistente ofrece una solución eficaz que mejora significativamente el proceso de clasificación y entrega de exámenes corregidos, reduciendo significativamente el tiempo invertido en estas actividades. Esto ha sido posible gracias a la integración exitosa de tecnologías avanzadas como la inteligencia artificial y el procesamiento de imágenes, aspectos fundamentales para el éxito del proyecto. Por otra parte, ha sido diseñado para ser intuitivo y respetuoso con el modelo de clasificación anterior, facilitando así su adopción, tanto a los usuarios actuales, como a futuros usuarios. Además, se ha tenido siempre en cuenta la seguridad del sistema para garantizar una experiencia de uso segura y robusta.

El desarrollo de este proyecto ha presentado desafíos técnicos que han requerido la adquisición de nuevos conocimientos y habilidades como el uso de inteligencia artificial para el reconocimiento de textos o el procesamiento de imágenes, que han sido áreas de aprendizaje cruciales, ya que estas tecnologías no se abordan durante los estudios universitarios. Sin embargo, ha proporcionado una valiosa oportunidad para explorar estas áreas y aplicarlas a un proyecto real. Por otra parte, se ha realizado un análisis exhaustivo y se han aplicado principios de seguridad y optimización de algoritmos que, aunque se abordan en el grado, han requerido un enfoque más profundo y especializado para aplicarlos con éxito al contexto y tecnologías del proyecto.

La elección de una metodología de trabajo ágil, como SCRUM, ha sido fundamental para gestionar de manera eficiente el desarrollo del asistente, permitiendo iteraciones rápidas y ajustes continuos que han ayudado a mejorar la calidad y efectividad del producto final. Este enfoque no solo ha optimizado el proceso de desarrollo, sino que también ha permitido una adaptación efectiva a unos requisitos que inicialmente no podían ser bien definidos y que podían variar a medida que se avanzaba en el desarrollo.

El proceso de desarrollo también ha permitido reforzar algunas de las competencias transversales [25] establecidas por la UPV, como la "responsabilidad y toma de decisiones", esenciales en el análisis y diseño del proyecto; el "compromiso social y medioambiental", al considerar el impacto que la solución puede tener en el contexto social y educativo, y la "comunicación efectiva", necesaria para argumentar y comunicar de manera clara y precisa los avances y decisiones tomadas durante el desarrollo del proyecto.

En conclusión, el trabajo realizado no solo cumple con los objetivos iniciales planteados, sino que también ha ofrecido una experiencia profundamente enriquecedora. Ha permitido adquirir nuevos conocimientos en áreas como la inteligencia artificial o el procesamiento de imágenes, así como el refuerzo y la aplicación de conocimientos obtenidos durante los estudios como la gestión de proyectos, la integración de sistemas o el desarrollo de software. Todo esto supone un crecimiento profesional significativo, que se traduce en una mayor capacidad para enfrentar desafíos complejos en el futuro, y en unos conocimientos sólidos que sin duda serán de utilidad en el entorno profesional y permitirán afrontar con éxito nuevos proyectos.

[25] https://www.upv.es/entidades/agt/competencias-transversales/

## 9.1 RELACIÓN DEL TRABAJO DESARROLLADO CON LOS ESTUDIOS CURSADOS

El proyecto desarrollado tiene una relación directa con los conocimientos adquiridos durante la etapa universitaria. Asignaturas como Gestión de proyectos y Proceso de software han sido esenciales para una correcta planificación y gestión del proyecto, así como la correcta aplicación de una metodología de trabajo. Otras asignaturas como Fundamentos de sistemas operativos, Concurrencia y sistemas distribuidos o Computación paralela han sido cruciales para el desarrollo de los componentes más complejos del proyecto, como los reconocedores de dígitos, donde la implementación de procesos concurrentes y la gestión eficiente de recursos eran indispensables para ofrecer un rendimiento optimizado del sistema. Por otra parte, asignaturas como Tecnología de sistemas de información en la red o Integración e interoperabilidad han sido fundamentales para el proceso de comunicación e integración entre los distintos módulos del proyecto. Finalmente, asignaturas como Programación o Estructura de datos y algoritmos han aportado las bases necesarias para el desarrollo de código efectivo y eficiente.

En resumen, el proyecto ha permitido aplicar de manera práctica y coordinada una amplia gama de conocimientos adquiridos durante la etapa universitaria, que han servido para alcanzar un mayor dominio en aquellas áreas clave y evidencia una relación adecuada entre el trabajo desarrollado y los estudios universitarios.

# 10 TRABAJOS FUTUROS

Durante la etapa de pruebas se detectó que tanto el reconocedor de calificaciones como el reconocedor de DNI presentaban fallos de predicción cuando los dígitos estaban muy cerca de los bordes del recuadro donde están contenidos. Esto, en el caso del reconocimiento de DNI, no era tan relevante, ya que, aunque tuviera algún fallo, el algoritmo empleado para la asociación de los DNI toleraba una cierta cantidad de fallos, permitiendo que la asociación final fuera correcta.

En cambio, en el caso del reconocimiento de calificaciones se requiere una precisión absoluta, ya que cualquier fallo en la predicción de un dígito invalida automáticamente toda la calificación. Aunque la implementación actual ofrezca unos resultados aceptables, afecta notablemente a la reducción del tiempo requerido por el usuario, pues debe introducir manualmente la calificación en caso de no ser correcta. Aparece entonces una posible mejora donde se incremente la precisión de este módulo. Para ello se plantea la realización de modificaciones en la estructura donde está contenida la calificación, aumentando el tamaño para que no se lleguen a tocar los bordes o bien diseñando una nueva estructura que permita mejorar la detección de los dígitos. Otra posible opción es añadir más procesamiento a las imágenes, añadiendo bordes artificialmente a las imágenes recortadas de los dígitos, aunque esto, al reducir el tamaño del dígito, podría afectar a la precisión general de la red neuronal. Finalmente, la opción más segura, aunque más complicada, sería el desarrollo de una nueva red neuronal que sea capaz de ignorar los bordes del cuadro donde están contenidos los dígitos sin afectar a la precisión del resultado.

# 11 BIBLIOGRAFÍA


[1] A. Martí Campoy, J. M. Cecilia Canales, M. Agustí Melchor y V. L. Atienza Vanacloig, «Estudio sobre la percepción del estudiantado de la retroalimentación de los exámenes presenciales,» de *IN-RED 2023: IX Congreso de Innovación Educativa y Docencia en Red*, Valencia, España, 2023.

[2] F. G. F. Paglia, «RiuNet,» 2022. [En línea]. Available: http://hdl.handle.net/10251/185224.

[3] M. Laleu, «Python application to process images of handwritten digits,» DISCA-ENSEA internship report, 2023.

[4] J. Ferrandis Jorge, «RiuNet,» 2021. [En línea]. Available: http://hdl.handle.net/10251/174870.

[5] J. Maleplate, «Implementation of a Neural Network for handwritten digits recognition,» DISCA-ESIGELEG internship report, 2022.

[6] J. P. Mueller y L. Massaron, Machine learning for dummies, Hoboken, New Jersey: John Wiley & Sons, 2019.

[7] K. Schwaber y J. Sutherland, «The scrum guide,» *Scrum Alliance,* vol. 21, nº 1, pp. 1-38, 2011.

[8] H. Takeuchi y I. Nonaka, «The new new product development game,» *Harvard business review,* 1986.

[9] K. Schwaber, «Scrum development process,» de *OOPSLA 1995*, Austin, Texas, 1997.

[10] D. Wells, «A gentle introduction to Extreme Programming,» [En línea]. Available: http://www.extremeprogramming.org/.

[11] K. Beck, Extreme programming explained: embrace change, Addison-Wesley, 2000.

[12] R. Pressman, Software engineering : a practitioner's approach, New York: McGraw-Hill, 2006.

[13] P. Grassle, H. Baumann y P. Baumann, UML 2.0 in Action A Project-Based Tutorial, 1st ed., Packt Publishing, 2005.

[14] A. Garrido Tejero, «Comunicación entre capas. Patrón controlador y DAL (Data Access Layer),» Universitat Politècnica de València, 2016.

[15] A. J. Bernstein, «Analysis of programs for parallel processing,» *IEEE Transactions on Electronic Computers,* nº 5, pp. 757-763, 1966.

[16] D. Beazley, «Understanding the python GIL,» de *PyCON Python Conference*, Atlanta, Georgia, 2010.

[17] A. Freeman, Pro Angular : build powerful and dynamic web apps, Apress, 2022.

[18] M. Lutz, Learning python: Powerful object-oriented programming, Fifth ed., O'Reilly, 2013.

[19] V. I. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals, vol. 10, Soviet physics doklady, 1966, pp. 707-710.

# 12 ANEXO

*Anexo 1.* Plantilla script reconocedor DNI

| EXAMEN 31 de OCTUBRE de 2022<br>ETSIT FCO Primer parcial, temas 1 y 2. | DNI: | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Apellidos: Nombre: | | | | | | | | | | |

**Instrucciones.**
- Escriba el DNI, nombre y apellidos con CLARIDAD.
- Escriba las iniciales del nombre y apellidos en **todas** las hojas. Hágalo ahora.
- No retire la grapa ni separe las hojas.
- Responda las cuestiones en los espacios reservados.
- No puede utilizar ningún material ni equipamiento de ayuda.
- La duración del examen es de 2 horas desde que se le entregó este examen.

**Cuestión 1 (1.5 puntos).**

*Anexo 2.* Plantilla adaptada a reconocedor de calificaciones

| EXAMEN 31 de OCTUBRE de 2022<br>ETSIT FCO Primer parcial, temas 1 y 2. | DNI: | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Apellidos: | | | Nombre: | | | | | | | |

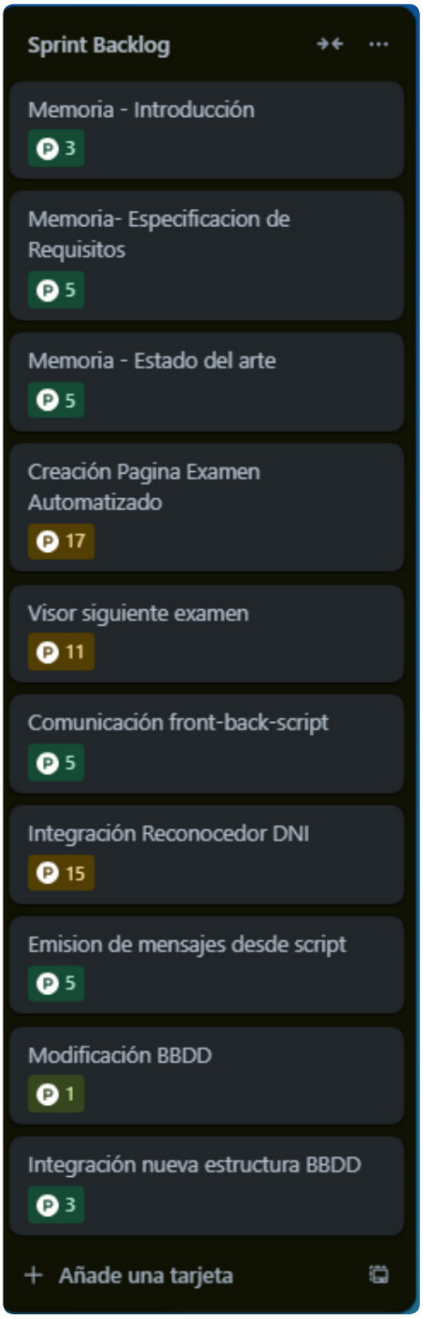
| **Instrucciones.**<br>- Escriba el DNI, nombre y apellidos con CLARIDAD.<br>- Escriba las iniciales del nombre y apellidos en **todas** las hojas. Hágalo ahora.<br>- No retire la grapa ni separe las hojas.<br>- Responda las cuestiones en los espacios reservados.<br>- No puede utilizar ningún material ni equipamiento de ayuda.<br>- La duración del examen es de 2 horas desde que se le entregó este examen. | | | , | | |
|---|---|---|---|---|---|

**Cuestión 1 (1.5 puntos).**

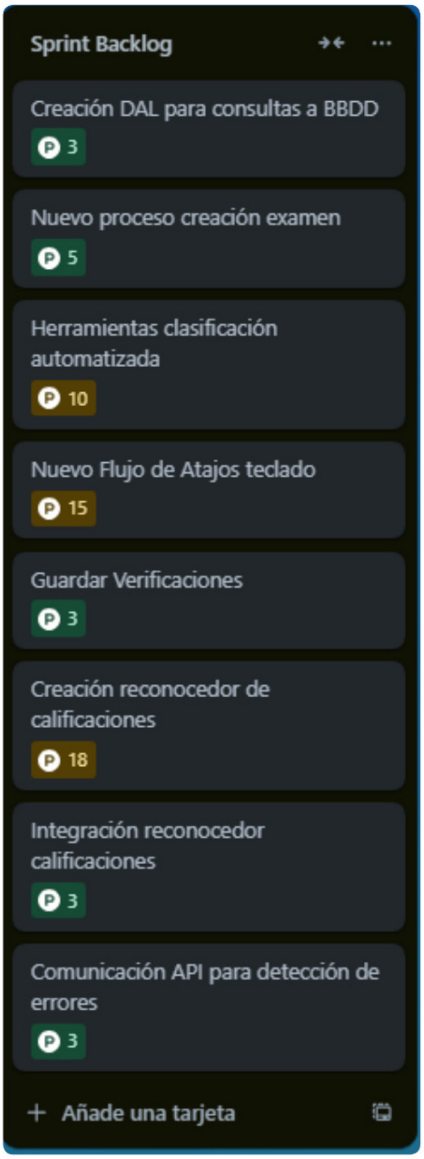
*Anexo 3.* Modelo de calidad ISO/IEC 25010

| | Característica | Subcaracterística |
|---|---|---|
| **ISO/IEC 25010** | Adecuación funcional | Completitud funcional<br>Corrección funcional<br>Pertinencia funcional |
| | Eficiencia de desempeño | Comportamiento temporal<br>Utilización de recursos<br>Capacidad |
| | Compatibilidad | Coexistencia<br>Interoperabilidad |
| | Capacidad de interacción | Reconocibilidad de adecuación<br>Aprendizabilidad<br>Operabilidad<br>Protección frente a errores de usuario<br>Involucración del usuario<br>Inclusividad<br>Asistencia al usuario<br>Auto-descriptividad |
| | Fiabilidad | Ausencia de fallos<br>Disponibilidad<br>Tolerancia a fallos<br>Recuperabilidad |
| | Seguridad | Confidencialidad<br>Integridad<br>No-repudio<br>Responsabilidad<br>Autenticidad<br>Resistencia |
| | Mantenibilidad | Modularidad<br>Reusabilidad<br>Analizabilidad<br>Capacidad de ser modificado<br>Capacidad de ser probado |
| | Flexibilidad | Adaptabilidad<br>Escalabilidad<br>Instalabilidad<br>Reemplazabilidad |
| | Protección | Restricción operativa<br>Identificación de riesgos<br>Protección ante fallos<br>Advertencia de peligro<br>Integración segura |

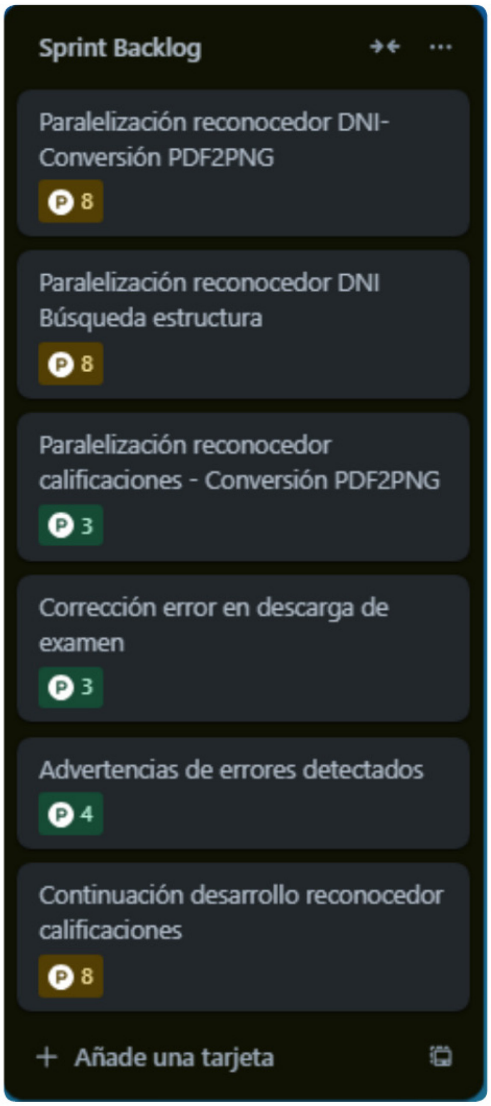
*Anexo 4.* Unidades de trabajo Sprint 1



*Anexo 5.* Unidades de trabajo Sprint 2



*Anexo 6.* Unidades de trabajo Sprint 3



*Anexo 7.* Archivo de instalación general

```
#Usado en un sistema Ubuntu 22.04.4 LTS
#En caso de otro sistema utilizar otro gestor (yum, apt, etc)
sudo apt-get update
sudo apt-get upgrade
#FRONTEND y BACKEND
sudo apt-get curl
#Instalar node v20
curl -sL https://deb.nodesource.com/setup_20.x | sudo -E bash -
sudo apt-get install nodejs -y
#Situate en la carpeta del front-end
#Instalar las dependencias del proyecto, en caso de no tenerlas (node-
modules)
npm i --legacy-peer-deps
sudo npm install -g @angular/cli

#Base de datos
sudo apt-get install mysql-server
sudo mysql
#Ya en MySQL, creamos un usuario con contraseña local ya que en las
últimas versiones se pone por defecto una contraseña SHA-2
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY
'Contraseña';
CREATE DATABASE MargotDB;
#Importar base de datos con el script BBDD.sql
exit #Salimos de MySQL
#Cambiar BBDD.sql por updateBBDD.sql en caso de actualizar
mysql -u root -p MargotDB < BBDD.sql
#reiniciamos
sudo service mysql stop
sudo service mysql start

#Reconocedores de dígitos Python
sudo apt-get install python3 #Instalar python3
sudo apt-get install python3-pip #Instalar pip
pip install PyPDF2
pip install pillow
pip install matplotlib
pip install scikit-image
pip install numpy
pip install opencv-python
pip install torch torchvision
pip install scikit-learn
pip install tqdm
pip install pandas
pip install pyarrow

#Instalación cpdf, sólo si no está ya instalado
# Descargar binario https://github.com/coherentgraphics/cpdf-binaries
sudo mkdir /usr/local/bin/cpdf
sudo mv cpdf /usr/local/bin/cpdf/
nano ~/.bashrc
#Añadir la siguiente línea al final del archivo .bashrc
export PATH=$PATH:/usr/local/bin/cpdf/
#Guardar los cambios y salir
source ~/.bashrc

#Guardar las credenciales de la base de datos
nano .env
#En el fichero añadimos las siguientes líneas con los datos necesarios
```

*Anexo 8.* Fichero BBDD.sql

```
CREATE TABLE `alumnos` (
  `id` int NOT NULL AUTO_INCREMENT,
  `apellidos` varchar(256) NOT NULL,
  `asignatura` int NOT NULL,
  `dni` varchar(64) NOT NULL,
  `nombre` varchar(256) NOT NULL,
  PRIMARY KEY (`id`)
)    ENGINE=InnoDB    AUTO_INCREMENT=1    DEFAULT    CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `asignaturas` (
  `id`   int   NOT   NULL   AUTO_INCREMENT   COMMENT   'id   numérico
autoincrement',
  `acronimo` varchar(16) NOT NULL COMMENT 'normalmente 3 carácteres,
pero se da cierto margen',
  `color` varchar(7) NOT NULL COMMENT 'hexadecimal incluyendo #',
  `curso` varchar(5) NOT NULL COMMENT 'xx/xx',
  `nombre` varchar(256) NOT NULL COMMENT 'nombre de la asignatura',
  `usuario` varchar(24) NOT NULL COMMENT 'id del usuario relacionado
con la asignatura',
  `visible` tinyint(1) NOT NULL COMMENT 'campo que se deja por las
dudas',
  PRIMARY KEY (`id`)
)    ENGINE=InnoDB    AUTO_INCREMENT=1    DEFAULT    CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `calificados` (
  `id` int NOT NULL AUTO_INCREMENT,
  `apellidos`   varchar(256)   CHARACTER   SET   utf8mb4   COLLATE
utf8mb4_0900_ai_ci NOT NULL,
  `dni` varchar(64) NOT NULL,
  `examen` int NOT NULL,
  `fin` int NOT NULL,
  `ini` int NOT NULL,
  `nombre` varchar(256) NOT NULL,
  `nota` float NOT NULL,
  `verificado` tinyint NOT NULL DEFAULT '1',
  PRIMARY KEY (`id`)
)    ENGINE=InnoDB    AUTO_INCREMENT=20    DEFAULT    CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `correo_validacion` (
  `usuario` varchar(24) NOT NULL,
  `token` varchar(32) NOT NULL,
  `creacion` timestamp NOT NULL,
  PRIMARY KEY (`usuario`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `examenes` (
  `id` int NOT NULL AUTO_INCREMENT,
  `asignatura` int NOT NULL,
  `fecha` bigint NOT NULL,
  `nombre` varchar(256) NOT NULL,
  `visible` tinyint(1) NOT NULL,
  `disabled` tinyint(1) NOT NULL,
  `saltoPag` int NOT NULL,
  `saltoPagActivo` tinyint(1) NOT NULL,
```

```
  `isAvanzado` tinyint(1) DEFAULT '0',
  `predicted` tinyint(1) NOT NULL DEFAULT '0',
  PRIMARY KEY (`id`)
)    ENGINE=InnoDB    AUTO_INCREMENT=5    DEFAULT    CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;


CREATE TABLE `preferencias` (
  `id` varchar(24) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci
NOT NULL,
  `cabecera` tinyint(1) NOT NULL,
  `encode`    varchar(24)    CHARACTER    SET    utf8mb4    COLLATE
utf8mb4_0900_ai_ci NOT NULL,
  `sep` varchar(1) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `usuarios` (
  `id` varchar(24) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci
NOT  NULL  COMMENT  'id  de  24  carácteres  alfabéticos  generados
aleatoriamente  que  se  usará  como  código  único  para  identificar  al
usuario',
  `correo` varchar(256) NOT NULL,
  `nombre`    varchar(256)    CHARACTER    SET    utf8mb4    COLLATE
utf8mb4_0900_ai_ci NOT NULL COMMENT 'nombre del usuario',
  `password`    varchar(256)    CHARACTER    SET    utf8mb4    COLLATE
utf8mb4_0900_ai_ci NOT NULL COMMENT 'contraseña que no codificaremos
aún',
  `token` varchar(64) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

*Anexo 9.* Fichero updateBBDD.sql

```
ALTER TABLE calificados
ADD COLUMN verificado TINYINT NOT NULL DEFAULT '1';

ALTER TABLE examenes
ADD COLUMN isAvanzado TINYINT(1) DEFAULT '0',
ADD COLUMN predicted TINYINT(1) NOT NULL DEFAULT '0';
```

*Anexo 10.* Fichero start_backend.sh

```
cd backend/
npm start > log.txt 2>error.txt &
echo $! > PID.txt
```

*Anexo 11.* Objetivos de desarrollo sostenible

# OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

| **Objetivos de Desarrollo Sostenibles** | **Alto** | **Medio** | **Bajo** | **No Procede** |
|---|---|---|---|---|
| ODS 1. **Fin de la pobreza.** | | | | × |
| ODS 2. **Hambre cero.** | | | | × |
| ODS 3. **Salud y bienestar.** | | | | × |
| ODS 4. **Educación de calidad.** | × | | | |
| ODS 5. **Igualdad de género.** | | | | × |
| ODS 6. **Agua limpia y saneamiento.** | | | | × |
| ODS 7. **Energía asequible y no contaminante.** | | | | × |
| ODS 8. **Trabajo decente y crecimiento económico.** | × | | | |
| ODS 9. **Industria, innovación e infraestructuras.** | | × | | |
| ODS 10. **Reducción de las desigualdades**. | | | | × |
| ODS 11. **Ciudades y comunidades sostenibles**. | | | | × |
| ODS 12. **Producción y consumo responsables.** | | | | × |
| ODS 13. **Acción por el clima.** | | | | × |
| ODS 14. **Vida submarina.** | | | | × |
| ODS 15. **Vida de ecosistemas terrestres.** | | | | × |
| ODS 16. **Paz, justicia e instituciones sólidas.** | × | | | |
| ODS 17. **Alianzas para lograr objetivos.** | | | | × |

El Trabajo de Fin de Grado es una gran oportunidad para que los estudiantes conecten su formación académica con los Objetivos de Desarrollo Sostenible (ODS). A través de sus investigaciones y proyectos, los estudiantes pueden contribuir al desarrollo sostenible, aplicando los conocimientos aprendidos durante su trayectoria universitaria para enfrentar desafíos globales. Esto no solo mejora la experiencia educativa, sino que ayuda a adquirir habilidades técnicas y profesionales, esenciales para tener un impacto positivo en la sociedad.

A continuación, se explicará cómo este proyecto se relaciona con algunas de los ODS, haciendo mención a metas concretas definidas en cada uno de ellos.

**ODS 4. Educación de calidad**

Meta 4.4 Aumentar considerablemente el número de jóvenes y adultos que tienen las competencias necesarias, en particular técnicas y profesionales, para acceder al empleo, el trabajo decente y el emprendimiento.

Este proyecto está relacionado en gran medida con esta meta, ya que permite a los estudiantes poder acceder fácilmente a sus exámenes corregidos, contribuyendo a aprender de sus errores y

obtener un aprendizaje profundo y personalizado, que se verá traducido en una mejora de sus capacidades técnicas y profesionales.

**ODS 8. Trabajo decente y crecimiento económico**

Meta 8.2 Lograr niveles más elevados de productividad económica mediante la diversificación, la modernización tecnológica y la innovación, entre otras cosas centrándose en los sectores con gran valor añadido y un uso intensivo de la mano de obra.

El proyecto automatiza la clasificación y publicación de exámenes corregidos, reduciendo el tiempo invertido en comparación con el proceso manual, permitiendo que el profesorado vea reducida la carga de trabajo destinada a ello y dando lugar a que puedan desarrollar actividades con mayor valor añadido como la enseñanza o el desarrollo de contenido educativo.

**ODS 9. Industria, innovación e infraestructuras.**

Meta 9.1 Desarrollar infraestructuras fiables, sostenibles, resilientes y de calidad, incluidas infraestructuras regionales y transfronterizas, para apoyar el desarrollo económico y el bienestar humano, haciendo especial hincapié en el acceso asequible y equitativo para todos.

El asistente web desarrollado en este proyecto puede ser considerado una infraestructura digital, que mejora la calidad y la fiabilidad del proceso de entrega de exámenes corregidos. Además, al ofrecer una solución tecnológica asequible y equitativa, se facilita el acceso a una educación de calidad, contribuyendo así al bienestar humano y al desarrollo económico.

**ODS 16. Paz, justicia e instituciones sólidas**

Meta 16.6 Crear a todos los niveles instituciones eficaces y transparentes que rindan cuentas.

El proyecto apoya la transparencia y la rendición de cuentas en el proceso de evaluación educativa. Mediante el uso del asistente web, tanto estudiantes como profesores tendrán un acceso claro y directo a los exámenes, reduciendo la posibilidad de errores o malentendidos. Al permitir la fácil detección de problemas en las correcciones, se refuerza la confianza en las instituciones educativas y se garantiza que las correcciones sean justas y transparentes, contribuyendo a construir instituciones más sólidas y responsables.

En conclusión, este Trabajo de Fin de Grado no solo representa un avance en la automatización del proceso de clasificaciones de exámenes, sino que tiene un impacto significativo en estudiantes y profesores. Al alinearse con varios ODS, el proyecto promueve una educación más inclusiva y eficiente, a la vez que fomenta el crecimiento económico a través de la innovación tecnológica y fortalece las instituciones educativas mediante la transparencia. Así, se convierte en una plataforma que ayuda a avanzar hacia un futuro sostenible, equitativo y justo, mejorando la calidad de vida de las personas.