Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA DE SEGOVIA

Grado en Ingeniería Informática de Servicios y Aplicaciones

Desarrollo de un sistema para la autoevaluación de modelos conceptuales de datos utilizando técnicas de visión computacional

---

Alumno: Clara Gándara González

Tutores: Miguel Ángel Martínez Prieto
Jorge Silvestre Vilches

Fecha: 29 de junio de 2024

Desarrollo de un sistema para la autoevaluación de modelos conceptuales de datos utilizando técnicas de visión computacional

Clara Gándara González

29 de junio de 2024

A mi abuela por siempre confiar en mí de manera incondicional.
A mi abuelo por ser mi confidente.
A mis padres por no soltar nunca mi mano.
A mi hermana por ser mi cómplice.
A mi sobrino por alegrar mis días con su sonrisa.

¡Qué extraño es todo hoy!
¡Y ayer sucedía todo como siempre!
¿Habré cambiado durante la noche?
Pero si no soy la misma,
el asunto siguiente es ¿quién soy?
¡Ay, ese es el gran misterio!

"Alicia en el País de las Maravillas"

La educación no es preparación para la vida;
la educación es la vida en sí misma.

John Dewey

Agradecimientos

Me gustaría agradecer a todas aquellas personas que me han apoyado no solo en la realización de este TFG, sino también durante todos los años de la carrera.

Quiero empezar agradeciendo a mis tutores Miguel Ángel Martínez Prieto y Jorge Silvestre Vilches por darme la oportunidad de desarrollar este proyecto de investigación junto a una beca de colaboración y ayudarme con cada problema que he ido teniendo en su desarrollo. Pero sobretodo les estoy muy agradecida por todo lo que me han ayudado a lo largo de la carrera.

También quiero agradecer a mis padres por apoyarme económicamente, por confiar siempre en que acabaría la carrera, por apoyarme en todas las decisiones que he ido tomando a lo largo de la misma, por traerme tappers en época de exámenes, etc. A mi hermana porque aunque estemos a más de 8.000km siempre que la he necesitado ha estado para mí y por darme consejos de hermana mayor cada vez que los he necesitado. Y a mi sobrino por alegrarme los días por muy duros que fuesen.

Además, me gustaría dar las gracias a una persona muy especial, mi abuela. Gracias por siempre confiar de manera incondicional en mí, por recordarme una y otra vez lo orgullosa que estabas de mí, por escucharme hablar de las asignaturas de la carrera sin entender nada de lo que te decía. Y sobretodo gracias por darme fuerzas con tus palabras para continuar en los momentos más difíciles. Me hubiera encantado enseñarte la aplicación y contarte que por fin he acabado la carrera, pero estoy segura que en la defensa estarás presente de alguna forma. Y no me podía faltar, agradecer a mi abuelo por siempre ayudarme desde el cielo.

Asimismo, quiero agradecer a mis amigos por aguantarme en mis peores momentos, por permitirme desahogarme con ellos y por escuchar mis numerosas quejas. También, por esas salidas, quedadas y llamadas que hacían que todos los problemas de la universidad desaparecieran por un momento.

Por último, me gustaría dar las gracias a dos personas que me ha ayudado mucho en los últimos años: Gloria Díaz Pereira y Mario Santos. Gracias por recordarme en numerosas ocasiones que era capaz de sacarme la carrera y por darme herramientas para mejorar tanto a nivel personal como a nivel académico. Sin vuestro apoyo este camino hubiera sido mucho más complicado.

Resumen

Los estudiantes suelen tener dificultades para abstraer los requisitos de información en términos de entidades, relaciones, atributos y roles. Por ello, el aprendizaje del modelo entidad-relación suele ser todo un reto en las aulas.

En este trabajo de fin de grado se plantea la creación de una herramienta de asistencia al aprendizaje de diseño conceptual de datos, que permita al estudiante autoevaluar su aprendizaje de forma sencilla.

El proyecto se ha abordado como un *workflow* en 4 fases: procesamiento de los componentes, procesamiento de los contenidos, conectividad de los componentes y contenidos y validación. Para ello, se han utilizado técnicas de visión computacional y de reconocimiento óptico de caracteres, llegando a detectar y conectar más del 70 % de los componentes del modelo. Como resultado se ha obtenido una aplicación mediante la cual el estudiante sube una imagen con un modelo entidad-relación, valida los componentes identificados y el texto detectado, generándole un *feddback*.

Palabras claves: base de datos, modelo entidad-relación, aprendizaje automático, reconocimiento óptico de caracteres.

Abstract

Students often have difficulty abstracting information requirements in terms of entities, relationships, attributes and roles. Therefore, learning the entity-relationship model is often a challenge in the classroom.

In this final degree project we propose the creation of a tool to assist the learning of conceptual data design, which allows students to self-assess their learning in a simple way.

The project has been approached as a workflow in 4 phases: component processing, content processing, connectivity of components and content, and validation. For this purpose, computer vision and optical character recognition techniques have been used to detect and connect more than 70 % of the components of the model. As a result, an application has been obtained through which the student uploads an image with an entity-relationship model, validates the identified components and the detected text, generating a feddback to the student.

Keywords: database, entity-relationship model, machine learning, optical character recognition.

Índice general

Índice de figuras

Índice de tablas

Parte I

Descripción del proyecto

Capítulo 1

Introducción

Desde mediados del siglo XX hasta la actualidad, nuestra sociedad ha sido testigo de como los avances tecnológicos han cambiado por completo nuestra forma de vida [22]. Hasta el punto de encontrarnos rodeados de aplicaciones y sistemas que están compuestos por software. Detrás de la gran mayoría de los proyectos de software existe una base de dados que se encarga de gestionar y servir la información necesaria para satisfacer los requisitos de usuario del proyecto.

Según Jordi Casas Roma "el diseño conceptual es la segunda etapa en el proceso de diseño e implementación de una base de datos y tiene como objetivo crear un esquema conceptual de alto nivel, que describa los requisitos de información independientemente de la tecnología que se utilice para abordarlos" [47]. "Así, un esquema conceptual es una descripción concisa de los requisitos de información e incluye descripciones detalladas de las entidades que están involucradas, las relaciones entre estas entidades y las restricciones de integridad que tienen." Entre los esquemas conceptuales más utilizados encontramos el modelo entidad-relación (ER), un diagrama que describe los requisitos de información en términos de entidades, relaciones y atributos. El modelo entidad-relación se utiliza de forma generalizada en entornos de desarrollo software por lo que su aprendizaje es esencial para los estudiantes de Ingeniería Informática y suele abordarse en asignaturas introductorias de las bases de datos.

A pesar de su simplicidad, el aprendizaje del modelo ER suele ser desafiante para los estudiantes, sobre todo por la dificultad que les supone abstraer los requisitos de información en términos de entidades, relaciones y atributos. Por ello, es habitual adoptar un aprendizaje basado en el empirismo [10], de tal forma que los estudiantes aprenden a modelar en base a su experiencia modelando. Sirva como ejemplo el enfoque de la asignatura Sistemas de Bases de Datos [70] impartida en la titulación de Grado en Ingeniería Informática de Servicios y Aplicaciones, en el que el aprendizaje se aborda de forma incremental, resolviendo problemas de mayor complejidad a medida que avanza la asignatura.

Sin embargo, el éxito de este tipo de estrategias de aprendizaje está supeditado a la disponibilidad de enunciados de problemas y a la capacidad del profesor para evaluarlos y retroalimentar a los estudiantes en consecuencia. Actualmente esto no es viable, dado el tamaño de los grupos de estudiantes que cursan una asignatura universitaria y la capacidad del profesor. Por ello, en este proyecto planteamos la creación de un sistema informático orientado a la autoevaluación de modelos ER, tomando como referencia la especificación de requisitos de información (en forma de "problema") y una solución de referencia, proporcionada por el profesor. La aplicación ofrece a los estudiantes la posibilidad de autoevaluar su conocimiento, proporcionando para ello el modelo ER que resuelve el supuesto, desde su punto de vista, y recibiendo el feedback apropiado

respecto a la solución del profesor. Cabe destacar que el sistema estará capacitado para trabajar con modelos ER manuscritos, que es la forma en la que los estudiantes suelen crear sus diseños conceptuales durante su aprendizaje. De esta forma, conseguimos que los estudiantes pueden realizar una mayor cantidad de ejercicios, recibiendo un feedback inmediato. Uno de los retos que tenemos a la hora de crear esta aplicación, es que las imágenes que van a subir los usuarios van a ser manuscritas, tal cual se ilustra en la imagen 1.1.

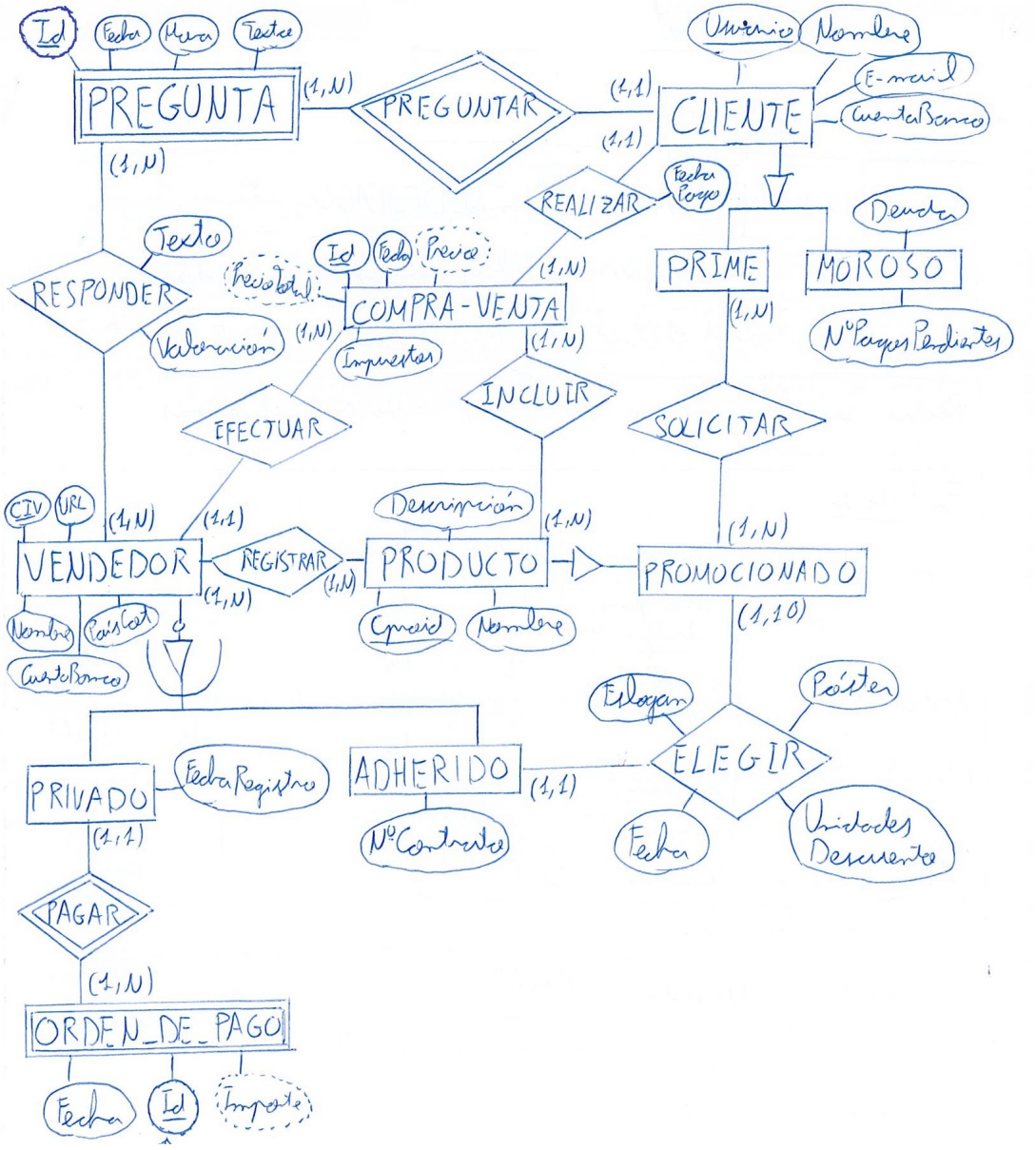


Figura 1.1: Ejemplo de uno de los diagramas ER que forma parte de la colección de datos creada en este proyecto.

Para realizar esta aplicación hemos hecho uso de la inteligencia artificial (IA), sobre todo de la visión computacional para la detección de las diferentes formas geométricas que conforman

los diagramas ER (rectángulos, óvalos, líneas, rombos, triángulos,...). También hemos empleado técnicas de reconocimiento de caracteres para identificar el contenido de estas figuras. Todo ello lo integramos en el flujo de datos que se presenta en la figura 1.2

1.1. Planteamiento del problema

El planteamiento del problema [27] proporciona una descripción de alto nivel del problema que se ha abordado en el proyecto, destacando las diferencias entre la situación ideal y la realidad actual. Además, explora las posibles consecuencias adversas de no abordar el problema y concluye al presentar una propuesta de solución que actúa como base para el desarrollo del proyecto.

- IDEAL: El conocimiento se fija mejor si se pone en práctica [10], por eso es necesario que los alumnos puedan evaluar su conocimiento de manera que puedan obtener, de forma inmediata, una evaluación detallada del mismo, automatizando en la medida de lo posible el proceso para facilitar la inmediatez de esta retroalimentación, y conseguir que esta carga de trabajo no descanse exclusivamente en el profesor.

- REALIDAD: Actualmente, el profesor recibe la solución del estudiante, corrige el modelo ER y le proporciona un feedback, que en algunos casos puede llegar retrasado dada la carga de trabajo que la tarea de evaluación puede suponerle al profesor.

- CONSECUENCIAS: Como se ha mencionado anteriormente este proceso de corrección requiere de una gran cantidad de tiempo del profesor, lo que conlleva a que no pueda corregir muchos ejercicios de los estudiantes. Así como que el estudiante no pueda recibir el feedback a su ejercicio de forma inmediata, lo cual puede tener incidencias negativas en su aprendizaje.

- PROPUESTA: En este proyecto implementamos un sistema informático capaz de evaluar automáticamente las soluciones de los estudiantes (respecto a una solución de referencia planteada por el profesor) y proporcionarles feedback inmediato.

1.2. Objetivos del proyecto

Este proyecto se centra en la interpretación y evaluación de modelos ER y la generación de un análisis comparativo inmediato. Siguiendo esta línea de trabajo hemos identificado los objetivos expuestos en la tabla 1.1 .

ID-OBJ	Objetivos
OBJ-01	Diseñar e implementar un proceso de interpretación de los contenidos de modelos Entidad-Relación creados por los estudiantes, a partir de una imagen digitalizada del documento manuscrito.
OBJ-02	Diseñar e implementar un mecanismo de corrección automatizado que sea capaz de evaluar los modelos Entidad-Relación generados por los estudiantes respecto a una solución de referencia.
OBJ-03	Diseñar e implementar un mecanismo de generación automatizada de feedback basado en la especificación del producto de aprendizaje planteada en UVAGILE [71].

Tabla 1.1: Objetivos.

1.2.1. Restricciones

Son las limitaciones en términos de disponibilidad de recursos o factores ajenos al proyecto, que puedan condicionar tanto su planificación como su desarrollo.

ID-REST	Restricciones
REST-01	El proyecto tiene una carga de trabajo de 12 ECTS.
REST-02	El programa realizará una comparación exacta del modelo ER del profesor y del estudiante.

Tabla 1.2: Restricciones de negocio.

1.3. Descripción general del Workflow

En esta sección, se presenta de forma general el flujo de datos sobre el que se construirá nuestro sistema informático. Este flujo consta de 6 etapas, que se ilustran en la figura 1.2.

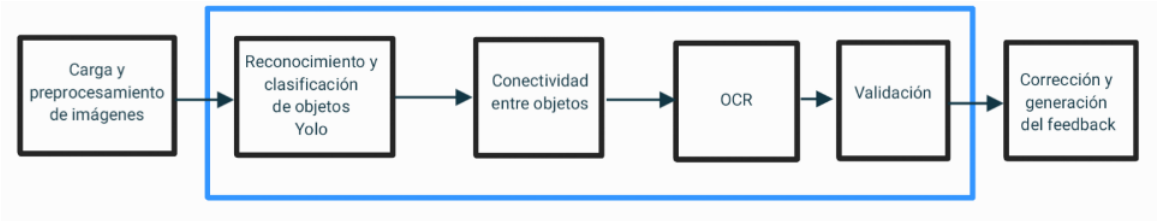


Figura 1.2: Workflow de la aplicación.

A continuación, se explica brevemente cada una de estas etapas:

1. **Carga y preprocesamiento de imágenes:** En este paso el usuario subirá una imagen a la aplicación y esta la cargará y guardará para utilizarla en los siguientes pasos.

2. **Reconocimiento y clasificación de objetos:** Aquí se detectan y clasifican los diferentes componentes de la imagen previamente subida (entidades, relaciones, atributos, roles y líneas), utilizando para ello visión computacional, concretamente YOLO [35].

3. **Conectividad de los componentes:** En este punto definimos un conjunto de reglas para conectar los diferentes componentes identificados en la etapa anterior, y con ello incorporar a la imagen la semántica de los diferentes conceptos utilizados en un modelo ER.

4. **Reconocimiento de caracteres (OCR):** Aquí se obtienen las diferentes piezas de texto utilizadas en el modelo y se vinculan a las componentes identificadas en el paso previo.

5. **Validación del usuario:** En esta fase, se proporciona al usuario (estudiante o profesor) una interfaz web para que valide que los componentes se han identificado y conectado correctamente.

6. **Corrección y generación del feedback:** En este paso se compara la solución del estudiante con la proporcionada previamente por el profesor y se le muestra al estudiante el feedback generado a través de la aplicación.

La implementación de este flujo de datos anterior se llevará a cabo ulizando el lenguaje de programación Python (y librerías como ultralytics, numpy, cv2, math, json o shutil) mientras que los componentes visuales se implementarán con Quasar y Vue.

1.4. Estructura de la memoria

La memoria de este proyecto se divide en los siguientes capítulos:

- **Capítulo 1: Introducción.** Motiva la importancia del diseño conceptual de datos y la necesidad de proporcionarle a los estudiantes herramientas que les asistan durante el proceso de aprendizaje. Además, plantea la descripción del problema y sus objetivos, así como una visión general de la solución que se construye en este proyecto.

- **Capítulo 2: Planificación.** Describe la metodología utilizada para llevar a cabo el proyecto, incluyendo la planificación, los presupuestos, sus balances temporal y económico.

- **Capítulo 3: Antecedentes.** Explica los conocimientos básicos necesarios para comprender el cómo y el porqué se lleva a cabo este proyecto, partiendo de una descripción precisa de qué es y qué rol juega el diseño conceptual de datos en el panorama tecnológico actual. Asimismo, se plantea una revisión general del estado del arte relacionado con la evaluación automática de modelos técnicos y se revisa el contexto científico-técnico en el que se desarrolla el proyecto., incluyendo los fundamentos de aprendizaje automático y de reconocimiento de objetos.

- **Capítulo 4: Descripción de la propuesta.** Presenta el análisis y el diseño del producto software a construir en este proyecto, así como la creación de la colección de datos necesaria para entrenar y evaluar los modelos de aprendizaje automático necesarios para implementar el flujo de datos presentado previamente.

- **Capítulo 5: Procesamiento de los componentes del modelo ER.** Explica cómo se identifican los componentes propios del modelo ER y se describen las diferentes reglas utilizadas para conectarlos. Asimismo, en este capítulo se evalúa la calidad de la propuesta realizada, utilizando métricas estandarizadas en el área de trabajo.
- **Capítulo 6: Procesamiento de los contenidos textuales.** Explica cómo se ha realizado el reconocimiento óptico de caracteres (OCR), cómo se ha conectado el texto con los diferentes componentes del modelo ER y se expone un análisis de la efectividad de OCR para abordar las necesidades de nuestro proyecto.
- **Capítulo 7: Validación del usuario.** Presenta el subsistema de validación del modelo reconocido, desde el que el usuario (estudiante o profesor) podrá corregir los defectos identificados o añadir aquellos elementos (visuales o de texto) que no se hayan reconocido.
- **Capítulo 8: Evaluación del modelo y generación de feedback.** Presenta el subsistema responsable de retroalimentar al estudiante, este muestra por pantalla una tabla con el feedback generado.
- **Capítulo 9: Conclusiones y trabajo a mayores.** Expone las conclusiones del proyecto y describe las líneas abiertas para su futura utilización para propósitos de aprendizaje en el ámbito de una asignatura universitaria.

Capítulo 2

Planificación

En este capítulo se presenta la metodología ASAP *(Agile Student Academic Projects)*, la cual se ha utilizado para llevar a cabo el proyecto. Así como, se explica como se ha planificado de forma incremental el proyecto utilizando la metodología anterior y se exponen los presupuestos y el balance temporal del proyecto.

En la sección 2.1 encontramos la metodología utilizada para llevar a cabo este proyecto, en la sección 2.2 la planificación previa al inicio del proyecto, en la sección los presupuestos del proyecto 2.3 y por último en la sección sección 2.4 el balance temporal y económico .

2.1. Metodología de trabajo

En este proyecto hemos optado por ASAP *(Agile Student Academic Projects)* [34] como metodología de ciclo de vida, dado su enfoque en obtener productos de aprendizaje de alta calidad en trabajos de fin de estudios. ASAP es una metodología que adopta prácticas de los marcos de trabajo ágiles [32] utilizadas comúnmente en el ámbito profesional, con el fin de abordar los objetivos de aprendizaje establecidos para el Trabajo Fin de Grado (TFG) en términos de planificación del proyecto, consolidación de sus antecedentes, desarrollo, aceptación del producto, y comunicación tanto oral como escrita del trabajo realizado. En este sentido, ASAP propone un conjunto de eventos, roles y artefactos destinados a establecer un ritmo de trabajo sostenido y constante durante todo el TFG. Además, busca facilitar la interacción periódica entre el estudiante y los tutores del proyecto, así como con otros estudiantes que están llevando a cabo otros TFGs en el mismo período de tiempo. Esto fomenta un entorno colaborativo que enriquece la experiencia de aprendizaje y permite compartir conocimientos entre los participantes. A continuación, se presentan los objetivos de un TFG según ASAp y los roles, eventos y artefactos propuestos en la metodología.

2.1.1. Objetivos

ASAP [34] presenta cinco objetivos de aprendizaje mediante los cuales pretende garantizar que los estudiantes obtengan una descripción clara del proyecto que van a desarrollar. A continuación, se detallan cada uno de estos objetivos:

- **Proyecto**: El principal objetivo del TFG es crear un proyecto siguiendo la metodología ASAP. Para ello, el desarrollo de dicho proyecto ha de hacerse de manera incremental e

iterativa, ya que la metodología ASAP es una metodología AGILE. Esta forma de trabajar ayuda a que el estudiante tenga una carga constante de trabajo durante todo el periodo de realización del TFG y minimiza de esta manera posibles retrasos en la presentación del proyecto.

- **Antecedentes:** Este objetivo busca que el estudiante se familiarice antes de empezar a implementar el proyecto con el área en el que lo va a desarrollarlo. Para ello, el estudiante buscará con la ayuda de sus tutores artículos relacionados tanto con las tecnologías que va a utilizar en la realización del proyecto, como artículos que desarrollen proyectos similares.

- **Desarrollo:** Este objetivo se enfoca en cumplir cada uno de los propósitos establecidos en el objetivo del proyecto, para que de esta manera se desarrolle de forma incremental e iterativa el producto.

- **Aceptación:** Este objetivo se centra en asegurarse de que el proyecto construido cumple todos los objetivos del proyecto.

- **Comunicación:** Este objetivo incluye dos subobjetivos: la creación de la memoria técnica del proyecto y su defensa. En la memoria técnica, se registra una breve descripción de la metodología ASAP, la planificación temporal del proyecto, los presupuestos, el balance temporal y económico, los resultados obtenidos de los objetivos antecedentes, desarrollo y aceptación previamente explicados, así como, las conclusiones del proyecto, el trabajo a futuro y las referencias bibliográficas utilizadas. Mientras que en la presentación oral el estudiante expone los puntos más destacados de cada uno de ellos explicados en profundidad en la memoria. Tanto la realización de la memoria como la defensa oral se realizan de manera iterativa a lo largo del proyecto, facilitando la obtención de retroalimentación por parte de los tutores y de otros estudiantes que realizan el TFG al mismo tiempo.

2.1.2. Roles

A continuación, se explican los 4 roles involucrados en la realización de un TFG utilizando la metodología ASAP [34]:

- **Estudiante:** Es el actor principal del proyecto y es responsable de identificar y ejecutar las tareas a realizar, desarrollar la planificación temporal, asegurarse de mejorar continuamente el producto, desarrollar e implementar el proyecto, así como mantener una comunicación activa con los tutores del proyecto.

- **Tutor:** Un TFG desarrollado con ASAP puede ser guiado por uno o más profesores, los cuales toman el papel de tutor o tutores. Este actor junto con el estudiante son los más activos del proyecto. El tutor debe hacer un seguimiento continuo del avance del estudiante, asegurándose de esta manera de que el estudiante va cumpliendo con los objetivos establecidos. Este también es participe junto al estudiante de la definición de los objetivos de cada *sprint*, así como le proporciona una retroalimentación al estudiante al finalizar cada *sprint*.

- **Comunidad:** Esta formada por todas las personas que participan en la realización del proyecto: profesores, estudiantes, expertos, etc. La comunidad participa al completo en las

reuniones de comunicación, en las que el estudiante expone su defensa del proyecto y el resto les proporcionan una retroalimentación.

- **Tribunal:** Esta constituida por tres profesores ajenos a la comunidad, que estarán presentes únicamente en el acto de defensa del TFG. Su función es evaluar el proyecto del estudiante.

2.1.3. Eventos

ASAP [34] cuenta con 5 eventos que se explican a continuación:

- **Sprint:** La realización del TFG se encuentra dividida en periodos de aproximadamente 4-5 semanas de duración, los cuales se denominan *sprints*. En cada uno de estos *sprints* el estudiante trabaja en objetivos específicos definidos por él y sus tutores al inicio de cada *sprint* en una reunión de inicio. La realización del proyecto en *sprints* facilita que su desarrollo se haga de manera incremental e iterativa. Al finalizar cada *sprint* el estudiante debe entregar la memoria del proyecto y hacer una defensa, de esta manera se garantiza que el mismo obtenga una retroalimentación de su trabajo de forma continua.

- **Reunión de Inicio:** Este evento marca el comienzo de cada *sprint* y su objetivo es definir los objetivos del *sprint* y planificar las tareas a realizar para conseguir dichos objetivos. En este evento participa tanto el estudiante como los tutores.

- **Reunión de Sincronización:** Este evento se realiza una vez a la semana, siempre que se pueda a la misma hora y día de la semana. Su duración es aproximadamente de 15 minutos y participan tanto el estudiante como los tutores. En ella, el estudiante le contará a los tutores los avances realizados durante la semana, así como los puntos de bloqueo. Los tutores por su parte buscan soluciones a dichos problemas y si es necesario ajustan la planificación del *sprint* sin modificar los objetivos establecidos en la reunión de inicio.

- **Comunicación de Progresos:** En este evento el estudiante presenta el trabajo realizado hasta el momento tanto a sus tutores como al resto de la comunidad. Al finalizar su exposición, la comunidad le proporciona una retroalimentación. Tanto la duración de la presentación del estudiante como la retroalimentación de la comunidad tiene una duración máxima de 30 minutos. La comunicación de progresos se realiza el último día del *sprint*.

- **Retrospectiva:** En este evento participa toda la comunidad y se realiza justo al finalizar la comunicación de progresos. Tiene una duración máxima de 30 minutos. En esta reunión se presentan de forma anónima todos los aspectos positivos y negativos del *sprint* y posteriormente se proponen diferentes soluciones para mejorar los aspectos negativos.

2.1.4. Artefactos

La metodología ASAP [34] cuenta con 2 artefactos. A continuación, se explica cada uno de ellos:

- **Incremento:** Representa el estado en el que se encuentra el proyecto al finalizar cada *sprint* y esta formado por los resultados obtenidos al terminar cada tarea del mismo. Al finalizar cada *sprint* se debe almacenar el incremento en el entorno de trabajo, para que los tutores puedan revisarlo y proporcionar al estudiante una retroalimentación.

- **Retroalimentación:** Es la generación de un feedback al estudiante sobre el incremento entregado al final de cada *sprint*. Existen dos tipos:
  - **Retroalimentación al concluir la comunicación de progresos:** Una vez el estudiante ha finalizado la defensa de su proyecto, la comunidad le proporcionará críticas constructivas. A partir de las cuales, el estudiante podrá realizar mejoras en los siguientes *sprints*. Al concluir este evento, el profesor subirá al entorno de trabajo un archivo con todas las mejoras sugeridas.
  - **Retroalimentación de la memoria del proyecto:** Una vez los tutores han revisado la memoria técnica del proyecto, subirán al entorno de trabajo un fichero con todas las mejoras y errores identificados en ella. La entrega de esta retroalimentación se llevará a cabo en la semana posterior a la finalización del *sprint*.

2.1.5. Entorno de trabajo

- **Espacio de trabajo compartido:** Se ha creado un canal de teams con dos secciones:
  - **Canal privado:** Este es accesible exclusivamente para el estudiante y los tutores de su TFG, garantizando de esta manera la confidencialidad de las discusiones y de los materiales específicos del proyecto. Dentro de este canal, encontramos el acceso al tablero de trabajo que se explica en el siguiente punto, y un enlace a una carpeta llamada archivos, donde se almacena todos los materiales del TFG en diferentes directorios: documentación, incrementos, personal, recursos y referencias.
  - **Canal público:** Toda la comunidad tiene acceso a este canal. En este se comparten los horarios de las reuniones generales, así como, archivos que pueden ser útiles para toda la comunidad.
- **Tablero del proyecto:** Es una herramienta que se utiliza para organizar y visualizar las distintas tareas del proyecto. La figura 2.1 muestra el tablero de trabajo de nuestro proyecto, en el que se pueden apreciar las 6 columnas que lo forman:
  - **Backlog de proyecto:** Contiene una lista de los objetivos principales del proyecto que aún no han sido abordados en ningunos de los *sprints* previos ni en el actual.
  - **Objetivo del *sprint*:** Incluye una lista de etiquetas que forman los objetivos seleccionados para el *sprint* actual.
  - **ToDo:** Tareas que se deben llevar a cabo en lo largo del *sprint* actual, pero que aún no se han empezado.
  - **Doing:** Contiene el conjunto de tareas que se están llevando a cabo actualmente.
  - **Blocked:** Formado por las tareas que se han empezado a realizar, pero que por algún motivo no se han podido completar.
  - **Done:** Contiene el conjunto de tareas terminadas.
- **Cuaderno de trabajo:** Es un documento de excel que recopila todas las actividades desarrolladas a lo largo del proyecto. Este cuaderno se compone de 6 columnas que detallan la fecha de inicio de cada tarea, el *sprint* en el que se realizaron, el tipo de tarea (dinámica de trabajo, lectura de documentación, programación, redacción de memoria y otros), el nombre

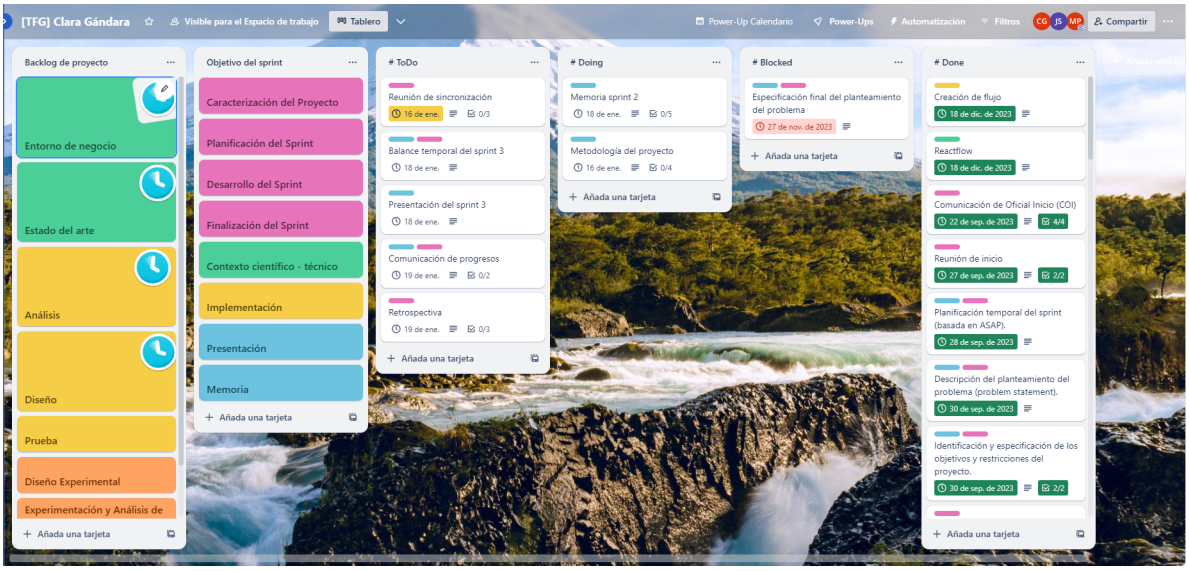


Figura 2.1: Tablero del proyecto.

de la tarea, el tiempo invertido en su realización y una descripción breve de la tarea llevada a cabo. El propósito principal del cuaderno de trabajo es mantener un registro detallado del trabajo realizado en el proyecto, que permita analizar la planificación del *sprint* y el esfuerzo invertido por el estudiante.

Alumno	Clara Gándara González	TFG	Desarrollo de un sistema para la autoevaluación de modelos conceptuales de datos utilizando técnicas de visión computacional

Fecha	Sprint	Actividad		Tiempo	Descripción
		Tipo	Tarea		
25/09/2023	Sprint #1	Dinámica de Trabajo	Preparar Incremento	1:00:00	Reunión de inicio del primer sprint.
25/09/2023	Sprint #1	Dinámica de Trabajo	Preparar Incremento	1:06.37	Descripción de las tareas del primer sprint.
26/09/2023	Sprint #1	Dinámica de Trabajo	Preparar Incremento	0:27:24	Planificación de las tareas del primer sprint.
26/09/2023	Sprint #1	Lectura de Documentación		0:40:00	Artículo 1.
27/09/2023	Sprint #1	Redacción de Memoria		0:15:36	Problem statement.
27/09/2023	Sprint #1	Redacción de Memoria		0:10:07	Objetivos y restricciones.
27/09/2023	Sprint #1	Lectura de Documentación		0:35:59	Leer artículos.
28/09/2023	Sprint #1	Lectura de Documentación		1:30:28	Leer artículos.
30/09/2023	Sprint #1	Lectura de Documentación		3:31:34	Leer artículos.
30/09/2023	Sprint #1	Redacción de Memoria		0:55:50	Descripción del modelo ER.
01/10/2023	Sprint #1	Redacción de Memoria		0:36:02	Diseño conceptual de BD y el ciclo de su vida en los SW.
02/10/2023	Sprint #1	Dinámica de Trabajo	Reunión Semanal	0:15:00	
02/10/2023	Sprint #1	Lectura de Documentación		0:16:00	Introducción general al reconocimiento de objetos
02/10/2023	Sprint #1	Redacción de Memoria		0:57:07	Redaccción de memoria.
04/10/2023	Sprint #1	Redacción de Memoria		1:10:33	Redaccción de memoria.
04/10/2023	Sprint #1	Lectura de Documentación		0:24:24	OpenCV.
05/10/2023	Sprint #1	Lectura de Documentación		0:28:29	OpenCV.
05/10/2023	Sprint #1	Redacción de Memoria		0:13:23	Contexto científico-técnico.
08/10/2023	Sprint #1	Redacción de Memoria		1:29:13	OpenCV.
08/10/2023	Sprint #1	Redacción de Memoria		0:55:29	Estado del arte: redacción del primer artículo.
09/10/2023	Sprint #1	Dinámica de Trabajo	Reunión Semanal	0:15:00	

Figura 2.2: Cuaderno de trabajo.

2.2. Planificación temporal

Como se ha indicado en el punto anterior este TFG se ha llevado a cabo usando la metodología ASAP. Por ello, su planificación temporal se ha dividido inicialmente en 5 *sprints*, con una carga de trabajo de 90 horas para cada *sprint*. A continuación, se muestra la duración de cada uno de los *sprints*:

- *Sprint* 1: 25/09/23 - 23/10/23
- *Sprint* 2: 23/10/23 - 01/12/23
- *Sprint* 3: 03/12/23 - 19/01/24
- *Sprint* 4: 19/01/24 - 15/03/24
- *Sprint* 5: 16/03/24 - 08/05/24

La planificación temporal de este proyecto se ha realizado de manera iterativa, organizándose *sprint* a *sprint*. En la reunión de inicio de cada *sprint*, se establecen los objetivos a alcanzar durante el mismo. Una vez finalizada la reunión, el estudiante define las tareas necesarias y establece la fecha de finalización de cada una de ellas para alcanzar los objetivos previamente definidos. En las siguientes subsecciones se presentan los objetivos y tareas de los cinco *sprints* planificados inicialmente, junto con un sexto *sprint* que se ha añadido posteriormente para compensar los retrasos acumulados en los *sprints* anteriores.

Las horas empleadas en un TFG de 12 créditos ECTS oscila entre 300 y 360 horas. Dado que este proyecto se ha llevado a cabo junto con una beca de colaboración, el número de horas empleadas se incrementa, situándose entre 420 y 480 horas. Por ello, para la planificación de este proyecto hemos considerado un total de 450 horas.

Para ilustrar la planificación temporal de cada *sprint*, se ha creado un diagrama de Gantt. Esta herramienta gráfica muestra la duración a lo largo del tiempo de las tareas asociadas a cada objetivo. Su finalidad principal es proporcionar una visión clara y comprensible del progreso del proyecto. El diagrama de Gantt se utiliza en la planificación para visualizar las fechas de inicio y finalización de las diferentes actividades. Cada tarea se representa como una barra horizontal verde en el gráfico, cuya longitud indica la duración de la tarea. Esto facilita la identificación de solapamientos, dependencias entre tareas y posibles retrasos, permitiendo una gestión más eficiente del proyecto.

2.2.1. *Sprint* #1

El primer *sprint* se ha llevado a cabo desde el 25 de septiembre de 2023 hasta el 23 de octubre del 2023. A continuación, se exponen las diferentes historias de proyecto junto con las tareas que se han realizado.

- **H1.** Caracterización del Proyecto
  - **T1.1** Descripción del planteamiento del problema.
  - **T1.2** Identificación y especificación de los objetivos y restricciones del proyecto.
- **H2.** Planificación del *sprint*

  - **T2.1** Definición del objetivo del *sprint*.
  - **T2.2** Planificación de tareas a realizar durante el *sprint*.
  - **T2.3** Balance temporal del *sprint*.
- **H3.** Desarrollo del *sprint*
  - **T3.1** Reuniones de sincronización semanales.
  - **T3.2** Actualización continua del tablero.
  - **T3.3** Reunión de comunicación.
- **H4.** Entorno de negocio
  - **T4.1** Caracterización del diseño conceptual de bases de datos y su relevancia en el ciclo de vida de los sistemas software.
  - **T4.2** Descripción del Modelo Entidad-Relación (concepto, notaciones...) y su propósito en el ámbito del diseño conceptual de bases de datos.
- **H5.** Estado del arte
  - **T5.1** Búsqueda y selección de trabajo relacionado con los objetivos del proyecto.
  - **T5.2** Propuesta de las dimensiones comparativas del trabajo relacionado en el contexto del proyecto.
  - **T5.3** Resumen individual de los diferentes trabajos relacionados, atendiendo a las dimensiones comparativas propuestas.
  - **T5.4** Análisis comparativo de los trabajos seleccionados y conclusiones sobre el estado del arte estududiado.
- **H6.** Contexto científico-técnico
  - **T6.1** Introducción general al reconocimiento de objetos.
  - **T6.2** Introducción general al reconocimiento de caracteres.
  - **T6.3** Caracterización de librerías disponibles en python.
- **H7.** Análisis
  - **T7.1** Modelado y especificación de los requisitos de usuario.
  - **T7.2** Modelado y especificación de los requisitos funcionales.
  - **T7.3** Modelado y especificación de los requisitos de información.
- **H8.** Diseño
  - **T8.1** Arquitectura lógica del sistema.
  - **T8.2** Diseño lógico de datos.
- **H9.** Implementación
  - **T9.1** Desarrollo del componente de reconocimiento de objetos.

- **T9.2** Desarrollo del componente de reconocimiento de caracteres.

- **H10.** Presentación
  - **T10.1** Elaboración de la presentación.

- **H11.** Memoria
  - **T11.1.** Redacción (parcial) del capítulo Introducción.
  - **T11.2** Redacción (parcial) del capítulo Planificación.
  - **T11.3** Redacción (parcial) del capítulo Antecedetes.
  - **T11.4** Redacción (parcial) del capítulo Descripción.

En la imagen 2.3 se muestra el diagrama de Gantt del primer *sprint*.

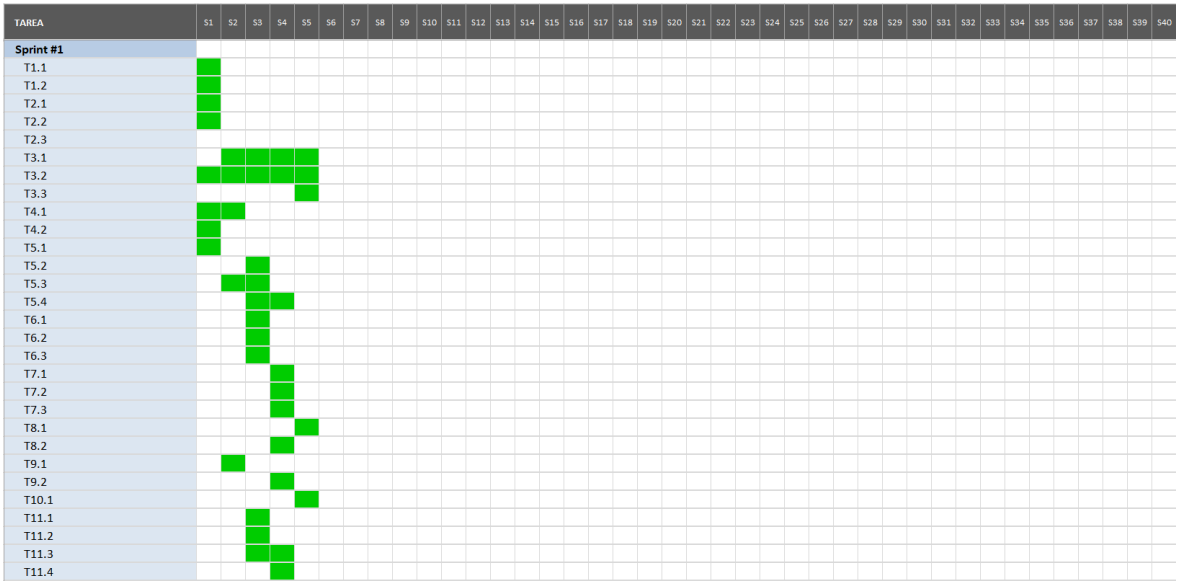


Figura 2.3: Diagrama de Gantt del *sprint* 1.

2.2.2. *Sprint* #2

El segundo *sprint* se ha llevado a cabo desde el 23 de octubre de 2023 hasta el 1 de diciembre del 2023. A continuación, se exponen las diferentes historias de proyecto junto con las tareas que se han realizado.

- **H1.** Caracterización del Proyecto
  - **T1.2** Identificación y especificación final de los objetivos y restricciones del proyecto.

- **H2.** Planificación del *sprint*
  - **T2.1** Definición del objetivo del *sprint*.
  - **T2.2** Planificación de tareas a realizar durante el *sprint*.

  - **T2.3** Balance temporal del *sprint*.

- **H3.** Desarrollo del *sprint*

  - **T3.1** Reuniones de sincronización semanales.
  - **T3.2** Actualización continua del tablero.
  - **T3.3** Reunión de comunicación.

- **H4.** Entorno de negocio

  - **T4.3** Corrección de los errores cometidos en el *sprint* previo en este apartado.

- **H5.** Estado del arte

  - **T5.3** Resumen individual de los diferentes trabajos relacionados, atendiendo a las dimensiones comparativas propuestas.
  - **T5.5** Análisis comparativo de los trabajos seleccionados y conclusiones sobre el estado del arte estududiado.

- **H6.** Contexto científico-técnico

  - **T6.4** Introducción general al reconocimiento de objetos.
  - **T6.5** Descripción general del transfer learning, como técnica de referencia para el reconocimiento de objetos
  - **T6.6** Descripción de la arquitectura YOLO para la detección de objetos.
  - **T6.7**Descripción de las técnicas OCR para el reconocimiento de caracteres.

- **H9.** Implementación

  - **T9.1** Desarrollo del componente de reconocimiento de objetos.
  - **T9.2** Desarrollo del componente de reconocimiento de caracteres.

- **H10.** Presentación

  - **T10.1** Elaboración de la presentación.

- **H11.** Memoria

  - **T11.1.** Redacción (parcial) del capítulo Introducción.
  - **T11.2** Redacción (parcial) del capítulo Planificación.
  - **T11.3** Redacción (parcial) del capítulo Antecedetes.
  - **T11.4** Redacción (parcial) del capítulo Descripción.

La imagen 2.4 presenta el diagrama de Gantt del segundo *sprint*.

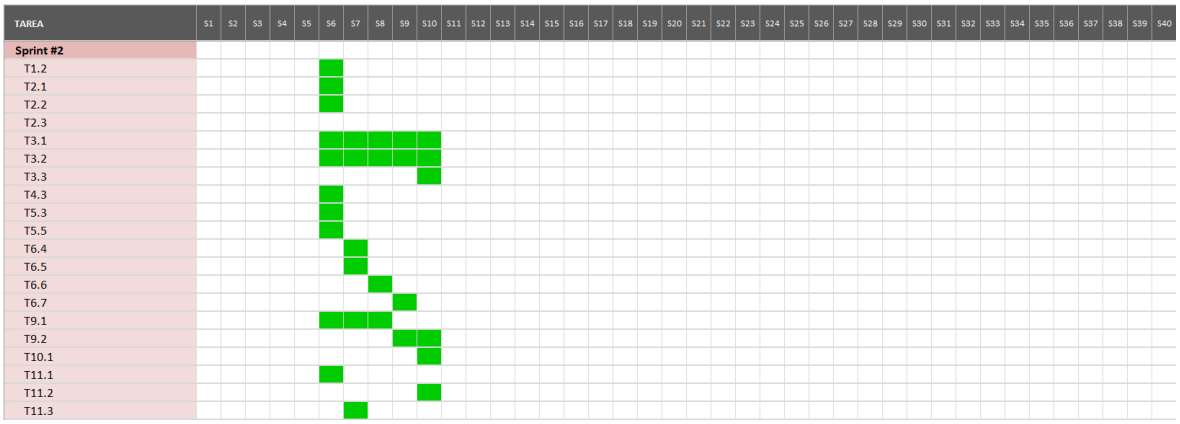


Figura 2.4: Diagrama de Gantt del *sprint* 2.

2.2.3. *Sprint* #3

El tercer *sprint* se ha llevado a cabo desde el 3 de diciembre de 2023 hasta el 19 de enero del 2024. A continuación, se exponen las diferentes historias de proyecto junto con las tareas que se han realizado.

- **H2.** Planificación del *sprint*
  - **T2.1** Definición del objetivo del *sprint*.
  - **T2.2** Planificación de tareas a realizar durante el *sprint*.
  - **T2.3** Balance temporal del *sprint*.
  - **T2.4** Medología de trabajo.
- **H3.** Desarrollo del *sprint*
  - **T3.1** Reuniones de sincronización semanales.
  - **T3.2** Actualización continua del tablero.
  - **T3.3** Reunión de comunicación.
- **H9.** Implementación
  - **T9.3** Desarrollo del componente de conectividad de objetos.
  - **T9.4** Implementación del backend (Flask)
- **H10.** Presentación
  - **T10.1** Elaboración de la presentación.

En la imagen 2.5 se muestra el diagrama de Gantt del tercer sprint.

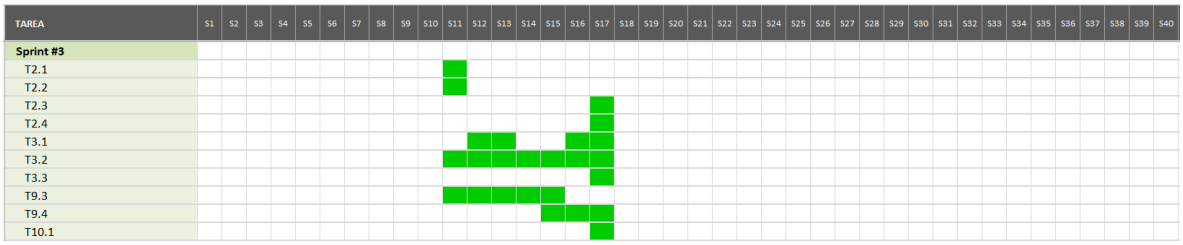


Figura 2.5: Diagrama de Gantt del *sprint* 3.

2.2.4. *Sprint* #4

El cuarto *sprint* se ha llevado a cabo desde el 19 de enero de 2024 hasta el 15 de marzo del 2024. A continuación, se exponen las diferentes historias de proyecto junto con las tareas que se han realizado.

- **H2.** Planificación del *sprint*
  - **T2.1** Definición del objetivo del *sprint*.
  - **T2.2** Planificación de tareas a realizar durante el *sprint*.
  - **T2.3** Balance temporal del *sprint*.
- **H3.** Desarrollo del *sprint*
  - **T3.1** Reuniones de sincronización semanales.
  - **T3.2** Actualización continua del tablero.
  - **T3.3** Reunión de comunicación.
- **H6.** Contexto científico-técnico
  - **T6.8** Descripción de las redes convolucionales (CNN).
- **H9.** Implementación
  - **T9.1** Desarrollo del componente de reconocimiento de objetos.
  - **T9.2** Desarrollo del componente de reconocimiento de caracteres.
  - **T9.4** Implementación del backend.
  - **T9.5** Implementación del frontend.
  - **T9.6** Creación del workflow de la aplicación.
  - **T9.7** Implementación de las funciones para cambiar el fichero json según la interacción del usuario.
- **H11.** Memoria
  - **T11.3** Redacción (parcial) del capítulo Antecedentes.
  - **T11.4** Redacción (parcial) del capítulo Descripción de la propuesta.
  - **T11.5** Redacción (parcial) del capítulo Procesamiento de los contenidos del modelo ER.

- **T11.6** Redacción (parcial) del capítulo Procesamiento de los contenidos del modelo ER.

La imagen 2.6 presenta el diagrama de Gantt del cuarto *sprint*.

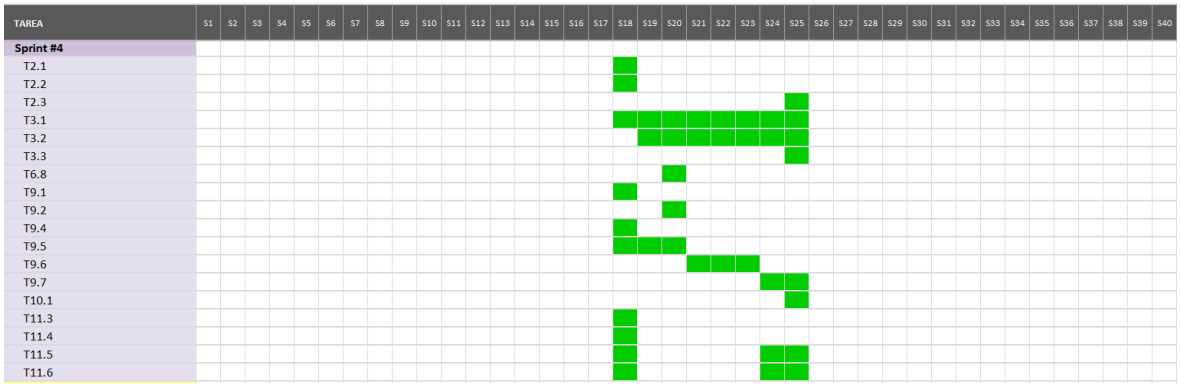


Figura 2.6: Diagrama de Gantt del *sprint* 4.

2.2.5. *Sprint* #5

El cuarto *sprint* se ha llevado a cabo desde el 16 de marzo de 2024 hasta el 8 de mayo del 2024. A continuación, se exponen las diferentes historias de proyecto junto con las tareas que se han realizado.

- **H2.** Planificación del *sprint*
  - **T2.1** Definición del objetivo del *sprint*.
  - **T2.2** Planificación de tareas a realizar durante el *sprint*.
  - **T2.3** Balance temporal del *sprint*.
- **H3.** Desarrollo del *sprint*
  - **T3.1** Reuniones de sincronización semanales.
  - **T3.2** Actualización continua del tablero.
  - **T3.3** Reunión de comunicación.
- **H9.** Implementación
  - **T9.4** Implementación del backend.
  - **T9.5** Implementación del frontend.
  - **T9.6** Creación del workflow de la aplicación.
  - **T9.7** Implementación de las funciones para cambiar el fichero json según la interacción del usuario.
- **H10.** Presentación
  - **T10.1** Elaboración de la presentación.

- **H11.** Memoria
  - **T11.2** Redacción (parcial) del capítulo Planificación.

La imagen 2.7 ilustra el diagrama de Gantt del quinto *sprint*.

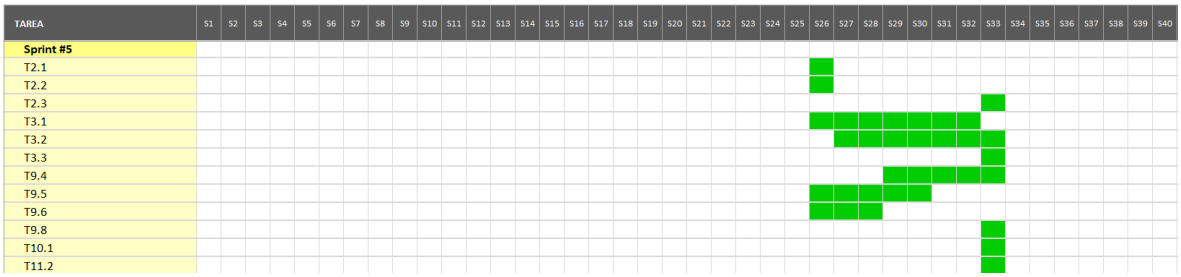


Figura 2.7: Diagrama de Gantt del *sprint* 5.

2.2.6. *Sprint* #6

El cuarto *sprint* se ha llevado a cabo desde el 9 de mayo de 2024 hasta el 28 de junio del 2024 y ha tenido una carga de trabajo de 90 horas. A continuación, se exponen las diferentes historias de proyecto junto con las tareas que se han realizado.

- **H2.** Planificación del *sprint*
  - **T2.1** Definición del objetivo del *sprint*.
  - **T2.2** Planificación de tareas a realizar durante el *sprint*.
  - **T2.3** Balance temporal del *sprint*.
- **H3.** Desarrollo del *sprint*
  - **T3.1** Reuniones de sincronización semanales.
  - **T3.2** Actualización continua del tablero.
  - **T3.3** Reunión de comunicación.
- **H9.** Implementación
  - **T9.4** Implementación del backend.
  - **T9.5** Implementación del frontend.
  - **T9.8** Implementación de las funciones para generar el feedback del estudiante.
  - **T9.9** Implementación de las funciones para generar el análisis de la conectividad.
- **H10.** Presentación
  - **T10.1** Elaboración de la presentación.
- **H11.** Memoria
  - **T11.1** Redacción (parcial) del capítulo Introducción.

- **T11.2** Redacción (parcial) del capítulo Planificación.
- **T11.3** Redacción (parcial) del capítulo Antecedentes.
- **T11.4** Redacción (parcial) del capítulo Descripción de la propuesta.
- **T11.5** Redacción (parcial) del capítulo Procesamiento de los contenidos del modelo ER.
- **T11.6** Redacción (parcial) del capítulo Procesamiento de los contenidos textuales.
- **T11.7** Redacción del capítulo Validación del usuario.
- **T11.8** Redacción del capítulo Evaluación del modelo y generación de feedback.
- **T11.9** Redacción del capítulo Conclusiones y trabajo a mayores.

La imagen 2.8 muestra el diagrama de Gantt del sexto *sprint*.

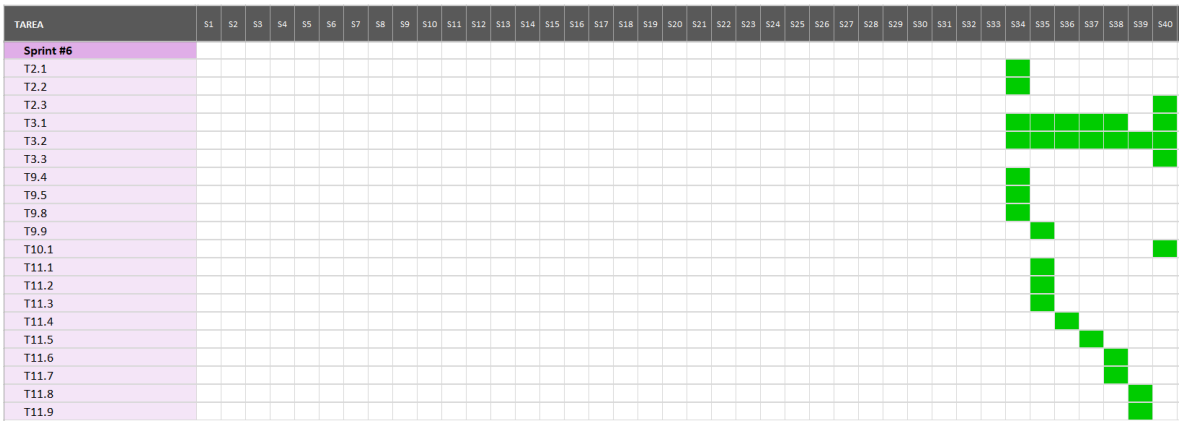


Figura 2.8: Diagrama de Gantt del *sprint* 6.

2.3. Presupuestos

En esta sección, se presenta el coste total estimado para llevar a cabo este proyecto, que resulta de la suma de los presupuestos dedicados al *hardware*, al *software* y a los recursos humanos. Cada uno de estos aspectos ha sido evaluado de forma independiente, proporcionando una visión completa del presupuesto general del proyecto. Para realizar el cómputo del coste del *hardware* y del *software*, se emplea la siguiente fórmula 2.1:

$$\text{C(euros)} = \frac{\text{Coste total (euros)}}{\text{Vida útil (meses)}} \times \text{Porcentaje de uso} \times \text{Tiempo de uso (meses)} \tag{2.1}$$

2.3.1. *Hardware*

Para llevar a cabo este proyecto, se ha utilizado un ordenador personal de gama media-alta con las siguientes especificaciones: procesador i7-1065G7, sistema operativo de 64 bits, 16 GB de RAM, 256 GB de almacenamiento SSD y 1 TB de almacenamiento HDD. Además, ha sido fundamental disponer de una buena conexión Wi-Fi para acceder a artículos y proyectos relacionados, así como para utilizar herramientas en línea como Overleaf, Google Colab, Teams,

entre otras. Sin embargo, no se han considerado los costes eléctricos relacionados con la carga del ordenador, ya que no se puede determinar con certeza su porcentaje de uso en este proyecto.

Al aplicar la fórmula 2.1, podemos calcular que el coste total relacionado con el *hardware* es de: 63,24€. La tabla 2.1 detalla el desglose del coste del *hardware*.

Herramienta	Coste total (€)	Vida útil (meses)	Porcentaje uso	Uso meses	Coste real (€)
Ordenador personal	1098	5*12=60	35	8	51,24
WI-FI	15 (al mes)		10	8	12
Total					63,24

Tabla 2.1: Presupuesto *hardware*.

2.3.2. *Software*

El coste total estimado del software es de 0€, dado que todas las herramientas empleadas en el desarrollo de este proyecto disponen de licencia gratuita, que ha sido sufiente para cubrir las necesidades de este proyecto. La tabla 2.2 muestra las herramientas que se van a emplear para llevar a cabo este proyecto.

Herramienta	Coste por mes	Porcentaje uso	Uso meses	Coste real (€)
Teams	0	95	8	0
Trello	0	100	8	0
Overleaf	0	100	8	0
Google Colab	0	100	5	0
Google Vision	0	100	3	0
Total				0

Tabla 2.2: Presupuesto *software*.

2.3.3. Recursos humanos

El estudiante encargado de este proyecto desempeñará distintos roles. Estos son los siguientes:

- **Analista de sistemas:** encargado de realizar el análisis del proyecto.
- **Arquitecto de software:** responsable de realizar el diseño del proyecto.
- **Desarrollador de *frontend*:** encargado de desarrollar la parte visual de la aplicación del proyecto.
- **Desarrollador de *backend*:** responsable de desarrollar las funciones lógicas de la aplicación utilizando el lenguaje de programación *Python*.

- ***Tester de software*:** encargado de evaluar el funcionamiento correcto del *software*.
- **Documentalista:** responsable de realizar la memoria del proyecto.

El presupuesto de recursos humanos se calculará como la suma ponderada de los salarios brutos de estos seis roles , además de la cotización correspondiente. La tabla 2.3 detalla el salario medio anual de cada rol, la cotización empresarial que el estudiante dedicará a cada uno, el coste total asociado a cada uno de estos roles para el proyecto, la cotización y el total de cada rol sumando la cotización, así como el total del proyecto. La empresa está obligada a dar de alta en la seguridad social (SS) al trabajador, esto supone un 31,90 % extra de su salario bruto anual.

Rol	Salario anual(€)	Salario por hora	Tiempo (h)	Total sin SS (€)	SS (€)	Total con SS (€)
Analista de sistema (17 %)	33.000 [51]	15,87	76,5	1214,06	387,29	1601,35
Arquitecto de software (5 %)	45.000 [52]	21,63	22,5	486,68	155,25	641,93
Desarrollador de *frontend* (4 %)	31.969 [55]	15,37	18	276,66	88,25	364,91
Desarrollador de *backend* (62 %)	34.497 [53]	16,59	279	4628,61	1476,53	6105,14
Tester software (2 %)	41.200 [57]	19,81	9	178,29	56,87	235,16
Documentalista (10 %)	23.585 [54]	11,34	45	510,3	162,79	637,09
Total			450			9585,58

Tabla 2.3: Presupuesto *recursos humanos*.

El presupuesto de recursos humanos se ha calculado en base a 450 horas, dado que este proyecto no solo comprende el TFG, sino también una beca de colaboración que incrementa las horas dedicadas de 300 a 450.

2.3.4. Presupuesto total del proyecto

Por lo tanto, el presupuesto total del proyecto asciende a 9648,82 euros. Esta cifra, como se muestra en la tabla 2.4, se ha obtenido sumando el presupuesto destinado al *hardware* (63,24€), al *software* (0€) y a los recursos humanos (9585,58€).

Coste *harware* (€)	Coste *software* (€)	Coste RRHH (€)	Coste total (€)
63,24	0	9585,58	9648,82

Tabla 2.4: Presupuesto total del proyecto.

2.4. Balance temporal y económico

En esta sección se presenta un análisis de la diferencia entre la planificación temporal y ecónomica con la real.

2.4.1. Balance temporal

Al igual que la planificación temporal, el balance temporal se ha realizado al finalizar cada *sprint*. Como se indica en la sección 2.2, aunque en un primer momento se planificó llevar a cabo el TFG en 5 *sprints*, finalmente se ha necesitado de un sexto *sprint* para poder terminar exitosamente el proyecto. Esto se ha debido, a que el estudiante tuvo algunos problemas personales en el *sprint* 1 impidiendole lograr los objetivos propuestos para dicho *sprint*, así como su incorporación a la vida laboral le llevo a tener algunos retrasos en el *sprint* 4.

Por otro lado, en la planificación se indicaba que debido a seguir la metodología ASAP todos los *sprints* tendrían la misma carga de trabajo (aproximadamento 90 horas cada uno). A continuación, se muestra el tiempo real empleado en cada *sprint*:

- *Sprint* 1: 48 horas
- *Sprint* 2: 85 horas
- *Sprint* 3: 106 horas
- *Sprint* 4: 111 horas
- *Sprint* 5: 90 horas
- *Sprint* 6: 120 horas

El total de horas empleadas (560 horas) en la realización de este proyecto han superado a las planificadas (450 horas) debido a que se han tenido que implementar en más de una ocasión las tareas de reconociento de caracteres e identificación de componentes. Esto se debió a que las primeras herramientas utilizadas en dichas tareas no nos han proporcionado los resultados deseasos. Por ello, se ha tenido que investigar e implementar otras herramientas. stos retrasos no afectan a la planificación individual de cada *sprint*, ya que esta se realiza al inicio de cada *sprint*, y es en esa misma reunión donde se plantean nuevas herramientas para la implementación de estas tareas. Sin embargo, sí afectan significativamente a la planificación temporal completa del proyecto.

En los siguientes puntos, se van a mostrar los diagramas de balance temporal de cada *sprint* y se va a explicar los desfases de realización de algunas tareas. Las celdas que se encuentran coloreadas en verde significan que la tarea se ha realizado en la fecha planificada, las celdas en rojo definen la fecha en la que se realizó realmente la tarea y las que están en gris significan cuando se había planificado la realización de una tarea.

Sprint 1

Como se indicaba previamente, debido a causas personales el estudiante no pudo realizar ninguna tarea en la tercera semana del proyecto, lo que provocó que la mayoría de las tareas se vieran retrasadas. En la imagen 2.9 se observa que las tareas T5.2, T5.3, T5.4, T6.1, T6.2, T6.3, T9.1, T9.2, T11.1, T11.2, T11.3, T11.4 se retrasaron.

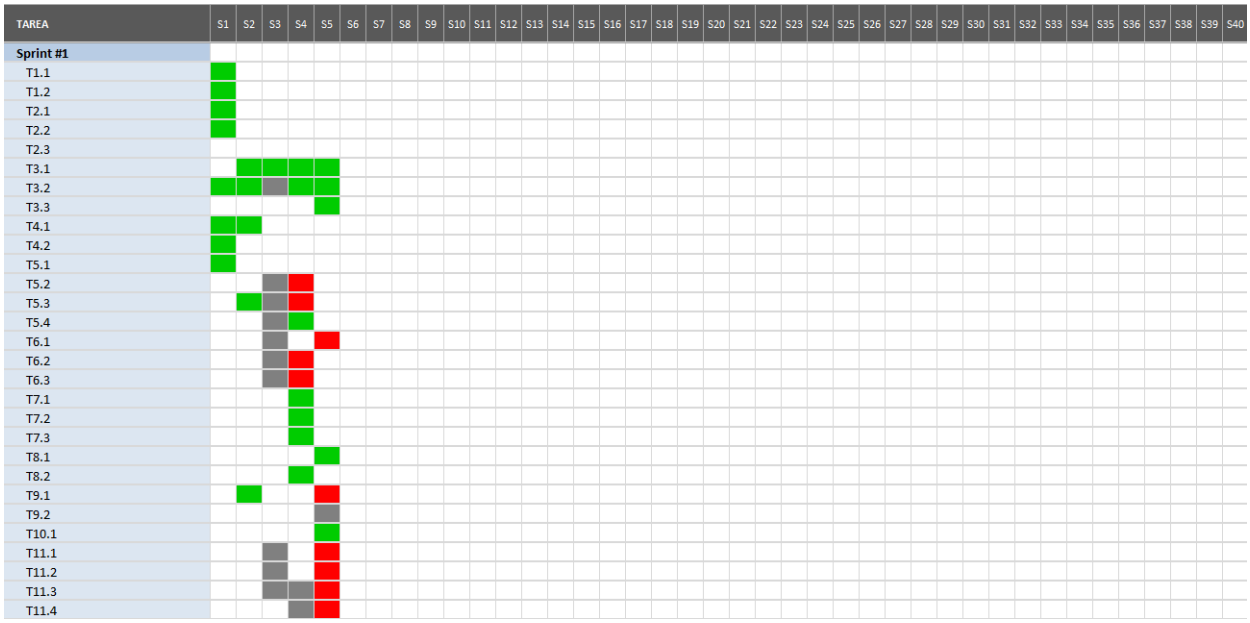


Figura 2.9: Diagrama de Gantt del balance temporal del *sprint* 1.

Sprint 2

En este segundo *sprint* el estudiante intentó realizar una mayor cantidad de tareas para compensar los retrasos del *sprint* anterior. Esto provocó ciertos desfases entre la planificación temporal de las tareas y su realización. Como se puede ver en la imagen 2.10, las tareas T6.4, T6.6 y T6.9 se adelantaron, mientras la tarea T9.2 se atraso y la tarea T6.7 no se llegó a realizar. Además, la reunión de sincronización (T3.1) de la tercera semana del *sprint* se canceló porque el estudiante se encontrba enfermo.

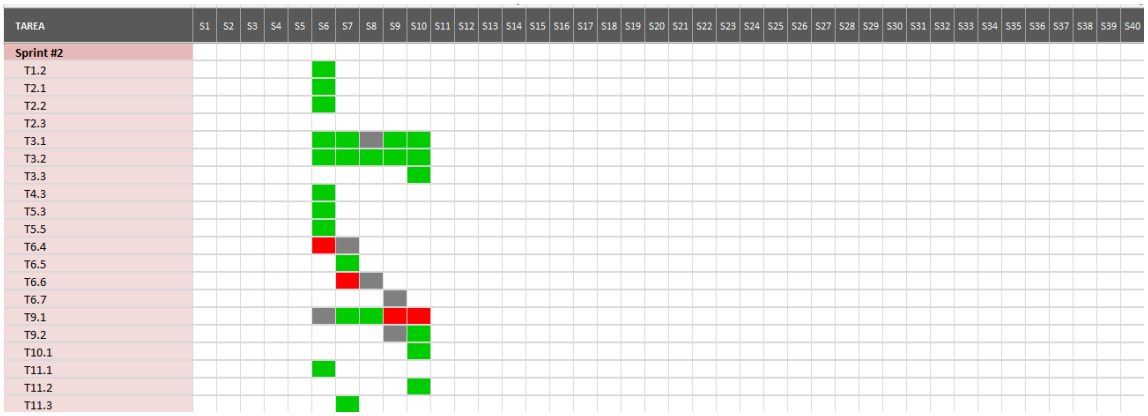


Figura 2.10: Diagrama de Gantt del balance temporal del *sprint* 2.

Sprint 3

En este *sprint* se cumplió casi a la perfección la planificación. La única diferencia que hubo con la realidad es que la tarea T9.3 se finalizó una semana antes. La imagen 2.11 muestra el

diagrama de Gantt del balance temporal del tercer *sprint*.

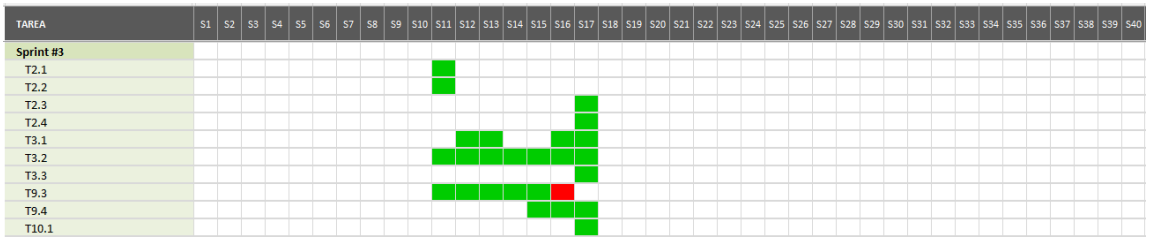


Figura 2.11: Diagrama de Gantt del balance temporal del *sprint* 3.

Sprint 4

En el cuarto *sprint*, el estudiante empezó a trabajar, lo que le provocó que su concentración a la hora de realizar tareas disminuyera. Esto, junto con que era la primera vez que realizaba una aplicación del calibre de este proyecto, le llevó a tener ciertos bloqueos en las tareas T9.4 y T9.5 que, según la planificación, deberían haberse finalizado en este *sprint*, pero que en realidad, aunque se trabajó en su implementación, no se consiguió. Debido al bloqueo con estas tareas, se vieron adelantadas las tareas T11.4 y T11.6, y retrasadas las tareas T1.5, T9.7. Por último, cabe destacar que por la falta de tiempo el estudiante no cumplió con la tarea de actualizar el tablero del proyecto las semanas 23 y 24. La imagen 2.11 muestra el diagrama de Gantt del balance temporal del cuarto *sprint*.

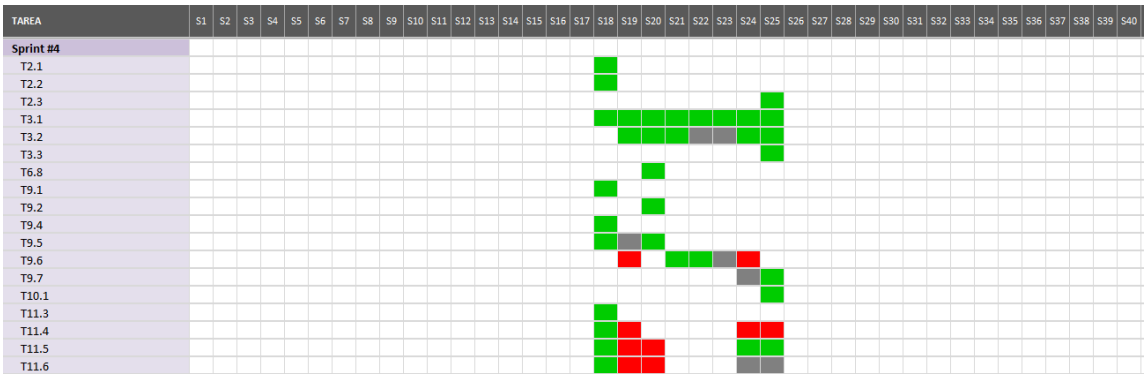


Figura 2.12: Diagrama de Gantt del balance temporal del *sprint* 4.

Sprint 5

Al inicio de este *sprint* se optó por añadir un sexto *sprint*, ya que los retrasos causados en el *sprint* anterior hacían casi imposible poder terminar el TFG en este quinto *sprint*. En este *sprint* el estudiante consiguió desbloquearse y acabar la tarea de implementación del backend (T9.4) tres semanas antes de lo previsto. Debido a que el estudiante le dio más importancia a la implementación de la aplicación se retrasó la tarea T11.2 a la última semana del *sprint* y se le olvidó actualizar el tablero del proyecto (T3.1) durante las semanas 26, 27, 28 y 29. La imagen 2.11 muestra el diagrama de Gantt del balance temporal del quinto *sprint*.

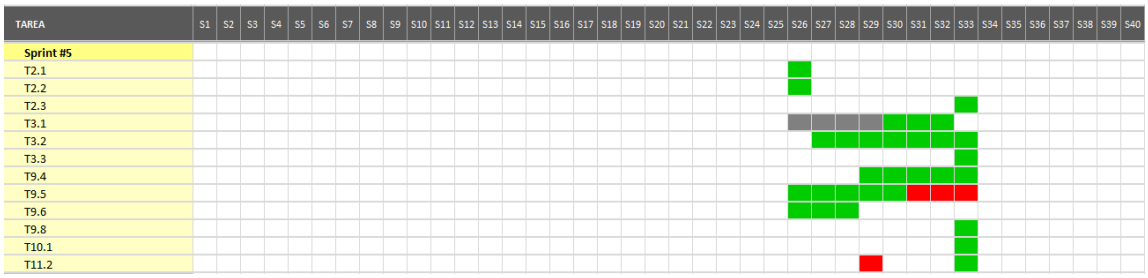


Figura 2.13: Diagrama de Gantt del balance temporal del *sprint* 5.

Sprint 6

En el último sprint no sé cumplió con la planificación de la redacción de los diferentes capítulos, debido a que en la planificación inicial del sprint no se consideró el tiempo necesario para la revisión de estos por parte del tutor, ni la posterior correción del estudiante de los mismos. La imagen 2.11 muestra el diagrama de Gantt del balance temporal del sexto *sprint*.

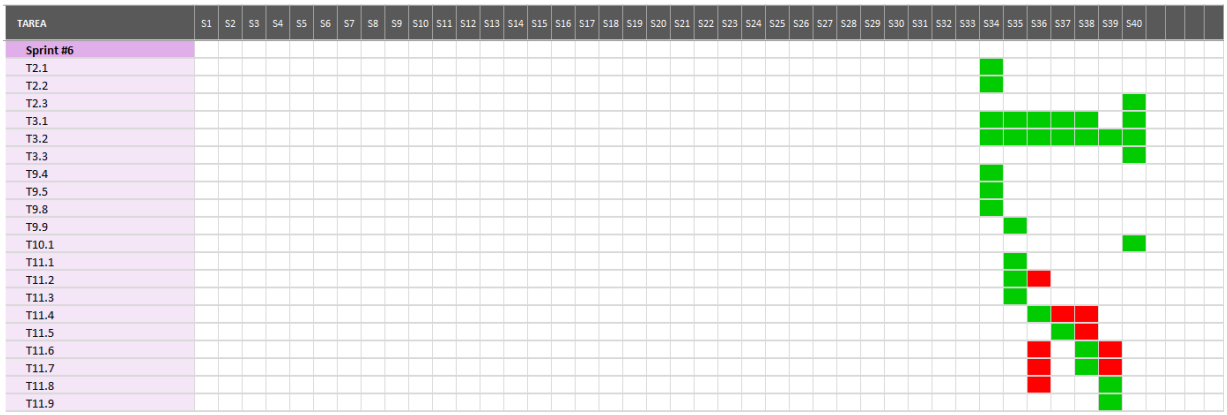


Figura 2.14: Diagrama de Gantt del balance temporal del *sprint* 6.

2.4.2. Balance económico

En este punto, se presenta el coste total real del proyecto, que resulta de la suma del coste real del *hardware*, el *software* y los recursos humanos. Inicialmente el proyecto fue planificado para realizarse en 8 meses, pero finalmente se han necesitado de 10 meses (no sé han contado las diferentes semanas de vacaciones que ha habido durante el curso). Este retraso temporal ha producido un aumento en el coste del proyecto.

El coste del *hardware* se ha visto incrementado de 63,24€ a 79,05€, es decir, ha aumentado 15,51€ . La tabla 2.5 muestra el calculo del coste real del *hardware*.

Herramienta	Coste total (€)	Vida útil (meses)	Porcentaje uso	Uso meses	Coste real (€)
Ordenador personal	1098	5*12=60	35	10	64,05
WI-FI	15 (al mes)		10	10	15
Total					79,05

Tabla 2.5: Coste real del *hardware*.

A pesar del aumento de horas del proyecto, el coste real del *software* es el mismo que su presupuesto (0€). Esto ha sido posible gracias a que todas las herramientas utilizadas disponen de licencias gratuitas, las cuales han sido suficientes para cubrir completamente las necesidades del proyecto.

El coste de los recursos humanos se ha visto también afectado debido al aumento de horas empleadas en el proyecto. Las horas dedicadas a este proyecto han sido finalmente de 560 como se puede ver en el artefacto "Cuaderno de trabajo". El coste real de los recursos humanos ha sido finalmente de 11973,51€, como se puede ver en la tabla 2.6. El presupuesto del mismo era de 9595,58€, lo que ha producido un incremento en el coste de personal de 2387,93€.

Rol	Salario anual(€)	Salario por hora	Tiempo (h)	Total sin SS (€)	SS (€)	Total con SS (€)
Analista de sistema (17 %)	33.000 [51]	15,87	95,2	1510,82	481,95	1992,77
Arquitecto de software (5 %)	45.000 [52]	21,63	28	605,64	193,20	798,84
Desarrollador de *frontend* (4 %)	31.969 [55]	15,37	22,4	344,29	109,83	454,12
Desarrollador de *backend* (62 %)	34.497 [53]	16,59	347,2	5760,05	1837,46	7597,51
Tester software (2 %)	41.200 [57]	19,81	11,2	221,87	70,78	292,65
Documentalista (10 %)	23.585 [54]	11,34	56	635,04	202,58	837,62
Total			560			11973,51

Tabla 2.6: Coste real de los *recursos humanos*.

Por lo tanto, como se muestra en la tabla 2.7, el coste total del proyecto asciende a 12052,56€ que se obtine al sumar del coste real del *hardware* (79,05€), el *software* (0€) y los recursos humanos (11973,51€). El presupuesto del proyecto era de 9648,82€, por lo que ha habido un incremento del coste del proyecto de 2403,74€.

Coste *harware* (€)	Coste *software* (€)	Coste RRHH (€)	Coste total (€)
79,05	0	11973,51	12052,56

Tabla 2.7: Coste real del proyecto.

Capítulo 3

Antecedentes

En este capítulo, se van a exponer los conocimientos básicos necesarios para entender el impacto del proyecto y la forma en la que se ha abordado la construcción del sistema informático. Para ello, se describe el diseño conceptual de datos (sección 3.1), el estado del arte (sección 3.2), el aprendizaje automático (sección 3.3), la visión computacional (sección 3.4) y el reconocimiento óptico de caracteres (sección 3.5).

3.1. Diseño conceptual de datos

En esta sección se explicará qué es la etapa de diseño conceptual en un proyecto informático y por qué es importante. Además, se describirá qué es el modelo ER y cómo se construye.

En la subsección se explica qué es un sistema de información (SI) y se describen las diferentes actividades del ciclo de vida de un SI. En la subsección 3.1.2 se presenta una definición de diseño conceptual y se enumeran los objetivos de esta etapa. Por último, en la subsección 3.1.3 se detalla qué es un modelo Entidad-Relación (ER), explicando cada uno de sus componentes.

3.1.1. Sistema de información y bases de datos

Un sistema de Información (SI) dispone los recursos necesarios para recolectar, gestionar y diseminar la información corporativa de una organización. Para ello, el SI delega en un sistema gestor de bases de datos (SGBD) las responsabilidades propias del mantenimiento y explotación de su base de datos, utilizando para ello una configuración hardware que satisfaga sus requisitos. Asimismo el SI dará soporte a los programas de aplicación que proporcionan la funcionalidad requerida por sus usuarios [30]. Los componentes que conforman un SI son: la base de datos, el SGBD sistema de gestión de la base de datos (SGBD), los programas de aplicación, los recursos hardware y las personas que lo desarrollan y lo usan.

"Una base de datos comprende una colección de datos relacionados de forma lógica, y el esquema que describe estos datos, para satisfacer las necesidades de información de una determinada organización." [58].

La imagen 3.1 muestra las diferentes actividades del ciclo de vida de la base de datos.

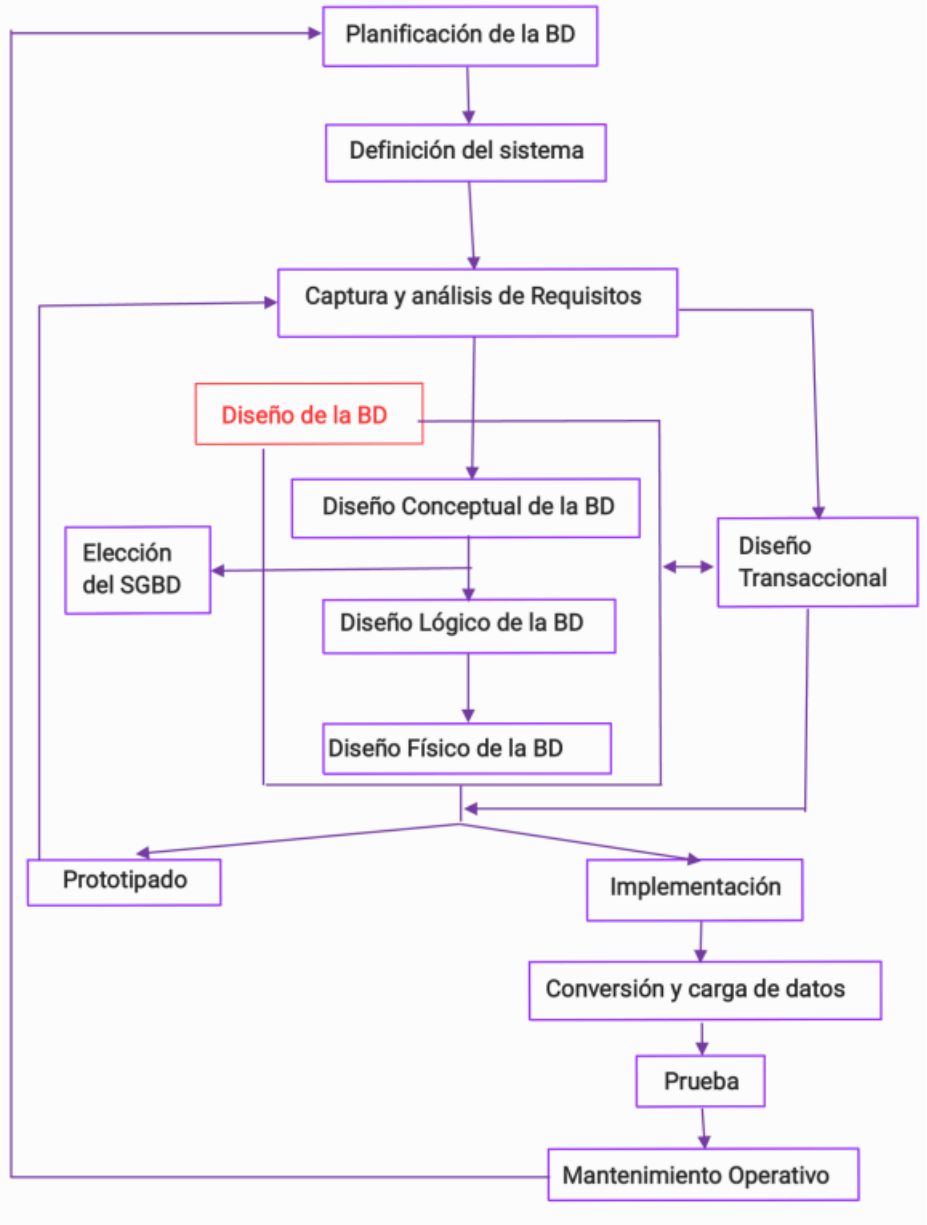


Figura 3.1: Actividades que forman el ciclo de vida de un SI.

3.1.2. Diseño conceptual

El diseño conceptual se realiza una vez se han identificado y analizado los requisitos del sistema a desarrollar. Consiste en "obtener un modelo de datos que describa los requisitos de datos y los flujos de información corporativos" [47]. Los objetivos de esta etapa son:

- Identificar los tipos de entidades.
- Identificar los tipos de relaciones.
- Determinar las restricciones de multiplicidad de las relaciones.

- Identificar y asociar los atributos con los tipos de entidades o relaciones correspondientes.
- Determinar los dominios de los atributos.
- Identificar la clave primaria y las claves alternativas de las entidades.
- Comprobar la existencia de redundancia.
- Comprobar que el modelo propuesto soporta las transacciones requeridas.
- Revisar el modelo con los usuarios y asegurarse que representa completamente sus requisitos.

En el diseño conceptual se construye tanto el modelo Entidad-Relación, como el diccionario de datos (de las entidades y relaciones).

¿Qué es un diccionario de datos? Se trata de un documento que describe de forma detallada las entidades y relaciones identificadas durante el diseño conceptual, prestando especial atención al dominio de los datos y a las reglas de negocio que les afectan. Este documento es fundamental a lo largo de toda la vida de la base de datos ya que contiene información que es transversal a todas las etapas del ciclo de vida de la base de datos.

3.1.3. Modelo Entidad-Relación (ER)

"El modelo ER es una herramienta que permite representar de manera simplificada cómo personas, objetos o conceptos se relacionan entre sí. Se utiliza para exponer cómo se organiza la información en una base de datos." [61].

El propósito del modelo entidad-relación en el diseño conceptual de bases de datos es proporcionar una representación clara y abstracta de la estructura de datos, de manera que ayude en la comunicación y validación de los requisitos, y que sirva como un punto de partida para el diseño lógico y la implementación de la BD. Por lo tanto, es una herramienta esencial en el proceso de diseño de BD que contribuye a la eficiencia y la calidad de los sistemas de información.

La imagen 3.2 muestra un diagrama ER simple, que describe un sistema de información de gestión académica. Este modelo ER nos proporciona la siguiente información:

- Un estudiante se caracteriza por su DNI, nombre, fecha de nacimiento, edad y teléfono.
- Un profesor se identifica por un identificador único.
- Una asignatura se identifica por un identificador único.
- Una matrícula se identifica por un código único y depende de la existencia de un estudiante.
- Un estudiante realiza una única matrícula.
- Una matrícula pertenece a un único estudiante.
- Una matrícula tiene una o más asignaturas.
- Una asignatura puede pertenecer a 0, 1 o más matrículas.
- Un profesor imparte una o más asignaturas.

- Una asignatura es impartida por uno o más profesores.
- Un profesor puede supervisar a uno o más profesores.
- Un profesor es supervisado por otro profesor.

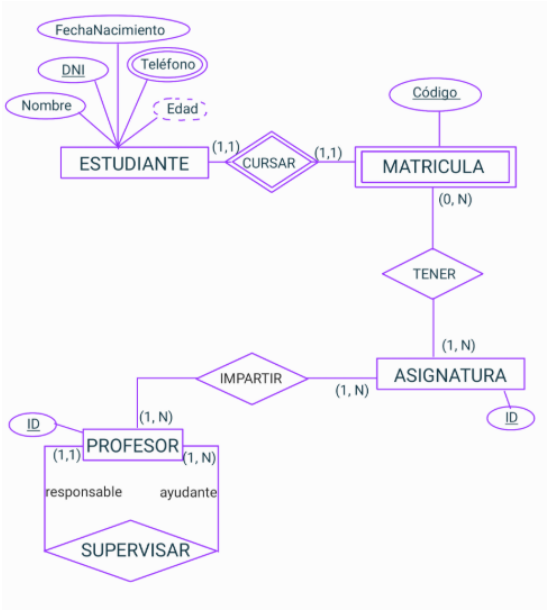


Figura 3.2: Ejemplo de un modelo entidad relación.

El modelo ER está compuesto de tres elementos:

Entidades. Describen conjuntos de elementos del mundo real que comparten las mismas propiedades y cuya existencia es independiente desde el punto de vista de la organización. Las entidades se pueden dividir en:

- Entidad fuerte: son las entidades cuya existencia no depende de ninguna otra entidad. Por ejemplo, un ESTUDIANTE existe por sí mismo, y se identifica unívocamente mediante el atributo DNI.
- Entidad débil: son las entidades cuya existencia depende de algún otro tipo de entidad. Por ejemplo, para identificar una entidad MATRÍCULA será necesario conocer a que ESTUDIANTE pertenece, puesto que un expediente debe estar asociado necesariamente a un alumno.

Las entidades se representan utilizando rectángulos. Las entidades fuertes tienen un borde simple, mientras que las entidades débiles están formadas por un borde doble. El nombre de las entidades se escribe en mayúsculas y en singular. Las imágenes 3.3 y 3.4 muestran respectivamente un ejemplo de entidad fuerte y uno de entidad débil.

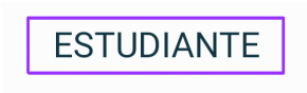


Figura 3.3: Ejemplo de entidad fuerte.

Figura 3.4: Ejemplo de entidad débil.

Atributos. Son propiedades que caracterizan las entidades y relaciones. Los atributos se pueden clasificar en simples o compuestos, dependiendo si están formados por un único elemento o varios elementos. Por otro lado, los atributos pueden ser monovaluados (si contienen un único valor en cada instancia de una entidad) o multivaluados (si contienen múltiples valores diferentes para cada instancia de una entidad). Por último, un atributo se clasifica como derivado cuando su valor se calcula o deduce a partir de otros atributos, los cuales pueden pertenecer o no a la misma instancia.

El conjunto formado por la menor cantidad de atributos que identifica de forma unívoca a una entidad se denomina identificador. Si este identificador está formada por dos o más atributos entonces lo denominaremos clave compuesta.

Los atributos se representan mediante eclipses. Los atributos multivaluados tienen un doble borde, mientras que los derivados tienen un borde discontinuo. La primera letra de los atributos se escribe en mayúscula y el resto de letras en minúscula. Aquellos atributos que forman la clave primaria de la entidad aparecerán subrayados.

Las siguientes 4 imágenes 3.5, 3.6, 3.7 y 3.8 muestran respectivamente un ejemplo de representación de un atributo simple, un atributo primario, un atributo derivado y un atributo multivaluado.

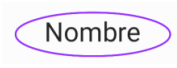


Figura 3.5: Ejemplo de un atributo.

Figura 3.6: Ejemplo de un atributo que es clave primaria.

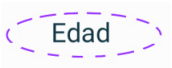


Figura 3.7: Ejemplo de un atributo derivado.

Figura 3.8: Ejemplo de un atributo multivaluado.

Relaciones. Describen las asociaciones significativas entre tipos de entidades. Las relaciones al igual que las identidades pueden contener atributos. Según el grado de relación (que determina el número de entidades que participan en la relación) encontramos los siguientes tipos:

- Binarias: asocian dos entidades.
- N-arias: asocian tres o más entidades.
- Unarias: relacionan a una clase consigo misma, desempeñando diferentes roles. Los roles en estas relaciones varían según el contexto de los datos; por ejemplo en la relación SUPERVISAR que se muestra en la imagen 3.11 la entidad PROFESOR juega el rol de "responable" y de "colaborador", atendiendo al hecho de que cada asignatura tiene un profesor responsable de su coordinación y, a su vez, puede tener otros profesores que colaboren con él.

Las relaciones se modelan utilizando rombos. Las relaciones débiles, que asocian una entidad fuerte con una débil, tienen un doble borde. El nombre de las relaciones debe escribirse en mayúscula y se expresa en infinitivo. Las figuras 3.9 y 3.10 representan respectivamente un ejemplo de relación fuerte y uno de relación débil.

Figura 3.9: Ejemplo de una relación.

Figura 3.10: Ejemplo de una relación débil.

A continuación, la imagen 3.11 representa un ejemplo de relación unaria, la 3.12 de relación binaria y la 3.13 de relación n-aria.

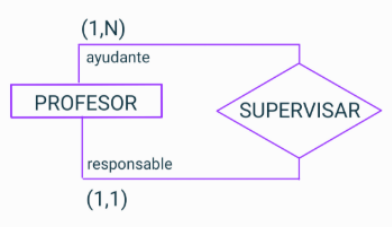


Figura 3.11: Ejemplo de una relación unaria.

Figura 3.12: Ejemplo de una relación binaria.

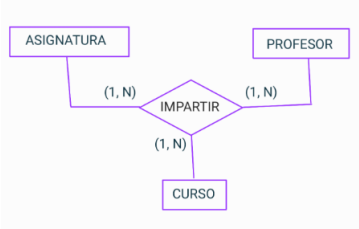


Figura 3.13: Ejemplo de una relación n-aria.

Además, cada relación está vinculada a una multiplicidad que define el número o rango de instancias de un tipo de entidad que puede relacionarse con una instancia de un tipo de entidad asociada a través de una relación específica. La multiplicidad impone dos restricciones fundamentales sobre la relación:

- Cardinalidad: Define el número máximo de instancias de cada tipo de entidad que participan en dicha relación. Existen tres tipos: uno a uno (1:1), uno a muchos (1,N) y muchos a muchos (N,N).

- Participación: Determina si todas las instancias de la entidad, o solo algunas, participan en la relación. La opcionalidad se representa con un 0 y la obligatoriedad con un 1.

Existe un tipo particular de relaciones que permiten vínculos jerárquicos entre entidades. Así, la entidad que actúa como raíz de la jerarquía se denomina "súperclase" mientras que las entidades subordinadas se denominan "subclases":

- Superclase, o entidad padre: es un tipo de entidad en cuyas instancias pueden distinguirse uno o más subgrupos
- Subclase, o entidad hija: es un subgrupo diferenciado de instancias de un tipo de entidad, que comparten alguna característica en común que las diferencia del resto de instancias pertenecientes a la superclase.

Estas relaciones tienen dos tipos de restricciones. La restricción de partición que determina si cada instancia de la superclase tiene que participar como instancia en alguna subclase. Y la restricción de disyunción que describe la relación entre las instancias de las subclases e indica si es posible que una instancia de la superclase sea, a su vez, instancia de una o más subclases. Por tanto, estas relaciones pueden ser obligatorias o optativas y disjuntas o no disjuntas. Las siguientes figuras (3.14, 3.15, 3.16 y 3.17 muestran los 4 tipos de relaciones IS-A existentes:

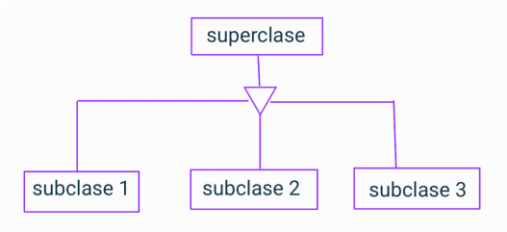


Figura 3.14: IS-A opcional y no disjunta.

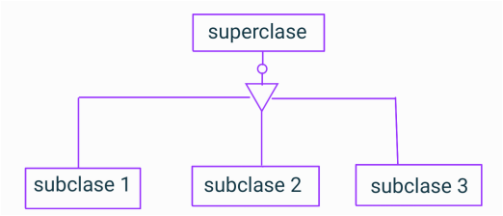


Figura 3.15: IS-A obligatoria y no disjunta.

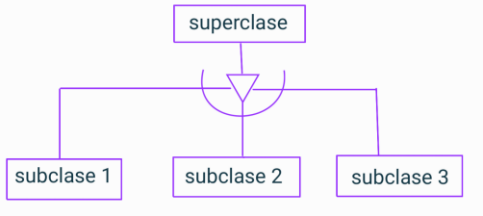


Figura 3.16: IS-A opcional y disjunta.

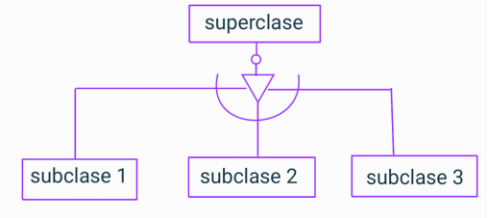


Figura 3.17: IS-A obligatoria y disjunta.

3.2. Estado del arte

En este apartado se hará una revisión general de los trabajos relacionados con la propuesta desarrollada en nuestro proyecto, para así proporcionar una visión más cercana del estado del arte. Esta revisión se llevará a cabo de acuerdo con 7 dimensiones comparativas: generación de feedback inmediato, utilización de diagramas externos, uso de la aplicación para practicar, contiene explicaciones de conceptos, generación de preguntas, capacidad de leer imágenes y comparación de imágenes, que serán la base sobre la que analizaremos nuestra propuesta.

Para ello se presentará un breve resumen de cada uno de los artículos seleccionados 3.2.1, se realizará un análisis comparativo 3.2.2, para el cuál hemos considerado 7 dimensiones (generación de feedback inmediato, utilización de diagramas externos, uso de la aplicación en exámenes, uso de la aplicación para practicar, contiene explicaciones de conceptos, generación de preguntas, capacidad de leer imágenes y comparación de imágenes).

3.2.1. Trabajos relacionados

Schildgen (MonstER Park - The Entity-Relationship-Diagram Learning Game)

MonstER Park [48] es un juego que enseña los fundamentos de los diagramas ER de forma divertida y dinámica a través de diferentes historias. Al iniciar el juego, lo que ve el jugador es una pantalla, como la de la imagen 3.18, en la que van a ir apareciendo diferentes personajes haciendo preguntas o contando una historia. En la parte inferior de la pantalla el usuario deberá seleccionar qué tipo de objeto va a insertar (entidad, atributo, relación,...), así como su nombre para solucionar cada pregunta y así pasar a la siguiente. Una vez insertado el objeto correspondiente, este se dibujará en la pantalla dentro de un rectángulo si es una entidad, de un ovalo si es un atributo,... Cada vez que aparece un concepto nuevo, se generará su correspondiente explicación. Además, el jugador recibe un feedback inmediato con sus errores y aciertos.

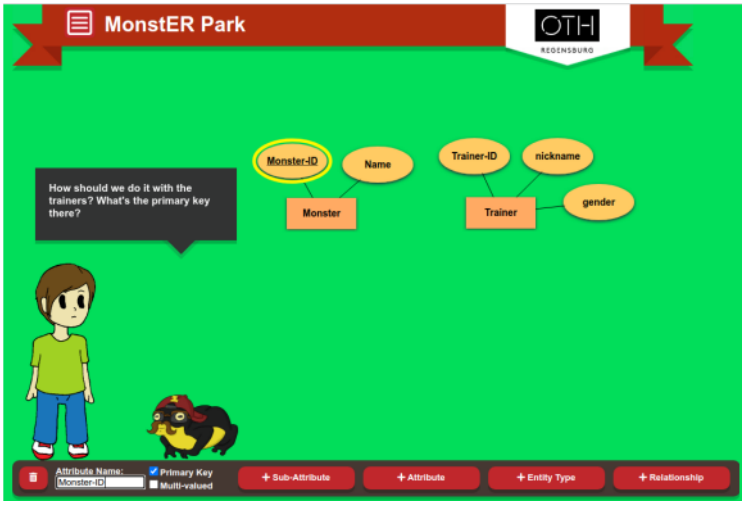


Figura 3.18: Pantalla del juego MonstER Park [48].

Sin embargo, la propuesta plantea una limitación específica. Aunque el programa almacena un conjunto de soluciones posibles para cada objeto, existe la posibilidad de que el usuario haya introducido un nombre que signifique lo mismo y que en cambio al no pertenecer a ese conjunto el programa se lo corregirá como erróneo. Por ejemplo, el usuario puede haber llamado a una entidad "ESTUDIANTE", pero el modelo solo considerar como válida la palabra "ALUMNO" y por tanto darle como errónea la entidad, cuando en realidad son sinónimos.

Foss, Urazova y Law-rence (Automatic Generation and Marking of UML Database Design Diagrams)

En este proyecto [14] se ha creado un sistema llamado AutoER para usarse en asignaturas de introducción a bases de datos. Los profesores pueden utilizar esta herramienta tanto para crear ejercicios con los que los estudiantes practiquen, como para hacer exámenes y evaluarlos de forma inmediata. AutoER permite a los profesores crear preguntas de forma manual. Los profesores pueden cambiar la interfaz de usuario, la calificación y retroalimentación de los problemas y la generación de preguntas. Uno de los mayores problemas a la hora de calificar un diagrama ER es que pueden existir varios diagramas válidos para un mismo problema. Esta aplicación ha resuelto este problema creando una interfaz en la que el usuario interactúa directamente con las preguntas y el texto del problema, y en base a las respuestas del estudiante, el programa va agregando los componentes correspondiente al diagrama ER. Cuando el estudiante solicita feedback, envía el diagrama para su evaluación al servidor y el programa le genera una retroalimentación inmediata. El número de veces que el estudiante puede enviar la solución, así como, los aspectos que aparecerán en la retroalimentación están bajo el control del profesor, que los habrá predefinido con anterioridad. AutoER se centra en estudiantes que están aprendiendo por primera vez los conceptos de diseño y el proceso de extracción de elementos a partir de un texto. La figura 3.19 muestra la interfaz de usuario de la aplicación AutoER. En esta imagen encontramos en la parte superior izquierda el enunciado del problema, donde el usuario puede ir pinchando en las diferentes palabras y añadirlas como atributos o entidades. En caso de añadirla como atributo se le muestra una lista con todas las entidades para que seleccione la entidad a la que corresponda dicho atributo. En la parte inferior izquierda se muestra el diagrama ER que se va creando y en la columna derecha de la imagen se muestran todas entidades con sus atributos y sus claves primarias. En la imagen 3.20 vemos un ejemplo de retroalimentación, en la cual aparece solo la información de calificación de los diferentes componentes (entidades, atributos, relaciones...).

Jaimez-González y Martínez-Samora (DiagrammER: A Web Application to Support the Teaching-Learning Process of Database Courses Through the Creation of ER Diagrams)

Este proyecto [23] presenta una aplicación web (DiagrammER) para ayudar en el proceso de la enseñanza de cursos de bases de datos. DiagrammER ofrece funcionalidades tanto para profesores como para alumnos. En el caso de los profesores, tienen la capacidad de cargar ejercicios, los cuales pueden incluir diagramas ER. Es importante destacar que los profesores tienen la flexibilidad de crear ejercicios sin un diagrama ER inicialmente y asociarlo posteriormente. Por otro lado, los alumnos pueden diseñar sus propios diagramas ER y completar los ejercicios previamente subidos por el profesor. Además, la plataforma permite a los alumnos acceder y revisar los diagramas ER que han creado, proporcionando así un entorno interactivo para el aprendizaje de bases de datos.

La aplicación DiagrammER muestra dos características. La primera es que permite a los estudiantes generar un número ilimitado de diagramas, y la segunda es que tiene disponible todas las figuras necesarias para desarrollar diagramas ER extendidos.

La figura 3.21 muestra el diseño de la interfaz de la aplicación web DiagrammER para crear diagramas ER una vez el usuario ya se ha identificado como usuario o como profesor. En ambos caso muestra esta misma pantalla, en la esquina superior izquierda de esta encontramos el listado de ejercicios guardados por el usuario, si el usuario pulsa sobre uno de ellos entonces podrá

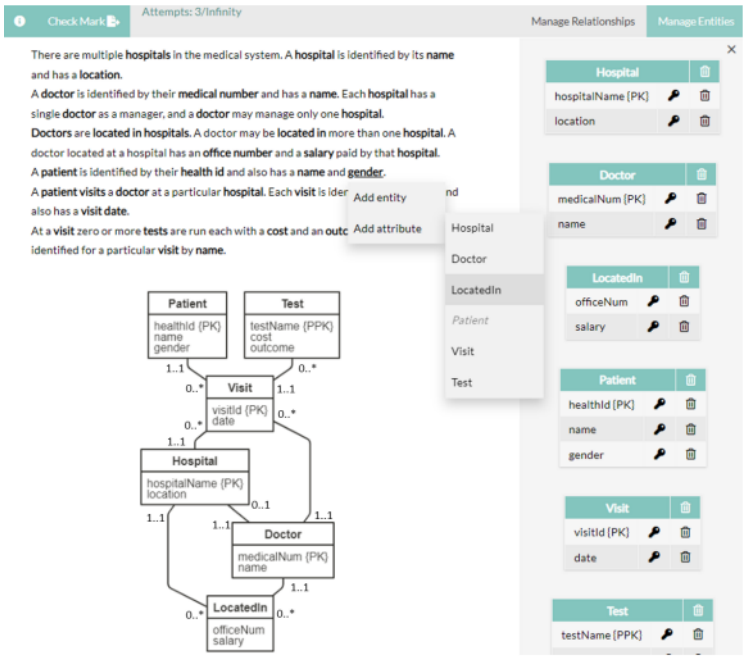


Figura 3.19: Interfaz de usuario y un estudiante respondiendo a una pregunta [14].

Entity name marks:	1.2 / 1.2
Entity attribute marks:	0.5 / 0.6
Entity primary key marks:	0.8 / 1.2
Weak entity key marks:	0.5 / 1.0
Total entity marks:	3.0 / 4.0
Relationship entity marks:	2.5 / 3.5
Relationship cardinalities marks:	2.5 / 3.5
Extra relationships:	-0.25
Total relationship marks:	4.75 / 7.0
Total marks:	7.75 / 11.0
Total scaled marks:	7.05 / 10.0

Figura 3.20: Ejemplo de retroalimentación [14].

visualizar la solución previamente generada y podrá actualizarlo si lo desea. En la parte central de la imagen se encuentra el área en el que el usuario irá creando el diagrama ER, para ello cuenta con un área de contenedores de figuras situado en la esquina izquierda inferior, así como de un área para editar el estilo del ttexto a la derecha de la imagen. En la parte superior de la imagen el usuario cuanta con la opción de guardar el modelo creado, crear un nuevo diagrama ER, exportar un diagrama, ver los ejercicios y acceder a la ayuda de la aplicación.

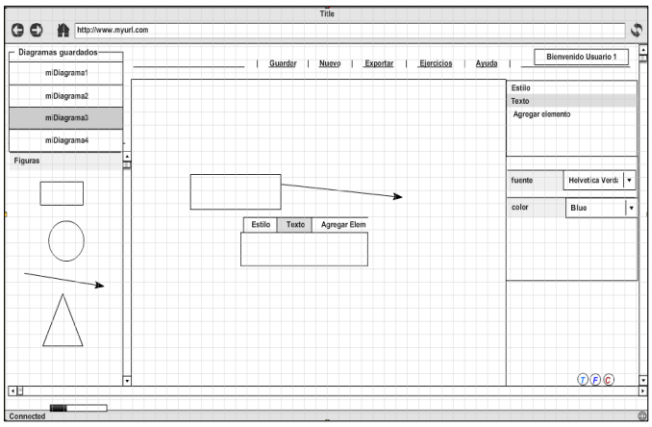


Figura 3.21: Interfaz de la aplicación web DiagrammeER para crear un diagrama ER [23].

Simanjunta (Proposed Framework for Automatic Grading System of ER Diagram)

Este artículo [50] presenta una investigación sobre el sistema de calificación automática de diagramas ER. Para ello utiliza un archivo XML que codifica el diagrama ER como entrada al sistema y propone dos enfoques para la evaluación de este:

- Algoritmo de distancia de edición de árbol para evaluar la similitud de los diagramas ER.
- Algoritmo de aprendizaje automático para construir un clasificador que califique automáticamente los diagramas ER.

Enterprise Architect es una herramienta de diseño y modelado visual basada en OMG UML. La plataforma soporta: el diseño y construcción de sistemas de software; modelado de procesos de negocios; y modelado de dominios basados en la industria [11]. Este sistema utiliza esta herramienta para exportar el diagrama en un archivo XMI, que consta de etiquetas XML. Las imágenes 3.22 y 3.23 muestran un diagrama ER y su correspondiente archivo XMI. En el XMI podemos ver como se añaden todos los componentes del modelo ER dentro de la etiqueta <elements>, y las conectividades de estos aparecerían dentro de la etiqueta <connectors>.

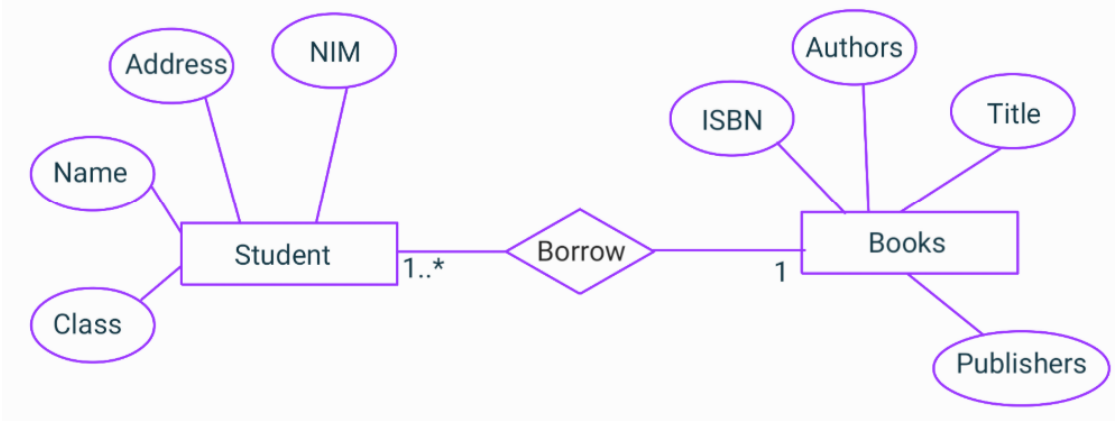


Figura 3.22: Diagrama ER [50].

```
<?xml version="1.0" encoding="windows-1252"?>
<xmi:XMI xmi:version="2.1" xmlns:uml="http://schema.omg.org/spec/UML/2.1"
xmlns:xmi="http://schema.omg.org/spec/XMI/2.1" xmlns:ERD="http://www.sparxsystems.com/profiles/ERD/1.0">
    <xmi:Documentation exporter="Enterprise Architect" exporterVersion="6.5"/>
    <uml:Model xmi:type="uml:Model" name="EA_Model" visibility="public">
    <xmi:Extension extender="Enterprise Architect" extenderID="6.5">
        <elements>
            <element xmi:idref="EAPK_193511AC_7ADC_45ec_B896_61B0479FF016">
            <element xmi:idref="EAPK_1BEB1EA3_9DCE_4a54_91EA_D02F113480DE" xmi:type="uml:Package" name="ERD View">
            <element xmi:idref="EAID_1D00AFEC_0385_4f99_9FE3_0EED91EE57A7" xmi:type="uml:Class" name="Address">
            <element xmi:idref="EAID_8D9DDE93_1B33_4c32_944E_E009AAB28971" xmi:type="uml:Class" name="Authors">
            <element xmi:idref="EAID_2657318E_0897_4218_B004_1AD359290315" xmi:type="uml:Class" name="Books">
            <element xmi:idref="EAID_31D5F802_929A_452f_853C_679C01FED0F6" xmi:type="uml:Class" name="Class">
            <element xmi:idref="EAID_AF405C88_85FF_48f6_BDCF_2003F076A015" xmi:type="uml:Class" name="ISBN">
            <element xmi:idref="EAID_C4AFAF7A_93D7_4de4_9C83_6C9CF44D3DBB" xmi:type="uml:Class" name="NIM">
            <element xmi:idref="EAID_450DC807_674E_4d4d_B218_3765EEFF6ADC" xmi:type="uml:Class" name="Name">
            <element xmi:idref="EAID_52ACA5DC_035F_4343_89D6_F67AF38B7B60" xmi:type="uml:Class" name="Publishers">
            <element xmi:idref="EAID_4AF44C38_57AE_4b8d_8AFF_5D0570433E41" xmi:type="uml:Class" name="Students">
            <element xmi:idref="EAID_5D3E9D93_849A_44a7_A9FA_78E56170C81E" xmi:type="uml:Class" name="Title">
        </elements>
        <connectors>...</connectors>
        <primitivetypes>...</primitivetypes>
```

Figura 3.23: Archivo XMI que consta de etiquetas XML [50].

- **Evaluación de la similitud del Diagrama ER utilizando el algoritmo de edición de árbol.** Tanto el diagrama ER del estudiante como el diagrama ER de la solución del profesor deben exportarse en un archivo XMI. El algoritmo compara ambos archivos en función de las etiquetas XML y proporciona como salida una puntuación o valor de similitud y comentarios sobre el diagrama ER del estudiante en función de la solución.
- **Sistema de calificación de diagramas ER con un enfoque de aprendizaje automático.** Se utilizarán diagramas de ER exportados en archivos XMI como datos de entrenamiento, así como una calificación proporcionada por un experto para cada dato de entrenamiento. Usando estos datos el proceso de aprendizaje automático dará como resultado un clasificador. Este se probará con algunos datos de prueba en archivos XMI para calificar los diagramas ER. La calificación obtenida del calificador se comparará con la calificación proporcionada por el experto y se actualizará la regla en el calificador para obtener el mejor resultado. Por último, se elegirá el mejor calificador para calificar automáticamente los diagramas ER. La imagen 3.24 muestra el proceso de calificación de diagramas ER usando un algoritmo de aprendizaje automático.

Singh, Anda y Tomovic (Parsing Structural and Textual Information from UML Class diagrams to assist in verification of requirement specifications)

En este proyecto [16] se ha utilizado un modelo de redes neuronales para detectar los diferentes componentes (clases y relaciones) de un diagrama UML y leer el texto de cada componente. El primer paso que han llevado a cabo es el preprocesamiento, que incluye el ajuste de tamaño, de brillo y contraste de la imagen, la eliminación de ruido y la detección de contornos. Posteriormente, han aplicado un detector de objetos basado en la arquitectura Faster-RCNN [1] sobre la imagen para detectar y localizar los diferentes componentes. A continuación, han usado redes neuronales recurrentes convolucionales [45] para analizar el texto de cada componente. Por último, han replicado esta misma estructura y texto, en formato digital aplicando una biblioteca de gráficos.

La imagen 3.25 muestra la arquitectura propuesta en este proyecto. Para entrenar este modelo han usado un conjunto de diagramas de clases UML dibujados a mano y otro conjunto creado de

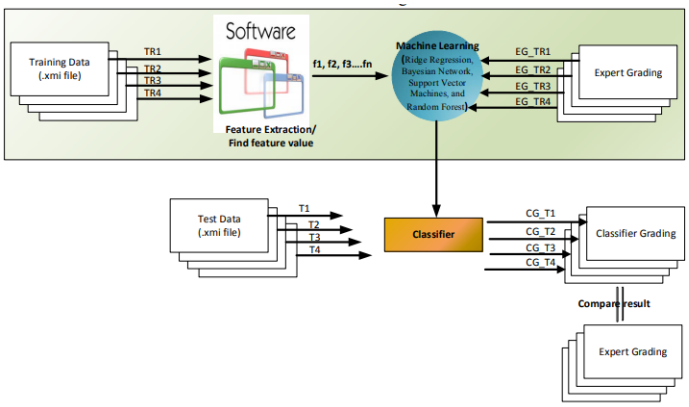


Figura 3.24: Proceso de calificación de diagramas ER usando un algoritmo de aprendizaje automático [50].

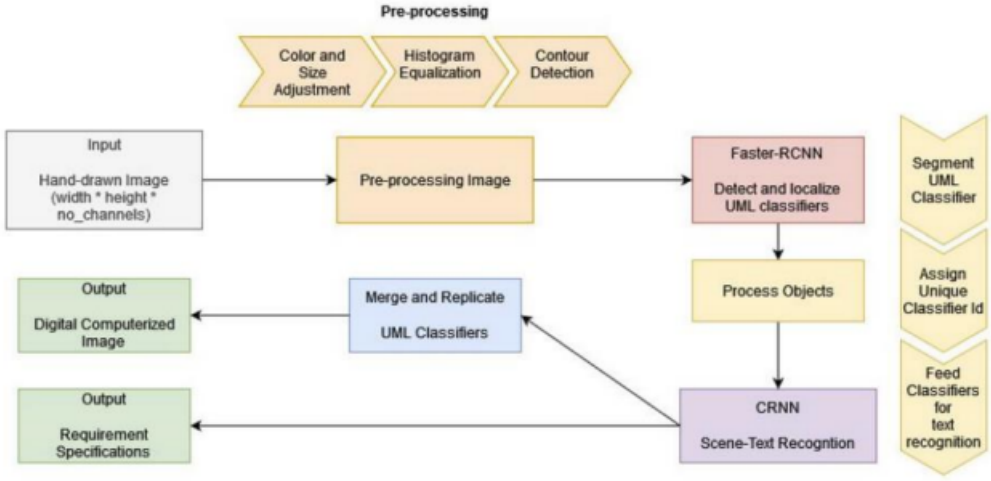


Figura 3.25: Arquitectura propuesta [16].

forma digital, mientras que para realizar la evaluación del modelo han utilizado 3 tipos diferentes de entradas, diagramas UML dibujados a mano, con tinta inteligente y de forma digital. Los resultados que obtuvieron indican que el modelo trabaja mejor con diagramas UML creados de forma digital, después con los de tinta inteligente y por último con los realizados a mano. Como conclusión, este modelo funciona muy bien en términos de detectar la mayoría de las clases de modelos (clases UML y herencia).

3.2.2. Análisis comparativo

En el análisis comparativo de los artículos seleccionados y nuestra propuesta, se evalúan ocho dimensiones consideradas clave en este estudio. Cada columna de la tabla 3.1 representa una dimensión específica, y cada celda indica si nuestra propuesta aborda dicha dimensión. Las dimensiones consideradas son:

- **Feedback inmediato:** Se refiere a la capacidad de la aplicación para proporcionar retro-

alimentación al estudiante de manera inmediata.

- **Diagramas Externos:** Evalúa si la aplicación incorpora diagramas ER externos en su funcionalidad.
- **Uso en exámenes:** Verifica si la herramienta se utiliza en la realización de exámenes.
- **Uso para practicar:** Examina si la aplicación se utiliza para prácticas en la creación de diagramas ER a partir de un enunciado.
- **Explicación de Conceptos:** Indica si la herramienta incluye explicaciones detalladas de conceptos durante su uso.
- **Generación de preguntas:** Evalúa si la aplicación genera preguntas relacionadas con el contenido para la creación de los diagramas ER.
- **Lectura de imágenes:** Verifica si la herramienta puede interpretar y analizar imágenes externas a ella.
- **Comparación de imágenes:** Indica si la aplicación realiza comparaciones entre diferentes imágenes.

Artículo	Feedback inmediato	Diagramas externos	Uso en examenes	Uso para practicar	Explicacione de conceptos	Genera preguntas	Leer imágenes	Comparar imágenes
Schildgen (2020)	**SI**	**NO**	**NO**	**SI**	**SI**	**SI**	**NO**	**NO**
Foss, Urazova y Lawrence (2022)	**SI**	**NO**	**NO**	**SI**	**NO**	**SI**	**NO**	**NO**
Jaimez-González y Martínez-Samora (2020)	**SI**	**NO**	**NO**	**SI**	**NO**	**NO**	**NO**	**NO**
Simanjuntak (2015)	**SI**	**SI**	**SI**	**SI**	**NO**	**NO**	**NO**	**NO**
Singh, Anda y Tomovic (2022)	**NO**	**SI**	**NO**	**NO**	**NO**	**NO**	**SI**	**NO**
Nuestra propuesta	**SI**	**SI**	**NO**	**SI**	**NO**	**NO**	**SI**	**SI**

Tabla 3.1: Dimensiones comparativas.

En los primeros 4 estudios se utilizan sistemas web para crear y/o corregir diagramas de ER. Las dos grandes ventajas que hemos encontrado en estos sistemas son la generación de un feedback inmediato al estudiante y la posibilidad de realizar más ejercicios para practicar antes de un examen. En el artículo [16] se propone una aplicación que permite calificar los exámenes de

estudiantes de bases de datos, además de poderse usar también para practicar subiendo ejercicios de diagramas ER.

Un aspecto a destacar es que en ninguna de estas aplicaciones se realiza una comparación de imágenes de diagramas ER. De hecho, los tres primeros sistemas crean el diagrama ER en la misma aplicación y en el último se exporta el diagrama ER en un fichero XMI. Es decir, todos trabajan con información estructurada que describe el diagrama ER.

La principal dificultad al comparar dos diagramas ER radica en la posibilidad de encontrar representaciones diferentes para los mismos elementos, todas ellas siendo correctas desde el punto de vista técnico. Para evitar este inconveniente, los dos primeros proyectos van haciendo preguntas o permiten que el usuario interactúe con el texto y es la propia aplicación la que va añadiendo los diferentes componentes al diagrama ER. Además, se ha estudiado un quinto artículo 3.2.1, en el que se crea, entrena y prueba un modelo de redes neuronales para la detección y lectura de los diferentes componentes de un diagrama UML, este puede haber sido creado a mano, con tinta inteligente o de forma digital.

Nosotros proponemos en este proyecto el desarrollo de una aplicación web que permita cargar imágenes con modelos Entidad-Relación (ER). Este sistema estará diseñado para analizar y comprender los distintos componentes presentes en el diagrama ER cargado. Posteriormente, se llevará a cabo una comparación entre el diagrama ER proporcionado por el estudiante y el del profesor, generando así un feedback inmediato para el estudiante. Nuestra propuesta se distingue al integrar las ventajas identificadas de los diferentes artículos estudiados, destacándose por la incorporación de la comparación de imágenes, un aspecto que no ha sido abordado por ninguna de las propuestas seleccionadas.

3.3. Aprendizaje automático

Esta sección se inicia con una breve introducción 3.3.1 a la Inteligencia Artificial (IA) y al aprendizaje automático, continua con una explicación de las redes neuronales 3.3.2, el aprendizaje profundo 3.3.3 y el aprendizaje por transferencia 3.3.4.

3.3.1. Introducción

Introducción a la Inteligencia Artificial (IA)

La IA "es un campo de la informática que se enfoca en crear sistemas que puedan realizar tareas que normalmente requieren inteligencia humana, como el aprendizaje, el razonamiento y la percepción [42]." A lo largo de la historia se han considerado 4 enfoques distintos:

- **Sistemas que actúan como humanos:** Se enfocan en crear sistemas que puedan realizar tareas de forma parecida a los humanos. Este enfoque incluye la implementación de sistemas capaces de razonar, planificar, aprender, comprender y comunicarse de manera similar a las personas [7]..
- **Sistemas que piensan como humanos:** Se centran en la imitación de la capacidad de los humanos para razonar y tomar decisiones. Este enfoque se suele llevar a cabo mediante experimentos psicológicos [7]..
- **Sistemas que piensan racionalmente:** Estos sistemas se concentran en el pensamiento racional y el cálculo lógico [7]..

- **Sistemas que actúan racionalmente:** Se trata de sistemas que toman decisiones de manera racional con el objetivo de maximizar la probabilidad de alcanzar sus objetivos. Estos sistemas aprenden y mejoran a partir de un conjunto de datos y experiencias. Algunos ejemplos que pertenecen a este enfoque son: sitemas de conducción autónoma, sistemas de diagnóstico médico, sistemas de análisis de imágenes por satélite, aspirador Roomba...[7]

Introducción al aprendizaje automático

El aprendizaje automático o *machine learning* es "la rama de la Inteligencia Artificial que se preocupa del diseño y desarrollo de algoritmos que permiten a los ordenadores mejorar su desempeño en la realización de una tarea a partir de la experiencia [7]."

El aprendizaje automático se puede utilizar en múltiples tareas: clasificación, clustering, reconocimiento de patrones, predicción, construcción de bases de conocimiento a partir de la experiencia, minería de datos, resolución de problemas, planificación y acción, detección/reconocimiento de anomalía, generación de patrones.

Existen diferentes tipos de aprendizaje:

- **Aprendizaje supervisado *(supervised learning)*:** El objetivo de este tipo de aprendizaje es entrenar modelos utilizando un conjunto de datos etiquetados para predecir o clasificar datos no observados previamente. En la programación tradicional introducíamos unos datos y un programa a un ordenador y este nos devolvía una salida. Sin embargo, en el aprendizaje automático lo que vamos a propocionarle es un conjunto de datos de entrada y otro de salida y el ordenador genera como salida un modelo (programa) [65].

- **Aprendizaje no supervisado *(unsupervised learning)*:** Consiste en visualizar, preprocesar o encontrar subgrupos sin utilizar ningún tipo de etiquetado en los datos. Existen dos tipos de aprendizaje no supervisado: *clustering* o agrupamiento (consiste en buscar agrupaciones entre los datos) y reducción dimensional (se basa en reducir la dimensionalidad de los datos sin perder información) [64].

- **Aprendizaje por refuerzo *(reinforcement learning)*:** La máquina aprende a partir de su propia experiencia, interaccionando con el entorno hasta dar con el comportamiento ideal. A partir de la información disponible, emprenderá acciones que repetirá y reforzará según las recompensas que obtenga, que pueden ser positivas o negativas. [4] La imagen 3.26 muestra un ejemplo de aprendizaje por refuerzo.

- **Aprendizaje semi-supervisado *(semi-supervised learning)*:** En este tipo de aprendizaje se tiene tanto datos etiquetados como datos sin etiquetar [6]. La imagen 3.27 muestra un ejemplo de aprendizaje semi-supervisado.

- **Aprendizaje por transferencia *(Transfer Learning)*:** Se utilizan patrones o modelos previamente entrenados para otro modelo para resolver un problema específico [68]. La imagen 3.28 presenta un ejemplo de aprendizaje por transferencia.

La imagen 3.29 muestra un ejemplo empleando un aprendizaje supervisado y uno de aprendizaje no supervisado. En este ejemplo, contamos con un conjunto de datos formado por cuadrados y triángulos rojos y azules. En el caso del aprendizaje supervisado se enseña a la máquina a clasificar formas según la cantidad de lados y los valores de píxeles asociados al color. Por ejemplo, se le

Figura 3.26: Aprendizaje por refuerzo.[3]

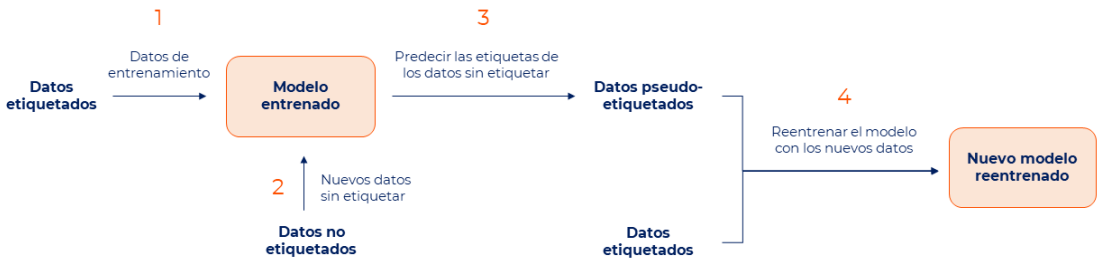
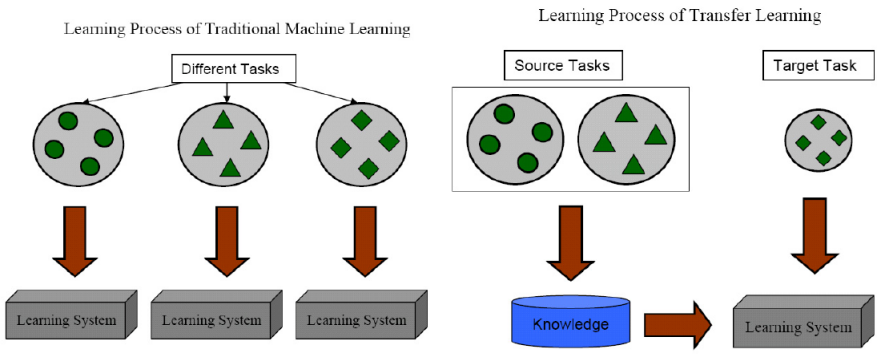


Figura 3.27: Aprendizaje semi-supervisado. [60]

indica que las formas con cuatro lados son cuadrados y se asignan etiquetas como “rojo” o “azul”. En el aprendizaje no supervisado, el algoritmo examina la similitud entre las formas del conjunto de datos, posiblemente considerando el número de lados o la coincidencia de color. Aunque no tiene información sobre los nombres específicos de las formas, el algoritmo agrupa los objetos con características similares y les asigna etiquetas propias. Esto permite que el algoritmo organice y clasifique las formas de manera autónoma, identificando patrones.



(a) Traditional Machine Learning (b) Transfer Learning

Figura 3.28: Aprendizaje por transferencia. [20]

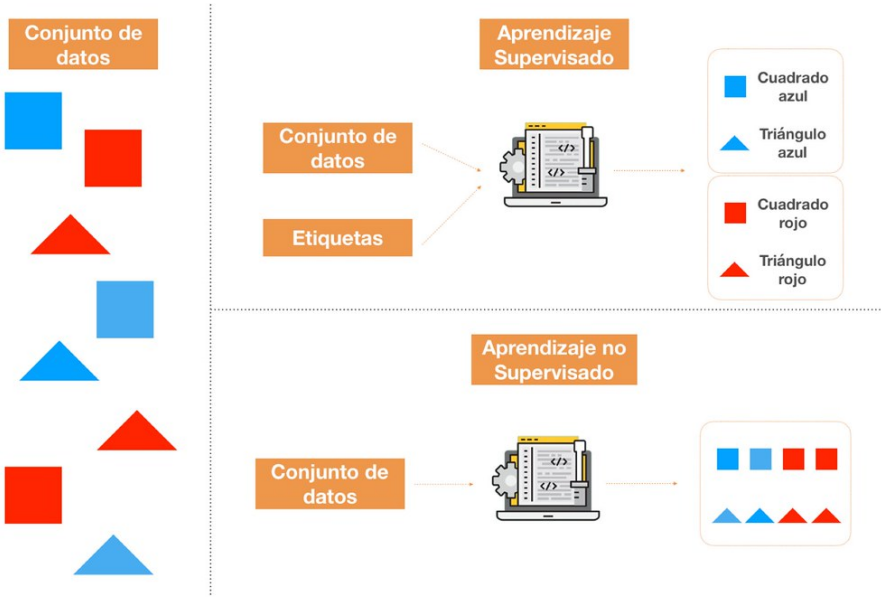


Figura 3.29: Aprendizaje supervisado vs aprendizaje no supervisado.[19]

3.3.2. Redes neuronales

Las redes neuronales son modelos matemáticos inspirados en la neurona biológica. Se componen de un conjunto de unidades interconectadas llamadas "neuronas", que trabajan de forma conjunta para procesar información. Una neurona es la unidad de procesamiento básica de una red neuronal y consiste en un conjunto de dentritas que reciben información de otras neuronas o del exterior. Cada conexión tiene un valor de entrada y está caracterizada por un peso. Una neurona artificial tiene una función sumatoria asociada que calcula la suma de las entradas (cuerpo de la neurona) y una función de activación que limita la amplitud de la salida de la neurona (simula el disparo de la neurona). En la imagen 3.30 podemos ver la gran similitud entre una neurona biológica (a la izquierda) y una artificial (a la derecha).

Es uno de los métodos más exitosos dentro de los algoritmos de aprendizaje supervisado.

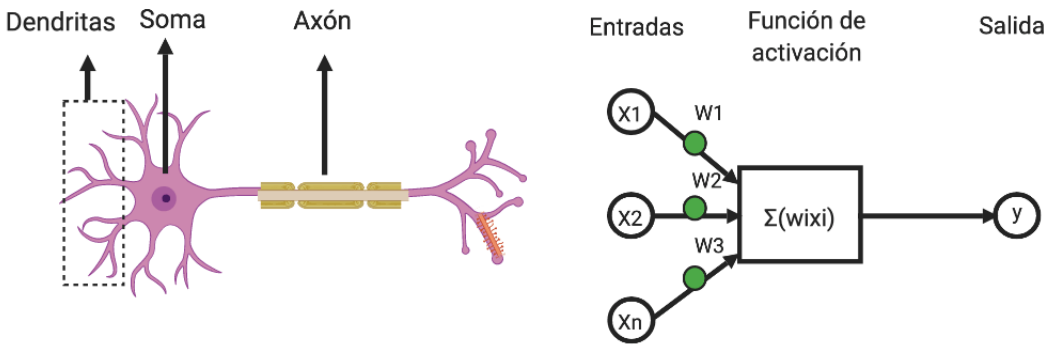


Figura 3.30: Neurona biológica vs neurona artificial. [15]

3.3.3. Aprendizaje profundo

El aprendizaje profundo es una técnica de aprendizaje automático basada en el uso de redes neuronales artificiales con múltiples capas para procesar y aprender patrones complejos a partir de grandes conjuntos de datos. En una red multicapa, las unidades se estructuran en capas, en las que las neuronas de cada capa reciben su entrada de la salida de las neuronas de la capa anterior. Podemos aumentar la capacidad expresiva de la red combinando neuronas en diferentes capas. Hay 3 tipos de capas en este tipo de redes:

- **Capa de entrada:** Es la capa en la que se sitúan las neuronas de entrada, las cuales reciben los datos de entrada que alimenta a la red neuronal. [36]
- **Capa de salida:** Es la capa formada por las unidades que generan los resultados.
- **Capas ocultas:** Son las capas en las que las neuronas no son ni de entrada ni de salida. Se denominan de esta manera, ya que a día de hoy desconocemos cuáles son sus datos de entrada y de salida.

La figura 3.31 muestra una red neuronal con 1 capa de entrada, 1 de salida y 1 capa oculta.

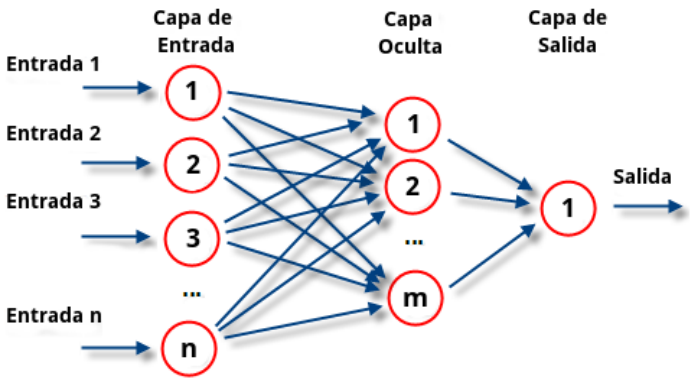


Figura 3.31: Red neuronal multicapa.[39]

Redes convolucionales (CNN)

Las Redes Neuronales Convolucionales o *Convolutional Neural Networks* (CNN) [44] son un tipo de red neuronal artificial especialmente diseñada para procesar datos que tienen una estructura de malla, como imágenes, vídeos o señales de audio. Las CNN utilizan una operación matemática llamada convolución para extraer características relevantes de los datos de entrada y aprender a reconocer patrones en ellos. Son particularmente útiles en tareas de visión por computadora, como clasificación de imágenes, detección de objetos y segmentación semántica, revolucionando el campo de la visión por computadora y permitiendo aplicaciones prácticas como la detección facial, la clasificación de imágenes en redes sociales y la conducción autónoma en vehículos.

En una Red Neuronal Convolucional (CNN), el proceso de reconocimiento de patrones y características en una imagen se lleva a cabo a través de tres capas fundamentales [29]:

- **Capa Convolucional:** La primera capa de una Red Neuronal Convolucional (CNN) es la capa convolucional, fundamental para detectar características simples en los datos de imagen, voz o audio. Puede ser seguida por capas adicionales de convolución o capas de agrupación. Su función principal es aplicar filtros, generalmente matrices de 3x3, a regiones de la imagen mediante operaciones de convolución. Estos filtros buscan capturar características locales como colores o bordes, generando mapas de características que destacan patrones esenciales en la imagen.

- **Transformación ReLu:** Después de la convolución, se aplica una transformación ReLu (*Rectified Linear Unit*) al mapa de características resultante. Esta transformación introduce no linealidades al modelo, permitiendo que la CNN aprenda patrones más complejos y representaciones más abstractas de los datos. La capa convolucional, junto con la transformación ReLu, convierte la información visual en valores numéricos, facilitando la interpretación y extracción de patrones por parte de la red neuronal.

- **Capa de Agrupación (*Pooling*):** su función principal es reducir la dimensionalidad de los mapas de características obtenidos. Utilizando filtros, esta capa selecciona valores representativos, disminuyendo la complejidad del modelo y evitando el sobreajuste. La reducción de dimensionalidad mejora la eficiencia computacional al limitar la cantidad de parámetros en la entrada, conservando las características esenciales aprendidas en la capa convolucional.

- **Capa Totalmente Conectada:** constituye la etapa final de la CNN, realizando la clasificación basada en las características extraídas. Cada nodo en esta capa está directamente vinculado a todos los nodos de la capa anterior, permitiendo una integración completa de la información. Utilizando una función de activación softmax, esta capa asigna probabilidades a las clases, logrando una clasificación precisa de las entradas. Es en esta fase donde se completa el proceso de reconocimiento de patrones y características, permitiendo que la CNN distinga y clasifique eficientemente objetos o elementos en la entrada.

En la figura 3.32 encontramos un ejemplo de red convolucional con una entrada de una imagen de un coche, a la que realizando diferentes capas de convoluciones y agrupaciones y una capa totalmente conectada (que se trata de una red neuronal multicapa) se clasifica la imagen finalmente como un coche.

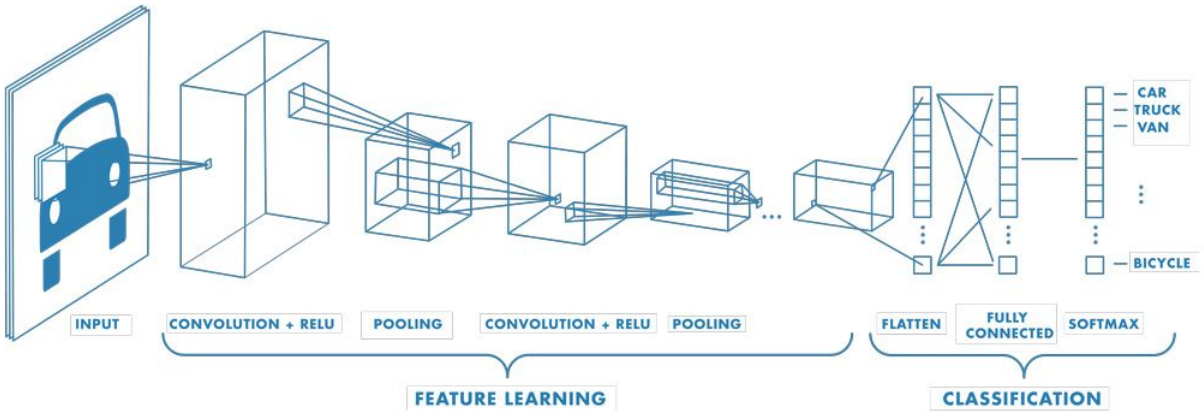


Figura 3.32: Ejemplo de CNN [69]

3.3.4. Aprendizaje por transferencia

Tokio School [13] define el aprendizaje por transferencia o "transfer learning" como una técnica de aprendizaje automático que permite aprovechar el conocimiento adquirido por un modelo en una tarea concreta, para facilitar el entrenamiento de un modelo distinto en la misma o en una tarea diferente. Esta técnica consiste en transferir las características aprendidas por el primer modelo al segundo, por ejemplo insertando fragmentos de la red neuronal en el nuevo modelo o copiando su configuración interna, siempre y cuando estas características sean de utilidad para realizar la nueva tarea para la que se desarrolla el segundo modelo. Se utiliza de forma habitual en campos como el procesamiento de lenguaje natural (por ejemplo, en traducción automática) y la visión artificial. La siguiente imagen 3.33 muestra las diferencias entre el enfoque tradicional y el enfoque de Transfer Learning.

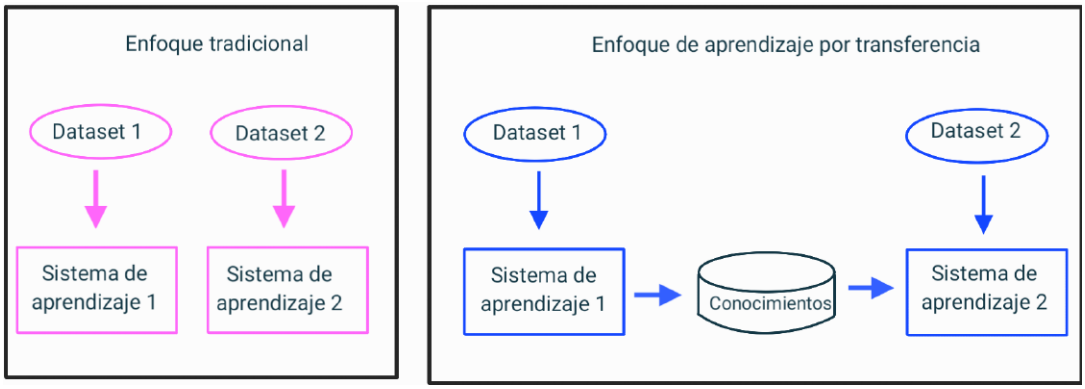


Figura 3.33: Enfoque tradicional vs. enfoque de Transfer Learning [67]

Aplicaciones del Transfer Learning para la resolución de problemas de Deep Learning

Según [67] Transfer Learning se aplica fundamentalmente de dos maneras:

1. **Utilización de modelos pre-entrenados como extractores de características:** Consiste en aprovechar redes neuronales ya entrenadas, excepto la capa final, utilizando las ca-

pas previas para extraer características útiles que pueden ser aplicadas a diferentes tareas. Por ejemplo, utilizar las primeras capas de un modelo como AlexNet (entrenado en ImageNet) para identificar diferentes especies de flores a partir de las imágenes. Este método demuestra un buen rendimiento en la práctica.

2. **Ajustes de modelos pre-entrenados:** Esta técnica consiste en reemplazar no solo la última capa de un modelo pre-entrenado, sino re-entrenar de manera selectiva las otras capas. Algunas capas se mantienen fijas mientras se ajustan las capas restantes para adaptarse al problema especifíco en cuestión. Esto permite aprovechar la estructura de la red y sus conocimientos, logrando de esta forma mejores resultados en menos tiempo de entrenamiento.

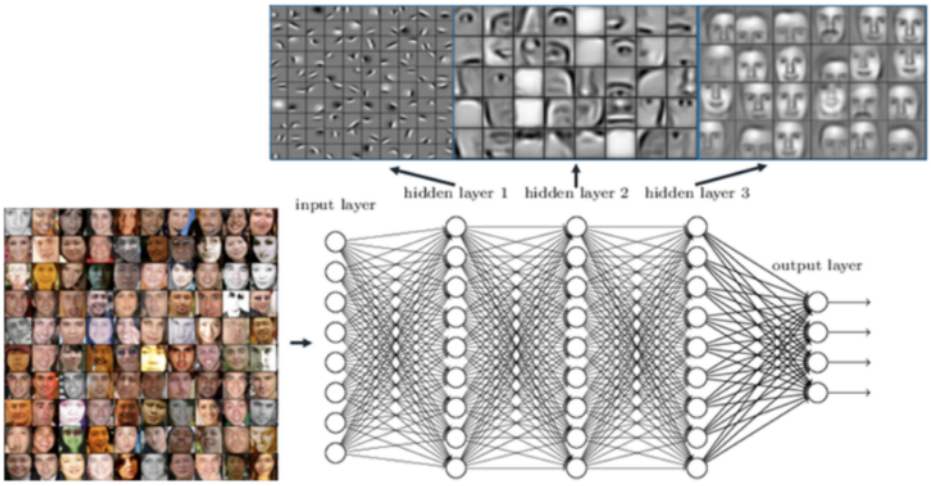


Figura 3.34: Ejemplo de modelo de red de neuronas utilizado para reconocimiento facial.[67]

3.4. Visión computacional

La detección de objetos es una rama de la visión computacional, que se centra en la aplicación de algoritmos y estrategias de aprendizaje automático para identificar patrones en las imágenes, permitiendo así la detección de objetos específicos.Tiene aplicaciones diversas, desde sistemas de vigilancia hasta la conducción autónoma y el análisis médico de imágenes.

Para llevar a cabo esta tarea, se emplean algoritmos y técnicas de aprendizaje automático para reconocer patrones en las imágenes y así identificar los objetos específicos de interés. En los últimos tiempos, han surgido diversos enfoques y modelos como Faster R-CNN, YOLO y SSD. En este proyecto hemos decidido usar YOLO.

Descripción de la arquitectura YOLO para la detección de objetos.

YOLO *(You Only Look Once)* es un algoritmo de código abierto que utiliza redes neuronales convolucionales (CNN) 3.3.3 para detectar en una única ejecución, como su propio nombre indica, varios objetos en tiempo real en una imagen. Se trata de un algoritmo con una gran

velocidad de detección de objetos en las imágenes, debido a que puede predecirlos en tiempo real y los resultados que genera son precisos con errores de fondo mínimos. Además, este algoritmo cuenta con una gran capacidad de aprendizaje, lo que significa que es muy bueno identificando y entendiendo cómo son los objetos para después usar esa información en la detección de esos objetos en otras imágenes o videos. [25]

YOLO utiliza tres técnicas: bloques residuales, regresión del cuadrado delimitados e intersección sobre unión (IOU).

Bloques residuales Primero la imagen se divide en varias cuadrículas de la misma dimensión SxS. Cada una de estas cuadrículas es responsable de localizar y predecir la clase del objeto que abarca, junto con el valor de probabilidad/confianza. [25]. La figura 3.35 muestra cómo una imagen de entrada se divide en cuadrículas de 13x18.

Figura 3.35: Imagen de entrada al algoritmo YOLO dividida en cuadrículas. [46]

Regresión del cuadro delimitador Un cuadro delimitador (*bounding box* es un contorno o borde que destaca un objeto específico dentro de una imagen [40]. Cada cuadro delimitador de la imagen consta de los siguientes atributos: ancho(b_w), altura(b_h), clase (representada por la letra c) y centro del cuadrado delimitador (b_x,b_y). La imagen 3.36 muestra un ejemplo de un cuadro delimitador de un coche [25]. En esta imagen se representa además la probabilidad o confianza (p_c) de que el objeto realmente esté dentro del cuadro delimitador.

Intersección sobre unión (IOU) IOU es una métrica que evalúa el grado de coincidencia entre dos áreas. Durante el entrenamiento, el modelo puede generar diferentes ventanas alrededor de un objeto, por lo que para determinar la ventana óptima se calcula su valor de IOU con respecto a la ventana de referencia (anotada manualmente). Esta métrica relaciona el tamaño del área común a ambas ventanas (intersección) con el tamaño del área ocupada por la unión de las dos ventanas. Si la coincidencia entre ambas ventanas es perfecta, el valor del IOU es 1, puesto que tanto la intersección como la unión de las áreas de ambas ventanas es igual al área de cualquiera de ellas. Cuanto más diferentes sean las ventanas, menor será su valor, hasta llegar a 0. Entre todas las ventanas generadas por YOLO, se escoge aquella con un valor máximo de IOU, puesto que es la que abarca un mayor grado del objeto detectado con el área total más pequeña[25]. En la figura 3.37 hay dos cuadros delimitadores. El azul se corresponde con el previsto, mientras

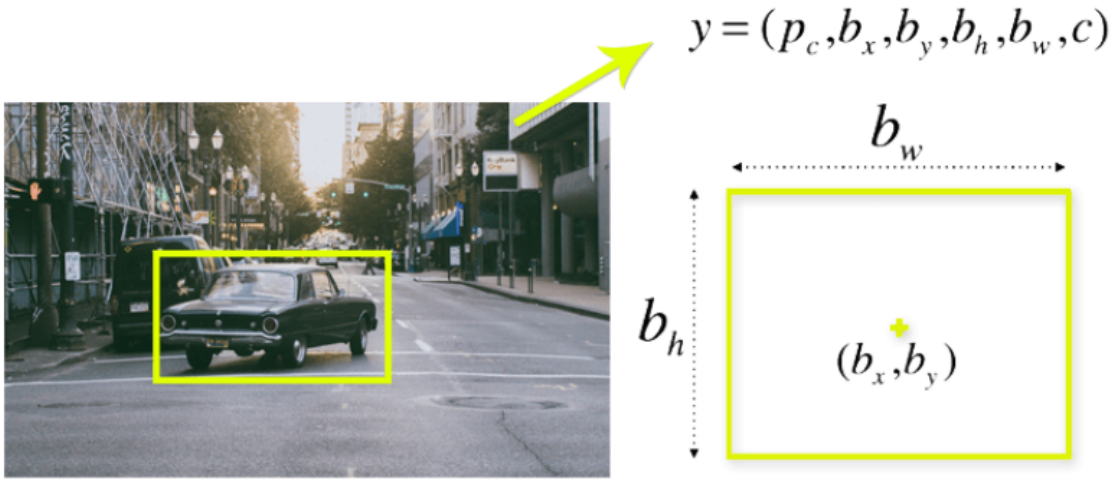


Figura 3.36: Ejemplo de cuadro delimitador. [46]

el verde con el real. El algoritmo YOLO nos asegura que ambos cuadros delimitadores sean idénticos.

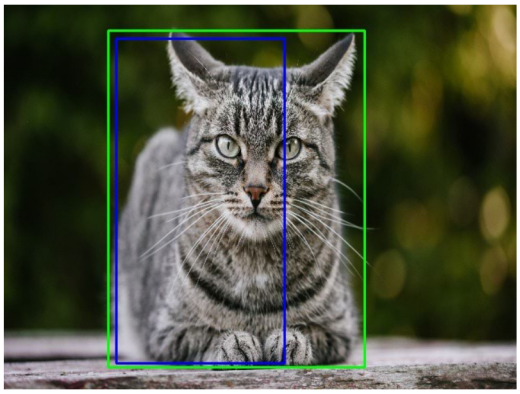


Figura 3.37: Ejemplo de intersección sobre unión (IOU). [46]

Combinación de las tres técnicas YOLO [24] utiliza una combinación de las tres técnicas explicadas anteriormente para generar los resultados finales de detección. La imagen 3.38 proporciona una representación visual del proceso de detección de objetos utilizando el algoritmo YOLO. Para detectar los objetos de esta imagen (perro, bicicleta y coche), el proceso se divide en cuatro etapas:

- **División en cuadrículas:** La imagen original se divide en una cuadrícula de 7x7 regiones, como se muestra en la parte izquierda de la figura.
- **Generación de cuadros delimitadores y confianza asociada:** Para cada región de la

cuadrícula, se superponen cuadros delimitadores potenciales, representados por los rectángulos en la parte superior central de la figura. Además, se calcula la confianza asociada a cada región, que indica la probabilidad de que un objeto esté presente dentro de ese cuadro delimitador. Los cuadros delimitadores con una alta confianza son considerados como detecciones válidas de objetos.

- **Mapa de probabilidad de clases:** De manera simultánea, YOLO genera un mapa de probabilidad de clases para cada región de la cuadrícula. Este mapa indica la probabilidad de presencia de objetos específicos dentro de cada región. Por ejemplo, en la parte inferior central de la figura, se muestra un mapa de probabilidad de clases donde las regiones con mayor probabilidad de contener un perro están resaltadas en azul, las de bicicletas en amarillo y las de coches en rosa.

- **Detecciones finales:** Finalmente, YOLO combina la información de los cuadros delimitadores y los mapas de probabilidad de clases para generar las detecciones finales de objetos. En la parte derecha de la figura, se muestran los objetos detectados, delimitados por cuadros coloreados según su clase (azul para el perro, amarillo para la bicicleta y rosa para el coche). Esto proporciona una representación completa del proceso de detección de objetos mediante YOLO.

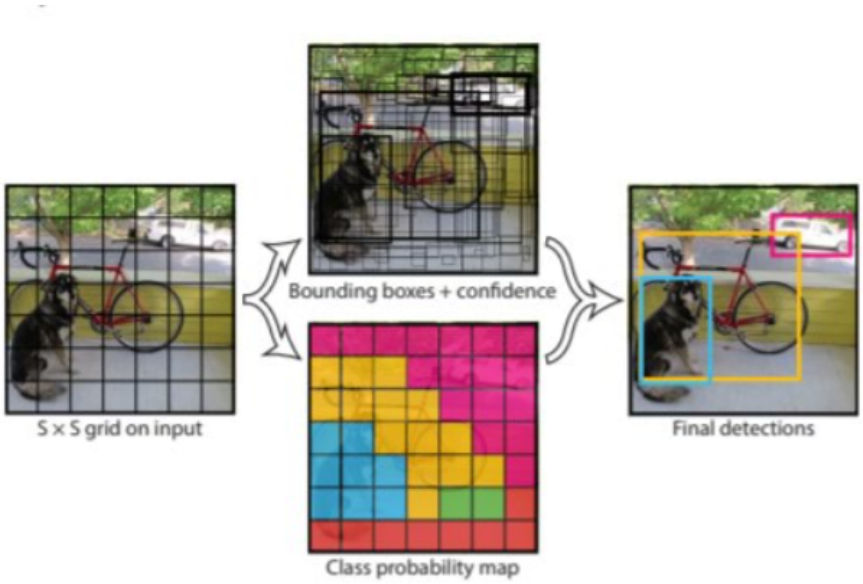


Figura 3.38: Ejemplo de detección de objetos usando YOLO. [46]

3.5. Reconocimiento óptico de caracteres (OCR)

El reconocimiento óptico de caracteres (OCR) es un proceso a través del cual se convierte una imagen con texto en un formato de texto que puede ser interpretado por las máquinas. [66] En

nuestro caso, vamos a utilizar el OCR para identificar los nombres de los diferentes componentes de las imágenes de los diagramas ER.

El proceso de ejecución de OCR comprende los siguientes pasos:

1. **Adquisición de imagen:** Se digitaliza la imagen y se convierte en datos binarios. El software de OCR analiza la imagen identificando áreas claras como fondo y áreas oscuras como texto. [66]

2. **Procesamiento previo:** El software limpia la imagen para corregir errores y prepararla para su lectura. Esto incluye enderezar o inclinar la imagen, eliminar manchas o suavizar bordes, limpiar cuadros y líneas, y reconocer guiones para la tecnología OCR multilingüe [66].

3. **Reconocimiento de texto:** Hay dos enfoques principales para reconocer texto en OCR: coincidencia de patrones y extracción de características. [66]

   a) **Coincidencia de patrones:** Aísla cada imagen de carácter del texto y lo compara con una lista de caracteres que tiene guardadas. Este tipo de reconocimiento de caracteres solo funciona correctamente si la escala y la fuente de ambos caracteres es parecida. Por ello este método solo es eficaz para documentos escritos en fuentes conocidas. [66]

   b) **Extracción de características:** Descompone los caracteres en líneas, circuitos cerrados y otras características. Luego, utiliza estas para encontrar la mejor coincidencia entre los caracteres almacenados [66].

4. **Postprocesamiento:** Después del análisis, el sistema convierte el texto extraído en un archivo digital. Algunos sistemas de OCR pueden generar archivos PDF con anotaciones que muestran versiones previas y posteriores de la imagen escaneada [66].

Para realizar el reconocimiento óptico de caracteres en nuestro proyecto, hemos estudiado dos herramientas: la librería Tesseract de Python [59] y la API de Google Vision [8]. Tras realizar pruebas con ambas, hemos observado que la API de Google Vision nos daba mejores resultados. Y por ello hemos decidido implementar la parte de OCR de nuestro proyecto con esta herramienta.

3.5.1. API de Google Vision

La API de Google Vision [8] proporciona dos métodos distintos para realizar reconocimiento óptico de caracteres (OCR):

- Procesando una imagen: la API de Google Vision es capaz de detectar y extraer todo el texto de la imagen. El docunento *JSON* que devuelve contiene las diferentes palabras obtenidas como cadenas de texto, junto las coordenadas de sus cuadros delimitadores. [9]

- Procesando un documento: esta API detecta no solo las palabras, sino también la página, el bloque, el párrafo y la división en la que se encuentra. El JSON que devuelve contiene toda esta información. [9]

Figura 3.39: Ejemplo de detección de texto de una imagen usando la API de Google Vision. [9]

Figura 3.40: Ejemplo de detección de texto de un documento utilizando la API de Google Vision. [9]

Parte II

Desarrollo de la propuesta

Capítulo 4

Descripción de la propuesta

En este capítulo se presentan las etapas de análisis (sección 4.1) y diseño (sección 4.2) del software y, además, se incluye una explicación detallada de cómo se ha construido el "*dataset*" (sección 4.3) necesario para el entrenamiento y evaluación del modelo de aprendizaje-automático que se planteará en el Capítulo 5.

4.1. Análisis

"El análisis de requisitos es el proceso de identificar, definir y documentar los requisitos de un sistema de software. El objetivo del análisis de requisitos es identificar las necesidades del usuario y traducirlas en requisitos específicos, medibles y alcanzables que el equipo de desarrollo de software pueda utilizar para diseñar y desarrollar el sistema"[2]. Se trata de un proceso iterativo durante el ciclo de vida del proyecto para afrontar posibles cambios conforme avanza el proyecto.

En esta sección se presenta la especificación y modelados de los diferentes tipos de requisitos utilizados en este proyecto.

4.1.1. Actores

Los actores son los usuarios que tendrán acceso a la aplicación del proyecto. En nuestro caso contamos con 2 tipos de actores: los estudiantes y los profesores. En la tabla 4.1 se presenta la descripción de cada uno de los actores.

ID	Actor	Descripción
A-01	Profesor	Este actor representa al profesor de una asignatura de bases de datos, que tiene la responsabilidad de proporcionar a sus estudiantes supuestos prácticos que les permitan consolidar su aprendizaje sobre diseño conceptual.
A-02	Estudiante	Este actor representa a los estudiantes de una asignatura de bases de datos y cuyo objetivo principal es aprender a realizar correctamente el diseño conceptual de una base de datos.

Tabla 4.1: Actores.

4.1.2. Requisitos de usuario

Los requisitos de usuario describen las tareas que los diferentes actores pueden realizar en la aplicación. Habitualmente su modelado y especificación se lleva a cabo en forma de casos de uso (CU).

La imagen 4.1 muestra el diagrama de casos de usos de nuestra aplicación*DBReader*. Este está formado por 5 casos de uso que describen la interacción de los actores con el sistema desarrollado. Por un lado, el profesor registra un supuesto práctico. Mientras que el estudiante registrar una entrega, la cual se evalúa y puede visualizar el feedback. En ambos casos, se realiza una validación de los modelos cargados.

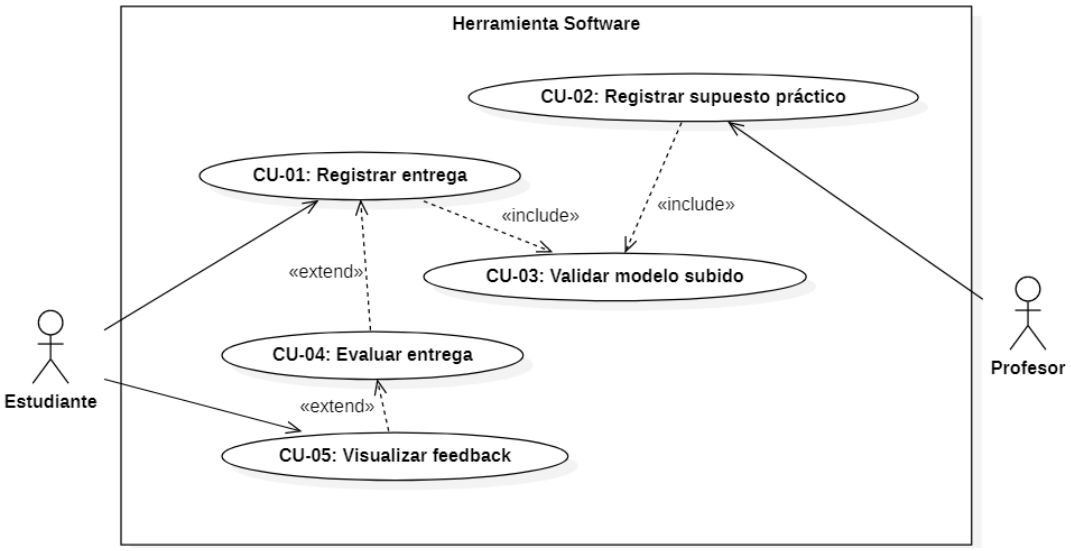


Figura 4.1: Diagrama de casos de uso.

A continuación, se muestran las especificaciones de los diferentes casos de uso identificados en la imagen 4.1. La tabla 4.2 contiene la especificación del CU-01 (registrar entrega), la 4.3 la del CU-02 (registrar supuesto práctico), la 4.4 la del CU-03 (validar modelo subido), la 4.5 la del CU-04 (evaluar entrega) y la 4.6 la del CU-05 (visualizar feedback).

CU-01	Registrar entrega.	
Descripción	Describe el proceso mediante el cual el estudiante sube un diagrama ER como solución a un problema.	
Autor	Estudiante.	
Actor	Analista de datos.	
Secuencia normal	Paso 1	El estudiante solicita registrar una nueva entrega.
	Paso 2	El sistema registra la nueva imagen.
	Paso 3	El sistema hace una inclusión al CU-03 (validar entega) y una extensión al CU-04 (visualizar feedback).
	Paso 4	El caso de uso finaliza satisfactoriamente.
Excepciones	E 1	El caso de uso vuelve al paso 4 si el formato de la imagen no se ajusta a lo especificado en el RNF-01 (4.8).

Tabla 4.2: Especificación CU-01.

CU-02	Registrar supuesto práctico	
Descripción	Describe el proceso mediante el cual el profesor sube el título del ejercicio y la solución al mismo.	
Autor	Analista de datos.	
Actor	Profesor.	
Secuencia normal	Paso 1	El profesor solicita registrar un nuevo supuesto práctico.
	Paso 2	El sistema solicita al profesor el título del supuesto práctico.
	Paso 3	El profesor proporciona el título solicitado.
	Paso 4	El sistema solicita al profesor que proporcione la imagen que contiene el modelo ER que se usará como solución de referencia.
	Paso 5	El profesor proporciona la imagen solicitada (cuyo formato se ajustará al especificado en el RNF-01).
	Paso 6	El sistema registra la nueva imagen.
	Paso 7	El sistema hace una inclusión al CU-03 (validar entrega).
	Paso 8	El caso de uso finaliza satisfactoriamente.
Excepciones	E 1	El caso de uso vuelve al paso 3 si el título proporcionado por el profesor ya ha sido asignado a otro supuesto práctico.
	E 2	El caso de uso vuelve al paso 5 si el formato de la imagen no se ajusta a lo especificado en el RNF-01 (4.8).

Tabla 4.3: Especificación CU-02.

CU-03	Validar modelo subido.	
Descripción	Describe el proceso mediante el cual el sistema valida el modelo subido por el estudiante o el profesor	
Autor	Estudiante o profesor.	
Actor	Analista de datos.	
Secuencia normal	Paso 1	El sistema valida el modelo subido por el estudiante o por el profesor.
	Paso 2	El caso de uso finaliza satisfactoriamente.

Tabla 4.4: Especificación CU-03.

CU-04	Evaluar feedback.	
Descripción	Describe el proceso mediante el cual el sistema realiza la evaluación de la entrega.	
Autor	Estudiante.	
Actor	Analista de datos.	
Secuencia normal	Paso 1	El sistema recibe el modelo del estudiante.
	Paso 2	El sistema evalúa el modelo.
	Paso 3	El caso de uso finaliza satisfactoriamente.

Tabla 4.5: Especificación CU-04.

CU-05	Visualizar feedback.	
Descripción	Describe el proceso mediante el cual el estudiante visualiza el feedback generado.	
Autor	Estudiante.	
Actor	Analista de datos.	
Secuencia normal	Paso 1	El estudiante solicita visualizar el feedback generado.
	Paso 2	El sistema muestra el feedback generado.
	Paso 3	El caso de uso finaliza satisfactoriamente.

Tabla 4.6: Especificación CU-05.

4.1.3. Requisitos funcionales

Un requisito funcional es una declaración de cómo debe comportarse un sistema. Define lo que el sistema debe hacer para satisfacer las necesidades o expectativas del usuario. Los requisitos funcionales se pueden considerar como características que el usuario detecta [43]. La tabla 4.7 muestra los requisitos funcionales de esta aplicación.

ID	Requisitos Funcionales
RF-01	El profesor podrá subir el título y la solución a un ejercicio.
RF-02	El estudiante podrá consultar la lista de ejercicios propuestos.
RF-03	El estudiante podrá subir un diagrama ER como solución a un ejercicio.
RF-04	El sistema evaluará cada entrega.
RF-05	El sistema validará cada modelo subido.
RF-06	Se emitirá un feedback por cada ejercicio resuelto.

Tabla 4.7: Requisitos funcionales.

4.1.4. Requisitos no funcionales

Los requisitos no funcionales son características y criterios que describen cómo debe ser el rendimiento, la seguridad, la usabilidad y otros aspectos de un sistema o software más allá de su funcionalidad básica [26]. La tabla 4.8 muestra los requisitos no funcionales de esta aplicación.

ID	Requisitos No Funcionales
RNF-01	El sistema admitirá imágenes en formatos *JPG* y *PNG*.

Tabla 4.8: Requisitos no funcionales.

4.1.5. Modelo Entidad Relación

"El modelo ER es una herramienta que permite representar de manera simplificada cómo personas, objetos o conceptos se relacionan entre sí. Se utiliza para exponer cómo se organiza la información en una base de datos." [61].

La imagen 4.2 ilustra el modelo ER de nuestro proyecto.El estudiante será identificado con el NIA (identificador único de la universidad, formado por la letra e y seguido por el DNI de la persona) y se guardará el nombre de pila de él. Este puede realizar una o más entregas, indicando la fecha de realización. La entrega se resuelve generando un supuesto. El cual se identifica con un *id* único, y se guarda el título de la solución, y el porcentaje de acierto tanto de las entidades, entidades débiles, entidades fuertes, relaciones, relaciones débiles, relaciones fuertes y relaciones IS-A. Tanto el supuesto como la entrega describen una o más clases de información. Las clases de información se encuentran identificadas mediante un *id* único y guardan además una breve descripción y su nombre. Estas clases de información pueden ser relaciones, entidades o IS-A. Y además, una clases de información puede contener uno o más atributos. Un atributo se identifica con *id* único, y guarda el valor de su nombre, una breve descripción y el tipo de atributo. Un atributo puede ser contenido por otro atributo, así como puede contener otros atributos. Por otro lado, una entidad puede participar en ninguna, una o más relaciones y una relación participa en una o más entidades. Una entidad puede ser padre o hija de una IS-A. Tanto de las entidades como de las relaciones se guarda el tipo de componente, mientras que de las IS-A se registra su obligatoriedad y disyuntividad.

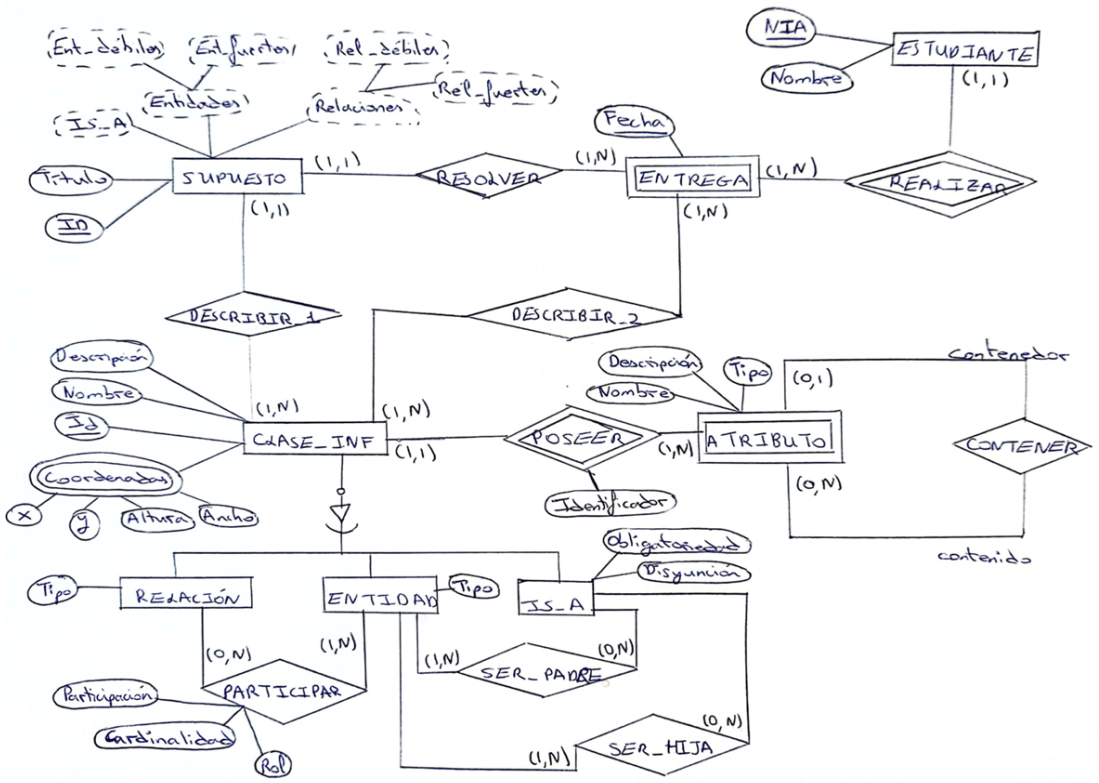


Figura 4.2: Modelo Entidad-Relación.

4.1.6. Diccionario de datos

El diccionario de datos documenta los metadatos del modelo ER. Existen dos tipos de diccionarios de datos, uno que especifica las entidades junto con sus atributos y otro que describe las relaciones asociadas a las entidades.

Diccionario de datos de entidades

Para cada entidad del modelo ER descrito en la sección 4.1.5 se ha creado un diccionario de datos. Las imágenes 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9 y 4.10 muestran respectivamente el diccionario de datos de las entidades ESTUDIANTE, ENTREGA, SUPUESTO, CLASE_INF, RELACIÓN, ENTIDAD, IS-A y ATRIBUTO.

ID	**ENT-01**										
Nombre:	**ESTUDIANTE**										
Definición:	La entidad ESTUDIANTE representa cada una de los estudiantes que utilizan la aplicación DBReader.										
Atributos:	**ID**	**Nombre**	**Definición**	**Tipo**	**Reglas**	**Derivado**	**Multivaluado**	**Compuesto**	**Único**	**Inicial**	**Nulo**
	A01.01	NIA	Identificador del estudiante.	String	Debe empezar por "e" y continuar con el formato de un DNI.	No	No	No	Si	No	No
	A01.02	Nombre	Nombre de pila del estudiante.	String		No	No	No	No	No	No

Identificadores:	**ID**	**Nombre Atributo**
	A01.01	NIA

Figura 4.3: Diccionario de datos de la entidad ESTUDIANTE.

ID	ENT-02
Nombre:	ENTREGA
Definición:	La entidad ENTREGA representa la solución a un ejercicio subida por el estudiante.

Atributos:	ID	Nombre	Definición	Tipo	Reglas	Derivado	Multivaluado	Compuesto	Único	Inicial	Nulo
	A02.01	Fecha	Representa la fecha de realización de la entrega del ejercicio.	Date	Sigue el formato Día/Mes/Año	No	No	No	No	No	No

Identificadores:	ID	Nombre Atributo
	A01.01	NIA
	A02.01	Fecha

Figura 4.4: Diccionario de datos de la entidad ENTREGA.

ID	ENT-03
Nombre:	SUPUESTO
Definición:	La entodad SUPUESTO representa la resolución de un ejercicio de un estudiante

Atributos:	ID	Nombre	Definición	Tipo	Reglas	Derivado	Multivaluado	Compuesto	Único	Inicial	Nulo
	A03.01	ID	Identificador del supuesto.	String		No	No	No	Si	No	No
	A03.02	IS-A	Número de IS-A que tiene el modelo.	Int		Si	No	No	No	No	No
	A03.03	Título	Título del modelo.	String		No	No	No	No	No	No
	A03.04	Entidades	Número de entidades que tiene el modelo.	Int		Si	No	No	No	No	No
	A03.04	Ent_débiles	Número de entidades débiles que tiene el modelo.	Int		Si	No	No	No	No	No
	A03.05	Ent_uertes	Número de entidades fuertes que tiene el modelo.	Int		Si	No	No	No	No	No
	A03.06	Relaciones	Número de relaciones que tiene el modelo.	Int		Si	No	No	No	No	No
	A03.07	Rel_débiles	Número de relaciones débiles que tiene el modelo.	Int		Si	No	No	No	No	No
	A03.08	Rel_fuertes	Número de relaciones fuertes que tiene el modelo.	Int		Si	No	No	No	No	No

Identificadores:	ID	Nombre Atributo
	A03.01	ID

Figura 4.5: Diccionario de datos de la entidad SUPUESTO.

ID	ENT-06
Nombre:	CLASE_INF
Definición:	La entidad CLASE_INF representa los componentes de un modelo ER.
Reglas:	Hay 3 tipos de CLASE_INF: RELACIÓN, ENTIDAD e IS-A

Atributos:	ID	Nombre	Definición	Tipo	Reglas	Derivado	Multivaluado	Compuesto	Único	Inicial	Nulo
	A06.01	Descripción	Breve definición del componente.	String		No	No	No	No	No	No
	A06.02	Nombre	Nombre del componente.	String		No	No	No	No	No	No
	A06.03	ID	Identificador del componente	String		No	No	No	Si	No	No
	A06.04	Coordenadas	Indica el las coordenadas del punto central del	Float		Si	No	No	Si	No	No

Identificadores:	ID	Nombre Atributo
	A06.03	ID

Figura 4.6: Diccionario de datos de la entidad CLASE_INF.

ID	ENT-07
Nombre:	RELACIÓN
Definición:	La entidad RELACIÓN representa a los componentes relaciones del modelo ER.

Atributos:	ID	Nombre	Definición	Tipo	Reglas	Derivado	Multivaluado	Compuesto	Único	Inicial	Nulo
	A07.01	Tipo	Representa la categoría de la relación.	String	Puede ser "débil" o "fuerte"	No	No	No	No	No	No

Identificadores:	ID	Nombre Atributo
	A06.03	ID

Figura 4.7: Diccionario de datos de la entidad RELACIÓN.

ID	**ENT-08**
Nombre:	**ENTIDAD**
Definición:	La entidad ENTIDAD representa a los componentes entidades del modelo ER.

	ID	Nombre	Definición	Tipo	Reglas	Derivado	Multivaluado	Compuesto	Único	Inicial	Nulo
Atributos:	A08.01	Tipo	Representa la categoría de la entidad.	String	Puede ser "débil" o "fuerte"	No	No	No	No	No	No

	ID	Nombre Atributo
Identificadores:	A06.03	ID

Figura 4.8: Diccionario de datos de la entidad ENTIDAD.

ID	**ENT-09**
Nombre:	**IS-A**
Definición:	La entidad IS-A representa las componentes IS-A del modelo

	ID	Nombre	Definición	Tipo	Reglas	Derivado	Multivaluado	Compuesto	Único	Inicial	Nulo
Atributos:	A09.01	Obligatoridad	Indica si las instancias de la superclase tienen que pertenecer a alguna subclase.	Boolean		No	No	No	No	No	No
	A09.02	Disjunción	Indica si las instancias de las superclases pueden pertenecer a más de una subclase.	Boolean		No	No	No	No	No	No

	ID	Nombre Atributo
Identificadores:	A06.03	ID

Figura 4.9: Diccionario de datos de la entidad IS-A.

ID	**ENT-10**
Nombre:	**ATRIBUTO**
Definición:	La entidad ATRIBUTO representa los componentes atributos de un modelo ER.

	ID	Nombre	Definición	Tipo	Reglas	Derivado	Multivaluado	Compuesto	Único	Inicial	Nulo
Atributos:	A10.01	Nombre	Nombre del atributo.	String		No	No	No	No	No	No
	A10.02	Descripción	Breve definición del componente.	String		No	No	No	No	No	No
	A10.03	Tipo	Representa la categoría del atributo.	String	Puede ser "compuesto", "derivado", "multivaluado", "simple" o "primario"	No	No	No	No	No	No
	A10.04	ID	Identificador del atributo.	String		No	No	No	Si	No	No

	ID	Nombre Atributo
Identificadores:	A10.04	ID
	A06.03	ID

Figura 4.10: Diccionario de datos de la entidad ATRIBUTO.

Diccionario de datos de relaciones

Para cada relación del modelo ER descrito en la sección 4.1.5 se ha creado un diccionario de datos. Las imágenes 4.11, 4.12, 4.13, 4.14, 4.15, 4.16, 4.17, 4.18 y 4.19 muestran respectivamente el diccionario de datos de las relaciones REALIZAR, SUBIR, RESOLVER,DESCRIBIR_1, DESCRIBIR_2, PARTICIPAR, SER_PADRE, SER_HIJA, POSEER y CONTENER

ID	**REL_01**
Nombre:	**REALIZAR**
Definición:	Un estudiante puede realizar una o más entregas y una entrega es realizada por un estudiante.

	ID	Nombre Entidad	Participación	Cardinalidad
Entidades	ENT-01	ESTUDIANTE	1	N
	ENT-02	ENTREGA	1	1

Figura 4.11: Diccionario de datos de la relación REALIZAR.

ID	**REL_03**
Nombre:	**RESOLVER**
Definición:	Un supuesto es resuelto a partir de una entrega.

	ID	Nombre Entidad	Participación	Cardinalidad
Entidades	ENT-02	ENTREGA	1	1
	ENT-03	SUPUESTO	1	N

Figura 4.12: Diccionario de datos de la relación RESOLVER.

ID	**REL_04**
Nombre:	**DESCRIBIR_1**
Definición:	Un supuesto describe una o má clase_inf y una clase_inf está descrita en un supuesto.

	ID	Nombre Entidad	Participación	Cardinalidad
Entidades	ENT-03	SUPUESTO	1	N
	ENT-06	CLASE_INF	1	1

Figura 4.13: Diccionario de datos de la relación DESCRIBIR_1.

ID	**REL_05**
Nombre:	**DESCRIBIR_2**
Definición:	Una entrega describe una o más clase_inf y una clase_inf puede estar descrita en una o más entregas.

	ID	Nombre Entidad	Participación	Cardinalidad
Entidades	ENT-02	ENTREGA	1	N
	ENT-06	CLASE_INF	1	N

Figura 4.14: Diccionario de datos de la relación DESCRIBIR_2.

ID	**REL_06**
Nombre:	**PARTICIPAR**
Definición:	Una relación participa en una o más entidades, una entidad puede participar en ninguna, una o más relaciones.

	ID	Nombre Entidad	Participación	Cardinalidad
Entidades	ENT-07	RELACIÓN	1	N
	ENT-08	ENTIDAD	0	N

	ID	Nombre	Definición	Tipo	Reglas	Derivado	Multivaluado	Compuesto	Único	Inicial	Nulo
Atributos:	E06.01	Cardinalidad	Valor de la cardinalidad de la relación.	String		No	No	No	No	No	No
	E06.01	Rol	Nombre del rol.	String		No	No	No	No	No	Si
	E06.01	Participación	Valor de la participación de la relación.	String		No	No	No	No	No	No

Figura 4.15: Diccionario de datos de la relación PARTICIPAR.

ID	**REL_07**
Nombre:	**SER_PADRE**
Definición:	Una IS-A puede ser clase padre de una o varias entidades y una entidad puede ser padre de ninguna, una o más IS-A.

	ID	Nombre Entidad	Participación	Cardinalidad
Entidades	ENT-08	ENTIDAD	0	N
	ENT-09	IS-A	1	N

Figura 4.16: Diccionario de datos de la relación SER_PADRE.

ID	**REL_08**
Nombre:	**SER_HIJA**
Definición:	Una IS-A puede ser clase hija de una o varias entidades y una entidad puede ser hija de ninguna, una o más IS-A.

	ID	Nombre Entidad	Participación	Cardinalidad
Entidades	ENT-08	ENTIDAD	0	N
	ENT-09	IS-A	1	N

Figura 4.17: Diccionario de datos de la relación SER_HIJA.

ID	REL_09
Nombre:	POSEER
Definición:	Una clase_inf puede poseer uno o más atributos y un atributo puede ser poseido por una clase_inf.

	ID	Nombre Entidad	Participación	Cardinalidad
Entidades	ENT-06	CLASE_INF	1	N
	ENT-10	ATRIBUTO	1	1

	ID	Nombre	Definición	Tipo	Reglas	Derivado	Multivaluado	Compuesto	Único	Inicial	Nulo
Atributos:	E09.01	Identificador	Identificador de la relación.	String	No	No	No	No	No	No	No

Figura 4.18: Diccionario de datos de la relación POSEER.

ID	REL_10
Nombre:	COTENER
Definición:	Un atributo puede contener ninguno, uno o más atributos y un atributo puedes ser contenido por uno o ningún atributo

	ID	Nombre Entidad	Rol	Participación	Cardinalidad
Entidades	ENT-10	ATRIBUTO	contenedor	0	N
	ENT-10	ATRIBUT0	contenido	0	1

Figura 4.19: Diccionario de datos de la entidad CONTENER.

4.2. Diseño

Una vez terminada la fase de análisis del *Software*, se realiza el diseño del mismo. "En esta etapa se transforman los requerimientos en un diseño o modelo con el cual se construye el sistema. El resultado del diseño es la identificación de las partes del sistema que satisfarán esas necesidades y facilitarán que este sea construido de forma simultánea por los individuos que conforman el equipo de desarrollo"[31]. Esta fase comprende el diseño arquitectónico, el diseño de los componentes, el diseño de la interfaz de la aplicación y el diseño de la base de datos. En este proyecto no se ha realizado un diseño de la base de datos, debido a que la aplicación propuesta carece de ella.

4.2.1. Arquitectura física

La arquitectura física expone cuáles son los componentes físicos de la aplicación a desarrollar (cliente, servidor,...), así como la relación existe entre ellos [5].

El modelo de arquitectura que se ha elegido para este proyecto es el de 2 capas. En este proyecto, se ha optado por una arquitectura de 2 capas. La capa cliente incluye el dispositivo desde el cual los usuarios acceden a la aplicación. La capa de aplicación contiene el servidor que ejecuta la lógica de la aplicación. La imagen 4.20 muestra la arquitectura física del sistema de este proyecto.

Sin embargo, en un entorno de producción sería adecuado que la aplicación contara con una arquitectura física más compleja como la que se describe en la imagen 4.21. A continuación, se explican los diferentes componentes que conforman el diagrama:

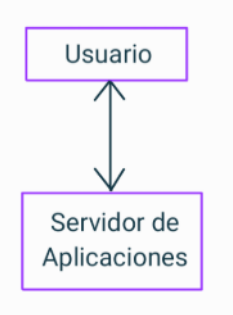


Figura 4.20: Arquitectura física del sistema actual.

- **Firewall:** se encuentra entre el cliente y el servidor, proporcionando una barrera de seguridad. Protege al servidor de acciones maliciosos, ataques de denegación de servicio, accesos no autorizados, etc.

- **Replicación de servidores:** para asegurar tanto la disponibilidad (en caso de que caiga uno por la razón que sea) como el mantenimiento de la calidad del servicio (si la carga de trabajo es alta).

- **Balanceadores de carga:** para que la distribución de la carga de trabajo sea equilibrada entre los servidores a fin de mantener la calidad del servicio. También se replican los balanceadores para asegurar la disponibilidad.

Una vez que la aplicación esté en funcionamiento, se espera que experimente picos de uso, especialmente durante el horario de estudio de los estudiantes y en los periodos de exámenes o entrega de tareas. Por ello, es importante que cuente con balanceadores de carga y replicadores de servidores para manejar estos picos de demanda. Además, al ser una aplicación web y por tanto accesible a través de internet, se necesita aplicar medidas de seguridad como el *firewall* propuesto.

Diagrama de despliegue

A continuación, en la imagen 4.22 se muestra el diagrama de despliegue del sistema. Este representa la arquitectura física del sistema, incluyendo nodos como entornos de ejecución de *hardware* o *software*, y el *middleware* que los conecta [28]. En nuestro diagrama de despliegue encontramos 2 dispositivos uno para aplicación cliente y otro para la aplicación servidor. La aplicación cliente está conectada con el entorno de ejecución del navegador *web*, y la aplicación servidor se encuentra asociada con los entornos de ejecución *webserver frontend* y *webserver backend*. A su vez este último entorno de ejecución se conecta a través de una llamada *HTTP* al sistema externo *Google Vision*.

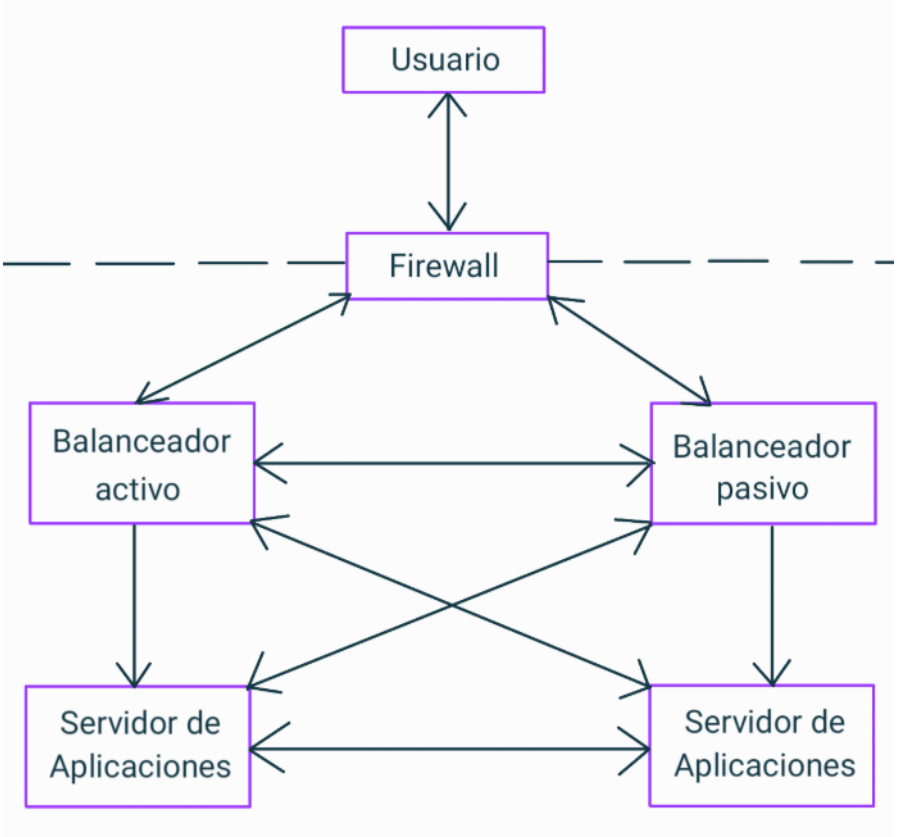


Figura 4.21: Arquitectura física ideal del sistema.

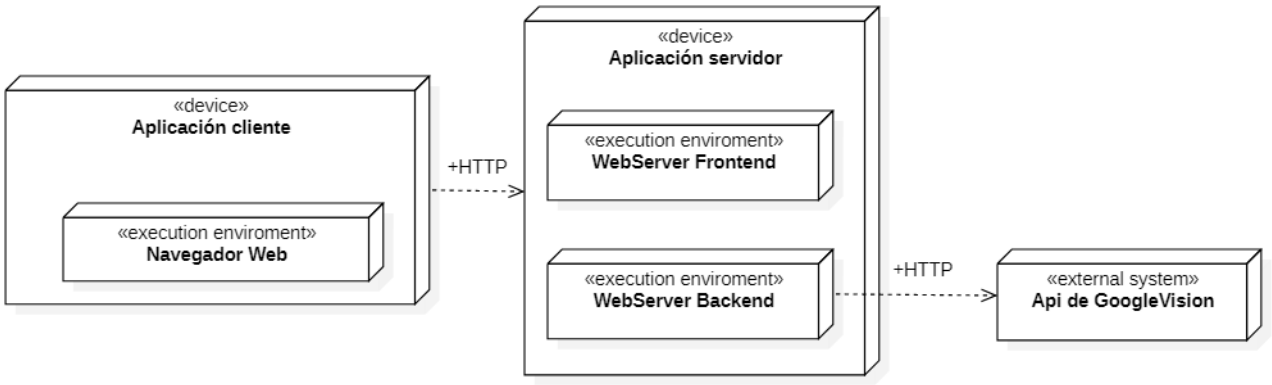


Figura 4.22: Diagrama de despliegue del sistema.

4.2.2. Arquitectura lógica

"La arquitectura lógica expresa cuáles son los componentes lógicos (subsistemas, o macro-funciones) que participan en nuestra proyecto, y la relación entre ellos" [5]. La imagen 4.23 muestra la arquitectura lógica del sistema que se va a desarrollar en este proyecto. Por un lado, encontramos el componente cliente que a través del navegador web se comunica con el servidor en el que hay una capa de presentación, que se ha desarrollado usando *Quasar Framework*. Este "es un marco de código abierto basado en *Vue.js* para crear aplicaciones con una única base de código" [41]. La capa de presentación se comunica a través de *Flask* con la capa de negocio, en la que encontramos los componentes de lógica de negocio formados por: el componente de reconocimiento de objetos, el componente de conectividad, el componente de reconocimiento de texto, el componente de validación y el componente de corrección y generación de feedback. En todos estos componentes encontramos diferentes funciones lógicas programadas con *Python*. El componente de validación se conecta con el sistema externo *Google Vision*. Todos los componentes de la lógica de negocio se conectan con un componente de acceso a datos.

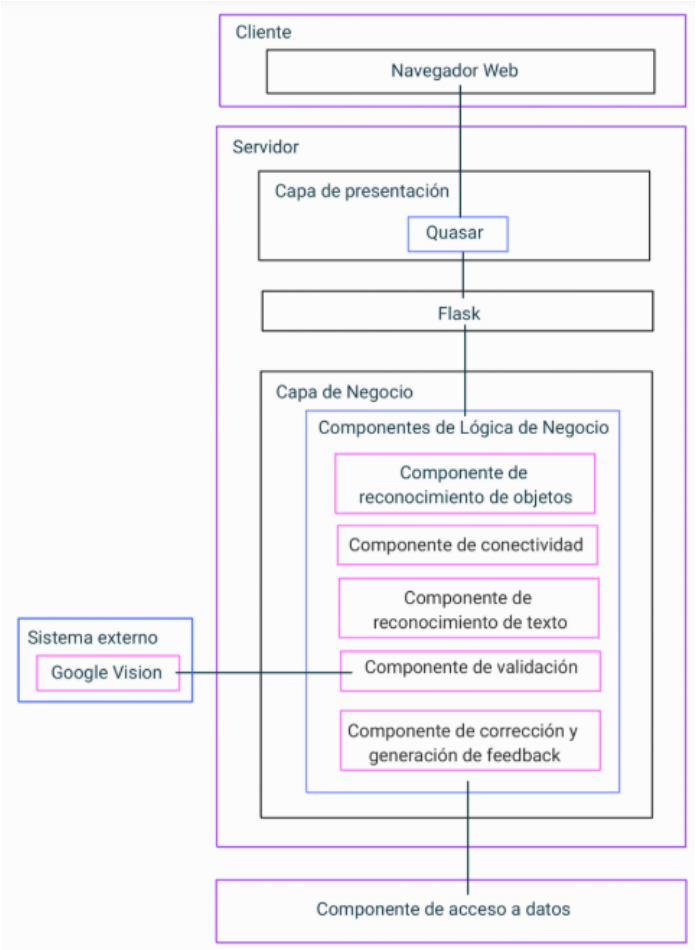


Figura 4.23: Arquitectura lógica del sistema.

4.2.3. Modelo lógico de datos

A continuación, se presenta el modelo lógico de datos. Todos los datos del modelo ER cargado se van a guardar dentro de una etiqueta llamada supuesto. Como ilustra la imagen 4.24 el supuesto almacena un *id*, el título del ejercicio, el conjunto de las entidades, relaciones e IS-A identificadas en el modelo, así como las conectividades relaciones-entidades. Para cada entidad almacenada en la etiqueta Entidades se guarda el id, el nombre, el tipo, las coordenadas (x,y,altura,anchura) y los atributos que tiene asociados. Para cada relación almacenada en la etiqueta Relaciones se guarda el id, el nombre, el tipo, las coordenadas y los atributos que tiene asociados. Para cada atributo almacenado en la etiqueta atributos se guarda el id, el nombre, el tipo y las coordenadas. En el caso de cada asociación relación-entidad se guarda el id de la relación, el id de la entidad, la cardinalidad, la participación y el rol. Y por último, para cada IS-A se almacena el id, el nombre, la descripción, la obligatoriedad y la disyunción. Las imágenes 4.25, 4.26, 4.27, 4.28 y 4.29 muestran respectivamente la estructura de datos de las entidades, relaciones, atributos, relaciones-entidades e IS-A.

```
Supuesto: {
   ID,
   Título,
   Entidades:{...},
   Relaciones:{...},
   IS-A{...},
   Relaciones-Entidades{...}
}
```

Figura 4.24: Estructura de datos del Supuesto.

```
Entidades:{
    Entidad:{
        Id,
        Nombre,
        Tipo,
        Coordenadas:{
            x,
            y,
            Altura,
            Ancho
        },
        Atributos:{...},
    },
    ...
}
```

Figura 4.25: Estructura de datos de Entidades.

```
Relaciones:{
    Relación:{
    Id,
    Nombre,
    Tipo,
    Coordenadas:{
        x,
        y,
        Altura,
        Ancho
    },
   Atributos:{...},
   ...
}
```

Figura 4.26: Estructura de datos de las Relaciones.

```
Atributos:{
    Atributo:{
    Id,
    Nombre,
    Tipo,
    Coordenadas:{
        x,
        y,
        Altura,
        Ancho
    },
    ...
}
```

Figura 4.27: Estructura de datos de los Atributos.

```
Relaciones-Entidades{
    Relacion-Entidad{
        Id_relacion,
        Id_entidad,
        Cardinalidad,
        Participación,
        Rol
    },
    ...
}
```

Figura 4.28: Estructura de datos de las Relaciones-Entidades.

```
IS-A{
    IS-A_1{
        Id,
        Nombre,
        Descripción,
        Obligatoriedad,
        Disyunción
    }
    ...
}
```

Figura 4.29: Estructura de datos de las IS-A.

4.2.4. Diseño de la interfaz de usuario

Para construir la interfaz de usuario, previamente creamos un primer boceto para cada una de las páginas de la aplicación. Las imágenes 4.30, 4.31, 4.32, 4.33, 4.34, 4.35, 4.36, 4.37, 4.38 y 4.39, ilustran respectivamente a las páginas subir_imagen, inicio, entidades, relaciones, roles, entidades-atributos, relaciones-atributos, entidades-relaciones, subir-imagen y generar-feedback.

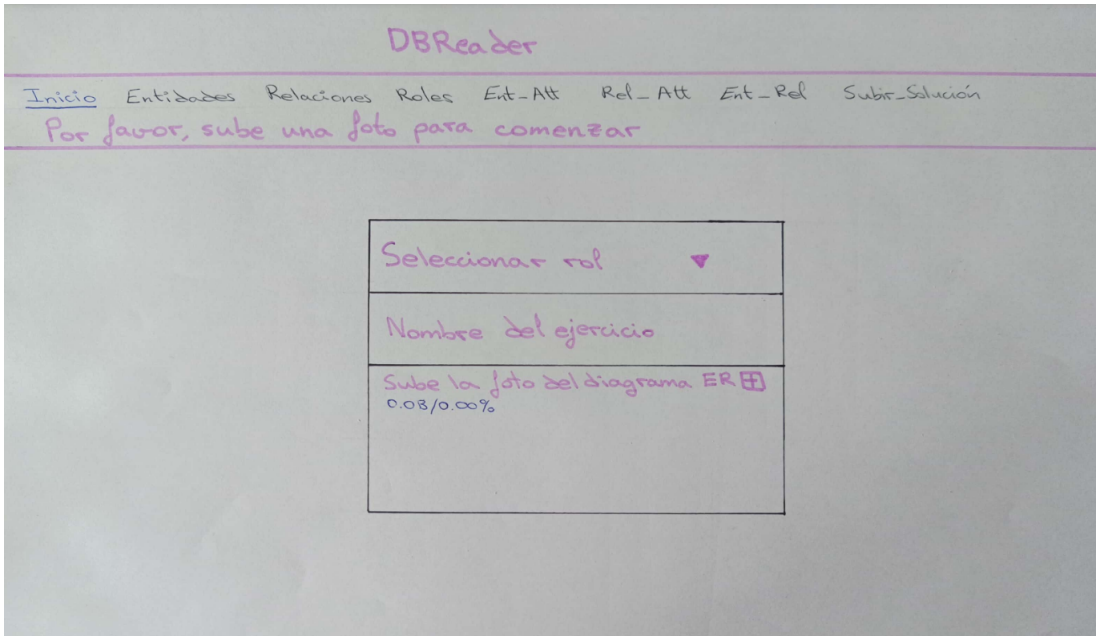


Figura 4.30: Página subir_imagen.

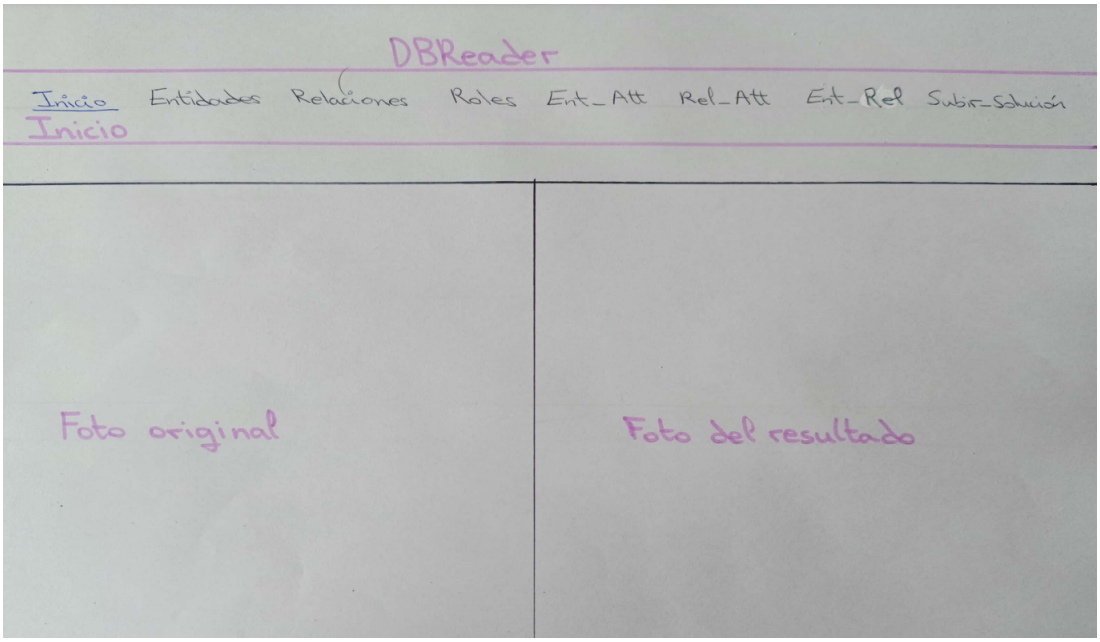


Figura 4.31: Página inicio.

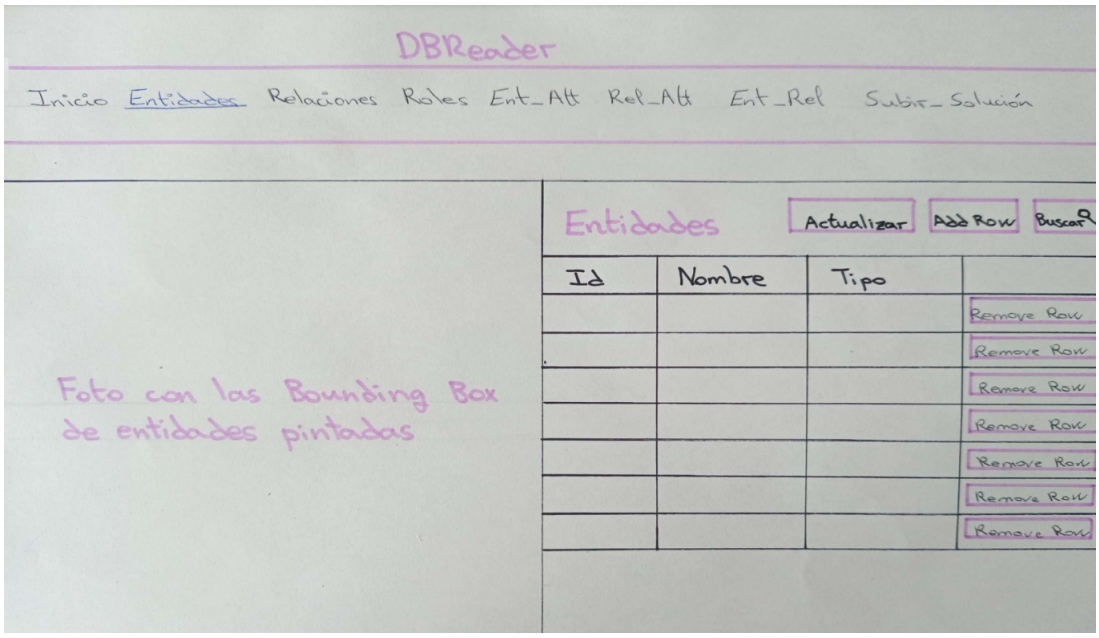


Figura 4.32: Página entidades.

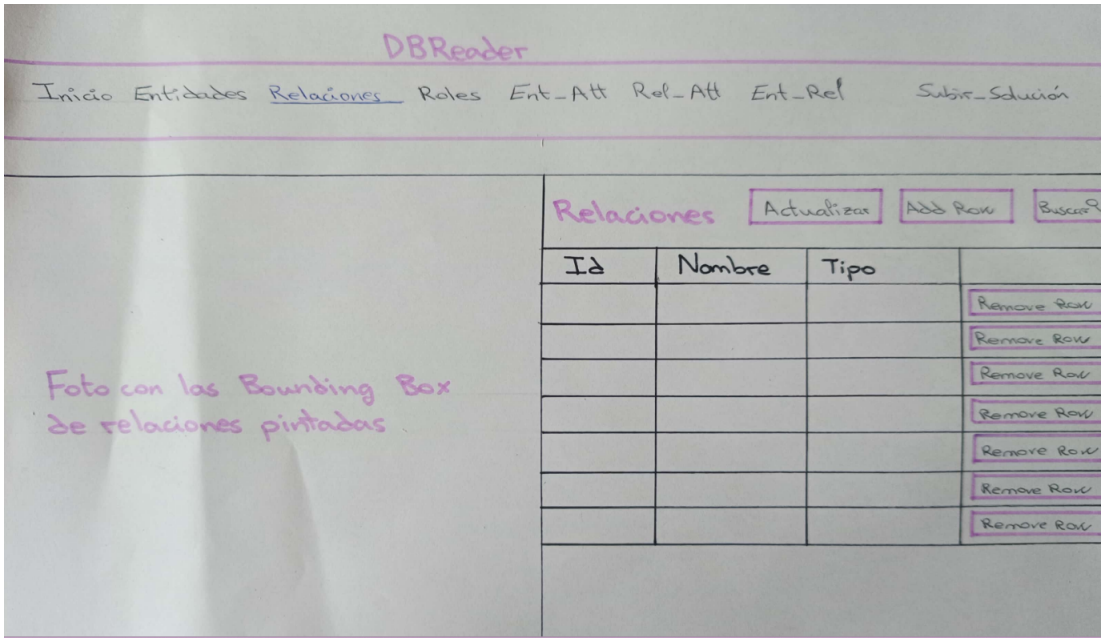


Figura 4.33: Página relaciones.

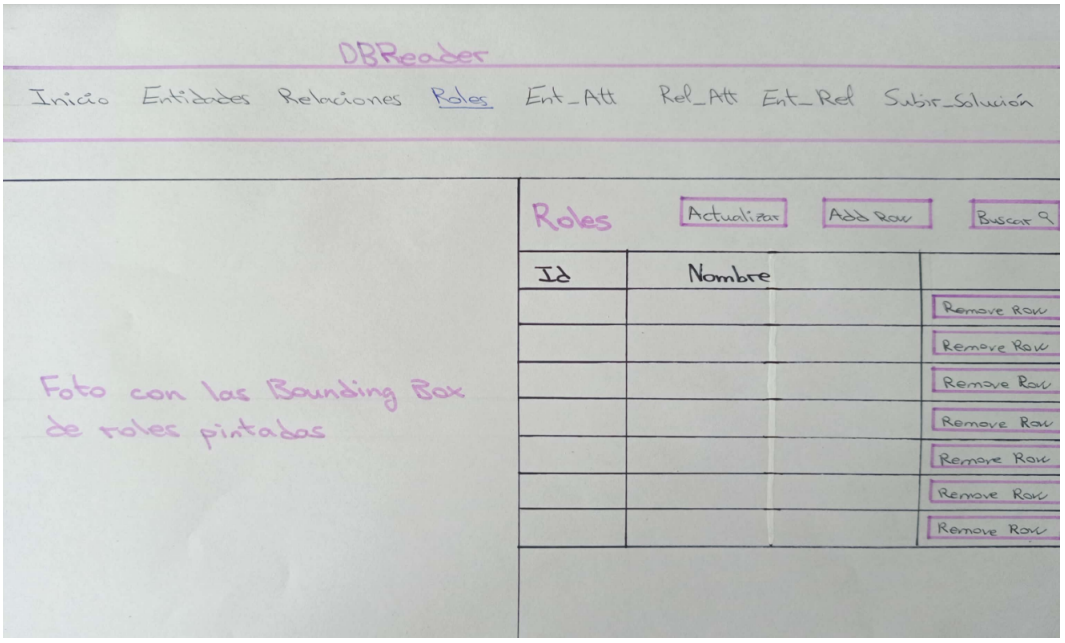


Figura 4.34: Página roles.

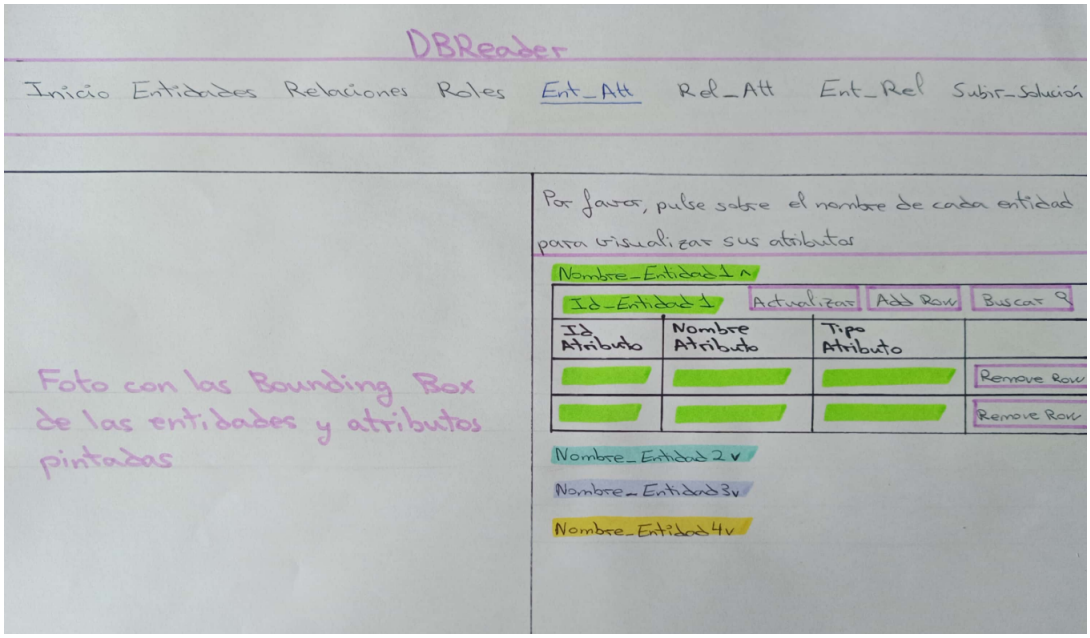


Figura 4.35: Página entidad-atributos.

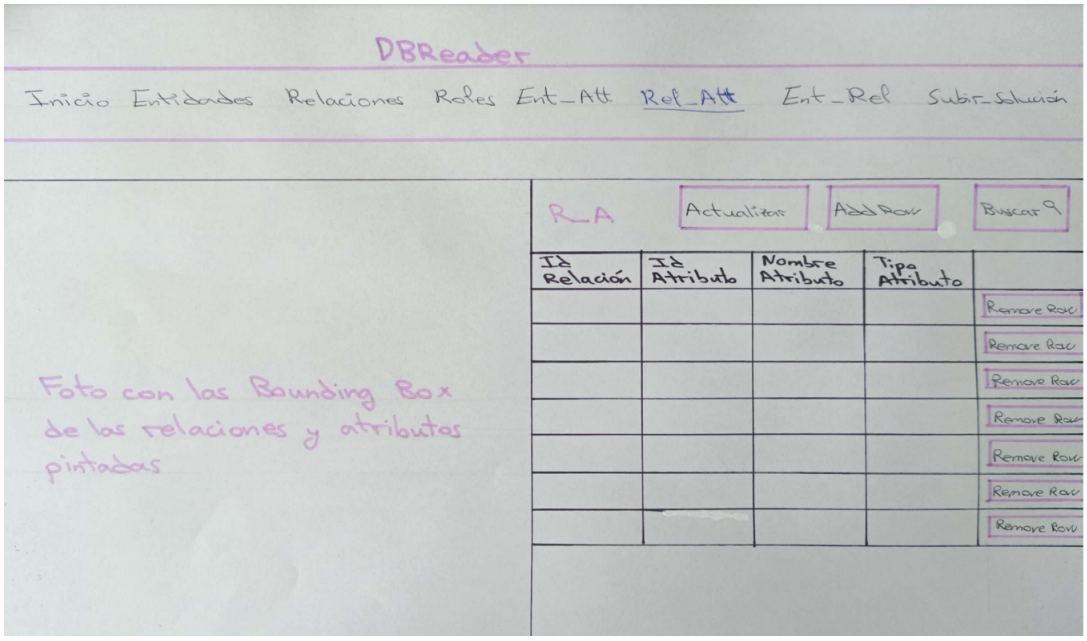


Figura 4.36: Página relación-atributos.

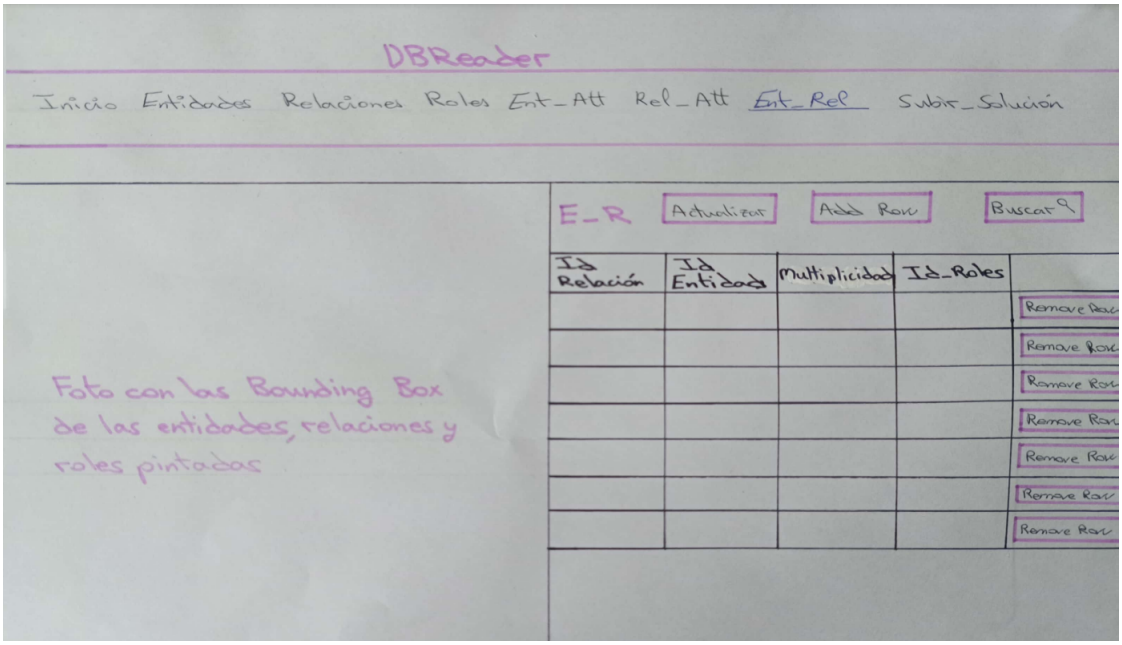


Figura 4.37: Página entidad-relación.

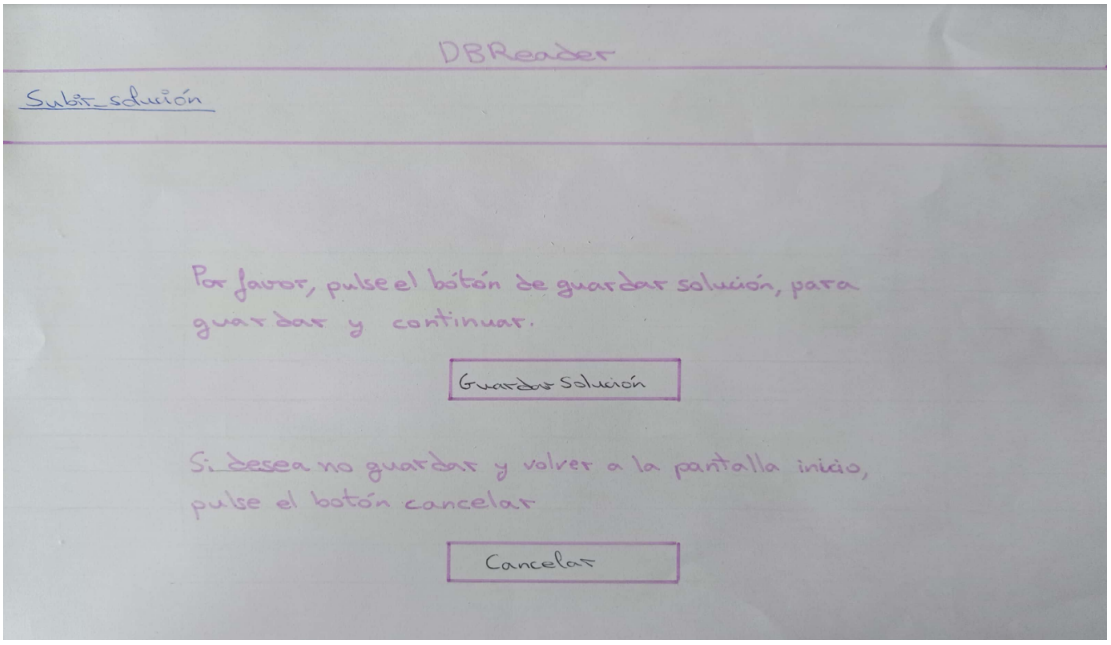


Figura 4.38: Página subir_solución.

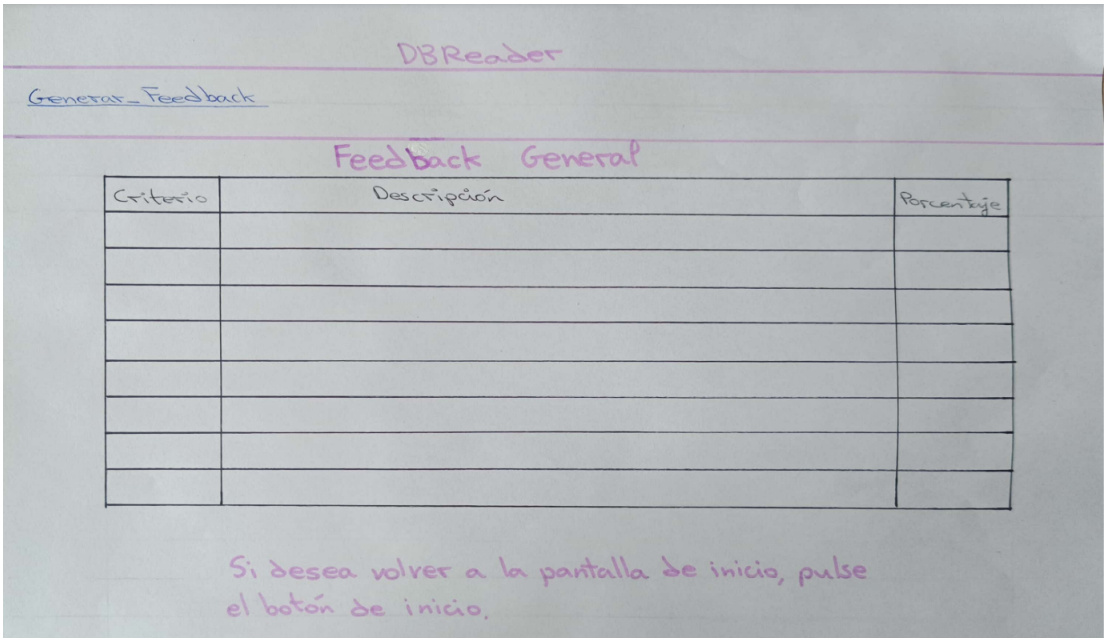


Figura 4.39: Página generar feedback.

4.3. Creación del *dataset*

4.3.1. Descripción general

El *dataset* utilizado en este proyecto está formado por 95 imágenes que fueron recopiladas a partir de exámenes (previamente anonimizados) correspondientes a la asignatura de Sistema de Bases de Datos del segundo curso del grado de Ingeniería Informática de Servicios y Aplicaciones de la Universidad de Valladolid (UVA). La figura 4.40 muestra un ejemplo de una imagen que forma parte de este *dataset*. Estas imágenes han sido etiquetadas manualmente utilizando la herramienta Label Studio. Para ello, se han utilizado 18 clases en el etiquetado de los datos. La tabla 4.9 detalla las clases junto con la cantidad de instancias correspondientes en el *dataset*. Durante el proceso de recopilación, se llevó a cabo una limpieza básica de las imágenes para eliminar "ruidos" (se borraron componentes tachados, así como componentes ilegibles) y posteriormente etiquetadas de acuerdo con el conjunto de etiquetas que se describen en el apartado siguiente.

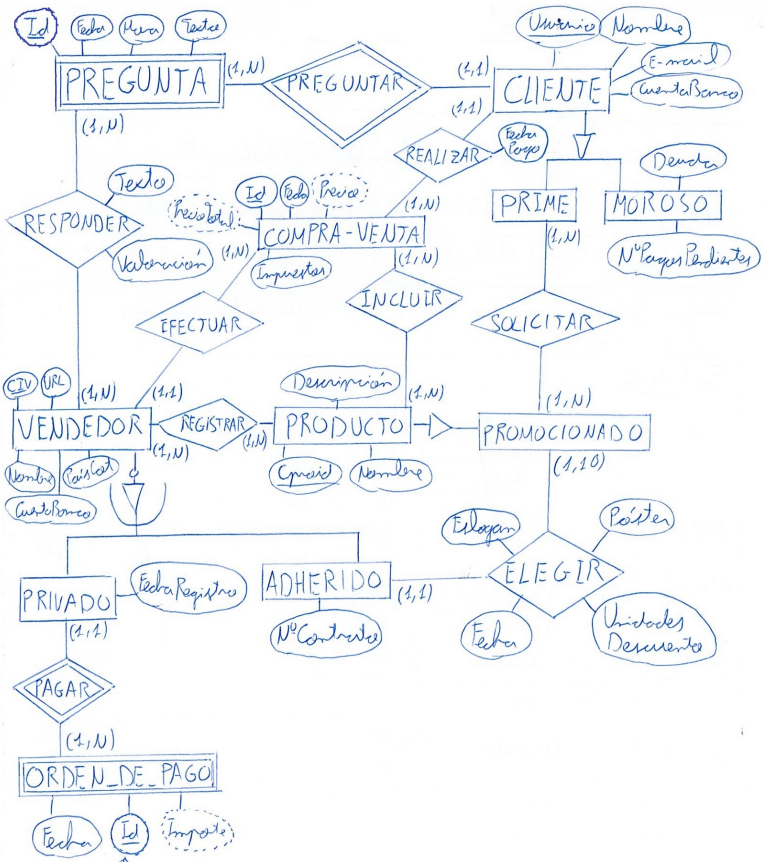


Figura 4.40: Ejemplo de uno de los diagramas ER que forma parte de la colección de datos creada en este proyecto.

Clase	Instancias
Atributo primario	616
Atributos derivados	142
Atributos multivaluados	55
Atributos simples	1962
Entidad débil	137
Entidad fuerte	770
IS-A no obligatoria/disjunta	9
IS-A no obligatoria/no disjunta	43
IS-A obligatoria/disjunta	55
IS-A obligatoria/no disjunta	23
Líneas	5799
Multiplicidad 0,1	51
Multiplicidad 0,N	235
Multiplicidad 1,1	416
Multiplicidad 1,N	686
Relación débil	134
Relación fuerte	678
Rol	22

Tabla 4.9: Número de instancias por clase en el *dataset*.

4.3.2. Etiquetado

Antes de nada hemos definido los nombres de las diferentes etiquetas que vamos a usar. El conjunto de clases que hemos creado está formado por 18 etiquetas, 4 de ellas representan a los atributos (Atributo primario, Atributos derivados, Atributos multivaluados, Atributos simples), 2 a las entidades (Entidad débil, Entidad fuerte), 4 a las relaciones IS-A (IS-A no obligatoria/disjunta, IS-A no obligatoria/no disjunta, IS-A obligatoria/disjunta, IS-A obligatoria/no disjunta), 4 a las multiplicidades (Multiplicidad 0,1, Multiplicidad 0,N, Multiplicidad 1,1, Multiplicidad 1,N), 2 a los tipos de relaciones débiles y fuertes (Relación débil, Relación fuerte), 1 para los roles (Rol) y una para las líneas que conectan los diferentes componentes (Lineas). Para etiquetar los diferentes componentes de las imágenes de nuestro *dataset* hemos utilizado la herramienta Label Studio [12]. Se trata de una herramienta de código abierto con una interfaz muy sencilla para etiquetar los datos de los proyectos de IA. Permite etiquetar diferentes tipos de datos: texto, imágenes, audio y vídeo. La figura 4.41 muestra un ejemplo de imagen etiquetada con Label Studio.

Una vez realizado el etiquetado de las imágenes, se han exportado los datos en formato YOLO [35] de manera que se ha obtenido una carpeta con las imágenes, otra con los ficheros txt que contienen la información de las etiquetas, un fichero classes.txt (contiene el nombre de las diferentes etiquetas usadas) y un archivo notes.json (formado por el nombre de las diferentes etiquetas y un identificador único para cada una de ellas). Las imágenes 4.42 y 4.43 ilustran la información del fichero notes.json, y la imagen 4.44 muestra el contenido del fichero classes.txt.

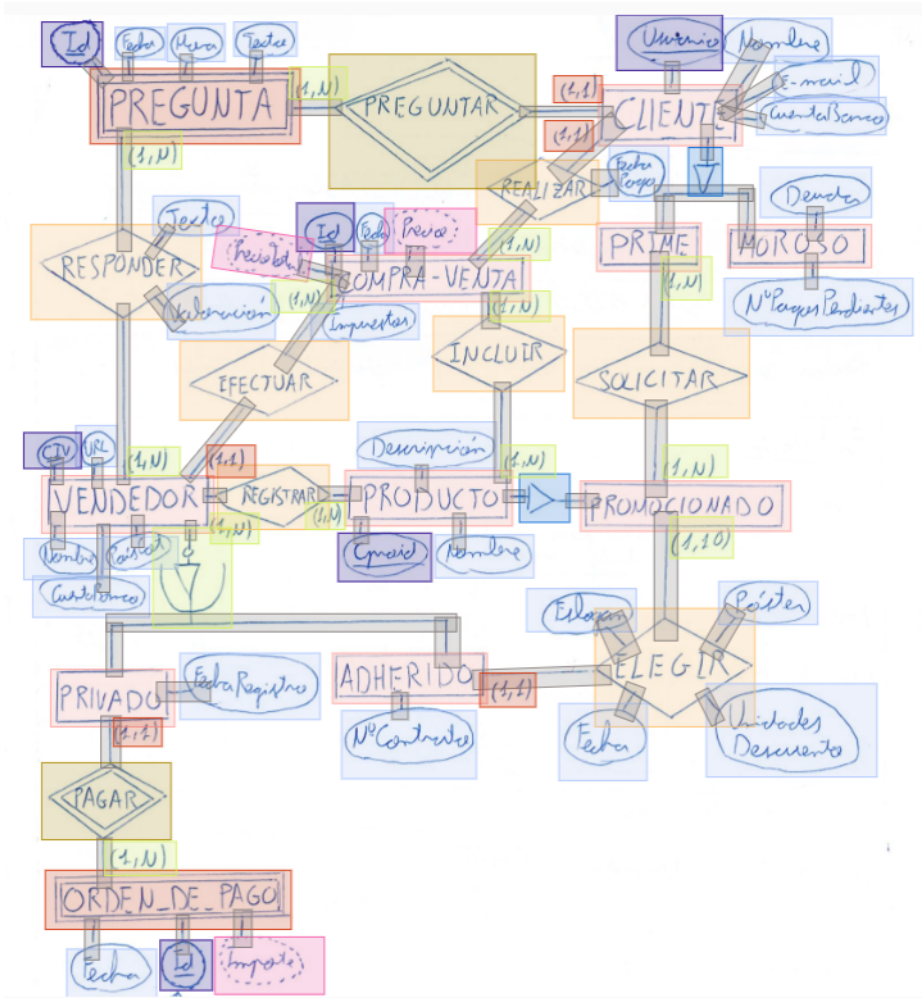


Figura 4.41: Ejemplo de una imagen etiquetada con Label Studio.

```
{
  "categories": [ ···
  ],
  "info": {
    "year": 2023,
    "version": "1.0",
    "contributor": "Label Studio"
  }
}
```

Figura 4.42: Contenido del fichero notes.json contraído.

```
{
  "categories": [
    {
      "id": 0,
      "name": "Atributo primario"
    },
    {
      "id": 1,
      "name": "Atributos derivados"
    },
    {
      "id": 2,
      "name": "Atributos multivaluados"
    },
    {
      "id": 3,
      "name": "Atributos simples"
    },
    {
      "id": 4,
      "name": "Entidad debil"
    },
    {
      "id": 5,
      "name": "Entidad fuerte"
```

Figura 4.43: Contenido del fichero notes.json extendido.

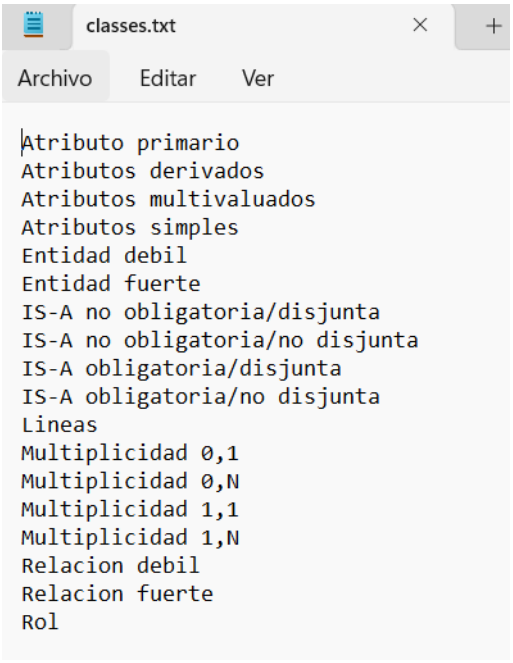


Figura 4.44: Contenido del fichero classes.txt.

4.3.3. División del *dataset*

A continuación, se explica como se ha dividido el *dataset* en tres conjuntos (entrenamiento, validación y prueba) para poder entrenar el modelo YOLO. Se recomienda que el conjunto de entrenamiento este formado por aproximadamente el 80 % de las imágenes y los conjuntos validación y prueba por el otro 20 % [49]. Por ello, de las 95 imágenes de nuestro *dataset* 75 conforman el conjunto de entrenamiento (79 %), 5 el de validación (5 %) y 15 el de prueba (16 %). Además. el formato de los archivos comprende imágenes almacenadas en jpg y png y etiquetas en archivos de texto. Cada archivo de etiquetas proporciona información sobre las coordenadas y las clases identificadas en cada una de las imágenes que componen el *dataset*.

Capítulo 5

Procesamiento de los componentes del modelo ER

En este capítulo se describe cómo se ha llevado a cabo la identificación de los diferentes componentes del modelo ER, se explican las distintas reglas que se han utilizado para conectarlos y se expone un análisis de la efectividad tanto del modelo propuesto para la identificación de los componentes, como de las reglas usadas para la conectividad.

En la subsección 5.2 se explica como se han identificado los diferentes componentes, en la 5.3 se exponen las reglas que se han utilizado para conectar los distintos componentes de los modelos ER, que corresponden respectivamente a los pasos 2 (reconocimiento y clasificación de objetos) y 3 (conectividad entre objetos) del pipeline de la aplicación que se muestra en la imagen 5.1. Es decir, empezaremos teniendo una imagen con un modelo ER, como la que se muestra en la figura 4.40, y en el primer paso identificaremos los diferentes componentes (entidades, relaciones, atributos y roles) y los añadiremos junto con sus coordenadas a un fichero json. En el segundo paso, conectaremos los atributos con sus entidades y relaciones correspondientes y las relaciones con sus respectivas entidades, multiplicidades y roles (en caso de que tengan) e incluiremos esta información en el archivo json anteriormente creado.

Para finalizar el capítulo, en la sección 5.4 se expone el diseño experimental y en la 5.5 un análisis de los resultados de identificación de los componentes y la conectividad de los mismos.

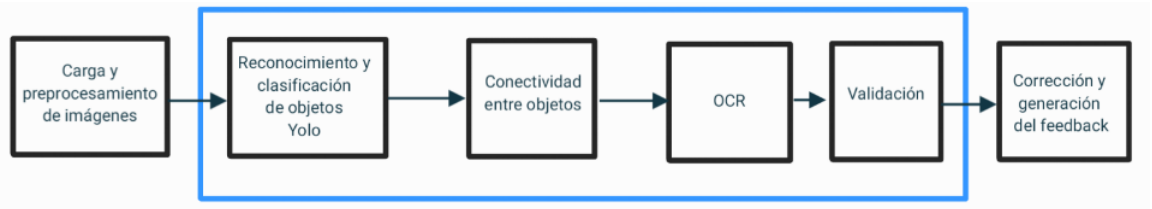


Figura 5.1: Workflow de la aplicación.

5.1. Introducción

El procesamiento de los componentes consiste por una parte en identificar los diferentes componentes del modelo ER y por otra parte conectarlos. Consiguiendo de esta manera al final del

proceso tener toda la información digitalizada del modelo ER de la imagen. Para poder identificar los componentes es necesario definir los diferentes componentes que conforman un modelo ER (entidades fuertes, entidades débiles, relaciones fuertes, relaciones débiles, IS-A obligatoria y disjunta, IS-A no obligatoria y disjunta, IS-A obligatoria y no disjunta, IS-A no obligatoria y no disjunta, atributos primarios, atributos simples, atributos derivados, atributos multivaluados, multiplicidades y roles), crear y entrenar un modelo que sea capaz de detectar estos componentes. Una vez identificados, se definen y programan una serie de reglas para conectar los atributos con sus correspondientes entidades y relaciones, así como las relaciones con sus entidades, multiplicidades y roles en caso de tener.

5.2. Identificación de los componentes

El segundo paso del flujo es la identificación de los diferentes componentes de un modelo ER. Para ello, hemos pasado la imagen con el modelo ER a un modelo de reconocimiento de objetos que hemos entrenado y seleccionado previamente como se explica en la sección 5.4.3. Este nos devuelve un fichero txt con todas las clases identificadas y sus correspondientes coordenadas. La figura 5.2 presenta un modelo ER, en el que se encuentran pintadas las Bounding Boxes de cada uno de sus componentes (18 etiquetas), cada una resaltada con un color único.

Posteriormente, se ha creado un JSON con las etiquetas entidades, relaciones, atributos y roles. En la etiqueta entidades se ha añadido una lista con todos los componentes que pertenecen a las clases "Entidad debil" y "Entidad fuerte", para cada uno de los cuales hemos guardado un identificador que es de la forma "ENT-000" siendo único e incremental, sus coordenadas y si se trata de una entidad débil o fuerte. En el caso de las relaciones, hemos guardado los componentes que pertenecen a las clases: "Relación debil", "Relación fuerte", "IS-A no obligatoria/disjunta", "IS-A no obligatoria/no disjunta", "IS-A obligatoria/disjunta", "IS-A obligatoria/no disjunta" y de nuevo hemos guardado de cada uno un identificador que sigue el formato "REL-000" que vuelve a ser único e incremental, sus coordenadas y el tipo de relación de la que se trata. Para los atributos hemos considerado los componentes de las clases: "Atributo primario", "Atributos derivados", "Atributos multivaluados" y "Atributos simple" y para cada uno de ellos hemos guardado un identificador único e incremental que sigue el formato "ATT-000", sus coordenadas y su tipo de atributo. Por último, en roles hemos añadido un identificador único e incremental de la forma "ROL-000" y las coordenadas de los componentes cuya clase es "ROL".

Las imágenes 5.3 y 5.4 muestran un ejemplo de los datos que se guardan respectivamente para una entidad y una relación identificada. En ambos casos se ha creado un identificador único, se han calculado las coordenadas de su *bounding box* y se ha guardado sus tipos.

5.3. Determinación de la conectividad

El siguiente paso del workflow de la aplicación consiste en conectar los diferentes componentes del modelo ER. El objetivo es conectar por un lado los atributos con su respectiva entidad y/o relación y por otro las relaciones con sus correspondientes entidades, multiplicidades y roles (en caso de que tengan). Para ello, se han considerado las *bounding-boxes* (coordenadas) de cada uno de los componentes y se han ido definiendo una serie de reglas. A continuación, se explican las reglas utilizadas en la conectividad de cada uno de los componentes:

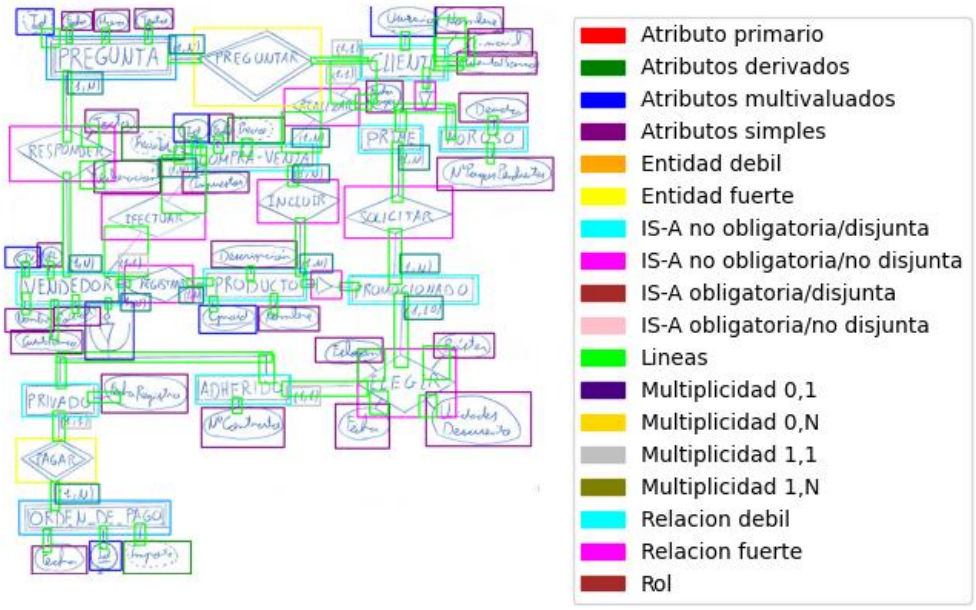


Figura 5.2: Ejemplo de todas las etiquetas identificadas en un modelo ER.


```
"id": "ENT-001",
"type": "fuerte",
"coords": {
    "x": 0.439252,
    "y": 0.671743,
    "width": 0.179263,
    "height": 0.0595294
}
```


Figura 5.3: Ejemplo de los datos que se guardan de una entidad identificada.


```
"id": "REL-008",
"type": "debil",
"coords": {
    "x": 0.463086,
    "y": 0.103483,
    "width": 0.218522,
    "height": 0.135798
}
```


Figura 5.4: Ejemplo de los datos que se guardan de una relación identificada.

1. **Conectividad Entidad-Atributos:** Consiste en identificar la entidad a la que pertenece cada uno de los atributos identificados, para lo cual se va a buscar el solapamiento entre el área de influencia de cada atributo y de las entidad próximas. Para implementarlo iniciamos creando un array con todos los atributos (primarios, simples, derivados y multivaluados) identificados en el diagrama ER. Procedemos a verificar si existe área común entre cada una

de las líneas y entidades identificadas en el modelo ER. Si encontramos una superposición, examinamos cada atributo del array previamente creado para determinar si comparten área con la línea. En caso afirmativo, el atributo se agrega a la entidad correspondiente. En la Figura 5.5, las entidades fuertes están resaltadas en azul celeste, las débiles en naranja, y los atributos en azul oscuro. Por ejemplo, a la entidad débil "ORDEN_DE_PAGO" se le ha asociado un atributo primario ("Id"), uno simple ("Fecha") y uno derivado ("Importe").

2. **Conectividad Relación-Atributos:** Trata de identificar la relación a la que pertenece cada uno de los atributos identificados, para ello se va a buscar el solapamiento entre el área de influencia de cada atributo y de las relaciones más cercanas. Para ello, aplicamos la misma regla que en la conectividad de entidades y atributos, con la única diferencia, que aquí descartamos los atributos que han sido conectados previamente con alguna entidad. En la Figura 5.5, las relaciones fuertes están resaltadas en amarillo, las débiles en rojo, y los atributos en azul oscuro. Por ejemplo, a la relación fuerte "REALIZAR" se le ha asociado 1 atributo simple ("Fecha Pago").

3. **Conectividad Entidad-Relación:** Radica en identificar la entidad a la que pertenece cada relación identificada en el modelo. En este paso, aplicamos el mismo razonamiento que en las conectividades anteriores. En esta ocasión, creamos un array que contiene todos los tipos de relaciones presentes en el diagrama ER (fuertes, débiles, IS-A disjunta y obligatoria, IS-A no disjunta y obligatoria, IS-A no disjunta y no obligatoria, e IS-A disjunta y no obligatoria). A continuación, evaluamos si existe área común entre cada entidad y línea del modelo. Si encontramos una superposición, verificamos si dicha línea se superpone con alguna relación. En caso afirmativo, añadimos la relación a la entidad correspondiente. Después de revisar todas las entidades, agregamos a un array todas las líneas que no tienen área en común con una entidad y una relación simultáneamente. Posteriormente, fusionamos estas líneas de dos en dos y repetimos el proceso. Este procedimiento se repite hasta un máximo de 6 veces. En la Figura 5.6, las entidades se encuentran resaltadas en azul celeste, y las relaciones en azul oscuro. Por ejemplo, a la relación fuerte "REALIZAR" se le han asociado dos entidades ("COMPRA-VENTA" y "CLIENTE").

4. **Conectividad de las multiplicidades:** Consiste en identificar para cada una de las multiplicidades la relación o entidad más cercana a ella. Esta conectividad se lleva a cabo después de haber establecido previamente la conectividad de Entidad-Relación. Cada línea de multiplicidad se vincula con la entidad o la relación más cercana, facilitada por el uso del algoritmo de distancia euclidiana, que evalúa los puntos medios tanto de la línea como de la entidad o relación. La información de multiplicidad, incluyendo el tipo y las coordenadas, se asocia a la entidad o relación identificada, y este proceso se repite para todas las líneas de multiplicidad en el modelo. La distancia máxima permitida entre la línea de multiplicidad y la entidad o relación más cercana se establece en 150 píxeles, ya que se ha visto que esta puede ser la máxima distancia en la que un estudiante escribe el valor de una multiplicidad de una línea. En la representación visual de este proceso, mostrada en la Figura 5.7, las relaciones se destacan en azul oscuro, las entidades fuertes en rojo, las entidades débiles en verde, y las multiplicidades en rosa. Por ejemplo, a la relación fuerte "REALIZAR" se le han asociado dos multiplicidades ("Multiplicidad 1,1" y "Multiplicidad 1,N").

5. **Conectividad de los roles:** Trata de identificar para cada una de los roles la entidad más cercana a ella. Esta conectividad se realiza de manera análoga a la conectividad de

las multiplicidades. Para cada rol, calculamos el punto medio de la línea más cercana y utilizando el algoritmo de distancia euclidiana identificamos la entidad más cercana a cada rol y añadimos dicho rol a esa entidad correspondiente. Este proceso se repite para cada rol en el modelo. La distancia máxima permitida entre la línea de rol y la entidad más cercana se establece en 150 píxeles, ya que se ha visto que esta puede ser la máxima distancia en la que un estudiante escribe el valor del rol de una línea. En la imagen que estamos utilizando como ejemplo para explicar el flujo de la aplicación no aparecen roles, por ello hemos tomado como excepción para este caso el ejemplo de otra imagen que tiene 2 roles y se muestra en la Figura 5.8. En ella, las relaciones están resaltadas en azul oscuro, las entidades fuertes en rojo, las entidades débiles en verde y los roles en rosa. Por ejemplo, a la relación fuerte "SEGUIR" se le han asociado dos roles ("usuario que sigue" y "usuario seguido").

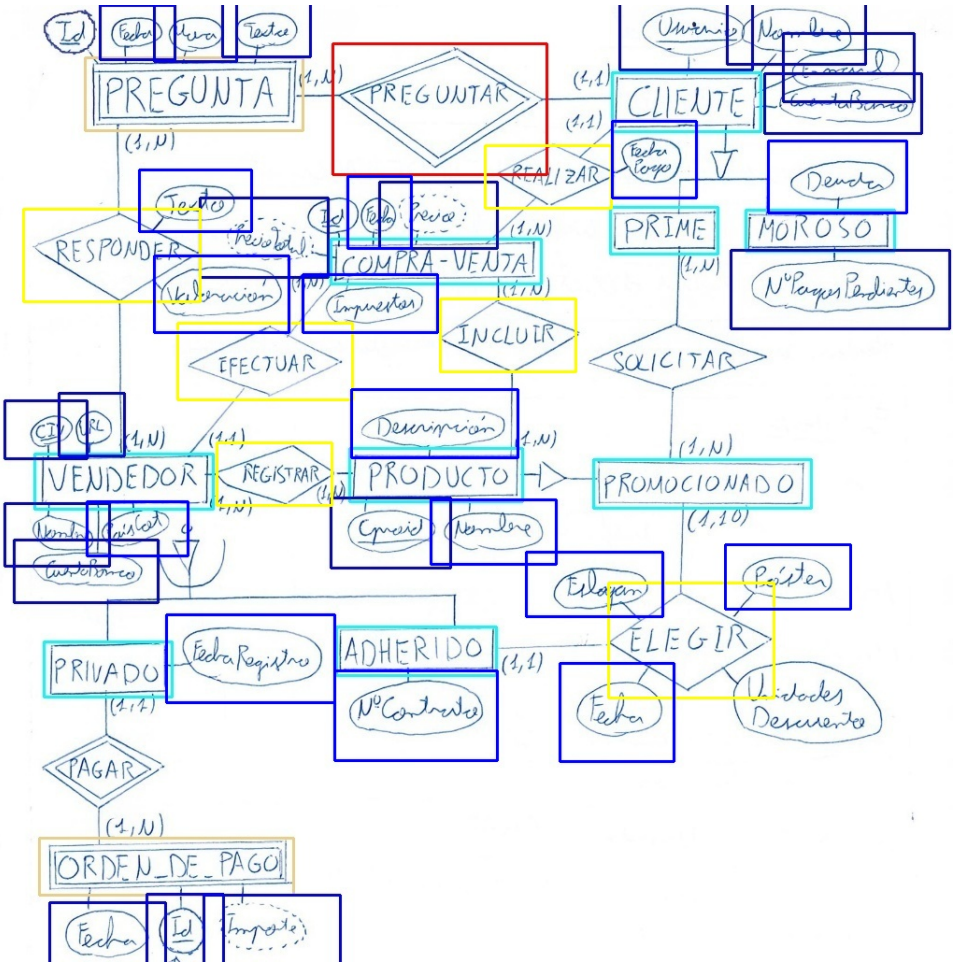


Figura 5.5: Ejemplo de elemetos Entidad-Atributos y Relacion-Atributosconectados.

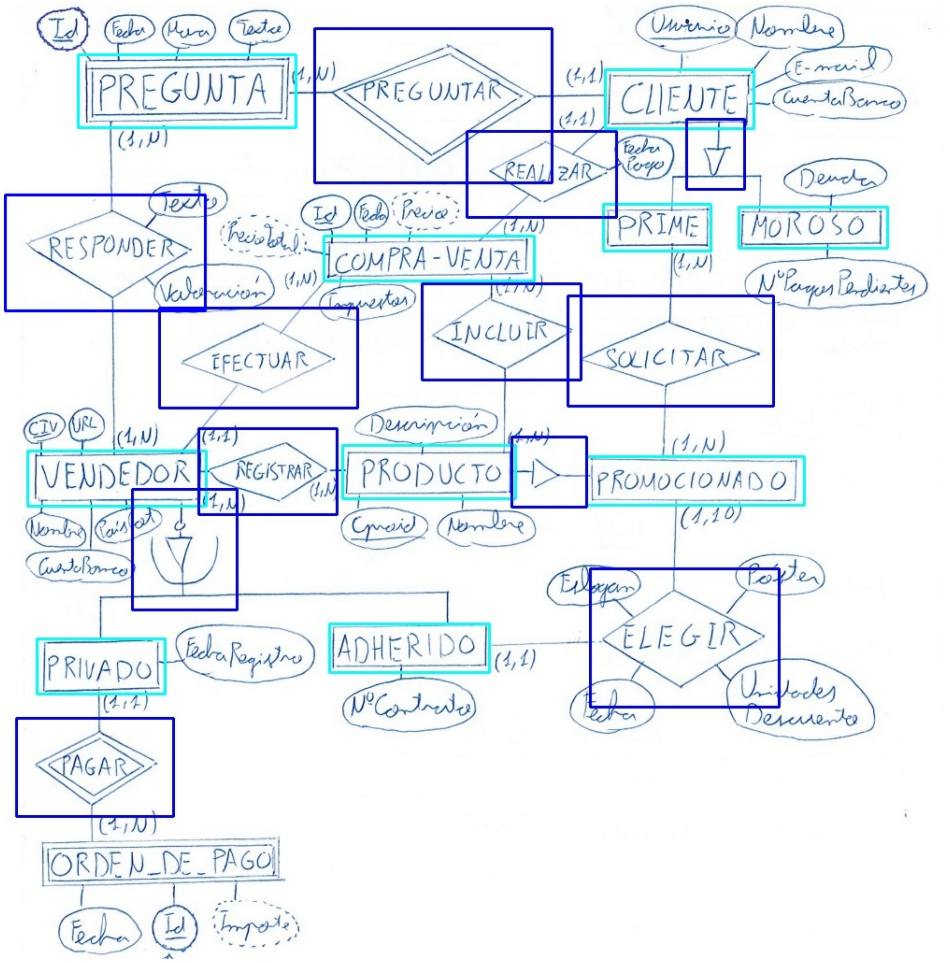


Figura 5.6: Ejemplo de elementos Entidad-Relacion conectados.

5.4. Diseño experimental

En esta sección, se exponen las distintas métricas utilizadas para evaluar los diferentes modelos configurados para la identificación de los componenetes de un modelo ER, y se explica cómo se ha realizado la elección del modelo utilizado en este proyecto.

5.4.1. Métricas de evaluación

Empezaremos explicando las métricas de evaluación que se han utilizado para realizar tanto el análisis de los modelos como el de la conectividad de los componentes. Hemos estudiado las siguientes:

- **Precisión:** es una medida que indica cuántos de los objetos detectados por el modelo son realmente objetos de interés en comparación con el total de objetos detectados. La precisión varía entre 0 y 1, donde 1 indica una precisión perfecta [62]. La precisión se calcula usando

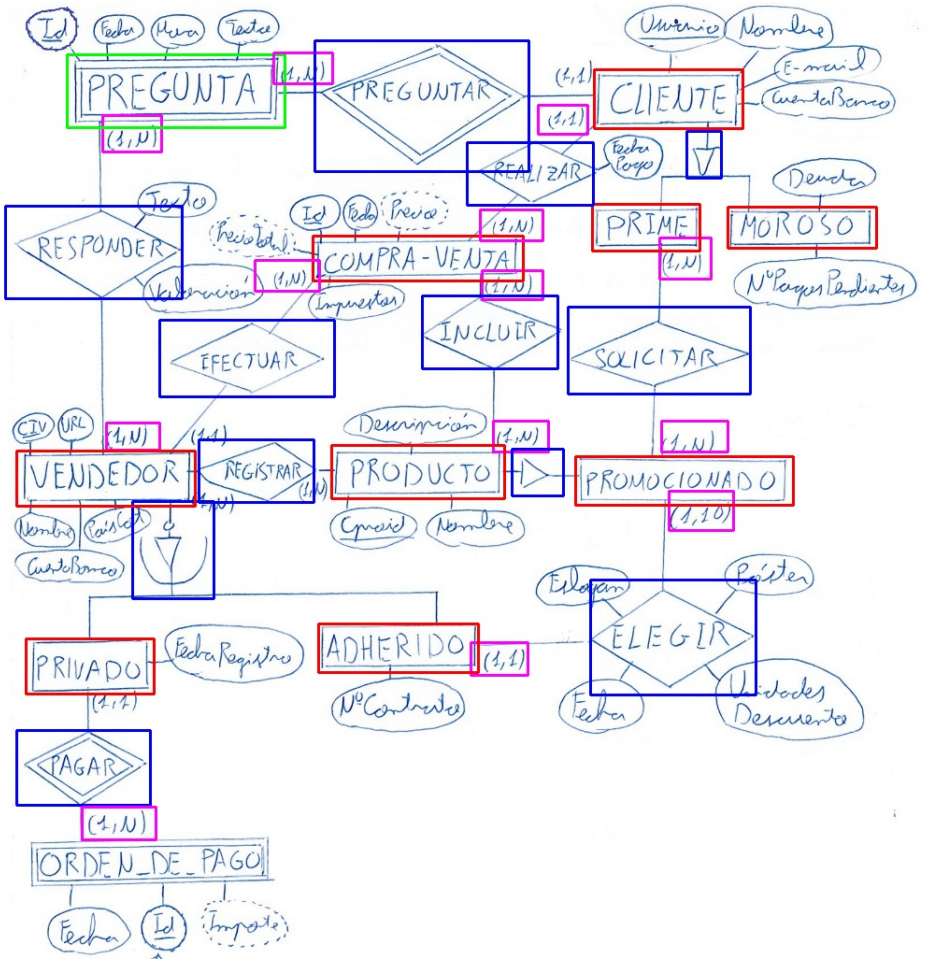


Figura 5.7: Ejemplo de elemetos Entidad-Relación-Multiplicidades conectados.

la siguiente fórmula:

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{5.1}$$

- **Métrica R:** se refiere a la tasa de *recall* o tasa de recuperación. El *recall* es una medida que evalúa la capacidad del modelo para detectar todos los elementos positivos (objetos de interés) en el conjunto de datos [62]. La métrica R (o *recall*) se calcula usando la siguiente fórmula:

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{5.2}$$

- **Métrica F1:** es una medida que combina la precisión y el recall en un solo valor, proporcionando una evaluación más completa del rendimiento del modelo en la detección de objetos. Esta métrica varía entre 0 y 1, donde 1 indica un equilibrio perfecto entre precisión y recall. Un valor más alto de F1 indica un mejor rendimiento en la detección de objetos

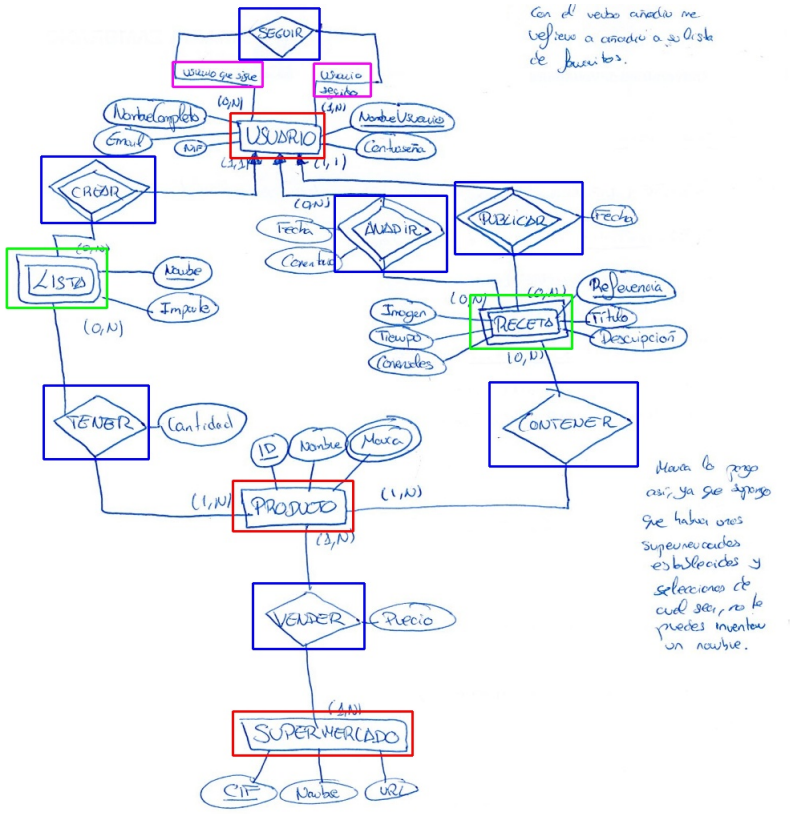


Figura 5.8: Ejemplo de elemetos Entidad-Relación y Roles conectados.

en comparación con valores más bajos [62]. La métrica F1 se calcula usando la siguiente fórmula:

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{5.3}$$

- **Métrica mAP50** ***(mean Average Precision at 50)*****:** es una métrica que evalúa la precisión promedio de detección de objetos, considerando solo las detecciones con una confianza del modelo superior al 50 %, siendo un valor más cercano a 1 indicativo de un mejor rendimiento [62]. La mAP50 se calcula usando la siguiente fórmula:

$$mAP_{50} = \frac{1}{N} \sum_{i=1}^{N} AP_{50}(i) \tag{5.4}$$

Donde N es el número total de clases y $AP_{50}(i)$ es la precisión media (que calcula el área bajo la curva de precisión-recuperación, proporcionando un valor único que engloba el rendimiento de precisión y recuperación (*recall*) del modelo con un umbral sobre unión de 0,5 [18]), el cual se calcula mediante la siguiente fórmula:

$$AP_{50}(i) = \frac{1}{M} \sum_{k=1}^{M} P_{50}(k, i) \tag{5.5}$$

Siendo M el número total de imágenes y P_{50} calcula la precisión con un umbral de intersección sobre unión (IoU) de 0,50.

5.4.2. Entrenamiento de los modelos

Hemos entrenado 6 modelos diferentes usando yolo8 [35]. Para el desarrollo de los modelos, se ha utilizado la implementación concreta de YOLO desarrollada por Rizwan Munawa [35]. En el momento de realización del proyecto, esta implementación se encuentra en la versión 8. Para cada uno de los modelos, se han definido diferentes combinaciones de valores de hiperparámetros del modelo, y se ha entrenado y evaluado utilizando los conjunto train y val definidos en la sección 4.3.1. Los hiperparámetros que se han estudiado son:

- *Epocas:* Una epoca implica un ciclo completo del conjunto de datos de entrenamiento que se compone de lotes e iteraciones del conjunto de datos, y el número de epoch necesarios para que un modelo se ejecute de manera eficiente se basa en los datos en sí y en el objetivo del modelo. [63]. En este proyecto se ha probado solo con 500 epocas, ya que se ha visto que aún cambiando el resto de los hiperparámetros el entrenamiento de los modelos termina antes de llegar a este valor.

- *Batch:* es el número de imágenes procesadas simultáneamente en una pasada hacia delante [17]. Para nuestro proyecto hemos probado con los valores 8, 10, 11, 12 y 16.

- *imgsz:* es el tamaño de la imagen para entrenamiento y validación [21]. En nuestro caso hemos probado con 768 píxeles y 1024 píxeles, viendo que el modelo se comporta mucho mejor con el segundo valor.

La tabla 5.1 indica las diferentes características de los modelos entrenados. El mejor valor obtenido en cada clase se muestra subrayado de color verde.

Modelo	Epocas	Batch	imgsz
Modelo 1	500	16	768
Modelo 2	500	8	1024
Modelo 3	500	10	1024
Modelo 4	500	11	1024
Modelo 5	500	12	1024
Modelo 6	500	16	1024

Tabla 5.1: Características de los modelos entrenados.

Las tablas 5.2, 5.3, 5.4 y 5.5 muestran respectivamente las métricas de precisión, recall, F1 y mAp50 de cada una de las clases obtenidas en cada uno de los modelos, así como, de una llamada "all" que muestra la media ponderada de todas las clases de los distintos modelos .El mejor valor obtenido en cada clase se muestra subrayado de color verde.

Clase	Modelo 1	Modelo 2	Modelo 3	Modelo 4	Modelo 5	Modelo 6
all	**0,695**	**0,746**	**0,844**	**0,812**	**0,643**	**0,797**
Atributos primario	0,802	0,844	0,911	0,97	0,893	0,933
Atributos derivados	0,911	1	0,949	0,931	0,914	0,994
Atributos multivaluados	0,536	0,536	0,721	0,719	0,512	0,744
Atributos simples	0,916	0,938	0,955	0,925	0,922	0,931
Entidad debil	0,861	1	1	1	0,843	1
Entidad fuerte	0,915	0,901	0,905	0,887	0,884	0,902
IS-A no obligatoria/ disjunta	1	0	1	1	0,21	1
IS-A no obligatoria/ no disjunta	1	1	0,846	1	0,724	1
IS-A obligatoria/ disjunta	0,274	0,593	1	0,73	0,316	0,579
IS-A obligatoria/ no disjunta	0,311	0,604	0,387	0,285	0,246	0,369
Lineas	0,592	0,739	0,789	0,699	0,609	0,757
Multiplicidad 0,N	0,175	0,668	0,717	0,66	0,343	0,433
Multiplicidad 1,1	0,475	0,639	0,75	0,708	0,532	0,61
Multiplicidad 1,N	0,462	0,563	0,639	0,582	0,502	0,665
Relacion debil	0,918	0,922	0,95	0,912	0,858	0,859
Relacion fuerte	0,977	0,983	0,983	0,982	0,978	0,983

Tabla 5.2: Métricas de precisión (“P”).

Clase	Modelo 1	Modelo 2	Modelo 3	Modelo 4	Modelo 5	Modelo 6
all	**0,767**	**0,776**	**0,767**	**0,803**	**0,835**	**0,822**
Atributos primario	0,943	0,925	0,943	0,931	0,971	0,914
Atributos derivados	0,9	0,972	1	0,9	0,9	1
Atributos multivaluados	0,667	0,667	0,882	0,875	0,667	0,977
Atributos simples	0,982	0,955	0,955	0,973	0,991	0,964
Entidad debil	0,8	0,763	0,8	0,767	0,8	0,769
Entidad fuerte	1	0,971	0,971	0,971	0,971	0,971
IS-A no obligatoria/ disjunta	0	0	0	0	0,419	0
IS-A no obligatoria/ no disjunta	0,614	0,812	0,667	0,626	0,896	0,863
IS-A obligatoria/ disjunta	1	1	0,904	1	1	1
IS-A obligatoria/ no disjunta	1	1	1	1	1	1
Lineas	0,778	0,765	0,752	0,765	0,747	0,797
Multiplicidad 0,N	0,154	0,31	0,2	0'385	0,385	0,308
Multiplicidad 1,1	0,528	0,54	0,472	0,743	0,664	0,782
Multiplicidad 1,N	0,909	0,731	0,727	0,909	0,955	0,812
Relacion debil	1	1	1	1	1	1
Relacion fuerte	1	1	1	1	1	1

Tabla 5.3: Métricas "R"

Clase	Modelo 1	Modelo 2	Modelo 3	Modelo 4	Modelo 5	Modelo 6
all	**0,729**	**0,760**	**0,804**	**0,807**	**0,728**	**0,809**
Atributos primario	0,867	0,882	0,927	0,950	0,931	0,923
Atributos derivados	0,905	0,986	0,974	0,915	0,906	0,997
Atributos multivaluados	0,588	0,585	0,793	0,791	0,586	0,846
Atributos simples	0,948	0,967	0,958	0,948	0,956	0,948
Entidad débil	0,874	0,869	0,889	0,873	0,800	0,870
Entidad fuerte	0,955	0,985	0,985	0,979	0,975	0,986
IS-A no obligatoria/ disjunta	0,476	0	0,571	0,571	0,277	0,571
IS-A no obligatoria/ no disjunta	0,761	0,896	0,761	0,769	0,801	0,926
IS-A obligatoria/ disjunta	0,432	0,598	0,548	0,846	0,481	0,733
IS-A obligatoria/ no disjunta	0,476	0,753	0,553	0,438	0,394	0,526
Lineas	0,681	0,753	0,770	0,735	0,672	0,789
Multiplicidad 0,N	0,252	0,443	0,315	0,477	0,429	0,496
Multiplicidad 1,1	0,495	0,585	0,592	0,725	0,588	0,687
Multiplicidad 1,N	0,617	0,833	0,778	0,712	0,660	0,767
Relación débil	0,957	0,959	0,975	0,954	0,924	0,924
Relación fuerte	0,988	0,992	0,992	0,991	0,990	0,992

Tabla 5.4: Métricas “F1”

Clase	Modelo 1	Modelo 2	Modelo 3	Modelo 4	Modelo 5	Modelo 6
all	**0,754**	**0,856**	**0,877**	**0,832**	**0,807**	**0,835**
Atributos primario	0,959	0,945	0,974	0,984	0,986	0,97
Atributos derivados	0,926	0,995	0,995	0,978	0,922	0,995
Atributos multivaluados	0,581	0,83	0,83	0,83	0,681	0,83
Atributos simples	0,982	0,983	0,989	0,982	0,977	0,982
Entidad débil	0,871	0,9	0,9	0,888	0,86	0,879
Entidad fuerte	0,988	0,958	0,962	0,983	0,948	0,963
IS-A no obligatoria/ disjunta	0	0,332	0,995	0	0,497	0
IS-A no obligatoria/ no disjunta	0,863	0,995	0,83	0,913	0,746	0,995
IS-A obligatoria/ disjunta	0,995	0,995	0,995	0,995	0,995	0,995
IS-A obligatoria/ no disjunta	0,995	0,995	0,995	0,995	0,995	0,995
Lineas	0,744	0,735	0,785	0,722	0,685	0,755
Multiplicidad 0,N	0,139	0,602	0,418	0,539	0,404	0,527
Multiplicidad 1,1	0,488	0,722	0,659	0,796	0,585	0,73
Multiplicidad 1,N	0,54	0,714	0,71	0,715	0,633	0,75
Relación débil	0,995	0,995	0,995	0,995	0,995	0,995
Relación fuerte	0,995	0,995	0,995	0,995	0,995	0,995

Tabla 5.5: Métricas “mAP50”

5.4.3. Elección del modelo

En la sección anterior se expusieron las métricas de precisión, *recall*, F1 y mAP50 de los 6 modelos entrenados. Debido a que la métrica F1 es una medida que combina la precisión y el *recall* en un único valor, se ha optado por elegir el modelo que mejor resultado proporciona usando esta métrica.

Como se puede ver en la tabla 5.4 el modelo 6 destaca por encima del resto en la métrica F1, siendo el que mejor valor tiene no solo en la clase *all*, sino también en 9 de las 16 clases restantes (atributos derivados; atributos multivaluados; entidad fuerte; IS-A obligatoria y disjunta; IS-A no obligatoria y disjunta; IS-A no obligatoria y no disjunta; líneas; multiplicidad 0,N; multiplicidad 1,1 y relación fuerte). Por ello, se ha elegido el modelo 6 para realizar la identificación de los

distintos componentes de los modelos ER.

5.5. Análisis de los resultados

En este apartado se expone un análisis tanto de la efectividad del modelo propuesto para la identificación de los componentes, como para las reglas utilizadas para conectar los diferentes componentes del modelo ER.

5.5.1. Identificación de los componentes

A continuación, se presenta la tabla 5.6 que contiene un análisis detallado de las métricas "P", "R", "mAP50" y "F1" de los diferentes componentes del modelo 6 seleccionado previamente en la subsección 5.4.3. Los valores superiores a 0.8 se muestran subrayados en verde y los inferiores a 0.5 en rojo. Los valores de las clases de las que más instancias contamos, son justo aquellas cuyos valores son superiores a 0.8 en las tres métricas (atributos primarios, atributos derivados, atributos simples, entidad débil, entidad fuerte, relación débil y relación fuerte). De igual modo, el modelo tiene una mala precisión en aquellas clases en las que contamos con menos instancias (IS-A no obligatoria/ disjunta, IS-A obligatoria/ disjunta y multiplicidades).

5.5.2. Determinación de la conectividad

En esta subsección, se presenta el análisis de los resultados de los tres tipos de conectividad más importantes: Entidad-Atributos, Relación-Atributos y Relación-Entidades. Para cada uno de ellos, se han calculado las métricas "P", "R" y "F1". Para realizar estos cálculos, se ha tomado una muestra aleatoria de 78 imágenes del dataset. Para cada imagen, se ha comparado el fichero "JSON" obtenido por la aplicación sin modificar con el fichero "JSON" editado por el usuario.

Como se muestra en la Tabla 5.7 los resultados obtenidos para la conectividad Entidad-Atributos son muy buenos, superando en las tres métricas el valor de 0,9. La conectividad de Relación-Atributos muestra para las tres métricas valores alrededor de 0,7. Estas son un poco más bajas que las anteriores debido a a que las reglas de conexión en este caso son más estrictas y también a que la cantidad de atributos que contienen las relaciones es muy pequeña en comparación con las que contienen las entidades. En el caso de la conectividad Relación-Entidades, las métricas son bastante buenas: tenemos un valor de 0,82 para la precisión, 0,88 para el "recall" y 0,84 para el "score F1". La conectividad Relación-Entidad-Multiplicidades tiene las métricas más bajas de todas las asociaciones, estando sus valores entre 0,54 y 0,58. Por último, la conectividad Relación-Entidad-Rol tiene las 3 métricas entre 0,8 y 0,9

5.6. Conclusiones

En este capítulo, se ha explicado la elección del modelo para la identificación de las diferentes componentes de los modelos ER y se han expuesto las reglas de conectividad utilizadas para conectar las distintas clases. Además, se ha concluido que tanto en la identificación como en la conectividad de los componentes se obtienen mejores resultados para aquellos componentes que cuentan con más instancias.

Clase	Métrica "P"	Métrica "R"	Métrica "mAP50"	Métrica "F1"
all	**0,629**	**0,818**	**0,746**	**0,712**
Atributos primario	0,829	0,979	0,946	0,898
Atributos derivados	0,776	1	0,994	0,875
Atributos multivaluados	0,602	0,778	0,885	0,679
Atributos simples	0,882	0,965	0,979	0,921
Entidad débil	0,904	1	0,993	0,95
Entidad fuerte	0,911	1	0,995	0,953
IS-A no obligatoria/ disjunta	0,265	0,368	0,348	0,307
IS-A no obligatoria/ no disjunta	0,454	1	0,847	0,624
IS-A obligatoria/ disjunta	0,331	0,989	0,636	0,497
IS-A obligatoria/ no disjunta	0,5	0,6	0,712	0,545
Lineas	0,561	0,821	0,768	0,666
Multiplicidad 0,1	0,335	0,2	0,239	0,25
Multiplicidad 0,N	0,395	0,839	0,578	0,535
Multiplicidad 1,1	0,366	0,76	0,49	0,493
Multiplicidad 1,N	0,431	0,924	0,514	0,589
Relación débil	1	0,997	0,995	0,998
Relación fuerte	0,962	1	0,995	0,981
Rol	0,815	0,5	0,507	0,622

Tabla 5.6: Métricas "P", "R", "mAP50" y "F1" de los datos de prueba del modelo seleccionado.

Tipo de conectividad	Métrica "P"	Métrica "R"	Métrica "F1"
Entidad-Atributos	0,90	0,98	0,94
Relación-Atributos	0,72	0,70	0,71
Relación-Entidades	0,82	0,88	0,84
Relación-Entidad-Multiplicidad	0,54	0,58	0,55
Relación-Entidad-Rol	0,81	0,87	0,83

Tabla 5.7: Métricas "P", "R" y "F1" de las conectividades Entidad-Atributos, Relación-Atributos, Relación-Entidad-Rol y Relación-Entidad-Multiplicidad.

Capítulo 6

Procesamiento de los contenidos textuales del modelo E-R

En este capítulo se explica cómo se ha llevado a cabo el reconocimiento óptico de caracteres, cómo se ha conectado el texto con los diferentes componentes del modelo ER detectados en la sección 5.2 y se expone un análisis de la efectividad del reconocimiento óptico de los caracteres.

Se inicializa con una breve introducción 6.1,se continúa ofreciendo una descripción del procesamiento de los contenidos textuales de los modelos ER. Posteriormente, en la subsección 6.2 se explica como se han detectado los contenidos textuales de los modelos ER y en la subsección 6.3 se explican las reglas utilizadas para conectar este texto identificado con los diferentes componentes detectados previamente en la sección 5.2. Estas secciones, corresponden respectivamente a los pasos 4 y 5 del pipeline de la aplicación que se muestra en la figura 6.1.

Para finalizar este capítulo, en la sección 6.4 se presenta un análisis de la lectura de texto y en la 6.5 unas breves conclusiones.

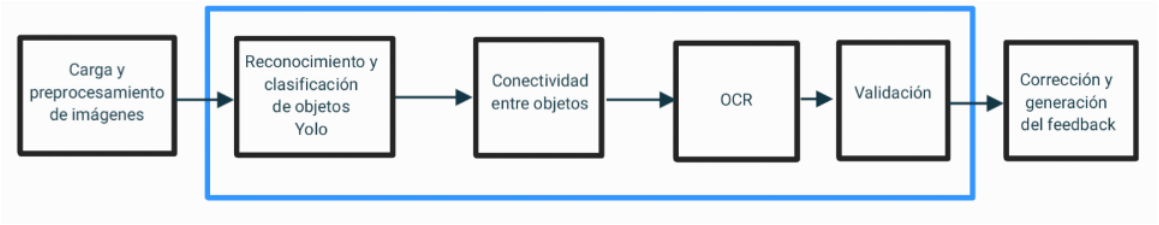


Figura 6.1: Workflow de la aplicación.

6.1. Introducción

El procesamiento de los contenidos textuales de una imagen con un modelo ER se realiza en dos fases. Primero se identifican y leen los diferentes caracteres de la imagen, obteniendo una lista con las coordenadas de localización de cada contenido textual identificado junto con los caracteres leídos. Después, se conectan estos contenidos con los componentes (entidades, relaciones, atributos y roles) previamente identificados en la sección 5.2. Obteniendo como resultado final de este proceso un fichero *json* con toda la información estructurada del modelo ER de la imagen.

6.2. Lectura de los contenidos textuales

En este cuarto paso del workflow de la aplicación, se detectan todos los contenidos textuales presentes en la imagen, incluyendo palabras, números y símbolos. Debido a que los modelos ER están escritos a mano por diferentes estudiantes y profesores, nos encontramos con dos grandes desafíos:

1. **Diferentes caligrafías:** Cada persona cuenta con una caligrafía distinta, por ello a la hora de detectar el contenido textual de los modelos ER nos vamos a encontrar con algunas letras más legibles que otras. Nuestro objetivo es conseguir detectar correctamente la mayor cantidad de caracteres independientemente de la caligrafía del usuario.

2. **Orientación del texto:** El texto en las imágenes puede estar dispuesto en diferentes orientaciones. En consecuencia, tenemos que asegurarnos que la *Api* de reconocimiento óptico de caracteres que usemos para detectar los caracteres pueda identificarlos en cualquier dirección.

Para realizar este proceso se pasa la imagen con el modelo ER a la *Api* de *Google Vision* y esta devuelve un fichero *json* con todos los componentes textuales detectados junto con las coordenadas donde los ha detectado. La imagen 6.2 muestra un fragmentos del fichero *json* generado con los caracteres identificados y sus respectivas coordenadas. En este vemos que se han detectado las palabras: "CLIENTE", "BANCO", "Fecha" y "REALIZAR". En la imagen 6.3 podemos ver todos los caracteres leídos pintados con sus respectivas *bounding boxes* sobre la imagen del modelo ER.

```
{
    "texto": "CLIENTE",
    "coords": {
        "x": 1.078,
        "y": 0.152,
        "width": 0.205,
        "height": 0.051
    }
},
{
    "texto": "Banco",
    "coords": {
        "x": 1.446,
        "y": 0.15,
        "width": 0.111,
        "height": 0.051
    }
},
```

```
{
    "texto": "Fecha",
    "coords": {
        "x": 1.076,
        "y": 0.251,
        "width": 0.074,
        "height": 0.023
    }
},
{
    "texto": "REALIZAR",
    "coords": {
        "x": 0.875,
        "y": 0.294,
        "width": 0.161,
        "height": 0.028
    }
},
```


Figura 6.2: Fragmento del fichero *json* con los caracteres detectados y sus coordenadas.

6.3. Asociación de los componentes y los contenidos textuales

En este punto, tenemos cada componente identificado en la imagen guardado como un identificador único, junto con el tipo de etiqueta y sus coordenadas de posición en la imagen en un fichero *json*. En la imagen 6.4 se muestra a la izquierda un fragmento de este fichero de las entidades identificadas y a la derecha de las relaciones. Así como, para cada componente textual

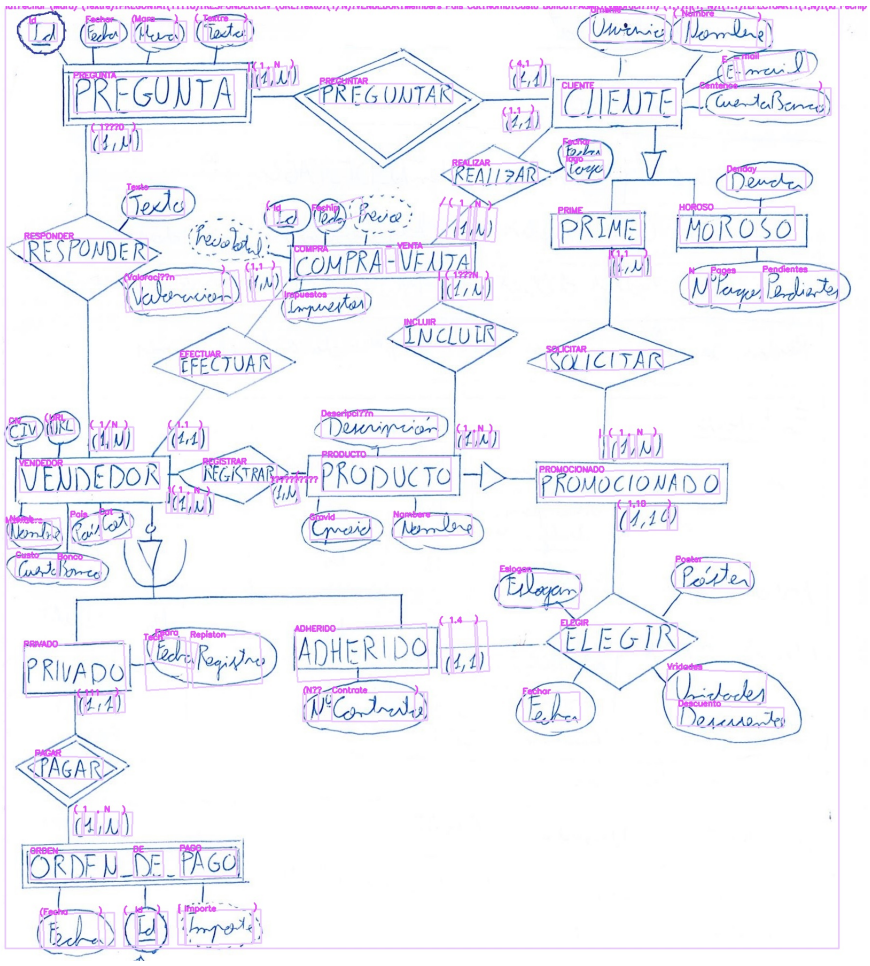


Figura 6.3: Ejemplo de caracteres identificados y pintados sobre el modelo ER.

tenemos guardado su texto junto a sus respectivas coordenadas en otro fichero *json*. La imagen 6.2 muestra un fragmentos de este segundo fichero *json*.

Para realizar la asociación de los componentes entidades, relaciones, atributos y roles se ha comprobado que contenidos textuales se encontraban dentro de estos componentes y se ha añadido dichos contenidos textuales a la etiqueta *"name"* de cada componente. Esto se ha realizado mediante la intersección de las coordenadas de los contenidos textuales del segundo fichero con las coordenadas de los componentes entidades, relaciones, atributos y roles del primer fichero. En el caso de que en un mismo componente hubiese más de un elemento textual detectado, se agregarían ambos separados por una coma. La imagen 6.5 presenta un fragmento del fichero *json* con las entidades (a la izquierda) y las relaciones (a la derecha) identificadas con sus correspondientes contenidos textuales.

La imagen 6.6 muestra sobre el modelo ER los diferentes componentes detectados junto a sus contenidos textuales identificados. Las relaciones están coloreadas en azul, las entidades en verde y los atributos en rojo.

```
"entidades": [
  {
    "id": "ENT-001",
    "name": "",
    "type": "fuerte",
    "coords": {
      "x": 0.439252,
      "y": 0.671743,
      "width": 0.179263,
      "height": 0.0595294
    }
  },
  {
    "id": "ENT-002",
    "name": "",
    "type": "fuerte",
    "coords": {
      "x": 0.858921,
      "y": 0.235207,
      "width": 0.16686,
      "height": 0.0526861
    }
  },
  {
    "id": "ENT-003",
    "name": "",
    "type": "fuerte",
    "coords": {
      "x": 0.719451,
      "y": 0.0999977,
      "width": 0.165977,
      "height": 0.069048
    }
  },
```

```
"relaciones": [
  {
    "id": "REL-008",
    "name": "",
    "type": "debil",
    "coords": {
      "x": 0.463086,
      "y": 0.103483,
      "width": 0.218522,
      "height": 0.135798
    }
  },
  {
    "id": "REL-009",
    "name": "",
    "type": "debil",
    "coords": {
      "x": 0.112162,
      "y": 0.796731,
      "width": 0.143246,
      "height": 0.0838392
    }
  },
  {
    "id": "REL-001",
    "name": "",
    "type": "fuerte",
    "coords": {
      "x": 0.724112,
      "y": 0.663044,
      "width": 0.186319,
      "height": 0.110967
    }
  },
```


Figura 6.4: Fragmento del fichero *json* con las entidades (a la izquierda) y relaciones (a la derecha) identificadas.

6.4. Evaluación

El procesamiento de los contenidos textuales de los modelos ER se ha realizado a través de la *Api* de *Google Vision*. A esta *Api* le hemos pasado la imagen con el modelo ER original y nos ha devuelto una lista con los contenidos textuales y las coordenadas de donde han sido detectados. Esta lista la hemos guardado en un fichero *json*. Para finalizar, hemos añadido los distintos contenidos textuales de este fichero a los componentes entidades, relaciones, atributos y roles como se explica en la sección 6.3.

La Api de *Google Vision* detecta el texto de la imagen palabra a palabra. Por ello, cuando un componente tiene un nombre compuesto, suele presentar más fallos en su detección y posterior asociación al componente, ya que tiende a asociar solo una de las palabras. Además, es importante recordar que las imágenes son modelos manuscritos por estudiantes en exámenes, por lo que la letra de cada modelo es diferente, hay muchas palabras que se encuentran escritas en vertical. Algunas palabras incluso pueden resultar difíciles de identificar para una persona. Considerando todos estos aspectos, se puede concluir que la API detecta los caracteres con precisión, aunque presenta dificultades con las tildes, las palabras escritas en horizontal y en los modelos donde la caligrafía del estudiante es más complicada de leer para las personas.

Por otra parte, observamos que las reglas definidas para asociar el texto con los componentes se comportan muy bien, ya que se asocian prácticamente todas las palabras.

```
"entidades": [
    {
        "id": "ENT-001",
        "name": "ADHERIDO",
        "type": "fuerte",
        "coords": {
            "x": 0.439252,
            "y": 0.671743,
            "width": 0.179263,
            "height": 0.0595294
        }
    },
    {
        "id": "ENT-002",
        "name": "MOROSO",
        "type": "fuerte",
        "coords": {
            "x": 0.858921,
            "y": 0.235207,
            "width": 0.16686,
            "height": 0.0526861
        }
    },
    {
        "id": "ENT-003",
        "name": "CLIENTE",
        "type": "fuerte",
        "coords": {
            "x": 0.719451,
            "y": 0.0999977,
            "width": 0.165977,
            "height": 0.069048
        }
    },
```

```
"relaciones": [
    {
        "id": "REL-008",
        "name": "PREGUNTAR",
        "type": "debil",
        "coords": {
            "x": 0.463086,
            "y": 0.103483,
            "width": 0.218522,
            "height": 0.135798
        }
    },
    {
        "id": "REL-009",
        "name": "PAGAR",
        "type": "debil",
        "coords": {
            "x": 0.112162,
            "y": 0.796731,
            "width": 0.143246,
            "height": 0.0838392
        }
    },
    {
        "id": "REL-001",
        "name": "ELEGIR, Unidades",
        "type": "fuerte",
        "coords": {
            "x": 0.724112,
            "y": 0.663044,
            "width": 0.186319,
            "height": 0.110967
        }
    },
```


Figura 6.5: Fragmento del fichero *json* con las entidades (a la izquierda) y relaciones (a la derecha) identificadas con sus correspondientes contenidos textuales.

6.5. Conclusiones

En este capítulo, se ha explicado cómo se ha llevado a cabo el reconocimiento óptico de caracteres (OCR) de las diferentes imágenes con modelos E-R, así como, se han descrito las reglas utilizadas para conectar el texto detectado con las entidades, relaciones, atributos, roles y multiplicidades previamente detectadas. La Api de *Google Vision* detecta el texto de las imágenes bastante bien, aunque suele fallar en aquellos caracteres que contienen tilde, las palabras escritas en horizontal y aquellas que cuentan con una mala caligrafía. Por otro lado, podemos concluir que la asociación del texto detectado con los componentes identificados es muy buena.

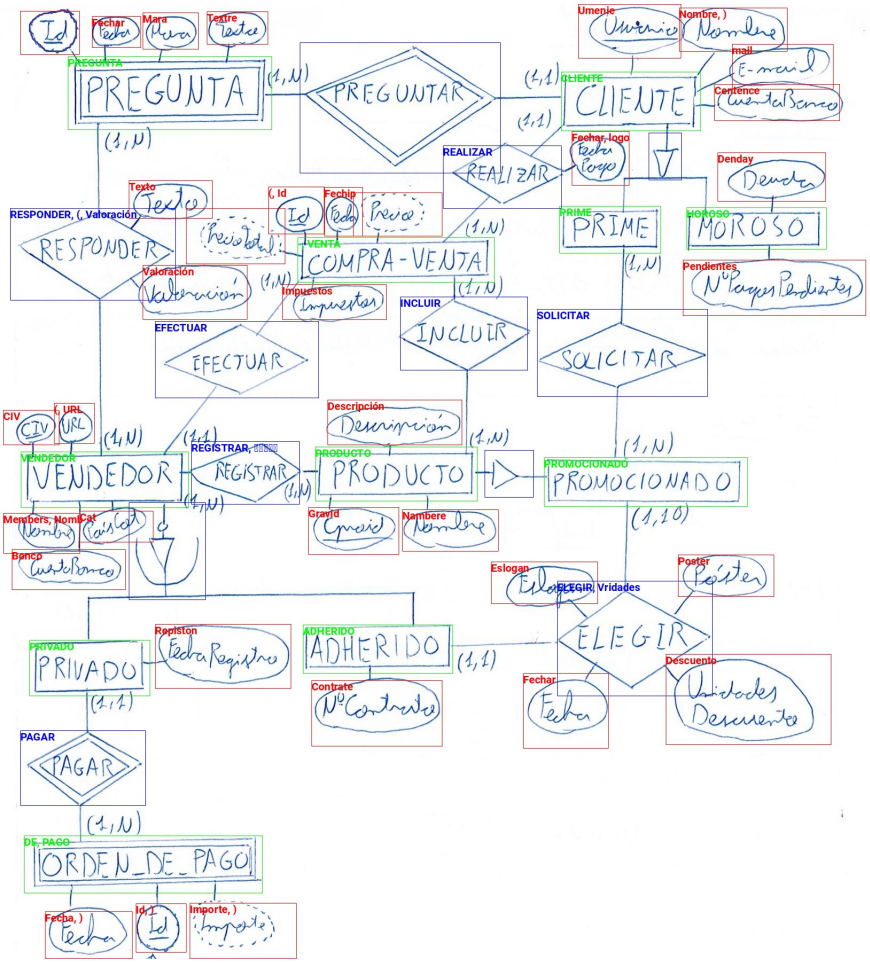


Figura 6.6: Ejemplo de asociación de los componentes detectados de un modelo E-R a los contenidos textuales identificados.

Capítulo 7

Validación del modelo con el usuario

En este capítulo presentamos el subsistema de validación del modelo reconocido. Este subsistema comprende un conjunto de páginas web que presentan los servicios necesarios para que el profesor o el estudiante proporcione una imagen con el modelo ER y corrija los errores o añadan aquellos elementos tanto visuales como de texto que no haya reconocido de forma correcta el modelo. La implementación de este componente se ha realizado con Vue y Quasar, utilizando CSS para dar estilo a las páginas.

En la sección 7.1 se muestra la página de inicio, en la 7.2 las páginas de componentes, en la 7.3 las páginas de conectividades y en la 7.4 la página para subir la solución el profesor.

7.1. Página de inicio

La página de inicio es la primera pantalla que ve cualquier usuario (profesor o estudiante) que acceda en la aplicación. En esta se le pide que proporcione una foto, utilizando para ello un botón con un símbolo "+" situado en la esquina inferior derecha, como se puede ver en la imagen 7.7. Una vez el usuario pincha dicho botón la aplicación le solicita que introduzca su rol de usuario (profesor o estudiante) y le muestra el botón de "Continuar" (imágenes 7.2 y 7.3). Este botón solo funciona si el usuario ha seleccionado el rol. En caso de haber seleccionado rol de profesor se le pedirá que introduzca el nombre del ejercicio (imagen 7.4) y si ha seleccionado estudiante, entonces se le mostrará un desglosable con todos los nombres de los ejercicios para los cuales previamente un profesor ha subido una solución (imagen 7.5). En ambos casos, cuando el usuario le da a continuar la aplicación le pedirá que suministre la imagen (en formato png o jpg) que contiene el modelo ER (imagen 7.6). Una vez el usuario ha seleccionado la imagen deseada, la aplicación inicia el proceso de validación, en el que se presenta la imagen original en la parte izquierda de la pantalla y la imagen de trabajo (sobre la que se llevará a cabo el proceso de validación) en la parte derecha, como se puede ver en la imagen 7.7.

Figura 7.1: Página de bienvenida.

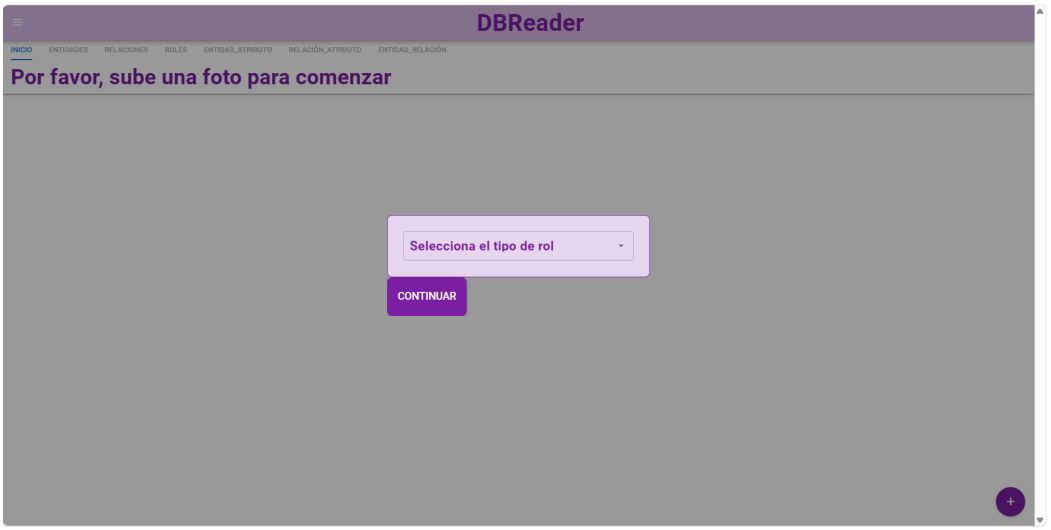


Figura 7.2: Seleccionar rol.

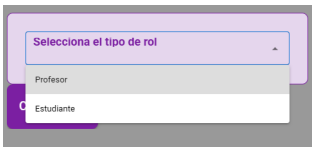


Figura 7.3: Desplegar roles.

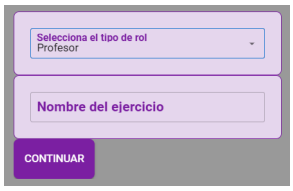


Figura 7.4: Introducir nombre de ejercicio.

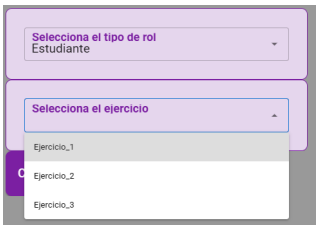


Figura 7.5: Seleccionar nombre de ejercicio.

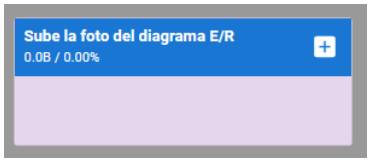


Figura 7.6: Seleccionar imagen.

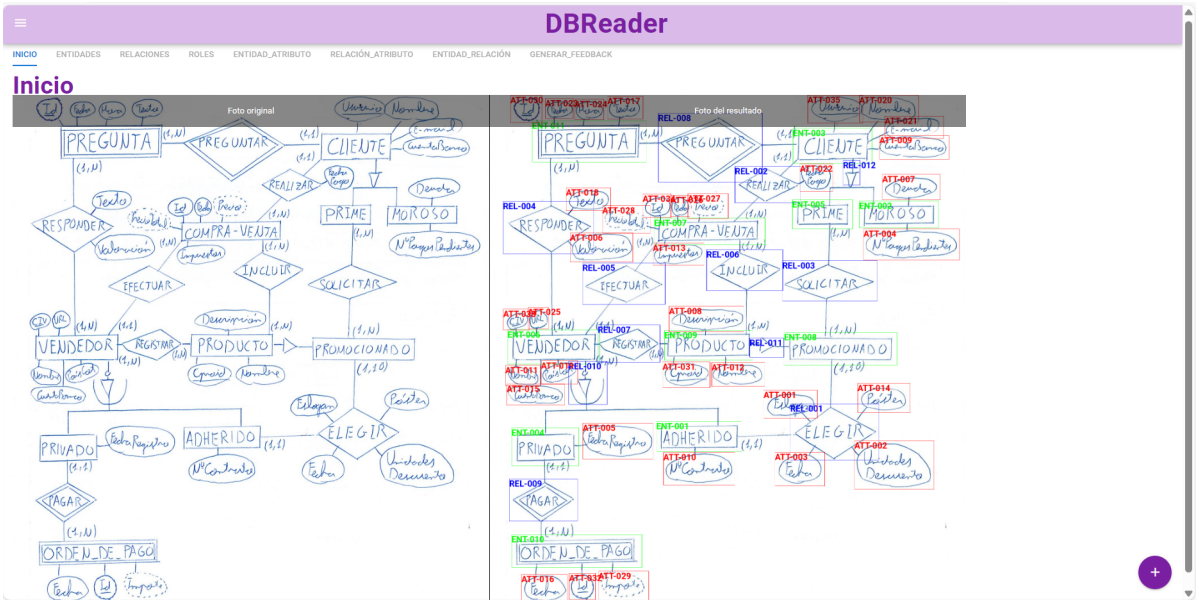


Figura 7.7: Página de inicio.

7.2. Páginas de componentes

En esta sección se presentan las páginas que contienen la información de los diferentes componentes (entidades, relaciones y roles) del modelo ER. Las páginas de entidades y relaciones tienen la misma estructura. En ellas se le muestra al usuario (a la izquierda) la imagen del modelo ER con las bounding boxes de las entidades o relaciones detectadas, junto al identificador que las representa. A la derecha presenta una tabla con el id, el nombre y el tipo de la relación o entidad. Para facilitar al usuario la identificación de los diferentes tipos de entidades o relaciones, se muestran en rojo los componentes débiles, en azul los fuertes y las relaciones IS-A en marrón. La página con la información relativa a los roles usados en las relaciones (cuando corresponda) muestra la etiqueta identificativa y el nombre del rol correspondiente. Estas páginas permiten al usuario eliminar componentes, modificar la información errónea que detecte el usuario en las columnas tipo y nombre, así como añadir un nuevo componente. Las imágenes 7.8, 7.10 y 7.12 muestran las interfaces de las páginas entidades, relaciones y roles respectivamente antes de ser modificadas. Mientras que las figuras 7.9 y 7.11 muestran las páginas entidades y relaciones respectivamente después de que el usuario ha modificado los fallos detectados en ellas. En ambos casos, el usuario solo ha modificado el valor de la columna "Nombre". En la tabla 7.1 se observan los cambios que ha hecho el usuario en ambas páginas.

ID	Nombre inicial	Nombre actual
ENT-007	-,VENTA	COMPRA-VENTA
ENT-010	DE,PAGO	ORDEN-DE-PAGO
ENT-011	PREGUNTA,I	PREGUNTA
REL-001	ELEGIR,Unidades	ELEGIR
REL-004	RSPONDER,(,Valeracion	RESPONDER
REL-007	REGISTRAR,(,FIN	REGISTRAR

Tabla 7.1: Modificaciones de las tablas de las páginas entidades y relaciones

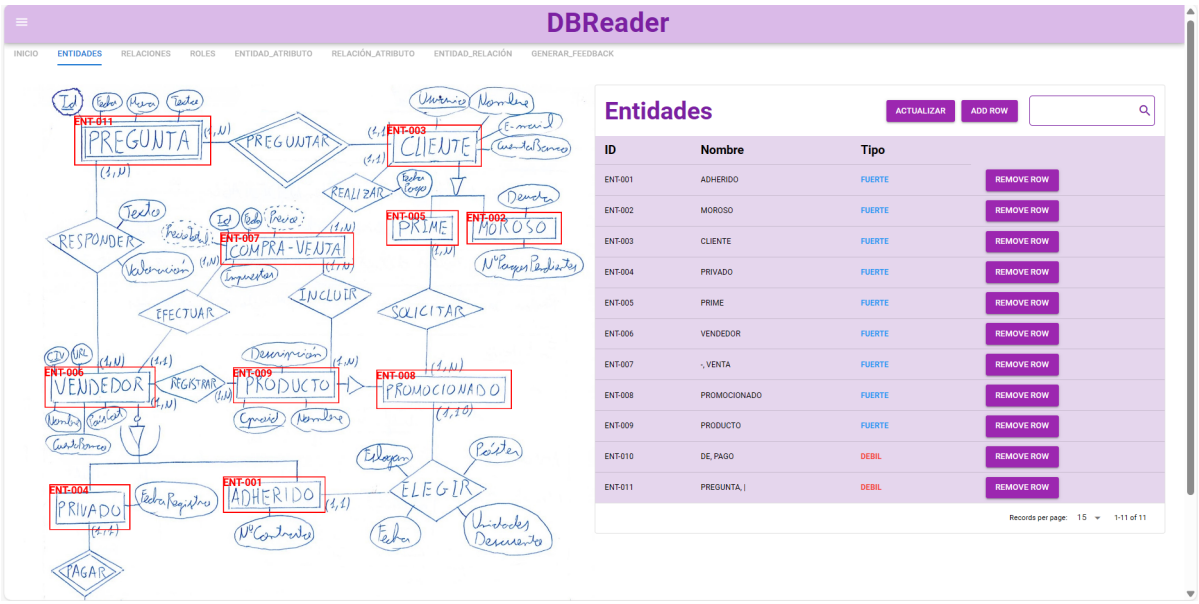


Figura 7.8: Página de entidades antes de ser modificada.

7.3. Páginas de conectividades

En esta sección se presentan las páginas en las que el usuario puede corregir los errores cometidos por el modelo en la conectividad de los componentes. Para ello se muestra al usuario tres páginas distintas, una con cada tipo de conectividad (entidad-atributos, relación-atributos, relación-entidad).

7.3.1. Página conectividad entidad-atributos

Esta página presenta a la izquierda la imagen del modelo E-R con las bounding boxes de todas las entidades y atributos detectados, y a la derecha el listado con el nombre de todas las entidades detectadas. Al pinchar el usuario en cada una de estas entidades, se desglosa una tabla con los atributos asociados a ella. En cada una de estas tablas el usuario puede agregar un

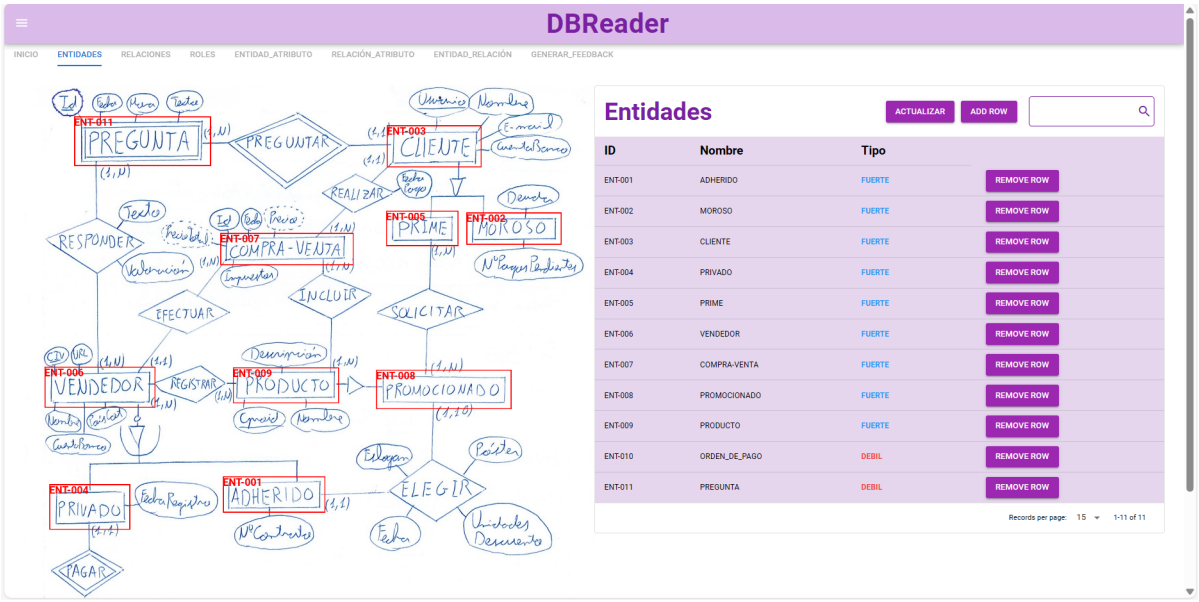


Figura 7.9: Página de entidades después de ser modificada.

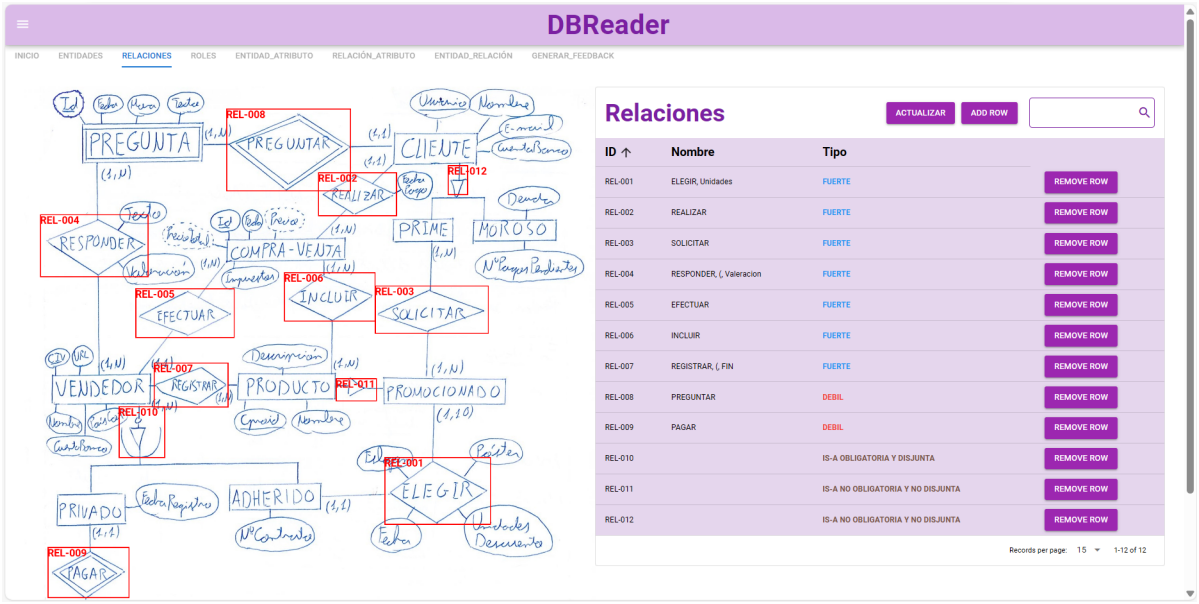


Figura 7.10: Página de relaciones antes de ser modificada.

nuevo atributo, eliminar o modificar la información de cualquier atributo de la tabla. La imagen 7.8 muestra la página de conectividad entidad-atributos antes de ser modificada por el usuario, mientras que la imagen 7.9 presenta la misma página con las correcciones del usuario.

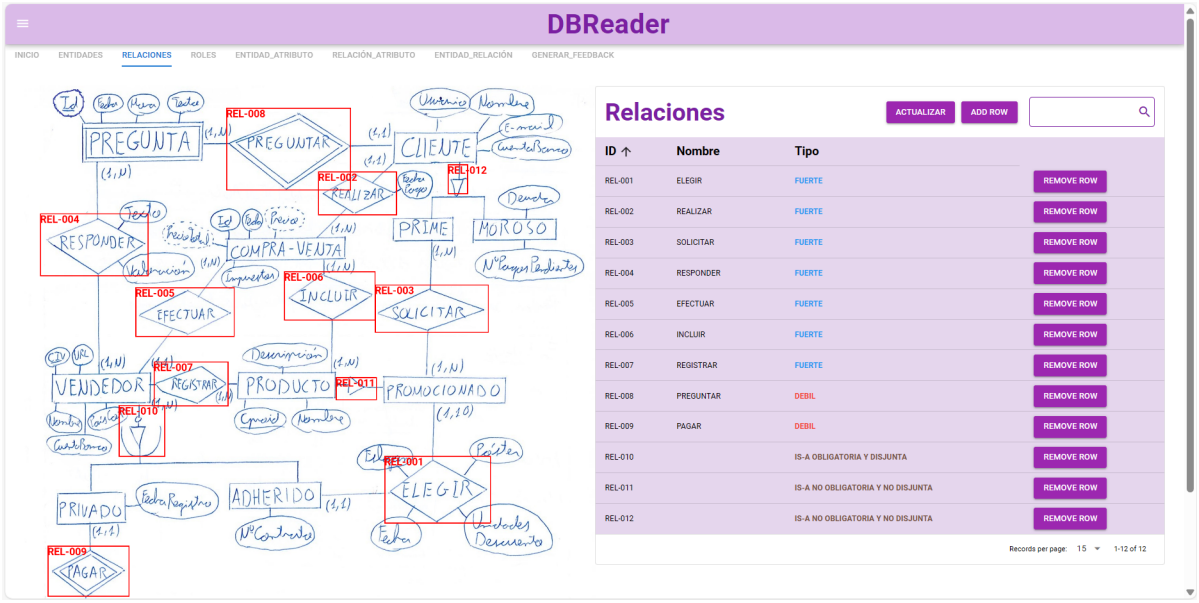


Figura 7.11: Página de relaciones después de ser modificada.

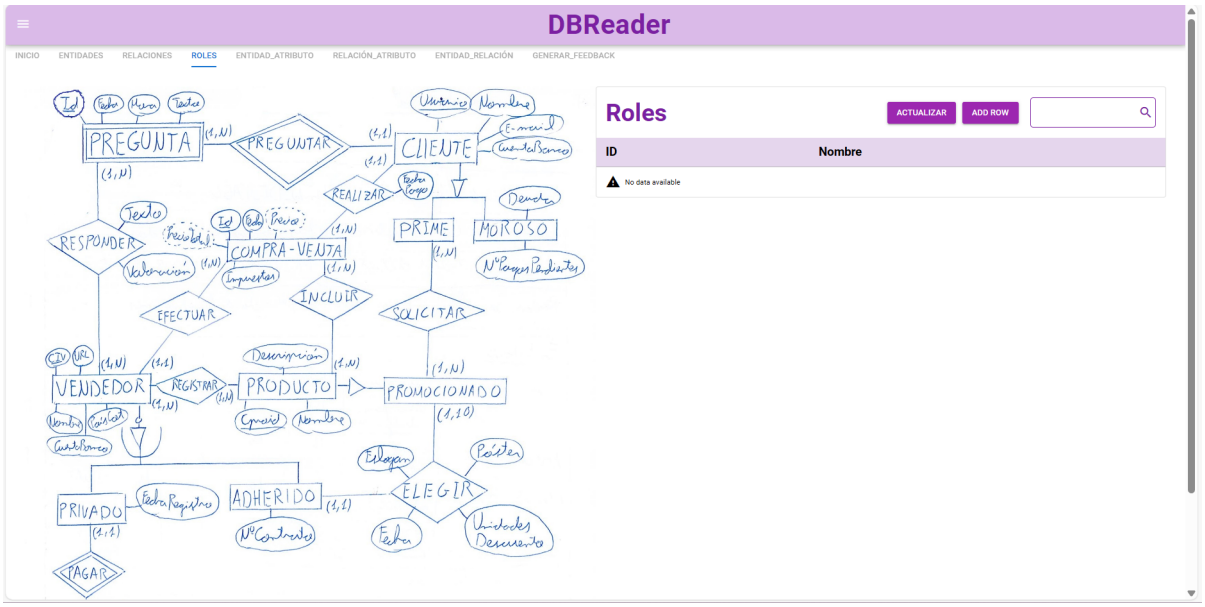


Figura 7.12: Página de roles.

7.3.2. Página conectividad relación-atributo

Esta pantalla muestra a la izquierda la imagen del modelo E-R con las bounding boxes de todas las relaciones y atributos detectados pintados. A la derecha presenta una tabla con las conectividades relación-atributo identificadas. Para cada asociación muestra el id de la relación, el id del atributo, el nombre del atributo y el tipo del mismo. Para facilitar la visualización de las relaciones con id impar aparecen en la tabla en color azul, mientras que las pares aparecen

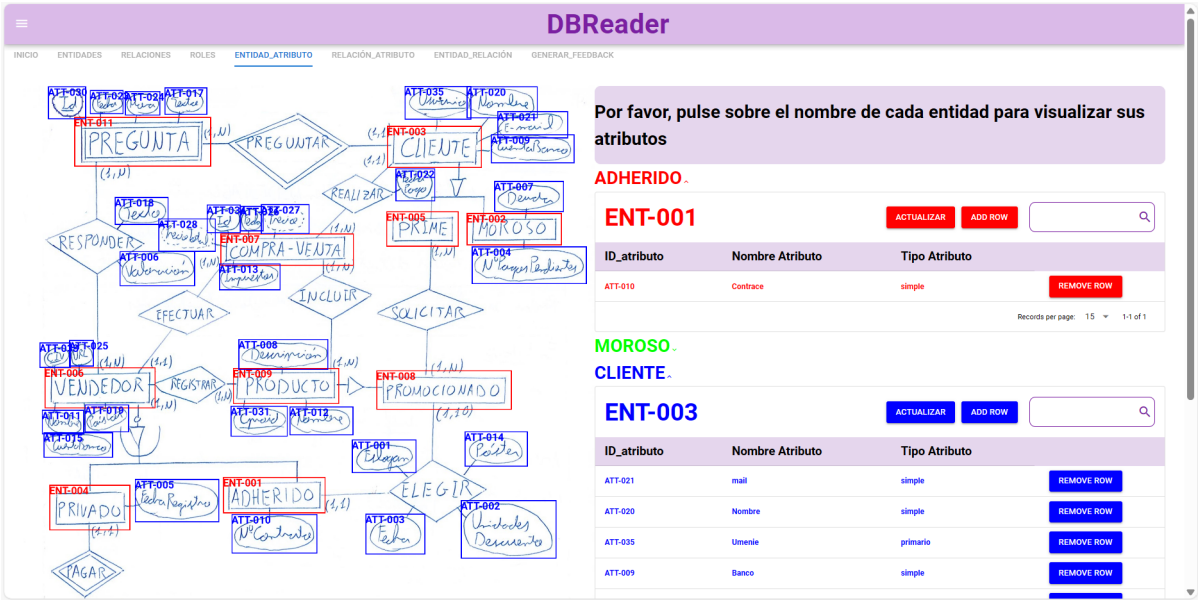


Figura 7.13: Página de conectividad de entidades-atributos antes de ser modificada.

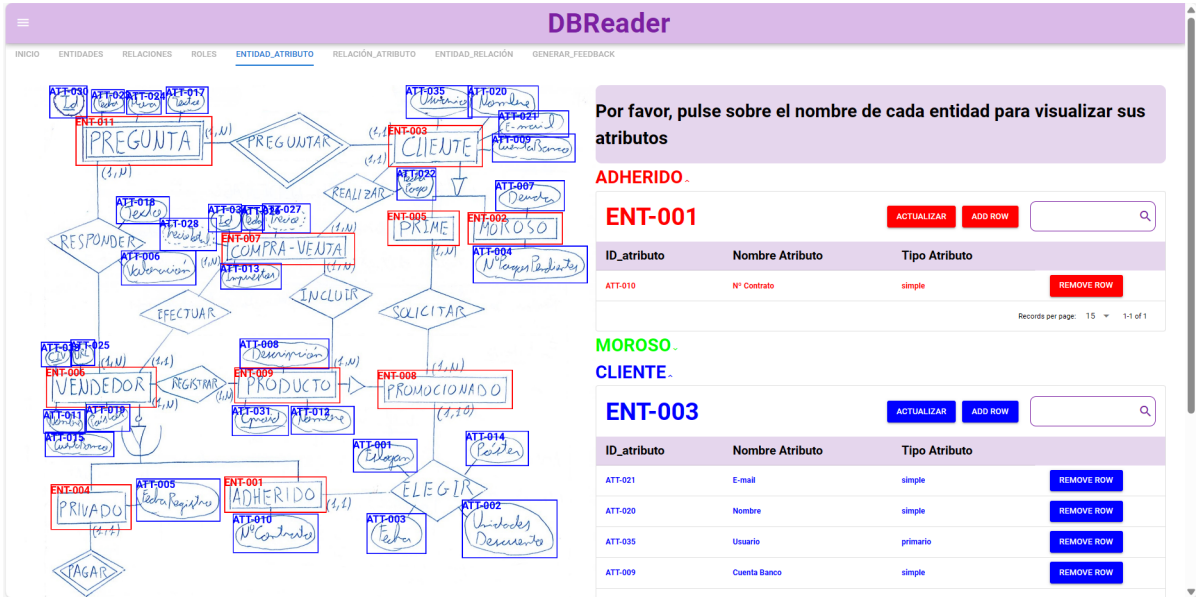


Figura 7.14: Página de conectividad de entidades-atributos después de ser modificada.

en color rojo. El usuario puede agregar una nueva asociación relación-atributo, eliminar una ya existente o modificar la información de cualquier celda de la tabla. La imagen 7.15 muestra la página de conectividad relación-atributo antes de ser modificada por el usuario, mientras que la imagen 7.16 presenta la misma página con las correcciones del usuario.

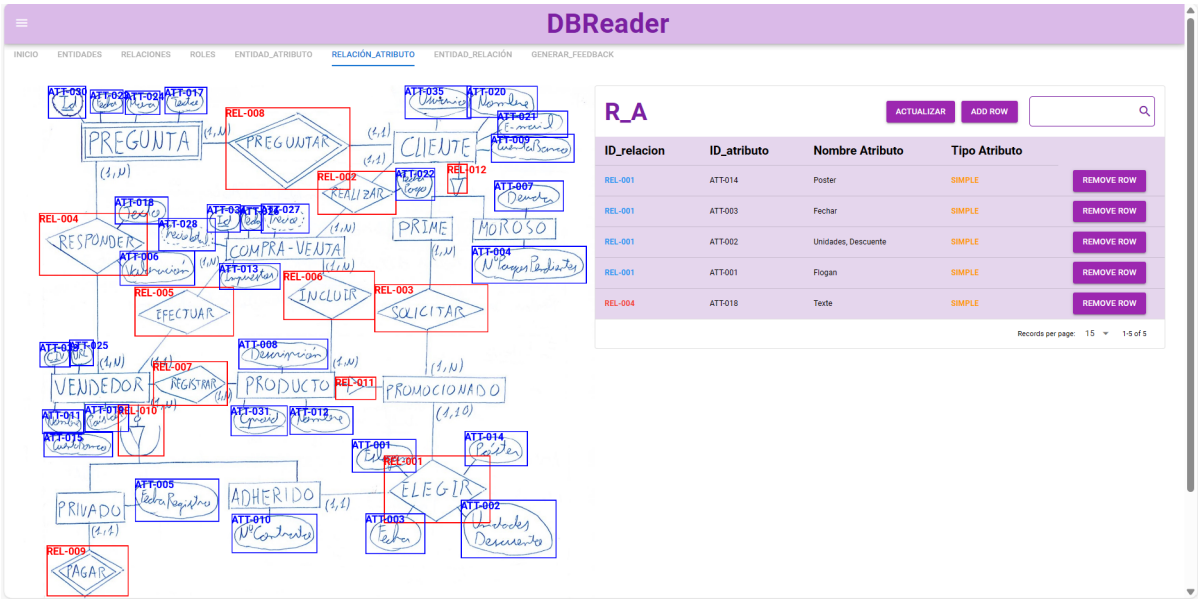


Figura 7.15: Página de conectividad de relaciones-atributos antes de ser modificada.

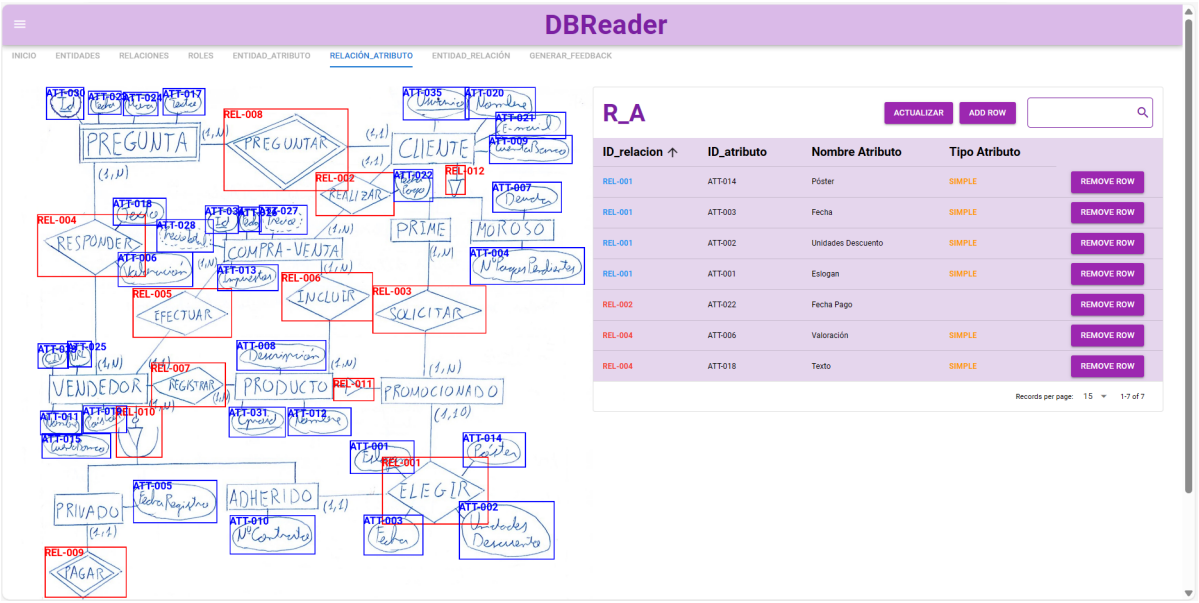


Figura 7.16: Página de conectividad de relaciones-atributos después de ser modificada.

7.3.3. Página conectividad relación-entidad

Esta página presenta al usuario a la derecha una tabla con todas las conectividades relación-entidad detectadas. Para cada una de ellas se muestra el id de la relación, el id de la entidad, la multiplicidad asociada a la relación y el id del rol en caso de que la relación sea unaria. Para facilitar la visualización de la información en la tabla, aparecen las relaciones con id impar en color azul y las relaciones con id par aparecen en color rojo. A la izquierda se vuelve a mostrar

la imagen del modelo E-R, esta vez con las bounding-boxes de todas las entidades y relaciones detectadas. De nuevo el usuario puede modificar la información de las celdas, así como eliminar y o agregar una fila a la tabla. La imagen 7.17 muestra la página de conectividad relación-entidad antes de ser modificada por el usuario, mientras que la imagen 7.18 presenta la misma página con las correcciones del usuario.

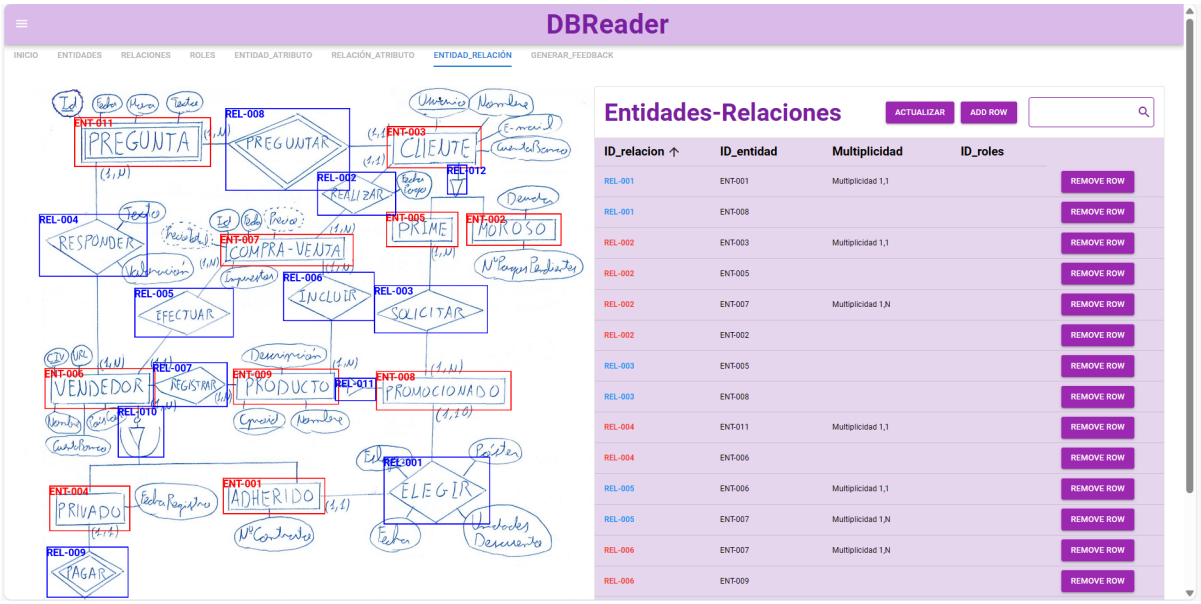


Figura 7.17: Página de conectividad de relaciones-entidades antes de ser modificada.

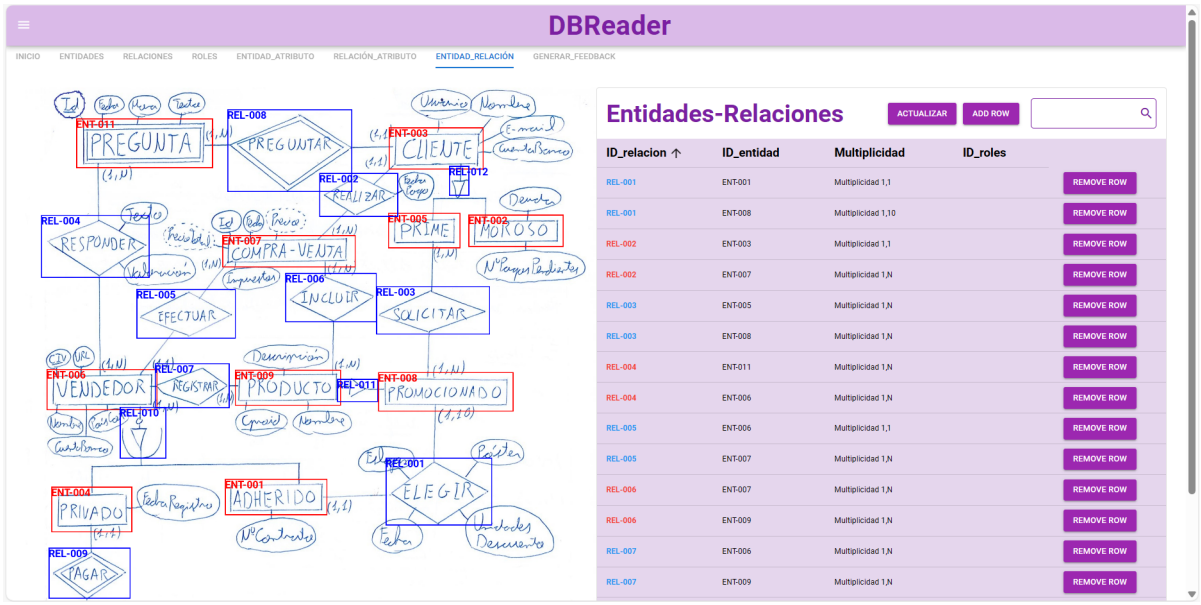


Figura 7.18: Página de conectividad de relaciones-entidades después de ser modificada.

7.4. Página subir solución profesor

Una vez el profesor ha corregido todos los errores, tanto de texto como visuales, accede a esta página al darle a la pestaña de "subir-solución". Al entrar en esta desaparecen las pestañas de las otras páginas y aparece un botón para guardar la solución y otro para cancelar (imagen 7.19). Si el profesor pulsa el botón "Guardar solución", el sistema almacena la información correspondiente al nuevo ejercicio y se le pregunta si desea subir otro ejercicio (imagen 7.20). Tanto si pulsa este botón como si pulsa el de "Cancelar" se le volverá a redirigir a la página de bienvenida.

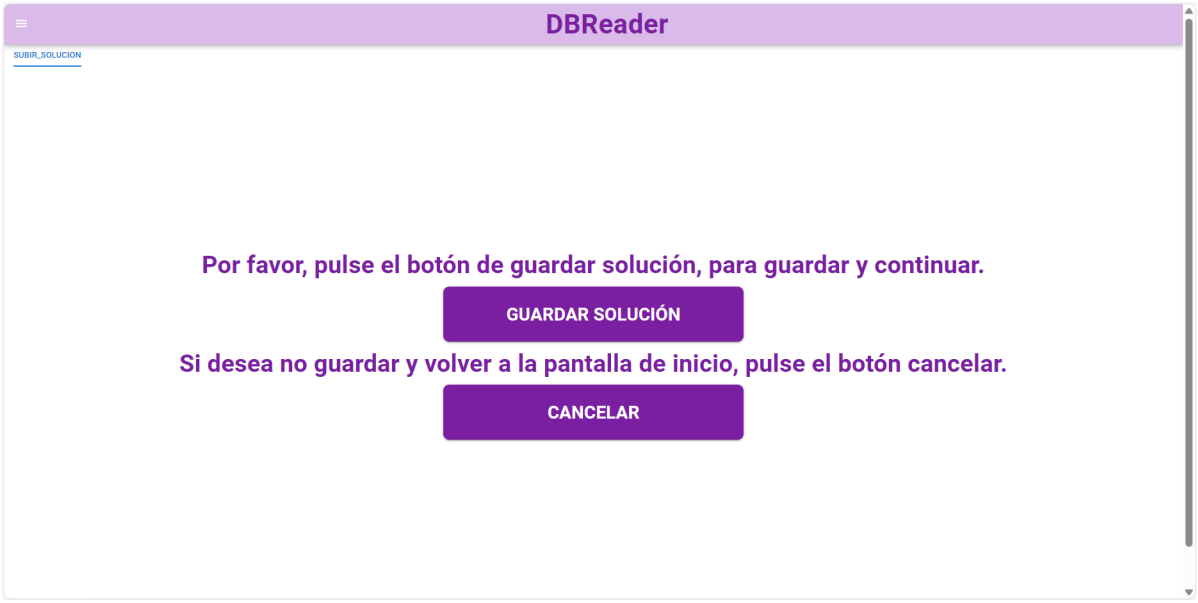


Figura 7.19: Página para subir la solución el profesor.

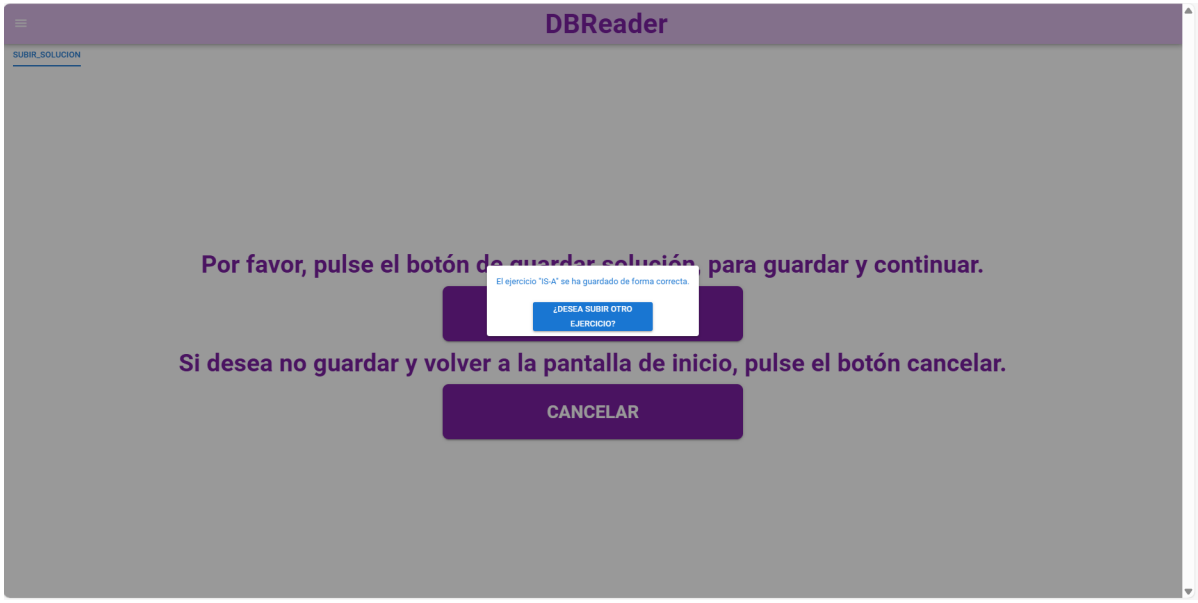


Figura 7.20: Página que pide aceptación al profesor para subir el ejercicio.

Capítulo 8

Evaluación del modelo y generación de *feedback*

En este capítulo se presenta el subsistema responsable de generar la retroalimentación al estudiante, este muestra por pantalla una tabla con el *feedback* generado. En la sección 8.1 se explican los diferentes criterios utilizados para evaluar el modelo y en la sección 8.2 se expone como se ha generado el *feedback*.

8.1. Evaluación del modelo

La evaluación de los modelo ER se ha realizado siguiendo la metodología UVAGILE [71] utilizada en la asignatura de Sistema de Bases de Datos del Grado de Ingeniería Informática de Servicios y Aplicaciones impartida en la Universidad de Valladolid. La evaluación del modelo ER en esta asignatura se lleva a cabo sobre 8 criterios de aceptación, que miden de forma independiente los diferentes aspectos que caracterizan el citado modelo. Para cada uno de ellos se calcula un porcentaje de acierto. Y se compara respecto a los valores de evaluación planteados: habitualmente, se considera que si la evaluación del criterio es inferior al 50 %, el aprendizaje alcanzado por el estudiante es insuficiente (rojo), mientras que si es superior al 60 % (verde), se considera aceptable. El rango entre 50-60 % se utiliza para buscar un efecto motivador en los estudiantes (amarillo), que aún sin alcanzar el mínimo del 60 %, presentan un aprendizaje valioso. En términos generales, la aceptación del modelo ER requiere que todos los criterios de aceptación sean aceptados, aunque en algunos casos pueden aceptarse algunos criterios en amarillo, para incentivar a los estudiantes a seguir mejorando su aprendizaje. A continuación, se explican los 8 criterios que utilizan para evaluar el modelo ER de cada estudiante.

1. **Criterio 1.6:** Al menos el 60 % de los nombres de Entidades, Relaciones y Atributos utilizados en el modelo ER respetan la sintaxis establecida en la asignatura. Las Entidades y las Relaciones están escritas en mayúscula, mientras que los Atributos tienen la primera letra en mayúscula y el resto en minúsculas.

2. **Criterios 2.3 y 2.4:** El modelo identifica correctamente, al menos el 60 % de las Entidades ("regulares" y "débiles") necesarias para abordar los requisitos establecidos en el pliego técnico del ejercicio.

3. **Criterio 2.5:** Las Entidades en el modelo ER declaran correctamente, al menos, el 60 % de los Atributos que caracterizan su información, de acuerdo con los requisitos establecidos en el pliego técnico del ejercicio.

4. **Criterio 2.6:** Al menos el 60 % de las Entidades planteadas en el modelo ER declaran un identificador válido, de acuerdo con los requisitos establecidos en el pliego técnico del problema.

5. **Criterio 2.7:** El modelo ER identifica correctamente, al menos, el 60 % de las Relaciones esperadas de acuerdo con el pliego técnico del ejercicio.

6. **Criterio 2.8:** El modelo ER establece correctamente las multiplicidades que describen a las Entidades participantes, al menos, el 60 % de las Relaciones esperadas de acuerdo con el pliego técnico del problema.

7. **Criterio 4.1:** El modelo ER determina correctamente, al menos, el 60 % de las Relaciones IS-A necesarias para abordar los requisitos en el pliego técnico del ejercicio.

8. **Criterio 4.2:** El modelo ER determina correctamente, al menos, el 60 % de las Relaciones de obligatoriedad y disyunción propias de las Relaciones IS-A necesarias para abordar los requisitos establecidos en el pliego técnico del problema.

Para cada uno de estos criterios se ha implementando una función en *Python* que calcula el porcentaje de acierto en cada caso y lo guarda en un fichero *JSON* que se llama "nombre_de,_la_imagen"_feedback.json.

8.2. Generación del *feedback*

Una vez creado el fichero *json*, este se le pasa al frontend de la aplicación donde utilizando diferentes componentes de *Quasar* se pintan los datos del fichero *json* en una tabla. En esta tabla aparece el número del criterio, una breve descripción del mismo y una celda coloreada de verde en caso de ser el porcentaje mayor del 60 %, en amarillo si se encuentra entre el 50 % y el 60 % y en rojo en caso de ser inferior al 50 %. La imagen 8.1 muestra un ejemplo de la página de la aplicación de generación de *feedback*. A esta página acceden solo los usuarios con rol estudiante, y lo hacen cuando consideran que han corregido todos los errores tanto visuales como de texto. Para acceder a ella, el estudiante pincha en la pestaña "generar-feedback" y al hacerlo desaparece el acceso al resto de las páginas. En la parte inferior de la página aparece un botón para que el estudiante pueda volver a la página de bienvenida y subir una nueva solución.

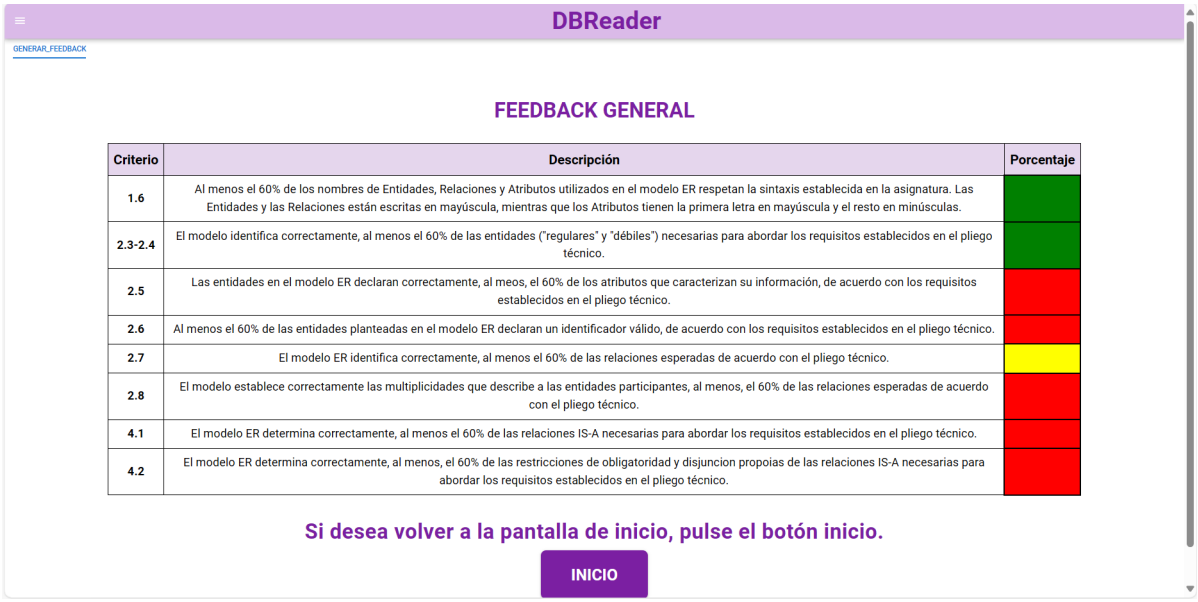


Figura 8.1: Página que muestra un ejemplo de generación de *feedback*.

Parte III

Conclusiones y trabajo futuro

Capítulo 9

Conclusiones y trabajo futuro

En este capítulo se presentan las conclusiones del proyecto y personales (sección 9.1) y se describen diferentes líneas abiertas que se podrían estudiar en un futuro para mejorar tanto la identificación de componentes, la asociación de los mismos, como la interfaz de la aplicación presentada (sección 9.2).

9.1. Conclusiones

En esta sección se exponen primero las conclusiones del proyecto, en la que se explica la utilidad de este en el futuro, se razona si se han cumplido los objetivos específicos descritos en la sección 1.2 y se hace una pequeña reflexión sobre como ha ayudado la metodología ASAP (sección 2.1) a la elaboración de este proyecto.

9.1.1. Perspectiva del proyecto

A continuación se muestran las conclusiones del proyecto:

- **Objetivos específicos:** Al inicio del proyecto se plantearon 3 objetivos, los cuales se han cumplido de forma satisfactoria. El OBJ-01 consistía en diseñar e implementar un proceso de interpretación de los contenidos de modelos ER creados por los estudiantes y profesores, a partir de una imagen digitalizada del documento manuscrito. Esto se ha conseguido usando el modelo YOLO (sección 5.2) para identificar los diferentes componentes y posteriormente se han conectado mediante las reglas descritas en la sección 5.3. El OBJ-02 era diseñar e implementar un mecanismo de corrección automatizado que fuese capaz de evaluar los modelos ER generados por los estudiantes respecto a una solución de referencia. Este objetivo se ha alcanzado mediante la construcción de funciones lógicas que comparan y evalúan la solución de los estudiantes frente a una solución previamente subida por el profesor. El OBJ-03 radicaba en diseñar e implementar un mecanismo de generación automatizada de feedback basado en la especificación de aprendizaje planteada en UVAGILE. Para lograr este objetivo, el feedback que se le proporciona al estudiante esta formado por una tabla en el que se indica una breve descripción del criterio evaluado y una celda al lado coloreada de verde si el porcentaje de acierto es superior al 60 %, de naranja si se encuentra entre el 50 % y 60 % y de rojo si es inferior al 50 %.

- **Utilidad para el futuro:** Nuestra aplicación ha sido diseñada para ser una herramienta educativa innovadora en las clases de Bases de Datos y otros cursos relacionados con el análisis y diseño de sistemas de *software*. Inicialmente, está enfocada en la enseñanza y práctica de la creación de modelos de ER, pero su utilidad se podría extender a otros diagramas utilizados en la ingeniería de software (como pueden ser: los diagramas de clases, diagramas de casos de uso, etc). En el futuro, pretendemos que esta aplicación pueda asumir el rol de evaluación automatizada de diferentes tipos de modelos, facilitando de esta manera la tarea de corrección y generación de feedback que actualmente realizan los profesores.

- **Desarrollo del proyecto/metodología de trabajo:** El uso de la metodología ASAP ha sido clave para lograr desarrollar este proyecto. Gracias a ella, se ha mantenido un ritmo constante de trabajo. Así como, ha ayudado a identificar y corregir errores conforme iban surgiendo, lo que ha sido fundamental para poder avanzar sin interrupciones significativas.

9.1.2. Perspectiva personal

El desarrollo de mi Trabajo de Fin de Grado, junto con la beca de colaboración, me ha permitido comprender como sería un proyecto de investigación. A lo largo de este proceso, me he enfrentado a diversos desafíos que me han permitido crecer tanto profesional como personalmente.

Para comenzar relataré todos aquellos aspectos más negativos que he encontrado durante el desarrollo del proyecto. Por un lado, en las primeras herramientas que utilicé para la identificación de componentes y la detección de texto daban resultados nefastos. Dicha situación me llevó a dedicar más tiempo del previsto, debido a que tuve que buscar otras herramientas que funcionasen mejor. Por otra parte, expresar un sentimiento negativo que he sentido a lo largo de estos meses, que no era otro que no visualizar el final de esta etapa. Los motivos que me llevaron a sentir esto fueron no ser capaz en los primeros meses de visualizar los resultados finales en las imágenes y por otro lado el desconocimiento en la realización de la aplicación, puesto que nunca a lo largo de mi vida académica había trabajado con ninguna de este calibre.

Sin embargo, la enseñanza positiva de estas situaciones comentadas me ha llevado a tener más paciencia y a entender que en los proyectos de investigación es muy normal que las primeras soluciones no siempre funcionen y sea necesario buscar otras alternativas. Además, su realización me ha permitido profundizar en varios campos de la informática, especialmente en la Inteligencia Artificial, la programación y el desarrollo de una aplicación web.

Para terminar con esta valoración personal cabe destacar que la realización de este proyecto me ha permitido ahondar en todos los conocimientos que he ido adquiriendo a lo largo de mi grado académico. También, a pesar de todas las adversidades descritas anteriormente mi intención por dedicar mi futuro profesional a la investigación se ha incrementado positivamente.

9.2. Trabajo futuro

En este proyecto hemos creado una aplicación que compara la solución manuscrita de un modelo ER del estudiante con la del profesor, proporcionandole una retroalimentación inmediata. Aunque hemos logrado desarrollar una herramienta que cumple con los tres objetivos descritos en la sección 1.2, aún existen varios aspectos en los que se debería seguir trabajando para conseguir una aplicación fiable que se pueda usar en las aulas. A continuación, destacamos las líneas de trabajo que quedan abiertas:

1. Entrenar el modelo de YOLO con un *dataset* más grande, para ver si de esta manera mejora la identificación de las multiplicidades y roles.
2. Mejorar las reglas de conectividad de los componentes.
3. Crear reglas para identificar las entidades padres e hijas en las relaciones IS-A.
4. Crear una API propia de reconocimiento óptico de caracteres (OCR).
5. Mejorar la interfaz de usuario de la aplicación.
6. Implementar una función en la aplicación que permita a los estudiantes acceder y visualizar el feedback detallado de los ejercicios que hayan subido previamente.
7. Implementar una función en la aplicación que permita a los profesores acceder a estadísticas detalladas sobre la corrección de los ejercicios propuestos. Estas podrían incluir el número total de estudiantes que lo han resuelto, la media de calificación de cada criterio, etc.
8. Extender la utilización de la aplicación a corrección de exámenes.
9. Ampliar el alcance de la aplicación a otros diagramas.

Bibliografía

[1] Oluwaseun Adeojo. *Desplegando la arquitectura y eficiencia de Fast y Faster R-CNN para la detección de objetos*. Accedido en Junio, 2024. Agosto 19, 2021. URL: https://docs.kanaries.net/es/topics/Python/fast-rcnn.

[2] Richard John Anthony. *Análisis de requisitos de software*. Accedido en Junio, 2024. 2016. URL: https://appmaster.io/es/blog/analisis-de-requisitos-de-software.

[3] *Aprende Machine Learning*. Accedido en Junio, 2024. Diciembre 24, 2020. URL: https://www.aprendemachinelearning.com/aprendizaje-por-refuerzo/.

[4] *Aprendizaje por refuerzo y Optimización*. Accedido en Junio, 2024. 2023. URL: https://www.iic.uam.es/inteligencia-artificial/aprendizaje-por-refuerzo/.

[5] *Arquitectura Física y Lógica*. Accedido en Junio, 2024. Octubre 2, 2012. URL: https://erlenb.blogspot.com/2012/10/arquitectura-fisica-y-logica.html.

[6] Dave Bergmann. *¿Qué es el aprendizaje semisupervisado?* Accedido en Junio, 2024. Diciembre 12, 2022. URL: https://www.ibm.com/es-es/topics/semi-supervised-learning.

[7] Margaret A Boden. *Inteligencia artificial*. Turner, 2017. ISBN: 9788416714902.

[8] *Cloud Vision API*. Accedido en Junio, 2024. 2023. URL: https://cloud.google.com/vision/?hl=es&utm_source=google&utm_medium=cpc&utm_campaign=emea-es-all-es-dr-bkws-all-all-trial-e-gcp-1707574&utm_content=text-ad-none-any-DEV_c-CRE_548685040546-ADGP_Hybrid+%7C+BKWS+-+EXA+%7C+Txt+-+AI+And+Machine+Learning+-+Vision+AI+-+v4-KWID_43700053288209441-kwd-15026601681-userloc_1005518&utm_term=KW_google%20vision-NET_g-PLAC_&&gad_source=1&gclid=CjwKCAiA8sauBhB3EiwAruTRJlt3O1T47ytBSO6xGQIXTKmoFbfE-Qx5NqRypC-BXAdTh6MNBg4RZBoCp-MQAvD_BwE&gclsrc=aw.ds.

[9] *Detecta texto en imágenes*. Accedido en Junio, 2024. URL: https://cloud.google.com/vision/docs/ocr?hl=es-419.

[10] *Empirismo*. Accedido en Junio, 2024. Mayo 17, 2024. URL: https://es.wikipedia.org/wiki/Empirismo.

[11] *Enterprise Architect (software)*. Accedido en Junio, 2024. Marzo 28, 2024. URL: https://en.wikipedia.org/wiki/Enterprise_Architect_(software).

[12] Sofia Eriksson. *Label Studio: Plataforma de etiquetado de datos de código abierto para el ajuste de modelos de IA*. Accedido en Junio, 2024. Junio 12, 2023. URL: https://vivevirtual.es/inteligencia-artificial/label-studio-plataforma-de-etiquetado-de-datos-de-codigo-abierto-para-el-ajuste-de-modelos-de-ia/.

[13] Esteban Canle Fernández. *Analizamos qué es y para qué se usa el Transfer Learning en el Deep Learning*. Accedido en Junio, 2024. Julio 5, 2022. URL: https://www.tokioschool.com/noticias/transfer-learning/.

[14] Sarah Foss, Tatiana Urazova y Ramon Lawrence. «Automatic Generation and Marking of UML Database Design Diagrams». En: *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education - Volume 1*. SIGCSE 2022. Providence, RI, USA: Association for Computing Machinery, 2022, págs. 626–632. ISBN: 9781450390705. DOI: 10.1145/3478431.3499376.

[15] Ulises García. *Introducción a las Redes Neuronales Pt. I*. Accedido en Junio, 2024. Junio 15, 2019. URL: https://futurelab.mx/redes%20neuronales/inteligencia%20artificial/2019/06/25/intro-a-redes-neuronales-pt-1/.

[16] Sandip Gautam. «Parsing Structural and Textual Information from UML Class diagrams to assist in v o assist in verification of r erification of requirement specifications ement specifications». En: *stcloudstat* (4, 2022), págs. 1-58.

[17] Glenn-jocher. *Ultralytics YOLO Guía de ajuste de hiperparámetros*. Accedido en Junio, 2024. Junio 10, 2024. URL: https://docs.ultralytics.com/es/guides/hyperparameter-tuning/#what-are-hyperparameters.

[18] Glenn-jocher, RizwanMunawar y Abirami-vina. *Profundización en las métricas de rendimiento*. Accedido en Junio, 2024. Junio 2, 2024. URL: https://docs.ultralytics.com/es/guides/yolo-performance-metrics/#class-wise-metrics.

[19] Ligdi Gonzalez. *Diferencia entre aprendizaje supervisado y no supervisado*. Accedido en Junio, 2024. Junio 15, 2018. URL: https://aprendeia.com/diferencia-entre-aprendizaje-supervisado-y-no-supervisado/.

[20] Enrique Blanco Henríquez. *Transfer Learning en modelos profundos*. Accedido en Junio, 2024. Septiembre 13, 2019. URL: https://telefonicatech.com/blog/transfer-learning-en-modelos-profundos.

[21] *Hiperparámetros para tareas de Computer Vision en el aprendizaje automático automatizado*. Accedido en Junio, 2024. Mayo 23, 2023. URL: https://learn.microsoft.com/es-es/azure/machine-learning/reference-automl-images-hyperparameters?view=azureml-api-2.

[22] *Historia de la ingeniería del software*. Accedido en Junio, 2024. Abril 15, 2024. URL: https://es.wikipedia.org/wiki/Historia_de_la_ingenier%C3%ADa_del_software#:~:text=Desde%20sus%20inicios%20en%20la,del%20rendimiento%20y%20la%20calidad..

[23] Carlos R. Jaimez-González y Jazmín Martínez-Samora. «DiagrammER: A Web Application to Support the Teaching-Learning Process of Database Courses Through the Creation of E-R Diagrams». En: *International Journal of Emerging Technologies in Learning (iJET)* 15.19 (oct. de 2020), págs. 4–21. DOI: 10.3991/ijet.v15i19.14745.

[24] Ross Girshick y Ali Farhadi Joseph Redmon Santosh Divvala. *You Only Look Once: Unified, Real-Time Object Detection*. 2015.

[25] Zoumana Keita. *Explicación de la detección de objetos YOLO*. Enero, 2024. URL: https://www.datacamp.com/es/blog/yolo-object-detection-explained.

[26] Arizbé Ken. *Requisitos no funcionales: ¿Por qué son importantes?* Accedido en Junio, 2024. Agosto 17, 2023. URL: https://www.gluo.mx/blog/requisitos-no-funcionales-por-que-son-importantes.

[27] Max Kush. *One Good Idea: The Statement Problem.* Accedido en Junio, 2024. Junio, 2015. URL: https://asq.org/quality-progress/articles/one-good-idea-the-statement-problem?id=6c8443c9748e4ddabc1badc97cbb84ae.

[28] *La Guía Fácil de los Diagramas de Despliegue UML.* Accedido en Junio, 2024. Octubre 18, 2022. URL: https://creately.com/blog/es/diagramas/tutorial-de-diagrama-de-despliegue/.

[29] *Las CNN mejoran el análisis de imágenes.* Accedido en Junio, 2024. Noviembre 16, 2021. URL: https://blog.softtek.com/es/las-cnn-mejoran-el-analisis-de-imagenes.

[30] Paola Lucena. *¿Qué son los sistemas de información y por qué son necesarios?* Accedido en Junio, 2024. Mayo 03, 2023. URL: https://www.cesuma.mx/blog/que-son-los-sistemas-de-informacion-y-por-que-son-necesarios.html#:~:text=Se%20define%20como%20un%20sistema,para%20las%20bases%20de%20datos..

[31] Perla Velasco-Elizondo Luis Castro Careaga. *Arquitectura de Software: Conceptos y Ciclo de Desarrollo.* Cengage Learning, Enero, 2015, pág. 2. ISBN: 978-607-522-456-5.

[32] Philipp Diebold y Marc Dahlem. «Agile Practices in Practice: A Mapping Study». En: *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering* (2014). DOI: 10.1145/2601248.2601254.

[33] Jesús Martínez. *Cómo Detectar Figuras Geométricas en OpenCV.* Accedido en Junio, 2024. Febrero 2021. URL: https://www.datasmarts.net/como-detectar-figuras-geometricas-en-opencv/.

[34] Miguel A. Martínez-Prieto, Jorge Silvestre, Anibal Bregon, Patricia Baz, Clara Gándara-González, Paula Mielgo e Irene Peñas. «Una metodología basada en prácticas ágiles para la realización de Trabajos Fin de Grado». En: *JENUI 8* (2023), págs. 1-8.

[35] Rizwan Munawar. *Ultralytics YOLOv8.* Accedido en Junio, 2024. 2024. URL: https://github.com/ultralytics/ultralytics.

[36] Oscar García-Olalla Olivera. *Redes Neuronales artificiales: Qué son y cómo se entrenan.* Accedido en Junio, 2024. Septiembre 16, 2019. URL: https://www.xeridia.com/blog/redes-neuronales-artificiales-que-son-y-como-se-entrenan-parte-i#:~:text=Las%20neuronas%20de%20la%20primera,como%20la%20capa%20de%20salida..

[37] *OpenCV.* Accedido en Junio, 2024. Agosto 2023. URL: https://es.wikipedia.org/wiki/OpenCV.

[38] *opencv-python 4.8.1.78.* Accedido en Junio, 2024. Junio 17, 2024. URL: https://pypi.org/project/opencv-python/.

[39] *Perceptrón multicapa.* Accedido en Junio, 2024. Septiembre 29, 2023. URL: https://es.wikipedia.org/wiki/Perceptr%C3%B3n_multicapa.

[40] Marco García Pérez. *¿Qué es el Bounding Box?* Accedido en Junio, 2024. Febrero 19, 2024. URL: https://msmk.university/ciencias/bounding-box.

[41] *Quasar Framework.* Accedido en Junio, 2024. Mayo 14, 2024. URL: https://en.wikipedia.org/wiki/Quasar_Framework.

[42] *Qué es la Inteligencia Artificial*. Accedido en Junio, 2024. Abril 19, 2023. URL: https://planderecuperacion.gob.es/noticias/que-es-inteligencia-artificial-ia-prtr#:~:text=La%20inteligencia%20artificial%20(IA)%20es,el%20razonamiento%20y%20la%20percepci%C3%B3n..

[43] *Qué son los requisitos funcionales: ejemplos, definición, guía completa*. Accedido en Junio, 2024. URL: https://visuresolutions.com/es/blog/functional-requirements/.

[44] *Redes neuronales convolucionales*. Accedido en Junio, 2024. 2021. URL: https://gamco.es/glosario/redes-neurales-convolucionales/.

[45] Jerson Rivas. *Redes Neuronales convolucionales y recurrentes*. Accedido en Junio, 2024. Julio 7, 2022. URL: https://medium.com/@jrivas_73036/redes-neuronales-convolucionales-y-recurrentes-bd0c343802b7.

[46] Daniel Eldan Rivera. *Detección de Humo de Incendios Forestales mediante Visión Artificial*. Accedido en Junio, 2024. Diciembre 30, 2022. URL: https://es.linkedin.com/pulse/detecci%C3%B3n-de-humo-incendios-forestales-mediante-daniel-eldan-rivera.

[47] Jordi Casas Roma. *Diseño conceptual de bases de datos*. Accedido en Junio, 2024. URL: https://cv.uoc.edu/annotation/cb826b689abc472d8fb5b2519840058b/699689/PID_00213704/PID_00213704.html#:~:text=Un%20esquema%20conceptual%20es%20una,restricciones%20de%20integridad%20que%20tienen..

[48] Johannes Schildgen. «MonstER Park - The Entity-Relationship-Diagram Learning Game». En: *ER Forum/Posters/Demos*. 2020, págs. 1-8.

[49] *Sets de Entrenamiento, Test y Validación*. Accedido en Junio, 2024. Marzo 03,2020. URL: https://www.aprendemachinelearning.com/sets-de-entrenamiento-test-validacion-cruzada/.

[50] Humasak Simanjuntak. «Proposed framework for automatic grading system of ER diagram». En: *2015 7th International Conference on Information Technology and Electrical Engineering (ICITEE)*. 2015, págs. 141-146. DOI: 10.1109/ICITEED.2015.7408930.

[51] *Sueldos para el puesto de Analista De Sistemas en España*. Accedido en Junio, 2024. Abril 11, 2024. URL: https://www.glassdoor.es/Sueldos/analista-de-sistemas-sueldo-SRCH_KO0,20.htm#:~:text=En%20Espa%C3%B1a%2C%20el%20sueldo%20medio,sistemas%20es%20de%20%E2%82%AC33.000%20..

[52] *Sueldos para el puesto de Arquirecto De Software en España*. Accedido en Junio, 2024. Abril 11, 2024. URL: https://www.glassdoor.es/Sueldos/arquitecto-de-software-sueldo-SRCH_KO0,22.htm.

[53] *Sueldos para el puesto de Backend Developer en España*. Accedido en Junio, 2024. Abril 11, 2024. URL: https://www.glassdoor.es/Sueldos/backend-developer-sueldo-SRCH_KO0,17.htm#:~:text=En%20Espa%C3%B1a%2C%20el%20sueldo%20medio,que%20trabajan%20de%20Backend%20developer..

[54] *Sueldos para el puesto de Documentalista en España*. Accedido en Junio, 2024. Mayo 6, 2024. URL: https://www.glassdoor.es/Sueldos/documentalista-sueldo-SRCH_KO0,14_IP4.htm.

[55] *Sueldos para el puesto de Frontend Developer en España.* Accedido en Junio, 2024. Abril 11, 2024. URL: https://www.glassdoor.es/Sueldos/frontend-developer-sueldo-SRCH_KO0,18.htm#:~:text=%C2%BFCu%C3%A1nto%20gana%20un%20Frontend%20developer,Frontend%20developer%20en%20tu%20zona..

[56] *Sueldos para el puesto de Python Developer en España.* Accedido en Junio, 2024. Abril 11, 2024. URL: https://www.glassdoor.es/Sueldos/python-developer-sueldo-SRCH_KO0,16.htm#:~:text=En%20Espa%C3%B1a%2C%20el%20sueldo%20medio,que%20trabajan%20de%20Python%20developer..

[57] *Sueldos para el puesto de Software Developer in Testing en España.* Accedido en Junio, 2024. Abril 11, 2024. URL: https://www.glassdoor.es/Sueldos/software-developer-in-testing-sueldo-SRCH_KO0,29.htm.

[58] Begg. T. Connolly C. *Database Systems: A Practical Approach to Design, Implementation, and Management.* 6.ª ed. Addison-Wesley, 2015, pág. 63.

[59] *Tesseract OCR: ¿Qué es y por qué lo deberías elegir en el 2024?* Accedido en Junio, 2024. Enero 9, 2024. URL: https://www.klippa.com/es/blog/informativo/que-es-tesseract-ocr/.

[60] *Tipos de aprendizaje que usan los algoritmos de Machine Learning.* Accedido en Junio, 2024. Noviembre 16, 2022. URL: https://decidesoluciones.es/tipos-de-aprendizaje-algoritmos-machine-learning/.

[61] Vive Unir. *¿Qué es el modelo entidad relación y para que se utiliza?* Accedido en Junio, 2024. Agosto 2023. URL: https://www.unir.net/ingenieria/revista/modelo-entidad-relacion/.

[62] Waster. *Explicación alternativa para accuracy, precision, recall y f1-score.* Accedido en Junio, 2024. 2019. URL: https://steemit.com/spanish/@waster/explicacion-alternativa-para-accuracy-precision-recall-y-f1-score.

[63] *¿QUÉ ES EPOCH EN MACHINE LEARNING?* Accedido en Junio, 2024. URL: https://ciberseguridad.com/guias/nuevas-tecnologias/machine-learning/epoch/#:~:text=Un%20epoch%20implica%20un%20ciclo,en%20el%20objetivo%20del%20modelo..

[64] *¿Qué es el aprendizaje no supervisado?* Accedido en Junio, 2024. URL: https://www.ibm.com/es-es/topics/unsupervised-learning.

[65] *¿Qué es el aprendizaje supervisado?* Accedido en Junio, 2024. URL: https://www.ibm.com/es-es/topics/supervised-learning.

[66] *¿Qué es el reconocimiento óptico de caracteres (OCR)?* Accedido en Junio, 2024. 2023. URL: https://aws.amazon.com/es/what-is/ocr/.

[67] *¿Qué es el Transfer Learning?* Accedido en Junio, 2024. Enero 6, 2024. URL: https://datascientest.com/es/que-es-el-transfer-learning.

[68] *¿Qué es el transfer learning y qué ventajas tiene?* Accedido en Junio, 2024. Noviembre 02, 2023. URL: https://www.unir.net/ingenieria/revista/transfer-learning/.

[69] *¿Qué son las redes neuronales convolucionales?* Accedido en Junio, 2024. URL: https://es.mathworks.com/discovery/convolutional-neural-network.html.

[70] Miguel Á. Martínez Prieto y Jorge Silvestre. *Proyecto/Guía docente de la asignatura Base de Datos para el curso 2023-2024.* Septiembre 07,2023.

[71] Miguel Á. Martínez Prieto, Jorge Silvestre, Aníbal Bregón Bregón y Diego García Álvarez. «Proyectos de aprendizaje (ÁGILES)». En: *Innovación docente en Educación Superior: interacción, participación y colaboración*. 2023, págs. 287-296.