Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención en Tecnologías de la Información

Creación de un servicio REST para la identificación y traducción de texto en imágenes mediante OCR

Alumno:
D. Yonatan Rodríguez Martín

Tutor:
D. Quiliano Isaac Moro Sancho

...

Agradecimientos

Agradecer a mi tutor Quiliano por apoyarme con el proyecto y ayudar a mejorarlo. Su orientación ha sido fundamental para el éxito del proyecto.

A mi familia por apoyarme durante la carrera, exámenes y entregas.

Y por último a mis amigos y compañeros de la carrera, por hacer de ésta una experiencia inolvidable, gracias a todos y todas.

En definitiva, gracias a todas las personas que de una manera más directa o indirecta han ayudado a que este proyecto de fin de grado haya salido adelante, y a la gente que ha mejorado la experiencia universitaria únicamente con su presencia.

Resumen

Este documento presenta el desarrollo de un servicio de identificación y traducción de texto en imágenes mediante OCR, haciendo posible la implementación de este servicio en aplicaciones web, móviles o de escritorio, como es el caso de la demostración simple desarrollada también para este proyecto.

Para el desarrollo se ha empleado una metodología ágil Scrum simplificada y adaptada al proyecto y se ha dividido en cuatro Sprints, cada uno con objetivos y tareas específicas.

En el primer Sprint, se ha realizado toda la fase de análisis del proyecto: planificación, objetivos, requisitos, casos de uso y modelo de dominio, sirviendo de cimientos para el resto de Sprints.

En el segundo Sprint, se ha profundizado en las tecnologías empleadas para el reconocimiento de texto y la traducción. Se han diseñado las estructuras de datos y algoritmos necesarios para cumplir los objetivos con las tecnologías seleccionadas.

El tercer Sprint se ha dedicado al procesado de las imágenes, análisis de algoritmos de "inpainting" y su implementación, así como la implementación de un servidor REST para dar servicio a la aplicación de muestra.

Finalmente, en el cuarto Sprint, se han realizado las pruebas de funcionamiento, empaquetado el servicio y desarrollado la aplicación para la demostración utilizando el servidor REST antes desarrollado.

Abstract

This paper presents the development of a service for the identification and translation of text inside images using OCR, making possible the implementation of this service in web, mobile or desktop aplications, as is the case of the simple demostrator also developed for this project.

For the development, a simplified and adapted agile Scrum methodology has been used and it has been divided into four Sprints, each one with specific objectives and tasks.

In the first Sprint, the entire project analysis phase was carried out: planning, objectives, requirements, use cases and domain model, serving as the foundation for the rest of the Sprints.

In the second Sprint, a deep study was made on the technologies used for text recognition and translation. The data structures and algorithms needed to meet the objectives with the selected technologies have been designed.

The third Sprint has been dedicated to imagen processing, analysis of "inpainting" algorithms and their implementation, as well as the implementation of a REST server to serve the sample application.

Finally, in the forth Sprint the functional tests have been performed, the service has been packaged and the demo application has been developed using the REST server previously developed.

Índice general

Lista de Figuras

Lista de Tablas

Siglas

G | J | O | R

G

GC

Garbage Collector. 22

J

JSON

JavaScript Object Notation. 38, 39

O

OCR

Optical Character Recognition. 1, 21, 35

R

REST

REpresentational State Transfer. 1, 2, 37

Capítulo 1

Introducción, objetivos y análisis de lo existente

1.1. Introducción y contexto

Con una internacionalización cada vez más presente, no es poco común encontrar instrucciones de funcionamiento, montaje o empleo en otro idioma y la falta de una traducción, por ejemplo, al Castellano. Realizar las traducciones de estos textos a otros idiomas supone un aumento de los costes tanto de diseño como de producción y recursos para las empresas, por lo que ofrecer un servicio en el que puedan ser traducidos a los idiomas necesarios supone una reducción de estos costes. La implementación de un servicio único que aúne la traducción y generación de imágenes listas para producción simplifica aún más dicho proceso.

1.2. Objetivos

El objetivo de este proyecto es la creación de un servicio basado en la tecnología REpresentational State Transfer (REST) para el consumo de imágenes y la posterior identificación, traducción y reemplazo de texto contenido en ellas, así como una aplicación de escritorio que implemente este servicio a modo de demostrador. Detalladamente serían:

- Creación de un servicio REST utilizando Spark en Java que consuma imágenes en codificación Base64.
- Identificar los cuadros de texto mediante Optical Character Recognition (OCR).
- Traducir el texto utilizando la API REST de DeepL.

- Reemplazar el texto en de la imagen original por la correspondiente traducción, detectando la familia, tamaño, color y estilo de la fuente empleada.
- Creación de interfaces y puntos de implementación, habilitando futuras implementaciones del servicio utilizando diferentes tecnologías de reconocimiento de texto o traducción.
- Desarrollo de una aplicación básica de escritorio para mostrar el funcionamiento del servicio paso a paso.

En cuanto a los objetivos académicos y de aprendizaje:

- Creación de un back-end con tecnología REST sobre el framework Spark, muy utilizado hoy en día.
- Estudio de técnicas de reconocimiento de texto digitales y visión por computador.
- Algoritmos de representación y colisión espacial bidimensional.
- Manipulación de imágenes y representación de texto utilizando la principal librería utilizada OpenCV.
- Llevar a cabo la documentación y desarrollo de un proyecto software desde cero.

1.3. Análisis de lo existente

Existen numerosos servicios de traducción en línea de texto en imágenes, pero cada uno de ellos ofrece características concretas: Google Translate [1] se centra en la velocidad de la traducción, y menos en la fidelidad de los resultados; mientras que DeepL Translator [2] proporciona traducciones mucho más correctas, pero en un tiempo mayor.

1.3.1. Google Lens y Google Translate

Google Lens [3] es una tecnología desarrollada por Google que permite identificar texto en tiempo real utilizando la cámara del dispositivo en caso de tenerla, imágenes tanto locales como en línea, documentos y páginas web. Una vez identificado el texto, presenta la opción de traducir el texto a múltiples idiomas utilizando la tecnología de Google Translate [1] y reemplazarlo en la imagen original, en tiempo real. Ver figura 1.1.

Puntos fuertes:

- Multiplataforma: dispone de aplicación web y móvil (Android e iOS).
- Amplio catálogo de 133 idiomas reconocidos y para traducir.

- Traducción en tiempo real.
- Integración con el resto de servicios de Google como Google Docs o Google Slides.

Puntos débiles:

- El reconocimiento de texto se realiza de forma local, teniendo un pobre rendimiento en equipos de menor desempeño.
- Menor fiabilidad de las traducciones, sólo algunas están verificadas.
- Requiere de los Servicios de Google instalados en Android, no presentes en todos los dispositivos.

Figura 1.1: Resultado de la traducción de muestra Google Translate

1.3.2. DeepL Translate

DeepL Translate [2] utiliza inteligencia artificial basada en aprendizaje automático para obtener traducciones de alta calidad y fiables. Se basa en el modelo Transformer [4] desarrollado por la empresa matriz de DeepL Translate: DeepL, Linguee [5], que utiliza el aprendizaje automático para entrenar un modelo de traducción mucho mejor al de otros competidores. Permite traducir texto y documentos PDF, Word y PowerPoint a 31 idiomas diferentes. Ver figura 1.2.

Puntos fuertes:

- Multiplataforma: cuenta con aplicaciones tanto web como móviles (Android e iOS).
- Alta precisión de la traducción. DeepL Translate es considerado uno de los servicios de traducción más precisos y fiables.
- Detección automática del idioma.
- Velocidad de la traducción: al utilizar un modelo con un número mejor de parámetros, los resultados se obtienen de manera más rápida y eficiente.

Puntos débiles:

- Menor número de idiomas detectados y disponibles para traducir, únicamente 31 idiomas.
- Posibilidad de traducir únicamente documentos de texto, no imágenes.
- La precisión de la traducción disminuye para idiomas menos comunes.
- Detección y traducción lenta y limitada para usuarios gratuitos de archivos de texto.

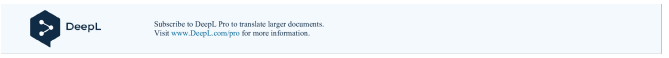
Se puede observar en la figura 1.2 el resultado de la traducción de la imagen de prueba convertida previamente a PDF.

1.3.3. Bing Microsoft Translator

Bing Microsoft Translator [6] es la alternativa desarrollada por Microsoft basada en la tecnología de traducción automática neuronal ("*Neural Machine Translation*" en inglés) entrenada con billones de traducciones para mejorar las respuestas del servicio. Permite traducir texto, documentos (en su versión para empresas integrada en los productos de Microsoft, como Office o Teams) e imágenes (únicamente utilizando su aplicación móvil). Ver figura 1.3.

Puntos fuertes:

- Multiplataforma: dispone de una aplicación web y móvil (Android e iOS).



In a place in La Mancha, whose name I don't want to remember, not long ago there lived a nobleman with a lance in a shipyard, an old-fashioned adarga, a skinny rocín and a running greyhound. A pot of something more cow than ram, salpicón on most nights, duels and quebrantos on Saturdays, lentils on Fridays, some extra pigeon on Sundays, consumed the three parts of his estate. The rest of the time he wore a velvet sackcloth, fleece tights for the feasts with his slippers of the same, and on weekdays he honored himself with his finest vellori. He had in his house a mistress who was over forty, and a niece who was not even twenty, and a farm hand who saddled the donkey as well as took the pruning shears. The age of our nobleman was around fifty years old, he was of a stout complexion, dry of flesh, wiry of face; he was a great early riser and a friend of hunting. They want to say that he had the nickname of Quijada or Quesada (in this there is some difference in the authors who write about this case), although by plausible conjectures it is understood that his name is Quijanai, but this matters little to our story; it is enough that in the narration 1 does not leave one point of the truth. It is to be known, then, that this aforesaid nobleman, the times he was idle (which were most of the year) was given to reading books of chivalry with such fondness and pleasure, that he almost completely forgot the exercise of hunting, and even the management of his estate; And so great was his curiosity and inattention in this that he sold many acres of arable land to buy books of chivalry to read; and so he took home as many of them as he could find; and none of them seemed to him as good as those composed by the famous Feliciano de Silva: because the clarity of his prose, and those intricate reasons of his, seemed to him pearls; and more when he came to read those requiebros and letters of challenge, where in many parts he found written: the reason of the unreason that to my reason is made, in such a way my reason enflaquece, that with reason I complain of your fermosura, and also when he read: the high heavens that of your divinity divinely with the stars are fortified, and make you deserving of the merit that deserves the

Figura 1.2: Resultado de la traducción de muestra DeepL Translate

- Integración en los productos Enterprise de Microsoft.
- Gran cantidad de idiomas reconocidos y disponibles para la traducción: 128 idiomas.
- Velocidad de traducción rápida gracias al gran poder computacional de los servidores de Microsoft.

Puntos débiles:

- Necesidad de tener una licencia de pago empresarial para sacar el máximo provecho del servicio.
- Traducción de imágenes disponible únicamente en las aplicaciones móviles.
- No reemplaza el texto original en la imagen tras la traducción.

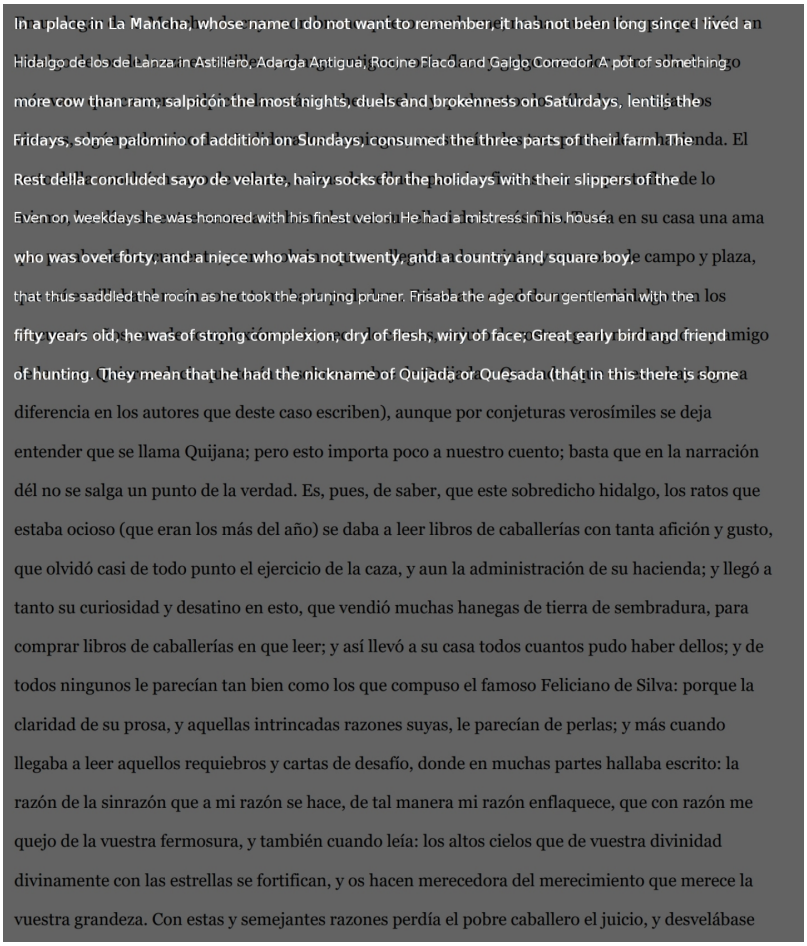


Figura 1.3: Resultado de la traducción de muestra Bing Microsoft Translator

1.3.4. Yandex Translate

Yandex Translate [7] se posiciona como la alternativa desarrollada en Rusia de los servicios de traducción. Al igual que Bing Microsoft Translator, se basa en la traducción automática neuronal, pero al utilizar un modelo menor y menos entrenado, la calidad de las traducciones y la fiabilidad se ven notablemente afectadas. Permite la traducción de texto, sitios web, documentos e imágenes. Ver figura 1.4.

Puntos fuertes:

- Multiplataforma: dispone de aplicación web y móvil (Android e iOS).
- Cuenta con 100 idiomas reconocidos y disponibles para la traducción.
- Integración con el resto de servicios de Yandex.
- Velocidad de la traducción a cambio de una menor precisión.

Puntos débiles:

- Baja calidad de las traducciones con respecto a otros competidores.
- Problemas de privacidad: la empresa Yandex ha sido investigada por la privacidad y seguridad de los datos de sus usuarios.
- Centrado en el idioma ruso, con menor fiabilidad para el resto de idiomas.

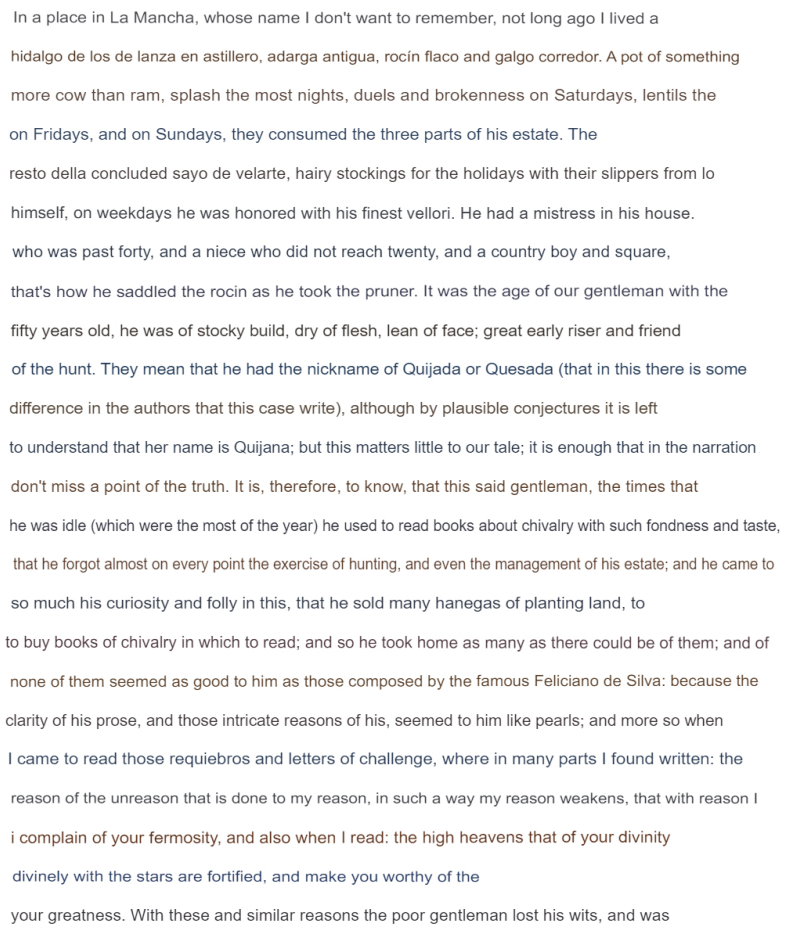


Figura 1.4: Resultado de la traducción de muestra Yandex Translate

1.4. Estructura de la memoria

El contenido de esta memoria se estructura en diferentes capítulos. Los capítulos 3 a 6, contienen el desarrollo del proyecto dividido en cuatro Sprints, de acuerdo a la metodología ágil Scrum empleada en el desarrollo.

Capítulo 1 Introducción, objetivos y análisis de lo existente

Capítulo 2 Planificación y requisitos

Capítulo 3 Sprint 1: análisis del proyecto: se presenta la fase de análisis completa del proyecto: planificación, objetivos, requisitos, casos de uso y modelo de dominio.

Capítulo 4 Sprint 2: reconocimiento y traducción de texto: centrado en el reconocimiento de texto, y su traducción, así como estructuras de datos y algoritmos empleados.

Capítulo 5 Sprint 3: procesado de imágenes y servidor REST: dedicado al procesado de las imágenes y algoritmos de eliminación de texto en imágenes y la implementación de un servidor REST.

Capítulo 6 Sprint 4: pruebas y aplicación de escritorio: pruebas de funcionamiento al sistema e implementación y pruebas de la aplicación de escritorio de muestra.

Capítulo 7 Conclusiones

Anexo A Manuales

Capítulo 2

Planificación y requisitos

2.1. Introducción

En el desarrollo de este proyecto participan el alumno Yonatan Rodríguez Martín y el profesor Quiliano Isaac Moro Sancho, y se llevará a cabo mediante una metodología ágil basada en Scrum.

La metodología Scrum se utiliza habitualmente para equipos de desarrollo de más de dos integrantes, por lo que se simplificará y adaptará al contexto de este proyecto con los roles detallados en el apartado 2.2.1 Roles.

2.2. Metodología

Como se ha mencionando anteriormente, para el desarrollo de este proyecto de fin de grado se utilizará una metodología ágil [8] Scrum . Ésta metodología se basa el el principio ágil de la creación de ciclos breves para el desarrollo, en este caso *Sprints*, que consisten en periodos de una a cuatro semanas para los cuales se realizan reuniones de planificación del desarrollo del proyecto en las que se divide en tareas que se añaden a la lista de tareas o *Backlog* [9]. Estas tareas son estimadas en cuanto al esfuerzo que va a suponer realizarlas utilizando el sistema de puntos de función IFPUG [10]. Tras la finalización de cada *Sprint* se valora la calidad de las tareas finalizadas y las no finalizadas pasan a formar parte del siguiente.

Adicionalmente, se realizan reuniones diarias o *daily meetings* en las que se planifican las tareas a llevar a cabo en el día de desarrollo. La no realización de estas reuniones será una de las adaptaciones realizadas a la metodología Scrum original, puesto que no son necesarias de acuerdo a los roles establecidos y también adaptados que se definen en el siguiente apartado; en su lugar, se realizarán reuniones semanales.

2.2.1. Roles

Los roles principales del proceso de Scrum son el de *Product Owner*, *Scrum Master* y equipo de desarrollo, que se detallan en la tabla 2.1 Roles en el proyecto. Fuera del proceso de Scrum existen roles igualmente necesarios como el de usuario, y los *Stakeholders* formado por el grupo de personas interesadas en que el producto tenga un beneficio. Ambos serán desempeñado en conjunto por el profesor y el alumno.

Rol	**Descripción y responsabilidades**
Product Owner	Responsable de definir los requisitos del proyecto, así como establecer objetivos y una visión global del proyecto y priorizar las funcionalidades a implementar. Proporciona orientación durante el desarrollo para asegurar el cumplimiento de los requisitos establecidos. Este rol será desempeñado por el profesor Quiliano Isaac Moro Sancho.
Scrum Master	Se encarga de organizar las reuniones necesarias para cada *Sprint* y mantener el *Backlog* actualizado junto al *Product Owner*, así como realizar un seguimiento del progreso del desarrollo y el mantenimiento del método Scrum. Será el alumno Yonatan Rodríguez Martín quién desempeñe este rol.
Equipo de desarrollo	Responsable de desarrollar el software necesario, crear la interfaz de usuario y llevar a cabo las pruebas necesarias para asegurar el funcionamiento del trabajo. Además se encarga de documentar el proceso y los resultados. Este rol será desempeñado por el alumno.

Tabla 2.1: Roles en el proyecto

2.3. Planificación

La planificación de este proyecto se realizará en Sprints, de acuerdo a la metodología Scrum mencionada anteriormente. El desarrollo se divide en cuatro Sprints, cada uno con sus correspondientes fases de análisis, diseño, desarrollo y creación las pruebas. En cada uno se incluye la documentación para la redacción de la memoria del proyecto.

La duración de cada Sprint será de tres a cuatro semanas. Ver figura 2.2.

Sprint	Descripción	Duración estimada
Sprint 1	Dedicado al análisis del proyecto.	Tres semanas.
Sprint 2	Análisis e implementación del reconocimiento y traducción de texto.	Cuatro semanas.
Sprint 3	Estudio de procesado de imágenes y servidor REST.	Cuatro semanas.
Sprint 4	Realización de pruebas unitarias, empaquetado final y desarrollo de aplicación de escritorio.	Cuatro semanas.

Tabla 2.2: Planificación en Sprints

2.4. Análisis de riesgos

En este apartado se analizarán los riesgos a los que se puede enfrentar el desarrollo del proyecto, así como la probabilidad de que ocurran, el impacto que tendrían y las estrategia y acciones a seguir en caso de que se presente cada uno de ellos.

RSK00	**Falta de experiencia**
Descripción	El alumno tiene experiencia en Java pero no en las tecnologías utilizadas como Tesseract u OpenCV
Probabilidad e impacto	Alta — Bajo
Estrategia	Mitigación
Acciones de mitigación	Estudiar la documentación
Acciones correctivas	Posibilidad de utilizar otras tecnologías

Tabla 2.3: Riesgo 00 Falta de experiencia

RSK01	**Retraso en la entrega**
Descripción	Posible demora en la entrega del proyecto debido a mala planificación o falta de recursos
Probabilidad e impacto	Media — Alto
Estrategia	Mitigación
Acciones de mitigación	Plan de trabajo detallado y realista, asignando recursos adecuados
Acciones correctivas	Revisar y ajustar la planificación y recursos

Tabla 2.4: Riesgo 01 Retraso en la entrega

RSK02	**Cambios en los requisitos**
Descripción	Posibilidad de que los requisitos del proyecto sean modificados durante el desarrollo
Probabilidad e impacto	Alta — Medio
Estrategia	Mitigación
Acciones de mitigación	Mantener comunicación constante con el tutor para cerrar los requisitos de manera rápida y precisa
Acciones correctivas	Evaluar el impacto de los cambios

Tabla 2.5: Riesgo 02 Cambios en los requisitos

RSK03	**Cambios en la tecnología**
Descripción	Posibilidad de que las tecnologías o herramientas utilizadas en el proyecto sufran cambios o actualizaciones, como una nueva versión de Java con cambios que inutilicen parte de la funcionalidad
Probabilidad e impacto	Baja — Medio
Estrategia	Mitigación
Acciones de mitigación	Utilizar versiones estables. Mantener las funcionalidades en módulos auto-contenidos
Acciones correctivas	Evaluar el impacto de los cambios, realizar pruebas de compatibilidad y adaptar el proyecto

Tabla 2.6: Riesgo 03 Cambios en la tecnología

RSK04	**Falta de disponibilidad del alumno o tutor**
Descripción	La falta de disponibilidad por enfermedad u otros motivos del alumno o tutor para el desarrollo puede afectar a los tiempos de entrega
Probabilidad e impacto	Alta — Bajo
Estrategia	Mitigación
Acciones de mitigación	Establecer una linea de comunicación alumno-profesor para asegurar la disponibilidad mutua
Acciones correctivas	Reasignar tareas, ajustar la planificación

Tabla 2.7: Riesgo 04 Falta de disponibilidad del alumno o tutor

RSK05	**Incompatibilidad de versiones de librerías**
Descripción	Posibilidad de que las librerías utilizadas en el proyecto sean incompatibles entre sí
Probabilidad e impacto	Baja — Alto
Estrategia	Mitigación
Acciones de mitigación	Validar compatibilidad de dependencias
Acciones correctivas	Ajustar o reemplazar los componentes con incompatibilidades en caso de ser posible

Tabla 2.8: Riesgo 05 Incompatibilidad de componentes

RSK06	**Limitaciones de traducción con DeepL API**
Descripción	Posibilidad de que la licencia gratuita de DeepL API tenga limitaciones en la cantidad de traducciones realizadas
Probabilidad e impacto	Baja — Bajo
Estrategia	Mitigación
Acciones de mitigación	Evaluar el volumen de traducciones requeridas y explorar alternativas o versiones de pago si es necesario
Acciones correctivas	Optimizar el uso de las traducciones disponibles, buscar soluciones alternativas de traducción o considerar la adquisición de una licencia de pago

Tabla 2.9: Riesgo 06 Limitaciones de traducción con DeepL API

RSK07	**Cambios en la API de DeepL**
Descripción	Posibilidad de que la API de DeepL sufra cambios en su estructura, métodos o funcionalidades durante el desarrollo del proyecto
Probabilidad e impacto	Baja — Medio
Estrategia	Mitigación
Acciones de mitigación	Mantenerse actualizado con las actualizaciones y cambios en la API de DeepL
Acciones correctivas	Analizar los cambios en la API de DeepL y ajustar la implementación en consecuencia

Tabla 2.10: Riesgo 07 Cambios en la API de DeepL

RSK08	**Limitaciones de procesamiento con Tesseract OCR**
Descripción	Posibilidad de que Tesseract OCR presente limitaciones en el procesamiento de ciertos tipos de imágenes lo que puede afectar la precisión de la extracción de texto
Probabilidad e impacto	Media — Medio
Estrategia	Mitigación
Acciones de mitigación	Realizar pruebas de rendimiento con diferentes tipos de imágenes, utilizar técnicas de preprocesamiento de imágenes para mejorar la calidad de entrada
Acciones correctivas	Identificar y abordar los problemas de procesamiento, explorar alternativas de OCR y ajustar las expectativas

Tabla 2.11: Riesgo 08 Limitaciones de procesamiento con Tesseract OCR

RSK09	**Mala planificación del proyecto**
Descripción	Posibilidad de una planificación inadecuada del proyecto, lo que puede llevar a retrasos en las entregas, incumplimiento de hitos y falta de coordinación entre las tareas
Probabilidad e impacto	Media — Alto
Estrategia	Mitigación
Acciones de mitigación	Realizar una planificación detallada, teniendo en cuenta los requisitos y restricciones del proyecto, asignar adecuadamente los recursos y establecer mecanismos de seguimiento y control del progreso
Acciones correctivas	Revisar y ajustar la planificación en función de las desviaciones identificadas y replantear tareas y asignación de recursos

Tabla 2.12: Riesgo 09 Mala planificación del proyecto

RSK10	**Estimación inadecuada de costes**
Descripción	Posibilidad de una estimación incorrecta de los costes del proyecto, lo que puede resultar en un agotamiento de recursos
Probabilidad e impacto	Alta — Medio
Estrategia	Mitigación
Acciones de mitigación	Realizar un análisis exhaustivo de los costes involucrados, incluyendo materiales, recursos humanos y otros gastos asociados, considerando posibles imprevistos y contingencias
Acciones correctivas	Identificar y gestionar los desvíos de costes, buscar formas de reducir los gastos sin comprometer la calidad del proyecto, y comunicar y negociar los cambios en el presupuesto con los interesados pertinentes.

Tabla 2.13: Riesgo 10 Estimación inadecuada de costes

RSK11	**Falta de experiencia estimando riesgos**
Descripción	Posibilidad de que la falta de experiencia del alumno en la estimación de riesgos pueda llevar a la omisión de riesgos importantes o a una valoración inadecuada de su probabilidad e impacto
Probabilidad e impacto	Alta — Bajo
Estrategia	Mitigación
Acciones de mitigación	Buscar información o asesoramiento en la gestión de riesgos y estudiar proyectos similares
Acciones correctivas	Utilizar la experiencia adquirida en la gestión de riesgos durante el proyecto y documentar el proceso para futuros proyectos

Tabla 2.14: Riesgo 11 Falta de experiencia estimando riesgos

2.5. Estimación de costes

La correcta estimación de los costes de un proyecto es fundamental para garantizar la viabilidad económica del proyecto y permitir una asignación correcta de los recursos. Esta estimación, se debe realizar en base a la experiencia de proyectos previos de alcance similar, pero debido a la falta de dichos proyectos y el acceso a sus costes, la estimación se realizará en base a costes conocidos y sin tener en cuenta posibles imprevistos que puedan surgir.

2.5.1. Costes hardware y software

Para el desarrollo del proyecto ha utilizado el ordenador del alumno. En el momento de despliegue, se requiere del servicio Amazon AWS EC2 [11]. Los costes hardware quedan detallados en la tabla 2.15.

En cuanto al software, se han empleado aplicaciones de código abierto y gratuitas como pueden ser Overleaf [12] o paint.net [13]. En la tabla 2.16 se detallan los costes de los programas y aplicaciones con licencias o costes que se han empleado, calculados para la duración del proyecto que es de tres meses [14][15][16].

Dispositivo	**Coste**
Ordenador	1.700,00€ aproximado
Amazon AWS EC2	41,85€ / mes

Tabla 2.15: Costes hardware

Aplicación/programa	**Coste**
Microsoft Office Project Profesional 2021	154,50€
IntelliJ IDEA Ultimate	169,00€
DeepL API Pro	14,97€ + variable por uso

Tabla 2.16: Costes software

2.5.2. Costes humanos

Otra parte importante a tener en cuenta a la hora de planificar los costes del proyecto, son los costes humanos, es decir, el tiempo invertido por el desarrollador -en este caso el alumno- en codificar el proyecto. Al tratarse de un proyecto desarrollado en Java, se han calculado estos costes en base al salario medio en España de un programador Java [17]. Este salario medio es de 16,15€ la hora, calculado para 300 horas que tiene de duración el proyecto, el coste de desarrollo es de 4.845€.

Capítulo 3

Sprint 1: análisis del proyecto

En este capítulo se detalla la fase de análisis del primer Sprint, en el que se incluye el análisis de los requisitos del proyecto (funcionales y no funcionales), los casos de uso que se pueden dar utilizando el proyecto y el modelo de dominio.

3.1. Planificación del Sprint

Las tareas asignadas para el primer Sprint se centran en realizar el análisis completo del proyecto. Su duración total estimada es de tres semanas, que se evaluará en el apartado 3.6. Finalización del Sprint. Ver tabla 3.1.

Tarea	**Descripción**	**Duración estimada**
1. Análisis de requisitos funcionales	Identificar y documentar los requisitos funcionales del proyecto.	3 días
2. Análisis de requisitos no funcionales	Identificar y documentar los requisitos no funcionales del proyecto.	3 días
3. Análisis de casos de uso	Identificar y documentar los casos de uso del sistema.	4 días
4. Modelo de dominio	Crear un modelo de dominio que represente las entidades y relaciones del sistema.	5 días

Tabla 3.1: Planificación Sprint 1

3.2. Análisis de requisitos funcionales

El análisis de requisitos se ha realizado de acuerdo a las necesidades del usuario y del *Product Owner*, en base a los objetivos establecidos para el proyecto y los conocimientos obtenidos estudiando ingeniería del software.

Los requisitos funcionales son las características y funciones que un sistema software debe tener para satisfacer las necesidades de los usuarios: definen lo que el sistema debe hacer, pero no cómo hacerlo.

En la tabla 3.2 Requisitos funcionales se pueden ver los requisitos funcionales identificados para el proyecto.

3.3. Análisis de requisitos no funcionales

Por otro lado, los requisitos no funcionales definen las restricciones y criterios que afectan a la calidad y rendimiento del sistema en términos de seguridad, usabilidad o disponibilidad entre otros. Este análisis se ha realizado teniendo en cuenta que el sistema debe ser rápido y fiable. Ver tabla 3.3 Requisitos no funcionales.

La rapidez del sistema se refiere a la capacidad para responder de manera ágil a las solicitudes de procesamiento, minimizando los tiempos de espera del usuario. Por otro lado la fiabilidad hace referencia al rendimiento constante y predecible del sistema, siendo capaz de gestionar situaciones inesperadas o de error.

3.4. Análisis de casos de uso

En este apartado se realizará un análisis de los casos de uso para los diferentes escenarios en los que el usuario puede verse al utilizar la aplicación. El objetivo principal del análisis de casos de uso es obtener una visión clara y comprensiva de las necesidades y expectativas del usuario, así como establecer el alcance del sistema.

Estos casos de uso representan las interacciones de un actor (en este caso, el usuario) con el sistema. Ver figura 3.1 Diagrama casos de uso.

Req	Requisito	Descripción
RF01	Subir imagen	El sistema debe permitir al usuario subir una imagen para el reconocimiento de texto.
RF02	Reconocer texto	El sistema debe ser capaz de, utilizando la librería Tess4J, realizar el OCR en la imagen proporcionada.
RF03	Traducir texto	Una vez reconocido el texto, el sistema debe utilizar la API REST de DeepL para traducirlo a un idioma seleccionado por el usuario.
RF04	Reemplazar texto en imagen	El sistema debe ser capaz de reemplazar el texto reconocido en la imagen original y generar una nueva imagen con el texto traducido utilizando la librería OpenCV.
RF05	Soporte de idiomas	El sistema debe ser capaz de reconocer y traducir texto en varios idiomas.
RF06	Proporcionar transcripción	El sistema debe proporcionar una transcripción del texto reconocido y traducido de la imagen.
RF07	Documentación de la API	El sistema debe proporcionar una documentación clara y completa de la API REST: *endpoints*, parámetros esperados y respuestas correspondientes.
RF08	Documentación de usuario	El sistema debe proporcionar una documentación detallada para los usuarios.
RF09	Gestión de errores	El sistema debe manejar adecuadamente los errores que puedan ocurrir durante el proceso de OCR, traducción o reemplazo en la imagen.
RF10	Configuración de parámetros	El sistema debe proporcionar la capacidad de configurar parámetros adicionales para el proceso de reemplazo de texto, como la fuente a emplear, el color de la fuente o la exclusión de los dígitos en la traducción.
RF11	Procesamiento de varias imágenes	El sistema debe ser capaz de procesar varias imágenes sin intervención manual en cada imagen.
RF12	Soporte de formatos de imagen	El sistema debe ser capaz de procesar los formatos de imagen comunes como JPEG o PNG.
RF13	Mejora de la precisión de OCR	EL sistema debe implementar técnicas o procedimientos para mejorar la calidad de las imágenes de entrada para mejorar la precisión del reconocimiento de texto, minimizando los errores y mejorando la calidad de los resultados.
RF14	Detección automática del idioma	El sistema debe ser capaz de detectar automáticamente el idioma del texto de la imagen proporcionada, eliminando la necesidad de que el usuario lo especifique manualmente.
RF15	Precisión de la traducción	El sistema debe proporcionar traducciones precisas y acorde al contexto del texto, traduciendo frases o párrafos enteros.

Tabla 3.2: Requisitos funcionales

Req	Descripción
RNF01	El sistema debe tener tiempos de respuesta menores de 10 segundos para el reconocimiento de texto, traducción y reemplazo en la imagen.
RNF02	El sistema debe estar disponible y accesible el 99 % del tiempo.
RNF03	El sistema debe ser capaz de manejar un volumen de usuarios moderado sin degradar el rendimiento.
RNF04	El código del sistema debe estar bien estructurado, modularizado y documentado para facilitar el mantenimiento.
RNF05	El sistema debe utilizar eficientemente los recursos del sistema, liberándolos cuando no sean necesarios como utilizar un Garbage Collector (GC) más agresivo [18][19].
RNF06	El sistema se debe poder utilizar como base de implementación de otros sistemas.
RNF07	El sistema debe ser compatible con los principales navegadores Chrome, Edge, Opera, Firefox así como *Sockets* [20] en cualquier lenguaje de programación que los implemente.

Tabla 3.3: Requisitos no funcionales

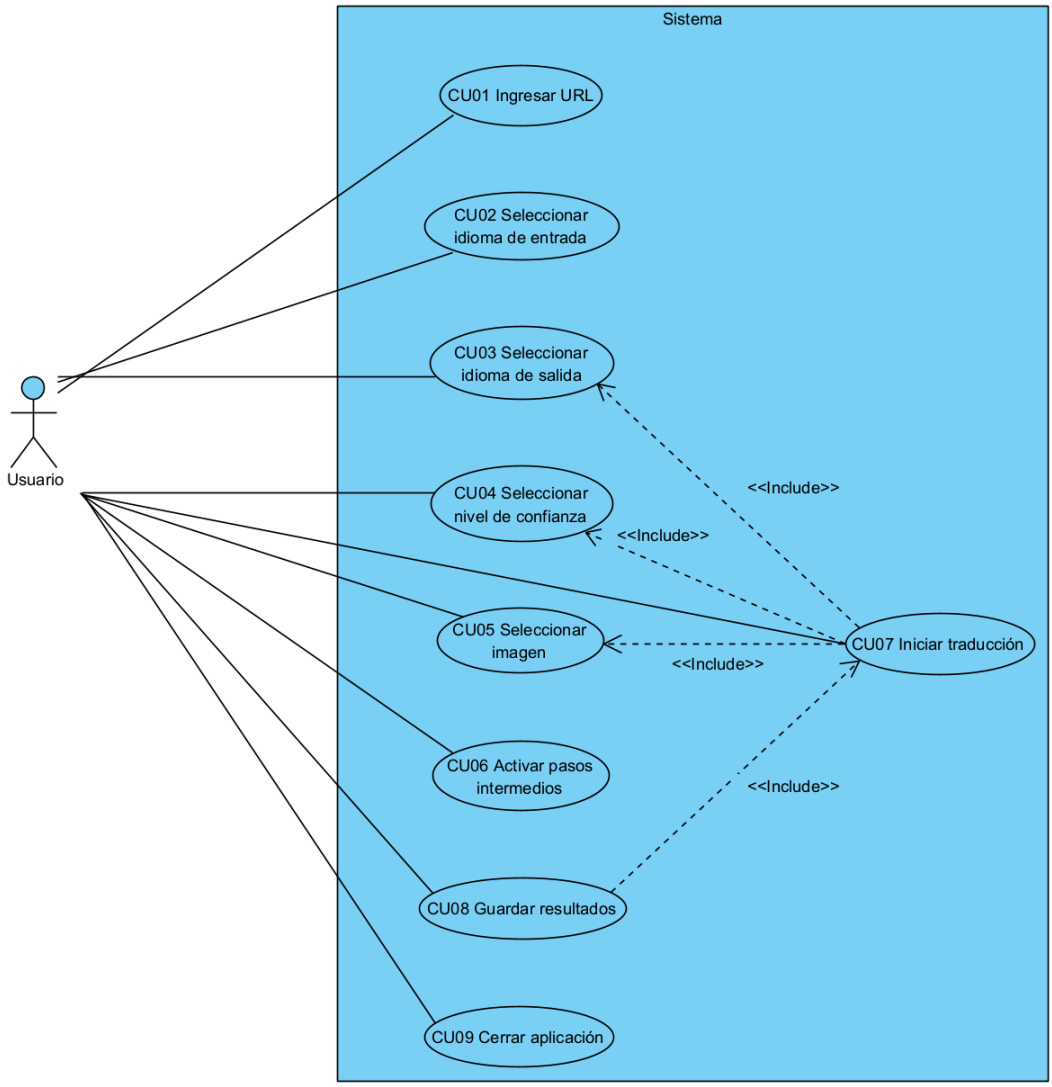


Figura 3.1: Diagrama casos de uso

3.4.1. Casos de uso detallados

Caso de uso	**CU01 - Ingresar URL**
Descripción	El usuario desea ingresar la URL del servidor REST en el campo de texto. La URL tiene un valor por defecto, por lo que no es necesario modificarla.
Actor	Usuario.
Pre-condición	-
Post-condición	La URL del servidor ha sido registrada por el sistema.
Secuencia base	1. El usuario ingresa la URL del servidor REST. 2. El sistema valida la URL.
Secuencias alternativas	1a. El usuario deja la URL por defecto y el caso de uso termina.

Tabla 3.4: Caso de uso 01 - Ingresar URL

Caso de uso	**CU02 - Seleccionar idioma de entrada**
Descripción	El usuario desea seleccionar el idioma de entrada a través de un selector desplegable. Si no se modifica, por defecto se auto-detecta el idioma.
Actor	Usuario.
Pre-condición	-
Post-condición	El idioma de entrada ha sido registrado por el sistema.
Secuencia base	1. El usuario selecciona el idioma de entrada. 2. El sistema registra el idioma de entrada seleccionado.
Secuencias alternativas	1a. El usuario no modifica el idioma de entrada, manteniéndolo en automático y el caso de uso termina.

Tabla 3.5: Caso de uso 02 - Seleccionar idioma de entrada

Caso de uso	**CU03 - Seleccionar idioma de salida**
Descripción	El usuario desea seleccionar el idioma de salida a través de un selector desplegable.
Actor	Usuario.
Pre-condición	-
Post-condición	El idioma de salida ha sido registrado por el sistema.
Secuencia base	1. El usuario selecciona el idioma de salida. 2. El sistema registra el idioma de salida seleccionado.
Secuencias alternativas	-

Tabla 3.6: Caso de uso 03 - Seleccionar idioma de salida

Caso de uso	**CU04 - Seleccionar nivel de confianza**
Descripción	El usuario desea seleccionar el nivel de confianza del reconocimiento de texto a través de un selector numérico. Si no se modifica, se utiliza el valor por defecto.
Actor	Usuario.
Pre-condición	-
Post-condición	El nivel de confianza ha sido registrado por el sistema.
Secuencia base	1. El usuario selecciona el nivel de confianza . 2. El sistema registra el nivel de confianza seleccionado.
Secuencias alternativas	1a. El usuario no modifica el valor por defecto del nivel de confianza y el caso de uso termina.

Tabla 3.7: Caso de uso 04 - Seleccionar nivel de confianza

Caso de uso	**CU05 - Seleccionar imagen**
Descripción	El usuario desea seleccionar una imagen a traducir haciendo click en el botón “Seleccionar Imagen” y utilizar un selector de archivos.
Actor	Usuario.
Pre-condición	-
Post-condición	La imagen seleccionada ha sido registrada por el sistema.
Secuencia base	1. El usuario hace click en el botón “Seleccionar Imagen”. 2. El sistema muestra el selector de archivos. 3. El usuario selecciona una imagen. 4. El sistema registra la imagen seleccionada.
Secuencias alternativas	3a. El usuario no selecciona ninguna imagen o cierra el selector de archivos y el caso de uso termina.

Tabla 3.8: Caso de uso 05 - Seleccionar imagen

Caso de uso	**CU06 - Activar pasos intermedios**
Descripción	El usuario desea activar la generación de los pasos intermedios que atraviesa la imagen marcando una caja de selección. Por defecto la caja de selección se encuentra desactivada.
Actor	Usuario.
Pre-condición	-
Post-condición	El sistema ha activado la generación de pasos intermedios.
Secuencia base	1. El usuario marca la casilla “Activar pasos intermedios”. 2. El sistema activa la generación de pasos intermedios.
Secuencias alternativas	1a. El usuario no marca la casilla y el caso de uso termina.

Tabla 3.9: Caso de uso 06 - Activar pasos intermedios

Caso de uso	**CU07 - Traducir**
Descripción	El usuario desea traducir la imagen seleccionada haciendo click en el botón "Traducir" para enviar la solicitud al servidor REST con los parámetros seleccionados previamente. Este botón solo se activa cuando se ha seleccionado el idioma de salida y una imagen.
Actor	Usuario.
Pre-condición	El idioma de salida y una imagen han sido seleccionados.
Post-condición	El sistema ha enviado la solicitud al servidor REST y se ha recibido la respuesta.
Secuencia base	1. El usuario hace click en el botón "Traducir". 2. El sistema envía la solicitud al servidor REST con los parámetros seleccionados.
Secuencias alternativas	-

Tabla 3.10: Caso de uso 07 - Traducir

Caso de uso	**CU08 - Guardar resultados**
Descripción	Tras recibir la respuesta del servidor REST, el usuario desea guardar los resultados de la traducción haciendo click en el botón "Guardar", seleccionando el directorio de salida mediante un selector de archivos. Este botón solo se activa cuando se ha recibido la respuesta a la petición tras el CU07 - Traducir.
Actor	Usuario.
Pre-condición	El CU07 - Traducir se ha ejecutado previamente.
Post-condición	El sistema ha guardado los archivos de salida en el directorio seleccionado.
Secuencia base	1. El usuario hace click en el botón "Guardar". 2. El sistema muestra un selector de archivos. 3. El usuario selecciona el directorio de destino. 4. El sistema guarda los archivos de salida en el directorio seleccionado.
Secuencias alternativas	3a. El usuario no selecciona ningún directorio o cierra el selector de archivos y el caso de uso termina. 4a. La opción Activar pasos intermedios se encuentra seleccionada y el sistema incluye los archivos con los pasos intermedios junto a los archivos de salida en el directorio seleccionado.

Tabla 3.11: Caso de uso 08 - Guardar resultados

Caso de uso	**CU09 - Terminar aplicación**
Descripción	En cualquier momento el usuario desea terminar el flujo de la aplicación y salir de ella.
Actor	Usuario.
Pre-condición	-
Post-condición	La aplicación se cierra.
Secuencia base	1. El usuario decide terminar la aplicación. 2. El sistema cierra la aplicación.
Secuencias alternativas	-

Tabla 3.12: Caso de uso 09 - Terminar aplicación

3.5. Modelo de dominio

Una vez definidos los requisitos del sistema y los casos de uso, los componentes del sistema se pueden agrupar en tres categorías según la función que desempeñan:

- **Reconocimiento y agrupado del texto:** encargado de ejecutar el reconocimiento de texto mediante OCR y agrupar las palabras reconocidas en frases completas.
- **Traducción del texto:** encargado de traducir el texto reconocido.
- **Procesado de imagen:** encargado del pre-procesado y post-procesado de las imágenes, así como el reemplazo del texto en la imagen original.

Esta separación funcional y las dependencias entre componentes se puede apreciar en diagrama de clases del dominio del sistema, que representa las clases que se van a definir en el sistema, sus atributos, funciones y relaciones entre ellas. Ver figura 3.2.

Cada componente funcional se puede ver en detalle desde la figura 3.3 hasta la figura 3.10, y en el siguiente apartado se describe la funcionalidad de cada componente del sistema.

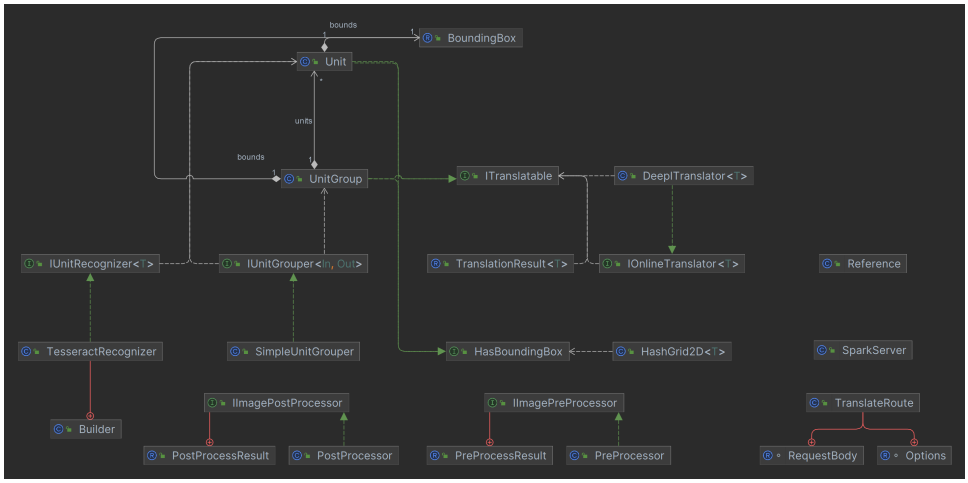


Figura 3.2: Modelo de dominio

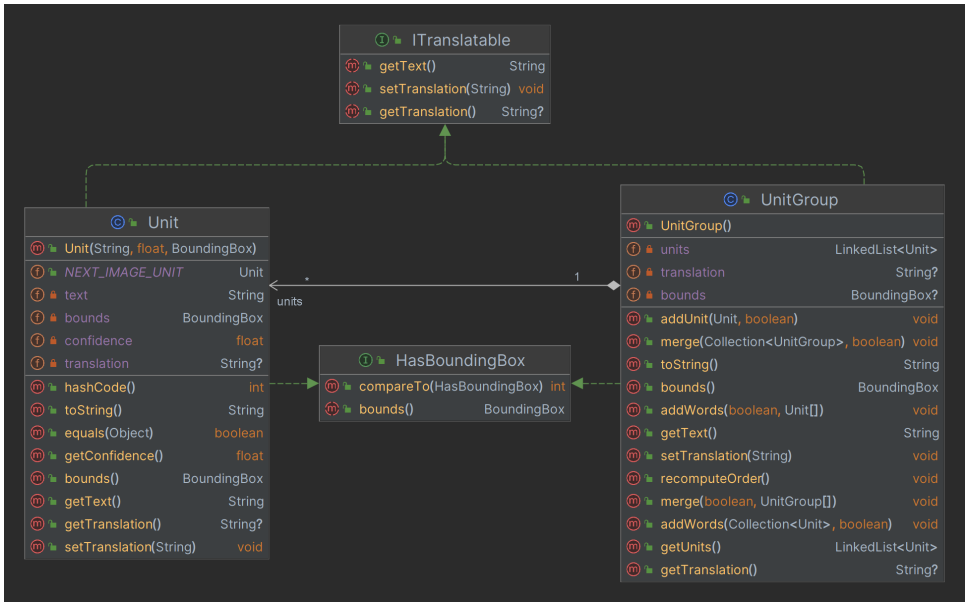


Figura 3.3: Modelo detallado - Modelo de datos

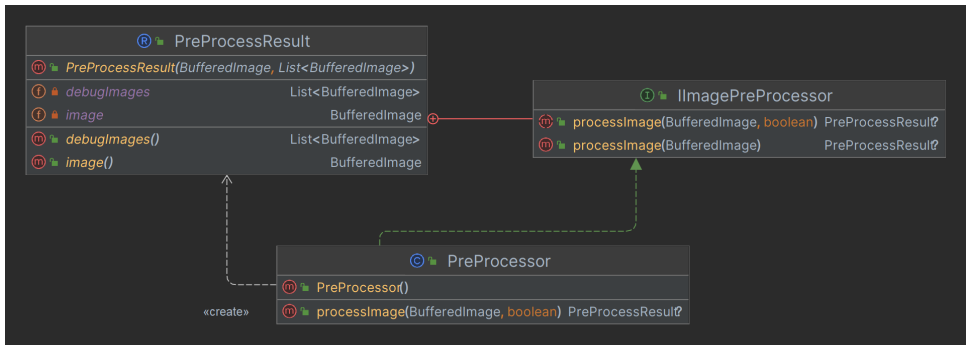


Figura 3.4: Modelo detallado - Pre-procesado de imagen

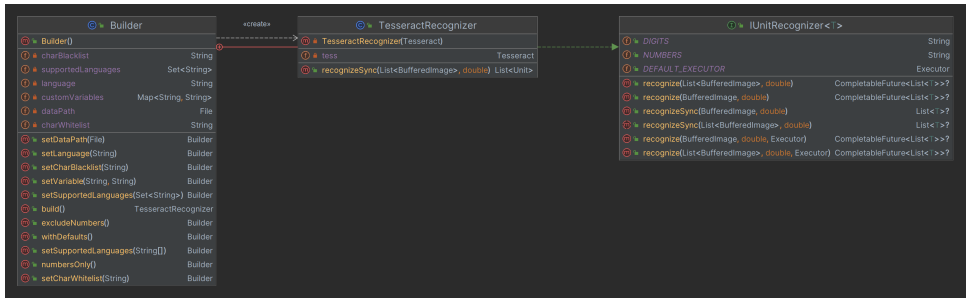


Figura 3.5: Modelo detallado - Reconocimiento de texto

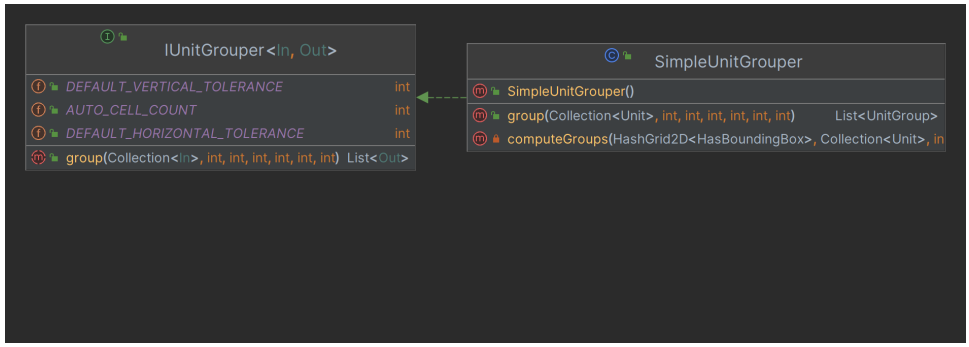


Figura 3.6: Modelo detallado - Agrupación de palabras

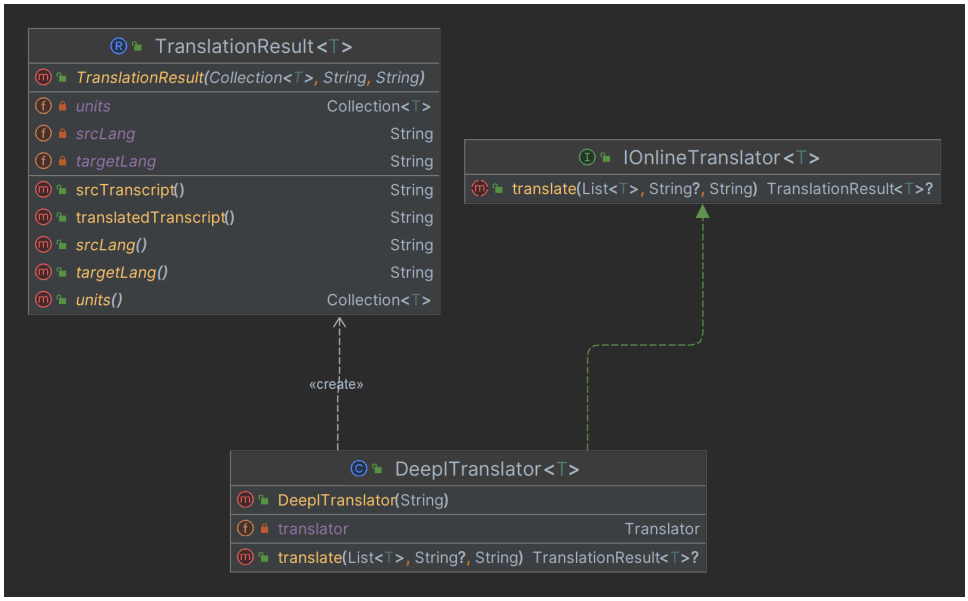


Figura 3.7: Modelo detallado - Traducción de texto

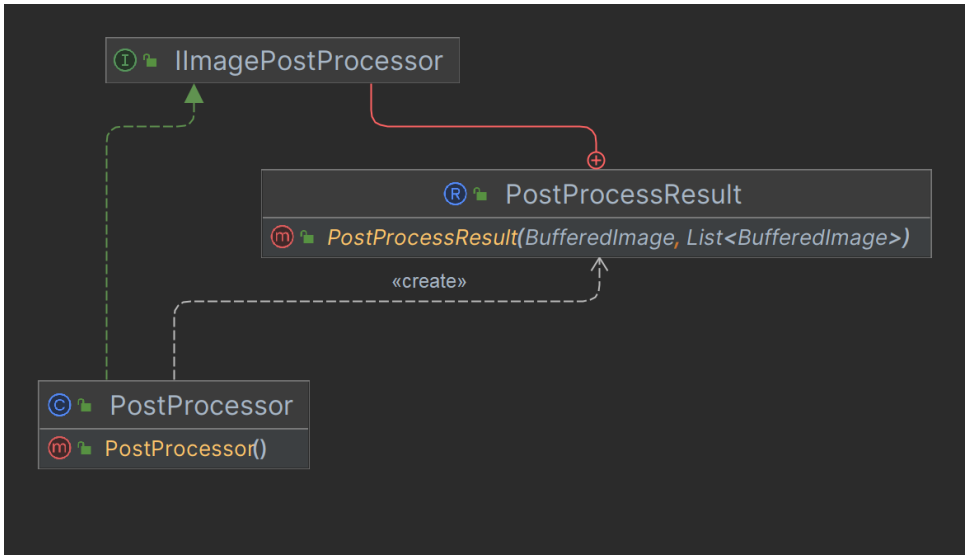


Figura 3.8: Modelo detallado - Post-procesado de imagen

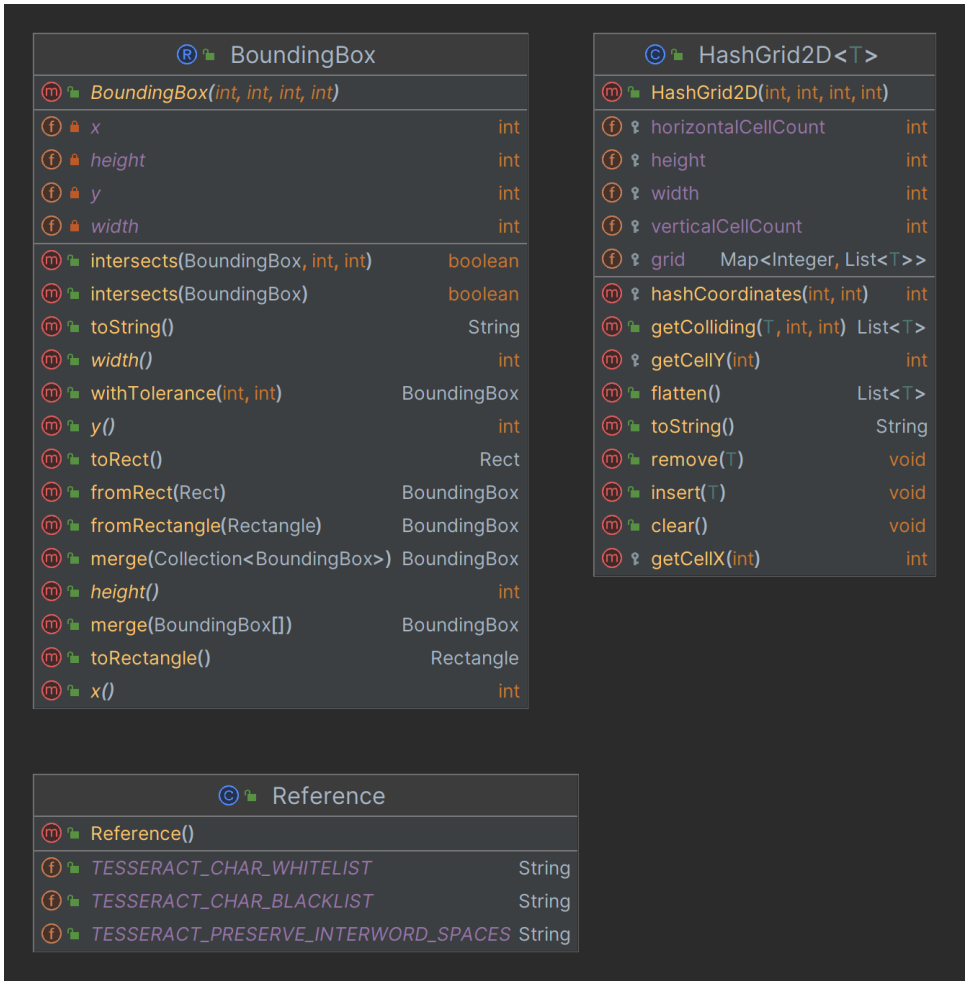


Figura 3.9: Modelo detallado - Utilidades

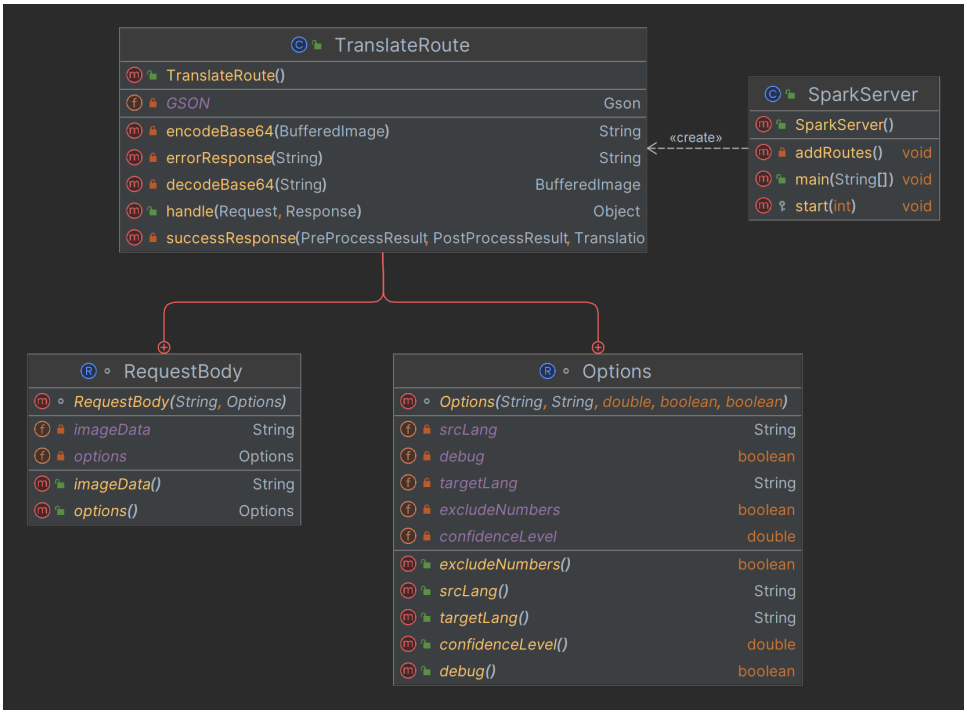


Figura 3.10: Modelo detallado - Servidor REST

3.5.1. Modelo de dominio detallado

Modelos de datos

Para el modelado de los datos se ha empleado el concepto de **Unit** y **UnitGroup**. Una Unit representa el mínimo componente reconocido, en este caso una palabra, y un UnitGroup está compuesto de una serie de Units relacionadas entre sí, es decir, una frase. Ver figura 3.3.

Ambos elementos implementan las interfaces **HasBoundingBox** y **ITranslatable**, que proporcionan la representación espacial 2D y la traducción de cada elemento respectivamente.

Procesado de imagen

Se encargan de realizar el pre-procesado y post-procesado de las imágenes las clases **PreProcessor** y **PostProcessor**, que implementan las interfaces **IImagePreProcessor** y **IImagePostProcessor** respectivamente. Esta implementación de interfaces permite que, mediante herencia, se creen implementaciones nuevas con funcionalidades más avanzadas o tecnologías diferentes en un futuro para el procesado de las imágenes. Las tecnologías utilizadas para implementar el procesado de la imagen se ven en detalle en el capítulo 4. Sprint 2: reconocimiento y traducción de texto. Ver figuras 3.4 y 3.8.

Reconocimiento de texto

El elemento principal de este componente es la interfaz **IUnitRecognizer**, que se encarga de abstraer el proceso de reconocimiento de texto en un sólo método (síncrono o asíncrono), permitiendo implementar cualquier tecnología de reconocimiento para ser utilizada en el sistema.

La librería de OCR elegida es Tesseract OCR, que se verá más en detalle en el siguiente capítulo. La implementación de esta librería se encuentra en la clase **TesseractRecognizer**. El proceso de inicialización del reconocedor de texto es complejo: requiere numerosos parámetros para poder iniciarse correctamente, y es ampliamente configurable, por lo que se ha empleado el **patrón de diseño constructor** [21]. Este patrón se emplea para construir de manera estructurada un objeto complejo a partir de otro siguiendo una serie de pasos.

Agrupado de texto

Para agrupar unidades independientes (también referidas cómo palabras) en unidades agrupadas (también referidas cómo frases) se ha empleado una estructura de datos basada en el trabajo de Lefebvre y Hoppe, Perfect Spatial Hashing [22] implementado en la clase **HashGrid2D** que se desarrolla en detalle en el siguiente capítulo.

La clase encargada de agrupar las palabras en frases es **SimpleUnitGrouper**, utilizando la estructura de datos antes mencionada y basada en la interfaz **IUnitGrouper**. Esta clase puede ser reemplazada por otro algoritmo de unión espacial gracias al uso de la herencia. Ver figuras 3.6 y 3.9.

Traducción de texto

Una vez agrupado el texto, llega a la fase de traducción, en la que un **IOnlineTranslator** se encarga de detectar el idioma de entrada - en caso de no haber sido seleccionado por el usuario previamente - y traducir el texto agrupado. La clase que implementa esta interfaz es **DeepLTranslator** y detalla en el capítulo 5. Ver figura 3.7.

Servidor REST

Finalmente, para unir todos los componentes en un único flujo y permitir utilizar el servicio como un **servidor REST**, la clase **SparkServer** implementa un sencillo servidor REST utilizando la biblioteca Spark, que se ve en profundidad en el capítulo 6.

Esta clase se encarga de recibir una petición POST HTML con la imagen a traducir y los parámetros de la traducción (representados en la clase **Options**). Una vez recibida la petición, comienza el flujo con el procesado de la imagen inicial, seguido del reconocimiento y agrupado del texto, tras ello se traduce y finalmente se sustituye el texto en la imagen original, en la fase final de procesado de la imagen, dando lugar a la respuesta del servidor.

3.6. Finalización del Sprint

De acuerdo a la metodología SCRUM, al finalizar el Sprint se llevan a cabo reuniones de revisión y retrospectivas, en el que el equipo de desarrollo (en este caso el alumno) presenta el trabajo realizado durante el Sprint, revisando las tareas completadas [9]. Ver tabla 3.13

Al final del primer Sprint se han completado todas las tareas que se habían asignado, por lo que ninguna tarea pasará al siguiente Sprint.

Tarea	**Estado**	**Desviación**
1. Análisis de requisitos funcionales	Completada	0 días
2. Análisis de requisitos no funcionales	Completada	0 días
3. Análisis de casos de uso	Completada	0 días
4. Modelo de dominio	Completada	0 días

Tabla 3.13: Revisión Sprint 1

Capítulo 4

Sprint 2: reconocimiento y traducción de texto

En este capítulo se detalla el trabajo realizado en el segundo Sprint: tras el análisis del anterior Sprint, se presentan las decisiones de diseño que se han tomado para el proyecto, así como las tecnologías a utilizar para las implementaciones de reconocimiento y traducción de texto, los problemas que han surgido y las estructuras de datos empleadas y algoritmos diseñados para resolverlos.

4.1. Planificación del Sprint

Las tareas incluidas en este segundo Sprint están centradas en el diseño del sistema de detección, agrupado y traducción del texto. La duración estiamda del Sprint es de cuatro semanas. Ver tabla 4.1.

4.2. Diseño del sistema

El sistema constará de dos componentes: un servidor REST, y una aplicación de escritorio a la que dará servicio mediante peticiones HTTP, siguiendo una **arquitectura cliente-servidor**.

El lenguaje de programación elegido para la implementación del servidor y cliente es **Java 19**, utilizando Maven [23] como herramienta de gestión de proyectos, dependencias y empaquetado de la aplicación.

A la hora de diseñar el sistema, se creará una capa de abstracción con interfaces para permitir la implementación de diferentes tecnologías de, por ejemplo, reconocimiento de

Tarea	Descripción	Duración estimada
1. Comparar librerías OCR	Analizar y comparar las diferentes librerías que existen para el reconocimiento de caracteres y determinar la mejor opción.	1 día
2. Comparar servicios de traducción	Analizar y comparar los diferentes servicios de traducción en línea que existen y determinar la mejor opción.	1 días
3. Implementar reconocimiento de texto	Implementar la librería elegida para el reconocimiento de texto.	5 días
4. Estructuras de datos	Crear las estructuras de datos necesarios para almacenar el texto reconocido.	3 días
5. Algoritmo agrupado de texto	Implementar un algoritmo para agrupar el texto: palabras en frases	4 días
6. Traducir texto	Implementar la traducción del texto utilizando el servicio elegido.	3 días
7. Diseño del sistema	Tomar las decisiones de diseño del sistema: arquitectura a utilizar, patrones, librerías y *frameworks*.	1 día

Tabla 4.1: Planificación Sprint 2

texto, sin modificar el comportamiento general del sistema, es decir: las implementaciones de los componentes y sus métodos están separados, convirtiendo cada componentes en software de caja negra para el sistema y el resto de componentes [24].

Para el reconocimiento de texto se utilizará la librería desarrollada por Google Tesseract OCR [25], en su empaquetado para Java Tess4J. Se reconocerán palabras individuales en lugar de párrafos completos, facilitando la separación en líneas individuales para el posterior reemplazo en la imagen final, sin afectar a la calidad de traducción.

La traducción se realizará mediante un servicio en linea (en lugar de uno local), este servicio es el de DeepL en su versión REST [26], al cual se enviará el texto a traducir mediante una petición HTTP utilizando el método POST, y se recuperará la traducción y el idioma detectado mediante una respuesta en formato JavaScript Object Notation (JSON).

La manipulación de imágenes se implementará utilizando la librería OpenCV [27], de código abierto; para la eliminación del texto original en la imagen se empleará un algoritmo Navier-Stokes (NS) [28].

Se utilizará Spark [29] como *framework* para el servidor REST en su versión Java. Se creará un punto de acceso /translate en el cual se realizará toda la comunicación cliente-servidor.

La aplicación de escritorio cliente seguirá un **patrón de diseño MVC** basado en el framework de interfaz nativo de Java: Swing [30].

4.2.1. Arquitectura cliente-servidor

El proyecto se centra en el desarrollo del servicio de reconocimiento y traducción, por lo que el grueso de la funcionalidad se encuentra en el servidor que será el encargado de ejecutar la lógica del sistema (reconocimiento, traducción y reemplazo del texto) de los datos que el cliente envíe mediante peticiones POST HTTP en formato JSON, mientras que la aplicación cliente se encargará de enviar al servidor las imágenes en Base64 junto a las opciones del usuario para la traducción. Ver figura 4.1.

La comunicación entre el cliente y el servidor se realizará mediante peticiones POST HTTP cuyo contenido será estructurado en formato JSON, como se ha mencionado anteriormente, y se detalla en los siguientes apartados.

Por otro lado, el cliente sigue el patrón de diseño modelo-vista-controlador (MVC) [31], y se detalla en el capítulo 6. Sprint 4: pruebas y aplicación de escritorio .

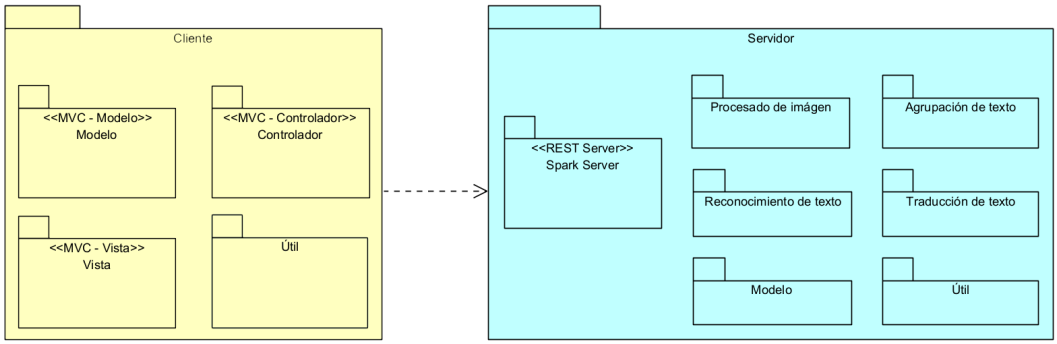


Figura 4.1: Arquitectura cliente-servidor

Petición del cliente

El contenido de la petición del cliente se divide en dos partes: la imagen a traducir, y las opciones de la traducción. Ver figura 4.2.

La imagen se almacena en formato Base64, que representa el contenido binario de la imagen en formato ASCII, es decir, caracteres que pueden ser enviados en una petición HTTP a través de internet.

Dentro de las opciones de traducción se encuentra el idioma del texto de la imagen de origen (opcional, si no aparece se detectará automáticamente), el idioma al que se desea traducir la imagen, el nivel de confianza del reconocimiento de texto y opcionalmente se puede activar la opción “debug” para que se incluyan las imágenes de los pasos intermedios que atraviesa la imagen.

Respuesta del servidor

La respuesta del servidor contiene la imagen final traducida en formato Base64, una transcripción del texto original de la imagen y de la traducción realizada, el idioma de origen (ya sea detectado o proporcionado por el usuario) y de destino, y un *flag* que indica si la petición se ha procesado correctamente. Opcionalmente, una lista con las imágenes de los pasos intermedios y un mensaje que se puede incluir cuando la petición es errónea. Ver figura 4.3.

```
{
    "imageData": "",
    "options": {
        "srcLang": "",
        "targetLang": "",
        "confidenceLevel": 0.0 - 1.0,
        "debug": true | false
    }
}
```

Figura 4.2: Contenido petición cliente

```
{
    "imageData": "",
    "srcTranscript": "",
    "translatedTranscript": "",
    "srcLang": "",
    "targetLang": "",
    "debugImages": [],
    "message": "",
    "success": true | false
}
```

Figura 4.3: Contenido respuesta servidor

4.3. Implementación

En este apartado, se presenta la implementación de las funcionalidades de detección, agrupado, y traducción del texto, así como las tecnologías y herramientas utilizadas.

4.3.1. Herramientas utilizadas

En el desarrollo tanto del servidor como del cliente se ha utilizado IntelliJ IDEA Ultimate 2023.1.2, para la detección del texto se han empleado los modelos entrenados por la comunidad de Tessdata para Tesseract OCR [32].

4.3.2. Reconocimiento de texto - Tesseract OCR

Existen muchas librerías y alternativas para el reconocimiento de texto en un computador, como servicios en la nube o implementaciones locales. Por un lado, los servicios en la nube tienen un coste económico elevado y no suponen ninguna mejora real para el usuario, puesto que el reconocimiento se realizará fuera de su dispositivo, por lo que se ha optado por ejecutar el reconocimiento de manera local.

Las principales alternativas valoradas para el reconocimiento de texto son:

- **Tesseract OCR**: de código abierto, precisión de reconocimiento moderada y de uso libre, con soporte multi-idioma.
- **Abbyy FineReader SDK**: mayor precisión de reconocimiento, software propietario y de pago.
- **GOCR**: menor precisión y falta de reconocimiento en múltiples idiomas, únicamente en inglés y alemán.
- **KerasOCR**: la necesidad de entrenar un modelo específico para el sistema hace que sea una solución inviable.

Tras analizar las principales librerías de OCR, se utilizará Tesseract OCR, puesto que es de código abierto, gratuita, tiene soporte para múltiples idiomas y una alta precisión de reconocimiento. Se trata de una librería altamente flexible que, aunque no tiene la mayor precisión del mercado, representa un punto medio entre versatilidad y facilidad de uso.

Tesseract OCR

Tesseract OCR es una tecnología desarrollada entre 1984 y 1994 por HP Labs [33] y mantenida actualmente por Google bajo una licencia de código abierto para realizar el reconocimiento de texto en imágenes [33]. El proceso de OCR que realiza, consta de varias fases: una fase inicial encargada de determinar las líneas en las que se agrupa el texto y determinar la línea base mediante un spline cuadrático; tras ello, se analiza cada línea de texto y se calcula la separación de sus caracteres individuales, que se recortan para finalmente, mediante un método de clasificación por características, ser identificados.

Para utilizar la librería en el lenguaje de programación Java, se ha empleado Tess4j como adaptador que, utilizando Java Native Interface (JNI) [19], permite llamar a las funciones en lenguaje C++ de la librería.

Se ha empleado el **patrón de diseño constructor** [21] para simplificar y facilitar la implementación de la librería en el sistema. Mediante este patrón, se pueden construir e inicializar los objetos requeridos por Tesseract paso a paso, mediante una serie de métodos para el control de variables, parámetros de iniciación y llamada al constructor.

Tras la ejecución del reconocimiento de texto, se encapsula el texto junto a sus coordenadas en un objeto **Unit** (ver figura 3.3. Modelo detallado - Modelo de datos).

4.3.3. Agrupación del texto

Tesseract OCR dispone de múltiples modos de reconocimiento de texto: símbolos individuales, palabras, línea a línea, que agrupa palabras en líneas horizontales, párrafos completos o en bloque, por el cual todo el texto del documento se trata como un único componente. Para hacer más eficiente la eliminación del texto en la imagen original, se ha decidido emplear el reconocimiento de palabras individuales, por lo que, para garantizar una traducción precisa y fiable, se deben reagrupar en frases completas. Esto también ha supuesto una mejora en la traducción de textos con disposición de artículo (varias columnas de texto), puesto que el modo de líneas completas por defecto de la librería, trata las líneas de todas las columnas como una sola, mezclando el texto.

Se ha utilizado la abstracción de unidad y grupo de unidades para representar palabras y grupos de ellas, puesto que no todas las unidades son palabras, pueden ser números o símbolos.

Estructuras de datos y algoritmos

El primer paso tras el reconocimiento de texto, es almacenar las unidades en una estructura de datos 2D para su posterior manipulación. Esta estructura de datos, llamada **HashGrid2D**, se inspira en el trabajo de Lefebvre y Hoppe, "Perfect Spatial Hashing" [22], dividiendo el espacio en celdas y asignando una celda a cada punto en el espacio; cada celda tiene un identificador único, generado mediante una función *hash* [34] con las coordenadas que representa. De esta forma, se pueden situar las unidades de forma espacial a la vez que acelera el proceso de formación de grupos de unidades. Ver figura 4.4.

Una vez almacenadas las unidades, se procede a la búsqueda de unidades que se encuentren al mismo nivel vertical, y lo suficientemente cerca horizontalmente para unirse en grupos de unidades. Al utilizar una estructura de datos especializada, se acelera el proceso de formación de grupos; esto se debe a que al situarse las unidades en celdas contiguas, en lugar de comprobar colisiones horizontales con todas las unidades del espacio, únicamente es necesario comprobarlas con las unidades de la misma celda o de las celdas inmediatamente adyacentes.

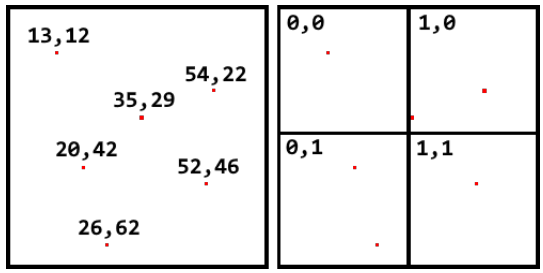


Figura 4.4: (A) Representación espacial de puntos. (B) Cuadrícula espacial.

Tanto la estructura de datos empleada para almacenar las unidades, como el algoritmo para detectar las colisiones se han elegido para maximizar la eficiencia del sistema, pero no son los únicos que se pueden implementar. De igual manera, se pueden implementar nuevas estructuras y algoritmos en el sistema gracias al énfasis en la creación de interfaces que abstraen el funcionamiento interno de cada componente.

4.3.4. Traducción del texto - DeepL API

Utilizando como base el análisis realizado de los diferentes servicios de traducción en el capítulo 1 Introducción, objetivos y análisis de lo existente, se ha utilizado la API REST de DeepL en su implementación Java [35], puesto que cuenta con un plan gratuito, su tiempo de respuesta es algo mayor que el de sus competidores, pero gracias a su modelo de transformador las traducciones son más fiables [4].

Tras la fase previa de agrupado, se ordenan los grupos de unidades por posición vertical en orden descendente - puesto que los párrafos se presentan de esta forma habitualmente - y se envía su contenido junto al idioma de entrada y salida a la API REST de DeepL. Recibida la respuesta, se almacenan las traducciones en sus correspondientes grupos de unidades, así como el idioma de entrada reconocido (en caso de que no haya sido proporcionado por el usuario).

Una vez traducido el texto, la transcripción del texto original y traducido son generadas para su posterior uso en la respuesta al cliente.

4.4. Finalización del Sprint

Una vez finalizado el Sprint, se ha llevado a cabo la reunión de revisión, en la que se ha podido apreciar que las tareas 4. Estructuras de datos y 5. Algoritmo agrupado de texto se han desviado de su estimación original, pero no ha tenido repercusión puesto que las tareas 1, 2 y 6 se han realizado en menor tiempo del esperado. Ver tabla 4.2.

Tarea	**Estado**	**Desviación**
1. Comparar librerías OCR	Completada	-0,5 días
2. Comparar servicios de traducción	Completada	-0,5 días
3. Implementar reconocimiento de texto	Completada	0 días
4. Estructuras de datos	Completada	+1 día
5. Algoritmo agrupado de texto	Completada	+1 día
6. Traducir texto	Completada	-1 día
7. Diseño del sistema	Completada	0 días

Tabla 4.2: Revisión Sprint 2

Capítulo 5

Sprint 3: procesado de imágenes y servidor REST

Las tareas asignadas para el tercer Sprint se centran en la mejora de las imágenes de entrada para aumentar la precisión de la detección de texto, y el reemplazo del texto en la imagen original por el texto traducido; también se incluye la implementación del servidor REST. Al igual que en el Sprint anterior, se presentan las tecnologías utilizadas y las implementaciones realizadas.

5.1. Planificación del Sprint

Las tareas asignadas a este Sprint se centran en el pre y post-procesado de imágenes, y la implementación del servidor REST. Ver tabla 5.1.

5.2. Mejoras de imagen para OCR

Como se ha visto en el capítulo anterior, Tesseract OCR no necesita letras perfectamente definidas para poder reconocerlas, únicamente que las características principales estén bien definidas [33]. Es por esto que el primer paso aplicado a la imagen ha sido la conversión de la imagen a escala de grises, reduciendo de ruido provocado por diferentes colores, dificultando el reconocimiento de caracteres individuales; tras ello, para mejorar la visibilidad de algunas tipografías demasiado finas o con características difusas, se lleva a cabo la binarización de la imagen [36].

La conversión a **escala de grises** se realiza eliminando la información de color de la

Tarea	Descripción	Duración estimada
Mejoras imagen original	Analizar e implementar técnicas de manipulación de imágenes para mejorar la imagen de entrada, aumentando la precisión del reconocimiento de texto.	2 días
Eliminar texto original	Eliminar el texto de la imagen original para poder reemplazarlo con el texto traducido.	5 días
Insertar texto traducido	Insertar el texto traducido en la imagen procesada (sin el texto) de modo que se ajuste al tamaño del original.	3 días
Servidor REST	Realizar la implementación del servidor REST que implemente todas las funcionalidades del sistema.	4 días

Tabla 5.1: Planificación Sprint 3

imagen original (figura 5.1), asignando a cada píxel su brillo (representado por la intensidad de rojo, verde y azul). Ver figura 5.2.

Con la imagen compuesta únicamente por el brillo de cada píxel, se lleva a cabo la **binarización**: se reducen los colores de la imagen al blanco puro y negro puro. Esto se hace estableciendo un límite a partir del cual se considerará como negro, y por debajo del cual se considerará blanco. La imagen resultante ha visto su ruido altamente reducido y, aunque las letras no tienen una claridad perfecta, sí presentan las características necesarias para poder ser reconocidas. Ver figura 5.3.

Éstas manipulaciones de imágenes se han realizado utilizando **Leptonica** [37], una librería de la que depende Tesseract OCR.

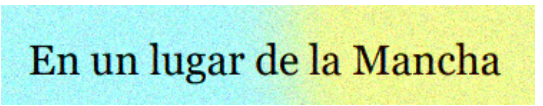


Figura 5.1: Imagen original

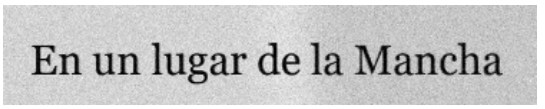


Figura 5.2: Imagen en escala de grises

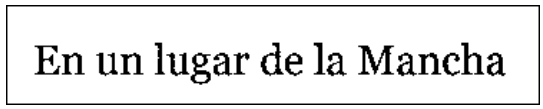


Figura 5.3: Imagen tras binarización

5.3. Eliminación de texto en imágenes

Un objetivo importante del sistema, es reemplazar el texto en la imagen original por el traducido sin que ésta sufra grandes modificaciones o sin tener un gran recuadro blanco tras el texto. Es por esto que se ha recurrido a técnicas de manipulación de imágenes para eliminar el texto de forma natural.

5.3.1. Leptonica

La primera implementación realizada ha sido utilizando la librería ya incorporada en Tesseract OCR, Leptonica [37]. Ésta implementación, más tarde descartada, consiste en la determinación de los colores principales presentes en cada segmento de la imagen mediante un histograma, para así poder establecer el color del fondo y el color del texto. Realizadas las pruebas de la implementación, se ha visto que para imágenes de texto sobre un color sólido producía buenos resultados, pero la baja calidad en imágenes con múltiples colores y la poca eficiencia al calcular un histograma para cada segmento de texto, provocando grandes tiempos de ejecución, han supuesto el abandono de éste método para la eliminación del texto, perdiendo tres días de desarrollo, pudiendo provocar la aparición del riesgo "RSK01 Retrasos en la entrega", debido a la materialización del riesgo "RSK03 Cambios en la tecnología", ver apartado 2.4. Análisis de riesgos.

5.3.2. OpenCV y técnicas de inpainting

Tras descartar la primera implementación, se ha realizado una nueva implementación utilizando la librería OpenCV, una de las más utilizadas para la manipulación de imágenes [27]. Analizando las diferentes técnicas de las que dispone la librería para la eliminación de partes específicas de imágenes, se ha optado por el método de **"inpainting"**.

Las técnicas de inpainting se utilizan para restaurar o reconstruir partes dañadas de una imagen, rellenando regiones de manera consistente con el contexto de la zona circundante [38]. La librería OpenCV cuenta con dos implementaciones de estas técnicas: Telea y Navier-Stokes (NS). Ambas implementaciones producen resultados simulares, comenzando a rellenar la zona desde los bordes hacia dentro: Telea emplea la suma ponderada de los píxeles que rodean la zona a rellenar para ir completando la imagen, mientras que NS intenta minimizar la varianza en los valores de los píxeles circundantes mediante técnicas heurísticas y algunas funciones de dinámicas de fluidos [28]. Tras realizar pruebas con ambos algoritmos, se ha

determinado utilizar **Navier-Stokes para eliminar el texto**, produciendo resultados más consistentes para tipografías más pesadas.

Ésta nueva implementación supone, a parte de la mejora en la calidad de la eliminación del texto, un aumento de la velocidad y eficiencia, puesto que emplea una máscara para determinar las áreas de la imagen que deben ser rellenadas. La máscara consiste en una imagen en blanco y negro, en la que los píxeles blancos representan las partes de la imagen a ser reemplazadas; una vez reconocido el texto y determinadas las secciones que contienen cada palabra individual, se genera esta máscara para eliminar todo el texto (reconocido) de la imagen.

5.4. Inserción del texto traducido

Una vez eliminado el texto de la imagen original se ha utilizado la librería gráfica incluida en el JDK de Java, Graphics2D, permitiendo la selección de la tipografía y tamaño del texto a dibujar. Utilizando el texto traducido y las posiciones del texto original en las que se ha eliminado el texto, se dibuja el texto traducido sobre la imagen original.

5.5. Servidor REST

El servidor REST se encarga de unir todos los componentes funcionales del sistema, de forma que puedan ser utilizados conjuntamente para dar servicio a las peticiones de los clientes.

5.5.1. Spark Framework

Para su implementación se ha empleado el *framework* Spark [29], que permite la creación de servidores REST en Java y Kotlin de manera sencilla sobre el servidor web Eclipse Jetty.

Como se puede ver en la figura 5.4, el funcionamiento del servidor es el siguiente:

1. Se registra la ruta /translate, encargada de recibir los datos de la imagen y opciones del usuario mediante una petición POST.
2. La imagen se decodifica de Base64 a imagen PNG para poder ser manipulada con normalidad.
3. Con la imagen decodificada, se aplican las técnicas para la mejora del reconocimiento del texto.
4. Se crea el reconocedor de texto con los parámetros introducidos por el usuario (idioma, nivel de confianza), produciendo una lista de palabras individuales y sus correspondientes representaciones espaciales en la imagen.

5. Tras la ejecución del OCR, se agrupan las palabras en frases completas, y se lleva a cabo la traducción del texto.

6. Una vez recibido el texto traducido, se procede a la eliminación del texto original de la imagen y la inserción del texto traducido.

7. La respuesta se serializa en formato JSON y se envía al cliente.

Cualquier error que se produzca durante la ejecución del flujo del servidor, terminará la ejecución y la correspondiente respuesta de error será enviada al cliente, también serializada en formato JSON.

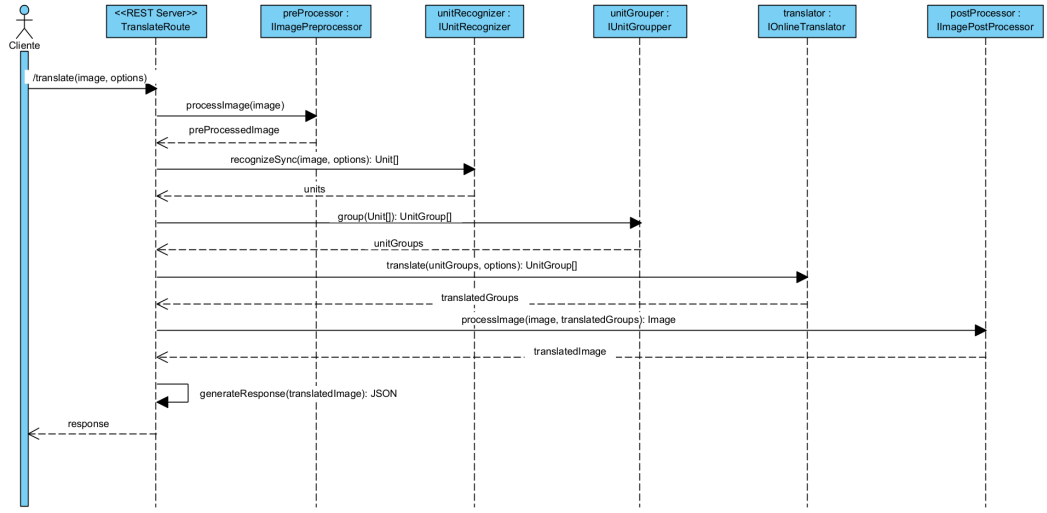


Figura 5.4: Diagrama de secuencia del servidor

5.6. Finalización del Sprint

Durante el desarrollo del Sprint, se ha producido un gran retraso para completar la tarea "2. Eliminar texto original", puesto que se ha tenido que descartar la primera implementación y comenzar de cero una segunda. Por otro lado, la tarea "4. Servidor REST", se ha realizado en menos tiempo del previsto. Ver tabla 5.2.

Pese al retraso sufrido, se ha decidido emplear tiempo extra para completar la tarea 4, en lugar de moverla al siguiente Sprint.

Tarea	**Estado**	**Desviación**
1. Mejoras imagen original	Completada	0 días
2. Eliminar texto original	Completada	3 días
3. Insertar texto traducido	Completada	0 días
4. Servidor REST	Completada	-1 día

Tabla 5.2: Revisión Sprint 3

Capítulo 6

Sprint 4: pruebas y aplicación de escritorio

En este capítulo, se presentan las pruebas realizadas al sistema. Una vez comprobado el correcto funcionamiento del servidor REST, se ha diseñado e implementado la aplicación cliente y se han realizado las pruebas pertinentes a la misma.

6.1. Planificación del Sprint

Las tareas asignadas a este Sprint se centran en las pruebas al sistema, diseño e implementación de la aplicación de escritorio y las pruebas realizadas a la misma. Ver tabla 6.1.

6.2. Pruebas al sistema

A continuación se presentan las pruebas más relevantes realizadas al sistema, puesto que se han realizado pruebas durante la fase de implementación del mismo, y los resultados obtenidos. Tras realizar todas las pruebas, no se han encontrado fallos.

Las pruebas al servidor REST se han realizado utilizando la herramienta Postman [39] y su función de realizar peticiones POST a un servidor, incluyendo el contenido en el *body* de la petición.

Tarea	Descripción	Duración estimada
1. Pruebas del sistema	Realizar pruebas al sistema y servidor REST para verificar su correcto funcionamiento.	1 día
2. Diseño aplicación de escritorio	Diseñar la interfaz de usuario de la aplicación de escritorio.	1 día
3. Implementación aplicación de escritorio	Implementar la aplicación de escritorio que conecte con la API REST del sistema.	5 días
4. Pruebas aplicación de escritorio	Realizar pruebas a la aplicación de escritorio para verificar su correcto funcionamiento.	1 día
Mejoras generales o imprevistos	Se ha asignado un tiempo para mejoras que necesite el sistema o cubrir posibles imprevistos que se den durante el desarrollo	5 días.

Tabla 6.1: Planificación Sprint 4

Prueba sistema	**01 - Procesamiento de imagen**
Descripción	Se ingresa una imagen con color y ruido al sistema y se verifica si la calidad mejora después de aplicar el pre-procesado de imagen.
Resultado esperado	La imagen de salida en blanco y negro y binarizada.
Resultado obtenido	Resultado esperado.

Tabla 6.2: Prueba sistema 01 - Procesamiento de imagen

Prueba sistema	**02 - Reconocimiento de texto**
Descripción	Se ingresa una imagen con texto al sistema y se verifica si el sistema extrae el texto correcto.
Resultado esperado	El texto de la imagen de entrada, con margen de error.
Resultado obtenido	Resultado esperado.

Tabla 6.3: Prueba sistema 02 - Reconocimiento de texto

Prueba sistema	**03 - Traducción de texto**
Descripción	Se ingresa un texto en el sistema y se verifica si la traducción es precisa y coherente.
Resultado esperado	La traducción obtenida debe ser precisa y coherente con el texto original.
Resultado obtenido	Resultado esperado.

Tabla 6.4: Prueba sistema 03 - Traducción de texto

Prueba sistema	**04 - Eliminación de texto**
Descripción	Se ingresa una imagen con texto al sistema y se verifica si el texto original es eliminado correctamente.
Resultado esperado	El área donde estaba el texto original debe ser rellenado de manera coherente al resto de la imagen.
Resultado obtenido	Resultado esperado.

Tabla 6.5: Prueba sistema 04 - Eliminación de texto

Prueba sistema	**05 - Dibujado del texto traducido**
Descripción	Se ingresa una imagen con texto eliminado al sistema y se verifica si el texto traducido se dibuja correctamente sobre la imagen.
Resultado esperado	El texto traducido debe estar ubicado en la imagen en la posición original.
Resultado obtenido	Resultado esperado.

Tabla 6.6: Prueba sistema 05 - Dibujado del texto traducido

Prueba sistema	**06 - Funcionamiento del servidor REST**
Descripción	Se realiza una solicitud HTTP POST al servidor REST y se verifica si se recibe una respuesta correcta y en el formato esperado.
Resultado esperado	Respuesta en formato JSON con los correspondientes campos: imagen traducida, transcripciones del texto, y opciones.
Resultado obtenido	Resultado esperado.

Tabla 6.7: Prueba sistema 06 - Funcionamiento del servidor REST

Prueba	**07 - Manejo de errores**
Descripción	Se realiza una petición errónea al sistema, con un idioma de entrada no soportado y se verifica si se recibe una respuesta de error, sin que el sistema se pare.
Resultado esperado	Respuesta en formato JSON con el contenido del error.
Resultado obtenido	Resultado esperado.

Tabla 6.8: Prueba sistema 07 - Manejo de errores

6.3. Diseño del cliente de escritorio

El diseño de la aplicación cliente se ha realizado utilizando la versión gratuita de Figma [40]. Como se puede ver en la figura 6.1, el diseño se centra en mostrar la funcionalidad del sistema y presentar todas las opciones de las que dispone, para poder ser accedidas por el cliente.

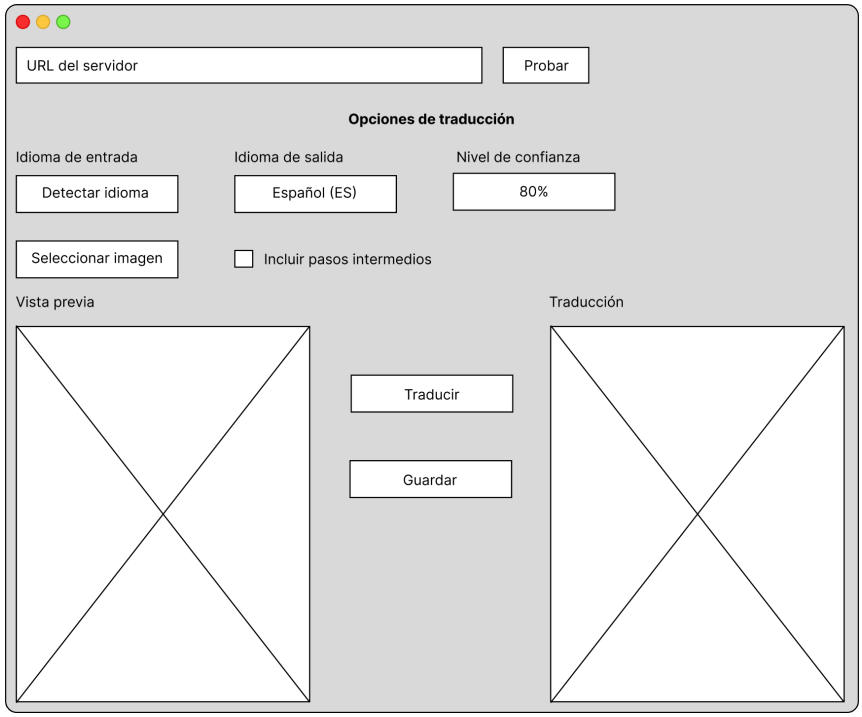


Figura 6.1: Diseño de aplicación de escritorio

6.4. Implementación del cliente de escritorio

Una vez verificado el correcto funcionamiento del sistema, y diseñada la aplicación de escritorio que dará servicio a los usuarios, se ha llevado a cabo la implementación utilizando el *framework* nativo de Java para interfaces de usuario, Java Swing [30]. Ver figura 6.2.

Finalmente la vista previa de la imagen seleccionada y su traducción, presentes en el diseño, se han descartado, puesto que al ser de un tamaño tan reducido, la legibilidad del texto era baja sino nula. Al haber sido implementada en Java utilizando librerías nativas del JDK, se puede ejecutar en cualquier plataforma para la que haya una implementación de Máquina Virtual de Java (JVM), como Windows, MacOS o Linux.

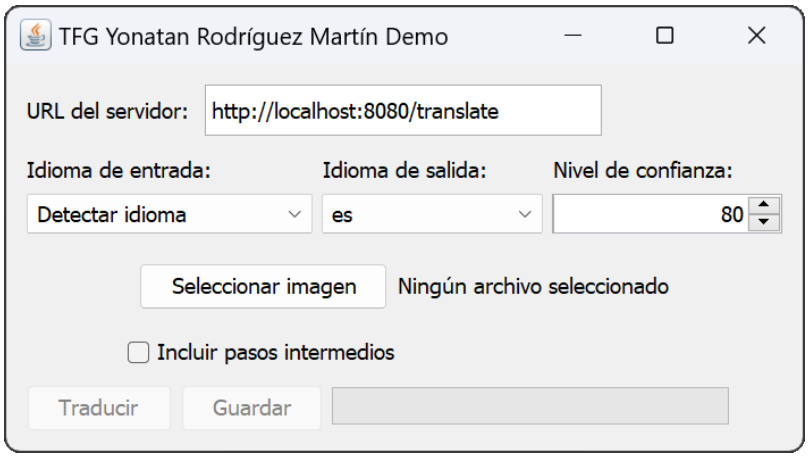


Figura 6.2: Implementación de aplicación de escritorio

6.5. Pruebas al cliente

Finalizada la implementación de la aplicación de escritorio, se han realizado las pruebas pertinentes. Estas se centran en la funcionalidad con la que interactúa el usuario y finalmente, la integración con el servidor REST del sistema. Adicionalmente, se ha probado la ejecución de la aplicación en Windows y Linux, siendo ésta correcta, y faltando la ejecución en MacOS por falta de disponibilidad de dispositivos para probarla. Tras la ejecución de las pruebas no se han encontrado fallos.

Prueba cliente	**01 - Selección de URL del servidor**
Descripción	Se modifica la URL del servidor en la aplicación de escritorio y se verifica que se conecte correctamente.
Resultado esperado	La URL del servidor se aplica para las peticiones del cliente.
Resultado obtenido	Resultado esperado.

Tabla 6.9: Prueba cliente 01 - Selección de URL del servidor

Prueba cliente	**02 - Selección de idioma de entrada**
Descripción	Se elige un idioma de entrada en la aplicación de escritorio y se verifica si se selecciona correctamente.
Resultado esperado	El idioma de entrada seleccionado se refleja correctamente en la interfaz de usuario y en el modelo.
Resultado obtenido	Resultado esperado.

Tabla 6.10: Prueba cliente 02 - Selección de idioma de entrada

Prueba cliente	**03 - Selección de idioma de salida**
Descripción	Se elige un idioma de salida en la aplicación de escritorio y se verifica si se selecciona correctamente.
Resultado esperado	El idioma de salida seleccionado se refleja correctamente en la interfaz de usuario y en el modelo.
Resultado obtenido	Resultado esperado.

Tabla 6.11: Prueba cliente 03 - Selección de idioma de salida

Prueba cliente	**04 - Selección de nivel de confianza**
Descripción	Se elige un nivel de confianza en la aplicación de escritorio y se verifica si se selecciona correctamente.
Resultado esperado	El nivel de confianza seleccionado se refleja correctamente en la interfaz de usuario y en el modelo.
Resultado obtenido	Resultado esperado.

Tabla 6.12: Prueba cliente 04 - Selección de nivel de confianza

Prueba cliente	**05 - Codificación de imagen**
Descripción	Se selecciona una imagen en la aplicación de escritorio y se verifica si se codifica correctamente en Base64.
Resultado esperado	La imagen seleccionada se codifica en Base64 correctamente y está lista para enviar al servidor
Resultado obtenido	Resultado esperado.

Tabla 6.13: Prueba cliente 05 - Codificación de imagen

Prueba cliente	**06 - Guardado del resultado de traducción**
Descripción	Se guarda el resultado de la traducción en una carpeta y se verifica si se guarda correctamente.
Resultado esperado	El resultado de la traducción se guarda correctamente en la carpeta especificada
Resultado obtenido	Resultado esperado.

Tabla 6.14: Prueba cliente 06 - Guardado del resultado de traducción

Prueba cliente	**07 - Generación de pasos intermedios**
Descripción	Se marca la opción de generación de pasos intermedios y se verifica si se generan correctamente.
Resultado esperado	El resultado de la traducción contiene los pasos intermedios de la traducción.
Resultado obtenido	Resultado esperado.

Tabla 6.15: Prueba cliente 07 - Generación de pasos intermedios

6.6. Finalización del Sprint

La estimación inicial para la implementación de la aplicación de escritorio era de cinco días, pero finalmente se han necesitado siete para completarla, puesto que el riesgo "RSK10 - Estimación inadecuada de costes" se ha materializado. Junto a los tres días de retraso anteriores, se acumula un retraso que completa las cuatro semanas asignadas al Sprint, puesto que se asignó una semana extra para imprevistos o mejoras, y la fecha de entrega prevista no se ha visto afectada. Ver tabla 6.16.

Tarea	**Estado**	**Desviación**
1. Pruebas del sistema	Completada	0 días
2. Diseño aplicación de escritorio	Completada	0 días
3. Implementación aplicación de escritorio	Completada	2 días
4. Pruebas aplicación de escritorio	Completada	0 días

Tabla 6.16: Revisión Sprint 4

Capítulo 7

Conclusiones y trabajo futuro

7.1. Conclusiones

Finalizado el desarrollo este proyecto de fin de grado, se detallan los objetivos cumplidos, competencias reforzadas y aprendidas por el alumno:

- Se ha implementado un servicio REST utilizando una tecnología nueva como es Spark.
- Se ha estudiado el funcionamiento de las técnicas de OCR.
- Se ha implementado el reconocimiento mediante OCR utilizando Tesseract OCR.
- Se ha diseñado a medida una estructura de datos bidimensional para el manejo del texto reconocido.
- Se ha diseñado un algoritmo eficiente para la unión del texto.
- Se ha utilizado una API REST externa para la traducción del texto, DeepL.
- Se han estudiado técnicas de eliminación de texto en imagen, como el "inpainting".
- Se ha utilizando la librería OpenCV para la eliminación de texto en imagen mediante el algoritmo Navier-Stokes de "inpainting".
- Se ha reforzado el conocimiento del lenguaje de programación Java.
- Se han aplicado técnicas y patrones de diseño estudiados: patrón constructor, modelo-vista-controlador, herencia, polimorfismo.
- Se ha diseñado una aplicación de escritorio que utiliza el servicio REST creado.
- Se ha implementado una aplicación de escritorio utilizando Java Swing, afianzando los conocimientos de desarrollo de aplicaciones multiplataforma.

7.2. Trabajo futuro

Pese a que la mayor parte de objetivos planteados para el proyecto se han conseguido, ya sea por razones técnicas u otro tipo, algunos no han podido cumplirse y quedan planteados como trabajo futuro, así como nuevas implementaciones de funcionalidades ya existentes para mejorar el sistema:

- Debido a la falta de un modelo entrenado para el reconocimiento de tipografías, no se ha podido implementar este objetivo, ya que estos modelos tienen un coste elevado, tanto temporal como económico, puesto que tienen que ser entrenados en servidores especializados por Google.
- Implementación de técnicas de reescalado de la imagen de entrada para mejorar la legibilidad para imágenes de menor tamaño, mejorando el reconocimiento de texto.
- Actualmente los modelos utilizados para el reconocimiento de texto empleados por Tesseract OCR son estáticos, es decir, en caso de actualizarse en un futuro quedarían obsoletos, por lo que se puede implementar un mecanismo para actualizarlos conforme avanza la tecnología.
- Implementación de otros servicios de traducción locales y en línea, para lo cual no es necesario modificar el sistema, únicamente realizar nuevas implementaciones de las interfaces ya disponibles.
- La técnica para la eliminación del texto se ha vuelto obsoleta durante el desarrollo del proyecto, apareciendo nuevas técnicas que utilizan inteligencia artificial para este propósito, obteniendo resultados de mayor calidad, por lo que se podrían aplicar dichas técnicas si se tuviera acceso a sus algoritmos.
- Implementación de un número mayor de aplicaciones cliente que utilicen el servicio REST, como un servidor web, una aplicación móvil o una nueva aplicación de escritorio realizada sobre un *framework* multiplataforma como puede ser Flutter.
- Realizar una nueva implementación del servidor REST en un lenguaje compilado como Rust o C++ para mejorar el uso de recursos y velocidad del sistema.

Apéndice A

Manuales

A.1. Contenido de la memoria USB

El contenido de la memoria USB es el siguiente:

- Servidor/: se encuentran los archivos relacionados con el servidor REST.
  - TFG/: código fuente del servidor en Java.
  - TFG Backend.jar: ejecutable compilado del servidor.
- Aplicación/: se encuentran los archivos relacionados con la aplicación de escritorio cliente.
  - TFG Frontend/: código fuente de la aplicación cliente en java.
  - TFG Frontend.jar: ejecutable compilado de la aplicación de escritorio.
- Memoria/: se encuentra esta memoria en formato PDF.

El código fuente del servidor [1] y del cliente [2] se encuentran también en repositorios del GitLab de la Escuela de Ingeniería Informática.

A.1.1. Requisitos de ejecución

Es necesaria la instalación previa de Java 19 (o versión superior) para la ejecución tanto servidor como de la aplicación de escritorio. La instalación se realizará desde la página de Oracle [3].

A.2. Manual de despliegue del servidor

A.2.1. Ejecución y uso del servidor

Se utilizará el archivo "TFG Backend.jar" utilizando un Garbage Collector especializado para los servicios de alta demanda como los servidores web: ZGC [19]. Para ello se ejecutará el siguiente comando:

```
> java -XX:+UseZGC -jar "TFG Backend.jar"
```

Tras lo cual el servidor mostrará las rutas registradas y el puerto en el que se está ejecutando. Ver figura A.1.

```
C:\Proyectos>java -XX:+UseZGC -jar "TFG Backend.jar"
Starting server on port 8080...
Adding routes...
Loaded routes:
        post /translate
Server started on port 8080
```

Figura A.1: Salida del comando de ejecución

Una vez iniciado el servidor, las peticiones se deben hacer mediante el método POST y en formato JSON, y deben contener los siguientes valores:

- imgData: cadena de caracteres que contiene la información de la imagen a traducir codificada en Base64
- options:
  - srcLang: opcional, idioma de la imagen de entrada
  - targetLang: idioma a traducir
  - confidenceLevel: nivel de confianza del reconocimiento de texto
  - debug: opcional, si se incluyen o no los pasos intermedios de la imagen

[1] Repositorio del servidor: `https://gitlab.inf.uva.es/yonrodr/tfg`
[2] Repositorio del cliente: `https://gitlab.inf.uva.es/yonrodr/tfg-demo`
[3] Página de descarga de java: `https://www.oracle.com/java/`

A.3. Manual de usuario

A.3.1. Ejecución y de la aplicación de escritorio

La aplicación de escritorio se puede ejecutar haciendo doble click sobre el ejecutable “TFG Frontend.jar” o desde la línea de comandos con el siguiente comando:

```
> java -jar "TFG Frontend.jar"
```

Una vez iniciada la aplicación (ver figura A.2), el usuario puede cambiar la URL del servidor a utilizar, el idioma de entrada - que por defecto lo detecta el sistema - y el idioma de salida y el nivel de confianza.

Seleccionados los parámetros de reconocimiento y traducción, al pulsar el botón “Seleccionar imagen”, un selector de archivos sólo de imágenes se abrirá, permitiendo al usuario seleccionar la imagen de entrada. Tras ello, si el usuario ha seleccionado una imagen, se activa el botón de traducir. Se puede activar o desactivar la generación de pasos intermedios en el correspondiente selector. Tras traducir el texto, el botón de guardado quedará activo, y al pulsarlo un nuevo selector de archivos se abrirá para elegir el directorio de salida, en el que se guardará la salida del sistema. Ver figura A.3.

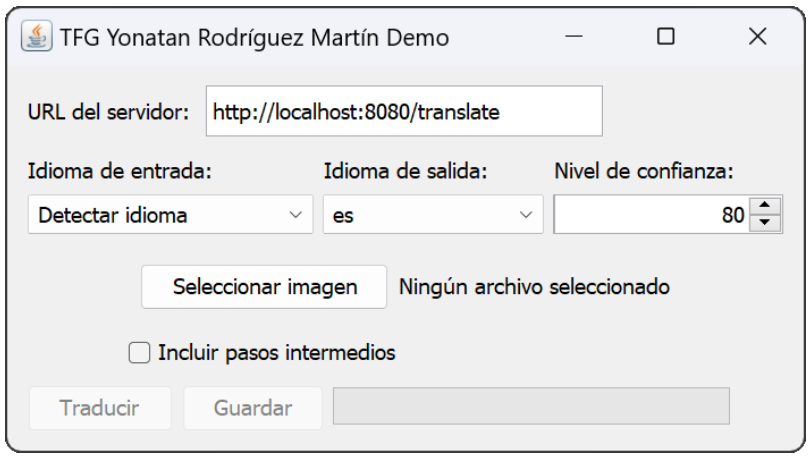


Figura A.2: Aplicación cliente

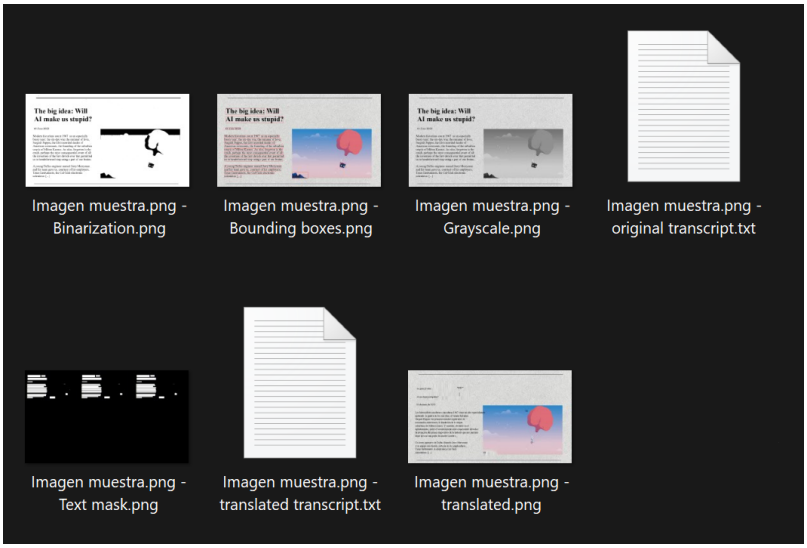


Figura A.3: Salida de la aplicación cliente

Bibliografía

[1] Google translate. https://translate.google.es/, Recuperado el 27 de abril de 2023.

[2] Deepl translator. https://www.deepl.com/translator/files, Recuperado el 27 de abril de 2023.

[3] Google lens. https://lens.google/, Recuperado el 27 de abril de 2023.

[4] Deepl transformer. https://www.deepl.com/en/blog/how-does-deepl-work, Recuperado el 03 de mayo de 2023.

[5] Deepl linguee. https://es.wikipedia.org/wiki/Linguee, Recuperado el 03 de mayo de 2023.

[6] Bing translator. https://www.bing.com/translator, Recuperado el 27 de abril de 2023.

[7] Yandex translate. https://translate.yandex.com/es/, Recuperado el 27 de abril de 2023.

[8] José H Canós, Patricio Letelier, and María Carmen Peñadés. Metodologías ágiles en el desarrollo de software. *Universidad Politécnica de Valencia, Valencia*, pages 1–8, 2003. Recuperado el 15 de mayo de 2023.

[9] Manuel Trigás Gallego. Metodología scrum. *Universitat Oberta de Catalunya*, 2012. Recuperado el 15 de mayo de 2023.

[10] International Function Point Users Group. IFPUG Standards. https://ifpug.org/ifpug-standards/fpa, Recuperado el 15 de mayo de 2023.

[11] Amazon ec2. https://aws.amazon.com/ec2/, Recuperado el 19 de mayo de 2023.

[12] J. Hammersley and J. Lees-Miller. Overleaf. https://es.overleaf.com/, 2012. Recuperado el 15 de mayo de 2023.

[13] dotPDN LLC. Paint.net. https://www.getpaint.net/, Recuperado el 19 de mayo de 2023.

[14] Microsoft project. https://www.microsoft.com/es-es/microsoft-365/project/, Recuperado el 19 de mayo de 2023.

[15] Jetbrains store. https://www.jetbrains.com/store/#personal, Recuperado 19 de mayo de 2023.

[16] Deepl pro developer. https://www.deepl.com/pro#developer, Recuperado el 19 de mayo de 2023.

[17] Talent.com salary - desarrollador java. https://es.talent.com/salary?job=desarrollador+java#:~:text=El%20salario%20desarrollador%20java%20promedio,hasta%20%E2%82%AC%2044.900%20al%20a%C3%B1o., Recuperado el 19 de mayo de 2023.

[18] Oracle. Java garbage collection basics. https://www.oracle.com/webfolder/technetwork/tutorials/obe/java/gc01/index.html, 2014. Recuperado el 07 de junio de 2023.

[19] Oracle. Z garbage collector. https://docs.oracle.com/en/java/javase/11/gctuning/z-garbage-collector1.html, 2022. Recuperado el 07 de junio de 2023.

[20] Wikipedia contributors. Websocket. https://en.wikipedia.org/wiki/WebSocket, 2023. Recuperado el 07 de junio de 2023.

[21] Laurent Debrauwer. *Patrones de diseño en Java: los 23 modelos de diseño: descripciones y soluciones ilustradas en UML 2 et Java.* Ediciones ENI, 2018. Recuperado el 11 de junio de 2023.

[22] Sylvain Lefebvre and Hugues Hoppe. Perfect spatial hashing. https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/33418.pdf, 2007. Recuperado el 11 de junio de 2023.

[23] The Apache Software Foundation. Apache maven, 2023. Recuperado el 12 de junio de 2023.

[24] Wikipedia contributors. Caja negra (sistemas). https://es.wikipedia.org/wiki/Caja_negra_(sistemas), 2022. Recuperado el 14 de junio de 2023.

[25] Google, Inc. Tesseract ocr. https://github.com/tesseract-ocr/tesseract, 2023. Recuperado el 12 de junio de 2023.

[26] DeepL. Deepl api documentation. https://www.deepl.com/docs-api, 2023. Recuperado el 03 de mayo de 2023.

[27] Opencv documentation. https://docs.opencv.org/4.7.0/, Recuperado el 02 de mayo de 2023.

[28] Marcelo Bertalmio, Andrea L Bertozzi, and Guillermo Sapiro. Navier-stokes, fluid dynamics, and image and video inpainting. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1. IEEE, 2001. Recuperado el 12 de junio de 2023.

[29] Per Wender and Love Löfdahl. Spark, 2020. Recuperado el 12 de junio de 2023.

[30] Oracle Corporation. Java Swing. https://docs.oracle.com/javase/tutorial/uiswing/, 2021. Recuperado el 14 de junio de 2023.

[31] Yanette Díaz González and Yenisleidy Fernández Romero. Patrón modelo-vista-controlador. *Telemática*, 11(1):47–57, 2012. Recuperado el 14 de junio de 2023.

[32] Inc Google. Tessdata - best lstm models. `https://github.com/tesseract-ocr/tessdata_best`, 2017. Recuperado el 12 de junio de 2023.

[33] Ray at Google Inc. Smith. An overview of the tesseract ocr engine. `https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/33418.pdf`, 2007. Recuperado el 11 de junio de 2023.

[34] Oracle. Objects documentation. `https://docs.oracle.com/javase/8/docs/api/java/util/Objects.html#hash-java.lang.Object...-`, 2014. Recuperado el 07 de junio de 2023.

[35] DeepL. Deepl java api. `https://github.com/DeepLcom/deepl-java`, 2023. Recuperado el 03 de mayo de 2023.

[36] Tesseract OCR Documentation. Improving the quality of the output. `https://tesseract-ocr.github.io/tessdoc/ImproveQuality.html`, 2022. Recuperado el 15 de junio de 2023.

[37] Leptonica. Leptonica - An Image Processing Library. `http://www.leptonica.org/index.html`, Recuperado el 16 de junio de 2023.

[38] OpenCV. OpenCV Tutorial - Inpainting. `https://docs.opencv.org/3.4/df/d3d/tutorial_py_inpainting.html`, 2021. Recuperado el 16 de junio de 2023.

[39] Postman. `https://www.postman.com/`. Recuperado el 18 de junio de 2023.

[40] Figma. `https://www.figma.com`. Recuperado el 18 de junio de 2023.

[41] Universidad de Valladolid. Proyecto docente del trabajo de fin de grado 2020-2021 (Mención Tecnologías de la Información). `https://alojamientos.uva.es/guia_docente/uploads/2019/545/46977/1/Documento.pdf`, 2020. Recuperado el 14 de abril de 2021.

[42] Change Vision. Astah* professional. `https://tecnicos.blogs.inf.uva.es/astah-professional/`, 2020. Recuperado el 20 de mayo de 2021.

[43] Oracle. Java native interface documentation. `https://docs.oracle.com/javase/8/docs/technotes/guides/jni/spec/intro.html`, 2014. Recuperado el 14 de junio de 2023.

[]