



Universidad de Oriente
Sede “Julio Antonio Mella”
Facultad de Ingeniería en Telecomunicaciones, Informática y
Biomédica

Trabajo de Diploma

En opción al título de Ingeniería en Informática

Título: “Sistema informático para el reconocimiento de entidades nombradas en documentos digitales”.

Autor: Luis Andrés Licea Berenguer.

Tutores: Dr. C. Dionis López Ramos.

Ms. C. Jose Erinaldo Cruzata Ferrer

2024
Santiago de Cuba,
“Año 66 de la Revolución”

Mis agradecimientos a:

Mi abuela por ser la persona más importante de mi vida, cuya presencia y apoyo han sido mi fortaleza. A la memoria de mi abuelo Luis, que estaría muy orgulloso.

A mis padres por todo el amor, y la educación que me han dado. Mi madre, cuyo apoyo y aliento han sido invaluable, me demostró lo que es la superación en el estudio y tu fuerza y determinación han sido mi guía en este viaje académico. Mi padre, cuya pasión por la carrera que he elegido ha sido mi inspiración y ejemplo a seguir. Es un privilegio para mí tenerlos como padres.

A mi tía, por ser una segunda madre para mí e impulsarme a seguir cada día.

A mi padrastro, por su apoyo y enseñanzas de vida. Tu confianza en mí y tu constante ánimo han sido un impulso invaluable.

Mi novia, por su amor incondicional, comprensión y paciencia durante los momentos más desafiantes de este viaje. Gracias por formar parte de mi vida.

A mis abuelos Héctor y Elena y mi tía Tatica por todo el cariño y apoyo.

A mis hermanas, espero que esta investigación les sirva de ejemplo a seguir.

Al Dr. C. Dionis López Ramos por guía, comprensión y empatía sin las que el trabajo no hubiera sido posible.

Al centro de desarrollo de software DATYS por acogerme para la realización de esta investigación y en especial al tutor del centro Ms.C. Jose Erinaldo Cruzata.

Al colectivo de profesores de la carrera de Ingeniería Informática.

A mi familia, amigos y compañeros de aula por todos los momentos divertidos.

RESUMEN

La habilidad de discernir entidades nombradas, como personas, lugares, organizaciones y otros elementos específicos en el texto, es un componente crítico en la extracción de conocimiento y toma de decisiones. En la actualidad, el reconocimiento de entidades nombradas desempeña un papel fundamental en diversos contextos, desafiando la capacidad de las aplicaciones para identificar y comprender información clave en grandes volúmenes de datos no estructurados. Este trabajo y sus resultados tributan a la empresa de desarrollo de software DATYS, mediante el desarrollo de un sistema informático para el reconocimiento de entidades nombradas utilizando la biblioteca spaCy y que permite el reentrenamiento de modelos computacionales especializados en el reconocimiento de estas entidades. Se emplean técnicas de procesamiento de lenguaje natural e inteligencia artificial para lograr estos objetivos. En el reentrenamiento de los modelos se utiliza un gran modelo de lenguaje nombrado Llama2 como herramienta para la generación de un nuevo conjunto de datos, aportando un valor significativo a la herramienta y cumpliendo con los requerimientos iniciales.

Palabras clave: Reconocimiento de entidades nombradas, spaCy, procesamiento del lenguaje natural, inteligencia artificial.

ABSTRACT

The ability to discern named entities such as people, places, organizations, and other specific elements in text is a critical component in knowledge extraction and decision-making. In today's world, named entity recognition plays a fundamental role in various contexts, challenging the capability of applications to identify and understand key information in large volumes of unstructured data. This work contributes to the software development company DATYS by developing a computer system for named entity recognition using the spaCy library, which allows for the retraining of computational models specialized in recognizing these entities. The use of natural language processing and artificial intelligence techniques is employed to achieve these objectives. In the retraining of the models, a large language model called Llama2 is used as a tool for generating a new set of data, adding significant value to the tool and meeting the initial requirements demanded.

Keywords: Named Entity Recognition, spaCy, Natural Language Processing, Artificial Intelligence, Unstructured Data.

Tabla de contenido

INTRODUCCIÓN	1
CAPITULO 1 . MARCO TEÓRICO	6
1.1 Procesamiento del Lenguaje Natural	6
1.2 Reconocimiento de entidades nombradas	6
1.3 Estado del Arte de los marcos de procesamiento del lenguaje natural para la detección de entidades	7
1.3.1 Google NLP	8
1.3.2 ChatGPT	8
1.3.3 IBM Watson Discovery	8
1.3.1 Estado del arte de modelos o bibliotecas para el reconocimiento de entidades nombradas	9
1.3.2 Comparación de los modelos de reconocimiento de entidades	10
1.4 Herramientas, lenguajes de programación y tecnologías empleadas en el sistema propuesto en la investigación	11
1.4.1 Marco de trabajo del módulo de la capa del cliente	12
1.4.2 Marco de trabajo del módulo de la capa del servidor	13
1.4.3 Herramientas utilizadas para el desarrollo del sistema	14
1.4.4 Base de datos utilizada	15
1.5 Metodología XP	15
1.6 Conclusiones del capítulo	16
CAPITULO 2 . ORGANIZACIÓN Y DISEÑO	17
2.1 Modelo de negocio para el REN	17
2.2 Arquitectura del sistema	19
2.3 Flujo del sistema	19
2.4 Requerimientos para el despliegue del sistema	20
2.5 Diseño de la base de datos	21
2.6 Historias de Usuario	23
2.7 Usuarios del Sistema	27
2.8 Diagrama de Secuencia de las interacciones de los objetos del sistema a lo largo del tiempo	27
2.9 Diagrama de Actividades del flujo del sistema	29
2.10 Conclusiones del capítulo	30
CAPITULO 3 . IMPLEMENTACIÓN Y PRUEBA	31

3.1	Instalación de los requisitos para el funcionamiento del sistema	31
3.2	Patrones de diseño empleados	31
3.3	Diagrama de clases.....	32
3.4	Algoritmos Importantes	33
3.4.1	Reconocimiento de Entidades Nombradas a un índice de Elasticsearch.....	33
3.4.2	Salvar resultados del reconocimiento en el índice seleccionado	35
3.4.3	Generar datos de entrenamiento	36
3.4.4	Reentrenar el modelo de spaCy evitando el olvido catastrófico	38
3.5	Medidas de Desempeño	40
3.6	Análisis económico del costo de producción del sistema.....	41
3.7	Pruebas al sistema	42
3.8	Ejemplo de resultados obtenidos	48
3.9	Implementación de la Interfaz de Usuario para el REN	50
3.10	Conclusiones del capítulo.....	50
CONCLUSIONES Y RECOMENDACIONES		51
REFERENCIAS BIBLIOGRÁFICAS.....		52
ANEXOS.....		57

INTRODUCCIÓN

El término "entidad nombrada" fue introducido por primera vez en la Conferencia de Comprensión de Mensaje en Maryland, EEUU (Message Understanding Conference, MUC-6) en 1996 (Sundheim,1996). En la misma se abordaron aspectos relevantes implicados en la evaluación de sistemas de extracción de información aplicados a una tarea común; enfocándose en la identificación de nombres de organizaciones, personas, ubicaciones geográficas, y expresiones monetarias en textos. Desde entonces, el interés en el reconocimiento de entidades nombradas (REN) ha crecido significativamente, siendo un componente esencial en el procesamiento del lenguaje natural (PLN) y la inteligencia artificial (IA) (Carvalho,2023).

Un ejemplo de un REN a una oración sería: “**Fernanda Morales**, originaria de **España** se unió a la **ONU** para trabajar en proyectos de desarrollo en **África**” donde se extraen y clasifican a Fernanda Morales de tipo persona, España y África de tipo localización, y ONU de tipo organización. El resultado del REN es valioso en áreas como el monitoreo de redes sociales, donde el seguimiento de la actividad y la interacción entre personas, organizaciones y ubicaciones puede revelar información crucial para la toma de decisiones estratégicas, la detección de tendencias, así como la comprensión de la dinámica social.

En la actualidad existen diferentes modelos y herramientas para hacer el REN entre las que destacan spaCy, Stanford, Flair, BERT entre otras. Estos requieren de datos etiquetados para su entrenamiento y ajuste, lo que significa que necesitan ejemplos de texto con las entidades nombradas ya identificadas y clasificadas para aprender a reconocerlas de manera efectiva, un ejemplo de datos etiquetados se encuentra en el [Anexo I](#). Sin embargo, la creación de estos datos de forma manual puede suponer un desafío debido al tiempo de trabajo y la cantidad de personal requerido para esta tarea (Nasar, 2021).

Para abordar la problemática anterior, la inteligencia artificial generativa (IAG) emerge como una herramienta útil (Bernabeu, 2022). Esta IAG posee modelos generadores de oraciones como Llama2 que pueden ser de gran utilidad para facilitar este proceso, proporcionando ejemplos de textos con entidades ya etiquetadas que pueden servir como base para el entrenamiento de los modelos de REN (Shelar, 2020).

Tomando en consideración que, desde la aparición de la internet, la cantidad de información disponible en distintos formatos y fuentes ha crecido a pasos agigantados, por lo que el REN puede ser considerado una solución viable para encontrar información específica, ya sea, en un solo documento o en un conjunto de documentos (Liu, 2022; Carvalho, 2023).

Empresas e investigadores han desarrollado sistemas y modelos para el REN mostrando una alta precisión para el idioma inglés, así como una gran diversidad de clasificaciones de los tipos de entidades nombradas, muchos de estos son de pago dificultando su utilización (Shelar, 2020). Sin embargo, para el resto de los idiomas incluido el español no ocurre de igual manera, debido a que la mayoría de las mejores soluciones son de habla inglesa y se requiere un gran volumen de datos etiquetados para entrenar los modelos de PLN para el REN (Shelar, 2020; Yadav, 2019).

DATYS una empresa de desarrollo de software con una sucursal en Santiago de Cuba, se dedica a la creación de soluciones tecnológicas para diversos sectores. Varias de sus soluciones de software se enfrentan a la tarea de procesar y analizar grandes volúmenes de datos en español almacenados en la base de datos Elasticsearch. Debido a estas circunstancias el desarrollo de herramientas digitales que permitan hacer el reconocimiento de entidades nombradas y almacenarlas para su posterior estudio es un requerimiento de varios clientes de los sistemas y servicios que tiene la empresa.

Problema de investigación:

La empresa DATYS de Santiago de Cuba tiene limitaciones en el reconocimiento de entidades nombradas en español al procesar volúmenes masivos de datos diariamente y ser almacenados en el motor de búsqueda de texto completo Elasticsearch.

De acuerdo con los autores (Pardo Gómez, 2021; Hernández-Sampieri, 2020) el objeto de investigación se define como:

- La cultura desde la cual se puede dar solución al problema y en la que este último se manifiesta.
- Permite delimitar el área del saber en el que se produce el problema que da lugar a la investigación.

- Debe distinguirse entre el objeto de la investigación y el objeto, proceso o fenómeno que es investigado: el “objeto de la investigación” no es la realidad misma, sino una abstracción de esa realidad; una expresión de la realidad en un plano mental.

En esta investigación se define como **objeto de investigación**:

Los modelos computacionales para el procesamiento del lenguaje natural.

Teniendo en cuenta lo expuesto en (Pardo Gómez, 2021; Hernández-Sampieri, 2020) campo de acción se define como:

- Aquel subproceso del objeto de investigación con el que trabaja específicamente el investigador y desde donde transforma el objeto de investigación.
- Al ser una delimitación del objeto de la investigación expresa la cultura desde la cual de manera específica el investigador pretende dar solución al problema presente en el “objeto, es decir, desde donde el mismo pretende efectuar su aporte.
- Debe hacerse mediante conceptos o categorías precisas, que permitan definir con claridad qué parte del objeto es la que va a recibir directamente la acción del investigador.

En esta investigación se define como **campo de acción**:

Sistemas informáticos para el reconocimiento de entidades nombradas en español como parte del procesamiento del lenguaje natural.

Es importante considerar que la extracción de entidades nombradas es una tarea de investigación del procesamiento de la lengua natural y los sistemas informáticos (ed., bibliotecas, marcos de trabajo o aplicaciones) utilizan modelos computacionales para la extracción de entidades nombradas.

Objetivo general:

Elaboración de un sistema informático para el reconocimiento de entidades nombradas en el procesamiento de grandes volúmenes de documentos digitales en español en la empresa DATYS.

Objetivos específicos:

1. Investigar las técnicas en el campo del procesamiento del lenguaje natural para la detección de entidades nombradas.

2. Diseñar una aplicación web como sistema informático para el reconocimiento de entidades nombradas en información textual en español, según los requerimientos de la empresa DATYS
3. Implementar el sistema diseñado para el reconocimiento de entidades nombradas y la generación de datos de entrenamiento.
4. Implementar un algoritmo que permita resolver el reto de poder extraer nuevas entidades nombradas al reentrenar un sistema para el reconocimiento de entidades nombradas.
5. Realizar evaluaciones para verificar las medidas de desempeño del sistema desarrollado.

Hipótesis:

Un sistema informático para el reconocimiento de entidades nombradas en español permite contribuir a disminuir las limitaciones actuales en el procesamiento de grandes volúmenes de documentos digitales en español en la empresa DATYS al ser almacenados en el motor de búsqueda de texto completo Elasticsearch.

Métodos de investigación:

- **Método histórico-lógico:** Se utilizó principalmente para analizar la evolución de las tecnologías y métodos utilizados en el REN, como ha ido cambiando la tarea a lo largo del tiempo, desde métodos basados en reglas hasta el uso de tecnologías de aprendizaje automático.
- **Método de análisis y síntesis:** se aplicó al realizar el análisis de todo el proceso llevado a cabo en el proyecto y sintetizar las ideas que fueron surgiendo; extrayendo los elementos comunes al objeto de estudio, analizar las diferentes herramientas, bibliotecas y sistemas que realizan el REN.
- **Método de modelado:** Se empleó para realizar los diagramas necesarios para documentar el software.
- **Método de entrevistas:** Se utilizó para entrevistar a la directora del centro de desarrollo de software DATYS, así como a los lingüistas de la entidad, en la cual se abordó temas sobre las problemáticas que tenían en el análisis de grandes volúmenes de datos para la extracción de información.

Aportes de la Investigación:

- Se contará con un sistema que reconozca las entidades nombradas en español en información textual de grandes volúmenes de datos almacenados en Elasticsearch.
- El sistema tendrá la capacidad de reentrenamiento por parte del usuario en caso de surgir o no detectar una nueva entidad.
- El sistema contará con un espacio para la generación de datos de entrenamiento y prueba de modelos computacionales especializados en la detección de entidades nombradas.

CAPITULO 1 . MARCO TEÓRICO

En este capítulo se explican los principales aspectos teóricos, los conceptos básicos de las tecnologías y la caracterización de las herramientas computacionales utilizadas.

1.1 Procesamiento del Lenguaje Natural

El PLN es un campo de la inteligencia artificial y la lingüística computacional que brinda a los ordenadores la capacidad de interpretar, manipular y comprender el lenguaje humano. Hoy en día, las organizaciones tienen grandes volúmenes de datos de varios canales de comunicación, como correos electrónicos, mensajes de texto, fuentes de noticias en redes sociales, vídeo, audio y más. Utilizan software de PLN para procesar de forma automática estos datos, analizan la intención o el sentimiento del mensaje y responden en tiempo real a la comunicación humana (Gelbukh, 2010; Vásquez, 2009; Moreira, 2021).

Esta área de investigación de la lingüística computacional tiene varias tareas de investigación como:

- El análisis de correferencias.
- El análisis de sentimientos.
- La traducción automática.
- El reconocimiento de entidades nombradas.

1.2 Reconocimiento de entidades nombradas

Una entidad nombrada es una palabra o una frase que identifica claramente un elemento de un conjunto de otros elementos que tienen similares atributos (Albuquerque, 2023). Estas pueden ser clasificadas en: organización, nombres de personas y lugares en el dominio general, tal como se muestra en la [Fig 1.1](#). En este caso las organizaciones son clasificadas con la etiqueta de ORG, las personas como PERSON, y las localizaciones como LOC.

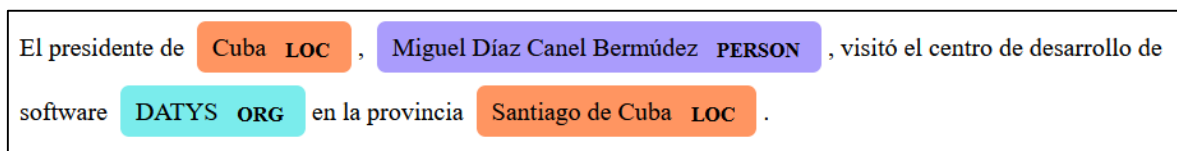


Fig 1.1 Ejemplo de reconocimiento de entidades nombradas

El REN, una tarea del procesamiento del lenguaje natural, es el proceso de localizar y clasificar entidades nombradas en el texto en categorías de entidades predefinidas. El término "Entidad Nombrada" fue utilizado por primera vez en la sexta Conferencia de

Comprensión de Mensajes (MUC-6, como la tarea de identificar nombres de organizaciones, personas y ubicaciones geográficas en el texto, así como como expresiones de moneda, tiempo y porcentaje. Desde MUC 6 ha habido un creciente interés en REN, y varios eventos científicos que dedican mucho esfuerzo a este tema (Wang, 2020; Wang, 2022; Missaoui, 2019). El REN varía mucho en función del idioma en el que han sido entrenadas. Cuando se realiza el análisis histórico lógico de los sistemas para el REN se evidencia como la evolución de las tecnologías influyó en los avances en el PLN.

En una primera etapa del desarrollo de los sistemas REN, se utilizaban métodos basados en reglas. Estos métodos se centraban en la aplicación de un conjunto predeterminado de reglas para la extracción de información, basándose en patrones morfológicos de las palabras o en el contexto en el que se encontraban dentro del documento (Goyal, 2018; Liu, 2022).

Con el avance de las tecnologías se introdujeron métodos basados en el aprendizaje automático para el REN. Este enfoque implicaba la anotación de datos para entrenar el modelo, permitiendo que pudiera aprender a identificar y clasificar entidades a partir de ejemplos previamente etiquetados (Goyal, 2018; Liu, 2022).

El aprendizaje profundo marcó una etapa significativa en la evolución de estas tecnologías. Las cuales son capaces de comprender las relaciones semánticas y sintácticas entre las palabras de un texto, lo que permite una identificación y clasificación más precisas de las entidades nombradas (Goyal, 2018; Liu, 2022).

1.3 Estado del Arte de los marcos de procesamiento del lenguaje natural para la detección de entidades

En el ámbito del PLN, el análisis del estado del arte de las soluciones para la REN representa un punto de partida crucial para cualquier investigación o desarrollo en este campo. Estas soluciones, que abarcan desde propuestas comerciales hasta herramientas de código abierto, han evolucionado significativamente para ofrecer capacidades avanzadas de análisis y extracción de información estructurada de textos. Entre las más relevantes se encuentran el Google NLP, ChatGPT e IBM Watson Discovery.

1.3.1 Google NLP

Google Cloud NLP es una interfaz de programación de aplicaciones (API, siglas en inglés) de Inteligencia Artificial enfocada en el PLN, parte de la suite de productos digitales de Google. Esta API se especializa en la creación de contenido que emula lo producido por humanos, ofreciendo un análisis de texto superior y más fluido que muchos programas de IA actuales (Gritta, 2020; Bisong, 2019). Entre sus ventajas, destaca su reconocimiento de entidades nombradas multilingüe y su eficiencia en comparación con la contratación de humanos con habilidades lingüísticas. Ofrece servicios en 10 idiomas y características como resumen de texto, generación de texto, traducción, detección de idioma, tokenización, lematización y análisis de sentimientos. Sin embargo, necesita una formación significativamente mayor para realizar menos tareas que otras opciones en el mercado y es más costosa, a pesar de ofrecer créditos gratuitos a nuevos clientes (Bisong, 2019).

1.3.2 ChatGPT

ChatGPT, un destacado sistema de procesamiento de lenguaje natural, se convirtió en un referente global a principios de 2023 gracias a su potente motor de análisis de lenguaje, GPT-4, y su capacidad para responder con éxito a demandas específicas de los usuarios (Diego, 2023; Wei, 2023). Además de lanzar su propia API para la creación de chatbots más precisos, esta tecnología ofrece ventajas como la reducción de costos en comparación con la contratación de personal humano y tiempos de respuesta rápidos. La existencia de modelos de aprendizaje automático pre-entrenados también facilita la implementación de aplicaciones de procesamiento de lenguaje natural (Diego, 2023). Sin embargo, entre las desventajas se encuentran el tiempo consumido en el proceso de aprendizaje para autoajustar el modelo ante nuevas entidades nombradas o cambios en las clasificaciones, lo que puede obstaculizar la implementación rápida de soluciones. Por otro lado, no es completamente fiable, ya que a menudo comete errores en la identificación y clasificación de entidades, lo que requiere una gestión cuidadosa y verificación de la información (De Kok, 2023; Lund, 2023).

1.3.3 IBM Watson Discovery

Watson, la plataforma de IA de IBM, posee una herramienta llamada Discovery que representa un gran avance en el PLN, permitiendo la evaluación, interpretación y extracción de información de archivos de texto mediante la creación de etiquetas que disciernen entre tablas, títulos y anotaciones (Chen, 2016; Martin, 2018). A pesar de sus ventajas, como su

capacidad avanzada de procesamiento para textos complejos y su habilidad para analizar una variedad de formatos, su implementación y mantenimiento pueden ser costosos y requerir un alto nivel de conocimiento técnico, además de estar condicionada por la calidad y estructura de los datos de entrada y presentar problemas de privacidad y seguridad al procesar grandes volúmenes de datos (Martin, 2018).

Es crucial reconocer que las tecnologías antes mencionadas son APIs que dependen de la consulta externa de la información, una característica que no se alinea con los requisitos de seguridad y privacidad de la empresa DATYS, dedicada a la creación de soluciones para la Defensa Nacional. Es necesario en este contexto avanzar hacia la independencia tecnológica y es un requerimiento de la empresa usar soluciones fuera de línea. En este sentido, es necesario analizar el estado del arte de soluciones locales (bibliotecas o marcos de trabajos) que se ajusten mejor a las necesidades del objetivo de esta investigación.

1.3.1 Estado del arte de modelos o bibliotecas para el reconocimiento de entidades nombradas

Para este proyecto se hace necesaria una biblioteca o modelo que permita realizar el REN, para esto se estudiaron diversas opciones. En la plataforma de HuggingFace¹ se encuentran diversos modelos los cuales poseen funcionalidades para realizar el REN entre los más empleados en el idioma español se encuentra spaCy, BERT, Flair, NLTK y Stanford NLP que se pueden descargar y utilizar de manera local sin depender de ninguna API.

1.3.1.1 spaCy

SpaCy es una biblioteca de PLN para Python que proporciona una serie de herramientas para el procesamiento de texto, incluyendo tokenización, etiquetado de partes del discurso, reconocimiento de entidades nombradas y más. Es reconocido por su velocidad, eficiencia, y especialmente útil para tareas de PLN que requieren un análisis rápido y preciso del texto. A pesar de esto puede tener dificultades para manejar texto en idiomas distintos al inglés (Partalidou, 2019).

¹ <https://huggingface.co/models>

1.3.1.2 BERT

BERT (Bidirectional Encoder Representations from Transformers) es un modelo de lenguaje preentrenado desarrollado por Google que utiliza la arquitectura Transformer y se enfoca en el reconocimiento de entidades nombradas. BERT fue diseñado para comprender el contexto completo de una entidad nombrada en una frase considerando todas las palabras antes y después de ella, lo que permite generar representaciones de palabras muy ricas y útiles para una variedad de tareas de procesamiento de lenguaje natural (PLN). Este enfoque bidireccional permite que BERT identifique entidades nombradas de manera precisa y efectiva. Sin embargo, la implementación de BERT puede ser difícil y requerir conocimientos técnicos avanzados (Hakala, 2019).

1.3.1.3 Flair

Flair es un marco de trabajo para el PLN que se centra en la simplicidad y la flexibilidad. Proporciona una interfaz fácil de usar para trabajar con diferentes tipos de incrustaciones de palabras, incluyendo BERT y otros modelos Transformer. Permite a los usuarios incorporar fácilmente estas representaciones avanzadas en sus propias aplicaciones de PLN. Además, personaliza el modelo mediante el ajuste fino (fine-tuning) para adaptarlo a tareas específicas. Es más lento que otros modelos de PLN y requiere una gran cantidad de recursos informáticos para su ejecución (Yadav, 2020).

1.3.2 Comparación de los modelos de reconocimiento de entidades

En la presente investigación se selecciona la biblioteca de spaCy debido a su precisión² del 90% en el REN para el idioma español, lo que es relativamente alto en comparación con los otros modelos mencionados ([ver Tabla 1.1](#)). El tamaño del modelo es de 541MB, lo que es menor que el de Flair (1.72GB). Esto puede resultar en tiempos de inferencia más rápidos y un menor consumo de memoria, lo cual es importante para aplicaciones en tiempo real o con recursos limitados. Además, spaCy permite personalizar y entrenar modelos en nuevos dominios o con nuevos tipos de entidades, lo que es muy importante para el reentrenamiento del mismo (Shelar, 2020; Schmitt, 2019).

²

$$Precision_{deteccion} = \frac{C_d + 0.5 * P_d}{C_d + P_d + S_d}$$

Tabla 1.1 Comparación de los modelos de reconocimiento de entidades

Modelo	Conjunto de datos	Tamaño del modelo	Componentes	Precisión
spaCy	OntoNotes 5.0	541MB	tok2vec, morphologizer, parser, senter, attribute_ruler, lemmatizer, ner	90%
BERT	CoNLL-2002	420MB	Transformer architecture, attention mechanism, fine-tuning capability	89.86%
Flair	Conll-03, WikiNER, AIDA-CoNLL-YAGO	1.72GB	Transformer architecture, attention mechanism, fine-tuning capability	86.65%

Estos modelos, a pesar de su precisión a menudo se enfrentan a desafíos significativos en entornos donde el lenguaje está en constante cambio, como es el caso del español. Una de las principales dificultades radica en la aparición de nuevas entidades, ya sea por términos específicos de un dominio, industria, la influencia de eventos actuales o de tendencias emergentes, ejemplo la entidad “**teamacere**” la cual es de tipo organización, que nombró al equipo de pelota cubano en su debut en el V Clásico Mundial de Béisbol. Los modelos preentrenados pueden no estar familiarizados con estas nuevas entidades y, por lo tanto, pueden tener dificultades para reconocerla correctamente. En este entorno es necesario y de suma importancia datos de entrenamiento etiquetados con esta nueva entidad.

Para la generación de estos datos etiquetados se utilizó el modelo de Llama2 debido a que no es una API de pago, es bastante eficiente y rápido en la generación de oraciones. Pese a existir más modelos para la generación de estos datos en el [Anexo II](#) se realizó un estudio comparativo en el que se evidencia el porqué de la elección.

1.4 Herramientas, lenguajes de programación y tecnologías empleadas en el sistema propuesto en la investigación

Partiendo desde el problema de investigación y los objetivos específicos planteados anteriormente se propone la creación de una aplicación web como sistema informático para el REN. Esta aplicación está conformada por un módulo orientado al cliente ([ver Fig 1.2](#)) y

otro módulo de servicios para procesar la lógica del negocio, el REN, la generación de oraciones y el reentrenamiento de los modelos.

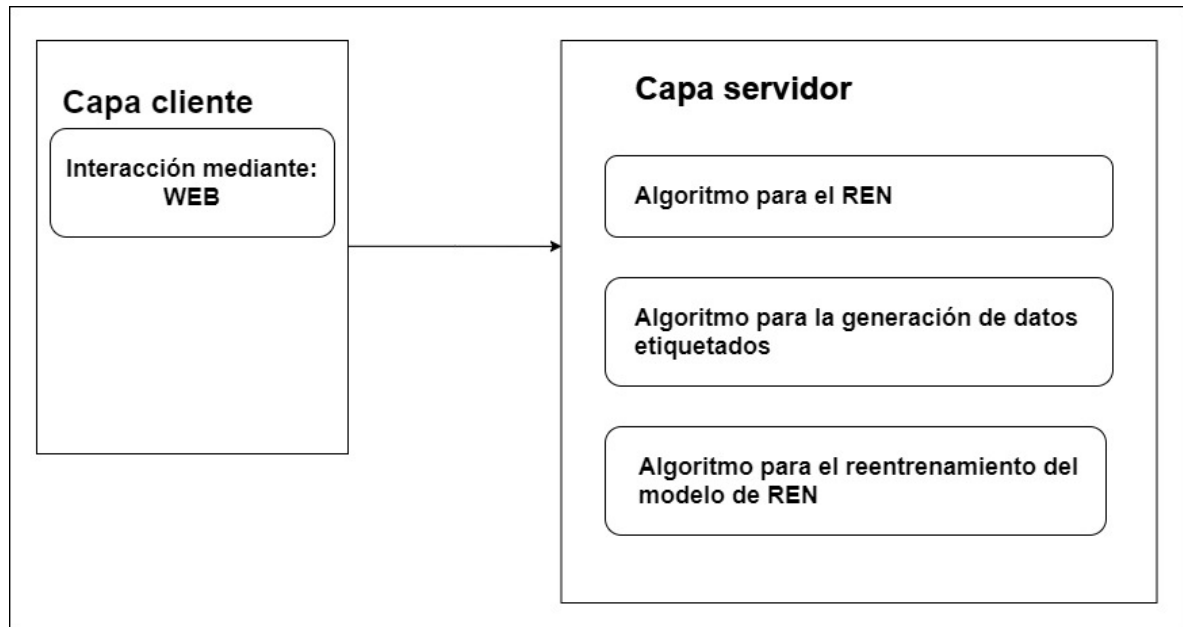


Fig 1.2 Capas del sistema.

Se analiza a continuación las herramientas y tecnologías empleadas en cada módulo de la aplicación.

1.4.1 Marco de trabajo del módulo de la capa del cliente

React 18 es una biblioteca de JavaScript utilizada para construir interfaces de usuario. Permite a los desarrolladores crear aplicaciones web con componentes reutilizables, lo cual facilita el desarrollo evitando la repetición de código. Algunas características clave de React incluyen la renderización del lado del cliente y alto rendimiento gracias al DOM virtual, esto puede hacerse de manera más eficiente, lo que resulta en tiempos de carga más rápidos y una mejor experiencia para el usuario (Chen, 2019; Cincović, 2020).

Angular 17 es un framework (marco de trabajo) de desarrollo de software de código abierto creado por Google, diseñado para construir aplicaciones web de una sola página, tanto para dispositivos móviles como de escritorio. Utiliza TypeScript o JavaScript, ofreciendo herramientas adicionales como tipado estático y decoradores. Su arquitectura se basa en el patrón Modelo-Vista-Controlador, facilitando la organización del código y la colaboración entre equipos de desarrollo (Saks, 2019; Cincović, 2020).

Vue 3 es un framework de JavaScript de código abierto para la construcción de interfaces de usuario y aplicaciones de una sola página. Cuenta con una arquitectura de adaptación gradual que se centra en la representación declarativa y la composición de componentes. Se enfoca en la capa de vista y permite crear aplicaciones web complejas con una sintaxis sencilla y fácil de entender. Utiliza un sistema de plantillas para vincular el estado de los datos de la aplicación con la interfaz de usuario, lo que permite actualizar automáticamente la vista cuando cambian los datos (Saks.2019; Cincović, 2020).

Para este proyecto se selecciona el desarrollo con React 18 por encima de Angular 17 y Vue 3 ya que este se enfoca en la creación de componentes reutilizables, lo que permite construir aplicaciones web complejas y escalables de manera eficiente. Esta característica contrasta con Angular, que tiende a ser más rígido en su sintaxis. Otro aspecto es la capacidad de React para renderizar eficientemente el DOM virtual, lo que se traduce en un mejor rendimiento y una experiencia de usuario más fluida. En comparación, Angular y Vue pueden presentar dificultades para manejar cambios frecuentes en la interfaz de usuario, afectando el rendimiento de la aplicación. React cuenta con una gran comunidad de desarrolladores y empresas importantes como Facebook y Airbnb que respaldan su uso y contribuyen a su crecimiento.

1.4.2 Marco de trabajo del módulo de la capa del servidor

Python 3.10.8 es un lenguaje de programación de alto nivel, ampliamente utilizado en aplicaciones web, desarrollo de software, ciencia de datos, y aprendizaje automático. Su sintaxis sencilla y cercana al lenguaje humano lo hace fácil de aprender y usar. Es de código abierto, lo que permite su uso y distribución gratuitos, incluso para fines comerciales. Cuenta con una extensa comunidad y una gran cantidad de módulos y bibliotecas, lo que facilita la ampliación de sus capacidades (Challenger-Pérez, 2014; Mirjalili, 2020). Para este proyecto se empleó debido a que posee bibliotecas para su uso con spaCy, Elasticsearch y Llama2.

FastAPI versión 0.100.1 es un marco moderno y de alto rendimiento para construir APIs web con Python basado en estándares abiertos. Fue diseñado para ser rápido tanto en términos de velocidad de ejecución como de codificación, facilita el aprendizaje y reduce errores humanos (Lathklar, 2023).

Si bien existen otros marcos de trabajo como Flask y Django, FastAPI demuestra su velocidad en el manejo de la concurrencia, ligereza y escasa latencia al realizar consultas. Por lo tanto, se empleó en el sistema como marco principal para la construcción de la API web.

1.4.3 Herramientas utilizadas para el desarrollo del sistema

Visual Studio Code (VS Code) versión 1.82 es un editor de código fuente desarrollado por Microsoft, disponible para Windows, macOS y Linux de forma gratuita. Combina la simplicidad de un editor con herramientas poderosas como la finalización de código IntelliSense y la depuración (Microsoft, 2024). Se utilizó en el desarrollo del sistema ya que en el mismo editor se pudo programar para los diferentes lenguajes utilizando los diferentes componentes (plugins: término en inglés) de la herramienta.

Git versión 2.44.0 es un sistema de control de versiones distribuido que se utiliza para rastrear cambios en archivos y coordinar el trabajo entre programadores. Es el sistema de control de versiones más popular en el mundo y se utiliza para una amplia gama de proyectos, desde software hasta documentación. El mismo permite a los usuarios realizar un seguimiento de los cambios en los archivos de un proyecto, lo que facilita la colaboración y la resolución de conflictos. Además, puede revertir a una versión anterior de un proyecto si se produce un error o un bug. Esto es especialmente útil en un entorno de desarrollo de software, donde los errores son comunes y es necesario poder volver atrás rápidamente (Blischak, 2016; Loeliger, 2012). En el proyecto se creó un repositorio de Git que se empleó para subir las diferentes versiones del sistema, así como las de la investigación.

Ollama en su versión 0.1.29: es una herramienta disponible para macOS, Windows y Linux. Esta permite la utilización de modelos LLM de forma local en el ordenador, solo se necesita especificar mediante comandos el modelo o los modelos que se va a descargar, también permite hacer una interacción con el modelo mediante la terminal de comandos e incluso mediante bibliotecas de lenguajes como Python (Gruber, 2024). Para este proyecto se utilizó Ollama para descargar y utilizar el modelo de Llama2 para la generación de oraciones, las mismas servirán para la creación de datos etiquetados para reentrenar el modelo de REN.

JSON

JSON (JavaScript Object Notation) es un formato de texto ligero utilizado para el intercambio de datos, derivado de la sintaxis de los objetos de JavaScript. Es una

alternativa a XML, con una sintaxis más simple y legible tanto para humanos como para máquinas (Ihrig, 2013). Es útil para intercambiar datos entre diferentes dispositivos digitales y se emplean ampliamente en la web, aplicaciones móviles, programas computacionales y transferencia de documentos (Nolan, 2014). En el sistema se empleó para el envío de información a la base de datos ya que la misma está orientada a documentos JSON.

1.4.4 Base de datos utilizada

Elasticsearch versión 8.3.3 es una base de datos con un motor de búsqueda y análisis distribuido construido sobre Apache Lucene. Utiliza APIs basadas en REST (acrónimo en inglés de Transferencia de Estado Representacional) y documentos JSON sin esquema, lo que facilita la construcción rápida de aplicaciones para diversos casos de uso, y sus operaciones suelen completarse en menos de un segundo. Es altamente escalable horizontalmente y sus índices albergan colecciones de documentos que pueden dividirse en fragmentos para una distribución eficiente en entornos distribuidos (Dhulavvagol, 2020; Shaik, 2017; Thacker, 2018).

Si bien existen otras bases de datos no relacionales en el proyecto se utilizó Elasticsearch debido a que es uno de los requerimientos de la empresa DATYS, además esta posee capacidades de búsqueda superiores a las demás bases de datos no relacionales. Está diseñada para manejar grandes volúmenes de datos y su arquitectura distribuida permite una escalabilidad horizontal. Toda la información que se va a utilizar en el sistema desde los usuarios, las trazas, los datos a los que se les va a realizar el REN, se encuentran almacenados en esta base de datos.

1.5 Metodología XP

La metodología de software XP se centra en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo (Gonzaga, 2023). XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP es adecuada para proyectos con requisitos imprecisos y donde existe un alto riesgo técnico (Bautista-Villegas, 2022). En el proyecto se utilizó como enfoque principal para la gestión y ejecución del mismo.

1.6 Conclusiones del capítulo

En el área del REN existen actualmente diversos sistemas que se utilizan en el ámbito comercial y científico. Los mismos muestran una baja precisión para los textos en español. Se evidenció que el modelo para el REN del marco de trabajo de spaCy es una alternativa mucho más eficiente que el resto de los modelos, aun así, tiene la limitación de que es incapaz de reconocer nuevas entidades. Es un reto lograr generar datos de entrenamiento etiquetados para el posterior reentrenamiento, donde el modelo de Llama2 es seleccionado debido a sus ventajas tanto de velocidad, como de eficiencia con respecto a otros modelos existentes. Para el desarrollo del sistema, se empleó el lenguaje de programación Python debido a las bibliotecas que el mismo posee para el PLN y para el uso de Elasticsearch. Para la interfaz de usuario, se utilizó React por ser más eficiente en cuanto a escalabilidad y componentes reutilizables. Como herramientas se usaron el controlador de versiones Git, para administrar los cambios en la investigación, el editor de código Visual Studio Code porque permite en un solo entorno trabajar en la construcción del servidor y la aplicación web y Ollama para descargar y utilizar el modelo generador de oraciones Llama2. También se utilizó FastAPI para la construcción de los servicios web y la base de datos de Elasticsearch como requisito de la empresa DATYS. Como metodología de desarrollo de software se usó XP ya que permitió un enfoque ágil y flexible en el desarrollo de software, fomentando la comunicación constante con la empresa DATYS y la entrega de soluciones simples. Todo ello contribuyó al desarrollo y culminación del sistema propuesto.

CAPITULO 2 . ORGANIZACIÓN Y DISEÑO

En el siguiente capítulo, se describe el proceso de organización y diseño del sistema, aplicando diversas técnicas y herramientas, para la empresa DATYS en su modelo de negocio. Se muestran los diferentes diagramas utilizados para la modelación y desarrollo del sistema de REN para el idioma español en grandes volúmenes de datos, actuando como una guía para futuros proyectos similares. Aunque la metodología ágil utilizada es XP, se utilizan algunos artefactos de RUP (Proceso Unificado Racional en español) para una mejor documentación, como es el caso de los requisitos no funcionales, el diagrama de secuencia y el de actividades.

2.1 Modelo de negocio para el REN

El modelo de negocio es una disciplina que se enfoca en comprender los procesos, estructuras y dinámicas de una organización para el desarrollo de un software. Su objetivo principal es establecer una ventaja competitiva a través del análisis y modelado de los procesos, identificando los problemas actuales, los requerimientos de la empresa para resolver esta tarea, y la propuesta de valor (Aros, 2008; Paredes, 2019).

En el caso de la empresa de desarrollo de software DATYS con sede en Santiago de Cuba, el proceso de REN comienza recibiendo un flujo de información de varios centros del país. Esta información es procesada y almacenada en Elasticsearch, donde el sistema propuesto realizaría el REN, la generación de datos de entrenamiento en caso de ser necesario y el reentrenamiento del modelo tal como se muestra en la [fig 2.1](#).

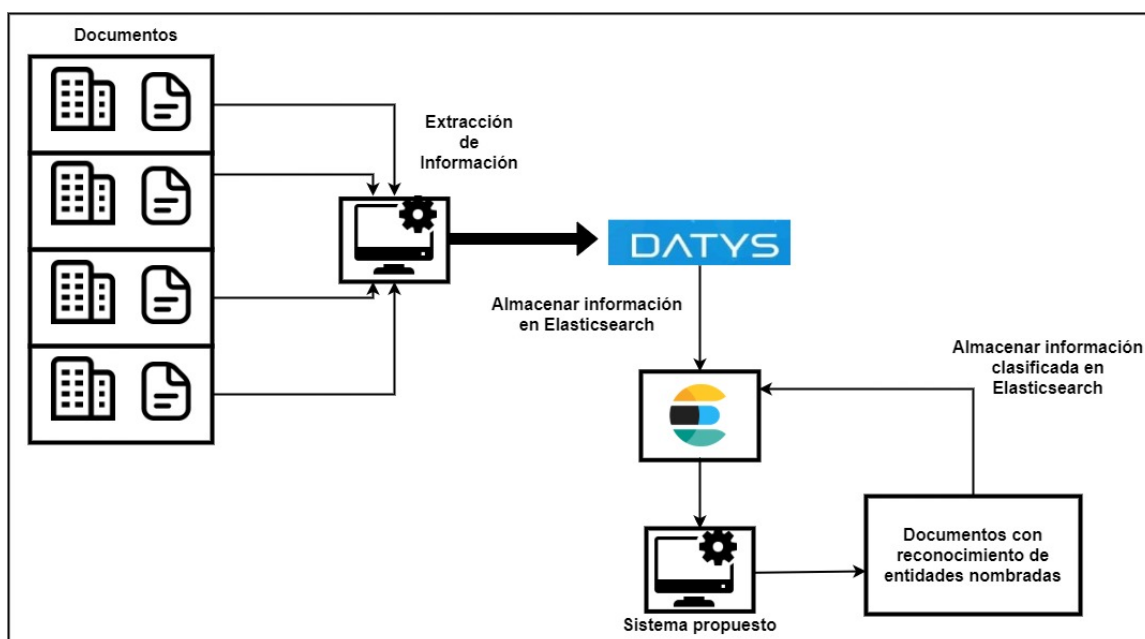


Fig 2.1 Diagrama del modelo de negocio de la empresa DATYS en el reconocimiento de entidades nombradas

El sistema además debe cumplir con las siguientes reglas basadas en los requerimientos del negocio:

- La extracción de información se realizará una vez que los datos hayan sido almacenados en Elasticsearch.
- Estos datos no deben ser modificados.
- Se utilizarán modelos computacionales locales y no se conectarán a fuentes externas.
- El sistema debe reconocer y clasificar las entidades nombradas, así como la posición en el texto donde aparecen.
- El procesamiento de las entidades se realizará sobre el texto original que ha sido almacenado en Elasticsearch.

La propuesta de valor del sistema se enfoca en el procesamiento seguro y eficiente de grandes volúmenes de datos almacenados en Elasticsearch, utilizando modelos computacionales locales para garantizar la privacidad y seguridad de la información. Además, el sistema tendrá la capacidad de reentrenamiento del modelo en caso de no reconocer una entidad.

2.2 Arquitectura del sistema

La arquitectura del sistema propuesto es de tipo cliente-servidor. En esta, una o más computadoras, denominadas servidores, brindan servicios y recursos a otras computadoras, denominadas clientes, a través de una red, ya sea local o remota (Cuyutupa Calderón, 2023; Lituma-Sarmiento, 2023). La elección de esta arquitectura para el desarrollo del sistema permite centralizar la lógica de negocio, facilitando así la administración, el mantenimiento y la escalabilidad.

El diseño del sistema se divide en tres capas, donde cada capa tiene responsabilidades específicas. La capa de presentación es la aplicación web con la que el cliente interactúa, mostrando los resultados y permitiendo la interacción del usuario. La capa de lógica de negocio es responsable de procesar los datos, realizar operaciones y ofrecer servicios, generando las respuestas que se enviarán a la capa de presentación. Por último, la capa de acceso a datos representa la fuente de información que el sistema utiliza para almacenar los datos, en este caso la base de datos de Elasticsearch de la empresa DATYS tal como se muestra en la [Fig 2.2](#).

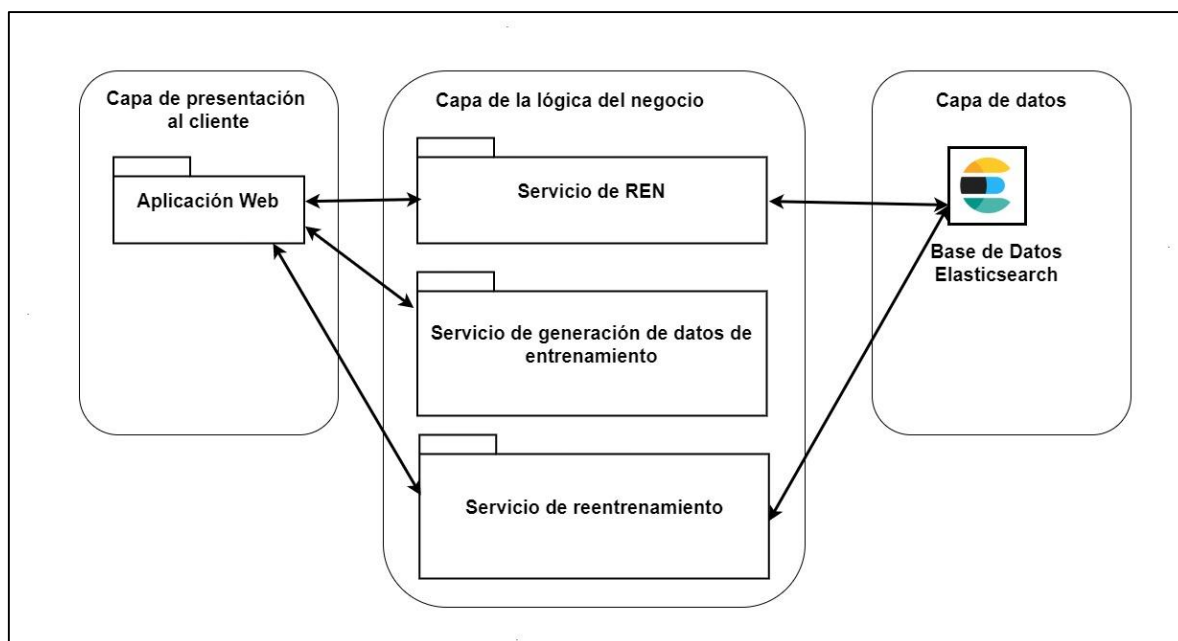


Fig 2.2 Diagrama de arquitectura cliente-servidor de tres capas

2.3 Flujo del sistema

Con el objetivo de dar solución al problema investigativo, se propone el diagrama que aparece en la figura: [\(Fig 2.3\)](#). El primer paso consiste en que el usuario introduzca los

datos en Elasticsearch. Posteriormente, en el segundo paso, el usuario accede al sistema propuesto, el cual proporciona acceso a todos los índices disponibles. Tras seleccionar un índice, el sistema mostrará las entidades identificadas. Si todo es correcto y las entidades se han extraído y clasificado adecuadamente, se procederá al tercer paso, donde el usuario almacenará los resultados en la base de datos. En caso de que el sistema no reconozca una entidad, se procederá al cuarto paso, en el cual se podrá reentrenar el modelo especificando la entidad, su tipo y una descripción. El sistema mostrará los datos de entrenamiento generados, y en el quinto paso, el usuario reentrenará el modelo para reconocer la entidad proporcionada.

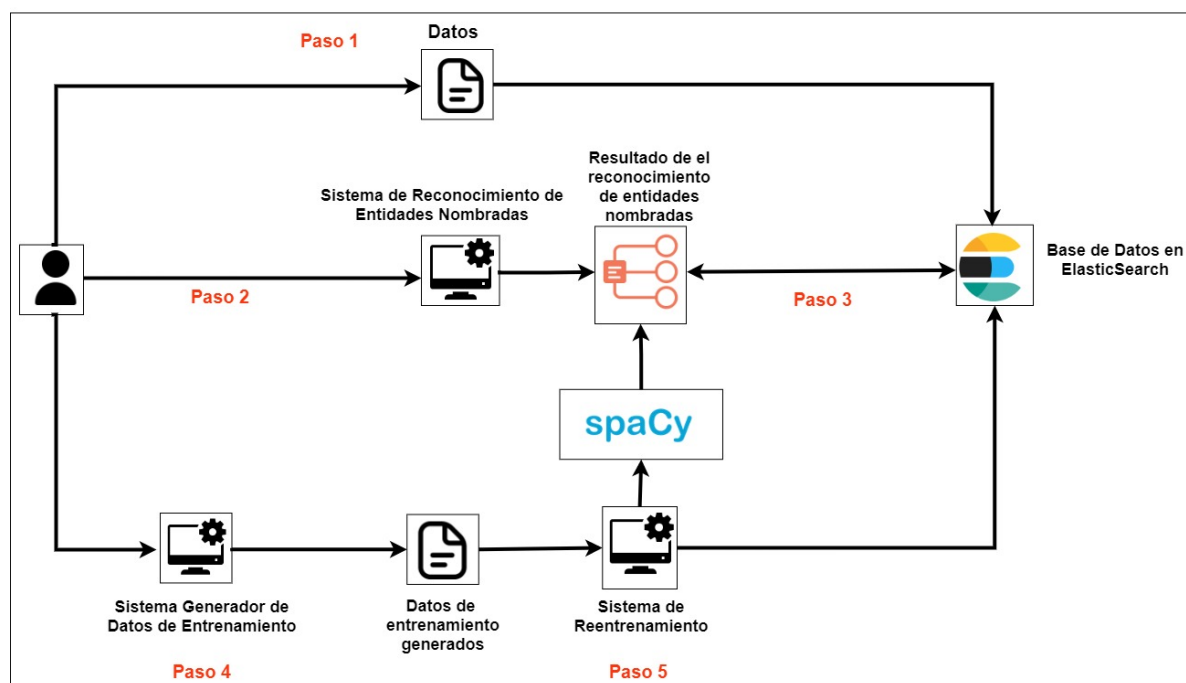


Fig 2.3: Diagrama de flujo del sistema.

2.4 Requerimientos para el despliegue del sistema

Los requisitos no funcionales se refieren a las restricciones y condiciones que el sistema debe cumplir, pero que no están directamente relacionadas con las funcionalidades específicas que el sistema debe proporcionar (Gómez, 2010). Estos requisitos son críticos para el éxito del proyecto, ya que afectan la calidad, el rendimiento, la seguridad, la usabilidad, entre otros aspectos del sistema. Aunque los documentos de RUP no se centran específicamente en los requisitos no funcionales, la metodología en sí misma promueve la consideración de estos aspectos a través de su enfoque en la arquitectura del sistema, la calidad del software, y la gestión de riesgos (Jain, 2024; Molina 2019).

Requisitos de Software

- Sistema operativo Windows 10 o 11
- Node versión 20.11.0 o superiores
- Python versión 3.10.8
- Elasticsearch versión 8.3.3
- spaCy versión 3.6.1

Requisitos de Hardware

- Procesador Intel Core i5 12ma generación o superiores
- RAM: 16GB de 3200MHz o superiores
- Almacenamiento: buen espacio de almacenamiento

Se sugiere realizar pruebas en distribuciones para el sistema operativo Linux. La implementación de versiones de Elasticsearch más actualizadas, así como la del modelo de spaCy. Además, este sistema se probó con 16GB de RAM, pero para mejorar el rendimiento debido a que la ejecución de algoritmos de PLN puede requerir una capacidad de cómputo considerable se sugiere la utilización de 32 GB de RAM y una GPU NVIDIA GeForce GTX 1650 para un mejor desempeño.

2.5 Diseño de la base de datos

El modelo entidad-relación es una herramienta que permite representar de manera simplificada los componentes que participan en un proceso de negocio y el modo en el que estos se relacionan entre sí. Se utiliza para exponer cómo se organiza la información en una base de datos (Molina, 2019; Pantaleo, 2015). Pese a ser Elasticsearch una base de datos NoSQL, es una buena práctica la creación del modelo de entidad relación el cual refleje las tablas y campos que se van a manejar por el sistema en la base de datos.

Los índices que se representan son [\(ver Fig 2.4\)](#):

- **Usuarios:** Esta representa los usuarios del sistema. Cada usuario tiene un nombre de usuario único (username), un rol y una contraseña (password). El rol determina los permisos que tiene el usuario.
- **Índice:** Este índice contiene campos de texto que se utilizan para realizar el reconocimiento de entidades.
- **Entidades:** Esta representa el proceso de reconocimiento de entidades que se realiza en los campos de texto. Después del reconocimiento, los datos se

almacenan nuevamente en el mismo índice, pero con información adicional sobre las entidades encontradas.

- **Traza:** Esta representa las entradas al sistema del usuario, los índices salvados por él al realizar el reconocimiento de entidades, los reentrenamientos al modelo y el envío de información a especialistas.
- **Modelo:** Esta representa la información cada vez que un usuario realiza un reentrenamiento al modelo de spaCy en el cual guarda los campos de la fecha en la que se realizó en reentrenamiento, la precisión resultante del modelo, y el usuario que realizó el reentrenamiento.
- **Medidas de desempeño:** Esta representa la información obtenida al realizar el reentrenamiento del modelo.

Las relaciones entre las tablas son:

- Un usuario puede estar asociado con varios documentos en el índice de texto. Esta relación es de uno a muchos, ya que un usuario puede haber creado o interactuado con múltiples documentos.
- El reconocimiento de entidades se aplica a los textos en el índice de texto. Esto es una acción o proceso que se realiza en los datos almacenados en el índice de texto. Por lo que la relación es de mucho a mucho. Un índice puede tener varias entidades nombradas, y una entidad nombrada puede estar presente en varios índices.
- Un usuario puede hacer varios reentrenamientos a un modelo mientras que un modelo es reentrenado por varios usuarios.
- Un usuario puede tener varias trazas mientras que una traza pertenece a un usuario por lo que la relación es de uno a muchos.
- Un modelo puede tener varias medidas de desempeño ya que puede ser reentrenado varias veces, pero una medida de desempeño nada más puede ser de un modelo.

El diseño de la base de datos en Elasticsearch no se normaliza ni se representa mediante diagramas “Entidad Relación” debido a las diferencias fundamentales en la forma en que estas bases de datos manejan los datos y las relaciones entre ellos. En lugar de esquemas fijos y relaciones predefinidas, Elasticsearch utiliza un enfoque más flexible y dinámico, permitiendo una mayor adaptabilidad a los cambios en los requisitos de datos y las estructuras de estos.

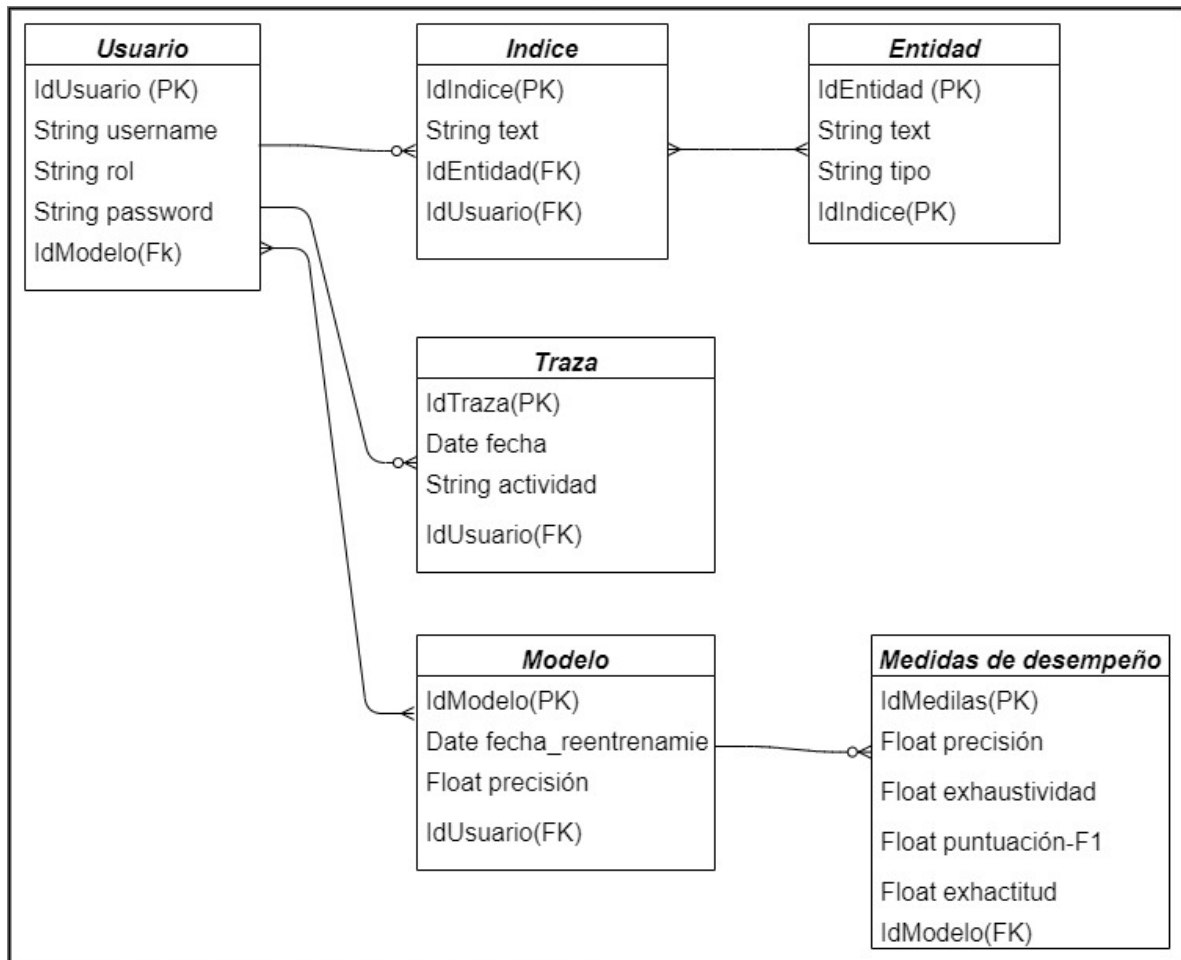


Fig 2.4 Modelo Entidad Relación de la base de datos.

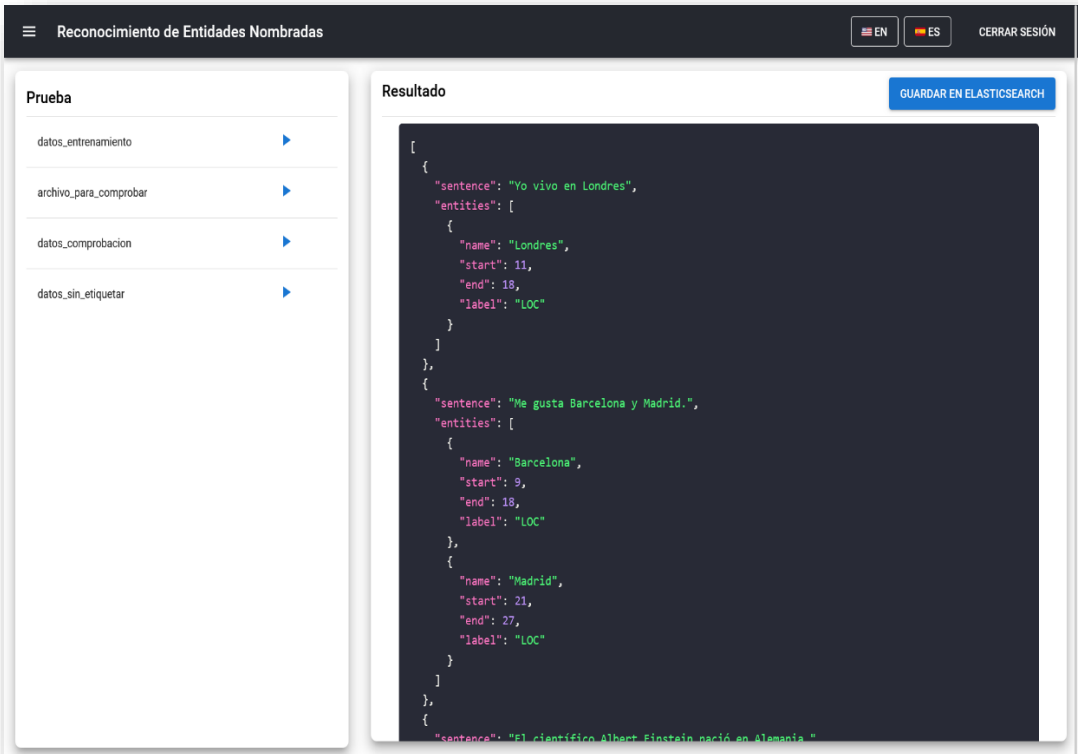
2.6 Historias de Usuario

Las historias de usuario (HU) son una herramienta esencial en el desarrollo ágil de software, que permite a los equipos centrarse en las necesidades y deseos de los usuarios finales. Estas historias describen de manera concisa y en lenguaje natural lo que un usuario quiere lograr con un producto o servicio, y por qué es importante para él (Menzinsky, 2018). Las HU del sistema son:

- HU.1: Autenticar usuario en el sistema.
- HU.2: Iniciar sistema con datos de prueba.
- HU.3: Reconocer entidades nombradas en los índices de Elasticsearch.
- HU.4: Reentrenar modelo.
- HU.5: Administrador de usuarios.
- HU .6: Soporte en español e inglés.
- HU .7: Reentrenar por especialista.

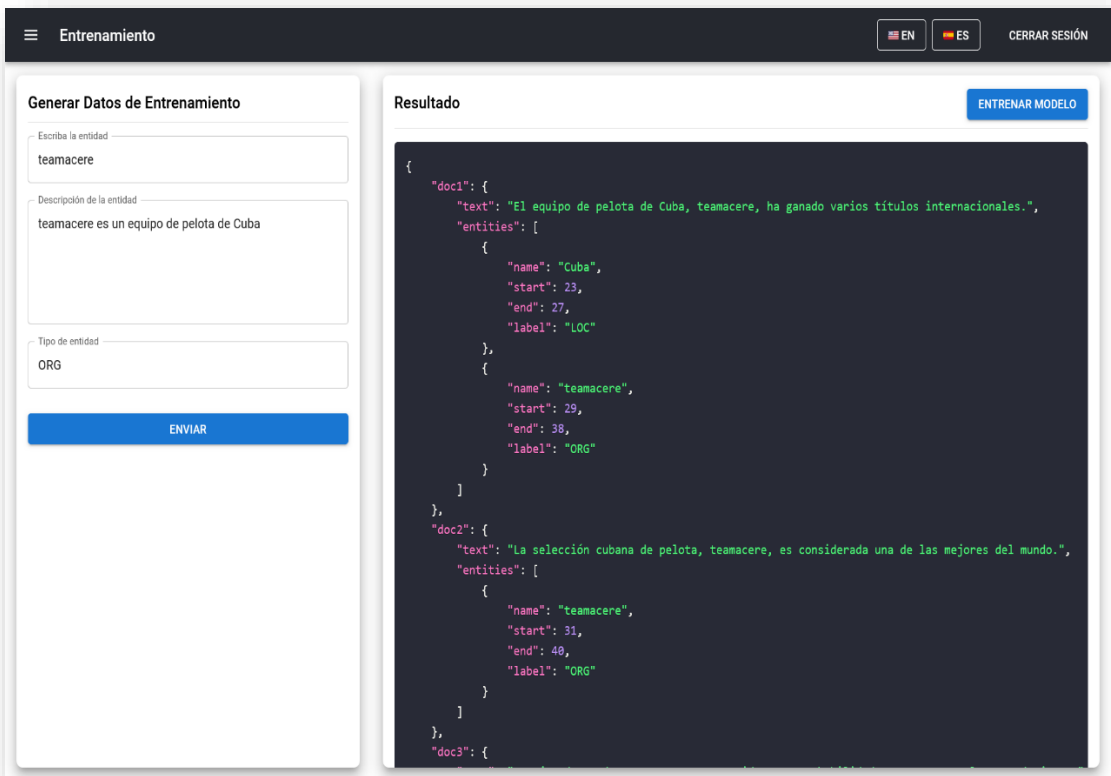
Dentro de estas HU las más importante son: reconocer entidades nombradas en los índices de Elasticsearch, reentrenar el modelo y reentrenar por especialista, las cuales se explican a continuación.

Tabla 2.1 Historia de Usuario: Reconocer entidades nombradas en los índices de Elasticsearch

Historia de usuario	
Número: 3	Nombre de Historia de Usuario: Reconocer entidades nombradas en los índices de Elasticsearch
Usuario: Los trabajadores del centro	
Prioridad en negocio: Alta	Riesgo de desarrollo: Alto
Interacciones del usuario: <ul style="list-style-type: none"> ➤ Acceder a la página de reconocimiento de entidades nombradas ➤ Mostrar los índices disponibles para el reconocimiento de entidades nombradas ➤ Seleccionar índice para el reconocimiento ➤ Ver resultados obtenidos ➤ Almacenar los resultados en Elasticsearch 	
	

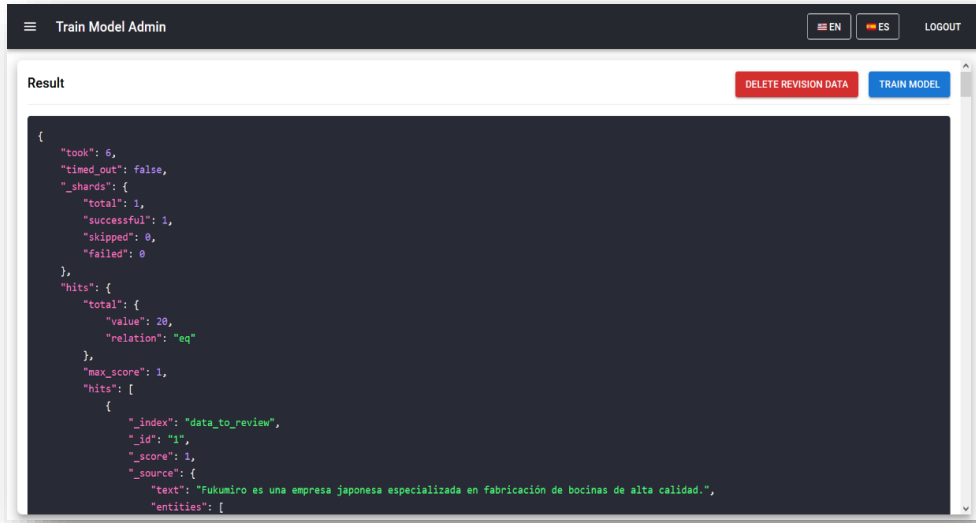
Los usuarios serán capaces de seleccionar el índice al cual se le desea hacer un reconocimiento de entidades y en caso de estar correcto almacenarlo en la base de datos de Elasticsearch ([ver Tabla 2.1](#)).

Tabla 2.2 Historia de Usuario: Reentrenar el modelo

Historia de usuario	
Número: 4	Nombre de Historia de Usuario: Reentrenar modelo
Usuario: Usuarios del sistema	
Prioridad en negocio: Alta	Riesgo de desarrollo: Alto
Interacciones del usuario: <ul style="list-style-type: none"> ➤ Entrada a la página de entrenamiento y generación de datos ➤ Llenar formulario con el nombre de la entidad a reconocer, una descripción y el tipo de entidad ➤ Enviar formulario para la generación de datos de entrenamiento ➤ Visualizar los datos generados ➤ Reentrenar el modelo o enviar los datos para que lo reentrene un especialista 	
	

Los usuarios serán capaces de reentrenar el modelo pasando los parámetros solicitados para la generación de dichos datos de entrenamiento. Al generar los datos para el reentrenamiento los usuarios lo evaluarán para el reentrenamiento del modelo. Este modelo se guardará y se notificará al usuario que se guardó solo si no se pierde conocimiento ([ver Tabla 2.2](#)).

Tabla 2.3: Historia de usuario: Reentrenar por especialista

Historia de usuario	
Número: 7	Nombre de Historia de Usuario: Reentrenar por especialista
Usuario: Administradores del centro	
Prioridad en negocio: Alta	Riesgo de desarrollo: Bajo
Interacciones del usuario: <ul style="list-style-type: none"> ➤ Entrada al apartado de entrenamiento por el administrador ➤ Revisar los datos de reentrenamientos enviados ➤ Reentrenar el modelo ➤ Eliminar los datos enviados 	
 <p>The screenshot shows a web interface titled 'Train Model Admin' with a dark theme. At the top right, there are language toggles for 'EN' and 'ES', and a 'LOGOUT' button. Below the header, there's a 'Result' section with a red 'DELETE REVISION DATA' button and a blue 'TRAIN MODEL' button. The main content area displays a JSON object representing the training result. The JSON includes fields for 'took' (6), 'timed_out' (false), '_shards' (total: 1, successful: 1, skipped: 0, failed: 0), 'hits' (total: 20, max_score: 1), and a list of hits. One hit is shown with '_index': 'data_to_review', '_id': '1', '_score': 1, and '_source' containing a text snippet about a Japanese company and its products.</p>	

Los usuarios que no sepan reentrenar el modelo pueden almacenar los datos de entrenamiento generados y enviarlos a un especialista para que realice el reentrenamiento ([ver Tabla 2.3](#)).

2.7 Usuarios del Sistema

La entrada de usuarios al sistema está dividida en dos roles: usuario y administradores los cuales tienen tareas específicas que en dependencia de su rol podrán realizar en el sistema ([ver Tabla 2.4](#))

Tabla 2.4 Usuarios del sistema

Actor	Tareas
Usuario	Visualizar, realizar reconocimiento de entidades, agregar a base de datos
Administrador	Administra la base de datos del sistema, visualizar, realiza reconocimiento de entidades, agrega a la base de datos, reentrena el modelo de spaCy

2.8 Diagrama de Secuencia de las interacciones de los objetos del sistema a lo largo del tiempo

Un diagrama de secuencia en la ingeniería de software es una representación gráfica que muestra la interacción de objetos en un sistema a lo largo del tiempo. Estos capturan la secuencia de mensajes intercambiados entre objetos y el orden en que ocurren estas interacciones, presentándolos como líneas de vida verticales y flechas horizontales. Los diagramas de secuencia son una herramienta esencial en el Lenguaje Unificado de Modelado (UML) para describir cómo y en qué orden un grupo de objetos funcionan en conjunto, facilitando la comprensión de los requisitos de un sistema nuevo o documentando un proceso existente (Vidal, 2012).

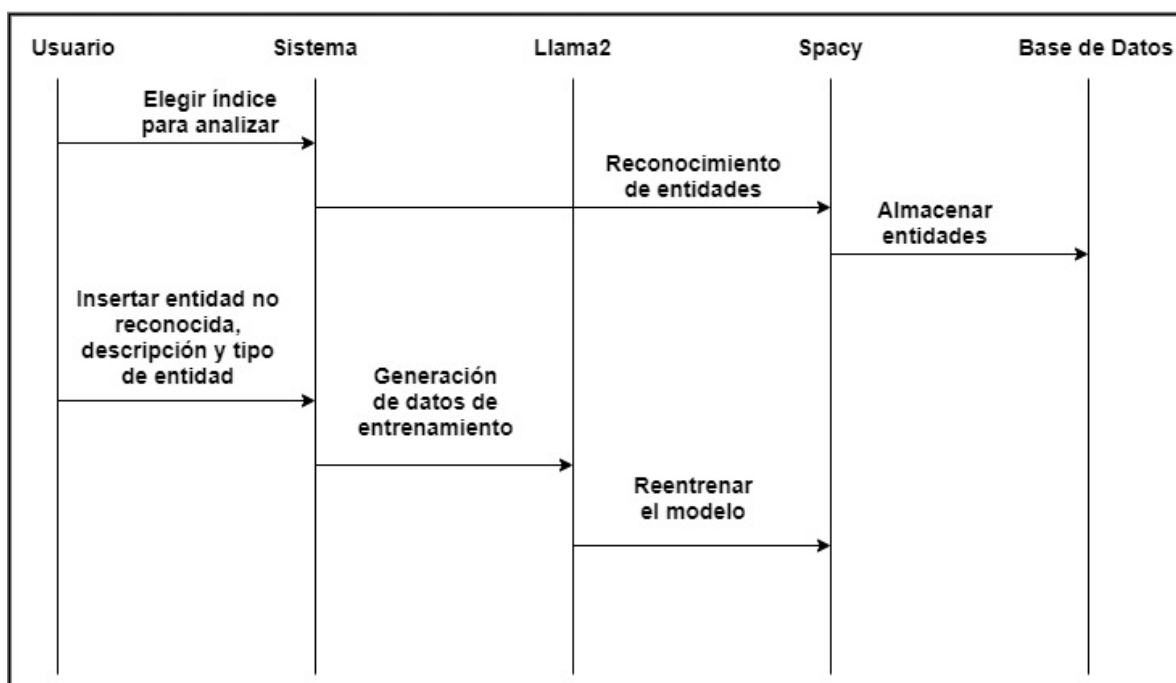


Fig 2.5 Diagrama de secuencia de las interacciones de los objetos a lo largo del tiempo

El diagrama de secuencia de la [Fig 2.5](#) muestra la interacción entre cuatro objetos:

- **Usuario:** El usuario que interactúa con el sistema.
- **Sistema:** El sistema que proporciona el servicio al usuario.
- **Llama2:** Un modelo de lenguaje que genera oraciones.
- **spaCy:** Un modelo de lenguaje que realiza el reconocimiento de entidades.
- **Base de datos:** Elasticsearch sería la base de datos de nuestro sistema donde se almacenaría los índices y los resultados obtenidos.

En la [Fig 2.5](#) se muestra cómo estos objetos interactúan para generar oraciones a partir de un índice proporcionado por el usuario. La interacción funciona de la siguiente manera:

- El usuario elige un índice.
- El sistema envía el índice a spaCy para el reconocimiento de entidades nombradas.
- Se almacenan los resultados de las entidades encontradas en la base de datos.
- En caso de no reconocer una entidad el usuario proporciona al sistema los datos de la misma.
- El modelo de Llama 2 genera los datos de entrenamiento.
- Reentrenamos el modelo de spaCy para el reconocimiento de la entidad proporcionado por el usuario.

Este diagrama cumple un papel crucial en el proceso de diseño y comprensión del sistema. Proporciona una representación clara de cómo los diferentes objetos dentro del sistema interactúan entre sí a lo largo del tiempo, mostrando la secuencia de intercambio y el orden en que estas ocurren.

2.9 Diagrama de Actividades del flujo del sistema

Un diagrama de actividades en la ingeniería de software es una representación visual que muestra el flujo de actividades y procesos dentro de un sistema o proceso de negocio. Este tipo de diagrama es útil para demostrar la lógica de un algoritmo, describir los pasos realizados en un caso de uso UML, ilustrar un proceso de negocios o flujo de trabajo entre los usuarios y el sistema, y simplificar y mejorar cualquier proceso clarificando casos de uso complicados. Además, permite modelar elementos de arquitectura de software, como métodos, funciones y operaciones (García-Holgado, 2020; Marcillo Ligua, 2021).

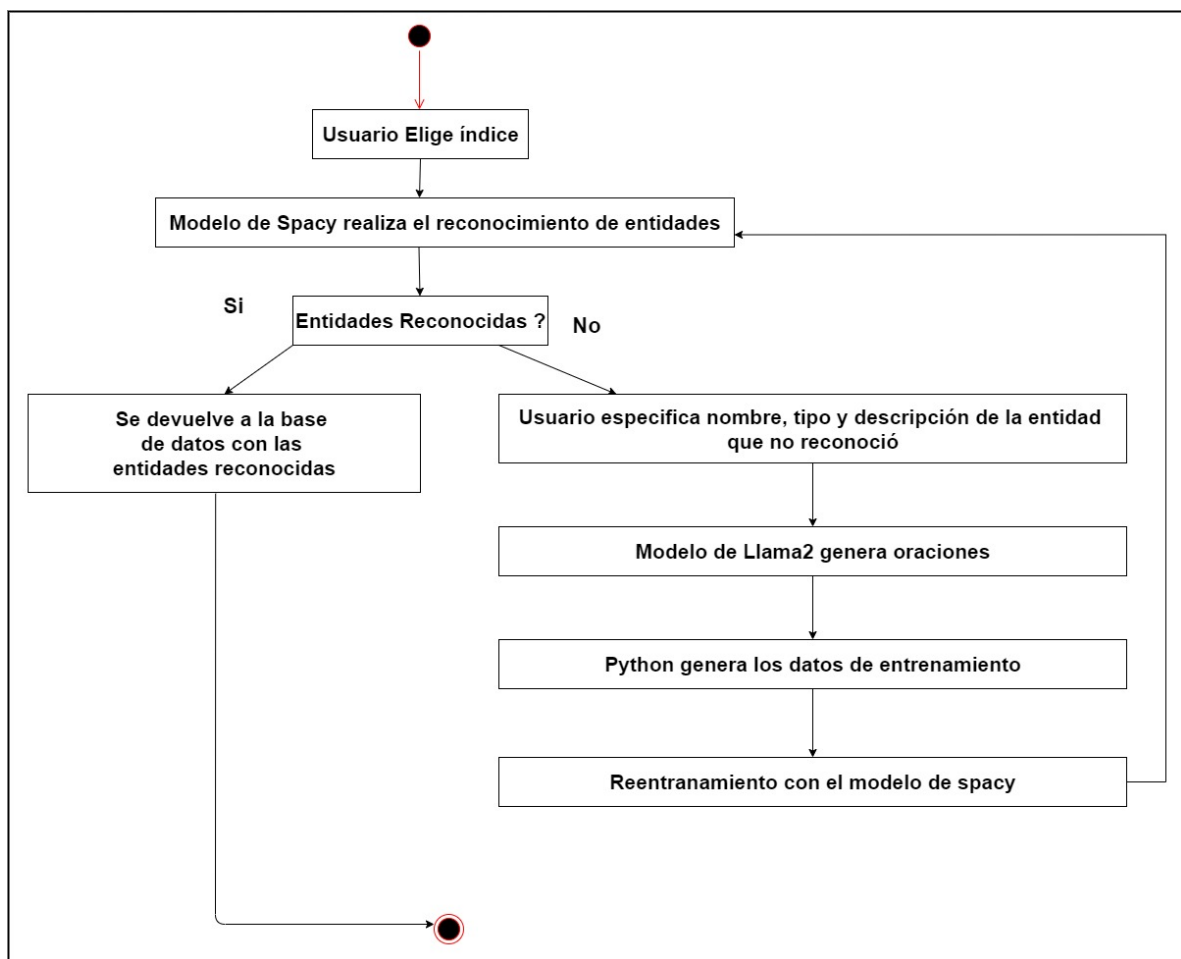


Fig 2.6 Diagrama de Actividades del flujo del sistema

El diagrama de la [figura 2.6](#) muestra el sistema que utiliza un modelo para generar oraciones y otro para el reconocimiento de entidades nombradas representándolo de la siguiente forma:

- El usuario elige un índice.
- El modelo de spaCy realiza el reconocimiento de entidades en el índice.
- Si se encuentran entidades, se devuelven a la base de datos con las entidades reconocidas.
- Si no se encuentran entidades, el usuario especifica el nombre, el tipo y la descripción de la entidad que no se reconoció.
- El modelo de Llama 2 genera oraciones a partir del índice y las entidades reconocidas.
- Se generan los datos de entrenamiento para el modelo de spaCy.
- El modelo de spaCy se reentrena con los nuevos datos de entrenamiento.

Este diagrama es especialmente útil ya que muestra las actividades relacionadas con el sistema y los usuarios. Permite modelar los elementos de la arquitectura de software, como métodos, funciones y operaciones. Al comprender claramente cómo se realizan las actividades del sistema, se pudo identificar de manera más eficiente los problemas, optimizar los procesos y tomar decisiones informadas para mejorar la calidad y eficiencia del sistema en general.

2.10 Conclusiones del capítulo

En este capítulo, se ha delineado el proceso de organización y diseño del sistema para la empresa de desarrollo de software DATYS. Se centró en una arquitectura cliente-servidor de tres capas, con Elasticsearch como base de datos y una interfaz de usuario en React. Los diagramas utilizados en el proceso de diseño resultaron útiles para representar la interacción entre los diferentes componentes del sistema, mientras que la definición de los roles y las historias de usuario permitieron una mejor comprensión de las necesidades. La consideración de los requisitos no funcionales y las recomendaciones para mejorar el rendimiento fueron aspectos importantes del proceso de diseño. Este enfoque detallado sienta las bases para el desarrollo eficiente del sistema, asegurando su adaptabilidad y cumplimiento de los objetivos planteados.

CAPITULO 3 . IMPLEMENTACIÓN Y PRUEBA

En este capítulo se explorarán las funcionalidades clave del sistema diseñado, así como los detalles de su implementación y las pruebas realizadas. Se discuten aspectos específicos de la implementación, los componentes utilizados y las consideraciones relevantes que surgieron durante el desarrollo. Además, se presentan los resultados obtenidos de los casos de prueba realizados una vez finalizado el proyecto.

3.1 Instalación de los requisitos para el funcionamiento del sistema

En el repositorio de github donde se encuentra el código fuente del proyecto³ se explica de forma detalla los pasos y procedimientos para la instalación y despliegue del sistema.

3.2 Patrones de diseño empleados

Los patrones de diseño son soluciones optimizadas y reutilizables para problemas comunes en el desarrollo de software. No son código en sí mismos, sino descripciones de cómo abordar y diseñar soluciones a estos problemas. Estos patrones actúan como plantillas que pueden ser implementadas en diferentes contextos y lenguajes de programación, facilitando la comunicación y la eficiencia en el diseño de software (Blas,2019; Alvarez, 2022).

En este proyecto se utilizaron varios patrones de diseño entre los que se encuentran: el principio de responsabilidad única, el principio abierto/cerrado, y el principio de inversión de dependencias.

El principio de responsabilidad única establece que cada función debe de tener una única responsabilidad o propósito. Lo que ayuda a evitar el acoplamiento innecesario y reduce la complejidad en cada cambio. Unos ejemplos de esto se encuentran en las funciones de *prepare_train_data* para la preparación de datos de entrenamiento para el modelo de REN, *configure_ner* esta función lo que hace es desactivar las demás funcionalidades del modelo y deja solo la del REN.

El principio de abierto/cerrado sostiene que las entidades de software deben de estar abiertas para su extensión y cerradas para su modificación. Esto significa que las clases y funciones se deben de diseñar para que sus funcionalidades principales puedan extenderse a otras entidades sin alterar el código fuente de la inicial. Promoviendo así la flexibilidad y extensibilidad del código. Ejemplo de su uso se encuentra en el APIRouter de FastAPI y la definición de las rutas ya de manera modular y extensible, lo que facilita la adición de nuevas rutas o la modificación de las existentes sin alterar el código central de la aplicación.

³ <https://github.com/luislicea1/NER-with-spacy-elasticsearch-and-Llama2>

El principio de inversión de dependencias es una abstracción de interfaz entre componentes de software de alto y bajo nivel para eliminar las dependencias entre ellos. Esto permite cambiar componentes de alto y bajo nivel sin afectar a otras clases, siempre que no se cambien las abstracciones de interfaz. El principio de inversión es fundamental para crear sistemas desacoplados y modulares, facilitando la modificación y extensión del software sin comprometer su estabilidad. Un ejemplo de su uso es en la utilización de *react-i18next* para la internacionalización del sistema o de *react-router-dom* para la navegación en la aplicación web. Esto permitió que dependan de abstracciones en vez de implementaciones concretas.

3.3 Diagrama de clases

Un diagrama de clases es una representación gráfica que muestra las clases de un sistema orientado a objetos, sus atributos, métodos y las relaciones entre ellas. Este tipo de diagrama es fundamental en el desarrollo de software, especialmente en proyectos que utilizan la programación orientada a objetos (OO), ya que permite a los desarrolladores visualizar la estructura del sistema y facilita la comprensión de cómo las diferentes partes interactúan entre sí (Cueto, 2016; Bonaparte, 2012).

A continuación, en la [figura 3.1](#) se representa el diagrama de clases del sistema para el REN y la generación de datos de entrenamiento. Donde se puede apreciar las clases Entidad, Datos Entrenamiento, Dato Etiquetado, Modelo y Medidas de Desempeño. Las relaciones que existen entre ellos son:

Relaciones:

- **Entidad-DatosEtiquetado:** Una entidad puede estar presente en uno o más objetos DatoEtiquetado.
- **DatosEntrenamiento-DatoEtiquetado:** Un conjunto de datos de entrenamiento puede contener uno o más objetos DatoEtiquetado.
- **Modelo-MedidasDeDesempeño:** Un modelo tiene un objeto MedidasDeDesempeño que almacena sus medidas de rendimiento.
- **Modelo-DatoEtiquetado:** Un modelo reconoce las entidades en objetos DatoEtiquetado.

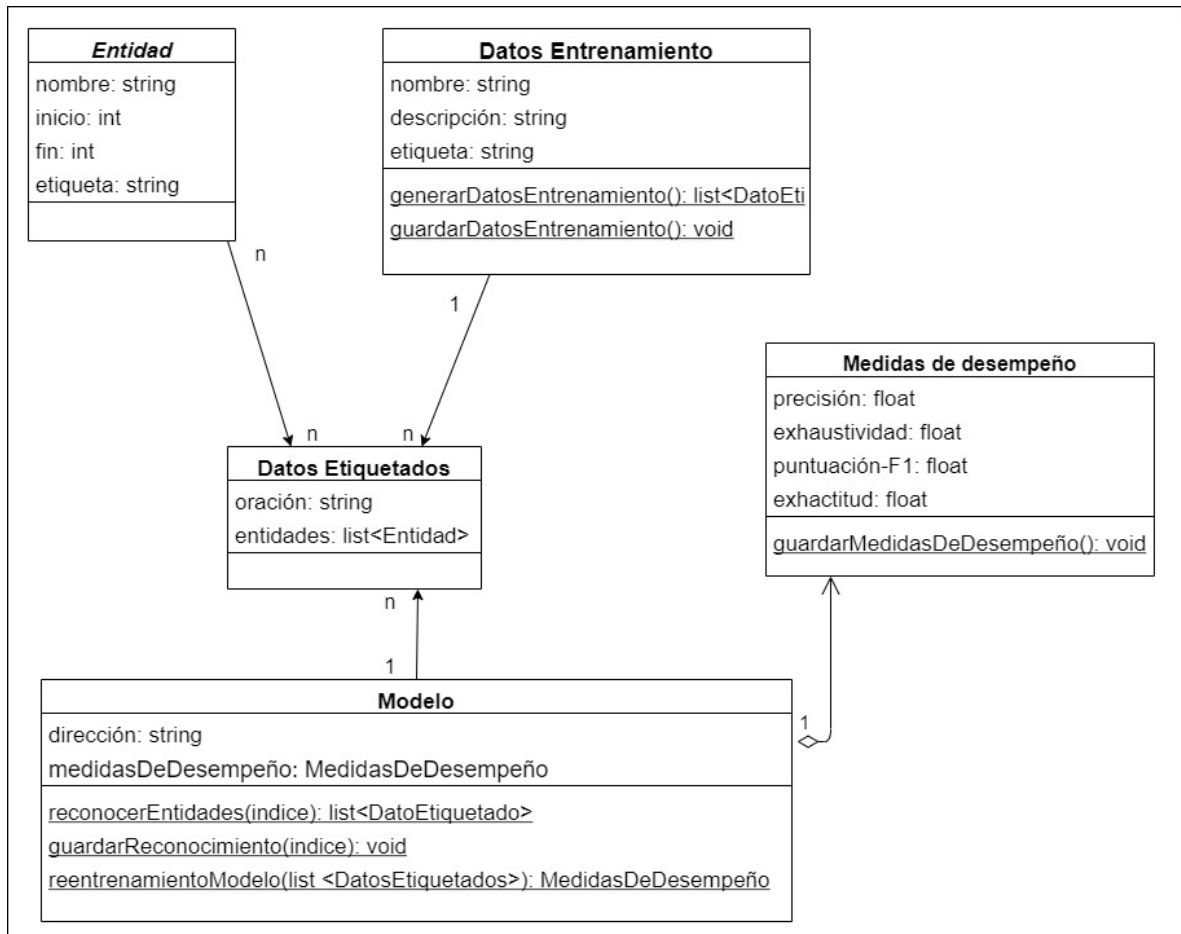


Fig 3.1 Diagrama de clases del sistema para el REN y la generación de datos de entrenamiento.

3.4 Algoritmos Importantes

A continuación, se analizan los algoritmos más importantes del sistema que le dieron cumplimiento al problema de investigación y a los requerimientos del sistema.

3.4.1 Reconocimiento de Entidades Nombradas a un índice de Elasticsearch

El código de la [Fig 3.2](#) define una función `post_index_ner_result` en FastAPI que se encarga de realizar el REN a un índice en Elasticsearch. Esta recibe un objeto de tipo `Post` el cual contiene información del índice seleccionado por el usuario.

```

@elastic_router.post("/ner-index-result")
async def post_index_ner_result(post: Post):
    try:
        nlp = load_nlp_model()
        documents = fetch_documents_from_elasticsearch(post.indice)
  
```

```

        results = process_documents(documents, nlp)
        return results
    except Exception as e:
        print(f"Error en /ner-index-result: Error: {e}")
        raise HTTPException(status_code=500, detail=str(e))

```

Fig 3.2 Función principal del reconocimiento de entidades nombradas

Cargamos el modelo de spaCy definido en `path.py` de la carpeta “*sentence-gen-api/path.py*” ([ver Fig 3.3](#)) y lo asignamos a `nlp`.

```

def load_nlp_model():
    try:
        return spacy.load(output_dir)
    except Exception as e:
        raise HTTPException(status_code=500, detail=f"Error al cargar el modelo de Spacy: {e}")

```

Fig 3.3 Función para cargar el modelo de spaCy.

Luego se hace llamada a la función “*fetch_documents_from_elasticsearch*” ([ver Fig 3.4](#)) a la cual se le asigna el índice seleccionado por el usuario. Esta función se conecta con Elasticsearch y recupera los documentos de dicho índice. Cada documento se devuelve como un diccionario que incluye el ID y su contenido.

```

def fetch_documents_from_elasticsearch(index_name: str) -> List[Dict]:
    es = Elasticsearch(["http://localhost:9200"], basic_auth=("elastic", "elastic"))
    try:
        response = es.search(index=index_name, body={"query": {"match_all": {}}}, size=1000)
        return [{"_id": doc["_id"], "_source": doc["_source"]} for doc in response["hits"]["hits"]]
    except Exception as e:
        raise HTTPException(status_code=500, detail=f"Error al recuperar documentos de Elasticsearch: {e}")

```

Fig 3.4 Función para extraer los datos del índice seleccionado

Luego se procesa los documentos en la función “*process_documents*” ([ver Fig 3.5](#)) la cual toma una lista de documentos y el modelo de spaCy como entrada. Procesa cada documento para realizar el REN.

```

def process_documents(documents: List[Dict], nlp) -> List[SentenceWithEntities]:
    results = []

```



```

for doc in documents:
    if "_source" in doc and "text" in doc["_source"]:
        spacy_doc = nlp(doc["_source"] ["text"])
        entities = [Entity(name=ent.text, start=ent.start_char,
end=ent.end_char, label=ent.label_) for ent in spacy_doc.ents]
        results.append(SentenceWithEntities(sentence=doc["_source"] ["text"],
entities=entities))
return results

```

Fig 3.5 Función para procesar los documentos y extraer las entidades nombradas

3.4.2 Salvar resultados del reconocimiento en el índice seleccionado

Una vez realizado el reconocimiento de entidades a un índice si el usuario considera que se encuentra correcto podrá salvarlo en la base de datos, la función para esto se recoge en [en la Fig 3.6:](#)

```

@elastic_router.post('/save_in_elastic')
async def post_save_in_elastic(post: SaveElastic):
    try:
        es = Elasticsearch(["http://localhost:9200"], basic_auth=("elastic",
"elastic"))
        indexName = post.indice
        if es.indices.exists(index=indexName):
            es.indices.delete(index=indexName)
        es.indices.create(index=indexName, body=settings)
        transformed_data = {}
        for i, item in enumerate(post.data, start=1):
            doc_key = f"doc{i}"
            transformed_data[doc_key] = {
                "text": item.sentence,
                "entities": [{"start": ent.start, "end": ent.end, "label": ent.label}
for ent in item.entities]
            }
        for doc_key, doc_value in transformed_data.items():
            try:
                es.index(index=indexName, id=doc_key, body=doc_value)
            except Exception as e:
                raise HTTPException(status_code=500, detail=str(e))
        return {"message": "Datos guardados exitosamente"}
    except Exception as e:
        print(f"Error en /save_in_elastic: Error: {e}")
        raise HTTPException(status_code=500, detail=str(e))

```

Fig 3.6 Función para salvar los resultados del reconocimiento de entidades nombradas

El código de la [Fig 3.6](#) define una función *post_save_in_elastic* que se encargará de guardar los datos procesados en un índice de Elasticsearch. La función acepta un objeto *SaveElastic* como entrada que contiene el nombre del índice y los datos de este. En la función se establece conexión con Elasticsearch, define la configuración del índice, incluyendo campos de text y entities. Se verifica si el índice especificado ya existe. Si es así, se elimina el índice existente y se crea un nuevo índice con la configuración definida anteriormente. Se transforman los datos proporcionados en el formato adecuado para Elasticsearch. Cada elemento de la lista data se convierte en un documento con un campo text y un campo entities que contiene una lista de entidades nombradas. Luego, se indexan estos documentos en Elasticsearch.

3.4.3 Generar datos de entrenamiento

Teniendo en cuenta el flujo del sistema representado en la [\(Fig 2.3\)](#) donde al surgir una nueva entidad en el idioma español puede generar datos de entrenamiento para usarlos para reentrenar el modelo de REN. Para ello desde la interfaz de usuario de entrenamiento este debe de llenar un formulario donde especifique el nombre de la entidad, el tipo y una descripción para generar así datos de entrenamiento en el formato requerido para entrenar el modelo de spaCy.

La función principal de la [Fig 3.7](#) se encarga de generar oraciones basadas en un modelo de lenguaje y un conjunto de entradas proporcionadas por el usuario. Se les asigna a las variables *prompt*; y *refine_prompt* los cuales se usan para guiar la generación de oraciones. Luego se almacena los resultados de *load_summarize_chain*, esta función es una parte crucial del proceso de resumen de texto en el contexto de la biblioteca *LangChain*, se utiliza para cargar una cadena de resumen que puede procesar textos y generar resúmenes de ellos.

```
@app.post("/generate-sentences")
async def sentences_generation(input: SentenceInputModel):
    try:
        entity_type = input.entity_type
        entity = input.entity
        prompt, refine_prompt = get_prompts(input.entity, input.text)
        question_list = []
```

```

docs = text_process(input.text)
for _ in range(1):
    questions = question_gen_chain.run(docs)
    question_list.append(questions)

oraciones = []
for texto in question_list:
    oraciones.extend(extraer_oraciones(texto))

json_oraciones = [{"sentences": oracion} for oracion in oraciones]
json_oraciones = generar_data_train(json_oraciones, entity, entity_type)
documents = transform_to_documents_format(json_oraciones)

return documents
except Exception as e:
    print(f"Error al generar las oraciones: {e}")

```

Fig 3.7 Función principal para generar los datos de entrenamiento

Luego de la descripción dada por el usuario se le asigna a la función “*text_process*” la cual se encarga de dividir un texto largo en fragmentos más pequeños, para así manejar grandes volúmenes de texto ([ver Fig 3.8](#)). Luego se ejecuta la cadena de procesamiento en los fragmentos de texto para generar oraciones. Este proceso se repite dos veces para obtener diferentes conjuntos de oraciones. Luego extraemos los resultados y preparamos para estructurarlas en un formato JSON específico.

```

def text_process(text: str) -> List[Document]:
    text_splitter_question_gen = CharacterTextSplitter(
        chunk_size=3000, chunk_overlap=10
    )
    text_chunks_question_gen = text_splitter_question_gen.split_text(text)
    docs_question_gen = [Document(page_content=t) for t in
text_chunks_question_gen]
    return docs_question_gen

```

Fig 3.8 Función para dividir un texto largo en fragmentos más pequeños

Después se generan los datos de entrenamiento en la función: “*generar_data_train*” ([ver Fig 3.9](#)). Esta función carga el modelo de spaCy guardado por el usuario inicialmente, elimina partes de la oración que no son relevantes para el reconocimiento de entidades como son los paréntesis, corchetes y comillas simples, después se itera sobre cada una de las oraciones generadas para el REN, cada entidad extraída se almacena en una lista de

diccionarios, donde cada diccionario contiene el nombre de la entidad, su posición de inicio y fin en el texto, y su etiqueta. Luego comprobamos si la entidad objetivo especificada por el usuario ya está presente para evitar duplicar entidades, si la entidad objetivo no se encuentra en las entidades extraídas, se añade manualmente. Al recibir los resultados en la función principal se mandan a la función “*transform_to_document_formats*” que le dan el formato necesario para reentrenar el modelo de spaCy.

```
def generar_data_train(json_oraciones, entity, entity_type):

    nlp = spacy.load(output_dir)

    for i, oracion in enumerate(json_oraciones, start=1):
        doc_text = procesar_oracion(oracion)
        entities = identificar_entidades(nlp, doc_text)
        entities = agregar_entidad_si_no_existe(entities, entity, entity_type,
nlp(doc_text))
        entities = ordenar_entidades(entities)
        json_oraciones[i-1]['entities'] = entities

    return json_oraciones
```

Fig 3.9 Función para generar datos de entrenamiento

3.4.4 Reentrenar el modelo de spaCy evitando el olvido catastrófico

El olvido catastrófico en el contexto del REN se refiere a la pérdida de habilidades o conocimientos previamente adquiridos por un modelo de IA cuando se le enseña nuevas tareas, especialmente en el ámbito del PLN. Este fenómeno es particularmente relevante en el REN, donde el modelo de IA se entrena para identificar y clasificar entidades como nombres de personas, organizaciones, lugares, entre otros, en textos (Alonso Beortegui, 2022).

Cuando un modelo de IA es reentrenado con un conjunto de datos de la misma tarea (p.ej: REN) o si se le enseña a realizar otras tareas de PLN, como la clasificación de texto o la traducción automática, puede sufrir olvido catastrófico. Esto significa que, aunque el modelo aprenda nuevas habilidades, puede perder su capacidad para identificar y clasificar correctamente las entidades nombradas en textos, lo que afecta su rendimiento en la tarea original de REN.

El equipo de la Universidad de Catania, Italia, liderado por Concetto Spampinato, ha trabajado para abordar este problema mediante el desarrollo de un método de aprendizaje consolidado de vigilia y sueño. Este enfoque intenta replicar el proceso de aprendizaje

humano, donde las personas afianzan conceptos y aprendizajes durante el sueño. En el contexto de la IA, se combina una fase de entrenamiento "despierto", donde el modelo se entrena con un conjunto de datos, y una fase de "sueño", donde el modelo analiza los datos de la fase despierta y también resúmenes de cargas de datos anteriores para evitar el olvido catastrófico (Campaña Rosero, 2023).

Los resultados de las pruebas realizadas por este equipo mostraron mejoras significativas en la precisión del reconocimiento de entidades nombradas, con incrementos entre un 2 y un 12% en comparación con el método de aprendizaje tradicional.

3.4.4.1 Algoritmo propuesto para resolver el olvido catastrófico

Para resolver el problema anterior adaptado a la problemática de investigación se diseñó el siguiente algoritmo ver [Fig 3.10](#):

Entrada: Datos de entrenamiento generados

Salida: Modelo reentrenado o mensaje de pérdida de conocimiento

- 1- **Fase Entrenamiento “Despierto”**: primero se toman los datos generados por el modelo de llama2 y se dividen en una proporción de 75% para datos de entrenamiento y 25% para datos de prueba.
- 2- **Fase Entrenamiento “Despierto”**: Los datos de entrenamiento generados se concatenan con los datos de entrenamiento almacenados en la base de datos de Elasticsearch al igual que los datos de prueba con su índice correspondiente.
- 3- **Fase Entrenamiento “Despierto”**: Se reentrena el modelo de spaCy con la unión de los datos nuevos y viejos.
- 4- **Fase “Sueño”**: Al modelo obtenido se comprueba la precisión, la exactitud, puntuación F1, y la exhaustividad con los datos de prueba.
- 5- **Fase “Sueño”**: Comparamos estas medidas con las anteriores del modelo almacenadas en la base de datos, si no existe pérdida de conocimiento entonces guardamos los datos de entrenamiento, los de prueba, y las métricas del modelo calculadas.
- 6- **Fase “Sueño”**: Si se pierde conocimiento se le envía un mensaje de error al usuario para que vuelva a generar los datos.

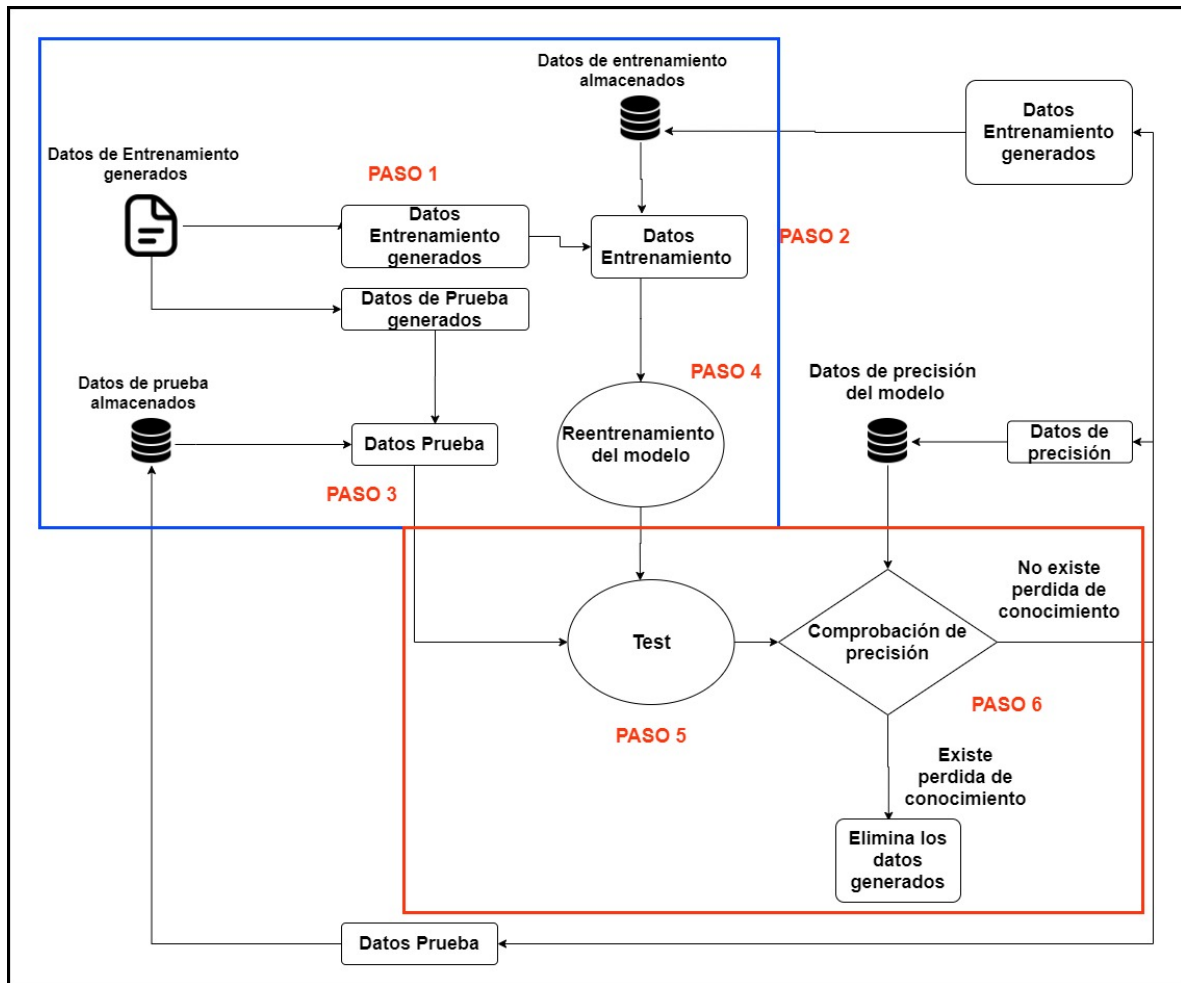


Fig 3.10 Diagrama de reentrenamiento para evitar el olvido catastrófico

3.5 Medidas de Desempeño

Para la detección de entidades, se diferenciarán cuatro casos:

- **Correcta (Cd):** Una detección será correcta cuando sobre en una frase se encuentra una anotación con el mismo inicio y final.
- **Parcial (Pd):** Una detección es parcial si en una frase se solapan las detecciones. Por ejemplo, que comiencen en la misma posición de inicio, pero finalicen en una posición final diferente. Uno de los casos en los que se puede producir es cuando el modelo nuevo predice una entidad en varias partes, mientras que en el gold standard es solo una.
- **Falta (Md):** Una detección será clasificada como falta, cuando se encuentra en el estándar, pero no en las detectadas por el nuevo modelo.
- **Falso positivo (Sd):** Este último caso sería la inversa de la falta. Se da cuando el modelo ha detectado una entidad que no está en el estándar (Sánchez Pérez, 2022).

Con estos cuatros elementos, se calcularán la precisión (precision), exhaustividad (recall), puntuación F1 (F1 score) y la exactitud (accuracy) del modelo. Para ello, se utilizan las siguientes fórmulas ver [Fig 3.11](#):

$$\begin{aligned}
 Precision_{deteccion} &= \frac{C_d + 0.5 * P_d}{C_d + P_d + S_d} \\
 Recall_{deteccion} &= \frac{C_d + 0.5 * P_d}{C_d + P_d + M_d} \\
 F1 - Score_{deteccion} &= \frac{Precision_{deteccion} * Recall_{deteccion}}{Precision_{deteccion} + Recall_{deteccion}} \\
 Accuracy_{deteccion} &= \frac{C_d}{C_d + M_d + S_d}
 \end{aligned}$$

Fig 3.11 Fórmulas de las medidas de desempeño del modelo para el REN

Para el conjunto de datos de comprobación con las nuevas entidades a reconocer se calcularon las siguientes métricas [ver Tab 3.1](#).

Tabla 3.1 Métricas de los modelos original y el modelo reentrenado para un conjunto de datos

Métrica	Modelo Original	Modelo Reentrenado
Precisión	0.7833	0.882
Exhaustividad	0.5465	0.849
Puntuación - F1	0.321	0.432
Exactitud	0.4791	0.762

Las métricas de evaluación muestran claramente que el modelo reentrenado para el reconocimiento de entidades nombradas presenta un rendimiento considerablemente mejorado en comparación con el modelo original. Esto indica que el proceso de reentrenamiento del modelo ha sido efectivo y ha llevado a mejoras significativas en su capacidad para reconocer y clasificar correctamente las entidades nombradas en los datos de prueba.

3.6 Análisis económico del costo de producción del sistema.

En el desarrollo de un sistema de reconocimiento de entidades nombradas, la planificación y el control del esfuerzo, costo y tiempo son fundamentales. La utilización de técnicas de PLN puede ser crucial para estimar los costos asociados con este tipo de proyecto. Lo que puede ayudar a extraer, estructurar e interpretar información relevante de diversas fuentes. Esto es esencial para una estimación precisa de los recursos necesarios y el tiempo requerido para el desarrollo de un sistema de reconocimiento de entidades nombradas.

El método de punto de casos de uso es una técnica utilizada en la gestión de proyectos de desarrollo de software para estimar el esfuerzo necesario y los recursos requeridos para llevar a cabo el proyecto. Consiste en cuatro etapas que involucran el cálculo de varios factores, incluyendo los puntos de historias de usuarios sin ajustar (UUCP), el factor de complejidad técnica (TCF), y el factor de ambiente (EF). En la primera etapa, se calculan los puntos de historias de usuarios sin ajustar considerando el peso de los actores. Luego, se ajustan estos puntos considerando el TCF y el EF para obtener los puntos de historias de usuarios ajustadas (UCP).

A través de los UCP, se estima el esfuerzo en horas hombre para el proyecto. Finalmente, se distribuye este esfuerzo en distintas etapas del proyecto, y se calcula el costo total del proyecto en función de las tarifas horarias promedio y otros costos asociados. Este método permite una estimación detallada y precisa del esfuerzo y el costo involucrado en el desarrollo de un proyecto de software, lo que facilita la planificación y gestión efectiva del mismo.

En la realización del análisis económico ([ver Anexo III](#)) se llegó a la conclusión que el costo total del proyecto sería aproximadamente de **\$ 145 431** en CUP.

En este análisis no se tuvo en cuenta los costos por electricidad e internet necesarios para el desarrollo del sistema desde el hogar.

3.7 Pruebas al sistema

Un escenario de prueba establece un método de evaluación del sistema, mediante la introducción de datos para verificar si se produce los resultados esperados bajo las condiciones específicas en el que se está probando. La metodología XP enfatiza en el proceso de pruebas, promoviendo la prueba lo más ampliamente posible para reducir el número de errores no detectados y disminuir el tiempo entre la aparición de un error y su detección.

Las pruebas de funcionalidad y de aceptación se desarrollan basándose en las historias de usuario, en cada ciclo de iteración del desarrollo del software, permitiendo confirmar que la historia ha sido implementada correctamente. En caso de que múltiples pruebas fallen, deben indicar el orden de prioridad para su resolución. A lo largo de la implementación de la aplicación, se diseñaron un conjunto de escenarios de prueba para verificar su funcionamiento de acuerdo a los requerimientos descritos en las HU, que fueron definidas en el capítulo 2.

La [Tabla 3.2](#), denominada "Caso de Prueba 'Autenticación'", presenta diversos escenarios de entrada y los resultados esperados correspondientes. En el primer caso, al ingresar una

contraseña, se espera que esta no se muestre mientras se escribe, lo que indica un nivel básico de seguridad en el proceso de autenticación. Por otro lado, al proporcionar un usuario y contraseña válidos, el sistema debe almacenar esta información en la base de datos, registrando así al usuario y permitiéndole acceder al sistema. Sin embargo, en el caso de proporcionar una contraseña incorrecta o un nombre de usuario erróneo, el sistema debe notificar al usuario sobre el error, asegurando así que se intenta iniciar sesión con credenciales válidas. Esta tabla detalla los escenarios clave que se deben considerar y verificar durante el proceso de autenticación, garantizando un sistema robusto y seguro.

Tabla 3.2 Caso de Prueba “Autenticación”

Entrada	Resultados	Condición
Una contraseña	No se muestra la contraseña al escribir	Se escribe la contraseña para iniciar sesión
Usuario y contraseña	Se guarda en base de datos la información de registro e inicio de sesión del usuario actual	Se registra y entra al sistema
Una contraseña incorrecta o usuario incorrecto	Se notifica al usuario del usuario o la contraseña es incorrecta	Se trata de registrar con usuarios y contraseñas erróneos

La [Tabla 3.3](#), titulada "Caso de Prueba: 'Inicio al sistema con datos de prueba'", describe los procedimientos de inicio del sistema utilizando datos de prueba. En el primer escenario, al aceptar los datos de inicio para prueba, se espera que los índices se indexen en Elasticsearch, lo que permite al usuario realizar pruebas en el sistema. Esto se activa al presionar el botón de aceptar datos de pruebas iniciales. Por otro lado, en el segundo caso, si se aceptan los datos de inicio para prueba por segunda vez, el sistema notificará al usuario que los índices ya han sido creados, lo que sugiere que no es necesario volver a crearlos. Este proceso se activa al presionar nuevamente el botón de aceptar datos de pruebas iniciales, con los índices ya existentes en la base de datos. La tabla proporciona una guía clara sobre cómo iniciar el sistema con datos de prueba y cómo el sistema responde en diferentes situaciones, asegurando un flujo eficiente durante las pruebas.

Tabla 3.3: Caso de Prueba: “Inicio al sistema con datos de prueba”

Entrada	Resultados	Condición
Aceptar datos de inicio para prueba	Se indexan índices en Elasticsearch para la prueba	Se presiona el botón de aceptar datos de pruebas iniciales

	del sistema por parte del usuario	
Aceptar datos de inicio para prueba por segunda vez	Se notifica al usuario que los índices ya han sido creados	Se presiona el botón de aceptar datos de pruebas iniciales con los índices ya existentes en la base de datos

La [Tabla 3.4](#), bajo el título "Caso de Prueba: REN", detalla los diferentes escenarios relacionados con el proceso de reconocimiento de entidades nombradas utilizando Elasticsearch. En el primer caso, al intentar acceder a la dirección del reconocimiento de entidades nombradas, se notifica al usuario que la base de datos no está en funcionamiento, indicando que Elasticsearch no está corriendo. En otro escenario, si se accede a la misma dirección, pero no se muestra ningún índice disponible, esto sugiere que no existen índices almacenados en Elasticsearch. Sin embargo, si se muestran los índices disponibles, confirma la presencia de índices en Elasticsearch. Cuando se selecciona un índice, pero no está en el formato correcto, se muestra un error en los resultados indicando el campo con formato incorrecto. Por el contrario, si el índice seleccionado está en el formato correcto, se muestran los resultados obtenidos del reconocimiento de entidades nombradas. Finalmente, al pulsar el botón de salvar en Elasticsearch, los resultados se almacenan en la base de datos, lo que completa el proceso de guardar los resultados. Esta tabla proporciona una guía clara sobre los diferentes casos que pueden surgir durante el REN, asegurando un funcionamiento fluido y efectivo del sistema.

Tabla 3.4: Caso de Prueba: "REN"

Entrada	Resultados	Condición
Entrada a la dirección del reconocimiento de entidades nombradas a los índices de Elasticsearch	Se notifica al usuario que la base de datos no está corriendo	La base de datos Elasticsearch no está corriendo
Entrada a la dirección del reconocimiento de entidades nombradas a los índices de Elasticsearch	No se muestra ningún índice disponible para ejecutar el reconocimiento de entidades nombradas	No existen índices almacenados en Elasticsearch
Entrada a la dirección del reconocimiento de entidades	Muestra los índices disponibles en Elasticsearch	Existen índices de Elasticsearch

nombradas a los índices de Elasticsearch		
Índice seleccionado	Muestra el error en el apartado de los resultados, cual es el campo que se encuentra en el formato incorrecto	Seleccionar un índice de Elasticsearch que no se encuentre en el formato correcto
Índice seleccionado	Muestra los resultados obtenidos del reconocimiento de entidades nombradas	Seleccionar un índice de Elasticsearch que fue almacenado en el formato correcto
Pulsar botón de salvar en Elasticsearch	Almacena los resultados en la base de datos	Guardar los resultados en la base de datos

La [tabla 3.5](#) muestra los resultados en cuanto a cantidad de datos a los que se le realizó el REN y el tiempo en que se demoró el modelo, teniendo en cuenta que el mismo no ha sido reentrenado. Se aprecia que el tiempo de procesamiento aumenta linealmente con la cantidad de datos. En cuanto a la precisión el modelo no mostro ningún cambio ya que se mantuvo en la media entre 77% y 79%. Como estos conjuntos de datos tienen entidades que el modelo no reconoce se recomienda realizar el reentrenamiento del modelo.

Tabla 3.5 Resultados del modelo sin reentrenar al realizar REN a conjuntos de datos

Cantidad de datos	Tiempo de REN	Precisión
200	1.78 segundos	78.33%
2000	20.10 segundos	77.91%
9000	49.17segundos	78.10%

La [Tabla 3.6](#), titulada "Caso de Prueba: 'Reentrenamiento del modelo'", enumera diversos escenarios relacionados con el proceso de reentrenamiento del modelo en un contexto específico, probablemente relacionado con el uso de spaCy. En el primer caso, al ingresar datos que incluyen el nombre de la entidad, su descripción y el tipo de entidad, el sistema genera oraciones que se utilizan como datos de entrenamiento para el modelo de spaCy. Esto ocurre cuando se introduce información para generar datos de entrenamiento. En otro escenario, si se vuelve a ingresar correctamente los datos al sistema, se confirma la reintroducción de información para generar datos de entrenamiento para el modelo de spaCy. Sin embargo, si se proporciona un conjunto de datos de entrenamiento erróneos, el sistema emite una alerta y ofrece la opción de repetir la generación de datos, indicando así

la creación de datos de entrenamiento incorrectos. Además, al elegir la opción de generar datos de entrenamiento, si el modelo pierde precisión, los cambios realizados durante el reentrenamiento no se guardan, lo que señala que el proceso de reentrenamiento puede comprometer la precisión del modelo. Esta tabla proporciona una visión detallada de los diferentes casos que pueden surgir durante el proceso de reentrenamiento del modelo, destacando las acciones y resultados esperados en cada situación para garantizar un reentrenamiento efectivo y preciso del modelo.

Tabla 3.6: Caso de Prueba “Reentrenamiento del modelo”

Entrada	Resultados	Condición
Datos (Nombre de la entidad, descripción, y tipo de entidad)	Generación de oraciones que da como resultado datos de entrenamiento para el modelo de spaCy	Entrada de información para generar datos de entrenamiento para el modelo de spaCy
Entrada de datos (Nombre de la entidad, descripción, y tipo de entidad)	Volver a entrar los datos al sistema de manera correcta	Entrada de información para generar datos de entrenamiento para el modelo de spaCy
Conjunto de datos de entrenamiento erróneos	El sistema lanza una alerta, y da la opción de volver a repetir la generación de datos	El sistema crea datos de entrenamiento erróneos
Elige la opción de generar datos de entrenamiento	EL modelo si pierde precisión no se guardan los cambios realizados	Reentrenamiento del modelo pierde precisión

La [tabla 3.7](#) muestra los resultados al reentrenar el modelo con distintos dataset que se encuentran en la plataforma de HuggingFace esto se realizó con el propósito de mostrar la precisión del modelo al reentrenarse en comparación con el tiempo que se demoró el reentrenamiento y la cantidad de datos utilizados. Los resultados obtenidos resaltan la importancia del tamaño y la calidad del conjunto de datos de entrenamiento en el rendimiento del modelo, demostrando mejoras en la capacidad de REN. Sin embargo, en conjuntos de datos extremadamente grandes puede requerir una inversión significativa de tiempo y recursos computacionales, teniendo en cuenta los requisitos no funcionales mínimos abordados en el capítulo 2 se sugiere el uso de más memoria RAM, así como de una GPU para mejorar el rendimiento del sistema.

Tabla 3.7 Resultados del reentrenamiento a diferentes dataset

Dataset	Cantidad de datos	Tiempo de reentrenamiento	Precisión
Podcast-ner-es	Entrenamiento: 209 Prueba: 53	29.35 segundos	51.12%
CONLL-NERC	Entrenamiento: 8323 Prueba: 1517	21 minutos	86.27%
Ner-massive	Entrenamiento: 471342 Prueba: 11136	9 horas	99.07%

La [Tabla 3.8](#), bajo el título "Caso de Prueba: 'Probar el modelo reentrenado para el reconocimiento de entidades nombradas'", presenta un escenario específico relacionado con la evaluación del modelo reentrenado para el reconocimiento de entidades nombradas, probablemente utilizando spaCy u otro sistema similar. En este caso, la entrada consiste en los datos del índice seleccionado, que se refiere a la selección de un índice que contiene una entidad que el modelo de spaCy no reconocía previamente como una entidad. Como resultado de esta entrada, se espera que el sistema realice el reconocimiento de las entidades, incluyendo la entidad previamente no reconocida. Esta tabla describe un caso de prueba crucial para verificar la capacidad del modelo reentrenado para reconocer entidades nombradas con precisión, lo que garantiza su efectividad y mejora continua en el proceso de reconocimiento de entidades.

Tabla 3.8: Caso de Prueba "Probar el modelo reentrenado para el reconocimiento de entidades nombradas"

Entrada	Resultados	Condición
Datos del índice seleccionado	Reconocimiento de las entidades incluyendo la entidad antes no reconocida	Seleccionar índice que contiene la entidad que el modelo de spaCy no reconocía como una entidad

La [tabla 3.9](#) muestra los resultados del REN a los mismos conjuntos de datos luego de realizar el reentrenamiento.

Tabla 3.9 Caso de Prueba: "REN luego de reentrenar el modelo con las entidades que no reconocía"

Cantidad de datos	Tiempo de REN	Precisión
200	1.81 segundos	89.75%
2000	20.15 segundos	90.01%

9000	49.26 segundos	89.32%
------	----------------	--------

Con los resultados obtenidos al realizar el REN del modelo sin reentrenar, el reentrenamiento con distintos conjuntos de datos y el REN luego de realizar el reentrenamiento podemos llegar a las conclusiones siguientes:

1. Mejora consistente en la precisión:

- En general, se observa una mejora notable en la precisión del modelo en el reconocimiento de entidades en todos los conjuntos de datos luego del reentrenamiento.
- Por ejemplo, en el caso del conjunto de datos de 200 instancias, la precisión pasó de 78.33% a 89.75%, lo que representa una mejora significativa.

2. Estabilidad en el tiempo de procesamiento:

- Aunque el tiempo de procesamiento varía ligeramente entre los diferentes conjuntos de datos, no hay un aumento significativo en el tiempo requerido para el proceso de reconocimiento a medida que aumenta la cantidad de datos.
- Esta estabilidad en el tiempo de procesamiento sugiere que el modelo es capaz de manejar conjuntos de datos más grandes sin un aumento desproporcionado en la carga computacional.

3. Importancia de la cantidad de datos:

- Los resultados resaltan la importancia de la cantidad de datos de entrenamiento en el rendimiento del modelo de reconocimiento de entidades.
- A medida que aumenta la cantidad de datos, el modelo tiende a mejorar su precisión, lo que sugiere que la exposición a una mayor variedad de ejemplos facilita al modelo aprender patrones más complejos y generalizar mejor.

3.8 Ejemplo de resultados obtenidos

Con los datos iniciales de prueba indexados en Elasticsearch se realizó un reconocimiento de entidades a los mismo en el cual se pudo apreciar el siguiente resultado de la [fig 3.12](#):

```
{
  "sentence": "El teamacere perdió el juego de pelota contra el equipo
de los Estados Unidos",
  "entities": [
    {
```

```

    "name": "Estados Unidos",
    "start": 63,
    "end": 77,
    "label": "LOC"
  }
]
}

```

Fig 3.12 Resultados de Reconocimiento de Entidades al índice de Elasticsearch antes del reentrenamiento

Como se puede ver en la figura el modelo no reconoce la entidad teamacere de tipo organización por lo tanto se generan datos de entrenamiento para reentrenar el modelo de spaCy. Un ejemplo de los datos de entrenamiento generados por el modelo se encuentran en el [Anexo IV](#)

Al reentrenar el modelo y volverlo a pasar por el índice de Elasticsearch nos da el siguiente resultado ([ver Fig 3.13](#))

```

{
  "sentence": "El teamacere perdió el juego de pelota contra el equipo de los Estados Unidos",
  "entities": [
    {
      "name": "teamacere",
      "start": 3,
      "end": 12,
      "label": "ORG"
    },
    {
      "name": "Estados Unidos",
      "start": 63,
      "end": 77,
      "label": "LOC"
    }
  ]
}

```

Fig3.13 Resultados del Reconocimiento de Entidades al índice de Elasticsearch despues del reentrenamiento

Por lo que el reentrenamiento fue completado con exito reconociendo asi la entidad teamacere de tipo ORG.

3.9 Implementación de la Interfaz de Usuario para el REN

El resultado de la implementación de la interfaz de usuario está representado en la [Fig 3.14](#). El mismo es minimalista ya que no posee elementos innecesarios que puedan distraer al usuario de la tarea en cuestión. Es fácil de navegar ya que todos los links se encuentran en el botón del menú de navegación. Además, se redimensiona automáticamente para adaptarse al tamaño de la pantalla del usuario.

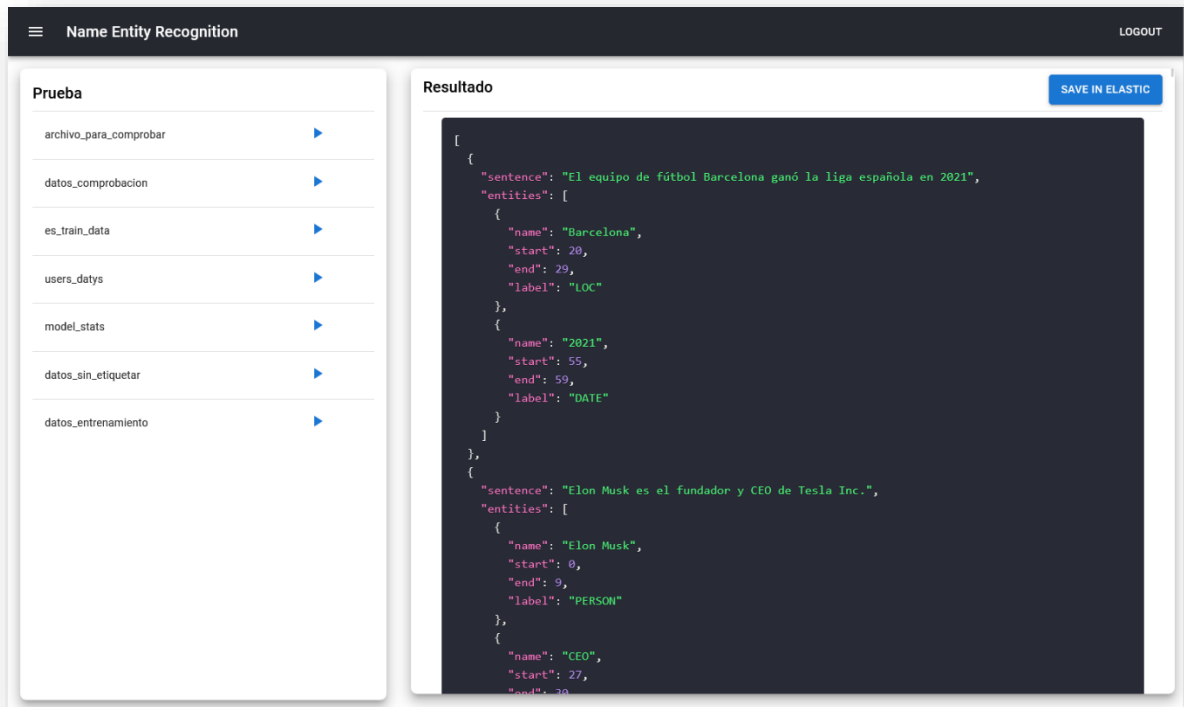


Fig 3.14 Interfaz de usuario para el REN

3.10 Conclusiones del capítulo

En el capítulo, se abordaron las funcionalidades claves, la implementación y las pruebas del sistema diseñado, así como los resultados obtenidos. Se detallaron los pasos de instalación, se presentó un diagrama de clases y se analizaron los algoritmos cruciales, como el diseño para mitigar el olvido catastrófico en el REN. Los resultados de las pruebas a tres conjuntos de datos de diferentes tamaños mostraron mejoras significativas en la precisión del modelo tras el reentrenamiento, validando su efectividad. Se realizó el análisis económico ahorrando a la empresa una cifra de **\$ 145 431** en CUP aproximadamente. Se destacó la importancia de la implementación de una interfaz de usuario intuitiva, proporcionando así una visión integral del desarrollo y desempeño del sistema.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

En la investigación se realizó un análisis crítico del estado del arte sobre las técnicas actuales en el campo del PLN para el REN y su evolución con el desarrollo de las tecnologías. Se evidencia su importancia en la extracción de información relevante en grandes volúmenes de datos, incidiendo en cuanto a los recursos y herramientas de PLN. Se diseñó e implementó un sistema informático para la empresa DATYS utilizado para el reconocimiento de entidades nombradas en el idioma español, empleando el modelo de spaCy. Aunque el modelo no reconozca entidades nuevas en el idioma o tenga problemas para clasificar una ya existente, se desarrolló un módulo para la generación de datos de entrenamiento con el modelo de Llama2 permitiendo tener un conjunto de datos etiquetados para el reentrenamiento de modelos computacionales para el REN y evitar el olvido catastrófico durante esta fase. Los resultados luego de realizar el reentrenamiento fueron satisfactorios en la evaluación de las medidas de desempeño, demostrando la necesidad de un gran corpus en el idioma español para obtener mejores resultados. Este sistema fue aceptado por los directivos de la empresa otorgando un aval de cumplimiento de los objetivos [ver Anexo V](#). Logrando un aporte a la independencia tecnológica y al proceso de informatización del país.

Recomendaciones

- Integrar otras funcionalidades al flujo de procesamiento de documentos como el procesamiento de lenguaje natural para realizar otras tareas como la traducción de texto, el análisis de sentimientos en las oraciones, similitud en textos.
- Integrar modelos de reconocimiento de entidades nombradas en otros idiomas como inglés, francés, portugués.

REFERENCIAS BIBLIOGRÁFICAS

- Albuquerque, H. O., Souza, E., Gomes, C., Pinto, M. H. D. C., Ricardo Filho, P. S., Costa, R., ... & Oliveira, A. L. (2023). Named entity recognition: a survey for the portuguese language. *Procesamiento del Lenguaje Natural*, 70, 171-185.
- Alonso Beortegui, A. (2022). Estudio y comparación de algoritmos de aprendizaje incremental de clases.
- Alvarez, O. D. G., Larrea, N. P. L., & Valencia, M. V. R. (2022). Análisis comparativo de Patrones de Diseño de Software. Polo del Conocimiento: Revista científico-profesional, 7(7), 2146-2165.
- Aros, C. G. (2008). RuP: Metodología en los sistemas y aplicaciones basadas en la web. Avances: Investigacion en Ingenieria, 1(8), 83-87.
- Bautista-Villegas, E. (2022). Metodologías ágiles XP y Scrum, empleadas para el desarrollo de páginas web, bajo MVC, con lenguaje PHP y framework Laravel. *Revista Amazonía Digital*, 1(1), e168-e168.
- Bernabeu Pérez, P. (2022). Reconocimiento de entidades nombradas y categorización de textos periodísticos (Doctoral dissertation, Universitat Politècnica de València).
- Bisong, E., & Bisong, E. (2019). Google automl: Cloud natural language processing. *Building machine learning and deep learning models on google cloud platform: a comprehensive guide for beginners*, 599-612.
- Blas, M. J., Leone, H. P., & Gonnet, S. M. (2019). Modelado y Verificación de Patrones de Diseño de Arquitectura de Software para Entornos de Computación en la Nube.
- Blischak, J. D., Davenport, E. R., & Wilson, G. (2016). A quick introduction to version control with Git and GitHub. *PLoS computational biology*, 12(1), e1004668.
- Bonaparte, U. J. (2012). Proyectos UML Diagramas de clases y aplicaciones JAVA en NetBeans 6.9. 1. Proyecto. Universidad Tecnológica Nacional–UTN, Facultad Regional Tucumán.
- Campaña Rosero, J. D. (2023). Plasticidad sináptica para mitigar el problema del olvido catastrófico durante el aprendizaje continuo en redes neuronales artificiales.
- Carvalho, A., Oliveira, A. L., Albuquerque, H. O., Souza, E., Gomes, C., Pinto, M. H. D. C., ... & da Silva, N. F. (2023). Reconocimiento de Entidades Nombradas: una investigación para el idioma Portugués. *Procesamiento del lenguaje natural*, (70), 171-185.
- Challenger-Pérez, I., Díaz-Ricardo, Y., & Becerra-García, R. A. (2014). El lenguaje de programación Python. *Ciencias Holguín*, 20(2), 1-13.
- Chen, S., Thaduri, U. R., & Ballamudi, V. K. R. (2019). Front-End Development in React: An Overview. *Engineering International*, 7(2), 117-126.

- Chen, Y., Argentinis, J. E., & Weber, G. (2016). IBM Watson: how cognitive computing can be applied to big data challenges in life sciences research. *Clinical therapeutics*, 38(4), 688-701.
- Cincović, J., & Punt, M. (2020). Comparison: Angular vs. React vs. Vue. Which framework is the best choice. Universidad de Belgrade.
- Cueto, M. J. J. F., & Zuñiga, C. B. Diagrama de clases en UML. *Diagrama de clases en UML.[En línea][Citado el: 25 de febrero de 2016.] <http://es.scribd.com/doc/31096724/Diagrama-de-Clases-en-UML#scribd>*.
- Cuyutupa Calderón, E. (2023). Sistema de rentas para mejorar la recaudacion de impuesto a la propiedad con arquitectura cliente servidor en la Municipalidad Distrital de Huacrapuquio-Huancayo.
- De Kok, T. (2023). Generative LLMs and textual analysis in accounting:(Chat) GPT as research assistant?. Available at SSRN. Disponible en: <https://ssrn.com/abstract=4429658> fecha de consulta 7 de marzo del 2024.
- Dhulavvagol, P. M., Bhajantri, V. H., & Totad, S. G. (2020). Performance analysis of distributed processing system using shard selection techniques on elasticsearch. *Procedia Computer Science*, 167, 1626-1635.
- Diego Olite, F. M., Morales Suárez, I. D. R., & Vidal Ledo, M. J. (2023). Chat GPT: origen, evolución, retos e impactos en la educación. *Educación Médica Superior*, 37(2).
- García-Holgado, A., García, M., Vázquez-Ingelmo, A., & García-Peñalvo, F. J. (2020). UML. Unified Modeling Language.
- Gelbukh, A. (2010). Procesamiento de lenguaje natural y sus aplicaciones. *Komputer Sapiens*, 1, 6-11.
- Gómez, O. T., López, P. P. R., & Bacalla, J. S. (2010). Criterios de selección de metodologías de desarrollo de software. *Industrial data*, 13(2), 70-74.
- Gonzaga, M. K. C., Pazos, W. J. O., Meneses, L. J. U., & Esteban, J. A. (2019). Metodología Híbrida de Desarrollo de Software combinando XP y SCRUM. *Mikarimin. Revista Científica Multidisciplinaria*, 5(2), 109-116.
- Goyal, A., Gupta, V., & Kumar, M. (2018). Recent named entity recognition and classification techniques: a systematic review. *Computer Science Review*, 29, 21-43.
- Gritta, M., Pilehvar, M. T., & Collier, N. (2020). A pragmatic guide to geoparsing evaluation: Toponyms, Named Entity Recognition and pragmatics. *Language resources and evaluation*, 54, 683-712.
- Gruber, J. B., & Weber, M. (2024). rollama: An R package for using generative large language models through Ollama. arXiv preprint arXiv:2404.07654.

- Hakala, K., & Pyysalo, S. (2019, November). Biomedical named entity recognition with multilingual BERT. In *Proceedings of the 5th workshop on BioNLP open shared tasks* (pp. 56-61).
- Hernández-Sampieri, R., & Mendoza, C. (2020). Metodología de la investigación: las rutas cuantitativa, cualitativa y mixta.
- Ihrig, C. J., & Ihrig, C. J. (2013). Javascript object notation. *Pro Node. js for Developers*, 263-270.
- Jain, A. (2024, 2 de abril). Qué son los Requisitos No Funcionales: Ejemplos, Definición, Guía Completa. Visure Solutions. <https://visuresolutions.com/es/blog/requerimientos-no-funcionales/>
- Lathkar, M. (2023). Introduction to fastapi. In *High-Performance Web Apps with FastAPI: The Asynchronous Web Framework Based on Modern Python* (pp. 1-28). Berkeley, CA: Apress.
- Lituma-Sarmiento, A. F., & Vizñay-Durán, J. K. (2023). Análisis y Diseño de una propuesta de sistema integral de Gestión Empresarial basado en una arquitectura Cliente-Servidor. *MQRInvestigar*, 7(1), 2262-2290.
- Liu, P., Guo, Y., Wang, F., & Li, G. (2022). Chinese named entity recognition: The state of the art. *Neurocomputing*, 473, 37-53.
- Loeliger, J., & McCullough, M. (2012). *Version Control with Git: Powerful tools and techniques for collaborative software development*. " O'Reilly Media, Inc."
- Lund, B. D., & Wang, T. (2023). Chatting about ChatGPT: how may AI and GPT impact academia and libraries?. *Library hi tech news*, 40(3), 26-29.
- Luzniak, K. (2023). GPT-4 vs. GPT-3. OpenAI Models' Comparación. <https://neoteric.eu/blog/gpt-4-vs-gpt-3-openai-models-comparison/>
- Marcillo Ligua, W. E. (2021). Análisis del modelado UML en el diseño y desarrollo del Software de calidad (Bachelor's thesis, Jipijapa. UNESUM).
- Martin, R. L., Iraola, D. M., Louie, E., Pierce, D., Tagtow, B. A., Labrie, J. J., & Abrahamson, P. G. (2018). Hybrid natural language processing for high-performance patent and literature mining in IBM Watson for Drug Discovery. *IBM Journal of Research and Development*, 62(6), 8-1.
- Menzinsky, A., López, G., Palacio, J., Sobrino, M. Á., Álvarez, R., & Rivas, V. (2018). Historias de usuario. Ingeniería de requisitos ágil.
- Microsoft. (2024). Visual Studio Code. <https://code.visualstudio.com/>
- Mirjalili, V., & Raschka, S. (2020). *Python machine learning*. Marcombo.
- Missaoui, S., MacFarlane, A., Makri, S., & Gutierrez-Lopez, M. (2019). DMINR at TREC News Track. In *TREC*
- Molina Hernández, Yenisel, Granda Dihigo, Ailec, & Velázquez Cintra, Alionuska. (2019). Los requisitos no funcionales de software. Una estrategia para su desarrollo en el Centro de

- Informática Médica. *Revista Cubana de Ciencias Informáticas*, 13(2), 77-90. Recuperado en 14 de abril de 2024, de http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2227-18992019000200077&lng=es&tlng=es.
- Moreira, D., Cruz, I., Gonzalez, K., Quirumbay, A., Magallan, C., Guarda, T., ... & Castillo, C. (2021). Análisis del Estado Actual de Procesamiento de Lenguaje Natural. *Revista Ibérica de Sistemas e Tecnologias de Informação*, (E42), 126-136.
- Nasar, Z., Jaffry, S. W., & Malik, M. K. (2021). Named entity recognition and relation extraction: State-of-the-art. *ACM Computing Surveys (CSUR)*, 54(1), 1-39.
- Nolan, D., Temple Lang, D., Nolan, D., & Lang, D. T. (2014). Javascript object notation. *XML and Web Technologies for Data Sciences with R*, 227-253.
- Pantaleo, G., & Rinaudo, L. (2015). Ingeniería de software. Alpha Editorial.
- Pardo Gómez, M. E. (2021). La Lógica del proceso de investigación científica. Recuperado de <https://eva.uo.edu.cu/mod/folder/view.php?id=123721>
- Paredes, D. A. V., Martínez, L. C. C., Bermúdez, R. M. L., & Mendoza, S. R. P. (2019). Análisis de la metodología RUP en el desarrollo de software académico mediante la herramienta DJANGO. *RECIMUNDO*, 3(2), 964-979.
- Partalidou, E., Spyromitros-Xioufis, E., Doropoulos, S., Vologianidis, S., & Diamantaras, K. (2019, October). Design and implementation of an open source Greek POS Tagger and Entity Recognizer using spaCy. In *IEEE/WIC/ACM International Conference on Web Intelligence* (pp. 337-341).
- Peeters, R., & Bizer, C. (2023). Entity matching using large language models. *arXiv preprint arXiv:2310.11244*.
- Saks, E. (2019). JavaScript Frameworks: Angular vs React vs Vue.
- Sánchez Pérez, A. (2022). Generación de datos sintéticos para el refuerzo de modelos de aprendizaje automático en entornos reales para la extracción de información.
- Schmitt, X., Kubler, S., Robert, J., Papadakis, M., & LeTraon, Y. (2019, October). A replicable comparison study of NER software: StanfordNLP, NLTK, OpenNLP, SpaCy, Gate. In *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)* (pp. 338-343). IEEE.
- Shaik, S., Naga, N., & Rao, M. (2017). A review of elastic search: performance metrics and challenges. *International Journal on Recent and Innovation Trends in Computing and Communication*, 5(11), 222-229.
- Shelar, H., Kaur, G., Heda, N., & Agrawal, P. (2020). Named entity recognition approaches and their comparison for custom ner model. *Science & Technology Libraries*, 39(3), 324-337.

- Shelar, H., Kaur, G., Heda, N., & Agrawal, P. (2020). Named entity recognition approaches and their comparison for custom ner model. *Science & Technology Libraries*, 39(3), 324-337.
- Sundheim, B. M. (1996, May). The message understanding conferences. In TIPSTER TEXT PROGRAM PHASE II: Proceedings of a Workshop held at Vienna, Virginia, May 6-8, 1996 (pp. 35-37).
- Thacker, U., Pandey, M., & Rautaray, S. S. (2018). Review of Elasticsearch Performance Varying the Indexing Methods. In *Progress in Intelligent Computing Techniques: Theory, Practice, and Applications: Proceedings of ICACNI 2016, Volume 2* (pp. 3-8). Springer Singapore.
- Vásquez, A. C., Quispe, J. P., & Huayna, A. M. (2009). Procesamiento de lenguaje natural. *Revista de investigación de Sistemas e Informática*, 6(2), 45-54.
- Vidal, C. L., Schmal, R. F., Rivero, S., & Villarroel, R. H. (2012). Extensión del diagrama de secuencias uml (lenguaje de modelado unificado) para el modelado orientado a aspectos. *Información tecnológica*, 23(6), 51-62.
- Wang, J., Shou, L., Chen, K., & Chen, G. (2020, July). Pyramid: A layered model for nested named entity recognition. In *Proceedings of the 58th annual meeting of the association for computational linguistics* (pp. 5918-5928).
- Wang, Q., & Su, X. (2022). Research on named entity recognition methods in chinese forest disease texts. *Applied Sciences*, 12(8), 3885.
- Wei, X., Cui, X., Cheng, N., Wang, X., Zhang, X., Huang, S., ... & Han, W. (2023). Zero-shot information extraction via chatting with chatgpt. *arXiv preprint arXiv:2302.10205*.
- Yadav, H., Ghosh, S., Yu, Y., & Shah, R. R. (2020). End-to-end named entity recognition from english speech. *arXiv preprint arXiv:2005.11184*.
- Yadav, V., & Bethard, S. (2019). A survey on recent advances in named entity recognition from deep learning models. *arXiv preprint arXiv:1910.11470*.

ANEXOS

ANEXO I Ejemplo de datos etiquetados para el reentrenamiento

```
{
  "text": "El escritor Gabriel García Márquez nació en Aracataca, Colombia",
  "entities": [
    {'start': 12, 'end': 34, 'label': 'PERSON'},
    {'start': 44, 'end': 53, 'label': 'LOC'},
    {'start': 55, 'end': 63, 'label': 'LOC'}]
},
```

Fig 4.1 Ejemplo de datos etiquetados

ANEXO II Comparación de los modelos generadores de oraciones

Para generar un conjunto de datos de entrenamiento etiquetados con la entidad que el modelo de spaCy no reconozca se utiliza el modelo de llama2, aunque existen otros modelos como GPT-3.5 y GPT-4, los cuales se comparan a continuación.

GPT – 3.5

Es una herramienta poderosa en el campo del PLN, capaz de generar textos de alta calidad y con una gran cantidad de funcionalidades. Sin embargo, aún tiene limitaciones en términos de coherencia semántica y fiabilidad de la información proporcionada. Es la versión mejorada de GPT-3 lanzada en marzo de 2022 por OpenAI (Luzniak, 2023).

GPT-4

Sucesor de GPT-3.5, lanzado en marzo de 2023, GPT-4 es el modelo de lenguaje más reciente y capaz de OpenAI. Destaca por su capacidad para procesar texto e imágenes, lo que le permite utilizarse para una variedad de propósitos más amplios. GPT-4 está disponible en dos variantes: gpt-4-8K y gpt-4-32K (Luzniak, 2023).

Llama2

Llama2 es una versión de un modelo de lenguaje grande desarrollado por Facebook. Este modelo ha sido entrenado para generar texto en un estilo similar al de un humano, y es especialmente útil para tareas de generación de texto y procesamiento del lenguaje natural. Llama2 puede generar respuestas a las preguntas, continuar las historias y realizar una variedad de tareas de generación de texto (Luzniak, 2023). Sin embargo estos modelos presentan diferencias entre ellos tal como se muestra ([ver Tabla 4.1](#)) .

Tabla 4.1 Comparación entre Llama 2, GPT-3.5 y GPT-4 para generar oraciones

Modelos	Llama2	GPT-3.5	GPT-4
Parámetros	70.000 millones	154.000 millones – 175.000 millones	1.000.000 millones – 176.000.000 millones
Contenido Máximo	4096	4096 8001 16384	8192 32768
Modalidades	Texto solamente	Texto solamente	Texto e imagen
Precisión	68.9%	70%	86.4%
Complejidad	Baja	Alta	Alta
Velocidad	Rápido	Lento	Lento
Eficiencia	Más eficiente	Menos eficiente	Menos eficiente

- **Tamaño y Parámetros:** Llama 2 es significativamente más pequeño que GPT-3.5 y GPT-4. Este parámetro no determina necesariamente la calidad o rendimiento del modelo.
- **Contenido máximo:** Llama 2 tiene el mismo límite de tokens que la variante base de GPT-3.5-turbo, mientras que la variante base de GPT-4 tiene el doble. Si se requieren entradas y salidas más largas, GPT-4 sería la opción preferida.
- **Modalidades:** GPT-4 es el único capaz de procesar entradas visuales estáticas, mientras que Llama 2 y GPT-3.5 solo manejan texto.
- **Precisión y Complejidad de Tareas:** En pruebas de rendimiento, Llama 2 muestra resultados similares a GPT-3.5, pero GPT-4 supera a ambos en complejidad y creatividad. Sin embargo, en el modelo de Llama 2 se utiliza una técnica llamada Ghost Attention que mejora su habilidad para controlar el diálogo a lo largo de múltiples turnos.
- **Creatividad:** GPT-4 tiene el mejor nivel de creatividad entre los tres modelos, capaz de generar contenido poético y metáforas sofisticadas. Llama 2 y GPT-3.5 son menos creativos, sin embargo, resultan adecuados para tareas menos complejas y menos artísticas.
- **Velocidad y Eficiencia:** Debido a su tamaño reducido, Llama 2 es más rápido y eficiente que GPT-3.5 y GPT-4, lo que puede ser crítico para proyectos donde la velocidad es importante (Luzniak, 2023; Peeters, 2023).

Para lograr crear un set de datos de entrenamiento elegimos el modelo generador de oraciones de Llama2 debido a sus ventajas en la comparación realizada.

ANEXO III Análisis Económico

El método de punto de casos de uso consta de cuatro etapas, en la que se desarrolla los siguientes cálculos:

Ecuación 3.1: Cálculo de los Puntos de Historias de Usuarios sin ajustar

$$\text{UUCP} = \text{UAW} + \text{UUCW}$$

En la cual:

- **UUCP:** Puntos de Historias de Usuario sin ajustar.
- **UAW:** Factor de Peso de los Actores sin ajustar.
- **UUCW:** Factor de Peso de Historias de Usuarios sin ajustar.

Factor de Peso de los Actores sin ajustar (UAW)

Este valor se calcula mediante un análisis de la cantidad de Actores presentes en el sistema y la complejidad de cada uno de ellos. En la tabla 4.1 se presenta el Facto de Peso de los Actores sin ajustar.

Tabla 4.1: Factor de peso de los actores sin ajustar

Tipo	Descripción	Peso	Cantidad por peso
Simple	Otro sistema que interactúa mediante una interfaz de programación de aplicaciones. (API)	2	1*2
Medio	Otro sistema que interactúa mediante un protocolo o una persona interactuando con una interfaz basada en texto.	2	2*2
Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica. (GUI)	3	1*3
Total			2+4+3 = 9

Total, de Puntos de Actores sin ajustar (UAW): 9

Cálculo de UUCW

Para calcular el UUCW(unadjusted Use Case Weight) para las historias de usuario, primero se necesita determinar la complejidad de cada una de ellas según el número de transacciones que involucran. Luego se multiplica este valor por el peso asignado a cada tipo de historia de usuario. En la tabla 4.2 se muestra la clasificación de complejidad y cálculo de peso.

Tabla 3.2 Peso de las Historias de Usuario sin Ajustar

Tipo	Descripción	Peso	Cantidad por peso
Simple	La HU contiene de 1 a 3 transacciones.	5	2*5
Medio	La HU contiene de 4 a 7 transacciones.	10	2*10
Complejo	La HU contiene más de 8 transacciones	15	1*15
Total			10+20+15 = 55

Cálculo de los Puntos de Historias de Usuarios ajustadas

Una vez que se tienen los Puntos de Historias de Usuarios, se debe ajustar este valor como se muestra en la ecuación 3.2.

Ecuación 3.2: Calculo de los Puntos de Historias de Usuarios ajustadas.

$$UCP = UUCP * TCF * EF$$

Donde:

- **UCP:** Puntos de Historias de Usuarios ajustados.
- **UUCP:** Puntos de Historias de Usuarios sin ajustar.
- **TCF:** Factor de complejidad técnica.
- **EF:** Factor de ambiente.

Factor de complejidad técnica (TCF).

Este coeficiente se calcula mediante la cuantificación de un conjunto de factores que determinan la complejidad técnica del sistema. Cada uno de los factores se cuantifica con un valor de 0 a 5, donde 0 significa un aporte irrelevante o nulo y 5 un aporte muy importante. En la siguiente tabla 4.3 se muestra factor de complejidad técnica con su significado y el peso de cada uno de estos factores.

Tabla 4.3 Factor de complejidad técnica

Factor	Descripción	Peso	Valor	(Peso-i * Valor-i)
T1	Sistema distribuido	2	2	4
T2	Rendimiento o tiempo de respuesta	1	5	5
T3	Eficiencia del usuario final	1	2	2
T4	Procesamiento interno complejo	1	4	4
T5	El código debe ser reutilizable	1	3	3
T6	Facilidad de instalación	0.5	2	1
T7	Facilidad de uso	0.5	5	2.5
T8	Portabilidad	2	3	6
T9	Facilidad de cambio	1	5	5
T10	Concurrencia	1	4	4

T11	Incluye objetivos especiales de seguridad	1	5	5
T12	Acceso directo a terceras partes	1	2	2
T13	Se requieren facilidades especiales de entrenamiento a los usuarios	1	2	2
Total				45.5

Para calcular TCF: Factor de complejidad técnica se muestra la ecuación 3.3 del factor de complejidad técnica.

Ecuación 3.3: Cálculo del Factor de complejidad técnica

$$\text{TCF} = 0.6 + 0.01 * \Sigma (\text{Peso}_i * \text{Valor}_i)$$

$$\text{TCF} = 0.6 + 0.01 * 45.5$$

$$\text{TCF} = 1.055$$

Factor Ambiente (EF)

El factor ambiente está relacionado con las habilidades y entrenamiento del grupo de desarrollo. Cada factor se cuantifica con un valor desde 0 (aporte irrelevante) hasta 5 (aporte muy relevante). En la siguiente tabla 4.4 se muestra factor de ambiente con su significado y el peso de cada uno de estos factores.

Tabla 4.4: Factor Ambiente

Factor	Descripción	Peso	Valor	(Peso-i * Valor-i)
E1	Familiaridad con el modelo de proyecto utilizado	1.5	3.5	5.25
E2	Experiencia en la aplicación	0.5	4	2
E3	Experiencia en orientación a objetos	1	3	3
E4	Capacidad del analista líder	0.5	4	2
E5	Motivación	1	4.5	4.5
E6	Estabilidad de los requerimientos	2	4	8
E7	Personal part-time	-1	0	0
E8	Dificultad del lenguaje de programación	-1	2	-2
Total				22.75

Para Calcular EF: Factor de ambiente se muestra la ecuación 3.4 cálculo del factor ambiente.

Ecuación 3.4: Cálculo del Factor de ambiente

$$\text{EF} = 1.4 - 0.03 * \Sigma (\text{Peso}_i * \text{Valor}_i)$$

$$\text{EF} = 1.4 - 0.03 * 22.75$$

$$\text{EF} = 0.7175$$

$$\text{Luego: UCP} = \text{UUCP} * \text{TCF} * \text{EF}$$

$$UCP = 32 * 1.03 * 0.7175$$

$$UCP = 23.76488$$

Estimación de esfuerzo a través de los Puntos de Historias de Usuarios

Ecuación 3.5: Esfuerzo estimado en horas hombres

$$E = UCP * CF$$

Donde:

- **E:** Esfuerzo estimado en horas hombres
- **UCP:** Punto de historias de usuarios ajustadas
- **CF:** Factor de conversión

Para obtener el factor de conversión (CF) se cuentan cuántos valores de los que afectan el factor ambiente (E1 a E6) están por debajo de la media (**<3**), y los que están por encima (**>3**) para los restantes (E7 a E8). Si el total (nos da **0**) es 2 o menos se utiliza el factor de conversión 20 Horas- Hombre / Punto de historias de usuarios. Si el total es 3 o 4 se utiliza el factor de conversión 28 Horas-Hombre / Punto de historias de usuarios. Si el total es mayor o igual que 5 se recomienda efectuar cambios en el proyecto ya que se considera que el riesgo de fracaso es demasiado alto. En este caso:

$$CF = 20 \text{ Horas-hombres} / \text{Puntos de historias de usuarios}$$

Luego

$$E = 23 * 20 \text{ horas-hombre}$$

$$E = 474 \text{ horas-hombre}$$

En la siguiente tabla (ver Tabla 4.5) se muestra distribución del esfuerzo por etapas.

Tabla 4.5 Distribución del esfuerzo por etapas

Actividad	% Esfuerzo	Valor esfuerzo
Planificación	5	50
Diseño	20	80
Codificación	45	260
Prueba	15	44
Sobrecarga	15	40
Total	100	474

Una vez estimado el tiempo de desarrollo del proyecto y conociendo la cantidad de desarrolladores y el pago que recibe cada uno de estos se puede llevar a cabo una estimación del costo total del proyecto referidos a los recursos humanos.

K: Coeficiente que tiene en cuenta los costos indirectos (1,5 y 2,0).

THP: Tarifa Horaria Promedio. El salario promedio mensual de los trabajadores en este caso es de \$12 000 CUP dividido entre 176h es igual a \$67.39 por hora.

176 horas (horas de trabajo para un 1 mes, esto se toma a razón de 24 días, ya que no se cuentan los fines de semana ni sábados cortos).

Tiempo = $474 \text{ horas} / 176 \approx$ equivalente a 2 meses y 69 días, este es el tiempo que tomaría desarrollar el proyecto empleando una sola persona.

2 meses y 69 días a razón de 24 días laborables por mes, representan 109 días. En nuestro caso como empleamos 3 trabajadores para el desarrollo del proyecto, el tiempo para su culminación quedará reducido a 36 días aproximadamente.

Entonces el costo total del proyecto:

$$C = E(\text{Total}) * K * \text{THP}$$

$$C = 474 * 1.5 * [3 * (12000/176)] = \underline{\underline{\$ 145\,431}}$$

Para el salario a los trabajadores se investigó como se mueve en diferentes organizaciones o sucursales en Santiago de Cuba de diferente sector ya sea estatal o privado, dada las nuevas regulaciones y tasas de cambio. Los datos se obtuvieron a través de trabajadores de las entidades y por anuncios laborales. (Ver Tabla 4.6)

Tabla 4.6 Situación actual de pago

Organización (Sector)	Pago/Mensual (Moneda Nacional)
XETID (Estatad)	9 000 CUP
DATYS (Estatad)	9 000 CUP
DESOFT (Estatad)	9 000 CUP
MYPIMES (Privado)	25 000 CUP
Freelancer o persona autónoma (privado)	700 MLC a 120 CUP (tasa oficial actual) son 84 000 CUP

Anexo IV Datos generados con la entidad “teamacere” de tipo ORG

```
{
  "doc1": {
    "text": "El equipo de pelota teamacere se encuentra en una intensa práctica para el próximo partido.",
    "entities": [
      {
        "name": "teamacere",
        "start": 20,
        "end": 29,
        "label": "ORG"
      }
    ]
  },
  "doc2": {
    "text": "La líder del equipo de pelota teamacere, Ana, está muy motivada para ganar el campeonato.",
    "entities": [
      {
        "name": "teamacere",
        "start": 30,
        "end": 39,
        "label": "ORG"
      },
      {
        "name": "Ana",
        "start": 41,
        "end": 44,
        "label": "PERSON"
      }
    ]
  }
}
```

Fig 4.2 Ejemplos de datos generados

Anexo V Aval del sistema por la empresa DATYS

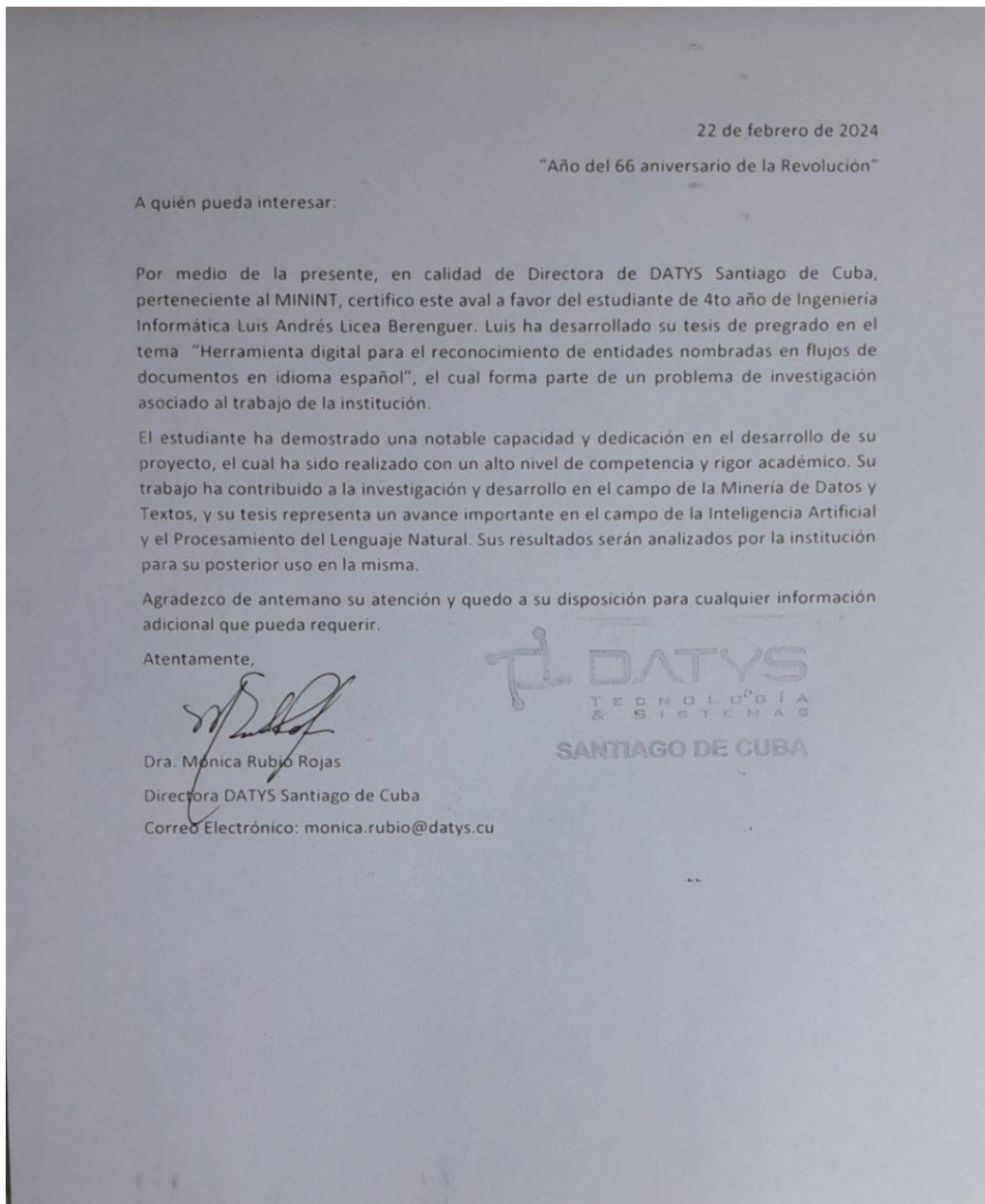


Fig 4.3 Aval del sistema por la empresa DATYS