

Universidad de Oriente

Sede “Julio Antonio Mella”

Facultad de Ingeniería en Telecomunicaciones, Informática y Biomédica

Trabajo de Diploma

En opción al título de Ingeniera en Informática

**Título:** “Sistema informático para el reconocimiento de entidades nombradas en el procesamiento de documentos digitales en la empresa DATYS”.

**Autor:** Luis Andrés Licea Berenguer.

**Tutores:** Dr. C. Dionis López Ramos.

Ms. C. Jose Ernaldo Cruzata Ferrer

2024

Santiago de Cuba,

“Año 66 de la Revolución”

Resumen

El reconocimiento de entidades nombradas es crucial en el procesamiento de datos no estructurados, permitiendo identificar y comprender información clave como personas, lugares y organizaciones. Este estudio se centra en desarrollar un sistema para el reconocimiento de entidades nombradas utilizando la biblioteca SpaCy en Python, que no solo identifica entidades nombradas, sino que también permite su reentrenamiento para adaptarse a nuevas entidades desconocidas. Se emplean técnicas de procesamiento de lenguaje natural y aprendizaje automático, utilizando un conjunto de datos etiquetados para reentrenar el modelo. El mismo demuestra alta precisión y exhaustividad en el reconocimiento de entidades, lo que permite una adaptación dinámica a nuevas entidades. Para el reentrenamiento, se crea un conjunto de datos con Llama2, aportando un valor significativo al campo del procesamiento de lenguaje natural.

**Palabras clave:** Reconocimiento de entidades nombradas, SpaCy, Python, reentrenamiento, Llama2.

*ABSTRACT*

Named Entity Recognition (NER) is crucial in the processing of unstructured data, allowing the identification and understanding of key information such as people, places, and organizations. This study focuses on developing a system for NER using the SpaCy library in Python, which not only identifies named entities but also allows for their retraining to adapt to new, unknown entities. Advanced techniques of natural language processing and machine learning are employed, using a labeled dataset to retrain the model. The model demonstrates high precision and exhaustiveness in entity recognition, enabling dynamic adaptation to new entities. For retraining, a dataset is created with Llama2, contributing significantly to the field of natural language processing.

**Keywords*:***Named Entity Recognition, SpaCy, Python, retraining, Llama2.

Tabla de contenido

[**INTRODUCCIÓN** 1](#_Toc164009632)

[**CAPITULO 1 .** **MARCO TEÓRICO** 4](#_Toc164009633)

[1.1 Procesamiento del Lenguaje Natural 4](#_Toc164009634)

[1.2 Reconocimiento de entidades nombradas (NER) 4](#_Toc164009635)

[1.3 Estado del Arte de los marcos de procesamiento del lenguaje natural para la detección de entidades 5](#_Toc164009636)

[1.3.1 Google NLP 6](#_Toc164009637)

[1.3.2 ChatGPT 6](#_Toc164009638)

[1.3.3 IBM Watson Discovery 7](#_Toc164009639)

[1.4 Herramientas, Lenguaje de programación y Tecnologías 8](#_Toc164009640)

[1.4.1 Python 8](#_Toc164009641)

[1.4.2 Visual Studio Code 8](#_Toc164009642)

[1.4.3 JSON 8](#_Toc164009643)

[1.4.4 Elasticsearch 9](#_Toc164009644)

[1.4.5 Git 9](#_Toc164009645)

[1.4.6 Modelos para el reconocimiento de entidades nombradas 10](#_Toc164009646)

[1.4.7 Comparación de los modelos de reconocimiento de entidades 11](#_Toc164009647)

[1.4.8 Modelos generadores de oraciones 12](#_Toc164009648)

[1.4.9 React 13](#_Toc164009649)

[1.4.10 FastAPI 14](#_Toc164009650)

[1.5 Metodología XP 14](#_Toc164009651)

[1.6 Conclusiones del capítulo 14](#_Toc164009652)

[**CAPITULO 2 .** **ORGANIZACIÓN Y DISEÑO** 16](#_Toc164009653)

[2.1 Propuesta del sistema 16](#_Toc164009654)

[2.2 Arquitectura del sistema creado 16](#_Toc164009655)

[2.3 Requisitos Funcionales 17](#_Toc164009656)

[2.4 Requisitos no funcionales del sistema 18](#_Toc164009657)

[2.5 Historias Técnicas 19](#_Toc164009658)

[2.5.1 Usabilidad 19](#_Toc164009659)

[2.5.2 Confiabilidad 19](#_Toc164009660)

[2.5.3 Seguridad 19](#_Toc164009661)

[2.6 Diseño de la base de datos 19](#_Toc164009662)

[2.7 Historias de Usuario 21](#_Toc164009663)

[2.8 Usuarios del Sistema 28](#_Toc164009664)

[2.9 Diagrama de Secuencia 28](#_Toc164009665)

[2.10 Diagrama de Actividades 30](#_Toc164009666)

[2.11 Diseño de la Interfaz 31](#_Toc164009667)

[2.12 Conclusiones del capítulo 32](#_Toc164009668)

[CAPITULO 3 . IMPLEMENTACIÓN Y PRUEBA 33](#_Toc164009669)

[3.1 Instalación de los requisitos para el funcionamiento del sistema 33](#_Toc164009670)

[3.2 Algoritmos Importantes 33](#_Toc164009671)

[3.2.1 Insertar Usuario 33](#_Toc164009672)

[3.2.2 Autenticación del Usuario 34](#_Toc164009673)

[3.2.3 Reconocimiento de Entidades Nombradas a un índice de Elasticsearch 35](#_Toc164009674)

[3.2.4 Salvar resultados del reconocimiento en el índice seleccionado 37](#_Toc164009675)

[3.2.5 Generar datos de entrenamiento 38](#_Toc164009676)

[3.2.6 Reentrenar el modelo de Spacy evitando el olvido catastrófico 40](#_Toc164009677)

[3.3 Medidas de Desempeño 42](#_Toc164009678)

[3.4 Análisis económico del costo de producción del sistema. 44](#_Toc164009679)

[3.5 Estimación de costo y tiempo 44](#_Toc164009680)

[3.6 Pruebas al sistema 45](#_Toc164009681)

[3.7 Resultados obtenidos 50](#_Toc164009682)

[3.8 Conclusiones del capítulo 51](#_Toc164009683)

[**CONCLUSIONES Y RECOMENDACIONES** 52](#_Toc164009684)

[**REFERENCIAS BIBLIOGRÁFICAS** 53](#_Toc164009685)

[**ANEXOS** 58](#_Toc164009686)

**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 (NER, según sus siglas en inglés) ha crecido significativamente, siendo un componente esencial en el procesamiento del lenguaje natural (NLP, según sus siglas en inglés) y la inteligencia artificial (IA) (Carvalho,2023).

Se ha demostrado que más del 70% de las consultas contra motores de búsqueda modernos contienen una entidad nombrada, y se ha sugerido que más del 30% de las palabras portadoras de contenido en textos de noticias corresponden a nombres propios. Por lo tanto, el indexado de documentos basado en entidades es deseable. El NER es altamente beneficioso en la extracción de información, o la actividad de encontrar datos dentro de grandes volúmenes de textos no estructurados. La extracción de hechos salientes sobre tipos predefinidos de entidades en textos libres es en efecto una parte esencial de la respuesta a preguntas y el monitoreo de medios (Bernabeu, 2022; Nasar, 2021). 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, esto 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).

Debido a estas circunstancias el desarrollo de herramientas que permitan hacer el reconocimiento de entidades nombradas y almacenarlas para su posterior estudio se ha vuelto necesario y de suma importancia. Empresas e investigadores han desarrollado sistemas, modelos para el reconocimiento de entidades nombradas 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 en español no ocurre de igual manera, debido a que la mayoría de los desarrolladores son de habla inglesa y se requiere un gran volumen de datos etiquetados para entrenar los modelos de PLN para el NER (Shelar, 2020; Yadav, 2019).

**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 en Elasticsearch.

**Objeto de investigación:**

Modelos de procesamiento del lenguaje natural.

**Campo de acción:**

Reconocimiento de entidades nombradas en español.

**Objetivo general:**

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

**Objetivos específicos:**

1. Investigar las técnicas actuales en el campo del procesamiento del lenguaje natural para la detección de entidades nombradas.
2. Diseñar una propuesta de sistema 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. Realizar evaluaciones para verificar las medidas de desempeño del sistema desarrollado en idioma español.

**Hipótesis:**

Si se desarrolla un sistema de procesamiento de lenguaje natural para el reconocimiento de entidades nombradas en español, se podrá contribuir a disminuir las limitaciones actuales en el reconocimiento de las mismas en grandes volúmenes de datos en Elasticsearch.

**Métodos de investigación:**

* Método histórico-lógico: se aplicó al realizar el análisis de la existencia de otros sistemas que den solución al problema en cuestión.
* 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.

**Aportes de la Investigación:**

* Se contará con un sistema que reconozca las entidades nombradas en español en grandes volúmenes de datos en Elasticsearch.
* El sistema tendrá la capacidad de reentrenamiento de forma supervisada 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 la entidad que no reconozca el modelo de procesamiento del lenguaje natural.
* EL modelo diseñado tiene la capacidad de retroceder en caso de perder conocimiento al realizar el reentrenamiento.

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.

## Procesamiento del Lenguaje Natural

El procesamiento de lenguaje natural (NLP) es una tecnología de machine learning que brinda a las computadoras 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 NLP 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).

Dentro del campo del procesamiento del lenguaje natural se pueden diferenciar dos aproximaciones principales: procesamiento estadístico del lenguaje natural, y el procesamiento lingüístico del lenguaje natural, en inglés conocido como rule-based Natural Language Processing. Estos se basan en modelos formales del conocimiento lingüístico, como: máquinas de estado, sistemas de reglas, lógica, o modelos probabilísticos. Actualmente se ha demostrado que no todos los sistemas NLP toleran adaptarse a teorías lingüísticas, ni si quiera los de procesamiento simbólico (Moreira, 2021).

## Reconocimiento de entidades nombradas (NER)

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. NER 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 NER, y varios eventos científicos que dedican mucho esfuerzo a este tema (Wang, 2020; Wang, 2022; Missaoui, 2019). El reconocimiento de las entidades nombradas varía mucho en función del idioma en el que han sido entrenadas.

Cuando se realiza el análisis histórico lógico de los sistemas para el reconocimiento de entidades nombradas se evidencia como la evolución de las tecnologías influyó en los avances en el NLP.

**Etapa inicial**

En una primera etapa del desarrollo de los sistemas NER, 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.

**Aparición de Tecnologías de Aprendizaje Automático**

Con el avance de las tecnologías de aprendizaje automático, se introdujeron métodos basados en machine learning para el NER. Este enfoque implicaba la anotación de datos para entrenar el modelo, permitiendo que el modelo NER pudiera aprender a identificar y clasificar entidades nombradas a partir de ejemplos previamente etiquetados. Este método representó un gran avance en la precisión y eficiencia del reconocimiento de entidades nombradas.

**Implementación de Deep Learning**

El uso de Deep learning para el NER marcó una etapa significativa en la evolución de estas tecnologías. Los modelos basados en Deep learning 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. Este enfoque ha demostrado ser particularmente efectivo para el español, un idioma con una rica morfología y complejidades gramaticales que pueden ser desafiantes para los modelos de NLP.

**Herramientas y Bibliotecas Actuales**

Hoy en día, existen diversas herramientas y bibliotecas disponibles para implementar NER, como spaCy y DocumentCloud (que utiliza Google Cloud NLP) entre otras. Estas herramientas permiten a los desarrolladores y analistas de datos aplicar técnicas de NER a sus proyectos, facilitando la extracción de información valiosa de grandes volúmenes de textos (Goyal, 2018; Liu, 2022).

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

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

### Google NLP

Google Cloud NLP es una interfaz de programación de aplicaciones (API, según sus siglas en inglés) de Inteligencia Artificial que se centra en el NLP. La misma es parte de la suite de productos digitales de Google, que están en constante crecimiento. Esta se enfoca en la creación de contenido que se asemeje lo más posible a lo que un humano podría crear, proporcionando un análisis de texto a un nivel superior que fluye de manera más clara que lo que muchos programas de IA actuales son capaces de proporcionar. Esta API es la opción más compleja de Google, lo que significa que se dirige a empresas más grandes que su producto básico AutoML (Gritta, 2020; Bisong, 2019).

Entre las ventajas de Google Cloud NLP, destaca su reconocimiento de entidades nombradas multilingüe y su eficiencia en comparación con la contratación de humanos con habilidades de idioma para manejar proyectos similares. Esta API ofrece servicios en 10 idiomas diferentes y proporciona características como la resumación de texto, la generación de texto, la traducción, la detección de idioma, la tokenización, la lematización y el análisis de sentimientos.

Sin embargo, Google Cloud NLP se observan algunas limitaciones en los tipos de información que es capaz de manejar. Muchos usuarios han encontrado que esta API necesita una formación significativamente mayor para ser capaz de realizar menos tareas que muchas de las otras opciones en el mercado. Por otra parte, es una opción más costosa que muchas de sus competidoras, aunque proporciona créditos gratuitos a nuevos clientes, el costo de este programa puede no valer la pena una vez que necesita pasar a una suscripción continua (Bisong, 2019).

### ChatGPT

ChatGPT es, probablemente, el referente inmediato para millones de personas sobre lo que es un sistema de procesamiento natural de lenguaje. Esta tecnología está equipada con un potente motor de análisis de lenguaje conocido como GPT-4. Esta herramienta ganó popularidad a principios del 2023 ya que es una de las pioneras en poner al servicio de las personas una herramienta capaz de responder a demandas concretas con un alto nivel de éxito. Además, esta plataforma lanzó su propia API que puede ser integrada en programas para la creación de chatbots mucho más potentes, útiles y precisos con las respuestas que ofrecen a sus usuarios (Diego, 2023; Wei, 2023).

Entre las ventajas de su uso se encuentra que es menos costoso en términos de tiempo que la contratación de personal humano para la tarea, los tiempos de respuesta son bastante rápidos. Además, existen modelos de aprendizaje automático preentrenados que facilitan la implementación de diferentes aplicaciones de NLP, lo que reduce el tiempo y esfuerzo necesarios para desarrollar soluciones desde cero.

Entre las desventajas de su uso se encuentran que, si se necesitara desarrollar un nuevo modelo sin utilizar un modelo preentrenado, el proceso de formación puede llevar mucho tiempo, lo que puede ser un obstáculo para la implementación rápida de soluciones. No es 100% fiable la mayoría de las veces comete errores en cuanto al tipo de entidad, la posición inicial y final de la misma, su uso requeriría una gestión cuidadosa y verificación de la información (Diego, 2023; De Kok, 2023; Lund, 2023).

### IBM Watson Discovery

Watson es la plataforma de IA creada por IBM que tiene como objetivo acercar el uso de datos a los negocios de hoy. De entre sus herramientas, Discovery representa el mayor avance en aplicación del procesamiento de lenguaje natural.

Esta herramienta es un procesador de documentos que evalúa, interpreta y extrae información de archivos de texto con el fin de hacer más simple la consulta y lectura de materiales escritos. Mediante la creación de etiquetas, el sistema aprende a detectar información de relevancia, así como a distinguir entre tablas, títulos y anotaciones (Chen, 2016; Martin, 2018).

Posee varias ventajas para el NER entre las que se encuentran su capacidad de procesamiento avanzado para analizar textos complejos, lo que le permite una extracción de información más precisa y detallada. Puede analizar textos en una variedad de formatos, incluyendo páginas web y redes sociales lo que lo hace crucial para la extracción de información de diversas fuentes.

Entre las desventajas de Watson se encuentran que su implementación y mantenimiento pueden ser costosos y requieren un nivel de conocimiento técnico considerable. La efectividad depende en gran medida de la calidad y la estructura de los datos de entrada. Al procesar grandes volúmenes de datos, especialmente en entornos empresariales, puede existir la ocurrencia de perdida de información, privacidad y seguridad de sus datos (Martin, 2018).

## Herramientas, Lenguaje de programación y Tecnologías

Las herramientas son objetos elaborados a fin de facilitar la realización de una tarea. Se diseñan y fabrican para cumplir uno o más propósitos específicos. Un lenguaje de programación es un lenguaje formal que especifica una serie de instrucciones para que una computadora produzca diversas clases de datos. Los lenguajes de programación pueden usarse para crear programas que pongan en práctica algoritmos específicos los cuales controlan el comportamiento físico y lógico de una computadora. Por su parte, la tecnología es la ciencia aplicada a la resolución de problemas concretos. Constituye un conjunto de conocimientos científicamente ordenados, que permiten diseñar y crear bienes o servicios que facilitan la adaptación al medio ambiente y la satisfacción de las necesidades humana.

Entre las herramientas empleadas para el desarrollo del sistema se encuentra el Visual Studio Code como editor de código, Git como controlador de versiones.

En el caso de los lenguajes de programación una de los más empleados en el desarrollo de sistemas de IA se encuentra Python y para la creación de interfaces React. La base de datos es Elasticsearch orientada a documentos JSON, el modelo utilizado para el reconocimiento de entidades es Spacy y el generador de datos de entrenamiento es el modelo de Llama2.

### Python

Python 3.10.8 es un lenguaje de programación muy popular utilizado en una amplia gama de aplicaciones, desde el desarrollo web hasta la ciencia de datos y el machine learning. Los desarrolladores prefieren Python por su eficiencia, facilidad de aprendizaje y capacidad para ejecutarse en múltiples plataformas. Además, este software es gratuito, se integra fácilmente en diferentes sistemas y acelera el proceso de desarrollo (Challenger-Pérez, 2014; Mirjalili, 2020).

### Visual Studio Code

Visual Studio Code (VS Code) 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).

### JSON

JSON (JavaScript Object Notation) es un formato de texto ligero utilizado para el intercambio de datos, derivado de la sintaxis de JavaScript. A diferencia de la programación, JSON se enfoca en el acceso, almacenamiento e intercambio de datos. Es una alternativa a XML, con una sintaxis más simple y legible tanto para humanos como para máquinas (Ihrig, 2013).

Son útiles para intercambiar datos entre diferentes dispositivos digitales y se emplean ampliamente en el diseño web, aplicaciones móviles, programas computacionales y transferencia de documentos (Nolan, 2014).

### Elasticsearch

Elasticsearch es un motor de búsqueda y análisis distribuido construido sobre Apache Lucene. Proporciona soporte para varios lenguajes de programación como: Java, Python, PHP, JavaScript, Node.js, Ruby, entre otros. Permite enviar datos en forma de documentos JSON a través de su API. Esto le permite procesar grandes volúmenes de datos en paralelo y encontrar rápidamente las mejores coincidencias para las consultas. Viene integrado con Kibana, una herramienta popular de visualización y reporte, y ofrece integración con Beats y Logstash, facilitando la transformación de datos de origen y su carga en el clúster de Elasticsearch. También se pueden utilizar varios plugins de código abierto para añadir funcionalidades ricas a las aplicaciones (Dhulavvagol, 2020).

Elasticsearch ofrece APIs basadas en REST, una interfaz HTTP simple y utiliza documentos JSON sin esquema, lo que facilita el inicio y la construcción rápida de aplicaciones para diversos casos de uso. Las operaciones como la lectura o escritura de datos, suelen completarse en menos de un segundo, lo que permite utilizarla para casos de uso de tiempo casi real, como el monitoreo de aplicaciones y la detección de anomalías. Además, es altamente escalable horizontalmente, facilitando la posibilidad de añadir más capacidad y confiabilidad a los nodos y clústeres (Shaik, 2017).

Sus índices son espacios lógicos que albergan colecciones de documentos. Cada documento está compuesto por campos clave-valor, y pueden dividirse en fragmentos para una distribución eficiente en entornos distribuidos. A diferencia de las bases de datos relacionales, no requieren un esquema definido previamente, lo que facilita la importación de datos y la adaptación a campos inesperados (Thacker, 2018; Dhulavvogol, 2020).

### Git

Git 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).

### Modelos para el reconocimiento de entidades nombradas

En la plataforma de Hugging Face encontramos diversos modelos de NLP en el cual poseen funcionalidades para realizar el NER entre los más empleados en el idioma español se encuentra Spacy, Bert, Flair, NLTK y Stanford.

#### Spacy

SpaCy es una biblioteca de NLP 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 NLP que requieren un análisis rápido y preciso del texto (Partalidou, 2019).

#### Bert

BERT (Bidirectional Encoder Representations from Transformers) es un modelo de lenguaje preentrenado desarrollado por Google que utiliza la arquitectura Transformer. BERT fue diseñado para comprender el contexto completo de una palabra en una frase considerando todas las palabras antes y después de ella. Este enfoque bidireccional permite que BERT genere representaciones de palabras muy ricas y útiles para una variedad de tareas de NLP (Hakala, 2019).

#### Flair

Flair es un marco de trabajo para el NLP 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 NLP. Además, personaliza el modelo mediante el ajuste fino (fine-tuning) para adaptarlo a tareas específicas (Yadav, 2020).

#### NLTK

NLTK, o Natural Language Toolkit, es una biblioteca gratuita y de código abierto para el procesamiento avanzado de NLP en Python. Se utiliza para simplificar los datos textuales y obtener información profunda de los mensajes de entrada. NLTK proporciona interfaces fáciles de usar para más de 50 corpus y recursos léxicos, como WordNet, junto con una suite de bibliotecas de procesamiento de texto para clasificación, tokenización, etiquetado, análisis y razonamiento semántico. Es reconocido por su amplitud en algoritmos y la calidad de sus recursos, lo que lo convierte en una herramienta poderosa para la educación y la investigación en NLP. Sin embargo, su curva de aprendizaje puede ser pronunciada y, en ocasiones, puede no coincidir con las demandas de uso en producción del mundo real debido a su ritmo de desarrollo más lento (Yogish, 2019) .

#### Stanford

Stanford NLP (Natural Language Processing) es un conjunto de herramientas de software desarrolladas por el grupo de Stanford NLP para resolver problemas computacionales de lenguaje natural. Estas herramientas incluyen procesamiento estadístico de NLP, aprendizaje profundo de NLP y herramientas basadas en reglas para tareas importantes de lingüística computacional (Sulaiman, 2017).

### Comparación de los modelos de reconocimiento de entidades

En la presente investigación se selecciona la biblioteca de Spacy debido a su precisión del 90% en el NER para el idioma español, lo que es relativamente alto en comparación con los otros modelos mencionados [(ver Tabla 1.1)](#comparacion_modelo_ner). 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 modelo (Shelar, 2020; Schmitt, 2019).

**Tabla 1.1 Comparación de los modelos de reconocimiento de entidades**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Modelo** | **Dataset** | **Peso del modelo** | **Componentes** | **Precisión** |
| **Spacy** | OntoNotes 5.0 | 541MB | [tok2vec](https://spacy.io/api/tok2vec), [morphologizer](https://spacy.io/api/morphologizer), [parser](https://spacy.io/api/dependencyparser), [senter](https://spacy.io/api/sentencerecognizer), [attribute\_ruler](https://spacy.io/api/attributeruler), [lemmatizer](https://spacy.io/api/lemmatizer), [ner](https://spacy.io/api/entityrecognizer) | 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% |
| **NLTK** | Personalizable | N/A | Reglas basadas, etiquetado de POS, tokenización | N/A |
| **Standford** | OntoNotes 5.0 | 635MB | Tokenización, etiquetado de POS, análisis de dependencia, reconocimiento de entidades | N/A |

### Modelos generadores de oraciones

En la realización del reentrenamiento del modelo de Spacy es necesario contar con un set de datos etiquetados en el cual este presente la entidad que no reconoce el modelo. Para generar un conjunto de oraciones de ejemplo en el cual debe estar presente dicha entidad 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 NLP, 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 1.2)](#comparacion_modelo_oracion_gen) .

**Tabla 1.2** **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.

### React

React 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. React introduce JSX, que permite combinar la lógica de JavaScript con la de la interfaz de usuario, proporcionando una sintaxis similar a XML que es fácil de leer y escribir (Chen, 2019).

Algunas características clave de React incluyen la renderizacióndel lado del cliente y alto rendimiento gracias al DOM virtual de React, esto puede hacerse de manera más eficiente, lo que resulta en tiempos de carga más rápidos y una mejor experiencia para el usuario.

### FastAPI

FastAPI 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).

Algunas características clave de FastAPI incluyen:

* **Alto rendimiento**: Es comparable en rendimiento con NodeJS y Go, gracias a su uso de Starlette para el manejo HTTP y Pydantic para la validación de datos.
* **Rápido para codificar**: Permite aumentar significativamente la velocidad de desarrollo de las características.
* **Reducción de errores**: Ayuda a reducir aproximadamente el 40% de los errores inducidos por el humano (desarrollador).
* **Basado en estándares**: Compatible con los estándares abiertos para APIs, OpenAPI y JSON Schema.

## 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).

## Conclusiones del capítulo

En el área del NER existen actualmente diversos sistemas que se utilizan en el ámbito comercial y científico. Estos sistemas se desarrollan en combinación de técnicas de PLN para el reconocimiento de estas entidades. Los mismos muestran una baja precisión para los textos en español, lo que es abordado desde las funcionalidades de diversos modelos. Se evidencio que 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. Lo que evidencia la necesidad de generar datos de entrenamiento etiquetados para el posterior reentrenamiento, donde el modelo de Llama2 es seleccionada debido a sus ventajas tanto de velocidad, como de eficiencia.

1. **ORGANIZACIÓN Y DISEÑO**

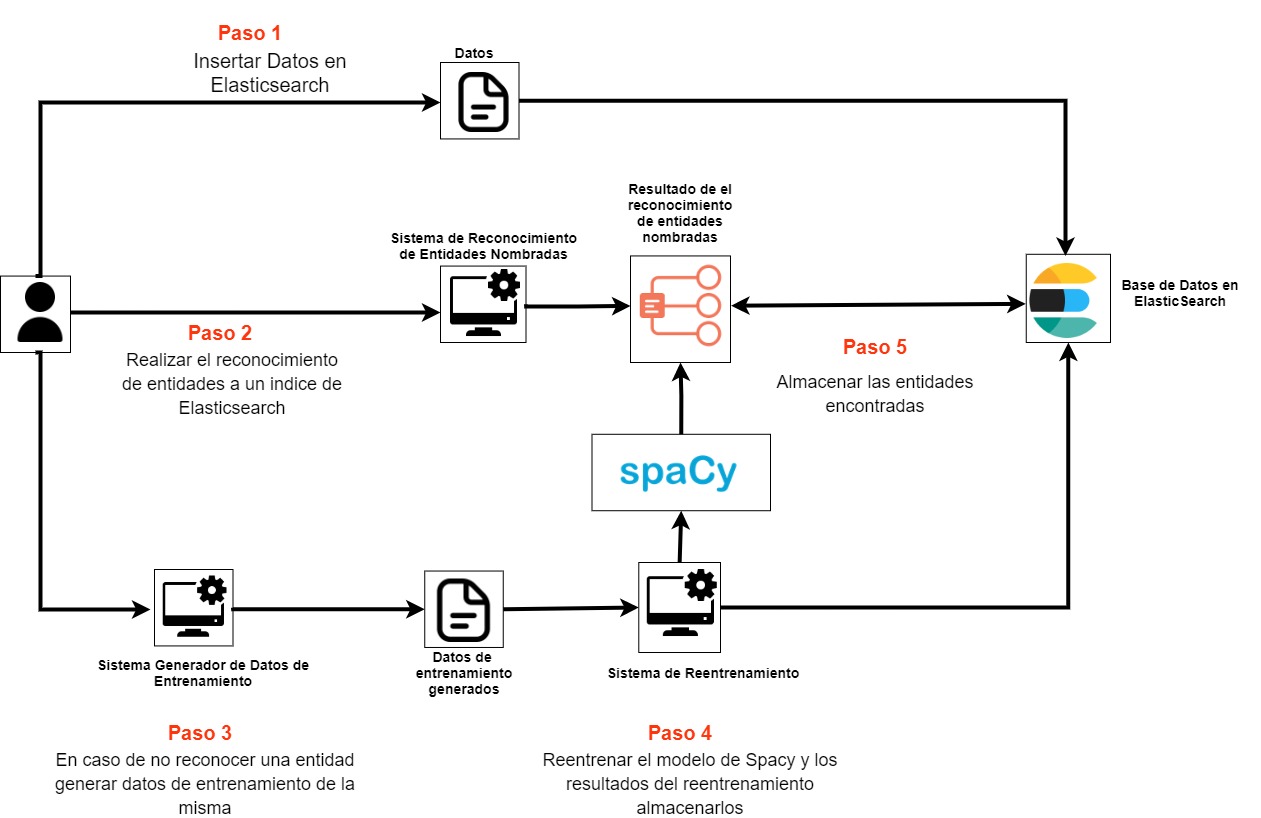
En el siguiente capítulo, se describe el proceso de organización y diseño del sistema, aplicando diversas técnicas y herramientas. Se muestran los diferentes diagramas utilizados para la modelación y desarrollo del sistema de NER para el idioma español en grandes volúmenes de datos, actuando como una guía para futuros proyectos similares.

## Propuesta del sistema

Se propone la creación de un sistema informático que utilice las bibliotecas del marco de trabajo Spacy y el Gran Modelo de Lenguaje (LLM, por sus siglas en inglés) para la generación de oraciones Llama2. Este sistema permitiría el reconocimiento y edición de entidades nombradas. Los usuarios del centro u otra entidad que necesite realizar el reconocimiento de entidades nombradas en un flujo de documentos indexados en Elasticsearch podrían utilizar este sistema. Además, en caso de no reconocer una entidad, el usuario podría especificar el nombre, el tipo y una descripción de la misma, para así generar datos de entrenamiento y reentrenar el modelo de Spacy para que reconozca dicha entidad.

## Arquitectura del sistema creado

Con el objetivo de dar solución al problema investigativo, se propone el diagrama que aparece en la figura: [(Fig 2.1)](#diagrama_arquitectura). Primeramente, el usuario inserta los datos a los que desea hacer el reconocimiento de entidades nombradas en Elasticsearch. Luego, accede al sistema que se propone, el cual ofrece acceso a todos los índices disponibles. Una vez seleccionado un índice, el sistema mostrará las entidades encontradas. En caso de que todo esté correcto, el usuario guarda los resultados en la base de datos. En caso de que no reconozca una entidad, podrá reentrenar el modelo, especificando la entidad, el tipo y una descripción. El sistema muestra los resultados obtenidos y, en caso de estar bien, el usuario reentrenará el modelo.



**Fig 2.1: Diagrama de Arquitectura**

## Requisitos Funcionales

Los requisitos funcionales en la metodología Rational Unified Process (RUP) se refieren a las declaraciones que identifican los atributos, capacidades, características y cualidades que un sistema o sistema de software debe cumplir para tener valor y utilidad para el usuario. Estos requisitos muestran qué elementos y funciones son necesarias para un proyecto, formando un modelo completo que representa el sistema total a algún nivel de abstracción (Gómez, 2010). Los requisitos necesarios son:

* **RF 1** – Autenticar usuario: Permite a los usuarios autenticarse en el sistema.
* **RF 2** – Registrar usuario: Permite al administrador insertar nuevos usuarios al sistema.
* **RF 3** – Mostrar índices de Elasticsearch: Muestra una lista de índices disponibles en Elasticsearch para que el usuario pueda seleccionar uno.
* **RF 4** – Mostrar resultados del reconocimiento de entidades y generar datos de reentrenamiento: Muestra los resultados del proceso de reconocimiento de entidades en formato JSON para el índice seleccionado y permite al usuario generar datos de reentrenamiento basados en un contexto específico.
* **RF 5** – Reentrenar el modelo Spacy y comprobar la pérdida de conocimiento: Permite al usuario reentrenar el modelo Spacy utilizando los datos generados y verifica que el modelo no haya perdido su capacidad de reconocimiento de entidades después del reentrenamiento.
* **RF 6** – Guardar el índice con las entidades encontradas: Permite al usuario guardar el índice actualizado con las entidades reconocidas.

La integración de los requisitos funcionales de la interfaz de usuario y la lógica de negocio en un conjunto unificado es esencial para ofrecer una visión completa del sistema, abarcando tanto las interacciones directas del usuario con el sistema como las operaciones internas necesarias para satisfacer esas interacciones. Esta unificación es fundamental para asegurar que el sistema no solo cumpla con las expectativas del usuario, sino que también realice sus funcionalidades de manera eficiente y efectiva.

## Requisitos no funcionales del sistema

Los requisitos no funcionales en la metodología Rational Unified Process (RUP) 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. 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 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 se sugiere la utilización de 32 GB de RAM para un mejor desempeño.

## Historias Técnicas

Las funcionalidades son complementadas por las historias técnicas que se enfocan en el diseño o la implementación. Las historias técnicas constituyen las propiedades o cualidades que el producto debe tener.

### Usabilidad

El sistema brinda al usuario una clara navegabilidad de las funciones del sistema

### Confiabilidad

El sistema opera de manera eficiente y confiable cada vez que se emplea, asegurando que las funcionalidades para las que fue diseñado produzcan los resultados que el usuario anticipa, y que estos resultados sean completos y lo más preciso posible.

### Seguridad

El sistema garantiza la seguridad del espacio de trabajo de cada usuario mediante un proceso de autenticación personal, permite el acceso y la realización de acciones con la herramienta. La información recopilada se almacena en bases de datos, por lo que es crucial que el sistema proteja la información que se maneja dentro de estas bases de datos, asegurando, por ejemplo, que las contraseñas estén cifradas para prevenir el acceso no autorizado.

## 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.

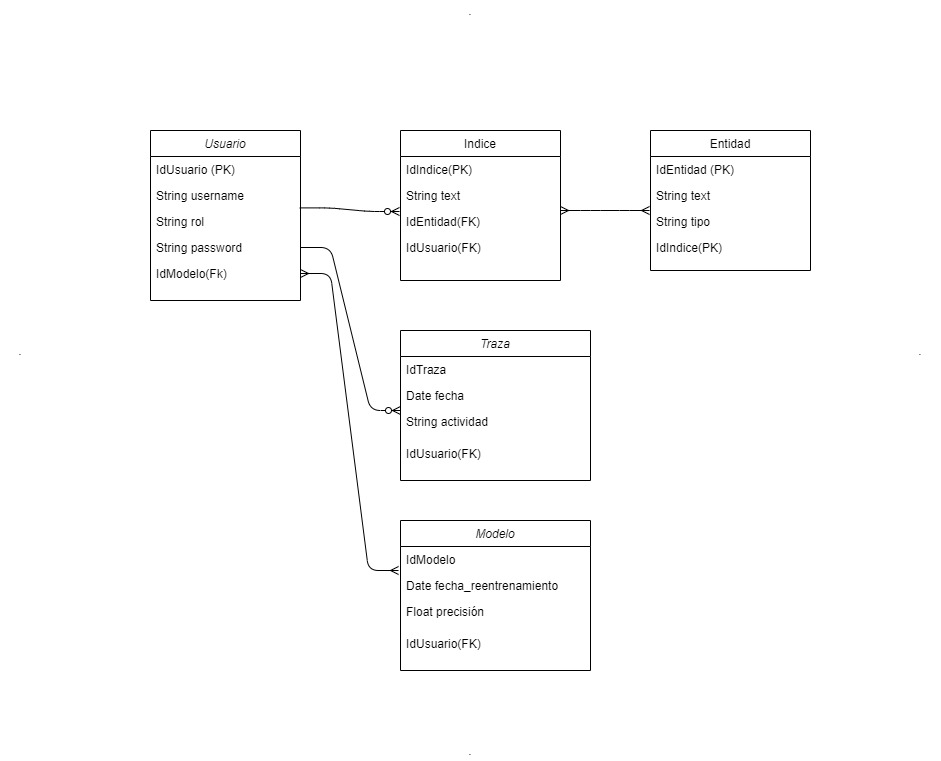
Los índices que se representan son [(ver Fig 2.2)](#modelo_entidad_relacion):

* **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 realizo el reentrenamiento.

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.

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 2.2 Modelo Entidad Relación**

## 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).

* 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 ingles
* HU .7: Reentrenar por especialista

**Tabla 2.1 Historia de Usuario: Autenticación de usuario en el sistema**

|  |  |
| --- | --- |
| **Historia de usuario** | |
| **Número**: 1 | **Nombre de Historia de Usuario**: Autenticar usuario en el sistema |
| **Usuario**: Los trabajadores del centro | |
| **Prioridad** **en negocio**: Alta | **Riesgo de desarrollo**: Bajo |
| **Interacciones del usuario**:   * El usuario accede a la página de inicio de sesión * Llenar formulario de autenticación al sistema * En caso de tener el usuario y contraseña correctos accede al sistema * Caso contrario salta error de autenticación | |
|  | |

Al iniciar la entrada al sistema por parte de un usuario es necesario que este se autentique primero para una mayor seguridad a los datos de la empresa y para definir el rol que este posee (ver tabla 2.1). En caso de no tener una cuenta el administrador deberá de crearle una.

**Tabla 2.2 Historia de Usuario: Inicio al sistema con datos de prueba**

|  |  |
| --- | --- |
| **Historia de usuario** | |
| **Número**: 2 | **Nombre de Historia de Usuario**: Iniciar sistema con datos de prueba |
| **Usuario**: Los trabajadores del centro | |
| **Prioridad** **en negocio**: Alta | **Riesgo de desarrollo**: Baja |
| **Interacciones del usuario:**   * El usuario accede a la página de Home la cual es la primera que aparece cuando se autentica en el sistema * El usuario podrá adquirir datos de prueba para poder empezar a utilizar el sistema * En caso de tener los datos de prueba saldrá un mensaje de que ya los tiene | |
| **D:\DOWNLOAD\Screen Shot 2024-04-06 at 22.21.35.png** | |

En el apartado de Inicio el usuario podrá hacer una revisión rápida de los modelos con datos de prueba que le proporcionamos (ver Tabla 2.2). De tal forma que si no cuenta con ningún índice en la base de datos podrá contar con un nuevo índice al cual realizar el reconocimiento de entidades nombradas.

**Tabla 2.3 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:**   * 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 | |
| **D:\DOWNLOAD\Screen Shot 2024-04-06 at 22.26.18.png** | |

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 2.3).

**Tabla 2.4 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:**   * 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 | |
| **D:\DOWNLOAD\Screen Shot 2024-04-06 at 22.19.38.png** | |

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 evaluaran 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 2.4).

**Tabla 2.5 Historia de Usuario: Administrador de usuarios**

|  |  |
| --- | --- |
| **Historia de usuario** | |
| **Número**: 5 | **Nombre de Historia de Usuario**: Administrador de usuarios |
| **Usuario**: Administradores del centro | |
| **Prioridad** **en negocio**: Alta | **Riesgo de desarrollo**: Alto |
| **Interacciones del usuario**:   * Entrada al administrador de usuarios * Visualizar los datos de los usuarios del sistema * Agregar nuevos usuarios * Editar los valores del usuario * Eliminar los usuarios | |
| **D:\DOWNLOAD\Screen Shot 2024-04-06 at 22.21.31.png** | |

Los encargados de administrar los usuarios serán capaces de insertar nuevos usuarios, editarlos y eliminarlos del sistema (ver Tabla 2.5).

**Tabla 2.6: Historia de usuario: Soporte en español e ingles**

|  |  |
| --- | --- |
| **Historia de usuario** | |
| **Número**: 6 | **Nombre de Historia de Usuario**: Soporte en español e inglés. |
| **Usuario**: Usuarios | |
| **Prioridad** **en negocio**: Alta | **Riesgo de desarrollo**: Bajo |
| **Interacciones del usuario**:   * Seleccionar idioma del sistema | |
|  | |

Los usuarios podrán cambiar el idioma al sistema, para español o para el inglés según su preferencia, esto permitirá que sea usado por un mayor número de usuarios (ver Tabla 2.5).

**Tabla 2.6: Historia de usuario: Reentrenar por especialista**

|  |  |
| --- | --- |
| **Historia de usuario** | |
| **Número**: 6 | **Nombre de Historia de Usuario**: Reentrenar por especialista |
| **Usuario**: Administradores del centro | |
| **Prioridad** **en negocio**: Alta | **Riesgo de desarrollo**: Bajo |
| **Interacciones del usuario:**   * Entrada al apartado de entrenamiento por el administrador * Revisar los datos de reentrenamientos enviados * Reentrenar el modelo * Eliminar los datos enviados | |
|  | |

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 2.6).

## Usuarios del Sistema

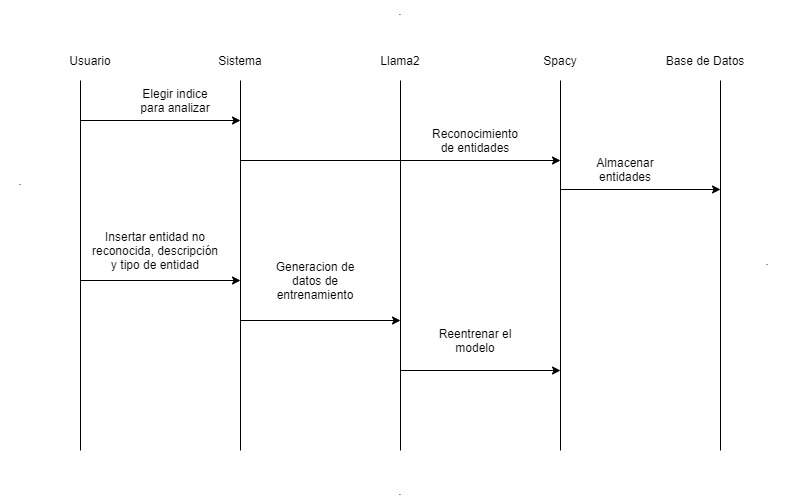
La entrada de usuarios al sistema está dividida en dos roles: usuario y administradores los cuales tienes tareas en específico que en dependencia de su rol podrán realizar en el sistema [(ver Tabla 2.7)](#fig_user_sistem)

**Tabla 2.7 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 |

## Diagrama de Secuencia

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 diagramas 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 2.3 Diagrama de Secuencia**

El diagrama de secuencia de la [Fig 2.3](#diagrama_secuencia) muestra la interacción entre cuatro objetos:

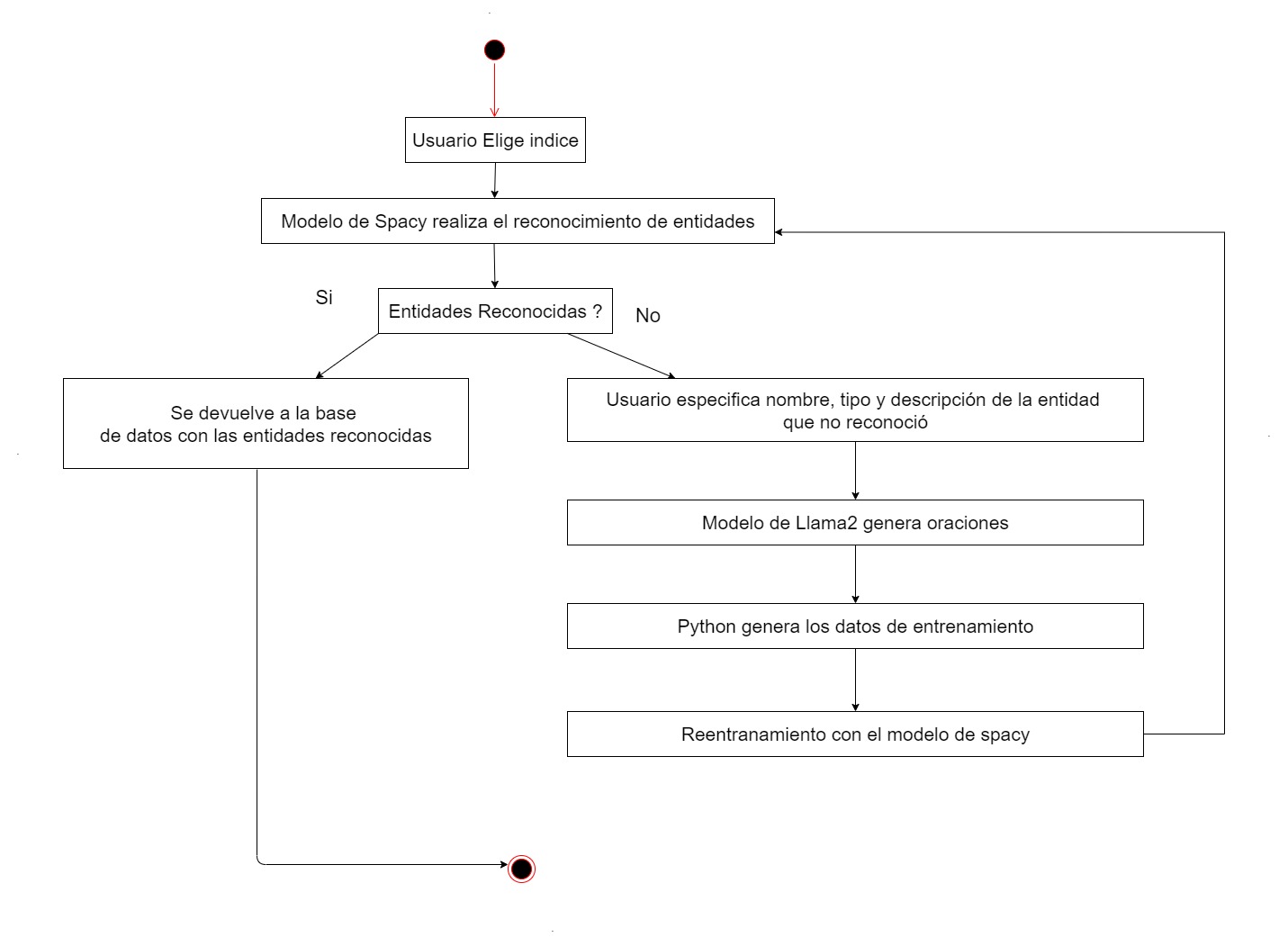
* **Usuario:** El usuario que interactúa con el sistema.
* **Sistema:** El sistema que proporciona el servicio al usuario.
* **Llama2:** Un modelo de lenguaje que genera oraciones.
* **Spacy:** Un modelo de lenguaje que realiza el reconocimiento de entidades.
* **Base de datos:** Elasticsearch sería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

En la figura 2.3 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 Llama2 genera los datos de entrenamiento.
* Reentrenamos el modelo de Spacy para el reconocimiento de la entidad proporcionado por el usuario

## Diagrama de Actividades

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).



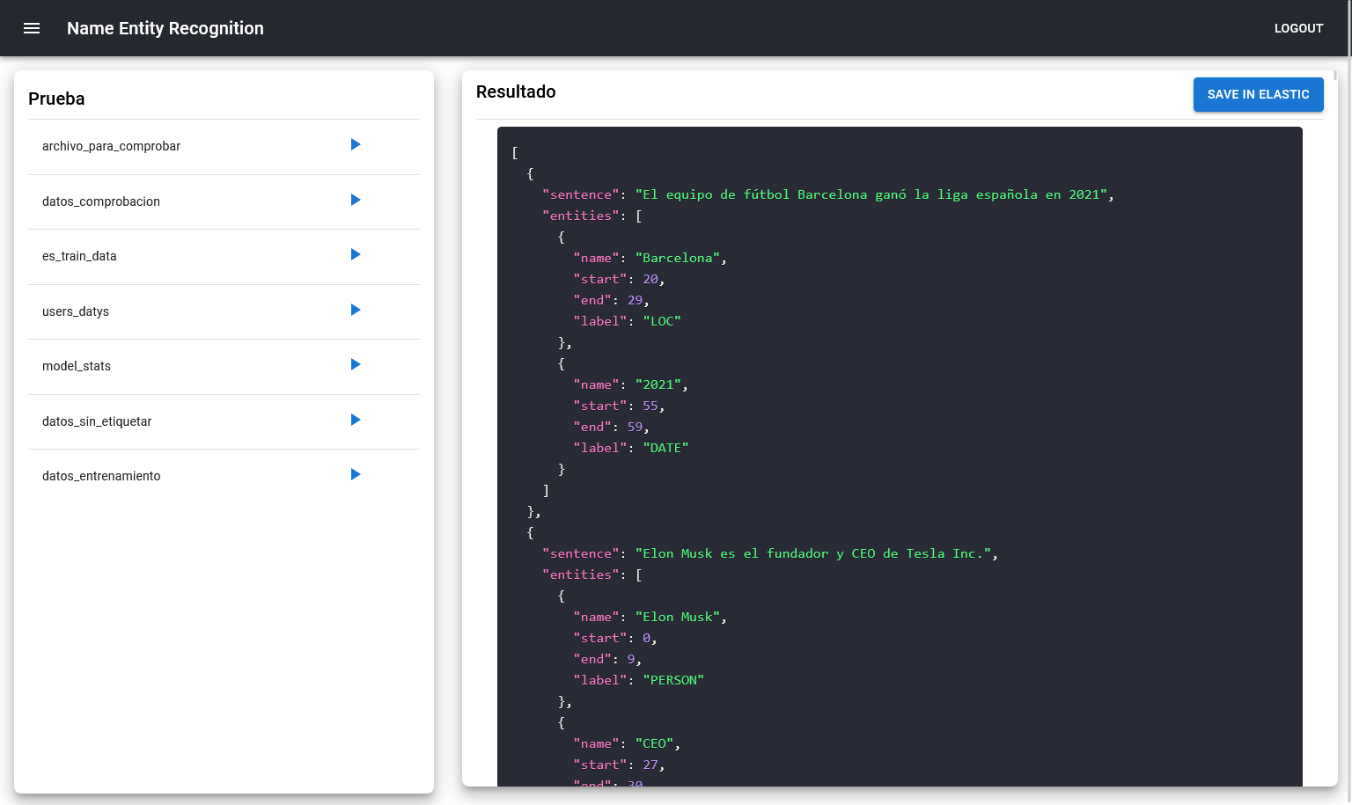
**Fig 2.4 Diagrama de Actividades**

El diagrama de la [figura 2.4](#diagrama_actividades) muestra el flujo de control de un sistema que utiliza un modelo para generar oraciones y otro para el reconocimiento de entidades nombradas. El diagrama lo representa 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 Llama2 genera oraciones a partir del índice y las entidades reconocidas.
* Python genera los datos de entrenamiento para el modelo de Spacy.
* El modelo de Spacy se reentrena con los nuevos datos de entrenamiento.

## Diseño de la Interfaz

El diseño de la interfaz de usuario presentado en la [Fig 2.5](#diseno_interfaz) posee varios puntos fuertes y áreas susceptibles a mejoras, con el fin de optimizar la experiencia de usuario:

****

**Fig 2.5 Diseño de la Interfaz**

**Aspectos positivos:**

* **Claridad:** La interfaz de usuario (UI por sus siglas en inglés) presenta una estructura clara y organizada, facilitando la comprensión del usuario sobre la ubicación y función de los diferentes elementos.
* **Simplicidad:** Se observa un diseño minimalista, libre de elementos superfluos que puedan generar confusión o dificultar la interacción.
* **Consistencia:** Se mantiene una uniformidad en la presentación de los elementos a lo largo de las diferentes pantallas, favoreciendo la familiaridad y el aprendizaje del usuario.
* **Estética:** La paleta de colores y la composición visual general resultan atractivas y agradables a la vista.

**Aspectos a mejorar:**

* **Espacio en blanco:** La incorporación estratégica de espacio en blanco podría mejorar la legibilidad del contenido y la organización de los elementos, optimizando la percepción visual del usuario.
* **Contraste:** Se recomienda ajustar el contraste entre algunos elementos para garantizar una mejor accesibilidad a usuarios con dificultades visuales.
* **Tamaño de fuente:** El tamaño de la fuente podría incrementarse ligeramente, especialmente en dispositivos con pantallas pequeñas, para facilitar la lectura del contenido.
* **Interactividad:** La implementación de elementos interactivos como botones, animaciones o transiciones contribuiría a una experiencia de usuario más dinámica y atractiva.

**Recomendaciones:**

* **Diseño de botones con diferentes colores y estilos:** La diferenciación visual de los botones aportaría claridad y permitiría al usuario identificar su función de forma intuitiva.
* **Incorporación de animaciones o transiciones:** La integración de elementos interactivos de este tipo enriquecería la experiencia del usuario, haciéndola más fluida y atractiva.

## Conclusiones del capítulo

Este capítulo ha delineado el proceso de planificación y diseño del sistema propuesto, presentando los requisitos necesarios para su implementación. Se destaca la importancia de una estrategia integral que aborde aspectos clave como la arquitectura, los requisitos funcionales y no funcionales, así como las historias de usuario y técnicas. Este enfoque detallado sienta las bases para el desarrollo eficiente y efectivo del sistema, asegurando su adaptabilidad y cumplimiento de los objetivos planteados.

# 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.

## Instalación de los requisitos para el funcionamiento del sistema

En el repositorio de github donde se encuentra el código fuente del proyecto[[1]](#footnote-1) se explica de forma detalla los pasos y procedimientos para la instalación y despliegue del sistema y en el Anexo I.

## 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.

### Insertar Usuario

El usuario inicialmente deberá estar guardado en el sistema, la función para que el administrador agregue usuarios al sistema sería [(ver Fig 3.1)](#fig_insertar_usuario):

@elastic\_router\_users\_admin.post("/insert\_user")

async *def* register(user: User):

    try:

        es = Elasticsearch(["http://localhost:9200"], *basic\_auth*=("elastic", "elastic"))

        index = "users\_datys"

        user\_exists = es.search(*index*=index, *body*={"query": {"match": {"username": user.username}}})

        if user\_exists['hits']['total']['value'] > 0:

            return {"message": "El usuario ya existe"}

        hashed\_password = pwd\_context.hash(user.password)

        user\_doc = {

            "username": user.username,

            "password": hashed\_password,

            "rol": user.rol

        }

        es.index(*index*=index, *body*=user\_doc)

        return {"message": "Usuario añadido con éxito"}

    except *Exception* as e:

        print(f"Error en elastic router users admin post: Error: {e}")

        raise HTTPException(*status\_code*=500, *detail*="Error al añadir el usuario")

**Fig 3.1 Función para Insertar un usuario en el sistema**

El código anterior define una función register en FastAPI que se encarga de registrar un nuevo usuario en un índice de Elasticsearch llamado “users\_datys”. Primero, se verifica si el usuario ya existe en el índice mediante una búsqueda por nombre de usuario. Si el usuario ya existe, se devuelve un mensaje indicando que el usuario ya existe. Si no existe, se procede a encriptar la contraseña utilizando “pwd\_context.hash(user.password)”, se crea un documento con el nombre de usuario, la contraseña encriptada y el rol del usuario. Luego se indexa este documento en Elasticsearch. Finalmente, se devuelve un mensaje indicando que el usuario se ha añadido con éxito. En caso de que ocurra algún error durante este proceso, se captura la excepción y se lanza una excepción HTTP con un código de estado 500 y un mensaje de error.

### Autenticación del Usuario

El usuario cuando necesite acceder al sistema es necesario que se autentique. La función para esto sería [(ver Fig 3.2)](#fig_auth_user)

@elastic\_router\_users\_admin.post("/insert\_user")

async *def* register(user: User):

    try:

        es = Elasticsearch(["http://localhost:9200"], *basic\_auth*=("elastic", "elastic"))

        index = "users\_datys"

        user\_exists = es.search(*index*=index, *body*={"query": {"match": {"username": user.username}}})

        if user\_exists['hits']['total']['value'] > 0:

            return {"message": "El usuario ya existe"}

        hashed\_password = pwd\_context.hash(user.password)

        user\_doc = {

            "username": user.username,

            "password": hashed\_password,

            "rol": user.rol

        }

        es.index(*index*=index, *body*=user\_doc)

        return {"message": "Usuario añadido con éxito"}

    except *Exception* as e:

        print(f"Error en elastic router users admin post: Error: {e}")

        raise HTTPException(*status\_code*=500, *detail*="Error al añadir el usuario")

**Fig 3.2 Función para autenticar un usuario al sistema**

Este código define una función post\_login\_result en FastAPI que se encarga de autenticar a un usuario mediante su nombre de usuario y contraseña. Primero, se establece una conexión con Elasticsearch en el índice “users\_datys” y se realiza una búsqueda para encontrar un usuario que coincida con el nombre de usuario proporcionado. Si se encuentra un usuario, se extrae la contraseña y el rol del usuario del documento encontrado. Luego, se verifica si la contraseña proporcionada por el usuario coincide con la contraseña almacenada en la base de datos. Si la contraseña es correcta, se imprime un mensaje de "Autenticación correcta" y se devuelve el rol del usuario. Si la contraseña no coincide, se imprime un mensaje de "Fallo en la autenticación".

### Reconocimiento de Entidades Nombradas a un índice de Elasticsearch

Para realizar el reconocimiento de entidades nombradas el usuario debe de elegir un índice de Elasticsearch al cual se le va a realizar el reconocimiento de entidades nombradas. La función que realiza esto se ilustra en la en la Fig 3.3.

El código de la [Fig 3.3](#fig_ner_principal_function) define una función post\_index\_ner\_result en FastAPI que se encarga de realizar el reconocimiento de entidades nombradas 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 3.3 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 3.4)](#fig_load_spacy_model) 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 3.4 Función para cargar el modelo de Spacy**

Luego se hace llamada a la función “fetch\_documents\_from\_elasticsearch” [(ver Fig 3.5)](#fig_extraer_index_data) a la cual se le asigna el índice seleccionado por el usuario. Esta función se conecta con Elasticsearch y recupera los documentos del índice seleccionado. Cada documento se devuelve como un diccionario que incluye el ID del documento 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 3.5 Función para extraer los datos del índice seleccionado**

Luego se procesa los documentos en la función “process\_documents” [(ver Fig 3.6)](#fig_procesar_extraer_entidades) la cual toma una lista de documentos y el modelo de Spacy como entrada. Procesa cada documento para reconocer las entidades nombradas.

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 3.6 Función para procesar los documentos y extraer las entidades nombradas**

### 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 3.7](#fig_salvar_res_ner):

@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 3.7 Función para salvar los resultados del reconocimiento de entidades nombradas**

El código de la [Fig 3.7](#fig_salvar_res_ner) define una función post\_save\_in\_elastic que se encargara 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. El tipo de campo de entities es nested ya que permite almacenar y consultar datos estructurados anidados. 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.

### Generar datos de entrenamiento

En caso de no reconocer una entidad lo más recomendable es reentrenar el modelo para que la reconozca, 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 3.8](#fig_principal_generar_datos_entrenamient) 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 3.8 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 que pueden ser difíciles de procesar de una sola vez, especialmente en aplicaciones que requieren análisis o generación de texto basada en el contenido del texto [(ver Fig 3.9)](#fig_dividir_texto_largo_en_pequeno). Luego ejecutamos 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 las oraciones generadas 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 3.9 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 3.10)](#fig_generar_datos_entrenamiento). 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 reconocimiento de las entidades nombradas, 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. Esto se hace buscando el texto de la entidad objetivo en el documento procesado y añadiendo su posición de inicio y fin, junto con su etiqueta y se ordenan por su posición de inicio en el texto para asegurar que están en el orden correcto. 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 3.10 Función para generar datos de entrenamiento**

### Reentrenar el modelo de Spacy evitando el olvido catastrófico

El olvido catastrófico en el contexto de NER 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 NLP. Este fenómeno es particularmente relevante en el NER, 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 se entrena para realizar tareas de NER, puede sufrir el olvido catastrófico si se le enseña a realizar otras tareas de NLP, como la clasificación de texto o la traducción automática. Esto significa que, aunque el modelo pueda aprender 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 NER.

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 (WSCL). Este enfoque intenta replicar el proceso de aprendizaje humano, donde las personas afianzan conceptos y aprendizajes durante el sueño, lo que les permite recordar mejor las cosas. En el contexto de la IA, el WSCL 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.

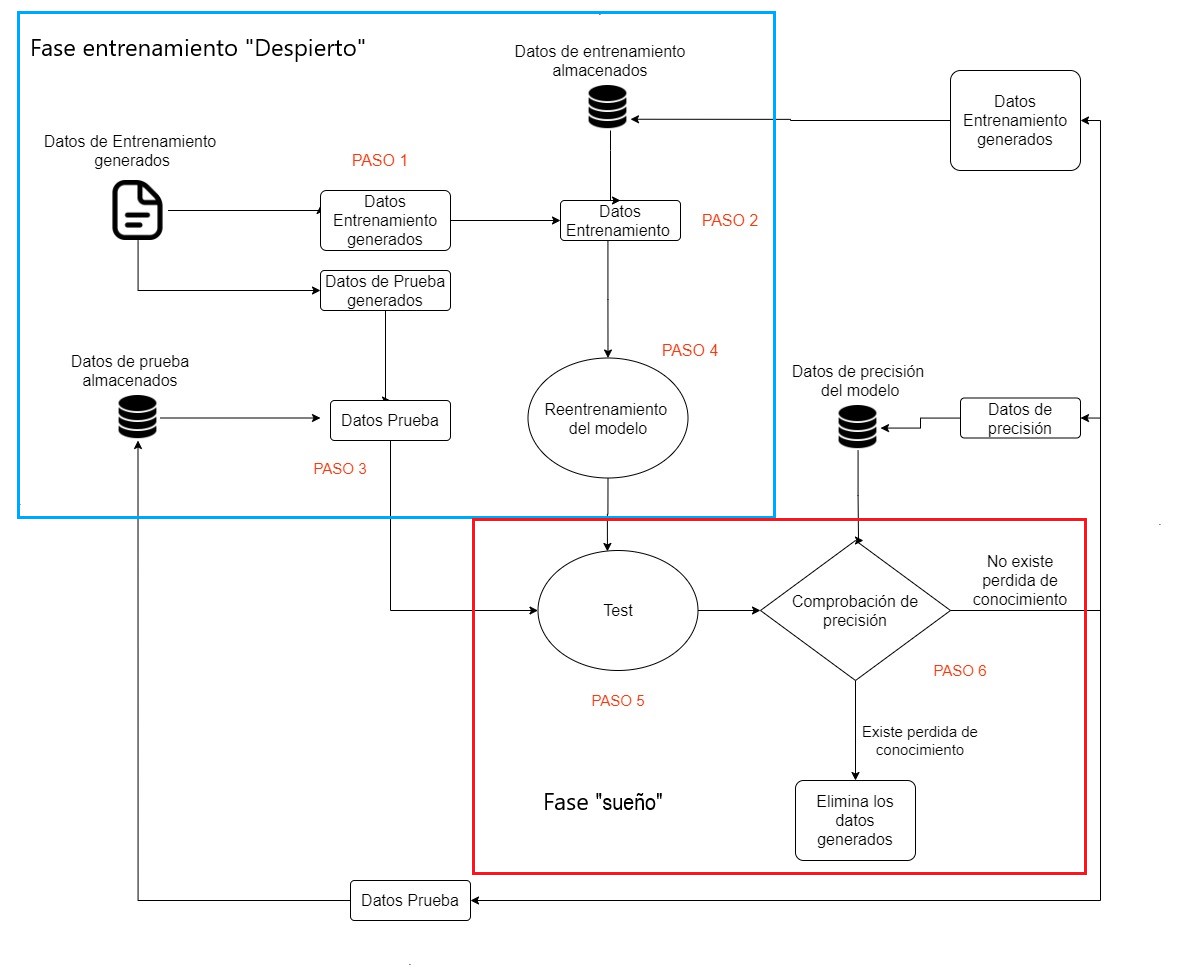
#### Algoritmo propuesto para resolver el olvido catastrófico

Para resolver el problema anterior adaptado a la problemática de investigación se diseñó el siguiente algoritmo ver Fig 3.11:

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 3.11 Diagrama de reentrenamiento para evitar el olvido catastrófico**

## 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 missing (falta), cuando se encuentra en el gold standard pero no en las detectadas por el nuevo modelo.
* Spurious (Sd): Este último caso sería la inversa del missing. Se da cuando el modelo ha detectado una entidad que no está en el gold standard.

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 [34]:

Para el conjunto de datos de comprobación con las nuevas entidades a reconocer se calcularon las siguientes métricas [ver Fig 3.6.](#tabla_metricas)

**Tabla 3.6 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.

## 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 NLP 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 y las historias de usuario sin ajustar. 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 1)](#anexo1) se llegó a la conclusión que el costo total del proyecto sería aproximadamente de **$ 48 477** 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.

## Estimación de costo y tiempo

Para calcular los costos y el tiempo necesarios para el desarrollo de sistemas, se empleó el método de puntos en casos de uso [35], un enfoque eficaz que se ajusta adecuadamente al modelo de caso de uso para la definición de requisitos. Este método se basa en el concepto de transacción de caso de uso, que representa la unidad mínima de medición. Este análisis se realizó considerando las Historias de Usuario, proporcionadas por la metodología XP, como base para la estimación.

## Pruebas al sistema

Un escenario de prueba establece un método para evaluar el sistema, mediante la introducción de datos para verificar si el sistema 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 3.7](#casoPruebaAuth), 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 ésta 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 3.7** 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 3.8](#casoPruebaInicioDatosPrueba), 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 3.8:** 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 del sistema por parte del usuario | Se presiona el botón de aceptar datos de pruebas iniciales |
| 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 3.9](#casoPruebaNER), bajo el título "Caso de Prueba: 'Reconocimiento de entidades nombradas'", 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 reconocimiento de entidades nombradas utilizando Elasticsearch, asegurando un funcionamiento fluido y efectivo del sistema.

**Tabla 3.9:** Caso de Prueba: “Reconocimiento de entidades nombradas”

|  |  |  |
| --- | --- | --- |
| 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 nombradas a los índices de Elasticsearch | Muestra los índices disponibles en Elasticsearch | Existen 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 3.10](#casoPruebaReentrenamiento), 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 3.10**: 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 creo 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 3.11](#casoPruebaPrueba), 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 3.11**: 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 3.12](#casoPruebaEscalabilidad), titulada "Caso de Pruebas 'Pruebas de Escalabilidad'", presenta dos casos de prueba específicos relacionados con la evaluación del modelo reentrenado para reconocimiento de entidades nombradas en situaciones de escalabilidad y aumento de la complejidad de los datos. En el primer caso, "Incremento gradual de datos de entrada", se verifica si el modelo puede reconocer las entidades con la misma precisión a medida que aumenta progresivamente el flujo de datos de entrada. La condición asociada implica un aumento progresivo del flujo de datos para evaluar la capacidad del modelo para escalar adecuadamente.

En el segundo caso, "Aumento de la complejidad de los datos", se evalúa si el modelo puede manejar correctamente las entidades que reconoce, incluso cuando se enfrenta a un aumento gradual en la complejidad de los datos de entrada. La condición en este caso implica un incremento gradual en la complejidad de los datos para evaluar la capacidad del modelo para adaptarse a diferentes contextos. Estas pruebas son fundamentales para garantizar que el modelo reentrenado pueda manejar eficazmente diferentes niveles de carga y complejidad de datos, lo que asegura su efectividad en el proceso de reconocimiento de entidades.

Tabla 3.12 Caso de Pruebas “Pruebas de Escalabilidad”

|  |  |  |
| --- | --- | --- |
| Entrada | Resultados | Condición |
| Incremento gradual de datos de entrada | El modelo reconoce las entidades con la misma precisión con la que fue reentrenado | Aumento progresivo del flujo de datos para verificar la capacidad del modelo para escalar adecuadamente. |
| Aumento de la complejidad de los datos | El modelo maneja correctamente las entidades que reconoce y posee algunos fallos en entidades nuevas | Incremento gradual en la complejidad de los datos de entrada para evaluar la capacidad del modelo para adaptarse a diferentes contextos. |

## 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:

{

"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 III](#anexo3)

Al reentrenar el modelo y volverlo a pasar por el indice 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.

## Conclusiones del capítulo

El análisis al sistema mostro mejores métricas de desempeño en volúmenes de datos de entrenamiento más grandes, además se le dio una solución al olvido catastrófico en caso de pérdida de conocimiento ya adquirido. Se evidenció los resultados luego de realizado el reentrenamiento, mostrando la entidad que no resultaba reconocida, indexando así los resultados de nuevo en la base de datos para su posterior estudio. El análisis económico realizó una estimación de costos del mismo, ahorrándose una cifra aproximada de **$ 48 477** en CUP y contribuyendo a la informatización del país.

**CONCLUSIONES Y RECOMENDACIONES**

**Conclusiones**

Se realizó un análisis crítico del estado del arte sobre las técnicas actuales en el campo del NLP para el NER 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 NLP. Se diseñó e implementó un sistema para la empresa DATYS utilizado para el reconocimiento de entidades nombradas en el idioma español, empleando los modelos de Spacy y Llama2. El mismo obtuvo resultados satisfactorios en la evaluación de las medidas de desempeño, demostrando la necesidad de un gran corpus de información en idioma español para reentrenamiento con mejores resultados. Este sistema estableció un enfoque de optimización continua, evitando el olvido catastrófico durante el reentrenamiento del modelo, asegurando su relevancia y eficacia en el contexto de la búsqueda y análisis de datos.

**Recomendaciones**

* Integrar más funcionalidades al sistema como la posibilidad de tener varios modelos de 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.

Bautista-Villegas, E. (2022). Metodologías agiles 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.

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.

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.

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.

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).

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.

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.

Partalidou, E., Spyromitros-Xioufis, E., Doropoulos, S., Vologiannidis, 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*.

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.

Sulaiman, S., Wahid, R. A., Sarkawi, S., & Omar, N. (2017). Using stanford NER and Illinois NER to detect malay named entity recognition. *International Journal of Computer Theory and Engineering*, *9*(2), 147-150.

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 Variating 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*.

Yogish, D., Manjunath, T. N., & Hegadi, R. S. (2019). Review on natural language processing trends and techniques using NLTK. In *Recent Trends in Image Processing and Pattern Recognition: Second International Conference, RTIP2R 2018, Solapur, India, December 21–22, 2018, Revised Selected Papers, Part III 2* (pp. 589-606). Springer Singapore.

**ANEXOS**

**ANEXO I Pasos de Instalación**

**Instalación de Python 3.10.8**

* **Descargar Python 3.10.8:** Visitar el sitio oficial de Python en <https://www.python.org/downloads/release/python-3108/> y descargar el instalador de Python 3.10.8. Verificar que se seleccionó la versión correcta al sistema operativo (Windows, macOS, Linux) y arquitectura (32 o 64 bits).
* **Ejecutar el instalador**: Ejecutar el archivo de instalación. Para el caso de Windows es recomendado habilitar la opción “Add Python 3.10 to PATH” durante la instalación para que Python se agregue automáticamente al PATH del sistema.
* **Configuración de la instalación**: Seguir las instrucciones del instalador. Puede elegir la ubicación de instalación y otras opciones.
* **Finalización de la instalación**: Una vez terminada la instalación, para verificar que se instaló correctamente, se abre una terminal y se escribe el siguiente comando: “python --version”

**Instalación de Elasticsearch 8.3.1**

* **Descargar Elasticsearch 8.3.1**: Visitar el sitio oficial de Elasticsearch en <https://www.elastic.co/downloads/past-releases/elasticsearch-8-3-1> y descargar el instalador de Elasticsearch 8.3.1. Verificar que se seleccionó la versión correcta al sistema operativo (Windows, macOS, Linux) y arquitectura (32 o 64 bits).
* **Descomprimir el archivo descargado**: descomprimir el archivo descargado en la carpeta donde queremos ejecutarlo.
* **Configuracione**s: Ir a la dirección del archivo descomprimido “config/Elasticsearch.yml” y copiar esta línea: “action.auto\_create\_index: .monitoring\*,.watches,.triggered\_watches,.watcher-histroy\*,.ml”
* **Ejecutamos el Elasticsearch**: Ejecutar el Elasticsearch ubicado en la dirección “bin/elasticsearch.bat” y salvamos el usuario y la contraseña
* **Retornamos al archivo:** “config/Elasticsearch.yml” y cambiamos la siguiente línea: “xpack.security.http.ssl: enabled: false “y “xpack.security.transport.ssl: enabled: false “
* **Volvemos a correr Elasticsearch**
* **Finalización**: Abrimos en el navegador el puerto 9200 e insertamos el usario y la contraseña guardadas anteriormente

**Instalación de las dependencias de python**

Para instalar las dependencias de Python ejecutamos el siguiente comando ‘ pip install requirements.txt ’

**Instalación del modelo de spacy es\_core\_news\_lg**

Para instalar el modelo el modelo es\_core\_news\_lg de spacy para el reconocimiento de entidades nombradas tenemos que ejecutar el siguiente comando: “python -m spacy download es\_core\_news\_lg”.

**Instalación del modelo llama2**

* **Descargar Ollama**: Para instalar el modelo de llama2 es necesario tener instalado el Ollama el cual lo podemos descargar en el siguiente link: <https://ollama.com/download> tenemos que seleccionar nuestro sistema operativo
* **Ejecutar el instalador**: Ejecutamos el archivo del instalador y aceptamos todos los términos y condiciones de uso
* **Descargar el modelo de llama2**: Para descargar el modelo de llama2 según la documentación del sitio en el siguiente link: <https://ollama.com/library/llama2> corresponde a ejecutar el siguiente comando: “ollama run llama2”

**Instalación de las herramientas empleadas en la Interfaz de Usuario**

**Instalación de node 20.11.0**

* **Descargar Node.js 20.11.0**: Visitar el sitio oficial de Node.js en <https://nodejs.org/en> y seleccionamos la version 20.11.0 de node. Se selecciona el instalador compatible con el sistema operativo y la arquitectura
* **Ejecutar el instalador**: Una vez completada la descarga, se ejecuta el archivo de instalación y y se sigue las instrucciones para completar la instalación
* **Verificar la instalación**: Para verificar la instalación. Se ejecuta esta línea de comando: “node --version” y “npm --version”

**Instalación de las librerías y dependencias del proyecto para el front**

En la carpeta del proyecto: frontend/ se ejecuta el siguiente comando: “npm install” este descargara todas las dependencias necesarias.

**Pasos para correr la aplicación**

Si es la primera vez que se va a correr la apliacion, entramos a la carpeta inicio en esta modifcamos el valor del Path para seleccionar la carpeta donde se guarde el modelo de spacy que se va a estar reentrenando y se ejecuta el siguiente comando: “python .\main.py”, esto guardaría una instancia del modelo de spacy: es\_core\_news\_lg en la carpeta seleccionada y almacenaría un índice de usuarios necesarios para la autenticación en el sistema. En todos los demás casos los pasos serian:

* **Ejecutar Elasticsearch**: En el directorio donde se instalo Elasticsearch se dirige a la carpeta: “bin/elasticsearch.bat”
* **Ejecutar el backend**: En el directorio del proyecto se dirige a la carpeta: “sentence-gen-api/” y se ejecuta el comando “python .\main.py”.
* **Ejecutar el frontend**: En el directorio del proyecto se dirige a la carpeta “frontend/” y se ejecuta el comando “npm run dev”
* Al levantarse el proyecto se entra a la dirección del puerto que imprime en consola, en el sistema aparecerá el login: user: admin, password: admin y se accederá con los permisos de administrador.

**ANEXO II 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

**UUCP = UAW + 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

**TCF = 0.6 + 0.01 \* Σ (Pesoi\* Valori)**

**TCF = 0,6 + 0.01 \* 45.5**

**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

**EF = 1.4 - 0.03 \* Σ (Pesoi \* Valori)**

**EF = 1.4 - 0.03 \* 22.75**

**EF = 0.7175**

**Luego: UCP=UUCP \* TCF \* 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 Horas-hombres / Puntos de historias de usuarios

Luego

**E = 23 \* 20 horas-hombre**

**E = 474 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 = 474horas/4 ≈ 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(Total) \* K \* THP**

C = 474 \* 1.5 \*12000/176 = **$ 48 477**

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 (Estatal) | 9 000 CUP |
| DATYS (Estatal) | 9 000 CUP |
| DESOFT (Estatal) | 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 III 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"

}

]

},

"doc3": {

"text": "El técnico del equipo de pelota teamacere, Carlos, ha estado trabajando duro con los jugadores para mejorar su juego.",

"entities": [

{

"name": "teamacere",

"start": 32,

"end": 41,

"label": "ORG"

},

{

"name": "Carlos",

"start": 43,

"end": 49,

"label": "MISC"

}

]

},

"doc4": {

"text": "El equipo de pelota teamacere es conocido por su furia y determinación en el campo de juego.",

"entities": [

{

"name": "teamacere",

"start": 20,

"end": 29,

"label": "ORG"

}

]

},

"doc5": {

"text": "El presidente del club de pelota teamacere, Juan, está orgulloso de cómo ha crecido el equipo en las últimas temporadas.",

"entities": [

{

"name": "teamacere",

"start": 33,

"end": 42,

"label": "ORG"

},

{

"name": "Juan",

"start": 44,

"end": 48,

"label": "PERSON"

}

]

},

"doc6": {

"text": "La jugadora más experimentada del equipo de pelota teamacere, María, ha sido clave en varios partidos importantes.",

"entities": [

{

"name": "teamacere",

"start": 51,

"end": 60,

"label": "ORG"

},

{

"name": "María",

"start": 62,

"end": 67,

"label": "PERSON"

}

]

},

"doc7": {

"text": "El equipo de pelota teamacere tiene una gran base de fanáticos en todo Cuba que les apoyan en todas sus competencias.",

"entities": [

{

"name": "teamacere",

"start": 20,

"end": 29,

"label": "ORG"

},

{

"name": "Cuba",

"start": 71,

"end": 75,

"label": "LOC"

}

]

},

"doc8": {

"text": "La fecha del próximo partido de teamacere está programada para el domingo a las 10 AM en el estadio local.",

"entities": [

{

"name": "teamacere",

"start": 32,

"end": 41,

"label": "ORG"

},

{

"name": "10 AM",

"start": 80,

"end": 85,

"label": "DATE"

}

]

}

}

1. <https://github.com/luislicea1/NER-with-spacy-elasticsearch-and-Llama2> [↑](#footnote-ref-1)