

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:** “Herramienta informática de procesamiento de lenguaje natural para el reconocimiento de entidades”.

**Autor:** Luis Andrés Licea Berenguer.

**Tutor:** Dr. C. Dionis López Ramos.

Santiago de Cuba,

“Año de la Revolución”

Resumen

En la actualidad, el reconocimiento de entidades nombradas desempeña un papel fundamental en diversos contextos, desafiando la capacidad de las aplicaciones para identificar y comprender información clave en grandes volúmenes de datos no estructurados. La habilidad de discernir entidades, como personas, lugares, organizaciones y otros elementos específicos en el texto, es un componente crítico en la extracción de conocimiento y toma de decisiones.

Este estudio se enfoca en la creación de un modelo de reconocimiento de entidades mediante la implementación de la biblioteca SpaCy y el lenguaje de programación Python. El modelo no solo tiene la capacidad de identificar entidades, sino que también incorpora una función de reentrenamiento supervisado para abordar la aparición de nuevas y desconocidas entidades.

Para lograr este propósito, se aplican técnicas avanzadas de procesamiento de lenguaje natural y aprendizaje automático. Se utiliza un conjunto de datos etiquetados para entrenar el modelo, cuyo desempeño se evalúa mediante métricas de precisión, exhaustividad y F1-score. Los resultados obtenidos demuestran que el modelo es altamente preciso y exhaustivo en su capacidad para reconocer entidades.

La característica distintiva de este modelo radica en su funcionalidad de reentrenamiento supervisado, lo que permite una adaptación dinámica a nuevas entidades, garantizando un rendimiento óptimo en tiempo real. En resumen, este trabajo aporta significativamente al campo del procesamiento de lenguaje natural al proporcionar un modelo preciso y adaptable que satisface la creciente importancia del reconocimiento de entidades en el entorno contemporáneo.

**Palabras clave:** Reconocimiento de entidades, SpaCy, Python, reentrenamiento supervisado

*ABSTRACT*

In the present day, named entity recognition plays a fundamental role across various contexts, challenging applications to identify and comprehend critical information within extensive volumes of unstructured data. The ability to discern entities, such as people, places, organizations, and other specific elements within text, is a critical component in knowledge extraction and decision-making.

This study focuses on the development of a named entity recognition model through the implementation of the SpaCy library and the Python programming language. The model not only possesses the capability to identify entities but also incorporates a supervised retraining function to address the emergence of new and unfamiliar entities.

To achieve this objective, advanced techniques in natural language processing and machine learning are employed. A labeled dataset is used to train the model, and its performance is evaluated using precision, recall, and F1-score metrics. The results demonstrate that the model exhibits a high degree of accuracy and comprehensiveness in its entity recognition capabilities.

The distinctive feature of this model lies in its supervised retraining functionality, allowing dynamic adaptation to novel entities, ensuring optimal real-time performance. In summary, this work significantly contributes to the field of natural language processing by providing a precise and adaptable model that meets the growing importance of named entity recognition in the contemporary environment.

**Keywords*:***Named Entity Recognition, SpaCy, Python, Supervised Retraining

Tabla de contenido

[**INTRODUCCIÓN** 1](#_Toc163212745)

[**CAPITULO 1 .** **MARCO TEÓRICO** 5](#_Toc163212746)

[1.1 Procesamiento del Lenguaje Natural 5](#_Toc163212747)

[1.2 Estado del Arte de los marcos de procesamiento del lenguaje natural para la detección de entidades 5](#_Toc163212748)

[1.2.1 Google NLP 5](#_Toc163212749)

[1.2.2 ChatGPT 6](#_Toc163212750)

[1.2.3 IBM Watson Discovery 6](#_Toc163212751)

[1.2.4 Amazon Comprehend 7](#_Toc163212752)

[1.2.5 Dandelion 7](#_Toc163212753)

[1.3 Herramientas, Lenguaje de programación y Tecnologías 8](#_Toc163212754)

[1.3.1 Python 8](#_Toc163212755)

[1.3.2 Visual Studio Code 9](#_Toc163212756)

[1.3.3 Elasticsearch 10](#_Toc163212757)

[1.3.4 Git 11](#_Toc163212758)

[1.3.5 Modelos para el reconocimiento de entidades nombradas 11](#_Toc163212759)

[1.3.6 Modelos generadores de oraciones 13](#_Toc163212760)

[1.3.7 React 14](#_Toc163212761)

[1.3.8 FastAPI 15](#_Toc163212762)

[1.3.9 XP 15](#_Toc163212763)

[1.4 Conclusiones del capítulo 17](#_Toc163212764)

[**CAPITULO 2 .** **ORGANIZACIÓN Y DISEÑO** 18](#_Toc163212765)

[2.1 Propuesta del sistema 18](#_Toc163212766)

[2.2 Arquitectura del sistema creado 18](#_Toc163212767)

[2.3 Requisitos Funcionales 19](#_Toc163212768)

[2.3.1 Requistos funcionales del sistema 19](#_Toc163212769)

[2.4 Requisitos no funcionales del sistema 20](#_Toc163212770)

[2.4.1 Requisitos de Software 20](#_Toc163212771)

[2.4.2 Requisitos de Hardware 20](#_Toc163212772)

[2.5 Historias Técnicas 21](#_Toc163212773)

[2.5.1 Usabilidad 21](#_Toc163212774)

[2.5.2 Confiabilidad 21](#_Toc163212775)

[2.5.3 Seguridad 21](#_Toc163212776)

[2.6 Modelo Entidad Relación 21](#_Toc163212777)

[2.7 Historias de Usuario 23](#_Toc163212778)

[2.8 Usuarios del Sistema 30](#_Toc163212779)

[2.9 Diagrama de Secuencia 30](#_Toc163212780)

[2.10 Diagrama de Actividades 31](#_Toc163212781)

[2.11 Diseño de la Interfaz 33](#_Toc163212782)

[2.12 Conclusiones del capítulo 34](#_Toc163212783)

[CAPITULO 3 . IMPLEMENTACIÓN Y PRUEBA 35](#_Toc163212784)

[3.1 Instalación de los requisitos para el funcionamiento del sistema 35](#_Toc163212785)

[3.2 Algoritmos Importantes 35](#_Toc163212786)

[3.2.1 Insertar Usuario 35](#_Toc163212787)

[3.2.2 Autenticación del Usuario 36](#_Toc163212788)

[3.2.3 Reconocimiento de Entidades Nombradas a un índice de Elasticsearch 37](#_Toc163212789)

[3.2.4 Salvar resultados del reconocimiento en el índice seleccionado 38](#_Toc163212790)

[3.2.5 Generar datos de entrenamiento 40](#_Toc163212791)

[3.2.6 Reentrenar el modelo de Spacy 42](#_Toc163212792)

[3.3 Análisis económico 44](#_Toc163212793)

[3.4 Estimación de costo y tiempo 45](#_Toc163212794)

[3.5 Pruebas al sistema 45](#_Toc163212795)

[3.6 Conclusiones del capítulo 48](#_Toc163212796)

[**1.** **CONCLUSIONES Y RECOMENDACIONES** 49](#_Toc163212797)

[**1.** **REFERENCIAS BIBLIOGRÁFICAS** 50](#_Toc163212798)

[**2.** **GLOSARIO DE TÉRMINOS** 52](#_Toc163212799)

[**4.** **ANEXOS** 53](#_Toc163212800)

**INTRODUCCIÓN**

El término “entidad nombrada” fue usado por primera vez en la Message Understanding Conference(MUC-6), organizada por Grishman y Sundheim (1996), y se refirió a la tarea de identificar nombres de organizaciones, personas y ubicaciones geográficas en textos, así como expresiones monetarias de tiempo y porcentajes. Desde la conferencia MUC-6, el interés en NER fue incrementándose hasta hacerse presente en varios eventos científicos.[1]

Ya en la actualidad elReconocimiento de Entidades Nombradases una técnica basada en el procesamiento de lenguaje natural (NLP) que se utiliza para extraer, identificar y clasificar información en documentos de texto. Detecta entidades (es decir, partes de la oración) y las clasifica en una categoría predeterminada, como nombre o código de país.[2]

Las categorías NER pueden ser genéricas e indicar, por ejemplo, palabras que significan una organización, una persona o una época. Sin embargo, también pueden personalizarse en función de un caso de uso específico. Por poner un ejemplo, el modelo NER puede crearse para reconocer categorías como “nombre del paciente” y “fecha de nacimiento” en documentos médicos o “nombre del comerciante” y “fecha de compra” en facturas. Las posibilidades son infinitas.[2]

Desde la aparición de distintas tecnologías, como la Internet, la cantidad de información disponible en distintos formatos y fuentes ha crecido a pasos agigantados. El tamaño de las colecciones almacenadas dificulta su manejo y organización, así como la posibilidad de encontrar información específica, ya sea, en un solo documento o en un conjunto de documentos. Además, la información puede no tener una estructura definida, como es el caso de la información textual, donde se encuentran principalmente secuencias de palabras. [3].

Dadas estas dificultades 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.

**Problema de investigación:**

La empresa DATYS-CERPAMID de Santiago de Cuba tiene limitaciones en el reconocimiento de entidades nombradas al procesar volúmenes masivos de datos diariamente en Elasticsearch.

**Objeto de Estudio:**

Reconocimiento de Entidades Nombradas.

**Campo de Estudio:**

Modelos de procesamiento del lenguaje natural.

**Objetivo General:**

Crear una herramienta de procesamiento del lenguaje natural para el reconocimiento de entidades nombradas en volúmenes masivos de datos en Elasticsearch.

**Objetivo Específicos:**

* **Extracción de Datos de Elasticsearch:** El programa se encargará de establecer la conexión con Elasticsearch y extraer los datos almacenados en la base de datos, asegurando la integridad y calidad de la información recuperada.
* **Reconocimiento de Entidades (NER):** Se desarrollará una funcionalidad de NER utilizando el modelo de Spacy para la identificación y clasificación automática de entidades presentes en los datos extraídos. Esto incluye la detección de nombres de personas, organizaciones, ubicaciones, fechas, entre otros.
* **Aprendizaje Supervisado:** En caso de que el modelo Spacy no reconozca una entidad específica en los datos, se implementará un proceso de aprendizaje supervisado. El programa permitirá a los usuarios etiquetar manualmente las nuevas entidades, reentrenando así el modelo para reconocer estas entidades en el futuro.
* **Almacenamiento de Resultados en Elasticsearch:** Una vez identificadas las entidades, el programa guardará los resultados de manera estructurada en Elasticsearch, permitiendo un acceso eficiente y consultas posteriores.
* **Evaluación de Rendimiento:** Se realizará una evaluación del rendimiento del programa, incluyendo la precisión y la eficiencia del reconocimiento de entidades, así como la efectividad del proceso de aprendizaje supervisado.
* **Optimización Continua:** El objeto de investigación incluye la posibilidad de optimizar y mejorar el programa a medida que se recopilen más datos y se identifiquen nuevas entidades.

**Campo de Acción:**

Aplicación de modelos de procesamiento del lenguaje natural, como Spacy, para detectar entidades, proponiendo el reentrenamiento del modelo cuando no se reconocen correctamente las entidades. Además, se integra una base de datos en Elasticsearch para almacenar y analizar los resultados del reconocimiento de entidades.

**Hipótesis:**

Si se desarrolla una herramienta de procesamiento de lenguaje natural para el reconocimiento de entidades nombradas, se podrá contribuir a disminuir las limitaciones actuales en el reconocimiento de las mismas en grandes volúmenes de datos en Elasticsearch. Esta herramienta buscará mejorar la eficiencia y efectividad en la identificación y clasificación de entidades, facilitando así el estudio lingüístico y la gestión de datos masivos para fines de análisis.

**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.

**Interrogantes Científicas:**

* ¿Cómo se puede mejorar la precisión y exhaustividad del modelo de reconocimiento de entidades utilizando técnicas de procesamiento de lenguaje natural y aprendizaje automático?
* ¿Cómo se puede implementar una funcionalidad de reentrenamiento supervisado para adaptar el modelo a nuevas entidades en tiempo real?
* ¿Cuál es el impacto de utilizar la biblioteca Spacy y el lenguaje de programación Python en el desarrollo y desempeño del modelo de reconocimiento de entidades?

**Aportes de la Investigación:**

* Un sistema que reconozca las entidades de los textos de la base de datos de Elasticsearch.
* Capacidad de reentrenamiento de forma supervisada por parte del usuario o la entidad en caso de surgir o no detectar una nueva entidad.
* Generación de datos de entrenamiento y testing de la entidad que no reconozca el modelo.
* Capacidad de retroceder en caso de el modelo 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. [4]

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

En el ámbito del procesamiento del lenguaje natural (NLP), el estado del arte de los marcos para la detección de entidades nombradas (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.

### Google NLP

Google Cloud NLP es una API de Inteligencia Artificial (IA) que se centra en el procesamiento del lenguaje natural (NLP). Esta API es parte de la suite de productos digitales de Google, que están en constante crecimiento y amplían las capacidades de la tecnología para las empresas.[5]

La API de Google Cloud NLP 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.[5]

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 tiene algunas limitaciones. Aunque es una opción fuerte en las áreas en las que proporciona, la API actualmente es más limitada 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. Además, Google Cloud NLP es una opción más costosa que muchas de sus competidoras, lo que significa que puede ser más difícil de experimentar que otras opciones. Aunque esta API proporciona créditos gratuitos a nuevos clientes, el costo más alto de este programa puede no valer la pena una vez que su negocio utiliza estos créditos y necesita pasar a una suscripción continua.

### 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 (interfaz de programación de aplicaciones, por sus siglas en inglés) 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.

Hoy en día, esta herramienta está detrás de miles de programas y aplicaciones, como son tiendas online, buscadores, aplicaciones de aprendizaje y hasta plataformas de redes sociales. Además, es un gran apoyo si quieres investigar sobre un tema altamente especializado, si deseas reescribir textos o si necesitas ideas creativas para redactar tus propios contenidos. [6]

Entre las ventajas del uso de ChatGPT para el reconocimiento de entidades nombradas es 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 de que existen modelos de aprendizaje automático preentrenados que facilitan la implementación de diferentes aplicaciones de PLN, lo que reduce el tiempo y esfuerzo necesarios para desarrollar soluciones desde cero.

Entre las desventajas de su uso para el reconocimiento de entidades 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.

### IBM Watson Discovery

Watson es la plataforma de inteligencia artificial 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, esencialmente, 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.[6]

IBM Watson Discovery posee varias ventajas para el reconocimiento de entidades nombradas 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 HTML, 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 su implementación y mantenimiento pueden ser costosos y requieren un nivel de conocimiento técnico considerable. La efectividad de Watson Discovery 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 y sobre la privacidad y seguridad de sus datos.

### Amazon Comprehend

Otra excelente herramienta para el procesamiento de texto en documentos es Amazon Comprehend, que forma parte del universo de Amazon Web Services. Este sistema es capaz de extraer información tanto de documentos como de tickets de atención al cliente, reseñas, correos electrónicos y redes sociales.

Por estas funciones es que Comprehend resulta tan útil para tiendas en línea o empresas con gran actividad en medios de comunicación masiva. Con algunos simples pasos, puedes saber qué opinión tienen los clientes de ti, qué piensan de tus productos y hasta el tono de tus interacciones con ellos a través de tus canales de atención.

También, hay empresas que usan este sistema para la administración de enfoques legales y procesamiento de documentos financieros, ya que pueden detectar detalles que hasta tu equipo legal y de ventas puede pasar por alto en los términos y condiciones de alguna operación.[6]

Una de las desventajas notables es su dependencia de modelos preentrenados que puedan no adaptarse completamente a contextos específicos o dominios de uso, lo que puede resultar en errores de clasificación o falta de precisión en ciertos casos.

### Dandelion

Dandelion es una API que se centra en el análisis semántico y de texto para ayudar a los usuarios a entender si los documentos que se están visualizando son principalmente positivos, negativos o neutrales. Esto permite a los usuarios aprender más sobre las opiniones generales sobre ciertos temas, lo que puede ayudarles a determinar qué ángulo sería el mejor para trabajar con la información que aprenden.[5]

Esta API se dirige principalmente a las empresas que están interesadas en investigar lo que su público objetivo está diciendo sobre varios temas que están relacionados con su industria, así como cómo sus competidores están manejándolos. Aunque no se adentra tanto en lo que los usuarios pueden hacer con esta información, Dandelion proporciona una visión general sólida de qué información relevante está actualmente disponible.[5]

Las ventajas de utilizar Dandelion para el reconocimiento de entidades nombradas (NER) son notables, destacando por su capacidad de procesamiento avanzado y su eficiencia en la automatización de procesos de negocio. La integración con otras herramientas de IBM, como PrimeQA, y su habilidad para analizar texto en formatos no estructurados, incluyendo HTML, páginas web y redes sociales, son características que lo hacen especialmente útil para la extracción de información relevante de diversas fuentes. Además, su capacidad para proporcionar insights empresariales esenciales de manera más eficiente, gracias a su uso de tecnologías de procesamiento del lenguaje natural (PLN), lo convierte en una opción atractiva para organizaciones que buscan optimizar sus operaciones y tomar decisiones informadas.

Sin embargo, a pesar de sus ventajas significativas, Dandelion también presenta desafíos que deben ser considerados. La complejidad y los costos asociados con su implementación y mantenimiento pueden ser barreras para algunas organizaciones, especialmente aquellas con recursos limitados. Además, la dependencia de la calidad y estructura de los datos de entrada, así como las limitaciones en la personalización de sus algoritmos, pueden requerir una inversión significativa en formación y ajustes para adaptarse a las necesidades específicas de cada negocio. Aunque ofrece créditos gratuitos a nuevos clientes, el costo de estos créditos puede no valer la pena una vez que el negocio utiliza estos créditos y necesita pasar a una suscripción continua. Por último, las preocupaciones sobre la privacidad y seguridad de los datos al procesar grandes volúmenes de información son aspectos que deben ser cuidadosamente considerados al evaluar esta solución.

## 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, por lo que son generalmente artefactos con una función técnica. 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. 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.

### Python

Python en su versión 3.10.8 es un lenguaje de programación ampliamente utilizado en las aplicaciones web, el desarrollo de software, la ciencia de datos y el machine learning (ML). Los desarrolladores utilizan Python porque es eficiente y fácil de aprender, además de que se puede ejecutar en muchas plataformas diferentes. El software Python se puede descargar gratis, se integra bien a todos los tipos de sistemas y aumenta la velocidad del desarrollo [7].

Se utilizó Python porque ofrece los siguientes beneficios:

* Los desarrolladores pueden leer y comprender fácilmente los programas de Python debido a su sintaxis básica similar a la del inglés.
* Python permite que los desarrolladores sean más productivos, ya que pueden escribir un programa de Python con menos líneas de código en comparación con muchos otros lenguajes.
* Python cuenta con una gran biblioteca estándar que contiene códigos reutilizables para casi cualquier tarea. De esta manera, los desarrolladores no tienen que escribir el código desde cero.
* Los desarrolladores pueden utilizar Python fácilmente con otros lenguajes de programación conocidos, como Java, C y C++.
* La comunidad activa de Python incluye millones de desarrolladores alrededor del mundo que prestan su apoyo. Si se presenta un problema, puede obtener soporte rápido de la comunidad.
* Hay muchos recursos útiles disponibles en Internet si desea aprender Python. Por ejemplo, puede encontrar con facilidad videos, tutoriales, documentación y guías para desarrolladores.
* Python se puede trasladar a través de diferentes sistemas operativos de computadora, como Windows, macOS, Linux y Unix.

### Visual Studio Code

Visual Studio Code (VS Code) es un editor de código fuente ligero pero potente desarrollado por Microsoft. Está disponible para Windows, macOS y Linux y es gratuito para descargar y utilizar. VS Code combina la simplicidad de un editor de código fuente con potentes herramientas para desarrolladores, como la finalización de código IntelliSense y la depuración. [8]

Algunas de las características clave de VS Code:

1. Soporte integrado para varios lenguajes de programación: VS Code viene con soporte integrado para JavaScript, TypeScript y Node.JS, y tiene un rico ecosistema de extensiones para otros lenguajes como C++, C#, Java, Python, PHP, Go, y para tiempo de ejecución como .NET y Unity [8].
2. Interfaz de línea de comandos potente: VS Code tiene una potente interfaz de línea de comandos que te permite controlar cómo lanzas el editor. Puedes abrir diferentes archivos, instalar extensiones, e incluso cambiar el idioma de visualización al inicio [8].
3. Integración con Git: VS Code viene con una integración con Git que te permite hacer commit, pull y push de tus cambios de código a un repositorio Git remoto [8].
4. Extensibilidad: Puedes elegir entre miles de extensiones para personalizar tu IDE. Esto te permite agregar funcionalidades adicionales a VS Code para adaptarlo a tus necesidades específicas [8].

### Elasticsearch

Esta base de datos es un motor de analítica y búsqueda de RESTful distribuido que almacena de forma centralizada tus datos para que puedas buscar, indexar y analizar datos de todas formas y tamaños[9].

Algunas características de Elasticsearch son:

* Desarrollado en Java
* Open Source
* Distribuido
* Escalable
* Basado en Lucene

Lucene es una librería de búsqueda de texto[10]:

* Desarrollado en java
* Open Source
* Escalable
* Con alto rendimiento
* Basada en índices invertidos



Fig 1.0

Como se muestra en la figura 1.0 un índice directo en una base de datos es que a cada texto o indexación a la base de datos se guarda con un id, mientras que en índices invertidos se le asigna un id a cada palabra y lo que se guarda en la base de datos es el conjunto de ubicaciones donde se encuentra esta palabra, haciendo así una búsqueda más rápida.

### 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 [11].

Git 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, Git permite a los usuarios 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 [11].

Git también es compatible con todos los sistemas operativos disponibles y puede acceder directamente a otros repositorios de control de versiones remotos. Esto significa que los usuarios pueden cambiar fácilmente a Git sin tener que mover sus archivos de esos repositorios al repositorio de Git. Además, Git es altamente escalable y más rápido que otros sistemas de control de versiones, lo que permite manejar eficientemente proyectos grande [11].

### Modelos para el reconocimiento de entidades nombradas

**Spacy**

SpaCy es una biblioteca de procesamiento de lenguaje natural (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. spaCy es conocido por su velocidad y eficiencia, y es especialmente útil para tareas de NLP que requieren un análisis rápido y preciso del texto [12].

**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.[13]

**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. Flair también ofrece clases específicas para incrustar palabras y documentos completos. Esto permite a los usuarios incorporar fácilmente estas representaciones avanzadas en sus propias aplicaciones de NLP. Además, Flair permite personalizar el modelo mediante el ajuste fino (fine-tuning) para adaptarlo a tareas específicas. [14]

**NLTK**

NLTK, o Natural Language Toolkit, es una biblioteca gratuita y de código abierto para el procesamiento avanzado del lenguaje natural (NLP) en Python [15]. 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, stemming, etiquetado, análisis y razonamiento semántico. También incluye envoltorios para librerías de NLP de nivel industrial y un foro de discusión activo. NLTK 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 [16].

**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 [17].

**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 |

**Tabla 1.0 Comparación de los modelos de reconocimiento de entidades**

Para este proyecto se elige la biblioteca de Spacy debido a su precisión del 90% lo que es relativamente alto en comparación con los otros modelos mencionados [(ver Tabla 1.0)](#comparacion_modelo_ner). El tamaño del modelo Spacy 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. Spacy permite personalizar y entrenar modelos en nuevos dominios o con nuevos tipos de entidades, lo cual vamos a utilizar para el reentrenamiento.

### Modelos generadores de oraciones

**GPT – 3.5**

Es la versión mejorada de GPT-3 lanzada en marzo de 2022 por OpenAI. Se conoce como GPT-3.5 y es parte de la familia GPT-3.5 que incluye varias variantes optimizadas, siendo la más avanzada gpt-3.5-turbo.

**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.

**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[18].

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Modelos** | **Llama2** | **GPT-3.5** | **GPT-4** |
| **Parámetros** | 70.000.000.000.000 | 154.000.000.000.000 – 175.000.000.000.000 | 1.000.000.000.000.000 – 176.000.000.000.000.000 |
| **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 |

**Tabla 1.1** **Comparación entre Llama 2, GPT-3.5 y GPT-4 para generar oraciones**

* **Tamaño y Parámetros**: Llama 2 es significativamente más pequeño que GPT-3.5 y GPT-4, con el modelo más grande de Llama 2 alrededor del 40-45% menor que GPT-3.5 y aproximadamente el 96% menor que GPT-4. Sin embargo, el tamaño no determina necesariamente la calidad o rendimiento del modelo ([ver Tabla 1.1)](#comparacion_modelo_oracion_gen).
* **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. Llama 2 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, siendo adecuados para tareas menos complejas y menos artísticas.
* **Token Limit**: 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.
* **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.
* **Caso de Uso Ideal para Llama 2**: Para construir un chatbot para pequeñas empresas, donde se requiere un manejo de solicitudes relativamente pequeñas y de complejidad moderada. También es una buena opción para herramientas que generen contenido de redes sociales de bajo riesgo debido a su nivel de creatividad limitado.[19]

### React

React es una biblioteca de JavaScript utilizada para construir interfaces de usuario. Permite a los desarrolladores crear aplicaciones web con componentes reutilizables que conforman partes de la interfaz de usuario, lo cual facilita el desarrollo evitando la repetición de código. React introduce JSX, una extensión de la sintaxis de JavaScript que permite combinar la lógica de JavaScript con la lógica de la interfaz de usuario, proporcionando una sintaxis similar a XML que es fácil de leer y escribir [20].

Algunas características clave de React incluyen:

* **Componentes Reutilizables**: Los componentes en React tienen su propia lógica y pueden ser reutilizados en toda la aplicación, lo que reduce la necesidad de repetir código.
* **Renderizado del lado del cliente**: React se utiliza principalmente para construir la interfaz de usuario en el lado del cliente, y se puede utilizar para construir cualquier tipo de aplicación web que se ejecute en la web.
* **Rendimiento**: Gracias al DOM virtual de React, la renderización de las páginas 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, facilitar el aprendizaje y reducir errores humanos. Su rendimiento puede compararse con NodeJS y Go y se considera uno de los marcos de trabajo más rápidos disponibles.[21]

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).
* **Intuitivo**: Ofrece excelente soporte de editor, autocompletado y verificación de tipos.
* **Fácil de usar y aprender**: Está diseñado para ser sencillo y requiere menos tiempo leyendo documentación.
* **Robusto**: Proporciona código listo para producción con documentación interactiva automática.
* **Basado en estándares**: Compatible con los estándares abiertos para APIs, OpenAPI y JSON Schema.[22]

### 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. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

Los principios y prácticas son de sentido común pero llevadas al extremo, de ahí proviene su nombre. Kent Beck, el padre de XP, describe la filosofía de XP en sin cubrir los detalles técnicos y de implantación de las prácticas. Posteriormente, otras publicaciones de experiencias se han encargado de dicha tarea. A continuación, presentaremos las características esenciales de XP organizadas en los tres apartados siguientes: historias de usuario, roles, proceso y prácticas.[23]

El ciclo de vida de XP consiste en seis fases:

* **Exploración**: En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.
* **Planificación de la Entrega:** En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días. Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la "velocidad" de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración. La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias. Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuántos puntos se pueden completar. Al planificar según alcance del sistema, se divide la suma de puntos de las historias de usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación.
* **Iteraciones:** Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción.
* Los elementos que deben tomarse en cuenta durante la elaboración del Plan de la Iteración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores. Wake en [18] proporciona algunas guías útiles para realizar la planificación de la entrega y de cada iteración.
* **Producción:** La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase. Es posible que se rebaje el tiempo que toma cada iteración, de tres a una semana. Las ideas que han sido propuestas y las sugerencias son documentadas para su posterior implementación (por ejemplo, durante la fase de mantenimiento).
* **Mantenimiento:** Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura.
* **Muerte del Proyecto:** Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.[23]

## Conclusiones del capítulo

Este capítulo proporciona una base para la comprensión del reconocimiento de entidades nombradas (NER) y su relevancia en el procesamiento del lenguaje natural (NLP). Se destaca la importancia histórica del término "entidad nombrada" y su evolución hasta convertirse en una técnica esencial en el procesamiento de lenguaje, utilizada para extraer, identificar y clasificar información en documentos de texto.

El capítulo también aborda la creciente necesidad de herramientas que permitan el reconocimiento de entidades nombradas en volúmenes masivos de datos, especialmente en contextos donde la información es textual, como es el caso de los datos almacenados en Elasticsearch. Se plantea el problema de investigación relacionado con las limitaciones de la empresa DATYS-CERPAMID en el reconocimiento de entidades nombradas al procesar grandes volúmenes de datos diariamente en Elasticsearch, y se establece el objetivo general de desarrollar una herramienta de NLP para abordar este problema.

Se describen los objetivos específicos del estudio, que incluyen la extracción de datos de Elasticsearch, el reconocimiento de entidades mediante el uso de modelos NER como Spacy, el aprendizaje supervisado para mejorar la precisión del modelo, el almacenamiento de resultados en Elasticsearch, y la evaluación y optimización continua del programa.

Además, se presenta un análisis detallado del estado del arte de los marcos de NLP para la detección de entidades, incluyendo Google Cloud NLP, ChatGPT, IBM Watson Discovery, Amazon Comprehend y Dandelion. Cada uno de estos marcos es evaluado en términos de sus ventajas y limitaciones, proporcionando una visión integral de las opciones disponibles para el desarrollo de la herramienta de NLP propuesta.

1. **ORGANIZACIÓN Y DISEÑO**

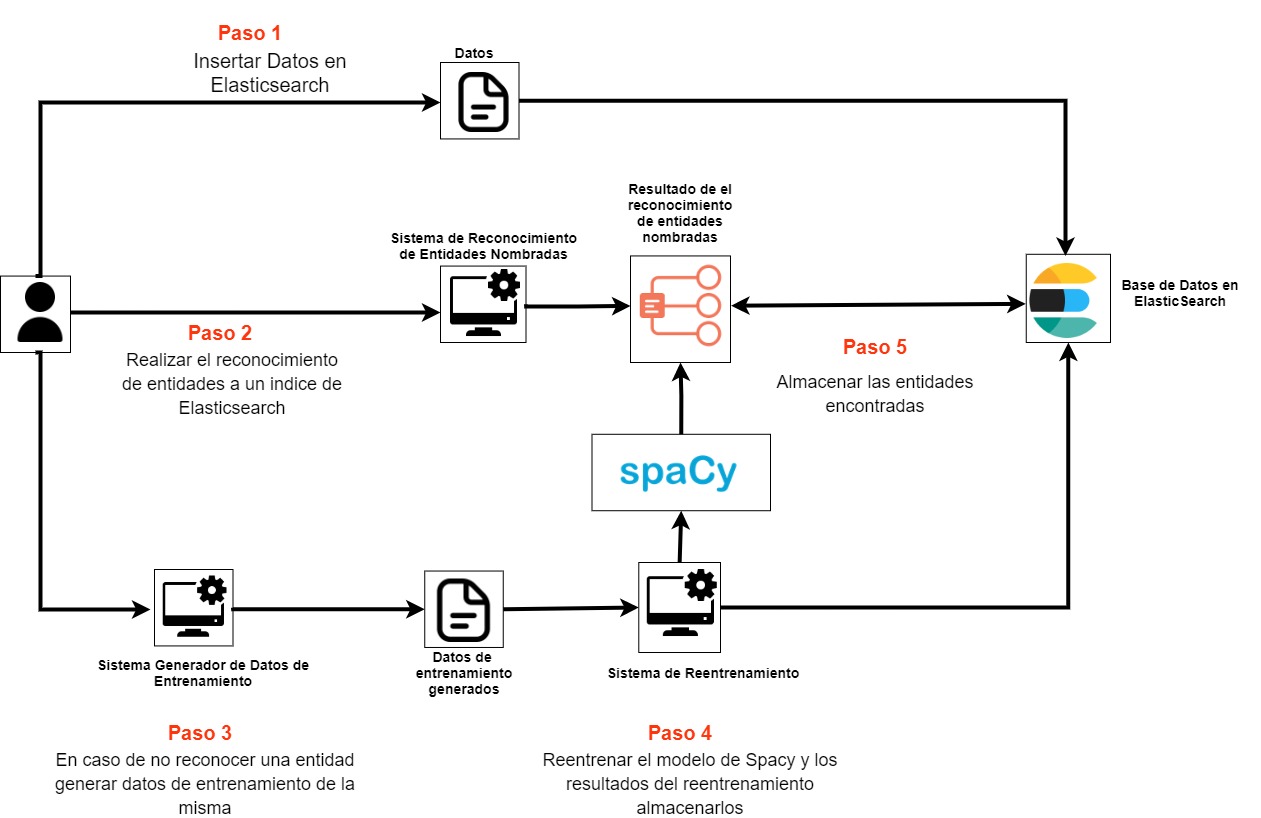
En el siguiente capítulo, se describirá el proceso de planificación y diseño del sistema, aplicando diversas técnicas y herramientas para construir un modelo detallado y preciso de la aplicación. Se exhibirán los diagramas utilizados para la modelación de la aplicación y la estrategia de desarrollo. Se analizarán los criterios de selección de las herramientas y técnicas empleadas, explicando las ventajas de su aplicación durante la fase de desarrollo del sistema. Este capítulo tiene como propósito ofrecer una visión integral y detallada del proceso de planificación y diseño del sistema, actuando como una guía para futuros proyectos similares.

## Propuesta del sistema

Para abordar el problema de investigación, 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 Elasticsearch podrían utilizar este sistema siempre que sus índices cumplan con las condiciones correspondientes para su procesamiento. Además, en caso de no reconocer una entidad, el usuario podría especificar el nombre de la entidad, 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 planteado anteriormente, se propone el siguiente diagrama: [(ver Fig 2.0)](#diagrama_arquitectura). Lo primero es que el usuario debe insertar los datos que desea hacer el reconocimiento de entidades en Elasticsearch en el formato correcto. Luego, podrá acceder al sistema que se propone, el cual ofrece acceso a todos los índices disponibles para el reconocimiento de entidades. Al seleccionar un índice, el sistema mostrará las entidades encontradas. En caso de que todo esté correcto, el usuario podrá guardar 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 mostrará los resultados obtenidos y, en caso de estar bien, el usuario reentrenará el modelo.



**Fig 2.0: 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 [24].

### Requistos funcionales del sistema

* RF 1 – Autenticación del usuario: Permite a los usuarios autenticarse en el sistema.
* RF 2 – Registro de 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.[25]

### 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 gen 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 debe brindar al usuario una clara navegabilidad de las funciones del sistema

### Confiabilidad

El sistema debe operar 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 precisos.

### Seguridad

El sistema garantizará la seguridad del espacio de trabajo de cada usuario mediante un proceso de Autenticación personal, permitiendo el acceso y la realización de acciones con la herramienta. La información recopilada se almacenará 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.

## Modelo Entidad Relación

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.[26]

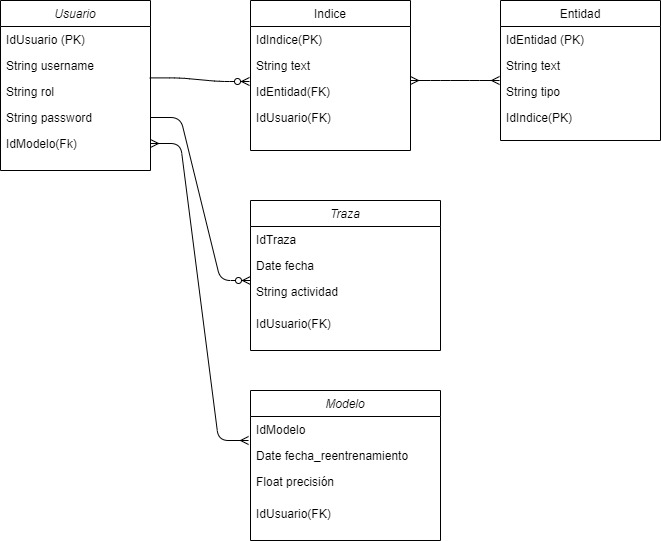
Pese a ser Elasticsearch una base de datos NOSQL, es una buena práctica la creación del modelo de entidad relación el cual refleje las tablas y campos que se van a manejar por el sistema en la base de datos.

Los índices que se representan son [(ver Fig 2.1)](#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 el al realizar el reconocimiento de entidades, los reentrenamientos al modelo, 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.

La relación entre las tablas representa las siguientes requerimientos o lógicas del negocio:

* 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.



**Fig 2.1 Modelo Entidad Relación**

## Historias de Usuario

Las historias de usuario 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[26].

* HU.1: Autenticación del usuario en el sistema
* HU.2: Inicio al sistema con datos de prueba
* HU.3: Reconocimiento de entidades nombradas de los índices de Elasticsearch
* HU.4: Reentrenamiento del modelo
* HU.5: Administrador de usuarios
* HU .6: Soporte en español e ingles
* HU .7: Reentrenamiento por especialista

**Tabla 2.0 Historia de Usuario: Autenticación de usuario en el sistema**

|  |  |
| --- | --- |
| **Historia de usuario** | |
| **Número**: 1 | **Nombre de Historia de Usuario**: Autenticación del usuario en el sistema |
| **Usuario**: Los trabajadores del centro | |
| **Prioridad** **en negocio**: Alta | **Riesgo de desarrollo**: Bajo |
| **Descripción**: Por temas de seguridad es necesario la autenticación al sistema por parte de los usuarios del centro. En caso de no tener cuenta deberá de registrarse | |
|  | |

**Tabla 2.1 Historia de Usuario: Inicio al sistema con datos de prueba**

|  |  |
| --- | --- |
| **Historia de usuario** | |
| **Número**: 2 | **Nombre de Historia de Usuario**: Inicio al sistema con datos de prueba |
| **Usuario**: Los trabajadores del centro | |
| **Prioridad** **en negocio**: Alta | **Riesgo de desarrollo**: Baja |
| **Descripción**: El usuario podrá hacer una revisión rápida de los modelos con datos de prueba que le proporcionamos. 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 | |
| **D:\DOWNLOAD\Screen Shot 2024-04-02 at 13.02.58.png** | |

**Tabla 2.2 Historia de Usuario: Reconocimiento de entidades nombradas de los índices de Elasticsearch**

|  |  |
| --- | --- |
| **Historia de usuario** | |
| **Número**: 3 | **Nombre de Historia de Usuario**: Reconocimiento de entidades nombradas de los índices de Elasticsearch |
| **Usuario**: Los trabajadores del centro | |
| **Prioridad** **en negocio**: Alta | **Riesgo de desarrollo**: Alto |
| **Descripción**: 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 | |
| **D:\DOWNLOAD\Screen Shot 2024-04-02 at 13.04.03.png** | |

**Tabla 2.3 Historia de Usuario: Reentrenamiento del modelo**

|  |  |
| --- | --- |
| **Historia de usuario** | |
| **Número**: 4 | **Nombre de Historia de Usuario**: Reentrenamiento del modelo |
| **Usuario**: Usuarios del sistema | |
| **Prioridad** **en negocio**: Alta | **Riesgo de desarrollo**: Alto |
| **Descripción**: 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 | |
| **D:\DOWNLOAD\Screen Shot 2024-04-02 at 13.14.27.png** | |

**Tabla 2.4 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 |
| **Descripción**: Los encargados de administrar los usuarios serán capaces de insertar nuevos usuarios, editarlos y eliminarlos del sistema | |
| **D:\DOWNLOAD\Screen Shot 2024-04-02 at 13.14.16.png** | |

**Tabla 2.5: 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 |
| **Descripción**: Los usuarios pueden cambiar el idioma al sistema, para español o para el ingles | |
|  | |

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

|  |  |
| --- | --- |
| **Historia de usuario** | |
| **Número**: 6 | **Nombre de Historia de Usuario**: Reentrenamiento por especialista |
| **Usuario**: Administradores del centro | |
| **Prioridad** **en negocio**: Alta | **Riesgo de desarrollo**: Bajo |
| **Descripción**: 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 | |
|  | |

## Usuarios del Sistema

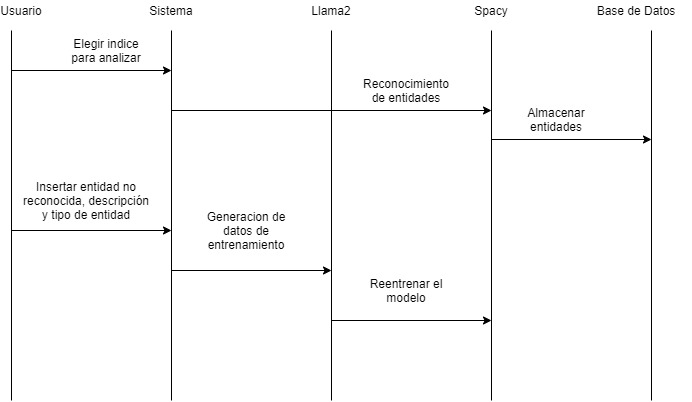
La entrada de usuarios al sistema está dividida en dos roles: usuario y administradores los cuales tienes tareas en específico que en dependencia de su rol podrán realizar en el sistema [(verFig 2.5)](#fig_user_sistem)

|  |  |
| --- | --- |
| **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 |

**Tabla 2.5 Usuarios del sistema**

## 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. [27]

****

**Fig 2.2 Diagrama de Secuencia**

El diagrama de secuencia de la [Fig 2.2](#diagrama_secuencia) muestra la interacción entre cuatro objetos:

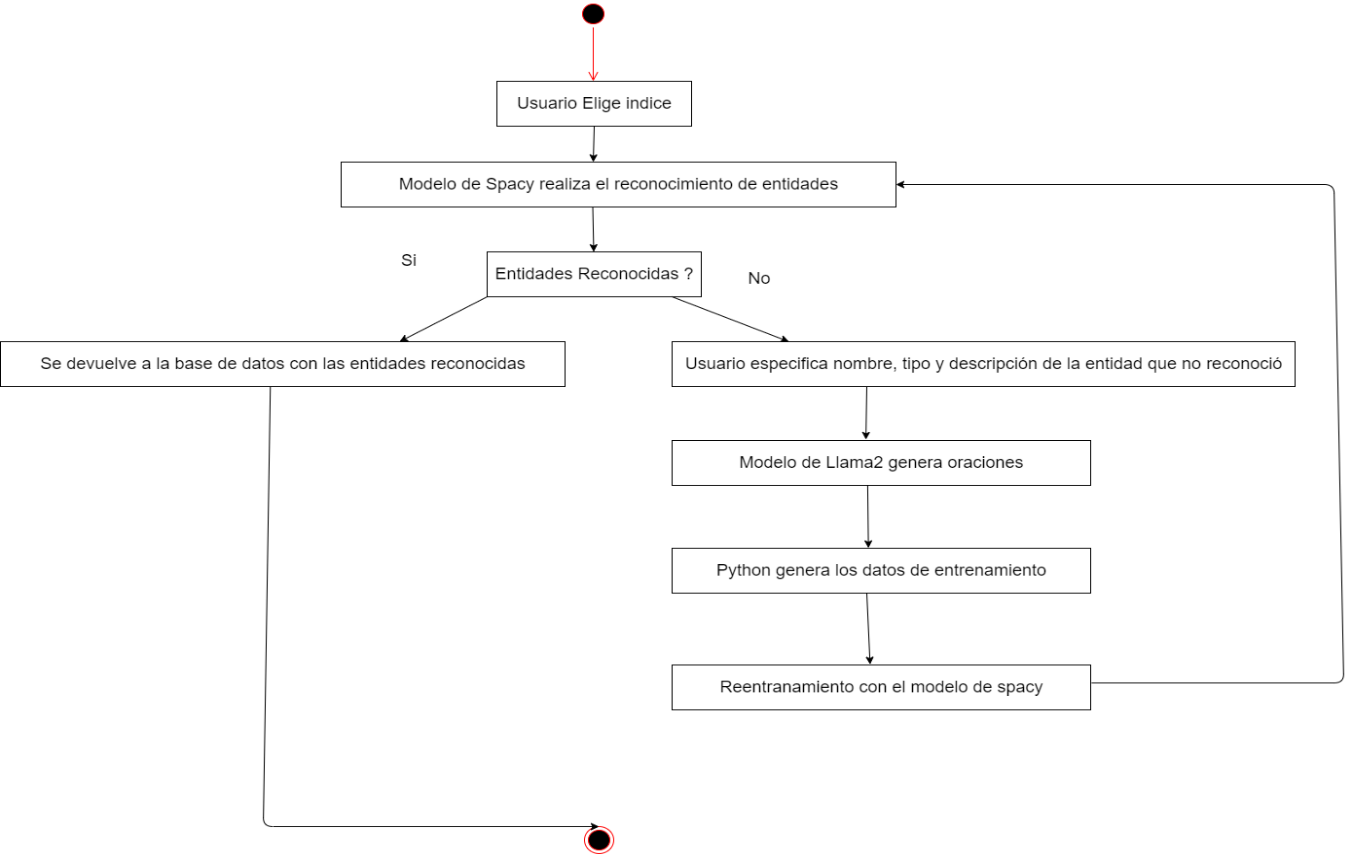
* **Usuario:** El usuario que interactúa con el sistema.
* **Sistema:** El sistema que proporciona el servicio al usuario.
* **Llama2:** Un modelo de lenguaje que genera oraciones.
* **Spacy:** Un modelo de lenguaje que realiza el reconocimiento de entidades.
* **Base de datos:** Elasticsearch sería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

En la figura X 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. [28]

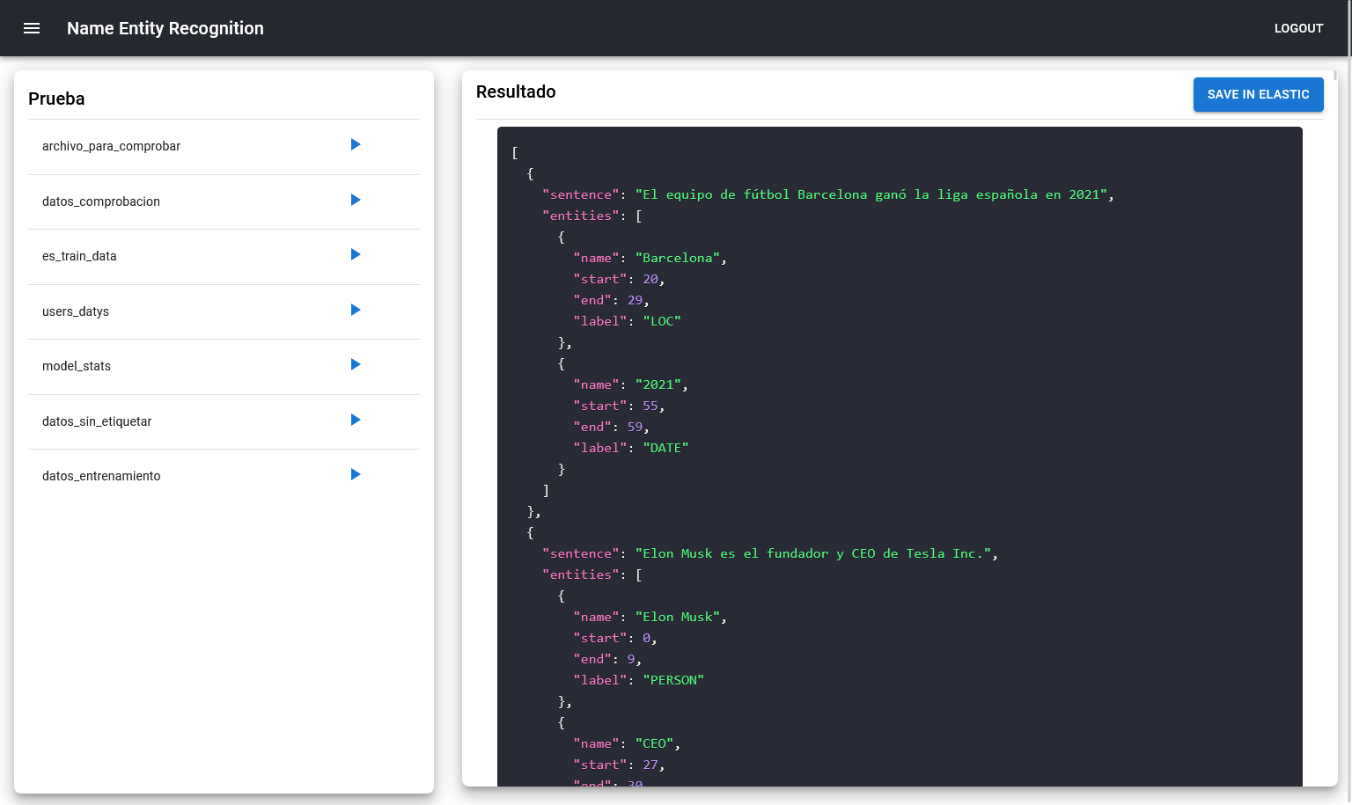
****

**Fig 2.3 Diagrama de Actividades**

El diagrama de la [figura 2.3](#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

****

**Fig 2.4 Diseño de la Interfaz**

El diseño de la interfaz de usuario presentado en la [Fig 2.4](#diseno_interfaz) posee varios puntos fuertes y áreas susceptibles a mejoras, con el fin de optimizar la experiencia de usuario:

**Aspectos positivos:**

* **Claridad:** La UI 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

En el capítulo actual se abordaron temas vinculados a las etapas de planificación y diseño en el desarrollo de software. Se describieron las funcionalidades a desarrollar, establecidas por los artefactos propuestos por la metodología XP, incluyendo los requisitos funcionales y no funcionales, así como las historias de usuario y las historias técnicas. Se presentó la base de datos no relacional empleada en la investigación y sus características particulares. También se resaltaron aspectos cruciales para la aplicación, como la seguridad, la usabilidad y la confiabilidad.

# 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 discutirán aspectos específicos de la implementación, los componentes utilizados y las consideraciones relevantes que surgieron durante el desarrollo. Además, se presentarán 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 X**.

## Algoritmos Importantes

A continuación, se analizarán 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):

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

@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")

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 hashear la contraseña del usuario utilizando “pwd\_context.hash(user.password)”, se crea un documento con el nombre de usuario, la contraseña hasheada y el rol del usuario, y 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)

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

@elastic\_router\_login.post("/login")

async *def* post\_login\_result(user: UserLogin):

    try:

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

        index = "users\_datys"

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

        for hit in user\_search.get("hits", {}).get("hits", []):

                user\_find = hit.get("\_source", {})

        password\_db, rol\_db =  user\_find['password'], user\_find['rol']

        if pwd\_context.verify(user.password, password\_db):

            print("Autenticación correcta")

            return rol\_db

        else:

            print("Fallo en la autenticación")

    except *Exception* as e:

        raise HTTPException(*status\_code*=500, *detail*=*str*(e))

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 sería [(ver Fig 3.3)](#fig_ner_principal_function):

**Fig 3.3 Función principal del reconocimiento de entidades nombradas**

@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 3.4 Función para cargar el modelo de Spacy**

*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 3.5 Función para extraer los datos del índice seleccionado**

*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 3.6 Función para procesar los documentos y extraer 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*

El código de la [Fig3.3](#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. Cargamos el modelo de spacy definido en path.py de la carpeta “sentence-gen-api/path.py” [(ver Fig3.4)](#fig_load_spacy_model) y lo asignamos a nlp. Luego se hace llamada a la función “fetch\_documents\_from\_elasticsearch” [(ver Fig3.5)](#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. Luego se procesa los documentos en la función “process\_documents” [(ver Fig3.6)](#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.

### 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 sería [(ver Fig 3.7)](#fig_salvar_res_ner):

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

@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

        settings = {

            "mappings": {

                "properties": {

                    "text": {"type": "text"},

                    "entities": {"type": "nested"}

                }

            }

        }

        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))

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 inidice 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.

**Fig 3.8 Función principal para generar los datos de entrenamiento**

@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\_gen\_chain = load\_summarize\_chain(

*llm*=LLM,

*chain\_type*="refine",

*verbose*=True,

*question\_prompt*=prompt,

*refine\_prompt*=refine\_prompt,

        )

        question\_list = []

        docs = text\_process(input.text)

        for \_ in range(2):

            questions = question\_gen\_chain.run(docs)

            question\_list.append(questions)

        oraciones = []

        for texto in question\_list:

            oraciones.extend(extraer\_oraciones(texto))

*# Estructurar las oraciones en el formato JSON deseado*

        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 3.9 Función para dividir un texto largo en fragmentos más pequeños**

from typing import List

from langchain.docstore.document import Document

from langchain\_text\_splitters import CharacterTextSplitter

*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*

**Fig3.10 Función para generar datos de entrenamiento**

*def* generar\_data\_train(*json\_oraciones*, *entity*, *entity\_type*):

    output\_dir = Path("D:\Tesis2\modelo-nuevo-es")

    nlp = spacy.load(output\_dir)

    for i, oracion in enumerate(json\_oraciones, *start*=1):

        doc\_text = oracion['sentences']

        doc\_text = re.sub(*r*'\(.\*?\)|\[.\*?\]|\'.\*?\'', '', doc\_text)

        doc = nlp(doc\_text)

        entities = []

        for ent in doc.ents:

            entities.append({'name': ent.text, 'start': ent.start\_char, 'end': ent.end\_char, 'label': ent.label\_})

        entity\_exists = any(e['name'] == entity for e in entities)

        if not entity\_exists:

            for token in doc:

                if token.text == entity:

                    start = token.idx

                    end = token.idx + len(token.text)

                    entities.append({'name': token.text,'start': start, 'end': end, 'label': entity\_type})

        entities.sort(*key*=*lambda* *x*: x['start'])

        json\_oraciones[i-1]['entities'] = entities

    return json\_oraciones

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. 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 Fig3.9)](#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. 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.

### Reentrenar el modelo de Spacy

Una vez que el usuario haya revisado los datos de entrenamiento generados por el modelo y verifique que reconoce correctamente la entidad entonces puede reentrenar el modelo.

**Fig 3.11 Función principal del reentrenamiento del modelo**

@elastic\_router\_train\_model.post('/train\_model\_es')

async *def* post\_save\_in\_elastic(post: ModelTrainData):

    try:

        save\_train\_data(post, "es\_train\_data")

        arr = getPythonzonas()

        TRAIN\_DATA = prepare\_train\_data(arr)

        n\_iter = 100

        nlp = load\_or\_create\_model(output\_dir)

        configure\_ner(nlp)

        examples = convert\_to\_examples(TRAIN\_DATA, nlp)

        train\_model(nlp, examples, n\_iter)

        nlp.to\_disk(output\_dir)

    except *Exception* as e:

        print(f"Error en train model es: Error: {e}")

        raise HTTPException(*status\_code*=500, *detail*=*str*(e))

**Fig 3.12 Función para preparar los datos de entrenamiento**

*def* prepare\_train\_data(*arr*):

    TRAIN\_DATA = []

    for el in arr:

        texto = el["\_source"]["text"]

        entities = el["\_source"]["entities"]

        entities\_formatted = [(entity["start"], entity["end"], entity["label"]) for entity in entities]

        resultado = (texto, {"entities": entities\_formatted})

        TRAIN\_DATA.append(resultado)

    return TRAIN\_DATA

**Fig 3.13 Función para cargar o crear el modelo de reconocimiento de entidades**

*def* load\_or\_create\_model(*output\_dir*):

    if os.path.exists(output\_dir):

        print("Cargando el modelo desde", output\_dir)

        return spacy.load(output\_dir)

    else:

        print("Creando un nuevo modelo en blanco")

        return spacy.blank('es')

**Fig 3.14 Función para configurar el componente NER del modelo Spacy**

*def* configure\_ner(*nlp*):

    if 'ner' not in nlp.pipe\_names:

        ner = nlp.create\_pipe('ner')

        nlp.add\_pipe('ner', *last*=True)

    else:

        ner = nlp.get\_pipe('ner')

    ner.cfg["noisereduce"] = True

**Fig 3.15 Función para convertir los datos de entrenamiento en ejemplos**

*def* convert\_to\_examples(*TRAIN\_DATA*, *nlp*):

    examples = []

    for text, annotations in TRAIN\_DATA:

        examples.append(Example.from\_dict(nlp.make\_doc(text), annotations))

    return examples

**Fig 3.16 Función para reentrenar el modelo de Spacy**

*def* train\_model(*nlp*, *examples*, *n\_iter*):

    other\_pipes = [pipe for pipe in nlp.pipe\_names if pipe != 'ner']

    with nlp.disable\_pipes(\*other\_pipes):

        optimizer = nlp.begin\_training()

        for itn in range(n\_iter):

            random.shuffle(examples)

            losses = {}

            for batch in spacy.util.minibatch(examples, *size*=compounding(4.0, 32.0, 1.001)):

                nlp.update(

                    batch,

*drop*=0.5,

*sgd*=optimizer,

*losses*=losses)

            print(losses)

Las funciones proporcionadas anteriormente son parte de un proceso para reentrenar el modelo de Spacy. Cada función contribuye al proceso general [(verFig 3.11)](#fig_principal_reentrenamiento) por ejemplo la función “prepare\_train\_data(arr)” [(ver Fig 3.12)](#fig_preparar_datos_entrenamiento) la cual procesa los datos de entrada para formatearlos adecuadamente para el entrenamiento del modelo. La función “load\_or\_create\_model(output\_dir)” [(ver Fig 3.13)](#fig_cargar_modelo_ner_o_nuevo) se encarga de verificar si existe un modelo previamente entrenado en el directorio especificado por el usuario en output\_dir, si el modelo existe lo carga, sino crea uno nuevo. La función “configure\_ner(nlp)” [(ver Fig 3.14)](#fig_config_ner) se encarga de las configuraciones del componente NER de Spacy. Luego la función “convert\_to\_example “([ver Fig 3.15](#fig_convertir_datos_en_examples)) que se encarga de convertir los datos de entrenamiento formateados, en ejemplos que el modelo Spacy puede utilizar para el reentrenamiento. Y la función “train\_model” [(ver Fig 3.16)](#fig_reentrenar_spacy) la cual es la que se encarga de entrenar el modelo con los ejemplos proporcionados, durante un numero especificado de iteraciones. Durante el entrenamiento se desactiva todos los otros componentes del modelo excepto el reconocimiento de entidades que es por el cual se está reentrenando.

## 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 procesamiento del lenguaje natural (PLN) puede ser crucial para estimar los costos asociados con este tipo de proyecto. El PLN puede ayudar a extraer, estructurar e interpretar información relevante de diversas fuentes, lo que 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. 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.

## 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 [29], 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 bajo las cuales debe ser probado. 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.

**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 |

**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 |

**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 |

**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 erroneos | El sistema lanza una alerta, y da la opción de volver a repetir la generación de datos | El sistema creo datos de entrenamiento erroneos |
| Elige la opción de generar datos de entrenamiento | EL modelo si pierde precisión no se guardan los cambios realizados | Reentrenamiento del modelo pierde precisión |

**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 |

## Conclusiones del capítulo

En este capítulo se dio los pasos para realizar la instalación y uso de las herramientas y tecnologías usadas. Se describió la implementación de las principales funcionalidades del sistema. Se realizó un análisis económico y una estimación de costos del mismo. Se realizaron pruebas que permitieron ver las capacidades para responder en diferentes situaciones que se pueden dar.

1. **CONCLUSIONES Y RECOMENDACIONES**

**Conclusiones**

* Se logró establecer una conexión exitosa con Elasticsearch y extraer los datos almacenados en la base de datos, asegurando la integridad y calidad de la información recuperada.
* Se desarrolló y puso en práctica una funcionalidad de reconocimiento de entidades utilizando el modelo de Spacy, logrando la identificación y clasificación automática de entidades como nombres de personas, organizaciones y ubicaciones.
* Se implementó un proceso de aprendizaje supervisado, para cuando el usuario generará datos de entrenamiento se reentrenará el modelo para así mejorar su precisión en el reconocimiento de entidades.
* Los resultados identificados de manera estructurada se almacenaron en Elasticsearch, facilitando un acceso eficiente y consultas posteriores.
* Se realizó una evaluación del rendimiento del programa, mostrando resultados satisfactorios en la precisión y eficiencia del reconocimiento de entidades y la efectividad del proceso de aprendizaje supervisado.
* Se estableció un enfoque de optimización continua para mejorar el programa a medida que se recopilan más datos y se identifican nuevas entidades, 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.
* Integrarle nuevas tecnologías como es el caso del uso de otras bases de datos como MongoDB.

1. **REFERENCIAS BIBLIOGRÁFICAS**

**BIBLIOGRAFIA**

[1] R. Viana, «Reconocimiento de entidades nombradas: Una breve introducción», SoldAI. Accedido: 8 de noviembre de 2023. [En línea]. Disponible en: https://medium.com/soldai/reconocimiento-de-entidades-nombradas-una-breve-introducci%C3%B3n-ad6e2a3f911a

[2] V. Osorio, «¿Qué es el Reconocimiento de Entidades Nombradas?», Klippa. Accedido: 25 de octubre de 2023. [En línea]. Disponible en: https://www.klippa.com/es/blog/informativo/reconocimento-entidades-nombradas/

[3] «SanchezPCR.pdf».

[4] «¿Qué es el procesamiento de lenguaje natural? - Explicación del procesamiento de lenguaje natural - AWS», Amazon Web Services, Inc. Accedido: 7 de octubre de 2023. [En línea]. Disponible en: https://aws.amazon.com/es/what-is/nlp/

[5] «Top 5 named entity recognition APIs to use in your app». Accedido: 17 de diciembre de 2023. [En línea]. Disponible en: https://datavid.com/blog/named-entity-recognition-apis

[6] D. Santos, «Procesamiento de lenguaje natural: qué es, ejemplos y herramientas». Accedido: 9 de noviembre de 2023. [En línea]. Disponible en: https://blog.hubspot.es/marketing/procesamiento-de-lenguaje-natural

[7] «¿Qué es Python? - Explicación del lenguaje Python - AWS», Amazon Web Services, Inc. Accedido: 9 de noviembre de 2023. [En línea]. Disponible en: https://aws.amazon.com/es/what-is/python/

[8] «Why Visual Studio Code?» Accedido: 9 de noviembre de 2023. [En línea]. Disponible en: https://code.visualstudio.com/docs/editor/whyvscode

[9] «Lista de características de Elasticsearch | Elastic». Accedido: 7 de octubre de 2023. [En línea]. Disponible en: https://www.elastic.co/es/elasticsearch/features

[10] «Qué es Lucene. Qué es Lucene y qué relación tiene con Solr.» Accedido: 7 de octubre de 2023. [En línea]. Disponible en: https://solrtutorial.es/que-es-lucene.html

[11] B. Marijan, «What Is Git and What Is It Used For? {+ Main Features}», Knowledge Base by phoenixNAP. Accedido: 9 de noviembre de 2023. [En línea]. Disponible en: https://phoenixnap.com/kb/what-is-git

[12] «Probamos SpaCy: mucho más que una librería para crear proyectos reales de procesamiento del lenguaje natural | datos.gob.es». Accedido: 7 de octubre de 2023. [En línea]. Disponible en: https://datos.gob.es/es/blog/probamos-spacy-mucho-mas-que-una-libreria-para-crear-proyectos-reales-de-procesamiento-del

[13] «BERT 101 - State Of The Art NLP Model Explained». Accedido: 6 de febrero de 2024. [En línea]. Disponible en: https://huggingface.co/blog/bert-101

[14] «flair | flair». Accedido: 6 de febrero de 2024. [En línea]. Disponible en: https://flairnlp.github.io/

[15] J. Karajgikar, «Guides: Text Analysis: NLTK Package». Accedido: 6 de febrero de 2024. [En línea]. Disponible en: https://guides.library.upenn.edu/penntdm/python/nltk

[16] «NLTK :: Natural Language Toolkit». Accedido: 6 de febrero de 2024. [En línea]. Disponible en: https://www.nltk.org/

[17] «Stanford NLP | Stanford NLP Python | Stanford NLP Tutorial». Accedido: 6 de febrero de 2024. [En línea]. Disponible en: https://www.analyticsvidhya.com/blog/2019/02/stanfordnlp-nlp-library-python/

[18] K. Luzniak, «6 main differences between Llama 2, GPT-3.5 & GPT-4», Neoteric. Accedido: 25 de octubre de 2023. [En línea]. Disponible en: https://neoteric.eu/blog/6-main-differences-between-llama2-gpt35-and-gpt4/

[19] K. Luzniak, «6 main differences between Llama 2, GPT-3.5 & GPT-4», Neoteric. Accedido: 10 de febrero de 2024. [En línea]. Disponible en: https://neoteric.eu/blog/6-main-differences-between-llama2-gpt35-and-gpt4/

[20] «Qué es React: definición, características y funcionamiento». Accedido: 6 de febrero de 2024. [En línea]. Disponible en: https://www.hostinger.es/tutoriales/que-es-react

[21] «Features - FastAPI». Accedido: 6 de febrero de 2024. [En línea]. Disponible en: https://fastapi.tiangolo.com/features/

[22] S. Ramírez, «tiangolo/full-stack-fastapi-postgresql». 6 de febrero de 2024. Accedido: 6 de febrero de 2024. [En línea]. Disponible en: https://github.com/tiangolo/full-stack-fastapi-postgresql

[23] P. Letelier y P. Letelier, «Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP)», www.cyta.com.ar/ta0502/v5n2a1.htm. Accedido: 20 de febrero de 2024. [En línea]. Disponible en: http://www.cyta.com.ar/ta0502/b\_v5n2a1.htm

[24] «Requerimientos – RUP». Accedido: 2 de abril de 2024. [En línea]. Disponible en: https://rupnoobs.wordpress.com/2016/03/12/requerimientos/

[25] A. Jain, «Qué son los Requisitos No Funcionales: Ejemplos, Definición, Guía Completa», Visure Solutions. Accedido: 2 de abril de 2024. [En línea]. Disponible en: https://visuresolutions.com/es/blog/requerimientos-no-funcionales/

[26] «¿Qué es el modelo entidad relación en BBDD?», UNIR. Accedido: 2 de abril de 2024. [En línea]. Disponible en: https://www.unir.net/ingenieria/revista/modelo-entidad-relacion/

[27] M. Alam, «¿Qué es el diagrama de secuencia? Diagramas de definición y secuencia en UML», IdeaScale. Accedido: 3 de abril de 2024. [En línea]. Disponible en: https://ideascale.com/es/blogs/que-es-el-diagrama-secuencia/

[28] «Tutorial de diagrama de actividades UML», Lucidchart. Accedido: 3 de abril de 2024. [En línea]. Disponible en: https://www.lucidchart.com/pages/es/tutorial-diagrama-de-actividades-uml

[29] «Estimating With Use Case Points». Accedido: 3 de abril de 2024. [En línea]. Disponible en: https://www.mountaingoatsoftware.com/articles/estimating-with-use-case-points

1. **GLOSARIO DE TÉRMINOS**

Se utiliza cuando durante el desarrollo se han utilizado palabras, conceptos, siglas, etc., que no expresan una idea clara, precisa y concisa, o que puedan tener múltiples interpretaciones

Cuando aparece una sigla por primera vez, ella se escribe normalmente y entre paréntesis se pone el significado de sus términos en el idioma original y a su vez se pone en el Glosario de términos, cuando aparece nuevamente no es necesario poner entre paréntesis el significado de la sigla.

El glosario de términos se anexa después de la bibliografía y antes de los anexos y se escribe en orden alfabético, Se recomienda no abusar de los anglicismos y si se utilizan deben escribirse en itálicas

1. **ANEXOS**

En este apartado se abordará sobre el análisis económico y el sistema en funcionamiento

**Anexo I** 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 3.1 se presenta el Facto 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 |

**Tabla 3.1: Factor de peso de los actores sin ajustar**

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 3.2 se muestra la clasificación de complejidad y cálculo de peso:

|  |  |  |  |
| --- | --- | --- | --- |
| **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** |

**Tabla 3.2 Peso de las Historias de Usuario sin Ajustar**

**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 3.3 se muestra factor de complejidad técnica con su significado y el peso de cada uno de estos factores:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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** |

**Tabla 3.3 Factor de complejidad técnica**

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 3.4 se muestra factor de ambiente con su significado y el peso de cada uno de estos factores:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 |

**Tabla 3.4: Factor Ambiente**

**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 Tabla3.5) se muestra 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 |

**Tabla3.5 Distribución del esfuerzo por etapas**

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.

|  |  |
| --- | --- |
| **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 |

**Tabla 3.6 Situación actual de pago**

**Anexo II** Sistema en Funcionamiento

Resultados de Reconocimiento de Entidades a Indice de Elasticsearch

{

"sentence": "El teamacere perdio el juego de pelota contra el equipo de los Estados Unidos",

"entities": [

{

"name": "Estados Unidos",

"start": 63,

"end": 77,

"label": "LOC"

}

]

}

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:

{

"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"

}

]

}

}

Al reentrenar el modelo y volverlo a pasar por el indice de Elasticsearch nos da el siguiente resultado:

{

"sentence": "El teamacere perdio 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"

}

]

}

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

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