ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

GRADO EN INGENIERÍA TELEMÁTICA

EXTRACCIÓN DE CONCEPTOS MÉDICOS EN HISTORIAS CLÍNICAS ELECTRÓNICAS

Autor: Pío Iglesias Estévez

Director: Israel Alonso Martínez

Madrid

Junio 2018

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Extracción de conceptos médicos en historias clínicas electrónicas

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

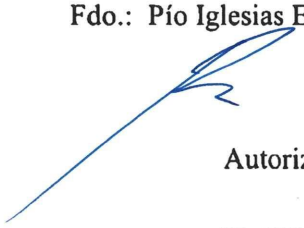
curso académico 2017/18 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: Pío Iglesias Estévez Fecha: 12 / Julio / 2018



Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Israel Alonso Martínez Fecha: 12 / Julio / 2018

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor D. Pío Iglesias Estévez

DECLARA ser el titular de los derechos de propiedad intelectual de la obra: Extracción de conceptos médicos en historias clínicas electrónicas, que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente).*

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 12 de Julio de 2021

ACEPTA

Fdo. Pío Iglesias Estévez

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

GRADO EN INGENIERÍA TELEMÁTICA

EXTRACCIÓN DE CONCEPTOS MÉDICOS EN HISTORIAS CLÍNICAS ELECTRÓNICAS

Autor: Pío Iglesias Estévez

Director: Israel Alonso Martínez

Madrid

Junio 2018

Agradecimientos

Gracias a toda mi familia, mi pareja, mis abuelos, mis padres (1,2 o incluso 3), mis tíos destacando a mi tía que ha sido como una madre para mí durante mi estancia en Madrid por apoyarme durante toda esta etapa, pero sobre todo a mi madre que si no fuese por la exigencia a la que me ha sometido en el día a día en todos los aspectos de la vida probablemente no estaría escribiendo estas líneas.

Gracias a todas las personas que me han apoyado a nivel académico cuando más lo necesitaba, compañeros y amigos que nunca dudaron a la hora de echarme un cable, de prestarme unos apuntes o de dejarme ver la solución de un problema que no fuese capaz de resolver.

Gracias a las personas responsables del Trabajo de fin de grado, al coordinador y sobre todo a mi director que ha sido comprensivo, paciente y me ha ayudado enormemente a cumplir los plazos de entrega.

EXTRACCIÓN DE CONCEPTOS MÉDICOS EN HISTORIAS CLÍNICAS ELECTRÓNICAS

Autor: Iglesias Estevez, Pio.
Director: Alonso Martínez, Israel.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas.

RESUMEN DEL PROYECTO

Palabras clave: Historias clínicas electrónicas, software libre, UMLS, conceptos clínicos

1. Introducción

En España en los últimos datos recogidos en 2017 se cuenta con la historia clínica de 35 millones de personas, un 73% de la población y un avance exponencial desde los 6 millones que cubrían en 2011[1] por ello surge la posibilidad de manejar esos datos con un ordenador para procesarlos con mayor velocidad y para darles mayor utilidad. Existe un gran Metatesauro de términos que se denomina UMLS[2], en este se encuentran gran cantidad de términos clínicos y las relaciones entre ellos en cuanto a similitud semántica.

Gracias a la existencia del diccionario UMLS y de las historias clínicas en formato electrónico, y la herramienta de Apache llamada UIMA que permite a las aplicaciones descomponerse en varios componentes y así analizar información y relaciones, nace Apache cTAKES[3]. Apache cTAKES con el soporte UIMA analiza historias clínicas electrónicas y extrae información de estas de todo tipo, separa palabra por palabra semánticamente y etiqueta los términos que sean clínicos conforme a su código en el diccionario de términos dentro de UMLS. Esta información se devuelve en un archivo y en forma de código. cTAKES cuenta con dos entornos gráficos para su uso además de poder usarlo mediante línea de comandos, a pesar de ser un entorno amable ambos no son muy útiles para su uso por parte de personas no técnicas y los resultados que se obtienen no sirven de gran utilidad directa.

2. Definición del proyecto

Este Proyecto trata de la construcción de un entorno software para darle una mayor utilidad a la herramienta Apache cTAKES a nivel usuario en formato web. Este entorno está programado en Python y actúa como una capa entre el usuario y la aplicación de Apache facilitando los datos que necesita para su funcionamiento ya sea la selección de la carpeta de la que se desea extraer información o la introducción de las credenciales de UMLS para poder contrastar la información con el diccionario, también facilita la forma en la que se devuelven los resultados devolviendo solamente un resumen con los términos clínicos encontrados en ella. Permite a los usuarios después de ello recabar más información acerca del término clínico en el diccionario o guardar los resultados en un archivo de texto para un uso futuro.

3. Descripción del modelo/sistema/herramienta

El elemento principal del sistema es la herramienta Apache cTAKES.

El entorno de la aplicación web está desarrollado en Python y la web también con la ayuda del Framework Django. El sistema además está formado por un server de cTAKES de elaboración propia y el servidor de UMLS. A ambos, la web envía peticiones y recibe como respuesta los resultados que pretende para llevar a cabo su función.

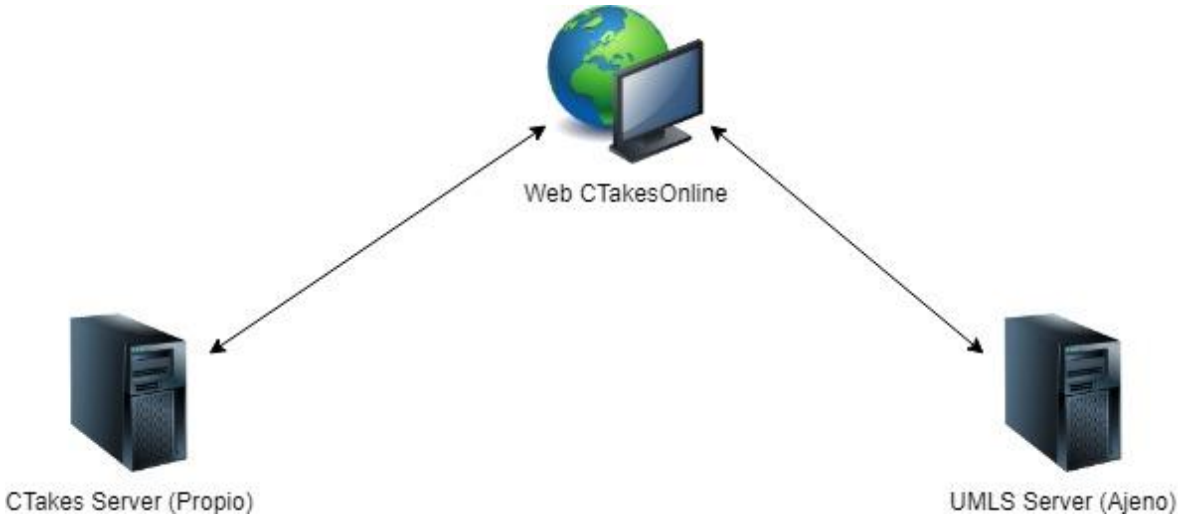


Ilustración 1 – Arquitectura del sistema

La web envía al server de cTAKES un texto, este le devuelve el texto analizado del cual la web se queda solamente con los términos clínicos y su código de identificación única (CUI).

En el caso del server de UMLS la web envía los CUIs para obtener de ella la descripción del término, los términos relacionados y el tipo semántico del mismo.

4. Resultados

El sistema permite extraer información de historias clínicas obteniendo todos los términos clínicos que aparecen en ellas y asociados a ellas sus códigos de identificación única en el diccionario de términos médicos, se permite guardar al usuario esta información en un archivo de texto para su posterior uso y para un ahorro en la cantidad de información que almacena. También se permite al usuario buscar toda la información acerca de ese término clínico con solamente un clic, aunque esta función esté todavía en pruebas y sea defectuosa en algunos casos por causa de los límites que tiene UMLS en su API, probablemente pidiendo que incorporen el usuario utilizado a una “lista blanca” sería suficiente para resolver esa cuestión.

Ilustración 2 – Vista de resultados

5. Conclusiones

Gracias al desarrollo de este proyecto se ha conseguido una manera más sencilla de utilizar la herramienta de extracción de información de historias clínicas Apache cTAKES mediante una página web en la que poder escribir las historias clínicas o subirlas desde un archivo de texto para extraer los términos clínicos en ellas y que además a posteriori permite ver información a mayores sobre los términos clínicos encontrados, además de ser todo de manera totalmente gratuita al utilizar software libre y código abierto.

6. Referencias

[1] Redacción. "La historia clínica en España: 35,7 millones digitales; 11 millones físicas". Redacción Médica. 2017 https://www.redaccionmedica.com/secciones/parlamentarios/la-historia-clinica-en-espana-35-7-millones-digitales-11-millones-fisicas-7937

[2] Wikipedia. "Unified Medical Language System" https://en.wikipedia.org/wiki/Unified_Medical_Language_System

[3] Apache cTAKES. http://cTAKES.apache.org/

EXTRACTION OF MEDICAL CONCEPTS FROM ELECTRONIC MEDICAL RECORDS

Author: Iglesias Estevez, Pio.
Director: Alonso Martínez, Israel.
Partner entity: ICAI – Universidad Pontificia Comillas.

ABSTRACT

Palabras clave: Electronic medical records, free software, UMLS, medical concepts

1. Introduction

In Spain in the last data recovered in 2017 the medical records of 35 million people is in electronic format, 73% of the population and an exponential increasement since the 6 million that have it in 2011[1]. Because of that, emerges the possibility of managing this data with a computer to process them faster and to make them more meaningful. There is a big Methathesaurus of terms called UMLS[2], in which a high number of medical terms and relationships between them by semantic type are represented.

With the existence of UMLS (the medical terms dictionary) and the medical records in electronic format and Apache tool called UIMA, a tool which allows applications to be divided into modules and to analyze information and relations, Apache cTAKES [3] is born.Apache cTAKES supported by UIMA analyzes medical records and extracts all kind of information from them, it splits word by word semantically and tags the clinical terms found with the correspondent code in the UMLS dictionary. This information is given in a file in coding format. cTAKES has two graphic interfaces to use it and also can be run from command shell even thought is not an easy to use tool for people without technical background and the results are not very useful directly.

2. Project definition

This project is about building a software environment to give more utility to Apache cTAKES for non-technical users and to be accessible via web. This environment is programmed in Python and it is used as an interface between the user and cTAKES making it easier to take advantage of it. It is no need to be registered in UMLS or to know which pipeline to use, everything is given. Also, the results of the extraction are modified to only get the medical terms and its information. It also allows the user to continue to search more in advance about a medical term or instead to print the results into a text file to use them in the future.

3. System description

The main element in the system is Apache cTAKES tool.

The environment of the web application is also programmed in Python and the web is also developed with the help of Django Framework. The parts of the system are, a cTAKES server self-done and the UMLS server. Both send and receive requests from the web application.

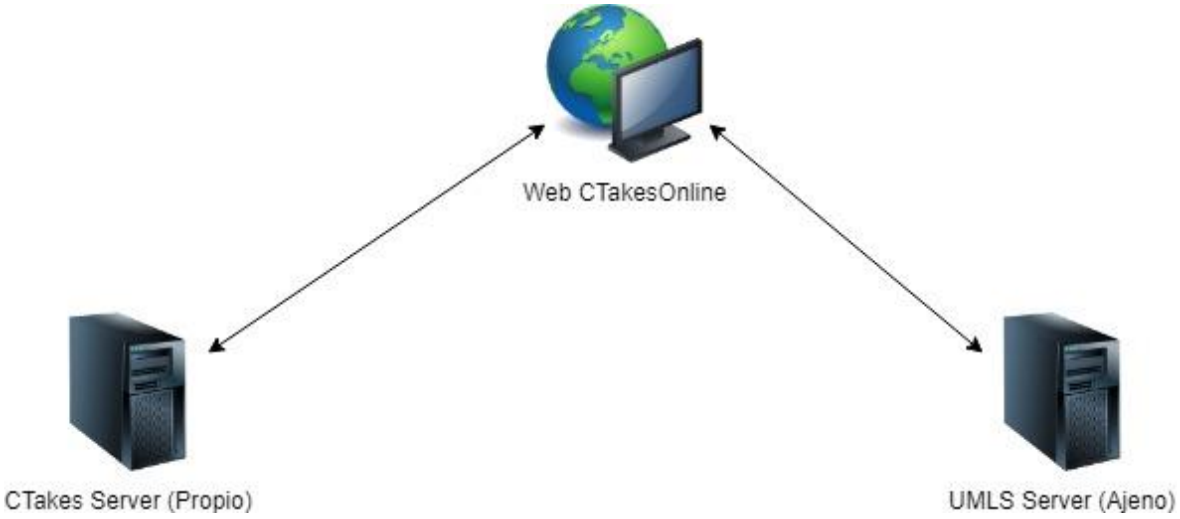


Ilustración 1 - Arquitectura del sistema

The web sends toc TAKES server a medical record text and it send back the text analyzed, with that the web filters the texts and only shows medical terms and its code of unique identification (CUI).

In this case the UMLS server send the CUIs to obtain from them the description of the medical term, the terms related to it and the semantic type.

4. Results

The system allows to extract medical concepts from medical records obtaining all the medical concepts which appear in them and pairing to them its code of unique identification in the UMLS dictionary, making it possible to save that information in a file which if it is processed in the future it will be faster and less heavy to save it in terms of size. It also allows the user to look for all the information about the medical term with only a click, even thought that function sometimes is not running because of the limits on the UMLS licence to use their API.

Ilustración 2 - Vista de resultados

5. Conclusions

Thanks to the development of this project it was possible to make it easier to use the medical records information extraction tool Apache cTAKES with a web applications where it is possible to write medical records or to upload them from a text file and to extract the medical concepts in it and after that to be able to search for more information about the medical concepts found in addition to be free to use free software and open source.

6. References

[1] Redacción. “La historia clínica en España: 35,7 millones digitales; 11 millones físicas”. Redacción Médica. 2017 https://www.redaccionmedica.com/secciones/parlamentarios/la-historia-clinica-en-espana-35-7-millones-digitales-11-millones-fisicas-7937

[2] Wikipedia. “Unified Medical Language System” https://en.wikipedia.org/wiki/Unified_Medical_Language_System

[3] Apache cTAKES. http://cTAKES.apache.org/

Índice de la memoria

Índice de ilustraciones

Índice de tablas

Capítulo 1. INTRODUCCIÓN

En España en los últimos datos recogidos en 2017 muestran que se cuenta con la historia clínica de 35 millones de personas, un 73% de la población, existiendo un avance exponencial desde los 6 millones que cubrían en 2011 por ello surge la posibilidad de manejar esos datos digitalmente para procesarlos con mayor velocidad y para darles mayor utilidad. [1]

Esta cuestión, en el campo de la medicina y en demás campos en el mundo actual, hace que también haya una carrera por lograr las herramientas más eficientes posibles para poder llevar a cabo esa mejora en la gestión de los datos. Nos enfrentamos a un volumen cada vez mayor de éstos, pero también hay que tener en cuenta que las mejoras en la potencia de los equipos informáticos también favorecen la mejora en la gestión de estas cantidades cada vez mayores de información.

Las mejoras en la gestión de los datos almacenados van totalmente de la mano de la inteligencia artificial, con ella se puede dotar a una máquina de diversas instrucciones, entrenarla, para poder llevar a cabo los procesos por ella misma.

Aunando todo ellos surgen los programas que analizan las historias clínicas, analizando el lenguaje natural y etiquetan los términos clínicos que aparecen en ellas, actualmente son todas ellas herramientas que funcionan a nivel código y que permiten obtener solamente resultados de este tipo en su uso estándar.

Para mejorar este aspecto y permitir el uso de este tipo de programas por parte de usuarios del mundo de la medicina no familiarizados con el mundo de la informática surge este proyecto. En él se lleva a cabo un análisis exhaustivo de la herramienta de software open-source Apache cTAKES, la más precisa de las que existen en la actualidad, se estudia las posibilidades que ofrece de manera estándar y se lleva a cabo un entorno gráfico y un desarrollo que permite utilizarla con mayor facilidad. Al ser una herramienta que además

funciona por módulos podría incluso en el futuro incluirse los desarrollos como parte de la propia herramienta e incluir más funcionalidades. Se analizará por su puesto también cómo funcionan los diccionarios UMLS y la manera de utilizarlos por parte de este programa. Este es en todo caso, un primer paso a la hora de facilitar su uso, pero según vaya mejorando la herramienta, con el tiempo, se podrán dar muchos más.

Todavía queda mucho margen de mejora en este campo, como todos los sistemas en los que se analiza lenguaje natural siempre hay error, aunque cada vez sea menor. También la ampliación en los diccionarios a distintos idiomas no solamente en inglés que es el de mayor tamaño, donde la raíz etimológica coincide en muchos términos de manera global pero no en todos.

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

En este capítulo se presentan las diferentes tecnologías empleadas en el desarrollo de este Trabajo de Fin de Grado, con el objetivo de facilitar la comprensión en mayor detalle de la implementación realizada.

En este caso se describirán las tecnologías utilizadas por la aplicación Apache cTAKES y las tecnologías a nivel software utilizadas complementariamente para el desarrollo de la aplicación web de la que trata el proyecto.

2.1 SOFTWARE

En este proyecto todo el desarrollo es software, se ha elegido una herramienta basada en Java como es Apache cTAKES que puede utilizarse en todos los sistemas operativos del mercado. No obstante, no es de gran importancia dado que en este proyecto la aplicación se lanza en un servidor al cual se tiene acceso mediante un Webservice Restful. El uso de REST es clave en este proyecto ya que, el cliente (la web) se comunica continuamente tanto con el servidor propio de la herramienta de extracción, tanto con los servicios ofrecidos por UMLS en su API Rest, para obtener más información acerca de las historias clínicas.

A la hora de programar, el lenguaje que más facilita el uso de las API Rest es Python, por lo tanto, ha sido el lenguaje elegido. También, como innovación se buscó la manera de poder llevar a cabo esta web completamente en Python cosa que se pudo lograr gracias al web Framework Django que es lo más parecido que existe al desarrollo web en Java para el lenguaje Pyhon. Gracias a Django se mantiene la uniformidad en el proyecto permitiendo darle cohesión a todo desde un mismo entorno.

Django está inspirado en la filosofía de desarrollo modelo vista controlador (MVC) a pesar de que la nomenclatura interna del framework no se corresponde con este nombre ya que el teórico controlador se llama vista en Django y la teórica vista se llama plantilla.[2]

Debe hacerse mención también a la herramienta Apache Maven, utilizada en numerosas ocasiones en programas de código abierto en internet y que es utilizada para la composición del servidor de Apache cTAKES creado, esta herramienta permite construir un proyecto desde solamente un archivo de información además de los archivos de código, lanzando la herramienta Maven se descargarán todas las librerías mencionadas en el archivo de información y se compilará y construirá el proyecto.

Apache UIMA es la implementación que usa cTAKES y otras muchas aplicaciones que gestionan información no estructurada, así se descomponen en distintos componentes que envían como respuesta un archivo descriptivo XML que el framework se encarga de gestionar, así como el flujo de datos entre ellos.

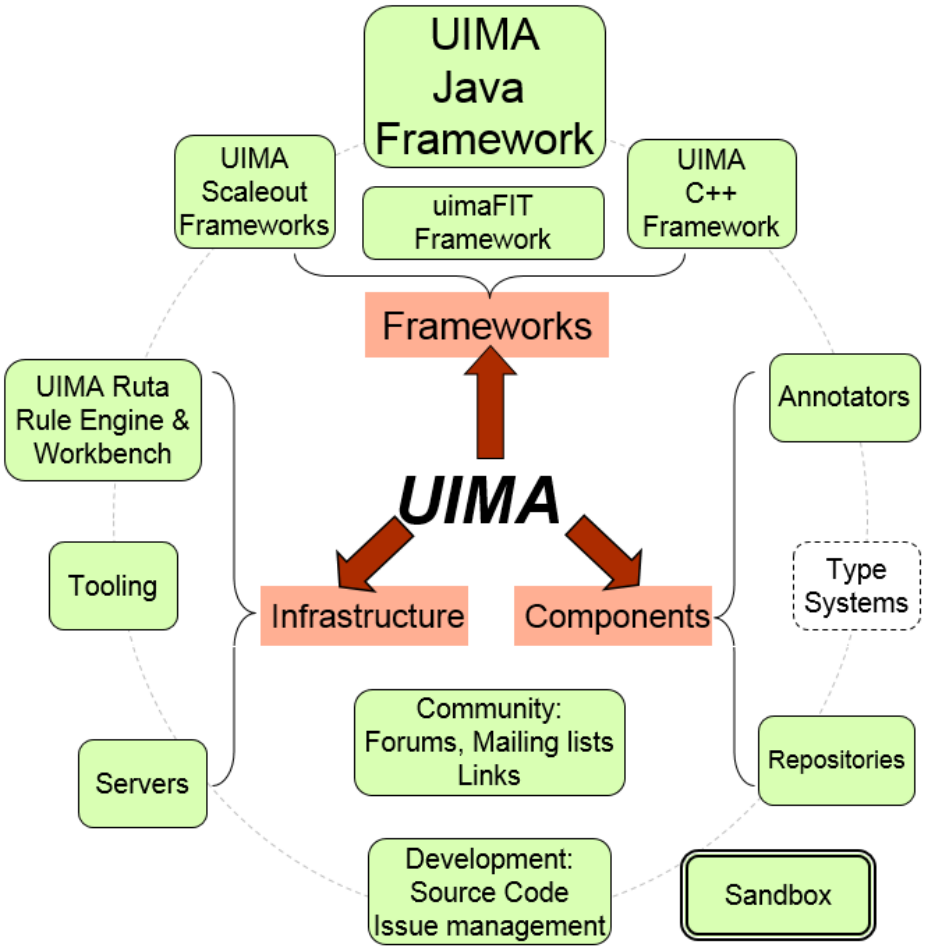


Ilustración 1. Diagrama de componente de Apache Uima. Fuente: https://uima.apache.org/images/UimaIs.png

Open NLP es la librería de apache basada en machine learning enfocada al procesamiento de lenguaje natural en textos que utiliza cTAKES. Soporta las tareas más comunes de NLP como son la detección del lenguaje, la tokenización, la segmentación de oraciones, el etiquetado "part-of-speech", la extracción de entidades nombradas, parseo y la resolución de referencias cruzadas.

También el uso de expresiones regulares para así obtener la información que se quería ha sido bastante útil gracias a la web regex101.com

2.2 DICCIONARIOS Y METATESAURO DE TÉRMINOS CLÍNICOS. UMLS

El Unified Medical Language System (UMLS) es un compendio de distintos recursos controlados orientados al campo de las ciencias biomédicas. Proporciona una estructura mapeada de los mismos y permite por lo tanto traducir mediante varios sistemas de terminología. También es utilizado como un índice de términos y conceptos biomédicos. Actualmente facilita mucho las tareas del procesado de lenguaje natural y es, como es en este caso, utilizado en gran medida para temas relacionados entre la medicina y la informática.

El UMLS fue diseñado por la Biblioteca Nacional de medicina de los Estados Unidos, es actualizado cuatrimestralmente y se puede usar de manera totalmente gratuita.

En cuanto a su propósito y aplicaciones los recursos para los investigadores son casi ilimitados, aunque esto también puede ser un problema por la cantidad de documentación que hay asociada cuando se busca dentro de la bibliografía médica. El propósito de UMLS es facilitar el acceso a esa bibliografía.

El Metatesauro es la base del UMLS y comprende más de 1 millón de términos biomédicos y 5 millones de nombres de conceptos todos ellos bajo el umbral de 100 alfabetos, estos son ICD-10, SNOMED CT, MeSH, Rx Norm, etc.

Está organizado por conceptos y cada uno tiene atributos específicos definiendo su significado y está asociado a su nombre conceptual en el alfabeto correspondiente. Numerosas relaciones entre los conceptos están representantes, por ejemplo, las relaciones jerárquicas como “is-a” para subclases y “forma parte de” para subunidades y asociativas como “es causado por” o “suele suceder junto con”.

El alcance del Metatesauro está determinado por el alcance de los alfabetos de los que está compuesto. Si los distintos alfabetos utilizan nombres distintos para un mismo concepto, o si usan el mismo nombre para diferentes conceptos, esto será reflejado de manera fidedigna en el Metatesauro. Toda la información jerárquica de los alfabetos está incluida en el Metatesauro. Los conceptos de este pueden también enlazar a otros recursos fuera de la base de datos como por ejemplo a una base de datos de secuencias de genes.

Cada uno de estos conceptos está asignado a uno o más tipos semánticos o categorías, los conceptos están relacionados mediante estas relaciones semánticas. La red semántica es un catálogo de todos los tipos semánticos. Esta es una clasificación muy amplia, hay 127 tipos semánticos y 54 relaciones en total. Los tipos semánticos principales son: los organismos, estructuras anatómicas, funciones biológicas, sustancias químicas, eventos, objetos físicos y conceptos o ideas. Los enlaces entre los tipos semánticos definen la estructura de la red y enseñan relaciones de importancia entre "grupings" y conceptos. El enlace primario entre los tipos semánticos es el "is-a", que establece jerarquía entre los tipos. La red cuenta también con 5 categorías principales de relaciones de tipo no-jerárquico (ascociativo), que constituyen los 53 tipos de relaciones restantes. Estas son "relacionado físicamente a", "espacialmente relacionado con", "relacionado funcionalmente con" y "relacionado conceptualmente a".

La información sobre el tipo semántico incluye un identificador, definición, ejemplos, información jerárquica sobre lo que abarcan el tipo y sus relaciones asociativas. Las relaciones asociativas dentro de la red semántica son muy débiles. Capturan en la mayoría de los casos en los que la relación tiene un significado. Por ejemplo "puede causar" junto con fumar y cáncer de pulmón forman una relación asociativa en "fumar puede causar cáncer de pulmón".

Otra característica de UMLS es el SPECIALIST Lexicon que contiene información sobre vocabulario común inglés, términos biomédicos, términos encontrados en MEDLINE y términos encontrados en el Metatesauro de UMLS. Cada entrada contiene información sintáctica, morfológica y ortográfica. Un set de programas java utiliza el lexicon para trabajar

acerca de las variaciones producidas en términos biomédicos relacionando palabras con partes del discurso, que pueden ser de ayuda en búsquedas web o en búsquedas sobre una historia clínica electrónica.

A pesar de todo UMLS cuenta con inconsistencias y errores dado su tamaño y su complejidad y la permisividad en cuanto a añadir nuevos términos, la ambigüedad y la redundancia son errores típicos que se pueden dar dentro de él. [3]

2.3 SEGURIDAD

El acceso a la API Rest de UMLS se encuentra protegido por un protocolo de autenticación del tipo Kerberos, el cual es un protocolo creado por el MIT en el que se distinguen tres "cabezas" o entidades, de ahí el nombre: un cliente, el cual quiere acceder a un servicio, un servidor que ofrece dicho servicio y un tercero de confianza: el Key Distribution Center (KDC) o centro de distribución de claves, compuesto por un Authentication Server (AC) o servidor de autenticación y un Ticket Granting Server (TGS) o servidor emisor de tickets.

Ilustración 2. Protocolo Kerberos. Fuente oficial: http://web.mit.edu/kerberos/

En este caso, el rol de cliente corresponde a nuestra aplicación, el rol de servidor (que oferta un servicio) le corresponde a la API de UMLS, concretamente al endpoint "**https://uts-ws.nlm.nih.gov/rest**", al cual se realizan las consultas. Finalmente, el role del KDC queda relegado al endpoint "**https://utslogin.nlm.nih.gov**", identificado en la propia documentación de la API de UMLS como el Authentication Service Endpoint. [4]

A continuación, se explica el funcionamiento de este protocolo en este entorno, el cual queda ilustrado en la imagen inferior a estas líneas. En primer lugar, el cliente se autentica contra el Authentication Service Endpoint (dentro del KDC), el cual cuenta con métodos para obtener tanto un Ticket Granting Ticket (TGT), como Service Tickets (ST) de un único uso a partir del mismo. El cliente puede autenticarse proporcionando sus credenciales (usuario y contraseña válidos) o un API Key, el cual ha sido obtenido en este caso para llevar a cabo el proyecto. Una vez haya obtenido el TGT, el usuario (cliente) podrá autenticarse ante cualquier servidor que "confíe" en el Authentication Service Endpoint, pudiendo demostrar que ya ha sido reconocido por el KDC (el tercero de confianza) y que por tanto pasa a formar parte de su círculo de confianza. No obstante, sólo lo será de forma temporal, ya que la validez del TGT es limitada, concretamente, caduca a las 8 horas de su obtención, obligando al cliente a solicitar uno nuevo, autenticándose una vez más facilitando sus credenciales o API Key al KDC. La seguridad que proporciona este protocolo se refuerza aún más con un segundo tipo de ticket expedido por el KDC sólo a aquellos clientes que presenten un TGT válido: el Service Ticket (ST), el cual es un ticket "personalizado", ya que ha de ser solicitado por un cliente para un servicio (proporcionado por un servidor) en concreto. Esto significa que un ST sólo es válido si lo usa el cliente que lo haya solicitado y solamente si lo usa contra el servidor que proporciona el servicio solicitado previamente al KDC. Los ST caducan a los 5 minutos y tan sólo tienen un único uso, por lo que antes de cada consulta que se vaya a realizar a la API, el cliente (la aplicación desarrollada) tendrá que solicitar un ST nuevo al KDC, para lo cual deberá haber solicitado previamente un TGT, el cual sea todavía válido.

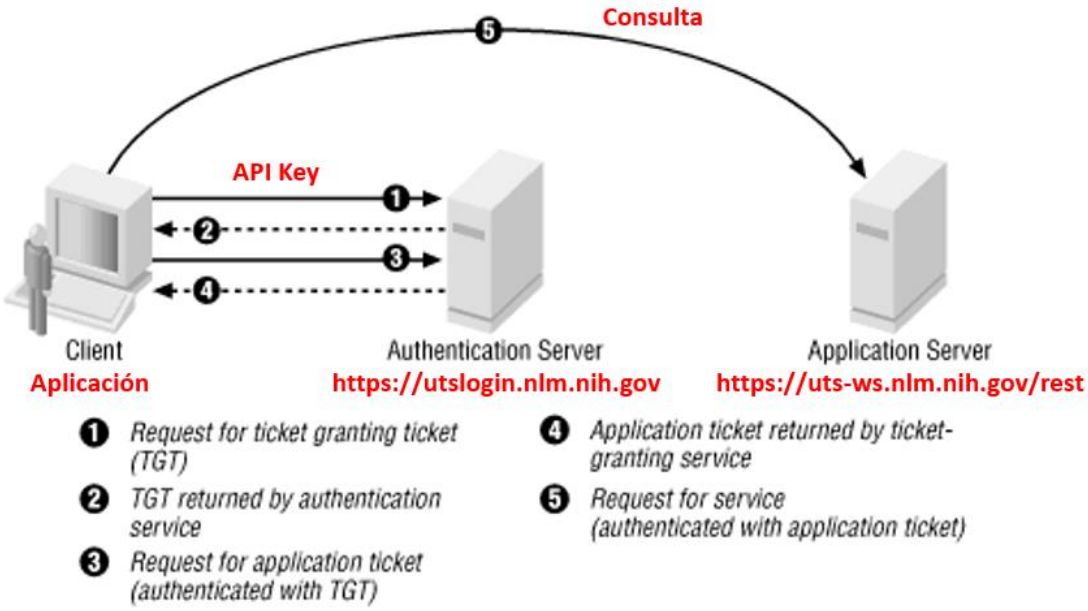


Ilustración 3. Protocolo Kerberos aplicado a la API REST de UMLS. Fuente: https://docs.typo3.org/typo3cms/extensions/ig_ldap_sso_auth/SSO/Kerberos.html

Capítulo 3. ESTADO DE LA CUESTIÓN

La narrativa clínica tiene características únicas que la diferencian de la literatura científica biomédica y del dominio general, esta requiere un esfuerzo centrado en sus propias metodologías dentro del campo de la NLP.

La Universidad de Columbia es la propietaria del Sistema de extracción y codificación de lenguaje médico (MedLEE) [7], este sistema fue diseñado para procesar informes de radiología, pero posteriormente se extendió a otros dominios siendo probado también por otras instituciones. MedLEE descubre qué términos son clínicos dentro de un conjunto de modificadores.

El extractor de textos de información sanitaria (HITEx) [9]es un sistema de NLP clínico open-source perteneciente al Brigham and Women's Hospital y la escuela médica de Harvard lo incorporó a su conjunto de herramientas para integrar la informática y la biología.

Los sistemas BioTeKS y MedKAT [8] de IBM fueron desarrollados para el dominio biomédico y son sistemas NLP.

SymText y MPLUS [10]son sistemas que han sido utilizados para interpretar escáneres de pulmón y detectar pneumonías.

Existen también otras herramientas desarrolladas con el objetivo de procesar artículos científicos que incluyen el MetaMap de la Biblioteca Nacional de Medicina y esto les ha permitido obtener como resultados mapeos para los conceptos incluidos en el metasaurio del Sistema de Lenguaje Médico Unificado (UMLS). También de diversas investigaciones para el cáncer por parte de universidades se han obtenido más conceptos para este MetaMap. En general, existe ya un gran número de herramientas open-source para el manejo de NLP. El OpenNLP suite implementa un clasificador machine learning de máxima entropía el cual permite maximizar la información derivada de un texto. Buyko y su equipo de trabajo han conseguido adaptar OpenNLP a la literatura biomédica científica demostrando que su exactitud es comparable a haber analizado un texto de las noticias. Este sistema contribuye

al progreso en materia del NLP al ser especialmente dedicado para la narrativa clínica, ser modular, extensible, robusto y opensource además de haber sido ya totalmente probado. [5]

3.1 HERRAMIENTAS EXISTENTES

3.1.1 GATE

General Architecture for Text Engineering es un suite de herramientas Java desarrollado por la Universidad de Sheffield desde 1995 y que actualmente es utilizado por gran cantidad de compañías para el procesamiento de lenguaje natural y extracción de información en múltiples idiomas, entre ellos para la extracción de información de historias clínicas como en el caso que nos concierne. [6]

3.1.2 MEDLEE

Medical Language Extaction and Encoding System:

El objetivo de MedLee es extraer, estructurar y codificar la información clínica contenida en las historias clínicas para que estos datos puedan ser utilizados a posteriori por sistemas automatizados. [7]

3.1.3 MEDKAT

MedKAT (Medical Knowledge Analysis Tool) es una herramienta orientada al campo médico y patológico, contiene compontes para extraer características específicas del cáncer de textos no estructurados. Está basado en los principios de NLP, y contiene componentes basados en reglas y en machine-learning. MedKAT está construido sobre el framework open source UIMA, que consiste en un conjunto de módulos (anotadores), cada uno con un archivo configuración en XML. En términos generales, los anotadores etiquetan un documento de texto no estructurado insertando “anotaciones” que están asociadas a una parte

particular del texto. La secuencia de ejecución está descrita también en un archivo de configuración, pero todos ellos son modificables.

El pipeline de MedKAT está dividido en los siguientes tipos de componentes:

Ingestión de documentos: Determinan la estructura del documento.

Procesamiento de lenguaje natural general: Componentes para tokenizar, acotar oraciones, etiquetar términos part-of-speech y parsear.

Búsqueda de conceptos: Los componentes que determinan los conceptos, basado en la terminología, detecta negaciones.

Búsqueda de relaciones: componente que popularizan el modelo de conocimiento de la enfermedad del cáncer y encuentra referencias. [8]

3.1.4 HITEX

HITEx (Health Information Text Extraction) es una herramienta open-source de procesado de lenguaje natural (NLP) desarrollada por un grupo de investigadores del Brigham and Women's Hospital y la Harvard Medical School. HITEx está construido sobre el framework GATE y también lo usa como plataforma. Es una colección de plug-ins para GATE que han sido desarrollados para resolver problemas del dominio de la medicina, como extracción del diagnóstico, extracción de la medicación. Trabaja conjuntando estos plug-ins en una aplicación pipeline y en conjunto con otros plug-ins estándar de NLP (algunos forman parte de Gate, como Part-of-Speech tagger o Noun Phrase Chunker). Cada plug-in del pipeline recoge el resultado del anterior. Permite reordenar la forma en la que se realiza el proceso. [9]

3.1.5 MPLUS

Mplus es una herramienta estadística que permite a los investigadores contar con una herramienta flexible para analizar sus datos, existe gran variedad de modelos, estimadores y algoritmos, este sistema también se ha entrenado para analizar historias clínicas pero para hacer confluencias de datos a gran escala en materia de genética. El uso de esta herramienta exige un gasto mínimo de 250 euros si eres de fuera de USA y el precio para estudiantes en USA entre 125 y 175 dólares. [10]

Algunas de estas herramientas, la mayoría, son de uso privado, en el caso de MEDLEE cuenta con su propia App Web como lo desarrollado en este proyecto, pero solamente pueden usarla los usuarios, no existe una herramienta de software libre a nivel usuario creada con el fin de que se pueda utilizar vía web para leer historias médicas clínicas y analizarlas de manera, todas son aplicaciones de escritorio que se consiguen de una manera u otra lanzar desde un servidor. Posiblemente tenga que ver con temas de protección de datos ya que las historias clínicas contienen información sensible de pacientes y si no hay un compromiso detrás de la página web de no almacenar esa información, o si lo hace que sea dentro de los criterios de la LOPD.

Capítulo 4. DEFINICIÓN DEL TRABAJO

En este capítulo se describen la justificación y motivaciones por las cuales se lleva a cabo este proyecto intentando hincapié en la elección de cTAKES en vez de otras herramientas para la extracción de términos clínicos y las justificaciones por las necesidades del mundo actual. Posteriormente se ofrece una explicación de los objetivos que se pretende alcanzar con la realización del proyecto, siendo estos enumerados y analizados globalmente. También se realiza un estudio del tiempo que conlleva el desarrollo del sistema, así como una estimación de los costes que, en este caso no ha tenido pero si fuese llevado a cabo por personas asalariadas.

4.1 JUSTIFICACIÓN

En el campo de la medicina es en uno de los campos en los que a mayor velocidad se ha producido el traslado de lo analógico a lo tecnológico, en apenas 7 años la situación ha cambiado totalmente, como se explicaba en la introducción, se ha pasado de contar con la historia clínica de apenas unos pocos millones de personas a contar con más de treinta y no se debe perder el tiempo a la hora de aprovechar las ventajas que tiene contar con la información en formato electrónico.

La extracción de conceptos médicos en historias clínicas es un proyecto estrictamente relacionado con las necesidades de hoy día, la gestión de grandes volúmenes de datos y conseguir darles una utilidad, no solamente contentarse con almacenarlos para que en un futuro se les pueda sacar partido, ese futuro es hoy. Gracias a lo implementado en este proyecto se conseguirá hacer más fácil la tarea de una persona que tiene ante ella una historia clínica con mucha información contenida, a la hora de buscar de qué términos clínicos se está hablando en ella y poder saber todo tipo de información acerca de esos términos en el caso de desconocer alguno o querer saber con qué puede estar relacionado.

A parte de la tarea concreta llevada a cabo es importante también el mero hecho de haber investigado acerca de las herramientas que existen para la extracción de información en historias clínicas, y sobre todo en concreto de la elegida, Apache cTAKES ya que la información sobre ellas en internet es escasa y existen numerosos foros de dudas, preguntas o reportes de errores sin ser contestados o resueltos y el hecho de haber trabajado con la herramienta y haberse enfrentado alguno de ellos y haber encontrado soluciones ayuda a la comunidad a continuar desarrollando sobre estas herramientas que están en pleno crecimiento y ayuda a que puedan crecer más.

Como se ha comentado en el capítulo precedente, existen diversas herramientas de software para la extracción de información de historias clínicas que utilizan el lenguaje natural. La mayoría de ellas hacen el mismo trabajo que la utilizada en este proyecto.

A pesar de ello, todas estas soluciones cuentan con distintos problemas:

- Algunas de ellas no son de código abierto, no son modificables por parte del usuario para poder utilizar a su antojo sus funcionalidades.

- No son ni gratuitas, ni está permitido su uso por parte de cualquiera, pertenecen a grupos de investigadores.

- No existe ninguna herramienta de extracción de información de historias clínicas que se pueda usar en una web, de manera cómoda y sencilla apta para cualquier usuario.

- La herramienta Apache cTAKES consigue extraer el término clínico y su identificador único, pero no devuelve directamente más información sobre el término buscado como por ejemplo su descripción completa o sus relaciones.

Una vez observadas y analizadas estas cuestiones se procede a pensar una solución que aúne todo, y esta es una web en la que se puedan escribir historias clínicas o leerlas de un archivo de texto y que nos permita obtener primeramente los términos clínicos que aparezcan en él

y tras ello o exportarlo en un archivo de texto como resumen de este o permitir al usuario buscar más información acerca de ellos.

4.2 OBJETIVOS

Elegir los objetivos del proyecto no fue una tarea sencilla. En este caso, dada la naturaleza de la herramienta utilizada, código abierto, que permite, si se consigue comprender su funcionamiento totalmente, llevar a cabo casi cualquier objetivo que se proponga en el campo de la extracción de información de historias clínicas tardando un menor o mayor tiempo y requiriendo un menor o mayor esfuerzo según la dificultad de los objetivos a alcanzar. En este caso dadas las dificultades para comprender en un comienzo el funcionamiento de la herramienta y posteriormente las dificultades al implementarla llevaron a reducir el alcance del proyecto y por lo tanto a ponerse objetivos más sencillos y realistas que continuasen siendo útiles pero que se pudiesen cumplir dentro de los plazos programados. Evidentemente esta situación fue producida por comenzar de cero absolutamente en la materia por lo que los objetivos en este caso no pueden ser los mismos que contando con un *background* inicial sabiendo exactamente lo que se quiere obtener del estudio y del sistema a desarrollar. En un principio el estudio fue centrado en qué se puede sacar de la herramienta cTAKES y posteriormente se fijaron los objetivos según lo observado, aunque antes de investigar acerca de cTAKES fue necesario un acercamiento al entorno de la interpretación de lenguaje natural y del diccionario UMLS y su funcionamiento.

A continuación, se mostrarán los objetivos propuestos y cumplidos finalmente por el sistema desarrollado.

Estos son los siguientes:

- Aprender acerca de la relación de la NLP con los conceptos clínicos.

- Entender el funcionamiento de Apache cTAKES, estudiar las posibilidades que puede ofrecer, investigar acerca de sus módulos y qué tarea lleva a cabo cada uno, además de aprender a instalarlo.

- Entender el funcionamiento de los diccionarios de UMLS. Sus conceptos, relaciones, como funciona y como sacarle el mayor partido posible a la información que contiene.

- Desarrollar una manera de usar Apache cTAKES que sea plausible para el usuario medio y desde cualquier lugar.

- El sistema debe de estar basado en componentes de código abierto y por lo tanto ser totalmente gratuito.

- Se debe de crear una interfaz amable para el usuario e intuitiva a la vez que apta para cualquier dispositivo con internet desde el cuál se quiera utilizar.

- El sistema debe ser seguro, evitando el almacenamiento de información sensible en acuerdo con el usuario, no guardar las historias clínicas en las bases de datos simplemente utilizarlas para la extracción y desecharlas.

- El sistema debe de permitir utilizar historias clínicas sea cual sea su tamaño y procesarlas en un tiempo razonable.

- El sistema debe permitir al usuario elegir qué quiere hacer en todo momento.

- El sistema debe de permitir la creación de resúmenes y así poder exportar los resultados obtenidos en un documento externo

- Permitir el acceso vía web a cualquier usuario que lo desee y así evitar el tedioso y no apto para nivel usuario, proceso de instalación de cTAKES.

- Proporcionar toda la información de utilidad acerca de los términos clínicos encontrados

4.3 METODOLOGÍA

Para poder alcanzar los objetivos que se han descrito, se ha utilizado una metodología que es iterativa e incremental pero también modular en ciertos aspectos.

El por qué de este uso de métodos distintos es que antes de poder desarrollar el sistema fue necesario un proceso de investigación largo acerca de la herramienta que se iba a utilizar, que ella misma está basada en un *framework* que también era importante comprender como es Apache UIMA, también acerca del procesamiento de lenguaje natural, todo lo que permite y como es usado por la herramienta y finalmente investigar acerca del diccionario UMLS y los datos que ofrece acerca de los conceptos clínicos. La primera vez para comprender todo fue una etapa mucho más larga que los análisis posteriores para repasar algún detalle o para resolver un error o problema surgido durante el desarrollo.

En lo relativo al diseño e implementación sí se usó la metodología iterativa e incremental más pura. Primeramente, se llevó a cabo de manera rápida una primera versión para ir comprobando qué elementos funcionan depurando cada una de las partes que generasen conflicto. En la siguiente versión se solucionarán los problemas anteriores y surgirán nuevos y así hasta que el modelo quede perfecto, en las últimas iteraciones sobre todo ya centradas en motivos de estética.

En un principio la primera versión del proyecto fue una aplicación java sin ningún tipo de servidor ni nada por el estilo, se llamaba a cTAKES por la línea de comandos y a partir de ahí se obtenían los datos que se consideraban necesarios. En la etapa de análisis previa a la

segunda versión el hecho de encontrar finalmente una base para un server REST de Apache cTAKES hizo que se trasladase el proyecto de lenguaje hacia Python y posteriormente una vez diseñado y codificado en Python todo el proyecto y todos los resultados obteniéndolos por consola se consideró centrar las siguientes versiones en hacer del proyecto una app web para que fuese accesible desde el navegador de internet y con vistas a que pueda ser utilizada por más gente sin necesidad de instalaciones de ningún tipo. Finalmente, las últimas versiones simplemente corrigen y mejoran la apariencia de la aplicación web.

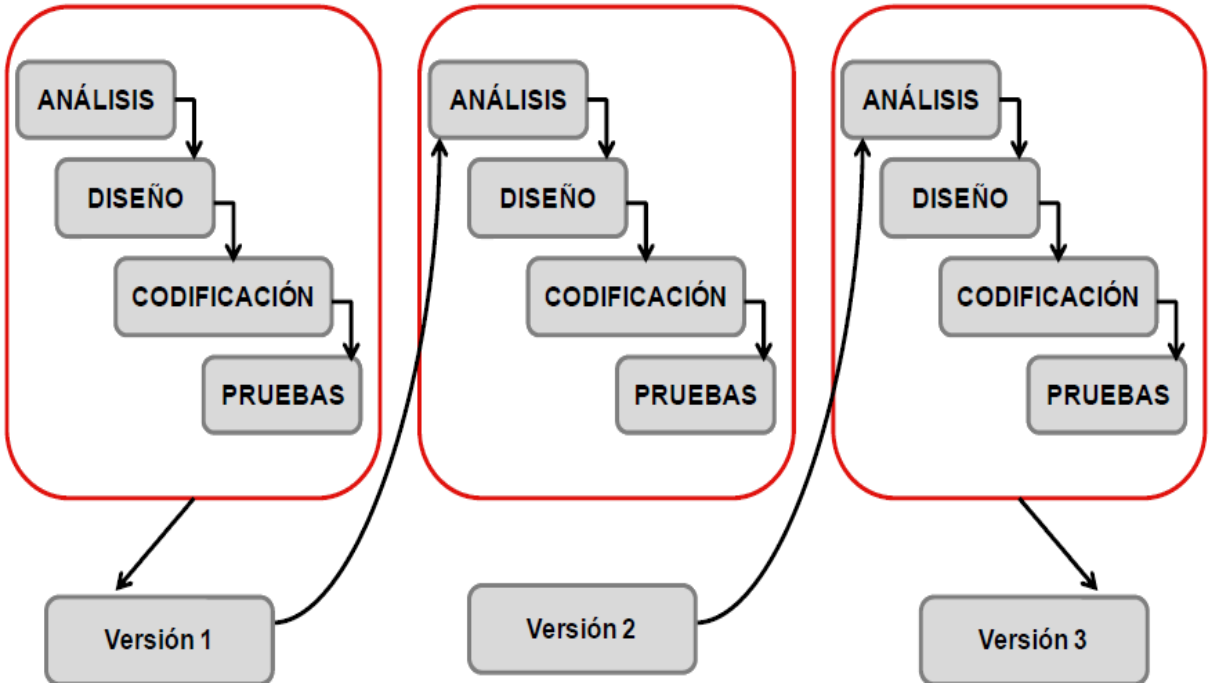


Ilustración 4.Diagrama de metodología iterativa incremental. Fuente:
https://www.emaze.com/@AZRQTRWI/Desarrollo-iterativo-e-incremntal

4.4 PLANIFICACIÓN Y ESTIMACIÓN ECONÓMICA

En este apartado se presenta la planificación que se ha seguido para llevar a cabo el proyecto, así como con todos los recursos utilizados.

4.4.1 CRONOGRAMA

A continuación, se presenta el cronograma que describe qué actividades han sido llevadas a cabo en este proyecto especificando en qué mes se ha llevado a cabo cada una de ellas.

Marzo	Abril	Mayo	Junio	Julio
Estudio de mercado	Concreción de objetivos	Estudio de Python	Desarrollo Web con Django	Redacción de la memoria
Estudio acerca de Apache cTAKES	Comienzo de la aplicación en Java EE	Desarrollo del Servidor de cTAKES	Redacción de la memoria	Detalles finales del sistema y de presentación
			Desarrollo en Python	

Tabla 1. Cronograma del proyecto

4.4.2 RECURSOS HUMANOS

Este proyecto está llevado a cabo en su totalidad por una única persona que se ha encargado de la puesta a punto y desarrollo desde su inicio del sistema. A continuación, se puede ver una estimación de las horas empleadas para cada tarea.

Actividad	Horas empleadas
Estudio de mercado	10
Diseño del sistema	30
Búsqueda de requisitos	5
Estudio de Python y Django	10
Estudio de Apache cTAKES	15
Desarrollo de servidor de Apache cTAKES	20
Estudio de UMLS	3
Desarrollo web	15
Versiones desechadas	20

Tabla 2. Tareas realizadas y horas empleadas

Total: 118 horas sin tener en cuenta la realización de este documento.

4.4.3 COSTES

Los costes del sistema han sido nulos ya que todo lo que se emplea es software libre y código abierto. Solamente se contaría con el coste humano, que contando 12 euros/hora como un sueldo medio de programador junior, el resultado sería de sería de 118*12euros/hora por lo tanto, 1416 euros de coste aproximado.

Capítulo 5. SISTEMA/ESTUDIO DESARROLLADO

En este capítulo se describen con detalle las particularidades del proyecto realizado. Para ello se comienza con un análisis del sistema en el que se incluyen las arquitecturas software y hardware empleadas. Después se describe el diseño del sistema, de las distintas plataformas web y finalmente como se ha llevado a cabo la implantación.

El proyecto se divide en dos tipos de tareas bien diferenciadas en un principio de las que surgen divisiones también. Estos dos tipos de tarea son: las tareas de investigación y las tareas de desarrollo.

Las tareas de investigación o estudio se centraron en la herramienta Apache cTAKES antes de poder desarrollar era importante conocer la herramienta y estudiar las posibilidades que ofrece por ello en este apartado se hablará de los componentes que forman parte de ella, el procedimiento de instalación, la utilidad que tiene y para qué se va a usar exactamente en este proyecto.

Las tareas de desarrollo y diseño del sistema están divididas según cada uno de los módulos que componen el sistema. El servidor de cTAKES en donde se explican los arreglos que fueron necesarios para que funcionase correctamente, el servidor de UMLS en donde se recaban las distintas llamadas que ofrece para obtener las respuestas que se buscan y la APP Web con su diseño dividido en *front-end y back-end*. Se comenta sobre cada una de las partes el código utilizado, se ilustra con imágenes la apariencia de la web y se documenta también con ellas las distintas funcionalidades que ofrece.

5.1 APACHE CTAKES: COMPONENTES, INSTALACIÓN Y UTILIDAD. ANÁLISIS DE LA HERRAMIENTA

Primeramente, el proyecto se centró en el análisis de la herramienta Apache cTAKES y las posibilidades que podía ofrecer a la hora de extraer información de historias clínicas, el proyecto, como se comentó con anterioridad se centra en esta herramienta ya que es open source, un software libre para su uso y configuración por parte de cualquiera.

Apache CTAKES está basado en la tecnología Apache UIMA que se describió en el apartado dedicado a las mismas y cuenta con una estructura pipeline en la que cada componente recibe la respuesta del anterior incorporando nuevas anotaciones y así consecutivamente hasta obtener un resultado.

Está formado por 6 componentes.

"Sentence boundary detector": Se encarga de saber cuál es el principio y el final de una oración, ya sea por los puntos, las exclamaciones o las interrogaciones.

"Tokenizer": Lo conforman dos componentes, el primero de ellos separa los componentes de la oración que estén separados por signos de puntuación o por espacios. El segundo identifica si los tokens consecutivos están relacionados por el contexto, ya sea un título personal, una fecha o un numeral.

"Normalizer": Es una capa de personalización del componente SPECIALIST Lexical Tools, representa cada palabra del texto entrante con un número por sus propiedades léxicas. Esto permite identificar como distintas, las palabras que son mencionadas varias veces en el

mismo texto pero que no significan lo mismo. Tanto la forma normalizada como la no normalizada son usadas por el diccionario que se describirá después.

"Part of Speech Tagger": Etiquetador que utiliza los módulos de OpenNLP para esta cuestión. Modelos supervisados y entrenados para leer información clínica.

"Named Entity Recognition": Mapea los conceptos encontrados con un concepto de su terminología en un diccionario. Se utiliza para ello un subset de UMLS que incluye los conceptos de SNOMED CT y RxNORM. Cada término en el diccionario pertenece a un tipo semántico, enfermedades, síntomas, procedimientos, anatomía y medicación. En este componente también es donde se anotan las negaciones, indicando si existe cerca de la palabra algo que indique que está negada. Se devuelve el texto correspondiente al término, el código del diccionario y si está negado o no.

A continuación, se explicará brevemente el proceso de instalación de CTAKES y sus modos de uso por defecto:

Es preciso contar con una versión de Java 1.8 o superior este también es uno de los motivos a favor de la herramienta ya que los programas creados en Java pueden ejecutarse correctamente y sin adaptaciones en cualquier sistema operativo y ese es uno de sus principales valores como lenguaje.

Se debe descargar la carpeta de CTAKES de la web oficial y descomprimirla, tras ello se debe de asignar una variable de entorno CTAKES_Home (Home a partir de ahora) a la carpeta ya que es utilizada por los procesos internos del programa. Posteriormente se descarga la carpeta "Resources" (recursos) de la web y se procede a copiarla en la carpeta Home sustituyendo todo lo que nos diga.

CTAKES nos da la posibilidad de trabajar con diccionarios propios en HSQLDB, simplemente hay que apuntar en el archivo "properties" del CTAKES a esta base de datos aunque en este caso no lo utilizamos, es una posibilidad que no se descarta en un futuro.

En el caso de no usar un diccionario propio es necesario solicitar un acceso al diccionario UMLS, registrándose en la web en apenas 24 horas se obtiene el acceso al mismo. Una vez hecho y para usar dos funcionalidades por defecto de CTAKES se debe de entrar en los archivos runcTAKES***.bat en la carpeta bin y añadirlas en la siguiente línea.

```
java                    -DcTAKES.umlsuser=<YOUR_UMLS_ID_HERE>                    -
DcTAKES.umlspw=<YOUR_UMLS_PASSSWORD_HERE>
```

Uno de estos servicios por defecto es el CAS Visual Debugger que sirve principalmente para comprobar si la herramienta está funcionando correctamente, esta interfaz gráfica se lanza con este comando:

```
bin\runcTAKESCVD.bat                                              desc\cTAKES-clinical-
pipeline\desc\analysis_engine\AggregatePlaintextFastUMLSProcessor.xml
```

Aparecerá un gran cuadro de texto donde se puede escribir la información clínica que se desee y una barra de menú donde se deberá seleccionar la opción "run" y a continuación la opción que diga "run" y a continuación "el nombre pipeline que se haya elegido", en el caso descrito el AggregatePlaintextFastUMLSProcessor.xml, existen tanto esta versión como el AggregatePlaintextUMLSProcessor.xml la única diferencia entre ellos es que en el primero se exportan menos anotaciones sobre el texto, las cuales incluyen todas a las que le damos utilidad por lo que no hace falta ser tan exhaustivo como para utilizar el que no es *"fast"*.
El programa devolverá la lista de las anotaciones que se han hecho sobre el texto introducido por pantalla. Como se comentó con anterioridad la utilidad que tiene es simplemente ver si todo funciona con corrección.

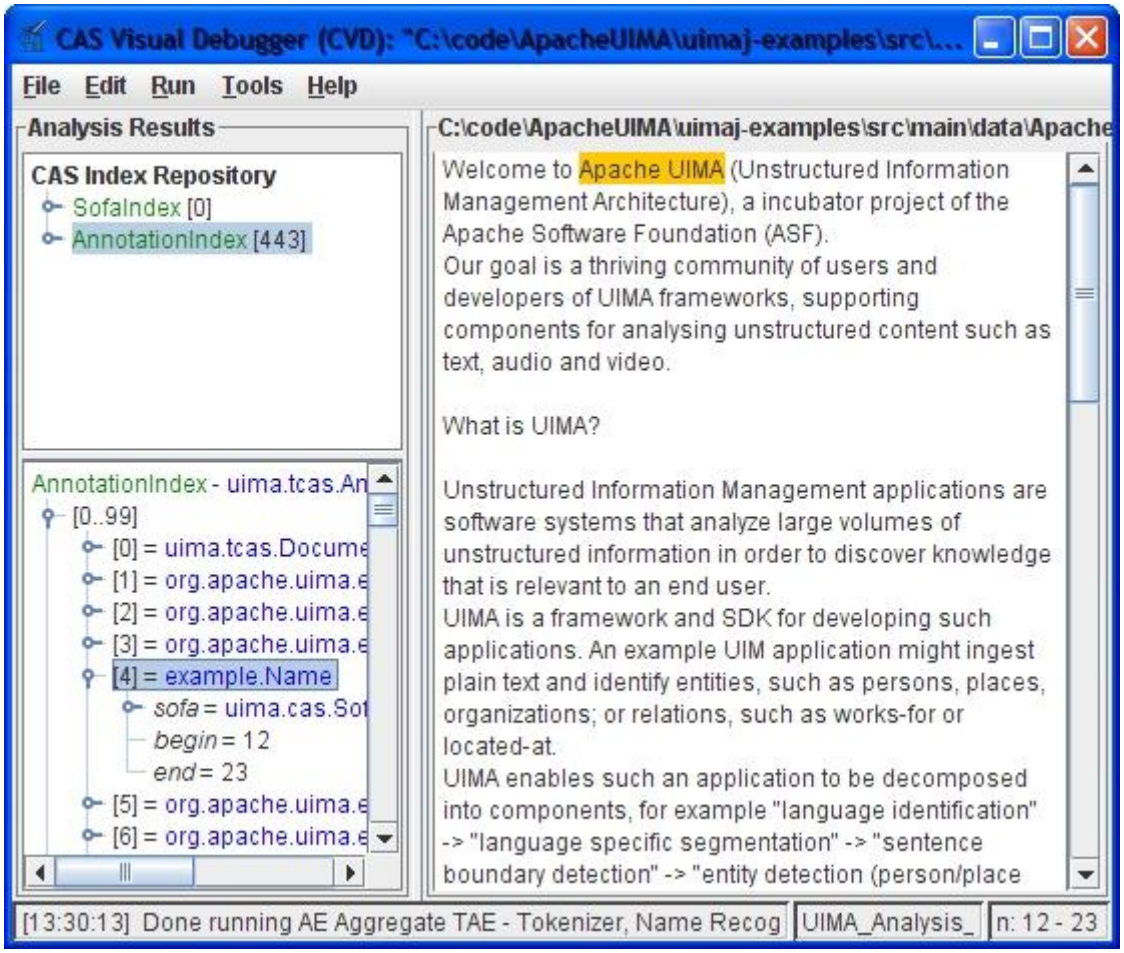


Ilustración 5.Vista del CVD. Fuente: http://cTAKES.apache.org/examples.html

El segundo servicio que ofrece es el Collection Processing Engine que también es una interfaz gráfica y que sirve para procesar múltiples documentos de texto a la vez. Se abre con este comando:

```
bin\runcTAKESCPE.bat
```

En la interfaz gráfica que aparece se indica el directorio donde se encuentran los archivos de texto que se quieren procesar y la carpeta de destino donde quieres que se guarde el resultado, que es un xml, uno por cada archivo de entrada, con todas las anotaciones sobre ese texto.

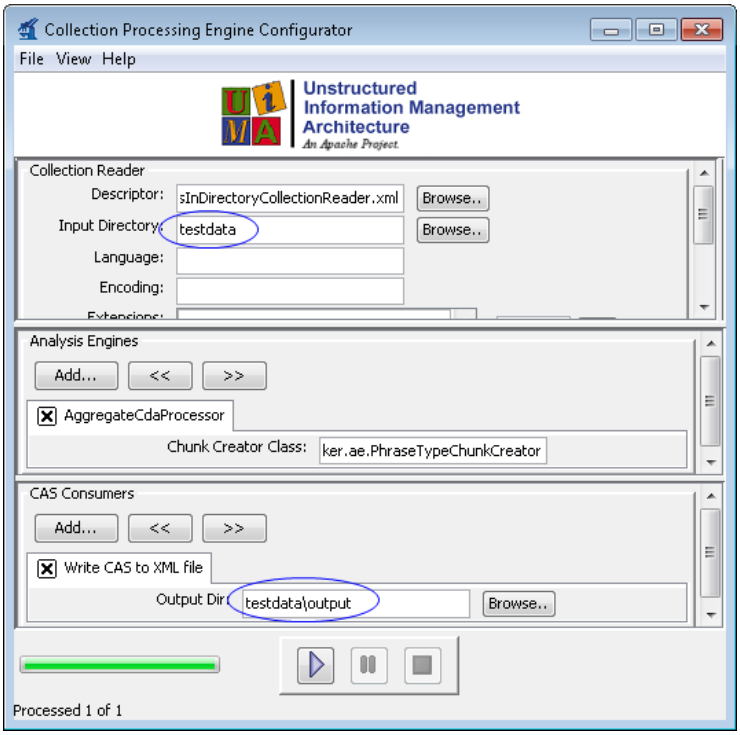


Ilustración 6. Vista del CPE.

Fuente:https://cwiki.apache.org/confluence/display/CTAKES/cTAKES+4.0+User+Install+Guide

La interfaz gráfica CPE permite leer todos los archivos de texto dentro de una carpeta y analizarlos todos a la vez, a pesar de no sacársele partido en el sistema que se relatará posteriormente es una de las grandes funcionalidades de CTAKES y que por supuesto se plantea sacarle un gran provecho en el futuro ya que para un entorno en el que se procesa grandes cantidades de información permitir procesar de una manera rápida gran cantidad de historias clínicas y analizarlas es algo a tener muy en cuenta, sobre todo en entornos Big Data.

Además de las maneras gráficas relatadas existe la posibilidad de utilizar CTAKES mediante línea de comandos, esto se puede hacer mediante un comando y varias variables adjuntas.

```
bin/runClinicalPipeline -i CarpetadeEntrada --xmiOut CarpetaDeSalida --user
umlsUsuario --pass umlsContraseña
```

Indicando en el comando el directorio de entrada y salida de archivos como se relataba en la funcionalidad anterior y el usuario y contraseña de UMLS se obtienen resultados como con la interfaz gráfica pero simplemente ejecutando una línea.

Tras analizar estas formas de utilizar cTAKES y explorando las posibilidades de código abierto que había en internet se toma la decisión de intentar crear un servidor de cTAKES que reciba las historias clínicas vía POST y utilizarlo dentro de una web de creación propia que haga peticiones tanto a cTAKES como a la API de UMLS y obtenga resultados conjuntándolas. Existe esta dirección: http://54.68.117.30:8080/index.jsp
Que es lo más similar al sistema que se desea desarrollar que existe utilizando Apache cTAKES, se elige el formato de la respuesta e incluye uno muy llamativo ya que representa de manera muy gráfica las anotaciones sobre el texto. Como podemos ver en la imagen [11] en la página siguiente.

Permite dar a cTAKES una respuesta muy atractiva a la vista y en la que se entienden perfectamente las anotaciones llevadas a cabo, en el caso del sistema realizado lo más importante para conseguir el objetivo era la relación entre los términos clínicos identificados dentro del texto y el CUI que le corresponde por lo que para ello obtener la respuesta en XML en la que ambos atributos forman parte de la anotación de UMLS hace más sencillo guardarlos en una variable para después operar con ellos, como sucede a la hora de después llamar a la API de UMLS para obtener toda la información acerca del concepto médico, el principal objetivo del proyecto.

```
SENTENCE:  Mr. Smith is  a  72 yo male with   COPD  , worsening gradually over the past year despite compliant
           NNP  NNP  VBZ DT    JJ  NN   IN     NN        VBG       RB      IN  DT   JJ   NN    IN       JJ
                                              |======|  |=======|           |================|         |=======|
                                             Disorder    Event                 Timex                 Event
                                             C0024117
TLINKS:    over the past year CONTAINS worsening , over the past year CONTAINS compliant

SENTENCE:   use  of XYZ meds, nebulizers and rescue inhalers.
            NN   IN NN  NNS      NNS     CC    NN     NNS
           |===|             |========|           |======|
           Event               Event               Event

SENTENCE:     PFT   ?s  (attached)
              NN     NN     JJ
           |=======|
           Procedure
           C0024119

SENTENCE:  demonstrate the decline in
               VBP     DT    NN    IN

SENTENCE:    lung   function over the last 12 months.
             NN       NN     IN  DT   JJ      NNS
           |======| |======| |====================|
           Anatomy   Event           Timex
           C0024109
TLINKS:    over the last 12 months CONTAINS function

SENTENCE:   Now  with the constant  use  of 2-3L NC O2 at home for the
            RB    IN  DT     JJ     NN   IN   NN NN NN IN  NN  IN  CD
           |===|                   |===|
           Timex                   Event
TLINKS:    Now CONTAINS use

SENTENCE:  last month, he  still can no longer walk  to the bathroom, about 30   feet   from his  bed without
            JJ   NN    PRP  RB   MD  RB   RB    VB   IN DT     NN      IN       NNS     IN  PRP$ NN    IN
           |========|                          |===|                          |======|
             Timex                             Event                        Anatomy
                                                                            C0016504
TLINKS:    last month CONTAINS walk

SENTENCE:  significant   SOB    and overall discomfort.
               JJ         NN    CC    JJ        NN
                        |======|            |========|
                       Finding              Finding
                       C0013404             C2364135

SENTENCE:  The kitchen is  further from his  bed.
           DT    NN    VBZ   RB     IN  PRP$ NN
               |=====|
                Event

SENTENCE:  He  says his  bed/bath
           PRP VBZ  PRP$ NN   NN

SENTENCE:  doorways and  halls   are wide enough for a  scooter that will bring him to his   toilet  , sink and
             NNS    CC    NNS    VBP  JJ    JJ   IN  DT   NN    WDT   MD   VB   PRP IN PRP$    NN       JJ  CC
           |======|     |======|         |====|                                            |=======|
            Event         Drug           Event                                           Procedure
                        C2356088                                                         C0184958
```

Ilustración 7. Resultado de la búsqueda de términos de manera gráfica

5.2 ANÁLISIS DEL SISTEMA

En este apartado se realiza un análisis de la plataforma desarrollada, citando sus componentes y la situación de unos con respecto a otros. En los apartados posteriores se encontrará como se ha llevado a cabo cada parte relatada y como funciona cada uno de los elementos.

El sistema está dividido en tres partes en cuanto a arquitectura, forman parte de él dos servidores, uno de Apache cTAKES y el de UMLS (externo) y por otro lado está también alojada la web que envía peticiones a ambos servidores para cumplir con su cometido.

La web en la que se basa el resultado del sistema se ha desarrollado íntegramente en Python ya que es un lenguaje que ofrece hacer las llamadas a servidores de manera muy sencilla y fue programada haciendo uso del IDE PyCharm y del framework Django para la creación de webs con Python, el servidor web corre sobre un servidor Apache Tomcat. El servidor de Apache cTAKES está desarrollado en Scala y se ejecuta de manera totalmente independiente al servidor web ocupando otro de los puertos de "localhost". Por otro lado el servidor de UMLS permite las peticiones que están reflejadas en su API y de la que se hacen uso de un número específico de ellas que son de utilidad para la web desarrollada.

La interfaz de la web es responsiva para permitir su uso desde cualquier dispositivo tanto smartphones como PC's.

5.2.1 ARQUITECTURA DE SOFTWARE

El funcionamiento interno de la web está basado en una clase controladora la cual cuenta con todas las funciones utilizadas en la web e interactúan con sus vistas correspondientes para así comunicar y hacer fluir la información entre el front-end y el back-end todo ello ha sido gracias al uso del Framework Django para el desarrollo de aplicaciones web con Python.

En este caso se ha basado la arquitectura en el uso de *web services* de tipo "Restful", uno de creación propia para poder hacer peticiones a cTAKES y obtener los resultados de la herramienta y otro proporcionado por la web de UMLS que permite hacer las peticiones propias de ese diccionario.

La web CTAKESOnline cuenta con un modelo cliente servidor como el ilustrado:

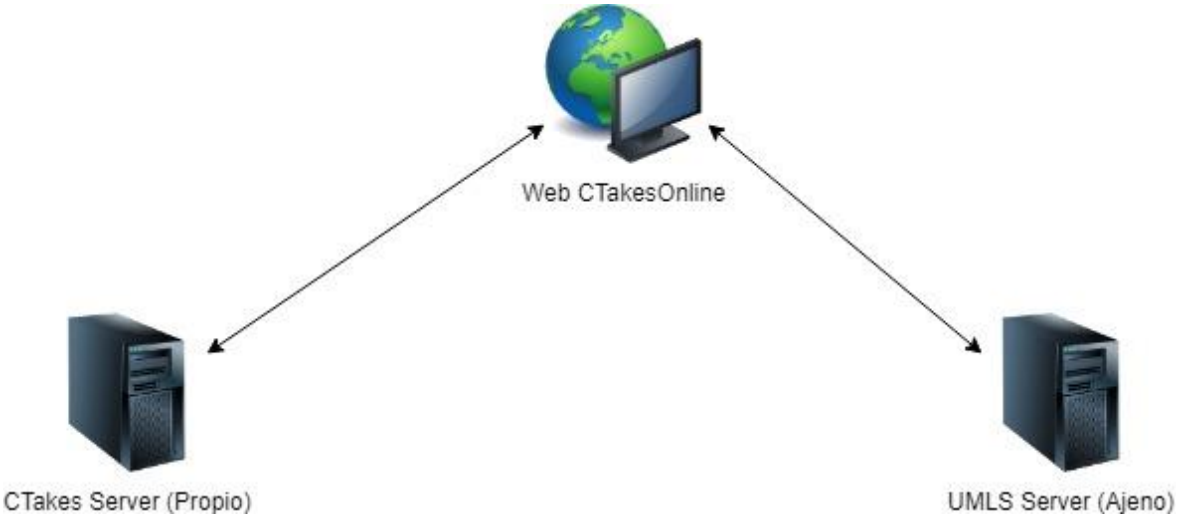


Ilustración 8. Arquitectura del sistema

La aplicación web lanza una petición al servidor de cTAKES cada vez que se rellena y envía el formulario de extracción de conceptos médicos en una historia clínica, éste recibe la historia clínica al completo y la procesa y responde a la web con un JSON en el que están incluidas todas las anotaciones que ha hecho sobre él.

Una vez recibida la información la web opera con ella hasta quedarse solamente con los conceptos médicos y sus CUIs correspondientes. En este caso si el usuario de la web desea obtener más información acerca de un CUI solamente tiene que pulsarlo y esa información se envía al server de UMLS, se ejecutan con él como atributo 3 llamadas al server para obtener 3 tipos de información distinta acerca del concepto médico, esta respuesta también de tipo JSON se envía a la aplicación web y ésta opera con los datos hasta mostrar en la web la información más reseñable obtenida acerca del concepto médico.

5.3 DISEÑO DEL SISTEMA

5.3.1 CREACIÓN DE UN SERVER PARA USAR APACHE CTAKES

Para el desarrollo de esta parte se utiliza como base código abierto encontrado en GitHUB para la creación de un *Rest-Server* de cTAKES simple hecho en SCALA cuyos requerimientos son tener cTAKES instalado en el ordenador y Maven para la obtención de los paquetes y librerías que utiliza y compilarlo. La implementación estaba pensada para Linux y fue necesario apuntar, crear un vínculo, a dos carpetas de CTAKES desde el directorio del server, la carpeta "desc" y la carpeta "resources" ya que son utilizadas por el servidor para poner en funcionamiento cTAKES y posteriormente modificar el server ya que resultaba siempre en "timeouts", por lo que hubo que modificar varias condiciones. [12]

```
val desc = if(args.length > 2) args(2)
      else "desc/cTAKES-clinical-
pipeline/desc/analysis_engine/AggregatePlaintextFastUMLSProcessor.xml"
```

La primera, que se utilizase el Pipeline "Fast", el cuál pues como dice el nombre es más rápido con lo cual, a pesar de perder un poco de precisión, se obtenía una rápida respuesta, muy importante en un server, cuyo código se encuentra sobre estas líneas.

Y también cambios en los criterios de creación del servidor maximizando todas las esperas para evitar los "timeouts" como se puede ver en el código bajo estas líneas.

```
val conf = ConfigFactory
      .parseString("""
    spray.can.client.request-timeout = infinie
    spray.can.client.response-timeout = infinie
    spray.can.host-connector.idle-timeout = infinite
    spray.can.host-connector.client.request-timeout = infinite
    spray.can.server.registration-timeout=infinite
```

```
    spray.can.client.connection-timeout= infinite
    """)
```

Para lanzar el servidor basta con un comando similar al que se utilizaba con CTAKES:

```
java -DcTAKES.umlsuser=<YOUR_UMLS_ID_HERE> -DcTAKES.umlspw=<YOUR_UMLS_PASSSWORD_HERE> -Xmx5g -cp target/cTAKES-server-0.1.jar:resources/ de.dfki.lt.cTAKES.Server host(e.g. localhost) port(e.g. 9999) desc/path/to/desc.xml
```

En él tras añadir las credenciales de UMLS apuntamos a la carpeta "resources" de CTAKES a la cual se le creó un vínculo en la principal del server y lo mismo para la carpeta "desc" posteriormente. Se elige el host y el puerto y el servidor ya queda lanzado. Por ejemplo, en localhost 8080 obtendríamos una respuesta al entrar en nuestro navegador en:

```
http://localhost:8080/cTAKES?text=Knee Pain.
```

Al hacer POST del link encima de estas líneas nos devuelve una respuesta JSON el cual es un archivo en formato .xml de las anotaciones sobre el texto enviado como parámetro "text" a esa dirección, se puede enviar vía código como atributo o simplemente usar la *url* como en el código superior y añadir el texto tras "text=". Esta respuesta la recuperaremos mediante una web que será el cliente que use el acceso a este servidor.

En la imagen se muestra un pequeño trozo de cómo es esta respuesta obtenida por parte del servidor de cTAKES creado.


```
[{"typ":"org.apache.ctakes.typesystem.type.syntax.ConllDependencyNode","annotation":
{"begin":0,"cpostag":null,"pdeprel":null,"lemma":null,"head":null,"postag":null,"id":0,"sofa":null,"end":20,"deprel":null,"feats
":null,"form":null,"phead":null}},{"typ":"org.apache.ctakes.typesystem.type.syntax.NP","annotation":
{"sofa":null,"begin":0,"chunkType":"NP","end":20}},{"typ":"uima.tcas.DocumentAnnotation","annotation":
{"sofa":null,"begin":0,"end":20,"language":"x-unspecified"}},
{"typ":"org.apache.ctakes.typesystem.type.syntax.TreebankNode","annotation":{"parent":null,"begin":0,"children":
[{"typ":"org.apache.ctakes.typesystem.type.syntax.TreebankNode","annotation":{"parent":null,"begin":0,"children":
[{"typ":"org.apache.ctakes.typesystem.type.syntax.TerminalTreebankNode","annotation":
{"parent":null,"begin":0,"children":null,"nodeValue":"Pain","nodeTags":null,"tokenIndex":0,"headIndex":0,"sofa":null,"nodeType":
"NN","leaf":"true","end":4,"index":0}}],"nodeValue":null,"nodeTags":
[],"headIndex":0,"sofa":null,"nodeType":"NP","leaf":"false","end":4}},
{"typ":"org.apache.ctakes.typesystem.type.syntax.TreebankNode","annotation":{"parent":null,"begin":5,"children":
```

5.3.2 UTILIZACIÓN DE LA API REST DE UMLS

Ya que se va a utilizar un modelo cliente servidor para el uso de CTAKES, para complementar la información que se obtiene mediante el uso del servidor anteriormente creado y lanzado. La API Rest de UMLS permite obtener prácticamente la misma información sobre un término que buscándolo en su propia web, pero con un problema, las llamadas son limitadas por lo que si el sistema creado se escalase en gran medida podría haber ciertas dificultades causadas por ello y probablemente sería necesario ampliar la licencia de uso de UMLS o instalar UMLS en el PC desde donde se fuese a lanzar la aplicación, su gestor y las bases de datos. Esta API Rest se utilizará para obtener la información sobre los términos clínicos que desee el usuario y así complementar el uso de CTAKES.

La API Rest de UMLS utiliza el sistema de protección Kerberos que mencionamos en las tecnologías utilizadas, para usar esta API se necesita primero tener unas credenciales UMLS y usar las mismas o una clave API que se tiene asociada a la cuenta para poder utilizarla.

El primer paso es la obtención de un TGT, este *Ticket-Granting Ticket* será válido para las siguientes 8 horas, para obtenerlo simplemente sirve con hacer un POST a la dirección https://utslogin.nlm.nih.gov/cas/v1/api-key en el caso de usar una API-KEY esto nos devolverá la dirección a la que debemos hacer los request posteriormente para la obtención de un *Service Ticket*, que expira con un uso o tras 5 minutos desde su activación, que es https://utslogin.nlm.nih.gov/cas/v1/tickets/{TGT}.

Una vez obtenido el *Service Ticket*, ST, ya podemos hacer llamadas a la API de UMLS para obtener la información que deseamos.

```
https://uts-ws.nlm.nih.gov/rest/content/current/CUI/C0018787?ticket=ST-134-
HUbXGfI765aSj0UqtdvU-cas
```

Este es un ejemplo de uso para la obtención de información acerca de un CUI lanzando esta dirección, con el código y el service ticket mediante un GET.
Se utilizarán las llamadas de esta API que utilizan los CUIs como término de búsqueda:

GET	/content/{version}/CUI/{CUI}	Retrieves information about a known CUI
GET	/content/{version}/CUI/{CUI}/atoms	Retrieves atoms and information about atoms for a known CUI
GET	/content/{version}/CUI/{CUI}/definitions	Retrieves definitions for a known CUI
GET	/content/{version}/CUI/{CUI}/relations	Retrieves NLM-asserted relationships for a known CUI

Tabla 3. Métodos de la API de UMLS utilizados. Fuente: https://documentation.uts.nlm.nih.gov/rest/home.html

Las urls para utilizar GET que aparecen en la tabla superior devuelven respuestas que son de gran utilidad para obtener toda la información acerca de los CUIs.

5.3.3 DESARROLLO CONJUNTO DE CTAKES JUNTO CON LA API REST DE UMLS

CTAKES se utiliza para analizar los textos clínicos, gracias a ello a partir de un texto escrito en lenguaje natural podemos obtener qué términos clínicos se mencionan en el mismo e incluso los etiqueta con su CUI correspondiente dentro del diccionario UMLS, eso nos permite tener ya una clave sobre la que buscar información en las bases de datos de UMLS mediante la API Rest anteriormente mencionada. El principal problema en este caso es, como anteriormente se mencionó, la limitación de estas llamadas a la API de UMLS, este problema tiene un agravante ya que dado que no está limitado en ningún momento la cantidad de caracteres del texto que se analizarán con cTAKES tampoco está limitada la cantidad de términos clínicos y sus correspondientes CUIs que se pueden obtener como resultado de la extracción por lo que las llamadas a la API de UMLS si se hiciese directamente sobre todos los CUIs superaría con creces los límites. Para solucionar esta

cuestión se ha permitido solamente buscar de cada vez información solamente acerca de un CUI. [4]

Al procesar el texto mediante el server de cTAKES, el usuario obtendrá como respuesta los términos clínicos que aparecen en él y su CUI correspondiente. Este CUI será un hipervínculo, que al ser pulsado redireccionará a una página en la que se explica con mayor profundidad el término, tipo semántico, descripción y se muestran sus relaciones.

Un ejemplo de respuesta de la llamada a la API Rest de UMLS cuando se busca el CUI: C0155502 sería:


```
{
    "pageSize": 25,
    "pageNumber": 1,
    "pageCount": 1,
    "result": [
        {
            "classType": "Definition",
            "sourceOriginated": true,
            "rootSource": "MSH",
            "value": "Idiopathic recurrent VERTIGO associated with POSITIONAL
NYSTAGMUS. It is associated with a vestibular loss without other neurological or
auditory signs. Unlike in LABYRINTHITIS and VESTIBULAR NEURONITIS inflammation in
the ear is not observed."
        }
    ]
}
```


Son respuestas en formato JSON que posteriormente se explicará (en el apartado de implementación) cómo se ha diseñado para que se obtenga lo que de verdad se necesita de ellas.

5.3.4 AGREGADO DE RESULTADOS

Se permitirá también crear un archivo en .txt de los resultados obtenidos tras el procesado de la historia clínica en la que solamente aparezcan los términos clínicos que aparecen en

ellas y los CUIs. Mediante un botón en la página de resultado se podrá crear un archivo .txt con los resultados obtenidos y el archivo tendrá el nombre deseado por el usuario. Se proporciona esta funcionalidad para permitir al usuario almacenar los resultados por si en un futuro desea trabajar con ellos o sobre todo si la historia clínica es muy larga desea tener más a mano los términos clínicos que se encuentran en la historia. Así como si se utiliza un método de búsqueda en el que se lea el contenido de los documentos para buscar un término en concreto será mucho más sencillo implementarlo sobre los agregados de resultados de las historias. El archivo de texto quedaría como en la imagen inferior. Los conceptos en parejas con sus códigos de identificación única.

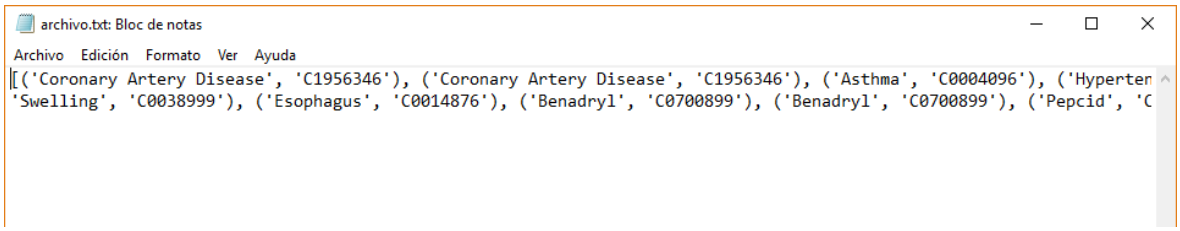


Ilustración 9. Agregado de resultados en .txt

5.3.5 DISEÑO DE LA INTERFAZ

La interfaz web de este sistema está pensada para ser utilizada tanto en smartphones como en todo tipo de tablets, ordenadores o en cualquier sistema que permita conexión a internet por lo tanto se puede decir que debe de estar adaptada para dar la mejor de las experiencias a los usuarios en esta gran cantidad de dispositivos.

En vez de crear distintas versiones de esta web se ha decidido optar por un diseño que sea responsivo y que permita interactuar de la manera adecuada con todas las plataformas. La web adapta así los contenidos y su disposición para permitir interactuar al usuario independientemente del tamaño del dispositivo o de la resolución que tenga su pantalla.

Para el diseño del front-end se ha hecho uso de las librerías de CSS de Bootstrap, las cuales ponen a disposición del usuario clases ya creadas y modificadas y que mejora en gran medida la apariencia que ofrecen las clases por defecto de CSS y facilitan su uso.

Se ha decidido un diseño sobrio con la web completamente blanca para dotar de seriedad a la web y legibilidad.

A continuación, se muestran diversas capturas de la web para demostrar que es responsiva.

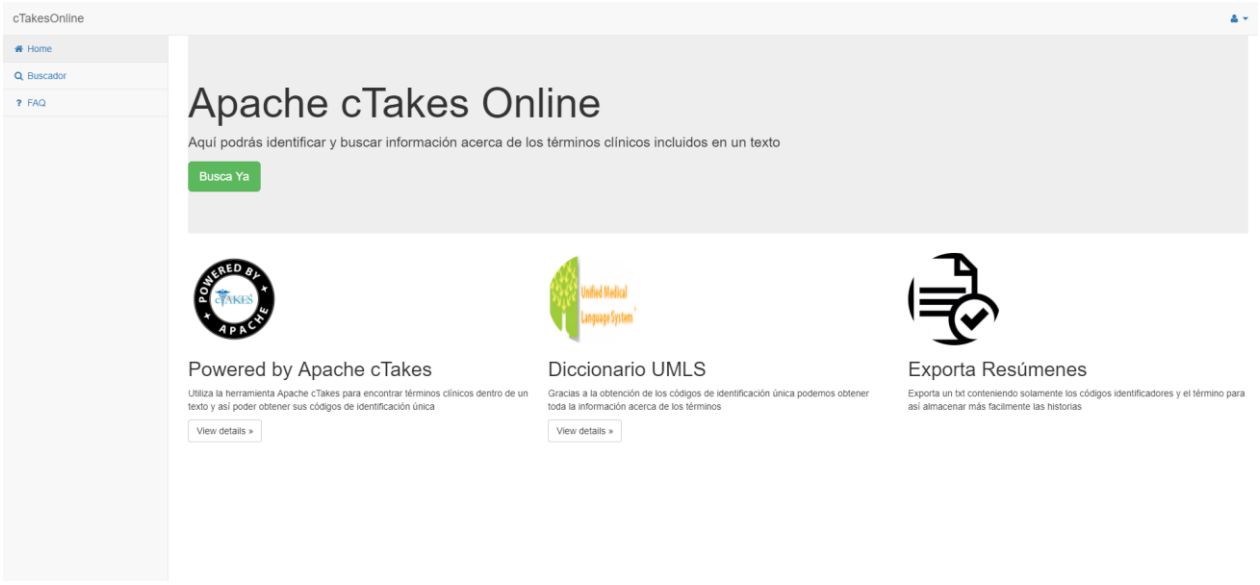


Ilustración 10. Página principal de la web

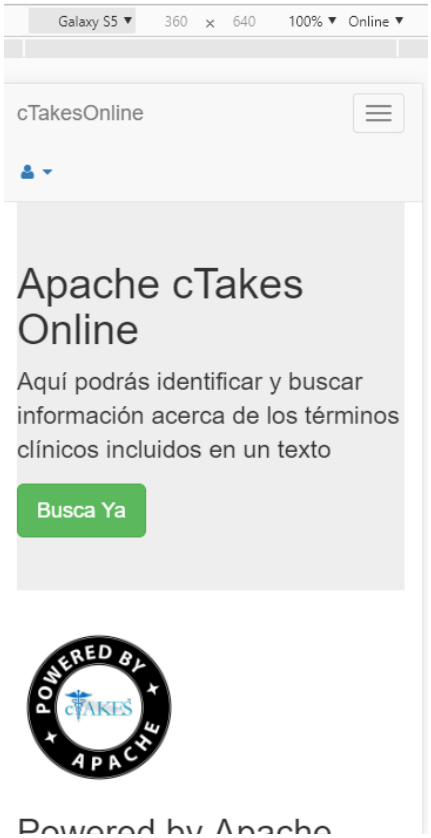


Ilustración 12. Vista de la página principa

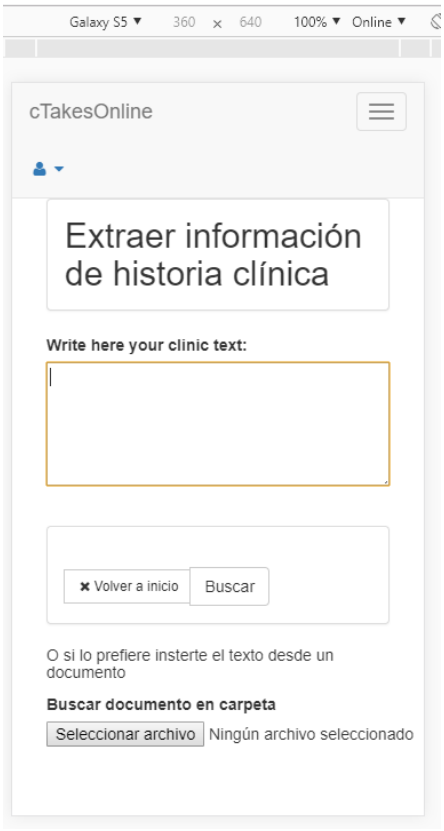


Ilustración 11. Vista página de búsquedas en Smartphone

5.3.6 DISEÑO DEL SERVIDOR WEB

El servidor del web ha sido creado totalmente en Python gracias al Framework Django el cual permite crear una web soporte principal y diversas aplicaciones contenidas en ella, en este caso solamente una.

En este modelo de trabajo se divide todo en componentes, existe un archivo en el que están las "urls" de la web a la cual a cada una se asigna un nombre, luego el archivo "views" desde el cual se captan las "requests" por parte de la web y que suele redirigir a las funciones hechas en la clase controladora y posteriormente las "templates", los archivos HTML que son el *front-end* de la aplicación.

5.4 IMPLEMENTACIÓN

En este apartado se detalla cómo, partiendo del diseño y las arquitecturas anteriormente descritas se ha implementado el sistema y cómo funciona.

5.4.1 PÁGINA PRINCIPAL

A modo de presentación existe una página principal en la que se describen las funcionalidades de la web para informar al usuario de estas, así como los links a las páginas web tanto de Apache cTAKES, como de UMLS por si se necesita información complementaria.

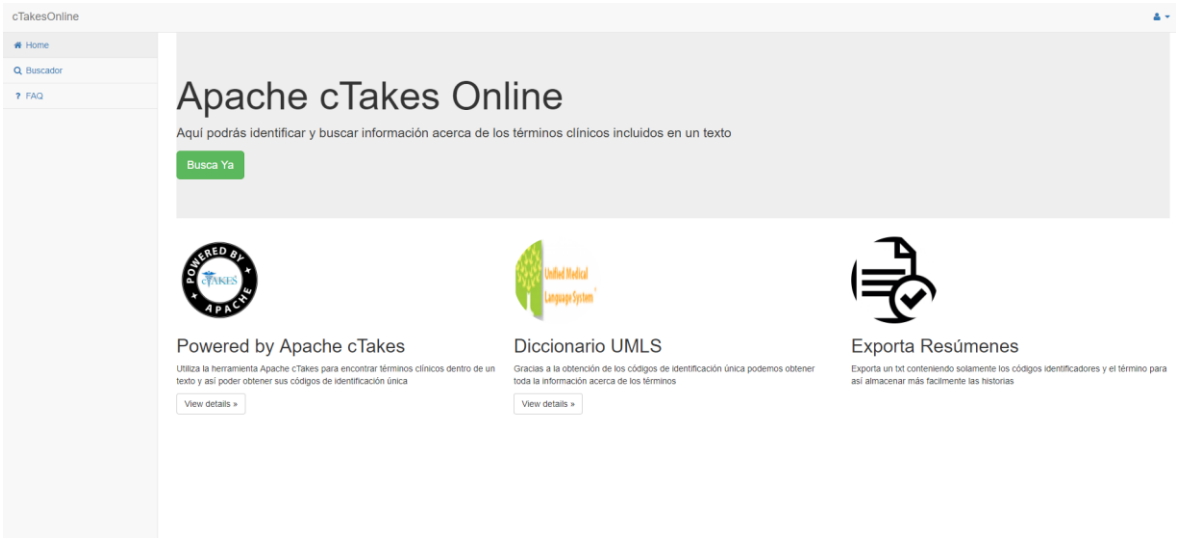


Ilustración 13. Página principal de la web

5.4.2 PÁGINA DE EXTRACCIÓN DE CONCEPTOS

A continuación, se mostrará la página encargada de la extracción de los conceptos médicos de las historias clínicas, cuenta con dos formas de uso:

- La primera mediante su escritura en la propia página en el cuadro de texto indicado para escribir también permite pegar el texto que se desee en el interior del mismo cuadro de texto.
- La segunda forma, pulsando el botón seleccionar archivo se abrirá el explorador de archivos y se podrá elegir un archivo .txt que contenga el texto que se desea analizar. Al seleccionarlo el texto escrito en el fichero con formato .txt aparecerá en ese mismo cuadro de texto, este se vuelca al cuadro de texto dando así la posibilidad al usuario de llevar a cabo alguna modificación de última hora sobre el mismo.

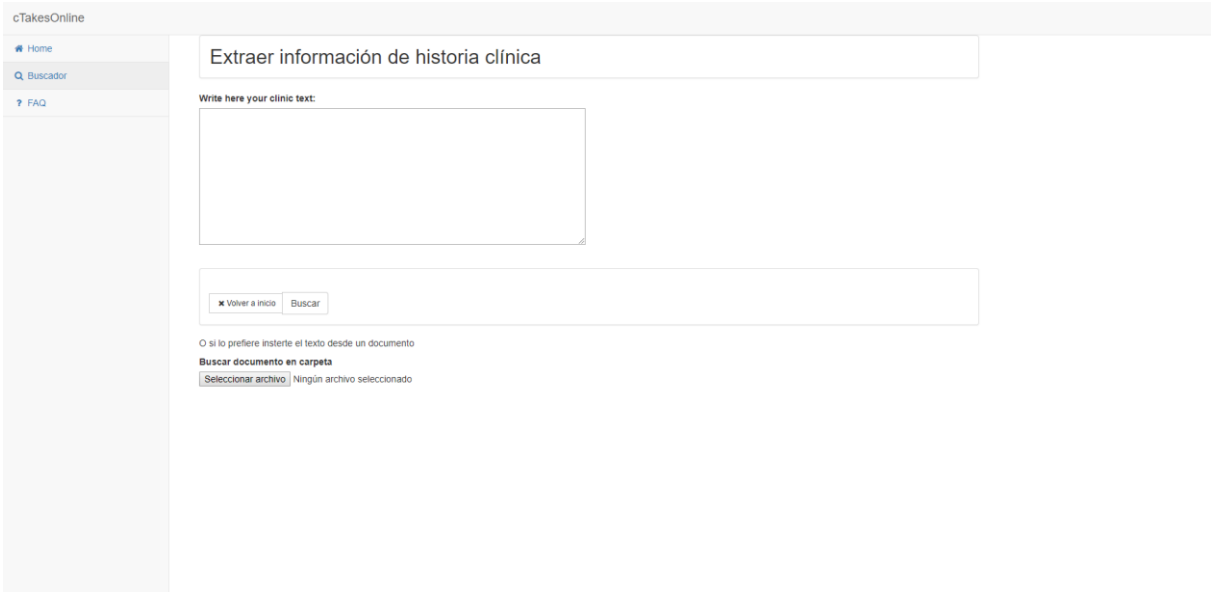


Ilustración 14. Página de extracción de la web

La página consiste en un formulario que nos permitirá enviar y procesar la información escrita en ese cuadro de texto y en cuanto a la selección de archivos se ha creado una clase en Javascript para leerlos y volcar su contenido al cuadro de texto con el siguiente código. La función FileReader es la que nos permite leerlo y después simplemente cambiamos el valor del "input" del texto por el contenido del archivo .txt que se ha leído.

```
document.getElementById("openFile").addEventListener('change',function(){
    var fr = new FileReader();
    fr.onload=function(){
    document.getElementById("btn-inputParam1").value = this.result;
```

```
    }
    fr.readAsText(this.files[0]);
})
```

5.4.3 PÁGINA DE RESULTADOS

La página con los resultados de una extracción sobre una historia clínica de ejemplo será la siguiente:

Ilustración 15. Página de resultados de la web

Como se puede ver la página está compuesta en el lado izquierdo por las parejas del nombre del término y su CUI asociado y en el lado derecho se encuentra la posibilidad de exportar los mismos a un archivo de texto con el nombre que desee el usuario si desea almacenarlos.

Se ha puesto el CUI como un botón porque al pulsarlos se redirige a la página que muestra la información sobre los mismos, se ha hecho de esta manera ya que los límites de la API Rest de UMLS son bastante estrechos y no permiten llevar a cabo muchas llamadas a la vez y ya que se desconoce el número de CUIs que tendrá la historia clínica es mejor que el usuario decida sobre cuales quiere saber más información.

A continuación, se mostrarán las partes del código que hacen esto posible:

```
def resultadohistorias(request):
                try:


            param1 = request.POST.get("btn-inputParam1")

            a = ExtraccionClinica.Extraccion(param1)
            return render(request, 'HistoriasClinicas/resultadohistorias.html',
a)



        except Exception as e:
            print("eeee")
            error = "Hubo un error Views"
            context = {'error': e}
            return render(request, 'HistoriasClinicas/error.html', context)
```

En el archivo "views.py" se recibe el *request* por parte del formulario, con el parámetro recibido gracias al atributo *name* del *input* en el *html* lanzamos la función Extracción de la controladora:

```
def Extraccion(param1):
    try:


        base_url_cTAKES = "http://localhost:9000/cTAKES?text="

                text = str(param1)
        r = requests.get(base_url_cTAKES + text)

        list_resultados = re.compile("preferredText\"[\:\"]*([A-
z\s]*)[\"\,]*score[\"\:0-9\.\,A-z\-\_\;]*\"cui\":\"(C[0-
9]*)\"}}").findall(r.text)

       context= {'resultados': list_resultados}
```

```
        return context

    except:
       error = "Hubo un error"
       context = {'error': error}
       return context
```

Contamos con la url del servidor de cTAKES que se relató su funcionamiento con anterioridad, le enviamos el texto leído y recibimos la respuesta JSON. Para quedarnos solamente con los CUIs y los nombres de los términos clínicos que es realmente lo que nos interesa, se utilizan expresiones regulares, mediante ella podemos filtrar el texto y quedarnos con los dos "grupos" que vamos a utilizar. Se utiliza la expresión regular: ("preferredText\"[\:\"]*([A-z\s]*)[\"\,]*score[\"\:0-9\.\,A-z\-_\;]*\"cui\":\"(C[0-9]*)\"}}") en ella nos quedamos con el contenido que se encuentra entre paréntesis que en este caso es el preferredText que equivale al nombre del término encontrado y el CUI. Estos se guardan en un diccionario, variable "context" ya que es la única manera de transferirlos de vuelta al frontend.

En la página de resultados se procesan de la siguiente forma:

```
{% for text, cui in resultados %}
                    </br>
                            <h4>{{text}}</h4>

                            <input name="cui" type="submit" value="{{cui}}">
                </br>
                        {% endfor %}
```

De esta manera obtenemos los dos campos que se querían y podemos poner el CUI como próximo elemento con el que interactuar, Django, metiéndolo ya como valor dentro de un input para utilizarlo en el siguiente formulario. Se parece bastante en estos casos a los JSP de Java pero se pueden utilizar muchas menos funciones dentro de los html.

En la otra columna se ofrece la posibilidad al usuario de exportar los resultados de la extracción de la historia clínica creando un archivo conteniendo los términos y los CUIs asociados a los mismos con el nombre que él desee, para ello se procede de la siguiente manera:

```
def Export(param1,param2):
    with open(str(param1)+".txt", 'w') as tgt_file:
        tgt_file.write(str(param2))
```

Siendo param1 el nombre que el usuario le da al archivo y param2 los resultados que se desean escribir en el mismo obteniendo un resumen de la historia clínica como este:

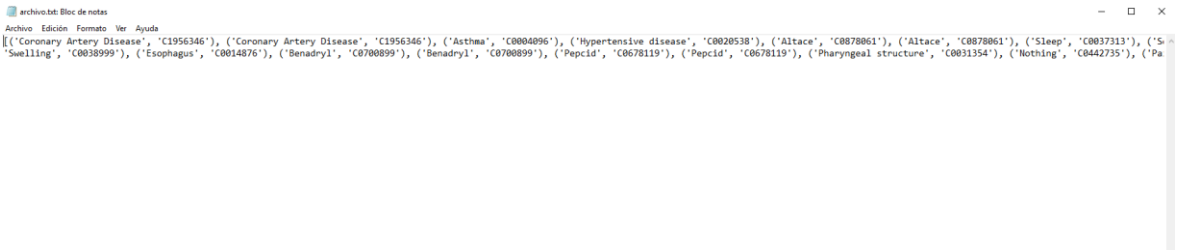


Ilustración 16. Captura de lo exportado en un archivo .txt

5.5 PÁGINA DE INFORMACIÓN EXTRA SOBRE LOS TÉRMINOS CLÍNICOS. API UMLS

Primeramente antes de poder buscar información en la API de UMLS es necesario obtener un TGT y un ST, para ello se envía el api-key del usuario de UMLS como request y se obtiene así el TGT, este es válido por 8 horas y tienen un límite de peticiones por lo que se procede a guardarlo en un archivo .txt auxiliar junto con la fecha en milisegundos, cada vez que haya una llamada a la API de UMLS comprobaremos si la antigüedad del mismo es menor o mayor a 8 horas, en el caso primero se utilizará el TGT guardado en el archivo .txt auxiliar llamado aquí “tgt.txt”, si fuese el caso opuesto, se pedirá un nuevo TGT.

```
def gettgt(api_key):
    with open("tgt.txt", 'w') as tgt_file:
                r = requests.post("https://utslogin.nlm.nih.gov/cas/v1/api-key",
data={'apikey': api_key})
        print(r)
        print(r.text)

        tgt = re.compile("<form\saction=\"([a-z\/0-9.:\"=\s\-]*\/TGT\-
.*)?\"\smethod=").findall(r.text)
        if tgt:

            # ESCRITURA (if >8 horas)
            tgt_file.write(str(tgt) + "^" + str(int(round(time.time() * 1000))))
            # print(tgt[0])
            return tgt[0]

        else:
            print("Error al recuperar TGT")



def getst(tgt):
    params = {'service': "http://umlsks.nlm.nih.gov"}
    # h = {"Content-type": "application/x-www-form-urlencoded", "Accept":
"text/plain", "User-Agent": "python"}
    r = requests.post(tgt, data=params)
    st = r.text
```

En el caso de los ST no existe este problema, son de un solo uso y se deben pedir con cada llamada, también tienen un límite de llamadas bastante escaso por lo que se debe intentar también sacar el mayor partido posible.

Con ello, se procede a obtener los datos que se precisan y el procedimiento será similar al utilizado a la hora de obtener información del servidor de cTAKES.

```
def umls(cui):
    api_key = "b0f4e7b5-0edf-4ef2-a7c4-0340288817e8"
    base_url_cui_description = "https://uts-ws.nlm.nih.gov/rest"
```

```
    with
open("C://Users/Usuario/PycharmProjects/ExtraccionClinica/ExtraccionClinicaWeb/Hi
storiasClinicas/tgt.txt",
              'r') as tgt_file:
        tgt, timestamp = str(tgt_file.readline()).split("^")
        # print(str(timestamp))
        # print(str(tgt))
        if int(round(time.time() * 1000)) - int(timestamp) > 28800000:  # han
pasado mas de 8 horas
            tgt = gettgt(api_key)
        else:
            print("no han pasado 8 horas => se va a utilizar el TGT guardado")

   # print("\nTGT:" + str(tgt) + "\n")


    st = getst(tgt)
    # st = "ST-3022924-dg9BfNaTwb9OTNfgFZR4-cas"
    # print(st)
    endpoint = "/content/current/CUI/" + cui + "/definitions"
    r = requests.get(base_url_cui_description + endpoint, params={'ticket': st})
    definition = re.compile("value\"[:\s]*\"([A-z\s\.,\-0-
9():\[\]]*)\"").findall(r.text)

    st = getst(tgt)

    endpoint = "/content/current/CUI/" + cui
    r = requests.get(base_url_cui_description + endpoint, params={'ticket': st})
    info = re.compile("semanticTypes[\":\s\[\{]*name[\"\:\s]*([A-z\s]*)[0-9A-
z\:\/\-\.\s\'\"\,\}]*name[\"\:\s]*([[0-9A-z\:\.\/\-\.\s\']*)").findall(r.text)

    st = getst(tgt)

    endpoint = "/content/current/CUI/" + cui + "/relations"
    r = requests.get(base_url_cui_description + endpoint, params={'ticket': st})
    relations = re.compile("relatedId\":[\s\"]*([0-9A-z\:\/\-
\.\s]*)[\"\,\n\s]*relatedIdName\":[\s\"]*([0-9A-z\:\.\/\-
\.\s\']*)").findall(r.text)

    context = {'definition': definition, 'info':info, 'relations': relations}
```

```
    return context
```

La función umls pide primeramente el TGT, si hiciese falta ya que lleva a cabo la comprobación de las 8 horas, posteriormente se pide un ST por cada llamada que hace a la API de UMLS, estas llamadas son: para obtener la definición del término clínico y para pedir el tipo semántico y las relaciones de este.

Como en el apartado anterior se obtiene de la respuesta la información que se desea mediante expresiones regulares con las cuales se filtra el texto que se desea, en la primera llamada se extrae del texto solamente lo incluido en el término definition, en la segunda llamada se extrae el texto incluido en la variable "semanticTypes" para obtener el tipo semántico del término y además el nombre exacto del término en el diccionario extrayendo lo que se encuentra en la variable "name", en el último caso se obtiene todo lo escrito en el interior de las variables "relatedID" y "relatedName" para obtener las relaciones, finalmente se proceden a guardar los resultados en un diccionario y así permitir enviarlo al front-end.

Capítulo 6. ANÁLISIS DE RESULTADOS

En este capítulo se hará un acercamiento a los resultados obtenidos durante el desarrollo del sistema sobre todo los erróneos y cómo se pudieron subsanar. Se analizarán los resultados de todas las fases del desarrollo desde el comienzo del proyecto y sobre todo durante el aprendizaje del uso de la herramienta Apache cTAKES para estudiar qué posibilidades ofrece. Se tendrán en cuenta resultados a la hora de intentar lanzar el servidor de cTAKES, ya que fue necesario investigar acerca de casos similares en otros servers para poder contrastar el error y darle una solución. Se estudian también, los resultados obtenidos por el otro servidor que forma parte de la web, el servidor de UMLS, sobre el cuál se testea su capacidad para soportar un alto número de peticiones en poco tiempo de manera insatisfactoria.

6.1 RESULTADOS OBTENIDOS

Los resultados obtenidos durante los primeros usos de la herramienta Apache cTAKES fueron bastante frustrantes. En el comienzo, durante las primeras pruebas realizadas sobre los casos de uso estándar de la herramienta, los resultados obtenidos eran simplemente de pruebas con el propósito de comprobar si funcionaban los componentes de la aplicación a expensas de la conexión con el diccionario UMLS. Los CUIs que se obtenían no eran correctos, fue comprobado cuando al buscar el CUI que resultaba, no coincidía con ninguno dentro del diccionario de UMLS.

En ese momento quedó claro qué "pipeline" no era el adecuado, los pipelines que no incluyen la palabra UMLS en su nombre, se utilizan para pruebas de los componentes de cTAKES sin conexión con el diccionario, para hacer el testeo más sencillo y más rápido. Para utilizar un "pipeline" o motor de análisis que conduzca a resultados consecuentes es necesario utilizar las credenciales de UMLS, escribirlas en los archivos de configuración del programa y llamar a los pipelines que incluyen en su nombre la palabra UMLS. Una vez usado el

Clinical Pipeline adecuado, ya sea el "Fast" o la versión con las anotaciones completas, los resultados eran los esperados como por ejemplo el código de identificación única, el cual era ya coincidente con la información recogida en los diccionarios de la UMLS. Además en los resultados se muestran todas las anotaciones sobre el texto que han llevado a la herramienta a identificar ese término como un término clínico.

Tras ello y tras una gran cantidad de pruebas a partir de código abierto encontrado por internet para la creación de un server con cTAKES para poder hacerle peticiones post desde la web creada, los errores producidos eran sobre todo de "timeout" y también se producían errores al utilizar pipelines distintos al más común, finalmente cambiando los criterios del servidor hacia espera infinita y utilizando el "pipeline" "fast" consiguieron subsanar esos problemas.

Probando el sistema de seguridad de la API de UMLS se decidió probar si existía alguna limitación en su uso ya que en la web no lo dice en ningún sitio. El día de la prueba fue necesario hacer cerca de 10 peticiones seguidas para que mandase un mensaje de error, el cuál indicaba que se había superado el número de peticiones que se permiten. A posteriori, ya no fueron tantas las peticiones que se debían hacer para recibir el mensaje de error, por lo que probablemente el usuario iniciado esté siendo reconocido por la web como peligroso por automatizar peticiones, identificado por causa de la gran cantidad de peticiones en poco tiempo.

Por otro lado, excepto no conseguir hacer funcionar el módulo encargado de las negaciones, el resto de los anotadores de cTAKES que fueron testeados y comprobados funcionaron con normalidad y no enviaron en ningún caso alguna respuesta errónea o nula.

Se han obtenido finalmente los resultados deseados por parte de cTAKES y por parte de la API de UMLS mediante el código implementado por la web y ambos devuelven la respuesta esperada y necesaria para cumplir con los objetivos marcados que debía cumplir el sistema.

6.2 ANÁLISIS CRÍTICO

Podría existir la posibilidad de obtener resultados más completos aún, mostrando absolutamente todos los datos que guarda UMLS acerca de un término, pero tanto la limitación en la API Rest de UMLS como la poca información acerca de la herramienta o de sistemas asociados que hay en internet han hecho que las complicaciones que podrían surgir a la hora de resolver errores fueran mayores y más dificultosos.

Finalmente se han conseguido tres objetivos tangibles, el uso de cTAKES en un server, el crear agregados de información de las historias clínicas y obtener la definición, tipo semántico y relaciones de un término clínico.

Probablemente algunas de estas funcionalidades se podrían implementar de manera modular directamente sobre cTAKES.

Capítulo 7. CONCLUSIONES Y TRABAJOS FUTUROS

En este último capítulo se desarrollarán las conclusiones acerca del trabajo realizado, los resultados obtenidos y las posibilidades que existen en el futuro de llevar a cabo trabajos de índole similar o usar este trabajo como catapulta hacia algo más grande.

El proyecto ha logrado un fin, el cuál es importante y propone una base sobre la que trabajar con la herramienta Apache cTAKES, también cuenta como resultado todo lo aprendido acerca de las plataformas utilizadas, lenguajes de programación utilizados y acerca de la extracción de información de historias clínicas electrónicas todo ello es un gran bagaje a la hora de tener un mayor conocimiento para llevar a cabo proyectos relacionados con la misma temática en el futuro donde desde un primer momento ya se contará con un *background* acerca del campo.

Es un campo en el que todavía queda mucho por trabajar y mucho que descubrir ya que actualmente con las soluciones Big Data en auge por los grandes volúmenes de datos a los que se enfrenta la sociedad. Por todo ello, la necesidad en este campo es brutal ya que pronto será absolutamente toda la información médica en formato electrónico.

7.1 CONCLUSIONES

Gracias al desarrollo de este proyecto se ha conseguido una manera más sencilla de utilizar la herramienta de extracción de información de historias clínicas Apache cTAKES, una manera gratuita para llevar a cabo esta extracción y siendo más aún al alcance de todo el mundo siendo ello vía web y de manera totalmente gratuita al utilizar todo software libre y código abierto.

El sistema permite extraer información de historias clínicas obteniendo todos los términos clínicos que aparecen en ellas y asociados a ellas sus códigos de identificación única en el diccionario de términos médicos, se permite guardar al usuario esta información en un

archivo de texto para su posterior uso y para un ahorro en la cantidad de información que almacena. También se permite al usuario buscar toda la información acerca de ese término clínico con solamente un clic, aunque esta función esté todavía en pruebas y sea defectuosa en algunos casos por causa de los límites que tiene UMLS en su API, probablemente pidiendo que incorporen el usuario utilizado a una "lista blanca" sería suficiente para resolver esa cuestión.

Por lo demás la herramienta cumple con las expectativas y sobre todo el uso dado en este sistema es un primer acercamiento hacia poder crear una web mucho más potente usando todo el valor potencial de cTAKES en cuanto a procesamiento de textos simultáneos, pero se deberá de tener en cuenta el cuello de botella que está provocando actualmente UMLS en cuanto a que permita hacer más llamadas a su API.

La aplicación cuenta con una interfaz amable y sencilla para el usuario a la par de intuitiva y cumple con las necesidades.

También ha sido de gran ayuda aprender el idioma Python, el cual hizo las llamadas Rest mucho más sencillas y también haber aprendido a desarrollar web en Python ha sido toda una experiencia.

En general los resultados a pesar de los innumerables cambios de planes han sido relativamente positivos y dan soluciones a cuestiones que no se encuentran ahora mismo en internet.

7.2 TRABAJOS FUTUROS

Esta web es simplemente un ejemplo a pequeña escala de lo que se puede llegar a hacer con el acceso a un servidor de cTAKES y en conjunto con el servidor de UMLS. Cómo ya se ha comentado anteriormente si se pudiese dar rienda suelta a la API de UMLS se podrían conseguir muchos más resultados porque permitiría saber absolutamente todos los datos relativos al término encontrado que existen en UMLS. Por otro lado, la consecución del

funcionamiento de la identificación de negaciones por parte de la herramienta Apache cTAKES se marca también como un objetivo para tener en cuenta en el futuro.

Dada la complejidad que se ha mencionado con anterioridad en este documento del diccionario UMLS, con la gran cantidad de términos y relaciones con las que cuenta se muestra todavía complicado para que por parte de UMLS se lleve a cabo una traducción completa al castellano y que fuese posible su uso por parte de la herramienta cTAKES. Por ahora el idioma inglés es suficiente para el desarrollo que se ha hecho, pero probablemente a la hora de ser usada por profesionales de la medicina de este país sería primordial contar con el diccionario traducido.

Por último, cabe mencionar las posibilidades en cuanto a conseguir resultados personalizados, estudiando qué términos se repiten más dentro de una historia clínica y cuáles son de mayor importancia para obtener un resumen de ésta. Esta tarea necesita una lógica que la haga funcionar y ello se plantea como un objetivo futuro.

Capítulo 8. BIBLIOGRAFÍA

[1] Redacción. "La historia clínica en España: 35,7 millones digitales; 11 millones físicas". Redacción Médica. 2017 https://www.redaccionmedica.com/secciones/parlamentarios/la-historia-clinica-en-espana-35-7-millones-digitales-11-millones-fisicas-7937

[2] Wikipedia. "Django (framework) https://es.wikipedia.org/wiki/Django_(framework)

[3] Wikipedia. "Unified Medical Language System" https://en.wikipedia.org/wiki/Unified_Medical_Language_System

[4] API REST UMLS https://documentation.uts.nlm.nih.gov/rest/home.html

[5] G. Savova, J. Masanz, P. Ogren, J. Zheng, S. Sohn, K. Kipper-Schuler, y C. Chute. Mayo clinical Text Analysis and Knowledge Extraction System (cTAKES): architecture, component evaluation and applications https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2995668/

[6] Wikipedia. "General Architecture for Text Engineering". https://es.wikipedia.org/wiki/General_Architecture_for_Text_Engineering

[7] Carol Friedman, Lyudmila Shagina, Socrates A. Socratous, y Xiao Zeng. "A WEB-Based Version of MedLEE: A Medical Language Extraction and Encoding System"

[8] MedKAT web. "The MedKAT Pipeline" http://ohnlp.sourceforge.net/MedKATp/

[9] HITEx Manual v2.0. "HITEx Manual v2.0" https://www.i2b2.org/software/projects/hitex/hitex_manual.html

[10] Mplus web. "Mplus General Description" https://www.statmodel.com/features.shtml

[11] Apache cTAKES User Install Guide. https://cwiki.apache.org/confluence/display/CTAKES/cTAKES+4.0+User+Install+Guide

[12] Weissenborn D. Apache cTAKES server. GIT. https://github.com/dirkweissenborn/cTAKES-server

ANEXO I

1. MANUAL DE INSTALACIÓN PARA APACHE CTAKES EN WINDOWS

1. Comprobar si se tiene Java 1.8 o superior instalado con el comando *java -version*
2. En la página de Downloads de cTAKES; http://ctakes.apache.org/downloads.html y en User Installation hacer click en Windows y se procederá a descargar cTAKES.
3. Descomprimir la carpeta en el directorio deseado.
4. Una vez descomprimida nombrar una variable de entorno que se llame <cTAKES_HOME> que tenga como dirección el directorio principal de Apache cTAKES.
5. Ir a esta web: https://sourceforge.net/projects/ctakesresources/files/ y descargar la última versión de los recursos para cTAKES.
6. Descomprimir la carpeta y copiarla entera en <cTAKES_HOME> aceptando que sobreescriba todos los archivos que sean necesarios hasta que se complete el copiado.
7. Solicitar en la web de UMLS: https://uts.nlm.nih.gov/license.html una licencia con usuario y contraseña hacer uso del diccionario. (Suele tardar 12 horas en verificarse)
8. Una vez registrado en UMLS se deben de buscar los siguientes archivos:

   ```
   <cTAKES_HOME>\bin\runctakesCVD.bat
   <cTAKES_HOME>\bin\runctakesCPE.bat
   ```
   Y en ellos sustituir en la siguiente línea las credenciales UMLS con las que te hayas registrado entrecomilladas “User” “Password”.

   ```
   java -Dctakes.umlsuser=<YOUR_UMLS_ID_HERE> -
   Dctakes.umlspw=<YOUR_UMLS_PASSSWORD_HERE> -cp ...
   ```

9. Una vez hecho todo esto ya está configurado y se pueden lanzar los distintos comandos de cTAKES para su uso:

- bin\runctakesCVD.bat desc\ctakes-clinical-pipeline\desc\analysis_engine\AggregatePlaintextFastUMLSProcessor.xml
  -Para ejecutar el CVD
- bin\runctakesCPE.bat
  -Para ejecutar el CPE
- bin/**runClinicalPipeline** -i *inputDirectory* --xmiOut*outputDirectory* --user *umlsUsername* --pass *umlsPassword*
  -Para ejecutar cTAKES desde consola