GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Sistema de traducción de lenguaje natural a lenguaje de programación mediante técnicas de NLP

Autor: Sergio Félix Giménez Suárez

Director: David Contreras Bárcena

Madrid

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

Aplicación de traducción de lenguaje natural a código

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

curso académico 2021/22 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.: Sergio Félix Giménez Suárez Fecha: 23/ 08/2022

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: David Contreras Bárcena Fecha: 24/08/2022

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Aplicación de traducción de lenguaje natural a código

Autor: Sergio Félix Giménez Suárez

Director: David Contreras Bárcena

Madrid

Agradecimientos

En un primer momento me gustaría agradecer a todos los profesores que he tenido a lo largo de la carrera, pues es gracias a ellos y al conocimiento que me han impartido que se ha podido realizar este trabajo, en especial a David como profesor de programación orientada a objetos pues es el tema principal sobre el que gira el trabajo.

Continuando con Juan Antonio, quien ha sido un gran compañero durante toda la carrera y quien siempre ha ayudado de manera desinteresada, mostrando que gran persona és.

Y por último a todos los compañeros que he tenido durante la carrera pues de una manera u otra, todos han influido en mí de alguna manera.

APLICACIÓN DE TRADUCCIÓN DE LENGUAJE NATURAL A CÓDIGO

Autor: Giménez Suárez, Sergio Félix
Director: Contreras Bárcena, David
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

El proyecto tiene como base la creación de un programa que, aplicando las distintas técnicas existentes de análisis del lenguaje natural, sea capaz en última instancia de generar el código referente al texto en dos lenguajes de programación diferentes como lo son Java y Python. Sirviendo este código generado como base para personas que busquen aprender a programar sin tener ningún tipo de conocimiento previo con respecto al tema.

Palabras clave: Java, Python, Procesamiento del lenguaje natural.

1. Introducción

Hoy en día encontramos diferentes dispositivos electrónicos a cada lado que miramos y la verdad es que estos están completamente integrados en nuestro día a día hasta llegar al punto de que muchas personas serían incapaces de vivir sin su ayuda.

Este estado actual de la tecnología hace necesaria la formación de manera constante de nuevos programadores, haciendo imprescindible la creación de nuevas herramientas o procesos que agilicen y simplifiquen la formación para poder satisfacer la alta demanda que existe en el sector. El objetivo del proyecto es el desarrollo de una herramienta capaz de generar el código referente a un programa, a partir de la definición realizada del mismo en lenguaje natural. El código resultante a dicho proceso se podrá generar en dos lenguajes de programación diferentes que poseen varias diferencias.

Actualmente existen diversas herramientas de manera pública que permiten y facilitan el acceso al procesamiento natural del lenguaje, mientras que en la generación de código dichas herramientas se encuentran en desarrollo.

2. Definición del proyecto

Este proyecto presenta diferentes dificultades. La primera que encontramos se trata de la capacidad de comprender qué es lo que quiere transmitir el usuario pues, aunque se use el mismo idioma cada persona utiliza el lenguaje de manera diferente, usando diferentes palabras y descripciones para un mismo problema. En este aspecto la dificultad es parecida al que se enfrentan hoy en día las diferentes aplicaciones de traducción de idiomas.

El segundo problema viene dado por los lenguajes de programación en los que se escribirá el código. Existen lenguajes tipados que se basan en prevenir que surjan errores durante la ejecución del código asignando de manera inequívoca un tipo de dato a cada variable y prohibiendo su uso como otro tipo sin antes hacer una correcta transformación. Por otro lado, los lenguajes no tipados, permiten tratar todo tipo de variables independientemente del tipo que tenga definido y para lo que esté definido.

En este caso los códigos generados por el programa como resultado se encuentran programados en Python y Java que se tratan de los lenguajes de programación más utilizados.

Los códigos generados han de ser ejecutables, pero no óptimos para cada tarea en específico pues la resolución optima de un problema implica una completa compresión del problema y de sus dificultades, lo que requeriría una descripción y comprensión del lenguaje perfectos.

3. Descripción de la herramienta

El programa, como se ha mencionado antes tiene dos secciones claramente diferenciables. La primera de todas se trata del procesamiento del lenguaje natural, en este caso se ha utilizado dos sistemas para su análisis. “CoreNLP”[10] y “SparkNLP”[11]. Se tratan de dos sistemas diferentes que permiten un tratamiento intenso del lenguaje. Ambos se basan en el entrenamiento de un modelo para identificar los diferentes elementos gramaticales de las oraciones. La segunda hace referencia a la parte de creación del código, en este caso el lenguaje final de salida será Java o Python.

Ambas partes requieren de un módulo que los conecte. Para esto una vez realizado el procesamiento del lenguaje, se crea un vector con las palabras clave y a partir de estas palabras clave se genera el código.

Así mismo el programa ha sido diseñado para poder intercambiar sendos módulos de manera simple siendo finalmente el vector de palabras clave intermedio que une ambos sistemas el único limitante.

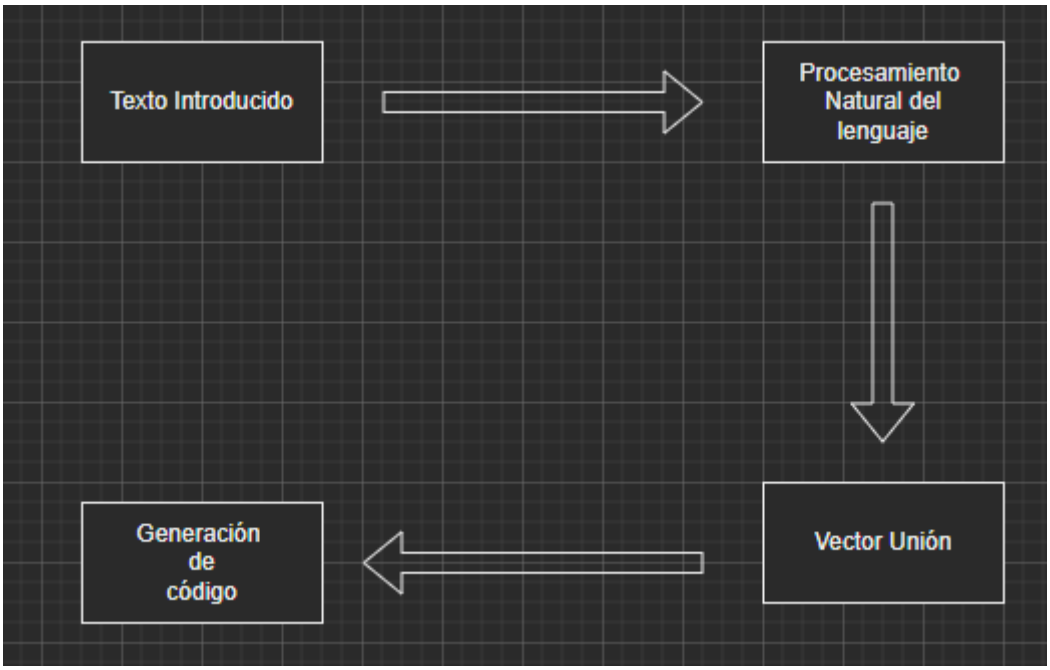

Ilustración 1 - Funcionamiento del programa

4. Resultados

El proyecto se ha dividido en varios apartados y la profundidad alcanzada en cada uno de ellos limita por sí solo el proyecto. Por ejemplo, a la hora de realizar el procesamiento del lenguaje natural se limitan en función de cómo se comprende el texto las funcionalidades finales del código. Por otra parte, aún si se tuviera una compresión perfecta del lenguaje y se supiese con claridad lo que se pretende si el sistema que genera el código esta poco elaborado se limita también el código generado.

En el proyecto se ha alcanzo un nivel parejo en ambos sistemas llegando a ser capaces de procesar un texto en español y generar un código muy simple del mismo. El código generado es muy simple pues se hace referencia a los primeros pasos que realizaría una persona a la hora de aprender a programar.

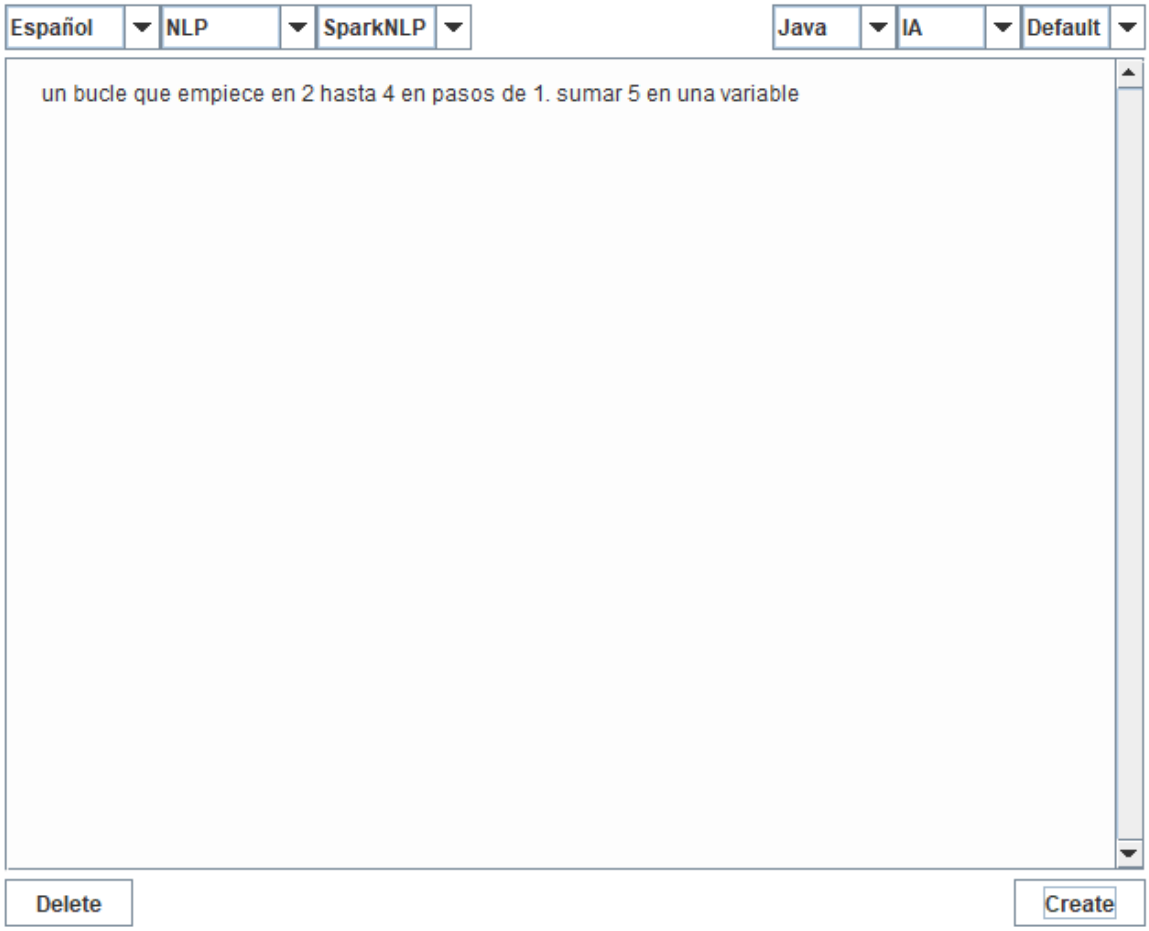

Ilustración 2 - Texto introducido en la aplicación

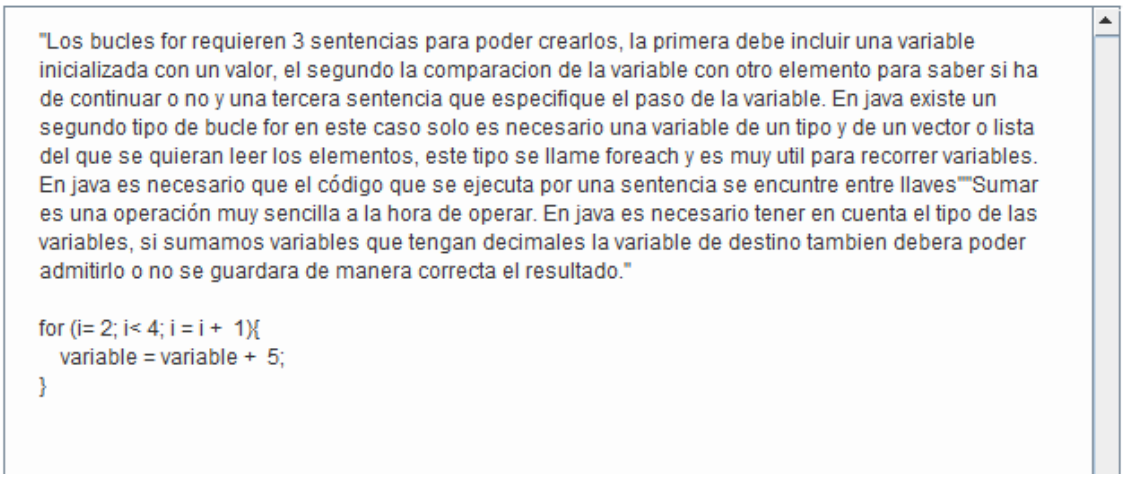

Ilustración 3 - Resultado de la generación de código

5. Conclusiones

La profundidad alcanzada en el proyecto alcanza los niveles más básicos de una aplicación que cumpla con el objetivo propuesto. Los códigos generados son simples para cualquier programador con cierta experiencia, pero para alguien nuevo en el mundo puede suponer un gran reto.

El procesamiento natural del lenguaje es complejo, pues si bien es cierto que existe una lengua común, esta puede ser utilizada de distintas maneras de tal forma que se podría decir que cada individuo tiene y desarrolla su propio lenguaje. Esto hace que sea difícil comprender del todo lo que un usuario intenta transmitir provocando que se puedan producir diferentes errores.

En cuanto a la generación de código un mismo programa puede ser escrito de varias maneras diferentes. Es por eso por lo que se ha buscado un método general y no específico de cada problema, pues se espera solucionar una gran cantidad de problemas y no optimizar la solución a un pequeño grupo de problemas. La generación de código presenta un problema en la generación de códigos extensos y se trata de la repetición de nombres de variables.

Finalmente, el programa puede servir de base y seguir desarrollándose en un futuro añadiendo diferentes módulos en ambos segmentos del proyecto, incrementado la profundidad de los ya existentes o finalmente expandiendo el árbol de posibilidades del vector unión de ambos segmentos.

6. Referencias

[1] Clement, C. B. (November 16–20, 2020). Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 9052–9065. Association for Computational Linguistics. Obtenido de https://aclanthology.org/2020.emnlp-main.728.pdf

[2] Transifex. (2013). Translating Natural Language to Code. Transifex. Obtenido de https://es.transifex.com/blog/2013/translate-natural-language-to-code/

[3] Weaver, W. (july 15, 1949). Translation. New mexico: The Rockefeller Foundation. Obtenido de https://gunkelweb.com/coms493/texts/weaver_translation.pdf

[4] M. Youssef, K. Zakaria, B. Jamal, B. Toumi and B. M. Gaouth, "Text to Code Conversion Using Deep Learning for NLP," 2020 International Symposium on Advanced Electrical and Communication Technologies (ISAECT), 2020, pp. 1-5, doi: 10.1109/ISAECT50560.2020.9523647

[5] The Code2Text Challenge: Text Generation in Source Code Libraries. https://arxiv.org/ftp/arxiv/papers/1708/1708.00098.pdf

[6] https://ml4code.github.io/base-taxonomy/generative.html

[7] https://metatext.io/datasets/codexglue:-concode

[8] Text to Code: Pseudo Code Generation. November 2019. DOI:10.1007/978-3-030-34365-1_3 In book: Context-Aware Systems and Applications, and Nature of Computation and Communication (pp.20-37). https://www.researchgate.net/publication/336928377_Text_to_Code_Pseudo_Code_Generation

[9] https://www.springerprofessional.de/en/text-to-code-pseudo-code-generation/17337634

[10] https://stanfordnlp.github.io/CoreNLP/

[11] https://nlp.johnsnowlabs.com

APPLICATION TO GENERATE CODE FROM NATURAL LANGUAGE

Author: Giménez Suárez, Sergio Félix.
Supervisor: Contreras Bárcena, David.
Collaborating Entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

The aim of this project is the creation of an application, that applying different natural language existing tools, ends up producing the code related to the text. The generated code will be available in two different programming language as Python and Java are. At last but not least, The final use of the application is to simplify the process of training new software developers.

Keywords: Python, Java, Natural Language Processing

1. Introduction

Nowadays, we can find multitude of electronic devices everywhere. Those devices require code to function or to complete some tasks. As society, we have reached to the point on relaying on electronics in all daily basis and some people will not be able to live without them.

The developing software industry is in constant expansion, and it is required more software developers to fulfill the vacancies. Due to this fact, it is required to simplify the process of training new developers. In order to be able to align with this increasing need we require to simplify and create new tools to train new people the art of programming. The application will be a new tool that will teach the basis of programming for those who have never seen anything from this world. To do this the application will generate the code from the text that the user insert.

Currently, we can find few tools regarding NLP (Natural language processing) but not so many for generating code, as those are still in development.

2. Project definition

The project has multiple difficulties. The first problem we face is the capability to fully understand what the user wants to say. This is because even using the same language different people use it in their own way utilizing different words and different descriptions for the same thing. This difficult is the same that the translation apps.

The second problem is given by the programming language to use. There are multiple languages with different characteristics. The most important one allows to classify them into typed and non-typed programming languages. Typed languages give the capability to prevent error in the execution of the code. This is because the variables are from a specific type and will not be able to be use in some specific functions.

In this specific case generated codes by the program are found in Java and Python languages, which are today the most used ones. As it is expected that we cannot

guarantee a perfect understanding form the text so the generated code will be executable but not optimized for this would require a perfect understanding of the languages.

3. Tool description

As it was said before, the program is divided in two different parts. The first one is related with the natural language processing; in this case we have used “CoreNLP”[10] and “SparkNLP”[11]. These two tools are used to process the text as a mean to get the grammatical elements of the words from a text. In order to get this information a model is trained. The second one makes reference to the part that generated the code. The code can be produced in Java or Python.

In addition, it is required a module that joins both sides. For this when the text is processed the key words are set in a vector that will be used to produce the code.

The program was designed to easily change any of the modules, natural language processing or the code generation.

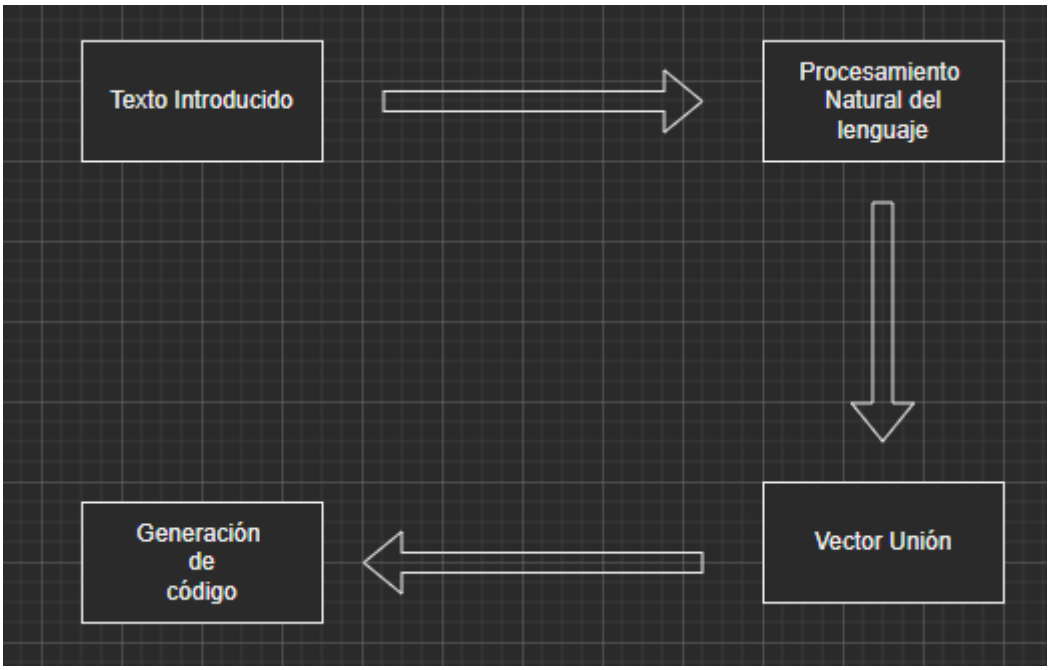

Ilustración 4 - Program function schema

4. Results

The Project was separated in multiple segments and the depth in each of them limits by them self the capability of the application. For example, the natural language processing limits the code generated as the understanding of the text may not be perfect and therefore the key words obtained may not produce a code at all. In the other hand, even

if the text was completely translated into the keywords, it doesn't produce a fully developed code as the keywords may not be processed by the generating code element.

The Project reached a similar level in both segments, being capable to process a Spanish text and generate the code from it. The generated code is simple as it makes reference to the first steps that a person will make when learning to program.

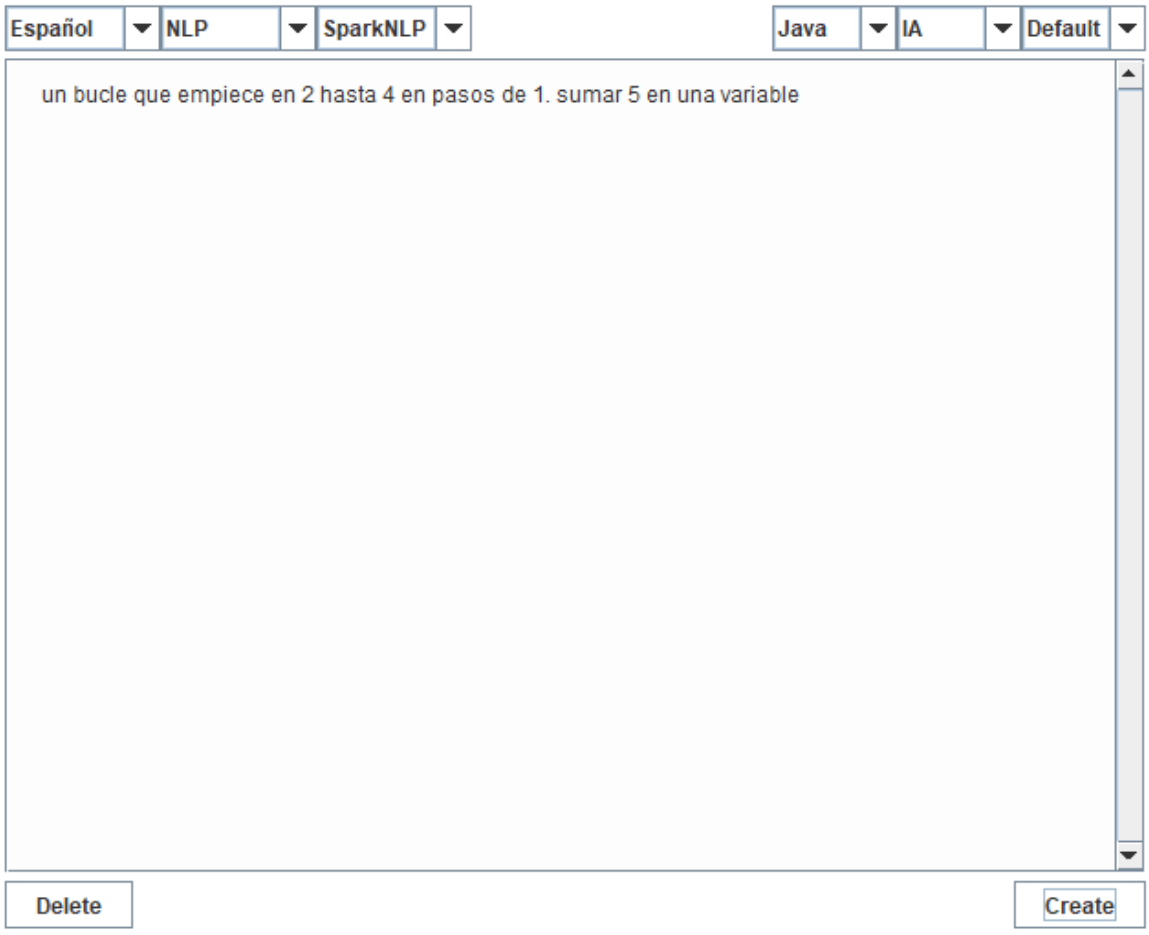

Ilustración 5 - Text introduced in the application

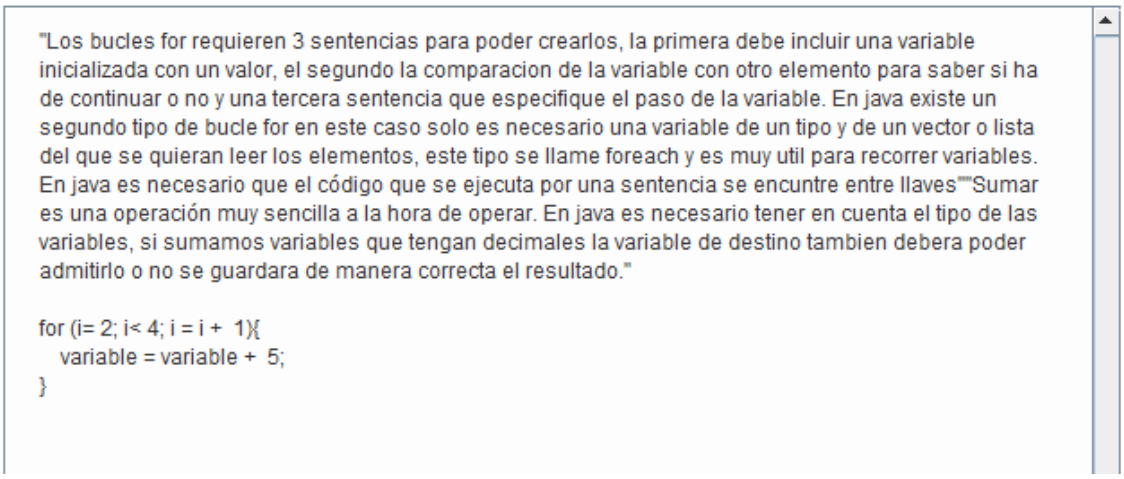

Ilustración 6 - Result of the code generation

5. Conclusions

The depth in the project reached the minimum required for an application to fulfill the pursued objective. The codes created are simple for any developer with some experience, but for someone just starting in this world it will cover the basis.

The natural language processing is complex. It is true that there exists a common language but each of us use it in our own way. This makes fully understanding a text extremely difficult, as these differences in the words use could be translated into errors.

The code from a specific program can be written in multiple ways. This is the reason why a general developing method was preferred over a specific one. Other reason for this is that it is expected to solve multiple problems but not optimizing any of them, as an optimized code will require a perfect explanation of what is wanted. The code generating presents a mayor problem when a long code is generated as the name of the variables cannot be repeated.

To finish, the program can be used as a base that can be expanded in the future as it was designed to add or replace modules easily, being the joint of both parts the limit.

6. Referencias

[1] Clement, C. B. (November 16–20, 2020). Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 9052–9065. Association for Computational Linguistics. Obtenido de https://aclanthology.org/2020.emnlp-main.728.pdf

[2] Transifex. (2013). Translating Natural Language to Code. Transifex. Obtenido de https://es.transifex.com/blog/2013/translate-natural-language-to-code/

[3] Weaver, W. (july 15, 1949). Translation. New mexico: The Rockefeller Foundation. Obtenido de https://gunkelweb.com/coms493/texts/weaver_translation.pdf

[4] M. Youssef, K. Zakaria, B. Jamal, B. Toumi and B. M. Gaouth, "Text to Code Conversion Using Deep Learning for NLP," 2020 International Symposium on Advanced Electrical and Communication Technologies (ISAECT), 2020, pp. 1-5, doi: 10.1109/ISAECT50560.2020.9523647

[5] The Code2Text Challenge: Text Generation in Source Code Libraries. https://arxiv.org/ftp/arxiv/papers/1708/1708.00098.pdf

[6] https://ml4code.github.io/base-taxonomy/generative.html

[7] https://metatext.io/datasets/codexglue:-concode

[8] Text to Code: Pseudo Code Generation. November 2019. DOI:10.1007/978-3-030-34365-1_3 In book: Context-Aware Systems and Applications, and Nature of Computation and Communication (pp.20-37). https://www.researchgate.net/publication/336928377_Text_to_Code_Pseudo_Code_Generation

[9] https://www.springerprofessional.de/en/text-to-code-pseudo-code-generation/17337634

[10] https://stanfordnlp.github.io/CoreNLP/

[11] https://nlp.johnsnowlabs.com

Índice de la memoria

Índice de figuras

Índice de tablas

Capítulo 1. INTRODUCCIÓN

En la actualidad la tecnología está completamente integrada con nuestro día a día llegando al extremo de que algunas personas serían incapaces de vivir sin ella. Todos estos dispositivos electrónicos poseen un código que les permite funcionar y este código hoy en día ha sido estrictamente desarrollado por un humano. Si nos paramos a observar cuantos dispositivos existen y cuantos códigos son necesarios para hacer que funcionen veríamos que la magnitud de lo realizado hasta la fecha es descomunal. Esto se puede ver con claridad en el siguiente ejemplo, sí una sola persona quisiera programar todo el código necesario para hacer funcionar un coche actual necesitaría de 20.000 años para completar la tarea. Es por esto qué se necesita un gran número de programadores.

El sector del desarrollo de software no ha parado de crecer desde que nació y cada año parece crecer más que el anterior, con miles de empresas invirtiendo millones para el desarrollo de sus proyectos. La existencia de tantas empresas en el mercado se da no solo por la cantidad de clientes existentes, sino también por la existencia de múltiples lenguajes de programación. Ciertos lenguajes de programación fueron creados específicamente para determinadas tareas y en el estado actual de las tecnologías no podríamos seleccionar a un lenguaje como el mejor entre todos, pues aquellos que son versátiles y permiten hacer multitud de funciones con distintas utilidades carece de profundidad en ellas o bien es más lento en el desempeño. A pesar de esto, dadas las características de cada lenguaje han existido claros "favoritos" que han tenido más apoyo que otros. Entre estos encontramos Java, un lenguaje antiguo, pero versátil ya que permite ser utilizado independientemente del dispositivo. Otro lenguaje más actual y que parece ir teniendo el apoyo de cada vez más programadores es Python, el cual también permite una gran versatilidad en cuanto a las funciones que se pueden desarrollar, siendo además un lenguaje muy simple, esto se debe a que programar en este lenguaje elimina parte de la sintaxis que hace tedioso la programación. Es por esto qué estos dos lenguajes has sido seleccionados para el proyecto y más adelante se especificará más sobre ellos.

A pesar de la infraestructura actual y el desarrollo del sector sigue habiendo una escasez de desarrolladores, y es que las empresas buscan en su mayoría solo a desarrolladores con experiencia, siendo muy pocas las empresas que ponen un esfuerzo en formar ellas mismas a estos programadores. Este problema se agrava pues existen más plazas de trabajo que trabajadores que puedan suplirlas, lo que da una posición de ventaja a los programadores. Esta escasez de programadores presenta un problema y se ha intentado suplir con distintos programas de enseñanza, como podría ser 42[1]. Otro tipo de soluciones que se han buscado

[1] Campus de programación innovador llevado a cabo por telefónica.

ha sido la creación de manera artificial de código, siendo actualmente el proyecto más vanguardista "Github Copilot" [2]de Microsoft, que actualmente se encuentra en desarrollo.

Hoy en día, los programadores se ayudan entre sí, y es muy raro no encontrar similitudes entre códigos realizados por diferentes personas, y es que gracias a los distintos foros existentes muchos programadores solucionan los problemas de otros o aportan sugerencias. Con esto último se quiere llegar a la idea de que, si bien existen millones de códigos para todos los dispositivos electrónicos, estos no tienen códigos únicos. La mayoría de los códigos que encontramos son copias o versiones muy ligeramente distintas pues la funcionalidad buscada es la misma y por tanto carecería de sentido que fueran muy diferentes.

El proyecto tiene como intención simplificar el proceso de aprendizaje de los nuevos programadores y en el mejor de los casos incluso evitar que una persona sobrecalificada se dedique al desarrollo de códigos simples. Esto se consigue gracias a la traducción de un elemento común entre todos los humanos, el lenguaje natural. Se busca un sistema que traduciendo desde el lenguaje natural sea capaz de crear el código referente a este dando las explicaciones más simples sobre el código generado.

La creación de este tipo de aplicación presenta ciertos problemas. Siguiendo un orden secuencial el primer problema con el que nos topamos se trata del sistema capaz de comprender el lenguaje natural, pues dependiendo de lo que se comprenda del texto el código puede ser completamente diferente o inclusive no llegar a producirse uno. Esto se debe a que, si bien se comparte una misma lengua, cada persona la usa de manera propia, creando pequeñas diferencias y haciendo que finalmente cada persona tenga su propio lenguaje, definiendo de manera ligeramente diferente lo mismo.

El segundo problema proviene de la parte que se encarga de la creación de código pues, aunque haya una comprensión perfecta del texto puede no necesariamente traducirse a código ya que no se pueda producir, bien por no estar implantada la lógica necesaria para ella o bien por que falten datos para el programa.

[2] Proyecto para la creación de código desarrollado por GitHub y OpenAI

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

En este apartado se describen las distintas tecnologías y herramientas que han sido utilizadas para la realización del proyecto.

2.1 VISUAL STUDIO CODE

Visual Studio Code es un entorno de desarrollo de software desarrollado por Microsoft. Este entorno de desarrollo es compatible con multitud de lenguajes de programación y cuenta con multitud de herramientas para simplificar el desarrollo de los programas.

Este entorno de desarrollo permite la navegación a través de los distintos archivos abiertos, la visualización simultanea del fichero, así como los cambios aplicados. Es compatible con GitHub, permitiendo subir de manera sencilla los cambios en el desarrollo. Además, permite el uso de la línea de comandos. Las ayudas al usuario son muchas y encontramos una gran variedad de extensiones compatibles para adaptar el entorno a nuestras necesidades.

Entre estas extensiones que ayudan al usuario encontramos algunas que nos permiten compilar y ejecutar el código sin necesidad de pasar por la línea de comandos añadiendo a mano cada archivo de dependencia del programa, simplificación a la hora de crear los métodos más comunes de los objetos o la capacidad de comprobar si una palabra es gramaticalmente correcta.

2.2 GITHUB

GitHub se usará para la sincronización de los archivos entre distintos dispositivos en los que se realice el desarrollo y como copia de seguridad del proyecto. GitHub aporta ciertas funciones que nos permitirán ver la evolución del proyecto. El entorno de desarrollo como se ha mencionado es compatible y permite simplificar el desarrollo.

GitHub es una plataforma online que permite crear proyectos tanto públicos como privados, sirviendo como almacenamiento en nube de millones de proyectos. Estos proyectos pueden tener múltiples ramas de desarrollo para que diferentes desarrolladores no interfieran unos con otros. Además, el servicio ofrece estadísticas de desarrollo para comprobar el ritmo y evolución del proyecto.

2.3 JAVA

Java es un lenguaje de programación muy conocido actualmente. Se trata de un lenguaje de programación orientada a objetos principalmente que simplifica mucho el desarrollo gráfico de aplicaciones. Este lenguaje además aporta la capacidad de poder ser ejecutado independientemente del hardware del dispositivo pues requiere de su propia máquina virtual. Java se trata de un lenguaje tipado, por lo que nos brinda la seguridad de que no se producirán errores durante la ejecución del código, si bien nos hace ser más estrictos durante el desarrollo.

2.4 PYTHON

Python en un lenguaje relativamente nuevo, que se está ganando el favor del público, creciendo de manera rápida gracias a la publicación constante de librerías por parte de los usuarios. Este hecho se ha dado gracias a que el lenguaje es simple y amigable con el usuario a la hora de programar. Gracias a la cantidad de librerías que se publican a diario la versatilidad el lenguaje es enorme y permite su uso en múltiples disciplinas diferentes. Además, permite la ejecución de otros lenguajes de programación por lo que es compatible con distintos lenguajes de uso muy específico.

2.5 PROCESAMIENTO DEL LENGUAJE NATURAL

El procesamiento natural del lenguaje se trata de uno de los elementos clave del sistema. Este procesamiento se puede llevar a cabo de múltiples maneras, pero en nuestro caso nos ayudaremos de herramientas ya desarrolladas y de uso libre.

2.5.1 CORENLP

"CoreNLP" es una herramienta de procesamiento natural del lenguaje compatible con múltiples idiomas, si bien solo podemos usar su completa capacidad procesando textos en Ingles. La herramienta está desarrollada en Java y permite su fácil integración con este leguaje.

CoreNLP está basada en la lematización[3], así como distintas herramientas para obtener la información del texto. Esta herramienta es capaza de obtener las distintas categorías gramaticales de las palabras que encontramos en el texto. Además, es capaz de encontrar relaciones entre las distintas palabras, así como la intencionalidad de la oración.

Esta herramienta solo presenta un problema para el proyecto, y es que, si bien permite la lematización de las palabras, esta funcionalidad solo la encontramos para el procesado de textos en inglés.

2.5.2 SPARKNLP

"SparkNLP" en otra herramienta para el procesamiento natural del lenguaje. Esta herramienta es más extensa que la anterior, permitiendo no solo procesar textos en diferentes idiomas, sino también su traducción. Esta herramienta cuenta con múltiples modelos entrenados y de uso libre. A diferencia de la anterior, esta herramienta es más compleja de utilizar y no cuenta con una guía de utilización en Java.

SparkNLP también está basada en la lematización, así como la implementación de distintas herramientas para comprender los diferentes textos. En este caso la herramienta está más pulida y se encuentran incluso para el mismo idioma y para cada funcionalidad distintos modelos entrenados. Además, se permite la opción de cargar de manera individual un modelo diferente en función del procesado que se quiera realizar pudiendo personalizar el

[3] Método de procesamiento de lenguaje natural basado en diccionarios

sistema de procesado para obtener una mayor precisión. Esta funcionalidad la encontramos en multitud de idiomas, lo que nos permite su fácil implementación en nuestro sistema.

2.6 HADOOP

Hadoop es un *framework*[4] que permite el desarrollo de software de manera confiable y escalable. Es una libraría para el procesamiento de grandes cantidades de datos permitiendo el uso desde un solo servidor hasta cientos de dispositivos. Esta herramienta es utilizada para poder llevar a cabo el procesamiento natural del lenguaje.

2.7 LUNACY

Lunacy es una aplicación que permite el diseño y creación de iconos. Esta aplicación fue desarrollada por el portal Icons8[5]. La aplicación ha sido utilizada para la creación de e todos los iconos que se puedan encontrar en la aplicación.

[4] Esquema trabajo que ofrece una estructura para el desarrollo
[5] Portal web gratuito con iconos de muestra para utilización en diferentes proyectos

Capítulo 3. ESTADO DE LA CUESTIÓN

El estado del arte se puede segmentar en varias partes. La primera de ella trata sobre los proyectos que de manera individual han buscado profundizar en el procesamiento del lenguaje natural. En segundo lugar, los proyectos que se basan en la generación automática, tema que se aproxima a los diferentes programas que se encargan de autocompletar el código. Por último, los proyecto que tienen un planteamiento parecido, pero con una ejecución diferente.

3.1 PROCESAMIENTO DEL LEGUAJE NATURAL

El procesamiento natural del lenguaje se ha estudiado a lo largo de años y ha sido un tema de interés desde que la programación se hizo un hueco en la historia. Se trata de una ciencia que trata de comprender y relacionar el lenguaje humano con el de las máquinas. Las primeras reglas de procesamiento del lenguaje natural eran complejas, extensas y realizadas a mano. Todo cambió con la aparición de las inteligencias artificiales capaces de auto aprender provocando así un auge del sector. Este tema guarda gran relación con los sistemas actuales de traducción de idiomas ya que es indispensable para su ejecución y tanto es así que la primera aproximación de este tema fue para realizar una traducción del ruso al inglés.

Todas estas distintas aproximaciones desembocaron en multitud de proyectos que sirvieron para establecer aspectos en común, pero también ciertas diferencias. La primera diferencia que se encontró es que la misma aproximación no servía para todas las lenguas pues cada una posee rasgos que la convierten en única, entre estos rasgos encontramos la estructura y ordenación de las oraciones, así como el uso de conectores, terminaciones y conjugaciones. Cabe destacar que de manera más interna cada lengua posee además ciertas metáforas cuyo significado viene dado por la historia y cultura de cada país, haciendo que cada lengua posea distintos refranes y dichos. Todos estos elementos, como se ha dicho hacen imposible un sistema idéntico para cada idioma, pero si permite utilizar una misma aproximación.

Existen distintas aproximaciones encontrando entre estas las siguientes.

3.1.1 STEMMING

Stemming hace referencia al sistema de procesado del lenguaje natural basado en la eliminación tanto de prefijos como de sufijos dejando así la raíz de la palabra. Así mismo estos prefijos y sufijos sirven para dotar de cierta información a la palabra, por ejemplo, haciendo referencia al tiempo o la persona. Usando estos datos podemos obtener información del significado de la frase.

Palabra	**Sufijo**	**Raíz**
niños	-os	niñ
comemos	-emos	com

Tabla 1 - Ejemplo de "Stemming"

Este sistema de procesado requiere saber las conjugaciones de cada lenguaje, incluyendo como en el español las formas irregulares.

3.1.2 LEMATIZACIÓN

La lematización es el sistema de procesado del lenguaje natural que se encarga de reducir cada palabra a su versión de diccionario. De este modo cualquier palabra en el texto es transformada a versión en infinitivo. Este método no tiene en cuenta las conjugaciones por lo que normalmente es acompañado de distintos sistemas para llevar a cabo una completa compresión del texto. Estas herramientas son usadas para obtener referencias entre palabras, intencionalidades, tiempos… Esto se puede realizar ya que las oraciones van acompañadas por palabras que aportan este significado.

Este tipo de procesado hace necesaria la existencia de un diccionario con todas la palabras de cada lengua. En caso de no encontrar una palabra en el diccionario esta no podría ser convertida a su forma de diccionario.

Palabra	Versión diccionario
Estamos	Estar
Coches	Coche

Tabla 2 - Ejemplo de lematización

3.1.3 BASE DE DATOS

El procesamiento del lenguaje natural no solo ha tenido aproximaciones a través de algoritmos. Entre estos métodos podemos encontrar las bases datos. Es caso es utilizado sobre todo por traductores de idiomas que se basan en encontrar el texto más parejo a lo escrito y dar como resultado su traducción. Este mecanismo es más simple que los anteriores y requiere de menos desarrollo para llevarlo a cabo, como contraparte se es necesario la existencia de grandes bases de datos. El grado de acierto de este método esta directamente relacionado con el tamaño de la base de datos y que esta sea independiente.

3.2 CREACIÓN AUTOMÁTICA DE CÓDIGO

En la actualidad los sistemas que se encargan de la creación de código suelen ser sistemas de ayuda para los programadores. Estos sistemas tienden a autocompletar el texto sobrante de métodos o de funciones ya definidas, así como terminar de escribir cierta parte de sintaxis. Estos sistemas llevan tiempo incorporados en los diferentes entornos de desarrollo y con el paso del tiempo han ido mejorando y aumentando su capacidad.

Otra aproximación son la aparición de diversos frameworks con topologías de programas ya preparados. Estos frameworks también añaden herramientas con las que con simple anotaciones se puede crear la estructura básica de ciertos objetos, es decir, se crean los métodos comunes a todos los objetos.

3.2.1 SPRINGBOOT

Springboot se trata de un framework para aplicaciones que requieren de la existencia de una base de datos. Este framework aporta los elementos básicos del back end [6] de una aplicación, entre estos encontramos elementos de seguridad, algunas clases básicas y el sistema de prueba para comprobar que no existen errores en la aplicación.

3.2.2 MAVEN

Maven es un framework de java que facilita la utilización de librerías externas en un código. En concreto, nos permite añadir librerías de una manera simple al especificar susodichas librerías en un fichero. Estas librerías podemos encontrarlas en su propia base de datos. Además, esta herramienta da al usuario diferentes códigos base diseñados en específico para distintos tipos de proyecto.

3.3 GITHUB COPILOT

GitHub Copilot es un proyecto en desarrollo que tiene como objetivo la creación de una herramienta capaz de a partir de lenguaje natural, ya sea este introducido como un texto o bien como un archivo de audio, generar el código subsecuente. Este proyecto lleva en desarrollo varios años y ha ido de la mano de Microsoft, quien ha invertido millones de dólares e incluso adquirido distintas empresas para el proyecto. Actualmente el proyecto ha publicado con acceso libre una versión beta[7] de su herramienta.

El proyecto se fundamenta en la utilización de los códigos, ya revisados por otros programadores y que cuentan con un gran número de apoyo. Estos códigos son ligeramente modificados para poder ser usados. Se eliminan nombres de variables y se otorgan definiciones generales que puedan ser cambiadas al momento de la generación del código.

[6] Parte de la aplicación que se encarga del manejo de datos y que es opaca para el usuario
[7] Fase de desarrollo de un programa que dispone de elementos funcionales

Capítulo 4. DEFINICIÓN DEL TRABAJO

4.1 MOTIVACIÓN

La motivación del proyecto nace de dos fuentes. Siendo la primera el estado actual de la industria del desarrollo de software, donde el número de plazas libres de trabajo es superior al número de personas cualificadas del sector. Haciendo que las distintas empresas tengan dificultad para mantener a un mismo individuo en la compañía por largos periodos de tiempo que no llegan a abarcar si quiera la duración de un solo proyecto. Este hecho dificulta y entorpece el desarrollo de los códigos y provoca atrasos en los desarrollo.

El segundo punto es el estado actual de la sociedad donde a cada día que pasa encontramos la tecnología cada vez más integrada entre nosotros. Hemos llegado al extremo donde personas del primer mundo no serían capaces de vivir ni un solo día sin estos dispositivos. Además, el nivel de desarrollo tecnológico es a día de hoy un factor que mide el estado de un país. Todos estos dispositivos que encontramos poseen un pequeña parte de Código que los hace funcionar, y en otros casos disponen de una cantidad inmensa de Código para poder llevar a cabo todas sus funciones. Un ejemplo de esto dos mundos serian un lavavajillas y un coche. El primero apenas necesita de instrucciones que definan el estado en el que se encuentra la máquina, mientras que un coche requiere de múltiples sensores y ajustes continuos. Si se quisiese desarrollar todo el código por una sola persona, le llevaría unos 20000 años de trabajo.

Esto hace que sea necesario el desarrollo de nuevas herramientas que, por un lado, simplifiquen el proceso de aprendizaje para así introducir a nuevas personas. Y por segundo, que evite si es posible la distracción de una persona cualificada en el desarrollo de un Código simple.

.

4.2 OBJETIVOS

La intención en el proyecto es la creación de una herramienta capaz de generar código a partir del lenguaje natural con la intención de obtener dos resultados.

El primero de ellos se trata de simplificar el proceso de aprendizaje de los nuevos programadores. Esto se consigue mediante la explicación de las bases de la programación en los códigos generados. Estas explicaciones se basan en el lenguaje de programación escogido, detallando las peculiaridades de cada uno.

El segundo de ellos es al aportar una aplicación capaz de generar códigos simples y que sea sencilla de usar, para dotar a personas sin conocimiento sobre la programación de la capacidad de producir programas simples, para así, de esta manera evitar que programadores experimentados se dediquen a programas simples cuando deberían dedicarse a proyectos más complejos y que requieren programadores con experiencia.

4.3 METODOLOGÍA

El desarrollo del proyecto se ha realizado utilizando el modelo de cascada, basado en la implementación de funcionalidad poco a poco. El proyecto ha tenido la duración de un curso académico empezando en septiembre y acabando en agosto del siguiente año.

	Septiembre	Octubre	Noviembre	Diciembre	Enero	Febrero
Tema						
Estado del Arte						
Diseño						

Tabla 3- Planificación del proyecto I

Esta primera planificación corresponde con la investigación y organización del proyecto, así como el desarrollo de la interfaz gráfica del proyecto.

	Febrero	Marzo	Abril	Mayo	Junio	Julio	Agosto
Módulo de NLP							
Módulo generación de código							
Desarrollo memoria							

Tabla 4- Planificación del proyecto II

La segunda planificación hace referencia tanto al desarrollo principal de las funcionalidades del proyecto, así como de la memoria descriptiva del proyecto.

Algunos de los desarrollos se han llevado en paralelo, esto se debe a la transición entre los distintos módulos. Ambos sistemas deben coincidir en un punto para que el producto de uno sirva como comienzo del otro.

Actividad	**Horas**
Tema	5 h
Estado del arte	25 h
Diseño	20 h
Módulo de NLP	100 h
Módulo generación de código	70 h
Desarrollo de la memoria	120 h
Total	340 h

Tabla 5- Número de horas utilizadas en el proyecto

4.4 ESTIMACIÓN ECONÓMICA

En esta sección se hará un resumen del coste del proyecto. Todos los programas utilizados son open source[8]. Entre estos encontramos tanto los sistemas de procesamiento natural del lenguaje como de los modelos empleados para este fin. Así mismo, también la aplicación para el desarrollo de los iconos utilizados en la aplicación.

En este caso encontramos los siguientes costes.

[8] Utilización gratuita, en algunos casos también se permite su modificación. En algunos casos se requiere de citar la herramienta.

Concepto	Coste	N.º de horas	Total
Programador	14,62 €	340 €	4.970,80 €
Portátil	900,00 €	-	900,00 €
Total			**5.870,80 €**

Tabla 6 - Coste del proyecto

Capítulo 5. SISTEMA/MODELO DESARROLLADO

En este capítulo se explicará todo el desarrollo realizado para el trabajo de fin de grado. Se dará detalle de los distintos aspectos de la aplicación, la razón detrás de ciertas decisiones tomadas durante el desarrollo. Así como una explicación sobre distintas partes fundamentales del código de la aplicación.

5.1 ANÁLISIS DEL SISTEMA

La aplicación busca solventar dos problemas actuales en la industria del desarrollo de software. Estos son como se han mencionado antes tanto la simplificación del aprendizaje, así como evitar la distracción de los desarrolladores ya formados.

Esto nos hace tener una aplicación que disponga de pocos casos de uso. En concreto tenemos dos casos de uso en función de la intención del usuario, si bien es cierto que no cambia en nada la ejecución del programa. En caso de querer solamente el código generado se ignorará los comentarios añadidos al código, mientras que si el usuario busca aprender a programar realizará múltiples consultas y aprenderá gracias a los distintos comentarios que se realizan en el código.

5.2 DISEÑO

Durante el planteamiento de la aplicación se ha tenido en mente la posibilidad de que esta crezca con el tiempo, por lo que se es posible añadir módulos tanto para el procesamiento del lenguaje natural como para la generación de código. El único limitante de ambos es el vector unión que enlaza ambos sistemas.

Es decir, el sistema permite añadir de manera sencilla distintas formas de realizar procesamiento del lenguaje natural, idiomas, sistemas de generación de código y lenguajes de programación.

El diseño de la aplicación se ha dividido en distintas partes. Debido a que se realizan distintas funciones dentro del programa.

- Interfaz gráfica: Permite la interacción del usuario con el sistema.
- Módulo de procesamiento del lenguaje natural.
- Módulo de generación de código.
- Segmento de unión de los diferentes módulos.

5.2.1 INTERFAZ GRÁFICA

El diseño de la interfaz sólo cuenta con una ventana donde el usuario dispone de 3 paneles. El primer panel representa una caja donde el usuario introducirá el texto a procesar y podrá seleccionar las distintas opciones para el procesado y la generación de código. El segundo panel es un cuadro informativo del estado de la aplicación. Sirve para indicar al usuario del avance y progreso. Por último, una última caja de texto donde se hallará el código generado por la aplicación.

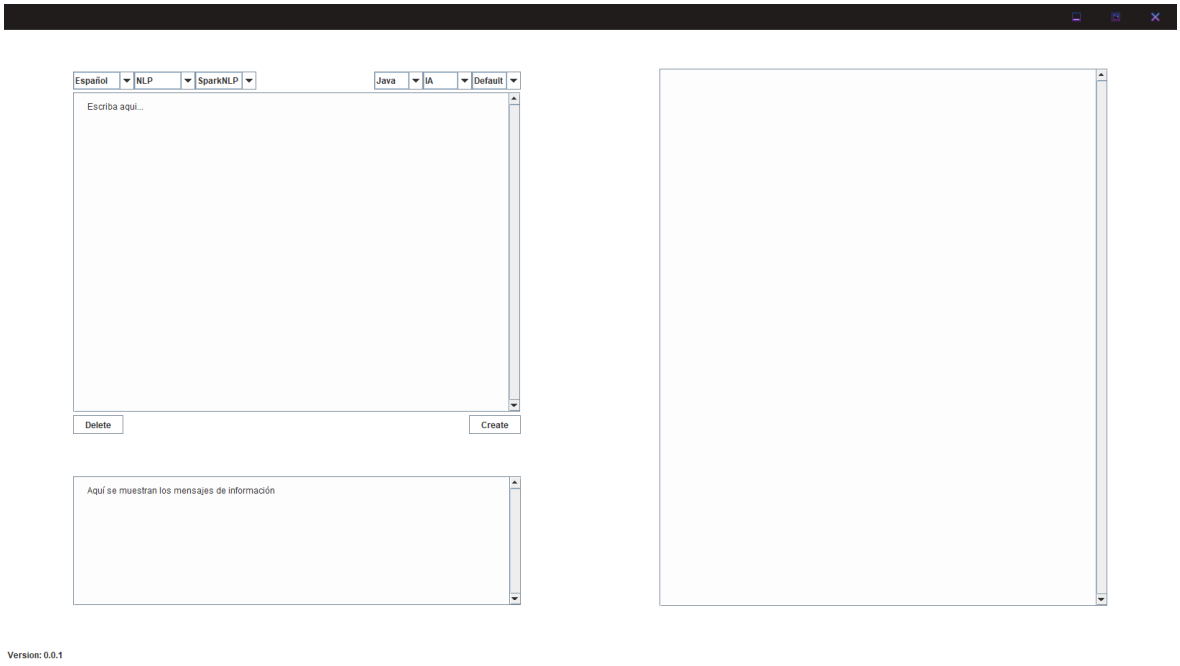

Ilustración 7- Interfaz gráfica de la aplicación

5.2.2 MÓDULO DE PROCESAMIENTO DEL LENGUAJE NATURAL

Este módulo se encarga de recoger el texto y procesarlo para hallar las palabras claves que servirán para la creación del código. Se han escogido dos herramientas diferentes para la realización de este proceso.

5.2.2.1 CoreNLP

Se ha usado esta herramienta para el análisis del texto en una versión simplificada. Esta herramienta solo es utilizada para obtener la categoría gramatical de las palabras y las relaciones entre estas.

Estos dos valores se han usado para la creación de vectores intermedios que especifiquen tanto el tipo de palabra, así como su relación. Estos vectores son revisados consecutivamente para formar un último vector que contenga las palabras clave necesarias.

5.2.2.2 SparkNLP

Se ha usado esta herramienta para el análisis del texto en mayor profundidad. Esta herramienta se utiliza, no sólo para obtener las categoría gramatical de las palabras y sus relaciones, sino que también realiza lematización, convirtiendo de esta manera las palabras a su forma de diccionario para su uso. Gracias a este hecho se puede realizar una búsqueda en un diccionario de sinónimos para las distintas palabras clave que aparezcan.

Estos tres vectores son recorridos e inspeccionados múltiples veces para generar el vector con las palabras clave para la generación del código.

5.2.3 MÓDULO DE GENERACIÓN DE CÓDIGO

Es módulo recibe como entrada un vector con palabras clave y tiene como salida un String[9] que contiene tanto los comentarios como el código que ha generado. Este módulo en función del número de parámetros introducidos llama a distintas funciones que van acoplando el código referente.

En caso de faltar palabras clave el programa presentaría un comportamiento anómalo. Estos comportamientos pueden ser desde la generación del código sin la presencia de los nombres de variables, hasta el fallo por completo del sistema provocando que no se genere ningún código.

5.2.4 SEGMENTO DE UNIÓN DE LOS DIFERENTES MÓDULOS

La aplicación usa un vector común como método de unión de los módulos. Este vector presenta las palabras clave de los códigos, y especifica tanto la función que se quiere realizar, valores, nombres de variables y subsecuentes funciones como podrían necesitar los bucles. En función de la longitud entre las palabras clave que definen las funciones básicas (sumas, bucles, creación de variables entre otros) se obtiene cuantos parámetros han sido especificados para la función. Estas palabras clave siguen un orden, donde primero se

[9] Cadena de caracteres, desde una palabra a un texto completo.

especifica la función, después los valores, nombres y por último las funciones subsecuentes en caso de ser necesarias. Si se introdujera una función subsecuente en una función que no precisa de esta, esta se ignoraría.

5.3 IMPLEMENTACIÓN

En esta sección se mostrará la implementación del código que se ha realizado para el proyecto. El código se ha dividido en secciones.

Todo el desarrollo del proyecto se ha realizado en Java. Se ha escogido este lenguaje debido a que permite ser ejecutado independientemente del hardware del dispositivo, siempre que se disponga de la máquina virtual.

5.3.1 INTERFAZ GRÁFICA

La interfaz gráfica se compone de un panel principal que contiene todos los elementos. Este panel se divide en 3 secciones. La sección norte contiene la barra de herramientas, la zona central toda la funcionalidad del programa y el panel sur contiene la información de la aplicación.

El panel tiene dos funciones, una para inicializar configuración y otro para establecer los componentes.

- Panel principal:

```
private void initConfig(){

        // set the icon of the application
        try {
            this.setIconImage((new
ImageIcon("src/main/resources/icons/Main_icon.png").getImage()));
        } catch (Exception e){
            e.printStackTrace();
        }

        ////////////////////////////
        // Generic configuration //
        ////////////////////////////
        setUndecorated(true); // Delete the common control bar of the app
```

```
        // setShape(new RoundRectangle2D.Double(0, 0, Screen.getScreenWidth(),
Screen.getScreenHeight(), 25, 25)); // Set rounded corners
        setMaximumSize(new Dimension((int)(Screen.getScreenWidth()*
0.85),(int)(Screen.getScreenHeight()*0.85))); // Set the maximum size of the
window
        setMinimumSize(new Dimension((int)(Screen.getScreenWidth()*
0.85),(int)(Screen.getScreenHeight()*0.85))); // set the minimum size of the
window
        setSize(new Dimension((int)(Screen.getScreenWidth()*
0.85),(int)(Screen.getScreenHeight()*0.85))); // Set the size of the window
        setResizable(false); // Prevent changing the size of the app
        setTitle(title);  // Set the title
        setVisible(true);  // Make the window visible
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  //The process will end
when the app closes
        setLocationRelativeTo(null);   // The app will start centered in the
screen
        requestFocus(); // Sets the focus on this component
        ////////////////////
        // Window Events //
        ////////////////////

        addWindowListener(new WindowAdapter() { // Confirmation to close the
program
            /**
             * Ask the user for a confirmation to close the app
             * @param evt When the app is about to be closed
             */
            public void windowClosing(WindowEvent evt) {
                Object[] options = {"si", "No"};
                int answer =
javax.swing.JOptionPane.showOptionDialog(null,"¿Seguro quieres salir?",
                 "Exit?",javax.swing.JOptionPane.DEFAULT_OPTION,
javax.swing.JOptionPane.WARNING_MESSAGE,new
ImageIcon("src/main/resources/icons/Alert.png"),options,null);

                if (answer == JOptionPane.YES_OPTION) {
                    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE); // Finish
the app
                } else {
                    setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE); //
Cancels the closing operation by doing nothing
                }
            }
        });

        addWindowListener(new WindowAdapter() { // Return to the previous state
            public void windowDeiconified(WindowEvent evt) {
                setExtendedState(state);
            }
        });

    }
```

Y para inicializar los componentes

```
private void initComponents(){

        ////////////
        // Layout //
        ////////////

        setLayout(new BorderLayout()); // window Layout type

        // North panel
        JPanel pnlNorth = new JPanel(new FlowLayout(FlowLayout.RIGHT)); //
Initiates the north panel with a basic layout
        pnlNorth.setBackground(Colors.COD_BLACK); // Change background color of
the panel

        // Minimize
        JButton btnMinimize = new JButton(); // Create a blank button
        btnMinimize.setIcon(new
ImageIcon("src/main/resources/icons/Minimize.png")); // Set an icon to the button
        btnMinimize.setBorderPainted(false); // Delete the border
        btnMinimize.setBackground(Colors.COD_BLACK); // Change button color

        // Resize
        JButton btnSize = new JButton(); // Create a blank button
        btnSize.setIcon(new ImageIcon("src/main/resources/icons/Size.png")); //
Set an icon to the button
        btnSize.setBorderPainted(false); // Delete the border
        btnSize.setBackground(Colors.COD_BLACK); // Change button color

        // Close
        JButton btnClose = new JButton(); // Create a blank button
        btnClose.setBorderPainted(false); // Delete the border
        btnClose.setIcon(new ImageIcon("src/main/resources/icons/Close.png"));//
Set an icon to the button
        btnClose.setBackground(Colors.COD_BLACK); // Change button color

        // Add the components to the north panel
        pnlNorth.add(btnMinimize);
        pnlNorth.add(btnSize);
        pnlNorth.add(btnClose);

        // Add north panel to the main window
        this.add(pnlNorth, BorderLayout.NORTH);

        // Add middle panel to the main window
        add(PnlMiddle.pnlMiddle, BorderLayout.CENTER); // Panel with all
functionality
```

```
        // South panel
        JPanel pnlSouth = new JPanel(new FlowLayout(FlowLayout.LEFT)); //
Initiates the south panel with left alignment
        pnlSouth.setBackground(Colors.WHITE);

        // version label
        JLabel lblVersion = new JLabel("Version: 1.0"); // Creates a label to
specify the version of the app
        // lblVersion.setForeground(Colors.WHITE); // Change the font color

        // Add the components to the south panel
        pnlSouth.add(lblVersion);

        // Add south panel to the main window
        add(pnlSouth, BorderLayout.SOUTH);

        ////////////////////
        // Button Events //
        ////////////////////

        // Minimize button action
        btnMinimize.addActionListener(new ActionListener() {
            /**
             * Button with change the state of the window
             * @param e when the button is pressed
             */
            // @Override
            public void actionPerformed(ActionEvent e){
                state = getExtendedState();
                setExtendedState(JFrame.ICONIFIED);
            }
        });

        // Size button action
        btnSize.addActionListener(new ActionListener() {
            /**
             * Change the size of the screen
             * @param e when the button is pressed
             */
            // @Override
            public void actionPerformed(ActionEvent e) {
                if (getExtendedState() == JFrame.MAXIMIZED_BOTH){ // Identify the
size of the window
                    setExtendedState(JFrame.NORMAL); // Change the recommended
size
                    setLocationRelativeTo(null);    // The app will start centered
in the screen
                    repaint(); // Repaint the app
                }
                else{
                    setExtendedState(JFrame.MAXIMIZED_BOTH); // Change to full
size
```

```
                    repaint(); // Repaint the size
                }
            }
        });
```

En el panel central se hayan otros 3 paneles. El primer panel es una caja para introducir el texto que será procesado. Este panel dispone de distintos botones para que el usuario seleccione la configuración del procesado y generación del código. El segundo panel contiene una caja de texto que sirve para mostrar el estado de la aplicación. El tercer panel se trata de una caja de texto donde se haya el código generado por la aplicación.

Todos estos paneles han sido creados utilizando el patrón singleton [10] ya que carece de sentido la existencia de varios de estos paneles en la misma aplicación. Tanto el panel para mostrar la información como el panel para mostrar el resultado heredan de la clase de java JScrollPane, añadiendo muy poco funcionalidad extra a este panel.

- Panel en el que se muestra el resultado:

```
public class PnlDisplayCode extends JScrollPane {

    /** Object of the type of the class to perform Singleton
     * programming as it makes no sense to have multiple instances
     * of this object in the program
      */
    public static PnlDisplayCode pnlDisplayCode = new PnlDisplayCode();

    /** Main component of the panel, it will display the code generated
     * by the program
     */
    JTextArea txtInfo = new JTextArea();

    /** Constructor method */
    private PnlDisplayCode(){
        initComponents();
        initConfig();
    }

    /** Method to initiate the components related to the panel */
    private void initComponents(){
        txtInfo.setLineWrap(true);
```

[10] Patrón de programación por el cual se limita el número de instancias del objeto a 1

```
        txtInfo.setBackground(Colors.WHITE_LIGHT);
        txtInfo.setMargin(new
Insets(Screen.getScreenHeight(0.01),Screen.getScreenWidth(0.01),Screen.getScreenH
eight(0.01),Screen.getScreenWidth(0.01)));
        txtInfo.setEditable(false);
        txtInfo.setWrapStyleWord(true);
    }

    /** Method to set the settings */
    private void initConfig(){
        setViewportView(txtInfo);
        setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
    }

    /** Method to clean the text box */
    public void clear(){
        txtInfo.setText(null);
    }

    /** Method to show the code created*/
    public void showCode(String[] text){
        this.clear();
        txtInfo.setText(text[0] + "\n\n" + text[1]);
    }

}
```

- Panel informativo:

```
public class PnlDisplayInfo extends JScrollPane{


    /** Object of the type of the class to perform Singleton
     * programming as it makes no sense to have multiple instances
     * of this object in the program
      */
    public static PnlDisplayInfo pnlDisplayInfo = new PnlDisplayInfo();

    /** Main component that will display the status of the program */
    private JTextArea txtInfo = new JTextArea("Aquí se muestran los mensajes de
información");

    /** Constructor method */
    private PnlDisplayInfo(){
        initComponents();
        initConf();
    }

    /** Method to instantiate the components */
    private void initComponents(){
```

```
        txtInfo.setLineWrap(true);
        txtInfo.setBackground(Colors.WHITE_LIGHT);
        txtInfo.setMargin(new
Insets(Screen.getScreenHeight(0.01),Screen.getScreenWidth(0.01),Screen.getScreenH
eight(0.01),Screen.getScreenWidth(0.01)));
        txtInfo.setEditable(false);
        txtInfo.setWrapStyleWord(true);
    }

    /** Method to set the settings */
    private void initConf(){
        setViewportView(txtInfo);
        setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
    }

    /** Method to clear the panel text*/
    public void clear(){
        pnlDisplayInfo.txtInfo.setText(null);
    }

    /** method to display a message */
    public void showMessage(String text){
        pnlDisplayInfo.txtInfo.setText(pnlDisplayInfo.txtInfo.getText() + "\n" +
text);
    }

}
```

Se han añadido un par de métodos para poder escribir en la caja de texto y borrar su contenido.

- Panel donde introducir el texto:

Se poseen distintos atributos

```
public static PnlTextBox PnlTextBox = new PnlTextBox();
    /** Component with the possible natural languages */
    private JComboBox<String> cbLanguages = new JComboBox<String>();
    /** Component with the possible programming languages */
    private JComboBox<String> cbCodes = new JComboBox<String>();
    /** Component with the possible methods for the natural language */
    private JComboBox<String> cbLanguagesMethods = new JComboBox<String>();
    /** Component with the possible methods for the programming language */
    private JComboBox<String> cbCodesMethods = new JComboBox<String>();
    /** Component with the possible options for the method and natural language
*/
```

```
    private JComboBox<String> cbLanguagesMethodsOptions = new
JComboBox<String>();
    /** Component with the possible options for the method and programming
language */
    private JComboBox<String> cbCodesMethodsOptions = new JComboBox<String>();
    /** Constraints to place the different components*/
    private GridBagConstraints gbc = new GridBagConstraints();
    /** Component that will show the text */
    private JTextArea txtNatural = new JTextArea("Escriba aqui...");
    /** Boolean to see if events are active or not */
    private boolean eventsActive = true;
```

para inicializar los distintos componentes

```
private void initComponents(){

        ////////////
        // Layout //
        ////////////
        setLayout(new GridBagLayout());

        // Instantiate the panel with the languages
        JPanel pnlLanguages = new JPanel(new GridLayout(1,3));
        setLanguages(); // Load the languages
        cbLanguages.setBackground(Colors.WHITE); // Set background color to white
        setLanguagesMethods(); // Load the possible methods
        cbLanguagesMethods.setBackground(Colors.WHITE); // Set background color to white
        setLanguagesMethodsOptions(); // Load the possible options
        cbLanguagesMethodsOptions.setBackground(Colors.WHITE); // Set background color to
white
        pnlLanguages.add(cbLanguages); // Add the component
        pnlLanguages.add(cbLanguagesMethods); // Add the component
        pnlLanguages.add(cbLanguagesMethodsOptions); // Add the component

        gbc.weightx = 1; // Extra space distribution
        gbc.weighty = 1; // Extra space distribution
        gbc.gridwidth = 1; // Number of columns
        gbc.gridheight = 1; // Number of rows
        gbc.gridx = 0; // Column place
        gbc.gridy = 0; // Row place
        gbc.anchor = GridBagConstraints.ABOVE_BASELINE_LEADING; // Place it in the upper
left corner of the main component
        add(pnlLanguages,gbc); // Adding the component to the grid with gbc settings

        // Panel with possible codes
        JPanel pnlCodes = new JPanel(new GridLayout(1,3)); // Grid with 1 row, 2 columns
        setCodes(); // Load the codes
        cbCodes.setBackground(Colors.WHITE); // Set Background color to white
        setCodesMethods(); // Load the possible methods
        cbCodesMethods.setBackground(Colors.WHITE); // Set the background color to white
        setCodesMethodsOptions(); // Load possible options
        cbCodesMethodsOptions.setBackground(Colors.WHITE); // Set the background color to
white
        pnlCodes.add(cbCodes); // Add the codes
        pnlCodes.add(cbCodesMethods); // Add the methods
```

```
        pnlCodes.add(cbCodesMethodsOptions); // Add the component

        gbc.weightx = 1; // Extra space distribution
        gbc.weighty = 1; // Extra space distribution
        gbc.gridwidth = 1; // Number of columns
        gbc.gridheight = 1; // Number of rows
        gbc.gridx = 3; // Column place
        gbc.gridy = 0; // Row place
        gbc.anchor = GridBagConstraints.ABOVE_BASELINE_TRAILING; // Place it in the upper
right corner of the main component
        add(pnlCodes,gbc); // Add the component to the grid with the gbc settings

        // Text box where the user will write
        JScrollPane pnlScroll = new JScrollPane();
        pnlScroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);


        txtNatural.setLineWrap(true); // End of line continue in the lower one
        txtNatural.setBackground(Colors.WHITE_LIGHT); // Set background color to white
        txtNatural.setMargin(new
Insets(Screen.getScreenHeight(0.01),Screen.getScreenWidth(0.01),Screen.getScreenHeight(0.01
),Screen.getScreenWidth(0.01))); // Sides margin
        txtNatural.setWrapStyleWord(true);
        pnlScroll.setViewportView(txtNatural); // Adding the panel

        gbc.anchor = GridBagConstraints.BASELINE_LEADING; // Place it in the middle as the
main component
        gbc.fill = GridBagConstraints.BOTH; // Makes the bag use all space to the sides
        gbc.weightx = 4; // Extra space distribution
        gbc.weighty = 48; // Extra space distribution
        gbc.gridwidth = 4; // Number of columns
        gbc.gridheight = 48; // Number of rows
        gbc.gridx = 0; // Column place
        gbc.gridy = 1; // Row place
        add(pnlScroll,gbc); // Add the component

        // Delete button
        JButton btnDelete = new JButton("Delete");
        gbc.weightx = 1; // Extra space distribution
        gbc.weighty = 1; // Extra space distribution
        gbc.gridwidth = 1; // Number of columns
        gbc.gridheight = 1; // Number of rows
        gbc.gridx = 0; // Column place
        gbc.gridy = 49; // Row place
        gbc.fill = GridBagConstraints.NONE; // Make the component fill to both sides
        gbc.anchor = GridBagConstraints.BELOW_BASELINE_LEADING; // Place the component to
the lower left corner of the main component
        add(btnDelete,gbc); // Add the component

        // Button create
        JButton btnCreate = new JButton("Create");
        gbc.weightx = 1; // Extra space distribution
        gbc.weighty = 1; // Extra space distribution
        gbc.gridwidth = 1; // Number of columns
        gbc.gridheight = 1; // Number of rows
        gbc.gridx = 3; // Columns place
        gbc.gridy = 49; // Rows place
        gbc.anchor = GridBagConstraints.BELOW_BASELINE_TRAILING; // Place the component to
the lower right corner of the main component
        add(btnCreate,gbc); // Add the component
```

```
        ////////////
        // Events //
        ////////////

        cbLanguages.addActionListener (new ActionListener () {
            /**
             * When a language is set the methods are modified
             * @param e Click the button action
             */
            public void actionPerformed(ActionEvent e) {
                if (eventsActive){
                    setLanguagesMethods(); // Reload the methods
                    setLanguagesMethodsOptions(); // Reload the options
                }

            }
        });

        cbLanguagesMethods.addActionListener(new ActionListener () {
            /**
             * When a method is set the options are modified
             * @param e Click the button action
             */
            public void actionPerformed(ActionEvent e) {
                if (eventsActive)
                    setLanguagesMethodsOptions(); // Reload the options
            }
        });

        cbCodes.addActionListener (new ActionListener () {
            /**
             * When a programming language is set the methods are modified
             * @param e Click the button action
             */
            public void actionPerformed(ActionEvent e) {
                if (eventsActive){
                    setCodesMethods(); // Reload the methods
                    setCodesMethodsOptions(); // Reload the options
                }
            }
        });

        cbCodesMethods.addActionListener(new ActionListener () {
            /**
             * When a method is set the options are modified
             * @param e Click the button action
             */
            public void actionPerformed(ActionEvent e) {
                if (eventsActive)
                    setCodesMethodsOptions(); // Reload the options
            }
        });

        btnDelete.addActionListener(new ActionListener() {
            /**
             * Remove all the text from the text box
             * @param e Click the button action
             */
            @Override
```

```
            public void actionPerformed(ActionEvent e) {
                if (eventsActive)
                    txtNatural.setText(null); // Deletes the text from the box
            }
        });

        btnCreate.addActionListener(new ActionListener(){
            /**
             * Start the process to create the code from the text
             * @param e Click the button action
             */
            @Override
            public void actionPerformed(ActionEvent e) {
                if (eventsActive){


                    // call to natural language processing
                    PnlDisplayInfo.pnlDisplayInfo.clear();
                    PnlDisplayInfo.pnlDisplayInfo.showMessage("Starting to process the
text");
                    ArrayList<String> optionsLanguage = new ArrayList<String>(); // Create
a variable that contains all the options
                    optionsLanguage.add((String)cbLanguages.getSelectedItem()); // Add the
selected language
                    optionsLanguage.add((String)cbLanguagesMethods.getSelectedItem()); //
Add the selected method
                    optionsLanguage.add((String)cbLanguagesMethodsOptions.getSelectedItem()
); // Add the selected option
                    String result = NaturalLanguage.processText(txtNatural.getText(),
optionsLanguage); // Call to process the text written by the user with the chosen options
                    PnlDisplayInfo.pnlDisplayInfo.showMessage("Starting to generate the
code");
                    if (result.contains("error")){

                    } else {
                        ArrayList<String> optionsCode = new ArrayList<String>(); // Create
a variable that contains all the options
                        optionsCode.add((String)cbCodes.getSelectedItem()); // Add the
selected language
                        optionsCode.add((String)cbCodesMethods.getSelectedItem()); // Add
the selected method
                        optionsCode.add((String)cbCodesMethodsOptions.getSelectedItem());
// Add the selected option
                        PnlDisplayCode.pnlDisplayCode.showCode(ProgrammingLanguage.language
Chooser(optionsCode,result,(String)cbLanguages.getSelectedItem()));
                    }
                    PnlDisplayInfo.pnlDisplayInfo.showMessage("process ended");
                }
            }
        });

    }
```

5.3.2 PROCESAMIENTO DEL LENGUAJE NATURAL

El módulo de procesamiento del lenguaje natural ha sido implementado con dos herramientas diferentes. Existe una clase que redirecciona a un método u otro en función de las opciones escogidas.

5.3.2.1 CoreNLP

Primero se procesa el texto y se crean los vectores con la información necesaria como la categoría gramatical o las relaciones entre las palabras. Esto se realiza para cada sentencia.

```
 // Options for the NLP
        Properties props = StringUtils.argsToProperties(new String[]{"-props",
"StanfordCoreNLP-spanish.properties"});
        StanfordCoreNLP pipeline = new StanfordCoreNLP(props);
        CoreDocument document = new CoreDocument(text);
        pipeline.annotate(document);

        // Support for variables
        String result = new String();
        ArrayList<String> dictionary = Files.getDictionary("Español");

        // Text processing
        for (int k = 0; k< document.sentences().size();k++){
            CoreSentence sentence = document.sentences().get(k);

            SemanticGraph dependencyParse = sentence.dependencyParse();
            String processed = dependencyParse.toCompactString(true);
            String[] chain = processed.split(">");
            ArrayList<String> middle = new ArrayList<String>();

            for (int i=0;i<chain.length;i++){
                if (chain[i].indexOf("[") == 0){
                    int pos = chain[i].indexOf("/");
                    middle.add(chain[i].substring(1, pos));
                } else {
                    String support = chain[i].substring(0,chain[i].indexOf("/"));
                    middle.set(middle.size()-1,middle.get(middle.size()-1)+";"+
support);
                }
            }

            ////////////////////////////////////////////////////////////////
            CoreDocument doc = pipeline.processToCoreDocument(text);
            // display tokens
            for (CoreLabel tok : doc.tokens()) {
                System.out.println(String.format("%s\t%s", tok.word(),
tok.lemma()));
```

```
            }
```

Después encontramos cada una de las funciones básicas de la programación que son comunes entre los distintos lenguajes de programación. Se da prioridad a la información más necesaria e importante y después a los detalles. Se utiliza la categoría gramatical para saber qué tipo de palabra se trata, ya sea un nombre, un numero o un verbo. Con las relaciones entre palabras obtenemos que nombres van con que acción. En este caso se produce una búsqueda de palabras en un diccionario.

- Creación de parámetros

Se hace una comprobación primero de los tipos de variable, seguido por el valor y finalizado por el posible nombre de la variable.

```
// Create parameter
            for (String i:middle){
                ArrayList<String> keyWords = new
ArrayList<String>(Arrays.asList(dictionary.get(0).split(",")));
                keyWords.addAll(Files.getSynonyms("Español","crear"));
                for (String kw:keyWords){
                    if (i.contains(kw)){
                        result = result + "createParam,";
                        for (String j:middle){
                            if (j.contains("boolean")){
                                result = result + "boolean,";
                                String[] keyWords3 =
dictionary.get(10).split(",");
                                for (String v:keyWords3){
                                    for (String m:middle){
                                        if (m.contains(v)){
                                            String[] parts = m.split(";");
                                            result = result + parts[parts.length-
1] + ",";
                                        }
                                    }
                                }
                                break;
                            } else if (j.contains("int")){
                                result = result + "int,";
                                String[] keyWords3 =
dictionary.get(10).split(",");
                                for (String v:keyWords3){
                                    for (String m:middle){
                                        if (m.contains(v)){
```

```
                                        for (String n:chain){
                                            if(n.contains("NUM")){
                                                if (n.contains("[")){
                                                    result = result  +
n.substring(1,n.indexOf("/")) + ",";
                                                } else {
                                                    result = result  +
n.substring(0,n.indexOf("/")) + ",";
                                                }
                                            }
                                        }
                                    }
                                }
                            }
                            break;
                        }else if (j.contains("float")){
                            result = result + "float,";
                            String[] keyWords3 =
dictionary.get(10).split(",");
                            for (String v:keyWords3){
                                for (String m:middle){
                                    if (m.contains(v)){
                                        for (String n:chain){
                                            if(n.contains("NUM")){
                                                if (n.contains("[")){
                                                    result = result  +
n.substring(1,n.indexOf("/")) + ",";
                                                } else {
                                                    result = result  +
n.substring(0,n.indexOf("/")) + ",";
                                                }
                                            }
                                        }
                                    }
                                }
                            }
                            break;
                        } else if (j.contains("double")){
                            result = result + "double,";
                            String[] keyWords3 =
dictionary.get(10).split(",");
                            for (String v:keyWords3){
                                for (String m:middle){
                                    if (m.contains(v)){
                                        for (String n:chain){
                                            if(n.contains("NUM")){
                                                if (n.contains("[")){
                                                    result = result  +
n.substring(1,n.indexOf("/")) + ",";
                                                } else {
                                                    result = result  +
n.substring(0,n.indexOf("/")) + ",";
                                                }
```

```
                                                }
                                            }
                                        }
                                    }
                                }
                                break;
                            } else if (j.contains("short")){
                                result = result + "short,";
                                String[] keyWords3 =
dictionary.get(10).split(",");
                                for (String v:keyWords3){
                                    for (String m:middle){
                                        if (m.contains(v)){
                                            for (String n:chain){
                                                if(n.contains("NUM")){
                                                    if (n.contains("[")){
                                                        result = result  +
n.substring(1,n.indexOf("/")) + ",";
                                                    } else {
                                                        result = result  +
n.substring(0,n.indexOf("/")) + ",";
                                                    }
                                                }
                                            }
                                        }
                                    }
                                }
                                break;
                            } else if (j.contains("long")){
                                result = result + "long,";
                                String[] keyWords3 =
dictionary.get(10).split(",");
                                for (String v:keyWords3){
                                    for (String m:middle){
                                        if (m.contains(v)){
                                            for (String n:chain){
                                                if(n.contains("NUM")){
                                                    if (n.contains("[")){
                                                        result = result  +
n.substring(1,n.indexOf("/")) + ",";
                                                    } else {
                                                        result = result  +
n.substring(0,n.indexOf("/")) + ",";
                                                    }
                                                }
                                            }
                                        }
                                    }
                                }
                                break;
                            } else if (j.contains("byte")){
                                result = result +"byte,";
```

```
                                String[] keyWords3 =
dictionary.get(10).split(",");
                                for (String v:keyWords3){
                                    for (String m:middle){
                                        if (m.contains(v)){
                                            for (String n:chain){
                                                if(n.contains("NUM")){
                                                    if (n.contains("[")){
                                                        result = result  +
n.substring(1,n.indexOf("/")) + ",";
                                                    } else {
                                                        result = result  +
n.substring(0,n.indexOf("/")) + ",";
                                                    }
                                                }
                                            }
                                        }
                                    }
                                }
                                break;
                            } else if (j.contains("char")){
                                result = result + "char,";
                                String[] keyWords3 =
dictionary.get(10).split(",");
                                for (String v:keyWords3){
                                    for (String m:middle){
                                        if (m.contains(v)){
                                            String[] parts = m.split(";");
                                            result = result + parts[parts.length-
1] + ",";
                                        }
                                    }
                                }
                                break;
                            } else if (j.contains("String")){
                                result = result +  "String,";
                                String[] keyWords3 =
dictionary.get(10).split(",");
                                for (String v:keyWords3){
                                    for (String m:middle){
                                        if (m.contains(v)){
                                            String[] parts = m.split(";");
                                            result = result + parts[parts.length-
1] + ",";
                                        }
                                    }
                                }
                                break;
                            }
                        }

                        if (result.split(",").length <2){
                            result = result + ",";
```

```
                            String[] keyWords3 = dictionary.get(10).split(",");
                            int cont = 0;
                            for (String v:keyWords3){
                                for (String m:middle){
                                    if (m.contains(v)){
                                        for (String n:chain){
                                            if(n.contains("NUM")){
                                                if (n.contains("[")){
                                                    result = result  +
n.substring(1,n.indexOf("/")) + ",";
                                                } else {
                                                    result = result  +
n.substring(0,n.indexOf("/")) + ",";
                                                }
                                            } else if (n.contains("NOUN")){
                                                if (cont >=1){
                                                    if (n.contains("[")){
                                                        result = result  +
n.substring(1,n.indexOf("/")) + ",";
                                                    } else {
                                                        result = result  +
n.substring(0,n.indexOf("/")) + ",";
                                                    }
                                                }
                                                cont++;
                                            }
                                        }
                                    }
                                }
                            }

                            for (String j:chain){
                                if(j.contains("PROPN")){
                                    if (j.contains("[")){
                                        if
(!j.substring(1,j.indexOf("/")).equals("String")){
                                            result = result  +
j.substring(1,j.indexOf("/")) + ",";
                                            break;
                                        }

                                    } else {
                                        if
(!j.substring(1,j.indexOf("/")).equals("String")){
                                            result = result  +
j.substring(0,j.indexOf("/")) + ",";
                                            break;
                                        }

                                    }
```

```
                            }
                        }

                        break;
                    }

                }
```

El código referente a las distintas operaciones aritméticas, primero se buscan valores numéricos y después nombres.

- Sumar:

```
// Add
                keyWords = new
ArrayList<String>(Arrays.asList(dictionary.get(1).split(",")));
                keyWords.addAll(Files.getSynonyms("Español","sumar"));
                int contName = 0;
                for (String kw: keyWords){
                    if(i.contains(kw)){
                        result = result + "add,";
                        int cont = 0;
                        for (String j:chain){
                            if(j.contains("NUM")){
                                if (j.contains("[")){
                                    result = result  +
j.substring(1,j.indexOf("/")) + ",";
                                    cont++;
                                } else {
                                    result = result  +
j.substring(0,j.indexOf("/")) + ",";
                                    cont++;
                                }
                                break;
                            }
                        }
                        for (String j:chain){
                            if (cont == 0){
                                if(j.contains("PROPN")){
                                    if (j.contains("[")){
                                        result = result  +
j.substring(1,j.indexOf("/")) + ",";
                                        contName++;
                                    } else {
                                        result = result  +
j.substring(0,j.indexOf("/")) + ",";
                                        contName++;
                                    }
                                    break;
```

```
                                }
                            }

                        }
                        int repName = 0;
                        for (String j:chain){
                            if(j.contains("PROPN")){
                                if (contName <= repName){
                                    if (j.contains("[")){
                                        result = result  +
j.substring(1,j.indexOf("/")) + ",";
                                        break;
                                    } else {
                                        result = result  +
j.substring(0,j.indexOf("/")) + ",";
                                        break;
                                    }
                                }
                                repName++;
                            }
                        }

                    }
                }
```

- Restar:

```
// Subtract
                keyWords = new
ArrayList<String>(Arrays.asList(dictionary.get(2).split(",")));
                keyWords.addAll(Files.getSynonyms("Español","restar"));
                for (String kw:keyWords){
                    if (i.contains(kw)){
                        result = result +  "subtract,";
                        int cont = 0;
                        for (String j:chain){
                            if(j.contains("NUM")){
                                if (j.contains("[")){
                                    result = result  +
j.substring(1,j.indexOf("/")) + ",";
                                    cont++;
                                } else {
                                    result = result  +
j.substring(0,j.indexOf("/")) + ",";
                                    cont++;
                                }
                                break;
                            }
                        }
                        for (String j:chain){
```

```
                            if (cont == 0){
                                if(j.contains("PROPN")){
                                    if (j.contains("[")){
                                        result = result  +
j.substring(1,j.indexOf("/")) + ",";
                                        contName++;
                                    } else {
                                        result = result  +
j.substring(0,j.indexOf("/")) + ",";
                                        contName++;
                                    }
                                    break;
                                }
                            }

                        }
                        int repName = 0;
                        for (String j:chain){
                            if(j.contains("PROPN")){
                                if (contName <= repName){
                                    if (j.contains("[")){
                                        result = result  +
j.substring(1,j.indexOf("/")) + ",";
                                        break;
                                    } else {
                                        result = result  +
j.substring(0,j.indexOf("/")) + ",";
                                        break;
                                    }
                                }
                                repName++;
                            }
                        }
                    }
                }
```

- Multiplicar:

```
// Multiply
                keyWords = new
ArrayList<String>(Arrays.asList(dictionary.get(3).split(",")));
                keyWords.addAll(Files.getSynonyms("Español","multiplicar"));
                for (String kw:keyWords){
                    if (i.contains(kw)){
                        result = result + "multiply,";
                        int cont = 0;
                        for (String j:chain){
                            if(j.contains("NUM")){
                                if (j.contains("[")){
```

```
                                    result = result  +
j.substring(1,j.indexOf("/")) + ",";
                                    cont++;
                                } else {
                                    result = result  +
j.substring(0,j.indexOf("/")) + ",";
                                    cont++;
                                }
                                break;
                            }
                        }
                        for (String j:chain){
                            if (cont == 0){
                                if(j.contains("PROPN")){
                                    if (j.contains("[")){
                                        result = result  +
j.substring(1,j.indexOf("/")) + ",";
                                        contName++;
                                    } else {
                                        result = result  +
j.substring(0,j.indexOf("/")) + ",";
                                        contName++;
                                    }
                                    break;
                                }
                            }

                        }
                        int repName = 0;
                        for (String j:chain){
                            if(j.contains("PROPN")){
                                if (contName <= repName){
                                    if (j.contains("[")){
                                        result = result  +
j.substring(1,j.indexOf("/")) + ",";
                                        break;
                                    } else {
                                        result = result  +
j.substring(0,j.indexOf("/")) + ",";
                                        break;
                                    }
                                }
                                repName++;
                            }
                        }
                    }
                }
```

- Dividir:

```
// Split
                keyWords = new
ArrayList<String>(Arrays.asList(dictionary.get(4).split(",")));
                keyWords.addAll(Files.getSynonyms("Español","dividir"));
                for (String kw: keyWords){
                    if (i.contains(kw)){
                        result = result + "split,";
                        int cont = 0;
                        for (String j:chain){
                            if(j.contains("NUM")){
                                if (j.contains("[")){
                                    result = result  +
j.substring(1,j.indexOf("/")) + ",";
                                    cont++;
                                } else {
                                    result = result  +
j.substring(0,j.indexOf("/")) + ",";
                                    cont++;
                                }
                                break;
                            }
                        }
                        for (String j:chain){
                            if (cont == 0){
                                if(j.contains("PROPN")){
                                    if (j.contains("[")){
                                        result = result  +
j.substring(1,j.indexOf("/")) + ",";
                                        contName++;
                                    } else {
                                        result = result  +
j.substring(0,j.indexOf("/")) + ",";
                                        contName++;
                                    }
                                    break;
                                }
                            }

                        }
                        int repName = 0;
                        for (String j:chain){
                            if(j.contains("PROPN")){
                                if (contName <= repName){
                                    if (j.contains("[")){
                                        result = result  +
j.substring(1,j.indexOf("/")) + ",";
                                        break;
                                    } else {
                                        result = result  +
j.substring(0,j.indexOf("/")) + ",";
                                        break;
                                    }
                                }
```

```
                                repName++;
                            }
                        }
                    }
                }
```

- Mostrar por pantalla:

En este caso el texto necesita ir entre comillas para ser interpretado de manera correcta.

```
// Show on screen
                keyWords = new
ArrayList<String>(Arrays.asList(dictionary.get(5).split(",")));
                keyWords.addAll(Files.getSynonyms("Español","mostrar"));
                for (String kw:keyWords){
                    if (i.contains(kw)){
                        result = result + "show,";
                        if (text.contains(String.valueOf('"'))){
                            result = result +
text.substring(text.indexOf(String.valueOf('"')),
text.lastIndexOf(String.valueOf('"'))+1);
                        }
                    }
                }
```

- Bucles *FOR* y *WHILE:*

La diferencia entre usar un bucle u otro se basa en la cantidad de parámetros especificados por el usuario. Esto se sabe en función del número de parámetros entre palabras clave que definen una función en el vector que sirve de enlace entre los módulos.

```
// Repetitions
                keyWords = new
ArrayList<String>(Arrays.asList(dictionary.get(6).split(",")));
                keyWords.addAll(Files.getSynonyms("Español","bucle"));
                for (String kw:keyWords){
                    // While
                    if (i.contains(kw)){
                        if (middle.size() <= 3){
                            result = result + "while,";
                            for (String j:chain){
                                if(j.contains("NUM")){
                                    if (j.contains("[")){
                                        result = result  +
j.substring(1,j.indexOf("/")) + ",";
```

```
                                        break;
                                    } else {
                                        result = result  +
j.substring(0,j.indexOf("/"))+ ",";
                                        break;
                                    }

                                }
                            }
                        // For
                        } else {
                            result = result + "for,";
                            int cont = 0;
                            ArrayList<Integer> values = new ArrayList<Integer>();
                            for (String j:chain){
                                if(j.contains("NUM")){
                                    if (j.contains("[")){
                                        result = result  +
j.substring(1,j.indexOf("/")) + ",";
                                        values.add(Integer.valueOf(j.substring(1,
j.indexOf("/"))));
                                        cont++;

                                    } else {
                                        result = result  +
j.substring(0,j.indexOf("/")) + ",";
                                        values.add(Integer.valueOf(j.substring(0,
j.indexOf("/"))));

                                        cont++;

                                    }
                                }

                                if (cont == 2){
                                    if (values.get(0) < values.get(1)){
                                        result = result + "<,";
                                        cont++;
                                    } else if (values.get(0) > values.get(1)){
                                        result = result + ">,";
                                        cont++;
                                    }
                                }

                            }
                        }
                    }
                }
```

- IF ELSE:

```
// IfElse
                keyWords = new
ArrayList<String>(Arrays.asList(dictionary.get(7).split(",")));
                keyWords.addAll(Files.getSynonyms("Español","comprobar"));
                for (String kw:keyWords){
                    if (i.contains(kw)){
                        result = result + "ifElse,";
                        for (String j:chain){
                            if(j.contains("PROPN")){
                                if (j.contains("[")){
                                    result = result  +
j.substring(1,j.indexOf("/")) + ",";
                                } else {
                                    result = result  +
j.substring(0,j.indexOf("/")) + ",";
                                }
                                break;
                            }
                        }
                        if (text.contains("menor que")){
                            result = result + "<,";
                        } else if (text.contains("menor o igual")){
                            result = result + "<=,";
                        } else if(text.contains("mayor que")){
                            result = result + ">,";
                        } else if (text.contains("mayor o igual")){
                            result = result + ">=,";
                        } else if (text.contains("igual")){
                            result = result + "==,";
                        }
                        int cont = 0;
                        for (String j:chain){
                            if(j.contains("PROPN")){
                                if (cont == 1){
                                    if (j.contains("[")){
                                        result = result  +
j.substring(1,j.indexOf("/")) + ",";
                                    } else {
                                        result = result  +
j.substring(0,j.indexOf("/")) + ",";
                                    }
                                    break;
                                }
                                cont ++;
                            } else if (j.contains("NUM")){
                                if (j.contains("[")){
                                    result = result  +
j.substring(1,j.indexOf("/")) + ",";
                                } else {
                                    result = result  +
j.substring(0,j.indexOf("/")) + ",";
                                }
                                break;
```

```
                        }
                    }
                }
            }
```

- SWITCH CASE:

Este caso es particular pues no todos los lenguajes de programación disponen de esta sentencia, ya que se puede imitar con clausular IF ELSE anidadas[11].

```
// Switch Case
                keyWords = new
ArrayList<String>(Arrays.asList(dictionary.get(8).split(",")));
                keyWords.addAll(Files.getSynonyms("Español","opción"));
                for (String kw:keyWords){
                    if (i.contains(kw)){
                        result = result + "switchCase,";
                        for (String j:chain){
                            if(j.contains("NUM")){
                                result = result  + j.substring(0,j.indexOf("/"))
+ ",";
                                break;
                            }
                        }
                    }
                }

            }
```

5.3.2.2 SparkNLP

En este caso cuando se procesa el texto, no solo se crea los vectores con las categorías gramaticales y las relaciones entre las palabras, sino que también se crea uno con las versiones de diccionario de las palabras.

```
String result = new String();

        SparkSession spark = SparkSession
        .builder()
        .appName("Programable")
        .config("spark.master", "local")
        .getOrCreate();
```

[11] En las clausulas IF ELSE cuando un else se continua con otro if para discernir entre más opciones.

```
PretrainedPipeline pipeline =
PretrainedPipeline.fromDisk("src/main/resources/data/Models/SparkNLP/explain_docu
ment_md_es_3.0.0_3.0_1616431976931");
ArrayList<String> dictionary = Files.getDictionary("Español");
        Map<String, Seq<String>> annotation = pipeline.annotate(text);
        System.out.println(pipeline.annotate(text));
        ArrayList<String> lemma = new
ArrayList<String>(Arrays.asList(annotation.get("lemma").get().toString().split(",
")));
        ArrayList<String> pos = new
ArrayList<String>(Arrays.asList(annotation.get("pos").get().toString().split(",")
));
        ArrayList<String> token = new
ArrayList<String>(Arrays.asList(annotation.get("token").get().toString().split(",
")));
        ArrayList<String> sentences = new
ArrayList<String>(Arrays.asList(annotation.get("sentence").get().toString().split
(",")));
        new ArrayList<String>(Arrays.asList(dictionary.get(0).split(",")));
        lemma.set(0, lemma.get(0).substring(5));
        lemma.set(lemma.size()-1, lemma.get(lemma.size()-
1).substring(0,  lemma.get(lemma.size()-1).length()-1));
        pos.set(0, pos.get(0).substring(12));
        pos.set(pos.size()-1, pos.get(pos.size()-
1).substring(0,  pos.get(pos.size()-1).length()-1));
        token.set(0, token.get(0).substring(5));
        token.set(token.size()-1, token.get(token.size()-
1).substring(0,  token.get(token.size()-1).length()-1));
        sentences.set(0,sentences.get(0).substring(5));
        sentences.set(sentences.size()-1, sentences.get(sentences.size()-
1).substring(0,  sentences.get(sentences.size()-1).length()-1));
        int sentencesNum = sentences.size();

        for (int i= 0; i<lemma.size();i++){
            if (lemma.get(i).contains(".")){
                lemma.set(i,lemma.get(i).substring(0,lemma.get(i).length()-1));
            }
        }

        for (int i= 0; i<pos.size();i++){
            if (pos.get(i).contains(".")){
                pos.set(i,pos.get(i).substring(0,pos.get(i).length()-1));
            }
        }
        for (int i= 0; i<token.size();i++){
            if (token.get(i).contains(".")){
                token.set(i,token.get(i).substring(0,token.get(i).length()-1));
            }
        }
```

```
        for (int k= 0;k < sentencesNum ; k++){
            if(sentences.get(k).charAt(0) == ' '){
                sentences.set(k, sentences.get(k).substring(1));
            }
            int sentencesLen = sentences.get(k).split(" ").length;
```

Como en el caso anterior, también encontramos las funciones básicas de la programación. Se realizan los mismos pasos que con el otro sistema. Sin embargo, se requiere de realizar ciertas modificaciones al código anterior debido a que los vectores creados no son idénticos.

- Creación de variables:

```
// Create parameter
                ArrayList<String> keyWords = new
ArrayList<String>(Arrays.asList(dictionary.get(0).split(",")));
                keyWords.addAll(Files.getSynonyms("Español","crear"));
                for (String kw:keyWords){
                    if (i.contains(kw)){
                        result = result + "createParam,";
                        for (String j:token.subList(0, sentencesLen)){
                            if (j.contains("boolean")){
                                result = result + "boolean,";
                                String[] keyWords3 =
dictionary.get(10).split(",");
                                for (String v:keyWords3){
                                    for (String m:token.subList(0,
sentencesLen)){
                                        if (m.contains(v)){
                                            if
(sentences.get(k).toLowerCase().contains("false")){
                                                result = result + "false" + ",";
                                            } else if
(sentences.get(k).toLowerCase().contains("true")){
                                                result = result + "true" + ",";
                                            }
                                        }
                                    }
                                }
                                break;
                            } else if (j.contains("int")){
                                result = result + "int,";
                                String[] keyWords3 =
dictionary.get(10).split(",");
                                for (String v:keyWords3){
                                    for (String m:token.subList(0,
sentencesLen)){
                                        if (m.contains(v)){
```

```
                                        result = result +
token.get(pos.indexOf(" NUM")) + ",";
                                    }
                                }
                            }
                            break;
                        } else if (j.contains("float")){
                            result = result + "float,";
                            String[] keyWords3 =
dictionary.get(10).split(",");
                            for (String v:keyWords3){
                                for (String m:token.subList(0,
sentencesLen)){
                                    if (m.contains(v)){
                                        result = result +
token.get(pos.indexOf(" NUM")) + ",";
                                    }
                                }
                            }
                            break;
                        } else if (j.contains("double")){
                            result = result + "double,";
                            String[] keyWords3 =
dictionary.get(10).split(",");
                            for (String v:keyWords3){
                                for (String m:token.subList(0,
sentencesLen)){
                                    if (m.contains(v)){
                                        result = result +
token.get(pos.indexOf(" NUM")) + ",";
                                    }
                                }
                            }
                            break;
                        } else if (j.contains("short")){
                            result = result + "short,";
                            String[] keyWords3 =
dictionary.get(10).split(",");
                            for (String v:keyWords3){
                                for (String m:token.subList(0,
sentencesLen)){
                                    if (m.contains(v)){
                                        result = result +
token.get(pos.indexOf(" NUM")) + ",";
                                    }
                                }
                            }
                            break;
                        } else if (j.contains("long")){
                            result = result + "long,";
                            String[] keyWords3 =
dictionary.get(10).split(",");
                            for (String v:keyWords3){
```

```
                                    for (String m:token.subList(0,
sentencesLen)){
                                        if (m.contains(v)){
                                            result = result +
token.get(pos.indexOf(" NUM")) + ",";
                                        }
                                    }
                                }
                                break;
                            } else if (j.contains("byte")){
                                result = result +"byte,";
                                String[] keyWords3 =
dictionary.get(10).split(",");
                                for (String v:keyWords3){
                                    for (String m:token.subList(0,
sentencesLen)){
                                        if (m.contains(v)){
                                            result = result +
token.get(pos.indexOf(" NUM")) + ",";
                                        }
                                    }
                                }
                                break;
                            } else if (j.contains("char")){
                                result = result + "char,";
                                String[] keyWords3 =
dictionary.get(10).split(",");
                                for (String v:keyWords3){
                                    for (String m:token.subList(0,
sentencesLen)){
                                        if (m.contains(v)){
                                            result = result +
token.get(token.indexOf(m)+1).substring(1) + ",";
                                        }
                                    }
                                }
                                break;
                            } else if (j.contains("String")){
                                result = result +  "String,";
                                String[] keyWords3 =
dictionary.get(10).split(",");
                                for (String v:keyWords3){
                                    for (String m:token.subList(0,
sentencesLen)){
                                        if (m.contains(v)){
                                            result = result +
token.get(token.indexOf(m)+1).substring(1) + ",";
                                        }
                                    }
                                }
                                break;
                            }
                        }
```

```
                        if (result.split(",").length <2){
                            result = result + ",";
                            String[] keyWords3 = dictionary.get(10).split(",");
                            for (String v:keyWords3){
                                for (String m:token.subList(0, sentencesLen)){
                                    if (m.contains(v)){
                                        if (pos.indexOf(" NUM") != -1){
                                            result = result +
token.get(pos.indexOf(" NUM")) + ",";
                                        } else if (pos.indexOf(" PROPN") != -1){
                                            result = result +
token.get(pos.indexOf(" PROPN")) + ",";
                                        }
                                    }
                                }
                            }

                        }

                        String[] keyWords3 = dictionary.get(9).split(",");
                        for (String v:keyWords3){
                            for (String m:token.subList(0, sentencesLen)){
                                if (m.contains(v)){
                                    result = result +
token.get(token.indexOf(m)+1).substring(1) + ",";
                                }
                            }
                        }

                    }
                }
```

Continuamos con las operaciones aritméticas.

- Suma:

```
// Add
                keyWords = new
ArrayList<String>(Arrays.asList(dictionary.get(1).split(",")));
                keyWords.addAll(Files.getSynonyms("Español","sumar"));
                for (String kw: keyWords){
                    if(i.contains(kw)){
                        result = result + "add,";

                        for (String m:pos.subList(0, sentencesLen)){
                            if (m.equals(" NUM")){
                                result = result + token.get(pos.indexOf(" NUM"))
+ ",";
                            }
```

```
                        }

                        if (result.split(",").length <2){
                            String[] keyWords3 = dictionary.get(10).split(",");
                            for (String v:keyWords3){
                                for (String m:token.subList(0, sentencesLen)){
                                    if (m.contains(v)){
                                        result = result +
token.get(token.indexOf(m)+1).substring(1) + ",";
                                    }
                                }
                            }
                        }

                        String[] keyWords3 = dictionary.get(9).split(",");
                        for (String v:keyWords3){
                            for (String m:token.subList(0, sentencesLen)){
                                if (m.contains(v)){
                                    result = result +
token.get(token.indexOf(m)+1).substring(1) + ",";
                                }
                            }
                        }

                    }
                }
```

- Resta:

```
// Subtract
                keyWords = new
ArrayList<String>(Arrays.asList(dictionary.get(2).split(",")));
                keyWords.addAll(Files.getSynonyms("Español","restar"));
                for (String kw: keyWords){
                    if(i.contains(kw)){
                        result = result +  "subtract,";

                        for (String m:pos.subList(0, sentencesLen)){
                            if (m.equals(" NUM")){
                                result = result + token.get(pos.indexOf(" NUM"))
+ ",";
                            }
                        }

                        if (result.split(",").length <2){
                            String[] keyWords3 = dictionary.get(10).split(",");
                            for (String v:keyWords3){
                                for (String m:token.subList(0, sentencesLen)){
                                    if (m.contains(v)){
```

```
                                        result = result +
token.get(token.indexOf(m)+1).substring(1) + ",";
                                    }
                                }
                            }
                        }

                        String[] keyWords3 = dictionary.get(9).split(",");
                        for (String v:keyWords3){
                            for (String m:token.subList(0, sentencesLen)){
                                if (m.contains(v)){
                                    result = result +
token.get(token.indexOf(m)+1).substring(1) + ",";
                                }
                            }
                        }

                    }
                }
```

- Multiplicar:

```
// Multiply
                keyWords = new
ArrayList<String>(Arrays.asList(dictionary.get(3).split(",")));
                keyWords.addAll(Files.getSynonyms("Español","multiplicar"));
                for (String kw: keyWords){
                    if(i.contains(kw)){
                        result = result +  "multiply,";

                        for (String m:pos.subList(0, sentencesLen)){
                            if (m.equals(" NUM")){
                                result = result + token.get(pos.indexOf(" NUM"))
+ ",";
                            }
                        }

                        if (result.split(",").length <2){
                            String[] keyWords3 = dictionary.get(10).split(",");
                            for (String v:keyWords3){
                                for (String m:token.subList(0, sentencesLen)){
                                    if (m.contains(v)){
                                        result = result +
token.get(token.indexOf(m)+1).substring(1) + ",";
                                    }
                                }
                            }
                        }

                        String[] keyWords3 = dictionary.get(9).split(",");
```

```
                        for (String v:keyWords3){
                            for (String m:token.subList(0, sentencesLen)){
                                if (m.contains(v)){
                                    result = result +
token.get(token.indexOf(m)+1).substring(1) + ",";
                                }
                            }
                        }

                    }
                }
```

- Dividir:

```
// Split
                keyWords = new
ArrayList<String>(Arrays.asList(dictionary.get(4).split(",")));
                keyWords.addAll(Files.getSynonyms("Español","dividir"));
                for (String kw: keyWords){
                    if(i.contains(kw)){
                        result = result +  "split,";

                        for (String m:pos.subList(0, sentencesLen)){
                            if (m.equals(" NUM")){
                                result = result + token.get(pos.indexOf(" NUM"))
+ ",";
                            }
                        }

                        if (result.split(",").length <2){
                            String[] keyWords3 = dictionary.get(10).split(",");
                            for (String v:keyWords3){
                                for (String m:token.subList(0, sentencesLen)){
                                    if (m.contains(v)){
                                        result = result +
token.get(token.indexOf(m)+1).substring(1) + ",";
                                    }
                                }
                            }
                        }

                        String[] keyWords3 = dictionary.get(9).split(",");
                        for (String v:keyWords3){
                            for (String m:token.subList(0, sentencesLen)){
                                if (m.contains(v)){
                                    result = result +
token.get(token.indexOf(m)+1).substring(1) + ",";
                                }
                            }
                        }
```

```
                    }
                }
```

- Mostar por pantalla:

En este caso requiere que el texto que se desea mostrar se encuentre entre comillas.

```
// Show on screen
                keyWords = new
ArrayList<String>(Arrays.asList(dictionary.get(5).split(",")));
                keyWords.addAll(Files.getSynonyms("Español","mostrar"));
                for (String kw:keyWords){
                    if (i.contains(kw)){
                        result = result + "show,";
                        if (text.contains(String.valueOf('"'))){
                            result = result +
text.substring(text.indexOf(String.valueOf('"')),
text.lastIndexOf(String.valueOf('"'))+1);
                        }
                    }
                }
```

- Bucles *FOR* y *WHILE:*

```
// Repetitions
                keyWords = new
ArrayList<String>(Arrays.asList(dictionary.get(6).split(",")));
                keyWords.addAll(Files.getSynonyms("Español","bucle"));
                for (String kw:keyWords){
                    // While
                    if (i.contains(kw)){
                        if (sentencesLen <= 6){
                            result = result + "while,";
                            for (String j:pos.subList(0, sentencesLen)){
                                if(j.contains("NUM")){
                                    result = result + token.get(pos.indexOf("
NUM")) + ",";
                                }
                            }
                        // For
                        } else {
                            result = result + "for,";
                            System.out.println("estamos en un bucle for");
                            int cont = 0;
                            ArrayList<Integer> values = new ArrayList<Integer>();
                            for (String j:pos.subList(0, sentencesLen+1)){
```

```
                                if(j.contains("NUM")){
                                    result = result + token.get(pos.indexOf("
NUM")) + ",";
                                    values.add(Integer.valueOf(token.get(pos.inde
xOf(" NUM")).substring(1)));
                                    pos.set(pos.indexOf(" NUM"), "num");
                                    System.out.println(cont);
                                    cont++;
                                }
                                if (cont == 2){
                                    if (values.get(0) < values.get(1)){
                                        result = result + "<,";
                                        cont++;
                                    } else if (values.get(0) > values.get(1)){
                                        result = result + ">,";
                                        cont++;
                                    }
                                }

                            }
                        }
                    }
                }
```

- IF ELSE:

```
// IfElse
                keyWords = new
ArrayList<String>(Arrays.asList(dictionary.get(7).split(",")));
                keyWords.addAll(Files.getSynonyms("Español","comprobar"));
                for (String kw:keyWords){

                    if (i.contains(kw)){
                        result = result + "ifElse,";
                        String[] keyWords3 = dictionary.get(9).split(",");
                        for (String v:keyWords3){
                            for (String m:token.subList(0, sentencesLen)){
                                if (m.contains(v)){
                                    result = result +
token.get(token.indexOf(m)+1).substring(1) + ",";
                                    break;
                                }
                            }
                        }
                        if (text.contains("menor que")){
                            result = result + "<,";
                        } else if (text.contains("menor o igual")){
                            result = result + "<=,";
                        } else if(text.contains("mayor que")){
                            result = result + ">,";
```

```
                        } else if (text.contains("mayor o igual")){
                            result = result + ">=,";
                        } else if (text.contains("igual")){
                            result = result + "==,";
                        }
                        int cont = 0;
                        for (String j:pos.subList(0, sentencesLen)){
                            if(j.contains(" PROPN")){
                                for (String v:keyWords3){
                                    for (String m:token.subList(0,
sentencesLen)){
                                        if (m.contains(v)){
                                            if (cont == 1){
                                                result = result +
token.get(token.lastIndexOf(m)+1).substring(1) + ",";
                                                cont++;
                                                break;
                                            } else{
                                                cont++;
                                            }
                                        }
                                    }
                                }
                            } else if (j.contains(" NUM")){
                                result = result + token.get(pos.indexOf(" NUM"))
+ ",";
                                break;
                            }
                        }
                    }
                }
```

- SWITCH CASE:

```
// Switch Case
                keyWords = new
ArrayList<String>(Arrays.asList(dictionary.get(8).split(",")));
                keyWords.addAll(Files.getSynonyms("Español","opción"));
                for (String kw:keyWords){
                    if (i.contains(kw)){
                        result = result + "switchCase,";
                        for (String j:pos.subList(0, sentencesLen)){
                            if (j.contains(" NUM")){
                                result = result + token.get(pos.indexOf("
NUM")).substring(1) + ",";
                                break;
                            }
                        }
                    }
                }
```

5.3.3 MÓDULO DE GENERACIÓN DE CÓDIGO

La generación de código se ha llevado a cabo en dos lenguajes de programación diferentes. Ambos lenguajes parten del mismo vector con las palabras clave, obteniendo como resultado el código referente a susodicho lenguaje.

Entre los parámetros comunes encontramos el número de iteraciones realizadas para saber cuánto espaciado debemos seguir para el código. Esto tiene mayor o menor importancia en función del lenguaje escogido, si bien siempre es recomendable que el código quede legible visualmente.

5.3.3.1 Java

Las funciones básicas de programación cuentan con múltiples formas de implementación en función de los parámetros especificados.

- Creación de variables:

En este caso es necesaria la especificación del tipo de variable.

```
// create specific type of variable
    public static String[] createVariable(int iteration,String type,String
language){
        String info = null;
        if (iteration < 1){
            info = Files.getInfo(language,"Java","createParam");
        }
        String code = null;
        code = new String(new char[4 * iteration]).replace("\0", " ");
        if (type.equals( "boolean")){
            code = code + "boolean variable = null;";
        } else if (type.equals("int")){
            code = code + "int variable = null;";
        }else if (type.equals("String")){
            code = code +  "String variable = null;";
        } else if (type.equals("float")){
            code = code + "float variable = null;";
        } else if (type.equals("double")){
            code = code + "double variable = null;";
        } else if (type.equals("short")){
            code = code + "short variable = null;";
```

```
        } else if (type.equals("long")){
            code = code + "long variable = null;";
        } else if (type.equals("byte")){
            code = code +"byte variable = null;";
        } else if (type.equals("char")){
            code = code + "char variable = null;";
        }
        String[] result= {info,code};
        return result;
    }

    public static String[] createVariable(int iteration,String type, String
value,String language){
        String info = null;
        if (iteration < 1){
            info = Files.getInfo(language,"Java","createParam");
        }
        String code = null;
        code = new String(new char[4 * iteration]).replace("\0", " ");
        if (type.equals("boolean")){
            code = code + "boolean variable = " + value +  ";";
        } else if (type.equals("int")){
            code = code + "int variable = " + value +  ";";
        }else if (type.equals("String")){
            char comillas = '"';
            code = code +  "String variable =  " + comillas + value + comillas
+   ";";
        } else if (type.equals("float")){
            code = code + "float variable = " + value +  "f;";
        } else if (type.equals("double")){
            code = code + "double variable = " + value +  ";";
        } else if (type.equals("short")){
            code = code + "short variable = " + value +  ";";
        } else if (type.equals("long")){
            code = code + "long variable = " + value +  ";";
        } else if (type.equals("byte")){
            code = code +"byte variable = " + value +  ";";
        } else if (type.equals("char")){
            code = code + "char variable = " + "'" + value + "'" +  ";";
        }
        String result[] = {info, code};
        return result;
    }
    public static String[] createVariable(int iteration,String type, String
value, String name,String language){
        String info = null;
        if (iteration < 1){
            info =  Files.getInfo(language,"Java","createParam");
        }
        String code = null;
        code = new String(new char[4 * iteration]).replace("\0", " ");
        if (type.equals("boolean")){
            code = code + "boolean " + name + "= " + value +  ";";
```

```
        } else if (type.equals("int")){
            code = code + "int "+ name + "= " + value +  ";";
        }else if (type.equals("String")){
            char comillas = '"';
            code = code +  "String " + name + " = " + comillas + value + comillas
+   ";";
        } else if (type.equals("float")){
            code = code + "float "+ name + " = " + value +  "f;";
        } else if (type.equals("double")){
            code = code + "double "+ name + " = " + value +  ";";
        } else if (type.equals("short")){
            code = code + "short "+ name + " = " + value +  ";";
        } else if (type.equals("long")){
            code = code + "long "+ name + " = " + value +  ";";
        } else if (type.equals("byte")){
            code = code +"byte "+ name + " = " + value +  ";";
        } else if (type.equals("char")){
            code = code + "char " + name +  " = " + "'" +value+ "'" +  ";";
        }

        String result[] = {info, code};
        return result;
    }
```

- Suma:

```
// Add
    public static String[] add(int iteration,String value,String language){
        String info = Files.getInfo(language,"Java","add");
        String code = new String(new char[4 * iteration]).replace("\0", " ");
        code = code + "variable = variable + " + value + ";";
        String result[] = {info,code};
        return result;
    }

    public static String[] add(int iteration,String value, String name,String
language){
        String info = Files.getInfo(language,"Java","add");
        String code = new String(new char[4 * iteration]).replace("\0", " ");
        code = code + name + " = " + name + " + " + value + ";";
        String result[] = {info,code};
        return result;
    }
```

- Resta:

```
// Subtract
    public static String[] subtract(int iteration, String value,String language){
```

```
        String info = Files.getInfo(language,"Java","subtract");
        String code = new String(new char[4 * iteration]).replace("\0", " ");
        code = code + "variable = variable - " + value + ";";
        String result[] = {info,code};
        return result;
    }

    public static String[] subtract(int iteration, String value, String
name,String language){
        String info = Files.getInfo(language,"Java","subtract");
        String code = new String(new char[4 * iteration]).replace("\0", " ");
        code = code + name + " = " + name + " - " + value + ";";
        String result[] = {info,code};
        return result;
    }
```

- Multiplicar:

```
// Multiply
    public static String[] multiply(int iteration, String value,String language){
        String info = Files.getInfo(language,"Java","multiply");
        String code = new String(new char[4 * iteration]).replace("\0", " ");
        code = code + "variable = variable * " + value + ";";
        String result[] = {info,code};
        return result;
    }

    public static String[] multiply(int iteration, String value, String
name,String language){
        String info = Files.getInfo(language,"Java","multiply");
        String code = new String(new char[4 * iteration]).replace("\0", " ");
        code = code + name + " = " + name + " * " + value + ";";
        String result[] = {info,code};
        return result;
    }
```

- Dividir:

```
// Split
    public static String[] split(int iteration, String value,String language){
        String info = Files.getInfo(language,"Java","split");
        String code = new String(new char[4 * iteration]).replace("\0", " ");
        code = code + "variable = variable / " + value + ";";
        String result[] = {info,code};
        return result;
    }
```

```
    public static String[] split(int iteration, String value, String name,String
language){
        String info = Files.getInfo(language,"Java","split");
        String code = new String(new char[4 * iteration]).replace("\0", " ");
        code = code + name + " = " + name + " / " + value + ";";
        String result[] = {info,code};
        return result;
    }
```

- For:

```
/ for
    public static String[] loopFor(int iteration,String init,String limit ,String
condition ,String step, ArrayList<String> task,String language){
        String info = null;
        if (iteration < 1){
            info = Files.getInfo(language,"Java","for");
        }


        String code = new String(new char[4 * iteration]).replace("\0", " ");
        code = code + "for (i=" + init + "; i" + condition + limit + "; i = i + "
+ step + "){\n";
        String[] call = callBack(iteration+1,task,language);
        if (call[0] != null){
            info = info + call[0];
        }
        code = code + call[1]+ "\n";
        code = code + new String(new char[4 * iteration]).replace("\0", " ")
+"}";

        String[] result = {info, code};
        return result;
    }
```

- While:

```
// while
    public static String[] loopWhile(int iteration,String limit,
ArrayList<String> task,String language){
        String info = null;
        if (iteration < 1){
            info = Files.getInfo(language,"Java","while");
        }


        String code = new String(new char[4 * iteration]).replace("\0", " ");
        code = code + "int variable = 1;\n";
```

```
        code = code + new String(new char[4 * iteration]).replace("\0", " ");
        code = code + "while(variable <"  + limit + "){ \n";


        String[] call = callBack(iteration+1,task,language);
        if (call[0] != null){
            info = info + call[0];
        }
        code = code + call[1]+ "\n";

        code = code + new String(new char[4 * (iteration+1)]).replace("\0", "
")  + "variable = variable + 1" + "\n";
        code = code + new String(new char[4 * iteration]).replace("\0", " ") +
"}";

        String[] result = {info, code};
        return result;
    }
```

- IF ELSE:

```
// if else
    public static String[] ifElse(int iteration, String variable, String
condition, String compare, ArrayList<String> task,String language){
        String info = null;
        if (iteration < 1){
            info = Files.getInfo(language,"Java","ifElse");
        }
        String code =  new String(new char[4 * iteration]).replace("\0", " ");
        code = code + "if(" + variable + condition + compare + "){ \n";

        String[] call = callBack(iteration+1,task,language);
        if (call[0] != null){
            info = info + call[0];
        }
        code = code + call[1]+ "\n";

        code = code +new String(new char[4 * iteration]).replace("\0", " ") + "}
else {\n";
        int position = 0;
            for (int j=1;j< task.size();j++){
                if (task.get(j).equals("createParam")  ||
task.get(j).equals("for" )|| task.get(j).equals("while" )||
task.get(j).equals("ifElse")){
                    position = j;
                    break;
                }
                if (task.get(j).equals("switchCase") || task.get(j).equals("add")
|| task.get(j).equals("subtract") || task.get(j).equals("multiply")){
                    position = j;
```

```
                    break;
                }
                if (task.get(j).equals("split") || task.get(j).equals("show")){
                    position = j;
                    break;
                }
            }
            for (int j = 0;j<position;j++){
                task.remove(0);
            }

        String[] call2 = callBack(iteration+1,task,language);
        if (call2[0] != null){
            info = info + call2[0];
        }
        code = code + call2[1]+ "\n";
        code = code + new String(new char[4 * iteration]).replace("\0", " ") +
"}";
        String[] result = {info, code};
        return result;
    }
```

- SWITCH CASE:

```
// switch case
    public static String[] switchCase(int iteration, String
options,ArrayList<String> task,String language){
        String info = null;
        if (iteration < 1){
            info = Files.getInfo(language,"Java","switchCase");
        }
        String code = new String(new char[4 * iteration]).replace("\0", " ");

        code = code + "switch(variable){ \n";
        int i = 0;
        while(Integer.valueOf(options) > i){
            code = code + new String(new char[4 * (iteration+2)]).replace("\0", "
") + "case "+ i + ": \n";
            String[] call = callBack(iteration+3,task,language);
            if (call[0] != null){
                info = info + call[0];
            }
            code = code + call[1] + "\n";

            code = code + new String(new char[4 * (iteration+3)]).replace("\0", "
") + " break; \n";
            int position = 0;
            for (int j=1;j< task.size();j++){
```

```
                if (task.get(j).equals("createParam")  ||
task.get(j).equals("for" )|| task.get(j).equals("while" )||
task.get(j).equals("ifElse")){
                    position = j;
                    break;
                }
                if (task.get(j).equals("switchCase") || task.get(j).equals("add")
|| task.get(j).equals("subtract") || task.get(j).equals("multiply")){
                    position = j;
                    break;
                }
                if (task.get(j).equals("split") || task.get(j).equals("show")){
                    position = j;
                    break;
                }
            }
            i++;
            if (Integer.valueOf(options) > i){
                for (int j = 0;j<position;j++){
                    task.remove(0);
                }
            }


        }


        code = code + new String(new char[4 * (iteration+2)]).replace("\0", " ")
+ "default: \n";
        int position = 0;
        boolean last = false;
        System.out.println(task);
        for (int j=1;j< task.size();j++){
            if (task.get(j).equals("createParam")  || task.get(j).equals("add"
)|| task.get(j).equals("subtract" )|| task.get(j).equals("multiply")){
                position = j;
                last = true;
                break;
            }
            if (task.get(j).equals("switchCase") || task.get(j).equals("for") ||
task.get(j).equals("while") || task.get(j).equals("ifElse")){
                position = j;
                last = true;
                break;
            }
            if (task.get(j).equals("split") || task.get(j).equals("show")){
                position = j;
                last= true;
                break;
            }
        }
        for (int j = 0;j<position;j++){
            task.remove(0);
```

```
        }

        if (last){
            String[] call = callBack(iteration+3,task,language);
            if (call[0] != null){
                info = info + call[0];
            }
            code = code + call[1] + "\n";
        }

        code = code + new String(new char[4 * (iteration+3)]).replace("\0", " ")
+ " break; \n";


        code = code +  new String(new char[4 * (iteration+1)]).replace("\0", " ")
+ "}";

        String[] result = {info, code};
        return result;

    }
```

- Mostrar por pantalla:

```
// show in window
    public static String[] show(int iteration,String text,String language){
        String info = Files.getInfo(language,"Java","show");
        String code = new String(new char[4 * iteration]).replace("\0", " ");
        code = code + "System.out.println("  + text  + ");";
        String[] result = {info, code};
        return result;
    }
```

- Función reiterativa:

Algunas funciones requieren de hacer llamadas a otras funciones básicas. Este método permite la llamada a cada método de manera independiente para ir completando el código.

```
// call back function
    private static String[] callBack(int iteration,ArrayList<String>
parameterChain,String language){
        String info = null;
        String code = null;
        int position = 0;
        for (int i=1;i< parameterChain.size();i++){
```

```
            if (parameterChain.get(i).equals("createParam")  ||
parameterChain.get(i).equals("for" )|| parameterChain.get(i).equals("while" )||
parameterChain.get(i).equals("ifElse")){
                position = i;
                break;
            }
            if (parameterChain.get(i).equals("switchCase") ||
parameterChain.get(i).equals("add") || parameterChain.get(i).equals("subtract")
|| parameterChain.get(i).equals("multiply")){
                position = i;
                break;
            }
            if (parameterChain.get(i).equals("split") ||
parameterChain.get(i).equals("show")){
                position = i;
                break;
            }
        }

        if (parameterChain.get(0).equals("createParam")){
            if (position == 4 || (position == 0 && parameterChain.size() == 4)){
                String[] result = LanguageJava.createVariable(iteration,
parameterChain.get(1), parameterChain.get(2), parameterChain.get(3),language);
                info = result[0];
                code = result[1];
            } else if (position == 3 || (position == 0 && parameterChain.size()
== 3)){
                String[] result = LanguageJava.createVariable(iteration,
parameterChain.get(1), parameterChain.get(2),language);
                info = result[0];
                code = result[1];
            } else if (position == 2 || (position == 0 && parameterChain.size()
== 2)){
                String[] result = LanguageJava.createVariable(iteration,
parameterChain.get(1),language);
                info = result[0];
                code = result[1];
            }
        }

        if (parameterChain.get(0).equals("add")){
            if (position == 3 || (position == 0 && parameterChain.size() == 3)){
                String[] result = LanguageJava.add(iteration,
parameterChain.get(1), parameterChain.get(2),language);
                info = result[0];
                code = result[1];
            } else if (position == 2 || (position == 0 && parameterChain.size()
== 2)){
                String[] result = LanguageJava.add(iteration,
parameterChain.get(1),language);
                info = result[0];
                code = result[1];
            }
```

```
        }

        if (parameterChain.get(0).equals("subtract")){
            if (position == 3 || (position == 0 && parameterChain.size() == 3)){
                String[] result = LanguageJava.subtract(iteration,
parameterChain.get(1), parameterChain.get(2),language);
                info = result[0];
                code = result[1];
            } else if (position == 2 || (position == 0 && parameterChain.size()
== 2)){
                String[] result = LanguageJava.subtract(iteration,
parameterChain.get(1),language);
                info = result[0];
                code = result[1];
            }
        }

        if (parameterChain.get(0).equals("multiply")){
            if (position == 3 || (position == 0 && parameterChain.size() == 3)){
                String[] result = LanguageJava.multiply(iteration,
parameterChain.get(1), parameterChain.get(2),language);
                info = result[0];
                code = result[1];
            } else if (position == 2 || (position == 0 && parameterChain.size()
== 2)){
                String[] result = LanguageJava.multiply(iteration,
parameterChain.get(1),language);
                info = result[0];
                code = result[1];
            }
        }

        if (parameterChain.get(0).equals("split")){
            if (position == 3 || (position == 0 && parameterChain.size() == 3)){
                String[] result = LanguageJava.split(iteration,
parameterChain.get(1), parameterChain.get(2),language);
                info = result[0];
                code = result[1];
            } else if (position == 2 || (position == 0 && parameterChain.size()
== 2)){
                String[] result = LanguageJava.split(iteration,
parameterChain.get(1),language);
                info = result[0];
                code = result[1];
            }
        }

        if (parameterChain.get(0).equals("for")){
            if (position == 5 || (position == 0 && parameterChain.size() == 5)){
                parameterChain.remove(0);
                String init = parameterChain.get(0);
                parameterChain.remove(0);
                String limit = parameterChain.get(0);
```

```
                parameterChain.remove(0);
                String condition = parameterChain.get(0);
                parameterChain.remove(0);
                String step = parameterChain.get(0);
                parameterChain.remove(0);
                String[] result = LanguageJava.loopFor(iteration,init ,
limit,condition,step,parameterChain,language);
                info = result[0];
                code = result[1];
            }
        }

        if (parameterChain.get(0).equals("while")){
            if (position == 2 || (position == 0 && parameterChain.size() == 2)){
                parameterChain.remove(0);;
                String limit = parameterChain.get(0);
                parameterChain.remove(0);
                String[] result =
LanguageJava.loopWhile(iteration,limit,parameterChain,language);
                info = result[0];
                code = result[1];
            }
        }

        if (parameterChain.get(0).equals("ifElse")){
            if (position == 4 || (position == 0 && parameterChain.size() == 4)){
                parameterChain.remove(0);
                String variable = parameterChain.get(0);
                parameterChain.remove(0);
                String condition = parameterChain.get(0);
                parameterChain.remove(0);
                String compare = parameterChain.get(0);
                parameterChain.remove(0);
                String[] result = LanguageJava.ifElse(iteration, variable,
condition,compare,parameterChain,language);
                info = result[0];
                code = result[1];
            }
        }

        if (parameterChain.get(0).equals("show")){
            String[] result = LanguageJava.show(iteration,
parameterChain.get(1),language);
            info = result[0];
            code = result[1];
        }

        String[] result = {info, code};
        return result;
    }
```

5.3.3.2 Python

En este caso también contamos con métodos sobrecargados[12] para generar el código en función de los parámetros especificados.

- Creación de variables:

En este caso no es necesario especificar el tipo de variable, en caso de hacerlo este se omite.

```
// create specific type of variable
    public static String[] createVariable(int iteration,String language){
        String info = null;
        if (iteration < 1){
            info = Files.getInfo(language,"Python","createParam");
        }
        String code = null;
        code = new String(new char[4 * iteration]).replace("\0", " ");
        code = code + "variable = null";
        String[] result= {info,code};
        return result;
    }

    public static String[] createVariable(int iteration, String value,String
language){
        String info = null;
        if (iteration < 1){
            info = Files.getInfo(language,"Python","createParam");
        }
        String code = null;
        code = new String(new char[4 * iteration]).replace("\0", " ");

        try {
            code = code + "variable = " + Integer.valueOf(value);
        } catch (NumberFormatException e){

            try {
                code = code + "variable = " + Float.valueOf(value);
            } catch (NumberFormatException f){
                if (value.toLowerCase().equals("true")){
                    code = code + "variable = True";
                } else if (value.toLowerCase().equals("false")){
                    code = code + "variable = False";
                } else {
                    char comillas = '"';
                    code = code + "variable = " + comillas + value + comillas;
                }
```

[12] Se dice de los métodos que llamados igual requieren de un distinto número de variables.

```
            }

        }

        String result[] = {info, code};
        return result;
    }
    public static String[] createVariable(int iteration, String value, String
name,String language){
        String info = null;
        if (iteration < 1){
            info =  Files.getInfo(language,"Python","createParam");
        }

        String code = null;
        code = new String(new char[4 * iteration]).replace("\0", " ");
        try {
            code = code + "variable = " + Integer.valueOf(value);
        } catch (NumberFormatException e){

            try {
                code = code + "variable = " + Float.valueOf(value);
            } catch (NumberFormatException f){
                if (value.toLowerCase().equals("true")){
                    code = code + "variable = True";
                } else if (value.toLowerCase().equals("false")){
                    code = code + "variable = False";
                } else {
                    char comillas = '"';
                    code = code + "variable = " + comillas + value + comillas;
                }
            }

        }

        String result[] = {info, code};
        return result;
    }
```

Seguimos con las distintas operaciones aritméticas

- Suma:

```
// Add
    public static String[] add(int iteration,String value,String language){
        String info = Files.getInfo(language,"Python","add");
        String code = new String(new char[4 * iteration]).replace("\0", " ");
        code = code + "variable = variable + " + value;
        String result[] = {info,code};
        return result;
```

```
    }

    public static String[] add(int iteration,String value, String name,String
language){
        String info = Files.getInfo(language,"Python","add");
        String code = new String(new char[4 * iteration]).replace("\0", " ");
        code = code + name + " = " + name + " + " + value;
        String result[] = {info,code};
        return result;
    }
```

- Resta:

```
// Subtract
    public static String[] subtract(int iteration, String value,String language){
        String info = Files.getInfo(language,"Python","subtract");
        String code = new String(new char[4 * iteration]).replace("\0", " ");
        code = code + "variable = variable - " + value;
        String result[] = {info,code};
        return result;
    }

    public static String[] subtract(int iteration, String value, String
name,String language){
        String info = Files.getInfo(language,"Python","subtract");
        String code = new String(new char[4 * iteration]).replace("\0", " ");
        code = code + name + " = " + name + " - " + value;
        String result[] = {info,code};
        return result;
    }
```

- Multiplicar:

```
// Multiply
    public static String[] multiply(int iteration, String value,String language){
        String info = Files.getInfo(language,"Python","multiply");
        String code = new String(new char[4 * iteration]).replace("\0", " ");
        code = code + "variable = variable * " + value;
        String result[] = {info,code};
        return result;
    }

    public static String[] multiply(int iteration, String value, String
name,String language){
        String info = Files.getInfo(language,"Python","multiply");
        String code = new String(new char[4 * iteration]).replace("\0", " ");
        code = code + name + " = " + name + " * " + value;
        String result[] = {info,code};
```

```
        return result;
    }
```

- Dividir:

```
// Split
    public static String[] split(int iteration, String value,String language){
        String info = Files.getInfo(language,"Python","split");
        String code = new String(new char[4 * iteration]).replace("\0", " ");
        code = code + "variable = variable / " + value;
        String result[] = {info,code};
        return result;
    }

    public static String[] split(int iteration, String value, String name,String
language){
        String info = Files.getInfo(language,"Python","split");
        String code = new String(new char[4 * iteration]).replace("\0", " ");
        code = code + name + " = " + name + " / " + value;
        String result[] = {info,code};
        return result;
    }
```

- For:

```
// for
    public static String[] loopFor(int iteration,String init,String limit ,String
condition ,String step, ArrayList<String> task,String language){
        String info = null;
        if (iteration < 1){
            info = Files.getInfo(language,"Python","for");
        }


        String code = new String(new char[4 * iteration]).replace("\0", " ");
        code = code + "for i in range(" + init + "," + limit + "," + step + "):
\n";
        String[] call = callBack(iteration+1,task,language);
        if (call[0] != null){
            info = info + call[0];
        }
        code = code + call[1]+ "\n";

        String[] result = {info, code};
        return result;
    }
```

- While:

```
// while
```

```
    public static String[] loopWhile(int iteration,String limit,
ArrayList<String> task,String language){
        String info = null;
        if (iteration < 1){
            info = Files.getInfo(language,"Python","while");
        }


        String code = new String(new char[4 * iteration]).replace("\0", " ");
        code = code + "variable = 1\n";
        code = code + new String(new char[4 * iteration]).replace("\0", " ");
        code = code + "while variable <"  + limit + ": \n";

        String[] call = callBack(iteration+1,task,language);
        if (call[0] != null){
            info = info + call[0];
        }
        code = code + call[1]+ "\n";

        code = code + new String(new char[4 * (iteration+1)]).replace("\0", "
")  + "variable = variable + 1" + "\n";

        String[] result = {info, code};
        return result;
    }
```

- IF ELSE:

```
// if else
public static String[] ifElse(int iteration, String variable, String condition,
String compare, ArrayList<String> task,String language){
String info = null;
if (iteration < 1){
info = Files.getInfo(language,"Python","ifElse");
}
String code =  new String(new char[4 * iteration]).replace("\0", " ");
code = code + "if " + variable + condition + compare + ": \n";

String[] call = callBack(iteration+1,task,language);
if (call[0] != null){
info = info + call[0];
}
code = code + call[1]+ "\n";

code = code +new String(new char[4 * iteration]).replace("\0", " ") + " else:
\n";
int position = 0;
for (int j=1;j< task.size();j++){
if (task.get(j).equals("createParam")  || task.get(j).equals("for" )||
task.get(j).equals("while" )|| task.get(j).equals("ifElse")){
```

```
position = j;
break;
}
if (task.get(j).equals("switchCase") || task.get(j).equals("add") ||
task.get(j).equals("subtract") || task.get(j).equals("multiply")){
position = j;
break;
}
if (task.get(j).equals("split") || task.get(j).equals("show")){
position = j;
break;
}
}
for (int j = 0;j<position;j++){
task.remove(0);
}

String[] call2 = callBack(iteration+1,task,language);
if (call2[0] != null){
info = info + call2[0];
}
code = code + call2[1]+ "\n";
String[] result = {info, code};
return result;
}

// Similar code to switch case as in python it does not exist
public static String[] switchCase(int iteration, String options,ArrayList<String>
task,String language){
String info = null;
if (iteration < 1){
info = Files.getInfo(language,"Python","switchCase");
}
String code = new String(new char[4 * iteration]).replace("\0", " ");
code = code + "if options == 0:";

String[] call = callBack(iteration+1,task,language);
if (call[0] != null){
info = info + call[0];
}
code = code + call[1]+ "\n";

int position = 0;
for (int j=1;j< task.size();j++){
if (task.get(j).equals("createParam")  || task.get(j).equals("for" )||
task.get(j).equals("while" )|| task.get(j).equals("ifElse")){
position = j;
break;
}
if (task.get(j).equals("switchCase") || task.get(j).equals("add") ||
task.get(j).equals("subtract") || task.get(j).equals("multiply")){
position = j;
break;
```

```
}
if (task.get(j).equals("split") || task.get(j).equals("show")){
position = j;
break;
}
}
for (int j = 0;j<position;j++){
task.remove(0);
}

int i = 1;
while(Integer.valueOf(options) > i-1){
code = code + new String(new char[4 * (iteration+2)]).replace("\0", " ") + "elif
variable == "+ i + ": \n";
String[] call2 = callBack(iteration+3,task,language);
if (call2[0] != null){
info = info + call2[0];
}
code = code + call2[1] + "\n";

code = code + new String(new char[4 * (iteration+3)]).replace("\0", " ") + "
break; \n";
position = 0;
for (int j=1;j< task.size();j++){
if (task.get(j).equals("createParam")  || task.get(j).equals("for" )||
task.get(j).equals("while" )|| task.get(j).equals("ifElse")){
position = j;
break;
}
if (task.get(j).equals("switchCase") || task.get(j).equals("add") ||
task.get(j).equals("subtract") || task.get(j).equals("multiply")){
position = j;
break;
}
if (task.get(j).equals("split") || task.get(j).equals("show")){
position = j;
break;
}
}
i++;
if (Integer.valueOf(options) > i){
for (int j = 0;j<position;j++){
task.remove(0);
}
}



}


code = code + new String(new char[4 * (iteration+2)]).replace("\0", " ") + "else:
\n";
position = 0;
```

```
boolean last = false;
for (int j=1;j< task.size();j++){
if (task.get(j).equals("createParam")  || task.get(j).equals("add" )||
task.get(j).equals("subtract" )|| task.get(j).equals("multiply")){
position = j;
last = true;
break;
}
if (task.get(j).equals("switchCase") || task.get(j).equals("for") ||
task.get(j).equals("while") || task.get(j).equals("ifElse")){
position = j;
last = true;
break;
}
if (task.get(j).equals("split") || task.get(j).equals("show")){
position = j;
last= true;
break;
}
}
for (int j = 0;j<position;j++){
task.remove(0);
}

if (last){
String[] call2 = callBack(iteration+3,task,language);
if (call2[0] != null){
info = info + call2[0];
}
code = code + call2[1] + "\n";
}

code = code + new String(new char[4 * (iteration+3)]).replace("\0", " ") + "
break; \n";

String[] result = {info, code};
return result;

}
```

Mostar por pantalla:

```
// show in window
    public static String[] show(int iteration,String text,String language){
        String info = Files.getInfo(language,"Python","show");
        String code = new String(new char[4 * iteration]).replace("\0", " ");
        code = code + "print("  + text  + ")";
        String[] result = {info, code};
        return result;
    }
```

- Función reiterativa:

```
// call back function
    private static String[] callBack(int iteration,ArrayList<String>
parameterChain,String language){
        String info = null;
        String code = null;
        int position = 0;
        for (int i=1;i< parameterChain.size();i++){
            if (parameterChain.get(i).equals("createParam")  ||
parameterChain.get(i).equals("for" )|| parameterChain.get(i).equals("while" )||
parameterChain.get(i).equals("ifElse")){
                position = i;
                break;
            }
            if (parameterChain.get(i).equals("switchCase") ||
parameterChain.get(i).equals("add") || parameterChain.get(i).equals("subtract")
|| parameterChain.get(i).equals("multiply")){
                position = i;
                break;
            }
            if (parameterChain.get(i).equals("split") ||
parameterChain.get(i).equals("show")){
                position = i;
                break;
            }
        }

        if (parameterChain.get(0).equals("createParam")){
            ArrayList<String> types = new ArrayList<String>(Arrays.asList("int",
"boolean", "float","double","char","String","long","short","byte"));
            if (types.contains(parameterChain.get(1))){
                if (position == 4 || (position == 0 && parameterChain.size() ==
4)){
                    String[] result = LanguagePython.createVariable(0,
parameterChain.get(2), parameterChain.get(3),language);
                    info = result[0];
                    code = result[1];
                } else if (position == 3 || (position == 0 &&
parameterChain.size() == 3)){
                    String[] result = LanguagePython.createVariable(0,
parameterChain.get(2),language);
                    info = result[0];
                    code = result[1];
                } else if (position == 2 || (position == 0 &&
parameterChain.size() == 2)){
                    String[] result = LanguagePython.createVariable(0,language);
                    info = result[0];
                    code = result[1];
```

```
                }
            } else {
                if (position == 4 || (position == 0 && parameterChain.size() ==
4)){
                    String[] result = LanguagePython.createVariable(0,
parameterChain.get(2), parameterChain.get(3),language);
                    info = result[0];
                    code = result[1];
                } else if (position == 3 || (position == 0 &&
parameterChain.size() == 3)){
                    String[] result = LanguagePython.createVariable(0,
parameterChain.get(2),language);
                    info = result[0];
                    code = result[1];
                } else if (position == 2 || (position == 0 &&
parameterChain.size() == 2)){
                    String[] result = LanguagePython.createVariable(0,language);
                    info = result[0];
                    code = result[1];
                }
            }
        }

        if (parameterChain.get(0).equals("add")){
            if (position == 3 || (position == 0 && parameterChain.size() == 3)){
                String[] result = LanguagePython.add(iteration,
parameterChain.get(1), parameterChain.get(2),language);
                info = result[0];
                code = result[1];
            } else if (position == 2 || (position == 0 && parameterChain.size()
== 2)){
                String[] result = LanguagePython.add(iteration,
parameterChain.get(1),language);
                info = result[0];
                code = result[1];
            }
        }

        if (parameterChain.get(0).equals("subtract")){
            if (position == 3 || (position == 0 && parameterChain.size() == 3)){
                String[] result = LanguagePython.subtract(iteration,
parameterChain.get(1), parameterChain.get(2),language);
                info = result[0];
                code = result[1];
            } else if (position == 2 || (position == 0 && parameterChain.size()
== 2)){
                String[] result = LanguagePython.subtract(iteration,
parameterChain.get(1),language);
                info = result[0];
                code = result[1];
            }
        }
```

```
        if (parameterChain.get(0).equals("multiply")){
            if (position == 3 || (position == 0 && parameterChain.size() == 3)){
                String[] result = LanguagePython.multiply(iteration,
parameterChain.get(1), parameterChain.get(2),language);
                info = result[0];
                code = result[1];
            } else if (position == 2 || (position == 0 && parameterChain.size()
== 2)){
                String[] result = LanguagePython.multiply(iteration,
parameterChain.get(1),language);
                info = result[0];
                code = result[1];
            }
        }

        if (parameterChain.get(0).equals("split")){
            if (position == 3 || (position == 0 && parameterChain.size() == 3)){
                String[] result = LanguagePython.split(iteration,
parameterChain.get(1), parameterChain.get(2),language);
                info = result[0];
                code = result[1];
            } else if (position == 2 || (position == 0 && parameterChain.size()
== 2)){
                String[] result = LanguagePython.split(iteration,
parameterChain.get(1),language);
                info = result[0];
                code = result[1];
            }
        }

        if (parameterChain.get(0).equals("for")){
            if (position == 5 || (position == 0 && parameterChain.size() == 5)){
                parameterChain.remove(0);
                String init = parameterChain.get(0);
                parameterChain.remove(0);
                String limit = parameterChain.get(0);
                parameterChain.remove(0);
                String condition = parameterChain.get(0);
                parameterChain.remove(0);
                String step = parameterChain.get(0);
                parameterChain.remove(0);
                String[] result = LanguagePython.loopFor(iteration,init ,
limit,condition,step,parameterChain,language);
                info = result[0];
                code = result[1];
            }
        }

        if (parameterChain.get(0).equals("while")){
            if (position == 2 || (position == 0 && parameterChain.size() == 2)){
                parameterChain.remove(0);;
                String limit = parameterChain.get(0);
                parameterChain.remove(0);
```

```
                String[] result =
LanguagePython.loopWhile(iteration,limit,parameterChain,language);
                info = result[0];
                code = result[1];
            }
        }

        if (parameterChain.get(0).equals("ifElse")){
            if (position == 4 || (position == 0 && parameterChain.size() == 4)){
                parameterChain.remove(0);
                String variable = parameterChain.get(0);
                parameterChain.remove(0);
                String condition = parameterChain.get(0);
                parameterChain.remove(0);
                String compare = parameterChain.get(0);
                parameterChain.remove(0);
                String[] result = LanguagePython.ifElse(iteration, variable,
condition,compare,parameterChain,language);
                info = result[0];
                code = result[1];
            }
        }

        if (parameterChain.get(0).equals("show")){
            String[] result = LanguagePython.show(iteration,
parameterChain.get(1),language);
            info = result[0];
            code = result[1];
        }

        String[] result = {info, code};
        return result;
    }
```

Capítulo 6. ANÁLISIS DE RESULTADOS

En esta sección se discute los resultados obtenidos del desarrollo de la aplicación.

La aplicación es capaz de procesar texto en español y generar un código tanto en Java como en Python. Estos códigos son de una naturaleza simple, son códigos cortos que carecen de complejidad de computación.

En cuanto al resultado de la aplicación procedemos a analizar un ejemplo de su ejecución seleccionando como ajustes español y "SparkNLP" para el procesamiento natural del lenguaje y Java como lenguaje para la generación del código.

Ilustración 8- Texto introducido en la aplicación

En la ilustración podemos observar múltiples oraciones. Cada una de ellas hace referencia a una posible función de un programa. Obteniendo como resultado lo siguiente.

```
"La sentencia switch case sirve para diferenciar el curso de accion en funcion del valor de una variable
de manera que existan multiples maneras de continuar el caso de uso más comun que se da es
cuando se crea un menu con diferentes opciones, asi dependiendo del valor de una varialbe podemos
actuar y realizar diferentes funciones.  En java es necesario que el código que se ejecuta por una
sentencia se encuntre entre llaves""Sumar es una operación muy sencilla a la hora de operar. En java
es necesario tener en cuenta el tipo de las variables, si sumamos variables que tengan decimales la
variable de destino tambien debera poder admitirlo o no se guardara de manera correcta el
resultado.""Restar es una operación muy sencilla a la hora de operar. En java es necesario tener en
cuenta el tipo de las variables, si sumamos variables que tengan decimales la variable de destino
tambien debera poder admitirlo o no se guardara de manera correcta el resultado."null"Sumar es una
operación muy sencilla a la hora de operar. En java es necesario tener en cuenta el tipo de las
variables, si sumamos variables que tengan decimales la variable de destino tambien debera poder
admitirlo o no se guardara de manera correcta el resultado.""Restar es una operación muy sencilla a la
hora de operar. En java es necesario tener en cuenta el tipo de las variables, si sumamos variables
que tengan decimales la variable de destino tambien debera poder admitirlo o no se guardara de
manera correcta el resultado."

switch(variable){
      case 0:
        resultado = resultado +  5;
         break;
      case 1:
        resultado = resultado -  4;
         break;
      case 2:
        if(referencia<resultado){
           resultado = resultado +  5;
        } else {
           variable = variable -  40;
        }
         break;
      default:
         break;
  }
```

Ilustración 9- Resultado de la ejecución

Se puede observar una primera parte de texto donde se explica la peculiaridad de la función más importante escogida. En este texto también se explican peculiaridades con respecto al lenguaje que se ha escogido. Disponiendo de otro mensaje si se trata de Python como se puede comprobar en la siguiente ilustración.

```
""" En python a diferencia de otros lenguajes de programación no dispone de una sentencia switch
case, en caso de querer imitar esta se realiza mediante la encadenación de multiples sentencias if
Else."""Sumar es una operación muy sencilla a la hora de operar. En Python es necesario tener en
cuenta el tipo de las variables. En caso de no tenerlo en cuenta se pueden producir errores durante la
ejecución al tratar de operar variables si son de tipos distintos."""""Restar es una operación muy
sencilla a la hora de operar. En Python es necesario tener en cuenta el tipo de las variables. En caso
de no tenerlo en cuenta se pueden producir errores durante la ejecución al tratar de operar variables si
son de tipos distintos."""null"""Sumar es una operación muy sencilla a la hora de operar. En Python es
necesario tener en cuenta el tipo de las variables. En caso de no tenerlo en cuenta se pueden producir
errores durante la ejecución al tratar de operar variables si son de tipos distintos."""""Restar es una
operación muy sencilla a la hora de operar. En Python es necesario tener en cuenta el tipo de las
variables. En caso de no tenerlo en cuenta se pueden producir errores durante la ejecución al tratar de
operar variables si son de tipos distintos."""""Restar es una operación muy sencilla a la hora de
operar. En Python es necesario tener en cuenta el tipo de las variables. En caso de no tenerlo en
cuenta se pueden producir errores durante la ejecución al tratar de operar variables si son de tipos
distintos."""
```

Ilustración 10- Comentario del código realizado en Python

Así mismo se puede observar que en este caso el código generado si corresponde a lo que se pide en un inicio. Sin embargo, el programa puede presentar ciertos fallos durante la ejecución.

En el procesado del lenguaje natural se realiza una comparación con de las palabras con un diccionario de sinónimos, si este presenta algún fallo o bien es incompleto no se generará las palabras clave adecuadas.

En cuanto al sistema de lematización el modelo empleado, como es normal, no presenta un acierto perfecto y se equivoca con algunas palabras por lo que en este caso podría perderse información importante para los códigos. Un ejemplo sería confundir ciertas palabras y considerar aquellas que no aportan información como palabras clave o viceversa.

En la generación de código pueden faltar palabras clave en el vector que impidan la selección de un método, ya sea bien por fallo del sistema o bien por falta de información por parte del

usuario. Esto se da cuando en un bucle no se especifica el número de iteraciones necesarias o el paso con el que debe avanzar.

Y por último podemos observar los estados de la ejecución.

Ilustración 11- Estados de la ejecución

A continuación, se muestra que es lo que ocurre cuando el programa no es capaz de procesar el texto y no se generan palabras clave sobre las funciones básicas.

Ilustración 12- Error durante el procesado del texto

Capítulo 7. CONCLUSIONES Y TRABAJOS FUTUROS

En conclusión, el proyecto ha cumplido con los objetivos impuestos en un principio, llegando a tener una aplicación capaz de procesar texto en español para producir el código referente a este.

El problema que se quiere solucionar requiere de mucha más inversión que la que se puede realizar en un TFG[13] por lo que el alcance de lo producido es insuficiente a lo requerido en la industria. El proyecto muestra una posible solución al problema, pero no la única. La idea general parece no distar con respecto a lo que se busca en la industria actualmente.

El proyecto presenta múltiples dificultades que se han de tener en cuenta durante la fase de diseño para no perjudicar la calidad del trabajo. En específico hay múltiples elementos que pueden limitar el alcance de la aplicación, encontrando el módulo de procesado de lenguaje natural, el módulo de generación de código y el sistema que permite unir sendos módulos.

El módulo del procesado del lenguaje natural presenta la primera barrera del programa y es que una mala compresión del texto introducido finaliza de manera inequívoca en código que no realice lo que se busca.

El módulo de generación de código puede limitar el grado de complejidad de los códigos creados, limitando el alcance de la aplicación.

El sistema que permite unir ambos módulos es una parte crítica del mismo pues una mala sincronización produce un fallo del sistema.

La elaboración de un programa refinado de estas características que sea capaz de realizar en mejores condiciones y que abarque un mayor rango de posibilidades requiere de un gran grupo de desarrollo. Un programa que se quede a medias, aunque sea capaz de realizar todas

[13] Trabajo de fin de grado

estas funciones a bajo nivel puede quedarse obsoleto en poco tiempo debido a la velocidad con la que cambia la industria.

En cuanto al futuro de la aplicación, esta se ha diseñado para que se pueda incrementar de manera individual, pudiendo agregar módulos de manera independiente sin afectar al resto de la aplicación.

El primer trabajo para la aplicación seria incrementar las posibilidades del código generado, para poder ofrecer una mayor profundidad, seguido de la inclusión de distintos lenguas de trabajo para incrementar el número de posibles usuarios. Terminando por añadir distintos lenguajes de programación.

La aplicación permitiría la integración de un nuevo paradigma de resolución de estos problemas permitiendo así probar su grado de valía. Lo que permite que distintos programadores puedan intentar realizar un acercamiento propio a la resolución del problema, abriendo el abanico de posibilidades.

Capítulo 8. BIBLIOGRAFÍA

[1] Clement, C. B. (November 16–20, 2020). *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 9052–9065.* Association for Computational Linguistics. Obtenido de https://aclanthology.org/2020.emnlp-main.728.pdf

[2] Transifex. (2013). *Translating Natural Language to Code.* Transifex. Obtenido de https://es.transifex.com/blog/2013/translate-natural-language-to-code/

[3] Weaver, W. (july 15, 1949). *Translation.* New mexico: The Rockefeller Foundation. Obtenido de https://gunkelweb.com/coms493/texts/weaver_translation.pdf

[4] M. Youssef, K. Zakaria, B. Jamal, B. Toumi and B. M. Gaouth, "Text to Code Conversion Using Deep Learning for NLP," 2020 International Symposium on Advanced Electrical and Communication Technologies (ISAECT), 2020, pp. 1-5, doi: 10.1109/ISAECT50560.2020.9523647.

[5] The Code2Text Challenge: Text Generation in Source Code Libraries. https://arxiv.org/ftp/arxiv/papers/1708/1708.00098.pdf

[6] https://ml4code.github.io/base-taxonomy/generative.html

[7] Text to Code: Pseudo Code Generation. November 2019. DOI:10.1007/978-3-030-34365-1_3 In book: Context-Aware Systems and Applications, and Nature of Computation and Communication (pp.20-37). https://www.researchgate.net/publication/336928377_Text_to_Code_Pseudo_Code_Generation

[8] https://www.springerprofessional.de/en/text-to-code-pseudo-code-generation/17337634

[9] https://stanfordnlp.github.io/CoreNLP/

[10] https://nlp.johnsnowlabs.com

[11] https://www.42madrid.com

[12] https://hadoop.apache.org

[13] https://iconos8.es/lunacy

[14] https://es.wikipedia.org/wiki/Procesamiento_de_lenguajes_naturales

[15] https://blog.bitext.com/what-is-the-difference-between-stemming-and-lemmatization/

[16] https://www.un.org/sustainabledevelopment/es/news/communications-material/

ANEXO I: ALINEACIÓN DEL PROYECTO CON LOS ODS

Los ODS u objetivos de desarrollo sostenido fueron definidos por la Organización de las naciones unidas (ONU). Fueron definidos en el marco del desarrollo futuro con la intención de establecerlos como finalidad de los diferentes proyectos que se lleven a cabo. En la imagen podemos observar los 17 objetivos.

Ilustración 13- Objetivos de desarrollo sostenible

Fuente: Organización de las naciones unidas

El proyecto se ha definido en el marco de la simplificación de la formación de los nuevos programadores por lo que durante el proceso de este se ha seguido de manera inequívoca el 4º objetivo. Así mismo, de manera indirecta se han seguido los objetivos de innovación en la industria que corresponde con el objetivo 9 y con el crecimiento económico que se corresponde con el objetivo 8, al intentar introducir a un mayor numero de personas cualificadas en el sector de desarrollo de software.

ANEXO II: GUÍA DE INSTALACIÓN

La aplicación requiere de distintos programas para su funcionamiento. A continuación, se detallan que programas son necesarios y como instalarlos.

Al tratarse de una aplicación desarrollada en Java será necesario tener la "Java virtual Machine". Este es el elemento más sencillo de instalar. En nuestro caso usaremos Java 11.

La instalación se llevará a cabo con el ejecutable y siguiendo los pasos del instalador.

https://adoptium.net/es/temurin/releases?version=11

Durante la instalación será muy importante que la ruta de instalación no contenga espacios y que se cree la variable de entorno de "JAVA HOME" para poder ejecutar comandos desde el símbolo de sistema.

La segunda aplicación que se necesita es Hadoop. En este caso usaremos la versión 3.3.3 de Hadoop, para ello seguimos los siguientes pasos

1. Descargamos Hadoop y descomprimimos los archivos.

Podemos descargar los archivos desde el siguiente enlace

https://hadoop.apache.org/release/3.3.3.html

2. Establecemos las variables de entorno.

Para ello necesitamos saber la ruta de instalación.

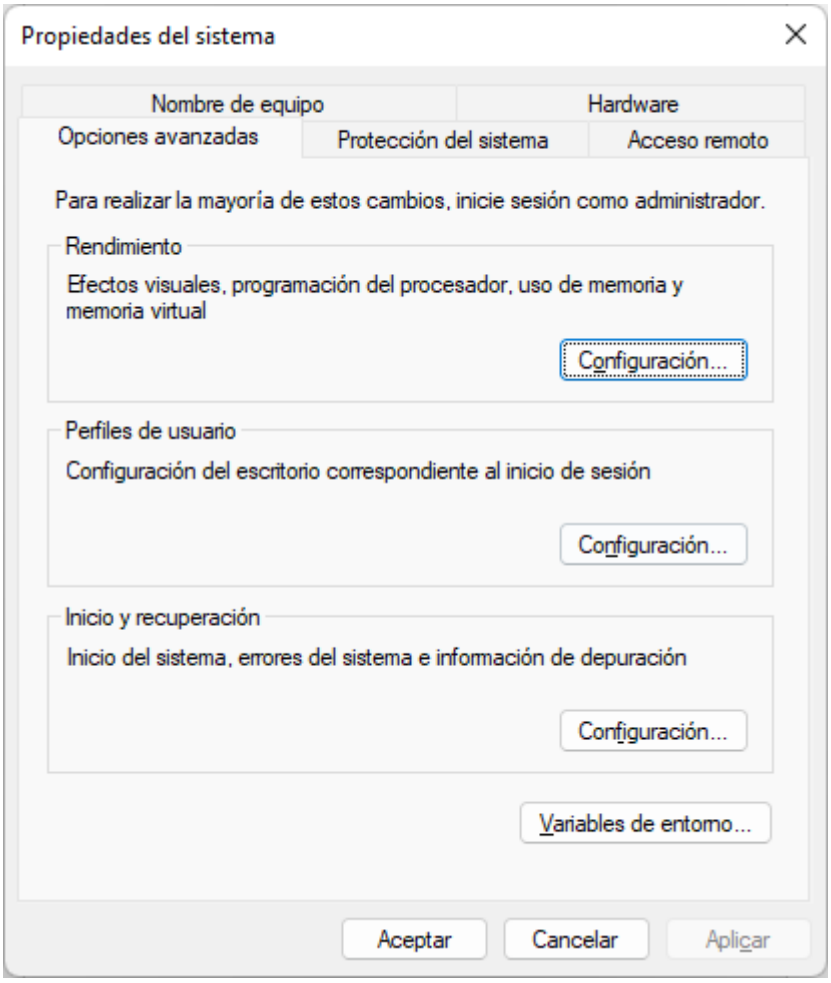

Ilustración 14- Variables de entorno

Clicamos en variables de entorno.

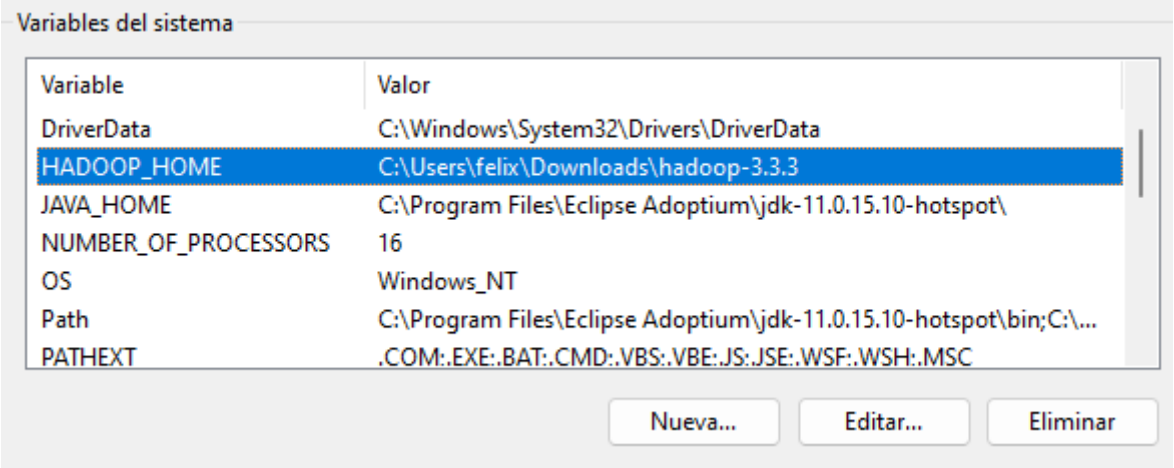

Ilustración 15- Hadoop Home

Añadimos a las variables del sistema la variable “HADOOP_HOME”.

3. Crear carpetas

En el raíz crear la carpeta “data” y dentro de esta crear las subcarpetas “datanode” y “namenode”

4. Configurar los archivos
    a. hadoop-env.cmd
    b. core-site.xml
    c. hdfs-site.xml
    d. mapred-site.xml

Estos 4 archivos se encuentran en etc/hadoop

En el primer archivo tenemos que especificar donde se encuentra “JAVA_HOME”, es importante que esta ruta no contenga espacios, de lo contrario provocara un error.

```
@rem The java implementation to use.  Required.
set JAVA_HOME=C:\Users\felix\Downloads\jdk-11.0.15.10-hotspot
```

Ilustración 16- Hadoop-env.cmd

En el segundo archivo debemos añadir el siguiente código:

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

En el tercer archivo hay que añadir el siguiente código:

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>C:\Users\felix\Downloads\hadoop-3.3.3\data\namenode</value>
  </property>
  <property>
    <name>dfs.datanode.name.dir</name>
    <value>C:\Users\felix\Downloads\hadoop-3.3.3\data\datanode</value>
  </property>
</configuration>
```

En el cuarto archivo hay que añadir el siguiente código:

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>
```

Con esto tendremos instalado lo necesario para ejecutar Hadoop, una vez lo hagamos ya estará todo listo.

ANEXO III: GUÍA DE USO

La aplicación presenta ciertas restricciones a la hora de especificar la información.

El primer punto es la presencia de ciertas palabras clave, pues sin ellas no se generará el vector que sirve de enlace entre los módulos.

El segundo sería la necesidad de separar en distintas oraciones las distintas funciones que se requieren. Un ejemplo sería especificar que es lo que se quiere realizar dentro de un bucle.

El tercero es que la aplicación es sensible a las faltas ortográficas.

A continuación, se dejan a modo de ejemplo distintos textos para el procesado:

- un bucle que realice 4 repeticiones. sumar 5 en una variable.
- un menú con 3 opciones. sumar 5 a una variable llamada resultado. restar 4 una variable llamada resultado. comprobar que una variable llamada referencia es menor que otra llamada resultado. sumar 5 a una variable llamada resultado. restar 40
- comprobar que una variable llamada resultado es menor que otra llamada referencia. multiplicar por 5 en una variable llamada resultado. dividir por 2 en una variable llamada referencia
- un bucle que empiece en 2 hasta 4 en pasos de 1. sumar 5 en una variable

ANEXO IV: CÓDIGO DE LA APLICACIÓN

Aquí podemos encontrar el código que falta de la aplicación pero que no hace referencia a la función principal del programa. Anteriormente se ha ignorado pues no corresponde con la funcionalidad principal de la aplicación:

Encontramos varias clases de apoyo

- Colors:

Define los diferentes colores que existen en la aplicación.

```
/**
 * This class defines the different colors that will be
 * used in the project to set the look and feel of the it.
 */
public class Colors {

    /** White color */
    public final static Color WHITE = new Color(255, 255, 255);

    /** Blue Color */
    public final static Color BLUE = new Color(84, 147, 184);

    /** Black color */
    public final static Color BLACK = new Color(0, 0, 0);

    /** Black color */
    public final static Color COD_BLACK = new Color(32,28,28);

    /** Egg white color */
    public final static Color WHITE_LIGHT = new Color(253, 253, 253);

    /**
     * Method to set the background color to {@linkplain WHITE}
     *
     * @param comp <code>JComponent</code> this is the component affected.
     */
    public static void setBackgroundWhite(JComponent comp) {
        comp.setBackground(WHITE);
    }

    /**
     * Method to set the background color to {@linkplain BLUE}
     *
```

```
     * @param comp <code>JComponent</code> this is the component affected.
     */
    public static void setBackgroundBlue(JComponent comp) {
        comp.setBackground(BLUE);
    }

    /**
     * Method to set the background color to {@linkplain BLACK}
     *
     * @param comp <code>JComponent</code> this is the component affected.
     */
    public static void setBackgroundBlack(JComponent comp) {
        comp.setBackground(BLACK);
    }

}
```

- Screen:

Sirve para obtener de manera dinámica las especificaciones de tamaño de la pantalla

```
public class Screen {

    /** Height of the screen*/
    private static int height=
(int)(Math.round((Toolkit.getDefaultToolkit().getScreenSize()).getHeight()));

    /** Width of the screen*/
    private static int width=
(int)(Math.round((Toolkit.getDefaultToolkit().getScreenSize()).getWidth()));

    /**
     * Method to get the width of the screen of the computer used
     * in case of multiple screens it returns the main window
     * @return Width value of the screen as an int
     */
    public static int getScreenWidth(){
        return width;
    }

    /**
     * Overloaded method to get the width of the screen multiplied by a factor
     * @param factor Double value
     * @return Width value of the screen multiply by a factor
     */
    public static int getScreenWidth(double factor){
        return (int)(factor*getScreenWidth());
    }

    /**
```

```
     * Method to get the height of the screen of the computer used
     * in case of multiple screens it returns the main window
     * @return Height value of the screen as an int
     */
    public static int getScreenHeight(){
        return height;
    }

    /**
     * Overloaded method to get the Height of the screen multiplied by a factor
     * @param factor Double value
     * @return Height value of the screen multiply by a factor
     */
    public static int getScreenHeight(double factor){
        return (int)(factor*getScreenHeight());
    }

}
```

- Files:

Sirve para cargar la información de los diferentes ficheros, como el diccionario o las opciones de procesado.

```
public class Files {

    /**
     * This method us used to pen a file
     * @param path Route of the file that wants to be opened
     * @return object file
     */
    public static File OpenFile(String path){
        File f = null;
        try{
            f = new File(path);
            return f;
        } catch(Exception e) {
            System.out.println("Archivo no encontrado en " + path);
            return null;
        }
    }

    /**
     * Method to get the possible codes of the app
     * @return ArrayList with the name of all possible programming language
     */
    public static ArrayList<String> getCodes(){
        ArrayList<String> codes = new ArrayList<String>();
        try{
            File data = OpenFile("src/main/resources/data/data.csv");
            FileReader  fr = new FileReader(data);
            BufferedReader br = new BufferedReader(fr);
```

```
            String line;
            while((line= new String(br.readLine().getBytes(), "UTF-8")) != null){
                String[] split = line.split(",");
                if(split[0].equals("Codes"))
                    break;
            }
            while((line=br.readLine()) != null){
                String[] split = line.split(",");
                if (split.length == 2)
                    codes.add(split[1]);
                if (split.length < 2)
                    break;
            }
            br.close();
        } catch (FileNotFoundException e){
            e.printStackTrace();
        } catch (Exception e){
            e.printStackTrace();
        }
        return codes;
    }


    /**
     * Loads the possible methods to use for the specific programming language
     * @param code Programing Language used to know which methods are available
     * @return ArrayList with the name of all possible methods
     */
    public static ArrayList<String> getCodesMethods(String code){
        ArrayList<String> codesMethods = new ArrayList<String>();
        try{
            File data = OpenFile("src/main/resources/data/data.csv");
            FileReader  fr = new FileReader(data);
            BufferedReader br = new BufferedReader(fr);
            String line;
            while((line= new String(br.readLine().getBytes(), "UTF-8")) != null){
                String[] split = line.split(",");
                if(split[0].equals("Codes"))
                    break;
            }
            while((line= new String(br.readLine().getBytes(), "UTF-8")) != null){
                String[] split = line.split(",");
                if(split[1].equals(code))
                    break;
            }
            while((line=br.readLine()) != null){
                String[] split = line.split(",");
                if (split.length == 3)
                    codesMethods.add(split[2]);
                if (split.length < 3)
                    break;
            }
            br.close();
```

```
        } catch (FileNotFoundException e){
            e.printStackTrace();
        } catch (Exception e){
            e.printStackTrace();
        }
        return codesMethods;
    }

    /**
     * Loads the possible methods to use for the specific programming language
     * @param code Programing Language used to know which methods are available
     * @param method Which method to look for
     * @return ArrayList with the name of all possible methods
     */
    public static ArrayList<String> getCodesMethodsOptions(String code,String
method){
        ArrayList<String> codesMethodsOptions = new ArrayList<String>();
        try{
            File data = OpenFile("src/main/resources/data/data.csv");
            FileReader  fr = new FileReader(data);
            BufferedReader br = new BufferedReader(fr);
            String line;
            while((line= new String(br.readLine().getBytes(), "UTF-8")) != null){
                String[] split = line.split(",");
                if(split[0].equals("Codes"))
                    break;
            }
            while((line= new String(br.readLine().getBytes(), "UTF-8")) != null){
                String[] split = line.split(",");
                if(split[1].equals(code))
                    break;
            }
            while((line= new String(br.readLine().getBytes(), "UTF-8")) != null){
                String[] split = line.split(",");
                if(split[2].equals(method))
                    break;
            }
            while((line=br.readLine()) != null){
                String[] split = line.split(",");
                if (split.length == 4)
                    codesMethodsOptions.add(split[3]);
                if (split.length < 4)
                    break;
            }
            br.close();
        } catch (FileNotFoundException e){
            e.printStackTrace();
        } catch (Exception e){
            e.printStackTrace();
        }
        return codesMethodsOptions;
    }
```

```
    /**
     * Method to get the possible languages of the app
     * @return ArrayList with the name of all possible natural language
     */
    public static ArrayList<String> getLanguages(){
        ArrayList<String> languages = new ArrayList<String>();
        try{
            File data = OpenFile("src/main/resources/data/data.csv");
            FileReader  fr = new FileReader(data);
            BufferedReader br = new BufferedReader(fr);
            String line;
            while((line=br.readLine()) != null){
                String[] split = line.split(",");
                if(split[0].equals("Languages"))
                    break;
            }
            while((line= new String(br.readLine().getBytes(), "UTF-8")) != null){
                String[] split = line.split(",");
                if (split.length == 2)
                languages.add(split[1]);
                if (split.length < 2)
                    break;
            }
            br.close();
        } catch (FileNotFoundException e){
            e.printStackTrace();
        } catch (Exception e){
            e.printStackTrace();
        }
        return languages;
    }

    /**
     * Loads the possible methods to use for the specific language
     * @param language Natural Language used to know which methods are available
     * @return ArrayList with the name of all possible methods
     */
    public static ArrayList<String> getLanguagesMethods(String language){
        ArrayList<String> LanguagesMethods = new ArrayList<String>();
        try{
            File data = OpenFile("src/main/resources/data/data.csv");
            FileReader  fr = new FileReader(data);
            BufferedReader br = new BufferedReader(fr);
            String line;
            while((line= new String(br.readLine().getBytes(), "UTF-8")) != null){
                String[] split = line.split(",");
                if(split[0].equals("Languages"))
                    break;
            }
            while((line= new String(br.readLine().getBytes(), "UTF-8")) != null){
                String[] split = line.split(",");
                if(split[1].equals(language))
                    break;
```

```
            }
            while((line=br.readLine()) != null){
                String[] split = line.split(",");
                if (split.length == 3)
                    LanguagesMethods.add(split[2]);
                if (split.length < 3)
                    break;
            }
            br.close();
        } catch (FileNotFoundException e){
            e.printStackTrace();
        } catch (Exception e){
            e.printStackTrace();
        }
        return LanguagesMethods;
    }

    /**
     * Loads the possible methods to use for the specific programming language
     * @param Language natural Language used to know which methods are available
     * @param method Which method is used to know all the options
     * @return ArrayList with the name of all possible methods
     */
    public static ArrayList<String> getLanguagesMethodsOptions(String
language,String method){
        ArrayList<String> languagesMethodsOptions = new ArrayList<String>();
        try{
            File data = OpenFile("src/main/resources/data/data.csv");
            FileReader  fr = new FileReader(data);
            BufferedReader br = new BufferedReader(fr);
            String line;
            while((line= new String(br.readLine().getBytes(), "UTF-8")) != null){
                String[] split = line.split(",");
                if(split[0].equals("Languages"))
                    break;
            }
            while((line= new String(br.readLine().getBytes(), "UTF-8")) != null){
                String[] split = line.split(",");
                if(split[1].equals(language))
                    break;
            }
            while((line= new String(br.readLine().getBytes(), "UTF-8")) != null){
                String[] split = line.split(",");
                if(split[2].equals(method))
                    break;
            }
            while((line=br.readLine()) != null){
                String[] split = line.split(",");
                if (split.length == 4)
                    languagesMethodsOptions.add(split[3]);
                if (split.length < 4)
                    break;
            }
```

```
            br.close();
        } catch (FileNotFoundException e){
            e.printStackTrace();
        } catch (Exception e){
            e.printStackTrace();
        }
        return languagesMethodsOptions;
    }

    /**
     * Method to get the possible languages of the app
     * @return ArrayList with the name of all possible natural language
     */
    public static ArrayList<String> getDictionary(String language){
        ArrayList<String> dictionary = new ArrayList<String>();
        try{
            File data =
OpenFile("src/main/resources/data/Dictionaries/Dictionary" + language + ".csv");
            BufferedReader br = new BufferedReader(new InputStreamReader(new
FileInputStream(data), "utf-8"));
            String line;
            while((line=br.readLine()) != null){
                dictionary.add(line);
            }
            br.close();
        } catch (FileNotFoundException e){
            e.printStackTrace();
        } catch (Exception e){
            e.printStackTrace();
        }
        return dictionary;
    }

    /**
     * Method to get the possible languages of the app
     * @return ArrayList with the name of all possible natural language
     */
    public static String getInfo(String language,String code,String method){
        String info = new String();
        try{
            File data = OpenFile("src/main/resources/data/code/" + language + "/"
+ code + "/" + method);
            BufferedReader br = new BufferedReader(new InputStreamReader(new
FileInputStream(data), "utf-8"));
            String line;
            while((line=br.readLine()) != null){
                info = info + line;
            }
            br.close();
        } catch (FileNotFoundException e){
            e.printStackTrace();
        } catch (Exception e){
            e.printStackTrace();
```

```
        }
        return info;
    }

    /**
     * Method to get the possible languages of the app
     * @return ArrayList with the name of all possible natural language
     */
    public static ArrayList<String>  getSynonyms(String language,String word){
        ArrayList<String> synonyms = new ArrayList<String>();
        try{
            File data = OpenFile("src/main/resources/data/Synonyms/" + language +
".txt");
            BufferedReader br = new BufferedReader(new InputStreamReader(new
FileInputStream(data), "utf-8"));
            String line;
            while((line=br.readLine()) != null){
                String[] words = line.split(",");
                if (words[0].equals(word)){
                    for (int i=1;i<words.length;i++){
                        synonyms.add(words[i]);
                    }
                }
            }
            br.close();
        } catch (FileNotFoundException e){
            e.printStackTrace();
        } catch (Exception e){
            e.printStackTrace();
        }

        System.out.print(synonyms);
        return synonyms;
    }
}
```