
Traductor de Texto a Lengua de Signos Española (LSE)



TRABAJO DE FIN DE GRADO 2019/2020

AUTORES

Sara Vegas Cañas
Miguel Rodríguez Cuesta
Alejandro Torralbo Fuentes

DIRECTORES

Virginia Francisco Gilmartín
Antonio Fernando García Sevilla

*Grado de Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid*

Convocatoria 2019/2020

Este documento está preparado para ser imprimido a doble cara.

Traductor de Texto a Lengua de Signos Española (LSE)

Trabajo de Fin de Grado en Ingeniería Informática

AUTORES

**Sara Vegas Cañas
Miguel Rodríguez Cuesta
Alejandro Torralbo Fuentes**

DIRECTORES

**Virginia Francisco Gilmartín
Antonio Fernando García Sevilla**

*Grado de Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid*

Convocatoria 2019/2020

Índice

1. Introducción	1
1.1. Motivación	1
1.2. Objetivos	2
1.3. Estructura del documento	3
2. Estado del Arte	5
2.1. La discapacidad auditiva	5
2.2. Sistemas Aumentativos y Alternativos de Comunicación (SAAC)	6
2.3. La Lengua de Signos	8
2.3.1. Reglas de la Lengua de Signos Española	10
2.3.2. Bancos de videos e imágenes LSE	20
2.4. Aplicaciones de Traducción de Texto a LSE	23
2.4.1. Text2Sign	23
2.4.2. TextoSign	24
2.4.3. StorySign	24
2.4.4. TeCuento	24
2.4.5. Conclusiones	26
2.5. Procesamiento del Lenguaje Natural	26
2.5.1. Estructura del Procesamiento del Lenguaje Natural . .	26
3. Tecnologías utilizadas	29
3.1. Servicios Web	29
3.1.1. Servicios Web SOAP	29
3.1.2. Servicios Web REST	30
3.1.3. Ventajas de Servicios Web	31
3.1.4. Desventajas de Servicios Web	31
3.2. Flask	31
3.3. Herramientas PLN	32
3.3.1. Spacy	32
3.3.2. NLTK	33
3.3.3. FreeLing	33

3.3.4. Conclusiones	33
3.4. Nginx	33
3.5. Google Colaboratory	34
4. Metodologías utilizadas	37
4.1. Metodologías de gestión del proyecto	37
4.2. Trello	39
4.3. Reuniones	40
4.4. Control de versiones	41
4.4.1. Estructura de Trabajo	41
5. Text2LSE	43
5.1. Arquitectura	43
5.2. Back-End	45
5.2.1. Servicio web de traducción de palabra a LSE (vídeo) .	45
5.2.2. Servicio web de traducción de palabra a LSE (imagen)	47
5.2.3. Servicio web para Traducción de texto en castellano a texto en LSE	49
5.2.4. Servicio web para traducción de texto castellano a tex- to en LSE dependiendo de los vídeos existentes	56
5.2.5. Servicio web para traducción de texto castellano a tex- to en LSE dependiendo de las imágenes existentes . .	56
5.2.6. Servicio web para traducción de texto a LSE (vídeo) .	58
5.3. Front-End	60
6. Trabajo individual	65
6.1. Sara Vegas Cañas	65
6.2. Alejandro Torralbo Fuentes	66
6.3. Miguel Rodríguez Cuesta	67

Índice de figuras

2.1. Representación del abecedario en el Sistema Braille	8
2.2. Representación de distintas palabras con pictogramas	9
2.3. Representación de la configuración de la mano en Lengua de Signos	11
2.4. Signo “ <i>ayudar</i> ” en LSE del banco de imágenes ARASAAC . .	11
2.5. Representación de la orientación de las manos en Lengua de Signos	12
2.6. Signo “ <i>visera</i> ” en LSE del banco de imágenes ARASAAC . .	13
2.7. Signo “ <i>oir</i> ” en LSE del banco de imágenes ARASAAC	13
2.8. Ejemplos de tipos de movimientos realizados al signar en LSE	14
2.9. Signo “ <i>mucho</i> ” en LSE del banco de imágenes ARASAAC . .	14
2.10. Signo “ <i>hombre</i> ” del banco de imágenes ARASAAC	15
2.11. Signo “ <i>mujer</i> ” del banco de imágenes ARASAAC	16
2.12. Signo “ <i>abuelo</i> ” del banco de imágenes ARASAAC	16
2.13. Signo “ <i>abuela</i> ” del banco de imágenes ARASAAC	16
2.14. Signo “ <i>niño</i> ” del banco de imágenes ARASAAC	17
2.15. Signo “ <i>niña</i> ” del banco de imágenes ARASAAC	17
2.16. Signo “ <i>uno</i> ” del banco de imágenes ARASAAC	18
2.17. Signo “ <i>muchos</i> ” del banco de imágenes ARASAAC	19
2.18. Signo “ <i>querer</i> ” en LSE del banco de imágenes ARASAAC . .	20
2.19. Signo “ <i>señalar</i> ” en LSE del banco de imágenes ARASAAC . .	20
2.20. Signo “ <i>hola</i> ” en LSE del banco de imágenes ARASAAC	21
2.21. Recursos LSE de la Fundación CNSE	22
2.22. Imagen de la página web SpreadTheSign	23
2.23. Maya, avatar usado por TextoSign.	24
2.24. Imagen de la aplicación StorySign.	25
2.25. Imagen de la aplicación TeCuento.	25
3.1. Vista de un cuaderno de Google Colaboratory	35
4.1. Estructura “ <i>Git</i> ” utilizada en el proyecto.	42

5.1. Aplicación web Text2LSE	44
5.2. Arquitectura de Text2LSE	46
5.3. Esquema Proxy Inverso	47
5.4. Flujo del servicio de Traducción de una palabra a vídeo en LSE	48
5.5. Vídeo devuelto por el servicio de traducción de la palabra “ <i>agua</i> ” a LSE	49
5.6. Flujo del servicio de Traducción de una palabra a imagen en LSE	50
5.7. Imagen devuelto por el servicio de traducción de la palabra “ <i>coche</i> ” a LSE	51
5.8. Flujo servicio de traducción de texto en castellano a texto en LSE	52
5.9. Información del token “Los niños meredaron chocolate” . . .	53
5.10. Displacy grafo de dependencias	54
5.11. Árbol de ls oración “Los niños merendaron chocolate”	55
5.12. Flujo servicio de traducción de texto en castellano a texto en LSE en función de los vídeos existentes	57
5.13. Flujo servicio de traducción de texto en castellano a texto en LSE en función de las imágenes existentes	59
5.14. Flujo servicio de traducción de texto en castellano a LSE en formato vídeo	61
5.15. Vídeo devuelto por el servicio de traducción del texto “ <i>Los niños comen chocolate</i> ” a LSE	62
5.16. Aplicación Web vista desde un dispositivo móvil	63

Capítulo 1

Introducción

En la sección 1.1 de este capítulo se explicará la motivación que hay detrás de la realización de este TFG, y en la sección 1.2 los objetivos que nos marcamos al comienzo del proyecto. Por otro lado, en la sección 1.3 se detallará la estructura que sigue el presente documento.

1.1. Motivación

En la sociedad actual en la que vivimos hay una gran cantidad de personas con distintas discapacidades: cognitivas, físicas, visuales, auditiva... En algunos casos, dichas discapacidades hacen que estas personas se sientan apartadas del resto de la sociedad por no tener acceso completo a ciertos servicios fundamentales, lo que dificulta su día a día. Gracias a los avances tecnológicos, se están desarrollando soluciones para ayudar a estos colectivos a acceder a distintos servicios y así integrarlos en la sociedad. Un ejemplo son las aplicaciones basadas en pictogramas (imágenes simples que representan un concepto), cuya finalidad es ayudar a comunicarse a personas con algún tipo de discapacidad cognitiva que les impide comunicarse usando el lenguaje natural.

En España, un 2,3 % de la población total (alrededor de un millón de personas) sufre algún tipo de discapacidad auditiva. Este colectivo usa distintos métodos de comunicación, que dependen tanto de la edad con la que empezaron a padecer sordera, como de su grado de pérdida auditiva. Las personas con discapacidad auditiva tienen las mismas necesidades de obtener información del entorno que el resto de la población, pero a pesar de disponer de las mismas capacidades cognitivas, se enfrentan a numerosas barreras comunicativas, que dificultan su proceso de aprendizaje y la capacidad de relacionarse con su entorno mediante la audición y la lengua oral. Esto se debe a que en muchas ocasiones el audio es el único canal para comunicar información, ya sea en películas, informativos, conversaciones cotidianas, ac-

tividades educativas, megafonía o vídeos.

El método alternativo al lenguaje oral más extendido entre las personas con discapacidad auditiva es la Lengua de Signos (LS). La LS no es universal, sino que varía en función de la región, es decir, no existe una lengua de signos común en todo el mundo, disponiendo cada país de una o varias. En España desde el año 2007 hay dos lenguas reconocidas oficialmente: la Lengua de Signos Española (LSE) y la Lengua de Signos Catalana (LSC). Es importante fomentar y facilitar el uso de la LS como lengua alternativa a la lengua oral, ya que hoy en día, en muchos casos, se cuenta con la posibilidad de añadir subtítulos al contenido audiovisual, pero no es suficiente. Los subtítulos distraen y se procesan de manera más lenta que la LS, que al ser ideográfica permite representar conceptos comunes con un solo gesto, permitiendo así expresar y asimilar información más rápidamente.

Contar con una herramienta capaz de traducir un texto a Lengua de Signos resultaría de gran ayuda para personas con discapacidad auditiva, ya que ofrecería la posibilidad de introducir la LS a cualquier audio que disponga de subtítulos, e incluso de traducir directamente cualquier audio mediante el apoyo de herramientas de traducción de voz a texto.

1.2. Objetivos

El objetivo principal del proyecto es crear una herramienta gratuita que sea capaz de traducir en tiempo real un texto en lenguaje natural escrito en castellano a la Lengua de Signos Española (LSE) en tiempo real. La aplicación creada permitirá incorporar la LSE a cualquier material audiovisual de manera sencilla, y además servirá de apoyo a las personas en proceso de aprendizaje de la LSE.

La aplicación estará basada en una arquitectura orientada a servicios (SOA), es decir, tendrá implementadas pequeñas funcionalidades alojadas en servicios web, siendo así fácilmente reutilizables por otros programadores que deseen integrar en sus aplicaciones las funcionalidades que vamos a desarrollar. Estos microservicios recibirán un texto en castellano y devolverán la traducción a LSE en varios formatos (video e imagen), haciendo uso de herramientas de Procesamiento de Lenguaje Natural (PLN) para ello.

En cuanto a los objetivos académicos, con este proyecto pretendemos poner en práctica los conocimientos adquiridos a lo largo del Grado en Ingeniería Informática, así como ampliar nuestros conocimientos y competencias.

1.3. Estructura del documento

En el Capítulo 2 se habla de la discapacidad auditiva y de los medios para comunicarse de los que disponen las personas con dicha discapacidad. También se da una visión más profunda de la Lengua de Signos, en concreto de la Lengua de Signos Española. Esto se complementa con un análisis de las herramientas que existen actualmente que ayudan a comunicarse y a aprender la LSE. En el Capítulo 3 se presentan las herramientas utilizadas para el desarrollo de la aplicación. En el Capítulo 4 se explica con detalle la metodología de trabajo seguida por los integrantes del grupo durante el desarrollo de este TFG. En el Capítulo 5 se explica en detalle el trabajo realizado. En el Capítulo 6 cada integrante del grupo explica el trabajo realizado a lo largo de todo el proyecto.

Capítulo 2

Estado del Arte

En este capítulo se presenta la información esencial acerca de la discapacidad auditiva y de la Lengua de Signos Española. En la sección 2.1 de este capítulo se introducen los tipos y características más importantes de la discapacidad auditiva. En la sección 2.2 se presentan las distintas vías de comunicación alternativa que utilizan las personas con discapacidad auditiva para relacionarse. A continuación, en la sección 2.3 se da una explicación detallada de qué es la Lengua de Signos y por qué es la vía de comunicación más usada por las personas con discapacidad auditiva. Posteriormente, en la sección 2.4 se muestran las soluciones de accesibilidad digital que existen para personas con discapacidad auditiva, así como tecnologías ya desarrolladas similares a la aplicación que se va a desarrollar en este TFG.

2.1. La discapacidad auditiva

La discapacidad auditiva es la dificultad que sufren algunas personas para percibir el sonido a través del oído. El colectivo de personas con déficit auditivo es muy heterogéneo, influyendo factores como la edad en la que aparece la sordera, el grado de ésta, así como factores de su entorno educativo y social. Existen distintas clasificaciones (?):

1. Según el grado de pérdida de audición:

- **Audición normal:** se perciben sonidos por encima de 20 decibelios.
- **Hipoacusia:** pérdida parcial de la audición.
 - **Leve:** no se perciben sonidos inferiores a 40 decibelios.
 - **Moderada:** se presentan pérdidas entre 40 y 70 decibelios

- **Severa:** pérdida de entre 70 y 90 decibelios. En este grado se requiere de ayudas auditivas, como prótesis o implantes. A partir de pérdidas de 75 decibelios la Seguridad Social considera al individuo persona sorda.
- **Sordera (Cofosis):** pérdida total de la audición. Se precisa de la ayuda de códigos de comunicación alternativa.

2. Según su etiología:

- **Genéticas:** factores hereditarios influyen en la pérdida de audición del individuo.
- **Adquiridas:** influyen factores externos como golpes o exposición a ruidos fuertes.
- **Congénitas:** la pérdida de audición está presente desde el nacimiento del individuo.

3. Según el momento de la aparición se distinguen:

- **Prelocutivas:** la sordera aparece antes de que el individuo haya aprendido el lenguaje oral. La gran mayoría de este colectivo no sabe ni leer ni escribir.
- **Postlocutivas:** la persona es capaz de comunicarse oralmente antes de la aparición de la discapacidad. En condiciones normales conoce el lenguaje escrito.

Muchas personas con discapacidad auditiva necesitan apoyar la comunicación oral con Sistemas Aumentativos y Alternativos de Comunicación (SAAC). En la siguiente sección se profundizará en estos métodos de comunicación y se explicarán con detalle los tipos que más se utilizan.

2.2. Sistemas Aumentativos y Alternativos de Comunicación (SAAC)

Los Sistemas Aumentativos y Alternativos de Comunicación (SAAC) (?) son un conjunto de recursos y formas de expresión distintas al lenguaje hablado, cuyo objetivo es facilitar la comprensión y la expresión del lenguaje a personas que tienen dificultades en la adquisición del habla y/o en la escritura. En los casos graves en los que la capacidad de expresión verbal es nula, estos sistemas se denominan sistemas alternativos. De esta forma, las personas con este tipo de dificultades pueden expresar sus deseos, intercambiar conocimientos, opiniones e incluso expresar su propia personalidad de

manera mucho más eficiente e inteligible para los demás, enriqueciendo así su campo de experiencia.

Las personas que más usan este tipo de sistemas son aquellas que presentan discapacidades motoras, las cuales carecen de un habla comprensible por los demás. También existen otros colectivos que requieren de estos sistemas, como pueden ser personas con discapacidad intelectual, cognitiva y física, así como discapacidades sensoriales, como la sordera.

Dependiendo de las necesidades de cada persona se pueden distinguir dos tipos de SAAC:

- **SAACs gestuales (?)**: No se apoya en ningún soporte físico (libros, dispositivos tecnológicos, etc), sino que se usa el propio cuerpo. Algunos ejemplos son los siguientes:
 - **Bimodal**: es el uso simultáneo de palabras articuladas y signos gestuales manteniendo la estructura sintáctica de la lengua oral.
 - **Oral signado**: es el uso simultáneo de palabras y signos manteniendo la estructura sintáctica de la lengua oral. La diferencia con el bimodal radica en que el bimodal signa preferentemente las palabras con contenidos semánticos, mientras que el oral signado signa de manera restrictiva todas y cada una de las palabras de la frase oral, hay paralelismo exacto entre palabras y signos.
 - **Lectura de labios**: es una técnica con la que una persona comprende lo que se le habla observando los movimientos de los labios de su interlocutor e interpretando los fonemas que éste produce.
 - **Palabra complementada**: es un sistema que posibilita la comunicación con personas sordas o con discapacidad auditiva mediante el uso simultáneo de la lectura de labios y una serie de gestos manuales que lo complementan.
 - **Dactilología**: Consiste en representar letras del alfabeto mediante formas manuales. Todas las lenguas de signos hacen uso de la dactilología en mayor o menor medida.
- **SAACs gráficos**: Se apoyan en un soporte físico que varía según las necesidades del individuo. Algunos ejemplos son:
 - **Braille**: es un sistema de lectura y escritura táctil, ya que los puntos que forman el sistema se encuentran elevados en la su-

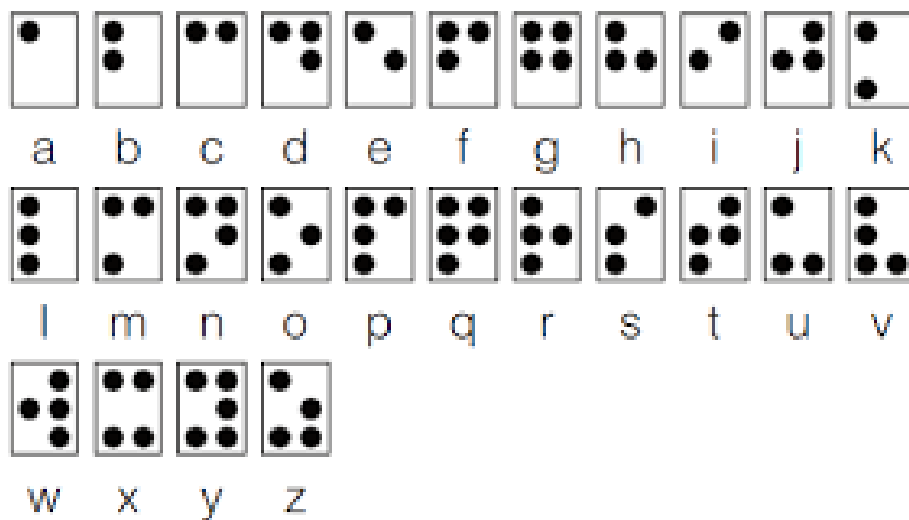


Figura 2.1: Representación del abecedario en el Sistema Braille

perficie. Es utilizado por las personas con discapacidad visual o ciega para poder escribir y leer *textos*, libros y documentos. En la Figura 2.1 se muestra el abecedario en Braille.

- **Pictogramas:** sistema que utiliza dibujos simples o símbolos para comunicarse de forma sencilla. Los símbolos son diseñados con el fin de representar las palabras y conceptos de uso más común. Existen diversos sistemas pictográficos, como por ejemplo MIC o ARASAAC, cuyos pictogramas son los más utilizados en España. En la Figura 2.2 se muestra la representación de varios elementos comunes mediante pictogramas.

2.3. La Lengua de Signos

La Lengua de Signos (LS) (?) es una lengua natural gestual que sirve para ayudar a integrarse a las personas con discapacidad auditiva o dificultad en el habla, y a personas que se quieran comunicar con ellas. La Lengua de Signos es considerada como SAAC gestual sin ayuda, es decir, no necesita apoyo de ningún sistema de información (imágenes, pictogramas).

La LS es una lengua rica y compleja gramaticalmente, es decir, no es una simple representación literal de la lengua oral. Además, es muy expresiva, ya que permite expresar sentimientos y emociones a la hora de comunicarse mediante el énfasis en los gestos y expresiones faciales.

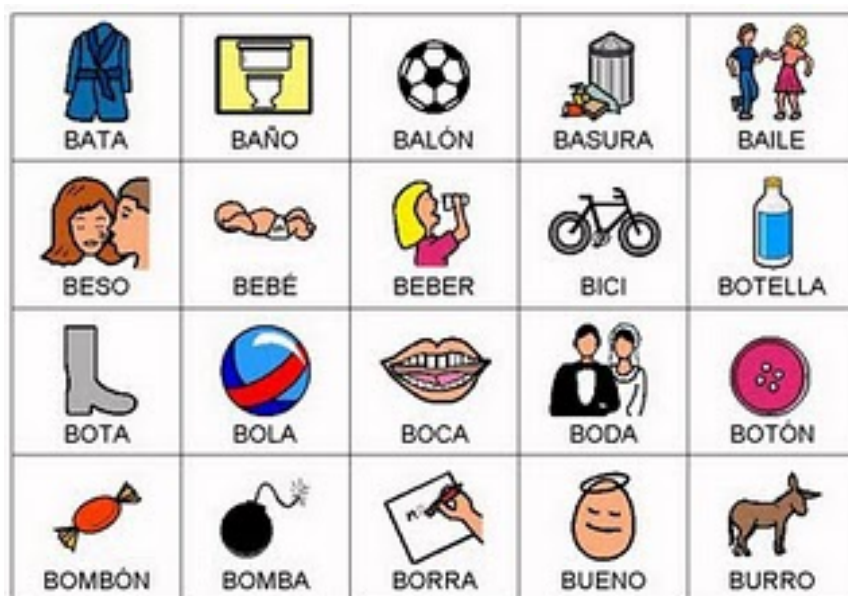


Figura 2.2: Representación de distintas palabras con pictogramas

El desarrollo de esta lengua necesita de unas capacidades tanto cognitivas como motrices, así como de un entrenamiento específico. Este sistema puede tener una doble finalidad:

- **Como elemento de comunicación:** Para hacer posible la comunicación de manera alternativa al habla o paliar las limitaciones que provoca la pérdida auditiva y así mejorar la integración social de las personas que sufren esta discapacidad.
- **Como elemento de desarrollo intelectual:** En personas sin ningún tipo de discapacidad auditiva, la capacidad lingüística es principalmente auditiva, siendo ésta la que más influye en la educación debido a que es la vía de adquisición del lenguaje. Para las personas que sí sufren de dicha discapacidad, el uso de estos sistemas es determinante en su desarrollo intelectual, sobre todo cuando el lenguaje verbal no esté adquirido.

La LS no es una lengua universal, es decir, no existe una Lengua de Signos común para todo el mundo ni para todos los idiomas. Cada país puede contar con una o varias LS oficiales y no son únicas para cada lengua. Esto quiere decir que dos países que comparten lengua oral oficial (como España y Argentina), tienen Lenguas de Signos totalmente diferentes. Incluso puede ocurrir que una misma LS presente diferencias dependiendo de en qué región del país se utilice.

En España hay alrededor de 1.064.000 personas mayores de seis años con algún grado de discapacidad auditiva, de las cuales el 1,25 % (13.300 personas) utilizan la Lengua de Signos para comunicarse, según los últimos datos aportados por el Instituto Nacional de Estadística (INE)¹. Esto se debe a que la mayoría de las personas con esta discapacidad son capaces de comunicarse a través del lenguaje oral, ya que su grado de discapacidad se lo permite y les ayuda a integrarse con el entorno debido a que no es común saber Lengua de Signos si no sufres de dicha discapacidad, siendo únicamente alrededor de 400.000 las personas que saben comunicarse mediante dicha lengua en nuestro país. Desde el año 2007 España cuenta con dos LS reconocidas oficialmente: la Lengua de Signos Española (LSE) y la Lengua de Signos Catalana (LSC), siendo la LSE la más utilizada en nuestro país.

La LSE es una lengua normativizada, es decir, consta de unas reglas que marcan el correcto uso de la misma, tal y como se muestra en la siguiente subsección.

2.3.1. Reglas de la Lengua de Signos Española

A continuación, se explica el conocimiento básico de las normas de la LSE en lo referente a la gramática, fonología, sintaxis y morfología.

2.3.1.1. Fonología de la Lengua de Signos Española

La fonología (?), en lo relativo a la LSE, se refiere al estudio de la estructura y organización interna de los signos.

Los signos de la LSE están formados por siete elementos esenciales:

- **Forma o configuración de la mano o manos** que intervienen en el signo. Es importante remarcar que no se utilizan los términos “mano izquierda” y “mano derecha”, ya que dependiendo de la persona, la mano dominante puede ser una u otra. Por eso se utilizan los términos “mano dominante” y “mano no dominante” para señalar con qué mano o manos signar. Para indicar qué mano es la dominante basta con levantarla y rotarla (tal y como se muestra en la Figura 2.3)
- **Orientación** de la mano o manos al signar respecto al cuerpo del individuo. Por ejemplo, gestos como “ayudar”, el cual podemos observar en la Figura 2.4, van acompañados de dirección para indicar a qué persona se refiere el emisor. En la Figura 2.5 se muestran las distintas direcciones que pueden acompañar a los gestos.

¹<https://www.ine.es/>



Figura 2.3: Representación de la configuración de la mano en Lengua de Signos



Figura 2.4: Signo “*ayudar*” en LSE del banco de imágenes ARASAAC

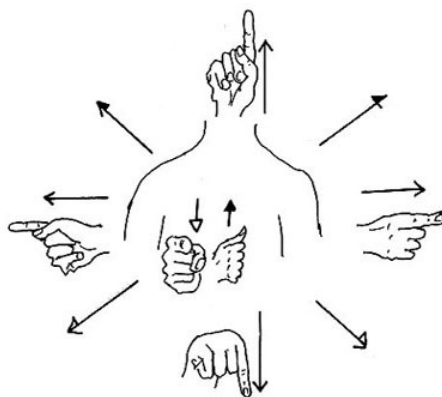


Figura 2.5: Representación de la orientación de las manos en Lengua de Signos

- **Lugar de articulación** del signo, es decir, el espacio en el que se realiza el movimiento. Puede ser ante el pecho, el hombro, la frente o los labios, entre otros. Por ejemplo, en la Figura 2.6, el gesto referente a la palabra “visera” se realiza ante la frente.
- **Plano** en el que se realiza el signo, es decir, la distancia de realización del signo con respecto al cuerpo del individuo. Sirve para indicar el tiempo verbal de los gestos. Los gestos signados más cerca del cuerpo indican pasado y lo más alejados futuro.
- **Punto de contacto** de la mano dominante con el cuerpo del individuo al realizar el signo. Por ejemplo, al signar “oir” el punto de contacto sería la oreja, tal y como se muestra en la Figura 2.7.
- **Movimiento** de las manos al realizar un signo. Puede ser giratorio, de vaivén o quebrado, entre otros. En la Figura 2.8 podemos observar la dirección y sentido con los que se realizan algunos de estos movimientos.
- **Componente no manual** del signo, que puede ser desde expresiones faciales hasta movimientos corporales del individuo. Por ejemplo, en la Figura 2.9, la intérprete quiere expresar una cantidad abundante, acompañando al signo de una expresión facial que enfatiza una gran cantidad.

2.3.1.2. Sintaxis de la Lengua de Signos Española

Las oraciones en la LSE no se estructuran igual que en castellano. La estructura básica de las oraciones se compone de sujeto-objeto-verbo, aunque existen excepciones. Por ejemplo, la oración “*Él come patatas*” en LSE se



Figura 2.6: Signo “*visera*” en LSE del banco de imágenes ARASAAC



Figura 2.7: Signo “*oir*” en LSE del banco de imágenes ARASAAC






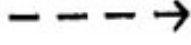


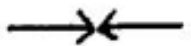
		
ondulado on	apulgado p	pinza pi
		
quebrado q	deslizamiento d	sacudidas s
		
vaivén v	repetición rep	choque ch

Figura 2.8: Ejemplos de tipos de movimientos realizados al signar en LSE



Figura 2.9: Signo “mucho” en LSE del banco de imágenes ARASAAC



Figura 2.10: Signo “*hombre*” del banco de imágenes ARASAAC

traduciría como “*Él patatas comer*”. Según se van incorporando elementos a las oraciones, la estructura se va volviendo más compleja siguiendo un orden temporal, es decir, las acciones se nombran en el orden en el que suceden. Por ejemplo, la oración “*Después de comer me fui a dormir*”, se traduciría a LSE como “*Yo comer fin dormir ir*”, indicando que primero se realizó la acción de comer y después la acción de irse a dormir.

2.3.1.3. Morfología de la Lengua de Signos Española

En la lengua oral, los morfemas de género sirven principalmente para marcar la concordancia entre el sustantivo y el adjetivo. En la LSE esto no existe, ya que tanto los adjetivos como los sustantivos son palabras invariables y además, no existen los artículos. Los elementos inanimados no tienen género en la LSE, por lo que no es necesario señalarlos, mientras que los elementos animados sí que precisan de distinción. En caso de que se quiera hacer énfasis en el género de alguna palabra, al terminar de signar la palabra en cuestión se añaden los signos de “*hombre*” (Figura 2.10) o de “*mujer*” (Figura 2.11).

En las Figuras 2.12 y 2.13 podemos observar cómo se añaden los signos “*hombre*” y “*mujer*” para marcar el género de las palabras “*abuela*” y “*abuelo*”.

Existen algunas excepciones, como el caso de los signos para “*niño*” y “*niña*”, los cuales tienen su propio signo para definir el género de la palabra. Podemos verlos en la Figura 2.14 y en la Figura 2.15 respectivamente.

Por otra parte, los morfemas de número en la LSE sirven únicamente para expresar cantidad. Existen varias formas de expresar cantidad según la



Figura 2.11: Signo “*mujer*” del banco de imágenes ARASAAC



Figura 2.12: Signo “*abuelo*” del banco de imágenes ARASAAC



Figura 2.13: Signo “*abuela*” del banco de imágenes ARASAAC



Figura 2.14: Signo “*niño*” del banco de imágenes ARASAAC



Figura 2.15: Signo “*niña*” del banco de imágenes ARASAAC



Figura 2.16: Signo “uno” del banco de imágenes ARASAAC

categoría gramatical de la palabra a la que acompañen:

■ **Sustantivos:**

- Si no queremos enfatizar el número o queremos expresar solo una unidad, se signa la palabra sin añadir nada a ella. Por ejemplo, la frase “*La puerta de la universidad es de color gris*” en LSE sería “*Universidad puerta color gris*”.
- Se pueden añadir palabras que expresen cantidad, como cuantificadores definidos, por ejemplo “uno”, cuyo signo podemos observar en la Figura 2.16, o cuantificadores indefinidos, por ejemplo “muchos”, cuyo signo podemos observar en la Figura 2.17.
- Se puede enfatizar la cantidad de un sustantivo mediante la expresión facial de la cual se puede distinguir si es mucha o poca la cantidad a la que se quiere referir el emisor. Por ejemplo, en la Figura 2.17 aparece una intérprete realizando el signo “muchos”. En ella podemos observar el gesto facial indicando una gran cantidad. Esta gesticulación puede sustituir al propio signo “muchos” en una oración si se realiza mientras se signa la oración.

■ **Adjetivos:** no varían en cuanto a número.

- **Verbos:** el plural se realiza principalmente mediante la repetición del verbo y recae sobre el objeto. Por ejemplo, en la oración “*Yo viajo a muchos sitios*” se repetiría el verbo viajar: “*Yo viajar viajar sitio*”.



Figura 2.17: Signo “muchos” del banco de imágenes ARASAAC

2.3.1.4. Verbos de la Lengua de Signos Española

El verbo en la LSE (?) presenta unos rasgos gramaticales propios que es necesario tener en cuenta. Se distinguen tres clases básicas de verbos en LSE:

- **Verbos planos:** Los verbos planos son aquellos cuyo signo no se modifica para marcar información complementaria, como por ejemplo el género, el número o a quién va dirigida la acción. Para añadir este tipo de información se añaden palabras independientes, como cuantificadores (uno, dos, etc), pronombres personales (yo, él, etc). Por ejemplo, *Querer*, *Comer*, *Pensar*, *Conducir*, etc. Podemos observar en la Figura 2.18 como el signo de la palabra “*Querer*” no aporta ningún tipo de información adicional por sí mismo.
- **Verbos de concordancia:** Los verbos de concordancia son aquellos que en la realización del propio signo se añade información adicional, como por ejemplo, a quién va dirigida la acción mediante la articulación del signo. Esto podemos observarlo en la Figura 2.4, que muestra el signo de la palabra “*ayudar*”, el cual termina señalando en el espacio a la persona a la que se está ayudando. Algunos ejemplos de verbos de concordancia son: *Avisar*, *Ayudar*, *Contar*, *Cuidar*, *Responder*, etc.
- **Verbos espaciales:** Los verbos espaciales son aquellos que no necesitan un signo específico para expresar su significado, ya que indican la situación en el espacio de algo o alguien a través de la dirección y la velocidad con la que se realiza el signo. Al igual que en los verbos planos, se puede añadir información adicional como género o número a través de palabras independientes, como cuantificadores. Por ejemplo, “*Hay un ratón ahí*” se traduce como “*Ratón (Señalando el sitio)*”. Esto



Figura 2.18: Signo “*querer*” en LSE del banco de imágenes ARASAAC



Figura 2.19: Signo “*señalar*” en LSE del banco de imágenes ARASAAC

podemos observarlo en la Figura 2.19, en la que se muestra el signo de la palabra “*señalar*”.

2.3.2. Bancos de videos e imágenes LSE

A continuación, se muestran diferentes bancos de vídeos e imágenes existentes de la Lengua de Signos Española.

2.3.2.1. ARASAAC

El Portal Aragonés de la Comunicación Aumentativa y Alternativa (ARASAAC)², es un proyecto desarrollado en el año 2007 por el gobierno de Aragón. Tiene como objetivo la creación de un sistema pictográfico de comuni-

²<http://www.arasaac.org/>



Figura 2.20: Signo “hola” en LSE del banco de imágenes ARASAAC

cación y un conjunto de herramientas de libre distribución. Estos recursos facilitan la accesibilidad de carácter comunicativo y cognitivo en diversos ámbitos de la vida a todas las personas que lo puedan requerir. ARASAAC proporciona un catálogo con más de 8.000 pictogramas en color, en blanco y negro, fotografías, así como un banco de vídeos e imágenes a color de signos de la Lengua de Signos Española. En la Figura 2.20 se muestra el signo “hola” en LSE del banco de imágenes de LSE de ARASAAC. Este contenido está disponible como contenido descargable con licencia Creative Commons, es decir, se puede usar libremente sin ánimo de lucro.

ARASAAC ofrece una API³ a través de la cual se pueden obtener sus pictogramas y los materiales generados con sus pictogramas a través de llamadas a los métodos de dicha API. Para obtener los recursos correspondientes a la LSE es necesario descargarlos a través del apartado de descargas de su página web⁴.

2.3.2.2. Banco de imágenes Fundacion CNSE

La Confederación Estatal de Personas Sordas (CNSE)⁵ es una organización que atiende los intereses de las personas sordas y sus familias en España. Es la primera entidad asociativa de la discapacidad de nuestro país, y desde su creación se ha ocupado de incentivar el desarrollo y la participación social de dicho colectivo. En ella se integran 17 federaciones de personas sordas, una por cada comunidad autónoma, que, a su vez, mantienen afiliadas a más de 120 asociaciones provinciales y locales de todo el Estado. No obstante, la CNSE atiende cualquier necesidad relacionada con el colectivo de personas

³<https://beta.arasaac.org/developers/api>

⁴<http://www.arasaac.org/descargas.php>

⁵<http://www.fundacioncnse.org/>



Figura 2.21: Recursos LSE de la Fundación CNSE

sordas, estén o no afiliadas a su movimiento asociativo.

En 1998, la CNSE constituye la Fundación CNSE para la Supresión de las Barreras de Comunicación. Se trata de una organización estatal sin ánimo de lucro, desde la que se impulsa la investigación y el estudio de la Lengua de Signos Española y se trabaja por mejorar la accesibilidad de las personas sordas en todos los ámbitos y se promueve el desarrollo de proyectos que mejoren la calidad de vida de las personas sordas y de sus familias.

Esta fundación cuenta con un banco de imágenes⁶ (ver Figura 2.21) y signos de la LSE, accesible a través de su página web. En esta página no existe la opción de descargar todas las imágenes de una vez, sino que es necesario hacerlo manualmente una a una, y no cuenta con un banco de videos de signos.

2.3.2.3. SpreadTheSign

SpreadTheSign⁷ es un diccionario online administrado por el Centro Europeo de Lenguas de Signos, que cuenta con más de 400.000 signos en vídeo (ver Figura 2.22) de diferentes Lenguas de Signos, como la española, estadounidense o alemana, entre muchas otras, así como de un alfabeto dactilológico para cada una de ellas y frases previamente almacenadas. Es una herramienta de autoaprendizaje gratuita, accesible a través de su página web y desde

⁶<http://www.fundacioncnse.org/educa/bancolse/>

⁷<https://www.spreadthesign.com/es.es/search/>



Figura 2.22: Imagen de la página web SpreadTheSign

su aplicación para Android⁸ e IOS⁹, aunque ninguna de estas aplicaciones permite la descarga de contenido.

2.4. Aplicaciones de Traducción de Texto a LSE

La variedad de recursos y aplicaciones orientadas a la integración social de las personas con discapacidad auditiva ha ido aumentando con el paso del tiempo. En esta sección se presentan aplicaciones con un objetivo similar al de nuestro TFG cuyo propósito es ayudar en la integración de las personas que sufren de discapacidad auditiva.

2.4.1. Text2Sign

Text2Sign¹⁰ es una aplicación tanto web como móvil gratuita que permite enviar un texto en formato PDF, imagen o Word a un equipo de intérpretes que se encarga de traducirlo a la LSE. Una vez realizada la traducción se puede descargar a través de la plataforma entre un periodo de 24 y 72 horas dependiendo de la longitud del texto ya que no es un servicio instantáneo y está limitado a una o dos páginas.

⁸https://play.google.com/store/apps/details?id=com.spreadthesign.androidapp_paid&hl=es

⁹<https://apps.apple.com/ni/app/spreadthesign/id438811366>

¹⁰<http://text2sign.es/>

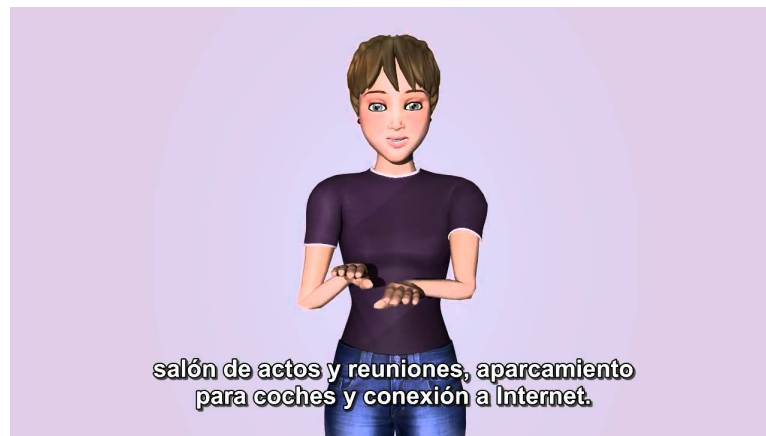


Figura 2.23: Maya, avatar usado por TextoSign.

2.4.2. TextoSign

TextoSign¹¹ es una herramienta software que funciona en páginas web, asistentes virtuales y dispositivos móviles que traduce en tiempo real un texto a la Lengua de Signos Española mediante un intérprete visual llamado Maya (Figura 2.23). Se trata de herramienta de pago y es utilizada hoy en día por La Caixa, Adif, Inturjoven y otras entidades.

2.4.3. StorySign

StorySign¹² es una aplicación para Android gratuita desarrollada por Huawei, que traduce en tiempo real una selección de cuentos infantiles a diferentes Lenguas de Signos como la española, italiana, francesa o inglesa. Para utilizarla es necesario abrir la aplicación y sostener el móvil encima del libro físico (especialmente diseñado para esta aplicación). Un avatar llamado Star interpreta el cuento en Lengua de Signos mientras la aplicación resalta cada palabra que Star interpreta como podemos ver en la Figura 2.24.

2.4.4. TeCuento

TeCuento¹³ es una aplicación Android gratuita que contiene diversos cuentos que se reproducen en video con subtítulos y LSE, como podemos apreciar en la Figura 2.25. No se traduce en el momento, son videos previamente grabados por personas físicas dando también la opción de poder escribir tu propio cuento y que lo traduzcan a LSE.

¹¹<http://textosign.es/>

¹²<https://play.google.com/>

¹³<https://play.google.com/>

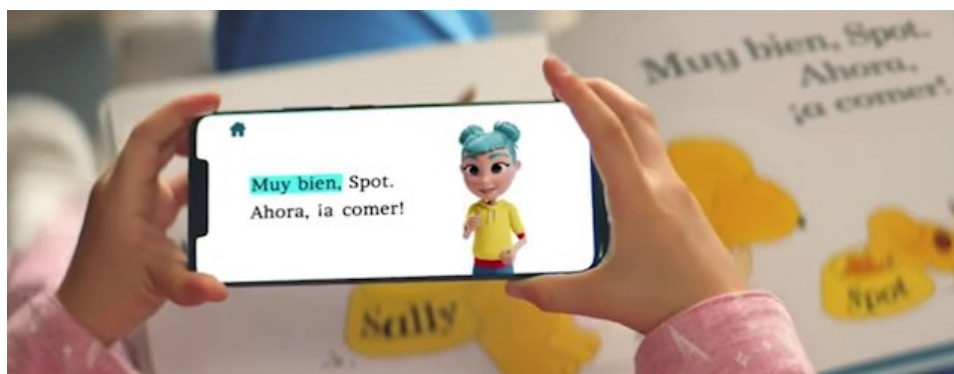


Figura 2.24: Imagen de la aplicación StorySign.



Figura 2.25: Imagen de la aplicación TeCuento.

2.4.5. Conclusiones

Como se ha podido comprobar existe un abanico amplio de aplicaciones que son capaces de traducir texto a LSE. Estas herramientas en su mayoría cumplen el propósito de facilitar el uso de la Lengua de Signos a las personas que quieran comunicarse con ella. No obstante, presentan algunas limitaciones como licencias de pago, tiempo de respuesta lento o disponibilidad en un solo tipo de dispositivo.

Nuestro proyecto Text2LSE pretende solventar muchas de estas limitaciones proporcionando una aplicación gratuita que sea accesible desde cualquier dispositivo, ya sea web o móvil. El objetivo de Text2LSE es ofrecer una traducción al instante de cualquier frase o palabra en castellano a LSE y la posibilidad de integrar nuestra herramienta mediante servicios web en aplicaciones que requieran su uso.

2.5. Procesamiento del Lenguaje Natural

El Procesamiento del Lenguaje Natural (PLN) es un campo dentro de la inteligencia artificial y la lingüística aplicada que analizan el lenguaje humano, lo interpretan y le dan significado para que pueda ser utilizado por una máquina de manera práctica.

Su evolución en los últimos años ha ido creciendo de forma exponencial debido a la capacidad computacional de la que disponemos hoy en día y los avances realizados en la algoritmia. Sus aplicaciones son muy variadas, ya sea desde traducción automática y chatbots hasta asistentes virtuales como Alexa de Amazon y Siri de Apple.

2.5.1. Estructura del Procesamiento del Lenguaje Natural

Para que una máquina sea capaz de entender y procesar el lenguaje natural, es necesario analizar la estructura de dicho lenguaje, teniendo en cuenta los siguientes aspectos:

- **Nivel Fonológico:** se encarga de analizar cómo las palabras se relacionan con los sonidos que representan.
- **Nivel Morfológico:** trata de cómo las palabras se construyen a partir de unas unidades de significado más pequeñas llamadas morfemas. Un ejemplo sería el morfema de género: *Chic-o (masculino)* y *Chic-a (femenino)*.
- **Nivel Sintáctico:** es el encargado de cómo se pueden unir las palabras para formar oraciones sintácticamente correctas, donde el orden de

las palabras es el necesario para que se establezca la comunicación. Un ejemplo sería la concordancia entre sujeto y verbo ya que estos comparten siempre misma persona y número.

Yo bebo agua: Yo (1ª per, Sg) y bebo (1ª per, Sg)

El bebe agua: Él (3ª per, Sg) y bebe (3ª per, Sg)

- **Nivel Semántico:** estudio del significado de las expresiones del lenguaje y de cómo los significados de las palabras se unen para dar sentido a una oración. Un ejemplo sería: *“Pedro habla la Lengua de Signos”*.
- **Nivel Pragmático:** trata de cómo las oraciones se usan en distintas situaciones y de cómo el uso afecta al significado de las oraciones.

El Procesamiento del Lenguaje Natural en este proyecto es imprescindible ya que como se ha indicado en el apartado 2.3.1.2 la LSE no es una simple representación del castellano. Para traducir el castellano a LSE hay que realizar un procesamiento previo del texto y obtener todos los datos gramaticales para así poder realizar una traducción que cumpla las reglas de la Lengua de Signos Española.

Capítulo 3

Tecnologías utilizadas

En este capítulo se habla de las tecnologías utilizadas a lo largo del desarrollo de nuestra aplicación, dando una breve descripción de cada una y comentando los motivos de su utilización con respecto a otras tecnologías.

3.1. Servicios Web

Según el W3C (World Wide Consortium)¹ un servicio web es un sistema software diseñado para soportar interacciones entre sistemas a través de la red. Proporcionan una forma estándar de interoperar entre aplicaciones software que se pueden ejecutar en diferentes plataformas, soportando aplicaciones desarrolladas en distintos lenguajes de programación.

Muchos de los servicios web están basados en la arquitectura SOA (Service Oriented Architecture). Es un estilo arquitectónico para la construcción de aplicaciones software en base a servicios disponibles los cuales interactúan entre sí y se pueden combinar obteniendo nuevos servicios con más funcionalidades.

Los dos tipos de servicios web más utilizados en la actualidad son: Servicios Web SOAP y Servicios Web REST.

3.1.1. Servicios Web SOAP

Los Servicios Web SOAP (?) son un tipo de servicios web cuyo propósito es que las aplicaciones implementadas en diferentes plataformas y lenguajes de programación sean capaces de intercambiar datos de una forma más sencilla.

¹<https://www.w3.org/>

Está basado en el protocolo SOAP (Simple Object Access Protocol) basado en el lenguaje XML el cual permite el intercambio de información entre aplicaciones. El mensaje se codifica como un documento XML que consta de un elemento *<Envelope>* que hace referencia a la raíz de cada mensaje y este a su vez incorpora una cabecera opcional *<Header>* y un cuerpo *<Body>* obligatorio. La primera es utilizada para pasar información de la aplicación y la segunda contiene la información dirigida al destinatario final del mensaje. Dentro del *<Body>* está el subelemento *<Fault>* que sirve para la notificación de errores.

Funciona por lo general con el protocolo HTTP, sin embargo no está limitado solo a este protocolo, si no que puede ser enviado por otros tipos de protocolos diferentes como TCP o POP3.

Para poder acceder al servicio web se necesita saber donde está ubicado y cuales son las funcionalidades del servicio. Para ello se utiliza el lenguaje de descripción WSDL (Web Services Description Language) el cual está basado en XML y permite especificar a los clientes qué llamadas se pueden hacer a un servidor SOAP y que tipo de respuesta esperar. A pesar de tener el archivo WSDL es necesario saber dónde está toda esta información. Para ello se dispone del UDDI (Universal Description, Discovery and Integration Directory) que es un estándar basado en XML que actúa como repositorio donde se puede acudir a realizar búsquedas de Servicios Web específicos.

3.1.2. Servicios Web REST

Un Servicio Web REST (?) es una interfaz para conectar varios servicios web basados en el protocolo HTTP que define una gran cantidad de métodos como son GET, POST, PUT y DELETE entre otros, los cuales pueden ser usados en diferentes circunstancias devolviendo los datos en distintos formatos como XML y JSON.

Rest tiene como peculiaridad que es un servicio sin estado, es decir, que por cada petición que recibe el servidor, los datos se pierden ya que no es necesario mantener sesiones. Esto permite un uso más eficiente de la memoria y una mayor escalabilidad permitiendo separar el cliente del servidor, facilitando que las distintas partes de un proyecto se puedan realizar de manera más independiente.

Los objetos son manipulados a través de una URI (Uniform Resource Identifier) haciendo de identificador único de cada recurso del sistema REST. Esto junto con los métodos del protocolo HTTP hace posible la transferen-

cia de datos en el sistema aplicando operaciones concretas sobre un recurso determinado.

Para el desarrollo de este proyecto, se ha utilizado un servicio web REST, debido a su facilidad de uso y configuración.

3.1.3. Ventajas de Servicios Web

Las principales ventajas de los servicios web son²:

- Permiten una mayor operabilidad entre las aplicaciones software sin importar sus propiedades ni la plataforma en la que estén instalados.
- Fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.
- Permiten y facilitan que software y servicios de diferentes compañías se puedan combinar formando un servicio funcional.

3.1.4. Desventajas de Servicios Web

Las principales desventajas de los servicios web son³:

- Las transacciones en este tipo de servicios están mucho menos desarrolladas que otros estándares abiertos.
- Su rendimiento es bajo si se compara con otros modelos de computación distribuida.
- Al estar muy ligados con HTTP, las medidas de seguridad que tratan de bloquear la comunicación entre programas se pueden burlar fácilmente.
- Existe poca información de servicios web para algunos lenguajes de programación.

3.2. Flask

Para el desarrollo del proyecto se ha decidido utilizar un servidor Flask para implementar los microservicios. Flask (?) es un “micro” framework escrito en Python y concebido para facilitar el desarrollo de Aplicaciones Web y APIs. La palabra micro hace referencia a que Flask únicamente trae por defecto las herramientas necesarias para crear una aplicación web básica, aunque si se necesitan añadir nuevas funcionalidades hay un conjunto muy

²<https://sites.google.com/site/preyctodetics/home/servicios-web>

³<https://sistemas3.wordpress.com/2007/06/14/web-services/>

grande de extensiones (plugins) que se pueden instalar fácilmente. Por ello, Flask es muy recomendable para el desarrollo de aplicaciones que no requieran muchas extensiones o que se necesiten implementar de una forma ágil y rápida. También es muy recomendable para implementar microservicios.

Algunas de las características de Flask por las que decidimos desarrollar nuestros microservicios con este framework son las siguientes:

- Rapidez y facilidad en la instalación y configuración, a diferencia de otros frameworks como Django, que son más complejos y difíciles de utilizar.
- Es compatible con Python. Nuestros microservicios están implementados en dicho lenguaje.
- Incluye un servidor web de desarrollo. No se necesita una infraestructura con un servidor web para probar las aplicaciones, sino que de una manera sencilla se puede levantar un servidor web para ir viendo los resultados que se van obteniendo.
- Cuenta con depurador. Si tenemos algún error en el código que se está desarrollando, se puede depurar ese error y ver los valores de las variables.
- Flask es Open Source y está amparado bajo una licencia BSD (?), que es la licencia utilizada para los sistemas operativos BSD (Berkeley Software Distribution), y tiene menos restricciones en comparación con otras licencias como la GPL, estando muy cercana al dominio público.

3.3. Herramientas PLN

En la actualidad hay una gran cantidad de herramientas para poder realizar el Procesamiento de Lenguaje Natural de una forma más sencilla y rápida. A continuación, vamos a explicar algunas de estas herramientas de Procesamiento de Lenguaje Natural que existen actualmente.

3.3.1. Spacy

Spacy⁴ es una librería de código abierto para realizar Procesamiento de Lenguaje Natural en Python. Permite el desarrollo de aplicaciones que sean capaces de procesar y entender grandes volúmenes de texto obteniendo un análisis completo del lenguaje. Actualmente es considerada una de las mejores herramientas para el PLN ya que proporciona modelos en nueve idiomas

⁴Web oficial de Spacy <https://spacy.io/>

distintos en los cuales está incluido el español y destaca por su rapidez y precisión a la hora de realizar análisis sintácticos comparado con otras librerías.

3.3.2. NLTK

Natural Language ToolKit (NLTK)⁵ es una librería dedicada al análisis del lenguaje natural en Python. En la actualidad incluye utilidades para más de 10 idiomas y fue creada por académicos e investigadores como una herramienta para crear funciones complejas de PLN. Esta biblioteca es muy útil cuando se quiere construir un modelo desde cero ya que proporciona acceso a muchos algoritmos y facilita el desarrollo de algoritmos propios permitiendo realizar un procesamiento del lenguaje más adaptado a un proyecto en concreto.

3.3.3. FreeLing

FreeLing⁶ es una herramienta PLN de código abierto bajo el lenguaje de programación C++ que ha sido desarrollada por la Universidad Politécnica de Cataluña. La biblioteca permite realizar análisis morfológico, detección de entidades con nombres o etiquetado gramatical para una gran variedad de idiomas.

Los desarrolladores pueden usar los recursos por defecto de FreeLing, ampliarlos, adaptarlos a dominios particulares e incluso desarrollar otros nuevos para idiomas específicos o necesidades particulares.

3.3.4. Conclusiones

Una vez analizada la posibilidad de usar una u otra se ha llegado a la conclusión de que Spacy es la herramienta adecuada, ya que proporciona un análisis rápido y bastante preciso en Español lo cual es de gran importancia en este proyecto y además es muy sencilla de aprender a manejar gracias a su manual de uso.

3.4. Nginx

Nginx⁷ es un servidor web de alto rendimiento, capaz de trabajar junto con diversas tecnologías de desarrollo y lenguajes. La asincronía es una de sus características fundamentales, junto con su rapidez, debido a que es un servidor web ligero. A día de hoy, además de sus capacidades de servidor

⁵Web oficial de NLTK <https://www.nltk.org/>

⁶Web oficial de FreeLing <http://nlp.lsi.upc.edu/freeling/index.php/node/1>

⁷Web oficial de Nginx <https://www.nginx.com/>

HTTP, NGINX también puede funcionar como un servidor proxy para correo electrónico (IMAP, POP3 y SMTP) y un proxy inverso y equilibrador de carga para servidores (HTTP, TCP y UDP).

Las ventajas de Nginx y el porqué de su uso en este TFG son:

- Nginx es un software multiplataforma que se puede usar en la gran mayoría de los sistemas operativos, tanto en sistemas basados en Unix como en Windows.
- Consume menos recursos que la mayoría de servicios que hacen su misma función ya que hace un uso de memoria de forma estática y se mantiene bastante estable independientemente del volumen de tráfico.
- Proporciona un alto rendimiento sobre todo en casos de mucho tráfico.
- Puede ser usado como Proxy inverso cacheando el contenido de nuestros sitios web.

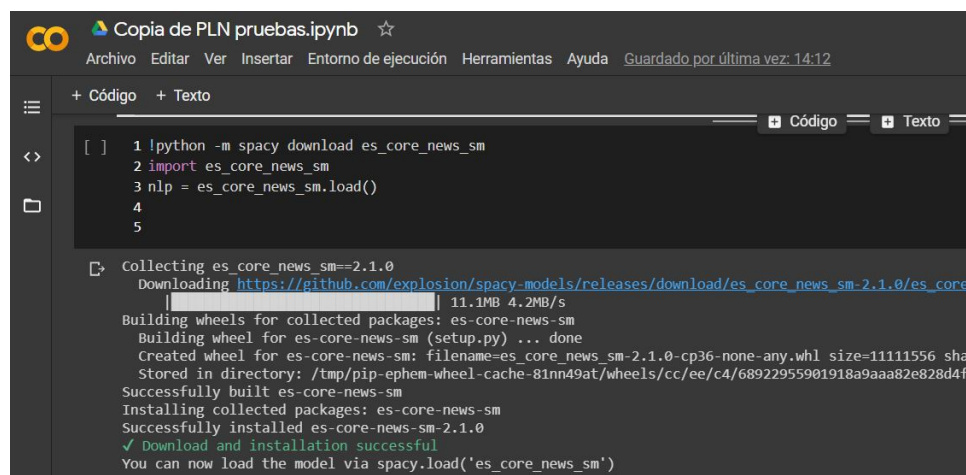
Se ha utilizado un servidor Nginx para alojar una aplicación web cuya finalidad es mostrar la funcionalidad de los diferentes microservicios implementados en el proyecto. Además, se ha configurado Nginx como servidor proxy inverso, redireccionando las peticiones según correspondan a la página web alojada en el propio Nginx o a los microservicios desarrollados alojados en el servidor Flask.

3.5. Google Colaboratory

Google Colaboratory⁸ es una herramienta gratuita de Google que permite programar en Python y ejecutar el código en tiempo real. El usuario final no necesita instalar nada en su propio sistema, ya que se pueden importar las librerías necesarias y toda la carga del procesamiento se localiza en los servidores de Google.

Con esta herramienta se crean documentos llamados “cuadernos de Colaboratory”, los cuales contienen celdas independientes en las que se puede introducir código ejecutable, texto, imágenes, comandos de la shell de Linux, código HTML, LaTeX, etc. Todos estos cuadernos son almacenados en la cuenta de Google Drive del usuario, haciendo que compartirlo con otras personas sea muy rápido y sencillo. Podemos ver un ejemplo de un cuaderno en la Figura 3.1

⁸<https://colab.research.google.com/notebooks/intro.ipynb>



```
[ ] 1 !python -m spacy download es_core_news_sm
    2 import es_core_news_sm
    3 nlp = es_core_news_sm.load()
    4
    5
```

```
Collecting es_core_news_sm==2.1.0
  Downloading https://pypi.org/project/es-core-news-sm/2.1.0/es_core_news_sm-2.1.0-cp36-none-any.whl (11.1MB)
    11.1MB 4.2MB/s
Building wheels for collected packages: es-core-news-sm
  Building wheel for es-core-news-sm (setup.py) ... done
  Created wheel for es-core-news-sm: filename=es_core_news_sm-2.1.0-cp36-none-any.whl size=11111556 sha256=81nn49at/wheels/cc/ee/c4/68922955901918a9aaa82e828d4f
  Stored in directory: /tmp/pip-ephem-wheel-cache-81nn49at/wheels/cc/ee/c4/68922955901918a9aaa82e828d4f
Successfully built es-core-news-sm
Installing collected packages: es-core-news-sm
Successfully installed es-core-news-sm-2.1.0
✓ Download and installation successful
You can now load the model via spacy.load('es_core_news_sm')
```

Figura 3.1: Vista de un cuaderno de Google Colaboratory

Una de las opciones más llamativas de Colaboratory es la opción de poder procesar el código que necesite gran potencia a través de un acelerador, en este caso consiste en una GPU proporcionada por Google.

Para este proyecto se ha utilizado esta herramienta para realizar pruebas con PLN utilizando Spacy y NLTK de manera más rápida y sencilla, teniendo acceso los tres miembros del equipo en tiempo real a dichas pruebas. En este caso no ha sido necesario la utilización de la GPU ya que la ejecución del código generado no ha sido tan potente como para necesitarla. Una vez que las pruebas eran satisfactorias, se comenzaba el traspaso de código a la API del proyecto.

Capítulo 4

Metodologías utilizadas

En este capítulo se habla de la metodología de trabajo seguida por el equipo de este TFG durante el desarrollo del mismo, junto con algunas de las herramientas utilizadas para organizar y estructurar las tareas a realizar y mantener el código de la aplicación accesible y con sus correspondientes copias de seguridad.

4.1. Metodologías de gestión del proyecto

Al comienzo de este TFG, los integrantes del proyecto desconocíamos tanto su alcance como su viabilidad, por lo que necesitábamos implantar una metodología que nos permitiera adaptar nuestro trabajo a necesidades que fueran surgiendo a medida que íbamos avanzando. Debido a esto, decidimos seguir una metodología de trabajo ágil.

Las metodologías ágiles (?) son aquellas que permiten adaptar la forma de trabajo a las condiciones particulares de cada proyecto, consiguiendo flexibilidad e inmediatez en la respuesta para amoldar el proyecto y su desarrollo a las circunstancias específicas del entorno.

Nuestros tutores han tomado el papel de Product Owner y han estado involucrados de manera muy activa a lo largo de todo el proyecto. A lo largo de todo el proyecto se les iba informando de los logros y progresos del mismo para optimizar el resultado final, obteniendo en todo momento una visión completa de su estado. La continua interacción entre los alumnos y los tutores tenía como objetivo asegurar que el resultado final sea exactamente lo que se busca y necesita. Además, de este modo se detectaban de forma rápida tanto errores como problemas que pudieran aparecer a lo largo del proyecto, por lo que es posible dar respuesta a todos aquellos contratiempos que puedan darse desde el inicio. Todo esto fue posible gracias a que estábamos en permanente contacto a través del correo electrónico para poder

resolver dudas puntuales, y a las constantes reuniones que se realizaron tanto entre los integrantes del grupo como entre los alumnos y los tutores. Estas reuniones se explican con más en detalle en la sección 4.3.

A lo largo de todo el TFG se fueron realizando entregas parciales del proyecto. De este modo, se entregaba en el menor intervalo de tiempo posible una versión funcional de la aplicación y de la memoria, para que los tutores pudieran revisar el trabajo realizado y comprobar que íbamos por buen camino.

La herramienta elegida para gestionar el trabajo realizado ha sido el conocido controlador de versiones Git, del que se habla con detalle en la sección ??

En nuestro proyecto seguimos algunas de las reglas de la metodología Kanban. Kanban (?) es una metodología ágil cuyo objetivo es gestionar de manera visual cómo se van completando las tareas. Kanban es una palabra japonesa que significa 'tarjetas visuales', donde Kan es 'visual', y Ban corresponde a 'tarjeta'.

En Kanban existen tres reglas (?) que ayudan a obtener el máximo rendimiento del flujo de trabajo. Dichas reglas son las siguientes:

- **Visualizar el flujo de trabajo:** una visualización de todas las tareas contribuye a que todos los miembros del equipo se mantengan al corriente del trabajo. Todos los miembros del equipo trabajan en el mismo tablero y colaboran en tiempo real. Esta metodología se basa en la implementación de un tablero que hará uso de una serie de tarjetas, que describirán las tareas, y que constará de varias columnas, una por cada estado de dichas tareas. La elección de los estados depende de las necesidades del proyecto, aunque los tres estados más comunes son "To Do", "Doing" y "Done". Según vayan surgiendo nuevas tareas, los miembros del equipo colocarán las tarjetas en la columna que corresponda, y las irán moviendo de columna a medida que el estado de la tarea avance.

Desde el comienzo, una de las mayores ventajas de la utilización de Kanban es la mejora de la implicación de los integrantes del equipo. Esta metodología permite a todos los miembros conocer el estado del proyecto en cualquier momento, y facilita tanto el reparto de tareas como que los compromisos sean acordados y aceptados por todos. Además, Kanban destaca por ser una técnica de gestión de las tareas muy visual, que permite ver a golpe de vista el estado de los proyectos, así como también pautar el desarrollo del trabajo de manera efectiva.

La herramienta utilizada para implementar el tablero Kanban ha sido Trello, explicada con detalle en la sección 4.2, y donde se explica en profundidad la forma en la que el equipo ha utilizado esta metodología de trabajo.

- **Limitar la cantidad de Trabajo en Proceso: WIP(?)**, o “Work In Progress”, hace referencia a la cantidad de tareas en las que el equipo está trabajando, y se utiliza para limitar el número de tareas en cada columna del tablero. Es importante establecer metas asequibles y limitar los trabajos en proceso para prevenir el exceso de cantidad de tareas, lo que podría originar “cuellos de botella” o incluso bloqueos. Los límites WIP aseguran que el equipo mantiene un ritmo de trabajo óptimo, sin exceder en su capacidad de trabajo.
- **Medir el flujo de tareas:** para medir el flujo de tareas, en kanban se utilizan los términos “tiempo de entrega”, o “leadtime”, y “tiempo de ciclo”, o “cycletime”.
 - **Leadtime:** es el periodo de tiempo que transcurre entre la aparición de una nueva tarea y su finalización.
 - **Cycletime:** es el periodo de tiempo que transcurre entre que alguien del equipo comienza a trabajar en una tarea hasta que se completa.

4.2. Trello

A lo largo de todo el proyecto se ha utilizado la herramienta Trello¹ para implementar el tablero Kanban, y así visualizar las tareas y su estado de manera clara y en tiempo real. Trello permite crear tableros con distintos estados, a los cuales se les pueden añadir tareas. Esta herramienta permite ver de manera sencilla el estado del proyecto, facilitando así la organización de las tareas a realizar.

Durante nuestro TFG se ha utilizado un único tablero, al que se le ha llamado “Text2LSE”, en el que se han englobado todas las tareas referentes a la investigación, desarrollo y pruebas del proyecto. Se ha diferenciado el tipo de tarea a través de los colores de cada tarjeta:

- **Azul:** Tareas de desarrollo del código de la aplicación web.
- **Rojo:** Tareas de desarrollo del código de los servicios web.

¹<https://trello.com/>

- **Verde:** Tareas relacionadas con la memoria.

Para una mayor organización, el tablero se ha estructurado en cuatro estados:

- **Lista de Tareas:** Lista de tareas a realizar.
- **En proceso:** Tareas en proceso de desarrollo. Estas tareas están asignadas a un miembro del equipo.
- **En revisión:** Tareas desarrolladas, pero pendientes de realizar pruebas para comprobar su correcto funcionamiento. Respecto a las tareas correspondientes al código, se realizan pruebas para ver que funcione el nuevo desarrollo correctamente sin que el resto de código realizado previamente deje de funcionar. Respecto a las tareas de memoria, se revisa cada nueva aportación por el resto de los compañeros.
- **Hecho:** Tareas completadas y comprobadas.

4.3. Reuniones

Para una buena organización, desde el principio el equipo de trabajo se ha reunido una vez a la semana para asignar las diferentes tareas, comentar el trabajo realizado y poner en común todos los conocimientos adquiridos por cada uno. A lo largo de la semana también había reuniones a través de Skype para comentar y resolver las posibles dudas que podrían surgir durante el proceso.

El equipo también se ha mantenido en contacto vía email con los tutores comentando el estado del proyecto. A esto se añadieron reuniones mensuales con ellos para realizar correcciones de la memoria, revisar el progreso, resolver dudas y comentar las tareas necesarias para las siguientes fases.

De esta forma, los integrantes del equipo y los tutores han estado informados constantemente del progreso del trabajo, se ha podido hacer un reparto claro de tareas y se han podido establecer tiempos de entrega realistas y factibles.

A partir de marzo, debido a las situación causada por el Covid-19, se tuvieron que suprimir las reuniones presenciales. Se han mantenido las reuniones semanales de los miembros del equipo, pero a distancia a través de Skype. También se ha mantenido el contacto vía email con los tutores, y las reuniones con ellos se han realizado a través de Google Meet.

4.4. Control de versiones

Para realizar la gestión del proyecto se ha decidido usar la herramienta GitHub que es una plataforma basada en el sistema de control de versiones Git que permite ver y llevar el registro de todos los cambios realizados en el proyecto, organizarlos y resolver cualquier conflicto que pueda surgir.

4.4.1. Estructura de Trabajo

Debido a la necesidad de trabajar los tres miembros del equipo de forma independiente, se ha decidido estructurar las ramas de trabajo de forma que se impida interferir en el trabajo de los demás compañeros. Por este motivo, se han creado tres ramas principales como se puede ver en la Figura 4.1: Memoria, API, y Web. Desde cada una de estas ramas se crearán ramas hijas según las funcionalidades a desarrollar, lo que permite que un miembro del equipo pueda trabajar sobre ella sin necesidad de modificar el resto del proyecto. Una vez finalizado el trabajo sobre esa rama hija, esta podrá eliminarse una vez realizada la correspondiente actualización sobre la rama padre. A continuación se detalla la estructura de las ramas de trabajo:

- **MASTER:** Rama principal del proyecto donde todo ha sido revisado. De ella dependen:
 - **Memoria:** Rama principal de la memoria que contiene la versión revisada más reciente. De ella se crean nuevas ramas hijas por capítulo y funcionalidad para poder trabajar cada uno en su parte de la memoria dentro de un mismo capítulo:
 - Memoria-Capítulo-N: *Memoria-Capítulo3*
 - ◇ Memoria-Capítulo-N-funcionalidad: *Memoria-Capítulo3-Flask*
 - **API:** Rama principal de la API que contiene el código más reciente y en funcionamiento. Contiene dos ramas hijas:
 - API-Video: Rama donde se desarrolla toda la parte de procesamiento de vídeo. A partir de ella se crean ramas hijas por funcionalidad a desarrollar.
 - ◇ API-Video-funcionalidad: *Api-Video-unaPalabra*
 - API-PLN: Rama donde se desarrolla el procesamiento de lenguaje natural. A partir de ella se crean ramas hijas por funcionalidad a desarrollar.
 - ◇ API-PLN-funcionalidad: *API-PLN-sujeto*

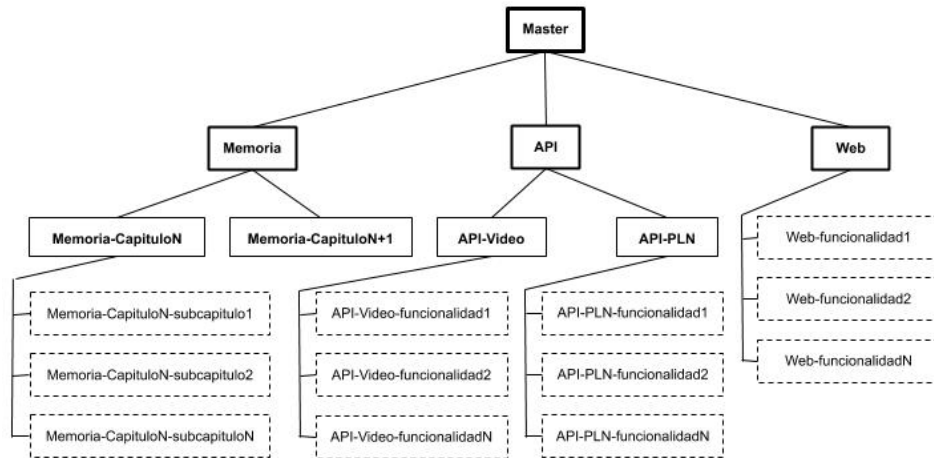


Figura 4.1: Estructura “Git” utilizada en el proyecto.

- **Web:** Rama principal del cliente que contiene el código más reciente y en funcionamiento. A partir de ella se crean ramas hijas por funcionalidad a desarrollar:
 - Web-funcionalidad: *Web-llamadaFetch*

Capítulo 5

Text2LSE

En este capítulo se muestra el desarrollo que se ha realizado a lo largo de este proyecto. Text2LSE es una aplicación web (ver Figura 5.1) que permite traducir textos escritos en castellano a LSE en formato vídeo y en formato imagen en tiempo real. Está basada en servicios web, los cuales están disponibles para todo el mundo de manera gratuita. Los vídeos e imágenes utilizados en la traducción a LSE son los recursos ofrecidos en el catálogo de LSE de ARASAAC.

En el apartado 5.1 se muestra la arquitectura de la aplicación web y cómo se comunica ésta con los servicios web. En el apartado 5.2 se explican en profundidad los servicios web desarrollados, tanto los utilizados en la aplicación web, como los desarrollados para aumentar las posibilidades de obtención de información para futuros desarrolladores. A continuación, en el apartado 5.3 se detalla más a fondo el desarrollo de la página web, tanto del diseño como de su funcionalidad.

La página web desarrollada es pública, y se puede acceder desde la siguiente url:

```
https://holstein.fdi.ucm.es/tfg-text2lse
```

5.1. Arquitectura

Text2LSE es una aplicación de traducción de castellano a LSE basada en servicios web, de manera que el código desarrollado sea fácilmente reutilizable. La arquitectura utilizada para el desarrollo de este proyecto es la arquitectura de cliente-servidor, es decir, un cliente muestra una web, la cual hace peticiones al servidor, que almacena los servicios web y devuelve

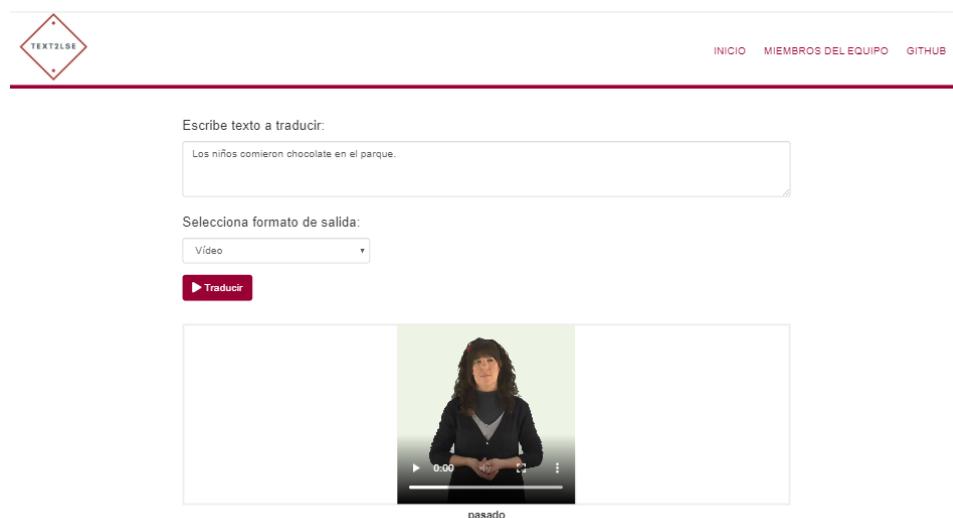


Figura 5.1: Aplicación web Text2LSE

la respuesta al cliente. Podemos ver un esquema de la arquitectura de este proyecto en la Figura 5.2.

Se ha optado por desarrollar una aplicación web debido a que ésta es accesible desde cualquier dispositivo que disponga de un navegador, ya sea un teléfono móvil, una tablet o un ordenador, sean de la marca y tamaño que sean. Para que se pueda ver de manera correcta en todos esos dispositivos, la aplicación web se ha desarrollado siguiendo un diseño responsive, es decir, que su visualización se adapte dependiendo de las dimensiones de la pantalla del dispositivo desde el cual se esté accediendo.

Respecto a la parte del servidor, se han desarrollado tres servicios web distintos, uno para devolver de manera rápida el vídeo en LSE de una sola palabra, otro para devolver el texto traducido a texto en LSE y otro para devolver el texto traducido a video en LSE.

Se ha utilizado un proxy inverso para poder acceder tanto a la página web como a los servicios web a través desde un mismo punto de acceso. Un Proxy inverso (?) es un método de redireccionamiento del tráfico a partes específicas de una infraestructura concreta. Las principales finalidades para las que se usa este tipo de servidores son:

- **Anonimización:** el proxy recibe todas las llamadas al servidor y se encarga de filtrarlas y redirigirlas como se haya configurado previamente. De esta manera, desde fuera del servidor no se va a poder obtener ningún tipo de información del servidor ni de los servicios que estén en

él, solo se podrá obtener información del proxy.

- **Protección y cifrado:** al utilizar un proxy inverso se tiene la posibilidad de instalar sistemas de control y filtros de paquetes que protegen al servidor, aumentando así la seguridad del sistema.
- **Balanceo de carga:** permite redirigir las distintas solicitudes entrantes por varios servidores, permitiendo repartir la carga de trabajo para no sobrecargar ningún servidor o equilibrar la carga en el caso de que falle uno de ellos.
- **Caché:** el proxy se puede configurar para que sea capaz de almacenar las respuestas del servidor temporalmente para ofrecer una mayor velocidad de respuesta. De esta forma, si se recibe una solicitud cuya respuesta la tiene almacenada el proxy en su caché, se manda la respuesta de manera inmediata haciendo que no reciba tanta carga de procedimiento el back-end.
- **Compresión:** un proxy inverso también se puede utilizar como compresor de datos, tanto entrantes como salientes.

En este proyecto, se ha configurado el servidor Nginx como servidor proxy inverso con el fin de redirigir las distintas solicitudes al servidor correspondientes. Se han especificado una serie de rutas para diferenciar qué llamadas de usuario redireccionar a la página web y cuáles a los servicios web. Esta estructura se puede observar en la Figura 5.3.

5.2. Back-End

En esta sección se explican con detalle los servicios web desarrollados y su implementación. La API donde están implementados estos servicios es pública y en los siguientes apartados se indica la URL para acceder a cada servicio.

```
https://holstein.fdi.ucm.es/tfg-text2lse
```

5.2.1. Servicio web de traducción de palabra a LSE (vídeo)

Este servicio permite obtener el video del signo en LSE correspondiente a una palabra determinada en lenguaje natural. Los videos utilizados en este servicio provienen del catálogo de vídeos de LSE de la web de ARASAAC. Para poder acceder a este servicio, se debe hacer una llamada GET a la API, indicando la palabra que se desea traducir a LSE de la siguiente forma:

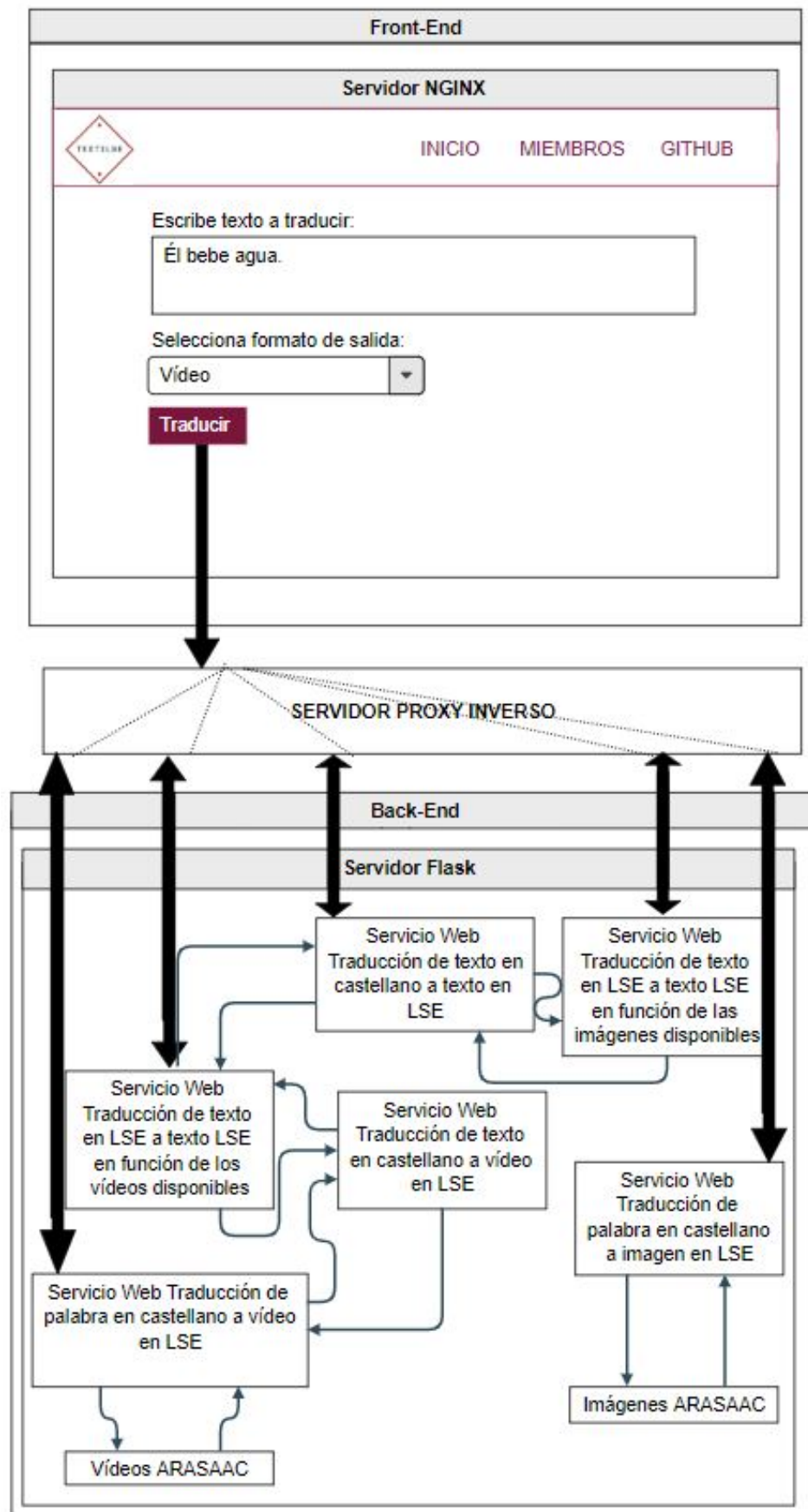


Figura 5.2: Arquitectura de Text2LSE

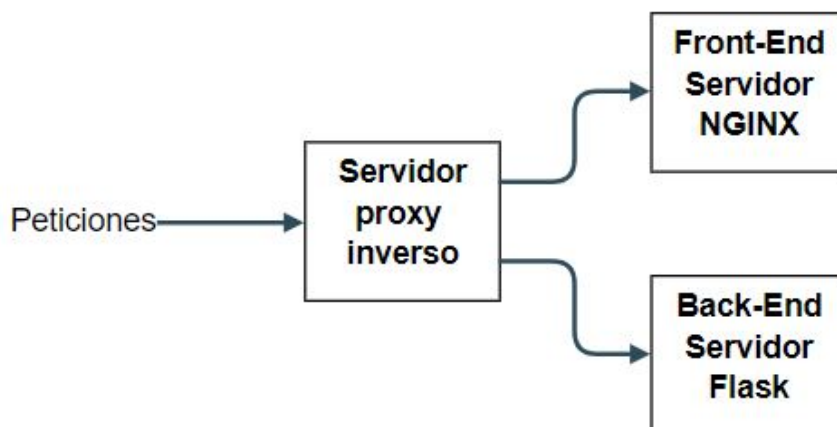


Figura 5.3: Esquema Proxy Inverso

```
https://holstein.fdi.ucm.es/tfg-text2lse/video/<palabra>
```

En el parámetro “*palabra*” se debe indicar la palabra de la que se desea obtener su signo en LSE.

En caso de que se encuentre el vídeo de la palabra buscada, este servicio devuelve el vídeo deseado en formato mp4. En caso de que no se encuentre, devuelve un vídeo de error en formato .mp4. Este flujo lo podemos observar en la Figura 5.4.

Por ejemplo, para obtener el vídeo en LSE de la palabra “*agua*”, como podemos ver en la Figura 5.5, se debe realizar la siguiente llamada:

```
https://holstein.fdi.ucm.es/tfg-text2lse/video/agua
```

5.2.2. Servicio web de traducción de palabra a LSE (imagen)

Este servicio permite obtener la imagen del signo en LSE correspondiente a una palabra determinada en lenguaje natural. Las imágenes utilizadas en este servicio provienen del catálogo de imágenes de LSE de la web de ARA-SAAC. Para poder acceder a este servicio, se debe hacer una llamada GET a la API, indicando la palabra que se desea traducir a LSE de la siguiente

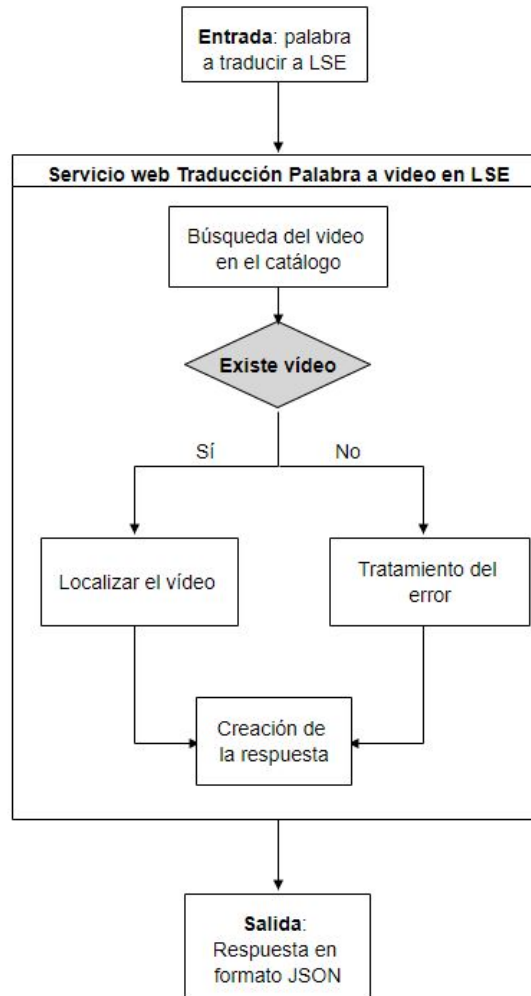


Figura 5.4: Flujo del servicio de Traducción de una palabra a vídeo en LSE



Figura 5.5: Vídeo devuelto por el servicio de traducción de la palabra “agua” a LSE

forma:

```
https://holstein.fdi.ucm.es/tfg-text2lse/imagen/<palabra>
```

En el parámetro “*palabra*” se debe indicar la palabra de la que se desea obtener su signo en LSE.

En caso de que se encuentre la imagen de la palabra buscada, este servicio devuelve la imagen deseada .jpg. En caso de que no se encuentre, devuelve una imagen de error en formato .jpg. Este flujo lo podemos observar en la Figura 5.6.

Por ejemplo, para obtener la imagen en LSE de la palabra “*coche*”, como podemos ver en la Figura 5.7, se debe realizar la siguiente llamada:

```
https://holstein.fdi.ucm.es/tfg-text2lse/imagen/coche
```

5.2.3. Servicio web para Traducción de texto en castellano a texto en LSE

Este servicio implementa la funcionalidad de traducción de un texto en castellano a LSE en formato texto. Para poder acceder a este servicio, se debe realizar una petición POST a la API en la siguiente URL:

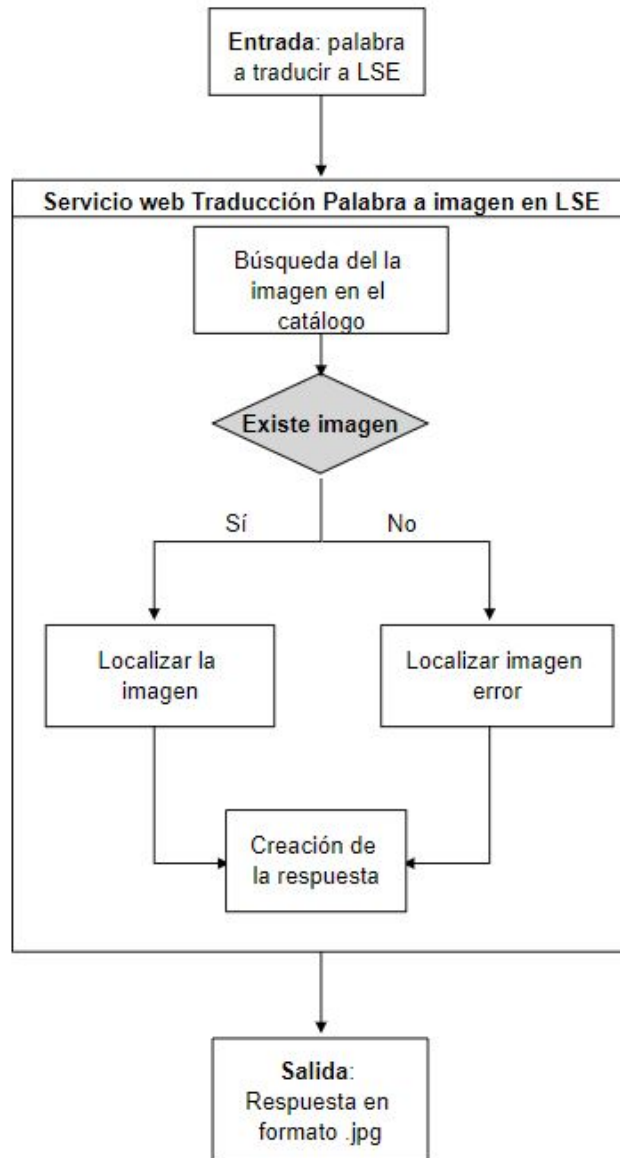


Figura 5.6: Flujo del servicio de Traducción de una palabra a imagen en LSE



Figura 5.7: Imagen devuelto por el servicio de traducción de la palabra “coche” a LSE

```
https://holstein.fdi.ucm.es/tfg-text2lse/TextoLSE
```

Los datos a enviar en la petición POST deben tener la siguiente estructura en JSON:

```
{ 'Texto': '<texto>' }
```

En el parámetro ‘texto’ se debe incluir el texto que se desea traducir a LSE. Un ejemplo de uso sería:

```
Entrada: { 'Texto': 'Él niño bebe agua.' }  
Salida: { 'Texto': 'ñiño agua beber.' }
```

El flujo de este servicio web lo podremos ver en la Figura 5.8.

Para que el servicio web sea capaz de traducir una frase a la LSE se ha tenido que realizar un procesamiento previo del texto en castellano para poder adaptarlo a la Lengua de Signos Española. A pesar de disponer herramientas PLN de la Universidad, debido a la singularidad de Lengua de Signos y a la falta de tiempo para aprender a usar una herramienta nueva debido a su complejidad el procesamiento del Lenguaje Natural ha sido implementado desde cero por los tres integrantes del equipo utilizando únicamente la herramienta Spacy.

La implementación del PLN está basada en un sistema de reglas las cuales filtran qué palabras deben de ser utilizadas y cuáles no. La elección a la hora de desarrollar el Procesamiento del Lenguaje Natural para nuestro proyecto estaba entre un sistema basado en reglas o mediante el uso de aprendizaje

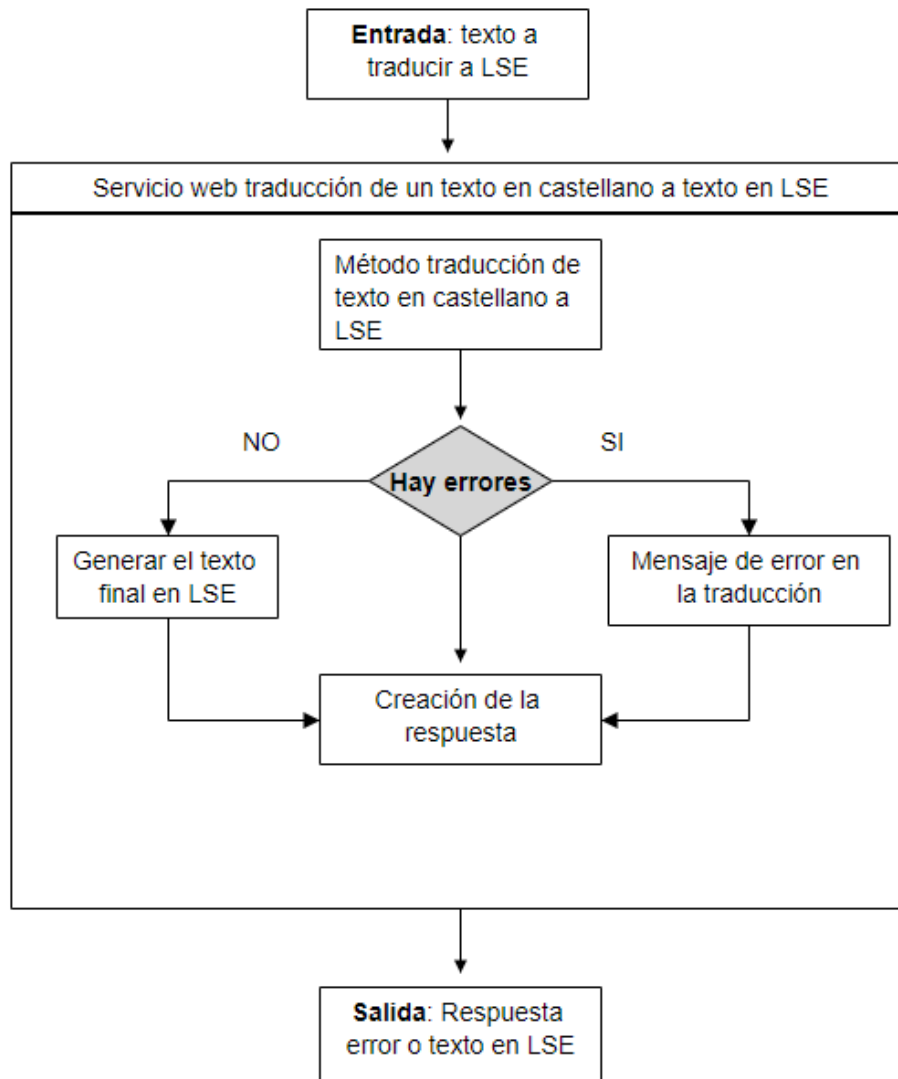


Figura 5.8: Flujo servicio de traducción de texto en castellano a texto en LSE

automático. La opción de desarrollar un aprendizaje automático no era viable debido a la dificultad y la falta de ejemplos en castellano para entrenar el sistema. Por este motivo se eligió el sistema basado en reglas ya que era más fácil de implementar y aunque puede limitar la capacidad de traducción de frases que no cumplan dichas reglas, cumple con los objetivos marcados en el proyecto que es traducir frases simples a LSE.

Nuestro PLN se divide en tres fases. Lo primero que hace es recopilar información de la frase en castellano para luego realizar un análisis sintáctico y morfológico de la frase y así poder ordenarla respetando la estructura de la LSE. A continuación se explicará con más detalle cada una de las fases:

5.2.3.1. Recopilar la información

Para poder realizar los análisis sintácticos y morfológicos de la frase se necesita obtener toda la información posible de la frase. Para ello se realizará el proceso de Tokenización el cual divide la frase en una lista de palabras las cuales pasan a denominarse *tokens* pero con la diferencia de que ahora cada token contiene gran cantidad de información. Esta información es:

- **Dependencia:** función de la palabra dentro de la frase. Ejemplo: ‘
- **Tags:** información de la palabra como el género, el número, el tiempo verbal, etc.
- **Part of speech:** tipo de palabra como nombre, adjetivo, etc.
- **Lema:** Lema de palabra.

Por ejemplo la información que se obtendría de la oración ‘*Los niños merendaron chocolate*’ sería 5.9:

```
LOS:
[DEPENDENCIA (det) --- TAGS {DET__Definite=Def|Gender=Masc|Number=Plur|PronType=Art} --- POS (DET) --- LEMMA (Los)]
NIÑOS:
[DEPENDENCIA (nsubj) --- TAGS {NOUN__Gender=Masc|Number=Plur} --- POS (NOUN) --- LEMMA (niño)]
MERENDARON:
[DEPENDENCIA (ROOT) --- TAGS {VERB__Mood=Ind|Number=Plur|Person=3|Tense=Past|VerbForm=Fin} --- POS (VERB) --- LEMMA (merendar)]
CHOCOLATE:
[DEPENDENCIA (obj) --- TAGS {NOUN__Gender=Masc|Number=Sing} --- POS (NOUN) --- LEMMA (chocolate)]
.:
```

Figura 5.9: Información del token “Los niños merendaron chocolate”

5.2.3.2. Análisis sintáctico

Una vez separado la oración en tokens hay que diferenciar que palabras forman parte del sujeto y cuales del predicado. Para ello hay que recorrer el árbol de dependencias mediante una función recursiva empezando desde el ROOT que es el token raíz del cual dependen todos los demás. Para entender

cómo funcionan las dependencias se va a utilizar como ejemplo la oración ‘Ayer los niños merendaban chocolate’. En la Figura 5.10 se muestran las dependencias de esta frase.

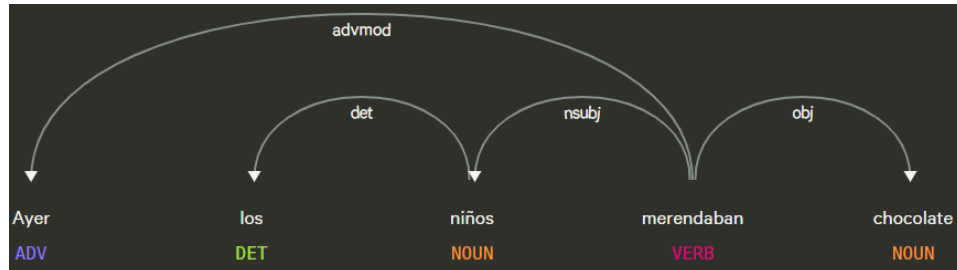


Figura 5.10: Displacy grafo de dependencias

Como se puede apreciar en el diagrama del token ‘merendaban’ dependen tres tokens (Ayer, niños y chocolate) y de ellos también dependen otros y así sucesivamente. Estas dependencias se traducen en un árbol donde el ROOT es la raíz como podemos ver en la Figura 5.11:

Para obtener las distintas partes de la oración se recorre el árbol de la siguiente forma. Lo primero es obtener el sujeto y para ello hay que buscar el token que contenga la dependencia “nsubj”. Una vez encontrado hay que volver a llamar a la función recursiva para obtener los hijos y guardarlos en una lista. En el ejemplo anterior “*Ayer los niños merendaban chocolate*” el sujeto sería (Los niños). Una vez obtenido los tokens que pertenecen al sujeto, los tokens restantes pertenecen al predicado (Ayer, merendaban y chocolate) y añaden a una lista.

- **Sujeto:** [Los, niños]
- **Predicado:** [Ayer, merendaban, chocolate]

5.2.3.3. Análisis morfológico

La Lengua de Signos Española tiene una estructura gramatical diferente al castellano y como el objetivo del TFG es llegar a traducir frases simples a LSE, la estructura de la oración que se ha tomado como referencia siguiente:

TIEMPO + SUJETO + OBJETO + VERBO.

A parte de seguir esta estructura, en la LSE las palabras pueden tener un orden distinto al castellano o se pueden añadir o eliminar palabras que la oración original en castellano no estaban.

El análisis morfológico que hemos seguido es a la hora de añadir o quitar palabras de la oración para poder obtener una traducción lo mas real posible es el siguiente:

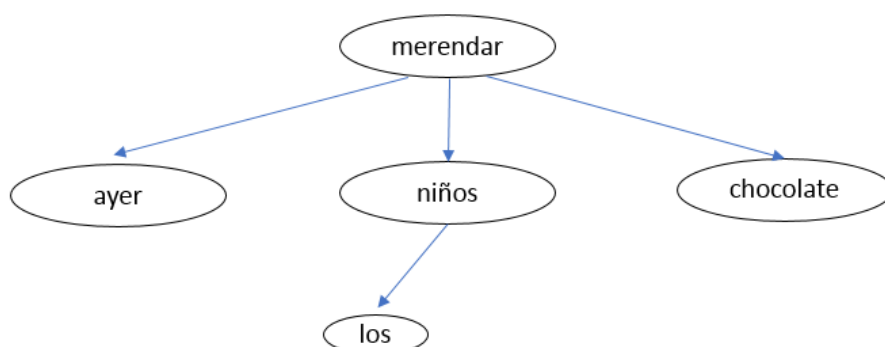


Figura 5.11: Árbol de la oración "Los niños merendaron chocolate"

- **Determinantes:** Los determinantes se omiten a la hora de hacer la traducción. *Ejemplo:* "El niño come carne" se traduce como "Niño carne comer".
- **Verbos copulativos:** Los verbos copulativos ser, estar o parecer se omiten a la hora de la traducción. *Ejemplo:* "yo soy bajo" se traduce como "yo bajo".
- **Posesivos:** Los determinantes posesivos se sustituyen por pronombres personales. *Ejemplo:* "Mi niño es bajo" se traduce como "yo niño bajo".
- **Preposiciones:** Algunas preposiciones se quitan y otras se añaden. *Ejemplo:* "Mi niño está en el parque" se traduce como "yo hijo parque".
- **Adjetivos:** Los adjetivos se ponen en masculino. *Ejemplo:* "Mi mamá es fea" se traduce como "yo mamá feo".
- **Temporalidad:** La temporalidad como vemos en la estructura de más arriba va al principio de la oración. *Ejemplo:* "Yo comí chocolate ayer" se traduce como "Ayer yo chocolate comer". Sin embargo no siempre hay adverbios de tiempo en la oración que indican la temporalidad. Quién indica en este caso el tiempo es el verbo por lo que habría que añadir la temporalidad con las signos PASADO o FUTURO. *Ejemplo:* "Yo comeré chocolate" se traduce como "Futuro yo chocolate comer".

Una vez añadido y quitado las palabras el sujeto y predicado, la frase se tiene que ordenar según la estructura tiempo + sujeto + objeto + verbo por lo que el resultado final de la frase usada como ejemplo "Los niños ayer merendaban chocolate" sería:

- **Sujeto:** [niños]

- **Predicado:** [Ayer, merendar, chocolate]
- **Frase ordenada:** “Ayer niños chocolate merendar”

5.2.4. Servicio web para traducción de texto castellano a texto en LSE dependiendo de los vídeos existentes

Este servicio implementa la funcionalidad de traducción de un texto en castellano a LSE en formato texto en función de los vídeos existentes en el sistema. Al igual que en el servicio anterior, los vídeos utilizados en este servicio provienen del catálogo de vídeos de LSE de la web de ARASAAC. Para poder acceder a este servicio, se debe realizar la siguiente petición POST a la API:

```
https://holstein.fdi.ucm.es/tfg-text2lse/TextoLSEVideos/
```

Los datos a enviar en la petición POST deben tener la siguiente estructura en JSON:

```
{ 'Texto': '<texto>' }
```

En el parámetro “*texto*” se debe incluir el texto que se desea traducir a LSE. Este flujo lo podemos observar en la Figura 5.12.

Por ejemplo, para traducir el texto “*Mis tías comen chocolate*” a LSE en formato texto en función de los vídeos existentes, se debe realizar la llamada POST indicada anteriormente con el siguiente JSON:

```
Entrada: { 'Texto': 'Mis tías comen chocolate' }  
Salida: { 'Texto': 'yo tío mujer otro chocolate comer' }
```

En este caso, al no encontrar el vídeo para la palabra “*tías*” busca el vídeo de esa palabra sin morfemas de género y número, en este caso busca el vídeo de la palabra “*tío*” añadiendo la palabra “*mujer*” para indicar el femenino y la palabra “*otro*” para indicar el plural.

5.2.5. Servicio web para traducción de texto castellano a texto en LSE dependiendo de las imágenes existentes

Este servicio implementa la funcionalidad de traducción de un texto en castellano a LSE en formato texto en función de las imágenes existentes en el sistema. Al igual que en el servicio anterior, las imágenes utilizados en este servicio provienen del catálogo de imágenes de LSE de la web de

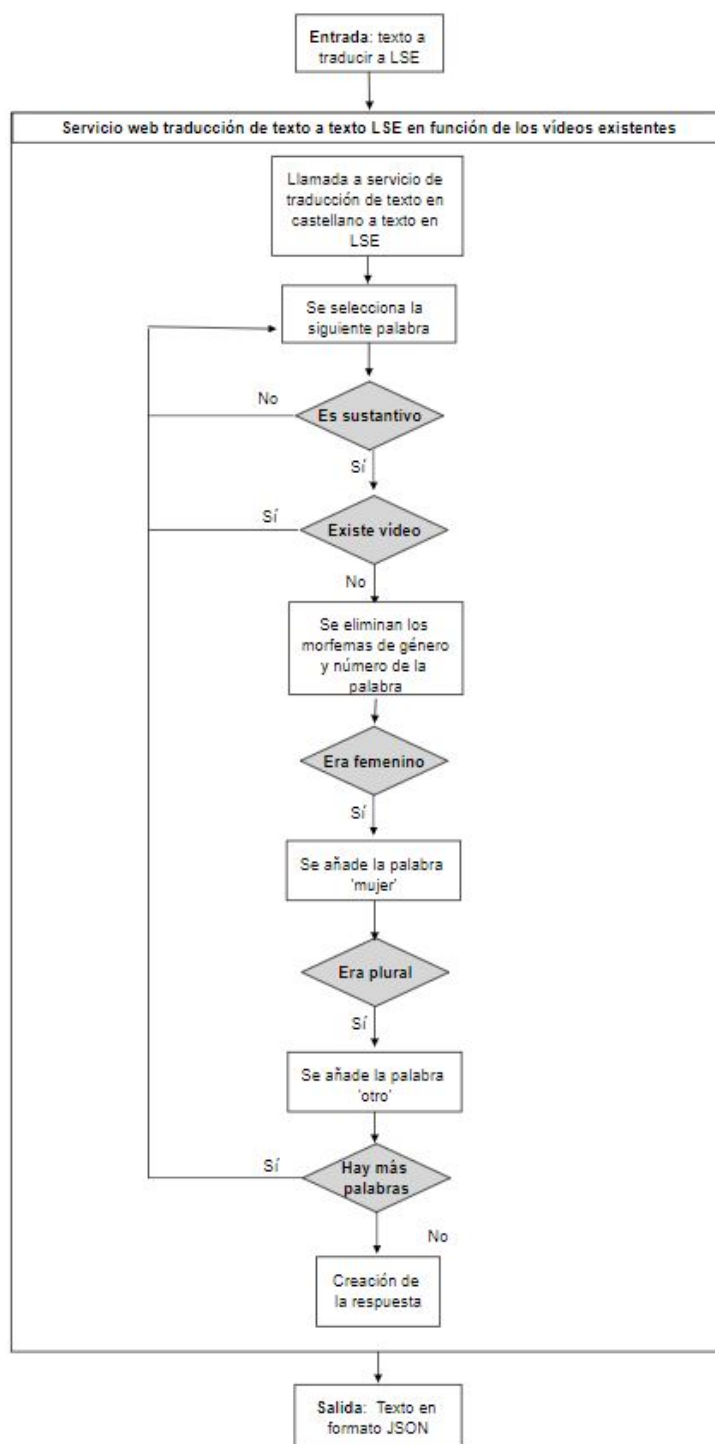


Figura 5.12: Flujo servicio de traducción de texto en castellano a texto en LSE en función de los vídeos existentes

ARASAAC. Para poder acceder a este servicio, se debe realizar la siguiente petición POST a la API:

```
https://holstein.fdi.ucm.es/tfg-text2lse/textoImagen/
```

Los datos a enviar en la petición POST deben tener la siguiente estructura en JSON:

```
{ 'Texto': '<texto>' }
```

En el parámetro *texto* se debe incluir el texto que se desea traducir a LSE. Este flujo lo podemos observar en la Figura 5.13.

Por ejemplo, para traducir el texto *“Los caballos son rápidos”* a LSE en formato texto en función de los vídeos existentes, se debe realizar la llamada POST indicada anteriormente con el siguiente JSON:

```
Entrada: { 'Texto': 'Los caballos son rápidos' }  
Salida: { 'Texto': 'caballo otro rápido' }
```

En este caso, al no encontrar la imagen para la palabra *“caballos”* busca el vídeo de esa palabra sin morfemas de género y número, en este caso busca el vídeo de la palabra *“caballo”* añadiendo la palabra *“otro”* para indicar el plural.

5.2.6. Servicio web para traducción de texto a LSE (vídeo)

Este servicio implementa la funcionalidad de traducción de un texto en castellano a LSE en formato video. Al igual que en el servicio anterior, los videos utilizados en este servicio provienen del catálogo de vídeos de LSE de la web de ARASAAC. Para poder acceder a este servicio, se debe realizar la siguiente petición POST a la API:

```
https://holstein.fdi.ucm.es/tfg-text2lse/video/
```

Los datos a enviar en la petición POST deben tener la siguiente estructura en JSON:

```
{ 'Texto': '<texto>' }
```

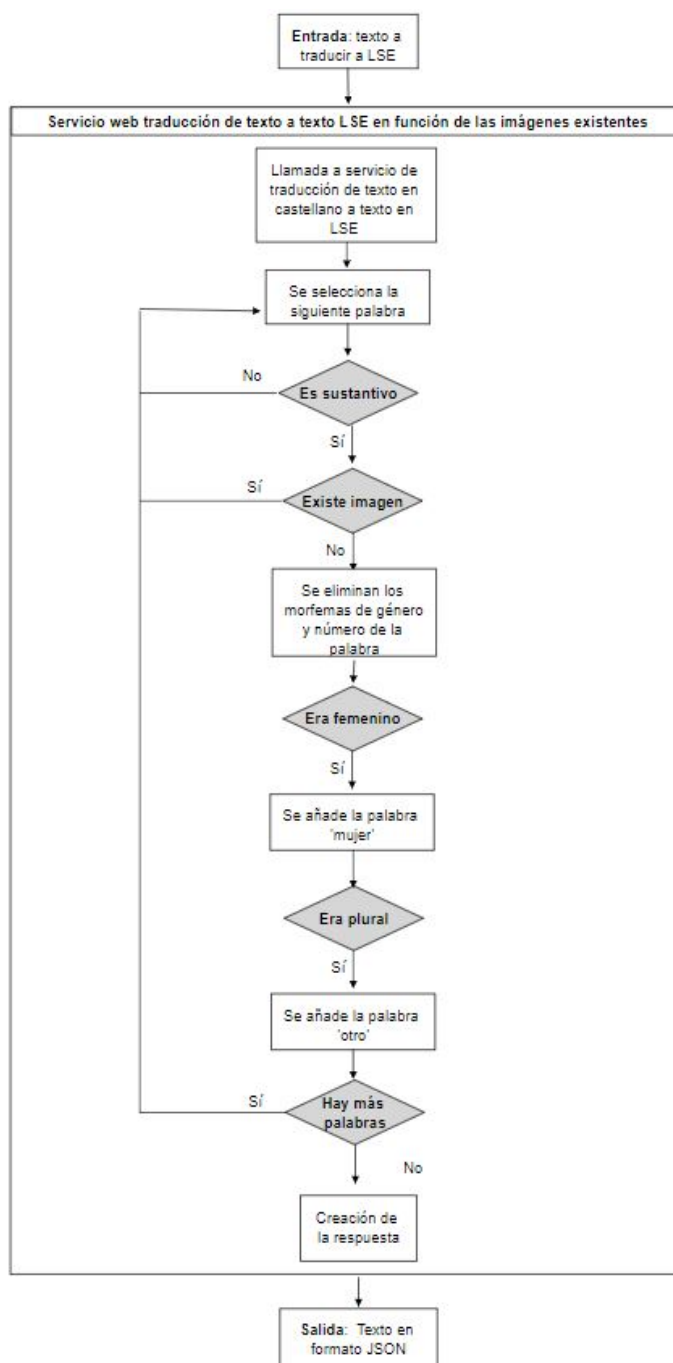


Figura 5.13: Flujo servicio de traducción de texto en castellano a texto en LSE en función de las imágenes existentes

En el parámetro *“texto”* se debe incluir el texto que se desea traducir a LSE. En caso de que se encuentren los recursos para traducir las palabras solicitadas en formato vídeo, se ejecuta un proceso que junta todos los vídeos en uno solo. Este vídeo es el que devuelve el servicio en formato mp4. Este flujo lo podemos observar en la Figura 5.14.

Por ejemplo, para obtener el vídeo en LSE del texto *“Los niños comen chocolate”*, como podemos ver en la Figura 5.15, se debe realizar la llamada POST indicada anteriormente con el siguiente JSON:

```
{ 'Texto': 'Los niños comen chocolate' }
```

5.3. Front-End

En esta sección se explica en detalle la aplicación web desarrollada, cuya finalidad es proporcionar a los usuarios una interfaz sencilla donde puedan introducir el texto que desean traducir y obtener la traducción del texto en LSE, tanto en imágenes como en vídeo. En la Figura 5.1 se muestra la interfaz de la aplicación, que consta de un input donde el usuario podrá introducir el texto en castellano que desee traducir, un desplegable donde podrá seleccionar el formato de salida y un botón para realizar la traducción.

Los formatos de salida disponibles en el desplegable son los siguientes:

- **Traducción a vídeo:** tras seleccionar este formato y pulsar el botón “traducir”, se realizará una llamada Fetch al microservicio de traducción a Texto Castellano a texto LSE dependiendo de los videos existentes, pasando por el servidor proxy, que a su vez redirigirá la petición a la API. Una función Fetch se encargará de recibir el texto resultante y de llamar por cada una de esas palabras al servicio que devuelve el video LSE correspondiente. Posteriormente se incrustan todos los vídeos en orden en el código HTML, de tal manera que se reproduce uno detrás de otro. Así, se podrá visualizar por pantalla un vídeo en el que se podrá ver a un intérprete de ARASAAC realizando los signos LSE correspondientes.
- **Traducción a imágenes:** de la misma manera que en el caso de los vídeos, tras seleccionar este formato y pulsar el botón “traducir”, se realizará una llamada Fetch al microservicio de traducción a Texto Castellano a texto LSE dependiendo de las imágenes existentes, pasando por el servidor proxy, que a su vez redirigirá la petición a la API. Una función Fetch se encargará de recibir el texto resultante y de llamar por cada una de esas palabras al servicio que devuelve la imagen

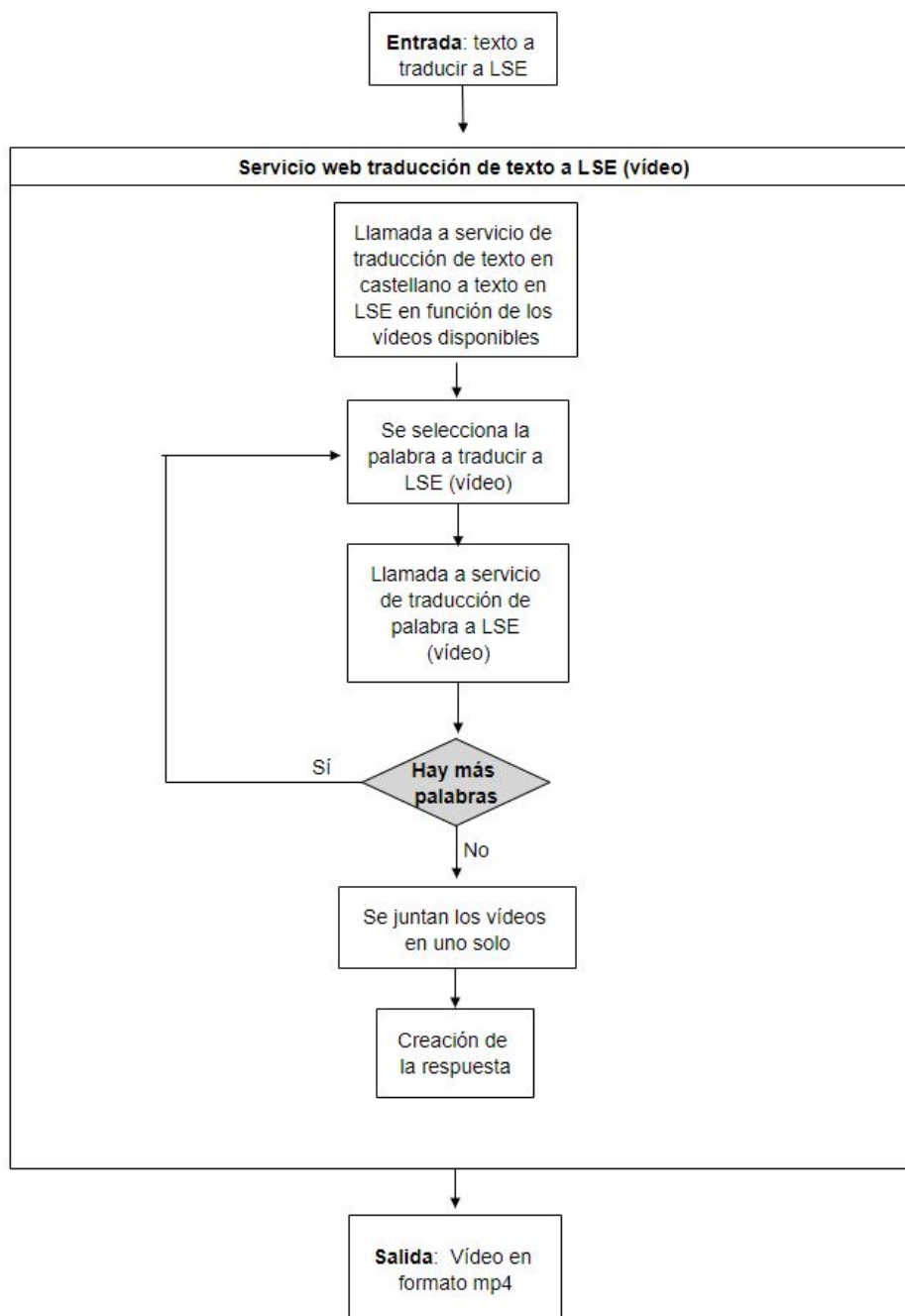


Figura 5.14: Flujo servicio de traducción de texto en castellano a LSE en formato vídeo



Figura 5.15: Vídeo devuelto por el servicio de traducción del texto “*Los niños comen chocolate*” a LSE

LSE correspondiente. Posteriormente se incrustan todas las imágenes en orden en el código HTML.

- **Traducción a texto LSE:** tras seleccionar este formato y pulsar el botón “traducir” se llamará a una función Fetch que realizará una petición a la url correspondiente. En caso de éxito aparecerá por pantalla dicho texto traducido a texto LSE.

En caso de error en cualquiera de las dos funcionalidades, la aplicación mostrará un modal con el tipo y detalle del error obtenido al realizar la petición a la API donde se encuentran nuestros servicios web.

Por otro lado, la interfaz también cuenta con un menú en la parte superior, donde además de a la página principal podemos acceder a una página con información sobre los miembros del equipo y de los tutores, y a otra con información sobre la aplicación. Todo ello se ha implementado siguiendo un diseño responsive, haciendo uso de Bootstrap 3, para garantizar que la herramienta sea accesible desde cualquier dispositivo. En la Figura 5.16 podemos observar la interfaz vista desde un dispositivo móvil.



Figura 5.16: Aplicación Web vista desde un dispositivo móvil

Capítulo 6

Trabajo individual

6.1. Sara Vegas Cañas

Lo primero de todo fue una investigación profunda de la discapacidad auditiva y de la Lengua de Signos Española por parte de los tres integrantes de este proyecto, de esta manera todos nos concienciamos de las dificultades que tienen todas las personas sordas para comunicarse en su día a día. Realicé una investigación de las herramientas que tienen estas personas a su disposición, tanto posibles traductores a LSE, como bancos de vídeos e imágenes. Encontré una variedad de aplicaciones, las cuales probé para poder ver hasta donde llegaban las soluciones actuales para el problema de comunicación de las personas sordas.

Cuando fijamos los objetivos de nuestro TFG, empezamos con la investigación de las posibles tecnologías que podríamos usar en el proyecto. Mi investigación en este tema se centró en servicios web, sobre todo de tipo REST, Python y en el Procesamiento del Lenguaje Natural, probando diferentes librerías, como Spacy y NLTK, para poder elegir la que más se ajustase a nuestras necesidades.

Una vez que terminó la primera parte de investigación, nos reunimos para juntar todas nuestras conclusiones. Con éstas comenzamos a desarrollar la motivación y objetivos de la memoria, junto con todos los apartados referentes a la Lengua de Signos, encargándome sobre todo de las Reglas de la LSE y de los bancos de vídeos e imágenes LSE.

A continuación, comenzamos a crear una primera base de la aplicación. Lo primero fue realizar la preparación del entorno local en Debian 10, utilizando Nginx para alojar la aplicación web, y Flask para ejecutar la API, preparando el modo debug para el posterior proceso de desarrollo de código. Construimos una web básica, en la cual podíamos escribir un texto y enviar-

lo a la API a través de JavaScript, mediante jQuery y Ajax, y una primera estructura de la API REST en Python. En este apartado me encargue junto con mi compañero Alejandro del desarrollo de una funcionalidad javascript capaz de lanzar una petición a la API con el texto y que esta nos devolviese un vídeo mp4. Más adelante decidimos cambiar las llamadas AJAX por llamadas Fetch.

Respecto a los servicios web, me dediqué a estructurar la función que devuelve un vídeo con varios signos y de desarrollar las llamadas a las funciones que devuelven diferente tipo de información en formato JSON.

Respecto a PLN, en un principio me centré en desarrollar un algoritmo recursivo que recorriera un árbol de dependencias desde el verbo a partir de una frase, complementándolo con el tratamiento de preposiciones y artículos colocando las palabras en orden de LSE. Junto con mis compañeros realizamos una estructura y desarrollo de la traducción con dos partes bien diferenciadas, el orden y la morfología.

Para poder observar de manera sencilla el funcionamiento de estos servicios, decidimos realizar un nuevo diseño de la página web que fuera totalmente responsive, en el cual participamos los tres miembros del proyecto.

Además de todo el trabajo técnico, participé en la redacción de la memoria. Me ocupé junto con mis compañeros del capítulo 1 y 2, que desarrollamos de forma conjunta. En el capítulo 3 me encargué de la redacción de la sección de Google Collaboratory y la de Python, que finalmente eliminamos. En cuanto al capítulo 4, donde explicamos la metodología de trabajo seguida en este TFG, desarrollé las secciones de Trello y Reuniones. Finalmente en el capítulo 5 me ocupe de la introducción, la sección de Arquitectura y las de los tres servicios que componen la traducción a LSE en formato vídeo. Por otro lado, trabajé en la corrección de todos estos apartados a medida que nuestros tutores nos fueron dando feedback.

6.2. Alejandro Torralbo Fuentes

Al comienzo del proyecto, los tres integrantes del grupo realizamos una fase de investigación, en la que adquirimos conocimientos sobre la discapacidad auditiva y la Lengua de Signos, y nos pusimos al día en los problemas a los que se enfrenta el colectivo de personas sordas para poder orientar nuestro proyecto a solucionar dichos problemas. A continuación, realizamos una búsqueda de recursos que pudiéramos utilizar en el desarrollo de nuestra aplicación, encontrando bancos de imágenes y vídeos de Lengua de Signos

Española como la de ARASAAC, entre otras herramientas.

Una vez recopilada la información necesaria, comenzamos con la instalación y preparación del entorno de desarrollo local: una máquina virtual con Debian 10, en la que instalamos un servidor para alojar el cliente con Nginx, y una API desarrollada en Python y ejecutada mediante Flask. Me encargué de configurar el entorno Flask, activando el modo debug para poder empezar a depurar el código de la API, e instalé la herramienta Postman para la realización de pruebas y el controlador de versiones Git, para trabajar en el código de forma conjunta con el resto de compañeros.

A partir de este punto, comenzamos con el desarrollo de la aplicación. Me encargué de desarrollar una web responsive con HTML, CSS y Javascript, alojada en el servidor Nginx. Junto con mi compañera Sara, conseguimos implementar una función en Ajax capaz de realizar una petición POST a la API, enviando el texto introducido por el usuario y recibiendo un vídeo en formato mp4 para mostrarlo en la web. Posteriormente mi compañera Sara y yo cambiamos estas llamadas Ajax por llamadas Fetch. Además, me encargué de que la API fuera capaz de juntar varios vídeos en función del texto recibido, y prepararlo para enviarlo a la página web. Posteriormente los tres integrantes decidimos estructurar la API y dividir esta funcionalidad en varias funciones diferentes, de modo que el usuario interesado en utilizar nuestros microservicios cuente con varias opciones de llamada. Mis compañeros Miguel y Sara se encargaron de dividir esta función. Una vez estructurado todo, los tres compañeros conjuntamente participamos en el desarrollo y pruebas de los microservicios de PLN.

Junto con el trabajo técnico, también participé en la redacción y corrección de la memoria. Los tres compañeros comenzamos desarrollando conjuntamente el capítulo 1 y 2. Además, desarrollé el apartado de Flask del capítulo 3 y metodologías de trabajo en el capítulo 4. En cuanto al capítulo 5, escribí la sección Front-End, la introducción de la sección Back-End y los servicios de traducción LSE en imágenes. A medida que fueron llegando las correcciones de nuestros tutores, me encargué de la corrección del capítulo 1 y parte del capítulo 2, junto con todas las secciones que yo mismo redacté en el resto de capítulos.

6.3. Miguel Rodríguez Cuesta

El comienzo del proyecto fue de investigación. Cada miembro del grupo buscó información de distinta variedad relacionada con el mundo de la Lengua de Signos para después realizar una puesta en común para empezar a

redactar la memoria.

Personalmente durante este período he ido recopilando información sobre qué es la LSE, quién la usa, como funciona, etc. También investigué sobre la comunidad sorda y sus problemas de adaptación la sociedad actual, discapacidad auditiva y una gran variedad de artículos de interés. Después de hacer una puesta en común de todo lo investigado, nos dividimos la memoria entre los tres, siendo mi parte reescribir servicios Web, aplicaciones de traducción de texto a LSE y procesamiento del Lenguaje Natural.

Para la segunda parte del proyecto decidimos los tres integrantes del equipo organizar una buena estructura para poder dividirnos bien el trabajo. Nos dividimos los nuevos apartados de la memoria siendo mi parte arreglar redacciones anteriores y crear los nuevos apartados de Nginx, Git y Github así como explicar en el capítulo 5 como funciona el Procesamiento de Lenguaje Natural de Text2LSE.

En cuanto al desarrollo de la aplicación he contribuido realizando un microservicio que devuelve el vídeo en Lengua de Signos correspondiente a una palabra. También he participado en el desarrollo del PLN creando junto con mis compañeros una gran cantidad de reglas así como arreglar fallos que he ido encontrando tanto en la web como en el servidor.