
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 |
| 2.2.1. ARASAAC | 8 |
| 2.2.2. Banco de imágenes Fundación CNSE | 9 |
| 2.2.3. SpreadTheSign | 11 |
| 2.3. La Lengua de Signos | 12 |
| 2.3.1. Fonología de la Lengua de Signos Española | 13 |
| 2.3.2. Sintaxis de la Lengua de Signos Española | 15 |
| 2.3.3. Morfología de la Lengua de Signos Española | 16 |
| 2.3.4. Verbos de la Lengua de Signos Española | 19 |
| 2.4. Aplicaciones de Traducción de Texto a LSE | 20 |
| 2.4.1. Text2Sign | 22 |
| 2.4.2. TextoSign | 22 |
| 2.4.3. StorySign | 23 |
| 2.4.4. TeCuento | 23 |
| 2.4.5. Conclusiones | 23 |
| 3. Tecnologías utilizadas | 25 |
| 3.1. Servicios Web | 25 |
| 3.1.1. Servicios Web SOAP | 26 |
| 3.1.2. Servicios Web REST | 26 |
| 3.1.3. Ventajas de Servicios Web | 27 |
| 3.1.4. Desventajas de Servicios Web | 27 |
| 3.1.5. Flask | 28 |
| 3.2. Procesamiento del Lenguaje Natural | 29 |

| | |
|---|-----------|
| 3.2.1. Estructura del procesamiento del lenguaje natural . . . | 29 |
| 3.2.2. Técnicas de PLN | 30 |
| 3.2.3. Herramientas PLN | 30 |
| 3.3. Python | 31 |
| 3.4. Nginx | 32 |
| 3.5. Google Colaboratory | 33 |
| 4. Metodologías utilizadas | 35 |
| 4.1. Metodologías Ágiles | 35 |
| 4.2. Kanban | 36 |
| 4.3. Trello | 37 |
| 4.4. Reuniones | 38 |
| 4.5. Control de versiones | 39 |
| 4.5.1. GIT | 39 |
| 4.5.2. GITHUB | 39 |
| 4.5.3. Estructura de Trabajo con GIT y GITHUB | 40 |
| 5. Text2LSE | 43 |
| 5.1. Arquitectura | 43 |
| 5.2. Back-End | 44 |
| 5.2.1. Servicios | 46 |
| 5.2.2. Procesamiento del Lenguaje Natural de TEXT2LSE | 48 |
| 5.3. Front-End | 54 |
| 6. Trabajo individual | 57 |
| 6.1. Sara Vegas Cañas | 57 |
| 6.2. Alejandro Torralbo Fuentes | 58 |
| 6.3. Miguel Rodríguez Cuesta | 59 |
| Bibliografía | 61 |

Índice de figuras

| | |
|---|----|
| 2.1. Representación de Sistema Braille | 8 |
| 2.2. Representación de Sistemas Pictográficos | 9 |
| 2.3. Pictograma “coche” del banco de imágenes ARASAAC | 10 |
| 2.4. Signo “hola” en LSE del banco de imágenes ARASAAC | 10 |
| 2.5. Recursos LSE de la Fundación CNSE | 11 |
| 2.6. Imagen de la página web SpreadTheSign | 12 |
| 2.7. Representación de la configuración de la mano en Lengua de Signos | 14 |
| 2.8. Representación de la orientación de las manos en Lengua de Signos | 15 |
| 2.9. Signo “oír” en LSE del banco de imágenes ARASAAC | 16 |
| 2.10. Signo “hombre” del banco de imágenes ARASAAC | 16 |
| 2.11. Signo “mujer” del banco de imágenes ARASAAC | 17 |
| 2.12. Signo “abuelo” del banco de imágenes ARASAAC | 17 |
| 2.13. Signo “abuela” del banco de imágenes ARASAAC | 18 |
| 2.14. Signo “niño” del banco de imágenes ARASAAC | 18 |
| 2.15. Signo “niña” del banco de imágenes ARASAAC | 18 |
| 2.16. Signo “uno” del banco de imágenes ARASAAC | 19 |
| 2.17. Signo “muchos” del banco de imágenes ARASAAC | 20 |
| 2.18. Signo “querer” en LSE del banco de imágenes ARASAAC | 21 |
| 2.19. Signo “ayudar” en LSE del banco de imágenes ARASAAC | 21 |
| 2.20. Signo “señalar” en LSE del banco de imágenes ARASAAC | 21 |
| 2.21. Maya, avatar usado por TextoSign. | 22 |
| 2.22. Imagen de la aplicación StorySign. | 23 |
| 2.23. Imagen de la aplicación TeCuento. | 24 |
| 3.1. Vista de un cuaderno de Google Colaboratory | 34 |
| 4.1. Tablero “Memoria” utilizado en el proyecto. | 38 |
| 4.2. Estructura “Git” utilizada en el proyecto. | 41 |
| 5.1. Aplicación web Text2LSE | 44 |

| | |
|--|----|
| 5.2. Arquitectura de Text2LSE | 45 |
| 5.3. Esquema Proxy Inverso | 46 |
| 5.4. Flujo del servicio de Traducción de una palabra a vídeo en LSE | 47 |
| 5.5. Flujo servicio de traducción de texto en castellano a video en LSE | 49 |
| 5.6. Flujo servicio de traducción de texto en castellano a texto en LSE | 50 |
| 5.7. Información del token merendaban | 52 |
| 5.8. Displacy grafo de dependencias | 52 |
| 5.9. Árbol binario de una oración | 53 |
| 5.10. Aplicación Web vista desde un dispositivo móvil | 56 |

Índice de Tablas

Capítulo 1

Introducción

"Lo que eres, lo eres por accidente de nacimiento; lo que soy, lo soy yo solo. Hay y habrá mil príncipes; solo hay un Beethoven".

Ludwig van Beethoven

En la sección 1.1 de este capítulo se explicará la motivación que hay detrás de la realización de este TFG, y en la 2.2 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, que en algunos casos hacen que estas personas se sientan apartadas del resto y no tengan acceso completo a ciertos servicios fundamentales, dificultando así su día a día. Gracias a los avances tecnológicos, se están desarrollando soluciones con el objetivo de ayudar a estos colectivos, facilitando de esta manera aspectos de su vida cotidiana. Un ejemplo son las aplicaciones basadas en pictogramas cuya finalidad es ayudar a comunicarse a personas con algún tipo de discapacidad cognitiva.

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. 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 lengua oficiales: la Lengua de Signos Española (LSE) y la Lengua de Signos Catalana (LSC).

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, actividades educativas, megafonía o vídeos. Hoy en día, en muchos casos, se cuenta con la posibilidad de añadir subtítulos al contenido audiovisual, pero no es suficiente, ya que 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 el texto a lengua de signos 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 speech to text, lo que resultaría de gran ayuda para personas con discapacidad auditiva.

1.2. Objetivos

El objetivo principal del proyecto es crear una herramienta que sea capaz de traducir un texto en lenguaje natural escrito en castellano a la Lengua de Signos Española (LSE).

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. Su diseño estará centrado en el usuario. Para ello, se debe conocer y comprender las necesidades, limitaciones, comportamiento y características de los potenciales usuarios de la aplicación. Estará basada en la arquitectura orientada a servicios SOA, es decir, tendrá implementadas pequeñas funcionalidades alojadas en un servicio web, siendo así fácilmente reutilizables.

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

Se ha organizado la memoria del proyecto de la siguiente manera:

- **Capítulo 1:** en este capítulo se explica la motivación que hay detrás de este proyecto y los objetivos del mismo. Además se incluye una sección con la estructura por capítulos del presente documento, junto con una breve descripción de cada apartado.
- **Capítulo 2:** en esta sección se habla de la discapacidad auditiva y de la Lengua de Signos Española en relación a este proyecto.
- **Capítulo 3:** en este apartado de la memoria se comentan las herramientas técnicas utilizadas para el desarrollo de nuestra API y Web.
- **Capítulo 4:** en este capítulo se explica con detalle la metodología de trabajo seguida por los integrantes del grupo.
- **Capítulo Trabajo Individual:** en este capítulo cada integrante del grupo explica el trabajo realizado a lo largo de todo el proyecto.

Capítulo 2

Estado del Arte

"La fuerza no viene de la capacidad corporal, sino de la voluntad del alma".

Mahatma Gandhi

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 este colectivo. Posteriormente, en la sección 2.4 se muestran las soluciones de accesibilidad digital que existen para personas con esta discapacidad, así como tecnologías ya desarrolladas similares a la aplicación que se ha desarrollado 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 (Discapnet, 2019):

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 uso 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) (ARASAAC, 2019) 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 y cuyas dificultades físicas no les permiten realizar movimientos con las manos para comunicarse. 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** (Torres, 2001): 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.

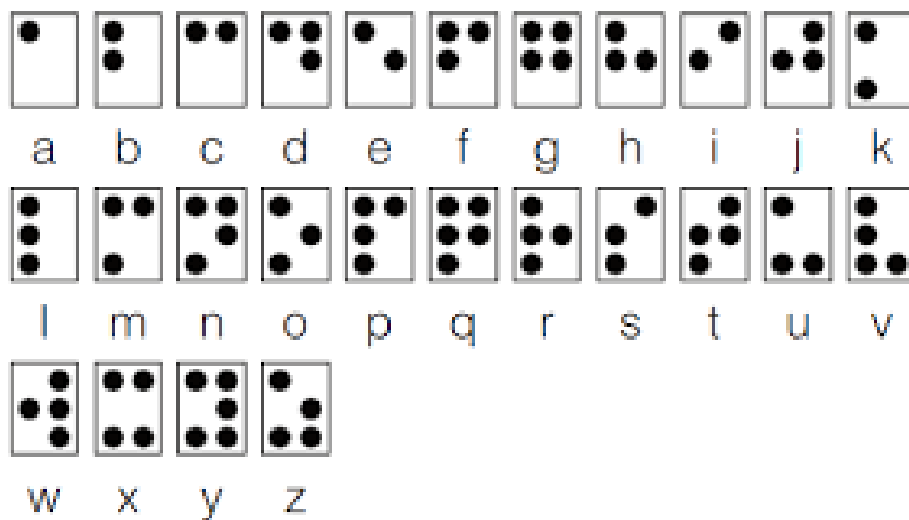


Figura 2.1: Representación de Sistema Braille

- **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 que utilizan las personas con discapacidad visual o ciegos 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.)

2.2.1. ARASAAC

El Portal Aragonés de la Comunicación Aumentativa y Alternativa (ARASAAC)¹, es un proyecto desarrollado en el año 2007 por el gobierno de Aragón. Tiene como objetivo la creación de un sistema pictográfico de comunicación y un conjunto de herramientas de libre distribución. Estos recursos

¹<http://www.arasaac.org/>

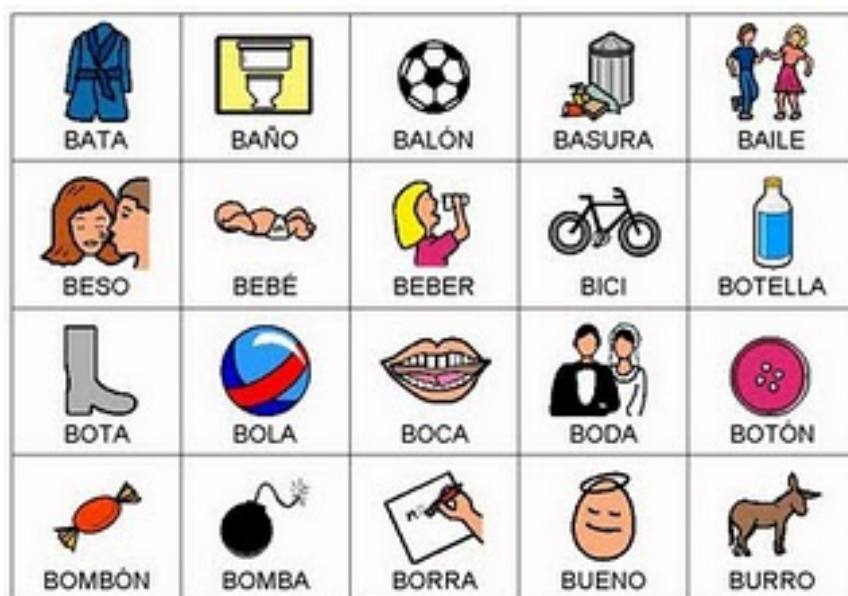


Figura 2.2: Representación de Sistemas Pictográficos

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 2.3 se muestra el pictograma de ARASAAC para la palabra “coche” y en la Figura 2.4 se muestra el signo “hola” en LSE. Este contenido está disponible como contenido descargable con licencia Creative Commons, es decir, se puede usar libremente sin ánimo de lucro.

ARASAAC ofrece una API² (Interfaz de Programación de Aplicaciones) 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 sus recursos correspondientes a la LSE es necesario descargarlos a través del apartado de descargas de su página web³.

2.2.2. Banco de imágenes Fundacion CNSE

La Confederación Estatal de Personas Sordas (CNSE)⁴ 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

²<https://beta.arasaac.org/developers/api>

³<http://www.arasaac.org/descargas.php>

⁴<http://www.fundacioncnse.org/>



Figura 2.3: Pictograma “*coche*” del banco de imágenes ARASAAC



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



Figura 2.5: Recursos LSE de la Fundación CNSE

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 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 imágenes⁵ (ver Figura 2.5) 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.

2.2.3. SpreadTheSign

SpreadTheSign⁶ es un diccionario online administrado por el Centro Europeo de Lenguas de Signos, que cuenta con más de 400.000 signos en vídeo

⁵<http://www.fundacioncnse.org/educa/bancolse/>

⁶<https://www.spreadthesign.com/es.es/search/>



Figura 2.6: Imagen de la página web SpreadTheSign

(ver Figura 2.6) 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 su aplicación para Android⁷ e IOS⁸, aunque ninguna de estas aplicaciones permite la descarga de contenido.

2.3. La Lengua de Signos

La Lengua de Signos (LS) es una lengua natural que, al ser gestual, se puede utilizar como un SAAC gestual para las personas sordas. Esta lengua 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.

Esta lengua es 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.

La Lengua de Signos es considerada como SAAC gestual sin ayuda, es decir, no necesita apoyo de ningún sistema de información (imágenes, 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:

⁷https://play.google.com/store/apps/details?id=com.spreadthesign.androidapp_paid&hl=es

⁸<https://apps.apple.com/ni/app/spreadthesign/id438811366>

- **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:** la audición es el órgano que más influye en la educación, ya que es vía de adquisición del lenguaje. El uso de estos sistemas es determinante en el desarrollo intelectual de las personas, 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 sólo 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)⁹. 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 oficiales: la Lengua de Signos Española (LSE) y la Lengua de Signos Catalana (LSC). La LSE es la LS 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. A continuación, se explica el conocimiento básico de estas normas en lo referente a la gramática, fonología, sintaxis y morfología.

2.3.1. Fonología de la Lengua de Signos Española

La fonología (Muñoz y et.al., 2002), en lo relativo a la LSE, se refiere al estudio de la estructura y organización interna de los signos.

⁹<https://www.ine.es/>



Figura 2.7: Representación de la configuración de la mano en Lengua de 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 2.7)
- **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 2.19, van acompañados de dirección para indicar a qué persona se refiere el emisor. En la Figura 2.8 se muestran las distintas direcciones que pueden acompañar a los gestos.
- **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.
- **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.

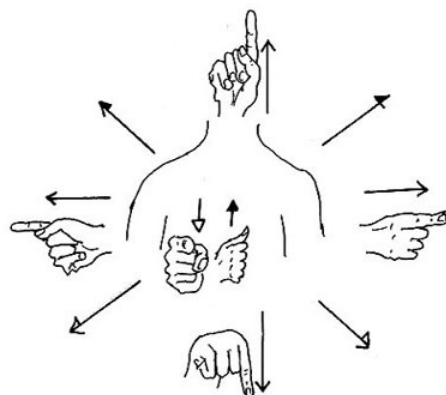


Figura 2.8: Representación de la orientación de las manos en Lengua de Signos

- **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 2.9.
- **Movimiento** de las manos al realizar un signo. Puede ser giratorio, vaivén o quebrado.
- **Componente no manual** del signo, que puede ser desde expresiones faciales hasta movimientos corporales del individuo.

2.3.2. Sintaxis de la Lengua de Signos Española

Las oraciones en la LSE no se estructuran igual que en el Castellano. La LSE es una lengua analítica, es decir, la estructura de las oraciones es muy simple, tendiendo a simplificar las frases para comunicar la información de la manera más concisa posible, al igual que lenguas como Japonés o Alemán.

La estructura básica de las oraciones se compone de sujeto-objeto-verbo, aunque existen excepciones. Por ejemplo, en la oración “*Él come patatas*” en LSE se 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.



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



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

2.3.3. 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 signarlos, 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



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



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

“*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 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*”.



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

- 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”.

2.3.4. Verbos de la Lengua de Signos Española

El verbo en la LSE (Fernández, 2008) 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



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

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 2.19, 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 podemos observarlo en la Figura 2.20, en la que se muestra el signo de la palabra “*señalar*”.

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



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



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



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

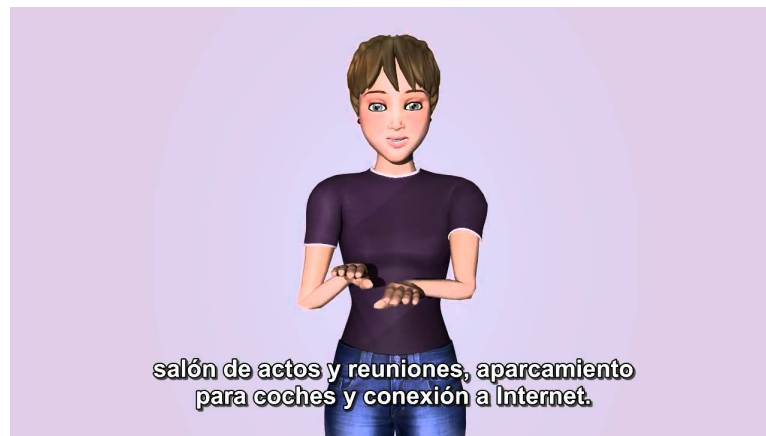


Figura 2.21: Maya, avatar usado por TextoSign.

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.

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, el cual observar en la Figura 2.21. Se trata de herramienta de pago y es utilizada hoy en día por La Caixa, Adif, Inturjoven y otras diversas entidades.

¹⁰<http://text2sign.es/>

¹¹<http://textosign.es/>

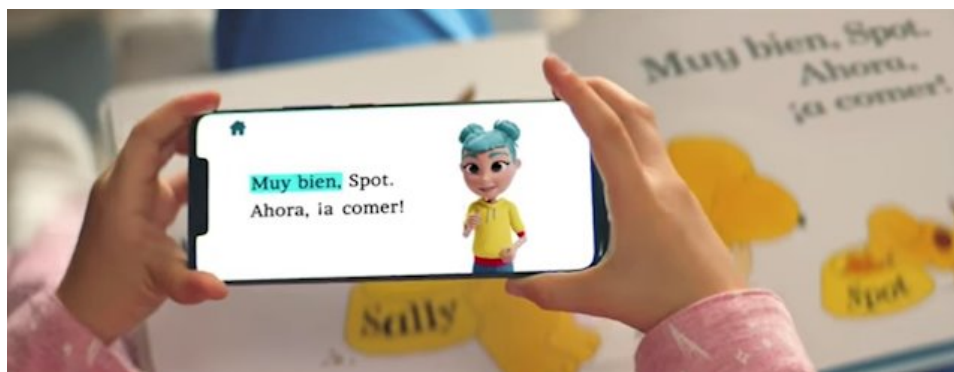


Figura 2.22: Imagen de la aplicación StorySign.

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 2.22.

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 2.23. 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.

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.

¹²<https://play.google.com/>

¹³<https://play.google.com/>



Figura 2.23: Imagen de la aplicación TeCuento.

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

Capítulo 3

Tecnologías utilizadas

Controlar la complejidad es la esencia de la programación".

Brian Kernigan

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

¹<https://www.w3.org/>

3.1.1. Servicios Web SOAP

Los Servicios Web SOAP (IBM Corporation, 2014) 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.

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 (Manuel Rosa Moncayo, 2018) 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 transferencia 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 son ²:

- Permiten una mayor operabilidad entre las aplicaciones software sin importar sus propiedades ni la plataforma en la que estén instalados.
- Los servicios Web 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 son ³:

- 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.

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

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

3.1.5. Flask

Para el desarrollo del proyecto se ha decidido utilizar un servidor Flask para montar la API REST.

Flask (Openwebinars, 2019) 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 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 nuestra API con este framework son las siguientes:

- Rapidez y facilidad en la instalación y configuración, a diferencia de otros frameworks como Django, que tiene una curva de aprendizaje mucho más baja.
- Es compatible con Python. Nuestra API está implementada 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 (Wikis FDI, 2017), 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.
- Cuenta con una muy buena documentación.

3.2. Procesamiento del Lenguaje Natural

El Procesamiento del Lenguaje Natural conocido como 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.

3.2.1. 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 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.

3.2.2. Técnicas de PLN

El procesamiento del lenguaje natural consta de distintas etapas, las cuales suelen variar de acuerdo al estudio que se va a realizar (Orellana Cordero y Trujillo Orellana, 2018). Algunas de ellas son:

- **Tokenización:** realiza la división del texto en palabras o frases.
- **Chunking:** Permite extraer conjuntos de palabras para que no pierdan el significado que forman entre ellas. Un ejemplo sería “El libro electrónico” en vez de separarlo en tokens lo separaría por grupos “el ” y “libro electrónico”.
- **Etiquetado gramatical:** Asigna a cada palabra del texto una categoría gramatical (sustantivo, adjetivo, verbo, etc).
- **Entidades Nombradas:** extraer y clasificar conjuntos de palabras en categorías predefinidas como personas, lugares, organizaciones, etc.

3.2.3. 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.

3.2.3.1. 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. Actualmente es considerada una de las mejores herramientas para el PNL ya que proporciona modelos en nueve idiomas 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 del mercado.

3.2.3.2. NLTK

Natural Language ToolKit (NLTK)⁵ es una librería dedicada al análisis del lenguaje natural bajo el lenguaje de programación 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

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

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

PLN. Esta biblioteca es muy útil cuando se quiere construir un modelo desde cero ya que proporciona acceso a muchos algoritmos y facilita el uso de algoritmos propios.

3.2.3.3. 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.

3.2.3.4. 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 más rápido y preciso que cualquier otra biblioteca y es muy sencilla de aprender a manejar gracias a su manual de uso.

3.3. Python

Python (Álvarez, 2003) es un lenguaje de programación desarrollado a lo largo de la década de los años 80 por Guido Van Rossum. El objetivo del creador era desarrollar un lenguaje de programación orientado a objetos de uso sencillo que sirviese para programar diversos tipos de tareas, no limitarlo a un solo tipo de uso. Sus principales características son:

- **Dinámico:** Se puede utilizar para desarrollar desde simples scripts o páginas web hasta para servidores de gran potencia.
- **Interpretado:** El programador no necesita realizar el paso de compilar el código, ya que Python se encarga de hacer esa compilación por sí mismo de manera que el desarrollador no tiene que realizar ese paso extra.
- **Multiplataforma:** Se puede utilizar en cualquier sistema informático, siempre y cuando se haya instalado previamente un intérprete compatible con él.

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

- **Interactivo:** Python contiene un intérprete por línea de comandos, a través del cual se pueden introducir sentencias y ver en tiempo real el resultado de dicha sentencia.
- **Orientado a objetos:** el código se estructura en elementos llamados clases, a través de las cuales se crean los objetos con funcionalidades específicas.
- **Funciones y librerías:** Python incluye una serie de funciones incluidas de base, como por ejemplo, la manipulación de strings, números, etc. A parte, existen multitud de librerías que se pueden importar al desarrollar un programa para cubrir necesidades específicas, como por ejemplo, el procesamiento del lenguaje natural.
- **Sintaxis clara:** La estructura del código se basa en los márgenes de obligado cumplimiento, siendo así muy visual. Esto hace que sea más sencillo distinguir las distintas partes del código.
- **Licencia de código abierto** (Python Software Foundation, 2020): Licencia compatible con la Licencia pública general de GNU, la cual permite que el usuario final pueda tener acceso al código, descargarlo, copiarlo y manipularlo.

Se ha decidido utilizar Python en este proyecto debido a la facilidad de uso y a la existencia de diversas librerías de procesamiento de lenguaje natural, como Spacy y NLTK.

3.4. Nginx

Nginx ⁷ 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 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:

- 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.

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

- 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 página web escrita en HTML, CSS y Javascript, cuya finalidad es proporcionar una interfaz gráfica para el uso de la API.

3.5. Google Colaboratory

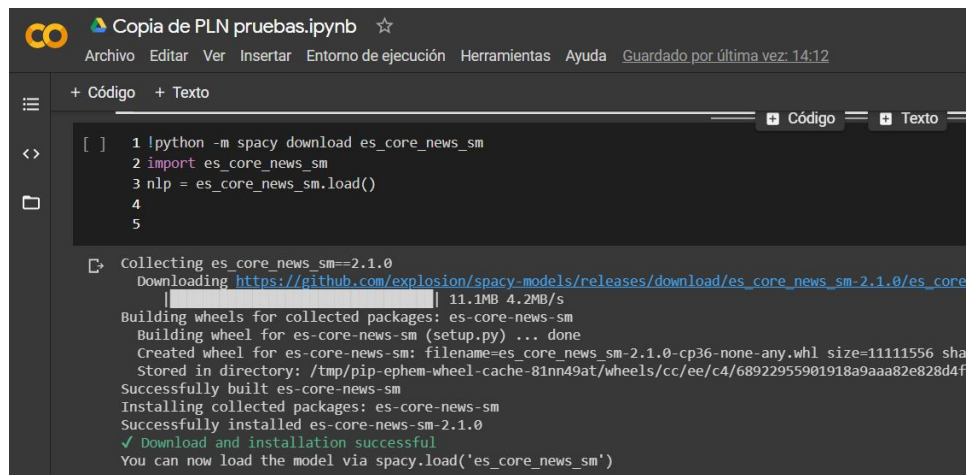
Google Colaboratory⁸ es una herramienta gratuita de Google que permite programar en Python y ejecutar el código en tiempo real. El usuario final no necesita instalar nada en su propio sistema, ya que se pueden importar las librerías necesarias y toda la carga del procesamiento se localiza en los servidores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

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.

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



The image shows a Google Colaboratory notebook interface. At the top, the title bar reads "Copia de PLN pruebas.ipynb" with a star icon. Below the title bar is a menu with options: Archivo, Editar, Ver, Insertar, Entorno de ejecución, Herramientas, Ayuda, and Guardado por última vez: 14:12. The notebook has two tabs: "Código" (active) and "Texto". The code cell contains the following Python code:

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

The output of the code cell shows the following text:

```
Collecting es_core_news_sm==2.1.0
  Downloading https://github.com/explosion/spacy-models/releases/download/es_core_news_sm-2.1.0/es_core_news_sm-2.1.0-py3-none-any.whl (11.1MB)
    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=...
      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

Capítulo 4

Metodologías utilizadas

*"Juntarse es un comienzo. Seguir juntos
es un progreso. Trabajar juntos es un
éxito".*

Henry Ford

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 Ágiles

Las metodologías ágiles (Iebschool, 2019b) 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.

La aplicación de metodologías ágiles permite involucrar a nuestros tutores a lo largo de todo el proyecto. En cada etapa se informa de los logros y progresos del mismo, con la visión de involucrarles directamente para sumar su experiencia y conocimiento, y así optimizar las características del resultado final, obteniendo en todo momento una visión completa de su estado. La continua interacción entre los alumnos y los tutores tiene como objetivo asegurar que el resultado final sea exactamente lo que se busca y necesita. Además, es posible detectar de forma rápida tanto errores como problemas que puedan 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 es posible gracias a las constantes reuniones que se realizan tanto entre

los integrantes del grupo como entre los alumnos y los tutores. Estas reuniones se explican con más en detalle en la sección 4.4.

Además, el trabajo se realiza con mayor velocidad y eficiencia, ya que se trabaja a través de entregas parciales del proyecto. De este modo, es posible entregar en el menor intervalo de tiempo posible una versión mucho más funcional de la aplicación y de la memoria. 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 4.5

Desde el comienzo, una de las mayores ventajas de la utilización de metodologías ágiles es la mejora de la implicación de los integrantes del equipo. Estas metodologías permiten 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. La herramienta utilizada para organizar todo el trabajo ha sido Trello, explicada en la sección 4.3.

Existen diversas modalidades de metodologías ágiles, algunas de las más conocidas son Scrum o Programación Extrema (XP). Concretamente, para el desarrollo de este proyecto se ha optado por la utilización de Kanban, que se explica en detalle en la sección 4.2.

4.2. Kanban

Kanban (Iebschool, 2019a) es una metodología agile cuyo objetivo es gestionar de manera general cómo se van completando las tareas a llevar a cabo. Kanban es una palabra japonesa que significa 'tarjetas visuales', donde Kan es 'visual', y Ban corresponde a 'tarjeta'.

La principal ventaja de esta metodología es que es muy fácil de utilizar, actualizar y asumir por parte del equipo. Además, 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 gestionar las tareas a realizar por los miembros del equipo ha sido Trello, explicada con detalle en la sección 4.3.

En Kanban existen una serie de principios básicos (Kanbantool, 2019) que ayudan a obtener el máximo rendimiento del flujo de trabajo. Algunos de los que se han aplicado en nuestro proyecto son los siguientes:

- **Visualizar lo que hace el flujo de trabajo:** una visualización de

todas las tareas contribuye a que todos los miembros del equipo nos mantengamos al corriente de nuestro trabajo. Todos los miembros del equipo son capaces de trabajar en el mismo tablero y colaborar en tiempo real. Además, el tablero digital Trello, a través de su aplicación móvil, permite acceder al flujo de trabajo desde cualquier sitio, compartir tareas con facilidad y comunicarse entre el equipo.

- **Limitar la cantidad de Trabajo en Proceso:** establecer metas asequibles y limitar los trabajos en proceso para prevenir el exceso de cantidad de tareas, que serían muy difíciles de completar.
- **Lectura fácil de indicadores visuales:** con Kanban es sencillo conocer lo que está ocurriendo de un solo vistazo. Se utilizan tarjetas de colores para distinguir los tipos de trabajo, prioridades, o fechas límite.

4.3. Trello

A lo largo de todo el proyecto se ha utilizado la herramienta Trello¹ para el reparto claro y equitativo de tareas entre los componentes del equipo de trabajo. Trello consiste en un conjunto de tableros, a los cuales se les pueden añadir una serie de tareas con diferentes estados. Esto se puede observar en la Figura 4.1, donde se muestra un ejemplo de tablero utilizado en el desarrollo del proyecto. Esta herramienta permite ver de manera sencilla el estado del proyecto, facilitando así la organización de las tareas a realizar. En el desarrollo de “Text2LSE” se han utilizado tres tableros distintos:

- **Investigación:** Incluye las tareas relacionadas con toda la investigación referente al proyecto, como por ejemplo LSE, Python, PLN, etc.
- **Desarrollo:** Este tablero contiene las distintas tareas de desarrollo de código, tanto de la aplicación web como de la API.
- **Memoria:** En este tablero se encuentran las distintas tareas relacionadas con el desarrollo y corrección de los apartados de la memoria.

Para una mayor organización, cada tablero se ha estructurado en tres estados:

- **Lista de Tareas:** Lista de tareas asignadas a un miembro del equipo en concreto que todavía están sin empezar.
- **En proceso:** Tareas que el propietario de la tarjeta tiene en proceso de desarrollo.

¹<https://trello.com/>

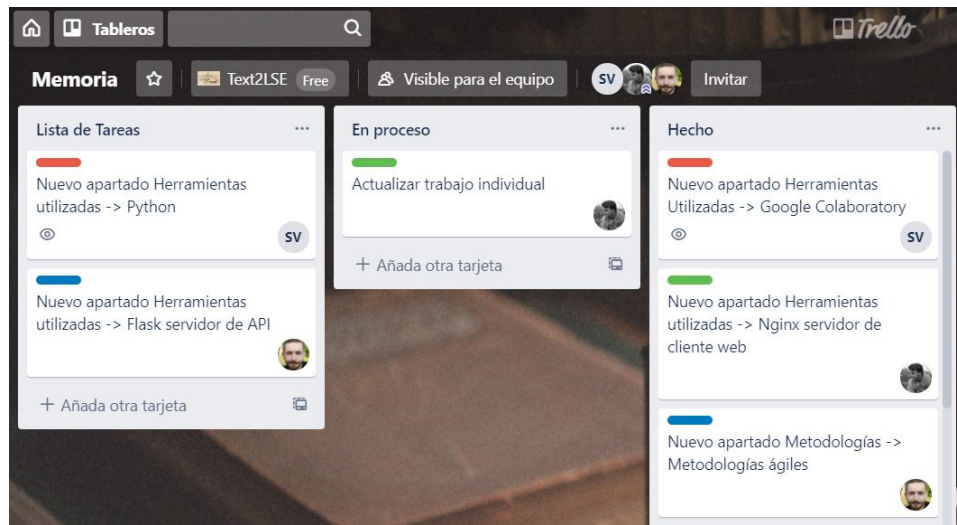


Figura 4.1: Tablero “*Memoria*” utilizado en el proyecto.

- **Hecho:** Tareas completadas y subidas al repositorio de GitHub.

Cada miembro del equipo ha sido el encargado de actualizar en tiempo real el estado de las tarjetas que tenía asignadas. De esta manera se ha conseguido tener una buena organización con respecto a las tareas a realizar.

4.4. 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 Skype² 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 entregas realistas y factibles.

²<https://www.skype.com/es/>

4.5. Control de versiones

4.5.1. GIT

Para el desarrollo del proyecto se ha utilizado Git³ que es un sistema de control de versiones distribuido que sirve para trabajar en equipo de una manera muy sencilla y optimizada. Git permite al usuario tener un control absoluto de su proyecto pudiendo ver todos los cambios realizados en la aplicación y en el código por cada uno de los miembros o incluso volver a versiones anteriores del proyecto.

Sus principales herramientas son:

- **Repository:** es un directorio donde se almacenan los archivos de tu proyecto. Puede estar ubicado en el almacenamiento de GitHub o en un repositorio local en tu computadora y tenerlos sincronizados.
- **Branch:** es una copia de tu repositorio cuyo desarrollo es independiente al repositorio central u otras ramas. Una vez realizados los cambios se puede combinar tu rama con otras ramas y con el repositorio central.
- **Commits:** son los cambios realizados en los archivos de tu repositorio local.
- **Pull:** acción que actualiza el repositorio local de tu ordenador con la última versión del proyecto alojado en Github.
- **Push:** permite subir a Github los commits realizados en el repositorio local.
- **Merge:** permite incorporar y fusionar los cambios con la rama principal.

4.5.2. GITHUB

GitHub⁴ es una plataforma que proporciona una interfaz que mediante las herramientas del control de versiones Git permite ver y llevar el registro de todos los cambios realizados en el proyecto, organizarlos y resolver cualquier conflicto que pueda surgir.

Github también funciona como una red social para desarrolladores ya que permite a otros usuarios ver y colaborar en tus repositorios así como resolver dudas que se tengan.

³<https://git-scm.com/>

⁴<https://help.github.com/>

4.5.3. Estructura de Trabajo con GIT y GITHUB

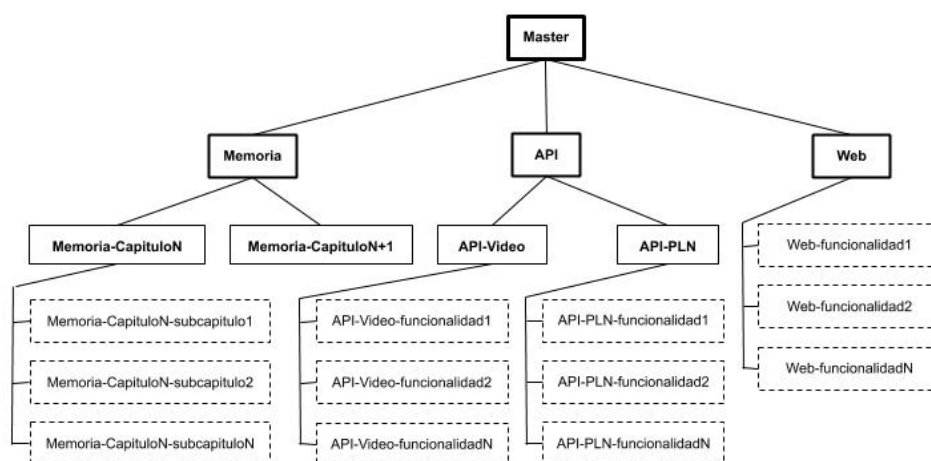
A la hora de desarrollar el proyecto con la herramienta Git, se ha establecido una estructura común para un correcto desarrollo del proyecto y poder gestionar así de manera eficiente y segura los cambios realizados.

La estructura desarrollada es la siguiente:

MASTER: Rama principal del proyecto donde todo ha sido revisado y está óptimo funcionamiento y 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:
 - 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*
- **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*

Nota: Todas las ramas creadas por funcionalidad se pueden borrar una vez se dejen de usar y se haya realizado un previo merge a su rama principal.

Figura 4.2: Estructura “*Git*” utilizada en el proyecto.

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 texto en tiempo real. Está basada en servicios web, los cuales están disponibles para todo el mundo de manera gratuita. Los vídeos que utilizan los servicios web del proyecto para realizar la traducción a LSE son recursos de los signos que ofrece 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 explica 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.

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 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,

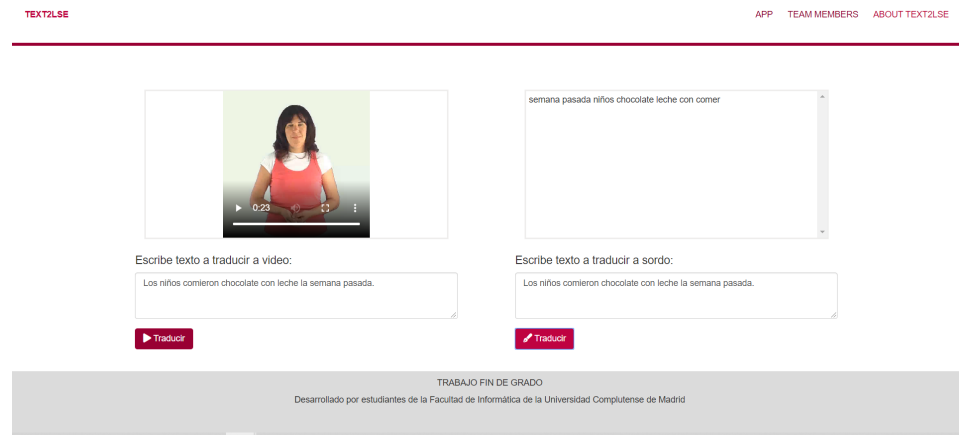


Figura 5.1: Aplicación web Text2LSE

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. Al no disponer de una API que nos devolviese directamente los vídeos de la LSE de ARASAAC, se descargaron desde su página web y se almacenaron en el servidor para poder acceder a ellos.

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. 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, así como los métodos utilizados en cada uno de ellos. Para comenzar COMPLETAR

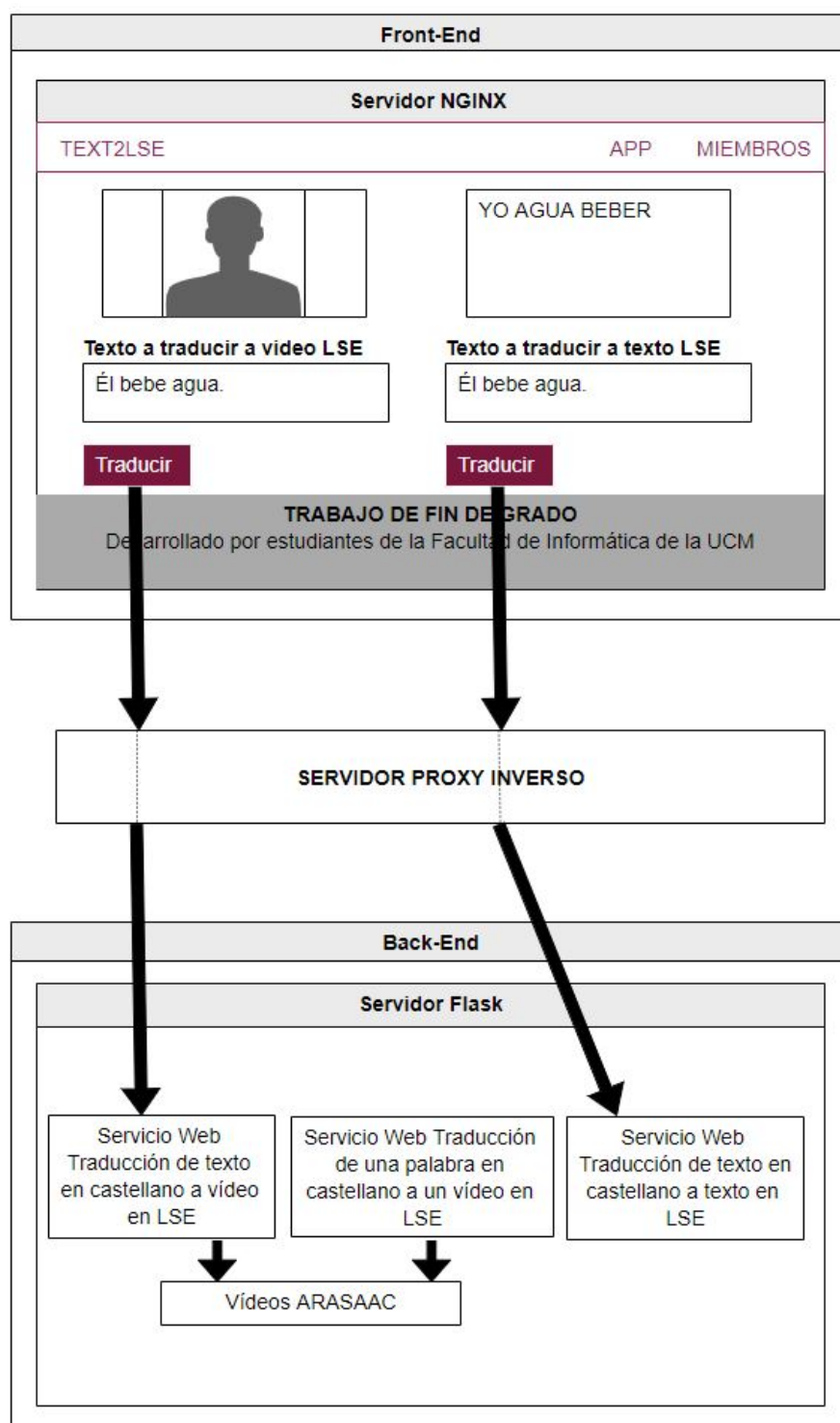


Figura 5.2: Arquitectura de Text2LSE

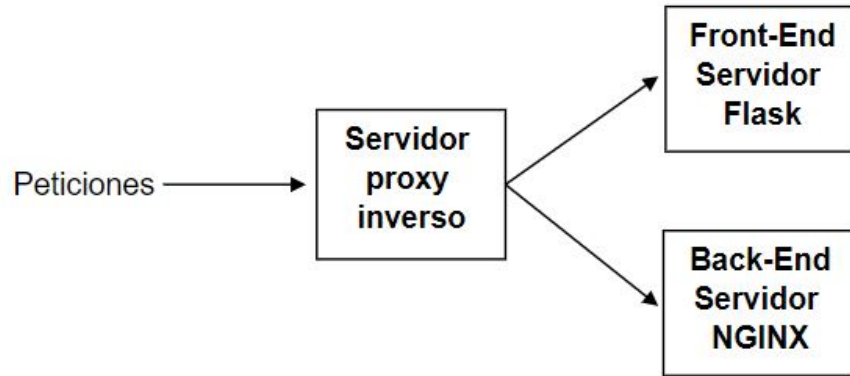


Figura 5.3: Esquema Proxy Inverso

5.2.1. Servicios

5.2.1.1. Servicio web para obtener el video de un signo en LSE de una determinada palabra

Este servicio permite obtener el video del signo en LSE correspondiente a una palabra determinada. 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 desde la siguiente url:

```
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 mensaje de error. Este flujo lo podemos observar en la Figura 5.4.

5.2.1.2. Servicio web para Traducción de texto a vídeo en LSE

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

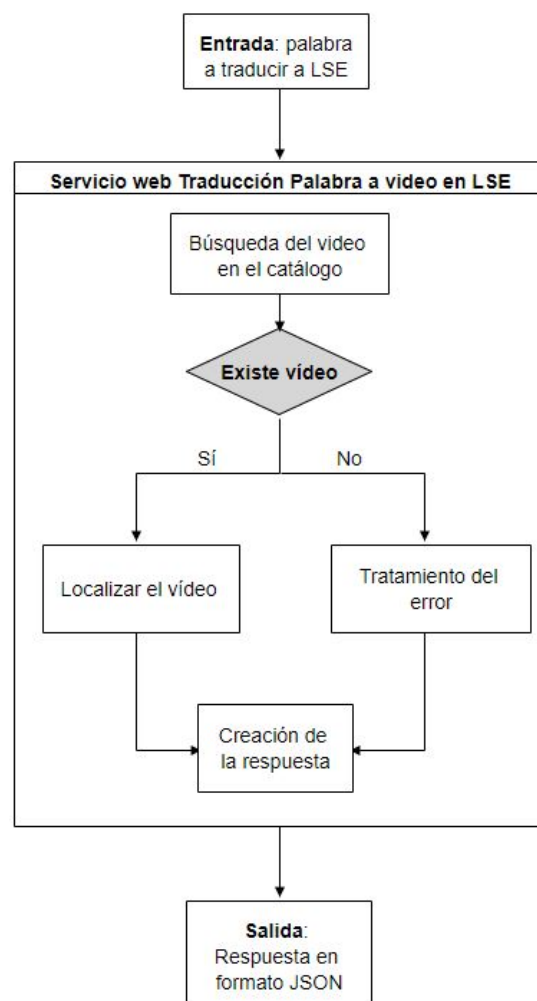


Figura 5.4: Flujo del servicio de Traducción de una palabra a vídeo en LSE

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 una petición POST a la API en la siguiente URL:

```
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>' }
```

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 a 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 5.5.

5.2.1.3. 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:

```
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. El flujo lo podremos ver en la Figura 5.6.

5.2.2. Procesamiento del Lenguaje Natural de TEXT2LSE

Como se ha visto en el apartado anterior ambos Servicios Web necesitan realizar un Procesamiento de Lenguaje Natural para poder adaptar el texto escrito en castellano a la LSE. En este apartado se explicará la implementación y el desarrollo del PLN utilizado en ambos servicios.

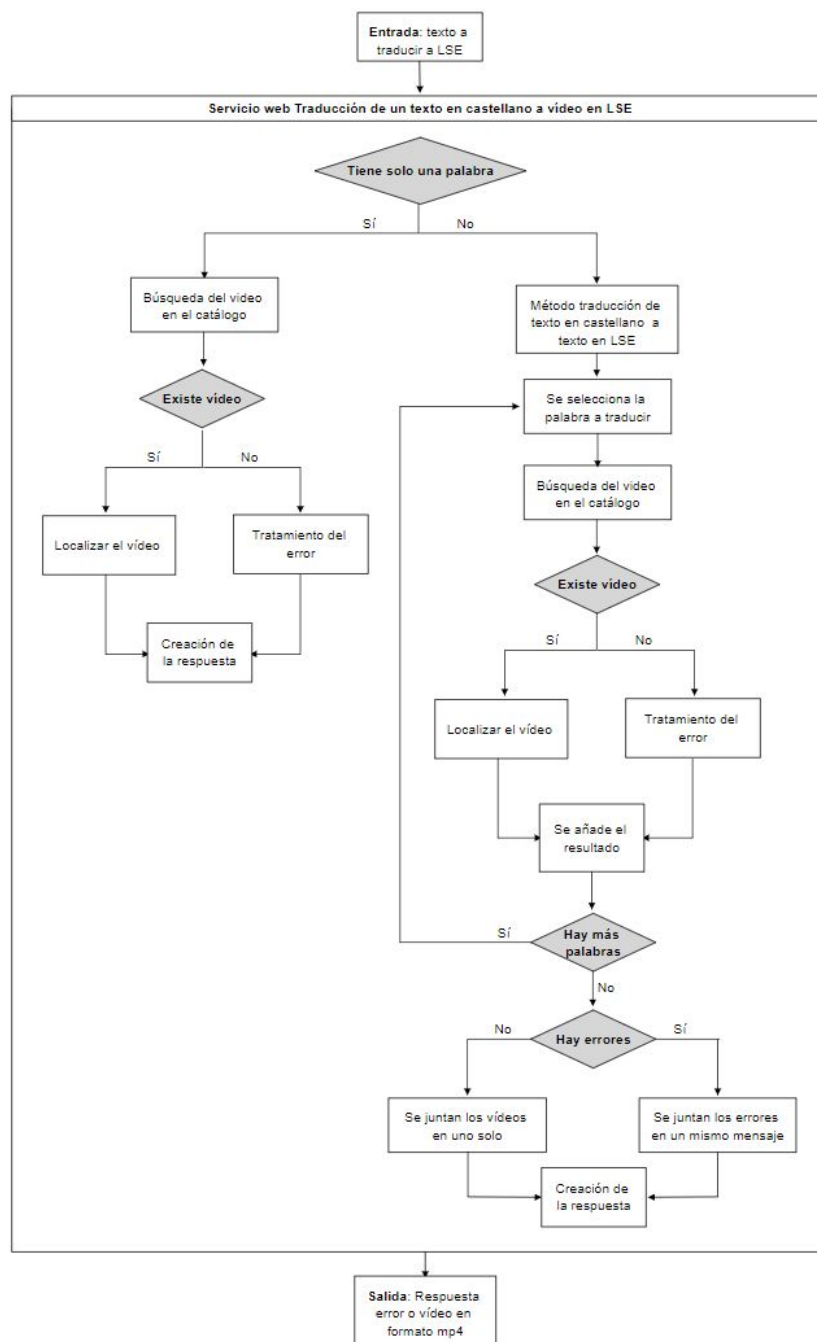


Figura 5.5: Flujo servicio de traducción de texto en castellano a video en LSE

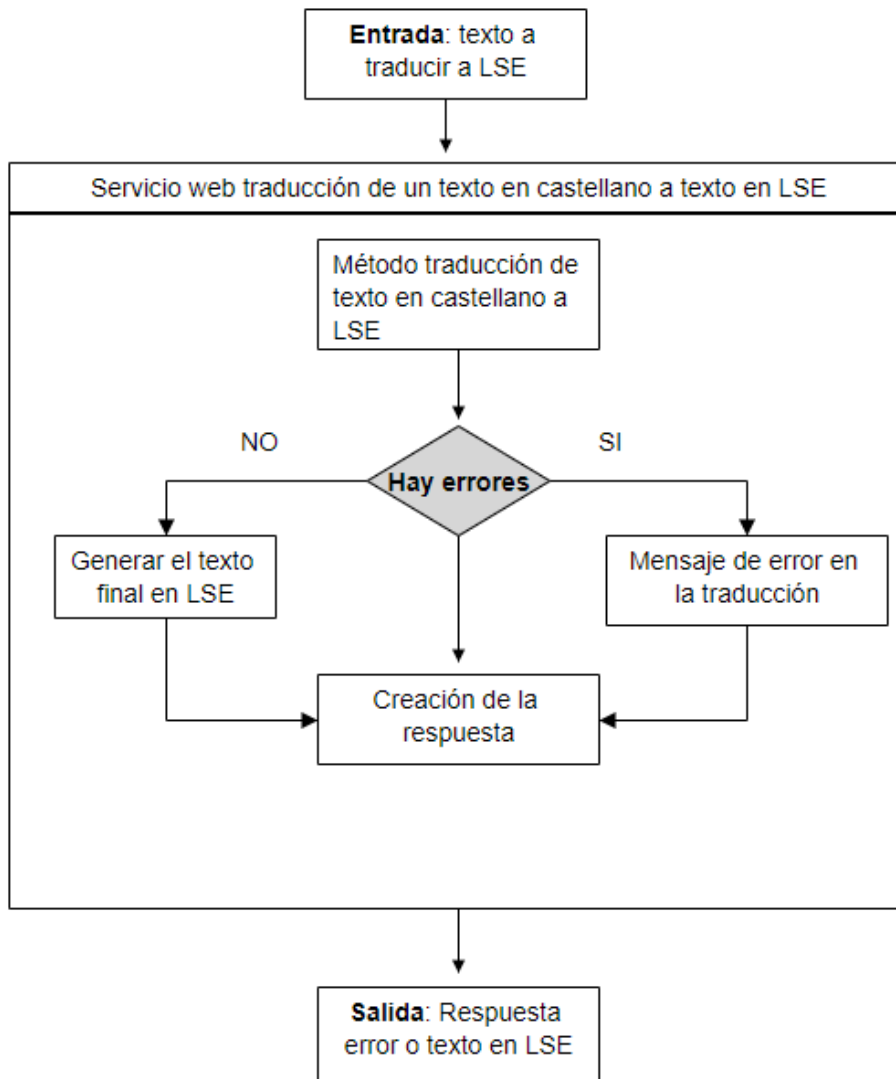


Figura 5.6: Flujo servicio de traducción de texto en castellano a texto en LSE

5.2.2.1. Implementación del PLN

El Procesamiento de Lenguaje Natural para el proyecto Text2LSE ha sido implementado desde cero por los tres integrantes del equipo. Debido a la singularidad de Lengua de Signos y la falta de documentación de otros servicios de PLN ya implementados de los cuales se podrían haber hecho uso, se llegó a la conclusión de realizar un procesamiento propio 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 cuales no. La elección a la hora de desarrollar el Procesamiento del Lenguaje para nuestro proyecto estaba entre un sistema basado en reglas o mediante el uso de aprendizaje 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 y compuestas a LSE.

5.2.2.2. Desarrollo del PNL

El desarrollo del Procesamiento del Lenguaje Natural está dividido en tres partes diferenciadas que son:

Recopilar la información: para poder empezar con el desarrollo se necesita conocer la información de cada palabra dentro de la oración. 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.
- **Tags:** información de la palabra como el género, el número, el tiempo verbal, etc.
- **Pos:** tipo de palabra como nombre, adjetivo, etc.
- **Lema:** Lema de palabra.

Por ejemplo la información que se obtendría del token ?merendaban? sería 5.7:

Análisis sintáctico y orden de la oración: una vez separado la oración en tokens hay que diferenciar cuales forman parte del sujeto y cuales



Figura 5.7: Información del token merendaban

del predicado. Para ello hay que recorrer el árbol de dependencias generado por Spacy 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’.

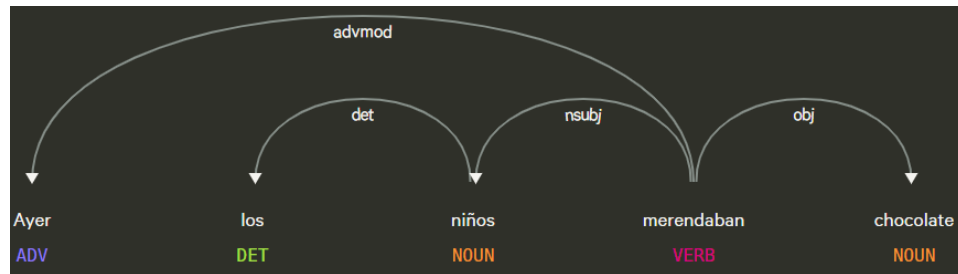


Figura 5.8: Displacy grafo de dependencias

Como se puede apreciar en el diagrama 5.8 hay un token ‘merendaban’ del cual dependen tres tokens [Ayer, niños, chocolate] y de ellos también dependen otros y así sucesivamente.

Estas dependencias se traducen en un árbol binario donde el ROOT es la raíz como podemos ver en la Figura 5.9:

- **El sujeto:** Para poder obtener el sujeto hay que recorrer el árbol buscando un 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, pero no todos ellos formarán parte del sujeto. Por lo tanto mediante reglas se analiza qué palabras se descartan del nuevo sujeto de la oración en Lengua de Signos. En el ejemplo anterior “Ayer los niños merendaban chocolate”:

- **Antes de PLN:** [Los, niños]
- **Después de PLN:** [niños]

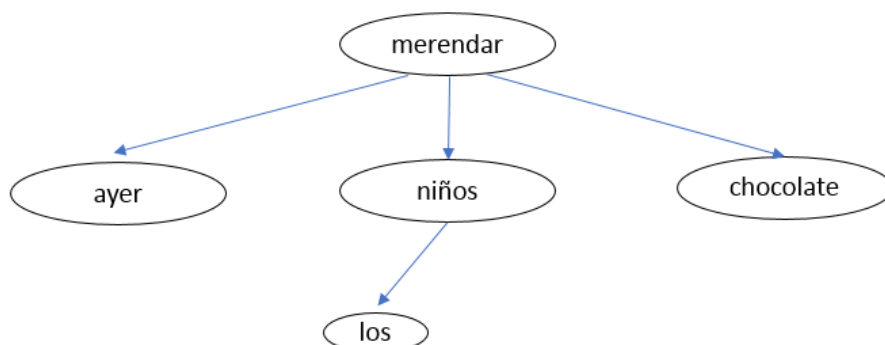


Figura 5.9: Árbol binario de una oración

- **El predicado:** El predicado se obtiene de manera muy parecida al sujeto. Como ya se ha encontrado un token con dependencia = “nsuj” y de él todos sus hijos, los demás tokens excepto el root, el cual se analiza de forma independiente, pertenecen al predicado y hay que analizarlos para ver si se descartan o no.

Una vez completadas las reglas y se haya completado el recorrido del árbol ya se tendrá el predicado guardado en una lista. En el ejemplo “Ayer los niños estaban en el parque”:

- **Antes de PLN:** [Ayer, estaban, en, el, parque]
- **Después de PLN:** [Ayer,parque]

Una vez terminado el análisis sintáctico de la frase “Ayer los niños estaban en el parque” se obtiene el Sujeto: [niños], el Predicado: [Ayer,parque] y el Root: [estar]

Como se ve la oración no tiene ningún sentido tan solo está separada sintácticamente por lo que ahora la hay que ordenarla para que cumpla la estructura de la LSE. Para ello primero se recorren las listas del sujeto y del predicado generadas en el análisis sintáctico cuyos tokens van siendo ordenados mediante reglas.

El orden general que se ha seguido para estructurar las frases ha sido el siguiente:

PALABRAS DE TIEMPO + SUJETO + OBJETO + VERBO.

En el ejemplo utilizado anteriormente, la frase ordenada quedaría:

“Ayer los niños merendaban chocolate en el parque” quedaría ahora como “Ayer niños chocolate parque merendar”

Análisis morfológico: Una vez realizado el análisis sintáctico y orden de la oración hay que hacer un análisis morfológico detectando el tiempo verbal de la oración, el género y el número de cada token, así como los determinantes posesivos los cuales son sustituidos por pronombres personales.

Para entender mejor cómo funciona este apartado se va a usar de ejemplo la oración: “Mis hermanos están de vacaciones”

Tiempo verbal: Si es PRESENTE la frase no se ve alterada, pero por ejemplo si está en PASADO o en FUTURO se añaden al principio de la frase los adverbios de tiempo “ayer” y “mañana” respectivamente ya que como se comentará en el siguiente apartado no tenemos suficientes palabras por lo que no podemos expresar el pasado o futuro de otra forma.

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 obtengan el resultado por pantalla, haciendo uso de los servicios web de traducción implementados mediante llamadas Fetch. En la Figura 5.1 se muestra la interfaz de la aplicación, donde se puede observar que se han desarrollado dos funcionalidades:

- **Traducción a vídeo:** esta sección consta de un input donde el usuario podrá introducir el texto en castellano que desee traducir, y tras pulsar el botón “traducir”, se realizará una llamada Fetch al servidor proxy, que redirigirá la petición a la API. Una función Fetch se encargará de recibir el resultado de la petición, incrustando el vídeo en el código HTML. 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 texto LSE:** de la misma manera que en la traducción a vídeo, en esta sección el usuario introducirá un texto en castellano y, tras 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 5.10 podemos observar la interfaz vista desde un dispositivo móvil.

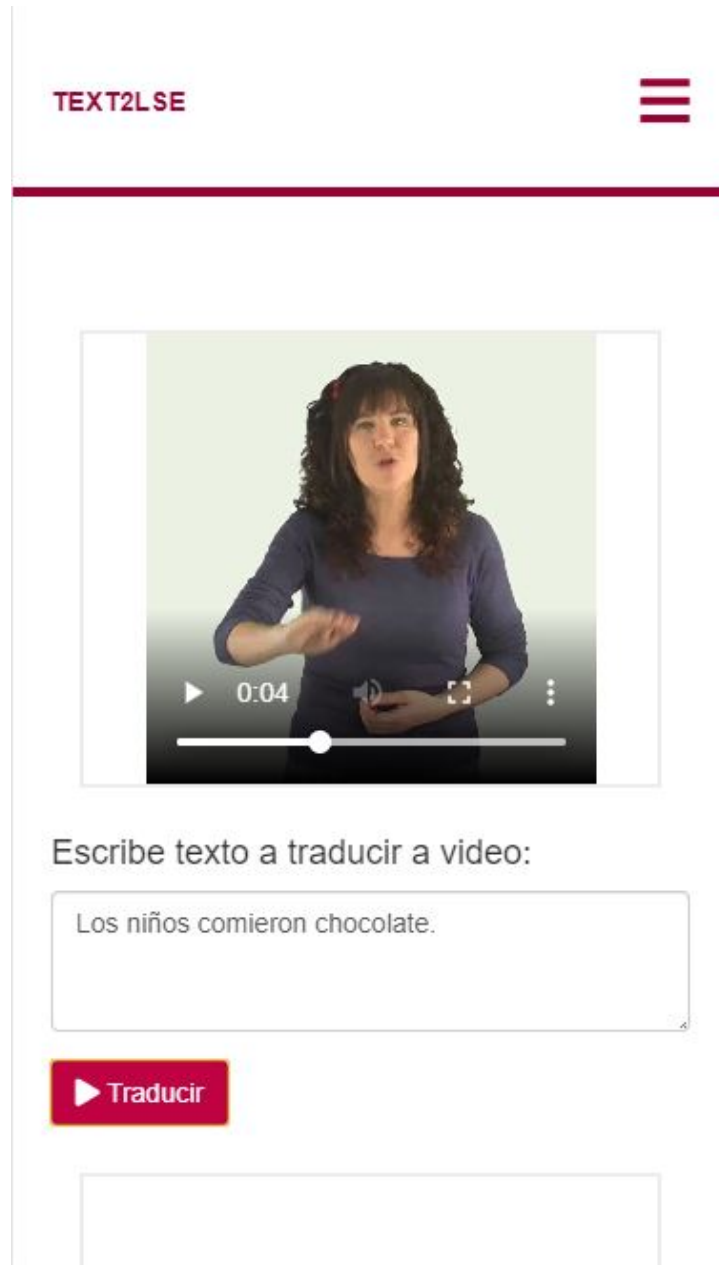


Figura 5.10: Aplicación Web vista desde un dispositivo móvil

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 llegamos a la conclusión del proyecto que queríamos hacer, 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 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 y comenzamos a desarrollar la primera parte de la memoria con los conocimientos adquiridos en la investigación previa.

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 enviarlo 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 video mp4.

En la fase posterior, me encargué de corregir mis apartados correspondientes de la memoria, y de desarrollar los apartados de Python y Google Colaboratory dentro de herramientas utilizadas, y de metodologías como Trello y Reuniones. Respecto a la web, ayudé a mi compañero Alejandro con el cambio de las llamadas AJAX a Fetch. También me encargué de estructurar la función de la API que devuelve un video con varios signos y de desarrollar las llamadas a las funciones que devuelven diferente tipo de información en Json. Respecto a PLN, en un principio me centré en desarrollar un algoritmo que recorra un árbol de dependencias desde el verbo a partir de una frase.

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, que realiza 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. 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 nuestra API cuente con varias opciones de llamada. Mis compañeros Miguel y Sara se encargaron de dividir esta función.

Por otro lado, decidimos cambiar las llamadas AJAX de la web por llamadas Fetch, y yo me encargué de programar estas llamadas con ayuda de mi compañera Sara.

Junto con el trabajo técnico, también participé en la redacción y corrección de la memoria, en la que trabajamos los tres compañeros conjuntamente. Me encargué de la corrección del capítulo 1 y parte del capítulo 2. Además, desarrollé el apartado de Flask del capítulo 3, y metodologías ágiles y kanban en el capítulo 4.

Por otro lado, decidimos cambiar las llamadas AJAX de la web por llamadas Fetch, y yo me encargué de programar estas llamadas con ayuda de mi compañera Sara.

Junto con el trabajo técnico, también participé en la redacción y corrección de la memoria, en la que trabajamos los tres compañeros conjuntamente. Me encargué de la corrección del capítulo 1 y parte del capítulo 2. Además, desarrollé el apartado de Flask del capítulo 3, y metodologías ágiles y kanban en el capítulo 4.

6.3. Miguel Rodríguez Cuesta

En 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 la puesta en común y de la primera corrección, 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 de proyecto 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.

En cuanto al desarrollo de código mi parte era realizar el apartado capaz de devolver una palabra en video mediante el método Get así como investigar como hacer una buena gestión de errores de la Api.

De forma independiente me estuve informando sobre como usar Git de forma correcta y así explicar a mis compañeros como usarlo mediante visual studio y también recibir sus consejos pues ellos también sabían muchas cosas nuevas. También me he informado e investigado como empezar a realizar el PLN para poder dividirnos el trabajo más adelante.

Para mí la gran importancia de esta segunda parte del trabajo ha sido la gran gestión del proyecto que hemos hecho que nos ha llevado más tiempo incluso que el desarrollo de código. Hemos organizado una estructura la cual nos permite trabajar de forma independiente y de forma eficiente en distintas partes del proyecto.

Bibliografía

*Y así, del mucho leer y del poco dormir,
se le secó el cerebro de manera que vino
a perder el juicio.*

Miguel de Cervantes Saavedra

- ÁLVAREZ, M. A. Qué es Python. Disponible en <https://desarrolloweb.com/articulos/1325.php>. Informe técnico, Desarrollo Web, 2003.
- ARASAAC. Qué son los SAAC. Disponible en <http://www.arasaac.org/aac.php>. Informe técnico, ARASAAC, 2019.
- DISCAPNET. Discapacidad auditiva. Disponible en <https://www.discapnet.es/areas-tematicas/salud/discapacidades/auditivas/discapacidad-auditiva>. Informe técnico, Discapnet, 2019.
- FERNÁNDEZ, A. *La cantidad a manos llenas. La expresión de la cuantificación en la Lengua de Signos Española*. Fundación CNSE, 2008.
- IBM CORPORATION. *¿Qué es un servicio web?*. Disponible en https://www.ibm.com/support/knowledgecenter/es/SSMKHH_9.0.0/com.ibm.etools.mft.doc/ac55710_.html, 2014.
- IEBSCHOOL. Metodologías ágiles. Disponible en <https://www.iebschool.com/blog/metodologia-kanban-agile-scrum/>. Informe técnico, IEBSchool, 2019a.
- IEBSCHOOL. Metodologías ágiles. Disponible en https://www.iebschool.com/blog/que-son-metodologias-agiles-agile-scrum/#metodologias_agiles. Informe técnico, IEBSchool, 2019b.
- KANBANTOOL. Metodologías ágiles. Disponible en <https://kanbantool.com/es/metodologia-kanban>. Informe técnico, Kanbantool, 2019.
- MANUEL ROSA MONCAYO, J. *¿Qué es REST? Conoce su potencia*. Disponible en <https://openwebinars.net/blog/que-es-rest-conoce-su-potencia/>, 2018.

- MUÑOZ, F. y ET.AL., A. *Apuntes de Lingüística de la Lengua de Signos Española*. Fundación CNSE, 2002.
- OPENWEBINARS. Qué es flask. Disponible en <https://openwebinars.net/blog/que-es-flask/>. Informe técnico, Flask, 2019.
- ORELLANA CORDERO, M. y TRUJILLO ORELLANA, A. A. Evaluación de técnicas de procesamiento de lenguaje natural para textos cortos en lenguaje español. 2018.
- PYTHON SOFTWARE FOUNDATION. *History and License*. Disponible en <https://docs.python.org/3/license.html>, 2020.
- TORRES, S. *Sistemas alternativos de comunicación*. Ed. Aljibe, 2001.
- WIKIS FDI. *Qué es la licencia BSD*. Disponible en http://wikis.fdi.ucm.es/ELP/Licencia_BSD, 2017.