

Universidad Nacional de La Pampa, Facultad de Ingeniería

PROYECTO FINAL DE GRADO DE INGENIERÍA EN SISTEMAS

“A-e-ii: Herramienta para monitorear el deterioro del habla en pacientes con enfermedades neurodegenerativas”

Autora: Sofía Florencia Rodríguez

Grado académico: Ingeniería en Sistemas

Tutora: Dra. María Belén Rivera - Cátedras Análisis y Diseño de Sistemas I y Sistemas Distribuidos II. Ayudante en Análisis y Diseño de Sistemas II

Lugar de presentación: General Plco

Año: 2023

Fecha de Aprobación: 15 de Septiembre

Jurados: Alfonso, Hugo - Martín, María de los Ángeles - Riesco, Daniel Edgardo. Filiación: Facultad Ingeniería UNLPam.

Resumen: En algunas demencias neurodegenerativas, las personas pueden presentar una disminución de habilidades lingüísticas que son difíciles de cuantificar con pruebas estandarizadas. El monitoreo y análisis del habla en estos pacientes puede proporcionar información valiosa sobre la evolución de su discurso hablado. Dentro de las pruebas y tratamientos, se incluye a la lectura en voz alta pero, rara vez, es aplicada sistemáticamente como parte integral de la terapia y poco se sabe (aunque es altamente prescrita) respecto a su eficacia. Este trabajo presenta el desarrollo de una aplicación web, denominada “A-e-ii”, que utiliza funciones de servicios alojados en servidores en la nube para transcribir audios de lecturas de pacientes con deterioro neurodegenerativo derivado en afasia progresiva primaria. Con las transcripciones generadas, se pretende monitorear los cambios en el discurso a través del tiempo de dichos pacientes. Para realizar este monitoreo, se diseñaron métricas que permiten medir distintos aspectos en relación a la fluidez de la lectura. Y que además, sirven para una posterior evaluación, a partir de indicadores que interpretarán los valores medidos en el tiempo. Desde el punto de vista tecnológico, se combinaron algoritmos que transcriben discurso oral a texto y de reconocimiento de partes de un discurso.

Palabras clave: Deterioro neurodegenerativo, Procesamiento de lenguaje natural, Computación en la nube, Programación web

Abstract: In some neurodegenerative dementias, people may have a decline in language skills that are difficult to quantify with standardized tests. Speech monitoring and analysis in these patients can provide valuable information about the evolution of their spoken speech. Among the tests and treatments, reading aloud is included, but it is rarely applied systematically as an integral part of therapy and little is known (although it is highly prescribed) regarding its effectiveness. This work presents the development of a web application, called “A-e-ii”, that uses service functions hosted on cloud servers to transcribe audio readings from patients with neurodegenerative deterioration resulting in primary progressive aphasia. With the transcripts generated, the aim is to monitor changes in the speech of these patients over time. To carry out this monitoring, metrics are designed to measure different aspects in relation to reading fluency. And they also serve for a subsequent evaluation, based on indicators that will interpret the values measured over time. From a technological point of view, algorithms that transcribe oral speech into text and recognition of parts of speech are combined.

Key Words: Degenerative Disease, Natural Language Processing, Cloud Computing, Web Programming

Universidad Nacional de La Pampa
Facultad de Ingeniería

Tesis presentada para obtener el grado de
Ingeniera en Sistemas

A-e-i!: Herramienta para monitorear el deterioro del habla en pacientes con enfermedades neurodegenerativas

Sofía Florencia Rodriguez
Legajo N° 5544

Directora: Dra. María Belén Rivera

General Pico, La Pampa

Junio, 2023

Agradecimientos

Este trabajo le pone fin a una etapa en la que crecí mucho, no solo a nivel académico, sino también a nivel personal. Quiero agradecer a todas y cada una de las personas que estuvieron presentes de alguna manera a lo largo de estos años.

Primero que nada, gracias a la Universidad Nacional de La Pampa, y sobre todo a la Facultad de Ingeniería, por lo brindado a lo largo de estos años.

Gracias a la Dra. María Belén Rivera, por su supervisión y guía en el desarrollo de esta tesis.

Gracias a mis amigos, tanto a los que conocí en la Facultad como a los de toda la vida, por su compañía y afecto, aún estando a veces a varios kilómetros de distancia.

Gracias a mi mamá, mis hermanos, mis abuelos y al resto de mi familia, por ser mi sistema de apoyo más fuerte.

Por último, gracias a mi papá, por haber recorrido conmigo parte del camino y por ser mi ejemplo a seguir.

Resumen

En algunas demencias neurodegenerativas, las personas pueden presentar una disminución de habilidades lingüísticas que son difíciles de cuantificar con pruebas estandarizadas. El monitoreo y análisis del habla en estos pacientes puede proporcionar información valiosa sobre la evolución de su discurso hablado. Dentro de las pruebas y tratamientos, se incluye a la lectura en voz alta pero, rara vez, es aplicada sistemáticamente como parte integral de la terapia y poco se sabe (aunque es altamente prescripta) respecto a su eficacia. Este trabajo presenta el desarrollo de una aplicación web, denominada “A-e-i!”, que utiliza funciones de servicios alojados en servidores en la nube para transcribir audios de lecturas de pacientes con deterioro neurodegenerativo derivado en afasia progresiva primaria. Con las transcripciones generadas, se pretende monitorear los cambios en el discurso a través del tiempo de dichos pacientes. Para realizar este monitoreo, se diseñaron métricas que permiten medir distintos aspectos en relación a la fluidez de la lectura, y que, además, sirven para una posterior evaluación, a partir de indicadores que interpretarán los valores medidos en el tiempo. Desde el punto de vista tecnológico, se combinaron algoritmos que transcriben discurso oral a texto y de reconocimiento de partes de un discurso.

Índice general

Índice de figuras	viii
Índice de tablas	x
Índice de códigos	x
1. Introducción	1
1.1. Problema	1
1.2. Solución	5
1.3. Objetivos	6
1.4. Estructura de la tesina	7
2. Fundamentos Teóricos	8
2.1. Reconocimiento automático de discurso	8
2.2. Procesamiento de lenguaje natural	10
2.2.1. Orígenes y evolución del procesamiento del lenguaje natural	11
2.2.2. Reconocimiento de partes de un discurso	15
2.3. Computación en la nube	17
2.3.1. Introducción	17
2.3.2. Proceso de Negocio como un servicio	18
2.3.3. Infraestructura como Servicio	20
2.3.4. Computación <i>serverless</i>	21

2.4. Servicios Web	21
2.4.1. Arquitectura REST	22
2.5. Enfermedades Neurodegenerativas	23
2.5.1. Demencia Frontotemporal	25
2.5.2. Afasia Progresiva Primaria	28
3. Estado del arte	33
3.1. Tecnologías de reconocimiento de discurso	33
3.1.1. Selección de la tecnología de reconocimiento de discurso	35
3.1.2. Justificación de la selección	36
3.2. Reconocimiento de partes de un discurso	37
3.2.1. Selección de herramientas para el reconocimiento de partes de un discurso	38
3.2.2. Justificación de la selección	40
3.3. Computación <i>serverless</i>	41
3.3.1. Selección de herramientas <i>serverless</i>	42
3.3.2. Justificación de la selección	42
3.4. Base de datos	43
3.4.1. Selección de herramientas para base de datos	44
3.4.2. Justificación de la selección	45
3.5. Trabajos Relacionados	45
4. Desarrollo de A-e-i!	51
4.1. Metodología de desarrollo	51
4.2. Especificación y análisis del proyecto	54
4.2.1. Propósito del sistema	54
4.2.2. Especificación de requisitos del sistema	54
4.2.3. Modelo de casos de uso	56

4.3. Diseño de A-e-i!	57
4.3.1. Diseño de la arquitectura	57
4.3.2. Diseño de la base de datos	59
4.3.3. Diseño de la interfaz	60
4.3.4. Diseño de métricas	61
4.4. Implementación de A-e-i!	64
4.4.1. Implementación de la lógica de negocio	64
4.4.2. Implementación de la interfaz de usuario	78
4.5. Despliegue de A-e-i!	81
4.5.1. Despliegue del servidor del <i>Backend</i>	81
4.5.2. Despliegue del servidor del <i>Frontend</i>	85
5. Caso de estudio	89
5.1. Introducción	89
5.2. Realización de pruebas	90
5.3. Generación de resultados	92
6. Conclusiones y Trabajos Futuros	96
6.1. Conclusiones finales	96
6.2. Trabajos futuros	98
Referencias	99
Anexo A. Wireframes	105
Anexo B. Tablas	108
Anexo C. Código	109

Índice de figuras

1.1. Esquema de la solución desarrollada.	6
2.1. Red semántica del Método Científico. Extraído de [Wikipedia, 2023]. . .	14
2.2. Modelos de servicios de computación en la nube. Extraído de [TIC, 2017].	19
2.3. Área afectada por la Afasia progresiva primaria semántica (APP-vs). Extraído de [Matías, 2020].	29
2.4. Área afectada por la Afasia progresiva primaria no fluente (APP-vnf). Extraído de [Matías, 2020].	30
2.5. Área afectada por la Afasia progresiva primaria logopénica (APP-vl). Extraído de [Matías, 2020].	32
3.1. Diagrama de <i>pipeline</i> de procesamiento de <i>SpaCy</i> . Extraído de [spaCy, 2023].	40
4.1. Fases para un proceso de investigación cualitativo. Extraído de [Hernández-Sampieri y Mendoza, 2020].	52
4.2. Diagrama de Gantt planificado para el presente trabajo final.	53
4.3. Diagrama de casos de uso.	56
4.4. Arquitectura de A-e-i!. Vista de componentes y conector.	58
4.5. Esquema tecnológico de la arquitectura de A-e-i!.	59
4.6. Diagrama de entidad relación de A-e-i!.	59
4.7. Pantalla de inicio de sesión.	60
4.8. Pantalla de resultados.	61

4.9. Petición realizada para registrar un paciente.	69
4.10. Ejemplo de etiquetado POS para la oración 'Es un día soleado'	76
4.11. Configuración de la base de datos.	77
4.12. Dirección de IP pública de la base de datos.	77
4.13. Interfaz gráfica para gestión de pacientes.	80
4.14. Interfaz gráfica para el registro de pacientes.	81
4.15. Creación del servicio de <i>Cloud Run</i> para el servicio web de Flask.	84
4.16. Opciones de configuración del servicio de Cloud Run.	84
4.17. Configuración del servicio de Cloud Run para la aplicación de React.	88
5.1. Lectura: "La hormiguita viajera".	91
5.2. Ejemplo de medidas obtenidas para un audio de una paciente. Extraído de la aplicación.	93
5.3. Valor de la métrica de velocidad del discurso en las grabaciones realizadas desde el 14 de Diciembre de 2022 al 22 de Febrero de 2023.	94
5.4. Valores medidos para cada categoría de palabra (correctas, eliminadas, sustituidas, insertadas) para la grabación del día 6 de Febrero del 2023.	94
5.5. Valores medidos para la categoría de palabra "correctas" para la grabación del día 6 de Febrero del 2023.	95
A.1. Pantalla de inicio de sesión.	105
A.2. Pantalla de bienvenida.	106
A.3. Pantalla de los audios junto a sus resultados.	106
A.4. Pantalla de los resultados obtenidos por paciente a través del tiempo.	107

Índice de tablas

2.1. Etiquetas POS definidas en <i>Treebank</i>	17
2.2. Clasificación de las demencias de tipo frontotemporal.	26
3.1. Resultados del cálculo de WER para distintos servicios de transcripción.	37
4.1. Etiquetas de tipo <i>Parts of Speech</i>	74
4.2. Ventajas del Framework Next.js sobre el uso de HTML, CSS y JavaScript.	79
B.1. Relaciones de dependencia posibles en un discurso.	108

Índice de códigos

4.1. Creación de una aplicación en Flask	65
4.2. Registro de rutas de vista de autenticación.	66
4.3. Registro de rutas.	67
4.4. Configuración de JWTManager.	67
4.5. Ejemplo de uso de <code>create_access_token()</code>	68
4.6. Creación de la instancia de configuración de voz.	69
4.7. Configuración de audio.	70
4.8. Algoritmo de <i>Speech Recognition</i>	70
4.9. Algoritmo de transformación de formato de audio.	71
4.10. Algoritmo de reducción de ruido.	71
4.11. Algoritmo de obtención de palabras más comunes en un texto.	72
4.12. Algoritmo para agregar una lista de palabras al reconocedor de discurso.	73
4.13. Carga de modelo de idioma.	74
4.14. Código del algoritmo de <i>Part-Of-Speech Tagging</i>	75
4.15. Raíz del proyecto de Next.js	79
4.16. Creación de archivo Dockerfile para el despliegue del servidor <i>backend</i>	82
4.17. Creación de archivo Dockerfile: Instalación de paquetes y ejecución	82
4.18. Carga de imagen en Google Cloud Registry	83
4.19. Configuración de Axios	85
4.20. Archivo <code>.dockerignore</code>	86
4.21. Configuración de Dockerfile para despliegue de <i>frontend</i> : Entorno de compilación	86
4.22. Configuración de Dockerfile para despliegue de <i>frontend</i> : Entorno de producción	87
4.23. Carga de imagen en Google Cloud Registry	87
C.1. Contenido del archivo <code>init.sql</code>	109
C.2. Algoritmo de WER basado en la distancia de Levenshtein	111
C.3. Algoritmo de prueba de servicio de transcripción de Facebook	113

C.4. Algoritmo de prueba de servicio de transcripción Google Speech Recognition	114
C.5. Algoritmo de prueba de servicio de transcripción de Microsoft Azure Cognitive Services	114
C.6. Algoritmo de prueba de servicio de transcripción IBM Watson	115

Capítulo 1

Introducción

Este trabajo de tesis es presentado para cumplimentar lo requerido en la Resolución N.º 069/21 y aprobar la carrera Ingeniería en Sistemas (plan 61217), perteneciente a la Facultad de Ingeniería, de la Universidad Nacional de La Pampa.

Su realización, es parte de una de las líneas de avance del proyecto de investigación denominado “Gestión del Conocimiento en Sistemas Domóticos”, acreditado por Resolución N.º 165/18 del Consejo Directivo.

En este primer capítulo, se introduce el problema detectado, la solución propuesta que motiva la realización del trabajo, así como también los objetivos establecidos a cumplir y, finalmente, la estructuración en capítulos de este trabajo.

1.1. Problema

Las enfermedades neurodegenerativas son actualmente la causa más frecuente de demencia. La demencia incluye un conjunto de síntomas que afectan la memoria, el pensamiento, la ejecución y las habilidades sociales, y son lo suficientemente graves como para interferir en la vida diaria de la persona afectada. Alrededor del 10-20 % de ellas son consideradas demencias frontotemporales, conocidas anteriormente como “enfermedad de Pick” [Henriksson y Laakso, 2020].

La demencia frontotemporal (DFT) es una enfermedad neurodegenerativa que afecta especialmente a los lóbulos frontales y temporales del cerebro. Estas, son las áreas que se asocian con la personalidad, la conducta y el lenguaje. En la demencia

frontotemporal, partes de estos lóbulos se encogen (se atrofian) y ciertas sustancias se acumulan en el cerebro. Generalmente, se desconoce la causa de estos cambios, aunque podría atribuirse a mutaciones genéticas. Los signos y síntomas varían según qué parte del cerebro esté afectada. Algunas personas con demencia frontotemporal desarrollan evidentes cambios en su personalidad y se vuelven socialmente inapropiadas, impulsivas o emocionalmente indiferentes, mientras que otras pierden la capacidad de usar el lenguaje adecuadamente.

La DFT es, por lo tanto, una causa frecuente de demencia. El diagnóstico clínico temprano a menudo es difícil y, dada la baja capacidad de introspección de los pacientes, la anamnesis a familiares es fundamental. Hasta la fecha, predecir la enfermedad subyacente sigue siendo difícil y, muchas veces, asociada con otras enfermedades neurodegenerativas que afectan otras áreas cerebrales, como lo es el Alzheimer. La definición de subtipos clínicos y la identificación de biomarcadores podrían ayudar a predecir la afección, y que, junto con el desarrollo de fármacos, se puedan ofrecer nuevas posibilidades terapéuticas [Cherney, 1995].

De acuerdo a los síntomas predominantes de la DFT, se distinguen tres síndromes clínicos principales: la DFT variante conductual, la demencia semántica y la afasia primaria progresiva no fluente [Cherney, 1995]. En las últimas dos, se sufre el deterioro o pérdida del habla, ocasionando los siguientes problemas [Buckingham Jr y Kertesz, 1974] :

- Dificultad creciente para usar y entender el lenguaje escrito y hablado.
- Dificultad para nombrar cosas, posiblemente reemplazando una palabra específica con una palabra más general.
- Desconocimiento del significado de las palabras.
- Tener un habla vacilante que puede sonar telegráfica.
- Cometer errores en la construcción de frases.

Afasia Progresiva Primaria (APP) es el término que se utiliza para la dificultad del lenguaje que se desarrolla progresivamente y que, por lo tanto, afecta la manera en que nos comunicamos. Puede afectar el habla, la escritura y la comprensión del lenguaje escrito y oral.

Los criterios de diagnósticos internacionales actuales distinguen tres variantes diferentes de la afasia primaria progresiva: la no fluida, la semántica y la logopénica, cada una de las cuales tiende a exhibir patrones específicos de déficits lingüísticos y una distribución característica de atrofia cerebral [Gorno-Tempini et al., 2011]. Las dos primeras se incluyen en el grupo de demencias frontotemporales.

Las personas que padecen la variante no fluída se caracterizan por un habla esforzada, presentando déficits morfosintácticos y omisión de palabras funcionales que conducen a agramatismo¹ y simplificación de la producción del lenguaje. Esto resulta en pérdida de la prosodia y errores articulatorios.

Los pacientes con la variante semántica muestran anomia severa y constantes dificultades en la comprensión de palabras sueltas. En tanto que, la persona que padece la variante logopénica, presenta dificultades para encontrar palabras, en la repetición de oraciones, ausencia de agramatismo, apraxia del habla y deterioro de la memoria semántica.

La afasia, en cualquiera de sus formas, ocasiona que la fluidez del lenguaje se vaya perdiendo de manera progresiva y selectiva. La persona comprende, pero no es capaz de articular las palabras que desde su pensamiento se generan. La capacidad de pronunciar fluidamente los sonidos del habla, recuperar la palabra adecuada o de expresar frases completas, es forzada y se va perdiendo lenta y progresivamente. Los seres humanos somos seres lingüísticos y, atravesar por deterioros que dañen a las neuronas que procesan el lenguaje, conduce a un sentimiento de frustración y distanciamiento social.

La detección y caracterización de las alteraciones del lenguaje juega un papel cada vez más importante en la identificación y diagnóstico de muchas enfermedades neurodegenerativas, como lo son, las demencias frontotemporales. Estos déficits del lenguaje se manifiestan por lo general, en etapas tempranas, sin mayores indicadores más que alguna latencia en el habla. Un análisis de la evolución del deterioro del lenguaje puede ser considerado como parte integral de la evaluación cognitiva de los pacientes afectados por alguna variedad de esta enfermedad neurodegenerativa [Boschi et al., 2017].

Actualmente, la mayoría de las evaluaciones cognitivas estándares que se realizan a estos pacientes² permiten conocer ciertos indicadores en relación a distintas dimensiones, como la independencia, la expresividad, las emociones, la identificación de imágenes, y secuencias de ejecución de distintas tareas, por mencionar solo algunos de los enfoques que tienen los diagnósticos. Pero, no se refieren específicamente a evaluar y monitorear el deterioro del habla en función de patrones léxicos de acuerdo a la

¹Trastorno neurológico caracterizado por la manifestación de deficiencias morfológicas y gramaticales en la construcción de frases. La morfología es la parte de la lingüística que estudia las reglas que rigen la flexión, la composición y la derivación de las palabras.

²Entre las más habituales se destacan: *Boston Diagnostic Aphasia Examination (BDAE)*, *Western Aphasia Battery*.

patología detectada. Y si bien, tanto la lectura en voz alta como el diálogo continuo, son indicados como “el mejor remedio”, actualmente los profesionales no cuentan con análisis cuantitativos permanentes de estas habilidades a fines de contrastar muchos casos y abordar conclusiones que les permitan mejorar las terapias ofrecidas a estos pacientes. Aún más, usualmente, no es objetivo de las terapias mejorar la lectura oral [Cherney, 1995]. Poco se analiza en pacientes con APP y DFT el discurso sintáctico, morfosintáctico, léxico semántico, así como la fonética, por mencionar algunos aspectos lingüísticos que caracterizan el lenguaje. El tratamiento de esta enfermedad neurodegenerativa sigue siendo sintomático o de soporte [Cherney, 1995], pero con escasas herramientas para monitorear el deterioro del habla.

Sin dudas que los actuales avances y los futuros, son fundamentales para ampliar la diversidad de acciones terapéuticas. Pero, surgen los siguientes interrogantes:

- ¿Qué hay respecto al monitoreo y control de estos pacientes que progresivamente comienzan a tener problemas con el lenguaje oral (expresión y comprensión) y con el lenguaje escrito (lectura y escritura)?
- ¿Cómo se mide el deterioro del habla?
- ¿Por qué “la lectura en voz alta” sigue siendo la terapia más prescrita por los especialistas (llámese neurólogos, fonoaudiólogos, terapistas)? Pero que, no tiene ningún tipo de evaluación ni seguimiento, más que la propia voluntad del familiar que se cerciore de que la persona afectada tenga como rutina dedicar un espacio a la lectura en voz alta, por mencionar un ejemplo de uno de los tratamientos indicados para mejorar la latencia del discurso.

De acuerdo a *American Speech-Language-Hearing Association*³, por lo general en esas patologías existen mayores dificultades con la lectura y escritura que con la expresión y comprensión orales.

Por lo expuesto, se evidencia la necesidad de contar con herramientas que permitan un monitoreo y posterior análisis del lenguaje oral, principalmente, el producido con la lectura. A la fecha, los especialistas cuentan con herramientas que pueden obtener datos del paciente (tal como imágenes, evaluaciones neuropsicológicas, entrevistas con el paciente y sus familiares) desde el propio centro médico y, a partir de ellos, determinan el tratamiento más adecuado a aplicar. Pero, podrían beneficiarse si contaran con datos del paciente obtenidos desde su propio hogar, provenientes de su rutina diaria o de

³<https://www.asha.org/>

tareas asignadas. En este sentido, un importante flujo de datos que se puede recopilar de forma discreta y económica es el que se obtiene del audio de voz [Peintner et al., 2008].

Así, proveer a médicos y especialistas herramientas que midan y controlen el deterioro del habla desde la propia comodidad de la casa del paciente, constituye una vía de control para el mismo y, también, un acervo de datos y estadísticas que pueda servir a futuro para mejorar las técnicas y diagnósticos aplicados.

1.2. Solución

En la sección anterior quedó expuesto que, en algunos trastornos neurodegenerativos, las personas pueden presentar una disminución de habilidades lingüísticas que son difíciles de cuantificar con pruebas estandarizadas [Fraser et al., 2014]. El análisis del discurso o “habla conectada” en estos pacientes puede proporcionar información valiosa sobre las capacidades lingüísticas de los mismos.

De acuerdo a [Peintner et al., 2008], una fuente de datos oportunos y representativos es la que se obtiene directamente del paciente en su propio ámbito, su propio hogar, y que puede sumar al flujo de datos obtenidos de análisis clínicos imágenes y pruebas psicológicas específicas. Además, dentro de las pruebas y tratamiento, se incluye a la lectura en voz alta pero, rara vez, es aplicada sistemáticamente como parte integral de la terapia y poco se sabe (aunque es altamente prescrita) respecto a su eficacia [Cherney, 1995].

Estas cualidades, junto con la creciente evidencia de que enfermedades neurodegenerativas, como la APP, alteran la producción del habla y el lenguaje, y los escasos avances en estudios de estas enfermedades y el idioma español, motivan la realización del presente trabajo. El problema a resolver radica en procesar el lenguaje natural de pacientes con DFT afásicos, a partir de transcripciones de audio correspondientes a lecturas en voz alta, utilizando patrones de reconocimiento léxico.

Por lo tanto, la presente tesis presenta el desarrollo de una aplicación web con arquitectura distribuida, denominada “A-e-i!”, que utiliza funciones de servicios alojados en servidores en la nube para transcribir audios de lecturas de pacientes con deterioro neurodegenerativo derivado en APP. Con las transcripciones generadas, se pretende monitorear los cambios en el discurso a través del tiempo de dichos pacientes. Para realizar este monitoreo, se diseñaron métricas que permiten medir distintos aspectos en

relación a la fluidez de la lectura y que, además, sirven para una posterior evaluación, a partir de indicadores que interpretarán los valores medidos en el tiempo. Desde el punto de vista tecnológico, se combinaron algoritmos de *machine learning* como los relacionados a *speech-to-text* (que transcriben discurso oral a texto) y de reconocimiento de partes de un discurso dentro de un texto (que permiten comparar y detectar omisiones, similitudes y nuevas ocurrencias de palabras comparando con un texto original). Además, la herramienta se implementó bajo una arquitectura REST distribuida que consume servicios de reconocimiento del discurso, alojados en plataformas en la nube y se propone un despliegue de la misma usando servicios de infraestructura en la nube. La Figura 1.1 ilustra un esquema de la solución propuesta para el problema planteado.

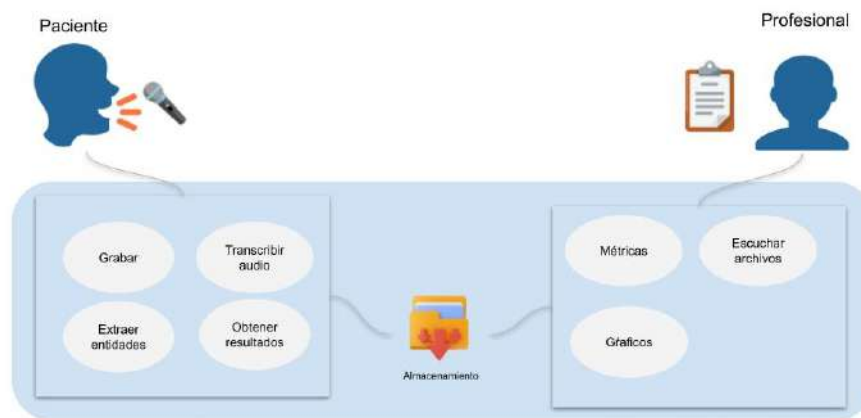


Figura 1.1 Esquema de la solución desarrollada.

El propósito final de la herramienta “A-e-i!”, es asistir a los profesionales fonoaudiólogos que atienden pacientes con deterioro neurodegenerativo, en esta instancia, enfocados en aquellos con DFT con consecuencias de APP. Así, el principal beneficio, desde la perspectiva de los profesionales de la salud, es el de facilitarles el monitoreo del tratamiento de estos pacientes en relación al deterioro del discurso hablado y poder arribar a conclusiones más precisas respecto a esta afectación. Se presenta un caso de estudio con una paciente real, que sirvió como referencia y principal motivación, para definir el problema detectado y delimitar la solución hallada.

1.3. Objetivos

El presente trabajo de tesis contempla como objetivo general el diseño e implementación de una aplicación con arquitectura distribuida de microservicios para

dar soporte al procesamiento y análisis del lenguaje natural español de pacientes afásicos, utilizando tecnologías de computación en la nube para consumir servicios y desplegar la aplicación.

Como parte de los objetivos específicos, se formularon los siguientes:

- Diseñar e implementar un microservicio de reconocimiento de voz que permita transcribir las muestras de lecturas realizadas.
- Diseñar e implementar un microservicio de clasificación de entidades de un texto para estructurar la información de una forma que facilite el análisis sintáctico y semántico.
- Diseñar e implementar una aplicación prototipo con conexión a dichos microservicios para asistir a profesionales neurólogos, fonoaudiólogos y/o logopedas, y contribuir con el estudio y análisis de la evolución del deterioro del habla de sus pacientes afásicos.

1.4. Estructura de la tesina

Este trabajo se estructura en seis capítulos:

- El primer capítulo presenta la introducción a esta tesis y describe: 1) la problemática a abordar, 2) la solución actual que se plantea para dicha problemática, 3) los objetivos que se estipularon alcanzar y 4) la estructura del trabajo.
- El segundo capítulo introduce al lector hacia los fundamentos teóricos esenciales para la comprensión del trabajo y que constituyeron el estudio que se tuvo que abordar para realizarlo.
- El tercer capítulo describe el estado del arte y proporciona una breve comparación de las diferentes herramientas y servicios a utilizar en el desarrollo de la propuesta.
- El cuarto capítulo detalla el diseño e implementación de la herramienta propuesta, haciendo mención al desarrollo tecnológico realizado.
- El quinto capítulo presenta el caso de estudio realizado.
- Por último, el capítulo seis expone las conclusiones derivadas de la realización de este trabajo y se detallan las propuestas para trabajos futuros.

Capítulo 2

Fundamentos Teóricos

El presente capítulo introduce el marco teórico necesario para el desarrollo de esta tesis, a saber: reconocimiento del discurso, procesamiento del lenguaje natural, computación en la nube, servicios web, servidores y finalmente, una introducción al tema de enfermedades neurodegenerativas.

2.1. Reconocimiento automático de discurso

El reconocimiento de voz automático es una técnica que necesita de una máquina para poder decodificar el contenido de lo hablado por una persona. Un sistema de reconocimiento de voz típico tiene como entrada un archivo en formato de audio o bien lo captado a través de un micrófono y, luego, mediante análisis específicos, se produce una salida. La salida generada es una transcripción del discurso hablado, reconociendo las palabras que lo componen.

El primer sistema de reconocimiento de voz fue desarrollado por investigadores de laboratorios Bell en el año 1952, diseñado para reconocer dígitos dictados por un individuo automáticamente. Se basaba en la división del espectro del discurso en dos bandas de frecuencias, una sobre los 900 cps (clicks por segundo) y otra, por debajo de ese valor. Luego, se determinaba un estándar para reconocer cada uno de los 10 dígitos y se guardaban en una memoria elemental para, posteriormente, ser comparados con las entradas.

En 1962 IBM presentó su primer sistema de reconocimiento de voz *IBM Shoebox*, que era capaz de reconocer 16 palabras relacionadas con operaciones matemáticas y 10

dígitos. Por ejemplo, identificaba comandos como “Cinco más tres más ocho más seis más cuatro menos nueve, total” e imprimía la respuesta correcta, es decir, 17.

Durante la década de 1970, la Agencia de Proyectos de Investigación Avanzada de Defensa de los Estados Unidos (DARPA) contribuyó mucho a la tecnología de reconocimiento de voz. DARPA financió un programa llamado *Speech Understanding Research* que tuvo como resultado el desarrollo de *Harpy* [Lowerre y Reddy, 1976], el primer sistema de reconocimiento de voz capaz de reconocer frases.

En la década de 1980, se aplicó el concepto de Modelo Oculto de Markov o *Hidden Markov Model* (HMM) al sistema de reconocimiento de voz. HMM es un modelo estadístico denominado así en honor al matemático que desarrolló gran parte de la teoría estadística que lo compone, Andrey Andreyevich Markov. Se utiliza para modelar los problemas que involucran información secuencial debido a que permite capturar información oculta de símbolos secuenciales observables. En un HMM se modela un sistema como un proceso de Markov, con parámetros desconocidos, y en base a esto se deben determinar los parámetros ocultos a partir de los parámetros observables. Esta técnica se ajusta a varios problemas de clasificación, el problema del reconocimiento de voz es uno de ellos.

En 2001, Google introdujo la aplicación *Voice Search* que permitía a los usuarios buscar consultas de manera hablada. Esta fue la primera aplicación habilitada para voz que fue muy popular.

En 2011, Apple lanzó *Siri*, que ofrecía una forma más rápida, sencilla y en tiempo real, de interactuar con los dispositivos de Apple a través de órdenes. Luego surgieron *Alexa* de Amazon y *Google Home*, asistentes virtuales basados en comandos de voz que permiten controlar una amplia gama de productos.

Hasta el día de hoy, se siguen dedicando muchos esfuerzos a los sistemas de reconocimiento de voz porque es una temática en la que aún quedan cosas a mejorar para aumentar su precisión. En estos sistemas, la precisión se ve afectada por distintas variables: el ruido del entorno, la forma de hablar de cada persona (que depende de su capacidad cognitiva y de cuestiones socioculturales), el idioma y la extensión del vocabulario. Este campo es multidisciplinario ya que incluye, no solamente a las ciencias de la computación, sino también a la lingüística, la acústica, la teoría de comunicación, estadística, fonoaudiología y psicología. A pesar de la dificultad que supone desarrollar un producto que supere los desafíos antes mencionados, actualmente existen diversas soluciones de software para reconocimiento de voz y transcripción, tales como *Dragon*

NaturallySpeaking, Google Cloud Speech Recognition, Embedded ViaVoice, LumenVox, Microsoft Azure Cognitive Services e IBM Watson, entre otros.

2.2. Procesamiento de lenguaje natural

El procesamiento del lenguaje natural (conocido como NLP por sus siglas en inglés provenientes de *Natural Language Processing*) es un campo de las ciencias informáticas que data de los años ‘50 y se lo reconoce dentro de la rama de la Inteligencia Artificial (IA). Esta disciplina enfoca sus esfuerzos de investigación y desarrollo en investigar la manera de comunicar las máquinas con las personas mediante el uso de lenguajes naturales, como el español, inglés, chino o cualquier otro. Comprende la traducción automática, el reconocimiento de entidades nombradas, el reconocimiento de partes de un discurso, el procesamiento de voz y la recuperación y extracción de información del lenguaje natural [Nadkarni et al., 2011].

Al procesamiento del lenguaje natural se lo suele denominar como un problema “NP-completo” [Ali y Shapiro, 1993], porque requiere un amplio conocimiento sobre el mundo exterior y la capacidad de interacción entre los agentes que lo componen. Además, los lenguajes son ambiguos por naturaleza, lo que complejiza aún más su análisis. Según Andrew NG¹, reconocido líder en el campo de la Inteligencia Artificial a nivel mundial, el reconocimiento del discurso combinado con procesamiento de lenguaje natural se convertirá en la principal forma en la que los humanos interactuaremos con las computadoras debido a que la precisión de las predicciones hoy en día se encuentra entre el 95 % y el 99 % [Merrett, 2015].

En la sección anterior se introdujo el concepto de reconocimiento automático de discurso, a partir del cual es posible generar transcripciones de audios. El procesamiento del lenguaje natural entra en escena una vez que se haya generado este reconocimiento de discurso. Es decir, tan pronto se haya convertido el formato de audio en formato de texto. De esta manera, es posible conocer la gramática y el significado del discurso y analizarlo en función de las estructuras gramaticales del idioma en cuestión. Así, NLP se enfoca en la intersección entre las ciencias de la computación, la inteligencia artificial y la lingüística computacional, permitiendo que se pueda estructurar el lenguaje para

¹El Dr. Andrew Ng es un líder mundialmente reconocido en IA. Es fundador de *DeepLearning.AI*, fundador y director ejecutivo de *Landing AI*, socio general de *AI Fund*, presidente y cofundador de *Coursera* y profesor adjunto en el Departamento de Ciencias de la Computación de la Universidad de Stanford. Más información sobre su trayectoria disponible en <https://www.andrewng.org/>.

analizarlo. Entre los análisis que se pueden aplicar, a la fecha se pueden mencionar los siguientes [Liddy, 2001]:

- Traducción automática: Es el procedimiento que tiene como entrada un texto escrito en un lenguaje y produce como salida un texto con el mismo sentido escrito en otro lenguaje.
- Creación de resúmenes automática: Permite obtener una versión resumida de un texto extrayendo las ideas principales sin perder de vista el contexto.
- Análisis de sentimiento: Es un tipo de procesamiento que se aplica sobre un texto para saber el sentimiento que exhibe o se cree que exhibe. Este análisis se usa mucho en sitios que poseen la opción de publicar una opinión o reseña.
- Clasificación de texto: Permite subdividir un texto y otorgarle una categoría a cada subdivisión. Para lograr la categorización de las palabras dentro de un discurso se usan algoritmos basados en estadística, como por ejemplo la estimación paramétrica, la estimación no paramétrica, el análisis de la distribución normal, de la distribución binomial y de la distribución multinomial [Collins, 2002]. La clasificación de texto es el tipo de análisis que se utiliza en el presente trabajo, etiquetando a los componentes del texto según sus categorías gramaticales.
- Respuesta automática de preguntas: Tal como se puede deducir de su nombre, intenta devolver una salida coherente a una pregunta o requisito de entrada expresada en lenguaje natural. Ejemplos de sistemas que funcionan así son *Siri*, *Alexa* y *OK Google*.

2.2.1. Orígenes y evolución del procesamiento del lenguaje natural

Los avances en el campo de NLP se pueden dividir en cuatro fases [Jones, 1994]. La cuarta fase tiene que ver con trabajos que se están realizando en el presente.

En la primera fase, el trabajo estaba enfocado en crear sistemas de traducción no humanos. El primer trabajo expuesto fue un traductor muy simple del idioma ruso al idioma inglés en 1954 en el diario MT (*Mechanical Translation*). El principal logro fue poder obtener una caracterización del lenguaje completa y precisa y aplicar estas descripciones, a través de un algoritmo. De todos los esfuerzos y estudios realizados en la época, uno de los métodos más importantes que surgió fue el método basado en las “reglas de producción de Chomsky” [Chomsky, 1956]. Las reglas de producción deben

estar constituidas por palabras que actúen como antecedentes y palabras que actúen como consecuentes para que, luego de su aplicación, una palabra se transforme en otra.

El apogeo de esta primera fase fue en 1961 durante el Congreso Internacional de Traducción Automática y Análisis de Idiomas, donde se reportaron trabajos provenientes de investigadores de varios países que tenían como objeto de estudio, tanto cuestiones de lingüística, como la morfología, la sintaxis y la semántica, como cuestiones más técnicas, como la interpretación y la generación de vistas desde el enfoque de hardware. Es esencial recordar que, para esa época, el hardware no tenía las capacidades que tiene hoy en día, se utilizaban tarjetas perforadas y el procesamiento era realizado por lotes. El almacenamiento era muy limitado y la velocidad de las máquinas era extremadamente baja. Es por esto que las primeras investigaciones no se enfocaron tanto en la programación, sino en las capacidades del hardware. A pesar de las limitaciones en cuanto a recursos, hubo avances desde el lado de la programación, como por ejemplo el desarrollo de una herramienta simple que permitía el análisis de oraciones. La mayoría de los esfuerzos tenían como objetivo resolver el análisis de la sintaxis.

En resumen, la primera fase tuvo como resultado la definición concreta de los requerimientos del procesamiento de lenguaje computacional y la investigación de uno de esos requerimientos, el análisis sintáctico, en base a las tecnologías disponibles.

La segunda fase del trabajo en NLP estuvo marcada por el uso de IA y la orientación hacia el estudio de la semántica de los lenguajes. En el año 1961, se presenta el sistema de preguntas y respuestas BASEBALL [Green Jr et al., 1961], en el cual las entradas posibles estaban restringidas y el procesamiento de lenguaje involucrado era muy simple, pero tenía la capacidad de realizar inferencias usando la base de conocimiento y dar respuesta a ciertas entradas en forma de pregunta. El sistema contenía información sobre la liga de baseball de Estados Unidos. Luego surgieron otros sistemas de preguntas y respuestas como ELIZA y LUNAR, que tenían el mismo comportamiento, con la diferencia de que las bases de datos de conocimiento escritas estaban orientadas a otros dominios. ELIZA [Weizenbaum, 1966] es considerado el primer *bot* de la historia y su objetivo era imitar el comportamiento del psicólogo Carl Rogers, intentando mantener una conversación de texto coherente con el usuario.

En 1975, respondiendo a la necesidad de capturar las relaciones e identidades conceptuales dentro de un discurso, se formuló la Teoría de la Dependencia Conceptual de Shank [Schank y Tesler, 1969]. La misma proveía un modelo de comprensión del lenguaje natural para representar el conocimiento con el objetivo de enviarle órdenes a la computadora escritas en lenguaje natural y que no tengan más de una interpretación.

Estos avances permitieron empezar a pensar en desarrollar arquitecturas modulares e interfaces y permitir la interacción de usuarios con estos sistemas. En el último período de esta fase surge la necesidad de identificar claramente los requerimientos de los usuarios.

En la segunda mitad de la década de 1970, los esfuerzos de investigación estuvieron direccionados en estudiar la funcionalidad del lenguaje y la necesidad de incorporar la comprensión del discurso en textos simples, tales como cuentos o noticias, generando así nuevos modelos de discurso.

Los modelos disponibles en esta etapa son los de concordancia de patrones semánticos y la lógica de primer orden. Los modelos de concordancia de patrones semánticos fueron diseñados por Ceccato en 1967 [Ceccato, 1967]. Usaban conocimiento del mundo dividido en categorías y *frames semánticos* (tipo de representación que define conceptos, sus atributos y sus interrelaciones) para interpretar y representar componentes lingüísticos. Por su parte, la lógica de primer orden consta de axiomas y reglas de inferencias que representan el mundo en forma de predicados, formando así un sistema capaz de hacer deducciones dada una entrada. Por estar basada en la lógica de predicados, permite eliminar la ambigüedad evitando así el exceso de posibles interpretaciones. El inconveniente que presenta la lógica de primer orden es que no es tan escalable, debido a que si el sistema crece debe ser modificada en su totalidad.

La tercer fase data del final de la década de 1970 hasta el final de la década de 1980. Esta fase tuvo un enfoque lógico-gramatical, motivado por el desarrollo de la teoría de la gramática en los años '70 y la aparición de la lógica para representar conocimiento. Los sistemas empezaron a gozar de una mayor capacidad de comprensión de textos. Para esto, se hizo uso del enfoque declarativo y se aprovechó el crecimiento de la programación lógica. Por el lado de la investigación, el período estuvo marcado por un crecimiento del interés en la estructura del discurso y por la publicación de los primeros trabajos basados en generación de texto.

Como final de esta tercer fase se destaca la utilización de diccionarios digitales. Con esto se generó mucha cantidad de material en formato digital, lo que facilitó la tarea de empezar a validar o mejorar textos, a desarrollar sistemas de diálogo y también sistemas de generación de explicaciones.

Con el surgimiento de la web, se empezaron a generar documentos que incentivaron el desarrollo de sistemas de extracción y resumen de texto. Por otro lado, empezó a crecer el campo de reconocimiento del habla.

Los modelos que protagonizaron esta fase no fueron solamente lingüísticos, como hasta ese momento, sino que surgieron modelos estadísticos que no solo se centraban en el análisis del discurso. Un ejemplo de modelo lingüístico clave para esta etapa lo constituyen las redes semánticas, y por el lado de los modelos estadísticos se destaca el HMM, introducido anteriormente.

Una red semántica es un tipo de representación del conocimiento formado por un conjunto de nodos y arcos interconectados. Los nodos representan conceptos y los arcos representan relaciones entre esos conceptos. Este tipo de representaciones pueden presentar definiciones o proposiciones [Cambria y White, 2014]. Se usan mucho en el área de la inteligencia artificial.

La Figura 2.1 ilustra una red semántica creada para describir el famoso Método Científico.



Figura 2.1 Red semántica del Método Científico. Extraído de [Wikipedia, 2023].

Por último, la cuarta fase se la identifica a principios de los años '90 y sigue vigente hasta el día de hoy. Los últimos años están marcados por el análisis de cantidades masivas de datos, los cuáles pueden ser analizados gracias al crecimiento exponencial de la tecnología. Los avances tecnológicos también permiten que se puedan procesar desde textos básicos hasta audio proveniente de archivos de vídeo. La cantidad de datos que se encuentra disponible hoy en día permite el entrenamiento de modelos que resultan ser muy precisos. De hecho, se pueden encontrar y obtener varios de estos modelos de manera gratuita a través de internet. En general, los modelos más

avanzados corresponden al idioma inglés, pero se pueden encontrar modelos, de menor precisión quizás, en todos los idiomas. Los sistemas de procesamiento de lenguaje natural actuales realizan actividades mucho más complejas que en las primeras fases. Sin embargo, aún se sigue trabajando para solucionar aspectos más complejos como los que tienen que ver con la comprensión de metáforas o la generación de respuestas razonables.

2.2.2. Reconocimiento de partes de un discurso

El reconocimiento de partes de un discurso, tal como su nombre lo indica, se utiliza para clasificar las palabras según el conjunto al que pertenecen. Por ejemplo, dentro de una oración se podrían identificar entidades de tipo “nombre propio”, “verbo”, “adjetivo”, entre otras.

Estos modelos funcionan en dos fases: En la primera fase se divide el texto en segmentos para clasificar cada uno. En la segunda fase, se clasifica cada segmento en categorías en forma de *tokens* [Costumero et al., 2014]. Todo este proceso se lleva a cabo ignorando cuestiones de formato, como por ejemplo, la presencia de mayúsculas.

Esta tecnología es de gran utilidad a la hora de analizar el contenido de los archivos de audio transcritos que comprende el presente trabajo y, así, obtener los valores de distintos resultados, que serán interpretados por un profesional.

Respecto al discurso de una persona, hay cuatro aspectos que pueden considerarse para analizar la forma de expresarse de las mismas:

- Métodos gramaticales: Analizar las estructuras morfosintácticas.
- Métodos prosódicos: Analizar, por ejemplo, tono, velocidad y cambios en la pronunciación.
- Métodos visuales: Analizar el gesto, la mirada y el movimiento del cuerpo. Por ejemplo, si se piensa en una persona que está dando una conferencia, un gesto frecuente es el de mirar al siguiente orador o al moderador cuando el turno está a punto de terminar.

En este trabajo se consideran los métodos gramaticales y los métodos prosódicos. A la fecha, los avances más significativos dentro del campo de reconocimiento de partes de un discurso se encuentran principalmente relacionado con el idioma inglés, aunque se están haciendo avances para mejorar la situación de los otros idiomas. Para poder analizar los métodos gramaticales se necesita realizar extracción de información y

más específicamente, extracción de relaciones, es decir, extraer las relaciones halladas entre las entidades en un texto. El método usado para este fin se llama tokenización o *tokenization* en inglés.

Tokenización

Tokenización es el proceso de dividir un texto en palabras, términos, oraciones, u otras unidades significativas a las que se las llama *tokens*. Se usa esta denominación en lugar de llamarlas “palabras” porque los *tokens* incluyen los signos de puntuación, las direcciones URL, etc.

La forma más sencilla de *tokenizar* un texto es usar los espacios en blanco como delimitadores en una cadena de palabras. El conjunto de *tokens* se agrupa en una estructura llamada *doc*. Cada *token* tiene una etiqueta denominada *Part Of Speech* (POS). Las etiquetas POS señalan si un *token* es un sustantivo, pronombre, adjetivo, verbo, es decir, qué función gramatical ocupa en la oración según su posición. Para realizar un etiquetado correcto se requiere una evaluación del contexto en el que dicho *token* se encuentre. Se pueden distinguir dos grupos grandes de etiquetas:

1. *Universal POS Tags*: Estas etiquetas se utilizan en *Universal Dependencies* (UD), un proyecto² que está desarrollando un corpus de anotaciones lingüísticas entre lenguas para muchos idiomas. Se basan en el tipo de palabras y su función dentro de una oración. Por ejemplo, se define la etiqueta NOUN (sustantivo común), ADJ (adjetivo), ADV (adverbio).
2. *Detailed POS Tags*: Estas etiquetas son el resultado de la división de las etiquetas POS universales en varias etiquetas, como NNS para los sustantivos plurales comunes y NN para el sustantivo común singular, ambos derivan de NOUN y se relacionan con sustantivos comunes. Estas etiquetas son específicas de cada idioma, pueden variar de un idioma a otro.

La mayoría de las librerías que realizan clasificación de *tokens* en etiquetas POS están basadas en el proyecto *Penn Treebank*³, pero éste solo contiene información sobre el idioma inglés. Las librerías que soportan reconocimiento de partes de un discurso (*Part-of-Speech Tagging*) en idioma castellano están conectadas a *Multiling RST Treebank*. Tanto *Penn Treebank* como *Treebank* son corpus lingüísticos que fueron anotados manualmente siguiendo la Teoría de la Estructura Retórica o *Rhetorical*

²<https://universaldependencies.org>

³<https://www ldc.upenn.edu/>

Structure Theory postulada por Mann y Thompson en 1988 [Mann y Thompson, 1987]. Las etiquetas que se encuentran en *Treebank* para el idioma español se muestran en Tabla 2.1:

Tabla 2.1 Etiquetas POS definidas en *Treebank*.

Clases abiertas	Clases cerradas	Otros
ADJ (Adjetivo)	ADP (Adposición)	PUNCT (Puntuación)
ADV (Adverbio)	AUX (Verbo Auxiliar)	SYM (Símbolo)
INTJ (Interjección)	CONJ (conjunción de coordinación)	X (Otro)
NOUN (Sustantivo)	DET (Determinante)	
PROPN (Nombre propio)	NUM (Número)	
VERB (Verbo)	PART (Partícula)	
PRON (Pronombre)		
SCONJ (Conjunción subordinante)		

2.3. Computación en la nube

2.3.1. Introducción

De acuerdo al Instituto Nacional de Estándares y Tecnologías, “*la computación en la nube es un modelo para habilitar el acceso de red ubicuo, conveniente y bajo demanda a un grupo compartido de recursos informáticos configurables (por ejemplo, redes, servidores, almacenamiento, aplicaciones y servicios) que se pueden aprovisionar y liberar rápidamente con un mínimo esfuerzo de administración o interacción con el proveedor de servicios. Este modelo de nube se compone de cinco características esenciales, tres modelos de servicio y cuatro modelos de implementación*”. [Cloud, 2011].

Se conoce entonces, como computación en la nube, a un grupo de servicios de tecnologías que pueden ir desde la provisión de infraestructura hasta la de almacenamiento. La computación en la nube surge como una respuesta a la creciente demanda de capacidad de almacenamiento y procesamiento de las empresas que, al ver que los costos de mantener su propio equipamiento crecían, empezaron a optar por tercerizar estos servicios.

A la fecha, los grandes prestadores de servicios en la nube son las compañías Amazon, Google, Microsoft e IBM [Qian et al., 2009]. Uno de los conceptos más importantes que introdujo el surgimiento de la computación en la nube son los modelos de negocio basados en servicios o, como se denominan comúnmente, *as a service*.

2.3.2. Proceso de Negocio como un servicio

El modelo de negocio que plantean las empresas que proveen servicios en la nube se conoce como BPAAS, “*Business Process As A Service*”, debido a que proveen un servicio, ya sea software, hardware, funcionalidad o consultoría a cargo de personal especializado. Estos servicios se ponen a disposición del cliente a través de internet, y éste paga por consumir dichos servicios. Los modelos de servicios en la nube son los siguientes [Docs, 2023a]:

- Infraestructura como servicio (IaaS, de *Infrastructure as a service*): Las empresas hacen disponible equipamiento, como por ejemplo máquinas virtuales, almacenamiento de datos y funciones de red para que el cliente que contrata este servicio, disponga de la infraestructura como si fuera local. Solo debe hacerse cargo de algunas cuestiones de configuración. El ejemplo comercial más conocido a la fecha de publicación de este trabajo es el de Amazon Web Services, cuyos servicios EC2 y S3 ofrecen cómputo y servicios de almacenamiento respectivamente.
- Plataforma como servicio (PaaS, de *Platform as a service*): Es una versión evolucionada de la IaaS ya que la empresa no solo provee diversos servicios, sino que también provee mano de obra especializada para realizar mantenimiento de software, planificar la capacidad y gestionar la escalabilidad, permitiendo así que el cliente no tenga que preocuparse por esas cuestiones y pueda enfocarse en el desarrollo de software. Los siguientes son los principales proveedores de plataformas como servicios actualmente:
 - *Amazon Web Services* (AWS) [Amazon, 2023] es una plataforma de servicios en la nube que ofrece una amplia gama de servicios de infraestructura, poder computacional, almacenamiento de bases de datos, etcétera.
 - *Google Cloud Platform* (GCP) [Google, 2023] es una plataforma de servicios en la nube que proporciona a los desarrolladores una variedad de herramientas y servicios para crear e implementar aplicaciones.
 - *Microsoft Azure* [Microsoft, 2023] es una plataforma de servicios en la nube que proporciona a los desarrolladores una gama de herramientas y servicios para crear e implementar aplicaciones web.

- Software como servicio (SaaS, de *Software as a service*): Esta opción delega la totalidad de las cuestiones técnicas a la empresa proveedora de servicios y le da al cliente mucha más comodidad al desligarlo aún más de las responsabilidades, haciendo así que la única preocupación del usuario sea utilizar el software. Ejemplo claro de este tipo son los servicios de mensajería electrónica, o los productos de Google, como *Google Drive*.
- Inteligencia artificial como servicio: En esta opción se combina tecnología de IA con el modelo de computación en la nube, por lo que el objetivo es permitir acceso a servicios de esta índole mediante la red, de manera ubicua, conveniente y bajo demanda haciendo uso de un conjunto compartido de recursos informáticos configurables. Esta alternativa surge de la necesidad de las organizaciones de afrontar uno de los grandes retos de la actualidad: la adopción e integración de la IA en los procesos de negocio. La dificultad se debe a la escasez de expertos en IA, la falta de capacidades y presupuestos de las organizaciones para configurar y mantener los recursos necesarios y el conocimiento limitado sobre cómo implementar y configurar estos sistemas de manera efectiva [Chui y Malhotra, 2018]. Para facilitar la aplicación de IA, los proveedores de la nube como Amazon, Google, IBM o Microsoft empezaron a ofrecer opciones basadas en *Machine Learning* y *Deep Learning*. De esta manera se pueden usar modelos de análisis de datos sin necesidad de aprender algoritmos.

La Figura 2.2 ilustra la denominada “pirámide tecnológica” de la computación en la nube y los productos comerciales que, a la fecha, son populares.



Figura 2.2 Modelos de servicios de computación en la nube. Extraído de [TIC, 2017].

Las razones por las que en muchos proyectos se decide incorporar servicios basados en computación en la nube son varias: a) para minimizar los costos, b) para no sopesar en las decisiones de infraestructura y c) para poder hacer uso de servicios ya

desarrollados. En el caso específico de la IA como servicio, su uso resulta beneficioso debido a que crear y entrenar modelos lleva una cantidad considerable de esfuerzo computacional.

2.3.3. Infraestructura como Servicio

Las empresas que ofrecen infraestructura como servicio, permiten desplegar aplicaciones en la nube utilizando componentes alojados en varios sitios. En un principio se acostumbraba a hacer el despliegue de la lógica de las aplicaciones en un servidor propio. Luego, con el avance tecnológico que habilitó la computación en la nube, surgieron proveedores de servicios que empezaron a ofrecer cómputo a través de sus servidores a cambio del pago de una cuota que variaba según las especificaciones computacionales. Este proveedor le asignaba a cada cliente uno o varios de sus nodos y permitía que cada uno tuviera administración total sobre los recursos debido a que se seguía un modelo *serverfull*. El modelo *serverfull* es una opción dentro de la computación en la nube donde el cliente puede desplegar sus aplicaciones y es responsable de sus recursos asignados. Las aplicaciones se ejecutan en ese servidor y los clientes son los responsables de aprovisionar y administrar los recursos para ella. Algunos de los problemas que conlleva la utilización de infraestructura como servicio con modelo *serverfull* son:

- Se cobra por mantener el servidor activo incluso cuando no se atiende ninguna solicitud.
- El mantenimiento del servidor y demás recursos queda en manos del cliente.
- Debe haber algún responsable de aplicar las actualizaciones de seguridad apropiadas al servidor.
- A medida que el uso aumenta, se necesita administrar la escala hacia arriba de manera manual. Así mismo, se debe administrar la escala hacia abajo cuando no se tiene tanto uso.

En las organizaciones más grandes, estas cuestiones están a cargo del equipo de infraestructura y, por ende, no supone un problema. Por otro lado, para pequeñas empresas y desarrolladores individuales esto puede resultar demasiado para gestionar, ya sea por falta de personal o bien por falta de especialización en infraestructura. Esta situación deriva a que se pierda el foco sobre el trabajo más importante, el de desarrollar y mantener la aplicación. Como resultado de esto surgió el modelo de computación sin servidor o computación *serverless*.

2.3.4. Computación *serverless*

La computación sin servidor (conocida por su abreviación en inglés como *serverless*) es un modelo de ejecución en el que el proveedor en la nube es responsable de ejecutar un fragmento de código mediante la asignación dinámica de los recursos y se encarga de facturar solo por la cantidad de recursos utilizados [Baldini et al., 2017]. Hay varias formas para calcular estos costos, pero las más comunes se basan en el tiempo de uso, en la memoria usada o en la cantidad de solicitudes recibidas. El código generalmente se ejecuta dentro de contenedores sin estado que pueden ser activados por una variedad de eventos tales como solicitudes HTTP, eventos de base de datos, servicios de edición-suscripción, alertas de monitoreo, carga de archivos, eventos programados, entre otros. El código que se envía al proveedor en la nube para la ejecución está, generalmente, estructurado en forma de función.

En este trabajo será de utilidad elegir un servicio de la nube que permita ejecutar aplicaciones en contenedores, ya que se puede crear la aplicación en menos tiempo al no ser necesaria la administración de la infraestructura donde se ejecutará el código. Los contenedores son un paquete de elementos que permiten crear un entorno dónde correr aplicaciones. Funcionan como un empaquetado de dependencias, ya que virtualizan un sistema operativo para que solamente contenga las librerías y paquetes necesarios para ejecutar una aplicación. Gracias a estos servicios se puede escribir una aplicación en cualquier lenguaje y poder ejecutarla en diferentes entornos muy diferentes de una manera simple, pero sin comprometer la complejidad de la arquitectura diseñada. La comparación entre los distintos servicios se realiza en el Capítulo 3, Sección 3.3.

2.4. Servicios Web

Los servicios web son sistemas de software que intercambian datos a través de estándares de transferencia con los que consiguen interactuar con aplicaciones de clientes y otros servicios. Estas interacciones se basan en la arquitectura cliente-servidor, donde el servidor le provee funcionalidades al cliente en forma de servicios encapsulados. Los servicios web se caracterizan por ser interoperables y extensibles.

2.4.1. Arquitectura REST

La lógica de la aplicación necesita estar enmarcada en una arquitectura de servicios web para que el cliente (navegador) pueda obtener la información necesaria para poder ser consultada o modificada por el usuario. REST (acrónimo de *Representational State Transfer*) significa transferencia de estado representacional y fue un concepto creado por el informático Roy Fielding [Fielding, 2000]. REST es un conjunto de criterios de diseño que definen un tipo de arquitectura para construir sistemas. Los principios de diseño REST son los siguientes:

- Cliente-servidor
- Identificador único (URI)
- Sin estado (*stateless*)
- Uso de operaciones HTTP

La arquitectura orientada a recursos (ROA) es la arquitectura que mejor cumple con los criterios REST. Una ROA expone datos internos que tienen que ver con la lógica de negocio a través de un simple interfaz de procesamiento de documentos.

Esta arquitectura se basa en solo cuatro conceptos (recursos, sus nombres (URIs), sus representaciones y los vínculos entre ellos) y cuatro propiedades (direccionabilidad, ausencia de estado, conectividad y una interfaz uniforme).

El siguiente procedimiento sirve para diseñar servicios web RESTful teniendo en cuenta todas las restricciones listadas [Richardson y Ruby, 2008].

1. Obtener el conjunto de datos
2. Dividir el conjunto de datos en recursos. Luego se procede con los pasos 3, 4, 5, 6, 7, 8 y 9 para cada recurso.
Para cada tipo de recurso:
 3. Nombrar el recurso asignándole un URI.
 4. Exponer un subconjunto de la interfaz uniforme.
 5. Diseñar las representaciones aceptadas del cliente.
 6. Diseñar las representaciones disponibilizadas al cliente.
 7. Integrar el recurso con los recursos existentes en caso de que sea necesario.
 8. Diseñar el flujo de control pensando en el curso típico y normal de los acontecimientos.
 9. Diseñar un flujo de control considerando las condiciones de error para crear respuestas apropiadas.

Como resultado, se obtiene un servicio web en el que el cliente manipula el estado del recurso enviando una representación como parte de una solicitud PUT o POST. Las solicitudes DELETE funcionan de la misma manera, pero no hay representación. El servidor manipula el estado del cliente enviando representaciones en respuesta a las solicitudes de tipo GET.

Luego, es necesario pensar en la creación de una API (*Application Programming Interface*) como un mediador entre los usuarios o clientes y los recursos o servicios web que se desea desarrollar. Es una forma de que una organización comparta recursos e información manteniendo la seguridad, el control y la autenticación.

2.5. Enfermedades Neurodegenerativas

Las enfermedades neurodegenerativas, también conocidas como enfermedades degenerativas del sistema nervioso, provocan progresivamente daños a las células nerviosas del cerebro y sus conexiones. A este proceso se lo conoce como neurodegeneración y los lóbulos del cerebro, dependiendo del tipo de afectación, se atrofian. A la fecha, está claro que esto ocurre como consecuencia de anomalías en el proceso de ciertas proteínas que, al acumularse en el tejido nervioso, dentro y fuera de las neuronas, producen manifestaciones clínicas, principalmente demencia [Mora Teruel y Segovia de Arana, 2002].

En todas las enfermedades neurodegenerativas hay algún tipo de proceso anormal de las proteínas neuronales. Estas proteínas defectuosamente procesadas se acumulan fácil cuando los mecanismos celulares para su eliminación son ineficaces. Cuando las proteínas específicas de cada circuito celular son procesadas de forma inadecuada se produce el mal funcionamiento de las diferentes clases de neuronas y las consiguientes manifestaciones de la enfermedad.

El genoma humano posee alrededor de 35000 genes [Institute, 2022]. Muchos de ellos codifican proteínas solamente en el sistema nervioso. Otros, codifican proteínas que se expresan en distintos tipos de neuronas. Así, ciertas poblaciones neuronales estarán más vulnerables a los cambios originados por variaciones genéticas, por variaciones ambientales o por la combinación de ambos. En una neurona, una pequeña perturbación puede eventualmente ser importante y tener consecuencias considerables. Los avances en genética molecular han permitido profundizar el estudio de las enfermedades neurodegenerativas y señalar un camino de esperanzas en los tratamientos.

De acuerdo a la Organización Mundial de la Salud (OMS)⁴ la demencia es un síndrome que provoca el deterioro de la memoria, el pensamiento, el comportamiento y la capacidad para realizar actividades cotidianas. Las personas con demencia pueden perderse en lugares familiares, experimentar cambios bruscos de humor, olvidar palabras sencillas y sufrir un deterioro de la memoria a corto plazo (el síntoma inicial más común de demencia). En América, más de 10 millones personas viven con demencia. Las estimaciones muestran que cada 20 años, se duplicará el número de personas con este trastorno. América Latina y el Caribe serán los más afectados, con un incremento de 3,4 millones de personas con demencias en 2010, a 7,6 millones en 2030. Con estas cifras y estadísticas, la OMS ya reconoce a la demencia, como una prioridad de salud pública.

Si bien a las demencias mayormente se las asocia con un origen degenerativo, pueden ser también clasificadas como: [De la Vega, R. y Zambrano, A., 2007]

- De origen metabólico o nutricional
- De origen vascular
- De origen infeccioso
- De origen tóxico
- De origen neoplásico

Dentro de las de origen degenerativo las más frecuentes son: [De la Vega, R. y Zambrano, A., 2007] la enfermedad de Alzheimer, la demencia con cuerpos de Lewy, la demencia frontotemporal o enfermedad de Pick, la asociada a la enfermedad de Parkinson, la enfermedad de Huntington y la parálisis supranuclear progresiva.

Las enfermedades degenerativas son la causa más frecuente de demencia en nuestro medio, el 5-10 % de ellas son demencias frontotemporales, antes llamada enfermedad de Pick. Este porcentaje aumenta cuando los pacientes son menores de 65 años, que es la segunda causa de demencia, después de la enfermedad de Alzheimer y la demencia con cuerpos de Lewy.

Este trabajo se relaciona con la demencia frontotemporal. En la sección siguiente, se introduce la misma.

⁴<https://www.paho.org/es/temas/demencia>

2.5.1. Demencia Frontotemporal

La demencia frontotemporal o enfermedad de Pick es un proceso caracterizado por una atrofia selectiva de neuronas en los lóbulos frontal y temporal y, en un menor grado, en el hipocampo [Mora Teruel y Segovia de Arana, 2002]. Clínicamente, esta demencia se define como un conjunto de síndromes caracterizados por una alteración progresiva del comportamiento y/o lenguaje, en relación con una atrofia focal de los lóbulos frontal y temporal. La región frontal es el área del cerebro encargada de regular la conducta, en tanto que la región temporal regula las capacidades lingüísticas.

La mayoría de los pacientes, incluso en etapas tempranas de la enfermedad, presentan una afectación importante de las funciones ejecutivas, el lenguaje y la memoria de trabajo. Por lo general, están preservadas las habilidades visoconstruccionales, a diferencia de la enfermedad de Alzheimer. La demencia frontotemporal, aunque es menos frecuente que la enfermedad de Alzheimer, es responsable de un porcentaje importante de las demencias degenerativas [Iragorri Cucalón, 2007]. Sus manifestaciones varían de acuerdo a la distribución de la atrofia en los lóbulos afectados.

Su hallazgo data de un siglo atrás y se lo atribuye a Arnold Pick, quien describió inicialmente las características clínicas de un paciente con atrofia lobar, circunscrita a los lóbulos frontal y temporal. Pick describió una serie de casos que presentaban cambios de personalidad y alteraciones conductuales, acompañadas con un deterioro del lenguaje o afasia y, lo que resultó más trascendente para la época, atrofia cerebral focalizada en la parte anterior de los lóbulos temporales y frontales [Pick, 1904].

Los avances años posteriores daban cuenta de atrofiaciones cerebrales focales o “demencias focales” y se empezó a notar que los estudios de las mismas diferían de lo descrito por Alois Alzheimer y su postulado sobre el deterioro en el área de la memoria [Alzheimer, 1911]. Pero no fue hasta la década de los 80, con el advenimiento de las técnicas modernas de neuroimágenes, que el tema volvió a tener interés. Así, se identificaron otros síndromes cognitivos progresivos que podrían confundirse con accidentes vasculares u otras patologías cerebrales focales. En 1982, Mesulam propuso la expresión afasia primaria progresiva, al reportar seis casos de pacientes que presentaban una afectación aislada del lenguaje [Mesulam, 2001]. Específicamente, estos casos presentaban afasia progresiva asociada a atrofia inespecífica de las regiones perisilvianas izquierdas⁵. Este

⁵Hacia finales del siglo XIX y comienzos del siglo XX, y basándose en observaciones clínicas, se propuso que existía un “área del lenguaje” en el cerebro que corresponde, en general, a la región perisilviana del hemisferio izquierdo. <https://neurologia.com/noticia/5536/el-area-del-lenguaje-en-el-cerebro-es-mas-amplia-de-lo-que-se-creia>

panorama habilitó a Mesulam a reunir conocimiento y experticia en clínica, y proponer así, una definición que se haría comúnmente aceptada, la de afasia progresiva primaria (APP). Afasia progresiva primaria engloba a un grupo de procesos neurodegenerativos que causan un deterioro del lenguaje, pero con relativa incidencia en otros dominios cognitivos. Desde la descripción de los seis casos de Mesulam, cerca de una centena más se han descrito en la literatura médica.

Los avances de la ciencia en el tema han permitido avanzar en la identificación de características clínicas, bioquímicas, patológicas y genéticas asociadas a la demencia frontotemporal. Los expertos en el tema coinciden en clasificarla de acuerdo a sus tres grandes variantes clínicas [Hodges y Miller, 2001]: la variante frontal o conductual, la variante semántica y la variante afasia primaria progresiva, siendo la más frecuente la variante conductual. La tabla 2.2 describe brevemente a cada una de ellas.

Tabla 2.2 Clasificación de las demencias de tipo frontotemporal.

Variante	Síntomas	Lesión Anatómica
Variante frontal	Cambio de personalidad y comportamiento	Corteza orbitofrontal bilateral
Afasia Progresiva Primaria	Afasia no fluente, alteración en la expresión, se conserva la comprensión	Área perisilviana izquierda
Demencia semántica	Afasia anatómica fluente con alteración en la comprensión y pérdida del significado	Corteza temporal inferolateral izquierda o bilateral

La variante frontal representa el 60 % de los casos de demencia frontotemporal [Swanson et al., 2019] y su manifestación son los cambios de comportamiento (conductas inapropiadas en público, impulsividad), de personalidad (cambios en la dieta, comportamientos repetitivos o compulsivos), alteraciones de las emociones (apatía, pérdida de empatía). En estos pacientes es característico el desorden en el comportamiento social y de personalidad. Por esto, manifiestan una conducta social inapropiada que parece insensible a las normas que regulan el comportamiento en sociedad. Aunque es difícil determinar cuándo comienza esta enfermedad, puede ser notorio en estos pacientes, comenzar haciendo algunas declaraciones extrañas o inapropiadas o expresar opiniones inusuales. Sin embargo, en un período corto de tiempo, dichos cambios en la interacción social no se restringen a eventos

singulares, sino que reflejan una pérdida subyacente de las reglas que regulan el comportamiento apropiado [DFT, 2023]. El perfil cognitivo de los pacientes con DFT variante conductual se manifiesta particularmente por fallas en las funciones ejecutivas, incluyendo dificultades en la atención selectiva y control inhibitorio, cambios de tareas y planificación y organización.

La variante semántica se caracteriza por una pérdida del significado de las palabras, aun cuando se conservan los aspectos fonológicos y sintácticos del lenguaje. Los pacientes evidencian un deterioro progresivo en la comprensión de palabras, especialmente nombres y en el reconocimiento de objetos. Conservan la habilidad para producir un discurso fluido pero sin la utilización de palabras claves y que puede llegar a ser progresivamente difícil de comprender. Asimismo, pierden la habilidad para reconocer el significado de palabras específicas o para denominar espontáneamente objetos familiares, cotidianos. El perfil cognitivo de los pacientes con DFT variante semántica se mantiene notablemente conservado [Ineco, 2023b].

La variante afasia primaria progresiva de la demencia se caracteriza con un lenguaje espontáneo lentificado, con pausas frecuentes debido a las anomias pero sin franco agramatismo. Se conserva además la articulación y la prosodia. Sin embargo, se presentan dificultades en la evocación, tanto en el lenguaje espontáneo como en tareas de denominación o repetición de oraciones, en la escritura y pueden presentar dificultades en la memoria de trabajo (o bucle fonológico). La comprensión de palabras aisladas se encuentra conservada así como la reproducción de palabras simples y cortas [Ineco, 2023a]. Se considera que las distintas variantes de APP representan aproximadamente el 40% de los casos de demencia frontotemporal [Swanson et al., 2019].

La demencia frontotemporal implica grandes dificultades para los cuidadores y alta dependencia de los pacientes. Independientemente de la forma de presentación, finalmente se produce un deterioro en el funcionamiento diario de la persona, que se vuelve cada vez más dependiente para realizar las actividades cotidianas. El avance y duración de la enfermedad es muy variable, dependerá del tipo de demencia frontotemporal que se padezca, del acervo intelectual previo de la persona afectada y de la calidad de vida que pueda llevar durante los años que tenga que convivir con esta enfermedad. A la fecha, no tiene cura.

La próxima sección, introduce la variante de afasia progresiva primaria.

2.5.2. Afasia Progresiva Primaria

La afasia progresiva primaria (APP) es un síndrome clínico caracterizado por un deterioro del lenguaje con origen neurodegenerativo. En comparación con otras patologías degenerativas, como la enfermedad de Alzheimer, la APP es, al momento, poco difundida, aunque los primeros reportes de pacientes con afectación progresiva del lenguaje se remontan a finales del siglo XIX (Pick, 1892; Sérieux, 1893). En lo que respecta a la literatura moderna, fue Mesulam (1982) quien reportó una serie de casos de pacientes con déficits lingüísticos de progresión gradual, sin mostrar afectación de ningún otro dominio cognitivo. Esto significa que tanto la memoria, como otras habilidades implicadas en el recuerdo de una persona (memoria episódica) se mantienen preservadas, y la limitación se centra en las actividades de la vida diaria y social, únicamente por la disfunción del lenguaje.

Actualmente, los criterios internacionales definen tres variantes de la Afasia Progresiva Primaria [Gorno-Tempini et al., 2011]: APP variante semántica (APP-vs), APP variante no fluente (APP-vnf) y APP variante logopénica. Si bien la característica común de las tres variantes es la afectación súbita y progresiva de las estructuras neurales que soportan el lenguaje, los dominios lingüísticos afectados en cada una son específicos, así como el área específica comprometida que puede observarse a partir de técnicas de neuroimágenes [Soriano y Martínez-Cuitiño, 2020].

Afasia Progresiva Primaria variante semántica (APP-vs)

La APP-vs también se la suele denominar “demencia semántica” implica en los pacientes un lenguaje fluido, es decir, sin síntomas de apraxia del habla, pero lleno de errores y anomias en la producción (dificultad para encontrar la palabra). También, se presentan errores en la comprensión de la palabra, ya sea en forma auditiva como visual.

La parte del cerebro afectada cuando se diagnostica esta variante, es la región temporal anterior, de forma bilateral, aunque de predominio en el hemisferio izquierdo (ver Figura 2.3). Esta es la región que permite el conocimiento del significado de las palabras [Matías, 2020]. Por este motivo, los pacientes con afasia semántica tienen dificultades en la comprensión de palabras y también presentan fallos en su denominación (por ejemplo tigre por león). En cambio, la emisión del lenguaje es fluida. Durante una conversación, los pacientes pueden preguntar por el significado de palabras oídas como escritas que no entienden, pero también pueden presentar

problemas para elegir qué imagen se relaciona con una palabra. Esto es así porque el área afectada compromete el funcionamiento del sistema semántico, que es el almacén de representaciones conceptuales de nuestra mente. Si la información de este almacén está degradada o es inaccesible no importa por qué modalidad se intente recuperarla, el resultado será siempre el mismo [Soriano y Martínez-Cuitiño, 2020]. Suelen también tener dificultades para el reconocimiento de rostros conocidos. Es decir, los pacientes saben que se trata de un rostro familiar aunque no pueden identificar de quien se trata (a diferencia del Alzheimer que, directamente, desconocen el rostro). Lo mismo puede suceder con objetos, voces, sonidos o gustos.

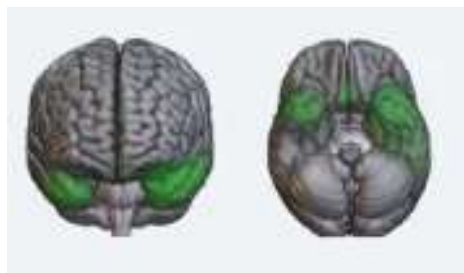


Figura 2.3 Área afectada por la Afasia progresiva primaria semántica (APP-vs). Extraído de [Matías, 2020].

Afasia Progresiva Primaria variante no fluente (APP-vnf)

La APP-vnf también es conocida como agramática, implica dificultades en la producción del lenguaje, en tanto que la comprensión del lenguaje y el resto de las funciones cognitivas se conservan hasta estadios avanzados de la enfermedad. Los pacientes que padecen esta variante de afasia sufren la pérdida de la fluidez del lenguaje, situación que puede llevarlo hasta el mutismo.

De acuerdo a estudios de neuroimagen hechos a pacientes con la variante no fluente de afasia, el área del cerebro que se ve afectada muestran atrofia de las regiones perisilvianas del hemisferio izquierdo, específicamente de la región frontal posterior y de la ínsula en el hemisferio izquierdo [Brambati et al., 2009]. La Figura 2.4 ilustra dicho área.

La característica principal del habla de estos pacientes es ser laboriosa y esforzada, pausada, lo que se conoce como apraxia del habla [Soriano y Martínez-Cuitiño, 2020]. Además, evidencian notorias dificultades para producir oraciones gramaticales correctas, conocido como agramatismo. Los pacientes realizan un enorme esfuerzo para producir un sonido específico, lo que lleva a articular sonidos que no son reconocidos por alguna

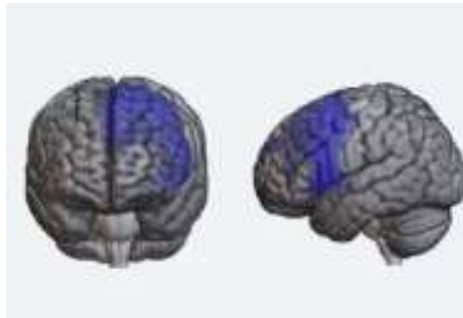


Figura 2.4 Área afectada por la Afasia progresiva primaria no fluente (APP-vnf). Extraído de [Matías, 2020].

lengua (parafrasias fonéticas) o bien, dificultad notoria para organizar la secuencia de sonidos de una palabra en forma correcta sin omitir, desplazar, agregar o transponer alguno (parafrasias fonémicas) [Soriano y Martínez-Cuitiño, 2020]. Estas dificultades están bien marcadas, no solo cuando el paciente intenta pronunciar una palabra sino también cuando se le solicita repetir.

El agramatismo se presenta tanto en la producción del lenguaje como así también en la capacidad de comprender oraciones afectando tanto la modalidad oral como la escrita [Wilson et al., 2010]. En relación con la producción oral, las oraciones que pronuncian son más cortas y sencillas. Si pronuncian una frase más elaborada, quedan sin poder completarla, quedando la misma carente de sentido, por faltar, justamente, las palabras más importantes para comprender el comunicado enunciado. Los pacientes suelen omitir los artículos, las conjunciones y las preposiciones, es decir, las palabras de función que complementan a las de contenido -sustantivos, verbos y adjetivos-. De esta forma el habla se convierte en una secuencia de palabras de contenido en la que no suele haber conjugación del verbo, un habla “telegráfica”.

Así, uno de los mayores inconvenientes gramaticales se registran en la formulación de los tiempos verbales. No solo al momento en que se los tiene que conjugar, sino también se presentan problemas a la hora de recuperarlos en forma aislada. Los sustantivos, en cambio, se encuentran relativamente conservados.

La lectura también se afecta en estos pacientes, presentando dificultades para la lectura de palabras no familiares y no-palabras (pseudopalabras)⁶ [Brambati et al., 2009]. Es decir, conservan la lectura de palabras conocidas (con mayor latencia y

⁶La noción de pseudopalabra se trata de una agrupación de letras que no constituye una palabra y que, por lo tanto, no permite representar una idea o un concepto, es un fragmento discursivo que no tiene significado.

algunas omisiones) pero la principal dificultad se evidencia cuando deben hacer uso de las reglas de conversión grafema-fonema. Estas reglas hacen referencia al hecho de que, cuando se lee, se asigna un sonido a las letras que forman una palabra. Algunas letras cambian de sonido según el lugar que ocupan, por ejemplo esto le pasa a la C, o sonidos que pueden venir por varias letras, como G/J.

De aquí, que el presente trabajo tiene como objetivo contribuir en este aspecto -el de la lectura- y medir y monitorear cuán afectada se encuentra la misma en pacientes con este tipo de afasia con el transcurso del tiempo, considerando que, conforme avanza la enfermedad, los pronósticos actuales indican que la persona llegará a no hablar, no escribir y, por supuesto, no leer.

Los pacientes tienen completa conciencia de todas estas dificultades, en especial, de la latencia en el habla. Y, aunque el resto de las funciones cognitivas inicialmente se conserva, a medida que la enfermedad progresa se detecta una disminución de los recursos atencionales, la afectación de la memoria de trabajo y de las funciones ejecutivas y de las praxias. A la fecha, el único tratamiento prescrito es la terapia con un especialista fonoaudiólogo para lentificar la progresión del deterioro.

Afasia Progresiva Primaria variante logopéica (APP-vl)

La variante logopéica es conocida también como fonológica y es la descrita más recientemente. Los pacientes que sufren esta variante experimentan dificultades para recuperar las palabras y un lenguaje lentificado debido a las anomias que presentan, es decir, a las dificultades para denominar un objeto o concepto, para acceder o producir el nombre o etiqueta con la que se lo designa. La presencia de anomias se manifiesta tanto en el lenguaje espontáneo como también ante una evaluación formal. También, se evidencian dificultades para repetir frases y oraciones, aunque estos pacientes pueden repetir adecuadamente palabras y no-palabras [Soriano y Martínez-Cuitiño, 2020].

Los estudios de neuroimagen de los pacientes con APP-vl muestran atrofia de la región perisilviana del hemisferio izquierdo y de la región temporo-parietal-izquierdo [Soriano y Martínez-Cuitiño, 2020]. Las zonas que se ven afectadas están relacionadas con la búsqueda de palabras y la memoria de trabajo verbal. La Figura 2.5 ilustra dicho área en el cerebro.

En la variante logopéica, los pacientes no tienen dificultades en la articulación. Los errores fonológicos que presentan se deben al intercambio u omisión de algún fonema que deriva en palabras que se parecen en su forma con la que buscan. Tampoco

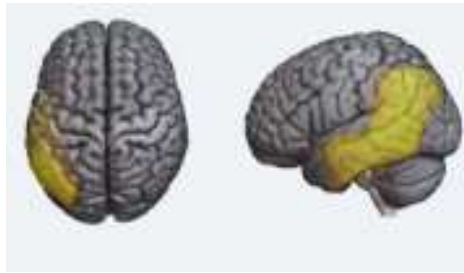


Figura 2.5 Área afectada por la Afasia progresiva primaria logopénica (APP-*vl*). Extraído de [Matías, 2020].

presentan pérdida de la prosodia normal del lenguaje [Mora Teruel y Segovia de Arana, 2002]. Aunque suelen producir oraciones un poco más cortas, el componente gramatical no se encuentra afectado [Gorno-Tempini et al., 2011]. La lectura en voz alta se afecta también, al igual que la variante no fluente, presentando alexia fonológica [Brambati et al., 2009]. Pero, a diferencia de éstos, la afectación sería más severa cuando deben leer no-palabras. La escritura también se altera según un perfil de alexia y agrafia fonológicas.

Los estudios indican que la principal función cognitiva comprometida de estos pacientes sería la memoria de trabajo, específicamente, el bucle fonológico. La memoria de trabajo es el sistema de memoria que mantiene y manipula la información de forma temporal, interviniendo en procesos cognitivos más complejos como la comprensión del lenguaje, la lectura o el razonamiento.

Nota: Los bucles son secuencias de instrucciones cíclicas que permiten repetir una acción mientras se está cumpliendo una determinada condición y el proceso no se para hasta que esta condición no se cumple. El típico ejemplo que se suele poner para entender mejor este componente es el de la persona a la que se le dice un número de varios dígitos (una clave, por ejemplo) y, hasta que logra encontrar un lugar donde apuntarlo, lo va repitiendo subvocalmente para que no se le olvide.

Capítulo 3

Estado del arte

El presente trabajo tiene como finalidad asistir a profesionales fonoaudiólogos en la realización del control y monitoreo de la evolución del discurso oral de lecturas, en pacientes que sufren afasia como consecuencia de alguna demencia. Para esto, se presenta una herramienta, denominada A-e-i! que, para su construcción, involucra los siguientes aspectos: procesamiento del lenguaje natural, reconocimiento de partes de un discurso y servicios provistos por plataformas de la nube.

El contenido de este capítulo se divide en dos. Por un lado, trata sobre el análisis de las herramientas que se tuvieron en cuenta para seleccionar aquellas más convenientes y emplearlas en el desarrollo de la aplicación. Por otro lado, se discuten trabajos relacionados que se asemejan a esta propuesta pero que no se ajustan completamente a la idea que motiva este trabajo.

3.1. Tecnologías de reconocimiento de discurso

Una alternativa para la transcripción de los audios a texto, es utilizar una red neuronal entrenada con audios de personas hispanohablantes. Otra alternativa es el uso de algoritmos estadísticos. Pero, como esto conlleva muchos costos de tiempo y de procesamiento, se optó por analizar distintos servicios en la nube de *speech-to-text* que proveen la transcripción de audio a texto a través de modelos preentrenados. Las opciones que se consideraron fueron:

- Facebook model (wav2vec2 large xlsr-53-spanish): Es un modelo del tipo codificador-decodificador de voz (seq2seq) diseñado para reconocimiento de voz

automático y traducción de voz. Utiliza en el fondo un modelo llamado Wav2vec 2.0 preentrenado como codificador. Wav2vec 2.0 recibe como entrada audio de voz, lo divide en tramos y cambia su representación codificando los tramos a través de una red neuronal convolucional multicapa. Luego, estas representaciones se envían a un decodificador basado en la biblioteca de transformadores *Huggingface* de Python [Matetext, 2023]. Como se puede ver en el nombre del modelo, tiene soporte para el idioma español pero solo con el dialecto de acento neutro.

- Google Speech Recognition: Este modelo, además de la posibilidad de transcribir audios, también permite obtener sugerencias de continuación para una frase, detección automática del idioma, filtrado de contenido inapropiado, etcétera. Funciona gracias a la existencia de una red neuronal de aprendizaje profundo. Tiene soporte para el idioma español en diferentes dialectos. La comunicación con el servicio puede ser síncrona o asíncrona. Los formatos de audio que soporta son: MP3, FLAC, LINEAR16, MULAW, AMR, AMR_WB, OGG_OPUS, SPEEX_WITH_HEADER_BYTE, WEBM_OPUS. Para acceder a este modelo se debe usar la API de Google Cloud *Speech-to-Text* [Docs, 2023b].
- Microsoft Azure Cognitive Services: Microsoft ofrece un componente de voz a texto en su plataforma Azure Cloud Services [Docs, 2023c]. Otras funciones relacionadas con el área de *speech-to-text* incluyen texto a voz, traducción de voz a determinado idioma y reconocimiento de la persona hablante. Estas funcionalidades son posibles gracias al aprendizaje por transferencia, técnica en la que se toma una red neuronal pre-entrenada, se elimina el clasificador de la red que se encuentra en la última capa y se reemplaza con nuevos clasificadores. Solo funciona de manera síncrona, no devuelve resultados intermedios durante la ejecución. Este componente tiene soporte para idioma español en varios dialectos. Los formatos de audio que soporta son: WAV, MP3, OPUS/OGG, FLAC, ALAW en contenedor wav y MULAW en contenedor wav.
- IBM Watson: IBM ofrece un componente de voz a texto en Watson Cloud Services [Docs, 2021]. Permite personalizar la experiencia de transcripción a texto a través de distintos parámetros. El sistema utiliza inteligencia artificial para combinar información sobre gramática y estructura del lenguaje con conocimiento sobre la composición de la señal de audio para transcribir la voz humana con precisión. A medida que procesa un archivo de audio se actualiza retroactivamente la transcripción. Los formatos de audio que soporta son: ALAW, basic, FLAC, G729, L16, MP3, formatos MPEG, MULAW, OPUS/OGG, WAV y WEBM. Permite tres tipos de comunicación, la síncrona, asíncrona o la conexión con

WebSockets para cuando se necesita lograr un intercambio de datos persistente, full-dúplex y de baja latencia entre el cliente y el servidor. Tiene soporte para el idioma español en varios dialectos.

Quizás la fuente más evidente de degradación del rendimiento en el reconocimiento de voz es la presencia de ruido. El ruido puede clasificarse como proveniente del ambiente, es decir, tráfico, lluvia, etc. o como proveniente de personas hablando, es decir, tos, estornudo, respiración, etc. Para poder comparar los distintos servicios es preciso saber qué métricas se usarán.

3.1.1. Selección de la tecnología de reconocimiento de discurso

Se realizó la comparación de las métricas BLEU y WER.

- BLEU: *Bi-Lingual Evaluation Understudy* (BLEU) es una métrica que puede calcularse utilizando más de una traducción de referencia, por lo que los resultados presentan una mayor robustez a la medida frente a traducciones libres realizadas por humanos. Se calcula normalmente a nivel de frases usando el resultado de la media geométrica para la cantidad de n-gramas a utilizar [Papineni et al., 2002]. En el estudio del lenguaje natural, los n-gramas engloban distintos tipos de elementos como por ejemplo fonemas, sílabas, letras, palabras.
- WER: *Word Error Rate* (WER) es una métrica común del rendimiento de un sistema de reconocimiento de voz o traducción automática. Se basa en que la dificultad general de medir el desempeño radica en el hecho de que la secuencia de palabras reconocidas, puede tener una longitud diferente de la secuencia de palabras correctas de referencia [Lippmann, 1997]. La fórmula de WER se deriva de la distancia de Levenshtein, cuyo algoritmo se detalla en la fórmula 3.1, trabajando a nivel de palabra en lugar de a nivel de fonema.

$$WER = (S + D + I)/N = (S + D + I)/(S + D + C) \quad (3.1)$$

donde:

S es el número de sustituciones,

D es el número de eliminaciones,

I es el número de inserciones,

C es el número de palabras correctas,

N es el número de palabras en la referencia ($N=S+D+C$)

Los términos “eliminación” y la “inserción” tienen que ver con la comparación

entre la referencia a la hipótesis. Entonces, considerando por ejemplo, la referencia “Buen día” y la hipótesis “Buen”, se afirma que se está frente a una eliminación. Por ejemplo, si hay 32 palabras en total en un archivo de transcripción y entre esas palabras habladas hay 14 errores entre sustituciones, inserciones y eliminaciones, el WER de esa transcripción sería 0.379. Eso se redondea a 0.38, lo que hace que el WER sea del 38 por ciento. Una desventaja del uso de una fórmula genérica como la de WER es que no se tiene en cuenta el efecto que los diferentes tipos de error pueden tener en la probabilidad de un resultado exitoso. Algunos errores pueden ser más perjudiciales que otros y algunos pueden corregirse más fácilmente que otros. Por otro lado, WER solo mide los errores de palabras a nivel superficial. No tiene en cuenta los roles contextuales y sintácticos de una palabra.

3.1.2. Justificación de la selección

La razón por la que se eligió trabajar con la métrica WER y no con otra métrica como por ejemplo el BLEU, pese a ser más específica para estos trabajos, fue su falta de información sobre el lenguaje español y sus hablantes, cosa que no afecta al WER [Chuang et al., 2020]. Luego, para decidir qué servicio en la nube utilizar, se calculó el WER y se tuvo en cuenta el rendimiento y el costo de cada uno.

Para ejecutar estos cálculos se creó un *Jupyter Notebook*. Un *Jupyter Notebook* permite crear documentos estructurados en forma de celdas independientes de entrada y de salida por lo que se puede ejecutar código, hacer cálculos, visualizar datos y resultados, de manera ordenada. Estas celdas contienen código en lenguaje Python en este caso. El *notebook* confeccionado para los cálculos contiene una función por cada servicio considerado que devuelve como resultado el WER calculado para un determinado grupo de archivos de audio, teniendo en cuenta el resultado esperado y el resultado arrojado por el servicio. El recurso en cuestión puede encontrarse en la plataforma *Google Colab*¹.

Se hizo la prueba con un total de 100 archivos de audio [Guevara-Rukoz et al., 2020] grabados por personas hispanohablantes de Argentina, para ello se ejecutaron las siguientes funciones que comparan la transcripción de referencia con la transcripción lograda:

¹Google Colaboratory, o “Colab”, como se lo denomina comúnmente, es un entorno que permite crear Jupyter Notebooks basado en la nube. Se ejecuta en un navegador web y permite que cualquier persona con acceso a Internet acceda a los documentos e interactúe con el código.

1. `test_google_speech_recognition`(archivo, referencia) (Código C.4 en Anexo)
2. `test_facebook_model`(archivo, referencia) (Código C.3 en Anexo)
3. `test_speech_azure`(archivo, referencia) (Código C.5 en Anexo)
4. `test_speech_ibm`(archivo, referencia) (Código C.6 en Anexo)

Tabla 3.1 Resultados del cálculo de WER para distintos servicios de transcripción.

Servicio	Promedio WER
Facebook Model	2.43
Google Speech Recognition	0.209
Azure Speech Recognition	0.181
IBM Watson	0.83

Los mejores resultados los arrojaron Microsoft Azure Cognitive Services y Google Speech Recognition. Analizando planes de pago, Microsoft Azure Speech Recognition es significativamente más económico que Google Speech Recognition. A la fecha, Microsoft ofrece 5 horas de transcripción gratuita por mes y luego cobra 1 USD por hora de audio. Google ofrece solo 1 hora de transcripción gratuita, después de lo cual el servicio cuesta 1.44 USD por hora de audio. Por estos motivos, se decidió usar Microsoft Azure Cognitive Services para tener acceso a Azure Speech Recognition. Otro de los beneficios de esta herramienta es su relación costo-beneficio y su escalabilidad. Debido a que el servicio está basado en la nube, los usuarios pueden escalar su uso según sea necesario, pagando solo por los recursos que se consumen. Esto lo convierte en una solución ideal para proyectos de todos los tipos y tamaños. Además, debido a que es de tipo *Software As a Service*, los usuarios no necesitan preocuparse por las complejidades de mantener y actualizar el código, lo que libera tiempo y recursos que pueden ser destinados a otras tareas.

3.2. Reconocimiento de partes de un discurso

Se optó por usar un servicio que permita realizar técnicas de procesamiento de lenguaje natural, en particular se necesita la técnica de reconocimiento de partes de un discurso o *Part-Of-Speech Tagging*. Desarrollar un sistema de reconocimiento de partes de un discurso propio sería muy costoso, ya que se necesita crear una infraestructura altamente disponible, rápida y escalable para que los modelos de aprendizaje automático que se requieren puedan ejecutarse. Esto supone una gran cantidad de recursos tanto de almacenamiento como de procesamiento.

3.2.1. Selección de herramientas para el reconocimiento de partes de un discurso

En la Sección 2.2.2 se describió el concepto de partes de un discurso, haciendo referencia a una aplicación de NLP que tiene la capacidad de reconocer a qué categoría gramatical pertenecen las palabras presentes en una oración y con esta información etiquetar cada una. Para escoger la herramienta que permitiera lograr esto, se optó por aquellas compatibles con Python. Además, se tuvieron en cuenta sólo aquellas que tenían soporte para el idioma español. Dentro de las herramientas más importantes que se encontraron, se distinguen las siguientes:

- *Language Understanding (LUIS)* de Microsoft Azure Cognitive Services: Es un servicio basado en la nube que incluye cuatro funciones principales: análisis de opiniones, extracción de frases clave, reconocimiento de partes de un discurso, reconocimiento de entidades nombradas y detección de idiomas. A la fecha, no tiene costo si se hacen menos de 10000 solicitudes por mes. A partir de esa cantidad, cuesta 1.50 USD cada solicitud [Microsoft Contributors, 2022].
- *Text Analysis APIs* de Komprehend AI: Es un servicio basado en la nube que proporciona reconocimiento de entidades nombradas, reconocimiento de partes de un discurso, análisis de sentimientos y detección de emociones. Su plan básico cuesta 79 USD por mes [Komprehend, 2020].
- *Cloud Natural Language API* de Google Cloud Platform: Está basado en la nube. Permite el análisis de entidades inspeccionando un texto ingresado como una cadena de *string* o mediante la lectura de un archivo. En el análisis de dicho texto se buscan entidades conocidas (nombres propios ya sea de figuras públicas como de instituciones, puntos de referencia, etc.). Para cada entidad conocida, se puede aplicar análisis de entidad que consiste en clasificar las opiniones sobre una entidad presente en un texto para tratar de determinar si genera reacciones positivas o negativas. Por otro lado, permite hacer análisis de sintaxis basándose en la estructura del idioma y clasificación de contenido del texto, es decir, clasificación de cada *token* según una categoría o etiqueta. Comienza a ser de pago a partir de las 5000 solicitudes mensuales, desde ahí hasta 1000000 cuesta 1 USD por solicitud y al superar esa cantidad, el precio de cada solicitud decrece a 0.50 USD [Google Cloud Platform, 2022].
- *TextRazor*: Está basado en la nube pero también tiene soporte para hosting local. Entre sus funcionalidades se encuentran la clasificación del texto en entidades y la división del mismo en tópicos según el tema tratado, el reconocimiento de

partes de un discurso, la extracción de relaciones entre las entidades y el análisis de opinión sobre cada una de ellas. El plan gratuito a la fecha, soporta hasta 500 solicitudes por día, de las cuales máximo 2 pueden ser simultáneas. Luego, el plan menos costoso cuesta 200 USD por mes [Text Razor, 2022].

- *Amazon Comprehend* de Amazon Web Services: Provee reconocimiento de entidades, el análisis de sentimientos, el análisis sintáctico con reconocimiento de partes del discurso, la extracción de frases claves y detección del idioma. Las solicitudes se miden en unidades de 100 caracteres (1 unidad = 100 caracteres) con un cargo mínimo de 3 unidades (300 caracteres) por solicitud, donde cada unidad cuesta 0.0001 USD [Amazon Comprehend, 2022].
- *Watson Natural Language Understanding* de IBM: Proporciona reconocimiento de entidades, reconocimiento de partes de un discurso, la extracción de palabras claves, análisis de emociones, análisis de sentimiento y detecta conceptos de alto nivel, a los que se refiera de forma indirecta un texto [IBM, 2021]. El plan básico cuesta 0.003 USD por elemento de NLU, donde NLU = número de unidades de datos * número de funciones utilizadas. Una unidad de datos es un grupo de 10000 caracteres como máximo.
- *SpaCy*: Brinda la posibilidad de realizar etiquetado, análisis sintáctico, reconocimiento de entidades nombradas, clasificación de texto según su función en el discurso y aprendizaje multitarea con transformadores preentrenados. Además, permite generar diagramas de análisis sintáctico. *SpaCy* es un software comercial de código abierto publicado bajo la licencia MIT por lo que no tiene costo [Honnibal y Montani, 2017].
- *NLTK*: Permite realizar clasificación de texto basándose en el análisis sintáctico y etiquetado de partes del discurso. Es un software libre y de código abierto, lo que lo hace ideal para expandir sus funcionalidades en caso de necesitarlo [Bird et al., 2009].

En resumen, se distinguen dos opciones: las de código abierto y las que usan servicios de la nube. En general, los principales beneficios de instalar un software de código abierto en lugar de usar directamente una API son el mayor rendimiento, la posibilidad de personalizar las configuraciones y hacer modificaciones y, por último, y no menos importante, su costo. La desventaja principal, comparando con un servicio en la nube, es la necesidad de conocimiento de un lenguaje de programación para poder realizar las configuraciones correctamente. Pero, en este caso no es un impedimento ya que las librerías de código abierto encontradas poseen soporte en el lenguaje de programación elegido para desarrollar la lógica de este trabajo (Python).

3.2.2. Justificación de la selección

Por lo expresado anteriormente y por ofrecer una mayor cantidad de características, se decidió utilizar *SpaCy*. *SpaCy* es un framework de código abierto para procesamiento de lenguaje natural escrito en Python. Para el idioma español hay modelos entrenados disponibles, sin embargo, una de las ideas detrás de *SpaCy* es que sus modelos no entrenados, que consisten en redes neuronales convolucionales, puedan usarse para entrenar un modelo en cualquier idioma [Spacy, 2023]. Sin embargo, en este trabajo se utilizará un modelo preentrenado, *es_core_news_sm*, se ha podido evidenciar que este modelo puede alcanzar una precisión del 98% en el reconocimiento de partes de un discurso [Company, 2023]. El *pipeline* (o camino) de procesamiento de *SpaCy* consiste en el preprocesamiento (*tokenización*), el etiquetado de POS, el análisis de dependencias, la lematización² de las palabras y el reconocimiento de entidades nombradas.

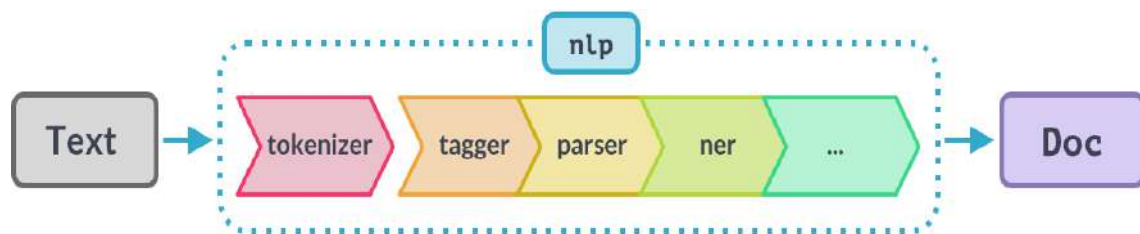


Figura 3.1 Diagrama de *pipeline* de procesamiento de *SpaCy*. Extraído de [spaCy, 2023].

Cada componente de ese *pipeline* está asociado a uno o más modelos entrenados. Por ejemplo, el componente *ner* que se puede ver en la Figura 3.1 funciona gracias a una red neuronal convolucional. El componente que interesa para poder clasificar las palabras es el *tagger*, debido a que se encarga de reconocer y etiquetar las distintas partes del texto.

SpaCy posee una *pipeline* entrenada con modelos estadísticos que permiten clasificar a qué etiqueta pertenece un *token*. Por ejemplo, una palabra que sigue a “el” o “la” en idioma castellano, tiene altas probabilidades de ser clasificada como sustantivo. Estos modelos estadísticos son entrenados usando un corpus de texto con etiquetas para aprender los patrones y las características de las diferentes partes del discurso. El modelo utiliza estos datos de entrenamiento para luego poder predecir la etiqueta de POS de cualquier palabra recibida en función del contexto en el que aparece y

²La lematización sirve para reducir variantes morfológicas de las formas de una palabra, para esto se necesita remover el plural, el tiempo, o los atributos finales de la palabra. Un ejemplo es la palabra “estudiando” luego de la lematización se transformaría en su forma base “estudiar”.

la probabilidad de que aparezcan diferentes etiquetas de POS en ese contexto. En resumen, los pasos para desarrollar un *tagger* con modelos estadísticos son:

1. Recopilar un corpus de texto previamente etiquetado y dividirlo en conjuntos de entrenamiento y testing.
2. Entrenar un modelo estadístico teniendo como base los datos de entrenamiento. Hay varias técnicas estadísticas que se pueden usar, la elección de la técnica dependerá de las características específicas del conjunto de datos y de los resultados en cuanto a rendimiento que se pretenden obtener.
3. Utilizar el modelo entrenado para predecir las etiquetas POS de las palabras del conjunto de testing.
4. Evaluar el rendimiento del modelo comparando las etiquetas predictas por el modelo con las etiquetas correctas de los datos de prueba y calculando métricas, como por ejemplo la precisión.
5. Ajustar el modelo y repetir los pasos anteriores hasta lograr el rendimiento deseado.

3.3. Computación *serverless*

Además de la herramienta de reconocimiento de discurso, fue necesario decidir respecto a las plataformas en las que se desplegará la aplicación y en la que se almacenará la información. La selección estuvo enfocada en las plataformas que proveen servicios *serverless* en la nube. El uso de servicios en la nube se ha vuelto cada vez más popular, no solo para el almacenamiento sino también para la creación de aplicaciones web. En este sentido, la computación *serverless* ha ido ganando popularidad en los últimos años. Con la computación sin servidor, los desarrolladores pueden concentrarse en escribir e implementar funciones pequeñas de un solo propósito en lugar de preocuparse por administrar servidores o infraestructura.

A continuación, se compararán los tres servicios en la nube populares a la fecha, para trabajar con computación *serverless*: Amazon Web Services (AWS), Google Cloud Platform (GCP) y Microsoft Azure.

3.3.1. Selección de herramientas *serverless*

Las siguientes son las ofertas *serverless* de los principales proveedores en la nube a la fecha:

- En la plataforma de Amazon Web Services: *AWS Lambda*, permite ejecutar código como funciones, cada función se ejecuta en un entorno aislado; *AWS Fargate*, permite ejecutar aplicaciones en contenedores en la nube y *Amazon SNS*, permite desarrollar sistemas de notificaciones.
- En la plataforma de Microsoft Azure: *Azure Container Apps* permite ejecutar aplicaciones en contenedores en la nube; *Azure Functions* permite ejecutar código como funciones, cada una en un entorno aislado y *Azure Kubernetes Service* permite crear aplicaciones sin servidor basadas en Kubernetes que tengan características de orquestación de servicios.
- En la plataforma Google Cloud Platform (GCP): *Cloud Functions* permite ejecutar código como funciones, cada una en un entorno aislado; *Cloud Run* y *App Engine* sirven para ejecutar aplicaciones en la nube. *App Engine* sirve para crear una aplicación sin servidor y es compatible con varios lenguajes de desarrollo sin necesidad de preocuparse por el soporte de la infraestructura. Su característica más importante es que simplifica la conexión a otros servicios de la plataforma, como *Cloud Storage*, *Cloud Pub/Sub* y *Cloud Firestore*, y que permite el escalamiento automático de recursos, pero no escala a cero cuando dejan de ser solicitados. A diferencia de *Google App Engine*, *Cloud Run* puede escalar contenedores sin estado de una manera rápida y aprovechar *Google Kubernetes Engine*, el escalamiento sucede de manera automática partiendo desde cero y volviendo a cero en caso de ser necesario, según las solicitudes de recursos. Por esta característica, *Cloud Run* permite trabajar con entornos de desarrollo más personalizados.

3.3.2. Justificación de la selección

En conclusión, las tres plataformas de servicios en la nube ofrecen valiosas herramientas y servicios para que los desarrolladores creen aplicaciones web. Cada plataforma tiene sus puntos fuertes y débiles, por lo que es importante prestar atención a los requisitos antes de elegir. AWS, GCP y Azure ofrecen un nivel gratuito, lo que facilita a los desarrolladores experimentar con las plataformas antes de comprometerse con un plan pago. Los tres proveedores de la nube ofrecen sólidos servicios informáticos

sin servidor, cada uno con sus propias características. Al elegir un servicio informático sin servidor, es importante tener en cuenta sus necesidades y requisitos específicos.

Debido a que las tres opciones cumplen con los requerimientos de la aplicación, se decidió utilizar *Cloud Run* de Google Cloud Platform debido a que permite el despliegue en contenedores y el primer millón de peticiones es gratis. Es por eso que, en un principio y debido a la cantidad de usuarios activos que pretende tener una aplicación de este tipo, el coste del uso de la plataforma en la nube de *Google Cloud* para el despliegue sería nulo.

3.4. Base de datos

Cuando se trata de elegir un servicio de base de datos, hay varias opciones disponibles. Un factor importante que divide a las bases de datos en dos, es su disponibilidad o no en la nube. Las bases de datos en la nube son cada vez más populares en el mundo de los negocios debido a los muchos beneficios que ofrecen. Estas son algunas de las ventajas clave de usar bases de datos en la nube:

1. Escalabilidad: Las bases de datos en la nube pueden gestionar fácilmente los cambios en el volumen de datos, lo que permite a las empresas ampliar o reducir su capacidad de almacenamiento según sea necesario. Esta escalabilidad facilita que las empresas se ajusten a las demandas cambiantes y eviten gastos innecesarios.
2. Accesibilidad: Se puede acceder a las bases de datos en la nube desde cualquier lugar con una conexión a internet, lo que permite acceder a los datos desde ubicaciones remotas. Esto puede mejorar la colaboración y la productividad, ya que los desarrolladores pueden trabajar juntos en proyectos sin estar en la misma ubicación física.
3. Confiabilidad: Las bases de datos en la nube están diseñadas para ser altamente confiables y tener disponibilidad alta ya que están preparadas para las fallas, debido a que cuentan con capacidades integradas de redundancia y solución de errores. Esto significa que las empresas pueden estar seguras de que sus datos siempre están accesibles y protegidos contra el tiempo de inactividad o la pérdida de datos.
4. Buena relación costo/beneficio: Las bases de datos en la nube eliminan la necesidad de que las empresas compren y mantengan hardware y software costosos. Además, las empresas solo pagan por el almacenamiento y los recursos que necesitan.

5. Seguridad: Los proveedores de bases de datos en la nube suelen contar con sólidas medidas de seguridad para protegerse contra las ciberamenazas, incluidos el cifrado, los cortafuegos y los controles de acceso. Esto ayuda a garantizar la confidencialidad, integridad y disponibilidad de sus datos. En este proyecto esta característica es importante debido a que se poseen datos personales de personas físicas.

En conclusión, las bases de datos en la nube ofrecen numerosos beneficios que pueden ayudar a mejorar la eficiencia, reducir costos y mejorar la seguridad.

3.4.1. Selección de herramientas para base de datos

A continuación, se describen comparaciones entre los proveedores de nube más conocidos a la fecha y sus productos de bases de datos.

- *Amazon RDS*: Servicio de base de datos relacional administrado por Amazon Web Services. Provee un servicio copias de seguridad automatizadas, múltiples zonas de disponibilidad que aseguran su funcionamiento. La desventaja es que este servicio está muy limitado en cuanto a que solo funciona con motores de base de datos específicos: PostgreSQL, MySQL, MariaDB, Oracle y SQL Server.
- *Google Cloud SQL*: Servicio de base de datos relacional administrado por Google Cloud Platform. Provee replicación automática para alta disponibilidad, integración perfecta con otros servicios de la plataforma de Google Cloud pero, al igual que Amazon RDS, está limitado a motores de base de datos específicos: MySQL, PostgreSQL y SQL Server.
- *SQL de Microsoft Azure*: Servicio de base de datos relacional administrado por Microsoft Azure. Entre sus prestaciones se encuentran las copias de seguridad automáticas, escalables y de alta disponibilidad. También se encuentra limitado a motores de base de datos específicos: Cosmos (NoSQL), MySQL, PostgreSQL y MariaDB.
- *MongoDB Atlas*: Servicio de base de datos NoSQL administrado por MongoDB. Es conocido por ser flexible, escalable y de alta disponibilidad. Su gran desventaja es que posee mayor costo en comparación con otras bases de datos en la nube.

3.4.2. Justificación de la selección

Debido a que se estipuló trabajar con información estructurada (ver Sección 4.3.2), fue necesario elegir una base de datos SQL, por lo que una no relacional, como lo es *Mongo DB Atlas*, quedó descartada.

En relación al costo de los servicios de base de datos analizados, no hay marcadas diferencias entre todos ellos. Por lo tanto, y debido a cuestiones de conveniencia, se optó por *Google Cloud SQL* ya que así, fue posible administrar las bases de datos desde la misma plataforma donde se administrarán los servidores. Como motor de base de datos se seleccionó MySQL debido a que se posee experiencia en su utilización y es un buen motor para manejar grandes cantidades de datos de manera eficiente por su estructura de base de datos relacional. Además, es escalable y esto constituye una buena característica, si se tiene en cuenta la posibilidad de que la aplicación amplíe su alcance.

3.5. Trabajos Relacionados

En [Peintner et al., 2008] los autores muestran resultados sobre la efectividad de aplicar técnicas de *machine learning* para el diagnóstico automático de ciertas enfermedades neurodegenerativas que alteran la producción del lenguaje y el discurso. Analizan audios de 3-5 minutos pertenecientes a un grupo de 9 pacientes de control y 30 diagnosticados con 3 de los subtipos de degeneración del lóbulo frontotemporal. Las grabaciones la realizan con micrófono y cámaras mientras el paciente realiza la parte I de las pruebas *Western Aphasia Battery*[Kertesz, 2007]. Transcriben estos audios y automáticamente generan modelos que predicen el diagnóstico del paciente. A partir de estos datos, extraen características del audio y de las palabras que el paciente usa, a partir de un algoritmo de reconocimiento automático del discurso (diseñado por ellos), su propia herramienta de medición de la duración del fonema y herramientas de análisis de contenido lingüístico y generan modelos que predicen el diagnóstico de pacientes. Los resultados demuestran que se pueden predecir diagnósticos más precisos y prevén futuros estudios con grabaciones de mayor calidad que mejorarán los resultados.

El trabajo que se propone en [Le et al., 2017] detecta parafasias fonémicas³ y neologismos⁴ a partir de ejemplos de discursos grabados desde *Aphasia Bank*⁵[MacWhinney et al., 2011]. Proponen un sistema de reconocimiento del discurso con modelos de lenguaje para transcribir automáticamente discursos de personas afásicas. Presentan un estudio piloto que investiga la viabilidad de la detección de parafasias fonémicas y neologismos automáticamente del habla afásica. Demuestran así, que cuando la transcripción objetivo es conocida, las parafasias fonémicas y neologismos se pueden distinguir con éxito del pronunciado de palabras. El sistema es capaz de superar la línea de base en la detección de la presencia de parafasias en los enunciados, y lograr una buena correlación en la estimación de la tasa de parafasia fonémica producida de cada locutor. Los resultados y análisis proporcionado en este trabajo ayudan a sentar las bases para el trabajo futuro, dirigido a la detección automática de parafasias.

Otros trabajos que merecen especial mención, son los llevados a cabo por la Dra. Leora Cherney, reconocida investigadora y médica clínica en el campo de la patología del habla y el lenguaje, particularmente, en el área de la rehabilitación de la afasia. Ha realizado importantes contribuciones en la evaluación y tratamiento de personas con afasia y enfocados sus investigaciones a la influencia de la lectura oral en el tratamiento de estas neurodegeneraciones ([Cherney, 1995], [Cherney, 2010]). Sus intereses se dirigen a habilidades lingüísticas específicas, habilidades cognitivo-lingüísticas y comunicación funcional. En tanto que sus investigaciones han explorado diversas técnicas de tratamiento, como la terapia del lenguaje con restricciones, el entrenamiento de guiones y las pruebas rápidas cognitivo-lingüísticas.

Además, Cherney ha llevado a cabo estudios para investigar la efectividad de intervenciones basadas en tecnología, la telepráctica y el impacto de programas intensivos de terapia en la recuperación de la afasia ([Cherney et al., 2008]). Sus trabajos han contribuido al desarrollo de protocolos clínicos y pautas para la rehabilitación de la afasia. Entre sus avances más significativos, se encuentra el método ORLA (*Oral Reading for Language in Aphasia*) [Cherney et al., 1986] que data del año 1986 y fue desarrollado para mejorar las habilidades de comprensión en la lectura. El método

³La parafasia fonémica es un tipo de afasia caracterizada por la sustitución de un fonema por otro o una palabra por otra que se parece en los fonemas que la componen.

⁴Un neologismo es, dentro de un determinado idioma, una palabra o expresión relativamente reciente y aislada que se está haciendo de uso corriente, sin llegarlo a ser del todo aún.

⁵<https://aphasia.talkbank.org/>

provee a los pacientes con práctica en lectura oral de oraciones y párrafos, ambos en conjunto con la terapia clínica. Consiste en los siguientes pasos:

1. El especialista se sienta enfrente del paciente.
2. El especialista lee en voz alta al paciente señalando cada palabra del texto base asignado. El material varía de 10 a 100 palabras (dependiendo el nivel educativo del paciente)
3. El especialista lee la oración o párrafo en voz alta junto con el paciente, señalando ambos, cada palabra a medida que se la lee. La velocidad y volumen se ajustan al paciente (entonación, etc.)
4. Para cada línea de oración, el especialista señala una palabra que el paciente debe identificar. Las palabras pueden ser *content words* o *function words*.
5. Para cada oración, el especialista señala una palabra para que el paciente lea en voz alta. De nuevo, se seleccionan tanto *content words* o *function words*.
6. La oración completa es leída de nuevo en voz alta al unísono con el paciente.

Con este método, se han realizado distintos estudios, documentados en, por ejemplo, [Cherney, 1995], donde dos pacientes con afasia de Broca participaron de un programa con el método ORLA. Dado que el programa proponía una instancia de lectura común y otra aplicando el método, las conclusiones arribadas fueron que hubo incremento en la precisión y tasa de lectura cuando se aplicó ORLA en la lectura, siguiendo los pasos antes descriptos.

En resumen, las contribuciones de Leora Cherney han avanzado en la comprensión de la afasia y han proporcionado conocimientos valiosos sobre enfoques efectivos de evaluación y tratamiento para personas con este trastorno de la comunicación.

Otro trabajo analizado fue el propuesto en [Adams et al., 2017] en el que se presenta un sistema para automáticamente detectar y clasificar categorías de errores en la producción de palabras característico de anomia del discurso conectado. Utilizan modelos estadísticos para identificar posibles errores y emplean un modelo de distancia fonológica para determinar similaridad fonológica entre el enunciado del sujeto y un conjunto plausible de palabras intencionadas. Luego, aplican técnicas de *machine learning* para determinar en cuales de las categorías cae un error de producción de la palabra. Para esto, utilizan el conjunto de datos provistos por el proyecto *AphasiaBank* y la historia del cuento infantil, Cenicienta, para identificar oraciones con instancias de los errores de interés: la parafasia fonológica o neologismo abstruso.

Estos trabajo analizados y, especialmente los de Cherney, permiten avalar el estudio del análisis del discurso de pacientes con afasia mediante la lectura en voz alta. En

tanto que difieren, en términos generales, con la propuesta del presente trabajo, en los siguientes aspectos:

- Desarrollan técnicas y algoritmos para trabajar con bases fonéticas en inglés, las pruebas están realizadas en ese idioma.
- Requieren la asistencia conjunta del paciente y su terapeuta (fonoaudiólogo u otro especialista).
- Utilizan técnicas de grabación por medio de micrófonos o vídeos que luego transcriben.
- Centran su análisis en el discurso completo y no en cada palabra.
- Trabajan los aspectos fonológicos y análisis de fonemas.

En tanto que, el presente trabajo:

- Se centra en el idioma castellano, por tanto, la cantidad de bases fonéticas es menor que las halladas para el idioma anglosajón.
- Parte de la hipótesis de que una terapia de lectura en voz alta podría ser mejor si el paciente la realiza desde la propia comodidad de su casa, sin mediar con el profesional que lo atiende, pero sí, saber de su supervisión continua.
- Utiliza una aplicación para que el paciente pueda grabar sus audios y enviarlos al profesional y éste, controlar y medir, distintas variables en relación a ese discurso. Se utilizan tecnologías de computación en la nube.
- Centra su análisis en cada palabra de cierta lectura asignada al paciente.
- No trabaja con el análisis de aspectos fonológicos y fonema debido a que, en esta instancia, no se tiene participación de profesionales en el tema.

Por el lado de las tecnologías de reconocimiento de discurso, se estudiaron trabajos relacionados en los que se planteaban alternativas al uso de servicios en la nube, es decir, el uso de algoritmos estadísticos o el entrenamiento de redes neuronales. Los algoritmos estadísticos que suelen usarse son los de GMM (mezcla gaussiana o *Gaussian Mixture*), HMM (modelos ocultos de Markov) o LDA (Análisis discriminante lineal *Linear discriminant analysis*). Por su parte, las redes neuronales que suelen usarse son RNNs (redes neuronales recurrentes) y las CNNs (redes neuronales convolucionales).

El uso de servicios que funcionan gracias a un modelo preentrenado (como los que se compararon en la Sección de 3.1), tiene la desventaja de que los conjuntos de datos con los que fueron entrenados y testeados provienen de la grabación de audios de personas que no tienen trastornos del habla, y esto puede afectar su precisión a la hora de ser probados con audios de personas que sí los tienen. En un artículo de la Universidad de

Shantou, China, [Mahmoud et al., 2023] se investigó el rendimiento de una red neuronal convolucional, de un algoritmo basado en LDA y de los servicios de reconocimiento de voz Google *Speech Recognition* y Azure *Speech Recognition*. Para el desarrollo de esa investigación se usaron datos de pacientes afásicos en idioma mandarín realizando:

- Entrenamiento LDA y CNN con audios de pacientes sin trastornos del habla y prueba de LDA, CNN con Azure *Speech Recognition* y Google *Speech Recognition*, con audios de pacientes sin trastornos del habla.
- Entrenamiento de LDA y CNN con audios de pacientes sin trastornos del habla y prueba de LDA, CNN utilizando Azure *Speech Recognition* y Google *Speech Recognition* con audios de pacientes afásicos.
- Entrenamiento de LDA y CNN con audios de pacientes afásicos y prueba de LDA, CNN con Azure *Speech Recognition* y Google *Speech Recognition* con audios de pacientes afásicos.

Esos resultados mostraron superioridad de la CNN desarrollada frente a los servicios en la nube de reconocimiento de voz en los tres escenarios, incluso, cuando para el entrenamiento se usaron audios de pacientes sin trastornos del habla. El problema es que el conjunto de datos utilizado, proveniente de 34 personas sanas y de 12 personas con afasia, se constituía por una serie de audios cuyo contenido transcripto, contiene solo 20 palabras en mandarín, por lo que es probable que esta red tenga un problema de sobreajuste. Es probable que si se cambiara el conjunto de datos de prueba, intentando que la red reconozca palabras desconocidas, se obtengan resultados con gran porcentaje de error, cosa contraria a lo que se esperaría de Azure *Speech Recognition* y Google *Speech Recognition* que están entrenados con conjuntos de datos más extensos.

Un inconveniente que hubiera presentado la elección de alguna de estas alternativas para el presente trabajo, es que no hay una base de datos de pacientes afásicos argentinos. Sin embargo, si el enfoque se dirige al idioma español sin tener en cuenta el país, se pueden encontrar datos de pacientes hispanohablantes de España y de pacientes hispanohablantes que residen en Estados Unidos en el sitio del proyecto de *Aphasia Bank*.

La poca disponibilidad de conjuntos de datos sobre pacientes latinoamericanos con APP evidencia la desigualdad a nivel de recopilación de información y la falta de representación que sufre la región respecto a regiones con otro nivel de desarrollo. Pero, es necesario señalar que, también se detectan desigualdades a la hora de realizar los diagnósticos. El problema es que, debido a los costos de los equipamientos, es común que en Argentina, y en general en toda América Latina, se cuente con equipos que no

son de última generación. En la región no se diagnostica correctamente la demencia debido a que se requiere de una gran cantidad de pruebas (en la Sección 2.5 se indican las estimaciones de la OMS para los índices de demencia en la región). Por ejemplo, uno de los signos de las enfermedades es la reducción del volumen de sustancia gris de las neuronas, característica que únicamente puede detectarse a partir de la interpretación de neuroimágenes, para lo cual se requiere mano de obra especializada además de imágenes de buena calidad. Es por esto que es difícil pensar en la recopilación de datos de calidad de pacientes afásicos y sanos en nuestra región cuando, para empezar, los casos están subdiagnosticados. Se estima que alrededor del 90 % de los pacientes que deberían recibir un diagnóstico de demencia no lo reciben por no someterse a la cantidad de pruebas necesarias.

Actualmente, se están realizando avances utilizando inteligencia artificial para poder subsanar estas desigualdades. En particular, científicos de la Universidad de California (entre ellos, un argentino) y el *Trinity College* de Irlanda desarrollaron un sistema que pretende automatizar los diagnósticos a partir de la clasificación de neuroimágenes y teniendo en cuenta el contexto regional y demográfico [Moguilner et al., 2023]. Esto permitiría abaratar los costos de la mano de obra especializada necesaria para realizar los diagnósticos y, en consecuencia, se ampliaría el número de personas que accedan a un diagnóstico correcto. En un futuro, podría ser factible la recopilación de datos necesaria para el entrenamiento de un sistema de reconocimiento de voz que se adecúe a los objetivos de este trabajo. [Leal, 2023].

Capítulo 4

Desarrollo de A-e-i!

En este capítulo se presenta la descripción del desarrollo de A-e-i!. Las secciones que lo componen tratan sobre:

- la metodología de desarrollo empleada,
- la especificación y análisis del proyecto que involucró este trabajo,
- el diseño de la aplicación,
- la descripción de la implementación en relación a la técnicas de procesamiento de lenguaje natural y transcripción de audio a texto empleadas, como también, la lógica de negocio (*backend*), e interfaz de usuario (*frontend*); y finalmente,
- el despliegue de la aplicación, detallando la puesta en producción de A-e-i! en los servidores de un proveedor de la nube.

4.1. Metodología de desarrollo

Este trabajo fue realizado en el contexto de un proyecto de investigación y, como tal, la metodología de trabajo aplicada estuvo enfocada en una primera instancia a tareas de investigación y luego, al desarrollo del caso práctico. Como metodología de investigación y, de acuerdo a [Hernández-Sampieri y Mendoza, 2020], se siguió un enfoque cualitativo guiado por áreas o temas de investigación. En este caso, el tema fue enfermedades neurodegenerativas y su impacto en la pérdida del habla.

Los estudios cualitativos pueden desarrollar preguntas de investigación más importantes y de allí derivar el camino de la investigación desarrollada. De acuerdo a la Figura 4.1, se comenzó con una idea preliminar, surgida a partir de observar un par

de años, una situación puntual en relación con el deterioro neurodegenerativo. Esto permitió delimitar y plantear un problema (expuesto en el Capítulo 1) y revisar la literatura al respecto (discutida en el Capítulo 3). A continuación, y por tratarse de un trabajo académico, se fijaron los objetivos de estudio y contribuciones, para que sea un trabajo factible a realizar y cumplir los requisitos académicos correspondientes (descritos también, en el Capítulo 1). El resto de las fases que comprende el proceso de investigación propuesto en [Hernández-Sampieri y Mendoza, 2020] involucran actividades destinadas al diseño de la investigación y al desarrollo de la solución en sí. En este sentido, el diseño de la investigación estuvo centrado en definir el propósito de la herramienta A-e-i!, para que su análisis y desarrollo tuviera límites bien definidos y, además, al análisis de las tecnologías existentes para la implementación de la misma. En conjunto, se fue definiendo la prueba de concepto que se realizaría, seleccionando la muestra a considerar (presentada en el Capítulo 5) para recolectar datos y proceder a mostrarlos.

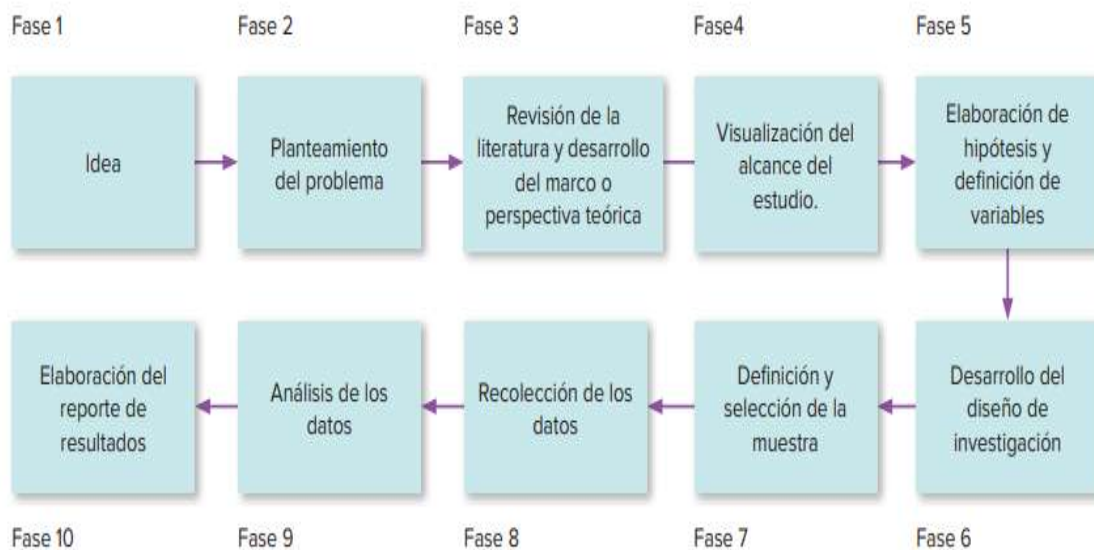


Figura 4.1 Fases para un proceso de investigación cualitativo. Extraído de [Hernández-Sampieri y Mendoza, 2020].

Para el desarrollo de la herramienta A-e-i!, que tiene su sustento teórico en la investigación realizada, se trabajó de manera iterativa e incremental, partiendo de un objetivo del sistema y un conjunto de requisitos que, a priori, son los que se definieron como parte de la visualización del alcance del estudio y la definición de variables y diseño de la investigación.

Así, se organizaron tareas y se asignó un tiempo de cuatro a ocho semanas de iteraciones. La estimación de la duración de cada tarea se puede observar en el diagrama de Gantt presentado en la Figura 4.2. Las tareas planificadas fueron las siguientes:

- Tarea 1: Estudiar el estado del arte referido a tecnologías de procesamiento y análisis del lenguaje natural en la nube y trabajos relacionados a dicho procesamiento en pacientes con APP. Tiempo estimado de dedicación: 1 mes.
- Tarea 2: Estudiar las tecnologías asociadas a *machine learning* para el reconocimiento de voz y procesamiento de lenguaje natural. Analizar plataformas en la nube y sus servicios respectivos para el despliegue de A-e-ii!. Tiempo estimado de dedicación: 2 meses.
- Tarea 3: Diseñar la arquitectura distribuida de microservicios para la aplicación propuesta. Diseñar las métricas para cuantificar distintas dimensiones relacionadas al discurso. Tiempo estimado de dedicación: 2 meses.
- Tarea 4: Implementar los microservicios para el procesamiento de lenguaje natural y reconocimiento de voz. Tiempo estimado de dedicación: 1 mes.
- Tarea 5: Implementar la aplicación web prototipo que consuma los servicios web de procesamiento de lenguaje natural y reconocimiento de voz. Tiempo estimado de dedicación: 1 mes.
- Tarea 6: Diseñar una prueba de concepto con un paciente real para demostrar la funcionalidad de la aplicación. Tiempo estimado de dedicación: 2 meses.
- Tarea 7: Escritura de la tesina. Tiempo estimado de dedicación: 3 meses.

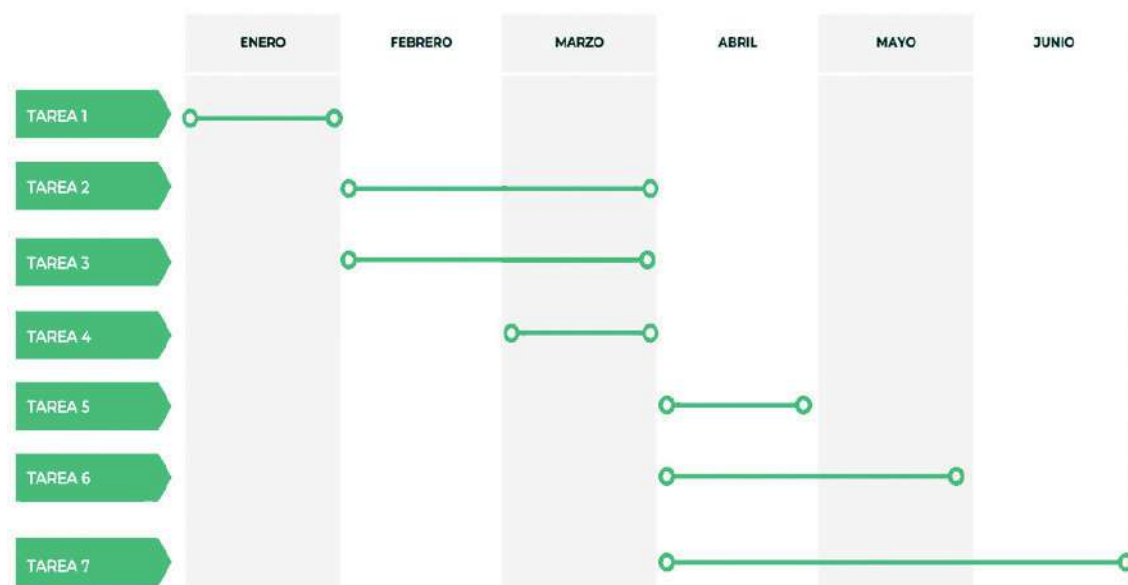


Figura 4.2 Diagrama de Gantt planificado para el presente trabajo final.

4.2. Especificación y análisis del proyecto

4.2.1. Propósito del sistema

El propósito de A-e-i! es ser una herramienta para la asistencia de profesionales fonoaudiólogos de pacientes con APP para controlar la evolución del discurso de estos pacientes. Además, se persigue que se optimicen los tiempos de los profesionales, permitiéndoles hacer controles rigurosos en cualquier momento. Si la tarea de grabar los audios, transcribirlos y analizarlos debiera realizarlos el mismo profesional en el centro médico, llevaría un tiempo considerable así como también, se vería comprometida la asistencia con asiduidad de los pacientes. No solo se incurriría en mayores costos sino que también se deberían realizar controles con, probablemente, menor frecuencia. Con esta aplicación, se permite al paciente grabar los audios desde su hogar y con la frecuencia deseada. La transcripción y el análisis del texto se realizan de forma automática y solamente es facultad del profesional acceder a los resultados generales y elaborar una conclusión a partir de ellos.

4.2.2. Especificación de requisitos del sistema

Requerimientos funcionales

Los requerimientos funcionales de un sistema están estrictamente ligados al comportamiento que debe tener el mismo al momento en que un usuario interactúa con él. Luego de realizar un relevamiento de información sobre el proceso de monitoreo de un paciente con APP y los aspectos a considerar en relación a los tratamientos para mejorar/controlar la fluidez del habla, se listaron los siguientes requerimientos:

1. El usuario profesional debe poder consultar la información de un paciente registrado en el sistema (edades, patologías diagnosticadas, etc.)
2. El usuario profesional debe poder actualizar la información de los pacientes.
3. El usuario profesional debe poder gestionar su propia información.
4. Los usuarios pacientes deben poder grabar sus audios de lecturas asignadas.
5. Los usuarios pacientes deben poder ver su historial de grabaciones.
6. El sistema permitirá la transcripción de audio a texto.
7. El sistema permitirá el análisis del texto proveniente de los audios mediante el uso de procesamiento de lenguaje natural devolviendo resultados tanto numéricos como categóricos.

8. El usuario profesional debe poder visualizar los resultados del análisis de los archivos de audio de los pacientes en formato texto.
9. El usuario profesional debe poder visualizar los resultados del análisis de los archivos de audio de los pacientes de forma gráfica. Idealmente estos gráficos deberían permitir interacciones.
10. El usuario profesional debe poder filtrar los resultados según el indicador que se quiera controlar (por ejemplo, mostrar solo la evolución de la cantidad de palabras enunciadas correctamente).
11. El sistema debe contar con un módulo de autenticación para tener el control de los usuarios y para proteger la información almacenada.

Requerimientos no funcionales

Además de los requerimientos funcionales, es necesario definir los requerimientos no funcionales del sistema. A los requerimientos no funcionales, se los define como requerimientos que no se refieren específicamente a la funcionalidad de un sistema, sino que identifican restricciones sobre el mismo. Dichas restricciones, deben ser tenidas en cuenta durante la implementación de los requerimientos funcionales, su no consideración, redunda directamente en la calidad del sistema.

1. Usabilidad: Este concepto tiene que ver con el nivel de uso o el grado en el que un sistema es fácil de utilizar. Lo ideal es que el usuario pueda interactuar con el sistema de manera simple, considerando que se trata de pacientes neurológicos, familiares a cargo y fonoaudiólogos que no debieran lidiar con un sistema complejo de operar.
2. Integridad de la información: Los datos almacenados o devueltos al usuario deben ser exactos, por lo que se debe asegurar que no se produzca su alteración ni destrucción por ningún motivo.
3. Confidencialidad: La información de los pacientes se debe mantener protegida para evitar su divulgación no autorizada.
4. Performance: Tiene que ver con el rendimiento del sistema, es decir, con todo lo relacionado a los tiempos de respuesta esperables al interactuar con el sistema.
5. Confiabilidad: Está relacionado a la cantidad de tiempo que se puede usar un sistema sin que el mismo falle.

4.2.3. Modelo de casos de uso

El modelo de casos de uso representa los requerimientos funcionales del sistema. Jacobson [Jacobson et al., 2000] propone el uso de los diagramas de casos de uso para representar, de manera gráfica, los requisitos funcionales de un sistema. Un caso de uso especifica el comportamiento de un sistema o de una parte del mismo porque describe la secuencia de acciones que debe llevar a cabo un actor para poder lograr el cumplimiento de una tarea. La Figura 4.3 ilustra el modelo de casos de uso de A-e-il.



Figura 4.3 Diagrama de casos de uso.

4.3. Diseño de A-e-i!

4.3.1. Diseño de la arquitectura

A-e-i! tiene una arquitectura distribuida en la nube de microservicios REST.

La Figura 4.4 presenta la vista de componentes y conectores de la arquitectura [Clements et al., 2003]. Cada elemento de una vista de componentes y conectores representa una pieza que tiene presencia en tiempo de ejecución, como procesos, servidores, clientes o fuentes de datos y estas piezas están conectadas entre sí por distintos tipos de interacción, como por ejemplo las interacciones de tipo invocaciones a servicios. En la arquitectura, los elementos que conforman esta vista se corresponden con los siguientes componentes:

- Componente *Frontend*: Para implementar la interfaz de usuario se utilizó Next.js¹ junto a Chakra UI². Next.js es un framework *frontend* basado en React para crear interfaces de usuario interactivas. Brinda elementos básicos para crear aplicaciones web rápidas. Next.js propone un marco de desarrollo para resolver los requisitos comunes de las aplicaciones, como el enrutamiento, la obtención de datos, las integraciones.
- Componente *Backend*: La lógica de la aplicación está diseñada como servicios web RESTful para que el cliente (navegador que despliega el *frontend*) obtenga la información necesaria para ser consultada o modificada por el usuario. Comprende la API de Autenticación, la API que gestiona la información y tareas de los doctores y la API que gestiona los audios, esto es, realiza la transcripción a texto utilizando *SpaCy* y los servicios cognitivos en la nube. Para el desarrollo de estos servicios se utilizó el módulo de Python conocido como Flask.
- Base de datos: El almacenamiento de los datos de A-e-i! se diseñó como un servicio en la nube. Por tanto, se eligió el servicio de *Cloud SQL* de *Google Cloud* y utilizando una base de datos MySQL.
- Componente de reconocimiento automático de voz: Es el encargado de recibir la transcripción del audio de una lectura, analizarlo y compararlo con un texto original. Los resultados del análisis se almacenan en la base de datos. Este componente se diseñó como un servicio en la nube, utilizando *Azure Speech Recognition* de *Microsoft Azure*.

¹<https://nextjs.org/>

²<https://chakra-ui.com/>

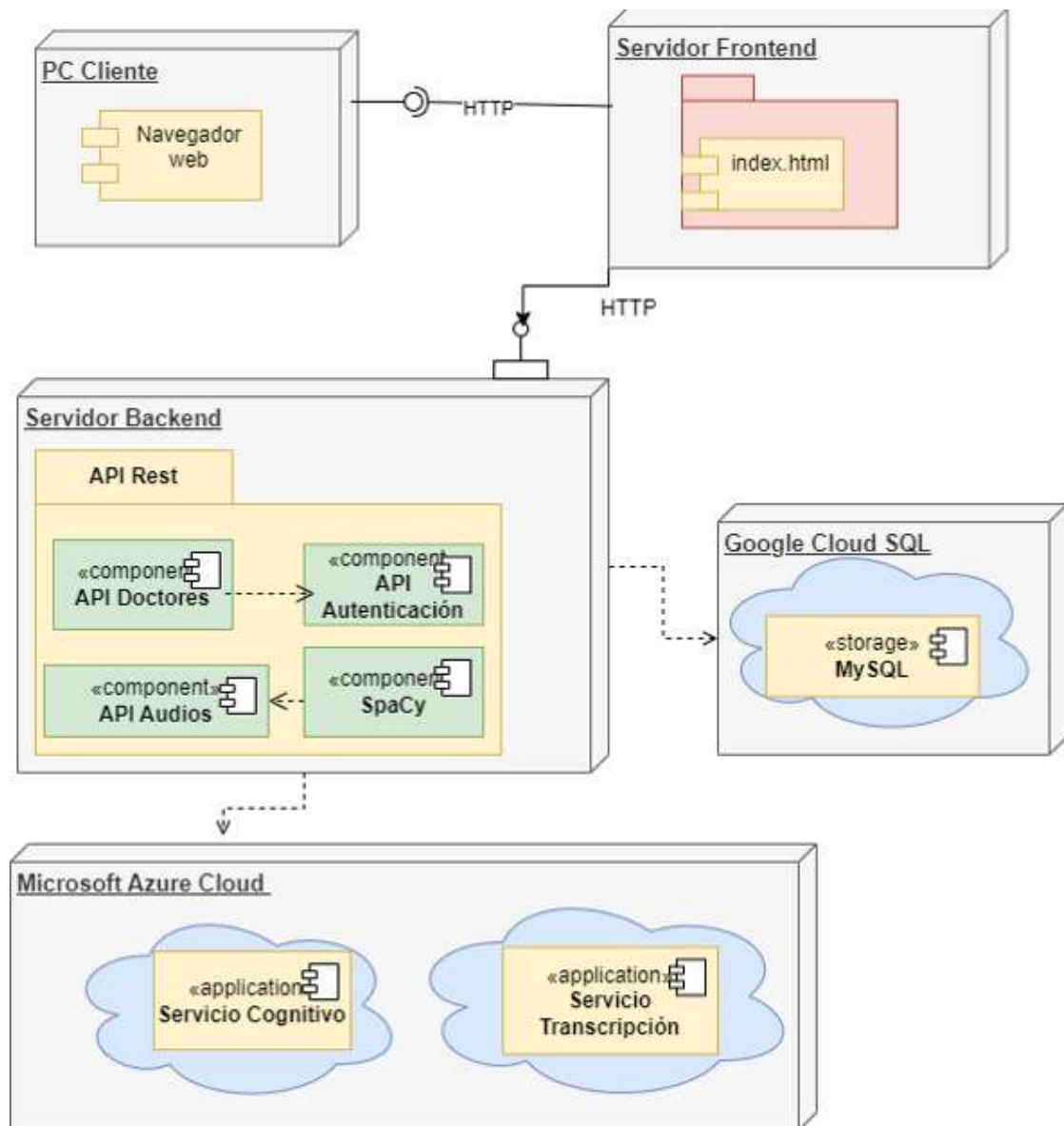


Figura 4.4 Arquitectura de A-e-i!. Vista de componentes y conector.

El usuario personal de salud interactúa con la aplicación a través del navegador web mediante solicitudes al servidor web *frontend* usando el protocolo HTTP. Este servidor web se comunica con el servidor *backend* a través del conector específico mediante HTTP, quien procesa la solicitud consultando la base de datos y devuelve al navegador la respuesta. La comunicación entre el cliente y el servidor no tiene estado y la única forma de guardar los datos intercambiados es almacenándolos en la base de datos.

La Figura 4.5 ilustra la arquitectura diseñada indicando las tecnologías de implementación utilizadas.

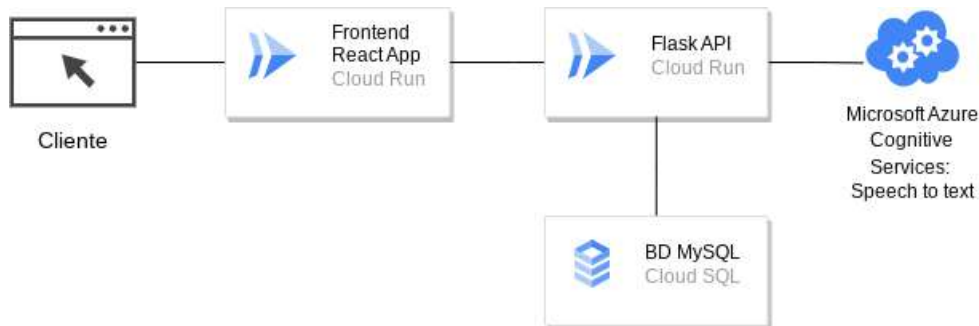


Figura 4.5 Esquema tecnológico de la arquitectura de A-e-i!.

4.3.2. Diseño de la base de datos

Para que la aplicación funcione como debe, es necesario un almacenamiento persistente de los datos, a partir de una base de datos. Para asegurar la integridad de los datos, la facilidad para acceder a los mismos y optimizar el uso de espacio es importante realizar el diseño de la base de datos utilizando un diagrama de entidad relación (DER).

La Figura 4.6 ilustra el DER diseñado contemplando las entidades principales y sus relaciones.

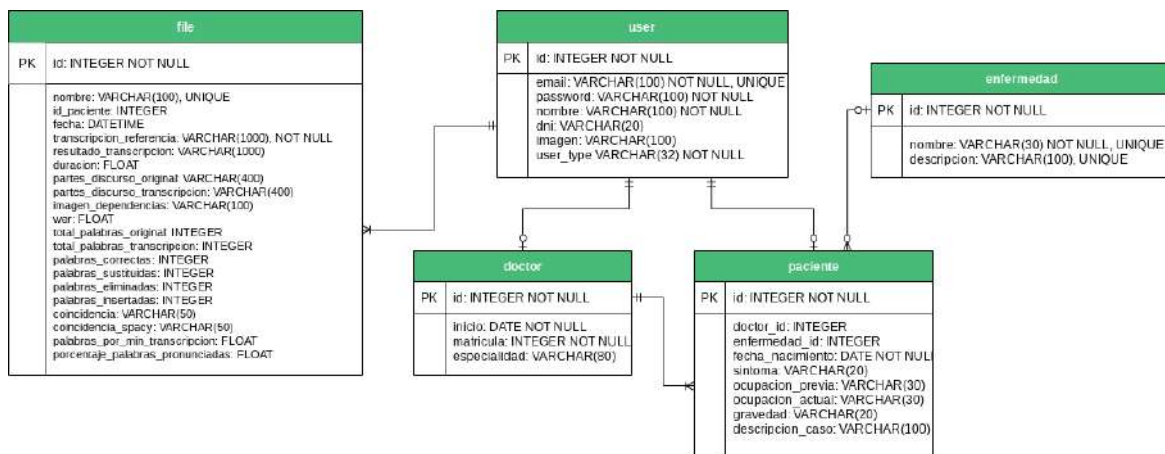


Figura 4.6 Diagrama de entidad relación de A-e-i!.

4.3.3. Diseño de la interfaz

La interfaz de usuario, o *frontend*, es la capa de presentación de la aplicación web como páginas HTML y es puesta a disposición al usuario a través del navegador web. Para obtener una aproximación más tangible de las funcionalidades del sitio, se diseñaron prototipos o *wireframes* usando la herramienta Figma³. Se logró un diseño simple teniendo en cuenta las necesidades y el nivel de experticia de cada tipo de usuario. Por otro lado, el uso de distintos tipos de visualizaciones (como por ejemplo los gráficos de barra, los gráficos lineales o los gráficos de torta) son factores determinantes que se usan para ayudar en la interpretación de los resultados.

La Figura 4.7 muestra un prototipo de la pantalla de ingreso mientras que la Figura 4.8 muestra un prototipo de la visualización de resultados. En el Anexo se puede ver el resto de prototipos diseñados.



Figura 4.7 Pantalla de inicio de sesión.

³Figma es una aplicación online para diseñar interfaces que se ejecuta en el navegador, por lo que permite el trabajo colaborativo y en tiempo real.



Figura 4.8 Pantalla de resultados.

4.3.4. Diseño de métricas

Para poder monitorear el deterioro del habla en pacientes con enfermedades neurodegenerativas, se diseñaron métricas que cuantifican distintas dimensiones del discurso hablado de un paciente. Esto es, a la transcripción realizada de la grabación del paciente, se aplicaron distintas métricas que permitieron obtener valores de interés, (o resultados) que la aplicación graficará. Posteriormente, esos valores medidos servirán para que los profesionales realicen una interpretación adecuada de esos resultados.

Este trabajo no contempla el diseño de indicadores que interpreten los valores medidos por requerir la ayuda de expertos para su realización.

A continuación se mencionan los distintos de algoritmos empleados para diseñar las métricas que permitieron calcular valores de interés del texto transcrito.

1. Las métricas diseñadas para realizar mediciones sobre el texto generado de la transcripción del paciente, aplicando como fórmula el algoritmo de *Part-Of-Speech Tagging* (4.14) de NLP, son las siguientes:

- Cantidad de adjetivos pronunciados correctos (#APC): Esta métrica mide cuántos adjetivos se pronunciaron correctamente respecto a los adjetivos del texto base de prueba.
 - Cantidad de adjetivos pronunciados (correctos o incorrectos) (#AP): Esta métrica permite calcular cuántos adjetivos se pronunciaron en total.
 - Cantidad de sustantivos pronunciados correctos (#SPC): Esta métrica permite calcular cuántos sustantivos se pronunciaron correctamente respecto a los sustantivos del texto base de prueba.
 - Cantidad de sustantivos pronunciados (correctos o incorrectos) (#SP): Esta métrica permite calcular cuántos sustantivos se pronunciaron en total.
 - Cantidad de verbos pronunciados correctos (#VPC): Esta métrica permite calcular cuántos verbos se pronunciaron correctamente respecto del total de verbos que tiene el texto base de prueba.
 - Cantidad de verbos pronunciados (correctos o incorrectos) (#VP): Esta métrica permite calcular cuántos verbos se pronunciaron en total.
2. Considerando como medida WER (*Word Error Rate*), se especificaron los siguientes cálculos:
- *Word Error Rate* (#WER): Esta métrica fue descrita en 3.1.1. La fórmula de WER se deriva de la distancia de Levenshtein, el algoritmo completo se detalla en el código C.2 del Anexo. El resultado del WER se obtiene con la fórmula 3.1.
 - Cantidad de palabras pronunciadas correctas (#PPC): Esta métrica permite calcular cuántas palabras se pronunciaron correctamente sobre el total de palabras del texto base.
 - Cantidad de palabras pronunciadas eliminadas (#PE): Esta métrica permite calcular cuántas palabras que forman parte del texto base, no fueron pronunciadas.
 - Cantidad de palabras pronunciadas insertadas (#PI): Esta métrica permite calcular cuántas palabras fueron insertadas, es decir, cuántas palabras que no forman parte del texto base, fueron pronunciadas adicionalmente por el paciente.
 - Cantidad de palabras pronunciadas sustituidas (#PS): Esta métrica permite calcular cuántas palabras fueron sustituidas, es decir, cuántas palabras que no forman parte del texto base, fueron pronunciadas en lugar de las palabras correctas.

3. Para contabilizar el total de palabras del texto base y las del texto transcripto, y generar medidas directas e indirectas (como el porcentaje de palabras pronunciadas), se usa un algoritmo que convierte cada texto en una lista de objetos de tipo *string* para luego utilizar el método *len()* de Python. El método *len()* devuelve la longitud de cada lista, lo que equivale a la cantidad de palabras de cada texto. Con este algoritmo, se diseñaron las siguientes métricas:
 - Cantidad de palabras del texto base (#P): Esta métrica calcula la cantidad de palabras que forman parte del texto de referencia.
 - Cantidad de palabras del texto de la transcripción (#T): Esta métrica calcula la cantidad de palabras que forman parte del texto leído por el paciente, luego de realizar la transcripción del audio.
 - Porcentaje de palabras pronunciadas (#PvsT): Relación entre el total de palabras de la transcripción (#T) y el total de palabras del texto base (#P).
 - Porcentaje de coincidencia (#PC): Es la relación entre el total de palabras pronunciadas correctas (#PPC) y el total de palabras del texto base (#P).
 - Velocidad del discurso (#VD): Es la medida del número de palabras pronunciadas en un período de tiempo determinado. Es la relación entre las palabras del texto (#P) y la duración del audio (#TDA).
4. Utilizando los algoritmos de cálculo de similitud con *SpaCy*, se diseñaron las siguientes métricas:
 - Porcentaje de coincidencia según *SpaCy* (#CsS): En *SpaCy* se pueden comparar dos objetos y predecir qué tan similares son, esto se logra debido a que cada palabra tiene una representación vectorial.
5. Finalmente, se consideró la duración del audio que se genera a partir de la lectura de un paciente. Esta duración se obtiene accediendo a la propiedad *duration_seconds* del objeto *AudioSegment* generado durante la ejecución del algoritmo 4.9.
 - Duración del audio (#TDA): Es la duración en segundos de cada audio. Como cada paciente tiene un texto asignado, esta métrica mide el tiempo que se tarda en leerlo en voz alta. Permitirá, a posterior, analizar la fluidez de la lectura.

4.4. Implementación de A-e-i!

4.4.1. Implementación de la lógica de negocio

Para la implementación de la lógica de negocio se eligió el framework Flask, módulo de Python que permite desarrollar aplicaciones fácilmente.

Flask no incluye un ORM (*Object-Relational Mapping*) ni otras funcionalidades como el enrutamiento o un motor de *templates*. Tampoco incluye una capa de abstracción de base de datos, ni validación de formularios o cualquier otra funcionalidad para la que ya existan diferentes librerías que puedan proveerlas. Por este motivo, Flask admite extensiones para agregar funcionalidades extras a las aplicaciones, a partir del uso de esas librerías. Existe un gran número de extensiones compatibles con Flask que brindan integración de bases de datos, validación de formularios, balanceo de carga, autenticación mediante distintos métodos, etc.

Si bien Flask es "micro", tiene la capacidad de extenderse para cubrir distintos requerimientos. Además, otra razón por la cual se escogió este microframework es que, en este caso solamente se necesitaba desarrollar la API REST y no la aplicación web completa, por lo que no fue necesario utilizar un framework tan robusto y poderoso, como por ejemplo podría ser Django. Flask, al igual que otros frameworks de desarrollo, está basado en el patrón de Modelo-Vista-Controlador (MVC), lo cual significa que se puede estructurar una aplicación de acuerdo al nivel que define ese patrón arquitectónico y aprovechar sus ventajas.

Instalación de Flask y configuración de la aplicación

Flask funciona con el lenguaje Python así que se debe chequear que este lenguaje esté instalado en la máquina, en una versión que no sea más antigua que la versión 3.7. Esto se realiza escribiendo en una terminal de comandos, el siguiente comando `python3` y chequeando que se inicie un interpretador del lenguaje.

Otro requisito es tener un motor de base de datos previamente instalado y configurado. Por defecto, el motor que se instala y configura con Flask es SQLite.

Para la instalación se debe usar el paquete de instaladores pip3⁴. El comando que debe utilizarse es `pip3 install Flask`.

Si el proceso de instalación finaliza sin error, se puede verificar la versión del framework que si acaba de instalar a partir comando `flask._version_`.

Para crear la aplicación, primero se importa la clase Flask. Una instancia de esta clase será la aplicación que se está creando. El punto de entrada de la aplicación se encontrará en un archivo llamado *app.py*.

A continuación, se crea una instancia de esta clase Flask. Como se puede observar en la línea 2 del Código 4.1, el primer argumento de esta clase, es el nombre del módulo o paquete de la aplicación. En este caso se usará `__name__`, que es una variable predefinida de Python que toma el nombre de la carpeta en donde se sitúa el archivo. Se habilita CORS⁵ para que la API solo acepte solicitudes de la interfaz de React.

```
1 def create_app():
2     app = Flask(__name__)
3     CORS(app)
4
5     db.init_app(app)
6
7     with app.app_context():
8         db.create_all()
9     return app
10 if __name__ == '__main__':
11     application = create_app()
```

Código 4.1 Creación de una aplicación en Flask

En la aplicación habrá un único punto de entrada, por lo tanto, se definieron todas las rutas que existirán para que cada petición pueda ejecutar el servicio que le corresponde. Las rutas se agruparon en las siguientes categorías:

- **Autenticación:** Agrupa las rutas de inicio de los servicios de inicio de sesión, cierre de sesión, registro de profesionales y pacientes y también, de obtención de la información del usuario autenticado en determinado momento.

⁴pip es un sistema de gestión de paquetes utilizado para instalar y administrar paquetes de software escritos en Python.

⁵CORS (*Cross-Origin Resource Sharing*) es un mecanismo o política de seguridad que permite controlar que las peticiones HTTP se pueden realizar desde un navegador a un servidor con un dominio diferente de la página cargada originalmente.

- Profesionales: Agrupa las rutas de los servicios que pueden acceder solamente los profesionales, esto es: listar pacientes del profesional, obtener la historia clínica de un paciente determinado, agregar patologías y listar patologías cargadas en el sistema.
- Archivos: Agrupa las rutas que mapean con los servicios para subir y analizar un archivo de audio y calcular los valores de todas las métricas. Además, obtener el listado de los archivos subidos y aquellos analizados para un paciente en particular.

El registro de estas rutas se realiza usando objetos *Blueprint*, este tipo de objeto sirve para registrar las operaciones a ejecutar dentro de una aplicación. Permite definir la colección de vistas para que la estructura de la aplicación sea modular y se pueda acceder a distintas funcionalidades a través de distintas URLs.

Este registro se realiza de la siguiente manera. En el Código 4.2 se muestra cómo deben registrarse las rutas utilizando el objeto *Blueprint*, definido en la línea número 1, y el decorador *route*. Las líneas 5, 8, 12, 16, 19 y 23 del mismo código muestran la definición de las URL.

```
1 autenticacion = Blueprint('autenticacion', __name__,
2 url_prefix="/autenticacion")
3
4 @autenticacion.errorhandler(404)
5 @autenticacion.route('/login', methods=['POST'])
6 def login_post():
7
8 @autenticacion.route('/registro', methods=['DELETE'])
9 @jwt_required()
10 def registro_delete():
11
12 @autenticacion.route('/registro', methods=['PUT'])
13 @jwt_required()
14 def registro_put():
15
16 @autenticacion.route('/registro', methods=['POST'])
17 def registro_post():
18
19 @autenticacion.route('/logout', methods=['GET'])
20 @jwt_required()
21 def logout():
```

```

22
23 @autenticacion.route("/usuario_autenticado", methods=["GET"])
24 @jwt_required()
25 def usuario_autenticado():

```

Código 4.2 Registro de rutas de vista de autenticación.

Por último, hay que registrar cada *Blueprint* definido en la aplicación para que esas rutas definidas estén disponibles para su acceso. El registro se hace en el módulo `__init__.py`, en las líneas 2 y 3 del código mostrado a continuación. Se puede ver cómo se importa el objeto *Blueprint* del módulo de autenticación que se inicializa en el código 4.3 y se registra el mismo en la aplicación.

```

1     # blueprint para rutas de autenticación
2     from views.autenticacion import autenticacion as autenticacion_blueprint
3     app.register_blueprint(autenticacion_blueprint)
4
5     # blueprint para rutas de profesionales
6     from .views.doctores import doctores as doctores_blueprint
7     app.register_blueprint(doctores_blueprint)
8
9     # blueprint para rutas de archivos
10    from .views.files import files as files_blueprint
11    app.register_blueprint(main_blueprint)

```

Código 4.3 Registro de rutas.

Para los servicios de autenticación, se utilizó `JWTManager` que gestiona los *tokens*⁶ de inicio de sesión. El fragmento de código 4.4, indica que se configuró un tiempo de validez de 1 hora para cada *token*.

```

1     app.config['SECRET_KEY'] = os.urandom(10)
2     app.config["JWT_ACCESS_TOKEN_EXPIRES"] = timedelta(hours=1)
3     login_manager = JWTManager(app)
4     login_manager.init_app(app)

```

Código 4.4 Configuración de `JWTManager`.

`JWTManager` provee los siguientes métodos:

⁶Un token es una cadena de texto encriptada con una clave que contiene información de un inicio de sesión para evitar que en todas las solicitudes se envíe el usuario y la contraseña.

- `create_access_token()` para crear cada *token*. El código 4.5 muestra la creación de un *token*.

```

1  @autenticacion.route('/login', methods=['POST'])
2  def login_post():
3      email = request.form.get('email')
4      psw = request.form.get('password')
5      user = User.query.filter_by(email=email).first()
6      if not user or not check_password_hash(user.password, psw):
7          return "Por favor revise su email o su contraseña.",
8              404
9
10     secret_key = app.config[SECRET_KEY]
11     access_token = create_access_token(identity=user.id)
12     session['user_id'] = user.id
13
14     return {"token": access_token, "user": user.to_dict() }
15

```

Código 4.5 Ejemplo de uso de `create_access_token()`.

- `jwt_required()` para proteger las rutas en las que se requiere autenticación.
- `get_jwt_identity()` para descryptar el *token* y obtener información del usuario que lo posee.

Por último, en la Figura 4.9 se muestra un ejemplo de una petición para registrar un usuario de tipo paciente, realizada a la API desde Postman.

Integración con el servicio de *Speech-to-text*

Como servicio de *Speech-to-text* se seleccionó Microsoft Azure Cognitive Services. Para lograr su integración con la API REST desarrollada en Flask, se cumplió con una serie de pasos previos:

1. Crear una suscripción en Azure.
2. Crear un recurso de voz en Azure Portal⁷.
3. Obtener la clave y la región del recurso de voz asignado. Deben configurarse como variables de entorno bajo los nombres `SPEECH_KEY` y `SPEECH_REGION`.

⁷Un recurso de Azure es un elemento administrable que está disponible para su uso en la plataforma de la Nube de Azure.

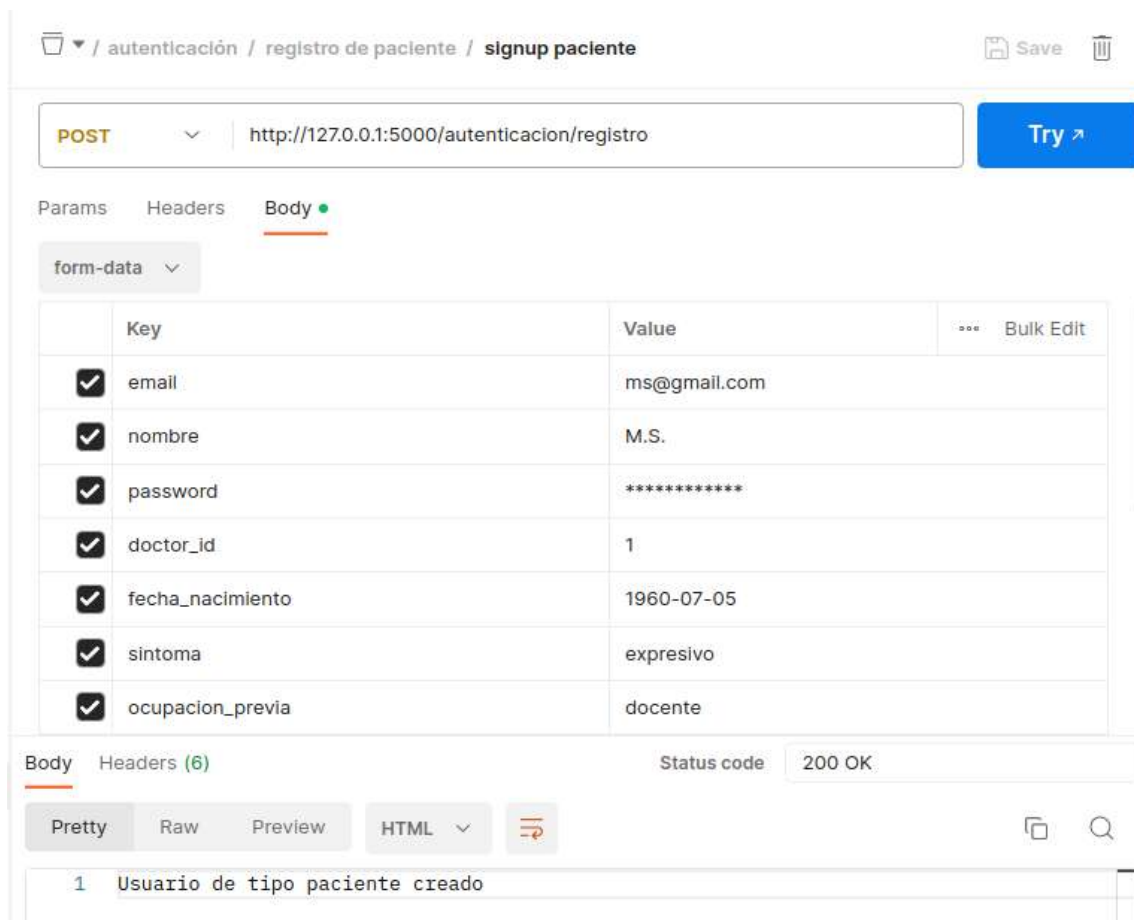


Figura 4.9 Petición realizada para registrar un paciente.

4. Instalar *Speech SDK*. Para instalar *Speech SDK* en Linux se debe ejecutar el comando `pip install azure-cognitiveservices-speech`.
5. Implementar el algoritmo de transcripción.

Para implementar el algoritmo de transcripción de audio, se utilizó el SDK de voz de Azure para lenguaje Python. Para poder acceder a los servicios de voz que se proveen desde la plataforma de Azure, se debe tener el *token* de acceso (`SPEECH_KEY`) que se obtuvo en paso 4) descrito anteriormente. Esto permite crear una instancia de una configuración de voz o *SpeechConfig* con una clave de suscripción, una región de servicio y un idioma específico. El fragmento de código 4.6 indica dichas configuraciones:

```

1  speech_key = os.getenv('SPEECH_KEY')
2  service_region = os.getenv('SPEECH_REGION')
3  languageCode = 'es-AR'
4  speech_config = speechsdk.SpeechConfig(subscription=speech_key,
5  region=service_region)

```

```
6 speech_config.speech_recognition_language = languageCode
```

Código 4.6 Creación de la instancia de configuración de voz.

Luego, para poder analizar un audio en particular, se creó una configuración de audio o *AudioConfig*. El audio será recibido como parámetro por la función de transcripción. Luego, esta instancia de configuración puede conectarse a un reconocedor de discurso o *SpeechRecognizer* que es quien inicia el proceso de transcripción, de la siguiente manera:

```
1 audio_input = speechsdk.AudioConfig(filename=audio_filename)
2 speech_recognizer = speechsdk.SpeechRecognizer(speech_config=speech_config,
3 audio_config=audio_input)
```

Código 4.7 Configuración de audio.

Una vez creada la instancia de *SpeechRecognizer* se debe configurar, o bien el reconocimiento de una sola captura o el reconocimiento continuo. El primero, reconoce una expresión única que se determina mediante la escucha de un silencio que marca un final o hasta que se procesa un máximo de 15 segundos de audio. En tanto que, para audios de más de 15 segundos, se debe usar el reconocimiento continuo, dado que permite establecer cuándo detener el reconocimiento. El reconocimiento continuo se ajusta mejor a las necesidades de la aplicación porque la duración de los audios será mayor a 15 segundos en todos los casos. Para marcar el fin del reconocimiento, se creó una variable booleana, denominada "fin", e inicializada en *false* y, otra variable de tipo lista, para almacenar los resultados parciales de la transcripción. También se conectó una función *callback*, denominada *stop*, encargada de detener el reconocimiento continuo al recibir un evento de cancelación o detenimiento. El código 4.8 muestra las sentencias codificadas para el algoritmo de reconocimiento del discurso.

```
1 fin = False
2 transcripcion = []
3
4 def stop(evt):
5     speech_recognizer.stop_continuous_recognition()
6     nonlocal done
7     done = True
8
9 # Conectar callbacks a los eventos disparados
10 speech_recognizer.recognized.connect
11 (lambda evt: full_transcription.append(evt.result.text.lower()))
12
```

```

13 speech_recognizer.session_stopped.connect(stop_cb)
14
15 speech_recognizer.canceled.connect(stop_cb)
16
17 speech_recognizer.start_continuous_recognition()
18
19 while not done: time.sleep(.5)
20 if done:
21     full_transcription = " ".join(full_transcription)
22 return full_transcription

```

Código 4.8 Algoritmo de *Speech Recognition*.

Previo a ejecutarse dicho algoritmo de reconocimiento de discurso, se debe realizar una conversión a formato .wav para asegurar la compatibilidad con los formatos soportados por el servicio de Azure. De esta manera, se captura el formato de audio del archivo recibido y, si no es de formato .wav, se lo convierte a un archivo con dicha extensión, tal como se describe en las siguientes líneas del código 4.9:

```

1 from pydub import AudioSegment
2 def transformar_formato(filename, format):
3     format=f"{filename.rsplit('.', 1)[1].lower()}"
4     if format != 'wav':
5         wav_audio=AudioSegment.from_file({UPLOAD_FOLDER}+{filename})
6
7         new_filename={UPLOAD_FOLDER}+{filename}.replace('m4a', 'wav')
8         wav_audio.export(new_filename, format='wav')
9         return {'wav_audio': wav_audio, 'new_filename': new_filename}
10 return filename

```

Código 4.9 Algoritmo de transformación de formato de audio.

Otro algoritmo que se desarrolló es el que permite reducir el ruido. El motor de reconocimiento de discurso tiene que distinguir entre el ruido del ambiente en que se graba cada audio y la voz del paciente, lo que le dificulta su tarea de identificar palabras.

```

1 import numpy as np
2 import librosa
3 from scipy.io import wavfile
4 import noisereducer as nr

```

```

5 def reducir_ruido(filename):
6     rate, data = wavfile.read(filename)
7     audio_modificado = nr.reduce_noise(y=data, sr=rate)
8     wavfile.write(filename, rate, audio_modificado)

```

Código 4.10 Algoritmo de reducción de ruido.

Por último, se mejoró la precisión del reconocimiento a partir de la clase *PhraseListGrammar* que permite agregar nuevas frases al reconocedor. Una lista de frases es una lista de palabras o frases de las que se quiere ayudar a mejorar su reconocimiento ya que, luego de ser añadidas al reconocedor de discurso, aumenta la probabilidad de que éste las identifique [Maheshwari Utkarsh, 2023]. Lo conveniente es agregar las palabras que es muy probable que sean pronunciadas. En este caso se eligieron 20 palabras, que coinciden con las 20 palabras más frecuentes del texto de referencia (en el Capítulo 5 se describe el caso de estudio y se especifica el texto de referencia utilizado), palabras que, con anticipación, se sabe que pueden llegar a ser pronunciadas en el audio. Para obtener las palabras más frecuentes en el texto de referencia se utilizó *SpaCy* para omitir los símbolos de puntuación, los espacios en blanco y las *stop words*⁸. Luego se utilizó la colección *Counter* para obtener la cantidad de apariciones de cada palabra y ordenar la lista de forma descendente.

```

1 from collections import Counter
2 def palabras_mas_comunes(texto):
3     nlp = spacy.load('es_core_news_sm')
4     doc = nlp(texto)
5
6     palabras_filtradas = [token.text
7         for token in doc
8         if not token.is_stop and not token.is_punct
9         and not token.is_space]
10
11     frecuencia_palabras = Counter(palabras_filtradas)
12     palabras_mas_comunes = frecuencia_palabras.most_common(20)
13
14     return palabras_mas_comunes

```

⁸Las *stop words* son las palabras más comunes de un idioma que no aportan información sustantiva sin su contexto, por ejemplo, “y” o “yo”. Por lo general, se eliminan del texto para reducir el tiempo de procesamiento.

15

Código 4.11 Algoritmo de obtención de palabras más comunes en un texto.

El código 4.12 muestra cómo agregar las palabras a la lista de frases de la instancia configurada en 4.7.

```

1 phrase_list_grammar = speechsdk.PhraseListGrammar.
2   from_recognizer(speech_recognizer)
3 lista_palabras_comunes = palabras_mas_comunes(texto_original)
4 for palabra in lista_palabras_comunes:
5     phrase_list_grammar.addPhrase(palabra)

```

Código 4.12 Algoritmo para agregar una lista de palabras al reconocedor de discurso.

Servicio de procesamiento del lenguaje natural

Para realizar el procesamiento de la transcripción obtenida se usó reconocimiento de partes de un discurso con *SpaCy*, tal cual se describió en la Sección 3.2.

Para instalar la biblioteca *SpaCy* se debe ejecutar el comando `pip install spacy`. También es necesario descargar el modelo *es_core_news_sm*. En general, la convención de nomenclatura que usa *SpaCy* es la siguiente:

[lenguaje]__[tipo]__[género]__[tamaño], donde:

- Lenguaje: Identifica el idioma en el que fue entrenado el modelo, en este caso “*es*” es la codificación del idioma español.
- Tipo: Son las capacidades del modelo. Por ejemplo, un modelo de tipo *core* cumple tareas de propósito general, como el etiquetado, la lematización y el reconocimiento de entidades nombradas.
- Género: Identifica el tipo de texto en el que se entrenó el modelo. Un ejemplo es el entrenamiento usando noticias (*news*) como fuente.
- Tamaño: Es el indicador de tamaño del modelo. Puede tomar el valor de *sm* (pequeño), *md* (mediano), *lg* (grande) o *trf* (transformer).

Para poder descargar el modelo se debe ejecutar en la consola el comando: `python -m spacy download es_core_news_sm`.

Si se ejecuta sin errores, se continúa con la implementación del algoritmo de reconocimiento de entidades. Para esto, se carga el modelo nombrado, utilizando la función *load*. Esta función devuelve un objeto de tipo *Language* que contiene todos

los componentes y datos del idioma necesarios para procesar el texto, en este caso el objeto será denominado *nlp*. En el fragmento de código 4.13 se observa cómo se debe instanciar este objeto.

```
1 import spacy
2 nlp = spacy.load('es_core_news_sm')
```

Código 4.13 Carga de modelo de idioma.

Luego, se realiza un preprocesamiento de la entrada de texto para eliminar las mayúsculas, usando la función *lower()*. Para procesar la entrada de texto proveniente de la transcripción se debe obtener un objeto *Doc*, que es una secuencia de objetos *token*. Luego, el resultado llamar al objeto *nlp* con la cadena de texto como parámetro devolverá el *Doc* procesado que se necesita. Cada uno de esos objetos *token* tiene información sobre sí mismo. Entre los atributos que se pueden consultar se encuentran:

- Text: El texto de la palabra original.
- Lemma: La forma base de la palabra.
- POS: La etiqueta, según la categorización de *Universal POS Tag*.
- Dep: Dependencia sintáctica, es decir, la relación entre *tokens*.
- Forma (*shape*): La forma de la palabra, esto es: mayúsculas, puntuación, dígitos.
- is_alpha: Booleano que especifica si un *token* es un carácter alfa o no.
- is_stop: Booleano que especifica si un *token* es parte de la lista de las palabras más comunes del idioma.

A continuación, la Tabla 4.1 ejemplifica los atributos anteriores, como resultado de analizar la oración “Es un día soleado”.

Tabla 4.1 Etiquetas de tipo *Parts of Speech*.

TEXT	LEMMA	POS	DEP	SHAPE	ALPHA	STOP
Es	ser	AUX	cop	xx	True	True
Un	uno	DET	det	xx	True	True
Día	día	NOUN	ROOT	xxx	True	True
Soleado	soleado	ADJ	amod	xxxx	True	False
S.	.	PUNCT	punct	.	False	False

La etiqueta que más interesa para lograr analizar el contenido sintáctico del texto es la denominada POS. La etiqueta POS (*Part Of Speech* o Parte del discurso) explica cómo se usa una palabra en particular en una oración. Hay ocho posibilidades:

1. Sustantivo (NOUN)

2. Pronombre (PRON)
3. Adjetivo (ADJ)
4. Verbo (VERB)
5. Adverbio (ADV)
6. Preposición (ADP)
7. Conjunción (CCONJ, SCONJ)
8. Interjección (INTJ)
9. Determinante (DET)

En el código 4.14, en las líneas 18 y 19, se puede visualizar cómo, por cada *token*, se obtiene su etiqueta POS.

Por otro lado, *SpaCy* incluye un visualizador de dependencias (o etiquetas *dep*) llamado *displaCy*. *DisplaCy* recopila las dependencias entre las frases de una oración para determinar su estructura gramatical. Para hacer esto, una oración se divide en secciones o unidades lingüísticas, en general se dividen en palabras pero también pueden dividirse en frases. El proceso se basa en la suposición de que existe una relación directa o dependencia entre cada una de estas secciones. Luego, el visualizador genera archivos de tipo .svg que muestran las dependencias sintácticas entre los distintos *tokens* que pueden estar contenidos en un objeto *Doc*.

En un gráfico de dependencias sintácticas, las relaciones entre las unidades lingüísticas se representan usando arcos dirigidos. De acuerdo al ejemplo anterior, el gráfico de dependencias generado es el que se ilustra en la Figura 4.8. Una etiqueta de dependencia indica la relación entre dos frases. Por ejemplo, la palabra “soleado” cambia el significado del sustantivo “día”. Como resultado, en la Figura 4.10 se puede identificar una dependencia entre día y soleado (**día -> soleado**). Esta dependencia tiene asociada la etiqueta *amod*, que significa “adjetivo modificador directo”. Actualmente, el modelo *es_core_news_sm* usa la taxonomía UD Spanish AnCora v2.8 Delor et al. [2008] que consta de 41 relaciones sintácticas comunes, como se muestra en la Tabla B.1 del Anexo.

En la línea 12 del fragmento de código 4.14 se puede observar cómo se genera el gráfico de dependencias sintácticas.

```
1 import spacy
2 from spacy import displacy
3 from pathlib import Path
4
5
```

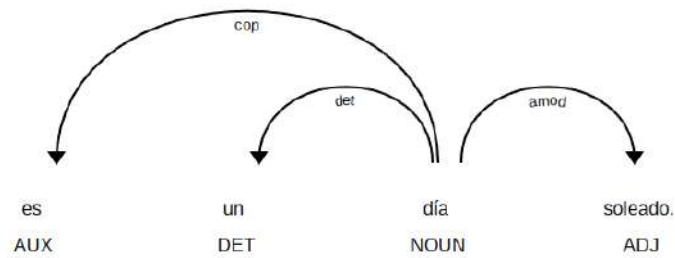


Figura 4.10 Ejemplo de etiquetado POS para la oración 'Es un día soleado'

```

6 def etiquetado_pos(text, filename):
7     nlp = spacy.load('es_core_news_sm')
8
9     doc = nlp(text.lower())
10    result = []
11
12    svg = displacy.render(doc, style="dep")
13
14    output_path = Path(f"./uploads/{filename}.svg")
15    # Crear una imagen de las dependencias entre palabras
16    output_path.open("w", encoding="utf-8").write(svg)
17
18    for token in doc:
19        result.append({"palabra": token.text, "pos": token.pos_})
20    return result

```

Código 4.14 Código del algoritmo de *Part-Of-Speech Tagging*.

Implementación de la base de datos

La base de datos se implementó siguiendo el modelo relacional detallado en la Sección 4.3.2 bajo un servicio en la nube. Como proveedor de la plataforma de la nube se optó por Google Cloud y su servicio *Cloud SQL*. Para esto, se creó una cuenta en la plataforma Google Cloud, obteniendo un crédito de 300 dólares como cliente nuevo.

Luego se creó un nuevo proyecto de Google Cloud y se instaló Google Cloud CLI localmente, la instalación depende del sistema operativo. A continuación, se inició el cliente mediante el comando `gcloud init`.

Como se utilizó una base de datos de MySQL, se creó una instancia de este motor dentro del panel de instancias de Cloud SQL. En este punto, se tiene que establecer la contraseña del usuario *root*, elegir una región y el tipo de disponibilidad para la instancia. La región debe coincidir con la región desde la cual se va a hacer acceso a la información.



Figura 4.11 Configuración de la base de datos.

Las opciones iniciales de cómputo de la Figura 4.11 pueden ser modificadas dinámicamente para atender a las necesidades de la aplicación. Una vez creada la instancia, se puede visualizar su IP pública, la que se deberá copiar para poder hacer la conexión con el cliente de Google Cloud Platform previamente instalado. En la Figura 4.12 se observa la IP y el nombre de la conexión.



Figura 4.12 Dirección de IP pública de la base de datos.

Para importar la base de datos definida se debe crear un *script sql*. El archivo se denominó *init.sql* y su contenido está en el código C.1 del Anexo, debe ser subido a Google Cloud Storage para luego ser importado en Cloud SQL.

Una vez hecho esto, se debe ejecutar este comando para la creación de la base de datos: `mysql -h 34.70.78.86 -u root -p < init.sql`.

Una vez creada la base, se puede acceder de forma remota a través de la misma dirección IP.

Dado que la base de datos almacenará datos de pacientes y su evolución clínica referida a su afectación neurodegenerativa, se tuvo en cuenta la protección de datos personales de acuerdo a la regulación establecida por la Constitución Nacional de la República Argentina, en la Ley 25.326 [Boletín Oficial de la República Argentina, 2000]. En la misma, se garantiza que todas las personas puedan controlar su información personal asentada en bases de datos públicas o privadas. Como se planea usar servicios de cómputo en la nube, se necesita pensar en un tratamiento de datos personales por terceros (las empresas prestadores de los servicios) por lo que se debe seguir lo estipulado en el artículo 25 de la mencionada Ley y el artículo 25 del Decreto 1558/01 [Boletín Oficial de la República Argentina, 2001] que la reglamenta. Esto impacta al desarrollo de la aplicación ya que se deberán almacenar datos personales de los pacientes y de los médicos. Esto implica que se debe pensar en la seguridad de los datos trabajados y la posibilidad de brindarlos al usuario cuando los requiera.

4.4.2. Implementación de la interfaz de usuario

Para implementar la interfaz de usuario se utilizó el framework Next.js junto a Chakra UI.

El uso de un framework *frontend* no es obligatorio para el desarrollo de una aplicación debido a que es posible construir una interfaz de usuario simple con solo tres archivos (HTML, CSS y JavaScript) pero, la utilización de alguno, presenta algunas ventajas importantes. En la Tabla 4.2 se destacan las ventajas de utilizar el *framework* Next.js comparándolo con el desarrollo de *frontend* solo con HTML, CSS y JavaScript.

Tabla 4.2 Ventajas del Framework Next.js sobre el uso de HTML, CSS y JavaScript.

	HTML, CSS y Javascript	Next.js
Reutilización	No.	Sí, basado en componentes.
Escalabilidad	La cantidad de archivos crece junto con la aplicación y el código.	Solo es necesario crear nuevos componentes o reutilizar componentes ya creados.
Mantenimiento	Mientras más código tenga la aplicación es más complicado.	Permite la reutilización y la codificación modular, lo que lo facilita.
Estructura del código	No especificada.	Estandarizada.
Comunidad activa	No.	Si.

Antes de instalar Next.js, se instaló en la máquina local el motor de Node js y el manejador de paquetes de npm. Para verificar si Node js está instalado se usa el comando `node -v`. Para verificar si npm está instalado se usa el comando `npm -v`.

A continuación, para crear la aplicación de Next.js se ejecutó el comando `npx create-next-app@latest nextjs-app`. Para iniciar el servidor de desarrollo se utiliza el comando `npm run dev`.

Como se describió al inicio de la sección, se utilizó además, Chakra UI, que es una biblioteca de componentes simple, modular y accesible que brinda los componentes básicos que se necesitan para crear una aplicación basada en React. Tiene la capacidad de adaptar la interfaz del sitio web al tamaño del dispositivo en que se visualice, o sea, sigue los lineamientos del “*responsive design*” o diseño adaptativo.

Para instalar Chakra UI se invoca el comando `npm i @chakra-ui/react @emotion/react @emotion/styled framer-motion`.

Después de la instalación, se debe configurar `ChakraProvider` en la raíz del proyecto para que los componentes estén disponibles en todos los archivos de la aplicación. En el caso de Next.js, la raíz del proyecto se encuentra en `_app.js`. En el código 4.15 se puede ver esta configuración.

```

1 function AEI({ Component, pageProps }) {
2
3   return (

```

```

4   <AuthProvider>
5     <AudioProvider>
6       <PacientesProvider>
7         <ChakraProvider theme={theme} cssVarsRoot="body" >
8           <Component {...pageProps} />
9         </ChakraProvider>
10        </PacientesProvider>
11        </AudioProvider>
12      </AuthProvider>
13    )
14  }
15
16  export default MyApp;

```

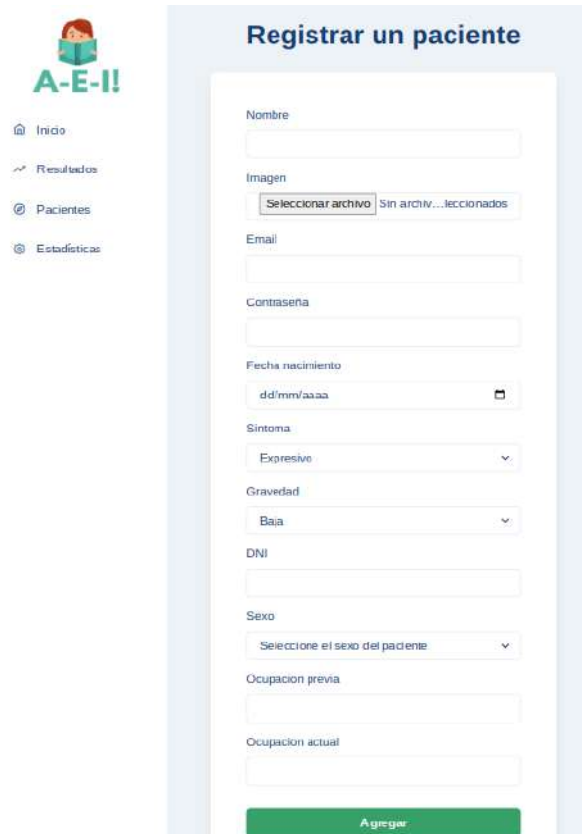
Código 4.15 Raíz del proyecto de Next.js

Una vez hecha la conexión entre la interfaz de usuario y la API REST desarrollada, se puede realizar un registro de un usuario como el que se observaba en la Figura 4.9 pero sin necesidad de usar Postman. Se usará la interfaz de registro de pacientes implementada.

Para acceder al formulario de registro, primero se debe acceder a la sección “Pacientes”, tal como ilustra la Figura 4.13.



Figura 4.13 Interfaz gráfica para gestión de pacientes.



The image shows a web interface for registering a patient. On the left is a sidebar with the 'A-E-I!' logo and navigation links: Inicio, Resultados, Pacientes, and Estadísticas. The main area is titled 'Registrar un paciente' and contains a form with the following fields: 'Nombre' (text input), 'Imagen' (file upload button labeled 'Seleccionar archivo'), 'Email' (text input), 'Contraseña' (password input), 'Fecha nacimiento' (date picker showing 'dd/mm/yyyy'), 'Síntoma' (dropdown menu with 'Expresivo' selected), 'Gravedad' (dropdown menu with 'Baja' selected), 'DNI' (text input), 'Sexo' (dropdown menu with 'Seleccione el sexo del paciente'), 'Ocupación previa' (text input), and 'Ocupación actual' (text input). A green 'Agregar' button is located at the bottom of the form.

Figura 4.14 Interfaz gráfica para el registro de pacientes.

4.5. Despliegue de A-e-i!

El despliegue en un entorno productivo de la aplicación A-e-i! se realizó en la plataforma de Google Cloud. Para ello, se utilizó la herramienta *serverless* que provee dicho proveedor de la nube, denominada *Google Cloud Run*. Este, es un servicio del tipo *Container as a Service*, se le llama así porque el usuario no tiene que desplegar ningún orquestador de contenedores.

4.5.1. Despliegue del servidor del *Backend*

Para desplegar el servicio web con flask en Cloud Run se debe crear un archivo `Dockerfile`⁹. Se comienza creando y configurando la región y zona horaria de la imagen

⁹Un archivo `Dockerfile` es una serie de instrucciones necesarias para crear la imagen de un contenedor.

nueva a partir usando como sistema operativo *ubuntu:20.04*, tal como se puede observar en el código 4.16.

```
1 FROM ubuntu:20.04
2 WORKDIR /
3
4 ENV TZ=America/Argentina
5 RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ >
6 /etc/timezone
```

Código 4.16 Creación de archivo Dockerfile para el despliegue del servidor *backend*

Luego, se realizan actualizaciones y se instalan los paquetes y librerías de Python necesarios para el funcionamiento de la aplicación que el contenedor debe tener. Las librerías que necesita la aplicación se encuentran listadas en un archivo denominado *requirements.txt*. Las actualizaciones e instalaciones empiezan en la primera línea y terminan en la decimosexta línea. En la última línea se especifica a través de la instrucción CMD (comando) el comando que inicia la ejecución de la aplicación en el contenedor.

```
1 RUN apt-get update
2 RUN apt-get install -y python3-pip
3 RUN pip install --upgrade pip
4 RUN apt-get install -y -q \
5     build-essential \
6     curl
7 RUN apt update &&
8 apt install gcc &&
9 apt install ffmpeg -y &&
10 apt install g++ &&
11 apt-get update -y &&
12 apt-get install -y &&
13 --no-install-recommends build-essential gcc \
14 libsndfile1
15 COPY . /
16 RUN pip install -r ./api/requirements.txt
17
18 CMD ["python3", "main.py"]
```

Código 4.17 Creación de archivo Dockerfile: Instalación de paquetes y ejecución

Con este archivo Dockerfile se puede construir una nueva imagen usando el comando `docker build . -t aeiaapp:latest`.

Luego, se debe ejecutar la imagen docker en Cloud Run. Para esto, se debe cargar la imagen en la plataforma Google Cloud Registry, el registro de imágenes de Google Cloud Platform. Como se ve en el fragmento de código 4.18, se coloca una etiqueta a la imagen previo a su subida a la plataforma.

```
1 docker tag <ID_IMAGEN> gcr.io/ID_PROYECTO/aeiflask:v.0.1
2 docker push <ID_IMAGEN> gcr.io/ID_PROYECTO/aeiflask:v.0.1
```

Código 4.18 Carga de imagen en Google Cloud Registry

Para publicar la imagen es necesario dirigirse a la consola de Cloud Run y crear un nuevo servicio. Se listan una serie de opciones de las cuales se eligió tener un servicio completamente administrado. La región del servicio se configura por defecto en *“us-central1 (Iowa)”*. Al nombre de servicio se le colocó *“aeiflask”* y respecto a la autenticación, se permitirán invocaciones sin autenticar debido a que esta restricción de seguridad ya está controlada en la interacción entre la interfaz de usuario y el servicio web. La Figura 4.15 muestra la pantalla de configuración del servicio de Cloud Run.

Por defecto, el puerto del contenedor que toma Google es el puerto 8080 pero esto puede ser modificado. Como se puede ver en la configuración avanzada del servicio que se muestra en la Figura 4.16, siempre habrá una instancia disponible y, si la cantidad máxima de solicitudes simultáneas excede las 80, se creará una segunda instancia. La cantidad máxima de solicitudes simultáneas por instancia también puede modificarse.

» Cloud Run « Crear servicio

Cada servicio expone un extremo único y ajusta de forma automática la escala de la infraestructura subyacente para controlar las solicitudes entrantes. No se puede cambiar el nombre del servicio ni la región más adelante.

Implementar una revisión desde una imagen de contenedor

URL de la imagen del contenedor [SELECCIONAR](#)

REALIZAR PRUEBAS CON UN CONTENEDOR DE MUESTRA
 Debe detectar las solicitudes HTTP en \$PORT y no depender del estado local. [¿Cómo se compila un contenedor?](#) [?](#)

Implementar de forma continua revisiones nuevas desde un repositorio de código fuente

Nombre del servicio *

Región * [¿Cómo se selecciona la región?](#) [?](#)

Precios y asignación de CPU [?](#)

La CPU solo se asigna durante el procesamiento de la solicitud
 Se te cobra por solicitud y solo cuando la instancia de contenedor procesa una solicitud.

La CPU siempre está asignada
 Se te cobra por todo el ciclo de vida de la instancia de contenedor.

Figura 4.15 Creación del servicio de *Cloud Run* para el servicio web de Flask.

Contenedores, herramientas de redes, seguridad [^](#)

[CONTENEDOR](#) [REDES](#) [SEGURIDAD](#)

General

Puerto de contenedor
 Las solicitudes se enviarán al contenedor de este puerto. Recomendamos detectar en \$PORT, en lugar de en este número específico.

Comando de contenedor
 Deja el campo en blanco para usar el comando de punto de entrada definido en la imagen de contenedor.

Argumentos de contenedor
 Argumentos pasados al comando del punto de entrada.

Aumento de CPU de inicio
 Asigna más capacidad de CPU durante el tiempo de inicio para iniciar los contenedores más rápido. [Más información](#) [?](#)

Capacidad

Memoria CPU
 Es la memoria para asignar a cada instancia de este contenedor. Es la cantidad de CPU virtuales asignadas a cada instancia de este contenedor.

Tiempo de espera de la solicitud seconds
 Tiempo en el que se daba mostrar una respuesta (máximo de 3600 segundos).

Cantidad máxima de solicitudes simultáneas por instancia

Figura 4.16 Opciones de configuración del servicio de *Cloud Run*.

Una vez creado el servicio, se otorga una URL que sirve para poder hacer solicitudes (<https://aeiflask-gwrvk2kx4a-uc.a.run.app>). Para el siguiente paso, se introduce brevemente a la librería Axios que fue utilizada en el desarrollo del *frontend* de la aplicación. Axios es una librería de JavaScript que permite hacer peticiones al contenido de un enlace HTTP.

Al obtener la dirección URL del servicio, se debe ingresar al código de la aplicación de React y modificar la configuración de Axios, ya que hasta este momento la URL del *backend* era la del servidor de desarrollo del entorno local (<http://localhost:5000>). En las líneas 3 y 4 del fragmento de código 4.19 se puede ver este cambio.

```
1 import axios from 'axios';
2 const axiosClient = axios.create({
3   #baseUrl: "http://localhost:5000",
4   baseUrl: "https://aeiflask-gwrvk2kx4a-uc.a.run.app",
5   headers: {
6     'Content-type': 'application/json',
7   }
8 });
```

Código 4.19 Configuración de Axios

4.5.2. Despliegue del servidor del *Frontend*

Para desplegar la aplicación de React en Cloud Run deben seguirse pasos similares a los del despliegue del servicio del *backend* de Flask, esto es:

1. Confeccionar un archivo Dockerfile.
2. Subir la imagen de Docker a Google Container Registry.
3. Configurar el servicio en Cloud Run.

Antes de comenzar con el primer ítem, que consiste en confeccionar un archivo Dockerfile, se procede a definir un archivo `.dockerignore`. El código 4.20 refleja su contenido.

En un proyecto React se generan las carpetas `node_modules` y `build` que contienen las dependencias del proyecto y los resultados de compilar el código respectivamente, por lo que su tamaño puede llegar a ser considerable y afectar la velocidad de la subida. Además, la información de las versiones de los paquetes, para volver a ser instalados, se encuentran en `package.json` y `package-lock.json`. El directorio de `build` se genera cada

vez que se compila el código, por lo que su inclusión no solo agrega sobrecarga, sino que directamente es innecesaria. Por eso se crea un archivo *.dockerignore* para evitar que estas carpetas entorpezcan la construcción del contenedor, todo lo que figura en este archivo es ignorado por el *daemon* de Docker. Por razones similares a la de la presencia de la carpeta *build*, se agrega a esta lista el archivo *Dockerfile* de la imagen, ya que la misma será construida a partir de este archivo, pero no necesita ser subido. También se agrega el propio archivo *.dockerignore* ya que su contenido resulta irrelevante en el proceso.

```
1 node_modules
2 build
3 .dockerignore
4 Dockerfile
```

Código 4.20 Archivo *.dockerignore*

Siguiendo los pasos enumerados anteriormente, se comienza a confeccionar el archivo *Dockerfile* importando una imagen base *node:alpine*¹⁰ que servirá para compilar la aplicación e instalar todas las dependencias, como podemos ver en el código 4.21. Luego se crea un ambiente de producción con una imagen *nginx:alpine* que será el sistema operativo en el que se importará el código ya compilado y donde sólo configurará lo necesario para el funcionamiento en producción. Al separar los entornos de compilación y de producción se construyen imágenes más eficientes, lo que conduce a implementaciones más rápidas.

```
1 FROM node:alpine as build
2
3 WORKDIR /app
4
5 ENV PATH /app/node_modules/.bin:$PATH
6 COPY package.json ./
7 COPY package-lock.json ./
8 RUN npm ci --silent
9 RUN npm install react-scripts@3.4.1 -g --silent
10
11 COPY . ./
12
```

¹⁰Alpine es una distribución de Linux.

```
13 RUN npm run build
```

Código 4.21 Configuración de Dockerfile para despliegue de *frontend*: Entorno de compilación

En la segunda etapa, una vez compilado el código, se puede copiar el contenido del proyecto desde el directorio *build*. Las líneas 2 y 3 del código 4.22 permiten que se ejecuten esas acciones. La línea 4 indica que se debe exponer el puerto 80 del ambiente de producción para permitir ese acceso.

```
1 FROM nginx:stable-alpine
2 COPY --from=build /app/build /usr/share/nginx/html
3 COPY nginx/nginx.conf /etc/nginx/conf.d/default.conf
4 EXPOSE 80
5 CMD ["nginx", "-g", "daemon off;"]
```

Código 4.22 Configuración de Dockerfile para despliegue de *frontend*: Entorno de producción

Luego de crear este archivo, se puede proceder a construir la imagen del contenedor Docker ejecutando el comando `docker build -t aei-react`.

El segundo paso enumerado es el de subir la imagen a Google Cloud Registry, para ello se etiqueta la imagen y luego se inicia su subida a la plataforma, de manera análoga a lo que se hizo en el despliegue del servidor del *backend*. El siguiente código muestra cómo se realizan estas acciones.

```
1 docker tag <ID_IMAGEN> gcr.io/ID_PROYECTO/aeireact:v.0.1
2 docker push <ID_IMAGEN> gcr.io/ID_PROYECTO/aeireact:v.0.1
```

Código 4.23 Carga de imagen en Google Cloud Registry

Una vez publicada la imagen, se puede acceder a la consola de Cloud Run para crear el servicio de la misma manera que se creó para el servicio web de Flask, se puede ver en la Figura 4.17 que la configuración es análoga.

Cuando el servicio es creado y configurado, se obtiene una URL que permite que se pueda acceder a la aplicación desde cualquier navegador.

Google Cloud gc-function-models cloud run

Cloud Run ← Crear servicio

Cada servicio expone un extremo único y ajusta de forma automática la escala de la infraestructura subyacente para controlar las solicitudes entrantes. No se puede cambiar el nombre del servicio ni la región más adelante.

Implementar una revisión desde una imagen de contenedor

URL de la imagen del contenedor [SELECCIONAR](#)

[REALIZAR PRUEBAS CON UN CONTENEDOR DE MUESTRA](#)
Debe detectar las solicitudes HTTP en SPOR y no depender del estado local. [¿Cómo se configura un contenedor? ↗](#)

Implementar de forma continua revisiones nuevas desde un repositorio de código fuente

Nombre del servicio *

Región * [¿Cómo se selecciona la región? ↗](#)

Precios y asignación de CPU

La CPU solo se asigna durante el procesamiento de la solicitud
Se te cobra por solicitud y solo cuando la instancia de contenedor procesa una solicitud.

La CPU siempre está asignada
Se te cobra por todo el ciclo de vida de la instancia de contenedor.

Ajuste de escala automático

Número mínimo de instancias * Número máximo de instancias

Configúralo en 1 para reducir los inicios en frío. [Más información ↗](#)

Pricing summary

Precios de Cloud Run

Nivel gratuito

Primeras 180,000 unidades de CPU virtuales segundo por mes

Primeros 360,000 GiB segundo por mes

2 millones de solicitudes por mes

→ [Verificar los detalles de los niveles pagados](#)

Se habilitó la API de Cloud Run Admin ×

Figura 4.17 Configuración del servicio de Cloud Run para la aplicación de React.

Capítulo 5

Caso de estudio

En este capítulo se describe el caso de estudio que sirvió para poner en práctica la aplicación desarrollada y comprobar la validez de la solución planteada.

5.1. Introducción

El caso de estudio consistió en trabajar con una paciente real con deterioro neurodegenerativo. Para contextualizar a la misma, sin entrar en términos clínicos, se detalla se trata de una mujer de 72 años, jubilada docente que hace unos años comenzó con problemas para hablar. Básicamente, su pensamiento iba más rápido que sus palabras. Luego de varios estudios, determinaron que estaba transitando APP y, las imágenes de tomografía computada en el cerebro daban cuenta de una atrofia frontal-izquierda.

Dado su interés incesante por la lectura y, siendo ésta una práctica sugerida para mejorar las habilidades lingüísticas, le indicaron que practique lecturas en voz alta. Si bien al inicio (hace 4 años) la paciente podía leer un libro entero en voz alta, la realidad es que la afasia, como su nombre lo indica, fue progresivamente avanzando, al punto que, con el correr del tiempo, apenas pudo llegar a leer algunos párrafos. Ante esta situación, los profesionales no tenían a disposición algo que le permitiera medir ese retroceso, más que la información aportada por los familiares que, claramente, era subjetiva ("lee más/lee menos").

5.2. Realización de pruebas

Hay dos tipos de síntoma en lo que a trastornos del lenguaje se refiere: los síntomas receptivos y los expresivos. Los síntomas receptivos tienen que ver con la capacidad del paciente de comprender el habla y los gestos, mientras que los síntomas expresivos tienen que ver con la capacidad en cuanto a vocabulario, es decir, en cuanto a desarrollar oraciones con vocabulario o estructuras complejas y recordar palabras

La prueba realizada con A-e-i! se desarrolló de la siguiente manera. Primero, se escogió un párrafo de un texto que era familiar para la paciente. Dado su avanzado deterioro neurocognitivo, el texto se seleccionó prestando atención al hecho de que sea de lectura fácil y con una historia conocida¹. Así se escogieron los dos primeros párrafos de "La Hormiguita viajera" de Constancio C. Vigil, libro que es posesión de la paciente desde su niñez.

Luego, se determinó la periodicidad de la prueba. Se estableció una frecuencia de dos días a la semana, a partir de diciembre 2022, siempre considerando primero, el estado emocional de la paciente, para no perturbarla. Por esta razón, la prueba no se ha realizado de manera regular. Y, en los meses de marzo-abril, no ha podido realizarse completamente.

El texto que se utilizó para que la paciente leyera es el siguiente: "*Esta hormiguita vive en un hormiguero que hay en el campo y tiene la entrada junto a una piedra grande. Todos los días sale en busca de lo que pueda ser útil, pues es una hormiguita exploradora. Si encuentra algo que merezca aprovecharse, vuelve inmediatamente al hormiguero para dar la noticia. Entonces salen y son guiadas por ella las obreras que trabajan en dividir y transportar los materiales. Cierta día encontró nuestra hormiguita una servilleta. Después de olerla, palparla y morderla, se dijo que era una hoja tan grande como no había otra, y tan seca y desabrida como una piedra. Pero encima de la servilleta descubrió una montaña de olor apetitoso y de riquísimo gusto.- ¡Qué bueno es esto! -exclamó emocionada.*"

Un familiar acompañando a la paciente, grabó cada una de sus sesiones de lectura de ese texto de referencia. Cada audio se subió a la aplicación, donde se calculan todos los valores de las métricas descritas en la sección anterior. La Figura 5.1 retrata un momento de dicha lectura.

¹Aquí la importancia de que el profesional, en conjunto con un familiar, evalúen el tipo de texto a asignar para la lectura. Depende el grado de avance de la enfermedad, el nivel intelectual previo del paciente, entre otros factores.



Figura 5.1 Lectura: "La hormiguita viajera".

La paciente tomó como hábito, la lectura de los mismos párrafos durante tres meses. Durante ese tiempo, su enfermedad mostró una marcada aceleración de síntomas, con predominio a mayores latencias, tanto en su desempeño diario como en la fluidez del habla y lectura. Si bien la profesional fonoaudióloga que la atiende, estuvo al tanto de estas pruebas, no participó como usuaria del sistema por estar el mismo en continuo desarrollo al mismo tiempo que las pruebas que con la paciente se realizaban.

Las pruebas se realizaron los siguientes días:

- 14-12-2022
- 21-12-2022
- 23-12-2022
- 31-12-2022
- 03-01-2023
- 06-02-2023
- 15-02-2023
- 22-02-2023
- 04-03-2023
- 25-05-2023

con lo cual, conforman el presente caso de estudio, un total de 10 audios que han sido transcritos y, el texto generado, analizado con el servicio de procesamiento de lenguaje natural descrito en la sección anterior. Además, a dicho texto, se le calcularon las métricas diseñadas, cuyos valores obtenidos, fueron luego graficados por la herramienta desarrollada. Estos gráficos no están a disposición de los pacientes, sino de los profesionales médicos que atiendan a los mismos y sean usuarios del sistema.

5.3. Generación de resultados

A continuación, se detallan los valores obtenidos para un audio cualquiera. En los días seleccionados para la prueba, se grabó solo un audio. En este caso, los valores medidos se muestran en la Figura 5.2 y corresponden al 6 de Febrero del 2023.

- Fecha: 6 de Febrero del 2023
- Transcripción: *“esta hormiguita vive en un hormiguero grande que hay y es el campo y que tiene la entrada junto a una piedra que es grande todos los días en buca de lo que pueda ser pues eh es una hormiguita y lo eh la dora si encuentra algo que me dejo queda a aprovechar pero pues ya se vuelve al hormiguero y pero para dar la noticia y entonces si y eso y por ella las obreras que trabajan en dividire y los materiales y el cierto dia o una servilleta era grande no no dejo desde la palparla y morderla se dijo que era un un hoja tan grande como no habia otra y esa vida como una piedra pero encima de la servilleta era una montana manana de color apetitoso y ni ni quisimos oh que bueno es esto exclamo”*

Transcripción 6 del paciente M.S. día 06 de Febrero del 2023	
Métrica	Medida
CANTIDAD DE ADJETIVOS PRONUNCIADOS (CORRECTOS O INCORRECTOS) (#AP)	14
CANTIDAD DE ADJETIVOS PRONUNCIADOS CORRECTOS (#APC)	10
CANTIDAD DE PALABRAS DEL TEXTO BASE (#P)	122
CANTIDAD DE PALABRAS DEL TEXTO DE LA TRANSCRIPCIÓN (#T)	143
CANTIDAD DE PALABRAS PRONUNCIADAS CORRECTAS (#PPC)	85
CANTIDAD DE PALABRAS PRONUNCIADAS ELIMINADAS (#PE)	4
CANTIDAD DE PALABRAS PRONUNCIADAS INSERTADAS (#PI)	25
CANTIDAD DE PALABRAS PRONUNCIADAS SUSTITUIDAS (#PS)	33
CANTIDAD DE SUSTANTIVOS PRONUNCIADOS (CORRECTOS O INCORRECTOS) (#SP)	18
CANTIDAD DE SUSTANTIVOS PRONUNCIADOS CORRECTOS (#SPC)	15
CANTIDAD DE VERBOS PRONUNCIADOS (CORRECTOS O INCORRECTOS) (#VP)	11
CANTIDAD DE VERBOS PRONUNCIADOS CORRECTOS (#VPC)	9
DURACIÓN DEL AUDIO (#TDA)	6.066
PORCENTAJE DE COINCIDENCIA (#PC)	69.672
PORCENTAJE DE COINCIDENCIA SEGÚN SPACY (#CSS)	87.165
PORCENTAJE DE PALABRAS PRONUNCIADAS (#PVST)	1.172
VELOCIDAD DEL DISCURSO (#VD)	23.575
WORD ERROR RATE (#WER)	0.508

Figura 5.2 Ejemplo de medidas obtenidas para un audio de una paciente. Extraído de la aplicación.

La Figura 5.3 muestra el gráfico de los valores medidos para la métrica de velocidad del discurso (#VD). En el eje x se encuentra los días de realizada la prueba y, en el eje y, los valores numéricos de los resultados que representan porcentajes. De igual manera, se pueden seleccionar otras métricas y ver la evolución en el tiempo de los valores medidos. En tanto que, en la Figura 5.4 se ilustra un gráfico de torta que es la información detallada para una prueba en concreta, es decir, seleccionando un punto del gráfico xey, el gráfico de torta se actualiza con los valores de las métricas del día que corresponde a la muestra seleccionada.

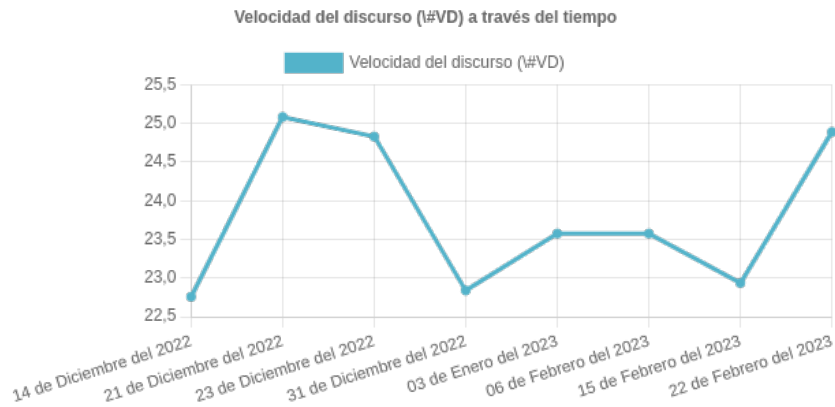


Figura 5.3 Valor de la métrica de velocidad del discurso en las grabaciones realizadas desde el 14 de Diciembre de 2022 al 22 de Febrero de 2023.

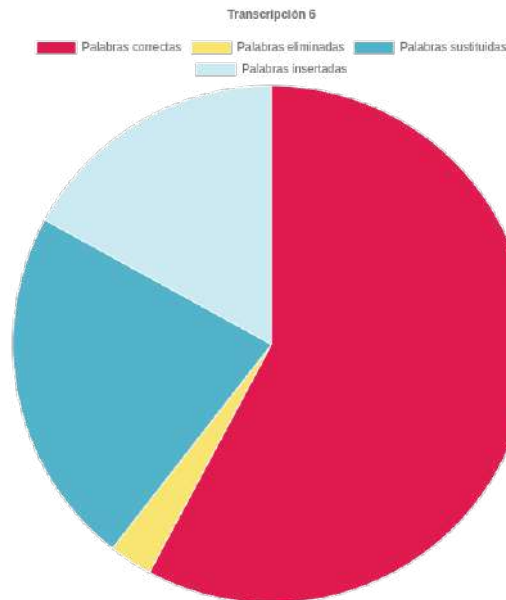


Figura 5.4 Valores medidos para cada categoría de palabra (correctas, eliminadas, sustituidas, insertadas) para la grabación del día 6 de Febrero del 2023.

Como los gráficos son interactivos, se pueden seleccionar una o más métricas para analizarlas por separado, como se puede observar en la Figura 5.5.

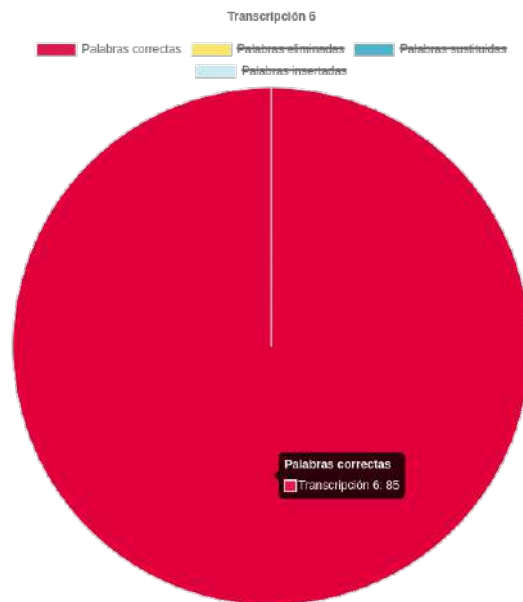


Figura 5.5 Valores medidos para la categoría de palabra “correctas” para la grabación del día 6 de Febrero del 2023.

Capítulo 6

Conclusiones y Trabajos Futuros

Este capítulo se estructura en los siguientes contenidos:

- Las conclusiones referidas a la contribución del trabajo, como así también, la experiencia personal de realizarlo.
- Los avances que se desprenden de esta investigación y los trabajos futuros que se persiguen.

6.1. Conclusiones finales

Este trabajo presentó el desarrollo de una aplicación que procesa audios de lecturas de pacientes con enfermedades neurodegenerativas (en esta instancia, aquellos con APP), transcribe y analiza dichos audios y calcula valores de interés que ayudan a profesionales que atienden a estos pacientes, a monitorear la evolución del deterioro del habla. La aplicación entonces, pretende ser una herramienta de asistencia a los profesionales, para que puedan controlar la terapia de lectura en voz alta de sus pacientes con APP y así, abordar conclusiones que sirvan, a futuro, para mejorar las prácticas a cada paciente y profundizar los estudios sobre cómo evoluciona el deterioro del habla.

Para esto, se utilizaron servicios en la nube para: el procesamiento del lenguaje natural, el almacenamiento persistente de los datos que administra y, el despliegue de la misma en servidores de producción. Por lo tanto, fue necesario estudiar técnicas de procesamiento de lenguaje natural y de reconocimiento de voz para incluir los

algoritmos necesarios que transcriban y procesen el lenguaje natural de los audios de las lecturas de los pacientes.

La elaboración de este trabajo contó con los siguientes puntos a destacar:

- Se analizaron y compararon varios servicios de distintos proveedores de computación en la nube que brindan servicios de *speech-to-text* o reconocimiento de discurso. A partir de esto, se observaron limitaciones a nivel cultural y lingüístico para el idioma castellano, porque si bien hay modelos disponibles para el dialecto argentino, dentro de nuestro país se puede reconocer la presencia de distintos dialectos según la región.
- Se analizaron distintos servicios de proveedores de computación en la nube que ofrecen técnicas de procesamiento de lenguaje natural. Sin embargo, se eligió la librería *open source SpaCy*, concluyendo que, cuando un proyecto tiene una comunidad fuerte, puede estar al mismo nivel que las herramientas de pago.
- Se completó el desarrollo de una aplicación que, si bien presenta cuestiones a mejorar, constituye una práctica avanzada de la utilización e integración de los conocimientos y técnicas de desarrollo de software adquiridas a lo largo de la carrera en varias asignaturas.

Este trabajo constituye un avance inicial, pero ha pretendido resaltar que la importancia de la terapia de leer en voz alta no alcanza si no se mantiene un seguimiento por parte de los profesionales, del discurso de sus pacientes, que progresivamente se va perdiendo.

Como conclusión personal, destaco que la realización de esta tesis me permitió adquirir nuevos conocimientos y reforzar otros. Por un lado, estudié sobre tecnologías no abordadas en la carrera, como reconocimiento de discurso y procesamiento de lenguaje natural, en particular, la técnica de reconocimiento de partes de un texto. Y su utilización en distintos servicios, como por ejemplo, los que ofrecen los proveedores de la nube.

Además, consolidé mis conocimientos sobre contenedores y realicé mi primer despliegue de una aplicación usando esa tecnología. También, este trabajo me abrió conocimiento sobre un tema relacionado a la salud sobre el que tenía escaso conocimiento. Así, estudiar sobre enfermedades neurodegenerativas y poder aplicar la informática en apenas, una arista de ese mundo, ha sido una valiosa motivación, con ansias de que, a futuro, se continúe con el trabajo y se llegue a lograr una herramienta complementaria que mejore la eficacia y la personalización de las terapias del habla de pacientes con estas patologías.

Por otro lado, fue necesario que desarrolle una aplicación web desde cero de manera individual, lo que me obligó a hacer un repaso de todos los componentes necesarios para lograr las funcionalidades deseadas y, en consecuencia, me llevó a reforzar conceptos estudiados durante la carrera y ponerlos en práctica en un desarrollo interdisciplinario.

Para finalizar, el procesamiento del lenguaje natural puede ser utilizado de diversas maneras para apoyar las terapias de enfermedades neurodegenerativas que afectan la capacidad de una persona para comunicarse y comprender el lenguaje, como por ejemplo, afasia. Pero, es importante destacar que aunque el NLP puede ser una herramienta valiosa, no reemplaza la importancia de la terapia tradicional realizada por profesionales de la salud especializados en el tratamiento de estas enfermedades. Sin embargo, puede ser una herramienta complementaria que ayude a los terapeutas a personalizar el tratamiento y realizar un seguimiento objetivo del progreso del paciente a lo largo del tiempo.

6.2. Trabajos futuros

Este trabajo tiene muchos temas que continuar explotando, de los cuales, se proyectan los siguientes:

- Extender la tarea interdisciplinaria y trabajar conjuntamente con especialistas en el tema para:
 - Mejorar y ampliar las funcionalidades de la herramienta presentada.
 - Considerar otras enfermedades neurodegenerativas que afecten el habla.
 - Ampliar los casos de prueba con más pacientes. Si bien, en la provincia no están del todos identificados con la problemática y muchos están internados en centros asistenciales, se pretende trabajar con algunos profesionales y, a partir de su contacto, incorporar nuevos casos y mejorar la evaluación de A-e-i!.
 - Trabajar en el diseño de indicadores que interpreten las medidas calculadas al momento y poder realizar un análisis de los resultados generados.
 - Realizar un estudio pormenorizado de las estructuras gramaticales del idioma español.

Desde el punto de vista tecnológico, se continuará con el estudio de servicios que mejoren la transcripción realizada y procesen el lenguaje natural, en especial, el idioma español.

Referencias

- Adams, J., Bedrick, S., Fergadiotis, G., Gorman, K., and Van Santen, J. (2017). Target word prediction and paraphasia classification in spoken discourse. En *BioNLP 2017*, páginas 1–8.
- Ali, S. S. and Shapiro, S. C. (1993). Natural language processing using a propositional semantic network with structured variables. *Minds and machines*, 3:421–451.
- Alzheimer, A. (1911). Concerning unusual medical cases in old age. *Zeitschrift Fur Die Gesamte Neurologie Und Psychiatrie*, 4:356–385.
- Amazon (2023). Aws cloud computing. [Online; Accedido el 5-03-2023].
- Amazon Comprehend (2022). Amazon. [Online; Accedido el 4-03-2023].
- Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., Mitchell, N., Muthusamy, V., Rabbah, R., Slominski, A., et al. (2017). Serverless computing: Current trends and open problems. *Research advances in cloud computing*, páginas 1–20.
- Bird, S., Klein, E., and Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc."
- Boletín Oficial de la República Argentina (2000). Ley n° 25.326: Ley de protección de datos personales. https://www.diputados.gov.ar/export/hcdn/secparl/dgral_info_parlamentaria/dip/archivos/Ley_25326.pdf. [Online; 20-04-2023].
- Boletín Oficial de la República Argentina (2001). Decreto reglamentario 1558 / 2001. <https://www.argentina.gob.ar/normativa/nacional/decreto-1558-2001-70368/actualizacion>. [Online; Accedido el 20-04-2023].
- Boschi, V., Catricala, E., Consonni, M., Chesi, C., Moro, A., and Cappa, S. F. (2017). Connected speech in neurodegenerative language disorders: a review. *Frontiers in psychology*, 8:269.
- Brambati, S. M., Ogar, J., Neuhaus, J., Miller, B. L., and Gorno-Tempini, M. L. (2009). Reading disorders in primary progressive aphasia: a behavioral and neuroimaging study. *Neuropsychologia*, 47(8-9):1893–1900.
- Buckingham Jr, H. W. and Kertesz, A. (1974). A linguistic analysis of fluent aphasia. *Brain and language*, 1(1):43–61.

- Cambria, E. and White, B. (2014). Jumping nlp curves: A review of natural language processing research. *IEEE Computational intelligence magazine*, 9(2):48–57.
- Ceccato, S. (1967). Concepts for a new systematics. *Information storage and retrieval*, 3(4):193–214.
- Cherney, L. R. (1995). Efficacy of oral reading in the treatment of two patients with chronic broca’s aphasia. *Topics in Stroke Rehabilitation*, 2(1):57–67.
- Cherney, L. R. (2010). Oral reading for language in aphasia: Impact of aphasia severity on cross-modal outcomes in chronic nonfluent aphasia. En *Seminars in speech and language*, volume 31, páginas 042–051. © Thieme Medical Publishers.
- Cherney, L. R., Halper, A. S., Holland, A. L., and Cole, R. (2008). Computerized script training for aphasia: Preliminary results.
- Cherney, L. R., Merbitz, C. T., and Grip, J. C. (1986). Efficacy of oral reading in aphasia treatment outcome. *Rehabilitation Literature*, 47(5-6):112–118.
- Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on information theory*, 2(3):113–124.
- Chuang, S.-P., Sung, T.-W., Liu, A. H., and Lee, H.-y. (2020). Worse wer, but better bleu? leveraging word embedding as intermediate in multitask end-to-end speech translation. *arXiv preprint arXiv:2005.10678*.
- Chui, M. and Malhotra, S. (2018). Ai adoption advances, but foundational barriers remain. *Mckinsey and company*.
- Clements, P., Garlan, D., Little, R., Nord, R., and Stafford, J. (2003). Documenting software architectures: views and beyond. En *25th International Conference on Software Engineering, 2003. Proceedings.*, páginas 740–741. IEEE.
- Cloud, H. (2011). The nist definition of cloud computing. *National Institute of Science and Technology, Special Publication*, 800(2011):145.
- Collins, M. (2002). Statistical methods in natural language processing. *AT&T Labs-Research*.
- Company, E. (2023). Spanish · spacy models documentation. https://spacy.io/models/es#es_core_news_sm. (Accedido el 05/29/2023).
- Costumero, R., García-Pedrero, Á., Gonzalo-Martín, C., Menasalvas, E., and Millan, S. (2014). Text analysis and information extraction from spanish written documents. En *Brain Informatics and Health: International Conference, BIH 2014, Warsaw, Poland, August 11-14, 2014, Proceedings*, páginas 188–197. Springer.
- De la Vega, R. y Zambrano, A. (2007). Clasificación etiológica de las demencias. <https://www.hipocampo.org/clasifica.asp>. [Online; Accedido el 28-03-2023].
- Delor, M., Martí, A., and Recasens, M. (2008). Ancora: Multilevel annotated corpora for catalan and spanish.

- DFT, I. (2023). Demencia frontotemporal (dft) - variante conductual – ineco. <https://www.ineco.org.ar/patologias/demencia-frontotemporal-dft-variante-conductual/>. (Accedido el 03/28/2023).
- Docs, A. C. (2023a). Modelos de servicio en la nube | tipos de cloud computing | aws. <https://aws.amazon.com/es/types-of-cloud-computing/>. (Accedido el 05/30/2023).
- Docs, G. C. (2023b). Speech-to-text: reconocimiento de voz automático | cloud speech-to-text | google cloud. <https://cloud.google.com/speech-to-text?hl=es>. (Accedido el 05/30/2023).
- Docs, I. C. (2021). Speech to text docs | ibm cloud docs. <https://cloud.ibm.com/docs/speech-to-text/index.html#languages>. (Accedido el 05/30/2023).
- Docs, M. A. (2023c). Speech to text – audio to text translation | microsoft azure. <https://azure.microsoft.com/en-us/products/cognitive-services/speech-to-text>. (Accedido el 05/30/2023).
- Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures*. University of California, Irvine.
- Fraser, K. C., Meltzer, J. A., Graham, N. L., Leonard, C., Hirst, G., Black, S. E., and Rochon, E. (2014). Automated classification of primary progressive aphasia subtypes from narrative speech transcripts. *cortex*, 55:43–60.
- Google (2023). Google cloud platform services summary. [Online; Accedido el 5-03-2023].
- Google Cloud Platform (2022). Ia de natural language. [Online; Accedido el 4-03-2023].
- Gorno-Tempini, M., Hillis, A., Weintraub, S., Kertesz, A., Mendez, M., Cappa, S., Ogar, J., Rohrer, J., Black, S., Boeve, B., Manes, F., Dronkers, N., Vandenberghe, R., Rascovsky, K., Patterson, K., Miller, B., Knopman, D., Hodges, J., Mesulam, M., and Grossman, M. (2011). Classification of primary progressive aphasia and its variants. *Neurology*, 76(11):1006–1014, ISSN: 0028-3878, doi:10.1212/WNL.0b013e31821103e6, <https://n.neurology.org/content/76/11/1006>.
- Green Jr, B. F., Wolf, A. K., Chomsky, C., and Laughery, K. (1961). Baseball: an automatic question-answerer. En *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*, páginas 219–224.
- Guevara-Rukoz, A., Demirsahin, I., He, F., Chu, S.-H. C., Sarin, S., Pipatsrisawat, K., Gutkin, A., Butryna, A., and Kjartansson, O. (2020). Crowdsourcing Latin American Spanish for Low-Resource Text-to-Speech. En *Proceedings of The 12th Language Resources and Evaluation Conference (LREC)*, páginas 6504–6513, Marseille, France. European Language Resources Association (ELRA), ISBN: 979-10-95546-34-4, <https://www.aclweb.org/anthology/2020.lrec-1.801>.
- Henriksson, I. and Laakso, K. (2020). Book talk and aphasia: the power of a book. *International Journal of Language & Communication Disorders*, 55(1):136–148.
- Hernández-Sampieri, R. and Mendoza, C. (2020). *Metodología de la investigación: las rutas cuantitativa, cualitativa y mixta*. McGraw-hill.

- Hodges, J. R. and Miller, B. (2001). The classification, genetics and neuropathology of frontotemporal dementia. introduction to the special topic papers: Part i. *Neurocase*, 7(1):31–35.
- Honnibal, M. and Montani, I. (2017). spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. [Online; Accedido el 4-03-2023].
- IBM (2021). Watson. [Online; Accedido el 4-03-2023].
- Ineco (2023a). Demencia frontotemporal (dft) - variante logopéica – ineco. <https://www.ineco.org.ar/patologias/demencia-frontotemporal-dft-variante-logopenica/>. (Accedido el 03/28/2023).
- Ineco (2023b). Demencia semántica (ds) – ineco. <https://www.ineco.org.ar/patologias/demencia-semantica-ds/>. (Accedido el 03/28/2023).
- Institute, N. H. G. R. (2022). Nhgri history and timeline of events. <https://www.genome.gov/about-nhgri/Brief-History-Timeline>. (Accedido el 05/29/2023).
- Iragorri Cucalón, Á. M. (2007). Demencia frontotemporal. *Revista colombiana de psiquiatría*, 36:139–156.
- Jacobson, I., Booch, G., and Rumbaugh, J. (2000). El lenguaje unificado de modelado uml. *JACOBSON, I*.
- Jones, K. S. (1994). Natural language processing: a historical review. *Current issues in computational linguistics: in honour of Don Walker*, páginas 3–16.
- Kertesz, A. (2007). Western aphasia battery–revised.
- Komprehend (2020). Text analysis apis. [Online; Accedido el 4-03-2023].
- Le, D., Licata, K., and Provost, E. M. (2017). Automatic paraphasia detection from aphasic speech: A preliminary study. En *Interspeech*, páginas 294–298.
- Leal, M. (2023). Demencia: cómo es el sistema de ia que promete dar un diagnóstico temprano a pacientes de américa latina-infobae. http://www.infobae.com/america/ciencia-america/2023/04/09/demencia_como_es_el_sistema_de_ia_que_promete_dar_un_diagnostico_temprano_a_pacientes_de_america_latina/. (Accedido el 05/31/2023).
- Liddy, E. D. (2001). Natural language processing.
- Lippmann, R. P. (1997). Speech recognition by machines and humans. *Speech communication*, 22(1):1–15.
- Lowerre, B. and Reddy, R. (1976). The harpy speech recognition system: performance with large vocabularies. *The Journal of the Acoustical Society of America*, 60(S1):S10–S11.
- MacWhinney, B., Fromm, D., Forbes, M., and Holland, A. (2011). Aphasiabank: Methods for studying discourse. *Aphasiology*, 25(11):1286–1307.

- Maheshwari Utkarsh, U. E. (2023). Improve recognition accuracy with phrase list - azure cognitive services | microsoft learn. <https://learn.microsoft.com/en-us/azure/cognitive-services/speech-service/improve-accuracy-phrase-list?pivots=programming-language-python&tabs=terminal>. (Accedido el 05/22/2023).
- Mahmoud, S. S., Pallaud, R. F., Kumar, A., Faisal, S., Wang, Y., and Fang, Q. (2023). A comparative investigation of automatic speech recognition platforms for aphasia assessment batteries. *Sensors*, 23(2):857, ISSN: 1424-8220, doi:10.3390/s23020857, <http://dx.doi.org/10.3390/s23020857>.
- Mann, W. C. and Thompson, S. A. (1987). *Rhetorical structure theory: A theory of text organization*. University of Southern California, Information Sciences Institute Los Angeles.
- Matetext (2023). facebook/wav2vec2-large-xlsr-53-spanish model - nlp hub - metatext. <https://metatext.io/models/facebook-wav2vec2-large-xlsr-53-spanish>. (Accedido el 05/30/2023).
- Matías, G. (2020). La afasia progresiva primaria y sus 3 tipos - asociación ayuda afasia. <https://afasia.org/blog-afasia-progresiva-primaria-tipos-preguntas/>. (Accedido el 03/28/2023).
- Merrett, R. (2015). Future of mobile, iot driven by speech recognition: Andrew ng - cio. (Accedido el 05/03/2023).
- Mesulam, M.-M. (2001). Primary progressive aphasia. *Annals of neurology*, 49(4):425–432.
- Microsoft (2023). What is paas? [Online; Accedido el 5-03-2023].
- Microsoft Contributors (2022). What is language understanding (luis)? <https://learn.microsoft.com/es-es/azure/cognitive-services/luis/what-is-luis>. [Online; Accedido el 4-03-2023].
- Moguilner, S., Whelan, R., Adams, H., Valcour, V., Tagliazucchi, E., and Ibáñez, A. (2023). Visual deep learning of unprocessed neuroimaging characterises dementia subtypes and generalises across non-stereotypic samples. *eBioMedicine*, 90:104540, ISSN: 2352-3964, doi:<https://doi.org/10.1016/j.ebiom.2023.104540>, <https://www.sciencedirect.com/science/article/pii/S2352396423001056>.
- Mora Teruel, F. and Segovia de Arana, J. (2002). Enfermedades neurodegenerativas. *Farmaindustria, Serie Científica: Madrid*.
- Nadkarni, P. M., Ohno-Machado, L., and Chapman, W. W. (2011). Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, 18(5):544–551.
- Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. En *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, páginas 311–318.

- Peintner, B., Jarrold, W., Vergyri, D., Richey, C., Tempini, M. L. G., and Ogar, J. (2008). Learning diagnostic models using speech and language measures. En *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, páginas 4648–4651. IEEE.
- Pick, A. (1904). Zur symptomatologie der linksseitigen schläfenlappenatrophie. *European Neurology*, 16(4):378–388.
- Qian, L., Luo, Z., Du, Y., and Guo, L. (2009). Cloud computing: An overview. En *Cloud Computing: First International Conference, CloudCom 2009, Beijing, China, December 1-4, 2009. Proceedings 1*, páginas 626–631. Springer.
- Richardson, L. and Ruby, S. (2008). *RESTful Web Services*. O’Reilly Media, ISBN: 9780596554606, <https://books.google.com.ar/books?id=XUaErakHsoAC>.
- Schank, R. C. and Tesler, L. (1969). A conceptual dependency parser for natural language. En *International Conference on Computational Linguistics COLING 1969: Preprint No. 2*.
- Soriano, F. G. and Martínez-Cuitiño, M. (2020). Presente, pasado y futuro de las afasias progresivas primarias: Una actualización bibliográfica. *Revista de Psicología*, 16(31):7–28.
- spaCy (2023). Language processing pipelines · spacy usage documentation. <https://spacy.io/usage/processing-pipelines>. (Accedido el 05/28/2023).
- Spacy (2023). Models and languages. [Online; Accedido el 4-04-2023].
- Swanson, S. J., Pillay, S. B., Elverman, K. H., and Umfleet, L. G. (2019). Primary progressive aphasia. *Handbook on the Neuropsychology of Aging and Dementia*, páginas 489–501.
- Text Razor (2022). Textrazor api. [Online; Accedido el 4-03-2023].
- TIC, L. (2017). Cloud computing. <https://laboratorio-tic.blogspot.com/2017/08/cloud-computing.html>. (Accedido el 05/28/2023).
- Weizenbaum, J. (1966). Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.
- Wikipedia (2023). Red semántica - wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Red_semantica. (Accedido el 05/28/2023).
- Wilson, S. M., Henry, M. L., Besbris, M., Ogar, J. M., Dronkers, N. F., Jarrold, W., Miller, B. L., and Gorno-Tempini, M. L. (2010). Connected speech production in three variants of primary progressive aphasia. *Brain*, 133(7):2069–2088.

Anexo A

Wireframes



Figura A.1 Pantalla de inicio de sesión.



Figura A.2 Pantalla de bienvenida.

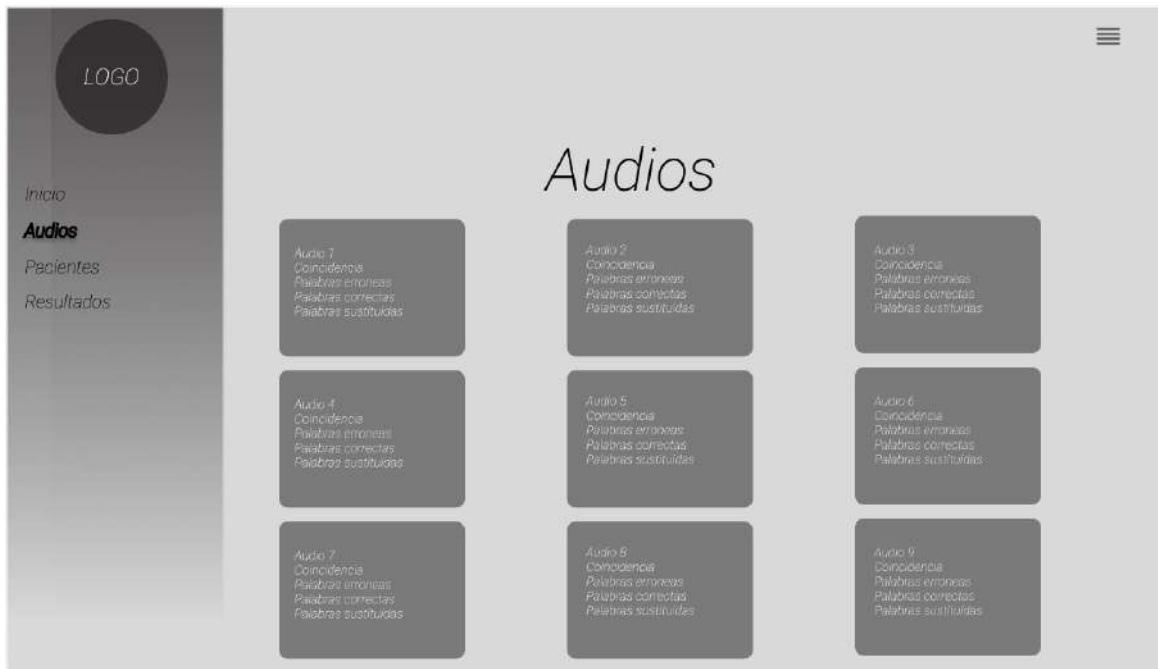


Figura A.3 Pantalla de los audios junto a sus resultados.



Figura A.4 Pantalla de los resultados obtenidos por paciente a través del tiempo.

Anexo B

Tablas

Tabla B.1 Relaciones de dependencia posibles en un discurso.

acl	nummod	csubj:pass
acl:relcl	obj	dep
advcl	obl	det
advmod	obl:agent	discourse
amod	obl:arg	expl
appos	orphan	expl:impers
aux	parataxis	expl:pass
aux:pass	punct	expl:pvt
case	root	fixed
cc	vocative	flat
ccomp	xcomp	
compound	list	
compound:lvc	mark	
conj	nmod	
cop	nsubj	
csubj	nsubj:pass	

Anexo C

Código

```
1
2 CREATE DATABASE AEIBD;
3 USE [AEIBD]
4 CREATE TABLE enfermedades (
5     id INTEGER NOT NULL,
6     nombre VARCHAR(30),
7     descripcion VARCHAR(100),
8     PRIMARY KEY (id),
9     UNIQUE (nombre),
10    UNIQUE (descripcion)
11 );
12
13 CREATE TABLE users (
14     id INTEGER NOT NULL,
15     email VARCHAR(100),
16     password VARCHAR(100),
17     nombre VARCHAR(100),
18     dni VARCHAR(20) NOT NULL,
19     sexo VARCHAR(10),
20     imagen VARCHAR(100),
21     user_type VARCHAR(32) NOT NULL,
22     PRIMARY KEY (id),
23     UNIQUE (email)
24 );
25
```

```
26 CREATE TABLE doctores (  
27   id INTEGER NOT NULL,  
28   inicio DATETIME NOT NULL,  
29   matricula INTEGER NOT NULL,  
30   especialidad VARCHAR(80),  
31   PRIMARY KEY (id),  
32   FOREIGN KEY(id) REFERENCES users (id)  
33 );  
34  
35 CREATE TABLE pacientes (  
36   id INTEGER NOT NULL,  
37   doctor_id INTEGER NOT NULL,  
38   fecha_nacimiento DATETIME NOT NULL,  
39   enfermedad_id INTEGER NOT NULL,  
40   sintoma VARCHAR(20),  
41   ocupacion_previa VARCHAR(30),  
42   ocupacion_actual VARCHAR(30),  
43   gravedad VARCHAR(20),  
44   descripcion VARCHAR(100),  
45   PRIMARY KEY (id),  
46   FOREIGN KEY(id) REFERENCES users (id),  
47   FOREIGN KEY(doctor_id) REFERENCES users (id),  
48   FOREIGN KEY(enfermedad_id) REFERENCES enfermedades (id),  
49   UNIQUE (descripcion)  
50 );  
51  
52 CREATE TABLE file (  
53   id INTEGER NOT NULL,  
54   nombre VARCHAR(100),  
55   id_paciente INTEGER NOT NULL,  
56   transcripcion_referencia VARCHAR(1000) NOT NULL,  
57   resultado_transcripcion VARCHAR(1000),  
58   wer FLOAT,  
59   duracion FLOAT,  
60   fecha DATETIME,  
61   partes_discurso_original VARCHAR(400),  
62   partes_discurso_transcripcion VARCHAR(400),  
63   imagen_dependencias VARCHAR(100),  
64   total_palabras_original INTEGER,
```

```

65 total_palabras_transcripcion INTEGER,
66 palabras_correctas INTEGER,
67 palabras_sustituidas INTEGER,
68 palabras_eliminadas INTEGER,
69 palabras_insertadas INTEGER,
70 coincidencia FLOAT,
71 coincidencia_spacy FLOAT,
72 palabras_por_min_transcripcion FLOAT,
73 porcentaje_palabras_pronunciadas FLOAT,
74 PRIMARY KEY (id),
75 UNIQUE (nombre),
76 FOREIGN KEY(id_paciente) REFERENCES users (id)
77 );

```

Código C.1 Contenido del archivo init.sql

```

1 def wer(ref, hip):
2
3     r = ref.split()
4     h = hip.split()
5     # costos almacenará los costos, como en el algoritmo de distancia de
6     # Levenshtein
7     costos = [[0 for inner in range(len(h)+1)] for outer in range(len(r)+1)]
8     # backtrace mantendrá las operaciones realizadas para luego poder
9     # retroceder, como lo exige el algoritmo WER.
10    backtrace = [[0 for inner in range(len(h)+1)] for outer in
11                range(len(r)+1)]
12
13    correcto = []
14
15    OP_OK = 0
16    OP_SUB = 1
17    OP_INS = 2
18    OP_DEL = 3
19
20    DEL_PENALTY=1
21    INS_PENALTY=1
22    SUB_PENALTY=1
23    # La primera columna representa el caso en el que se logra diferencia

```

```
24     # de cero entre las palabras de hipótesis y las de referencia.
25     for i in range(1, len(r)+1):
26         costos[i][0] = DEL_PENALTY*i
27         backtrace[i][0] = OP_DEL
28
29     for j in range(1, len(h) + 1):
30         costos[0][j] = INS_PENALTY * j
31         backtrace[0][j] = OP_INS
32
33     # cálculo
34     for i in range(1, len(r)+1):
35         for j in range(1, len(h)+1):
36             if r[i-1] == h[j-1]:
37                 costos[i][j] = costos[i-1][j-1]
38
39                 correctas.append(r[i-1])
40
41                 backtrace[i][j] = OP_OK
42             else:
43                 substitutionCost = costos[i-1][j-1] + SUB_PENALTY
44                 insertionCost     = costos[i][j-1] + INS_PENALTY
45                 deletionCost       = costos[i-1][j] + DEL_PENALTY
46
47                 costos[i][j] = min(substitutionCost, insertionCost,
48                                     deletionCost)
49                 if costos[i][j] == substitutionCost:
50                     backtrace[i][j] = OP_SUB
51                 elif costos[i][j] == insertionCost:
52                     backtrace[i][j] = OP_INS
53                 else:
54                     backtrace[i][j] = OP_DEL
55
56     # rastrear hacia atrás a través de la mejor ruta:
57     i = len(r)
58     j = len(h)
59     numSub = 0
60     numDel = 0
61     numIns = 0
62     numCor = 0
```

```

63
64 while i > 0 or j > 0:
65     if backtrace[i][j] == OP_OK:
66         numCor += 1
67         i-=1
68         j-=1
69
70     elif backtrace[i][j] == OP_SUB:
71         numSub +=1
72         i-=1
73         j-=1
74
75     elif backtrace[i][j] == OP_INS:
76         numIns += 1
77         j-=1
78
79     elif backtrace[i][j] == OP_DEL:
80         numDel += 1
81         i-=1
82
83
84 wer_result = round( (numSub + numDel + numIns) /
85 (float) (len(r)), 3)
86
87 return wer_result

```

Código C.2 Algoritmo de WER basado en la distancia de Levenshtein

```

1
2 def test_facebook_model(archivo, transcription):
3
4     audio, rate = librosa.load(archivo, sr = 16000)
5
6     limit = 150000
7     audio_chunks = [audio[i:i+limit] for i in range(0,audio.shape[0],limit)]
8
9     transcripcion_completa = []
10    try:
11        for audio_chunk in audio_chunks:
12            input_values = tokenizer(audio_chunk,

```

```

13     return_tensors = "pt").input_values
14
15     logits = model(input_values).logits
16
17     prediction = torch.argmax(logits, dim = -1)
18
19     transcription = tokenizer.batch_decode(prediction)[0]
20     transcripcion_completa.append(transcription)
21
22     transcripcion_completa = " ".join(transcripcion_completa)
23
24     file_name = 'audio_transcript__'+str(uuid.uuid4())+'.txt'
25
26     return wer(transcription, transcripcion_completa.lower())
27 except:
28     print("error")

```

Código C.3 Algoritmo de prueba de servicio de transcripción de Facebook

```

1 def test_google_sr(archivo, transcripcion):
2     AUDIO_FILE=archivo
3     r=sr.Recognizer()
4     with sr.AudioFile(AUDIO_FILE) as source:
5         r.adjust_for_ambient_noise(source)
6         audio=r.record(source)
7         transcription = r.recognize_google(audio, language = 'es-ES')
8
9     return wer(referencia, transcription.lower())

```

Código C.4 Algoritmo de prueba de servicio de transcripción Google Speech Recognition

```

1 def test_speech_azure(audio_filename, referencia):
2     speech_config = speechsdk.SpeechConfig(subscription=speech_key,
3     region=service_region)
4     speech_config.speech_recognition_language = languageCode
5
6     audio_input = speechsdk.AudioConfig(filename=audio_filename)
7     speech_recognizer = speechsdk.SpeechRecognizer(speech_config=
8     speech_config,
9     audio_config=audio_input)
10

```

```

11 done = False
12 transcripcion_completa = []
13
14 def stop_cb(evt):
15     speech_recognizer.stop_continuous_recognition()
16     nonlocal done
17     done = True
18
19 speech_recognizer.recognized.connect(lambda evt: transcripcion_completa
20     .append(evt.result.text.lower()))
21
22 speech_recognizer.session_stopped.connect(stop_cb)
23 speech_recognizer.canceled.connect(stop_cb)
24
25 speech_recognizer.start_continuous_recognition()
26 while not done:
27     time.sleep(.5)
28
29 if done:
30     transcripcion_completa = " ".join(transcripcion_completa)
31
32     return wer(transcripcion_completa, referencia)

```

Código C.5 Algoritmo de prueba de servicio de transcripción de Microsoft Azure Cognitive Services

```

1 def test_speech_ibm(archivo, referencia):
2     audio_file = sr.AudioFile(archivo)
3     r = sr.Recognizer()
4     with audio_file as source:
5         audio=r.record(source)
6         transcripcion_speech_ibm = r.recognize_ibm(audio,
7             username=os.getenv('USERNAME'),
8             password=os.getenv(PASSWORD))
9
10    return wer(transcripcion_speech_ibm, referencia)

```

Código C.6 Algoritmo de prueba de servicio de transcripción IBM Watson