ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN.

GRADO EN INGENIERÍA EN TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

ÁREA DE INGENIERÍA TELEMÁTICA

DESARROLLO DE UNA SKILL EN AMAZON ALEXA PARA MEJORAR EL HABLA DE NIÑOS CON TRASTORNOS COMUNICATIVOS

Dña. FONTERIZ CONCHESO, Lucía
TUTOR: D. CORCOBA MAGAÑA, Víctor

FECHA: Julio de 2023

Resumen

En los últimos años, el uso de asistentes virtuales ha crecido enormemente. Cada vez es más común la presencia de este tipo de dispositivos en los hogares, ya sea para buscar información en la red, marcar recordatorios, reproducir contenido multimedia...

Relacionado con esto, la tecnología juega papel fundamental en nuestro día a día. Tanto con motivos de ocio, como de trabajo o estudio, son múltiples las opciones que existen en el mercado tecnológico. Su presencia hace que incluso los más pequeños tengan un gran dominio y pasión por su uso. Y siendo así, ¿por qué no combinar la atracción que sienten por la tecnología con las necesidades de desarrollo que necesitan?

Con el fin de integrar la tecnología en el ámbito de la logopedia surge *HablaConmigo*, una skill para dispositivos Amazon Alexa que permita adaptar el contenido y las actividades a las necesidades terapéuticas que requiera cada niño, logrando un aprendizaje efectivo, pero siempre manteniendo el entretenimiento que les brinda la tecnología.

Contenidos

Figuras

Tablas

Acrónimos

- TEL: Trastorno Específico del Lenguaje
- IoT: Internet of Things
- TIC: Tecnologías de la Información y la Comunicación
- IA: Inteligencia Artificial
- VUI: Voice User Interface (Interfaz de Usuario de Voz)
- RAH: Reconocimiento Automático del Habla
- HMM: Hidden Markov Model (Modelo Oculto de Markov)
- NLP: Natural Language Processing (Procesamiento del Lenguaje Natural)
- APL: Alexa Presentation Language

1. Memoria

A continuación, se detallan los procedimientos seguidos en el desarrollo de este proyecto, así como una explicación sobre su funcionamiento e información sobre las herramientas utilizadas para su elaboración.

1.1.- INTRODUCCIÓN

El Trastorno Específico del Lenguaje (TEL) [1] [2] es un trastorno del neurodesarrollo persistente y serio que influye en la capacidad de adquirir y desarrollar el lenguaje hablado. Se caracteriza por un significativo y prolongado desarrollo atípico de la adquisición del lenguaje, en comparación con el desarrollo cronológico habitual, independientemente de deficiencias auditivas, motoras, cognitivas, conductuales y el trastorno del espectro autista, y se califica como un trastorno "heterogéneo", ya que nunca vamos a encontrar dos TEL iguales. Esto quiere decir que los síntomas varían mucho de un niño a otro y no siempre se presentan en la misma forma e intensidad. Además, puede afectar a uno o varios aspectos del lenguaje (área fonética y fonológica, semántica, morfosintáctica y pragmática).

Entre los niños que padecen trastornos del habla y/o comunicación se puede observar un elevado porcentaje de fracaso escolar, además de dificultades para la integración social. Quienes sufren dichos trastornos se enfrentan, entre otras cuestiones, a:

- Dificultades para la comprensión de narraciones y conversaciones (problemas para discernir entre información relevante a mantener e información irrelevante a eliminar)
- Dificultades para relacionarse con personas de su mismo grupo de edad: Tienden a relacionarse con adultos, además de ser excluidos por otros niños con desarrollo típico para su edad, lo que conlleva a problemas sociales (especialmente en el ámbito escolar)

Es importante destacar la pandemia de COVID-19 [3]. Si bien aún no se puede hacer un detallado estudio del efecto que ha tenido en esta parte de la población, es una innegable realidad que ha tenido un impacto clínico en la salud mental de los niños. Las recomendaciones específicas al desarrollo del lenguaje indican que sus posibles efectos negativos deben ser tomados en cuenta seriamente por el personal de salud y padres, debiéndose "buscar proactivamente facilitar la creación de un entorno de comunicación óptima para los niños".

En este contexto, la tecnología se presenta como una herramienta poderosa. El uso de las Tecnologías de la Información y Comunicación (TIC) ha transformado nuestra forma de vida, mejorando nuestras comunicaciones, entretenimiento, trabajo y aprendizaje. El Internet de las Cosas (IoT) ha experimentado un crecimiento significativo en los últimos años, conectando objetos físicos a través de redes, ya sea utilizando para ello Internet o una red privada. Estos dispositivos recopilan datos que son analizados mediante técnicas de machine learning o inteligencia artificial (IA), y tomando a continuación decisiones que se envían nuevamente a los dispositivos IoT para su posterior ejecución.

1.2.- OBJETIVOS Y MOTIVACIÓN

Conforme a lo expuesto en el apartado anterior, se concluye que el objetivo principal de este trabajo es el diseño de una skill para dispositivos Amazon Alexa que ayude a mejorar el habla en niños con TEL, utilizando la tecnología como una herramienta de apoyo en la terapia y motivando el aprendizaje del lenguaje de una forma interactiva y lúdica. Junto con este objetivo principal, también se ha perseguido lo siguiente:

- Explorar cómo desarrollar una skill para distintos tipos de dispositivos Amazon Alexa
- Estudiar la arquitectura de Amazon Alexa
- Aplicar Alexa Presentation Language (APL) para dispositivos Amazon Alexa con pantalla

- Uso de una API RESTful para las comunicaciones entre la base de datos y las aplicaciones del sistema
- Aprender a diseñar el frontend de una aplicación web
- Aprender a diseñar diagramas UML

1.3.- ESTUDIOS Y ANÁLISIS PREVIOS

Previo al desarrollo del proyecto, se realizó una breve investigación sobre el estado de los asistentes virtuales y qué áreas del lenguaje podrían trabajarse, junto con un "estudio de mercado" para comprobar la existencia de herramientas similares ya presentes y ver cómo podrían ser mejoradas.

1.3.1.- Asistentes virtuales

Según el diccionario de Cambridge, un asistente virtual es un "programa informático o dispositivo conectado a Internet capaz de entender preguntas e instrucciones habladas, diseñado para ayudarle a hacer planes, encontrar respuestas a preguntas, etc" [4].

Pese a que los asistentes virtuales suelen concebirse como una tecnología moderna, lo cierto es que su historia se remonta a décadas atrás [5]. En 1952, "Laboratorios Bell" desarrolló "Audrey", un sistema de reconocimiento automático del habla (RAH) capaz de entender números. Una década más tarde, en 1962, IBM presentó "Shoebox" [6]. Este dispositivo fue capaz de identificar 16 palabras, entre los que se incluían los números del "0" al "9" y operadores matemáticos como "más", "menos" y "total". El siguiente avance lo logró en 1970 la Universidad Carnegie Mellon junto con la DARPA (Agencia de Proyectos de Investigación Avanzados de Defensa), del Departamento de Defensa de los Estados Unidos. Se trataba del proyecto "Harpy", que logró reconocer un millar de palabras.

En la década de los 80, el mismo grupo de científicos desarrolló un sistema que podía analizar tanto palabras individuales como secuencias completas de palabras, basándose, para

ello, en el Modelo Oculto de Markov (HMM). Hasta ese momento solo se habían analizado números y palabras independientes, pero con el uso del HMM se comenzaron a diseñar sistemas capaces de analizar frases completas. Este modelo, aplicado al Procesamiento del Lenguaje Natural (NLP), permite establecer relaciones entre las palabras y calcular la probabilidad estadística de que una palabra siga a otra. Así, los primeros asistentes virtuales fueron contestadores automáticos y softwares médicos de dictáfonos digitales, utilizando para ello software de reconocimiento del habla.

Ya en los 90, compañías como IBM, Dragon, Philips, Lernout & Hauspie o Microsoft integraron el RAH en sus ordenadores personales, y en 1994, IBM presentó a "Simon", el primer teléfono inteligente del mercado. Este dispositivo incluía una pantalla tácil LCD, accesible por medio de un lápiz táctil, y un sistema operativo que permitía ejecutar aplicaciones, y sentó las bases para los asistentes virtuales inteligentes tal como los conocemos hoy en día. Tras esto, hubo un importante parón tecnológico debido al estallido de la *burbuja puntocom*.

Entre 2003 y 2008 la DARPA desarrolló el "Proyecto CALO" [7] (*Cognitive Assistant that Learns and Organizes* - Asistente Cognitivo que Aprende y Organiza). El objetivo de este proyecto era integrar diversas tecnologías de IA y aprendizaje automático en una única interfaz de comunicación, obteniendo un software cognitivo capaz de razonar, aprender, explicar y adaptarse a las preferencias del usuario. Este software se convertía así en un asistente digital, capaz de organizar la información y gestionar las tareas y la agenda del usuario, entre otras cosas.

Tras la finalización del proyecto, el SRI fundó Siri Inc., que continuó con la investigación basándose en CALO. En 2010 Apple compró todo el proyecto, y en 2011 presentó "Siri", el primer asistente virtual digital avanzando integrado en el sistema operativo de un dispositivo móvil. A Siri le siguieron los asistentes virtuales que conocemos hoy en día: en 2014 Amazon presentaba a "Alexa" y Microsoft a "Cortana", y en 2016 Google presentaba a "Google Assistant".

En la Figura 1.1 se muestra un pequeño resumen de esta evolución.

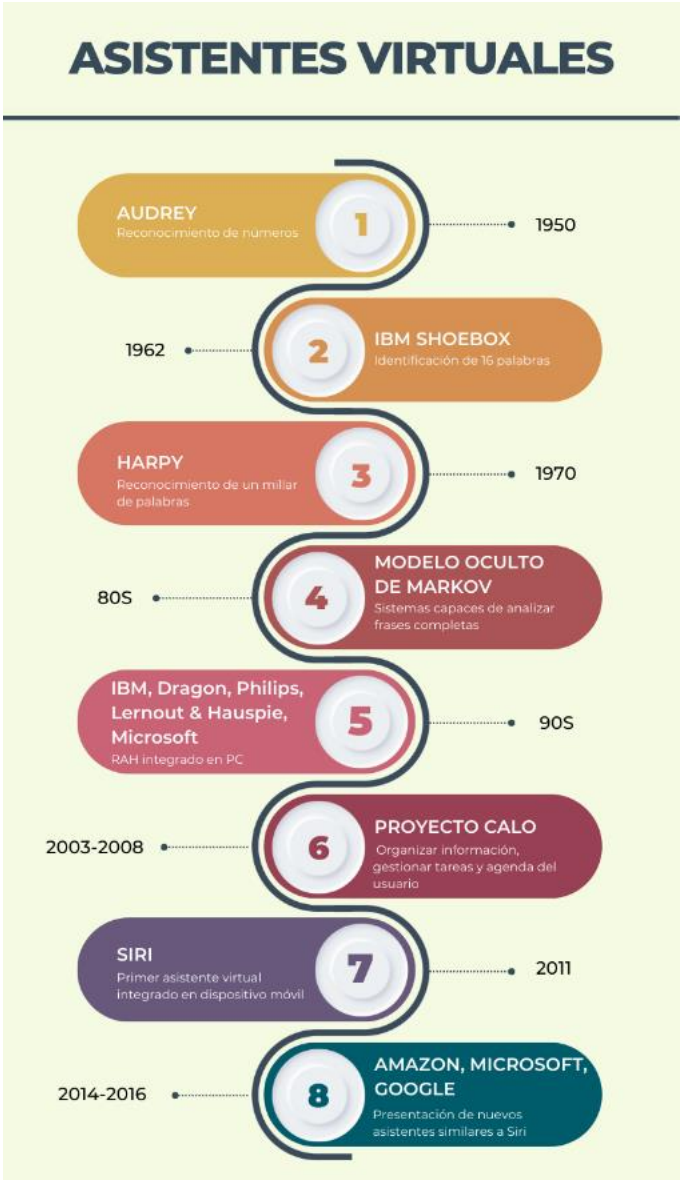

Figura 1.1.- Evolución de los asistentes virtuales. Fuente: Diseño propio con plantilla de Canva

1.3.2.- Estudio de áreas para trabajar

Como paso previo al diseño de la aplicación, se llevó a cabo una investigación para distinguir qué áreas del lenguaje se ven afectadas por este trastorno y la forma en que podría

ayudar en su terapia el uso de HablaConmigo, decidiendo centrarse en el trabajo de la conciencia fonológica, la adquisición de lenguaje y vocabulario y el tratamiento de las dislalias.

1.3.2.1.- Conciencia fonológica

La conciencia fonológica [8] es la habilidad que permite reconocer y manipular los sonidos del lenguaje hablado, algo fundamental para el proceso de aprendizaje de la lectura. Pese a que no existe una opinión común, la mayoría de los expertos coinciden en que esta habilidad se desarrolla principalmente entre los 3 y los 8 años de edad, aunque cada individuo experimenta este proceso de forma distinta.

Existen dos niveles de conciencia fonológica, la conciencia silábica y la conciencia fonémica. La conciencia silábica abarca todo lo relativo al manejo de las sílabas, lo que permite a los niños reconocerlas, contarlas, combinarlas para formar palabras... Además, tiene una gran importancia tanto en la lectura como en la escritura, ya que el conocimiento de la descomposición de palabras facilita la decodificación y comprensión de textos escritos. Por otro lado, la conciencia fonémica abarca todo lo relativo al manejo de los fonemas. Los fonemas son las unidades más pequeñas de las palabras, los sonidos que las componen, por lo que el desarrollo de esta habilidad permite a los niños adquirir una mayor comprensión de la relación existente entre los sonidos del habla y las letras escritas.

Así, una de las áreas del lenguaje que se buscará trabajar con la ayuda de HablaConmigo, es la conciencia fonológica, y, más concretamente, la conciencia silábica.

1.3.2.2.- Desarrollo del lenguaje

El desarrollo del lenguaje [9] es un proceso que comienza cuando el niño nace y que continúa desarrollándose hasta aproximadamente los 7 años. Durante este tiempo los niños pasan por dos grandes etapas:

- Etapa prelingüística: Abarca desde el nacimiento hasta aproximadamente los 18 meses. En esta etapa el niño pasa desde comenzar a diferenciar el habla humana de otros sonidos, al balbuceo reduplicativo hasta poder pronunciar sus primeras palabras.

- Etapa lingüística: Esta etapa abarca desde los 18-24 meses hasta los 7 años, comenzando con las primeras palabras del niño hasta que este es capaz de comunicarse con éxito. Así, en este tiempo, los niños adquieren la capacidad de pronunciar todos los fonemas, y experimentan un enriquecimiento de su vocabulario, utilizando palabras cada vez más complejas. En este periodo también desarrollan habilidades sociales y emocionales, cruciales para una comunicación exitosa.

Así, con el uso de HablaConmigo, centrada especialmente para niños entre 6 y 7 años, se buscará mejorar el desarrollo de su lenguaje mediante un aumento de su vocabulario, además de ayudar a que estimulen su razonamiento verbal.

1.3.2.3.- Dislalias

Pese a que los expertos no concretan una definición común para este término, la mayor parte de ellos coinciden en que la dislalia [10] [11] es un trastorno que afecta a la articulación y/o pronunciación de algunos fonemas o palabras, sin que la causa sea una alteración en el funcionamiento del sistema nervioso central. Se trata del trastorno del habla más común en la población infantil, teniendo en España una afección aproximadamente del 15% en edades preescolares. Conforme los niños crecen, este porcentaje se reduce al 3% [12].

Al igual que con la definición del trastorno, los distintos autores tampoco acuerdan una clasificación común de las dislalias. Para realizar una clasificación de estas, es necesario diferenciar previamente trastornos fonéticos (alteraciones derivadas de problemas motrices) de trastornos fonológicos (problemas derivados de una discriminación auditiva). Una vez

entendida esta distinción, puede realizarse una clasificación de las dislalias, siendo la más común en función de la etiología (Pascual, 1988):

- Dislalia evolutiva: Se trata de anomalías articulatorias presentes en las primeras etapas del desarrollo infantil. Se consideran normales, ya que aparecen durante el proceso de adquisición del lenguaje en el que se encuentran los niños, que en estos momentos aún no son capaces de reproducir exactamente lo que escuchan.

- Dislalia funcional: Se denomina así por la inexistencia de algún trastorno neurológico u orgánico que cause esta anomalía. Una incorrecta utilización de los órganos articulatorios, como los labios o la lengua, es la principal causa de este tipo de dislalia.

- Dislalia audiógena: Es causada por una pérdida de sensibilidad auditiva, que deriva en la confusión de ciertos sonidos similares y, por tanto, en una incorrecta articulación de los mismos.

- Dislalia orgánica: Es un trastorno de la articulación en el habla por lesiones del sistema nervioso, ya sea el central o el periférico. En función de qué afecten estas lesiones, se pueden observar dos tipos de dislalias orgánicas:
    - Disartrias: Las lesiones afectan al habla
    - Disglosias: Las lesiones afectan a los órganos del habla

Algunos problemas asociados a la presencia de una dislalia se mencionan a continuación [13]:

- Rotacismo: Omisión, sustitución o distorsión en la articulación del fonema /r/
- Sigmatismo: Omisión, sustitución o distorsión en la articulación del fonema /s/
- Lambdacismo: Defecto de pronunciación del fonema /l/
- Nasalización: Sustitución de fonema /d/ por fonema /n/
- Duplicación: Sustitución de un fonema por otro previo en la palabra
- Alteración del orden de los fonemas

- Alteración del orden de las sílabas

Mientras que la dislalia es un trastorno que se centra, principalmente, en las dificultades de articulación, el TEL abarca un conjunto de dificultades que engloba lo relacionado con todo el lenguaje, dando lugar a una mayor dificultad para comunicarse [14]. A pesar de ello, son muchos los casos en los que ambos trastornos pueden coexistir, un niño puede tener tanto TEL como dislalia.

Esto, junto con el nivel de incidencia en la población infantil, es la principal causa de elección para intentar tratar este trastorno con la ayuda de HablaConmigo.

1.3.3.- Software existente

Como paso previo al diseño de HablaConmigo, se realizó una investigación sobre softwares similares ya existentes.

Un ejemplo es *Lee Con Leo* [15], una herramienta lanzada a finales de 2017 de la mano de la logopeda cántabra María Domingo disponible para Android, iOS y Windows. Según su página web [16], "previene dificultades en la adquisición de la lectura y la escritura a través del juego y el refuerzo positivo", y ayuda a los niños con la adquisición de "los puntos de articulación utilizados en el habla, la conciencia fonológica, la lectura comprensiva y la escritura y su ortografía".

Figura 1.2.- Página principal de Lee Con Leo. Fuente: Lee Con Leo *[16]*

Otros ejemplos de aplicaciones similares [17] [18] son *Léxico Cognición* (iOS, para trabajar "conceptos y categorías semánticas, estructura gramatical y comprensión de instrucciones"), *Conversation Therapy Lite* (Android e iOS, para trabajar el lenguaje expresivo) o *My Playhome Store* (Android e iOS, para trabajar expresión oral).

Como se puede observar, existen aplicaciones con un propósito similar al de *HablaConmigo* en el mercado, si bien es cierto que no se han encontrado referencias sobre softwares que combinen este propósito con el uso de asistentes virtuales digitales.

1.4.- REQUISITOS MÍNIMOS

A continuación, se detallan los requisitos mínimos que el sistema (conjunto de skill HablaConmigo y aplicación web como herramienta de ayuda al logopeda) debe tener:

- Los usuarios de la skill deben poder registrarse con un identificador personal, su nombre y su edad.

- Los usuarios de la skill deben poder acceder a “su perfil” en sesiones posteriores usando su identificador personal.

- Los usuarios de la skill deben poder elegir entre varios juegos que les permitan trabajar distintas áreas del lenguaje.

- Los usuarios de la skill deben poder elegir entre varios niveles de cada juego, de forma que la skill se adapte a sus requerimientos personales.

- Los usuarios de la skill deben poder cambiar de juego y nivel durante el desarrollo de la sesión.

- El logopeda debe poder añadir nuevas entradas a los juegos utilizando la aplicación web.

- El logopeda debe poder realizar consultas con distintas opciones de periodos de tiempo sobre los resultados obtenidos por un usuario concreto, en un nivel y juego determinados.

1.5.- FUNDAMENTOS DE LAS TECNOLOGÍAS EMPLEADAS EN EL PROYECTO

Previo al desarrollo tanto de la skill como de la aplicación web, se determinaron las tecnologías a usar para su desarrollo, cuyas principales características se exponen a continuación.

1.5.1.- Node.js

Alexa Developer Console permite dos métodos para alojar los recursos de back-end de la skill desarrollada: Node.js y Python. La principal diferencia entre ambos [19] es que Python es un lenguaje de programación, mientras que Node.js es un entorno de ejecución de JavaScript de código abierto, multiplataforma y que no crea un nuevo hilo para cada petición. Node.js también permite la creación de aplicaciones web, además de ejecutar código JavaScript del lado del servidor. Esta última característica es una de las principales razones por las que se ha elegido Node.js para desarrollar HablaConmigo, ya que hace que el desarrollo de aplicaciones sea muy sencillo, y es, a su vez, el método recomendado por Amazon para el desarrollo de skills.

Node.js es uno de los entornos de ejecución que mayor crecimiento ha experimentado desde su aparición. Algunas de las ventajas [20] [21] que tiene su uso son las siguientes:

- Comunidad y soporte: El uso y crecimiento de Node.js, junto con el uso de plataformas como GitHub, ha conllevado la disponibilidad de una gran cantidad de documentación sobre el entorno, incluyendo tutoriales, guías y recursos educativos, lo que convierte a Node.js en una opción atractiva y sencilla para iniciarse en el desarrollo web.

- Escalabilidad: Como se mencionó anteriormente, Node.js utiliza un solo hilo. Esto hace que el entorno sea capaz de procesar una gran cantidad de conexiones de forma simultánea, algo muy útil en aplicaciones web, como chats, juegos en línea y aplicaciones de streaming.

- Rendimiento y eficacia: Relacionado con el punto anterior, cabe destacar que Node.js utiliza un modelo de E/S sin bloqueo, que, junto con su bajo consumo de recursos, lo convierten en un entorno con mejor rendimiento y tiempos de respuesta más rápidos.

- Open source: Node.js es una plataforma de código abierto. Así, no es necesaria ninguna licencia para su utilización, y es accesible para todo el mundo.

1.5.2.- JavaScript

JavaScript [22] es un lenguaje de programación interpretado [23] (utiliza un programa, denominado intérprete, para traducir el código a código máquina sin que sea necesario compilar el código completo, el programa se ejecuta línea a línea) y multiparadigma [24] (permite al programador combinar diferentes paradigmas de programación, lo que aporta flexibilidad y permite desarrollar código más adaptable). Uno de los paradigmas que utiliza JavaScript es la Programación Orientada a Objetos (POO), y más concretamente, un tipo de esta: la Programación Orientada a Prototipos.

A la hora de escribir el código fuente de un programa, hay una serie de reglas que deben seguirse para que el correspondiente lenguaje de programación lo considere correcto: es lo que se denomina sintaxis de un lenguaje. En el caso de JavaScript [25] [26], su sintaxis está influenciada en gran medida por la de Java y C, siendo sus características principales las siguientes:

- Es *case-sensitive*: Es decir, distingue entre mayúsculas y minúsculas. Esto implica que las letras son consideradas diferentes y no pueden intercambiarse indistintamente. Tanto las variables y los nombres de las funciones, como las palabras clave y los identificadores, son sensibles a mayúsculas y minúsculas. Para JavaScript, no es lo mismo una variable declarada como “miVariable” que declarada como “mivariable”, al igual que no es lo mismo definir una estructura con “if”, “If” o “IF”. Esta distinción entre mayúsculas y minúsculas obliga a que las palabras clave, como “if” o “else”, entre otras, deban escribirse siempre en minúsculas. De lo contrario, el lenguaje no las interpretará correctamente.

- No se define el tipo de las variables: No es necesario indicar el tipo de dato que una variable va a almacenar cuando esta se crea, lo que permite que una misma

variable pueda variar el tipo de dato que almacena durante la ejecución del programa.

- No es necesario terminar cada declaración con un punto y coma (;): A diferencia de otros lenguajes, en los que es obligatorio que cada sentencia termine con el carácter punto y coma (;), en JavaScript no lo es, siempre y cuando no se incluyan varias declaraciones en una misma línea, caso en el que sí que se requiere el uso de punto y coma. Aunque su uso no es imperativo, es recomendable utilizarlo para evitar errores de código.

Dos herramientas muy útiles cuando se programa son las librerías (bibliotecas) y los *frameworks* [27]. Las librerías son conjuntos de códigos que permiten a los programadores reutilizar el trabajo realizado por otros, reduciendo así la necesidad de escribir código desde cero y evitando la duplicidad de este. Para utilizar una librería en una aplicación, simplemente habría que añadir una línea como la de la Figura 1.3 en el <head> del archivo, indicando en el atributo *src* la ruta de origen de la biblioteca o la URL donde se encuentra.

```
<script src="./js/jquery-3.5.1.min.js"></script>
```

Figura 1.3.- Código para incluir una librería en JavaScript. Fuente: Diseño propio

Algunas de las bibliotecas de JavaScript más populares son jQuery (se explicará el punto 1.5.3), React.js (librería de código abierto utilizada para construir interfaces de usuario rápidas y dinámicas) y D3.js (librería utilizada para la manipulación de documentos basados en datos).

Por otro lado, los *frameworks* son marcos de aplicación que ofrecen una base para desarrollar un proyecto. Suelen incluir bibliotecas y utilidades adicionales para simplificar el desarrollo de código, de forma que se mejore la eficiencia y calidad de este. Se podría decir que es una especie de "plantilla" que el desarrollador modificará y adaptará a sus requisitos. Algunos de los *frameworks* más populares de JavaScript son Angular (utilizado para crear aplicaciones web de una sola página), Vue.js (*framework* con estructura flexible y escalable que permite construir interfaces de usuario) y Bootstrap (se explicará en el punto 1.5.4).

En resumen, JavaScript se ha convertido en uno de los lenguajes más populares en el desarrollo de aplicaciones web gracias a su capacidad para ejecutarse tanto en el lado del cliente como en el lado del servidor, además de su facilidad para crear aplicaciones web interactivas y dinámicas.

1.5.3.- jQuery

jQuery [28] [29] es, probablemente, la biblioteca de JavaScript más popular del mercado actual. Se trata de una librería de código abierto que enriquece la experiencia del usuario al añadir una capa de interacción AJAX entre la web y las aplicaciones desarrolladas, permitiendo, entre otras cosas, controlar eventos, crear animaciones y añadir efectos visuales.

El DOM (*Document Object Model*) [30] puede definirse como la estructura de un documento HTML. El DOM, o árbol DOM, conformaría del árbol de etiquetas de un documento HTML, como puede apreciarse en la Figura 1.4.

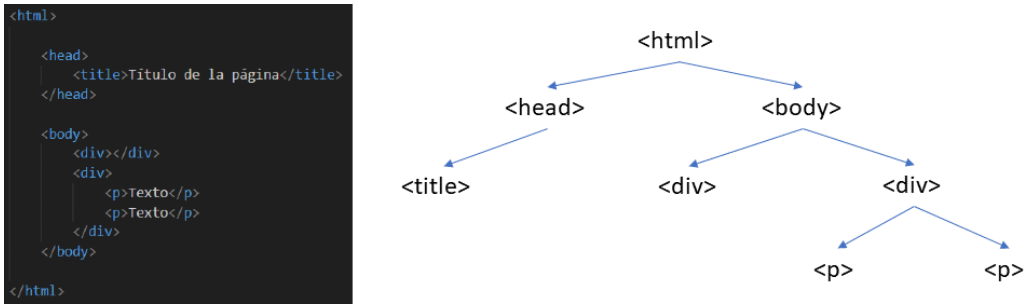

Figura 1.4.- Relación entre HTML y Dom. Fuente: Diseño Propio

Son muchos los elementos de una página web que permite modelar el DOM (la ventana del navegador, el historial...), además de elementos dentro de la propia página (párrafos, tablas, formularios...). A través de JavaScript se puede acceder a estos elementos mediante el DOM, lo que permite a los desarrolladores crear nuevos elementos, modificarlos, suprimirlos, recolocarlos... Aquí es donde entraría jQuery, que con su función de manipulación del DOM facilita en gran medida la realización de estas acciones.

Una característica de JavaScript es que no siempre todo su código es compatible con diferentes navegadores, y pueden ser necesarias diferentes líneas de código para su compatibilidad. jQuery incluye ciertas funciones que facilitan esta compatibilidad, por lo que también simplifica la escritura de código en JavaScript.

1.5.4.- Bootstrap

Bootstrap [31] [32] es un *framework* de CSS muy popular en el ámbito del desarrollo web gracias a su capacidad para facilitar la creación de aplicaciones y páginas web "responsive" [33]. Esto quiere decir que ayuda a que una página web adapte su diseño y contenido a cualquier tipo de pantalla o dispositivo, logrando una correcta visualización de la página y mejorando la experiencia del usuario.

Para lograr esa adaptabilidad, Bootstrap utiliza un sistema de cuadrícula (*grid*) [34] formado por 12 columnas por fila. De esta forma, pueden asignarse distintos anchos a los elementos situados en una misma fila. Otra característica de esta función es la posibilidad de asignar distinto número de columnas a un elemento en función del tamaño de la pantalla. Bootstrap define seis *breakpoints* [35] distintos, que se corresponden con seis tamaños de pantalla distintos (Figura 1.5), lo que permite, entre otras cosas, asignar diferente espacio a ocupar por un elemento en función del tamaño de la pantalla del dispositivo.

Breakpoint	Infijo de clase	Dimensiones
X-Small	*None*	<576px
Small	`sm`	≥576px
Medium	`md`	≥768px
Large	`lg`	≥992px
Extra large	`xl`	≥1200px
Extra extra large	`xxl`	≥1400px

Figura 1.5.- Breakpoints de Bootstrap. Fuente: Bootstrap *[32]*

Dos elementos pueden apreciarse perfectamente ocupando 6 columnas cada uno en una pantalla de ordenador. En cambio, en un dispositivo móvil, no se lograría una visión óptima de esta manera. Con esta funcionalidad de Bootstrap se puede hacer que un elemento ocupe 6 columnas en un tamaño de pantalla de ordenador y las 12 columnas enteras en una pantalla de móvil, por ejemplo. Para hacer esto solo habría que introducir una línea de código como la de la Figura 1.6.

```
<div class="col-lg-6 col-md-12 col-sm-12">
```

Figura 1.6.- Código para modificar espacio a ocupar en función del tamaño de la pantalla.
Fuente: Diseño propio

Una gran ventaja de Bootstrap es la gran cantidad de documentación que puede encontrase en su página web. Esta dispone de multitud de códigos de ejemplo, que con los archivos .css y .js que pueden descargarse desde la misma, permiten crear elementos de una forma muy rápida para el desarrollador. Este, solo tendría que hacer uso de CSS para personalizarlo y adaptarlo a los requerimientos.

Es importante saber que Bootstrap utiliza jQuery para crear interacciones y animaciones, por lo que es necesario incluir esta biblioteca siempre que se vaya a utilizar Bootstrap en un proyecto.

1.5.5.- HTML

HTML (*HyperText Markup Language* – Lenguaje de Marcado de Hipertexto) [36] [37] es el lenguaje que define el contenido, significado y estructura de una página web. Es un lenguaje de marcas (o marcado) porque utiliza etiquetas que añaden información sobre la estructura del texto y los elementos que incluye. Estas etiquetas incluyen el nombre del elemento entre “<” y “>”, y en su interior no se distingue entre mayúsculas y minúsculas. Generalmente, un elemento tiene una etiqueta de inicio y otra de cierre, como en el ejemplo de la Figura 1.7.

```
<div class="texto_formulario">NUEVO JUEGO</div>
```

Figura 1.7.- Ejemplo etiquetas apertura y cierre en HTML. Fuente: Diseño propio

Hay dos tipos de marcado: el marcado presentacional y el marcado hipertextual. El primero permite diseñar la apariencia visual del texto (poner letra en negrita, en cursiva...), mientras que el segundo se utiliza para conectar nuestra página con otras partes de esta o con otras páginas distintas. En la Figura 1.8 pueden observarse algunos ejemplos de etiquetas y qué efecto causan en una página web:

Etiquetas para dar formato a las fuentes		
Etiqueta	**Ejemplo de código**	**Apariencia**
	Negrita	**Negrita**
	Enfatizar el texto	*Enfatizar el texto*
	Cursiva	*Cursiva*
<strike>	<strike>Texto tachado</strike>	Texto tachado
	Texto reforzado por navegador	**Texto reforzado por navegador**
<sup>	Ir ^{superíndice}	Ir superíndice
<sub>	El _{subíndice}	El subíndice
<u>	<u>Subrayado</u>	Subrayado
Etiquetas para dar formato a líneas		
Etiqueta	**Ejemplo de código**	**Apariencia**
	Párrafo 1	Párrafo 1
	Párrafo 2	Párrafo 2
 	Esto es una línea. Esto es el retorno de carro.	Esto es una línea. Esto es el retorno de carro.
Enlaces web		
Etiqueta	**Ejemplo de código**	**Apariencia**
<a>	Enlace a www.google.com	Enlace a **www.google.com**

Figura 1.8.- Ejemplos de etiquetas HTML. Fuente: Proyecto Degardo *[38]*

Relacionados con las etiquetas se encuentran los atributos, valores que pueden añadirse a una etiqueta para ajustar un elemento a los requerimientos de los usuarios. Algunos ejemplos son pueden observarse en la Figura 1.9.

Nombre del atributo	Elementos aplicables	Descripción
align	<applet>, <caption>, <col>, <colgroup>, <hr>, <iframe>, <img>, <table>, <tbody>, <td>, <tfoot> , <th>, <thead>, <tr>	Especifica el alineamiento horizontal del elemento.
bgcolor	<body>, <col>, <colgroup>, <marquee>, <table>, <tbody>, <tfoot>, <td>, <th>, <tr>	Permite configurar el color de fondo del elemento que se esté aplicando como el color de fondo de una página, de una marquesina, otros.
size	<input>, <select>	Define el ancho del elemento (en píxeles). Si el atributo del elemento es del tipo text o password entonces es el número de caracteres.
src	<audio>, <embed>, <iframe>, <img>, <input>, <script>, <source>, <track>, <video>	La URL del contenido integrable.
border	<img>, <object>, <table>	Aplica efectos de borde al elemento.
alt	<applet>, <area>, <img>, <input>	Texto alternativo en caso de que la imagen no se pueda mostrar.

Figura 1.9.- Ejemplos de atributos HTML. Fuente: Recursos Ceainf *[39]*

HTML suele usarse en combinación con CSS para personalizar el diseño de la página web y con JavaScript para aportarle a esta dinamismo e interactividad.

1.5.6.- CSS

CSS (*Cascading Style Sheets* – Hojas de Estilo en Cascada) [40] [41] es un lenguaje de estilo que se utiliza en conjunto con HTML, permitiendo modificar el diseño de una página web para hacerla más atractiva para el usuario. Así, HTML se encarga de la estructura del contenido de la página, mientras que CSS estructura la presentación de esta. La diferencia entre usar CSS y no usarlo es notable (Figura 1.10), ya que una página web sin CSS constaría únicamente de un fondo blanco y distintos tamaños de letra, lo que la haría poco atractiva.

Figura 1.10.- Comparación entre página web con y sin CSS. Fuente: RailsBridge *[42]*

CSS utiliza una sintaxis muy simple, formada por un selector y un bloque de declaración. El selector "selecciona" el elemento HTML que se quiere personalizar. A continuación del selector se incluyen unas llaves {} entre las que se encontrarán las declaraciones. Cada una de estas declaraciones consta de la propiedad que se quiere personalizar y el valor que se le quiere dar. Además de modificar elementos HTML ya definidos, CSS también permite crear nuevos elementos.

Propiedades	Descripción
background-color	Definir color de fondo
background-image	Utilizar una imagen como fondo
background-repeat	Repetir imagen de fondo
background-position	Posicionar una imagen en un lugar determinado
text-align	Centrar un texto
color	Establecer un color de fuente determinado
width	Ancho
height	Alto
border-color	Establecer un color de borde determinado

Tabla 1.1.- Ejemplos de propiedades personalizables con CSS

Otra característica muy interesante de CSS son las pseudoclases y los pseudoelementos. Se trata de palabras clave que permiten personalizar partes específicas de un elemento, o la apariencia de este en un determinado estado. Las pseudoclases [43] [44] se utilizan para personalizar un elemento en un estado determinado, como la situación del cursor (*:hover*), la representación de un enlace que aún no se ha visitado (*:link*) o de uno ya visitado (*:visited*), o un elemento deshabilitado (*:disabled*), entre otros. Por otro lado, los pseudoelementos permiten seleccionar partes específicas de un elemento, y, a diferencia de las pseudoclases, estos se representan mediante dos puntos dobles (::). Una de las posibilidades que ofrece el uso de pseudoclases es insertar contenido antes (*::before*) o después (*::after*) del contenido de un elemento HTML.

1.5.7.- Axios

Axios [45] es una biblioteca de JavaScript que permite realizar las mismas operaciones que un Cliente HTTP para Node.js. Es isomórfico, ya que puede ejecutarse tanto en el navegador como en Node.js utilizado el mismo código base.

Para utilizar Axios es necesario tener este módulo instalado, o, en su defecto, incluirlo en nuestro proyecto. Esto puede hacerse de varias maneras: usando npm (Figura 1.11), bowe, yarn, CDN jsDelivr, CDN unpkg o importando el módulo (Figura 1.12). En el caso de HablaConmigo, se utilizó npm para la aplicación web y se importó el módulo para la skill.

```
> npm install axios
```

Figura 1.11.- Comando para instalar Axios utilizando npm. Fuente: Diseño propio

```
const axios = require('axios');
```

Figura 1.12.- Importar módulo Axios. Fuente: Diseño propio

La principal característica de Axios es su sencillez. Axios ofrece una interfaz muy simple que permite realizar peticiones GET, POST, PUT y DELETE (entre otras) empleando para ello muy pocas líneas de código. Tiene también una sintaxis muy sencilla, como puede verse en los siguientes ejemplos:

```
axios.post('/user', {
    firstName: 'Fred',
    lastName: 'Flintstone'
  })
  .then(function (response) {
    console.log(response);
  })
  .catch(function (error) {
    console.log(error);
  });
```

Figura 1.13.- Ejemplo de petición POST utilizando Axios. Fuente: Axios *[45]*

1.5.8.- Alexa Presentation Language (APL)

Alexa Presentation Language (APL) es un lenguaje de diseño creado por Amazon, basado en JSON y que permite la creación de interfaces que combinan visuales con vídeo y audio. Este lenguaje ayuda a los desarrolladores a incluir elementos como botones o "scrollers", imágenes, vídeos o texto personalizable por el usuario. La presencia de todos estos elementos visuales, junto con la interacción de voz de Alexa, enriquece enormemente la experiencia del usuario.

Dentro de la Alexa Skills Kit (ASK), se incluye en la pestaña de "Build" el apartado "Multimodal Responses" (Figura 1.14). Esto permite crear plantillas en APL, tanto en forma de código, como arrastrando y personalizando los campos como si de un formulario se tratase. En este segundo caso, la propia herramienta genera el código de la plantilla.

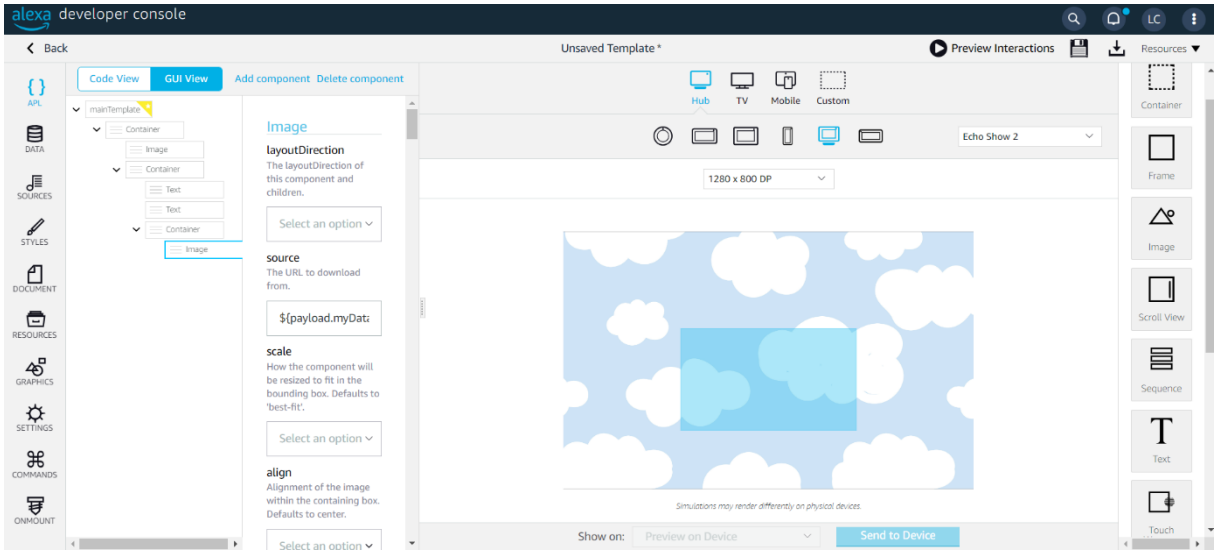

Figura 1.14.- Creación de plantillas APL con Multimodal Responses. Fuente: Diseño propio

1.6.- DEFINICIÓN DE LAS ETAPAS DEL PROYECTO

Previo a comentar la planificación temporal del proyecto, es necesario definir las etapas que lo componen:

1. Estudio del proyecto
2. Definición funcional
3. Diseño de la skill y aplicación web
4. Desarrollo de la skill y aplicación web
5. Fase de pruebas
6. Documentación del proyecto

1.6.1.- Estudio del proyecto

La primera etapa se corresponde con realizar un estudio de la situación actual de softwares similares en el mercado para ver qué incluyen y qué no, y cómo podrían mejorarse, además de definir qué se pretende lograr con el uso de la aplicación.

Para la realización del estudio del proyecto se estima una duración de 64 horas.

1.6.2.- Definición funcional

En esta segunda etapa se definen los requerimientos, tanto funcionales como no funcionales, que deberá cumplir el producto final, además de contextualizar sobre a quién va dirigido y por qué esta elección.

Para la realización de la definición funcional se estima una duración de 120 horas.

1.6.3.- Diseño de la skill y aplicación web

En tercer lugar, se diseña y detalla el sistema conforme a los requerimientos definidos en la etapa anterior: estructura, componentes, interfaces, elementos necesarios para el desarrollo...

Para la realización del diseño de la skill y la aplicación web se estima una duración de 120 horas.

1.6.4.- Desarrollo de la skill y aplicación web

En cuarto lugar, una vez definidos entornos de desarrollo, lenguajes de programación elegidos, y demás componentes y tecnologías, se procede al desarrollo del código fuente tanto

de la skill como de la aplicación web hasta lograr el producto final, plenamente funcional y cumpliendo todos los requerimientos. Se trata de la etapa más larga, ya que en ella pueden aparecer múltiples problemas no previstos en la planificación.

Para la realización del desarrollo de la skill y la aplicación web se estima una duración de 560 horas.

1.6.5.- Fase de pruebas

Una vez desarrollado el producto, este debe ser probado. Es una etapa fundamental, ya que es la última comprobación antes de que el cliente o usuario reciba el producto. Es necesario comprobar, entre otras cosas, que todos los componentes funcionan correctamente o que el flujo de funcionamiento es correcto.

Para la fase de pruebas se estima una duración de 96 horas.

1.6.6.- Documentación del proyecto

En último lugar, es necesario realizar la documentación que acompañará al proyecto: técnica, económica, temporal... Es muy importante que todo lo relativo al proyecto esté incluido en la documentación, ya sea información sobre cómo instalar la aplicación o un manual sobre cómo utilizarla, un desglose y explicación de los gastos...

Para la realización de la documentación del proyecto se estima una duración de 160 horas, si bien es cierto que esta tarea ha ido desarrollándose durante todo el tiempo empleado: conforme se avanzaba en una tarea, esto se reflejaba en la documentación.

1.7.- PLANIFICACIÓN TEMPORAL DEL PROYECTO

Aunque en un caso real este tipo de proyecto sea desarrollado por varias personas, en este caso ha sido una sola persona quien ha asumido todos los roles que se plantearán posteriormente (Apartado 1.8). En la Tabla 1.2 se indican las horas estimadas empleadas en cada una de las etapas del proyecto, además de la suma de estas, que se correspondería con el tiempo empleado para el desarrollo del proyecto completo.

Etapa	**Duración**
Estudio del proyecto	64 horas
Definición funcional	120 horas
Diseño de la skill y la aplicación web	120 horas
Desarrollo de la skill y la aplicación web	560 horas
Fase de pruebas	104 horas
Documentación	160 horas
TOTAL (estimado)	1128 horas

Tabla 1.2.- Planificación temporal estimada del proyecto

En esta tabla puede observarse que el total estimado de horas necesario para el desarrollo del proyecto son 1128. Un aspecto a tener en cuenta es que no todas las tareas se han realizado de forma secuencial, sino que algunas han ido desarrollándose paralelamente, como se observa en la Figura 1.15. Esta imagen muestra el Diagrama de Gantt del proyecto, una especie de gráfico de barras que sirve para ilustrar la cronología del mismo. Aquí se puede observar que la documentación del proyecto, si bien se estima un cálculo total de 160 horas empleadas, ha sido una tarea que ha ido desarrollándose de forma paralela a todas las demás, de modo que fuese quedando constancia, por ejemplo, de las conclusiones obtenidas durante la fase de estudio o de la evolución de ciertos aspectos de las aplicaciones.

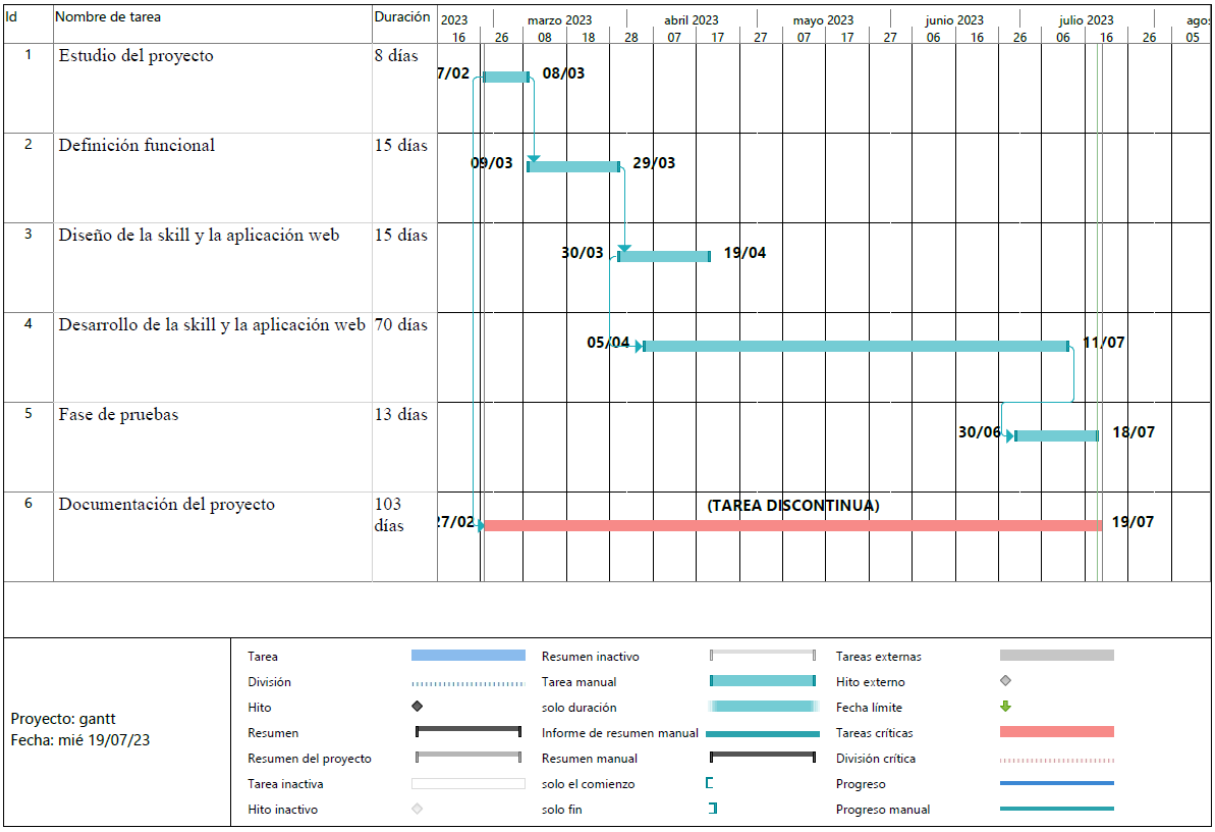

Figura 1.15.- Diagrama de Gantt del proyecto. Fuente: Diseño propio con Microsoft Project

Teniendo en cuenta que se ha planteado una jornada laboral de 8h/día, y que algunas tareas se realizaron paralelamente, la suma estimada de días necesarios para el proyecto asciende a 141.

1.8.- DEFINICIÓN Y REPARTO DE ROLES

Pese a que en el caso que nos ocupa todo el trabajo ha sido realizado por una sola persona, en un caso real habría varios implicados, que cumplirían los siguientes roles:

- Director del proyecto
- Analista
- Arquitecto
- Programador

- Tester
- Ingeniero de control de calidad

Cada uno de estos implicados tendría unas funciones y tareas distintas, descritas en los siguientes apartados.

1.8.1.- Director del proyecto

El director es el máximo responsable del proyecto. Por ello, debe asegurarse de que todo se desarrolle correctamente, cumpliendo los planes y plazos establecidos. Además, es el encargado de administrar los recursos tanto humanos como económicos y tecnológicos. La persona que desempeñe este puesto debe tener un gran poder comunicativo, tanto al cliente como al resto de implicados en el desarrollo del proyecto.

1.8.2.- Analista

El analista es la persona encargada de trabajar junto al cliente para realizar un estudio detallado del problema, desde lo general al detalle, y elaborar las especificaciones y requerimientos que el resto de implicados necesita para el correcto desarrollo del producto. Al igual que el director, el analista debe tener gran capacidad de comunicación para entender los problemas del cliente y trasladárselos de forma correcta al resto de miembros del equipo.

1.8.3.- Arquitecto

El arquitecto de software es el encargado de diseñar la arquitectura del producto basándose en los requisitos funcionales y no funcionales definidos por el analista. Además, debe decidir qué tecnología va a usarse para desarrollar el proyecto, basándose para ello en diversos factores, como pueden ser el coste o las licencias. El arquitecto debe ser una persona

con un alto nivel de conocimiento técnico, de cara a elegir la mejor solución en función de las necesidades del cliente.

1.8.4.- Programador

El programador es el encargado de convertir las especificaciones en código fuente, además de realizar tareas de mantenimiento de código para sea siempre lo más eficiente posible. El programador debe ser una persona con amplios conocimientos técnicos (varios lenguajes de programación, metodologías software, frameworks...), además de tener gran capacidad de trabajo en equipo.

1.8.5.- Tester

El tester es el encargado de realizar las pruebas necesarias para comprobar que el producto desarrollado cumple los requerimientos del cliente, y de informar de fallos, incompatibilidades o comportamientos fuera de lo esperado que se encuentre. El tester debe ser una persona con gran conocimiento técnico, tanto de lenguajes de programación como de metodologías de control de calidad software y de automatización.

1.8.6.- Ingeniero de calidad

El ingeniero de calidad es el encargado de asegurarse de que se cumpla la calidad del producto durante todas las fases de su desarrollo, de forma que se garantice la globalidad del proyecto. El ingeniero de calidad debe tener conocimientos tanto técnicos como de negocio, ya que tiene que poder comprender tanto la parte software como los estándares de calidad.

1.8.7.- Reparto de roles

Una vez definidos los distintos roles, se establece el reparto de tareas acorde a lo establecido en la definición de las etapas del proyecto (Apartado 1.6):

- Estudio del proyecto: Analista
- Definición funcional: Analista
- Diseño de la skill y la aplicación web: Arquitecto
- Desarrollo de la skill y la aplicación web: Desarrollador, arquitecto
- Fase de pruebas: Tester, Ingeniero de Calidad
- Documentación: Director

El director del proyecto, además, se encargará de supervisar todas las etapas, coordinando a los distintos profesionales y asegurándose de que se cumplen los plazos requeridos.

2. Presupuesto

Para el cálculo del presupuesto asociado al proyecto es necesario realizar el cálculo tanto del gasto en hardware y software como en personal. A continuación, se detalla el cálculo desglosado y el cálculo del presupuesto total.

2.1.- EQUIPAMIENTO HARDWARE

Para el desarrollo de la skill y de la aplicación web se ha utilizado un ordenador portátil HP Pavilion 15-eg2008ns, que tiene un precio de mercado de 970,00€.

Además, para las pruebas de la skill se ha utilizado un dispositivo Echo Show 8 de 2.ª generación, con un precio de mercado de 129,99€.

Así, se obtienen unos gastos totales en equipamiento hardware 1.099,99€, tal y como se puede observar en la Tabla 2.1.

Material	Cantidad	Precio/Unidad	Precio Total
HP Pavilion	1	970,00 €	970,00 €
Echo Show 8	1	129,99 €	129,99 €
		TOTAL:	1.099,99 €

Tabla 2.1.- Presupuesto de equipamiento hardware

2.1.1- Amortización equipamiento hardware

De acuerdo a lo que dicta la Agencia Tributaria Española respecto al Impuesto sobre la Renta de las Personas Físicas, el coeficiente lineal máximo aplicable para “Equipos para tratamiento de la información y sistemas y programas informáticos” es del 26% [46]. En este caso se ha utilizado el máximo.

Así, la amortización de equipamiento hardware concurre en:

Presupuesto hardware calculado	1.099,99 €
Porcentaje de amortización aplicado	26%
TOTAL AMORTIZADO	285,99 €

Tabla 2.2.- Presupuesto hardware amortizado

2.2.- EQUIPAMIENTO SOFTWARE

Para el desarrollo tanto de la skill como de la aplicación web, se han utilizado entornos de desarrollo de código libre, por lo que el proyecto no implicaría gastos en equipamiento software.

2.3.- RECURSOS HUMANOS

Como se detalló en el Apartado 1.8, el desarrollo del proyecto requeriría el trabajo de 6 profesionales (director, analista, arquitecto, programador, tester e ingeniero de calidad), y conforme a lo expuesto en los Apartados 1.6 y 1.7 en cuanto a horas trabajadas por cada uno de ellos, se calcula el siguiente gasto en personal:

Rol	Días de trabajo	Horas (8h/día)	Precio/Hora	Precio Total
Director	20	160	35,00 €	5.600,00 €
Analista	23	184	15,00 €	2.760,00 €
Arquitecto	22	176	25,00 €	4.400,00 €
Desarrollador	63	504	20,00 €	10.080,00 €
Tester	10	80	15,00 €	1.200,00 €
Ingeniero de calidad	3	24	15,00 €	360,00 €
TOTAL:	**141**	**1128**	-	**24.400,00 €**

Tabla 2.3.- Presupuesto de Recursos Humanos

2.4.- PRESUPUESTO TOTAL

Con el cálculo realizado en los apartados anteriores de los presupuestos parciales, se realiza el cálculo del presupuesto total. Para este es necesario tener en cuenta el Impuesto sobre el Valor Añadido (IVA), que asciende al 21%.

Presupuesto Hardware.....285,99 €
Presupuesto Software.....0,00 €
Presupuesto Recursos Humanos.....24.400,00 €
Total Costes Estimados.....24.685,99 €

Gastos generales (13%).....3.209,18 €
Beneficio industrial (6%).....1.481,16 €

Total sin IVA.....29.376,33 €
IVA (21%).....6.169,03 €

TOTAL ESTIMADO.....35.545,36 €

Así, el coste total estimado para la realización del proyecto es de treinta y cinco mil quinientos cuarenta y cinco euros con treinta y seis céntimos.

3. Documentos técnicos

3.1.- REQUISITOS

Durante la etapa de definición funcional se definieron los requisitos funcionales y no funcionales que debían cumplir la skill y la aplicación web. Estos requisitos se detallan a continuación.

3.1.1.- Requisitos funcionales

Los requisitos funcionales describen la actividad que debe realizar nuestra aplicación cuando los usuarios interactúan con ella para satisfacer sus necesidades y expectativas. En otras palabras, definen las funcionalidades y características específicas.

RF-1. Skill HablaConmigo

Identificador	Descripción
RF-1.1	La skill permitirá al usuario registrarse utilizando para ello un número de identificación personal elegido por este, y añadiendo su nombre y edad.
RF-1.2	La skill permitirá al usuario iniciar sesión utilizando su identificador.
RF-1.3	La skill ofrecerá seis juegos disponibles, entre los que el usuario podrá elegir el que quiera.
RF-1.4	La skill ofrecerá distintos niveles de dificultad para cada juego, permitiendo al usuario elegir el que mejor se adapte a sus necesidades.
RF-1.5	La skill permitirá al usuario cambiar de nivel mientras juega a un juego determinado.
RF-1.6	La skill permitirá al usuario cambiar de juego durante la sesión.
RF-1.7	La skill permitirá al usuario trabajar la conciencia silábica a través de los juegos de *Ordena las Sílabas*, *Cuenta las Sílabas* y *La Sílaba Oculta*.
RF-1.8	La skill permitirá al usuario trabajar su lenguaje a través de los juegos de *Las Adivinanzas* y *Los Antónimos.*
RF-1.9	La skill permitirá al usuario comprobar su pronunciación de las letras L, S, R y D a través del juego *Repite la Palabra.*

RF-1.10	La skill proporcionará retroalimentación a los usuarios, de forma que sepan si su respuesta ha sido correcta o incorrecta, además del número de aciertos correspondientes al nivel que estén jugando.

Tabla 3.1.- Requisitos funcionales de la skill HablaConmigo

RF-2. Aplicación web

Identificador	Descripción
RF-2.1	La aplicación web permitirá al logopeda consultar el número de aciertos y errores y el tiempo medio de respuesta del usuario de un nivel y juego determinado.
RF-2.2	La aplicación web permitirá elegir si se desean consultar todos los datos de juego del usuario, los datos a partir de una fecha, los datos hasta una fecha o los datos en un intervalo de tiempo.
RF-2.3	La aplicación web permitirá al logopeda añadir nuevas entradas a los juegos y niveles ya creados.

Tabla 3.2.- Requisitos funcionales de la aplicación web

3.1.2.- Requisitos no funcionales

Los requisitos no funcionales no determinan funcionalidades del sistema, sino que hacen referencia a cualidades, restricciones y características del software, como pueden ser la escalabilidad, el rendimiento o la seguridad, por ejemplo.

RNF-1. Skill HablaConmigo

Identificador	Descripción
RNF-1.1	La skill está implementada en español, por lo que el usuario debe conocer el idioma para poder utilizarla.
RNF-1.2	La skill debe ejecutarse en algún dispositivo de la familia Amazon Alexa, por lo que el usuario deberá poseer una para poder utilizarla.
RNF-1.3	La skill debe estar instalada en el dispositivo para poder utilizarla.
RNF-1.4	Alexa requiere conexión a Internet para funcionar correctamente, por lo que es necesaria esta conexión para poder utilizarla.
RNF-1.5	La skill debe ser accesible e intuitiva para que los niños puedan usarla sin necesidad de conocimientos técnicos.

RNF-1.6	La skill debe ir acompañada de un manual de usuario para que este pueda utilizarla correctamente

Tabla 3.3.- Requisitos no funcionales de la skill HablaConmigo

RNF-2. Aplicación web

Identificador	Descripción
RNF-2.1	La aplicación web está implementada en español, por lo que el logopeda debe conocer el idioma para poder utilizarla.
RNF-2.2	La aplicación web debe ejecutarse en un ordenador, por lo que el logopeda deberá poseer uno para poder utilizarla.
RNF-2.3	La conexión con la API REST requiere Internet para funcionar correctamente, por lo que es necesaria esta conexión para poder utilizar la aplicación web.
RNF-2.4	La aplicación web tendrá una interfaz de usuario sencilla e intuitiva, que permita al logopeda su uso sin necesidad de conocimientos técnicos.

Tabla 3.4.- Requisitos no funcionales de la aplicación web

3.2.- DISEÑO

En este apartado se detallará todo lo relativo al diseño de la solución: la arquitectura del sistema, qué entornos de desarrollo se han utilizado, que hardware y software se ha empleado y sus características principales.

3.2.1.- Arquitectura del sistema

En la Figura 3.1 se observa el esquema de la arquitectura del sistema, compuesta por el dispositivo que ejecuta la skill, una API REST, la base de datos y una aplicación web.

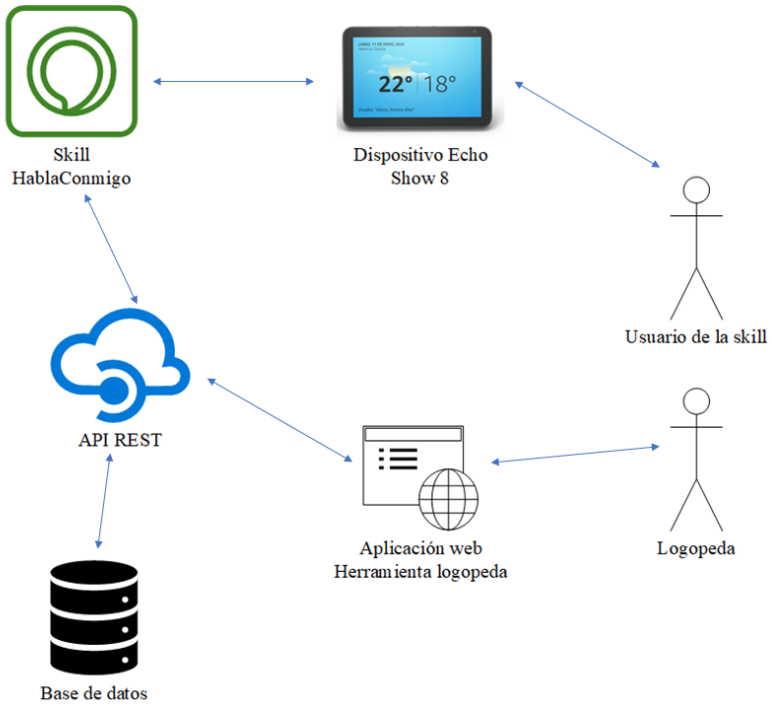

Figura 3.1.- Esquema de la arquitectura del sistema. Fuente: Diseño propio

El sistema está basado en una API REST (*REpresentational State Transfer* - Transferencia de Estado Representacional), elegida por su escalabilidad y flexibilidad: REST efectúa una separación total entre cliente y servidor, lo que optimiza las interacciones entre ellos y facilita que evolucionen de forma independiente. Esta API REST implementa un CRUD (*Create, Read, Update and Delete* – Crear, Leer, Actualizar y Borrar) parcial, ya que en este caso solo permite crear objetos y hacer consultas.

Esta API REST será quien se encargue de acceder a la base de datos del sistema, de tipo MySQL. En esta base de datos se almacenarán los datos correspondientes a los usuarios, los juegos y los resultados que obtienen los usuarios. Para acceder a ella, la API REST utilizará los correspondientes métodos de HTTP (POST para crear objetos y GET para hacer consultas). Los datos son almacenados y también devueltos al usuario en formato JSON.

Esta API interactúa también con la skill HablaConmigo y con la aplicación web que la acompaña. Por un lado, está la skill, diseñada para utilizarse (a priori) en dispositivos Amazon Echo Show 8. El niño interactuará con la skill a través del dispositivo para jugar, y la skill, a través de la API, accederá a la base de datos. En ella registrará a los usuarios, de quienes

podrá obtener luego sus datos a modo de "inicio de sesión". Además, solicitará los datos para que el niño juegue y guardará sus resultados para su posterior consulta por parte del logopeda. Por otro lado, la API interactúa con la aplicación web, una herramienta de ayuda para el logopeda. El logopeda accederá a esta aplicación para registrar nuevas entradas de juegos o para realizar consultas sobre los resultados de sus pacientes.

3.2.2.- Entorno de desarrollo

Para el desarrollo del sistema se han utilizado dos entornos de desarrollo distintos: Alexa Developer Console (para el desarrollo de la skill) y Visual Studio Code (para la aplicación web). Ambos entornos se explican a continuación.

3.2.2.1.- Alexa Developer Console

En el desarrollo de la skill HablaConmigo se utilizó Alexa Developer Console. Para hablar de ella, es necesario mencionar Alexa Skills Kit (ASK) [47]. Este kit de Amazon ofrece un conjunto de herramientas y recursos para desarrollar skills de Alexa. Una de estas herramientas es la Alexa Developer Console, una plataforma en línea para el desarrollo de skills. Se trata de un entorno muy sencillo e intuitivo, con gran cantidad de funcionalidades (desarrollo de código, personalización de la vista en pantalla, definición de interacciones de voz, depuración...). Además, incluye un apartado de prueba que permite simular el funcionamiento de la aplicación seleccionando el dispositivo Amazon Alexa deseado.

Antes de comenzar a desarrollar una skill hay ciertos conceptos del modelo de interacción de voz que se deben conocer:

- Palabra de activación: Indica a Alexa que debe comenzar a escuchar.

- Palabra de lanzamiento: Palabra de acción de transición que le indica a Alexa que probablemente irá seguida de la invocación de una habilidad. Ejemplos de estas palabras son “decir”, “preguntar”, “abrir”, “lanzar” o “usar”.

- Nombre de invocación: Nombre que utiliza como inicio de interacción la skill a la que se desea acceder.

- *Utterance* (enunciado): Es la solicitud que realiza el usuario, y puede tener múltiples resultados: invocar una habilidad, confirmar acciones...

- *Prompt*: Respuesta de Alexa que sirve como guía para el usuario durante el uso de la skill. Pueden utilizarse por parte de Alexa para solicitar información adicional al usuario o proporcionarle instrucciones, entre otras cosas.

- *Intent*: Acción que corresponde con la petición hecha por el usuario. Un intent puede tener opcionalmente argumentos con un valor indicado por el usuario. Estos argumentos se denominan “slots”.

- Valor del *slot*: Valor de entrada que proporciona el usuario en un intent, y que permite a Alexa comprender lo que desea hacer el usuario.

Una vez comprendido esto, se puede comenzar a desarrollar la skill con la Alexa Developer Console. Al tratarse de una aplicación web no es necesaria ninguna instalación. El primer paso para su uso sería acceder a la página web donde se aloja este servicio (https://developer.amazon.com/alexa/console/ask) y crear una cuenta. Una vez que la cuenta está creada y la sesión iniciada aparece la siguiente pantalla:

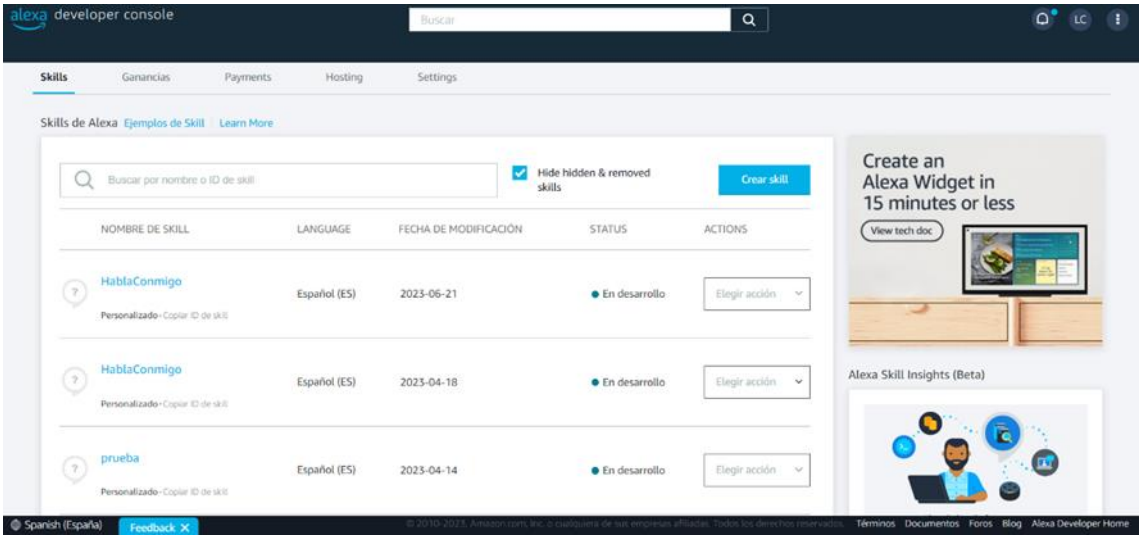

Figura 3.2.- Página de inicio Alexa Developer Console. Fuente: Diseño propio

En la parte superior aparecen 5 pestañas:

- *Skills*: Muestra las skills asociadas a tu cuenta, con su correspondiente configuración, y permite realizar diferentes acciones sobre ellas (probarlas, distribuirlas, eliminarlas...). Además, también es la pestaña en la que se pueden crear nuevas skills.

- *Earnings* (Ganancias): Permite ver las ganancias obtenidas con las skills publicadas. También permite filtrar por mercado y mes, y generar un CSV con los resultados obtenidos que se puede descargar.

- *Payments*: Muestra información relacionada con los pagos dentro de las skills, y permite a los desarrolladores monetizar su trabajo, con acciones como configurar precios u opciones de pago.

- *Hosting*: Muestra la información relacionada con el alojamiento y configuración de la infraestructura necesaria para la skill, como número de solicitudes, uso mensual o límite mensual. Al igual que en ganancias, también se puede filtrar la información por mes.

- *Settings*: Esta pestaña permite, principalmente, todo lo relacionado con gestionar accesos y usuarios, como, por ejemplo, añadir o quitar permisos a un usuario para una o varias skills. También permite consultar el ID del cliente y de proveedor asociado a la cuenta.

Como se mencionó con anterioridad de la pestaña *Skills*, es aquí donde se puede crear una nueva. Para ello hay que hacer clic en el botón azul que indica "Crear skill", que abrirá la pestaña de la Figura 3.3. Aquí se introduce el nombre que se le quiere dar a la skill en el campo *Skill name*, y se selecciona en el desplegable de la parte inferior de la página el idioma en el que se interactuará con la skill (posteriormente se pueden añadir más idiomas).

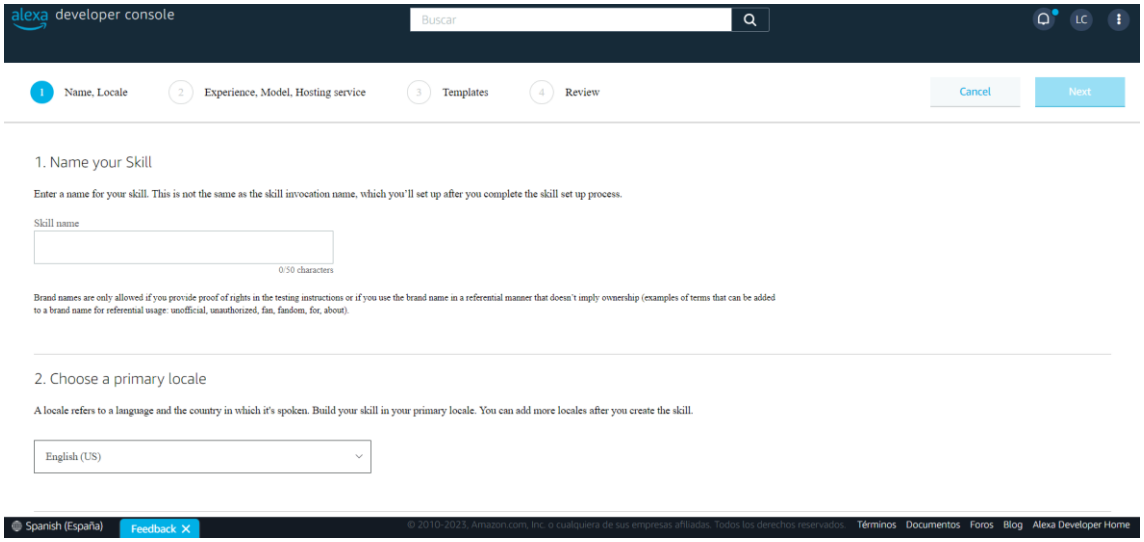

Figura 3.3.- Primera pantalla de creación de skill. Fuente: Diseño propio

Una vez introducidos estos datos, los botones de la parte superior de la pantalla se activarán y se podrá pinchar en *Next* para pasar a la siguiente pantalla (Figuras 3.4 y 3.5). Aquí se debe indicar, en primer lugar, qué tipo de skill se pretende desarrollar (una skill para jugar, una relacionada con películas, una que dé las noticias...). A continuación, se selecciona *Custom* (Personalizado) como modelo para la skill, y finalmente, se elige el método que se utilizará para alojar los recursos del back-end de la skill, que puede ser Node.js o Python (en el caso que nos ocupa, Node.js). Hay que seleccionar también la región de alojamiento más cercana, con el fin de reducir la latencia en el servicio. Con estas selecciones ya se puede continuar el proceso de creación.

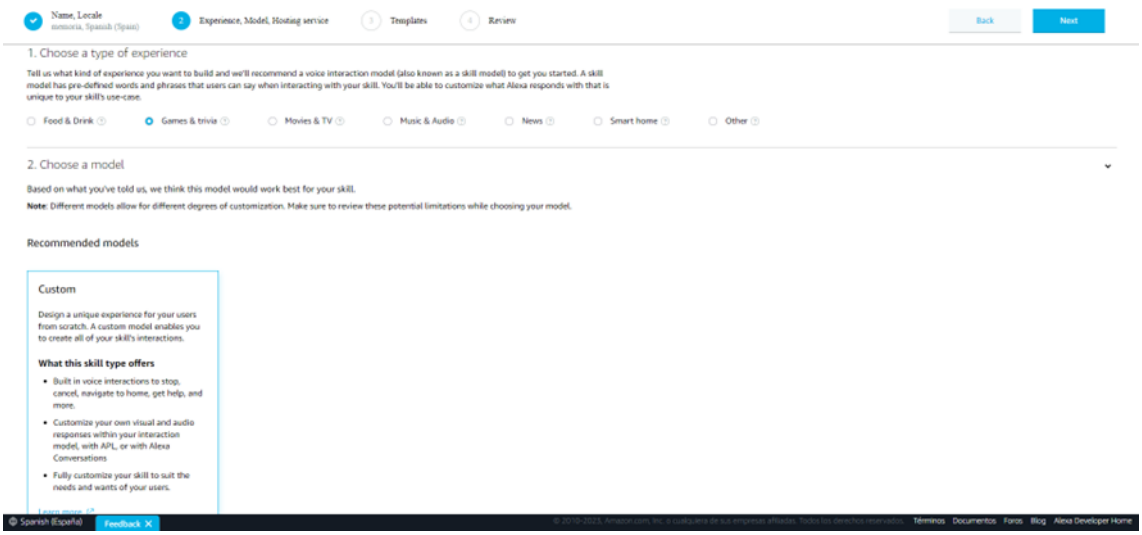

Figura 3.4.- Segunda pantalla de creación de la skill (Parte 1). Fuente: Diseño propio

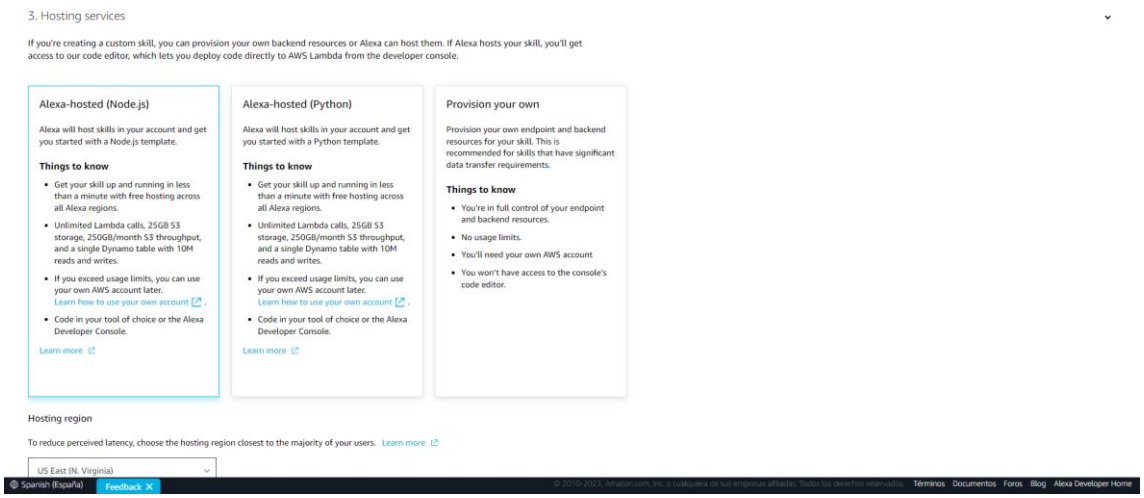

Figura 3.5.- Segunda pantalla de creación de la skill (Parte 2). Fuente: Diseño propio

La tercera pantalla (Figura 3.6) permite elegir una plantilla para añadir a la skill, aunque en este caso, no se utiliza ninguna y se empieza desde cero (*Start from Scratch*).

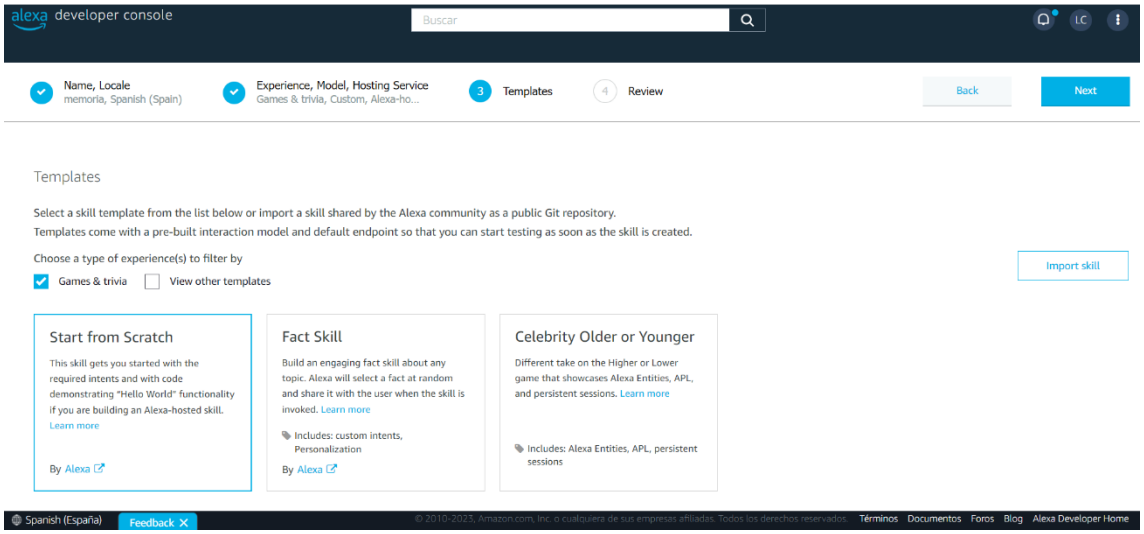

Figura 3.6.- Tercera pantalla de creación de la skill. Fuente: Diseño propio

Con todas estas características seleccionadas se llega a la cuarta pantalla (Figura 3.7), donde se muestra un resumen de las elecciones realizadas por si ha habido algún error y es necesario modificar alguna.

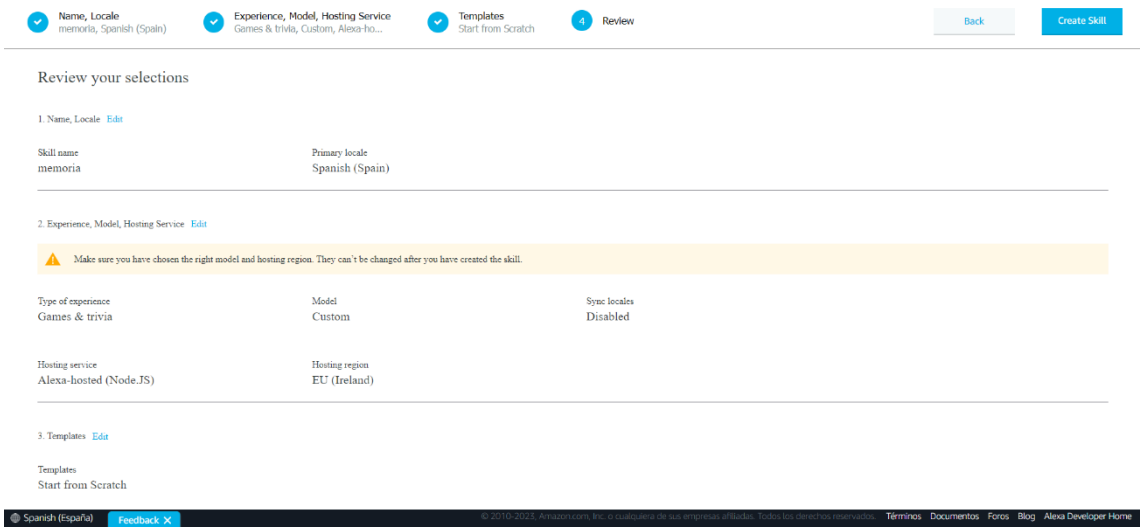

Figura 3.7.- Cuarta pantalla de creación de la skill. Fuente: Diseño propio

Una vez comprobado que todo está correcto, se hace clic en *Create Skill* en la esquina superior derecha para crear la skill.

Cuando Alexa Developer Console termina de procesar la solicitud de creación, muestra la pantalla principal de la skill:

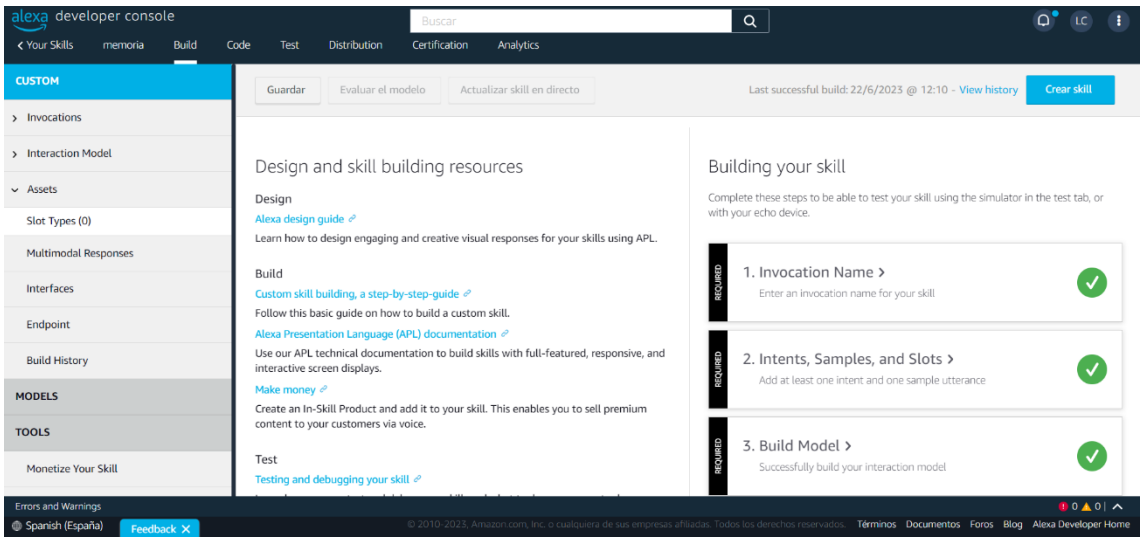

Figura 3.8.- Pantalla principal de la skill. Fuente: Diseño propio

El último paso del proceso de creación, que es, a su vez, el primer paso del de configuración, es modificar el nombre de invocación de la skill. Para ello, *Build >> Invocations >> Skill Invocation Name* (Figura 3.9), y se introduce en el campo *Skill Invocation Name* la forma en la que se desea "llamar" a la skill cuando se interactúe con Alexa (en este caso, *habla conmigo*). Es importante guardar esta configuración y, una vez guardada, hacer clic en *Crear skill* (*Build skill*) para aplicar los cambios realizados.

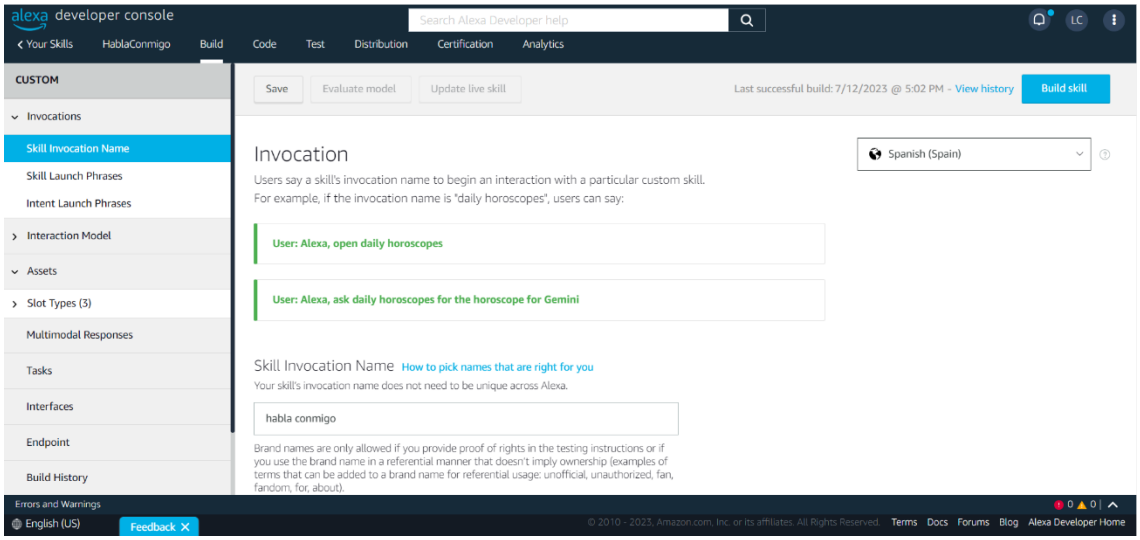

Figura 3.9.- Cambio del nombre de invocación de la skill. Fuente: Diseño propio

En la Figura se mostraba la "página principal" de la skill. En esta imagen se pueden observar seis pestañas distintas: *Build*, *Code*, *Test*, *Distribution*, *Certification* y *Analytics*.

La "página principal" de la skill se corresponde con la pestaña *Build*. Desde esta pestaña se puede acceder a los distintos elementos de la skill, crearlos, modificarlos, y realizar, en general, configuraciones de la skill. Estos elementos son accesibles desde el menú de la parte izquierda de la pantalla (Figura 3.10).

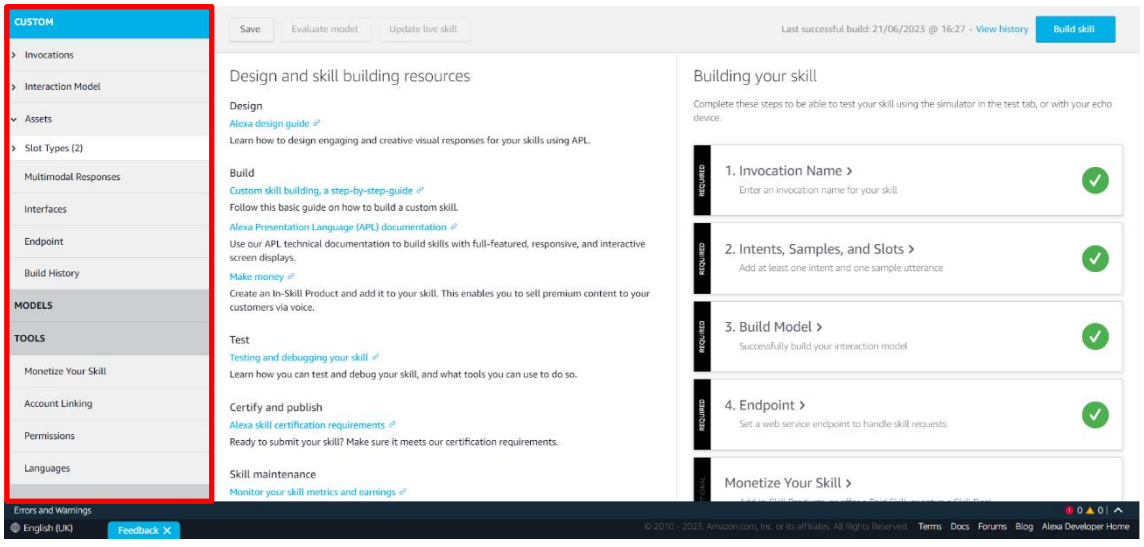

Figura 3.10.- Pestaña Build. Fuente: Diseño propio

En primer lugar, *Invocations* consta de tres apartados, *Skill Invocation Name* (para cambiar el nombre de invocación de la skill, ya utilizado), *Skill Launch Phrases* (para añadir frases que invoquen una skill sin indicar el nombre de esta) e *Intent Launch Phrases* (para añadir frases que permitan interactuar con intents específicos de una skill sin necesidad de indicar el nombre de esta). Estos dos últimos apartados solo pueden ser modificados si la skill está publicada.

En segundo lugar, se encuentra *Interaction Model*, que consta de cuatro apartados: *Intents*, *Annotation Sets* (conjunto de datos que permiten entrenar y mejora el modelo de lenguaje de la skill), *Intent History* (proporciona un registro de las interacciones de los usuarios con la skill) y *JSON Editor* (herramienta para edición y visualización de archivos JSON). Como se explicó anteriormente, los intents representan la acción que el usuario desea lograr, y se definen en la pestaña *Intents*. Al entrar por primera vez, se puede ver que aparecen cinco intents predefinidos (Figura 3.11), que deben mantenerse para el correcto funcionamiento de la skill (pueden editarse sus funciones).

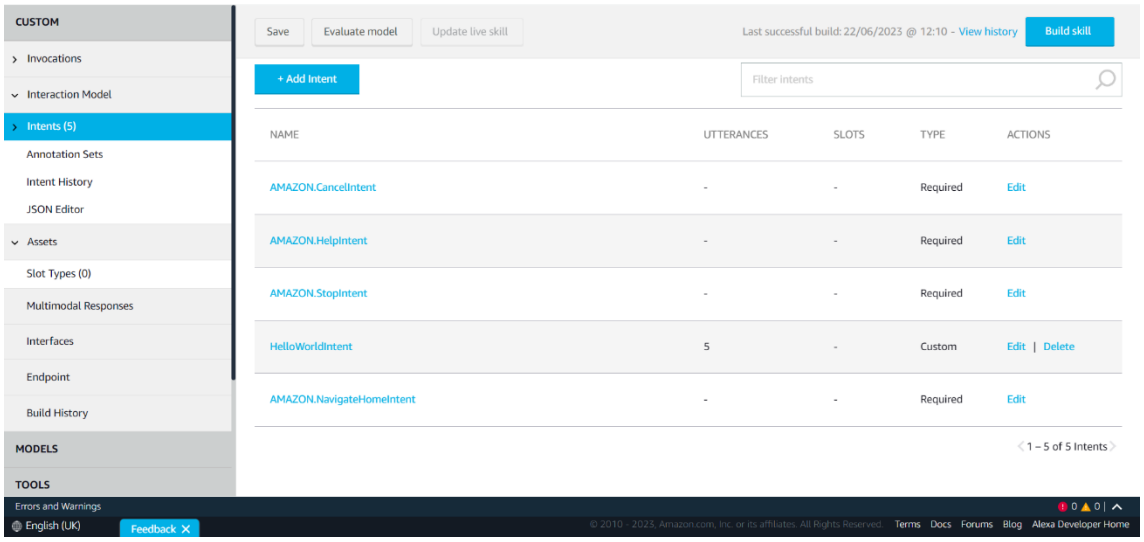

Figura 3.11.- Intents predefinidos. Fuente: Diseño propio

Para crear uno nuevo, se hace clic en el botón *Add Intent*, que muestra la pantalla de la Figura. Alexa Developer Console permite crear intents personalizados, adaptados a las funcionalidades que se desean incorporar a la skill, pero también permite añadir intents ya definidos (por ejemplo, *AMAZON.YesIntent* o *AMAZON.NoIntent*). En el caso de querer

añadir un intent, personalizado, se introduce el nombre en *Enter name for intent* y se hace clic en *Create custom intent*.

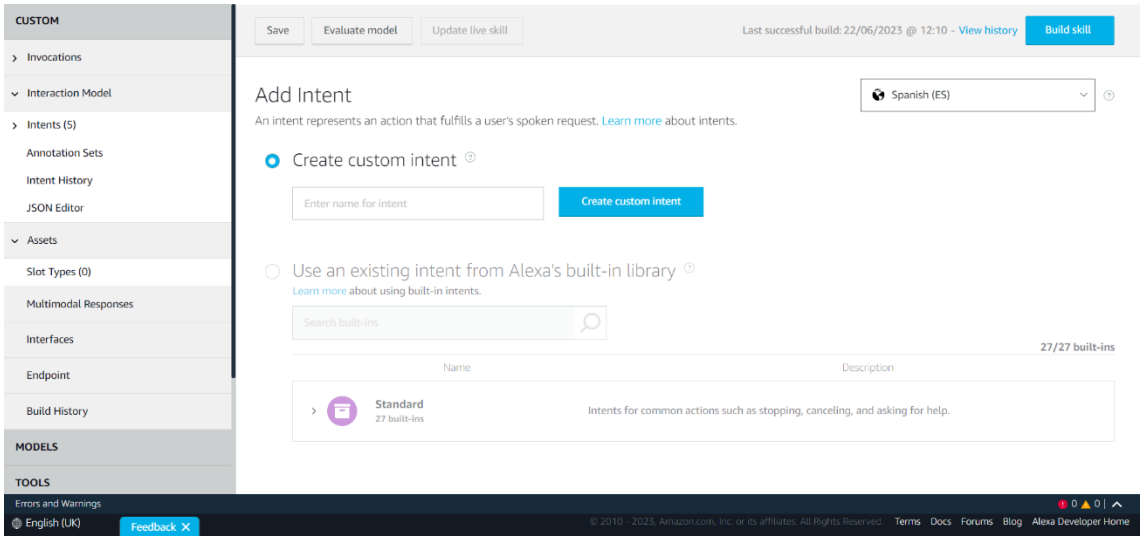

Figura 3.12.- Pantalla de adición de intent a la skill. Fuente: Diseño propio

Una vez definido el nombre del intent ya se puede configurar (Figura 3.13). En *Sample Utterances* se incluyen aquellas frases que cuando el usuario las diga, invocarán el intent (cuantas más frases se incluyan, más precisión tendrá Alexa). Estas *utterances* pueden incluir entre corchetes lo que se denomina *Intent Slots*, que permite capturar información dicha por el usuario. Es necesario asignarle un tipo a cada intent slot definido.

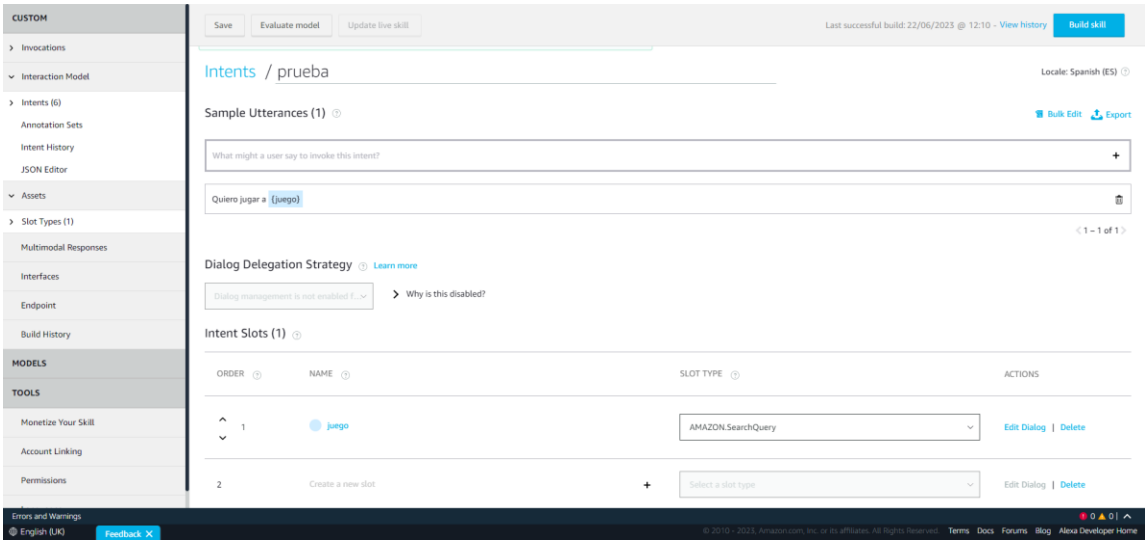

Figura 3.13.- Configuración de intent personalizado. Fuente: Diseño propio

Para hacer obligatorio un valor para el slot, se hace clic en él. Una vez hecho esto se muestra una nueva pantalla (Figura 3.14). Aquí se marca *Slot Filling* y se introducen *prompts*, que serán las frases que diga Alexa en el caso de recibir el intent pero no un valor para su correspondiente slot.

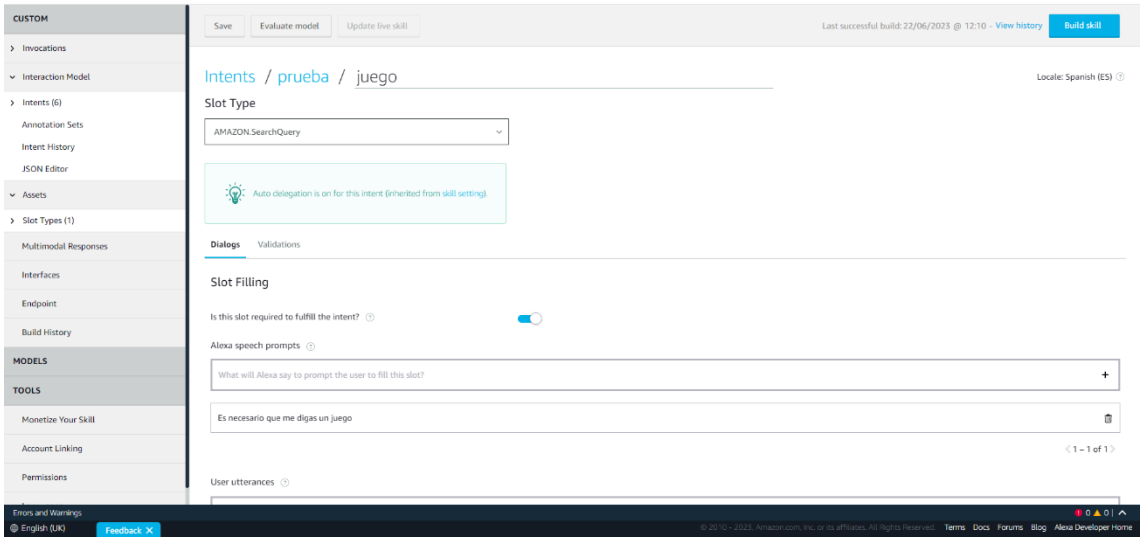

Figura 3.14.- Obligación de aportar valor al slot. Fuente: Diseño propio

Finalmente, se vuelve a la pantalla de edición del intent (Figura 3.15) y se activa la auto delegación. Así, la skill podrá manejar respuestas incompletas.

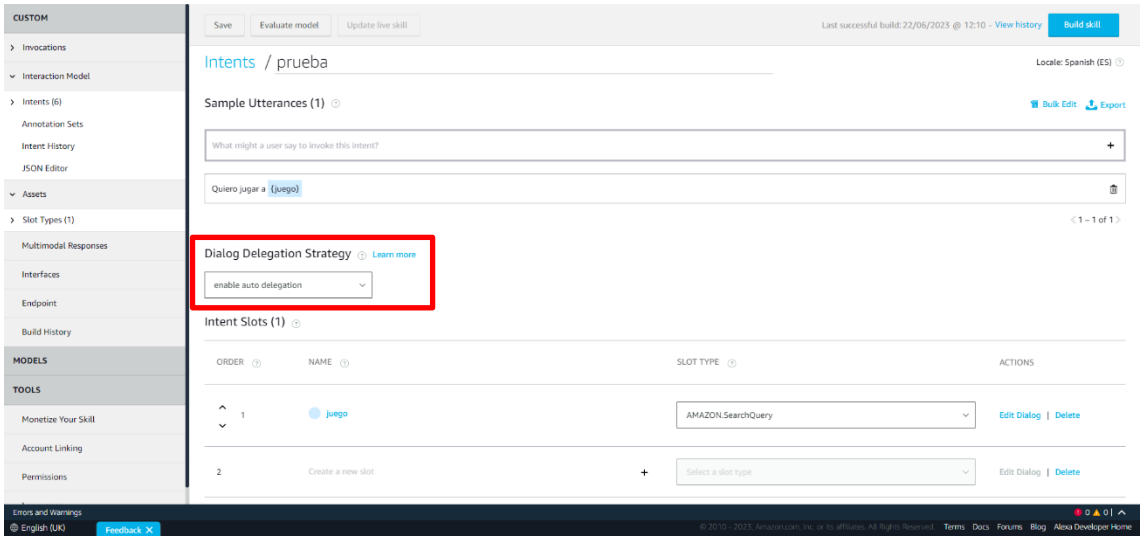

Figura 3.15.- Activar autodelegación. Fuente: Diseño propio

El siguiente apartado de *Build* sería *Assets*, que solo tiene un subapartado: *Slot Types*. Aquí se mostrarán los diferentes tipos de dato que se le han asignado a los *Slot Intents*, además de permitir ver qué *Slots* tienen cada tipo y a qué *Intent* están asociados.

A continuación, se encuentra *Multimodal Responses*, que permite añadir ayudas visuales a la skill. Para poder hacerlo es necesario ir a la siguiente pestaña, *Interfaces*, y activar el Lenguaje de Presentación de Alexa (APL) para los distintos dispositivos que se requiera. Al igual que APL, otras interfaces, como la reproducción de audio o vídeo se pueden habilitar en esta pestaña.

Finalmente, se encuentran *Endpoint*, que permite seleccionar el endpoint de la skill (ARN de AWS Lambda o HTTPS) y *Build History*, que muestra el historial de compilaciones de la skill.

Por último, otras funciones que permite la pestaña de *Build* son añadir o eliminar modelos de la skill, monetizarla, modificar los permisos o realizar configuraciones de los idiomas que utiliza.

Pasando a la siguiente pestaña de la barra superior, *Code*, se encuentra lo siguiente:

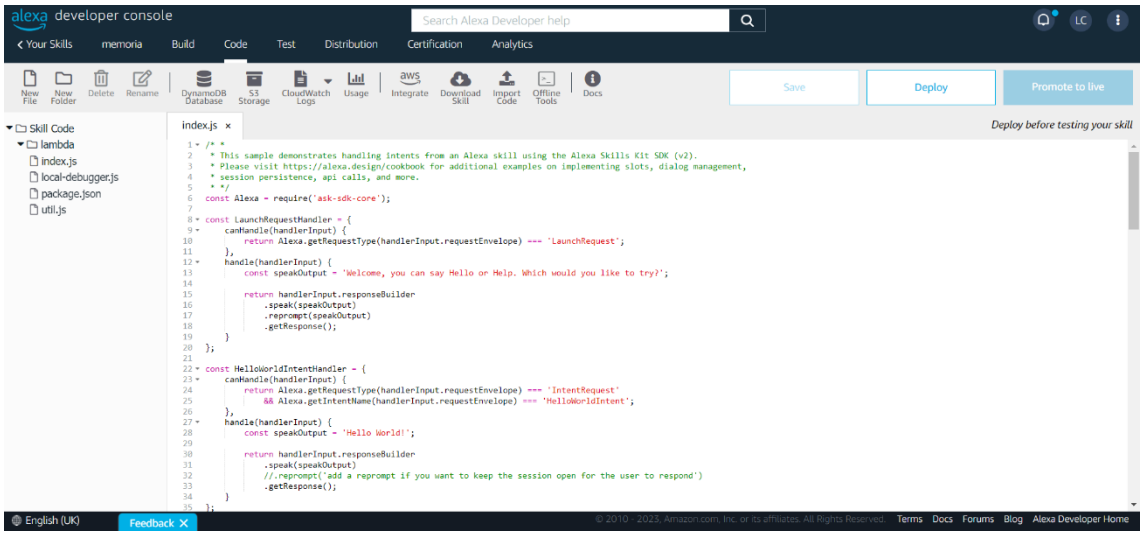

Figura 3.16.- Primera vista de la pestaña Code. Fuente: Diseño propio

En esta pestaña se puede ver que ya hay ciertos archivos creados:

- index.js: Aquí se incluye el código principal de la skill. Aparece ya desarrollado el código de los intents que anteriormente se mencionó que se creaban automáticamente.

- local-debugger.js: Incluye un script para depurar la skill de forma local.

- package.json: Archivo que incluye las dependencias utilizadas en el proyecto, autor, licencia, descripción de la skill...

- util.js: Código que proporciona una URL para acceder temporalmente a recursos almacenados en S3.

En la barra superior se encuentra, en primer lugar, todo lo relativo a archivos: crear un archivo, crear una carpeta, y eliminar y renombrar un archivo o carpeta. A continuación, se encuentran *DynamoDB* y *S3 Storage*, servicios complementarios que se utilizan para almacenar y recuperar datos (DynamoDB para datos estructurados y persistentes y S3 para archivos multimedia, fundamentalmente), y junto con ellos, *CloudWatch Logs*, que permite, entre otras cosas, registrar eventos, alarmas, y depurar registros, y *Usage*, que muestra las estadísticas de uso (enlace a la pestaña *Hosting*, explicada al principio). Finalmente, se encuentran *Integrate* (permite, principalmente, integrar la skill con otros servicios), *Download Skill* (permite descargar un ZIP con todos los archivos de la skill), *Import Code* (función contraria a *Download Skill*, permite importar un ZIP con el código fuente de la skill), *Offline Tools* (proporciona herramientas para desarrollo y depuración de la skill en entorno local sin conexión a Internet) y *Docs* (proporciona documentación y referencias para ayudar en el desarrollo de la skill).

En esta pestaña será donde se desarrolle el código de la skill. Es importante recordar guardar el progreso y compilarlo para su correcto funcionamiento (botones *Save* y *Deploy* de la esquina superior derecha, Figura 3.16).

La siguiente pestaña sería *Test*. Esta es una de las mayores ventajas de utilizar Alexa Developer Console para el desarrollo de la skill. *Test* permite simular el funcionamiento de la skill desarrollada es distintos dispositivos Alexa. Así, se puede desarrollar y probar una skill sin necesidad de poseer un dispositivo físico.

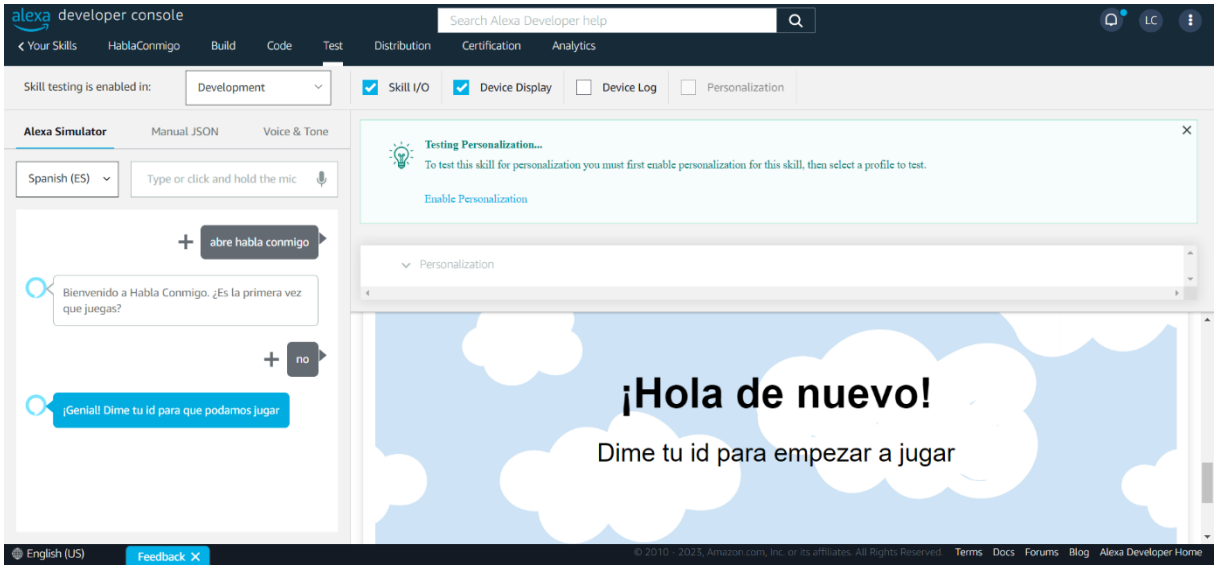

Figura 3.17.- Pestaña Test. Fuente: Diseño propio

Una vez completado el desarrollo de la aplicación, estaría lista para su distribución, que se haría en la pestaña *Distribution* (Figura 3.18). Aquí se deben rellenar los distintos campos con información de la skill (descripción de la skill, logo que debe mostrar, categoría...).

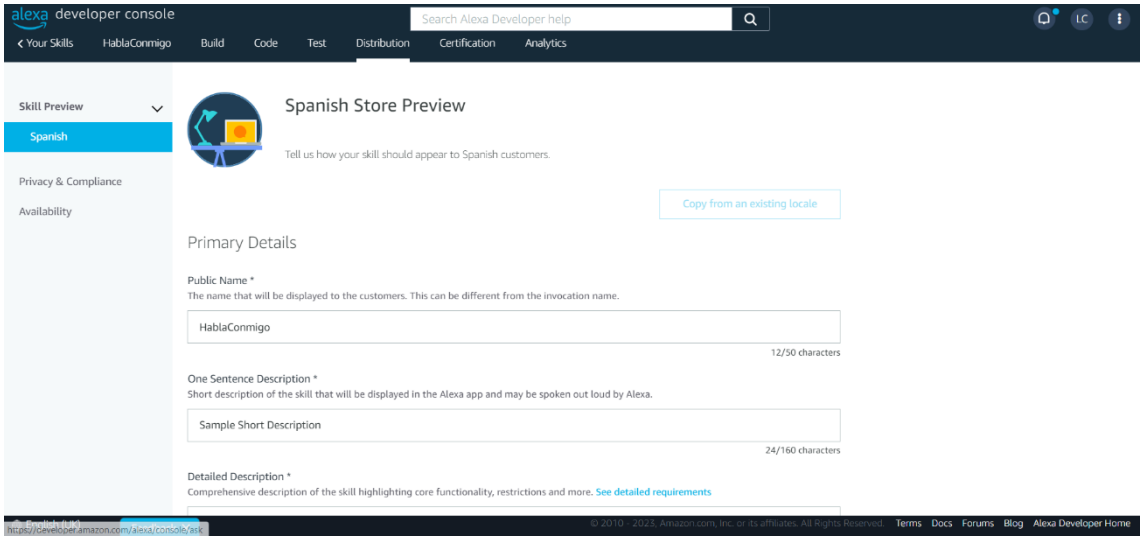

Figura 3.18.- Pestaña Distribution. Fuente: Diseño propio

En la parte de abajo de esta pestaña se encuentra el botón *Save and Continue*, en el que se debe clicar una vez rellenada toda la información sobre la skill, y que mostrará una vista previa de cómo se vería la skill en la tienda de skills de Alexa. Es necesario también certificar que la skill cumple los requisitos de cumplimiento de exportaciones en la pestaña *Privacy & Compliance* (menú izquierdo), así como configurar quién debería tener acceso a la skill y dónde le gustaría que estuviera disponible, todo esto en la pestaña *Availability* (también del menú izquierdo).

La penúltima pestaña, *Certification*, muestra en un principio, la página *Validation* (Figura 3.19), en la que se incluyen los problemas que requieren arreglo para que la skill "supere" la certificación.

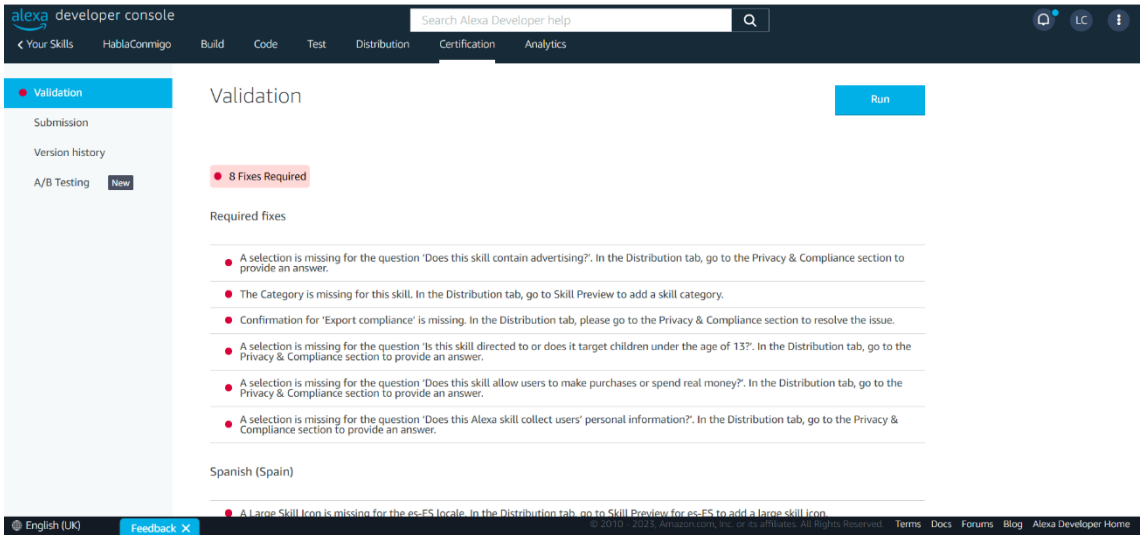

Figura 3.19.- Pestaña Certification. Fuente: Diseño propio

Una vez solventados los errores y revisadas las listas de verificación de envío (pestaña *Submission* del menú izquierdo), solo habría que hacer clic en *Run* (esquina superior derecha) para enviar la skill y que se revise su certificación.

El último punto de la barra superior de Alexa Developer Console es *Analytics* (Figura 3.20). Aquí se muestran datos como la duración promedio de la sesión, número de invocaciones de la skill o errores que experimenta la skill, entre otras cosas. Además, permite filtrar estos datos para observarlos durante un periodo de tiempo determinado, o ver los datos asociados a un intent determinado.

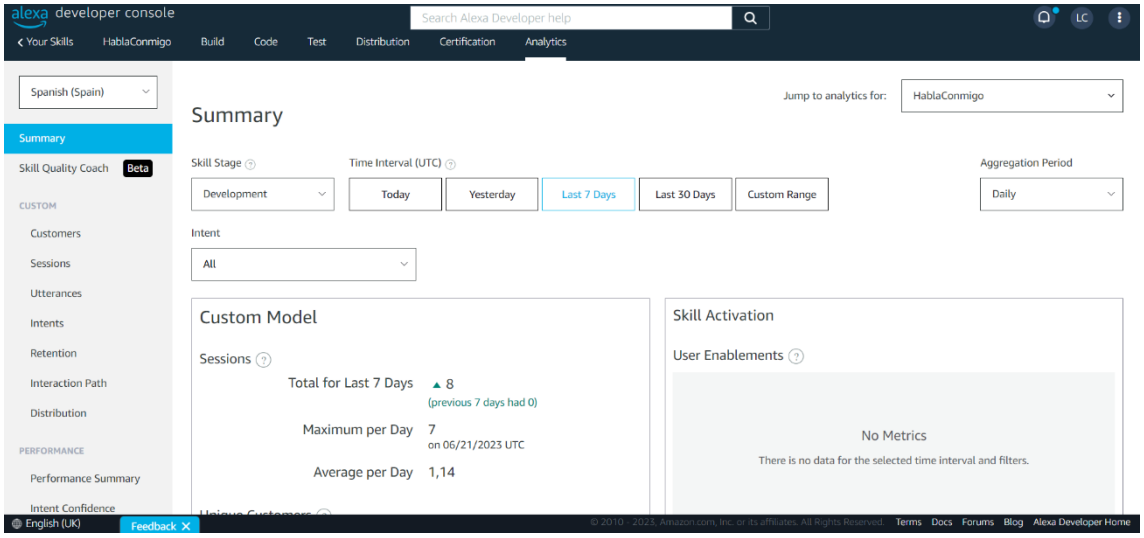

Figura 3.20.- Pestaña Analytics. Fuente: Diseño propio

3.2.2.2.- Visual Studio Code

Para el desarrollo de la herramienta de ayuda para logopedas de HablaConmigo, se utilizó Visual Studio Code (VS Code) [48], un editor de código fuente, de código abierto y de escritorio, desarrollado por Microsoft y disponible para Windows, Linux y macOS. Además, desde 2021 dispone también de una versión web. Es compatible con JavaScript, TypeScript y Node.js, además de disponer de extensiones para poder utilizar muchos otros lenguajes, como C++, C#, Java, Python o .NET. Esto permite trabajar con múltiples tecnologías y plataformas sin necesidad de utilizar distintos entornos de desarrollo. Junto con las extensiones para poder escribir código en otros lenguajes, son múltiples las que se pueden instalar para facilitar la tarea de programar. Un ejemplo de esta funcionalidad es Live Server [49], una extensión muy utilizada por desarrolladores de Front-End. Lo que hace es crear un servidor local conectado con la página que está siendo desarrollada que se actualiza automáticamente con cada cambio realizado, agilizando en gran medida el trabajo.

Una característica muy útil de VS Code es IntelliSense [50]. Este término engloba varias funciones muy útiles a la hora de desarrollar código, como pueden ser la “finalización de código”, la “asistencia de contenido” o las “sugerencias de código”. Además, Git está

integrado en VS Code. Esto permite clonar un repositorio fácilmente en VS Code, al igual que trabajar con distintas versiones y editar código en vivo y colaboración.

Todas estas características son las causantes de su gran popularidad. Según la encuesta realizada en mayo de 2021 por Stack Overflow a unos 80000 desarrolladores, VS Code es, y con mucha diferencia, el entorno de desarrollo más utilizado (Figura 3.21).

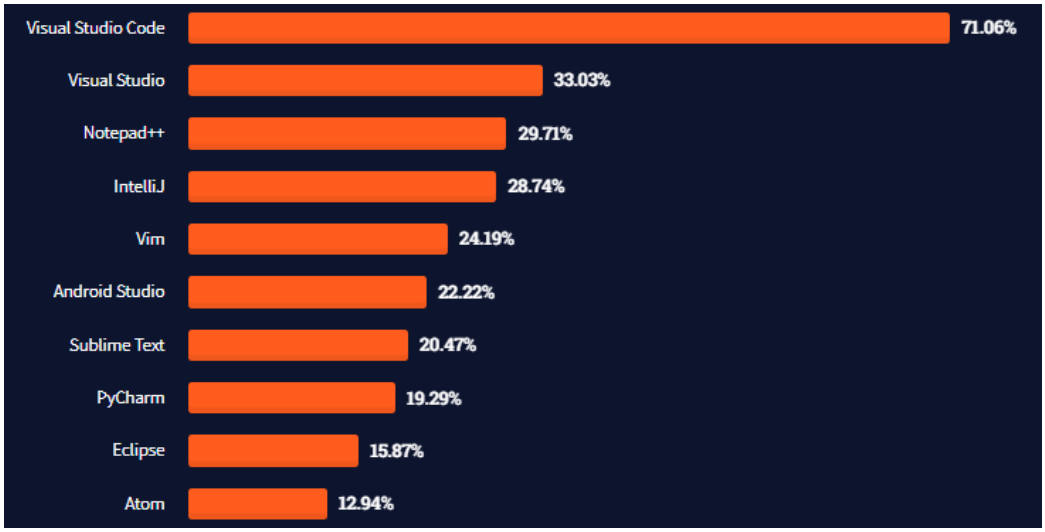

Figura 3.21.- Top 10 entornos de desarrollo elegidos por los usuarios. Fuente: Stack Overflow *[51]*

Si además se consultan los resultados de la encuesta para desarrolladores profesionales, en la que participaron unos 55000 usuarios, puede observarse que VS Code sigue siendo la opción preferida con una amplia ventaja:

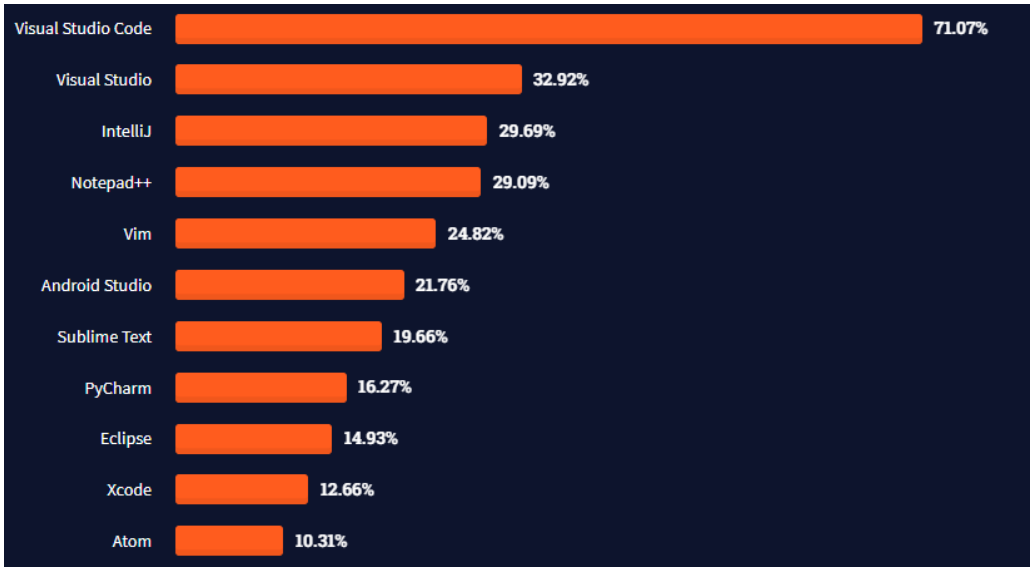

Figura 3.22.- Top 10 entornos de desarrollo elegidos por los desarrolladores profesionales. Fuente: Stack Overflow *[51]*

El primer paso para poder trabajar con Visual Studio Code es su instalación. Para descargar el archivo de instalación simplemente habría que entrar en la página web oficial (https://code.visualstudio.com) e ir a la pestaña de *Descargas*. En esta página se mostrarían las distintas versiones disponibles para los distintos Sistemas Operativos que soportan VS Code (Figura 3.23). Solo habría que seleccionar el deseado e instalarlo siguiendo el proceso habitual (si se desea utilizar la versión web, solo habría que ir a la página https://vscode.dev/).

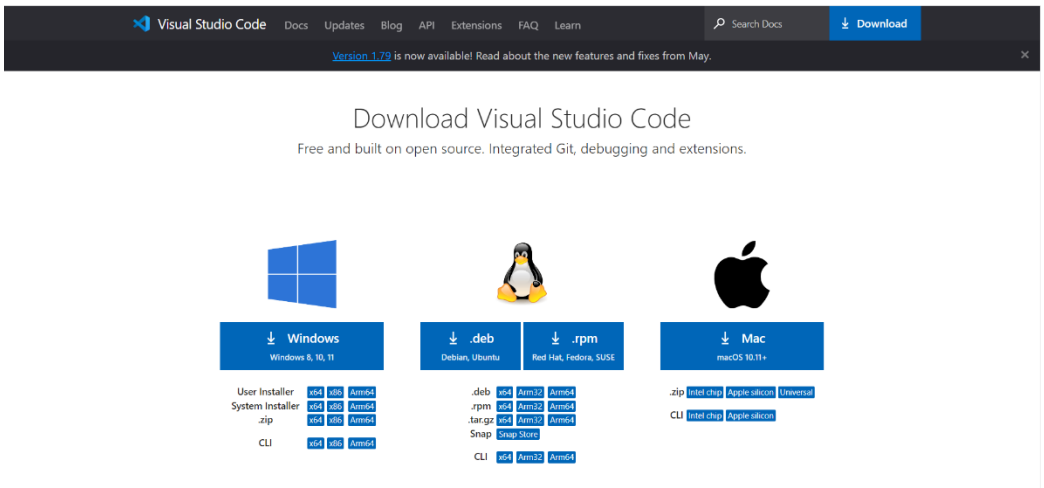

Figura 3.23.- Página de descarga de VS Code. Fuente: Diseño propio

Visual Studio Code es un IDE que permite su personalización en muchos aspectos, como pueden ser el idioma, el tema de color o atajos de teclado. Así, la primera vez que se accede se muestra una pantalla que permite personalizar el aspecto estético del programa:

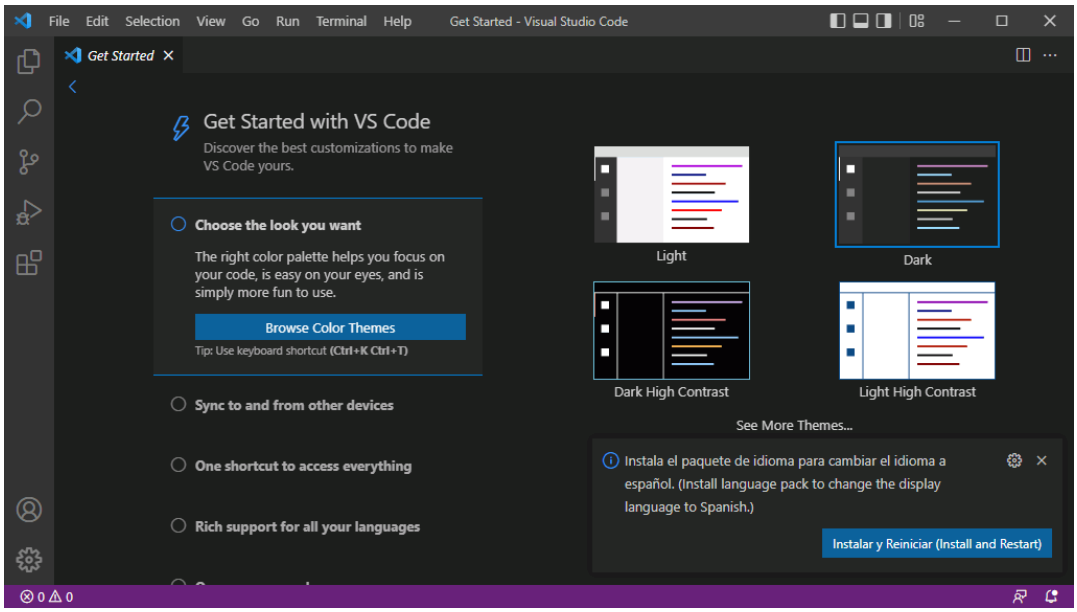

Figura 3.24.- Vista del primer acceso a VS Code. Fuente: Diseño propio

Una vez personalizado se puede empezar a desarrollar un proyecto. Lo primero sería crear el directorio que va a alojar la aplicación. Simplemente se haría *File >> Open Folder*. Con esto ya quedaría definido el espacio de trabajo del proyecto, y se podrían empezar a añadir los archivos necesarios.

3.2.3.- Hardware utilizado

Para el desarrollo del proyecto se ha hecho uso de un ordenador portátil HP Pavilion 15-eg2008ns, con las siguientes características:

- Sistema operativo Windows 11
- CPU con arquitectura de 64 bits
- Procesador Intel Core i7 12ª Generación
- 16 GB de memoria RAM

- SSD de 1 TB

Además, para las pruebas de la skill HablaConmigo se ha utilizado un dispositivo Echo Show 8, con las siguientes características:

- Pantalla HD de 8’’
- Cámara de 13 MP
- Dos altavoces de 2,0’’
- Dimensiones de 200x135x99 mm

3.2.4.- Software utilizado

Para el desarrollo de la skill HablaConmigo y de la aplicación web de ayuda para el logopeda se ha hecho uso de los siguientes softwares:

- Alexa Developer Console
- Visual Studio Code (Versión 1.79.2)

3.3.- DESCRIPCIÓN DEL SISTEMA

A continuación, se detalla el funcionamiento tanto de la skill como de la aplicación web de gestión, además de sus componentes y código.

3.3.1.- Casos de uso

Los casos de uso se utilizan para relacionar los requisitos funcionales de un sistema con las interacciones entre los usuarios y el sistema. A continuación, se exponen los casos de uso tanto de la skill HablaConmigo como de la aplicación web que la acompaña

3.3.1.1.- Casos de uso de la skill

Para poder jugar a alguno de los juegos que ofrece HablaConmigo, el usuario debe registrarse, o, en caso de haberlo hecho con anterioridad, iniciar sesión. A continuación, ya puede elegir uno de los seis juegos de los que dispone la skill, para después elegir un nivel de los disponibles para el juego seleccionado y empezar a jugar. Durante el juego, el usuario también podrá cambiar de nivel dentro de ese mismo juego o cambiar de juego.

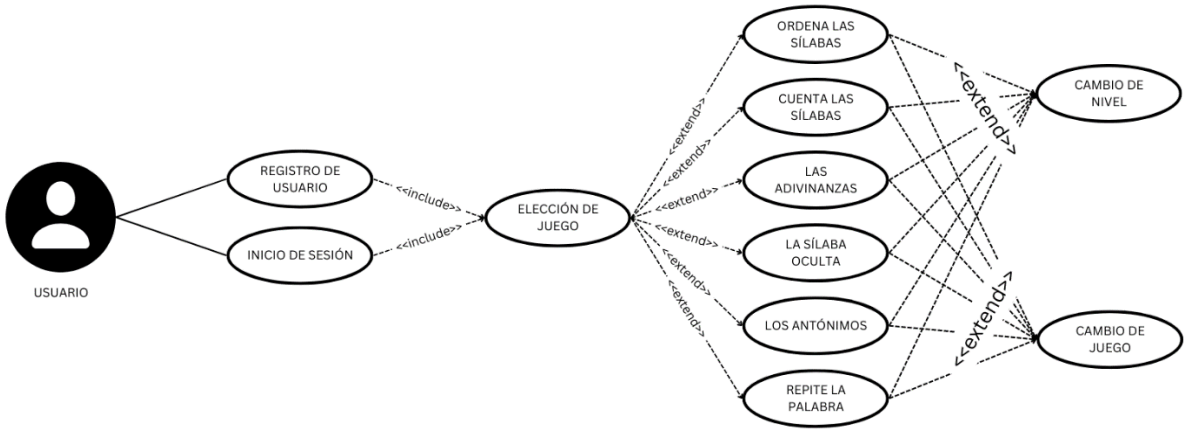

Figura 3.25.- Casos de uso de la skill. Fuente: Diseño propio con Canva

3.3.1.2.- Casos de uso de la skill – Registro de un nuevo usuario

Identificador	CUS-1
Nombre	Registro de usuario
Actores	Usuario (niño)
Requisitos	RF-1.1, RNF-1.1, RNF-1.2, RNF-1.3, RNF-1.4
Propósito	Utilizando un número que servirá de identificador, elegido por el niño, además de su nombre y edad, este podrá “crear un perfil” en la skill.
Precondiciones	El niño debe tener acceso a la pantalla de bienvenida para poder registrarse.
Flujo principal	1.- El niño abre la skill. 2.- El niño le indica a Alexa que “sí” es la primera vez que juega. 3.- El niño le indica a Alexa el id que quiere utilizar, junto con su nombre y edad, en una de las frases aceptadas para que interprete sus datos correctamente.

Flujo secundario	3a.- El niño intenta registrarse con un id que ya está registrado. Alexa le indica que ese id ya está registrado y se vuelve al paso 3 a la espera de que el niño elija un id válido. 3b.- El niño intenta proporcionar sus datos con una frase que Alexa no puede interpretar correctamente. Alexa le indica que no le ha entendido y se vuelve al paso 3 a la espera de que el niño pronuncie la frase correctamente.

Tabla 3.5.- Casos de uso de la skill - Registro de usuario

3.3.1.3.- Casos de uso de la skill – Inicio de sesión

Identificador	CUS-2
Nombre	Inicio de sesión
Actores	Usuario (niño)
Requisitos	RF-1.2, RNF-1.1, RNF-1.2, RNF-1.3, RNF-1.4
Propósito	Utilizando el id que utilizó con anterioridad para registrarse, el niño podrá iniciar sesión en la skill.
Precondiciones	El niño debe tener acceso a la pantalla de bienvenida para poder registrarse.
Flujo principal	1.- El niño abre la skill. 2.- El niño le indica a Alexa que “no” es la primera vez que juega. 3.- El niño le indica a Alexa su id para que esta recupere sus datos y el niño pueda empezar a jugar.
Flujo secundario	3a.- El niño intenta acceder con un id que no está registrado. Alexa le indica que no se ha encontrado ningún usuario registrado con ese id y vuelve al paso 3 para que el usuario proporcione un id válido de inicio de sesión. 3b.- El niño intenta proporcionar su id con una frase que Alexa no puede interpretar correctamente. Alexa le indica que no le ha entendido y se vuelve al paso 3 a la espera de que el niño pronuncie la frase correctamente.

Tabla 3.6.- Casos de uso de la skill – Inicio de sesión

3.3.1.4.- Casos de uso de la skill – Elección de juego

Identificador	CUS-3
Nombre	Elección de juego
Actores	Usuario (niño)
Requisitos	RF-1.3, RNF-1.1, RNF-1.2, RNF-1.3, RNF-1.4
Propósito	El usuario podrá elegir uno de los seis juegos que ofrece la skill.
Precondiciones	El niño debe haberse registrado o iniciado sesión previamente.
Flujo principal	1.- El niño se registra o inicia sesión. 2.- El niño elige un juego de los seis disponibles para empezar a jugar.
Flujo secundario	2a.- El niño elige un juego que no se encuentra en la lista. Alexa le indica que elija un juego disponible y se vuelve al paso 2 a la espera de que el niño elija un juego válido. 2b.- El niño elige el juego con una frase que Alexa no puede interpretar correctamente. Alexa le indica que no le ha entendido y se vuelve al paso 2 a la espera de que el niño pronuncie la frase correctamente.

Tabla 3.7.- Casos de uso de la skill - Elección de juego

3.3.1.5.- Casos de uso de la skill – Juego “Ordena las Sílabas”

Identificador	CUS-4
Nombre	Juego “Ordena las Sílabas”
Actores	Usuario (niño)
Requisitos	RF-1.4, RF-1.7, RF-1.10, RNF-1.1, RNF-1.2, RNF-1.3, RNF-1.4
Propósito	El usuario podrá elegir uno de entre los cuatro niveles disponibles de “Ordena las Sílabas”.
Precondiciones	El niño debe haber elegido como juego “Ordena las Sílabas” previamente y jugar un “acertijo”.
Flujo principal	1.- El niño elige “Ordena las Sílabas” como juego. 2.- El niño elige uno de los cuatro niveles disponibles. 3.- El niño da una respuesta al enunciado formulado por Alexa. 4.- Alexa comprueba si su respuesta es correcta o no y se lo indica al niño.

Flujo secundario	2a.- El niño elige un nivel que no está disponible para "Ordena las Sílabas". Alexa le indica que elija un nivel disponible para este juego y se vuelve al paso 2 a la espera de que el niño responda con un nivel válido. 2b.- El niño elige el nivel con una frase que Alexa no puede interpretar correctamente. Alexa le indica que no le ha entendido y se vuelve al paso 2 a la espera de que el niño pronuncie la frase correctamente. 3a.- El niño da una respuesta con una frase que Alexa no puede interpretar correctamente. Alexa le indica que no le ha entendido y se vuelve al paso 3 a la espera de que el niño pronuncie la frase correctamente.

Tabla 3.8.- Casos de uso de la skill - Juego "Ordena las Sílabas"

3.3.1.6.- Casos de uso de la skill – Juego "Cuenta las Sílabas"

Identificador	CUS-5
Nombre	Juego "Cuenta las Sílabas"
Actores	Usuario (niño)
Requisitos	RF-1.4, RF-1.7, RF-1.10, RNF-1.1, RNF-1.2, RNF-1.3, RNF-1.4
Propósito	El usuario podrá elegir uno de entre los dos niveles disponibles de "Cuenta las Sílabas".
Precondiciones	El niño debe haber elegido como juego "Cuenta las Sílabas" previamente y jugar un "acertijo".
Flujo principal	1.- El niño elige "Cuenta las Sílabas" como juego. 2.- El niño elige uno de los dos niveles disponibles. 3.- El niño da una respuesta al enunciado formulado por Alexa. 4.- Alexa comprueba si su respuesta es correcta o no y se lo indica.

Flujo secundario	2a.- El niño elige un nivel que no está disponible para "Cuenta las Sílabas". Alexa le indica que elija un nivel disponible para este juego y se vuelve al paso 2 a la espera de que el niño responda con un nivel válido. 2b.- El niño elige el nivel con una frase que Alexa no puede interpretar correctamente. Alexa le indica que no le ha entendido y se vuelve al paso 2 a la espera de que el niño pronuncie la frase correctamente. 3a.- El niño da una respuesta con una frase que Alexa no puede interpretar correctamente. Alexa le indica que no le ha entendido y se vuelve al paso 3 a la espera de que el niño pronuncie la frase correctamente.

Tabla 3.9.- Casos de uso de la skill - Juego "Cuenta las Sílabas"

3.3.1.7.- Casos de uso de la skill – Juego "Las Adivinanzas"

Identificador	CUS-6
Nombre	Juego "Las Adivinanzas"
Actores	Usuario (niño)
Requisitos	RF-1.4, RF-1.8, RF-1.10, RNF-1.1, RNF-1.2, RNF-1.3, RNF-1.4
Propósito	El usuario podrá elegir uno de entre los tres niveles disponibles de "Las Adivinanzas" y jugar un "acertijo".
Precondiciones	El niño debe haber elegido como juego "Las Adivinanzas" previamente.
Flujo principal	1.- El niño elige "Las Adivinanzas" como juego. 2.- El niño elige uno de los tres niveles disponibles. 3.- El niño da una respuesta al enunciado formulado por Alexa. 4.- Alexa comprueba si su respuesta es correcta o no y se lo indica.

Flujo secundario	2a.- El niño elige un nivel que no está disponible para "Las Adivinanzas". Alexa le indica que elija un nivel disponible para este juego y se vuelve al paso 2 a la espera de que el niño responda con un nivel válido. 2b.- El niño elige el nivel con una frase que Alexa no puede interpretar correctamente. Alexa le indica que no le ha entendido y se vuelve al paso 2 a la espera de que el niño pronuncie la frase correctamente. 3a.- El niño da una respuesta con una frase que Alexa no puede interpretar correctamente. Alexa le indica que no le ha entendido y se vuelve al paso 3 a la espera de que el niño pronuncie la frase correctamente.

Tabla 3.10.- Casos de uso de la skill - Juego "Las Adivinanzas"

3.3.1.8.- Casos de uso de la skill – Juego "La Sílaba Oculta"

Identificador	CUS-7
Nombre	Juego "La Sílaba Oculta"
Actores	Usuario (niño)
Requisitos	RF-1.4, RF-1.7, RF-1.10, RNF-1.1, RNF-1.2, RNF-1.3, RNF-1.4
Propósito	El usuario elegirá el único nivel de "La Sílaba Oculta" y podrá jugar un "acertijo".
Precondiciones	El niño debe haber elegido como juego "La Sílaba Oculta" previamente.
Flujo principal	1.- El niño elige "La Sílaba Oculta" como juego. 2.- El niño elige el único nivel disponible de este juego. 3.- El niño da una respuesta al enunciado formulado por Alexa. 4.- Alexa comprueba si su respuesta es correcta o no y se lo indica.

Flujo secundario	2a.- El niño elige un nivel que no está disponible para "La Sílaba Oculta". Alexa le indica que elija un nivel disponible para este juego y se vuelve al paso 2 a la espera de que el niño responda con un nivel válido. 2b.- El niño elige el nivel con una frase que Alexa no puede interpretar correctamente. Alexa le indica que no le ha entendido y se vuelve al paso 2 a la espera de que el niño pronuncie la frase correctamente. 3a.- El niño da una respuesta con una frase que Alexa no puede interpretar correctamente. Alexa le indica que no le ha entendido y se vuelve al paso 3 a la espera de que el niño pronuncie la frase correctamente.

Tabla 3.11.- Casos de uso de la skill - Juego "La Sílaba Oculta"

3.3.1.9.- Casos de uso de la skill – Juego "Los Antónimos"

Nombre	Juego "Los Antónimos"
Actores	Usuario (niño)
Requisitos	RF-1.4, RF-1.8, RF-1.10, RNF-1.1, RNF-1.2, RNF-1.3, RNF-1.4
Propósito	El usuario podrá elegir uno de entre los dos niveles disponibles de "Los Antónimos" y jugar un "acertijo".
Precondiciones	El niño debe haber elegido como juego "Los Antónimos" previamente.
Flujo principal	1.- El niño elige "Los Antónimos" como juego. 2.- El niño elige uno de los dos niveles disponibles. 3.- El niño da una respuesta al enunciado formulado por Alexa. 4.- Alexa comprueba si su respuesta es correcta o no y se lo indica.
Flujo secundario	2a.- El niño elige un nivel que no está disponible para "Los Antónimos". Alexa le indica que elija un nivel disponible para este juego y se vuelve al paso 2 a la espera de que el niño responda con un nivel válido. 2b.- El niño elige el nivel con una frase que Alexa no puede interpretar correctamente. Alexa le indica que no le ha entendido y se vuelve al paso 2 a la espera de que el niño pronuncie la frase correctamente. 3a.- El niño da una respuesta con una frase que Alexa no puede interpretar correctamente. Alexa le indica que no le ha entendido y se vuelve al paso 3 a la espera de que el niño pronuncie la frase correctamente.

Tabla 3.12.- Casos de uso de la skill - Juego "Los Antónimos"

3.3.1.10.- Casos de uso de la skill – Juego “Repite la Palabra”

Identificador	CUS-9
Nombre	Juego “Repite la Palabra”
Actores	Usuario (niño)
Requisitos	RF-1.4, RF-1.9, RF-1.10, RNF-1.1, RNF-1.2, RNF-1.3, RNF-1.4
Propósito	El usuario podrá elegir uno de entre los cuatro niveles disponibles de “Repite la Palabra” y jugar un “acertijo”.
Precondiciones	El niño debe haber elegido como juego “Repite la Palabra” previamente.
Flujo principal	1.- El niño elige “Repite la Palabra” como juego. 2.- El niño elige uno de los cuatro niveles disponibles. 3.- El niño da una respuesta al enunciado formulado por Alexa. 4.- Alexa comprueba si su respuesta es correcta o no y se lo indica.
Flujo secundario	2a.- El niño elige un nivel que no está disponible para “Repite la Palabra”. Alexa le indica que elija un nivel disponible para este juego y se vuelve al paso 2 a la espera de que el niño responda con un nivel válido. 2b.- El niño elige el nivel con una frase que Alexa no puede interpretar correctamente. Alexa le indica que no le ha entendido y se vuelve al paso 2 a la espera de que el niño pronuncie la frase correctamente. 3a.- El niño da una respuesta con una frase que Alexa no puede interpretar correctamente. Alexa le indica que no le ha entendido y se vuelve al paso 3 a la espera de que el niño pronuncie la frase correctamente.

Tabla 3.13.- Casos de uso de la skill - Juego "Repite la Palabra"

3.3.1.11.- Casos de uso de la skill – Cambio de nivel

Identificador	CUS-10
Nombre	Cambio de nivel
Actores	Usuario (niño)
Requisitos	RF-1.5, RNF-1.1, RNF-1.2, RNF-1.3, RNF-1.4
Propósito	El usuario podrá elegir jugar a otro nivel dentro del mismo juego al que estaba jugando.
Precondiciones	El niño debe haber elegido un juego y un nivel previamente.

Flujo principal	1.- El niño elige un juego y un nivel. 2.- El niño da una respuesta al enunciado formulado por Alexa y Alexa comprueba si su respuesta es correcta o no y se lo indica. 3.- El niño elige un nivel distinto. 4.- Alexa le muestra un nuevo enunciado perteneciente al nuevo nivel elegido.
Flujo secundario	3a.- El niño elige un nivel que no está disponible para el juego seleccionado con anterioridad. Alexa le indica que elija un nivel disponible para el juego actual y se vuelve al paso 3 a la espera de que el niño responda con un nivel válido. 3b.- El niño elige el nivel con una frase que Alexa no puede interpretar correctamente. Alexa le indica que no le ha entendido y se vuelve al paso 3 a la espera de que el niño pronuncie la frase correctamente.

Tabla 3.14.- Casos de uso de la skill - Cambio de nivel

3.3.1.12.- Casos de uso de la skill – Cambio de juego

Identificador	CUS-11
Nombre	Cambio de juego
Actores	Usuario (niño)
Requisitos	RF-1.6, RNF-1.1, RNF-1.2, RNF-1.3, RNF-1.4
Propósito	El usuario podrá elegir jugar a otro juego durante la misma sesión
Precondiciones	El niño debe haber elegido un juego y un nivel previamente.
Flujo principal	1.- El niño elige un juego y un nivel 2.- El niño da una respuesta al enunciado formulado por Alexa y Alexa comprueba si su respuesta es correcta o no y se lo indica 3.- El niño elige un juego distinto 4.- Alexa le indica al niño que elija un nivel del nuevo juego elegido para empezar a jugar a él
Flujo secundario	3a.- El niño elige un juego del que la skill no dispone. Alexa le indica que elija un juego disponible y se vuelve al paso 3 a la espera de que el niño responda con un juego válido. 3b.- El niño elige el juego con una frase que Alexa no puede interpretar correctamente. Alexa le indica que no le ha entendido y se vuelve al paso 3 a la espera de que el niño pronuncie la frase correctamente.

Tabla 3.15.- Casos de uso de la skill - Cambio de juego

3.3.1.13.- Casos de uso de la aplicación web

Para poder utilizar la aplicación web, lo único que necesita el logopeda es abrir el archivo .html que corresponde con la página principal ("index.html"). Una vez dentro, podrá elegir entre añadir una nueva entrada a uno de los juegos y niveles disponibles o consultar el progreso de un usuario.

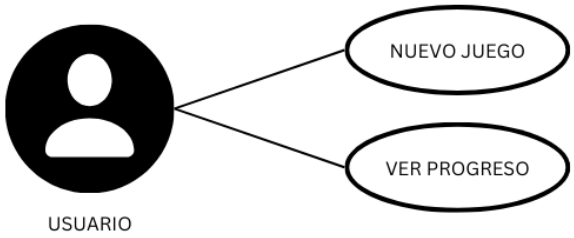

Figura 3.26.- Casos de uso de la aplicación web. Fuente: Diseño propio con Canva

3.3.1.14.- Casos de uso de la aplicación web – Nuevo entrada de juego

Identificador	CUAP-1
Nombre	Nueva entrada de juego
Actores	Usuario (logopeda)
Requisitos	RF-2.3, RNF-2.1, RNF-2.2, RNF-2.3, RNF-2.4
Propósito	El usuario podrá añadir una nueva entrada para uno de los niveles de uno de los juegos ya creados.
Precondiciones	El logopeda debe haber abierto previamente la aplicación web y accedido a la página de "Nuevo juego".
Flujo principal	1.- El logopeda abre la página para añadir una nueva entrada. 2.- El logopeda crea una nueva entrada con los campos (al menos "ID del juego", "Palabra, adivinanza", "Solución" y "Nivel") completos. Para los niveles que sea necesario, también se pueden rellenar "Pista de texto" y "Pista de imagen". 3.- El logopeda hace clic en "Enviar" y se guarda la nueva entrada en la base de datos. 4.- La aplicación muestra un mensaje de éxito en la subida.

Flujo secundario	3a.- El logopeda no rellena uno o más campos obligatorios. La aplicación muestra el mensaje "Error al subir el juego" y se vuelve al paso 2 a la espera de que el logopeda rellene el formulario correctamente. 4a.- Si se produce algún error en la subida, la aplicación muestra el mensaje "Error al subir el juego" y se vuelve al paso 3 a la espera de que el logopeda vuelva a enviar la entrada a la base de datos y esta pueda subirse correctamente.

Tabla 3.16.- Casos de uso de la aplicación web - Nueva entrada de juego

3.3.1.15.- Casos de uso de la aplicación web – Ver progreso

Identificador	CUAP-2
Nombre	Ver progreso
Actores	Usuario (logopeda)
Requisitos	RF-2.1, RF-2.2, RNF-2.1, RNF-2.2, RNF-2.3, RNF-2.4
Propósito	El usuario podrá añadir consultar los resultados obtenidos por un niño en un determinado nivel de un determinado juego.
Precondiciones	El logopeda debe haber abierto previamente la aplicación web.
Flujo principal	1.- El logopeda abre la página para consultar el progreso 2.- El logopeda rellena el formulario de consulta de progreso introduciendo, al menos, el "ID del usuario", el "ID del juego" y el "Nivel" (obteniendo todos los resultados de ese usuario). En caso de querer filtrar los resultados, puede añadirse una "Fecha y hora de inicio" (se mostrarían todos los resultados a partir de esa fecha), "Fecha y hora de fin" (se mostrarían todos los resultados hasta esa fecha) o pueden añadirse ambas fechas (se mostrarían todos los resultados obtenidos en el periodo de tiempo acotado por esas fechas). 3.- El logopeda hace clic en "Obtener resultados" 4.- La aplicación muestra los resultados en el apartado habilitado.
Flujo secundario	3a.- El logopeda no rellena uno o más campos obligatorios. La aplicación muestra el mensaje "Error al solicitar los resultados" y se vuelve al paso 2 a la espera de que el logopeda rellene todos los campos requeridos correctamente. 4a.- Si se produce algún error en la descarga de datos, estos no se mostrarán, y se vuelve al paso 3 a la espera de que el logopeda solicite otra vez los datos y estos puedan descargarse correctamente.

Tabla 3.17.- Casos de uso de la aplicación web - Ver progreso

3.3.2.- Diseño de la base de datos

Para almacenar los datos del sistema se ha utilizado una base de datos MySQL con la estructura y relaciones que se observan en la Figura 3.27.

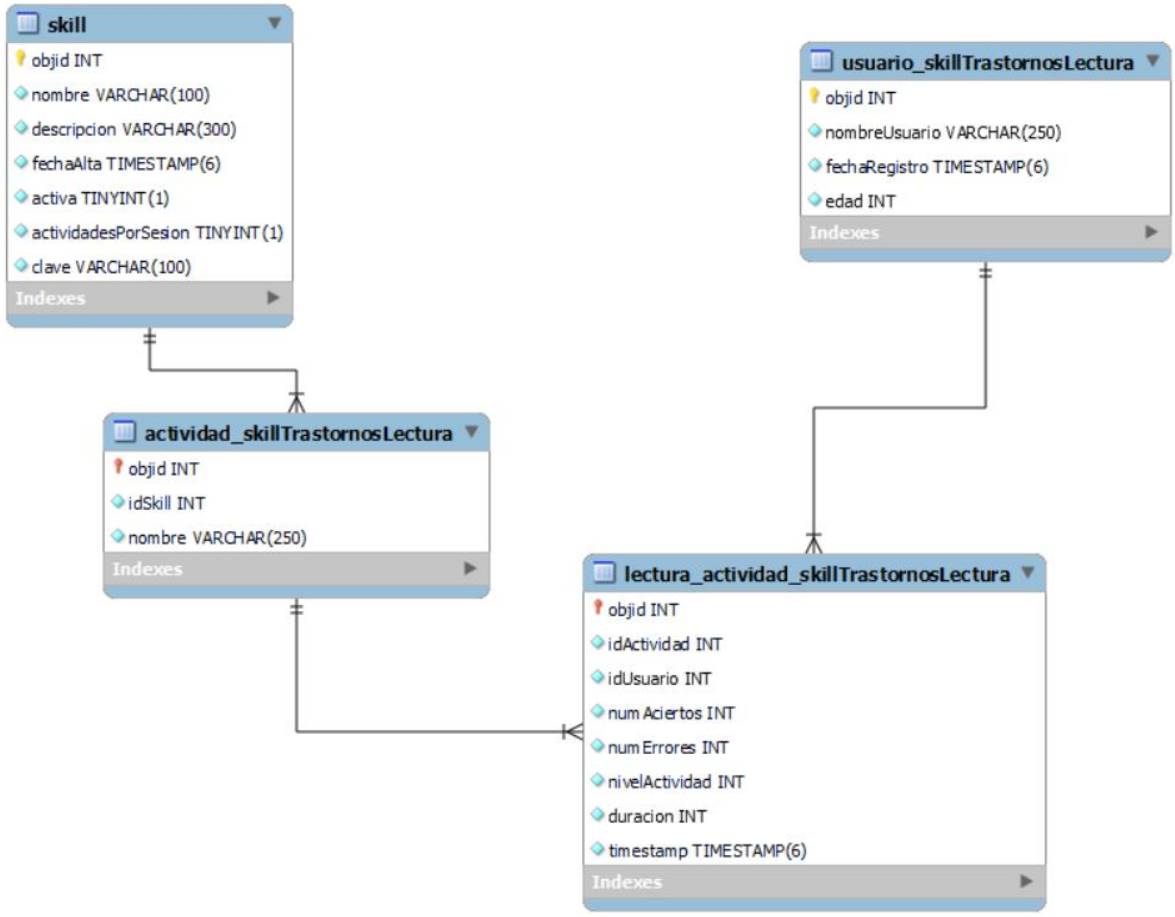

Figura 3.27.- Base de datos del sistema. Fuente: Diseño propio con MySQL Workbench

La base de datos consta de 4 tablas distintas:

- skill: En esta tabla se almacenan los datos generales relativos a la skill HablaConmigo, y consta de los siguientes campos:
    - objid: Identificador único de la skill HablaConmigo, que tiene un valor c7800aee8f9287a60ffe835984a38431.
    - nombre: Nombre de la skill.
    - descripcion: Campo para detallar el propósito de la skill.

  - fechaAlta: Fecha en la que se dio de alta la skill.
  - activa: Indicador booleano de si la skill está activa o no (toma el valor 1 si está activa y 0 si no lo está).
  - actividadesPorSesion: Número de actividades que se realizan por sesión. Por defecto, tiene un valor 5.
  - clave: Clave adicional para la skill.

- actividad_skillTrastornosLectura: En esta tabla se almacenan los nombres de los distintos juegos que dispone la skill, y consta de los siguientes campos:
  - objid: Identificador único de la actividad (juego) correspondiente.
  - idSkill: Clave foránea a la tabla skill.
  - nombre: Cadena de texto que almacena el nombre del juego.

- usuario_skillTrastornosLectura: En esta tabla se almacenan todos los datos personales de los usuarios. Consta de los siguientes campos:
  - objId: Identificador único del usuario.
  - nombreUsuario: Cadena de texto que almacena el nombre del usuario.
  - fechaRegistro: Campo de tipo timestamp que registra, en el momento en el que se añada una nueva entrada a la base de datos, la fecha en la que se da de alta un nuevo usuario. Tiene el formato YYYY-MM-DD hh:mm:ss.
  - edad: Número de tipo int que guarda la edad del usuario.

- lectura_actividad_skillTrastornosLectura: En esta tabla se almacenan los datos de juego de los usuarios. Consta de los siguientes campos:
  - objid: Identificador único del registro de la tabla.
  - idActividad: Clave foránea a la tabla actividad_skillTrastornosLectura. Hace referencia al identificador de la actividad correspondiente.
  - idUsuario: Clave foránea a la tabla usuario_skillTrastornosLectura. Hace referencia al identificador del usuario correspondiente.
  - numAciertos: Número de tipo int que almacena el número de aciertos del usuario en la sesión.

- numErrores: Número de tipo int que almacena el número de errores del usuario en la sesión.
- nivelActividad: Número de tipo int que almacena el nivel al que está jugando el usuario.
- duracion: Número de tipo int que guarda el tiempo (en segundos) que tarda el usuario en dar una respuesta desde que se formula la pregunta.
- timestamp: Campo de tipo timestamp que registra, en el momento en el que se añada una nueva entrada a la base de datos, la fecha en la que se da de alta un nuevo usuario. Tiene el formato YYYY-MM-DD hh:mm:ss.

3.3.3.- API REST

Como se mencionó con anterioridad, el sistema está conectado mediante una API RESTful que implementa un CRUD parcial (peticiones GET y POST del protocolo HTTP). Con el uso de la API se consigue una arquitectura cliente-servidor en la que ambas partes están totalmente separadas. Así, la skill y la aplicación web serán las encargadas de ejercer la función de cliente, pidiendo y publicando datos en la base de datos.

La comunicación con el servidor REST se realiza mediante URLs que permiten tanto a la skill como a la aplicación web realizar consultas y enviar datos nuevos utilizando el método HTTP. Para publicar o leer un registro se hace una petición utilizando el protocolo HTTP a www.it.uniovi.es, utilizando un mecanismo de autenticación Basic y un Content-Type application/json. Así, las peticiones se realizan a la URL https://usuarioAut:pass@www.it.uniovi.es/hosesbackend/skillTrastornosLectura.php, siendo usuarioAut *lfonteriz* y pass *aaa1674unTFG*.

Para la tabla usuario_skillTrastornosLectura, los servicios implementados son los siguientes:

- GET: Obtener los datos de un usuario
    - Parámetros en la URL:
        - idSkill (c7800aee8f9287a60ffe835984a38431)
        - idUsuario
    - Respuesta:
        - Código de estado HTTP 200 (OK), 400 (Bad Request) o 404 (Not Found)
        - Objeto JSON con la lista de registros
    - Ejemplo: https://lfonteriz:aaa1674unTFG@www.it.uniovi.es/hosesbackend/skillTrastornosLectura.php?idSkill=c7800aee8f9287a60ffe835984a38431&idUsuario=5

- POST: Añadir un nuevo usuario a la base de datos
    - Parámetros:
        - idSkill (c7800aee8f9287a60ffe835984a38431)
        - idUsuario
        - nombreUsuario
        - edad
    - Respuesta: Código de estado HTTP 201 (Created), 400 (Bad Request) o 406 (Not Acceptable)
    - Ejemplo:
        - https://lfonteriz:aaa1674unTFG@www.it.uniovi.es/hosesbackend/skillTrastornosLectura.php
        - En el cuerpo de la petición idSkill, idUsuario, nombreUsuario y edad

Para la tabla lectura_actividad_skillTrastornosLectura, los servicios implementados son los siguientes:

- GET: Obtener los resultados de un usuario en un periodo de tiempo determinado (si los campos timestamps no tienen valor devuelve todos los resultados de un usuario en ese nivel, si solo tiene valor el timestampInicio devuelve los resultados a partir de una fecha, si solo tiene valor el timestampFinal devuelve los resultados hasta una fecha, y si tienen valor tanto timestampInicio como timestampFinal devuelve los resultados obtenidos en el periodo delimitado por esas dos fechas).
    - Parámetros en la URL:
        - idSkill (c7800aee8f9287a60ffe835984a38431)
        - idActividad
        - nivelActividad
        - idUsuario
        - timestampInicio (opcional, formato YYYY-MM-DD hh:mm:ss)
        - timestampFinal (opcional, formato YYYY-MM-DD hh:mm:ss)
    - Respuesta:
        - Código de estado HTTP 200 (OK), 400 (Bad Request) o 404 (Not Found)
        - Objeto JSON con la lista de registros
    - Ejemplo que devuelve todos los registros de un usuario: https://lfonteriz:aaa1674unTFG@www.it.uniovi.es/hosesbackend/skillTrastornosLectura.php?idSkill=c7800aee8f9287a60ffe835984a38431&idActividad=5&nivelActividad=1&idUsuario=1
    - Ejemplo que devuelve los registros de un usuario a partir de una fecha: https://lfonteriz:aaa1674unTFG@www.it.uniovi.es/hosesbackend/skillTrastornosLectura.php?idSkill=c7800aee8f9287a60ffe835984a38431&idActividad=5&nivelActividad=1&idUsuario=1&timestampInicio=2023-06-28%2008:55:14
    - Ejemplo que devuelve los registros de un usuario hasta una fecha: https://lfonteriz:aaa1674unTFG@www.it.uniovi.es/hosesbackend/skillTrastornosLectura.php?idSkill=c7800aee8f9287a60ffe835984a38431&idActivi

dad=5&nivelActividad=1&idUsuario=1&timestampFinal=2023-06-28%2009:12:27

  - Ejemplo que devuelve los registros de un usuario en un periodo delimitado: https://lfonteriz:aaa1674unTFG@www.it.uniovi.es/hosesbackend/skillTrastornosLectura.php?idSkill=c7800aee8f9287a60ffe835984a38431&idActividad=5&nivelActividad=1&idUsuario=1&timestampInicio=2023-06-28%2008:00:00&timestampFinal=2023-06-28%2009:11:27

- POST: Añadir una nueva entrada con los resultados de un usuario
  - Parámetros:
    - idSkill (c7800aee8f9287a60ffe835984a38431)
    - idActividad
    - idUsuario
    - duracion
    - numAciertos
    - numErrores
    - nivelActividad
  - Respuesta: Código de estado HTTP 201 (Created), 400 (Bad Request) o 406 (Not Acceptable)
  - Ejemplo:
    - https://lfonteriz:aaa1674unTFG@www.it.uniovi.es/hosesbackend/skillTrastornosLectura.php
    - En el cuerpo de la petición idSkill, idActividad, idUsuario, duración, numAciertos, numErrores y nivelActividad

3.3.5.- Explicación de las partes principales del código

A continuación, se detallan las principales partes del código tanto de la skill como de la aplicación web.

3.3.5.1.- Código de la skill HablaConmigo

El primer paso para hablar del código de la skill HablaConmigo es explicar la estructura del proyecto. Como se puede ver en la Figura 3.28, el proyecto se aloja en una carpeta "lambda", que se encuentra, a su vez, dentro de una carpeta "Skill Code". En esta carpeta "lambda" se alojan, a su vez, otras dos carpetas ("documents" y "functions", cada una con sus respectivos archivos) y cuatro archivos más.

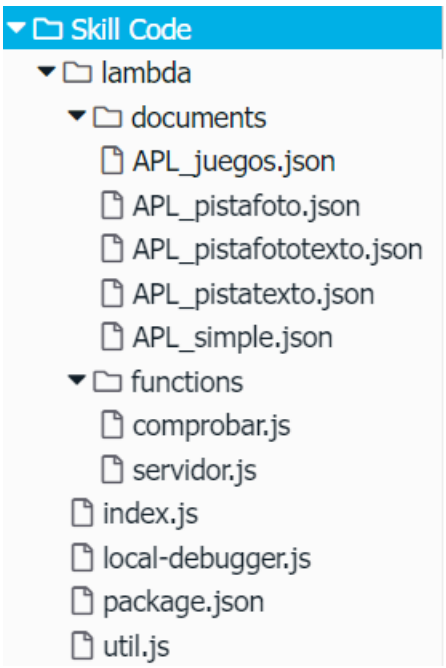

Figura 3.28.- Estructura del proyecto de la skill. Fuente: Diseño propio

En la primera carpeta, "documents" se alojan las distintas plantillas de Alexa Presentation Language (APL). Se trata de archivos de extensión ".json" que se utilizan para mostrar en la pantalla del dispositivo interacciones de Alexa con el usuario. La primera parte del código (Figura 3.29), es común a todos los archivos. Aquí se especifica el tipo de archivo ("APL"), la versión del APL utilizada ("2023.1") y el tema ("dark"). En la sección "mainTemplate", se comienza a definir la plantilla principal de la interfaz. Lo primero es definir un parámetro "payload", que recibirá un valor de entrada y permitirá personalizar la plantilla en función de la respuesta del usuario. Después, en "ítems", se crea un contenedor que alojará todo el contenido de la pantalla y lo centrará en esta. Dentro de este contenedor se

crea otro que representará el fondo de pantalla. En este contenedor se especifica en “source” la URL de la imagen que se va a formar, se utiliza “best-fill” para asegurarse de que la imagen se ajuste de forma óptima, sin distorsionarse, se definen las dimensiones de la imagen y se centra en el contenedor. Después, en cada archivo, se definen los elementos necesarios.

```
{
    "type": "APL",
    "version": "2023.1",
    "license": "",
    "settings": {},
    "theme": "dark",
    "import": [],
    "resources": [],
    "styles": {},
    "onMount": [],
    "graphics": {},
    "commands": {},
    "layouts": {},
    "mainTemplate": {
        "parameters": [
            "payload"
        ],
        "items": [
            {
                "justifyContent": "center",
                "items": [
                    {
                        "source":
"https://www.it.uniovi.es/hosesbackend/multimedia/fondo.jpg",
                        "scale": "best-fill",
                        "type": "Image",
                        "width": "110%",
                        "height": "110%",
                        "position": "absolute",
                        "alignSelf": "center"
                    },
                    {
                        "alignItems": "center",
                        "items": [
                            {
                                "text": "Elige un juego",
                                "fontSize": "50dp",
                                "color": "black",
                                "textAlign": "center",
                                "textAlignVertical": "center",
                                "fontWeight": "bold",
                                "type": "Text",
                                "width": "400dp",
                                "height": "75dp",
                                "paddingTop": "12dp",
                                "paddingBottom": "12dp"
                            },
```

Figura 3.29.- Inicio código archivos APL. Fuente: Diseño propio

En el caso de “APL_juegos.json” (Figura 3.30), se añade otro contenedor formado por un texto de título tamaño 50dp, en negro, centrado tanto vertical como horizontalmente, en negrita y con un espaciado tanto arriba como debajo de 12dp, y a continuación, 6 textos con el mismo formato (texto de tamaño 25dp, alineado a la izquierda, centrado verticalmente, de color negro y con un espaciado tanto arriba como debajo de 12dp). En cada uno de estos textos se muestra → *Nombre de un juego*.

```
{

    "fontSize": "25dp",
    "textAlign": "left",
    "textAlignVertical": "center",
    "color": "black",
    "text": "→ Ordena las sílabas",
    "type": "Text",
    "width": "250dp",
    "height": "50dp",
    "paddingTop": "12dp",
    "paddingBottom": "12dp"
},
```

Figura 3.30.- Código del archivo APL_juegos.json. Fuente: Diseño propio

En la Figura 3.31 se muestra el código de “APL_simple.json”. En él se definen dos campos de texto, uno en negrita, tamaño de 50dp y color negro, y otro de tamaño 20dp, color negro y espaciado de 20dp. En ambos se utiliza el parámetro “payload”. Esta funcionalidad permite presentar en la plantilla el texto deseado en cada campo según la interacción del usuario. La plantilla “APL_simple.json” es exactamente igual, solo que se añade un campo más de texto entre “title” y “body”, denominado “subtitle” y con el mismo formato que “body”.

```
"items": [
            {
              "type": "Text",
              "text": "${payload.myData.title}",
              "textAlign": "center",
              "fontWeight": "bold",
              "fontSize": "50dp",
              "color": "black"
            },
            {
              "type": "Text",
              "text": "${payload.myData.body}",
              "textAlign": "center",
              "fontSize": "30dp",
              "color": "black",
              "spacing": "20dp"
            }
          ]
```

Figura 3.31.- Código de APL_simple.json. Fuente: Diseño propio

También tienen la misma estructura las plantillas “APL_pistafoto.json” y “APL_pistafototexto.json” (Figura 3.32). Ambas constan de un campo de texto de título de tamaño 40dp, en negrita, y color negro que muestra el texto que se obtiene a través del parámetro “payload”. También es común entre ambas plantillas un contenedor para una imagen, cuya URL fuente se obtiene también a través del parámetro “payload”. La diferencia entre ambas plantillas es que “APL_pistafototexto.json” incluye un campo de texto más, “subtitle”, de tamaño 30dp, color negro y espaciado de 20dp, y cuyo valor también se obtiene a través del parámetro “payload”.

```
{
  "text": "${payload.myData.title}",
  "textAlign": "center",
  "fontWeight": "bold",
  "fontSize": "40dp",
  "color": "black",
  "type": "Text"
},
{
  "text": "${payload.myData.subtitle}",
  "textAlign": "center",
  "fontSize": "30dp",
  "color": "black",
  "type": "Text",
  "spacing": "20dp"
},
{
  "items": [
    {
     "source": "${payload.myData.foto}",
     "type": "Image",
     "width": "100%",
     "height": "100%"
     }
  ],
```

Figura 3.32.- Código de la plantilla APL_pistafototexto.json. Fuente: Diseño propio

La segunda carpeta del proyecto es “functions”. En ella se alojan dos archivos distintos con funciones que se utilizan en varias ocasiones en el código principal. De esta manera, se ahorra escribir código importando estas funciones.

El primer archivo de la carpeta “functions” es “comprobar.js” (Figura 3.33). Este archivo contiene “checkAnswer”, una función muy sencilla que recibe dos parámetros: “currentPalabra” y “palabraSolucion”. En primer lugar, crea un string “juego” al que le asigna el valor del campo “solución” de “currentPalabra”, y a continuación convierte a minúsculas tanto “juego” como el parámetro recibido “palabraSolucion”. Si ambas palabras son iguales, la función retorna “true”, en caso contrario, retorna “false”. Finalmente, se exporta la función para poder usarla en otros archivos (“module.exports”).

```
const checkAnswer = function(currentPalabra, palabraSolucion) {
    const juego = new String(currentPalabra.solucion);
    return juego.toLowerCase() === palabraSolucion.toLowerCase();
};

module.exports = {
    checkAnswer
}
```

Figura 3.33.- Código de la función checkAnswer. Fuente: Diseño propio

El segundo archivo de la carpeta "functions" es "servidor.js" (Figura 3.34). En la primera línea se importa el módulo "https" en Node.js, lo que permitirá realizar solicitudes HTTP seguras (utilizando HTTPS). A continuación, se define "postResultados", una función asíncrona que recibe como parámetros "idActividad", "idUsuario", "duración", "numAciertos", "numErrores" y "nivelActividad", y se asignan estos valores, junto con el idSkill, en el objeto "payload", el cuerpo de la petición. Después, se define un objeto "options" en el que se especifican los detalles de la solicitud: en "hostname" se incluye la dirección del servidor, se asigna un número de puerto 443, se define la ruta del recurso en "path" y se indica que el tipo de petición a realizar es POST en "method". El último apartado de "options" son los encabezados, donde se especifica que el "content-type" es "application/json", la longitud del contenido ("content-length"), que se obtiene calculando la longitud de la representación en bytes del cuerpo de la petición, y el tipo de autenticación, ("Basic"), con un esquema de nombre de usuario y contraseña codificados en Base64.

Una vez definida la petición se crea una nueva promesa que recoge la lógica de la solicitud. Utilizando "https.request" se envía la solicitud al servidor con los datos y opciones especificados anteriormente. Los datos se reciben en trozos (chunks). Así, según se van recibiendo (*res.on('data')*), se van agregando uno a uno a returnData, y cuando la respuesta se completa (*res.on('end')*), se resuelve la promesa con éxito (*resolve()*). Si durante el procesamiento de la respuesta se produce un error (*res.on('error')*), se rechaza la promesa indicando el error obtenido (reject(error)). Por último, se llama a *req.write* para asegurarse de que se incluyen los datos de "payload" en la solicitud y se llama a *req.end()* para finalizar la solicitud. Finalmente, en "module.exports", se exporta la función "postResultados" para que pueda ser utilizada en otros archivos.

```
const https = require('https');

const postResultados = async function(idActividad, idUsuario, duracion,
numAciertos, numErrores, nivelActividad) {
  const payload = {
    idSkill: 'c7800aee8f9287a60ffe835984a38431',
    idActividad: idActividad,
    idUsuario: idUsuario,
    duracion: duracion,
    numAciertos: numAciertos,
    numErrores: numErrores,
    nivelActividad: nivelActividad
  };

  const options = {
    hostname: 'www.it.uniovi.es',
    port: 443,
    path: '/hosesbackend/skillTrastornosLectura.php',
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
      'Content-Length': Buffer.byteLength(JSON.stringify(payload)),
      'Authorization': 'Basic ' +
Buffer.from('lfonteriz:aaa1674unTFG').toString('base64'),
    },
  };

  return new Promise((resolve, reject) => {
    const req = https.request(options, (res) => {
      res.setEncoding('utf8');
      let returnData = '';

      res.on('data', (chunk) => {
        returnData += chunk;
      });

      res.on('end', () => {
        resolve();
      });

      res.on('error', (error) => {
        reject(error);
      });
    });

    req.write(JSON.stringify(payload));
    req.end();
  });
};


module.exports = {
    postResultados
}
```

Figura 3.34.- Código del archivo servidor.js. Fuente: Diseño propio

En el archivo “index.js” se encuentra el código principal de la skill. En las primeras líneas (Figura 3.35) se importan dos módulos en Node.js: ‘ask-sdk-core’ (para manejar interacciones personalizadas con Alexa) y ‘https’ (para realizar solicitudes HTTPS y manejar las respuestas obtenidas). Esto permite utilizar las funciones de estos módulos en el resto del código (utilizando “Alexa.” o “https.”). También aquí se define una constante “juegos”. Este array contiene los nombres de los distintos juegos disponibles, y se utilizará para determinar a qué juego quiere jugar el usuario en función de su respuesta.

```
const Alexa = require('ask-sdk-core');
const https = require('https');
const juegos = ['ordena las sílabas', 'cuenta las sílabas', 'las
adivinanzas', 'la sílaba oculta', 'los antónimos', 'repite la palabra'];
```

Figura 3.35.- Primeras líneas de index.js. Fuente: Diseño propio

Todos los handlers, ya sea en la parte final de su código o tras evaluar una condición, incluyen el código de la Figura 3.36. “handlerInput” es un objeto que se utilizará durante el desarrollo del código, ya que representa la entrada de datos proporcionados por el usuario y dispone de muchos métodos para el tratamiento de la información recibida. En este caso se muestra “.responseBuilder”, un objeto que permite construir y personalizar la respuesta que dará Alexa al usuario, y se le especifican los siguientes métodos:

- “.speak(speakOutput)”: Determina qué le tiene que decir Alexa al usuario. En este caso, se le pasa la variable “speakOutput”, una cadena de texto cuyo contenido se modificará en función de la situación.

- “.reprompt(speakOutput)”: También determina qué le tiene que decir Alexa al usuario, solo que en este caso representa un mensaje solicitándole al usuario una respuesta válida.

- “.getResponse()”: Método que devuelve la respuesta construida al usuario.

```
return handlerInput.responseBuilder
            .speak(speakOutput)
            .reprompt(speakOutput)
            .getResponse();
```

Figura 3.36.- Código para personalizar respuesta. Fuente: Diseño propio

Otra parte común a lo largo del código es la referente al uso de plantillas APL (un ejemplo en la Figura 3.37). Dado que no todos los dispositivos Amazon Alexa tienen pantalla, lo primero que se hace es verificar si el dispositivo en el que se está utilizando la skill soporta Alexa Presentation Language (APL). En caso positivo, se importa de la carpeta "documents" el archivo APL que se quiera utilizar mediante "require". A continuación, mediante el método ".addDirective" de "responseBuilder" se indica a Alexa que deberá mostrar un contenido visual en la pantalla. Ese contenido tiene unos parámetros, donde destaca el objeto "datasources: { myData: { ... } }". Como se explicó anteriormente, las plantillas podían personalizarse en cada momento. Así, añadiendo las propiedades correspondientes de la plantilla (en el caso de la Figura 3.37, 'title' y 'body') dentro de "myData", se puede personalizar la muestra en pantalla con resultados obtenidos durante el desarrollo del juego.

```
if (

Alexa.getSupportedInterfaces(handlerInput.requestEnvelope)['Alexa.Present
ation.APL']
          ) {
            const APL_simple = require('./documents/APL_simple.json');
            responseBuilder.addDirective({
              type: 'Alexa.Presentation.APL.RenderDocument',
              token: 'myAPLToken',
              document: APL_simple,
              datasources: {
                myData: {
                  title: title,
                  body: body,
                },
              },
            });
          }
```

Figura 3.37.- Código de ejemplo de uso de plantilla APL. Fuente: Diseño propio

Al principio de cada handler (controlador) hay una condición de entrada. Esta condición se determina en la función "canHandle(handlerInput)" (Figura 3.38). Lo primero

que se muestra en esta imagen es "const sessionAttributes = handlerInput.attributesManager.getSessionAttributes()", muy recurrente a lo largo del código, ya que permite recuperar ciertos atributos almacenados durante la sesión y utilizar su valor en los distintos controladores. Después, para verificar la condición de entrada, lo primero que se hace es comprobar si la solicitud hecha por el usuario es de tipo "IntentRequest" (la solicitud del usuario se corresponde con alguno de los intents definidos para la skill). En "Alexa.getIntentName(handlerInput.requestEnvelope) === 'SeleccionarNivelIntent'" se compara el nombre del intent invocado con la solicitud del usuario con el intent que invoca ese handler (en este caso, el intent es "SeleccionarIntentHandler"). En este caso, se añade entre medias una condición más. Utilizando el atributo de sesión (recuperado previamente con ".getSessionAttributes()") "juegoElegido", se invoca al handler correspondiente con el juego elegido por el usuario (en este caso se compara con juegos[0], que es "ordena las sílabas", por lo que si el usuario eligiese este juego invocaría al handler que incluye el código de la Figura 3.38 en su "canHandle(handlerInput)"). De esta forma, Alexa determina si un controlador puede o no manejar una solicitud.

```
canHandle(handlerInput) {

        const sessionAttributes =
handlerInput.attributesManager.getSessionAttributes();

        return Alexa.getRequestType(handlerInput.requestEnvelope) ===
'IntentRequest'
            && sessionAttributes.juegoElegido === juegos[0]
            && Alexa.getIntentName(handlerInput.requestEnvelope) ===
'SeleccionarNivelIntent'
    },
```

Figura 3.38.- Código de ejemplo de invocación de un handler. Fuente: Diseño propio

Para el registro de usuarios, se realiza una solicitud POST similar a la que se explicó previamente ("postResultados" en "servidor.js"). La diferencia en este caso se encuentra en la respuesta de la promesa (Figura 3.39). Para facilitar la interacción con el usuario, se definen diferentes situaciones en función del código de estado HTTP de la respuesta:

- Código 201: Se indica al usuario que se ha creado su perfil correctamente, y se le dice que repita el id que acaba de registrar para pasar a la pantalla de selección de juego.

- Código 400: Se le indica al usuario que la solicitud ha sido mal formulada, esperando que la repita de nuevo correctamente.

- Código 406: Se le indica al usuario que el id que intenta registrar ya está registrado en la base de datos, esperando que formule de nuevo la solicitud con un id válido.

- Código distinto: Se muestra el mensaje de error no esperado y se rechaza la promesa.

```
res.on('end', () => {
          if (res.statusCode === 201) {

            speakOutput = `Usuario ${sessionAttributes.nombreUsuario}
creado con éxito. Repite tu id para empezar a jugar`;
            title = `Hola ${sessionAttributes.nombreUsuario}`;
            body = 'Registrado con éxito';

          } else if (res.statusCode === 400) {

            speakOutput = `Solicitud mal formulada`;
            title = 'Error 400';
            body = 'Solicitud mal formulada';

          } else if (res.statusCode === 406) {

            speakOutput = `Ya hay un usuario registrado con ese id`;
            title = 'Error 406';
            body = 'ID ya registrado';

          } else {
            reject(new Error(`Error ${res.statusCode}: Error
inesperado`));
          }
```

Figura 3.39.- Análisis de los códigos de estado de respuesta del registro de usuarios. Fuente: Diseño propio

Para el inicio de sesión, lo primero que se hace es recuperar la respuesta del usuario para identificar cuál es su id. Una vez hecho esto, se definen las “options” de la petición GET

de forma similar al POST (cambiando "method" POST por GET). Como la solicitud de tipo GET se realiza en formato querystring, lo que se mofica es el "path" de la petición, añadiendo el campo idUsuario acompañado de '${id}'. Esto permite modificar la URL en función de la respuesta del usuario.

```
const { requestEnvelope } = handlerInput;
    const id =
handlerInput.requestEnvelope.request.intent.slots.idUsuario.value;
    sessionAttributes.idUsuario = id;

    const options = {
      hostname: 'www.it.uniovi.es',
      port: 443,
      path:
`/hosesbackend/skillTrastornosLectura.php?idSkill=c7800aee8f9287a60ffe835
984a38431&idUsuario=${id}`,
      method: 'GET',
      auth: 'lfonteriz:aaa1674unTFG',
      headers: {
        'Content-Type': 'application/json'
      }
    };
```

Figura 3.40.- Formulación de petición GET para inicio de sesión. Fuente: Diseño propio

Una vez planteada la solicitud, se analiza la respuesta de forma similar al registro:

- Código de estado 200 (Figura 3.41): Si se encuentra un usuario registrado en la base de datos con el id proporcionado por el usuario, se convierte la respuesta obtenida en un objeto JavaScript (utilizando "JSON.parse()"). Dado que el objeto respuesta es un array de un solo elemento, se accede al primer y único objeto de esta lista en "usuarioTodo["0"]", que se almacena en "usuario". De esta forma, utilizando "usuario.campo" se puede acceder a los datos asociados a ese id (siendo "campo" "objid", "nombre", "edad" o "fechaRegistro"). Estos valores se guardan en el correspondiente atributo de sesión y se actualizan ("handlerInput.attributesManager.setSessionAttributes(sessionAttributes)") para usarlos en otros controladores mientras dure la sesión.

- Código de estado 404: Se le indica al usuario que no se ha encontrado ningún usuario registrado con ese id, esperando que el usuario responda indicando un id válido.

- Código de estado 400: Se le indica al usuario que la solicitud ha sido mal formulada, esperando que la repita de nuevo correctamente.

```
if (res.statusCode === 200) {

    // Parsea la respuesta JSON
    const usuarioTodo = JSON.parse(responseData);
    const usuario = usuarioTodo["0"];

    // Se encontró un usuario válido
    // Guardamos sus valores
    sessionAttributes.nombreUsuario = usuario.nombreUsuario;
    sessionAttributes.edad = usuario.edad;
    sessionAttributes.idUsuario = usuario.objid;

handlerInput.attributesManager.setSessionAttributes(sessionAttributes);

    speakOutput = `¡Hola ${sessionAttributes.nombreUsuario}! Hay seis
juegos disponibles. ¿A cuál te gustaría jugar?`;

    var APL_juegos = require('./documents/APL_juegos.json');

    if
(Alexa.getSupportedInterfaces(handlerInput.requestEnvelope)['Alexa.Presen
tation.APL']) {
        handlerInput.responseBuilder.addDirective({
            type: 'Alexa.Presentation.APL.RenderDocument',
            document: APL_juegos,
        });
  }
```

Figura 3.41.- Código de estado 200 de inicio de sesión. Fuente: Diseño propio

El manejo de los juegos se realiza en todos de la misma forma. En el primer handler se extrae el valor del nivel proporcionado por el usuario, se asigna ese valor al atributo de sesión correspondiente al nivel de ese juego y se actualizan los atributos de sesión (Figura 3.42).

```
const nivel =
handlerInput.requestEnvelope.request.intent.slots.numNivel.value;
const sessionAttributes =
handlerInput.attributesManager.getSessionAttributes();
sessionAttributes.nivelOrdenarSilabas = nivel;
handlerInput.attributesManager.setSessionAttributes(sessionAttributes);
```

Figura 3.42.- Primeras líneas de código de los handlers de los juegos. Fuente: Diseño propio

A continuación, se define la petición GET para obtener un enunciado de juego. Esta petición tiene el mismo formato que la de inicio de sesión, solo que en este caso los parámetros de la URL son "idActividad" (un número fijo que se modifica en función del juego) y "nivelActividad" (varía en función del atributo de sesión) (Figura 3.43).

```
const options = {
          hostname: 'www.it.uniovi.es',
          port: 443,
          path:
`/hosesbackend/skillTrastornosLectura.php?idSkill=c7800aee8f9287a60ffe835
984a38431&idActividad=1&nivelActividad=${sessionAttributes.nivelOrdenarSi
labas}`,
          method: 'GET',
          auth: 'lfonteriz:aaa1674unTFG',
          headers: {
            'Content-Type': 'application/json'
          }
        };
```

Figura 3.43.- Código de petición GET para enunciado de juego. Fuente: Diseño propio

En el código de la respuesta de la promesa (Figura 3.44), se almacena el valor de la respuesta en el array de JSON "juegosTodo", para después definir una función que devolverá un enunciado aleatorio para que el usuario juegue (en el caso de la Figura 3.44, "getRandomOrdenar"). Esta función recibe como parámetro un array "pastPalabras" en el que se almacenan los enunciados que ya se mostraron al usuario. Al utilizar el método ".filter", se eliminan los elementos obtenidos en la respuesta de la solicitud GET que se encuentren en "pastPalabras" (es decir, los que ya se hayan mostrado al usuario). Una vez filtrados los elementos de la respuesta, se analiza si queda algún elemento que no se haya mostrado ya:

- Si queda algún elemento disponible: Se selecciona un elemento aleatorio del array utilizando los métodos ".floor" y ".random".

- Si no queda ningún elemento disponible: Se devuelve un objeto con valores predeterminados ("0" y "null").

```
const juegosTodo = JSON.parse(responseData);

const getRandomOrdenar = function(pastPalabras = []) {
   const filtrado = Object.values(juegosTodo).filter(p =>
   !pastPalabras.find(pp => pp.objid === p.objid));

    return filtrado.length > 0
       ? filtrado[Math.floor(Math.random() * filtrado.length)]
       : {"objid": 0, "palabra": null, "solucion": null};
                    };
```

Figura 3.44.- Código de función que genera enunciado de juego aleatorio. Fuente: Diseño propio

Una vez hecho esto se entra en la lógica de identificación de nivel (Figura 3.45): una estructura if-else if en la que, tras analizar si la petición GET ha obtenido una respuesta válida ("juegosTodo" tiene un valor asignado), se compara el nivel elegido por el usuario con los distintos niveles disponibles. Si el usuario elige un nivel no disponible para ese juego, se le indicará que ese nivel es incorrecto y se esperará a que elija uno válido.

```
if (juegosTodo){

    if (sessionAttributes.nivelOrdenarSilabas === '1'){

    } else if (sessionAttributes.nivelOrdenarSilabas === '2') {

    } else if (sessionAttributes.nivelOrdenarSilabas === '3') {

    } else if (sessionAttributes.nivelOrdenarSilabas === '4') {

    } else {

        speakOutput = 'Error en el nivel'
        if (
            Alexa.getSupportedInterfaces(handlerInput.requestEnvelope)[
                'Alexa.Presentation.APL'
            ]
            ) {
                handlerInput.responseBuilder.addDirective({
                    type: 'Alexa.Presentation.APL.RenderDocument',
                    document: APL_simple,
                    datasources: {
                        myData: {
                            title: 'Error',
                            body: 'Nivel incorrecto',
                        },
                    },
                });
            }
    }
```

Figura 3.45.- Estructura selección nivel. Fuente: Diseño propio

Si, por el contrario, el usuario elige un nivel válido, se inicia la lógica de los juegos. Lo primero que se hace es comprobar si hay actualmente una palabra para ese nivel de ese juego (Figura 3.46). Para eso, se utiliza la propiedad de JavaScript ".hasOwnProperty()", que sirve para verificar si un objeto tiene una propiedad determinada, y se le pasa "current_wordNivelUnoOrdenar", que representa la palabra o enunciado actual. Si su valor no es "null", significa que el usuario no ha dado una respuesta al enunciado mostrado, por lo que se vuelve a mostrar y a preguntar la respuesta.

```
// Comprobamos si hay palabra actual. Si la hay, repetimos la pregunta y
salimos
if (
    sessionAttributes.hasOwnProperty('current_wordNivelUnoOrdenar') &&
    sessionAttributes.current_wordNivelUnoOrdenar !== null
) {
    speakOutput = `¿Cuál es la palabra oculta en
${sessionAttributes.current_wordNivelUnoOrdenar.palabra}?`;

    if (Alexa.getSupportedInterfaces(handlerInput.requestEnvelope)[
        'Alexa.Presentation.APL'
    ]
    ) {
        handlerInput.responseBuilder.addDirective({
            type: 'Alexa.Presentation.APL.RenderDocument',
            document: APL_pistatexto,
            datasources: {
                myData: {
                    title:
`${sessionAttributes.current_wordNivelUnoOrdenar.palabra}`,
                    subtitle: `Pista:
${sessionAttributes.current_wordNivelUnoOrdenar.pistaPalabra}`,
                    body: 'Es _',
                },
            },
        });
    }
    return handlerInput.responseBuilder
        .speak(speakOutput)
        .reprompt(speakOutput)
        .getResponse();
}
```

Figura 3.46.- Código de comprobación de existencia de palabra actual. Fuente: Diseño propio

Si no hubiese palabra actual se continúa el código. Lo que habría que hacer entonces es generar una palabra aleatoria (Figura 3.47), utilizando para ello la función "getRandom" explicada anteriormente. Una vez que se obtiene un objeto palabra, se puede acceder a los distintos valores que tienen los campos de este objeto para proporcionárselos al usuario (por ejemplo, el campo "pistaPalabra", Figura 3.47).

```
// Importamos las funciones de ordenar sílabas y pedimos una palabra
aleatoria
const palabraNivelUnoOrdenar =
getRandomOrdenar(sessionAttributes.past_wordsNivelUnoOrdenar);
const pistaUnoOrdenar = palabraNivelUnoOrdenar.pistaPalabra;
```

Figura 3.47.- Código de generación de palabra aleatoria y extracción de pistas. Fuente: Diseño propio

Como se explicó en la función “getRandom”, si todas las palabras disponibles ya se mostraron al usuario, se devuelve un objeto con unos valores predeterminados. Como el valor predeterminado del campo “objid” (el identificador único del objeto) es “0”, se comprueba si efectivamente este identificador tiene el valor “0”. De ser así significa que el usuario ha agotado las palabras disponibles, por lo que se le indica que elija otro nivel o juego para poder seguir jugando.

```
// Comprobamos si queda alguna palabra
if (palabraNivelUnoOrdenar.objid === 0) {

    speakOutput = `¡Te has pasado el juego! Elige un nivel u otro juego
para seguir jugando`;

    if (Alexa.getSupportedInterfaces(handlerInput.requestEnvelope)[
        'Alexa.Presentation.APL'
    ]
    ) {
        handlerInput.responseBuilder.addDirective({
            type: 'Alexa.Presentation.APL.RenderDocument',
            document: APL_simple,
            datasources: {
                myData: {
                    title: 'ENHORABUENA',
                    body: 'Te has pasado el juego. Elige un nivel u otro
juego para seguir jugando',
                },
            },
        });
       }
}
```

Figura 3.48.- Comprobación de si quedan palabras disponibles. Fuente: Diseño propio

Si por el contrario el identificador del objeto obtenido es distinto de “0” (Figura 3.49), significa que sí que hay una palabra disponible para que el usuario pueda seguir jugando en ese nivel. Se guarda entonces el objeto obtenido en el atributo de sesión correspondiente para mantenerlo durante el juego hasta que sea necesario modificarlo, y se actualizan los valores de los atributos de sesión. A continuación, se formula al usuario la pregunta y se muestra en pantalla la palabra o enunciado del juego (y, de corresponderse con el nivel, la pista de texto o foto asociada). Finalmente se crea un nuevo objeto de tipo “Date” (“inicio”) y se guarda el valor del tiempo en milisegundos de este objeto utilizando la función “.getTime()” en un

timestamp de la sesión. Este timestamp se calcula en el momento en el que se formula la pregunta al usuario, y servirá para calcular el tiempo que tarda este en dar una respuesta.

```
else {
    // Configuramos el atributo current_word
    sessionAttributes.current_wordNivelUnoOrdenar =
palabraNivelUnoOrdenar;

    // Guardamos los atributos de sesión

handlerInput.attributesManager.setSessionAttributes(sessionAttributes);

    // Hacemos la pregunta
    speakOutput = `¿Qué palabra se esconde en
${palabraNivelUnoOrdenar.palabra}?`;
    if (
        Alexa.getSupportedInterfaces(handlerInput.requestEnvelope)[
            'Alexa.Presentation.APL'
        ]
    ) {
        handlerInput.responseBuilder.addDirective({
            type: 'Alexa.Presentation.APL.RenderDocument',
            document: APL_pistatexto,
            datasources: {
                myData: {
                    title: title,
                    subtitle: subtitle,
                    body: body,
                },
            },
        });
       }

    // Creamos timestamp de inicio
    inicio = new Date();
    sessionAttributes.timestampInicio = inicio.getTime();
handlerInput.attributesManager.setSessionAttributes(sessionAttributes);

}
```

Figura 3.49.- Código de trabajo con palabra aleatoria obtenida. Fuente: Diseño propio

Una vez que se muestra el enunciado, el usuario debe dar una respuesta válida, atendiendo a las frases aceptadas por Alexa en cada juego, para comprobar si ha acertado o no. Lo primero que se hace para comprobar el resultado es obtener el atributo de sesión correspondiente al juego (Figura 3.50), y, en función del obtenido, se evalúa la respuesta del usuario entrando en el if correspondiente (misma estructura if-else if que en controlador anterior).

```
const sessionAttributes =
handlerInput.attributesManager.getSessionAttributes();
const nivelOrdenarComparar = sessionAttributes.nivelOrdenarSilabas;
```

Figura 3.50.- Extracción de nivel para identificar si la respuesta es correcta o no. Fuente: Diseño propio

Una vez hecho esto se procesa la respuesta del usuario (Figura 3.51), obteniendo la solución dada por ese mediante el valor del "slot" del intent que invoca al controlador (en el caso de la Figura 3.51, "respuestaPalabra"). En ese momento se crea otro objeto "Date" llamado "final", se obtiene su valor en milisegundos de la misma forma que en "inicio" y se guarda su valor en el atributo de sesión correspondiente. Haciendo la resta entre ambos atributos de sesión, y dividiendo el resultado entre 1000, se obtiene el tiempo en segundos que tarda el usuario en dar una respuesta desde que se formula el enunciado.

Para comprobar la respuesta, se importan las funciones del archivo "comprobar.js". Con esto se puede utilizar la función "checkAnswer" de este archivo, pasándole como parámetros el objeto obtenido aleatoriamente y la respuesta dada por el usuario. Internamente, la función comprueba si la respuesta del usuario es correcta o no.

El siguiente paso es añadir, mediante el método ".push()", el objeto actual a la lista de objetos que ya han aparecido para evitar que vuelva a ofrecérsele al usuario. Además, utilizando el método ".hasOwnProperty()", se comprueba si existe una propiedad que lleve la cuenta de los aciertos del usuario, y, en caso de no existir, la crea y se inicializa a 0.

```
// Obtenemos la palabra dicha por el usuario
var respuesta_usuarioNivelUnoOrdenar =
handlerInput.requestEnvelope.request.intent.slots.respuestaPalabra.value;
final = new Date();
sessionAttributes.timestampFinal = final.getTime();
duracionMili = sessionAttributes.timestampFinal -
sessionAttributes.timestampInicio;
duracion = Math.floor(duracionMili / 1000);

// Comprobamos la respuesta
const comprobarNivelUnoOrdenar = require('./functions/comprobar.js');

const aciertoNivelUnoOrdenar = comprobarNivelUnoOrdenar.checkAnswer(
    sessionAttributes.current_wordNivelUnoOrdenar,
    respuesta_usuarioNivelUnoOrdenar
);

// Añadimos la palabra a la lista de palabras que ya aparecieron
// Almacenamos el valor para el resto de la función y cambiamos la
palabra actual a null
sessionAttributes.past_wordsNivelUnoOrdenar.push(sessionAttributes.curren
t_wordNivelUnoOrdenar);
const palabra_mostrarNivelUnoOrdenar =
sessionAttributes.current_wordNivelUnoOrdenar.palabra;
const palabra_solucionNivelUnoOrdenar =
sessionAttributes.current_wordNivelUnoOrdenar.solucion;
sessionAttributes.current_wordNivelUnoOrdenar = null;

// Comprobamos si hay puntuación y si no la inicializamos
if (!sessionAttributes.hasOwnProperty('puntuacionNivelUnoOrdenar'))
    sessionAttributes.puntuacionNivelUnoOrdenar = 0;
```

Figura 3.51.- Código de comprobación de la respuesta del usuario al juego. Fuente: Diseño propio

Una vez hecho esto se comprueba si el usuario ha dado una respuesta correcta o no. Si ha acertado, se incrementa el atributo de sesión correspondiente a la puntuación del nivel y se asigna un “1” a la variable “acierto” y un “0” a la variable “error”. Si por el contrario el usuario da una respuesta incorrecta, se asigna un “0” a la variable “acierto” y un “1” a la variable “error”. Tras esto, en ambos casos se proporciona retroalimentación para que el usuario sepa si su respuesta es correcta o no.

```
// Comprobación respuesta
if (aciertoNivelUnoOrdenar) {
    sessionAttributes.puntuacionNivelUnoOrdenar += 1
    acierto = 1;
    error = 0;
    speakOutput = `¡Wow! ¡Adivinaste que
${palabra_solucionNivelUnoOrdenar} se escondía en
${palabra_mostrarNivelUnoOrdenar}.
        Ahora tu puntuación del nivel 1 es
${sessionAttributes.puntuacionNivelUnoOrdenar}. Elige un nivel u otro
juego para seguir jugando`;

    if (
        Alexa.getSupportedInterfaces(handlerInput.requestEnvelope)[
            'Alexa.Presentation.APL'
        ]
    ) {
            handlerInput.responseBuilder.addDirective({
                type: 'Alexa.Presentation.APL.RenderDocument',
                document: APL_simple,
                datasources: {
                    myData: {
                        title: '¡Enhorabuena!',
                        body: 'Elige un nivel u otro juego',
                    },
                },
            });

        }

} else {
    speakOutput = `Vaya... No has encontrado la palabra escondida en
${palabra_mostrarNivelUnoOrdenar}. Elige un nivel u otro juego para
seguir jugando`;
    acierto = 0;
    error = 1;

    if (
        Alexa.getSupportedInterfaces(handlerInput.requestEnvelope)[
            'Alexa.Presentation.APL'
        ]
    ) {
        handlerInput.responseBuilder.addDirective({
            type: 'Alexa.Presentation.APL.RenderDocument',
            document: APL_simple,
            datasources: {
                myData: {
                    title: 'Oh, vaya',
                    body: 'Elige un nivel u otro juego',
                },
            },
        });
       }
}
```

Figura 3.52.- Código de análisis acierto/error. Fuente: Diseño propio

En la última parte del controlador (Figura 3.53), se actualizan los atributos de sesión, se importan las funciones de "servidor.js" y se suben los resultados obtenidos por el usuario a la base de datos. Para esto se utiliza la función ".postResultados()" definida en ese archivo, pasándole como parámetros el id del juego (un número fijo que cambia según el juego), el atributo de sesión correspondiente al id de usuario, el valor obtenido con los timestamps de tiempo de respuesta del usuario, los valores que hayan tomado "acierto" y "error", y el id del nivel (también un número fijo que se modifica en cada "else-if"). Finalmente, se utiliza ".then()" para mostrar un mensaje en consola de éxito en la subida de los resultados y ".catch()" para mostrar el error en la consola en caso de fallo en la subida.

```
// Almacenamos y actualizamos los datos de la sesión
handlerInput.attributesManager.setSessionAttributes(sessionAttributes);
const servidor = require('./functions/servidor.js');
servidor.postResultados(1, sessionAttributes.idUsuario, duracion,
acierto, error, 1)
    .then(() => {
        // La solicitud se completó con éxito
        console.log('Solicitud de resultados enviada correctamente');
    })
    .catch((error) => {
        // Se produjo un error al realizar la solicitud
        console.error('Error al enviar la solicitud de resultados:',
error);
    });
```

Figura 3.53.- Código de subida de resultados a la base de datos. Fuente: Diseño propio

Finalmente, es necesario exportar el controlador de la skill (Figura 3.54), ya que actúa como punto de entrada de esta. Es necesario incluir todos los controladores anteriormente definidos, y en el orden en el que fueron definidos, ya que se procesan de arriba abajo.

```
exports.handler = Alexa.SkillBuilders.custom()
    .addRequestHandlers(
        LaunchRequestHandler,
        RegistroUsuarioIntentHandler,
        CrearUsuarioIntentHandler,
        UsuarioRegistradoIntentHandler,
        BienvenidaIntentHandler,
        SeleccionarJuegoIntentHandler,
        OrdenarSilabasHandler,
        OrdenarSilabasIntentHandler,
        ContarSilabasHandler,
        ContarSilabasIntentHandler,
        AdivinanzasHandler,
        AdivinanzasIntentHandler,
        SilabaOcultaHandler,
        SilabaOcultaIntentHandler,
        AntonimosHandler,
        AntonimosIntentHandler,
        RepitePalabraHandler,
        RepitePalabraIntentHandler,
        HelloWorldIntentHandler,
        HelpIntentHandler,
        CancelAndStopIntentHandler,
        FallbackIntentHandler,
        SessionEndedRequestHandler,
        IntentReflectorHandler)
    .addErrorHandlers(
        ErrorHandler)
    .withCustomUserAgent('sample/hello-world/v1.2')
    .lambda();
```

Figura 3.54.- Código final de index.js. Fuente: Diseño propio

3.3.5.2.- Código de la aplicación web

El primer paso para hablar del código de la aplicación web es explicar la estructura del proyecto. Como se puede ver en la Figura 3.55, el proyecto se aloja en una carpeta “herramientalogopeda”, dentro de la que hay varias carpetas y archivos.

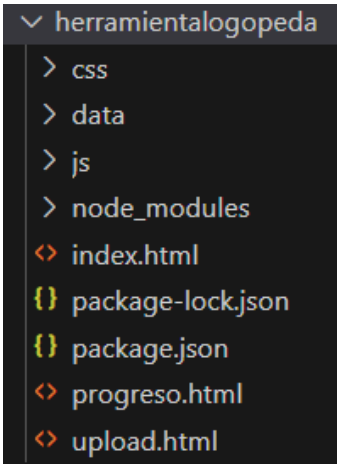

Figura 3.55.- Estructura del proyecto de la aplicación web. Fuente: Diseño propio

Lo primero es la carpeta “css”. En esta carpeta se alojan las hojas de estilo del proyecto, y se encuentran algunos archivos ya predefinidos de Bootstrap, además de uno de diseño propio, “estilos.css”.

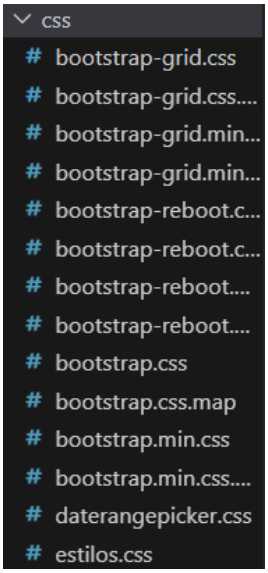

Figura 3.56.- Contenido de la carpeta "css". Fuente: Diseño propio

En “estilos.css” se encuentran definidos ciertas personalizaciones de la página web, como, por ejemplo, el estilo de los botones de la página principal:

Figura 3.57.- Muestra botón página principal. Fuente: Diseño propio

```
.boton_2 {
  text-decoration: none;
  padding: 15px 50px;
  font-size: 1.5rem;
  position: relative;
  margin: 10px;
  background-color: #FFFFF2;
  color: #103C76;
  border-radius: 50px;
  border: 3px solid #103C76;
  transition: transform 0.3s ease;
  font-family: Arial, Helvetica, sans-serif;
}

.boton_2:hover{
  text-decoration: none;
  background-color: #103C76;
  color: #FFFFF2;
}
```

Figura 3.58.- Código que define el botón. Fuente: Diseño propio

El código de la Figura 3.58 elimina el subrayado del texto del botón, define un relleno de 15px arriba y abajo y de 50px a izquierda y derecha, establece el texto del botón a un tamaño de 1.5rem, un color azul de código #103C76 y la fuente de texto como la familia de fuentes “Arial”, “Helvetica” y otras fuentes genéricas sans-serif, establece un borde de 50px de radio, 3px de grosor y el mismo color que el texto, un color de fondo amarillo con código #FFFFF2, y, finalmente, establece una transición de 0.3 segundos cuando el botón cambia de estado. Además, el apartado “:hover”, describe la apariencia del botón cuando el cursor del ratón se sitúa encima. En este caso simplemente se pondría el botón entero del anterior color del texto y el borde y el texto pasaría a ser del amarillo del fondo anterior.

La siguiente carpeta sería “data”. En esta carpeta se almacenan dos archivos: “logo.ico”, que contiene el logo de la aplicación para el favicon, y “logo.png”, que contiene el

logo de la aplicación en formato imagen para su uso en la página web (por ejemplo, el logo de la barra de navegación).

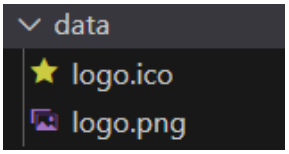

Figura 3.59.- Contenido de la carpeta data. Fuente: Diseño propio

La tercera carpeta es “js”. En esta carpeta se alojan archivos JavaScript necesarios para el correcto funcionamiento de la página web. Se encuentran, principalmente, archivos del framework Bootstrap (por ejemplo, los que permiten que cuando se reduce la pantalla funcione el botón desplegable de la barra de navegación), de la biblioteca jQuery (que permiten añadir animaciones a la página web) y Moment (biblioteca que permite trabajar con fechas, utilizada para las solicitudes de progreso del usuario).

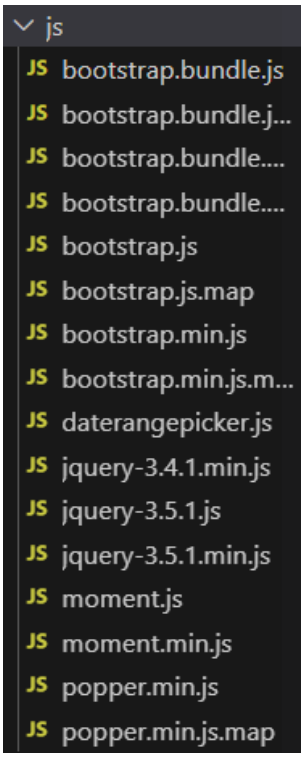

Figura 3.60.- Contenido de la carpeta js. Fuente: Diseño propio

La última carpeta es “node_modules”. Esta carpeta se crea automáticamente al instalar Node.js en el proyecto. En ella se encuentran todas las dependencias y paquetes que han sido instalados en el proyecto.

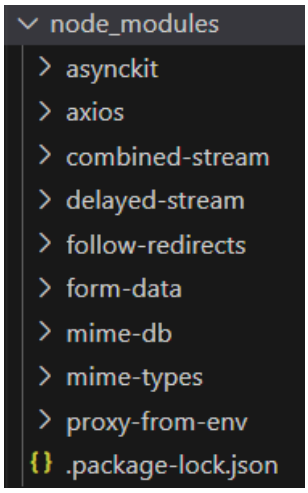

Figura 3.61.- Contenido de la carpeta node_modules. Fuente: Diseño propio

A continuación, ya se encuentran los archivos. El primero es “package-lock.json”, un archivo que se genera automáticamente por Node Package Manager (npm) cuando se intalan las dependencias de un proyecto. En él se detalla información sobre las versiones de los paquetes y dependencias del proyecto.

El siguiente es “package.json” (Figura 3.62). Se trata de un archivo de configuración utilizado en proyectos de Node.js. Contiene metadatos y configuraciones importantes relacionadas con el proyecto, así como la lista de dependencias y scripts de inicio.

```
{
  "name": "herramientalogopeda",
  "version": "1.0.0",
  "description": "Herramienta para el logopeda",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Lucía Fonteriz",
  "license": "ISC",
  "dependencies": {
    "axios": "^1.4.0"
  }
}
```

Figura 3.62.- Código de package.json. Fuente: Diseño propio

En este código no se ha modificado nada de las configuraciones por defecto, salvo la descripción y el autor. Además, al instalarse "axios" para su uso en el proyecto, se añade como dependencia.

Los siguientes archivos son las distintas páginas de la web ("index.html", "upload.html" y "progreso.html"). Una parte similar en todas ellas es el encabezado (en cada caso adaptado con el título o archivos requeridos, Figura 3.63). En primer lugar, se define la codificación de caracteres utilizada (UTF-8), y se definen las propiedades de visualización del contenido en dispositivos móviles (se establece el ancho del "viewport" coincidiendo con el de la pantalla del dispositivo y se establece su escala inicial a 1). Después, se importan las hojas de estilos que se van a necesitar para esa página mediante elementos "link" (en el caso de "index.html", se importan un css de Bootstrap y una hoja de estilos personalizada, "estilos.css"), y con otro elemento también de tipo "link" se define el icono del sitio web. A continuación, con la etiqueta "title", se define el título del documento HTML, que será el que se muestre en la barra de título del navegador. Finalmente, se enlazan archivos externos JavaScript que contienen funciones necesarias para ciertos componentes de la página mediante la etiqueta "script" (en el caso de "index.html", se importan la librería jQuery y el framework Bootstrap).

```
<!DOCTYPE html>
<html lang="es">
  <head>

    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <link rel="stylesheet" href="./css/bootstrap.css">
    <link rel="stylesheet" href="./css/estilos.css">
    <link rel="icon" type="image/icon" href="./data/logo.ico"/>

    <title>HablaConmigo | Inicio</title>

    <script src="./js/jquery-3.5.1.min.js"></script>
    <script src="./js/bootstrap.js"></script>

  </head>
```

Figura 3.63.- Encabezado de index.html. Fuente: Diseño propio

Otra parte que comparten del código HTML es la correspondiente con la barra de navegación (Figura 3.64).

```
<nav class="navbar navbar-expand-md navbar-dark" aria-label="Fourth
navbar example" style="background-color: #FFFFF2;">
      <div class="container-fluid">

        <a href="index.html">
          <img id="logo" src="data/logo.png" width="154" height="100">
        </a>

        <button class="navbar-toggler" style="background-color: #103C76;"
type="button" data-toggle="collapse" data-target="#navbarsExample04"
        aria-controls="navbarsExample04" aria-expanded="false" aria-
label="Toggle navigation">
          <span class="navbar-toggler-icon" style="background-color:
#103C76;"></span>
        </button>

        <div class="collapse navbar-collapse" id="navbarsExample04">
          <ul class="navbar-nav ml-auto mb-2 mb-md-0">
            <li class="nav-item">
              <a class="nav-link texto_barra" href="upload.html">Nuevo
juego</a>
            </li>
            <li class="nav-item">
              <a class="nav-link texto_barra" href="progreso.html">Ver
progreso</a>
            </li>
          </ul>
        </div>
      </div>
    </nav>
```

Figura 3.64.- Código HTML de la barra de navegación. Fuente: Diseño propio

Para el desarrollo de la barra de navegación se han utilizado clases predefinidas de Bootstrap (“navbar”, “navbar-expand-md” y “navbar-dark”), modificando manualmente su color (style="background-color: #FFFFF2;"). Después se establece un “div” que actúa como contenedor fluido, lo que proporciona el diseño “responsive” de la barra de navegación. Una vez hecho esto se definen los elementos que forman parte de la barra. El primer enlace, situado en la parte izquierda de la barra, sería una imagen que se corresponde con el logo de la aplicación y que apuntaría a la página principal (“index.html”). Lo siguiente que se define es el botón que se encargará de mostrar u ocultar los demás elementos de la barra cuando la pantalla sea más pequeña. Para él también se ha utilizado una clase de Bootstrap (“navbar-toogler), modificando el color del fondo para adaptarlo a la paleta de colores de la página. Finalmente se establecen los elementos que agrupará este botón: los botones de “Nuevo juego” y de “Ver progreso” (se conectan con ese botón a través del id “navbarExample04”

que había sido definido con anterioridad). Para estos elementos se crea una lista no ordenada (“<ul>”) con los elementos alineados a la derecha (“ml-auto”). Después se establecen dos elementos de lista (“<li>”) iguales: enlaces (“<a>”) identificados con la clase “nav-item” y “texto-barra” (este segundo definido su estilo en “estilos.css”), e identificada su referencia en el elemento “href” (“upload.html” para “Nuevo juego” y “progreso.html” para “Ver progreso”). Antes del cierre del elemento (“”), se indica el texto que debe mostrar el enlace.

En “index.html”, el resto del código HTML es muy sencillo (Figura 3.65). Todo se engloba en un contenedor con formato “content_index” (definido en “estilos.css”), con una fila (clase “row” de Bootstrap, que permite dividir el contenido en columnas) y una columna ("col-lg-12 col-md-12 col-sm-12"). Esta configuración de la columna utiliza el sistema *grid* de Bootstrap. Al asignar un 12 a cada tamaño de pantalla, se le indica que independientemente del tamaño, el contenido siempre debe ocupar toda la pantalla.

```
<div class="container content_index">

      <div class="row">
        <div class="col-lg-12col-md-12 col-sm-12">

          <div>
            <img src="data/logo.png" alt="HablaConmigo" width="340"
height="227">
          </div>

          <br>

          <h1>HERRAMIENTA PARA EL LOGOPEDA</h1>

          <br>

          <div class="container">

            <div class="row">
              <div class="col-lg-6 col-md-12 col-sm-12">

               <a href="upload.html" class="boton_2" href>Nuevo juego</a>


              </div>

              <br><br>

              <div class="col-lg-6 col-md-12 col-sm-12">

                <a href="progreso.html" class="boton_2">Ver progreso</a>

              </div>

              <br><br>

            </div>
          </div>
        </div>
      </div>

    </div>
```

Figura 3.65.- Código HTML de index.html. Fuente: Diseño propio

El primer elemento de este contenedor es un div que contiene una imagen (el logo de la aplicación), junto con la que se define un texto alternativo a mostrar en el caso de que esta no se cargase correctamente y se fijan sus dimensiones. A continuación, se muestra el texto "HERRAMIENTA PARA EL LOGOPEDA" con el formato de texto de encabezado de nivel 1, y, finalmente otro contenedor para los botones. Para este contenedor se crea una fila, pero, en lugar de una columna, se crean 2 si el tamaño de la pantalla es "lg" ("col-lg-6 col-md-12

col-sm-12"). De esta forma se consigue que los 2 botones que enlazan a las páginas de nueva entrada y consulta de progreso se ocupen la mitad del ancho disponible si la pantalla es grande, pero que en el momento que se alcance el tamaño de pantalla “md”, se sitúen uno encima de otro y pasen a ocupar cada uno todo el ancho disponible. Cada uno de estos botones es un elemento de tipo enlace, que referencia al html correspondiente y con el estilo de la clase “botón_2” (definido en “estilos.css”).

En el archivo “upload.html”, tras la barra de navegación, se establece un “container-fluid”, de forma que todo el contenido ocupe toda la página. Tras él, se define una fila (clase “row” de Bootstrap) y, al igual que con los botones de “index.html”, se definen dos columnas que ocupen 6 espacios para el tamaño de pantalla “lg” y 12 para el resto. Es decir, se van a tener dos elementos que en tamaño de pantalla grande se sitúan pareados pero que en el momento que la pantalla alcance el tamaño “md” se situarán uno encima de otro ocupando todo el ancho disponible.

El primer elemento es una tabla (Figura 3.66) que sirve como referencia al logopeda para tener una visión de la relación ID Juego-Nombre-Número de niveles. Para ello se utilizan las clases “table” y “table-bordered” de Bootstrap, usando “style” para personalizarlos. A continuación, se definen en “thead” los encabezados de las columnas y su correspondiente color de fondo, y, finalmente, se crean 6 elementos “tr” que se corresponden con las 6 filas de la tabla y sus valores.

```
<table class="table table-bordered" style="margin-top: 30px; margin-left:
40px;">
    <thead>
        <tr>
            <th style="background-color:#5271FF;">ID Juego</th>
            <th style="background-color:#F06525;">Nombre Juego</th>
            <th style="background-color:#51B59F;">Número Niveles</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>1</td>
            <td>Ordena las sílabas</td>
            <td>4</td>
        </tr>
        <tr>
            <td>2</td>
            <td>Cuenta las sílabas</td>
            <td>2</td>
        </tr>
        <tr>
            <td>3</td>
            <td>Adivinanzas</td>
            <td>3</td>
        </tr>
        <tr>
            <td>4</td>
            <td>Adivina la sílaba que falta</td>
            <td>1</td>
        </tr>
        <tr>
            <td>5</td>
            <td>Antónimos</td>
            <td>2</td>
        </tr>
        <tr>
            <td>6</td>
            <td>Repite la palabra</td>
            <td>4</td>
        </tr>
    </tbody>
</table>
```

Figura 3.66.- Código de la tabla de upload.html. Fuente: Diseño propio

El siguiente elemento es un formulario (Figura 3.67). Este formulario tiene aplicado el estilo de la clase “form_upload”, definido en “estilos.css”. El primer elemento que muestra es el texto “NUEVO JUEGO”, al que le aplica el estilo de la clase “texto_formulario” (también definida en “estilos_css”) y un párrafo explicativo de la finalidad del formulario con el formato de texto de cuerpo. A continuación, define 6 elementos iguales: una columna que ocupa todo el ancho de la pantalla en todos los tamaños de pantalla, el campo como tipo texto

aplicándole el estilo “form-control” de Bootstrap, le asigna un id para acceder posteriormente a su valor, se establece el texto que debe mostrar como guía para el usuario, y, en el caso de los 4 primeros, se incluye el atributo “required” para indicar que el campo es obligatorio. Tras los campos del formulario, se incluye un botón con el estilo de la clase “boton_4” (definido en “estilos.css”), un id que lo identifica y el texto “Enviar”. Finalmente se incluye un “div” vacío, con el id “mensajeEstado”, que se utilizará para mostrar un mensaje de éxito o error en la subida cuando se pulse el botón de enviar.

```
<form id="form_upload" class="form_upload" style="justify-content:
center; align-items: center;">
    <div class="texto_formulario">NUEVO JUEGO</div>
    <br>
    <p>Introduce, al menos, el ID del juego, la palabra o adivinanza, su
solución y el nivel</p>

    <div class="form-row align-items-center">
        <div class="col-lg-12 col-md-12 col-sm-12 my-1">
            <input type="text" class="form-control" id="idJuego"
placeholder="ID del juego" required>
        </div>
    </div>

    <div class="form-row align-items-center">
        <div class="col-lg-12 col-md-12 col-sm-12 my-1">
            <input type="text" class="form-control" id="palabra"
placeholder="Palabra, adivinanza" required>
        </div>
    </div>

    <div class="form-row align-items-center">
        <div class="col-lg-12 col-md-12 col-sm-12 my-1">
            <input type="text" class="form-control" id="solucion"
placeholder="Solución" required>
        </div>
    </div>

    <div class="form-row align-items-center">
        <div class="col-lg-12 col-md-12 col-sm-12 my-1">
            <input type="text" class="form-control" id="nivel"
placeholder="Nivel" required>
        </div>
    </div>

    <div class="form-row align-items-center">
        <div class="col-lg-12 col-md-12 col-sm-12 my-1">
            <input type="text" class="form-control" id="pistatexto"
placeholder="Pista de texto">
        </div>
    </div>

    <div class="form-row align-items-center">
        <div class="col-lg-12 col-md-12 col-sm-12 my-1">
            <input type="text" class="form-control" id="pistaimagen"
placeholder="Pista de imagen (introducir URL)">
        </div>
    </div>
    <br>
    <input type="button" class="boton_4" id="postJuego" value="Enviar">

    <div>
        <p></p><div id="mensajeEstado"></div>
    </div>
</form>
```

Figura 3.67.- Código del formulario de upload.html. Fuente: Diseño propio

La última parte de código de este archivo es un script para la subida del juego (Figura 3.68). Este código comienza definiendo la URL a la que se realizará la solicitud POST (y a la que, por tanto, se enviarán los datos del nuevo juego). Tras esto, se crea una constante "postJuego" que referencia al id "postJuego" (lo que permite detectar cuando se hace clic en ese botón) y se define la función "handlePOSTJuegoButton", que se ejecutará cuando se haga clic en el botón de enviar. A continuación, se define el funcionamiento de "handlePOSTJuegoButton". Primero, se obtienen los distintos valores de los campos del formulario. Como el campo de pista de texto y de imagen no son obligatorios, se hace una comprobación de si están en blanco, y en caso de que así sea, se les asigna el valor "N.A" (No Aplica). El siguiente paso es realizar la solitud POST a la URL especificada. Para ello se utiliza la función "fetch", en la que se configuran los encabezados (tipo de contenido y autorización) y se asignan en formato JSON los datos del juego en el cuerpo de la solicitud. Una vez hecho esto se analiza el resultado de la operación: si el código de estado HTTP devuelto es 201 (Created) se muestra en el campo mensajeEstado el texto "Juego subido correctamente". En caso contrario, se muestra el texto "Error al subir el texto". Finalmente, si ocurre algún error, se captura en ".catch()" y se muestra en la consola el error.

```
<script>
        const url =
"https://www.it.uniovi.es/hosesbackend/skillTrastornosLectura.php";
        const postJuego = document.getElementById("postJuego");
        postJuego.onclick = handlePOSTJuegoButton;

        function handlePOSTJuegoButton() {
          var idActividad = document.getElementById('idJuego').value;
          var palabra = document.getElementById('palabra').value;
          var solucion = document.getElementById('solucion').value;
          var pistaPalabra = document.getElementById('pistatexto').value;
          var pistaFoto = document.getElementById('pistaimagen').value;
          var nivelActividad = document.getElementById('nivel').value;

          if (pistaPalabra.trim() === '') {
            pistaPalabra = 'N.A';
          }

          if (pistaFoto.trim() === '') {
            pistaFoto = 'N.A';
          }

          // Solicitud a la API REST vía método POST
          fetch(url, {
            method: 'POST',
            port: 443,
            headers: {
              'Content-Type': 'application/json',
              //'Content-Length': Buffer.byteLength(JSON.stringify(body)),
              'Authorization': 'Basic ' + btoa('lfonteriz:aaa1674unTFG'),
            },

            // Datos del juego
            body: JSON.stringify({
              idSkill: 'c7800aee8f9287a60ffe835984a38431',
              idActividad: idActividad,
              palabra: palabra,
              solucion: solucion,
              pistaPalabra: pistaPalabra,
              pistaFoto: pistaFoto,
              nivelActividad: nivelActividad
            })
          })
          .then(response => {
            if (response.status === 201) {
              document.getElementById("mensajeEstado").innerHTML = 'Juego subido
correctamente';
            } else {
              document.getElementById("mensajeEstado").innerHTML = 'Error al
subir el juego';
            }
          })
          // Procesamiento de errores
          .catch(err => {
            // Depuración en consola del navegador
            console.error(err);
          });
        }

</script>
```

Figura 3.68.- Script para la subida de juego en upload.html. Fuente: Diseño propio

El último archivo de la aplicación web es “progreso.html” (Figura 3.69). Sigue una estructura muy similar a “upload.html”: tras la barra de navegación, un contenedor general, formado, en primer lugar, por una fila y la misma estructura de la columna. Al igual que para agregar nuevas entradas, lo primero que se encuentra es una tabla con la relación ID Juego-Nombre-Número de niveles y un formulario (que sigue también el mismo estilo). Este formulario también sigue la misma estructura que el de subida de archivos, con la diferencia de que tiene 2 campos (fecha y hora de inicio y fecha y hora de fin) que no son para introducir texto, sino que al hacer clic en ellos se abre un calendario que permite seleccionar fecha y hora. Además, estos campos no son obligatorios para realizar la consulta (los 3 campos de texto sí que son obligatorios). Por último, al igual que en el otro formulario, se incluye un botón de la misma clase que en este caso muestra el texto “Obtener resultados”.

```
<form class="form_upload" style="justify-content: center; align-items:
center;">
    <div class="texto_formulario">VER PROGRESO</div>
    <br>
    <p>Introduce el ID del usuario y selecciona ID del juego, nivel y
fecha de inicio y de fin</p>

    <div class="form-row align-items-center">
        <div class="col-lg-12 col-md-12 col-sm-12 my-1">
            <input type="text" class="form-control" id="usuario"
placeholder="ID del usuario" required>
        </div>
    </div>

    <div class="form-row align-items-center">
        <div class="col-lg-12 col-md-12 col-sm-12 my-1">
            <input type="text" class="form-control" id="idjuego"
placeholder="ID del juego" required>
        </div>
    </div>

    <div class="form-row align-items-center">
        <div class="col-lg-12 col-md-12 col-sm-12 my-1">
            <input type="text" class="form-control" id="nivel"
placeholder="Nivel" required>
        </div>
    </div>

    <div class="form-row align-items-center">
        <div class="col-lg-12 col-md-12 col-sm-12 my-1">
            <input type="text" class="form-control" id="fechaInicio"
placeholder="Fecha y hora de inicio">
        </div>
    </div>

    <div class="form-row align-items-center">
        <div class="col-lg-12 col-md-12 col-sm-12 my-1">
            <input type="text" class="form-control" id="fechaFin"
placeholder="Fecha y hora de fin">
        </div>
    </div>

    <br>

    <input type="button" class="boton_4" id="getResultados"
value="Obtener resultados">

    <div>
        <p></p><div id="mensajeEstado"></div>
    </div>

</form>
```

Figura 3.69.- Código del formulario de progreso.html. Fuente: Diseño propio

La principal diferencia con la página de subida de archivos es que a continuación de la tabla y el formulario, en la página de consulta de progreso se incluye otro contenedor para mostrar el resultado de la consulta, situado en el centro de la pantalla y que ocupa todo su ancho para todos los tamaños de pantallas (Figura 3.70). Aquí se añade un elemento con el mismo estilo que los formularios (clase “form_upload”, definida en “estilos.css”), con un texto de la clase “texto_formulario” (también definida en “estilos.css”) mostrando la frase “RESULTADOS DE LA CONSULTA”. Tras esto se incluye un elemento “ ” que engloba todos los elementos del formulario, cuya estructura sería, para cada elemento, una frase con el formato de texto de cuerpo indicando qué resultado se va a mostrar acompañado de un “ ” vacío con un id del elemento, donde se mostrará el valor correspondiente obtenido.

```
<div class="container-fluid">
    <div class="row">
        <div class="col-lg-12 col-md-12 col-sm-12" style="display: flex;
align-items: center; justify-content: center;">
            <div class="form_upload" style="justify-content: center;
align-items: center;">

                <div class="texto_formulario">RESULTADOS DE LA
CONSULTA</div>

                <br>

                <div style="color: #103C76; font-family: Arial,
Helvetica, sans-serif; text-align: left;">
                    <div>
                      <p>ID del usuario: </p><div
id="idUsuarioRespuesta"></div>
                    </div>
                    <br>
                    <div>
                      <p>ID del juego: </p><div
id="idJuegoRespuesta"></div>
                    </div>
                    <br>
                    <div>
                      <p>Nivel del juego: </p><div
id="nivelRespuesta"></div>
                    </div>
                    <br>
                    <div>
                      <p>Número de aciertos: </p><div
id="numAciertosRespuesta"></div>
                    </div>
                    <br>
                    <div>
                      <p>Número de errores: </p><div
id="numErroresRespuesta"></div>
                    </div>
                    <br>
                    <div>
                      <p>Tiempo medio de respuesta: </p><div
id="tiempoRespuesta"></div>
                    </div>

                </div>

            </div>

        </div>

    </div>

</div>
```

Figura 3.70.- Código de la muestra de resultados. Fuente: Diseño propio

Finalmente, el archivo “progreso.html” incluye tres scripts, dos que importan funciones de archivos JavaScript (flatpickr, importado desde una dirección web, para seleccionar fecha y hora en el formulario, y Moment, cuyo archivo se aloja en la carpeta “js” y se utiliza para manipular las fechas seleccionadas en el calendario) y un tercero de creación propio para realizar la solicitud de datos (Figuras 3.71 y 3.72).

En primer lugar, utiliza la biblioteca “flatpickr para” crear dos calendarios interactivos, que son los que se muestran en los campos de fecha de inicio y de fin del formulario. Después, se define la URL a la que se realizará la solicitud GET y se crea una constante “getResultados”, que referencia al id “getResultados” (lo que permite detectar cuando se hace clic en ese botón). A continuación, se define la función “handleGETResultadosButton”, que se ejecutará cuando se haga clic en el botón de solicitar datos. Esta función hace lo siguiente:

- Obtiene los valores de los distintos campos del formulario.
- Actualiza la URL concatenando los parámetros del id del juego, el nivel y el id del usuario (es decir, los campos obligatorios).
- Comprueba si se ha seleccionado alguna fecha, y en caso de ser así, la añade también a la URL. Para esto se utiliza Moment.js, ya que las fechas deben convertirse a un formato determinado (“YYYY-MM-DD hh:mm:ss):
- Utilizando “fetch”, se realiza la solicitud a la URL obtenida finalmente. En esta solicitud se añaden también los encabezados requeridos (tipo de contenido y autenticación).
- Una vez hecha la solicitud, se procesa la respuesta obtenida en formato JSON. En primer lugar, se calcula la suma de los aciertos y errores del usuario (suma de los campos “numAciertos” y “numErrores” de cada objeto en el JSON), y, en segundo lugar, se realiza el cálculo del tiempo medio de respuesta (resultado de realizar la media del campo “duración” de todos los objetos del JSON respuesta).
- Finalmente, se actualiza el contenido de los elementos HTML en los que se debe mostrar la respuesta (elementos que anteriormente se explicó que se mostraban vacíos y al hacer la solicitud se mostraban los resultados). Así, los resultados son visibles en la aplicación para el usuario que realiza la petición.

```
<script>
        const fechaInicioElegida = flatpickr("#fechaInicio", {
          enableTime: true,
          dateFormat: "Y/m/d H:i:S",
        });

        const fechaFinElegida = flatpickr("#fechaFin", {
          enableTime: true,
          dateFormat: "Y/m/d H:i:S",
        });

        var url =
"https://www.it.uniovi.es/hosesbackend/skillTrastornosLectura.php?idSkill
=c7800aee8f9287a60ffe835984a38431";
        const getResultados = document.getElementById("getResultados");
        getResultados.onclick = handleGETResultadosButton;

        function handleGETResultadosButton() {
          const fechaInicio = fechaInicioElegida.selectedDates[0];
          const fechaFin = fechaFinElegida.selectedDates[0];

          var idActividad = document.getElementById("idjuego").value;
          var nivelActividad = document.getElementById("nivel").value;
          var idUsuario = document.getElementById("usuario").value;

          if (idActividad === "" || nivelActividad === "" || idUsuario
=== "") {
            document.getElementById("mensajeEstado").innerHTML = 'Error
al solicitar los resultados';
            return;
          }

          else {

            url +=
`&idActividad=${idActividad}&nivelActividad=${nivelActividad}&idUsuario=$
{idUsuario}`;

          if (fechaInicio && fechaFin) {
            url += `&timestampInicio=${moment(fechaInicio).format("YYYY-
MM-DD HH:mm:ss")}&timestampFinal=${moment(fechaFin).format("YYYY-MM-DD
HH:mm:ss")}`;
          } else if (fechaInicio) {
            url += `&timestampInicio=${moment(fechaInicio).format("YYYY-
MM-DD HH:mm:ss")}`;
          } else if (fechaFin) {
            url += `&timestampFinal=${moment(fechaFin).format("YYYY-MM-DD
HH:mm:ss")}`;
          }
```

Figura 3.71.- Código del script para solicitar datos de progreso del usuario (Parte 1). Fuente: Diseño propio

```
// Realiza la solicitud GET utilizando la URL generada
          fetch(url, {
            method: 'GET',
            port: 443,
            headers: {
              'Content-Type': 'application/json',
              //'Content-Length':
Buffer.byteLength(JSON.stringify(body)),
              'Authorization': 'Basic ' + btoa('lfonteriz:aaa1674unTFG'),
          }})
            .then(res => res.json())
            .then(json => {

              var sumaAciertos = 0;
              var sumaErrores = 0;

              for (var i in json) {
                if (json.hasOwnProperty(i)) {
                  sumaAciertos += json[i].numAciertos;
                  sumaErrores += json[i].numErrores;
                }
              }

              var duraciones = [];
              // Convertir el objeto json en un array de valores
              var resultados = Object.values(json);
              // Iterar sobre cada resultado y realizar las operaciones
deseadas
              resultados.forEach(resultado => {
                duraciones.push(resultado.duracion);
              });

              var tiempoRespuesta = duraciones.length > 0 ?
duraciones.reduce((a, b) => a + b) / duraciones.length : 0;

              document.getElementById("idUsuarioRespuesta").innerHTML =
idUsuario;
              document.getElementById("idJuegoRespuesta").innerHTML =
idActividad;
              document.getElementById("nivelRespuesta").innerHTML =
nivelActividad;
              document.getElementById("numAciertosRespuesta").innerHTML =
sumaAciertos;
              document.getElementById("numErroresRespuesta").innerHTML =
sumaErrores;
              document.getElementById("tiempoRespuesta").innerHTML =
tiempoRespuesta;
            })
            .catch(err => {
              console.error(err);
            });
          }
        }
</script>
```

Figura 3.72.- Código del script para solicitar datos de progreso del usuario (Parte 2). Fuente: Diseño propio

3.4.- PRUEBAS

Una vez desarrollado el producto software, es necesario realizar ciertas pruebas para evaluar y verificar su correcto funcionamiento. En este apartado, se analizará el funcionamiento cada caso de uso planteado, tanto de la skill como de la aplicación web. Para cada uno de estos casos se crea una tabla con los siguientes campos:

- Prueba: Campo que pone nombre a la prueba realizada.

- Identificador: Campo en el que se asigna una especie de "clave" que identifica a la prueba realizada.

- Descripción: Campo en el que se explica qué se pretende verificar con la prueba realizada.

- Resultado esperado: Campo en el que se indica cuál es la respuesta que se espera que dé el software.

- Resultado obtenido: Campo en el que se indica cuál es, finalmente, la respuesta dada por el software.

- Validación: Campo en el que se indica si efectivamente la respuesta obtenida es la esperada.

- Correspondencia: Relación de la prueba realizada con el caso de uso correspondiente.

En los apartados siguientes, se detallan las pruebas efectuadas tanto para la skill HablaConmigo como para la aplicación web.

3.4.1.- Casos de prueba de la skill HablaConmigo

Identificador	PS-1
Prueba	Registrar un nuevo usuario con los datos necesarios
Descripción	Un usuario intenta registrarse en la skill proporcionando todos los datos necesarios
Resultado esperado	La skill debe registrar al usuario en la base de datos e indicarle que ha sido registrado correctamente, que le proporcione el mismo id para empezar a jugar.
Resultado obtenido	La skill contesta "Usuario *nombre* creado con éxito. Repite tu id para empezar a jugar".
Validación	Correcta
Correspondencia	CUS-1 → Registro de usuario

Tabla 3.18.- Prueba de uso de la skill 1 - Registrar un nuevo usuario con los datos necesarios

Identificador	PS-2
Prueba	Registrar un nuevo usuario con un id ya registrado
Descripción	Un usuario intenta registrarse en la skill utilizando un id que ya está registrado en la base de datos.
Resultado esperado	La skill debe indicarle al usuario que no es posible registrarle porque ya existe un usuario registrado con ese id, y esperar a que el usuario le dé un id válido para poder registrarlo.
Resultado obtenido	La skill contesta "Ya hay un usuario registrado con ese id", y se queda esperando a que el usuario proporcione un id válido.
Validación	Correcta
Correspondencia	CUS-1 → Registro de un nuevo usuario

Tabla 3.19.- Prueba de uso de la skill 2 - Registrar un nuevo usuario con un id ya registrado

Identificador	PS-3
Prueba	Iniciar sesión con un id ya registrado
Descripción	Un usuario intenta acceder a la skill proporcionando un id con el que se registró anteriormente.
Resultado esperado	La skill debe darle la bienvenida al usuario, decirle que hay 6 juegos disponibles en la skill y que elija uno para empezar a jugar.
Resultado obtenido	La skill contesta "¡Hola *nombre*! Hay seis juegos disponibles. ¿A cuál te gustaría jugar?", y se queda esperando a que el usuario elija un juego.
Validación	Correcta

Correspondencia	CUS-2 → Inicio de sesión

Tabla 3.20.- Prueba de uso de la skill 3 - Iniciar sesión con un id ya registrado

Identificador	PS-4
Prueba	Iniciar sesión con un id no registrado
Descripción	Un usuario intenta acceder a la skill proporcionando un id que no se registró previamente
Resultado esperado	La skill debe decirle al usuario que no hay ningún registro con el id proporcionado, y esperar a que este responda con un id válido
Resultado obtenido	La skill contesta "No se ha encontrado un usuario con ese ID", y se queda esperando a que el usuario proporcione un id válido
Validación	Correcta
Correspondencia	CUS-2 → Inicio de sesión

Tabla 3.21.- Prueba de uso de la skill 4 - Iniciar sesión con un id no registrado

Identificador	PS-5
Prueba	Elección de juego correcta
Descripción	Un usuario elige un juego de los disponibles utilizando una frase interpretable por Alexa.
Resultado esperado	La skill debe decirle al usuario el juego que ha escogido, el número de niveles que tiene y quedar esperando a que elija uno de los niveles para empezar a jugar.
Resultado obtenido	La skill responde "Has elegido jugar a *juego*. Este juego tiene *numniveles* niveles disponibles. ¿A cuál de ellos te gustaría jugar?", y se queda esperando a que el usuario elija un nivel.
Validación	Correcta
Correspondencia	CUS-3 → Elección de juego

Tabla 3.22.- Prueba de uso de la skill 5 - Elección de juego correcta

Identificador	PS-6
Prueba	Elección de un juego distinto a los de la lista proporcionada
Descripción	Un usuario elige un juego que no se encuentra en la lista de la skill y que, por tanto, la skill no reconoce.
Resultado esperado	La skill debe decirle al usuario el juego que elija un juego disponible y quedarse esperando a que dé una respuesta válida para empezar a jugar.
Resultado obtenido	La skill responde "Elige un juego disponible", y se queda esperando a que el usuario elija un juego disponible
Validación	Correcta

Correspondencia	CUS-3 → Elección de juego

Tabla 3.23.- Prueba de uso de la skill 6 - Elección de un juego distinto a los de la lista proporcionada

Identificador	PS-7
Prueba	Elección correcta de nivel de "Ordena las Sílabas"
Descripción	Un usuario elige uno de los cuatro niveles disponibles para "Ordena las Sílabas".
Resultado esperado	La skill responderá al usuario con un enunciado del juego y nivel elegido y se queda esperando a que este responda a la pregunta.
Resultado obtenido	La skill responde con "¿Qué palabra se esconde en *palabra*?", muestra por pantalla la palabra para ordenar y, en algunos niveles, la pista de texto correspondiente. Después, se queda esperando a que el usuario responda para comprobar si su respuesta es correcta o no.
Validación	Correcta
Correspondencia	CUS-4 → Juego "Ordena las Sílabas"

Tabla 3.24.- Prueba de uso de la skill 7 - Elección correcta de nivel de "Ordena las Sílabas"

Identificador	PS-8
Prueba	Elección incorrecta de nivel de "Ordena las Sílabas"
Descripción	Un usuario elige un nivel distinto de los cuatro disponibles para "Ordena las Sílabas".
Resultado esperado	La skill deberá indicarle al usuario que elija un nivel disponible, y quedarse esperando a que lo haga.
Resultado obtenido	La skill responde "Elige un nivel disponible para Ordena las Sílabas", muestra por pantalla los niveles disponibles y espera a que el usuario elija uno válido.
Validación	Correcta
Correspondencia	CUS-4 → Juego "Ordena las Sílabas"

Tabla 3.25.- Prueba de uso de la skill 8 - Elección incorrecta de nivel de "Ordena las Sílabas"

Identificador	PS-9
Prueba	Elección correcta de nivel de "Cuenta las Sílabas"
Descripción	Un usuario elige uno de los dos niveles disponibles para "Cuenta las Sílabas".
Resultado esperado	La skill responderá al usuario con un enunciado del juego y nivel elegido y se queda esperando a que este responda a la pregunta.

Resultado obtenido	La skill responde con "¿Cuántas sílabas tiene *palabra*?", muestra por pantalla la palabra para contar sus sílabas y se queda esperando a que el usuario responda para comprobar si su respuesta es correcta o no.
Validación	Correcta
Correspondencia	CUS-5 → Juego "Cuenta las Sílabas"

Tabla 3.26.- Prueba de uso de la skill 9 - Elección correcta de nivel de "Cuenta las Sílabas"

Identificador	PS-10
Prueba	Elección incorrecta de nivel de "Cuenta las Sílabas"
Descripción	Un usuario elige un nivel distinto de los dos disponibles para "Cuenta las Sílabas".
Resultado esperado	La skill deberá indicarle al usuario que elija un nivel disponible, y quedarse esperando a que lo haga.
Resultado obtenido	La skill responde "Elige un nivel disponible para Cuenta las Sílabas", muestra por pantalla los niveles disponibles y espera a que el usuario elija uno válido.
Validación	Correcta
Correspondencia	CUS-5 → Juego "Cuenta las Sílabas"

Tabla 3.27.- Prueba de uso de la skill 10 - Elección incorrecta de nivel de "Cuenta las Sílabas"

Identificador	PS-11
Prueba	Elección correcta de nivel de "Las Adivinanzas"
Descripción	Un usuario elige uno de los tres niveles disponibles para "Las Adivinanzas".
Resultado esperado	La skill responderá al usuario con un enunciado del juego y nivel elegido y se queda esperando a que este responda a la pregunta.
Resultado obtenido	La skill responde con "*adivinanza*", muestra por pantalla la adivinanza y, en los niveles que corresponde, la pista de texto o imagen. Después, se queda esperando a que el usuario responda para comprobar si su respuesta es correcta o no.
Validación	Correcta
Correspondencia	CUS-6 → Juego "Las Adivinanzas"

Tabla 3.28.- Prueba de uso de la skill 11 - Elección correcta de nivel de "Las Adivinanzas"

Identificador	PS-12
Prueba	Elección incorrecta de nivel de "Las Adivinanzas"

Descripción	Un usuario elige un nivel distinto de los tres disponibles para “Las Adivinanzas”.
Resultado esperado	La skill deberá indicarle al usuario que elija un nivel disponible, y quedarse esperando a que lo haga.
Resultado obtenido	La skill responde “Elige un nivel disponible para Las Adivinanzas”, muestra por pantalla los niveles disponibles y espera a que el usuario elija uno válido.
Validación	Correcta
Correspondencia	CUS-6 → Juego “Las Adivinanzas”

Tabla 3.29.- Prueba de uso de la skill 12 - Elección incorrecta de nivel de "Las Adivinanzas"

Identificador	PS-13
Prueba	Elección correcta de nivel de “La Sílaba Oculta”
Descripción	Un usuario elige el único nivel que tiene “La Sílaba Oculta”.
Resultado esperado	La skill responderá al usuario con un enunciado del juego y nivel elegido y se queda esperando a que este responda a la pregunta.
Resultado obtenido	La skill responde con “¿Qué palabra se esconde en *palabra*?”, muestra por pantalla la palabra y una pista de imagen. Después, se queda esperando a que el usuario responda para comprobar si su respuesta es correcta o no.
Validación	Correcta
Correspondencia	CUS-7 → Juego “La Sílaba Oculta”

Tabla 3.30.- Prueba de uso de la skill 13 - Juego "La Sílaba Oculta"

Identificador	PS-14
Prueba	Elección incorrecta de nivel de “La Sílaba Oculta”
Descripción	Un usuario elige un nivel distinto del “1”.
Resultado esperado	La skill deberá indicarle al usuario que elija el único nivel disponible, y quedarse esperando a que lo haga.
Resultado obtenido	La skill responde “Elige un nivel disponible para La Sílaba Oculta”, muestra por pantalla “Nivel 1” y espera a que el usuario lo elija.
Validación	Correcta
Correspondencia	CUS-7 → Juego “La Sílaba Oculta”

Tabla 3.31.- Prueba de uso de la skill 14 - Elección incorrecta de nivel de "La Sílaba Oculta"

Identificador	PS-15
Prueba	Elección correcta de nivel de “Los Antónimos”
Descripción	Un usuario elige uno de los dos niveles disponibles.

Resultado esperado	La skill responderá al usuario con un enunciado del juego y nivel elegido y se queda esperando a que este responda a la pregunta.
Resultado obtenido	La skill responde con "¿Cuál es el antónimo de *palabra*?", muestra por pantalla la palabra, y, en el nivel necesario, también muestra una pista de imagen. Después, se queda esperando a que el usuario responda para comprobar si su respuesta es correcta o no.
Validación	Correcta
Correspondencia	CUS-8 → Juego "Los Antónimos"

Tabla 3.32.- Prueba de uso de la skill 15 - Elección correcta de nivel de "Los Antónimos"

Identificador	PS-16
Prueba	Elección incorrecta de nivel de "Los Antónimos"
Descripción	Un usuario elige un nivel distinto de los dos disponibles.
Resultado esperado	La skill deberá indicarle al usuario que elija uno de dos los niveles disponibles, y quedarse esperando a que lo haga.
Resultado obtenido	La skill responde "Elige un nivel disponible para Los Antónimos", muestra por pantalla los niveles disponibles y espera a que el usuario elija uno.
Validación	Correcta
Correspondencia	CUS-8 → Juego "Los Antónimos"

Tabla 3.33.- Prueba de uso de la skill 16 - Elección incorrecta de nivel de "Los Antónimos"

Identificador	PS-17
Prueba	Elección correcta de nivel de "Repite la Palabra"
Descripción	Un usuario elige uno de los cuatro niveles disponibles.
Resultado esperado	La skill responderá al usuario con un enunciado del juego y nivel elegido y se queda esperando a que este responda a la pregunta.
Resultado obtenido	La skill responde con "`Repite la palabra: *palabra*" y muestra por pantalla la palabra. Después, se queda esperando a que el usuario repita la palabra para comprobar si su respuesta es correcta o no.
Validación	Correcta
Correspondencia	CUS-9 → Juego "Repite la Palabra"

Tabla 3.34.- Prueba de uso de la skill 17 - Elección correcta de nivel de "Repite la Palabra"

Identificador	PS-18
Prueba	Elección incorrecta de nivel de "Los Antónimos"
Descripción	Un usuario elige un nivel distinto de los cuatro disponibles.

Resultado esperado	La skill deberá indicarle al usuario que elija uno de los cuatro niveles disponibles, y quedarse esperando a que lo haga.
Resultado obtenido	La skill responde "Elige un nivel disponible para Repite la Palabra", muestra por pantalla los niveles disponibles y espera a que el usuario elija uno.
Validación	Correcta
Correspondencia	CUS-9 → Juego "Repite la Palabra"

Tabla 3.35.- Prueba de uso de la skill 18 - Elección incorrecta de nivel de "Los Antónimos"

Identificador	PS-19
Prueba	Cambio de nivel correcto
Descripción	Un usuario elige otro nivel de los disponibles para el juego seleccionado.
Resultado esperado	La skill responderá al usuario con un enunciado del juego y nivel elegido y se queda esperando a que este responda a la pregunta.
Resultado obtenido	La skill responde con el correspondiente enunciado según el nivel y juego elegido, y espera a la respuesta del usuario.
Validación	Correcta
Correspondencia	CUS-10 → Cambio de nivel

Tabla 3.36.- Prueba de uso de la skill 19 - Cambio de nivel correcto

Identificador	PS-20
Prueba	Cambio de nivel incorrecto
Descripción	Un usuario elige otro nivel de los disponibles para el juego seleccionado pero ese nivel no se encuentra en la lista de los disponibles para el juego elegido.
Resultado esperado	La skill deberá indicarle al usuario que elija un nivel disponible, y quedarse esperando a que lo haga.
Resultado obtenido	La skill responde "Elige un nivel disponible para *juego*", muestra por pantalla los niveles disponibles de ese juego y espera a que el usuario elija un nivel válido.
Validación	Correcta
Correspondencia	CUS-10 → Cambio de nivel

Tabla 3.37.- Prueba de uso de la skill 20 - Cambio de nivel incorrecto

Identificador	PS-21
Prueba	Cambio de juego correcto
Descripción	Un usuario elige otro juego de los disponibles en la lista.

Resultado esperado	La skill debe decirle al usuario el juego que ha escogido, el número de niveles que tiene y quedar esperando a que elija uno de los niveles para empezar a jugar.
Resultado obtenido	La skill responde "Has elegido jugar a *juego*. Este juego tiene *numniveles* niveles disponibles. ¿A cuál de ellos te gustaría jugar?", y se queda esperando a que el usuario elija un nivel.
Validación	Correcta
Correspondencia	CUS-11 → Cambio de juego

Tabla 3.38.- Prueba de uso de la skill 21 - Cambio de juego correcto

Identificador	PS-22
Prueba	Cambio de juego incorrecto
Descripción	Un usuario elige un juego que no se encuentra en la lista de la skill y que, por tanto, la skill no reconoce.
Resultado esperado	La skill debe decirle al usuario el juego que elija un juego disponible y quedarse esperando a que dé una respuesta válida para empezar a jugar.
Resultado obtenido	La skill responde "Elige un juego disponible", y se queda esperando a que el usuario elija un juego disponible.
Validación	Correcta
Correspondencia	CUS-11 → Cambio de juego

Tabla 3.39.- Prueba de uso de la skill 22 - Cambio de juego incorrecto

3.4.2.- Casos de prueba de la aplicación web

Identificador	PAP-1
Prueba	Nueva entrada de juego con todos los campos necesarios (al menos deben rellenarse los campos "ID del juego", "Palabra, adivinanza", "Solución" y "Nivel"). Si fuera necesario, también se rellenarían los campos "Pista de texto" o "Pista de foto" (en este último caso se añadiría la URL en la que se encuentra la imagen deseada).
Descripción	El logopeda agrega una nueva entrada para un juego con todos los campos necesarios (al menos, "ID del juego", "Palabra, adivinanza", "Solución" y "Nivel") rellenados de forma correcta.
Resultado esperado	La aplicación web debe subir la entrada proporcionada por el logopeda a la base de datos y proporcionarle una retroalimentación positiva.

Resultado obtenido	La aplicación sube correctamente la nueva entrada de juego y muestra el mensaje "Juego subido correctamente".
Validación	Correcta
Correspondencia	CUAP-1 → Nueva entrada de juego

Tabla 3.40.- Prueba de uso de la aplicación web 1 - Nueva entrada de juego con todos los campos necesarios

Identificador	PAP-2
Prueba	Nueva entrada de juego con algún campo obligatorio ("ID del juego", "Palabra, adivinanza", "Solución" o "Nivel") no rellenado.
Descripción	El logopeda agrega una nueva entrada para un juego dejando al menos uno de los campos requeridos ("ID del juego", "Palabra, adivinanza", "Solución" o "Nivel") vacío.
Resultado esperado	La aplicación web no debe subir la entrada, y debe mostrar un mensaje de retroalimentación negativa al logopeda.
Resultado obtenido	La aplicación muestra el mensaje "Error al subir el juego".
Validación	Correcta
Correspondencia	CUAP-1 → Nueva entrada de juego

Tabla 3.41.- Prueba de uso de la aplicación web 2 - Nueva entrada de juego con algún campo obligatorio no rellenado

Identificador	PAP-3
Prueba	Consulta de progreso formulada correctamente
Descripción	El logopeda realiza una petición para consultar el progreso de un usuario con todos los campos del formulario necesarios rellenados correctamente (al menos, deben rellenarse los campos "ID del usuario", "ID del juego" y "Nivel"). En caso de querer filtrar más los resultados, pueden indicarse una "Fecha y hora de inicio" (se mostrarían los resultados obtenidos a partir de esa fecha), "Fecha y hora de fin" (se mostrarían los resultados obtenidos hasta esa fecha) o ambas fechas (se mostrarían los resultados obtenidos en el periodo de tiempo acotado por las fechas elegidas).
Resultado esperado	La aplicación web debe realizar la petición, los cálculos internos necesarios y mostrar al logopeda el resultado de su solicitud en el apartado habilitado para ello.
Resultado obtenido	La aplicación muestra los resultados de la solicitud en el apartado habilitado para ello.
Validación	Correcta

Correspondencia	CUAP-2 → Ver progreso

Tabla 3.42.- Prueba de uso de la aplicación web 3 - Consulta de progreso formulada correctamente

Identificador	PAP-4
Prueba	Consulta de progreso mal formulada
Descripción	El logopeda realiza una petición para consultar el progreso de un usuario con algún campo requerido no rellenado (“ID del usuario”, “ID del juego” o “Nivel” vacío).
Resultado esperado	La aplicación web mostrará un mensaje de error y esperará a que el usuario rellene los campos requeridos correctamente.
Resultado obtenido	La aplicación muestra el mensaje “Error al solicitar los resultados” y espera que el usuario complete los campos restantes correctamente.
Validación	Correcta
Correspondencia	CUAP-2 → Ver progreso

Tabla 3.43.- Prueba de uso de la aplicación web - Consulta de progreso mal formulada

3.5.- MANUAL DE USUARIO

A continuación, se ofrece una guía de uso tanto de la skill HablaConmigo como de la aplicación web de ayuda para el logopeda.

3.5.1.- Uso de la skill HablaConmigo

El primer paso para usar la skill HablaConmigo sería instalar la skill en el dispositivo (explicación en Apartado 3.6). Una vez hecho esto, ya se puede empezar a jugar.

Para abrir la skill es necesario decir su nombre de invocación. Así, diciendo “Alexa, abre Habla Conmigo”, se inicializará la skill y se mostrará la pantalla de la Figura 3.73, donde Alexa pregunta al usuario si es la primera vez que juega.

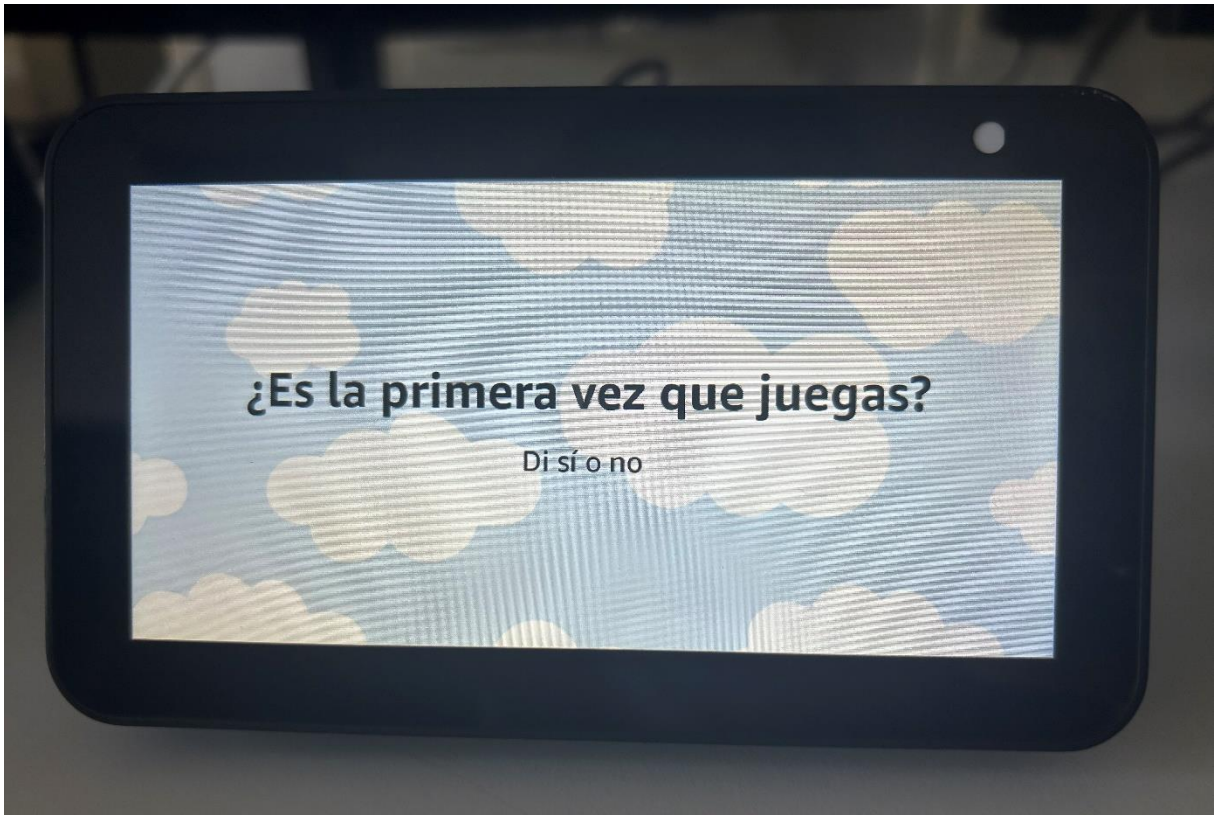

Figura 3.73.- Pantalla de bienvenida de la skill. Fuente: Diseño propio

Si el usuario responde “sí”, Alexa le llevará a la pantalla de registro (Figura 3.74). En esta pantalla se muestra una frase a modo de guía para que el usuario sepa cómo responderle a Alexa para que ella interprete correctamente sus datos. El usuario debe proporcionarle un número de identificación elegido por el mismo, su nombre y su edad. Las frases que Alexa interpreta para registrar a un nuevo usuario son las siguientes:

- “Mi id es el {id} soy {nombre} y tengo {edad} años”
- “Mi id es el número {id} me llamo {nombre} y tengo {edad} años”
- “Mi identificador es el número {id} me llamo {nombre} y tengo {edad} años”
- “Mi id es el número {id} soy {nombre} y tengo {edad} años”
- “Mi id es el {id} soy {nombre} y tengo {edad} años”

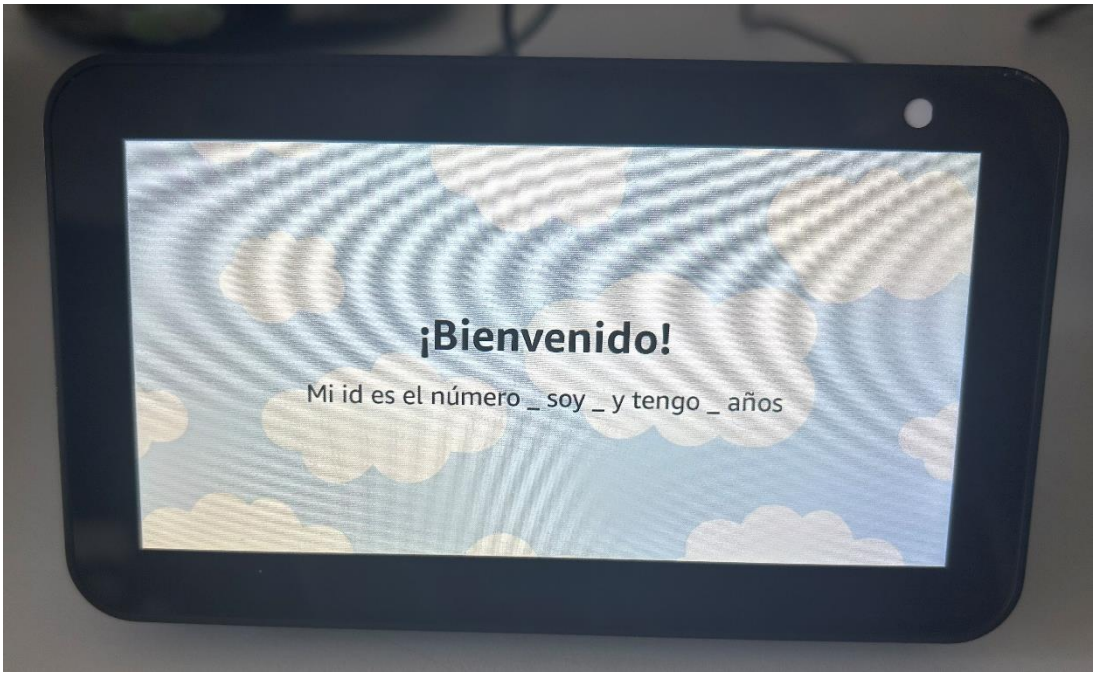

Figura 3.74.- Pantalla de registro. Fuente: Diseño propio

Si el usuario formula correctamente la frase, Alexa le registrará en la base de datos y le pedirá que le repita el id para empezar a jugar (Figura 3.75). Las frases con las que Alexa puede identificar el id de un usuario son:

- "El {idUsuario}"
- "Es el {idUsuario}"
- "El número {idUsuario}"
- "Mi id es el número {idUsuario}"
- "Mi id es {idUsuario}"

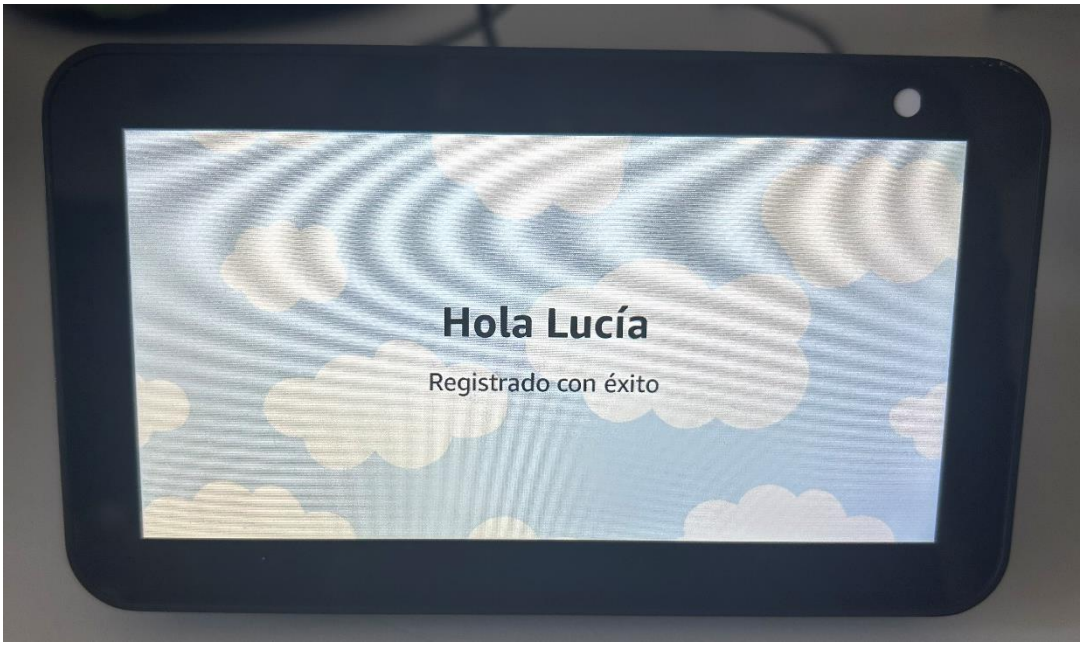

Figura 3.75.- Pantalla de usuario registrado con éxito. Fuente: Diseño propio

Si, por el contrario, en la pantalla de bienvenida (Figura 3.73) el usuario responde "no", Alexa le llevará a la página de inicio de sesión (Figura 3.76). Aquí Alexa pedirá al usuario que se identifique para iniciar el juego y seguir guardando sus resultados. Las frases que Alexa interpreta para el inicio de sesión son las mismas que en el caso de repetir el id para empezar a jugar.

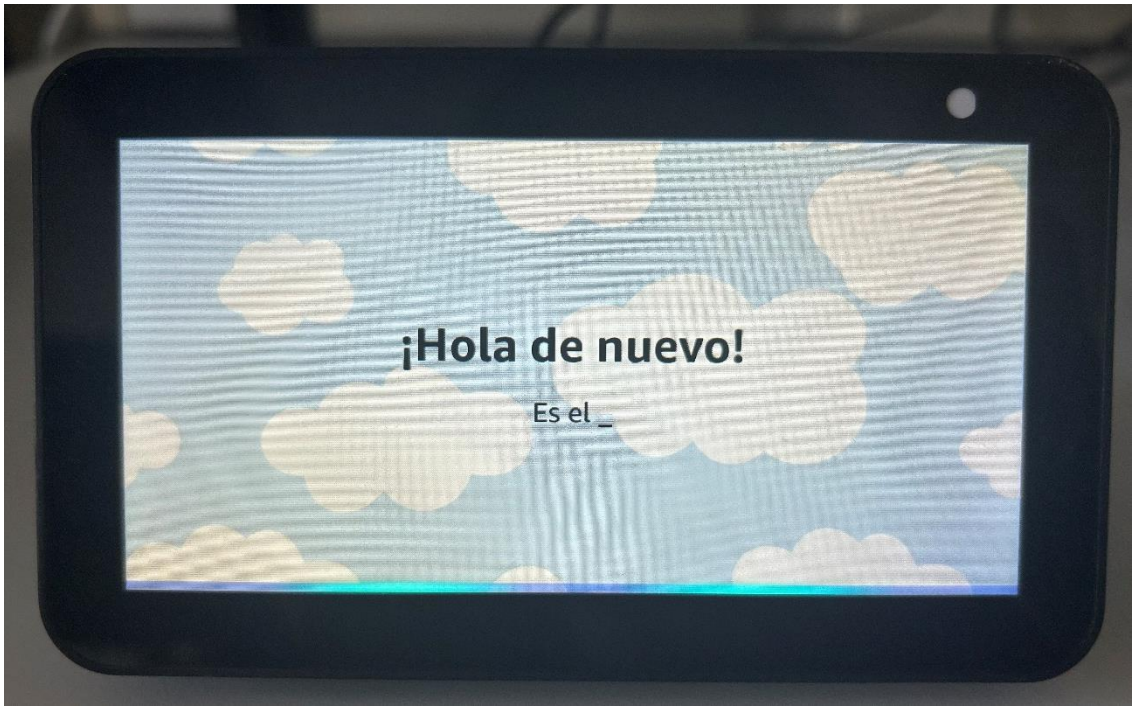

Figura 3.76.- Pantalla de inicio de sesión. Fuente: Diseño propio

Una vez identificado el usuario, la skill pide al usuario que elija uno de los 6 juegos disponibles, mostrando sus nombres por pantalla (Figura 3.77). Las frases que Alexa interpreta para la elección de juego son:

- "A {juegoElegido}"
- "Quiero jugar a {juegoElegido}"
- "Juego de {juegoElegido}"
- "Quiero jugar al juego de {juegoElegido}"

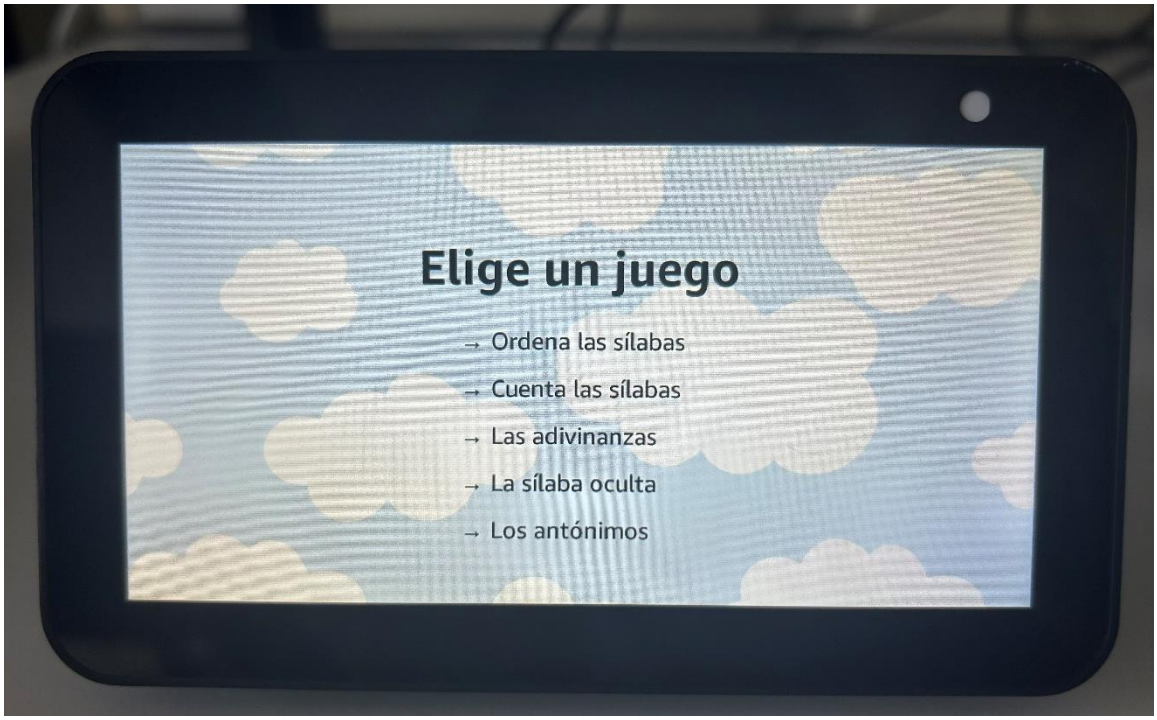

Figura 3.77.- Pantalla de selección de juego. Fuente: Diseño propio

Cada usuario tiene varios niveles de juego disponibles. Al elegir un juego, Alexa le pregunta al usuario a cuál de ellos le gustaría jugar (Figura 3.78). Las frases que Alexa interpreta para la elección de juego son:

- Al nivel {numNivel}
- Quiero jugar al nivel {numNivel}
- Nivel {numNivel}

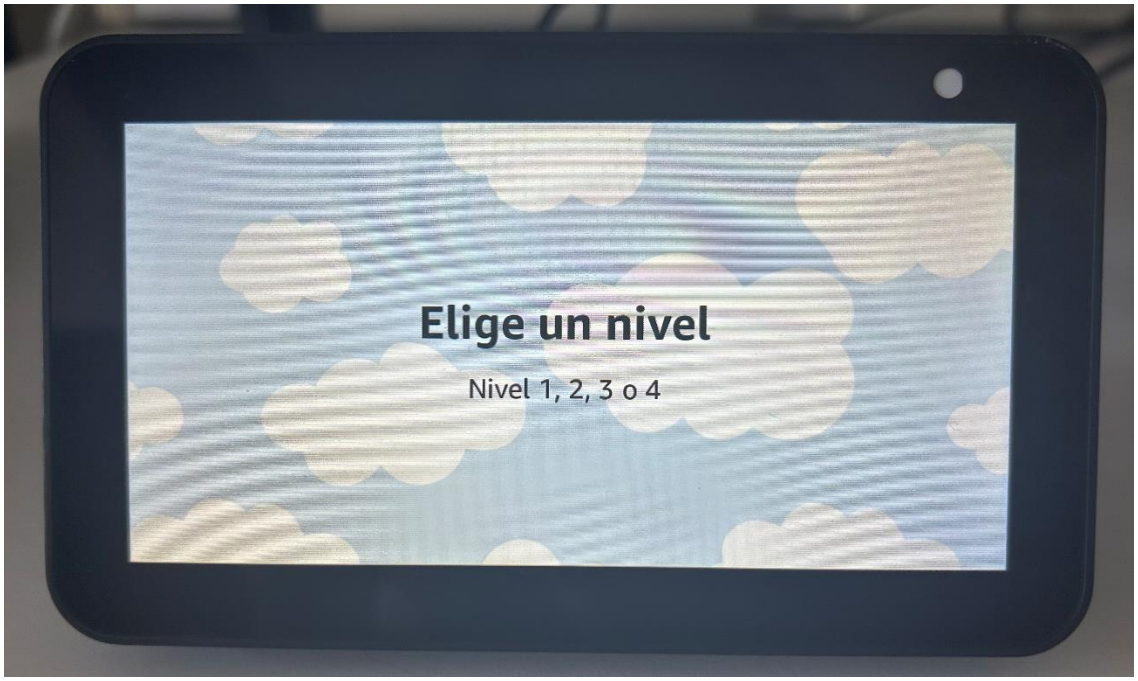

Figura 3.78.- Pantalla de selección de nivel. Fuente: Diseño propio

En el momento que el usuario elige un nivel, Alexa muestra el "enunciado" correspondiente, pista de foto o texto en el caso de necesitarla y una frase como guía de respuesta para el usuario (un ejemplo se muestra en la Figura 3.79).

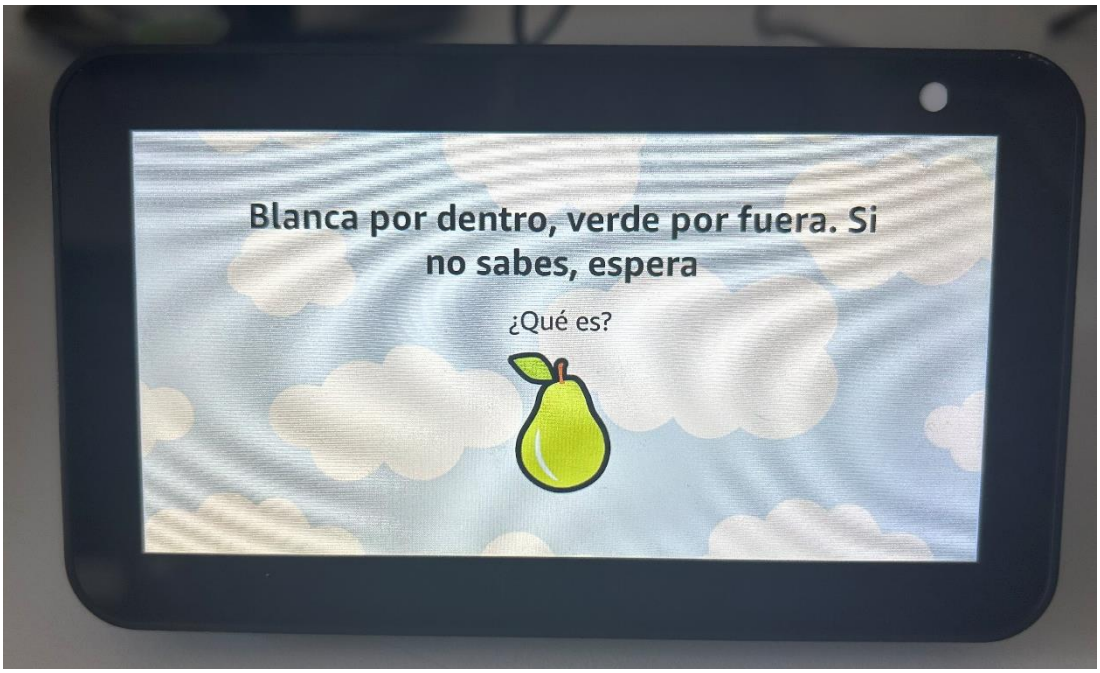

Figura 3.79.- Pantalla de muestra de enunciado. Fuente: Diseño propio

Hay distintas formas de responder en función del juego elegido. “Ordena las Sílabas”, “Las Adivinanzas”, “La Sílaba Oculta” y “Repite la Palabra” admiten las siguientes respuestas:

- “Se esconde {respuestaPalabra}”
- “La palabra oculta es {respuestaPalabra}”
- “La palabra escondida es {respuestaPalabra}”
- “La palabra es {respuestaPalabra}”
- “Es {respuestaPalabra}”

“Cuenta las Sílabas” admite las siguientes respuestas:

- “Tiene {numSilabas} sílabas”
- “Tiene {numSilabas}”
- “{numSilabas} sílabas”
- “La palabra tiene {numSilabas} sílabas”

Por último, “Los Antónimos” admite las siguientes respuestas:

- “Su antónimo es {antonimo}”
- “Su opuesto es {antonimo}”
- “Lo contrario es {antonimo}”
- “La palabra opuesta es {antonimo}”
- “El opuesto es {antonimo}”

Cuando Alexa recibe la respuesta del usuario, comprueba si es correcta (Figura 3.80) o no (Figura 3.81), proporciona retroalimentación al usuario para que este sepa si ha acertado o no y le indica que elija un nivel u otro juego para seguir jugando.

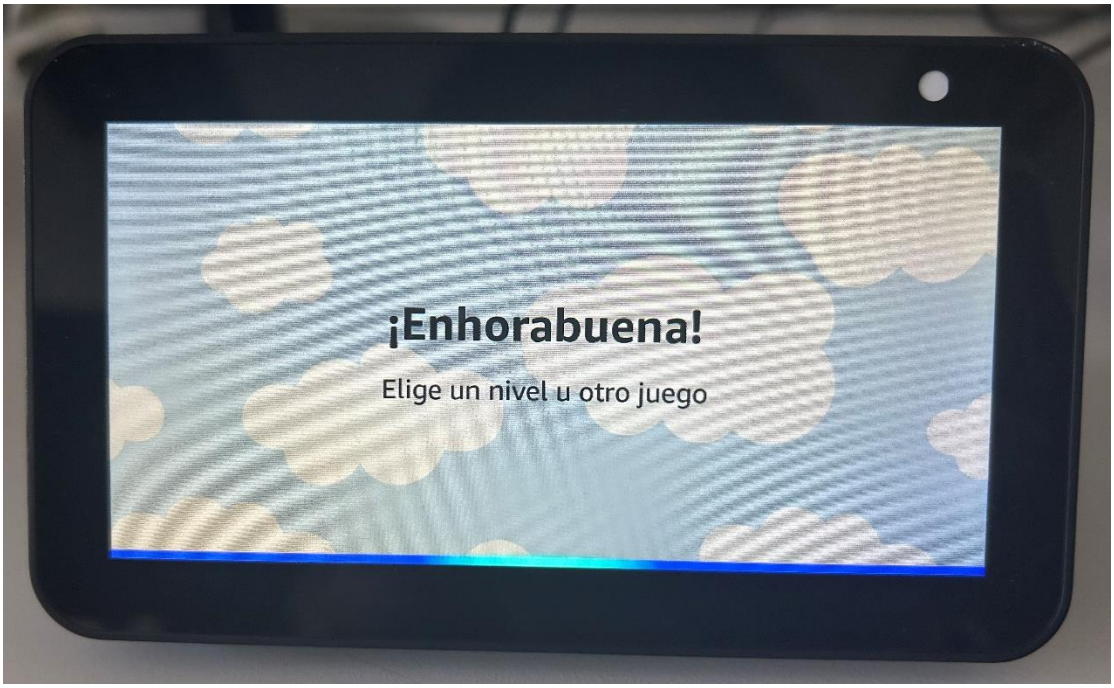

Figura 3.80.- Pantalla de acierto. Fuente: Diseño propio

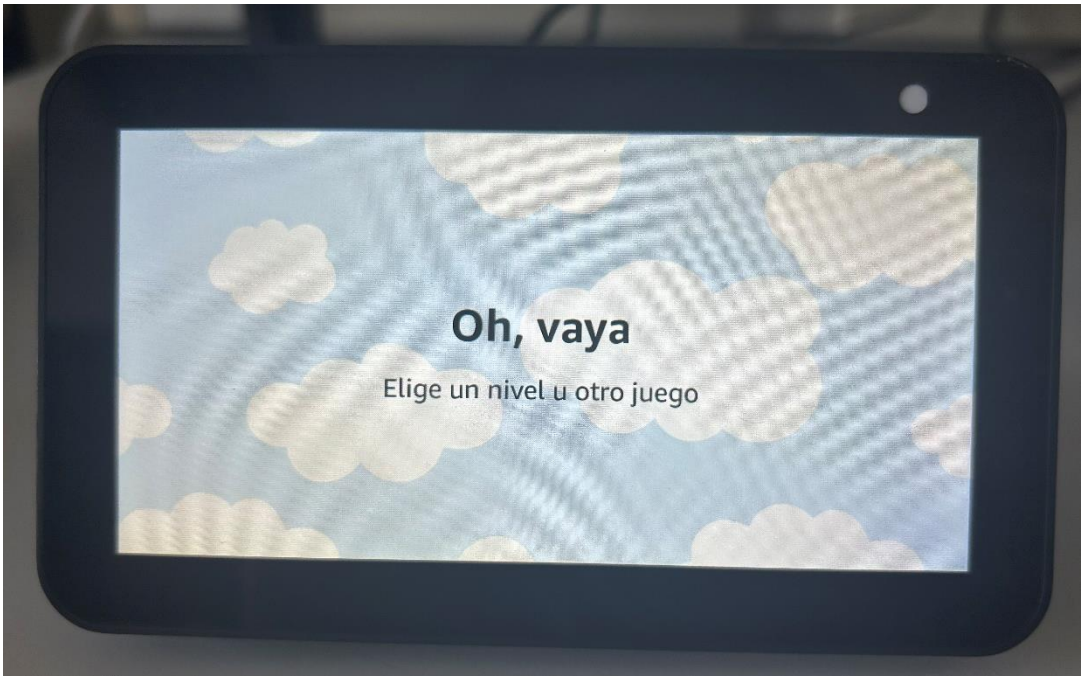

Figura 3.81.- Pantalla de error. Fuente: Diseño propio

Si el usuario quiere seguir jugando al mismo juego y al mismo nivel, simplemente tendría que decir una de las frases de selección de nivel anteriormente mencionadas con el

número de nivel al que estaba jugando y Alexa le mostrará otro "enunciado" de juego. Si quiere seguir jugando al mismo juego pero cambiar de nivel, sería igual pero con el nuevo número de nivel al que quiere jugar. Finalmente, si lo que quiere es cambiar de juego, tendría que decir el nombre del juego en una de las frases de invocación que se mencionaron en la pantalla de selección de juego.

Cuando el usuario desee dejar de jugar, solo debe decir "adiós" para despedirse de Alexa.

3.5.2.- Uso de la aplicación web

El primer paso para utilizar la aplicación web sería abrir el archivo "index.html", que se corresponde con la página de inicio de la web. Para ello solo hace falta hacer doble clic en el archivo.

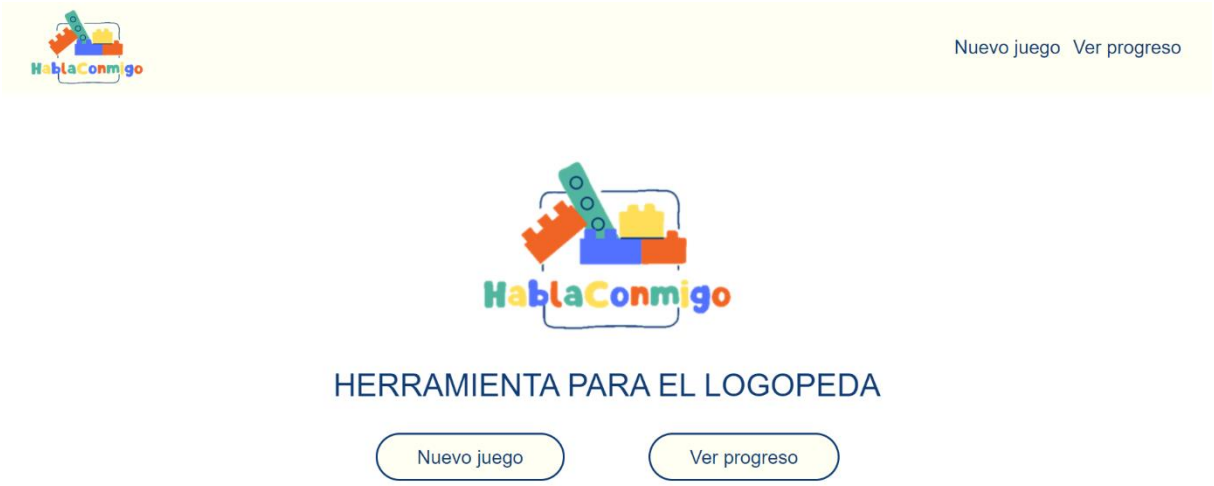

Figura 3.82.- Vista de página de inicio. Fuente: Diseño propio

En esta página se puede observar la barra de navegación y lo que sería la página principal. En la parte izquierda de la barra de navegación se encuentra el logo de la aplicación, que contiene un hipervínculo a la página principal (Figura 3.82), mientras que en

la parte derecha se encuentran dos botones, “Nuevo juego” y “Ver progreso”. El primero llevaría a la página que permite al logopeda añadir una nueva entrada a un juego (Figura 3.83), mientras que el segundo llevaría a la página que permite al logopeda consultar los resultados del usuario (Figuras 3.86 y 3.87). En la parte inferior se encuentra una imagen con el logo de la aplicación, acompañado de la descripción “Herramienta para el logopeda” y dos botones. Estos dos botones, al igual que los de la barra de navegación, permiten acceder a las páginas de subida de nueva entrada de juego y de consulta de progreso.

Haciendo clic en “Nuevo juego”, ya sea en el enlace de la barra de navegación o en el botón del contenedor central, se muestra la página que se puede observar en la Figura 3.83. Lo primero que se observa es la barra de navegación, en la que se muestra el enlace de la página actual en un color distinto, para que el usuario sepa en qué página se encuentra. El resto de la página se divide en dos partes: una tabla y un formulario. En la parte izquierda se encuentra la tabla, que se incluye a modo de referencia para que el logopeda sepa el id de cada juego y de cuantos niveles dispone este. En la parte derecha se encuentra un formulario para subir una nueva entrada.

Figura 3.83.- Vista de página de Nuevo juego. Fuente: Diseño propio

Este formulario consta de varios campos, siendo algunos requeridos (ID del juego, palabra, adivinanza, solución y nivel) y otros optativos (pista de texto y pista de imagen). En estos campos debe introducirse (de hacerlo) lo siguiente:

- ID del juego: Número que representa el juego al que quiere añadirse una nueva entrada.

- Palabra, adivinanza: Enunciado de la adivinanza, palabra de la que se deben ordenar o contar las sílabas... Es el campo en el que se introduce el "acertijo" para el niño.

- Solución: Campo en el que se introduce la solución al "acertijo" introducido en el campo anterior.

- Nivel: Número que identifica el nivel del juego al que se quiere añadir la nueva entrada.

- Pista de texto: Campo en el que se introduce, en los casos que sea necesario, una pista de texto para orientar al usuario de la skill en la respuesta.

- Pista de imagen (introducir URL): Campo en el que se introduce, en los casos que sea necesario, una pista de imagen para orientar al usuario de la skill en la respuesta. Con el planteamiento actual de la aplicación, no se subiría directamente la imagen a este formulario, sino que la imagen debe ser subida previamente al servidor y aquí se incluiría la URL en la que se encuentra esa imagen.

Una vez rellenados los campos ya se puede enviar el formulario. Si todo se ha procesado correctamente, aparecerá debajo del botón de enviar un mensaje avisando del éxito de la subida (Figura 3.84). Si en cambio, durante la subida del registro se produce algún error, se mostrará en el mismo sitio un mensaje de error (Figura 3.85).

Figura 3.84.- Muestra de mensaje de juego subido correctamente. Fuente: Diseño propio

Figura 3.85.- Muestra de mensaje de error en la subida del juego. Fuente: Diseño propio

Por último, haciendo clic en la barra de navegación en “Ver progreso” (o si se encuentra en la página de inicio también se puede acceder desde el botón del contenedor principal) se accede a la página que permite hacer consultas sobre el progreso de un niño (Figuras 3.86 y 3.87).

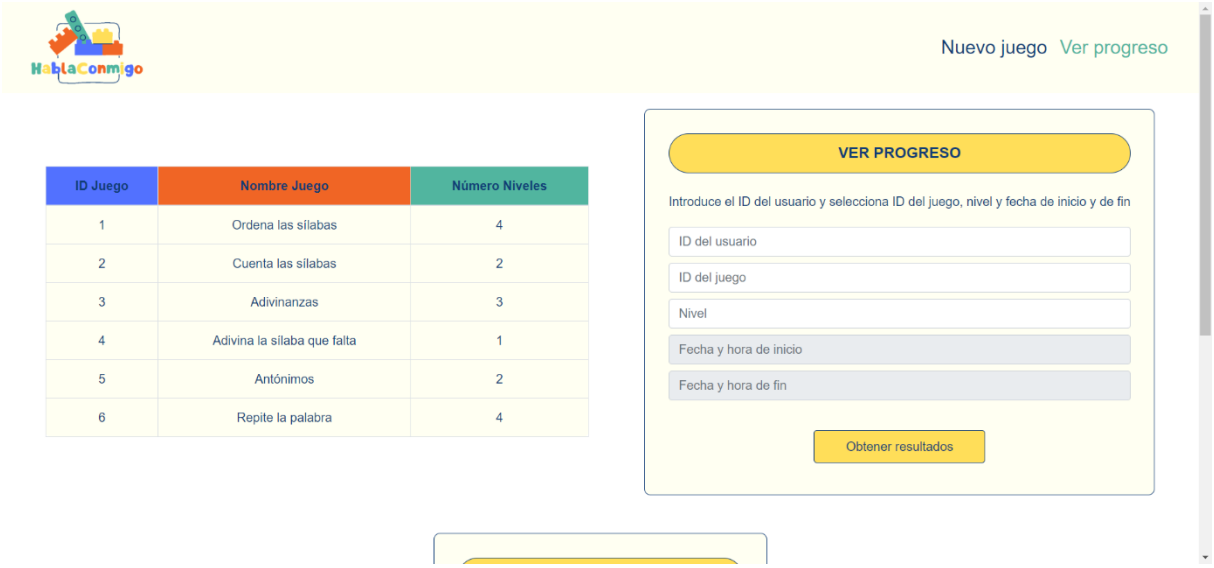

Figura 3.86.- Vista de Ver progreso (Parte 1). Fuente: Diseño propio

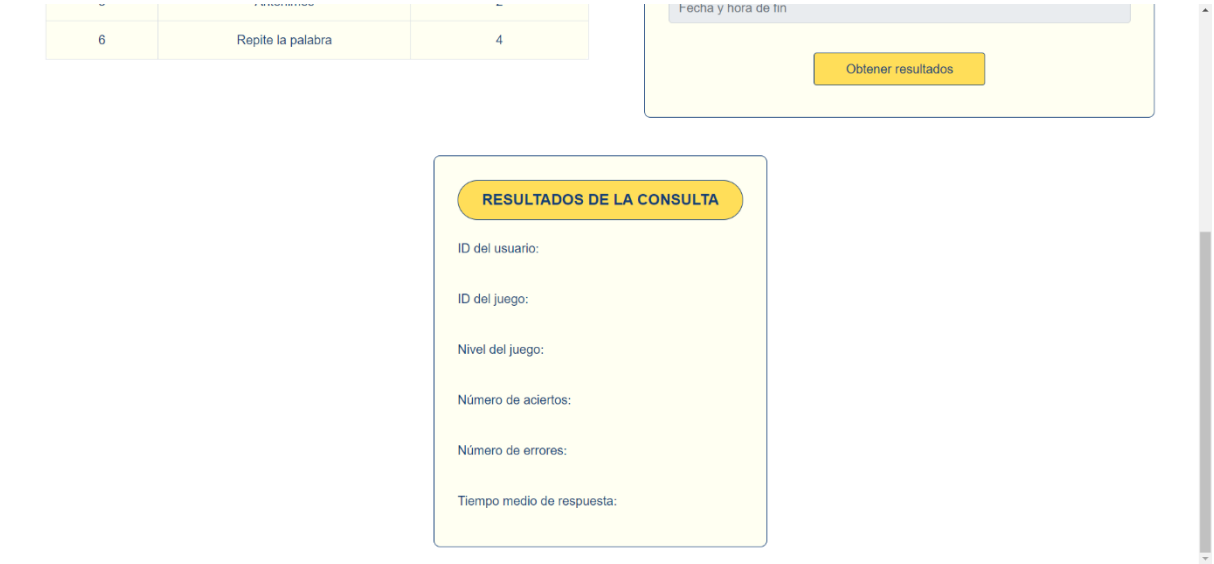

Figura 3.87.- Vista de Ver progreso (Parte 2). Fuente: Diseño propio

Al igual que en la página de "Nuevo juego", lo que se observa en esta página de primeras son una tabla (con la misma relación ID Juego-Nombre-Número de niveles) y un formulario. En esta ocasión el formulario consta de los siguientes campos:

- ID del usuario: Número que identifica al usuario del que se quiere consultar su progreso.

- ID del juego: Número que representa el juego del que se quiere consultar el progreso del usuario de la skill.

- Nivel: Número que representa el nivel exacto del que se quiere consultar el progreso.

- Fecha y hora de inicio: Campo de calendario que permite seleccionar una fecha inicial. No es obligatorio seleccionar una.

- Fecha y hora de fin: Campo de calendario que permite seleccionar una fecha final. No es obligatorio seleccionar una.

Figura 3.88.- Muestra de introducción de datos en el formulario. Fuente: Diseño propio

Los campos de ID del usuario, ID del juego y nivel deben rellenarse siempre. En cambio, los 2 campos de fechas pueden dejarse vacíos en función de lo que se quiera consultar:

- Los 2 campos vacíos: Se solicitarán todos los resultados obtenidos por ese usuario en ese nivel.

- Solo contiene fecha el campo “Fecha y hora de inicio”: Se solicitarán los resultados obtenidos a partir de la fecha seleccionada.

- Solo contiene fecha el campo “Fecha y hora de inicio”: Se solicitarán los resultados obtenidos hasta la fecha seleccionada.

- Tanto “Fecha y hora de inicio” como “Fecha y hora de fin” contienen fecha: Se solicitarán los resultados obtenidos en el periodo de tiempo delimitado por esas dos fechas.

En la Figura 3.89 se puede observar la parte de debajo de la página, con un contenedor con ciertas líneas de texto. Cuando se solicite un progreso, los datos obtenidos en la consulta aparecerán en este apartado (Figura 3.89).

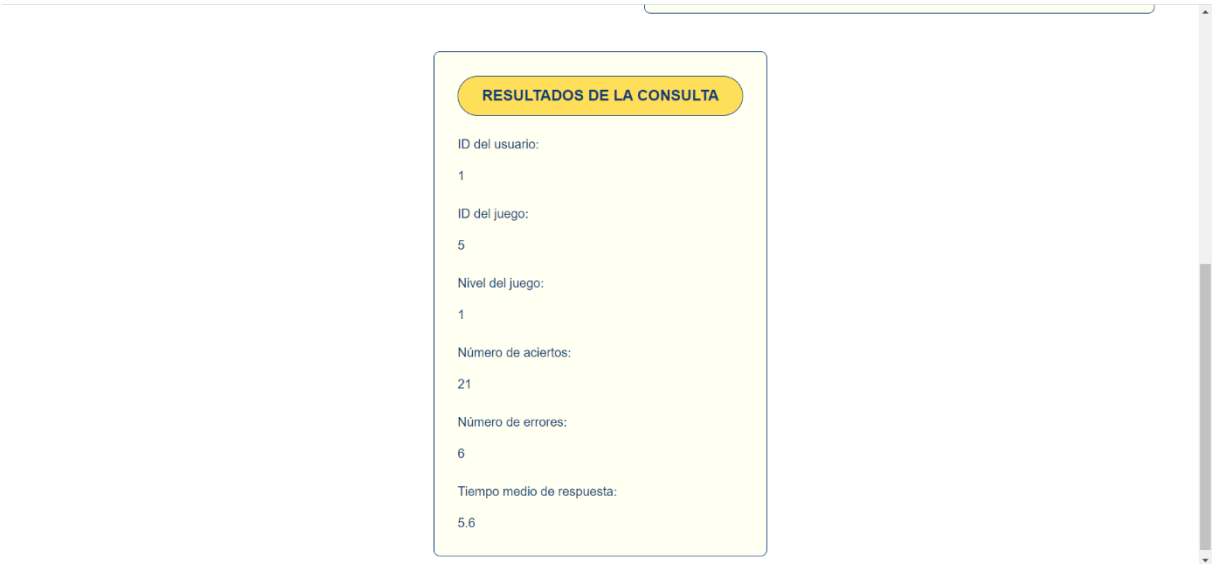

Figura 3.89.- Ejemplo de resultados obtenidos con consulta. Fuente: Diseño propio

Finalmente, destacar que la página web es “responsive”, por lo que se adaptará al tamaño de pantalla utilizado para su visualización (Figura 3.90).

Figura 3.90.- Muestra de página web adaptada a una pantalla menor. Fuente: Diseño propio

3.6.- MANUAL DE INSTALACIÓN

Hay varias opciones para añadir una skill a un dispositivo Amazon Alexa, cuya diferencia es que la skill esté o no publicada en la "tienda" de skills.

En primer lugar, para empezar a usar una skill publicada en la tienda, es necesario acceder a la página web de Alexa (https://www.amazon.es/b?ie=UTF8&node=13944662031) o abrir la aplicación de Alexa en un dispositivo móvil. Una vez dentro de la página web o de la aplicación móvil, es necesario iniciar sesión con la misma cuenta que se haya utilizado para configurar el dispositivo Alexa.

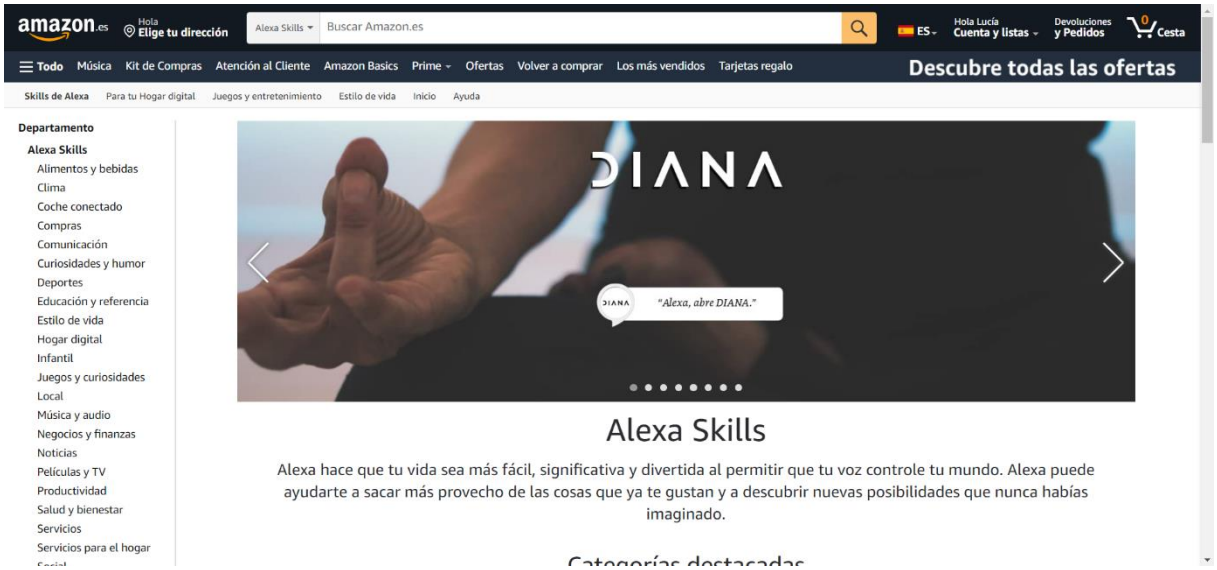

Figura 3.91.- Página principal de tienda de skills

En la tienda de skills, buscar HablaConmigo y hacer clic. Se abrirá una pantalla donde se muestra toda la información de la skill (Figura 3.92) (descripción, idiomas disponibles, comentarios...). Haciendo clic en "Activar" en la esquina superior derecha, se instalará la skill en todos los dispositivos asociados a esa cuenta, y diciendo "Alexa, abre Habla Conmigo", se abrirá la skill en el dispositivo elegido para empezar a jugar.

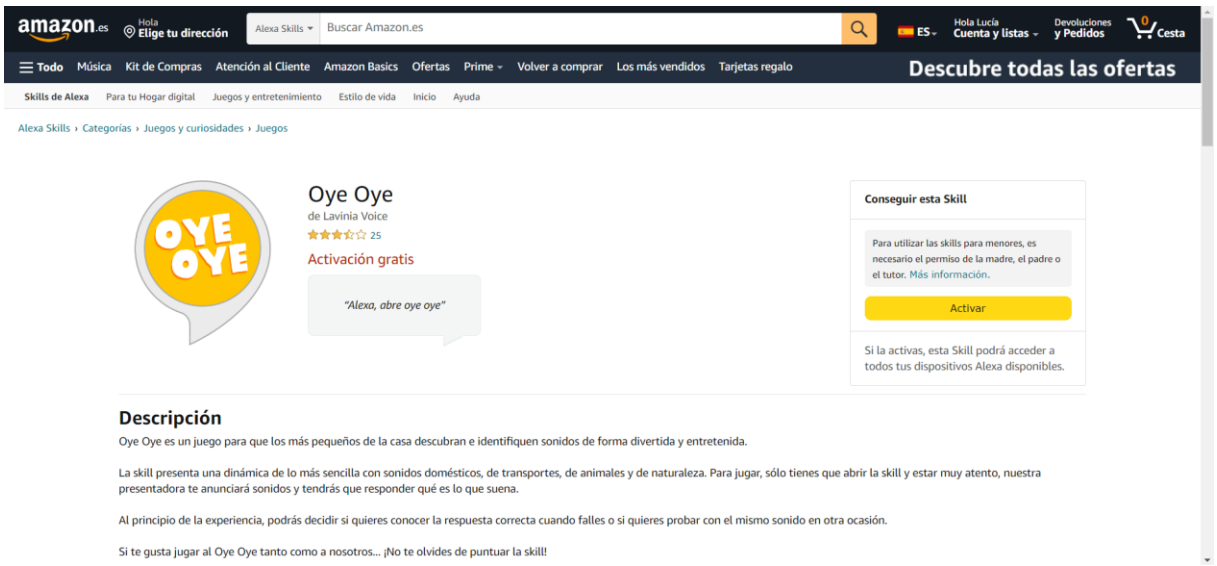

Figura 3.92.- Muestra de elección de skill en la tienda de skills para instalarla. Fuente: Diseño propio

En el caso de HablaConmigo, no se ha podido publicar debido a problemas asociados a la certificación de skills desarrolladas para niños. Para poder utilizarla, por tanto, es necesario ser un usuario "Beta tester". Esto implica solicitar al propietario de la skill acceso a ell. Este compartirá la skill con el correo electrónico del solicitante, que debe ser el que tiene la cuenta de Amazon asociada al dispositivo. Una vez hecho esto, el usuario ya podrá utilizar la skill en todos los dispositivos que utilicen su cuenta.

4. Trabajo futuro

Durante el desarrollo del proyecto, han sido muchas las nuevas ideas que han ido surgiendo para dar forma tanto a la skill HablaConmigo como a la aplicación web. Dado que algunas de estas ideas no han podido realizarse finalmente, se exponen a continuación como posibles futuras mejoras del sistema.

En primer lugar, se plantea la opción de poder subir las fotos de las pistas directamente desde la aplicación. Con el planteamiento actual, es necesario subir con anterioridad las fotos al servidor y lo que se incluye en el registro de una nueva entrada para un juego es la URL en la que se encuentra esa imagen en el servidor. De cara a facilitar el trabajo al logopeda, sería más eficiente que desde el propio formulario de la aplicación web se pudiese seleccionar una imagen del dispositivo que se añadiese en la petición y se subiese directamente.

También en relación web, sería interesante la opción de mostrar los resultados del usuario de una forma más gráfica. Por ejemplo, permitir seleccionar varios períodos de tiempo para que los represente en un gráfico y ver la evolución del paciente.

En cuanto a la skill, con el planteamiento actual, el usuario debe elegir un nivel tras dar una respuesta a un enunciado, aunque quiera seguir jugando al mismo nivel. Una posible mejora sería que una vez que el usuario elige un nivel se le mostrasen, por ejemplo, 4 enunciados distintos uno detrás de otro. De esta forma, el juego sería más dinámico y no tan repetitivo.

Finalmente, se plantea la posibilidad de añadir nuevos juegos a la skill que sirvan para trabajar otras áreas del lenguaje distintas a las actuales. Con esto, se lograría llegar a un público más amplio tratar a más usuarios.

5. Conclusiones

El propósito de este trabajo era el desarrollo de una skill para dispositivos "Amazon Alexa" que permitiera hacer más interactiva y atractiva la terapia en niños con trastornos comunicativos.

Previo a su desarrollo, fue necesaria una investigación sobre qué áreas del lenguaje eran las más afectadas con el fin de acotar una línea de trabajo. Además, se realizó un estudio sobre herramientas existentes en el mercado con la misma utilidad que la skill HablaConmigo, de cara a mejorar las opciones disponibles.

Durante el desarrollo de la skill se planteó la posibilidad de que el sistema incluyese una aplicación web de gestión, lo que permitiría al logopeda poder realizar consultas sobre la evolución de sus pacientes y añadir nuevas entradas a los juegos disponibles, utilizando para ello formularios web. Esto permite que el logopeda pueda ampliar y adaptar la skill a las necesidades de sus pacientes, sin necesidad de grandes conocimientos informáticos para ello.

Para la arquitectura del sistema se planteó un diseño similar al visto en la asignatura *Servicios Multimedia e Interactivos*, basado en una aplicación que realizaba peticiones a una API REST y esta API, a su vez, era la encargada de comunicarse con la base de datos del sistema. Para este trabajo se amplió ese diseño, ya que se disponía de una skill y de una aplicación web que necesitaban comunicación con la base de datos. Además, se eligió Node.js para el backend de la skill, al igual que en dicha asignatura. Por otro lado, para lograr que la aplicación web fuese totalmente adaptable a distintos tamaños de pantalla, se decidió utilizar Bootstrap, utilizado tanto en *Servicios de Comunicaciones Básicos* como en *Servicios Multimedia e Interactivos*.

Durante el desarrollo del proyecto han sido varios los "problemas" que han surgido y que ha habido que solucionar. Un ejemplo de esto fue que, hasta no comprender correctamente el uso de los atributos de sesión en el desarrollo de skills, no era posible cambiar de juego o nivel. También se realizaron distintos cambios en el diseño de la base de

datos. Por ejemplo, inicialmente se planteó que cuando se añadía una nueva entrada a la tabla de resultados, se añadiese el timestamp de inicio del juego y de fin. Al final, esto se modificó por la duración en milisegundos y un campo extra que generaba un timestamp en el momento que se añadía la nueva entrada a la tabla. Además, durante la fase de pruebas se mejoraron ciertos aspectos de la aplicación que no habían sido tenidos en cuenta durante el diseño y desarrollo de esta. Uno de estos aspectos es que previa a esta fase no se mostraba ningún tipo de mensaje que le indicase al logopeda si sus entradas se estaban añadiendo correctamente o no a la base de datos. Así, se modificó esto, de forma que cuando el logopeda añade una nueva entrada obtiene un mensaje de retroalimentación positiva en caso de una subida exitosa, o un mensaje de error en caso contrario.

En lo personal, la realización de este proyecto me ha supuesto un gran reto. Pese a haber trabajado previamente con ciertas tecnologías mencionadas anteriormente, como Node.js o Bootstrap, este trabajo ha supuesto ir más allá. Por ejemplo, he tenido que plantear desde 0 el diseño de una base de datos y la relación entre sus elementos, algo que no había hecho previamente. Tampoco había tenido la oportunidad de trabajar con el diseño de skills de Alexa. Esto ha supuesto emplear bastante tiempo en el desarrollo de la skill, ya que hay ciertas características (intents, utterances, APL...) únicos de la programación de skills para Alexa que requieren un total conocimiento y dominio previo al desarrollo de la aplicación.

En mi opinión, la creación de esta skill de Alexa para niños con trastornos comunicativos es una innovación prometedora en el campo de la logopedia, ya que combina el creciente uso de altavoces inteligentes con la terapia que necesitan estos niños, brindándoles una forma adicional de práctica, más dinámica y entretenida, y fácilmente accesible.

6. Bibliografía

[1] «NeuronUP,» 2022. [En línea]. Available: https://www.neuronup.com/estimulacion-y-rehabilitacion-cognitiva/trastornos-del-neurodesarrollo/trastorno-del-lenguaje-causas-sintomas-diagnostico-e-intervencion/#:~:text=El%20Trastorno%20del%20lenguaje%20(TEL,a%20encontrar%20dos%20TEL%20iguales.. [Último acceso: 05 Julio 2023].

[2] «Neuronas en crecimiento,» 2019. [En línea]. Available: https://neuropediatra.org/2019/01/11/tel-trastorno-especifico-del-lenguaje-tel/. [Último acceso: 05 Julio 2023].

[3] Daniel Eduardo Alvarez-Amado, Eduardo Javier Barragán-Pérez, «A propósito de la pandemia COVID-19: hablemos con los padres con respecto a los trastornos del desarrollo del lenguaje,» *Revista Médica Clínica Las Condes*, vol. 33, nº 5, pp. 450-457, 2022.

[4] D. d. Cambridge. [En línea]. Available: https://dictionary.cambridge.org/es/diccionario/ingles-espanol/virtual-assistant. [Último acceso: 05 Julio 2023].

[5] «Alisys,» [En línea]. Available: https://alisys.net//es/blog/de-audrey-a-siri-historia-y-uso-de-los-asistentes-de-virtuales. [Último acceso: 05 Julio 2023].

[6] I. Archives. [En línea]. Available: https://www.ibm.com/ibm/history/exhibits/specialprod1/specialprod1_7.html. [Último acceso: 05 Julio 2023].

[7] C. d. Castillo, «El Proyecto CALO: el desconocido 'padre' militar de Siri, Alexa, Cortana y el resto de asistentes virtuales,» *elDiario*, 21 Septiembre 2020.

[8] C. d. Colores, «Crecer de Colores,» 11 Agosto 2022. [En línea]. Available: https://www.crecerdecolores.com/post/qu%C3%A9-es-la-conciencia-fonol%C3%B3gica-y-cu%C3%A1l-es-su-importancia. [Último acceso: 05 Julio 2023].

[9] Bárbara Campo y María Belmonte, «Abaterapia,» [En línea]. Available: https://abaterapia.com/comportamiento/desarrollo-del-lenguaje/.

[10] N. T. y. J. Peña-Casanova. [En línea]. Available: https://drrodolfokestler.weebly.com/uploads/3/2/1/7/32170793/cap._9a._dislalias._pe%

C3%B1a-casanova_j..pdf. [Último acceso: 05 Julio 2023].

[11] S. Calvo Sánchez, «Las dislalias en la etapa de Educación Infantil,» *Publicaciones Didácticas*, nº 102, pp. 348-351, 2012.

[12] C. N. A. Alayón, «Centro Neurológico Antonio Alayón,» 2 Julio 2018. [En línea]. Available: https://www.antonioalayon.com/logopedia-la-dislalia-afecta-al-15-de-los-ninos-en-preescolar/. [Último acceso: 05 Julio 2023].

[13] E. d. E. e. Educación, «Universidad Internacional de Valencia,» 20 Noviembre 2014. [En línea]. Available: https://www.universidadviu.com/es/actualidad/nuestros-expertos/que-es-la-dislalia-de-l-y-como-corregirla. [Último acceso: 05 Julio 2023].

[14] «Psikipedia,» [En línea]. Available: https://psikipedia.com/libro/psicopatologia-1/dislalias-infantiles-retraso-simple-del-lenguaje-y-del-habla. [Último acceso: 05 Julio 2023].

[15] C. O. Pulido, «El País,» 14 Noviembre 2018. [En línea]. Available: https://cincodias.elpais.com/cincodias/2018/11/14/companias/1542195734_689097.html. [Último acceso: 05 Julio 2023].

[16] «Lee Con Leo,» [En línea]. Available: https://leeconleo.es/. [Último acceso: 05 Julio 2023].

[17] M. A. López, «Eres mamá,» 7 Agosto 2021. [En línea]. Available: https://eresmama.com/apps-ninos-trastorno-especifico-lenguaje/. [Último acceso: 05 Julio 2023].

[18] C. M. Carrero, «Educación 3.0,» [En línea]. Available: https://www.educaciontrespuntocero.com/recursos/aplicaciones-trastorno-especifico-lenguaje/#Hablando-con-Nok. [Último acceso: 05 Julio 2023].

[19] M. Khare, 2021. [En línea]. Available: https://kinsta.com/es/blog/nodejs-vs-python/. [Último acceso: 05 Julio 2023].

[20] «Kinsta,» 2023. [En línea]. Available: https://kinsta.com/es/base-de-conocimiento/que-es-node-js/. [Último acceso: 05 Julio 2023].

[21] «Hostinger,» 19 Abril 2023. [En línea]. Available: https://www.hostinger.es/tutoriales/que-es-node-js. [Último acceso: 05 Julio 2023].

[22] «MDN Web Docs,» 2023. [En línea]. Available: https://developer.mozilla.org/es/docs/Web/JavaScript. [Último acceso: 05 Julio 2023].

[23] «Open Bootcamp,» [En línea]. Available: https://open-bootcamp.com/aprender-programar/lenguaje-interpretado. [Último acceso: 05 Julio 2023].

[24] «Trucoteca,» [En línea]. Available: https://trucoteca.com/que-es-un-lenguaje-de-programacion-multiparadigma/. [Último acceso: 05 Julio 2023].

[25] «Uniwebsidad,» [En línea]. Available: https://uniwebsidad.com/libros/javascript/capitulo-1/sintaxis. [Último acceso: 05 Julio 2023].

[26] «MDN Web Docs,» 2023. [En línea]. Available: https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Grammar_and_types. [Último acceso: 05 Julio 2023].

[27] D. P. Acharya, «Kinsta,» 15 Junio 2023. [En línea]. Available: https://kinsta.com/es/blog/bibliotecas-javascript/#las-bibliotecas-javascript-ms-populares. [Último acceso: 05 Julio 2023].

[28] «Arimetrics,» [En línea]. Available: https://www.arimetrics.com/glosario-digital/jquery. [Último acceso: 05 Julio 2023].

[29] «Hostinger,» 5 Abril 2023. [En línea]. Available: https://www.hostinger.es/tutoriales/que-es-jquery#Ventajas_y_desventajas_de_jQuery. [Último acceso: 05 Julio 2023].

[30] «Desarrollo Web,» 18 Diciembre 2021. [En línea]. Available: https://desarrolloweb.com/articulos/que-es-el-dom.html. [Último acceso: 05 Julio 2023].

[31] «Arimetrics,» [En línea]. Available: https://www.arimetrics.com/glosario-digital/bootstrap. [Último acceso: 05 Julio 2023].

[32] «Bootstrap,» [En línea]. Available: https://getbootstrap.esdocu.com/docs/5.1/getting-started/introduction/. [Último acceso: 05 Julio 2023].

[33] «40 de Fiebre,» [En línea]. Available: https://www.40defiebre.com/que-es/diseno-responsive. [Último acceso: 5 Julio 2023].

[34] «Bootstrap,» [En línea]. Available: https://getbootstrap.esdocu.com/docs/5.1/layout/grid/. [Último acceso: 05 Julio 2023].

[35] «Bootstrap,» [En línea]. Available: https://getbootstrap.esdocu.com/docs/5.1/layout/breakpoints/. [Último acceso: 05 Julio

2023].

[36] «Desarrollo Web,» 2001. [En línea]. Available: https://desarrolloweb.com/articulos/que-es-html.html. [Último acceso: 05 Julio 2023].

[37] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/HTML. [Último acceso: 05 Julio 2023].

[38] «Proyecto Degardo,» [En línea]. Available: http://proyectodegardo.weebly.com/tipos-de-etiquetas.html. [Último acceso: 05 Julio 2023].

[39] «Recursos Ceainf,» 2016. [En línea]. Available: http://recursosceainf9.blogspot.com/2016/10/unidad-5-atributos-en-html.html. [Último acceso: 05 Julio 2023].

[40] «MDN Web Docs,» [En línea]. Available: https://developer.mozilla.org/es/docs/Learn/CSS/First_steps/What_is_CSS. [Último acceso: 05 Julio 2023].

[41] «Hostinger,» 11 Enero 2023. [En línea]. Available: https://www.hostinger.es/tutoriales/que-es-css. [Último acceso: 05 Julio 2023].

[42] «RailsBridge,» [En línea]. Available: http://es.railsbridge.org/frontend/CSS_basico. [Último acceso: 05 Julio 2023].

[43] «MDN Web Docs,» [En línea]. Available: https://developer.mozilla.org/es/docs/Web/CSS/Pseudo-classes. [Último acceso: 05 Julio 2023].

[44] «MDN Web Docs,» [En línea]. Available: https://developer.mozilla.org/es/docs/Web/CSS/Pseudo-elements. [Último acceso: 05 Julio 2023].

[45] «Axios,» [En línea]. Available: https://axios-http.com/es/docs/intro. [Último acceso: 05 Julio 2023].

[46] «Agencia Tributaria Española,» 2022. [En línea]. Available: https://sede.agenciatributaria.gob.es/Sede/ayuda/manuales-videos-folletos/manuales-practicos/folleto-actividades-economicas/3-impuesto-sobre-renta-personas-fisicas/3_5-estimacion-directa-simplificada/3_5_4-tabla-amortizacion-simplificada.html. [Último acceso: 05 Julio 2023].

[47] A. Developer, «Alexa Developer,» [En línea]. Available:

https://developer.amazon.com/es-ES/alexa/alexa-skills-kit. [Último acceso: 05 Julio 2023].

[48] Microsoft, «Visual Studio Code,» [En línea]. Available: https://visualstudio.microsoft.com/es/#vscode-section. [Último acceso: 05 Julio 2023].

[49] «EDteam,» 2020. [En línea]. Available: https://ed.team/blog/10-extensiones-de-visual-studio-code-que-debes-instalar-ya. [Último acceso: 05 Julio 2023].

[50] V. S. Code, «Visual Studio Code,» [En línea]. Available: https://code.visualstudio.com/docs/editor/intellisense. [Último acceso: 05 Julio 2023].

[51] «Stack Overflow,» 2021. [En línea]. Available: https://insights.stackoverflow.com/survey/2021#most-popular-technologies-new-collab-tools. [Último acceso: 05 Julio 2023].