ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN.

GRADO EN INGENIERÍA EN TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

ÁREA DE INGENIERÍA TELEMÁTICA

SISTEMA DE DETECCIÓN DE EMOCIONES

GRAÑA COLUBI, EVA
TUTOR: D.CORCOBA MAGAÑA VÍCTOR

FECHA: Julio 2023

Índice general

Índice de figuras

Índice de cuadros

1. Introducción

Alexa es un servicio de voz desarrollado por Amazon que se encuentra en sus dispositivos y ofrece una interacción fácil e intuitiva con su tecnología. Este servicio ofrece la posibilidad de añadir Skills, que son funcionalidades extra que puede crear cualquier persona para instalar en los dispositivos. En la página web de Amazon[1] se pueden encontrar muchas de estas Skills. En este proyecto se creará una skill que realice un seguimiento del estado anímico del usuario mediante preguntas directas y tests para evaluar la depresión, además de añadir algunos consejos para mejorar el ánimo.

1.1.- Objetivo y motivación

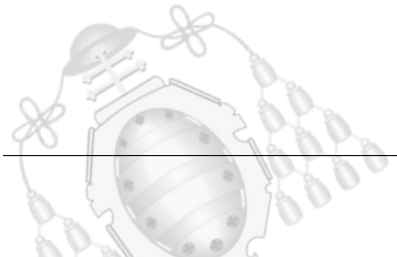
El principal objetivo de este proyecto es la realización de una Skill de Alexa que sirva de apoyo para personas con depresión y ayude a mejorar su ánimo.

La intención es hacer un seguimiento de su estado de ánimo de forma asidua para ayudar a comprender mejor sus propias emociones y tomar medidas para mejorar su bienestar mental. Para ello, se le hacen preguntas sobre cómo se encuentra y se aportan algunos consejos como que medite o salga a dar un paseo. Además, cuenta con tests que miden la depresión y el estado rumiativo del usuario, factor que influye mucho en el ánimo, y se pueden establecer recordatorios para que no se olvide de mantener un registro frecuente.

Por otro lado, el proyecto consiste en desarrollar el diseño y la implementación de una interfaz de usuario con Amazon Alexa, por lo que se establecen también unos objetivos específicos:

- Analizar los distintos métodos de evaluación de la depresión para escojer el que mejor se adapte al proyecto
- Investigar formas de mejorar el ánimo en personas con depresión para ofrecer consejos útiles

- Explorar las aplicaciones existentes para tratar la depresión o ayudar a mantener un ánimo positivo
- Estudiar la arquitectura de sistema de Amazon Alexa y análisis de su funcionamiento
- Aprender el uso de Alexa Developer Console, para el desarrollo frontend y backend de la skill
- Aplicar Alexa Presentation Language



2. Estudios y Análisis previos

La OPS (Organización Panamericana de la Salud) define la depresión como: "enfermedad que se caracteriza por una tristeza persistente y por la pérdida de interés en las actividades con las que normalmente se disfruta, así como por la incapacidad para llevar a cabo las actividades cotidianas, durante al menos dos semanas."[2]

En la actualidad, la salud mental es un tema de gran importancia y preocupación en todo el mundo. Muchas personas sufren de estrés, ansiedad, depresión y otros trastornos emocionales que afectan negativamente su calidad de vida y su capacidad para realizar sus actividades diarias.

Según el estudio de IHME (Institute for Health Metrics and Evaluation) Global Burden of Disease (GBD) de 2019, aproximadamente un 3,8 % de la población sufre depresión, incluyendo un 5 % de los adultos y un 5,7 % de los adultos de más de 60 años[3] (figura 2.1).

Además, la pandemia vivida recientemente ha afectado significativamente a la salud mental de la población. En un estudio de la INE (Instituto Nacional de Estadística) se comparan algunos indicadores de la salud mental antes y después de la pandemia[4] (figura 2.2).

Este mismo estudio indica los siguientes datos: "En 2020, se cifra en 5,4 % la población con algún tipo de cuadro depresivo, son 2,1 millones de personas. Atendiendo a la severidad de la sintomatología, 230.000 se consideran graves."[4]

La tecnología puede ser una herramienta útil para ayudar con este problema cada vez más acuciante. En este caso se pretende usar una aplicación que cuenta con una interfaz por voz.

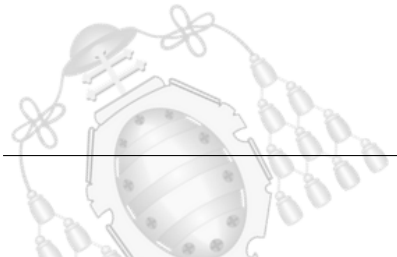
Las interfaces de usuario por voz (VUI, por sus siglas en inglés) son sistemas que permiten interactuar con una aplicaciones mediante el uso de la voz en

lugar de interactuar con un teclado o pantalla. Una VUI puede estar presente en muchos dispositivos diferentes como móviles, altavoces inteligentes, asistentes virtuales, automóviles, electrodomésticos y otros dispositivos electrónicos.

Esta tecnología se está usando cada vez más, ya que ofrece muchas ventajas, como ser más intuitiva y accesible para para personas con discapacidades motoras o auditivas. Por ello se puede encontrar en diversos escenarios, como pueden ser:

- Asistentes virtuales: el sistema usado en este proyecto es el asistente virtual de Amazon, Alexa. También existe Siri, de Apple, o Google Assistant. Se pueden encontrar integrados en distintos dispositivos como móviles, altavoces o tabletas y permiten al usuario hacerles consultas o realizar tareas como establecer recordatorios.
- Automatización del hogar: existen sistemas de domótica que pueden controlar ciertos dispositivos del hogar como luces, termostatos o cerraduras de las puertas y que utilizan VUI de forma que puedan controlarse mediante comandos de voz.
- Industria: esta tecnología también se usa en la industria, ya que puede ser de especial ayuda para trabajadores que deben realizar tareas complejas con las manos y no pueden usarlas para controlar otros dispositivos. Por ejemplo, el uso de comandos por voz puede ser útil para trabajadores en almacenes para realizar tareas como el seguimiento de inventario o la gestión de pedidos.

En resumen, es muy útil y muy usada a día de hoy en ámbitos muy diversos.



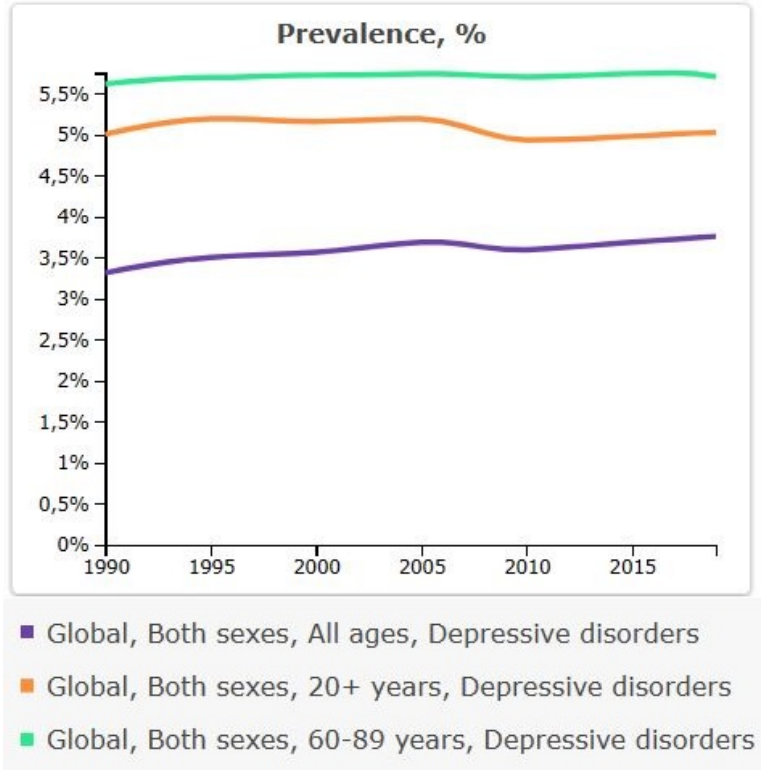


Figura 2.1.- Global Burden of Disease Study (Fuente: Institute for Health Metrics and Evaluation)

Figura 2.2.- La salud mental en pandemia. (Fuente: Instituto Nacional de Estadística)

2.1.- Escalas para medir el bienestar mental

Existen diversos métodos de evaluación psicológica, como son las entrevistas abiertas, los instrumentos estandarizados y los auto-registros. En este proyecto se van a usar los dos últimos, ya que las entrevistas abiertas requieren una persona que pueda registrar la respuesta, pues no son seleccionables entre items concretos. Como instrumentos estandarizados se usarán escalas y cuestionarios, que sirven para la cuantificación de la gravedad y el impacto de la depresión. En concreto, se usarán tres cuestionarios distintos, dependiendo de la edad del usuario.

Si este es mayor de 67 años, se hará uso de la escala de depresión geriátrica de Yesavage (GDS por sus siglas en inglés) en su versión abreviada adaptada al español (ver anexo 8), ya que los parámetros de fiabilidad y validez son similares a los del cuestionario original[5].

Si su edad está comprendida entre los 17 y los 67, se usará el Inventario de Depresión de Beck[6] (ver anexo 8), ya que es un instrumento muy utilizado para detectar y evaluar la gravedad de la depresión. Este consiste en un autoinforme compuesto por 21 ítems de tipo Likert, es decir, se evalúa el nivel de acuerdo o desacuerdo con la cuestión propuesta. Dado que no se busca una rigurosidad científica, sino hacer un pequeño análisis de cómo se encuentra el usuario, se ha optado por introducir una ligera modificación en este inventario. Para reducir el tiempo que se tarda en completarlo, se leerá solo una frase de cada item y se marcará el nivel de acuerdo con ella, en lugar de leerse las cuatro opciones disponibles, ya que hacía pesado el desarrollo de la actividad. Generalmente se intercalan items en positivo y negativo para evitar la aquiescencia, es decir, la tendencia a responder siempre de forma afirmativa con independencia del contenido. Pero, para facilitar la comprensión del test, se usarán solo ítems formulados en positivo, ya que hay estudios que demuestran que el uso de ítems negativos puede generar confusión e inconsistencia en las respuestas[7][8].

En el caso de que el usuario sea menor de 17 años, se usará el Inventario de Depresión para Niños (CDI) de Kovacks y Beck [9](ver anexo 8). Este es parecido al anteriormente citado, por lo que se le han practicado las mismas modificaciones. También se ha añadido un test que evalúa el estado rumiativo del usuario, ya que hay

estudios que correlacionan la depresión con este estado[10]. Este cuestionario, llamado Escala de Respuestas Rumiativas(ver anexo 8), fue desarrollado por Susan Nolen-Hoeksema y sus colaboradores [11]. Además, según un estudio de Moshe Bar y Malia F. Mason, se puede mejorar el estado de ánimo al pasar a un estilo de pensamiento ampliamente asociativo, evitando que los procesos mentales se centren en un tema limitado y negativo[12]. En este estudio, conseguían que los sujetos llegaran a este estado simplemente haciéndoles leer listas asociativas de palabras individuales que avanzan ampliamente. En la Skill se han cogido algunas de las listas usadas en este estudio para que Alexa las lea si el usuario da una puntuación baja en el estado de ánimo o si directamente las pide.

2.2.- Consejos

Para ayudar a mejorar el ánimo del usuario se han incorporado una serie de consejos que este puede seguir para mejorar su estado de ánimo. Por ejemplo, se le anima a dar un paseo o meditar, ya que estas actividades pueden contribuir a un aumento en el volumen del hipocampo, zona del cerebro que se ve deteriorada debido a la depresión[13][14]. Como ayuda para la meditación, Alexa muestra una pantalla con música relajante. También se le anima a leer, ya que hay estudios que correlacionan esta actividad con una mejora del estado depresivo. Por ejemplo, según el artículo "Entrenamiento cognitivo: efectos en la cognición, depresión y actividades de la vida diaria en sujetos institucionalizados"[15]:

"El entrenamiento cognitivo demostró ser útil para disminuir los índices de depresión y mejorar el estado cognitivo." [...] "Es interesante el estudio de Thompson y Foth, quienes mencionan que otras actividades que también estimulan la función mental son viajar, leer, tomar cursos continuos de educación, memorizar poesía, cantar en coros, aprender a tocar un instrumento musical, participar en juegos de cartas, crucigramas y rompecabezas, así como leer novelas de misterio, pues estas tienen elementos de razonamiento inductivo y deductivo."

O el estudio "Epidemiología de la depresión en el adulto mayor" de la "revista médica herediana" [16], que indica:

"un estudio transversal en 1012 personas mayores de 60 años, encontró menor frecuencia de depresión en personas con indicadores de envejecimiento activo, como tener amigos, trabajo y hábito de leer".

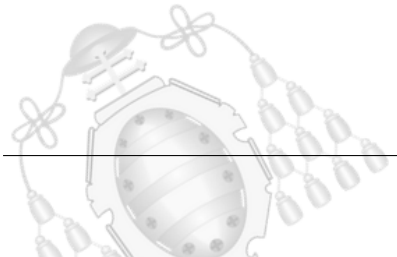
Siguiendo estas indicaciones, otro de los consejos que propone Alexa es que el usuario hable con algún amigo o familiar. Además, la skill cuenta con una lista de chistes que Alexa propone escuchar en la batería de consejos, ya que la risa cuenta con muchos beneficios. El estudio "El valor del humor en el proceso psicoterapuéutico" de la revista "Psicodebate"[17] menciona algunos de estos, que se pueden leer en el Anexo 8.

2.3.- Proyectos similares

En esta sección se van a comparar diferentes soluciones tecnológicas existentes para tratar la depresión con la propuesta.

Cómo me encuentro. [18]	Skill de Alexa que hace un seguimiento del estado anímico del usuario mediante preguntas y recordatorios.	No incluye consejos para mejorar el ánimo ni test que evalúen el nivel de depresión.
Calm. [19]	Aplicación de móvil que ofrece herramientas para la relajación como música o ejercicios de respiración.	No tiene test que evalúen el nivel de depresión y la mayoría de opciones son de pago.
Track your Happiness. [20]	Sitio web que hace una serie de preguntas durante varios días para ayudarte a descubrir qué te hace feliz.	No tiene test que evalúen el nivel de depresión ni ofrece consejos diarios para mejorar el ánimo.
What's Up? [21]	Aplicación de móvil que ofrece ejercicios de relajación, técnicas de respiración y un diario de hábitos.	No tiene test que evalúen el nivel de depresión ni tiene opción de usarla en español.
Feel like shit? [22]	Sitio web que hace una serie de preguntas para que el usuario identifique sus sensaciones y ofrece algunos consejos para que se sienta mejor.	No tiene test que evalúen el nivel de depresión ni recordatorios, además solo se encuentra en inglés

Cuadro 2.1.- Proyectos similares



3. Fundamentos de las tecnologías empleadas en el proyecto

En este proyecto se usan cinco tecnologías: Alexa Skills Kit, APL, Node.JS, SQL y angular.

3.1.- Alexa Skills Kit

Alexa Skills Kit [23] es un conjunto de herramientas y servicios proporcionados por Amazon para poder crear aplicaciones (denominadas Skills) para Alexa. Incluye una amplia gama de recursos para crear habilidades personalizadas para Alexa, incluidos tutoriales, plantillas, documentación y una interfaz de programación de aplicaciones.
Para poder crear Skills hace falta una cuenta de Amazon developer y se desarrollan principalmente en la consola de Amazon, aunque también se puede conectar la skill a un repositorio de GitHub para poder desarrollar localmente. La interfaz de programación está compuesta por dos partes: el front-end, donde creas los llamados "intents", que son las funcionalidades de la aplicación, y con qué frases se activan, y el back-end, donde se programa lo que hace cada intent. Para ello se puede utilizar node.js o python. En este proyecto se ha elegido la primera opción, ya que en los tutoriales seguidos para aprender a usar esta tecnología, es el entorno en el que se explica.
Para crear la skill hay que acceder a la consola de Amazon, donde aparecerá la pantalla que podemos ver en la figura 3.1. En ella podemos ver las skills que tenemos o crear una. Cuando entramos en una skill nos aparece la pantalla que podemos ver en la figura 3.2:

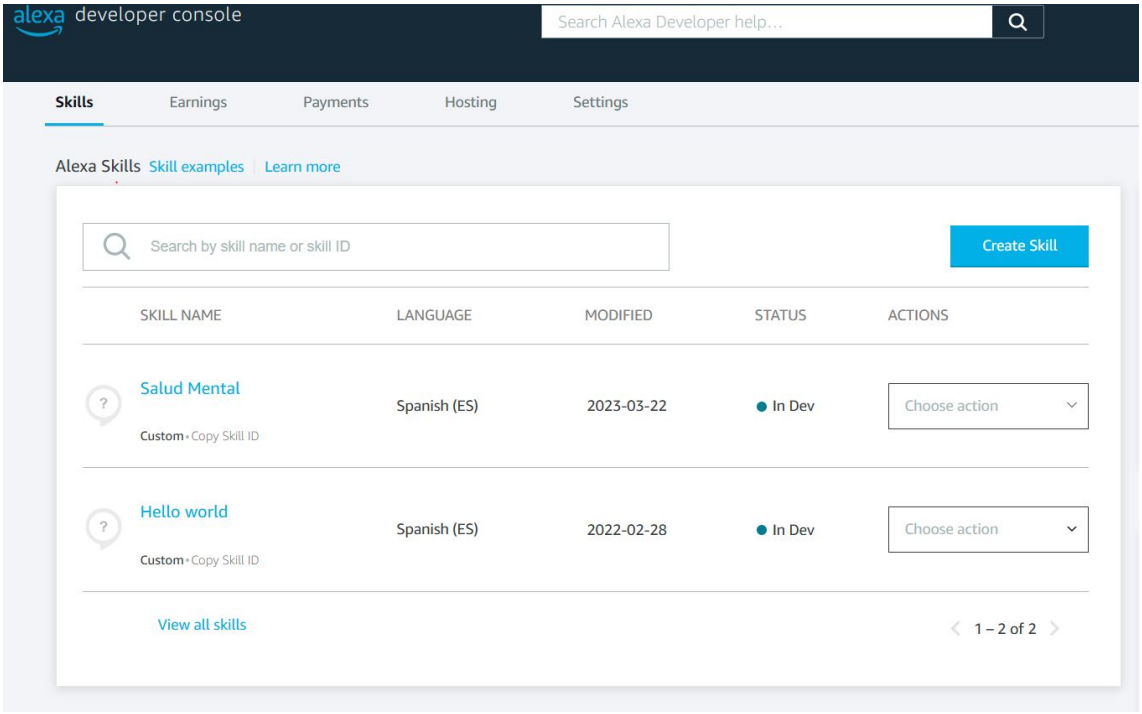


Figura 3.1.- Consola de desarrollo Alexa Skills (Fuente: propia)

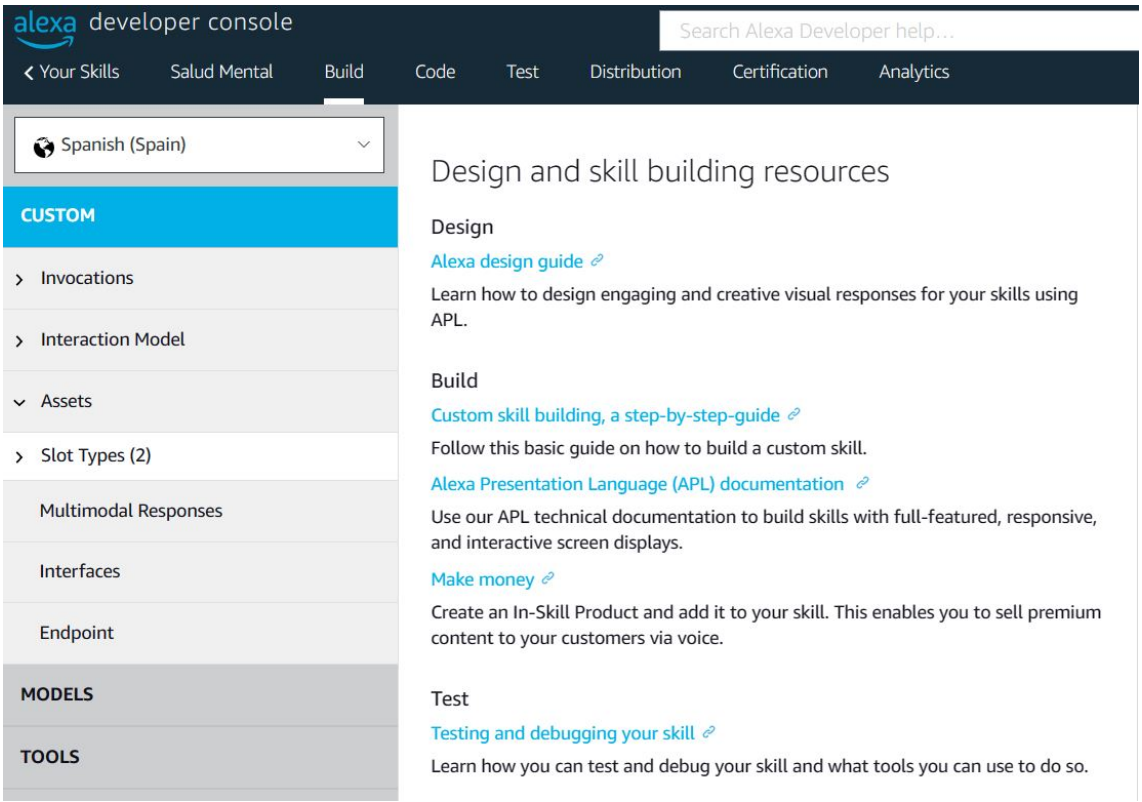


Figura 3.2.- Consola de desarrollo Alexa Skills (Fuente: propia)

Figura 3.3.- Consola de desarrollo de Amazon: Sample Utterances (Fuente: propia)

En la pestaña "Build" se elije el nombre de la skill, la palabra o frase con la que se abrirá y los *intents* y *slots*.

- Un *intent* representa una acción que se activará cuando el usuario diga el comando de voz asociado, llamado "sample utterance". Cada *intent* puede tener uno o varios asociados y se eligen al crearlo, como se puede ver en la figura 3.3. Existen varios *intents* predefinidos, llamados "built-in intents" con *sample utterances* asociados. Hay cuatro obligatorios: AMAZON.CancelIntent, AMAZON.HelpIntent, AMAZON.StopIntent y AMAZON.NavigateHomeIntent. Estos se añaden a la skill automáticamente al crearla. También se pueden añadir otros *built-in intents* útiles como el AMAZON.YesIntent o el AMAZON.NoIntent. AL añadir nuevos *intents*, da la opción de crear uno personalizado o añadir uno ya creado, como se ve en la figura 3.4.
- Para añadir información a los *intents*, se usan los llamados *slots*, que son los datos que proporciona el usuario y con los que se trabajará en el código. Para asegurarse de que el usuario rellena los *slots* necesarios y que Alexa los guarda correctamente, se puede usar la delegación de diálogo de forma que, si el usuario no da la información requerida, Alexa preguntará por ella explícitamente y, tras obtenerla, repetirá los datos para comprobar si son correctos. También existen *slots* predefinidos para datos típicos como un número o una fecha o se pueden crear personalizados. Además, se pueden añadir ciertas validaciones, por ejemplo,

si es un número, que esté entre 1 y 10, para no tener que controlarlo desde el código.

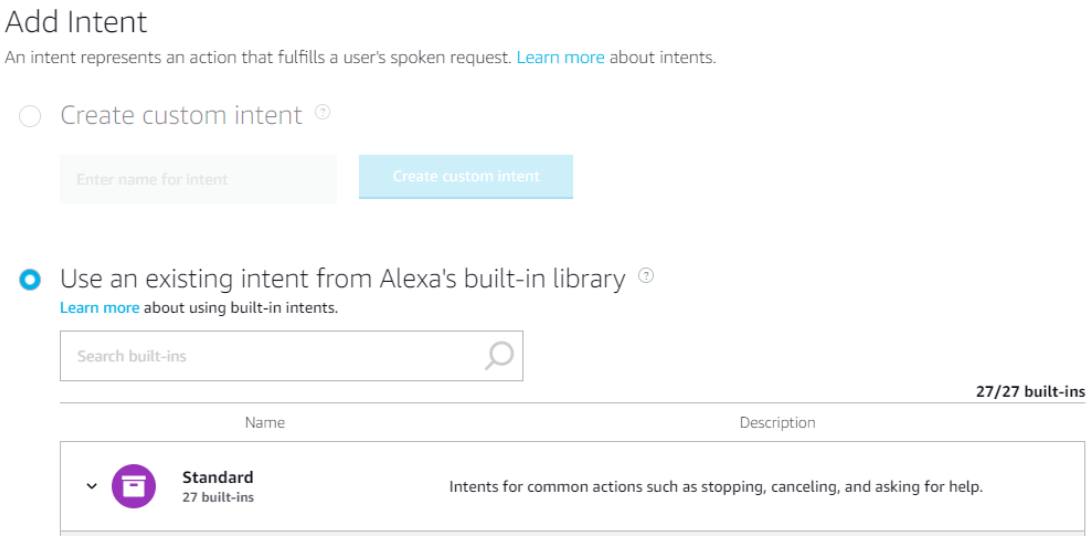


Figura 3.4.- Consola de desarrollo de Amazon: añadir Intents (Fuente: propia)

En la pestaña "Code" se programa lo que hará la skill dependiendo de los comandos de voz utilizados por el usuario. En la pestaña "Test" se puede probar la skill con asistente virtual similar al Alexa físico, pero desde la web. En la sección 5.2.2 se hablará más ampliamente de esta consola.

3.2.- Node.js

Node.js[24] es un entorno o conjunto de herramientas que permite ejecutar código javascipt y se utiliza para construir aplicaciones del lado del servidor. Se puede instalar y ejecutar en múltiples sistemas operativos. Se basa en el motor de JavaScript V8 de Google, que está optimizado para ejecutar código de JavaScript de manera rápida y eficiente. Además, Node.js incluye un conjunto de bibliotecas y herramientas integradas que facilitan la creación de aplicaciones web del lado del servidor.

Se puede descargar el código fuente o un instalador pre-configurado desde la página oficial. Para poder usar distintas versiones de node cambiando de una a otra fácilmente se usa NVM (Node Version Manager), el cual se puede instalar con el comando "curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh — bash" en Linux y Mac o, si se usa Windows, siguiendo las instrucciones del repositorio de GitHub

nvm-windows[25] de coreybutler. Con esta herramienta, se pueden instalar distintas versiones de node a través del comando “nvm install <versión>”, ver las distintas versiones instaladas con el comando “nvm list” y cambiar de versión activa con “nvm use <versión>”.
En este proyecto se usa Node.js tanto para la programación de la skill de Alexa como para la creación de una aplicación web con angular, donde se podrán ver las estadísticas del usuario. En el caso de la aplicación web, se usa la versión de node 14.20.0 y para la skill se usa la versión 16.

3.3.- Alexa Presentation Languaje

El Alexa Presentation Language (APL)[26] es un lenguaje de diseño y presentación que permite añadir imágenes, texto, sonido, vídeo o una combinación de varios en dispositivos de Alexa que lo permitan, como el Echo Shows, Fire TV y muchos más.
APL funciona con una estructura parecida a los comandos de voz, se envía una petición y se devuelve una respuesta. En este caso, en la respuesta se incluye una directiva para mostrar una respuesta visual. Esta directiva tiene dos elementos, el documento APL, que define qué plantilla se va a usar y permite usar condiciones para que se adapte al dispositivo que se esté usando, y el *APL data source*, donde irán los datos que se quieren mostrar. De esta forma se separa la forma en que se va a mostrar la información de los datos concretos.
Para crear una respuesta visual, hay que entrar en la opción “Multimodal Responses” de la pestaña “Build” en la consola de desarrollo (figura 3.2). Existen dos opciones, crearla desde cero con la herramienta “Athoring Tool”, para lo que se requiere cierto conocimiento de programación con APL, o hacerlo a partir de una plantilla con la herramienta “Response Builder”, que permitirá personalizarla de una forma más sencilla, ya que se edita a través de una interfaz gráfica, como puede verse en la figura 3.5. Una vez creada la plantilla, se generará automáticamente el código que se debe incluir en la skill para poder usarla.

Aunque el APL proporciona una experiencia más rica y envolvente para el usuario al añadir información visual o permitir al usuario interactuar con elementos en la pantalla,

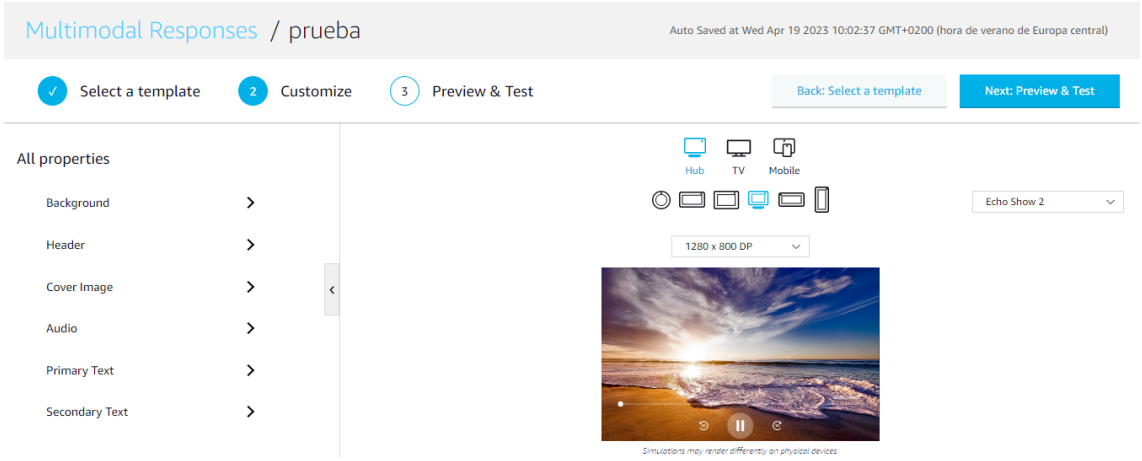


Figura 3.5.- Consola de desarrollo de Amazon: Response Builder (Fuente: propia)

como botones o deslizadores, al usar esta tecnología se debe tener presente que en una skill de Alexa lo principal es la voz. Es decir, puede apoyarse en imágenes o texto, pero toda la información importante debe ir en lo que dice Alexa, ya que la finalidad de este asistente es que el usuario se comunique con él de forma oral, sin necesidad de atender a una pantalla.

También hay que tener en cuenta que no todos los dispositivos de Alexa tienen pantalla, por lo que antes de añadir los documentos APL, habrá que comprobar si el dispositivo lo permite.

3.4.- SQL

SQL (Structured Query Language)[27] es un lenguaje de programación para trabajar con conjuntos de datos y las relaciones entre ellos. Fue desarrollado en la década de 1970 por IBM y actualmente es un estándar reconocido por la Organización internacional de Normalización (ISO) y el Instituto Nacional Estadounidense de Estándares (ANSI) .

Hay varios tipos de bases de datos SQL, las más populares son: Oracle, MySQL, Microsoft SQL Server, PostgreSQL y SQLite. En este proyecto en concreto se usa MySQL.

Este lenguaje permite a los usuarios crear, modificar y consultar datos almacenados en una base de datos relacional. Algunos de los comandos básicos son: SELECT, FROM,

WHERE, INSERT, DELETE y UPDATE. La instrucción SELECT, se usa para obtener los datos que buscamos, pasándole una descripción de los mismos, como las tablas donde se encuentran, que se especifica con el comando FROM. Con la instrucción WHERE se puede acotar más la búsqueda definiendo parámetros que deben cumplir los datos solicitados. Por ejemplo, en una base de datos donde se encuentra la tabla "tiendas", que contiene diversas tiendas de Asturias distribuidas entre sus ciudades, se pretende obtener solo las que se encuentren en Oviedo. Esto se conseguiría con la instrucción: *SELECT * FROM tiendas WHERE ciudad = "Oviedo"*. El operador "*" indica que se quieren obtener todos los campos de la tabla. Si se busca solo uno en concreto, por ejemplo, el nombre, se especificaría de la siguiente manera: *SELECT nombre FROM tiendas WHERE ciudad = "Oviedo"*. También se pueden unir tablas para seleccionar datos que cumplan ciertos criterios en relación a ambas. Por ejemplo, se quiere borrar de la tabla "clientes" aquellos que sean no usuarios de las tiendas de Oviedo. Esto se haría se la siguiente forma: *DELETE clientes FROM clientes INNER JOIN tiendas ON clienteId = clientes.Id WHERE NOT ciudad = "Oviedo"*. También se pueden unir varios criterios con los operadores AND y OR.
De esta forma se pueden conseguir datos precisos, crear o eliminar tablas, insertar datos, actualizar y eliminar registros y realizar multitud de consultas complejas.

3.5.- Angular

Angular[28] es una plataforma de desarrollo mantenida por Google y desarrollada en TypeScript que sirve para crear aplicaciones web. Ofrece una amplia variedad de librerías para realizar diversas acciones como el enrutado o la comunicación cliente-servidor.
Este framework se basa en componentes, que son los bloques que forman la aplicación. Un componente está formado por tres archivos: el archivo de typescript, donde se desarrolla toda la funcionalidad de la página, el archivo html, donde se diseña la parte visual, y el css, donde se especifica el estilo. Estos dos últimos archivos se especifican en el de typescript mediante el decorador @Component() para unirlos. Este decorador contiene la siguiente información:

- Selector: es un nombre que se le da al componente para poder usarlo en otros, generalmente empieza por “app-” seguido del nombre separado por guiones. Con esto, se puede incrustar el componente dentro del html de otro escribiendo el nombre entre “<>”. Por ejemplo: “<app-mi-componente></app-mi-componente>”
- TemplateUrl: hace referencia al fichero html del componente, donde se indica lo que se va a mostrar en la página.
- StylesUrls: con esta opción se enlazan, opcionalmente, los ficheros de estilos que modificarán la apariencia del html. Lo habitual es usar un fichero propio del componente, pero se pueden añadir otros.

Desde el html se pueden llamar a las variables definidas en el código mediante el uso de llaves dobles, por ejemplo: “<p> numero<p>”. Cada vez que esta variable cambie de valor, se actualizará en el html. También se pueden cambiar las propiedades de los elementos del html desde el código. Por ejemplo, se podría habilitar o deshabilitar un botón según una variable booleana definida en el código de la siguiente forma:

```
<button type="button" [disabled]="canClick"> boton < /button >
```

En el código se definiría la variable canClick y se igualaría a “true” o “false”.

También se puede cambiar el comportamiento de los elementos usando directivas. Las más comunes son “*ngIf” y “*ngFor”. Por ejemplo, se podría cambiar el botón anterior para que directamente no aparezca cuando la variable “canClick” está a false de la siguiente manera:

```
<button type="button" *ngIf="canClick"> boton </button >
```

La forma más fácil de usar angular es instalando la interfaz de línea de comandos Angular CLI. Esta herramienta es de NodeJs, por lo que habrá que tener este entorno instalado previamente. Una vez hecho esto, para instalar Angular CLI solo habría que escribir en una terminal el comando “npm install -g @angular/cli”. Los comandos más utilizados son:

- ng build: compila la aplicación.

- ng serve: compila y ejecuta la aplicación
- ng generate: crea o modifica archivos.
- ng new: crea la estructura básica de un proyecto de angular con una página de prueba.

Para crear un nuevo componente se usa el comando: “ng generate component miComponente”. Esto crea una carpeta llamada mi-componente con los archivos especificados antes. El componente principal es el app.component, que se crea automáticamente al ejecutar el escribir “ng new” y es al que se accede al ejecutar la aplicación. Para acceder a otros se usa el app-routing.module. En él, se escriben los componentes a los que se quiere acceder y sus rutas correspondientes, para poder llamarlas desde el html a través de la directiva [routerLink] o bien desde el código con lel método router.bavigateByUrl(). Para lo segundo es necesario importar la librería de la siguiente manera: ‘import { Router } from “@angular/router”’ y crear la propiedad router de la clase Router en el constructor.

En este proyecto se ha usado angular para la creación de una aplicación web donde poder ver las estadísticas del usuario. Para ello, se ha usado como plantilla el repositorio de GitHub de cmurestudillos, “graficos-angular”[29] con licencia MIT, que se puede leer en el anexo 8. Este repositorio ofrece ejemplos de gráficos de línea, barras, donuts y radas usando material, charts.js y ng2-charts. En este caso se usarán sólo el gráfico de líneas para mostrar los resultados de los test y del registro de ánimo y el de barras para mostrar cuántas veces se han realizado los consejos. En el caso del gráfico de le líneas (figura 3.6), a la derecha aparecerá un botón para mostrar tres opciones: que aparezcan los datos de los últimos 7 de días, los últimos 15 días o el último mes. Por defecto, aparecerá la primera opción. Además, en la leyenda del gráfico se podrán marcar y desmarcar los datos para elegir cuáles aparecen. En el caso del gráfico de barras(figura 3.7), aparecerá en la leyenda las opciones de los días, que también se podrán marcar y desmarcar. Al entrar a la aplicación, aparecerá un modal para introducir el usuario del que se quieren obtener los datos. En la esquina superior derecha aparecerá un botón con el nombre del usuario registrado, que se podrá cambiar la pulsarlo.

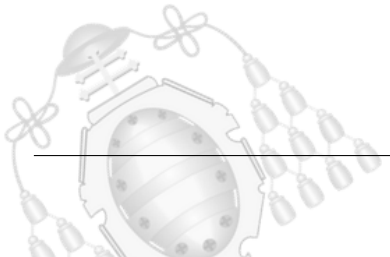
Figura 3.6.- Aplicación web: Gráfico de línea (Fuente: propia)

Figura 3.7.- Aplicación web: Gráfico de barras (Fuente: propia)

4. Planificación

En esta sección se verán las tareas en las que se desglosa el proyecto y el tiempo dedicado a cada una de ellas. El inicio del desarrollo del proyecto se ha establecido el día 11 de abril de 2022. Su finalización estimada tras el cálculo de la planificación es el día 15 de junio de 2023.

1. Análisis de la situación actual: se establecen los objetivos y se analiza la situación de partida, así como las necesidades para la realización del proyecto.
2. Definición de requisitos: se definen los requisitos, tanto de usuario como de sistema, que debe cumplir la aplicación.
3. Diseño: se diseña la arquitectura del sistema, se define el entorno de desarrollo y se estudian las soluciones planteadas, tanto de hardware como de software.
4. Descripción de sistema: se elabora el diagrama de casos de uso, la descripción de casos de uso, el diagrama de secuencia y la base de datos.
5. Desarrollo: se implementa la Skill, tanto la parte frontend como la parte backend.
6. Pruebas: se despliega la realización de pruebas para asegurar el correcto funcionamiento de la Skill.
7. Elaboración de manuales: elaboración de los distintos manuales para una información correcta acerca del uso de la aplicación.
8. Documentación: recogida de información y documentación de las actividades realizadas durante todo el proyecto.



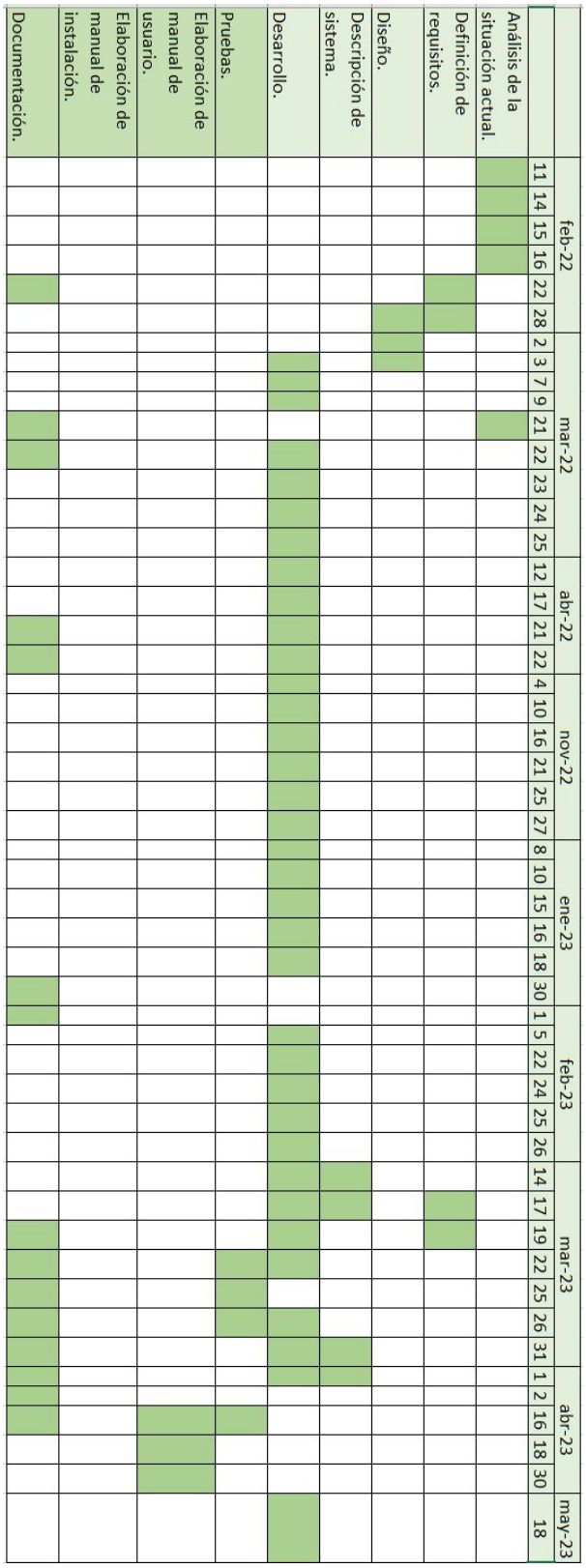


Figura 4.1.- Diagrama de la planificación (Fuente: elaboración propia)

5. Documentos Técnicos

5.1.- Requisitos

En este apartado se hará un análisis de los requisitos del sistema, tanto los funcionales como los no funcionales.

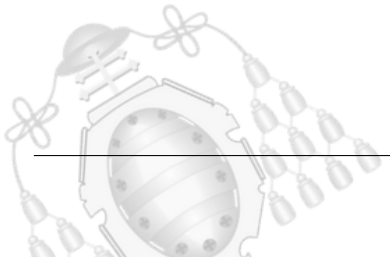
Cada requisito estará definido por:

- Identificador: es un valor único del requisito, su estructura es RFXX si es funcional o RNFXX si es no funcional y en donde XX indica el número de secuencia del requisito.
- Nombre: define la denominación que se le da al requisito en cuestión.
- Descripción: breve definición del requisito.
- Prioridad: puede tener 3 niveles distintos, A (Alta) más prioritaria, M (Media) prioridad media y B (Baja).

5.1.1.- Requisitos funcionales

ID	Nombre	Descripción	Prioridad
RF1	Inicio de la aplicación.	El sistema se iniciará mediante el comando de voz “salud mental”.	A
RF2	Solicitar ayuda.	El sistema mostrará todas las opciones disponibles para el usuario al usar el comando de voz “ayuda”.	A
RF3	Registrar ánimo.	Al iniciar la skill, se pedirá al usuario que indique en una escala del 1 al 10 su estado de ánimo.	A

Continúa en la siguiente página



Cuadro 5.1

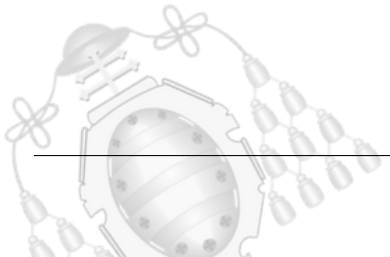
Id	Nombre	Descripción	Prioridad
RF.	Preguntas.	Tras registrar el animo, se realizarán una serie de preguntas y consejos para mejorarlo.	A
RF5	Test GDS.	El sistema permitirá realizar el test de depresión geriátrica de Yesavage.	M
RF6	Test Beck.	El sistema permitirá realizar el test de depresión de Beck.	M
RF6	Test Rumiación.	El sistema permitirá realizar un test para medir el estado rumiativo del usuario.	M
RF7	Reiniciar.	El sistema permitirá reiniciar el test que se esté realizando.	B
RF8	Recordatorios.	El sistema permitirá crear recordatorios diarios para que el usuario no olvide registrar su ánimo.	A
RF9	Música.	El sistema permitirá escuchar música relajante mediante el comando de voz "música", o propondrá escucharla en la batería de preguntas inicial.	M
RF10	Lista de asociaciones.	El sistema permitirá escuchar listas de asociaciones amplias mediante el comando de voz "listas", o propondrá escucharla en la batería de preguntas inicial.	M
RF11	Salir de la skill.	El sistema permitirá salir de la skill con el comando de voz "para".	A

Continúa en la siguiente página

Cuadro 5.1

Id	Nombre	Descripción	Prioridad
RF12	Registrar edad.	El sistema preguntará por la edad del usuario al iniciarse para poder usar el test que más se ajuste.	A
RF13	Cuestionario de depresión infantil.	El sistema permitirá realizar el cuestionario de depresión infantil (CDI).	M
RF14	Chistes.	El sistema permitirá escuchar un chiste, o propondrá escucharlo en la batería de preguntas inicial.	B
RF15	Frases motivacionales.	El sistema permitirá escuchar frases motivacionales, o propondrá escucharla en la batería de preguntas inicial.	B
RF16	Teléfono de la esperanza.	Si el usuario registra un ánimo muy bajo, el sistema le recomendará llamar al teléfono de la esperanza o la linea de atención a la conducta suicida.	B
RF17	Seguimiento.	Se podrá ver un seguimiento del usuario desde una aplicación web.	A

Cuadro 5.1.- Requisitos funcionales



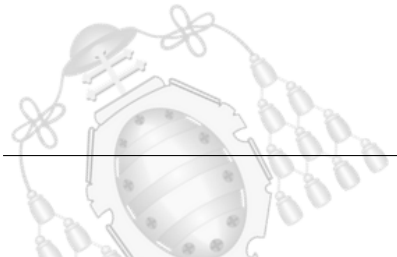
5.1.2.- Requisitos no funcionales

ID	Nombre	Descripción	Prioridad
RFN1	Rendimiento.	El sistema debe ser capaz de responder en menos de cinco segundos.	M
RFN2	Capacidad.	El sistema deberá ser capaz de dar soporte a cientos de usuarios simultáneamente.	M
RFN3	Fiabilidad.	El sistema debe ser robusto y estar disponible en todo momento. No debería caerse o desconectarse sin motivo aparente.	A
RFN4	Conexión a internet.	El dispositivo debe tener conexión a internet.	A
RFN5	Usabilidad.	El sistema debe ser fácil de usar e intuitiva para los usuarios. Debe ofrecer una experiencia de usuario coherente y satisfactoria.	M
RFN6	Lenguaje de programación.	El lenguaje de programación será node.js.	M
RFN7	Diseño de la interfaz.	La skill debe poder adaptarse a cualquier dispositivo de Alexa, tenga pantalla o no.	M

Cuadro 5.2.- Requisitos no funcionales

5.2.- Diseño

En esta sección se explicará el funcionamiento global del sistema y cada uno de los elementos que lo conforman



5.2.1.- Arquitectura del sistema

La arquitectura de una skill de Alexa cuenta con dos partes 2 partes:

- Frontend: el Servicios de Voz de Alexa se encarga de reconocer las palabras del usuario e identificar a qué *intent* está llamando y con qué *slots*, en caso de haberlos. Una vez identificados, se envía la información al backend, el cual procesará la información y devolverá un texto que se pasará a voz en ese servicio.
- Backend: este recibe qué *intent* se ha activado y ejecuta el código que le corresponde, devolviendo el texto que dirá Alexa y/o imágenes o sonidos, si el dispositivo cuenta con pantalla.

La siguiente figura 5.1, del blog "Piensa en software, desarrolla en colores"[30], representa esta arquitectura:

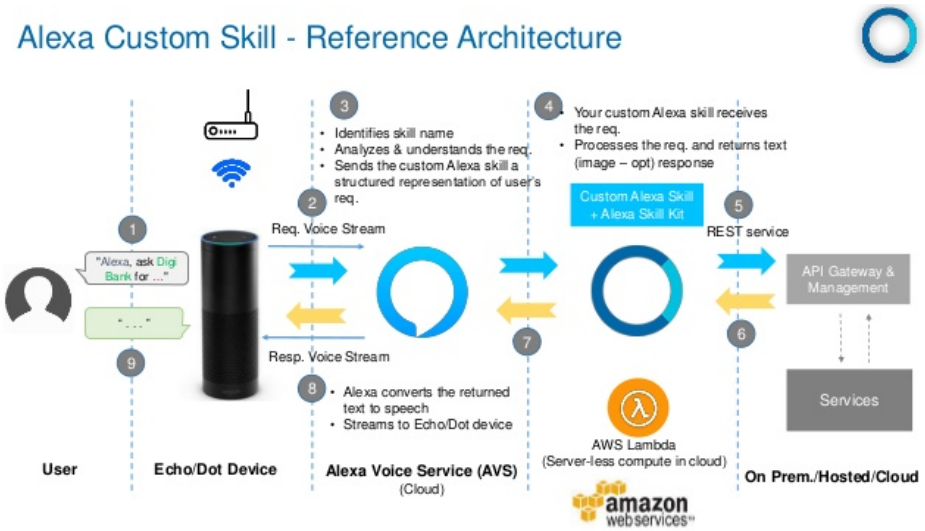


Figura 5.1.- Arquitectura de una skill de Alexa. Fuente: Piensa en software, desarrolla en colores

5.2.2.- Entorno de desarrollo

La skill se ha desarrollado usando al consola de desarrollo de Amazon. Esta cuenta con una interfaz que permite configurar el front-end de forma sencilla sin necesidad de

escribir código (figura 3.2). Dentro de la pestaña build, a la izquierda, aparecen varios desplegables que permiten configurar sus características (figura 5.2).

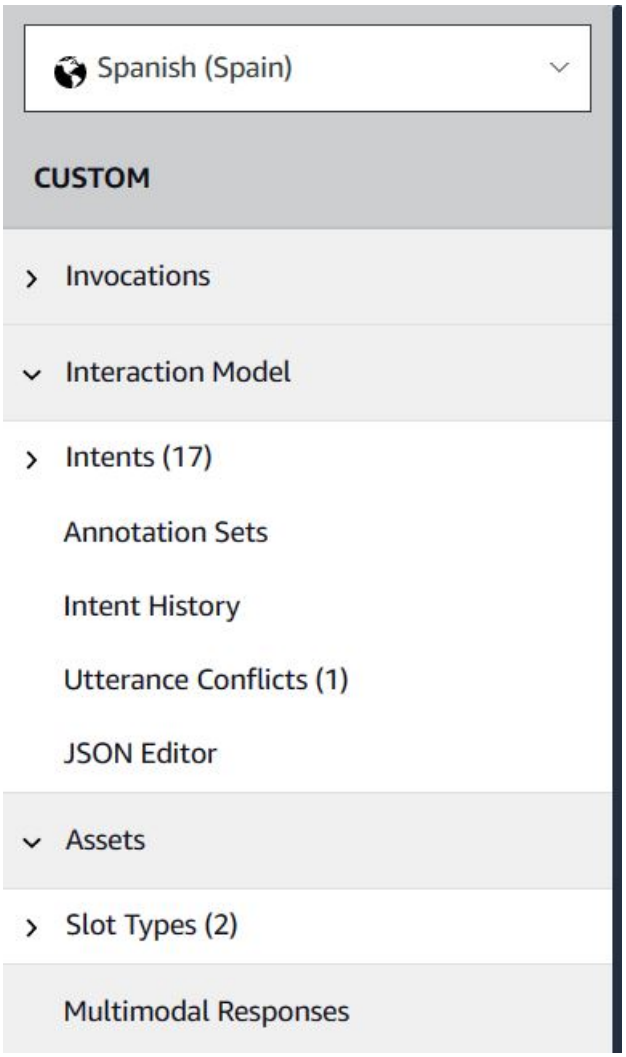


Figura 5.2.- Consola de de desarrollo Alexa Skills, desplegables de la pestaña build (Fuente: propia)

- La opción "Invocations" permite elegir la frase con la que se abrirá la skill.
- Dentro de "Interaction Model" se pueden elegir los *intents* con los que contará la skill. Como ya se ha explicado en el apartado 3.1, los *intents* son las acciones que se pueden realizar dentro de la skill y se activan mediante comandos de voz llamados "sample utterances". Estos pueden crearse personalizados o usar los preconfigurados por Amazon, llamados "Built-In Intents". En este proyecto, se han usado 10 *intents* personalizados y 7 preconfigurados (figura 5.3). Los *sample utterances* de los *intents* pueden contener información que se guardará en los

llamados *slots* para procesarla posteriormente en el código. En este caso, se cuenta con tres *intents* con *slots* (figura 5.4).

- Como se ha explicado en el apartado 3.1, los *slots* son variables con la información que puede proporcionar un usuario al activar un *intent* y, como tal, pueden ser de diferentes tipos. Al igual que los *intents*, estos pueden ser preconfigurados o se pueden personalizar. Los que se usen, aparecen dentro del desplegable *Slot Types*. En este caso, se han usado dos tipos preconfigurados (figura 5.5): AMAZON.NUMBER, para los *slots* "numero" y "anios", y AMAZON.TIME para el *slot* "recordatorio". Como ya se explicó antes, se puede usar el modo de diálogo "auto delegation" para que Alexa compruebe si se han dado los datos necesarios, si se han registrado correctamente o si el dato proporcionado es válido.

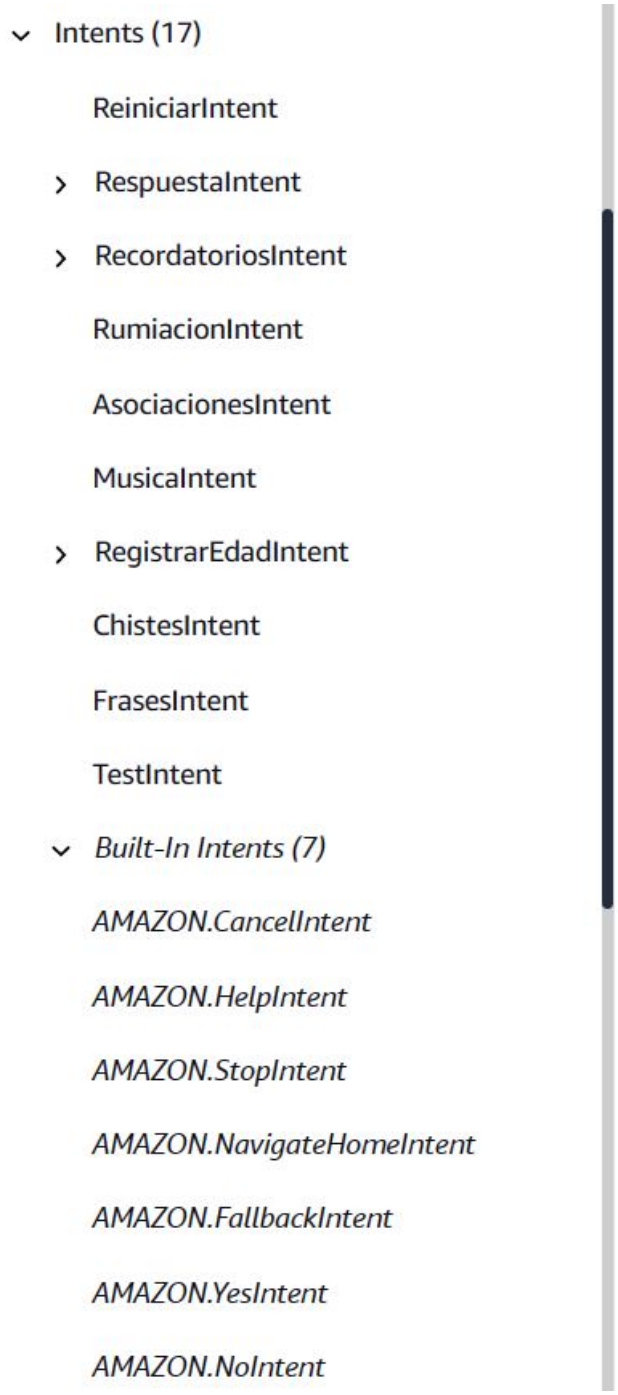


Figura 5.3.- Consola de de desarrollo Alexa Skills, *intents* (Fuente: propia)

En el caso del back-end (figura 5.6), la interfaz es bastante sencilla. A la izquierda se muestra la estructura de archivos, donde se pueden crear nuevos archivos y carpetas, y a la derecha se abren los archivos que se elijan y se escribe el código. También muestra una barra de herramientas justo encima con botones que permiten abrir recursos como

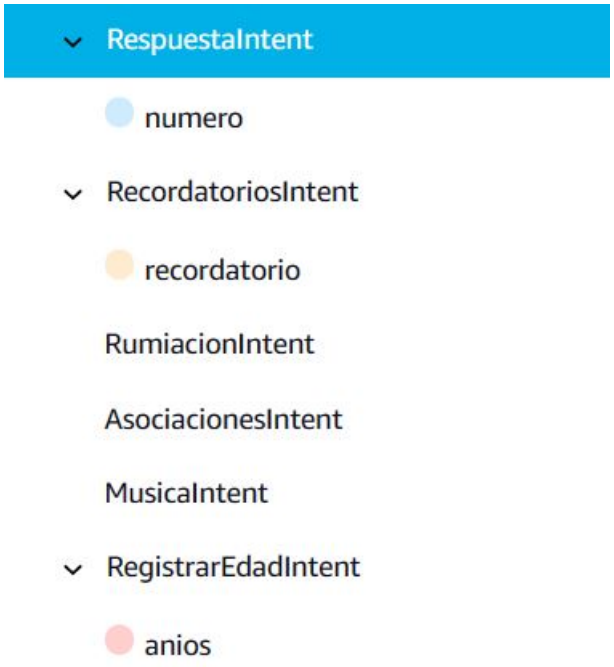


Figura 5.4.- Consola de de desarrollo Alexa Skills, *intents* con *slots* (Fuente: propia)

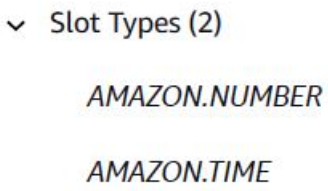


Figura 5.5.- Consola de de desarrollo Alexa Skills, tipos de *slots* (Fuente: propia)

la base de datos de Amazon DynamoDB o el CloudWatch, donde se almacenan los logs.

Esta consola también cuenta con una herramienta de testeo para probar la skill de forma similar a la que funcionaría en un dispositivo de Alexa. Se puede ver un ejemplo en la figura 5.7

5.2.3.- Hardware

Para el desarrollo de este proyecto se ha utilizado un ordenador Lenovo ideapad 520, que cuenta con un procesador Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz, una RAM instalada de 12,0 GB (11,9 GB usable) y un sistema operativo de 64 bits, procesador basado en x64, con Windows 10 Home.

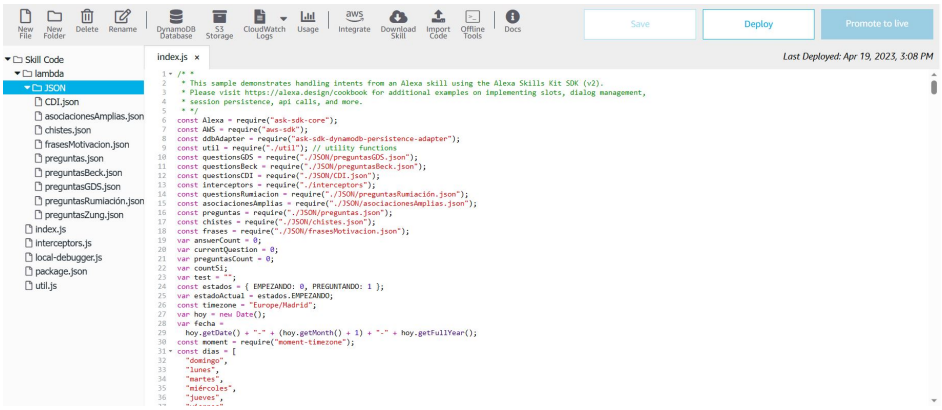


Figura 5.6.- Consola de de desarrollo Alexa Skills, pestaña “Code” (Fuente: propia)

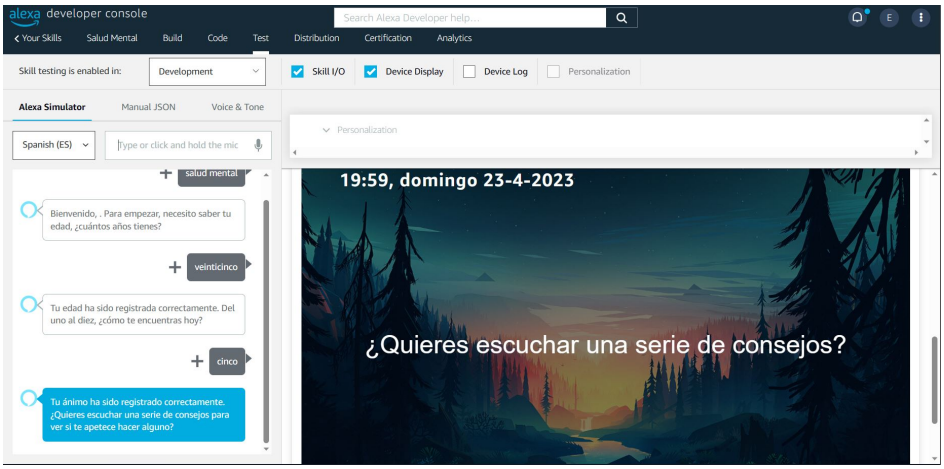


Figura 5.7.- Consola de de desarrollo Alexa Skills, pestaña “Test” (Fuente: propia)

5.2.4.- Software

5.2.4.1.- Alexa Skills Kit

Para el desarrollo de la Skill se ha usado el conjunto de herramientas y servicios proporcionado por Amazon, Alexa Skills Kit. Como se ha visto en el apartado 3.1, este conjunto de herramientas incluye diversos recursos para desarrolladores, como tutoriales, documentación y plantillas, además de ofrecer una consola donde se realiza el desarrollo de la skill, descrita en profundidad en el apartado 5.2.2. Para aprender el funcionamiento de esta tecnología se ha contado con la serie de tutoriales “zero to hero”[31], disponibles en youtube.

5.2.4.2.- Git

Git[32] es una herramienta de software libre ampliamente utilizada, diseñada para mantener un control de versiones distribuido. Para usarlo, se puede descargar desde la propia página, está disponible para macOS, Linux y Windows. Se puede usar por línea de comandos o usando alguna interfaz gráfica, las cuales se encuentran también la página oficial.

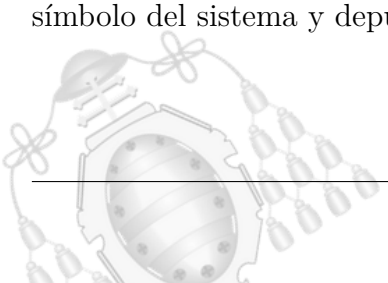
Git cuenta con un sistema basado en ramas que permite realizar cambios sin sobrescribir otras versiones y unirlas posteriormente. Cada rama es totalmente independiente y se puede crear partiendo de cualquier otra rama que se desee escribiendo el comando "git checkout -b <nombreRamaNueva>". Este comando crea una nueva rama desde la que se encuentra actualmente y se cambia a ella. Para mezclar dos ramas, se entra en una y se escribe el nombre de la otra tras el comando "git merge". Cada cambio realizado se debe confirmar con el comando "git commit -m <mensaje>", donde "mensaje" será una breve descripción de los cambios realizados.

En este proyecto se ha usado git para mantener un control de versiones de la skill gracias a la herramienta Alexa Skills (ASK) Toolkit for Visual Studio Code.

5.2.4.3.- Visual Studio Code

Visual Studio Code, también llamado VS Code, es un editor de código desarrollado por Microsoft y de código abierto. Ofrece diversas herramientas como un depurador de código, resaltado de sintaxis, auto-completado de código y git integrado. Además, se le pueden añadir multitud de extensiones. Está disponible para Windows, macOS and Linux y se puede descargar de la página oficial. Está orientado principalmente a JavaScript, TypeScript y Node.js, pero también se puede usar con otros lenguajes.

También incluye una terminal para poder ejecutar el código o realizar otras acciones sin tener que cambiar de ventana, y deja elegir entre PoweShell (por defecto), Git Bash, símbolo del sistema y depurador de JavaScript.



En este proyecto se ha usado VS Code junto con node.js y angular para realizar la aplicación que mostrará el seguimiento del usuario.

5.2.4.4.- OverLeaf

Para escribir la memoria, se ha usado OverLeaf[33], un editor de LaTeX en línea que permite redactar documentos de forma colaborativa e implementa un control de versiones. Además, ofrece multitud de plantillas para empezar un proyecto de forma sencilla y una documentación[34] detalla para aprender a usar esta herramienta. Para poder usarla solo hace falta crearse una cuenta. Se pueden crear proyectos ilimitados y buscar plantillas para partir de la que mejor se adapte al tema o usar una de ejemplo.

Cuenta con un sistema de ventanas de forma que se puede ver el PDF al lado del código e ir compilándolo según se escribe, aunque se puede modificar para ver sólo una de las ventanas o que aparezcan en pestañas separadas. A la izquierda, aparecen todos los archivos del proyecto y el esquema de secciones del documento abierto para navegar por él fácilmente. Además, cuenta con unos botones para mostrar en el PDF la localización actual del código y vice-versa. En la configuración se puede elegir entre distintos compiladores, cambiar el documento principal, el idioma para el corrector ortográfico, añadir ayudas de auto-completado y cambiar entre varias opciones de visualización.

5.2.4.5.- Node.js

Como ya se ha explicado en el apartado 3.2, Node.js es un conjunto de herramientas que permite ejecutar código JavaScript, lenguaje que inicialmente era exclusivamente del lado del cliente, es decir, solo se podía ejecutar en un navegador web, pero, gracias a Node.js, ahora se puede ejecutar fuera de estos. Este entorno usa un modelo de operaciones asíncrono y orientado a eventos, es decir, el procesamiento de una tarea no va a bloquear o retrasar la ejecución de otra. Además, cuenta con su propio gestor de paquetes, llamado npm (Node Package Manager). Mediante ese, se pueden gestionar fácilmente las dependencias de un proyecto, además de descargar librerías o programas

de terceros. Node.js tiene muchos usos, pero es especialmente útil para aplicaciones en tiempo real.

En este proyecto, se usa para la creación de la skill de la Alexa y para la aplicación web del seguimiento del usuario. Para esta última, se ha usado la versión 14.15.3, instalada a través de la herramienta nvm que, como ya se vio antes, es un administrador de versiones de Node.js.

5.2.4.6.- AWS Lambda

AWS Lambda es un servicio informático proporcionado por Amazon que permite ejecutar código sin que el desarrollador se preocupe por los servidores, ya que los gestiona el proveedor del servicio. Es decir, el desarrollador simplemente se encarga de escribir el código, sin necesidad de planificar las características del servidor como la capacidad, la configuración, el mantenimiento o la tolerancia a fallos. Además, no se paga por la infraestructura, solo por el tiempo de informática que se utiliza.

5.2.4.7.- PlantUML

PlantUML[35] es una herramienta en línea de código abierto que permite crear diagramas de forma rápida y sencilla a partir de texto plano. Ofrece diversos tipos de diagramas y varios ejemplos en cada uno para aprender la sintaxis, además de una guía. Según se escribe, se va creando el diagrama en tiempo real y, una vez finalizado, permite descargarlo con diferentes extensiones. En este caso, se ha usado la herramienta para crear el diagrama de casos de uso y los diagramas de secuencia de los más importantes, descargándolo con la extensión png.

5.2.4.8.- MySQL

MySQL[36] es un sistema de gestión de base de datos de código abierto. Como ya se mencionó en el apartado 3.4, es uno de los más populares y es la base de datos de código abierto más utilizada. También tiene una licencia comercial gestionada por oracle. Se utiliza en el desarrollo web para dar forma y facilitar la comunicación

entre web y servidores. Una de sus principales características es que trabajo con bases de datos relacionales, es decir, utiliza tablas múltiples que se interconectan entre sí para almacenar la información y organizarla correctamente. Existen muchas interfaces gráficas para MySQL, una de las más populares es MySQL Workbench. Permite administrar gráficamente las bases de datos y diseñar visualmente sus estructuras. Esta herramienta reemplaza el anterior conjunto de software MySQL Tools.

En este proyecto, se ha usado MySQL para guardar la información proporcionada por el usuario desde la aplicación de Alexa y acceder a ella desde la aplicación web.

5.2.4.9.- Otro software empleado

El sistema operativo del ordenador en el que se ha desarrollado todo el proyecto es Windows 10.

5.3.- Descripción del sistema

5.3.1.- Casos de uso

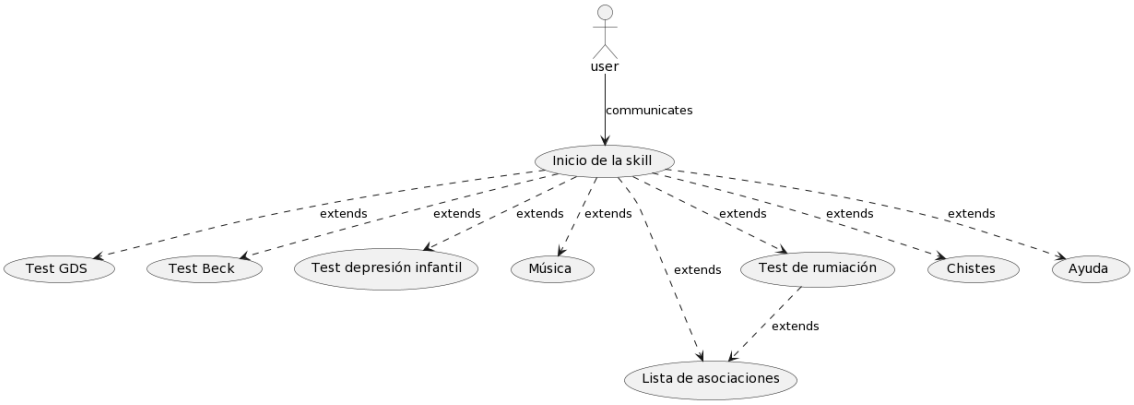


Figura 5.8.- Diagrama de casos de uso skill. (Fuente: Elaboración propia)

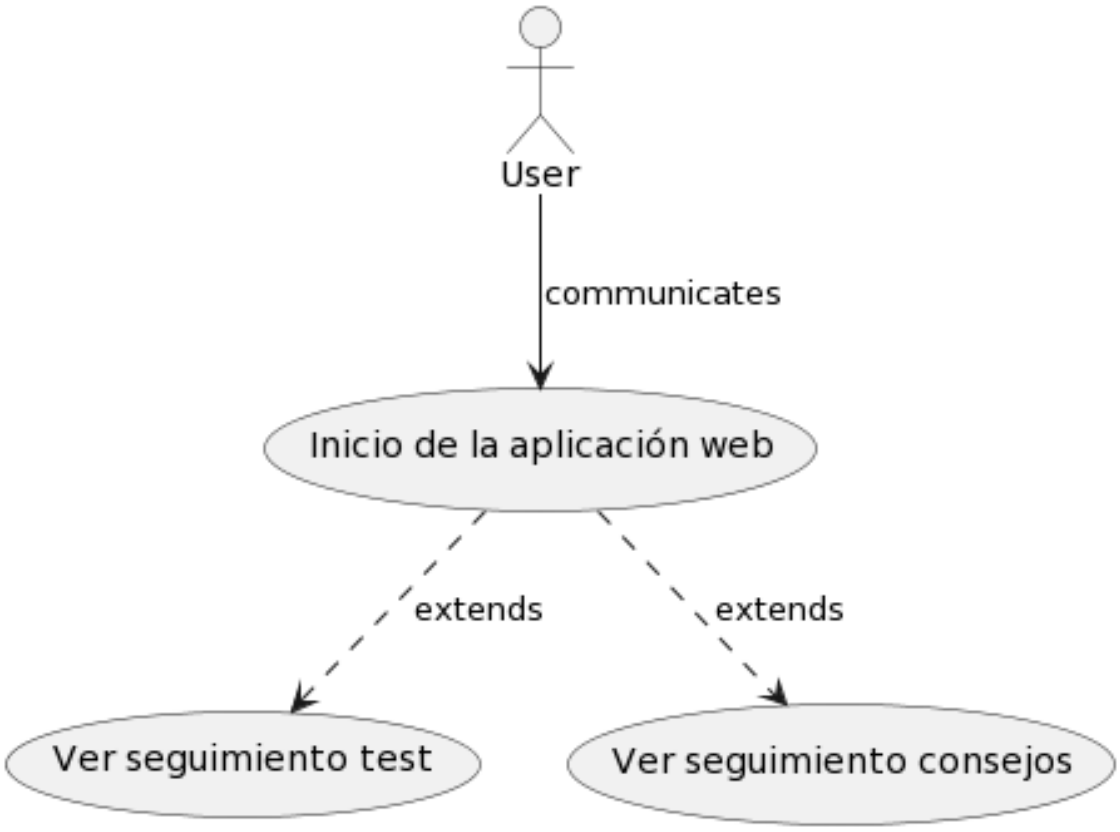


Figura 5.9.- Diagrama de casos de uso aplicación web. (Fuente: Elaboración propia)

En este apartado se van a definir los casos de uso correspondientes con los requisitos funcionales descritos anteriormente. Cada caso cuenta con un identificador de la forma CUXX, donde XX indica el número de secuencia del caso de uso.

CU1	Inicio de la skill.
Descripción	El usuario abre la skill.
Precondiciones	El usuario debe iniciar la skill.
Postcondiciones	El usuario accede a la skill.

Continúa en la siguiente página

Flujo principal.	1. El usuario usa el comando de inicio de skill. 2. Alexa le saluda diciendo la fecha y la hora y le pregunta cómo se encuentra. 3. El usuario contesta con un número del 1 al 10. *a*) Si el usuario contesta "0" o "1", antes de pasar al paso 4, Alexa le recomienda llamar al teléfono de la esperanza o la línea de atención a la conducta suicida. 4. Alexa lo registra y le pregunta si quiere recibir algún consejo. *a*) El usuario contesta que sí. 1) Alexa le propone un consejo. *b*) El usuario contesta que no 1) Alexa contesta con la lista de cosas que puede hacer.
Flujo secundario (primer inicio de la skill)	1. El usuario usa el comando de inicio de skill. 2. Alexa le da la bienvenida y le pregunta su edad. 3. El usuario le dice la edad. 4. Alexa le pide que cree un recordatorio. 5. El usuario abre la configuración de recordatorios. 6. Alexa le pregunta a qué hora quiere crear el recordatorio. 7. El usuario le da una fecha. 8. Se guarda el recordatorio y le pregunta cómo se encuentra. 9. Se inicia el flujo principal.

Continúa en la siguiente página

Flujo secundario	3. el usuario contesta un número fuera de rango. *a*) Alexa le pide que diga un número del 1 al 10. 3. el usuario contesta con algo que no es un número. *a*) Alexa vuelve a preguntarle al usuario cómo se encuentra y espera otra respuesta.
Dependencias	RF1: Inicio de la aplicación, R3: Registrar ánimo, RF4: Preguntas, RF2: Solicitar ayuda, RF16: Teléfono de la esperanza, RF8: Recordatorios, RF11: Registrar edad.

Cuadro 5.3.- CU2: Inicio de la skill

CU2	Salir de la skill.
Descripción	El usuario desea salir de la skill.
Precondiciones	El usuario debe haber iniciado la skill.
Postcondiciones	Se cierra la skill.
Flujo principal	1. El usuario usa el comando de cierre de la skill. 2. Alexa se despide del usuario y cierra la skill.
Dependencias	RF11: Salir de la skill.

Cuadro 5.4.- CU2: Salir de la skill.

CU3	Realizar test GDS.
Descripción	El usuario desea realizar un test para evaluar su depresión.
Precondiciones	El usuario debe haber iniciado la skill y ser mayor de 65 años.

Continúa en la siguiente página

Postcondiciones	Se registra el nivel de depresión del usuario.
Flujo principal	1. El usuario usa el comando de voz “quiero hacer un test”. 2. Alexa le indica al usuario que va a hacerle una serie de preguntas a las que debe contestar con “sí” o “no” y le pregunta si lo ha entendido. 3. El usuario contesta que sí. 4. Empieza a hacerle las preguntas. 5. Tras contestarlas todas, si sale una puntuación alta, le propone un consejo para mejorar el ánimo y, si es baja, le pregunta qué más quiere hacer.
Flujo secundario	3. El usuario contesta que no. 4. Alexa vuelve a indicarle l usuario que va a hacerle una serie de preguntas a las que debe contestar con “sí” o “no” y le pregunta si lo ha entendido. 3. El usuario da una respuesta no contemplada. 4. Alexa se queda a la espera de que conteste “sí” o “no”.
Dependencias	RF5: Test GDS.

Cuadro 5.5.- CU3: Realizar test GDS

CU4	Realizar test de Beck.
Descripción	El usuario desea realizar un test para evaluar su depresión.
	Continúa en la siguiente página

Precondiciones	El usuario debe haber iniciado la skill y tener una edad comprendida entre los 15 y los 65.
Postcondiciones	Se registra el nivel de depresión del usuario.
Flujo principal	1. El usuario usa el comando de voz “quiero hacer un test”. 2. Alexa indica al usuario que conteste 1, 2,3 o 4 a las afirmaciones que le vaya diciendo según lo identificado que se sienta con ellas y empieza con la primera. 3. El usuario contesta con un número entre el 1 y el 4. 4. Sigue diciendo afirmaciones hasta llegar al final. Si sale una puntuación alta, le propone un consejo para mejorar el ánimo y, si es baja, le pregunta qué más quiere hacer.
Flujo secundario	3. El usuario contesta con un número fuera de rango. 4. Alexa le indica que conteste 1, 2, 3 o 4. 3. El usuario da una respuesta no contemplada. 4. Alexa se queda a la espera de que conteste un número entre el 1 y el 4.
Dependencias	RF6: Test de Beck.

Cuadro 5.6.- CU4: Realizar test de Beck.

CU5	Realizar cuestionario de depresión infantil.
Descripción	El usuario desea realizar un test para evaluar su depresión
Precondiciones	El usuario debe haber iniciado la skill y ser menor de 15.
Postcondiciones	Se registra el nivel de depresión del usuario.

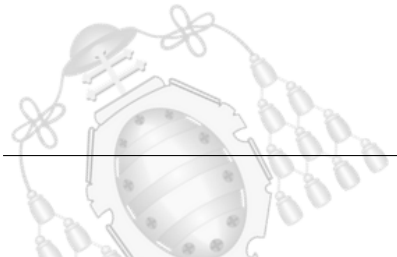
Continúa en la siguiente página

Flujo principal	1. El usuario usa el comando de voz “quiero hacer un test”. 2. Alexa indica al usuario que conteste 1, 2,3 o 4 a las afirmaciones que le vaya diciendo según lo identificado que se sienta con ellas y empieza con la primera. 3. El usuario contesta con un número entre el 1 y el 4. 4. Sigue diciendo afirmaciones hasta llegar al final. Si sale una puntuación alta, le propone un consejo para mejorar el ánimo y, si es baja, le pregunta qué más quiere hacer.
Flujo secundario	3. El usuario contesta con un número fuera de rango. 4. Alexa le indica que conteste 1, 2, 3 o 4. 3. El usuario da una respuesta no contemplada. 4. Alexa se queda a la espera de que conteste un número entre el 1 y el 4.
Dependencias	RF13: Cuestionario de depresión infantil.

Cuadro 5.7.- CU5: Realizar cuestionario de depresión infantil

CU6	Escuchar Música relajante.
Descripción	El usuario desea escuchar música relajante.
Precondiciones	El usuario debe haber iniciado la skill.
Postcondiciones	La skill muestra una imagen y música relajante.

Continúa en la siguiente página

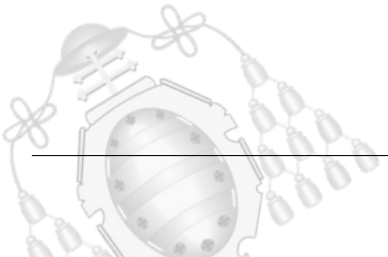


Flujo principal	1. El usuario usa cualquiera de los siguientes comandos de voz: *a*) "Escuchar música". *b*) "Meditar". *c*) "Música relajante". *d*) "Música". 2. Se abre la música.
Dependencias	RF9: Escuchar Música relajante.

Cuadro 5.8.- CU6: Música

CU7	Escuchar lista de asociaciones amplias.
Descripción	El usuario desea escuchar una lista de asociaciones amplias.
Precondiciones	El usuario debe haber iniciado la skill.
Postcondiciones	La skill reproduce una lista.
Flujo principal.	1. El usuario usa cualquiera de los siguientes comandos de voz: *a*) "Escuchar listas". *b*) "Listas". *c*) "Lista". *d*) "Lista de asociaciones amplias". *e*) "Asociaciones amplias". 2. La skill reproduce una lista.

Continúa en la siguiente página

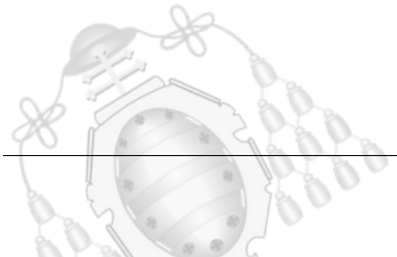


Flujo secundario	1. El usuario realiza el test de rumiación y le sale una puntuación mayor a 33. 2. La skill reproduce una lista.
Dependencias	RF10: Escuchar lista de asociaciones amplias.

Cuadro 5.9.- CU7: Lista de asociaciones amplias

CU8	Realizar test de rumiación.
Descripción	El usuario desea realizar un test para evaluar su estado rumiativo.
Precondiciones	El usuario debe haber iniciado la skill.
Postcondiciones	Se registra el nivel de rumiación del usuario.
Flujo principal	1. El usuario usa cualquiera de los siguientes comandos de voz: *a*) "Rumiación". *b*) "Test rumiación". 2. Alexa indica al usuario que conteste 1, 2,3 o 4 a las afirmaciones que le vaya diciendo según lo identificado que se sienta con ellas y empieza con la primera. 3. El usuario contesta con un número entre el 1 y el 4. 4. Sigue diciendo afirmaciones hasta llegar al final. Si sale una puntuación alta, le propone un consejo para mejorar el ánimo y, si es baja, le pregunta qué más quiere hacer.

Continúa en la siguiente página



Flujo secundario	3. El usuario contesta con un número fuera de rango. 4. Alexa le indica que conteste 1, 2, 3 o 4. 3. El usuario da una respuesta no contemplada. 4. Alexa se queda a la espera de que conteste un número entre el 1 y el 4.
Dependencias	RF6: Test rumiación.

Cuadro 5.10.- CU8: Realizar test de rumiación

CU9	Escuchar un chiste.
Descripción	El usuario desea escuchar un chiste.
Precondiciones	El usuario debe haber iniciado la skill.
Postcondiciones	La skill cuenta un chiste.
Flujo principal	1. El usuario usa cualquiera de los siguientes comandos de voz: *a*) "Cuéntame un chiste". *b*) "Otro chiste". 2. La skill reproduce un chiste.
Dependencias	RF14: Chistes.

Cuadro 5.11.- CU9: Escuchar un chiste

CU10	Escuchar una frase motivacional.
Descripción	El desea escuchar una frase motivacional.
Precondiciones	El usuario debe haber iniciado la skill.

Continúa en la siguiente página

Postcondiciones	La skill reproduce una frase motivacional.
Flujo principal	1. El usuario usa cualquiera de los siguientes comandos de voz: *a*) "Dime una frase motivacional". *b*) "Dime frases motivacionales". *c*) "Otra frase". 2. La skill reproduce una frase motivacional.
Dependencias	RF15: frases motivacionales.

Cuadro 5.12.- CU10: Escuchar una frase motivacional

CU11	Ayuda.
Descripción	El usuario desea abrir la ayuda.
Precondiciones	La skill debe de estar iniciada.
Postcondiciones	La skill reproduce el texto de ayuda.
Flujo principal	1. El usuario pide ayuda. 2. Alexa contesta con la lista de opciones que ofrece.
Dependencias	RF12: solicitar ayuda.

Cuadro 5.13.- CU11: Ayuda

CU12	Inicio de la aplicación web.
Descripción	El usuario desea ver los resultados de su seguimiento.
Precondiciones	La aplicación web debe estar operativa.
Postcondiciones	Se muestra un gráfico con el seguimiento del usuario.

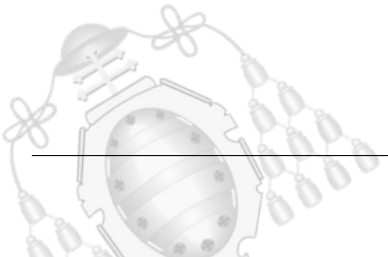
Continúa en la siguiente página

Flujo principal	1. El usuario entra en la aplicación web. 2. Aparece un cartel para introducir el nombre de usuario de Amazon. 3. El usuario introduce su nombre de usuario de Amazon. 4. Se muestra un gráfico con los resultados del registro de ánimo, el test de depresión y el de rumiación de los últimos 7 días.
Dependencias	RF17: Seguimiento.

Cuadro 5.14.- CU12: Inicio de la aplicación web.

CU13	Seguimiento test.
Descripción	El usuario desea ver los resultados de su seguimiento en los test.
Precondiciones	La aplicación web debe estar operativa y el usuario debe haber introducido su nombre.
Postcondiciones	Se muestra un gráfico con el seguimiento del usuario.
Flujo principal	1. El usuario pulsa el botón “mostrar más”. 2. Aparece una tabla que muestra los resultados por fechas y tres botones abajo para ver los resultados de los últimos 7, 15 y 30 días.

Continúa en la siguiente página

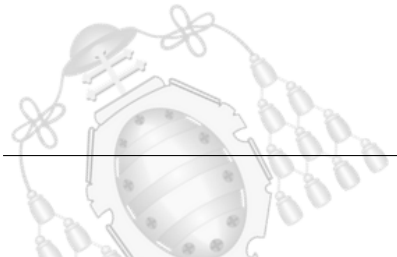


Flujo secundario	1. El usuario pulsa el botón “últimos 15 días”. 2. Se muestra un gráfico con los resultados del registro de ánimo, el test de depresión y el de rumiación de los últimos 15 días. 3. El usuario elige la opción ’‘último mes”. 4. Se muestra un gráfico con los resultados del registro de ánimo, el test de depresión y el de rumiación del último mes. 5. El usuario pulsa “Test rumiación” en la leyenda. 6. Se muestra un gráfico con los resultados del registro de ánimo y el test de depresión del último mes. 7. El usuario pulsa “Test depresión” en la leyenda. 8. Se muestra un gráfico con los resultados del registro de ánimo del último mes.
Dependencias	RF17: Seguimiento.

Cuadro 5.15.- CU13: Seguimiento 15 días

CU14	Seguimiento consejos.
Descripción	El usuario desea ver el seguimiento del los consejos que ha realizado.
Precondiciones	La aplicación web debe estar operativa y el usuario debe haber introducido su nombre.
Postcondiciones	Se muestra un gráfico de barras con los consejos propuestos por Alexa y la frecuencia con la que han sido realizados.

Continúa en la siguiente página

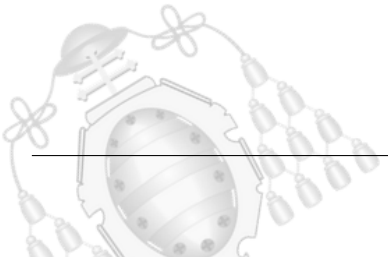


Flujo principal	1. El usuario pulsa “Barras” en el menú de la barra superior. 2. Se muestra un gráfico de barras con los consejos propuestos por Alexa y la frecuencia con la que han sido realizados en los últimos 7 días, 15 días y el último mes.
Flujo secundario	7. El usuario pulsa “7 días” en la leyenda. 8. Se elimina del gráfico las barras con los datos de los últimos 7 días. 9. El usuario pulsa “15 días” en la leyenda. 10. Se elimina del gráfico las barras con los datos de los últimos 15 días. 11. El usuario pulsa“30 días” en la leyenda. 12. Se elimina del gráfico las barras con los datos de los últimos 30 días.
Dependencias	RF18: Seguimiento.

Cuadro 5.16.- CU18: Seguimiento consejos

CU15	Reiniciar test.
Descripción	El usuario ha empezado a realizar un test y desea reiniciarlo.
Precondiciones	El usuario debe haber iniciado la skill de Alexa y haber empezado un test.
Postcondiciones	Se reinicia el test.

Continúa en la siguiente página



Flujo principal	1. El usuario dice el comando de voz “reiniciar”. 2. Alexa contesta que se ha reiniciado el test. Si lo desea, puede volver a empezarlo.
Dependencias	RF7: Reiniciar.

Cuadro 5.17.- CU19: Reiniciar test.

A continuación se muestra la matriz de cobertura de requisitos para comprobar que todos los requisitos funcionales son cubiertos por al menos un caso de uso.

	CU1	CU2	CU3	CU4	CU5	CU6	CU7	CU8	CU9	CU10	CU11	CU12	CU13	CU14	CU15
RF1	X														
RF2	X														
RF3	X														
RF4	X														
RF5			X												
RF6				X				X							
RF7															X
RF8	X														
RF9						X									
RF10							X								
RF11	X	X													
RF12											X				
RF13					X										
RF14									X						
RF15										X					
RF16	X														
RF17												X	X	X	

Figura 5.10.- Matriz de cobertura de requisitos. (Fuente: Elaboración propia)

5.3.1.1.- Diagramas de secuencia

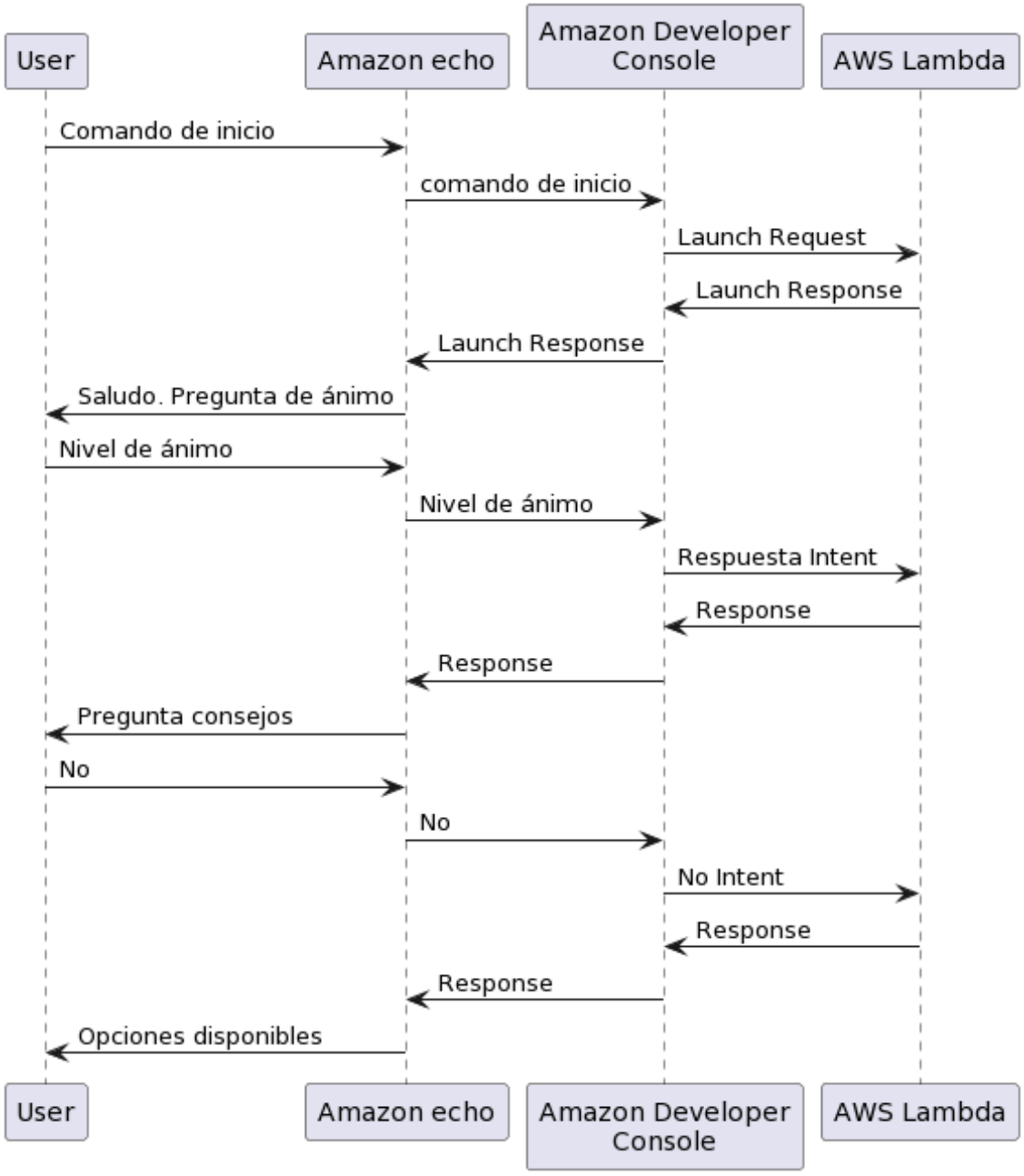


Figura 5.11.- Diagrama de flujo CU1. (Fuente: Elaboración propia)

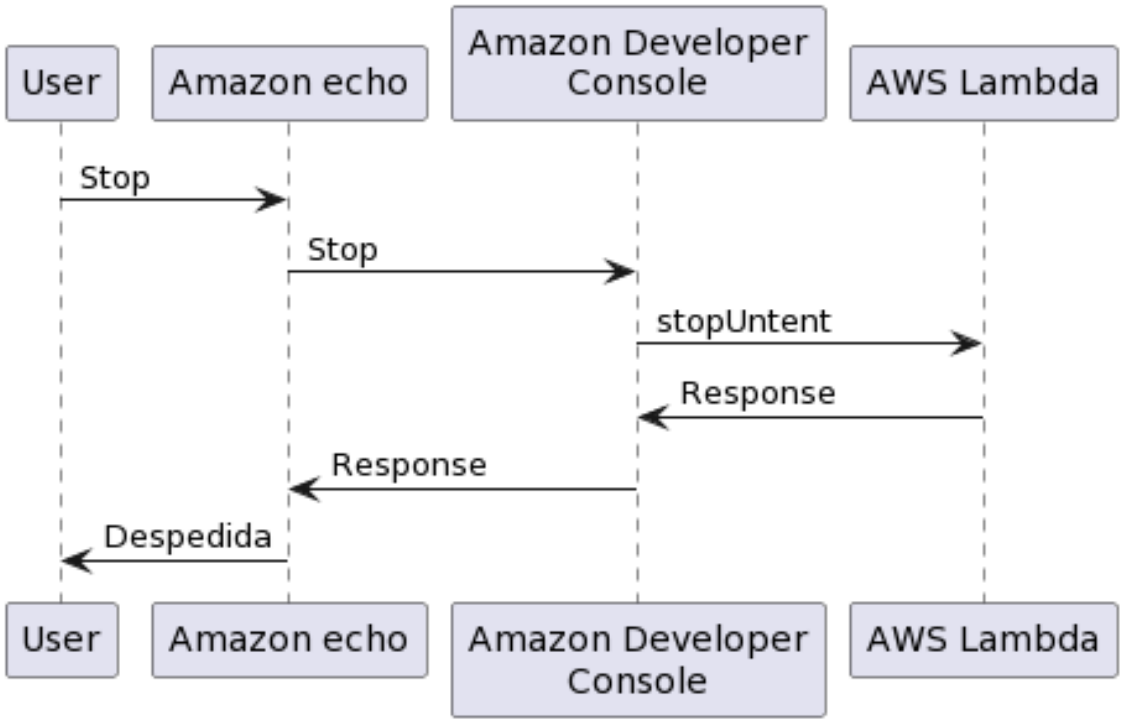


Figura 5.12.- Diagrama de flujo CU2: Salir de la skill. (Fuente: Elaboración propia)

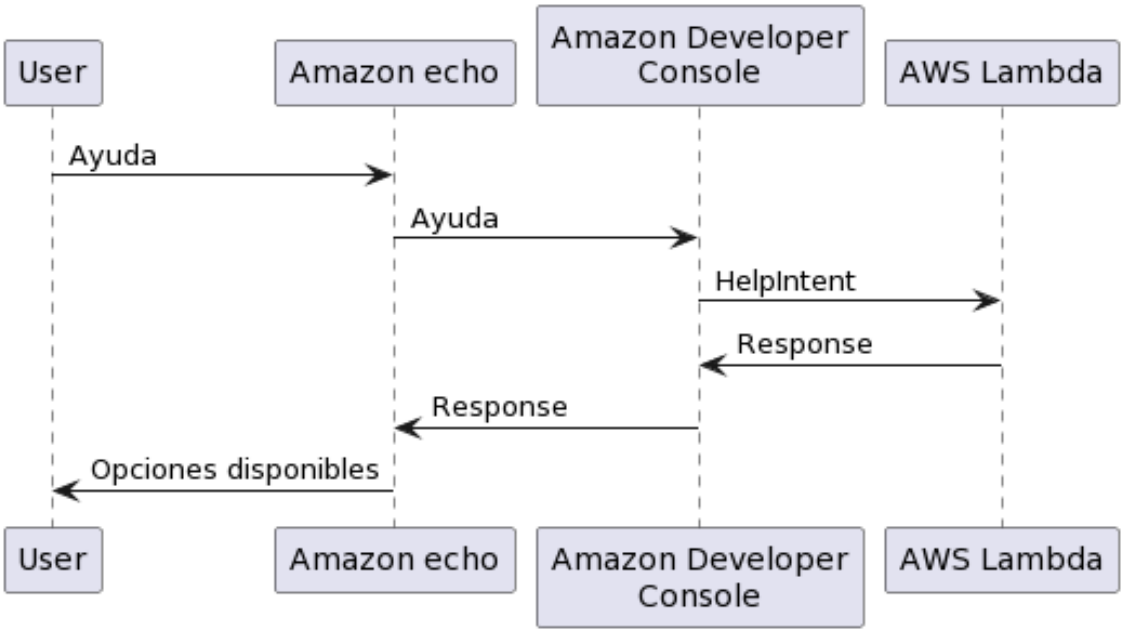


Figura 5.13.- Diagrama de flujo CU11: Ayuda. (Fuente: Elaboración propia)

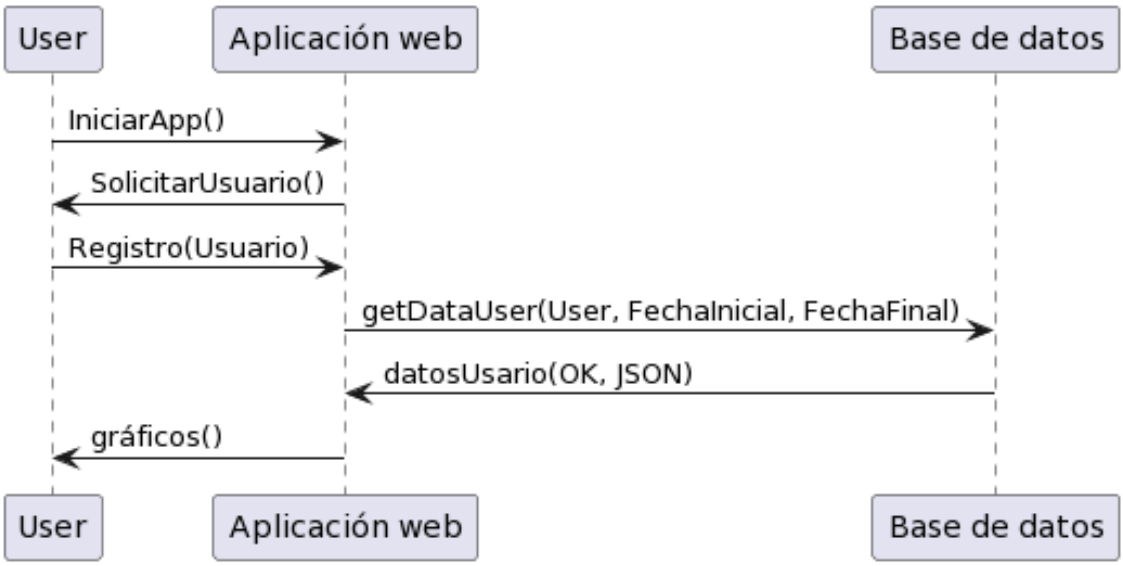


Figura 5.14.- Diagrama de flujo CU12: Iniciaio de aplicación web. (Fuente: Elaboración propia)

5.4.- Casos de prueba

En esta sección se van a ver las pruebas correspondientes a los casos de uso descritos anteriormente, para comprobar que todo funciona como lo esperado. Cada prueba unitaria está definida por:

- Identificador: es un valor único de la prueba, su estructura es PUXX, donde XX indica el número de secuencia.
- Descripción: detalla la prueba a realizar.
- Resultado esperado: lo que se espera obtener al realizar la prueba.
- Resultado obtenido: lo que realmente se obtiene al realizar la prueba.
- Validación: si la prueba ha salido bien o no.
- Correspondencia: el caso de uso asociado a la prueba.

PU1	Primer inicio de la skill.
Descripción	Verificación del correcto inicio de la skill.
Resultado esperado	La skill se abre y pregunta al usuario por su edad.
Resultado obtenido	La skill se abre y pregunta al usuario por su edad.
Validación	Correcta.
Correspondencia	CU1: Inicio skill.

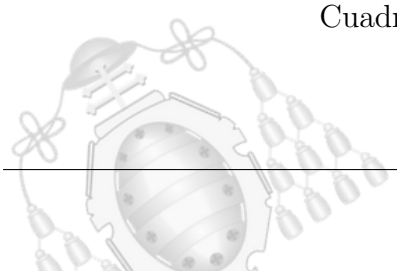
Cuadro 5.18.- PU1: Primer inicio de la skill

PU2	Registro de edad.
Descripción	Verificación de que la skill guarda la edad del usuario.
Resultado esperado	El usuario dice su edad y la skill indica que se ha registrado correctamente.
Resultado obtenido	La skill indica que la edad del usuario se ha guardado correctamente.
Validación	Correcta.
Correspondencia	CU1: Inicio de la skill.

Cuadro 5.19.- PU2: Registro de edad

PU3	Crear recordatorios.
Descripción	Verificación de que la skill guarda correctamente los recordatorios.
Resultado esperado	Tras indicar a la skill la hora en la que se quiere crear el recordatorio, indica que se ha guardado correctamente.
Resultado obtenido	La skill indica que el recordatorio se ha guardado correctamente.
Validación	Correcta.
Correspondencia	CU1: Inicio de la skill.

Cuadro 5.20.- PU3: Crear recordatorios

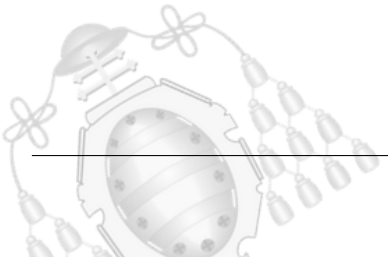


PU4	Inicio de la skill.
Descripción	Verificación del correcto inicio de la skill tras registrar los datos.
Resultado esperado	La skill se abre indicando la fecha y hora actual y le pregunta al usuario cómo se encuentra.
Resultado obtenido	La skill se abre indicando la fecha y hora actual y le pregunta al usuario cómo se encuentra.
Validación	Correcta.
Correspondencia	CU1: Inicio de la skill.

Cuadro 5.21.- PU4: Inicio de la skill

PU5	Registrar ánimo.
Descripción	Verificación de que la skill registra el ánimo indicado por el usuario.
Resultado esperado	La skill indica que ha registrado el ánimo y propone escuchar una lista de consejos.
Resultado obtenido	La skill indica que ha registrado el ánimo y propone escuchar una lista de consejos.
Validación	Correcta.
Correspondencia	CU2: Inicio de la skill.

Cuadro 5.22.- PU5: Registrar ánimo.



PU6	Test GDS.
Descripción	Tras haber registrado una edad mayor de 65 años y pedir hacer un test, la skill abre el test de GDS.
Resultado esperado	Se abre el test de GDS.
Resultado obtenido	Se abre el test de GDS.
Validación	Correcta.
Correspondencia	CU3: Test GDS.

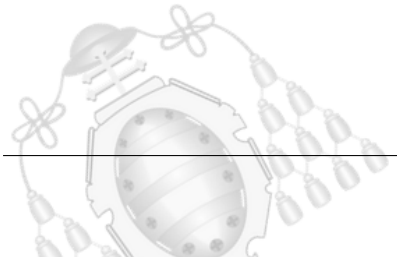
Cuadro 5.23.- PU6: Realizar test GDS

PU7	Test Beck.
Descripción	Tras haber registrado una edad entre 15 y 65 años y pedir hacer un test, la skill abre el test de Beck.
Resultado esperado	Se abre el test de Beck.
Resultado obtenido	Se abre el test de Beck.
Validación	Correcta.
Correspondencia	CU4: Test Beck.

Cuadro 5.24.- PU7: Realizar test de Beck.

PU8	Cuestionario de depresión infantil.
Descripción	Tras haber registrado una edad medor de 15 y pedir hacer un test, la skill abre el test CDI.
Resultado esperado	Se abre el test de CDI.
Resultado obtenido	Se abre el test de CDI.
Validación	Correcta.
Correspondencia	CU5: Realizar cuestionario de depresión infantil.

Cuadro 5.25.- PU8: Test CDI



PU9	Música.
Descripción	Comprobar que la skill reproduce música cuando lo pide el usuario.
Resultado esperado	La skill muestra un APL con una imagen y música relajante.
Resultado obtenido	La skill muestra un APL con una imagen y música relajante.
Validación	Correcta.
Correspondencia.	CU6: Escuchar música relajante.

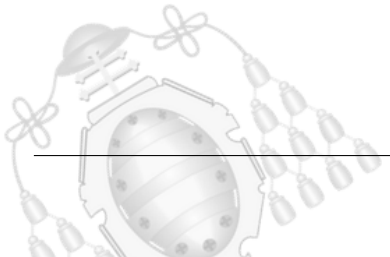
Cuadro 5.26.- PU9: Música

PU10	Lista de asociaciones amplias.
Descripción	Comprobar que la skill reproduce una lista de asociaciones amplias cuando lo pide el usuario.
Resultado esperado	La skill muestra una lista al azar.
Resultado obtenido	La skill muestra una lista al azar.
Validación	Correcta.
Correspondencia	CU7: Lista de asociaciones.

Cuadro 5.27.- PU10: Escuchar lista de asociaciones amplias.

PU11	Test rumiación.
Descripción	Comprobar que la skill abre el test de rumiación cuando lo pide el usuario.
Resultado esperado	La skill abre el test de rumiación.
Resultado obtenido	La skill abre el test de rumiación.
Validación	Correcta.
Correspondencia	CU8: Realizar test rumiación.

Cuadro 5.28.- PU11: Test rumiación



PU12	Chistes.
Descripción	Comprobar que la skill reproduce un chiste cuando el usuario lo pide.
Resultado esperado	La skill cuenta un chiste.
Resultado obtenido	La skill cuenta un chiste.
Validación	Correcta.
Correspondencia	CU9: Escuchar un chiste.

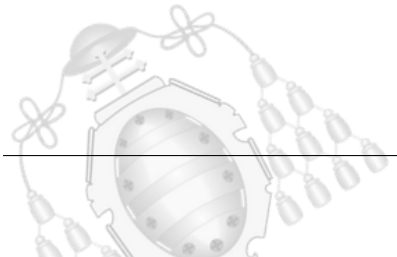
Cuadro 5.29.- PU12: Chistes

PU13	Frases motivacionales.
Descripción	Comprobar que la skill reproduce una frase motivacional cuando el usuario lo pide.
Resultado esperado	La skill dice una frase motivacional.
Resultado obtenido	La skill dice una frase motivacional.
Validación	Correcta.
Correspondencia	CU10: frases motivacionales.

Cuadro 5.30.- PU13: Escuchar una frase motivacional.

PU14	Ayuda.
Descripción	Comprobar que la skill reproduce la ayuda cuando el usuario lo pide.
Resultado esperado	La skill reproduce el texto de ayuda.
Resultado obtenido	La skill reproduce el texto de ayuda.
Validación	Correcta.
Correspondencia	CU11: Ayuda.

Cuadro 5.31.- PU14: Ayuda



PU15	Cierre de la skill.
Descripción	Comprobar que la skill se cierra correctamente.
Resultado esperado	La skill se cierra cuando lo indica el usuario.
Resultado obtenido	La skill se cierra cuando lo indica el usuario.
Validación	Correcta.
Correspondencia	CU2: Salir de la skill.

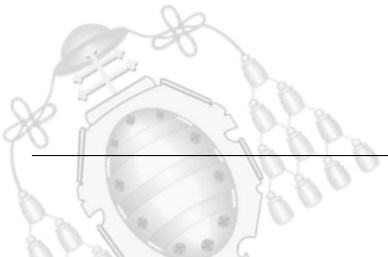
Cuadro 5.32.- PU15: Cierre de la skill

PU16	Intent inexistente.
Descripción	Comprobar que la skill no falla cuando el usuario usa un comando de voz no contemplado.
Resultado esperado	La skill no hace nada.
Resultado obtenido	La skill no hace nada.
Validación	Correcta.

Cuadro 5.33.- PU16: Intent inexistente

PU17	Número fuera de rango.
Descripción	Comprobar que la skill no falla cuando, en los test, el usuario contesta con número fuera de rango.
Resultado esperado	La skill pide al usuario que dé una respuesta válida.
Resultado obtenido	La skill pide al usuario que dé una respuesta válida.
Validación	Correcta.
Correspondencia	CU4: Realizar test de Beck, CU5: Realizar cuestionario de depresión infantil, CU8: Realizar test de rumiación.

Cuadro 5.34.- PU17: Número fuera de rango

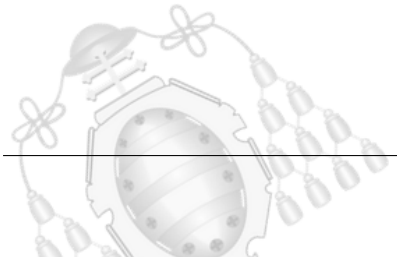


PU18	Inicio de skill con otro nombre.
Descripción	Comprobar que cuando se dice un nombre que no es el de la skill, no se inicia.
Resultado esperado	Alexa indica que no ha entendido lo que ha dicho el usuario y no se inicia la skill.
Resultado obtenido	Alexa indica que no ha entendido lo que ha dicho el usuario y no se inicia la skill.
Validación	Correcta.
Correspondencia	CU1: Inicio de la skill.

Cuadro 5.35.- PU18: Inicio de skill con otro nombre.

PU19	Inicio de la aplicación web.
Descripción	Comprobar que la aplicación web inicia correctamente.
Resultado esperado	La aplicación web muestra un gráfico de líneas con el seguimiento del usuario.
Resultado obtenido	La aplicación web muestra un gráfico de líneas con el seguimiento del usuario.
Validación	Correcta.
Correspondencia	CU12: Inicio de la aplicación web.

Cuadro 5.36.- PU19: Inicio de la aplicación web.

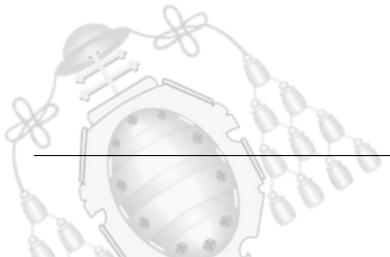


PU20	Seguimiento 15 días.
Descripción	Comprobar que la aplicación web muestra el seguimiento de los últimos 5 días.
Resultado esperado	Al pulsar el botón "últimos 15 días", la aplicación web muestra un gráfico de líneas con el seguimiento del usuario de los últimos 15 días.
Resultado obtenido	La aplicación web muestra un gráfico de líneas con el seguimiento del usuario de los últimos 15 días.
Validación	Correcta.
Correspondencia	CU13: Seguimiento 15 días.

Cuadro 5.37.- PU20: Seguimiento 15 días.

PU21	Seguimiento último mes.
Descripción	Comprobar que la aplicación web muestra el seguimiento del último mes.
Resultado esperado	Al pulsar el botón "último mes", la aplicación web muestra un gráfico de líneas con el seguimiento del usuario del último mes.
Resultado obtenido	La aplicación web muestra un gráfico de líneas con el seguimiento del usuario del último mes.
Validación	Correcta.
Correspondencia	CU14: Seguimiento último mes.

Cuadro 5.38.- PU21: Seguimiento último mes.

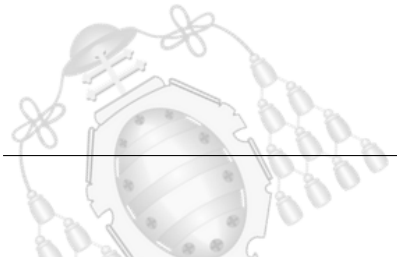


PU22	Seguimiento tests.
Descripción	Comprobar que la aplicación web muestra los resultados del test de rumiación.
Resultado esperado	Al pulsar en la leyenda los datos sobrantes, la aplicación muestra un gráfico únicamente con los datos de interés.
Resultado obtenido	La aplicación web muestra un gráfico únicamente con los datos de interés.
Validación	Correcta.
Correspondencia	CU15: Seguimiento test rumiación, CU16: seguimiento test depresión, CU17: seguimiento nivel ánimo.

Cuadro 5.39.- PU22: Seguimiento tests.

PU23	Seguimiento consejos.
Descripción	Comprobar que la aplicación web muestra el número de veces que ha realizado los consejos propuestos por Alexa.
Resultado esperado	Al elegir la pestaña ”barras”, la aplicación muestra un gráfico de barras con los consejos propuestos por Alexa y el número de veces que se han seguido.
Resultado obtenido	La aplicación web muestra un un gráfico de barras con los consejos propuestos por Alexa y el número de veces que se han seguido.
Validación	Correcta.
Correspondencia	CU18: Seguimiento consejos.

Cuadro 5.40.- PU23: Seguimiento consejos.



PU24	Reiniciar test.
Descripción	Comprobar que los resultados guardados del test en curso se reinician al pedirlo.
Resultado esperado	Tras usar el comando reiniciar test", Alexa indica que el test se reinició correctamente y al volver a abrirlo, empieza por la primera pregunta.
Resultado obtenido	Tras usar el comando reiniciar test", Alexa indica que el test se reinició correctamente y al volver a abrirlo, empieza por la primera pregunta.
Validación	Correcta.
Correspondencia	CU19: Reiniciar test.

Cuadro 5.41.- PU24: Reiniciar test.

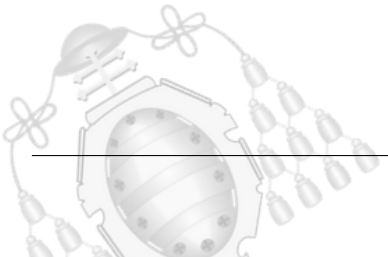
5.5.- Presupuesto

En esta sección se mostrará el presupuesto total del proyecto, divido en: recursos humanos, software, hardware, costes indirectos y beneficio industrial.

5.5.1.- Recursos humanos

Descripción	Cantidad	Unidad	Precio Unitario	Importe
Arquitecto	600	Horas	25,00€	15.000,00€
Analista	600	Horas	25,00€	15.000,00€
Programador	800	Horas	15,00€	12.000,00€
Total				**42.000,00€**

Cuadro 5.42.- Presupuesto Recursos Humanos



5.5.2.- Software

Descripción	Cantidad	Unidad	Precio Unitario	Amortización	Importe
Amazon Skills Kit	1	U	00		0,00€
Windows 10 Home	1	U	145,00€	2 %	2,90€
Visual Studio Code	1	U	0.00€		0,00€
Angular	1	U	0.00€		0,00€
Node.JS	1	U	0.00€		0,00€
Total					**2,90€**

Cuadro 5.43.- Presupuesto Software

5.5.3.- Hardware

Descripción	Cantidad	Unidad	Precio Unitario	Amortización	Importe
Portátil	1	U	749,00€	2 %	14,98€
Amazon Echo show 8	1	U	129,99€	10 %	13,00€
Total					**27,98€**

Cuadro 5.44.- Presupuesto Hardware

5.5.4.- Costes indirectos y Beneficio industrial

Se calculan los gastos indirectos como el 13 % de la suma de los gastos de software, hardware y recursos humanos y el beneficio industrial como el 6 % de ese total.

5.5.5.- Resumen del presupuesto

Categoría	Importe
Recursos humanos	42.000,00€
Software	2,90€
Hardware	27,99€
Costes indirectos (13 %)	5.464,02€
Beneficio industrial (6 %)	2.847,84€
Subtotal	50.342,75€
IVA (21 %)	10.571,98€
Total	**60.914,73€**

Cuadro 5.45.- Presupuesto Total

El presupuesto total asciende a la cuantía de SESENTA MIL NOVECIENTOS CATORCE EUROS CATORCE CÉNTIMOS #60.914,73€#.

6. Manual de usuario

6.1.- Requisitos mínimos

1. Tener un dispositivo que permita usar la tecnología de Alexa, como podría ser un altavoz Echo, o instalar la aplicación en un dispositivo con Android o iOS.
2. Conexión a Internet activa y estable.
3. Tener una cuenta de Amazon.
4. Configuración: para esta skill en concreto, hay configurar los permisos para que pueda acceder al nombre del usuario y establecer recordatorios.

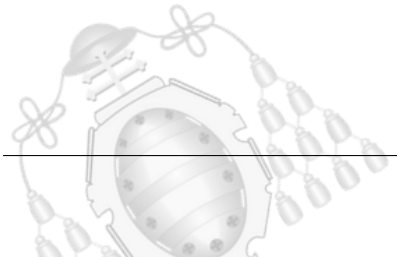
6.2.- Skill de Alexa

6.2.1.- Configuración

Al instalar la skill, esta pedirá permisos para acceder al nombre y los recordatorios. Se deben aceptar por lo menos los segundos, ya que es importante establecer recordatorios para registrar el ánimo todos los días y, si no se hace, la skill pedirá que se establezcan siempre que se inicie. El nombre es opcional, si no se aceptan los permisos, simplemente la skill no dirá el nombre del usuario.

6.2.2.- Primer inicio

Cuando se abra por primera vez, la skill preguntará al usuario por su edad para seleccionar más tarde el test que mejor se ajuste y le pedirá que establezca un recordatorio para que no se olvide de registrar su ánimo.



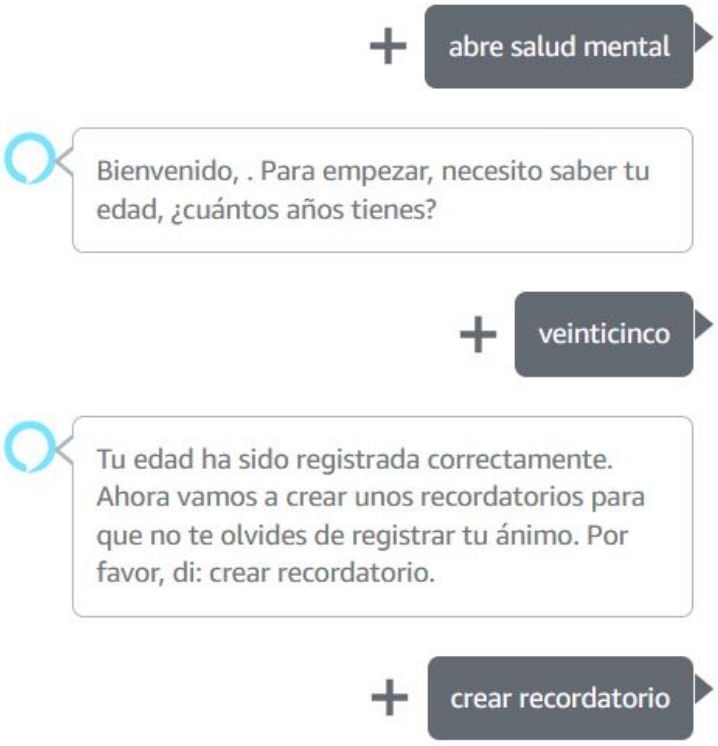


Figura 6.1.- Modo test: primer inicio (Fuente propia)

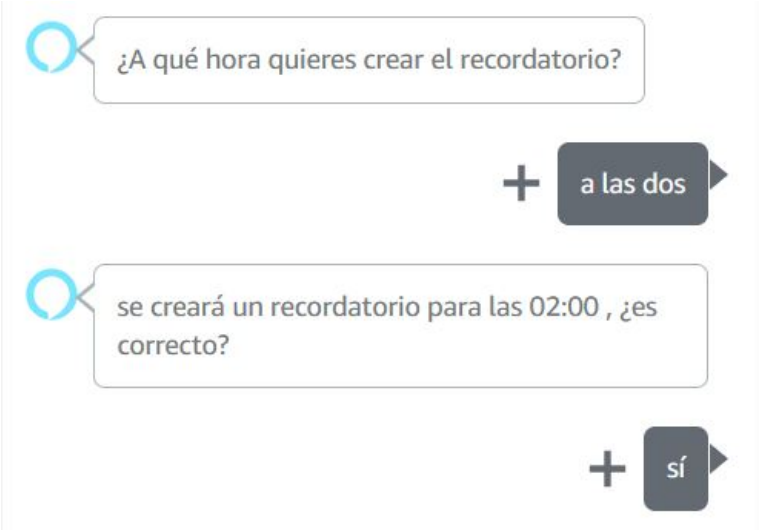


Figura 6.2.- Modo test: primer inicio (Fuente propia)

Una vez guardados estos datos, Alexa pregunta al usuario por su ánimo, que es como se inicia normalmente.

6.2.3.- Inicio habitual

Al abrir la skill con los datos ya guardados, Alexa dirá el saludo de inicio y preguntará al usuario por su ánimo en una escala del uno al diez. Cuando este conteste, le preguntará si quiere escuchar un consejo. Si contesta que no, saltará el texto de ayuda. Si contesta que sí, le dirá un consejo al azar de la siguiente lista:

- ¿Quieres salir a dar un paseo?
- ¿Quieres leer un libro?
- ¿Quieres hablar con algún amigo o familiar?
- ¿Quieres meditar o hacer ejercicios de respiración?
- ¿Quieres escuchar música relajante?
- ¿Quieres escuchar alguna de mis listas de asociaciones amplias?
- ¿Quieres escuchar un chiste?
- ¿Quieres escuchar una frase motivacional?
- ¿Quieres realizar alguno de mis test?

Si contesta que sí, Alexa se despedirá o abrirá el intent correspondiente, según cuál fue la pregunta. Si contesta que no, saltará el texto de ayuda, donde se enumeran todas las opciones de la skill, que son:

- Escuchar música relajante: se puede abrir con los comando de voz “escuchar música”, “meditar”, “música relajante” o “música”.
- Escuchar una lista de asociaciones amplias: se puede abrir con los comandos de voz “escuchar listas”, “lista”, “listas”, “lista de asociaciones amplias” o “asociaciones amplias”.
- Escuchar un chiste: se puede abrir con el comando de voz “cuéntame un chiste” u “otro chiste” si se quieren escuchar más.
- Escuchar una frase motivacional: se puede abrir con los comando de voz “dime una frase motivacional”, “dime frases motivacionales” u “otra frase”, si se quieren escuchar más.

- Realizar un test para evaluar el estado de depresión: se puede abrir con el comando de voz “quiero hacer un test” o “test depresión”. Alexa reproducirá uno de los tres test (el de GDS, Beck o CDI) según la edad del usuario.
- Realizar un test para evaluar el estado rumiativo: se puede abrir con el comando de voz “rumiación” o “test rumiación”.

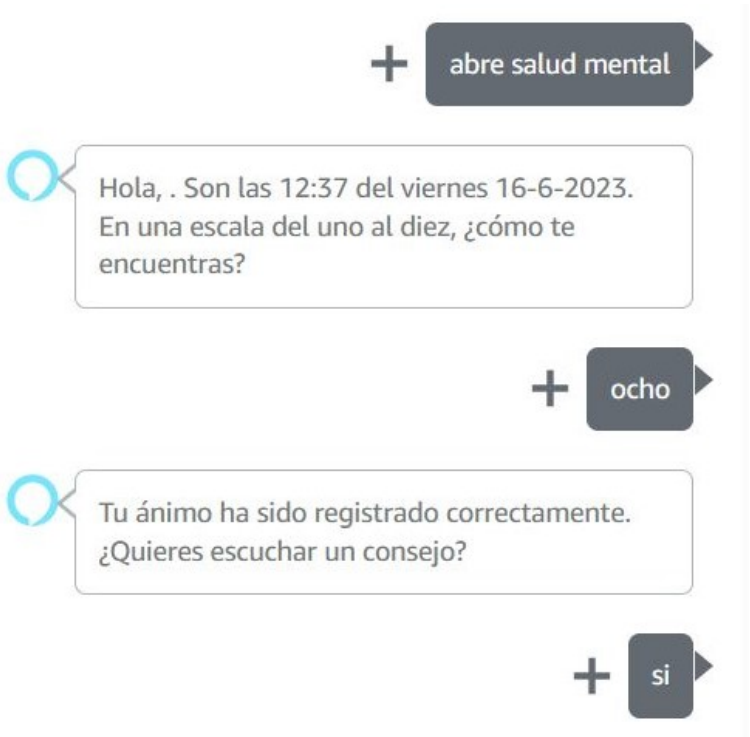


Figura 6.3.- Modo test: inicio habitual (Fuente propia)

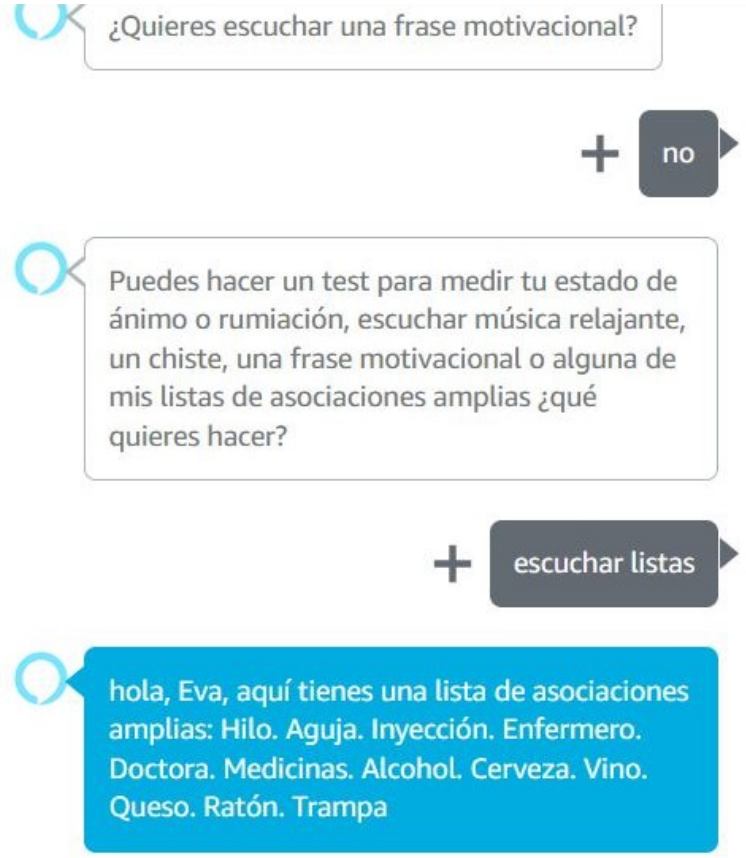


Figura 6.4.- Modo test: inicio habitual (Fuente propia)

6.3.- Aplicación web

Al iniciar la aplicación, aparecerá un cartel pidiendo el nombre de usuario de Amazon.

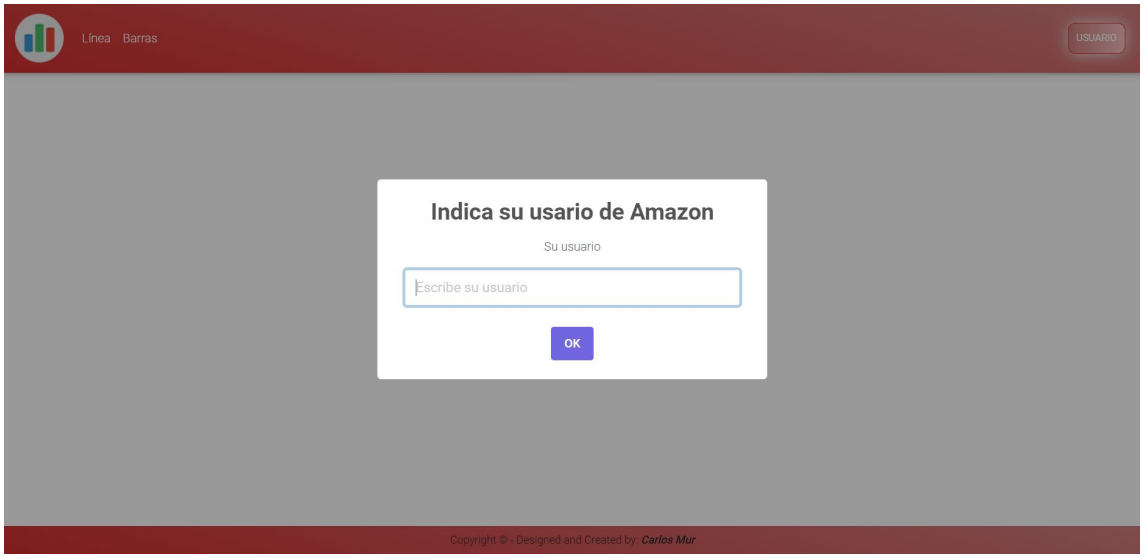


Figura 6.5.- Aplicación web: inicio (Fuente propia)

Tras introducirlo, aparecerá un gráfico de líneas con los datos registrados del nivel de ánimo, los resultados del test de depresión correspondiente al usuario y los del test de rumiación de los últimos 7 días.

Figura 6.6.- Aplicación web: Gráfico de líneas (Fuente propia)

Si se pulsa en el botón “mostrar más”, aparecerá una tabla con los resultados de los tres registros y 3 botones para elegir el número de días de los datos que se muestran.

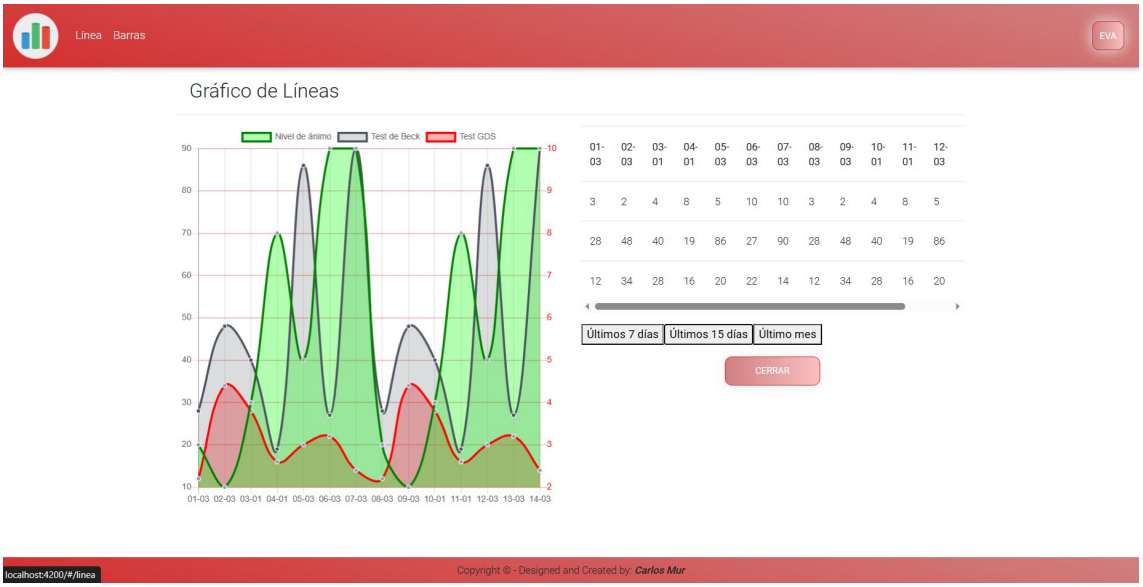


Figura 6.7.- Aplicación web: Gráfico de líneas 15 días (Fuente propia)

También es posible elegir los datos que se muestran en el gráfico pulsando sobre la leyenda para ocultar los que no se quieran ver. Para volver a mostrarlos, solo habría que volver a pulsar sobre el mismo.

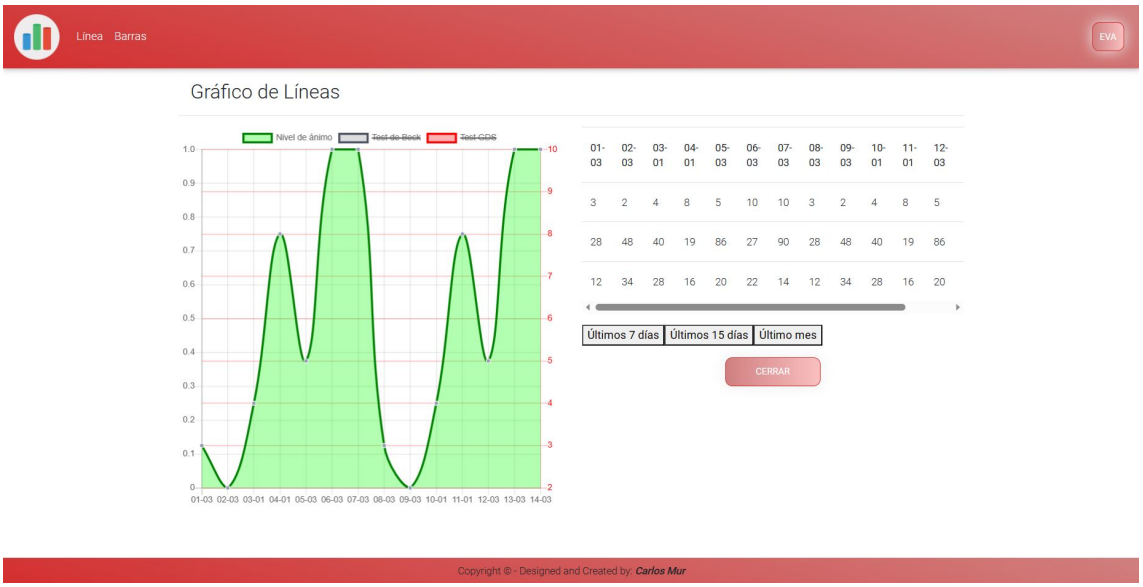


Figura 6.8.- Aplicación web: Gráfico de líneas registro de ánimo (Fuente propia)

Si se pulsa, en el menú de la barra superior, la opción “Barras”, se mostrará un gráfico de barras con los consejos propuestos por Alexa y el número de veces que han sido realizados en los últimos 7, 15 y 30 días.

Figura 6.9.- Aplicación web: Gráfico de barras. (Fuente propia)

Al igual que en el gráfico de líneas, se pueden ocultar o mostrar los datos de la leyenda pulsando sobre los mismos.

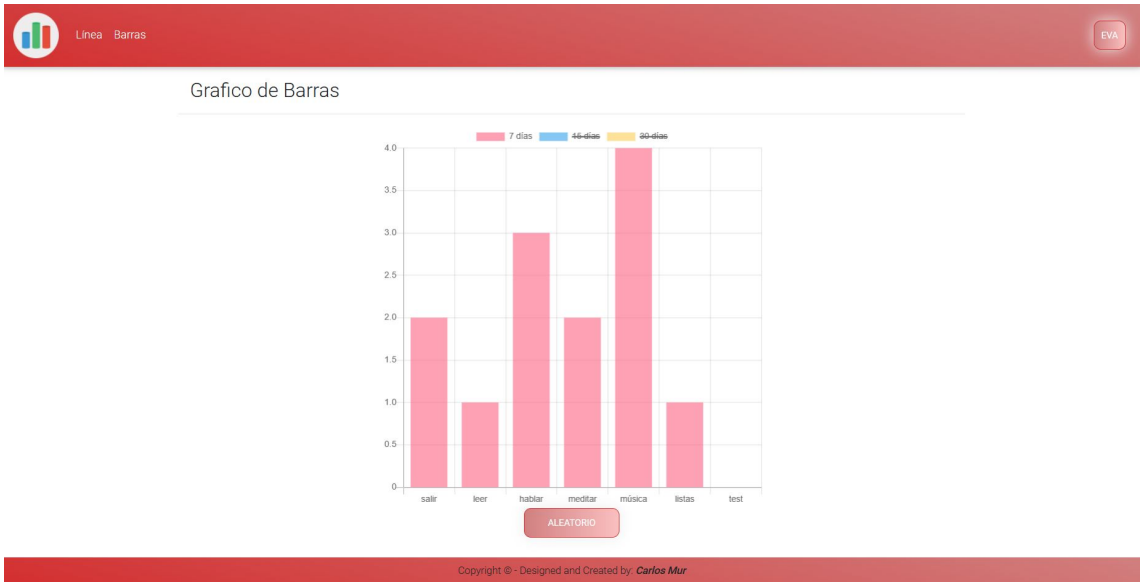


Figura 6.10.- Aplicación web: Gráfico de barras 7 días. (Fuente propia)

7. Manual de instalación

7.1.- Aplicación Web

Para instalar la aplicación web, se debe disponer de un equipo que tenga node.js y AngularCLI instalado, para más información sobre cómo instalar estos entornos, se pueden ver los apartados [24] y [28]. Para la creación del proyecto se ha usado la versión de node 14.20.0 y la versión de angular 12.1.4.

La aplicación se encuentra en el repositorio de GitHub https://github.com/kawalapiti/TFG. Para descargarla, se puede hacer click sobre el botón "Code" y escoger la opción "Download ZIP".

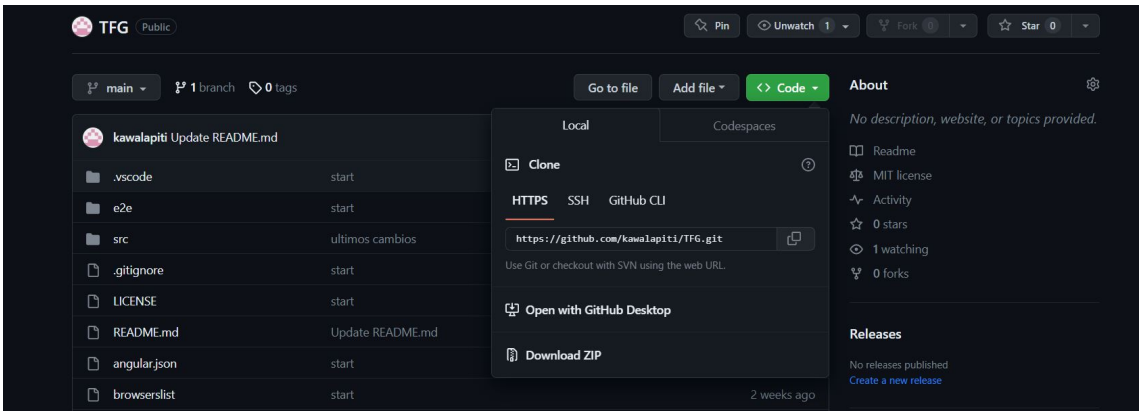


Figura 7.1.- Repositorio GitHub (Fuente propia)

Una vez descomprimida la carpeta, debe abrirse una terminal en la carpeta raíz y ejecutar el comando "ng serve". En cuanto termine de compilar, se podrá visualizar la aplicación entrando desde un navegador a la url http://localhost:4200.

7.2.- Skill Alexa

La aplicación aún no está publicada, pero se puede probar siendo beta-tester. Para ello, habría que añadir el correo que se va a usar para probarla en la configuración de la skill. Es importante que la cuenta de correo tenga asociada una cuenta de Amazon,

ya que si no, no llegará ningún mensaje. Una vez hecho, se enviará un correo con un link para acceder a la skill.

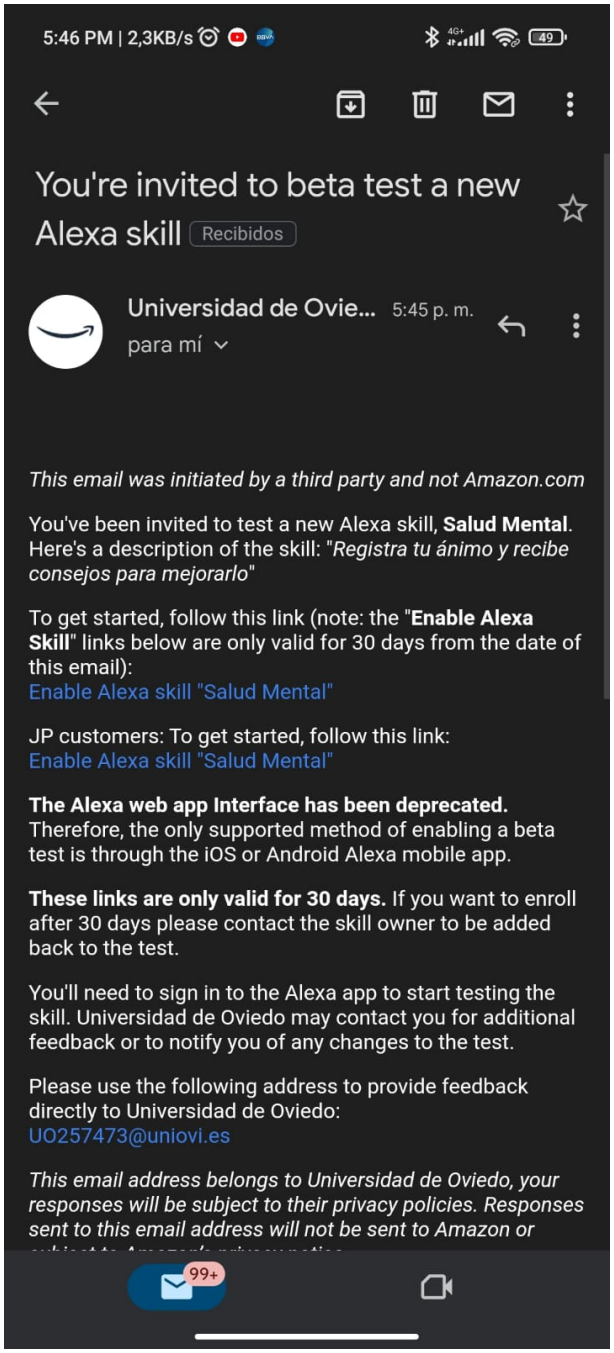


Figura 7.2.- Correo de acceso a la skill (Fuente propia)

Al pulsar en el link, se abrirá el navegador web con una página que muestra un botón para acceder a la aplicación de Alexa y un link para descargarla. Si se pulsas el

botón, da la opción de abrir la aplicación, si se tiene instalada, o acceder a ella desde el propio navegador.

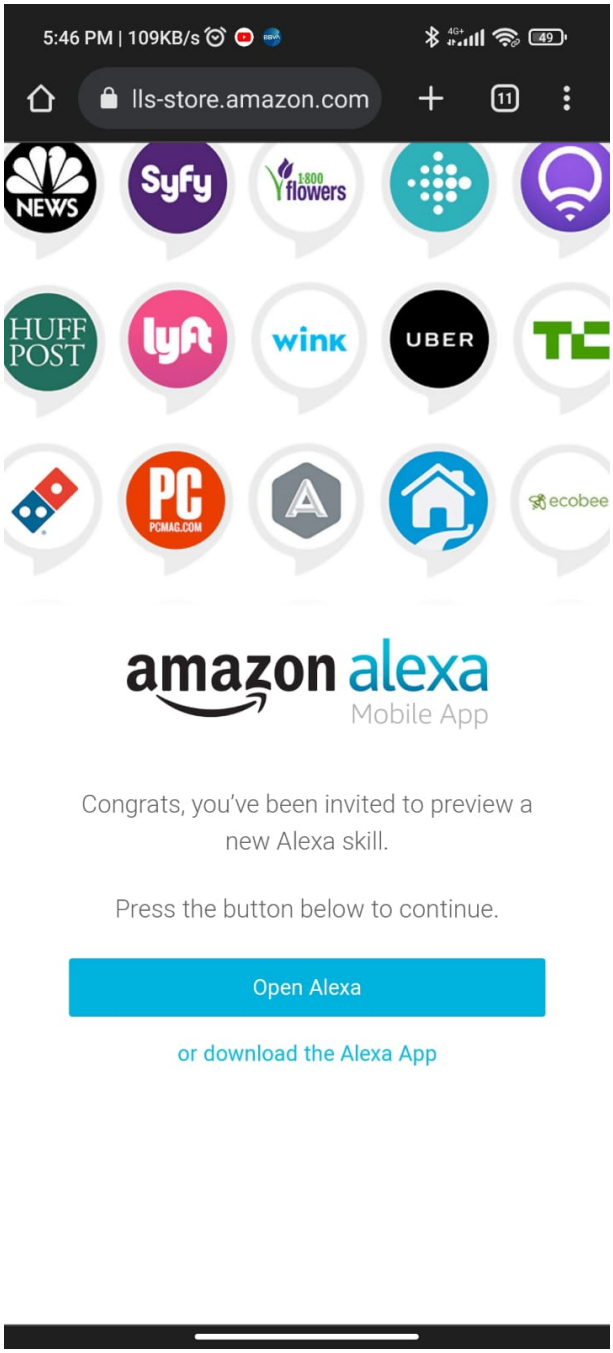


Figura 7.3.- Link de acceso a la skill (Fuente propia)

Tras acceder a la aplicación, se muestra un aviso de que se está accediendo a una skill de prueba.

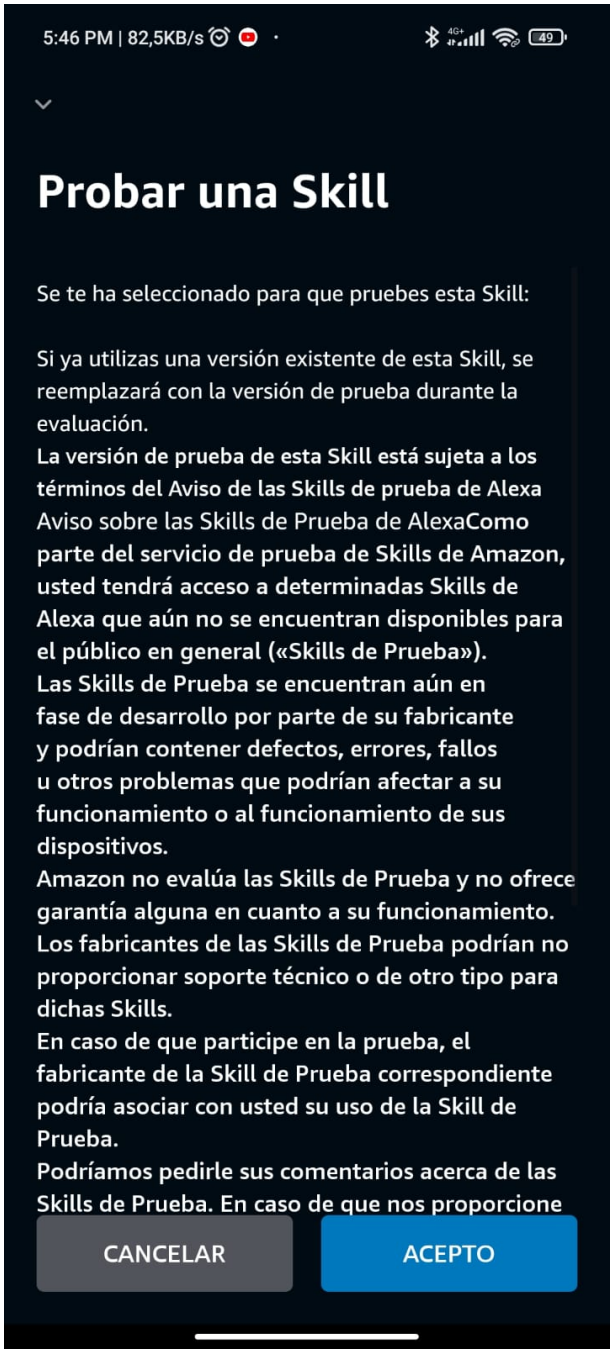


Figura 7.4.- Aviso (Fuente propia)

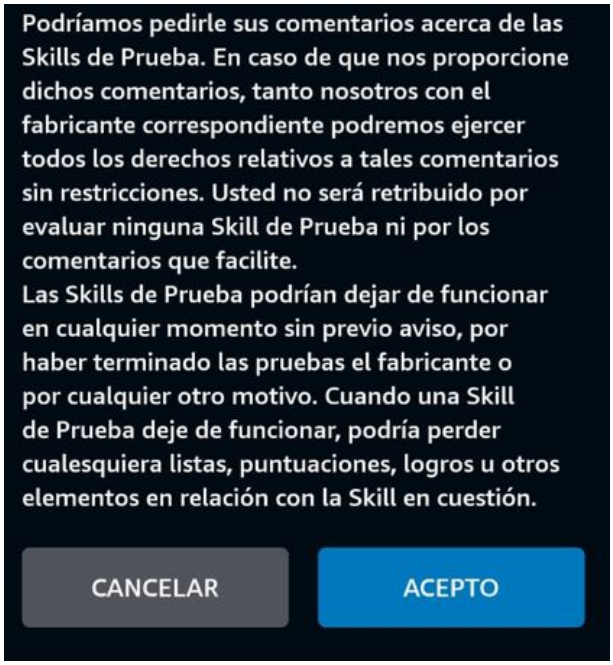


Figura 7.5.- Aviso (segunda parte) (Fuente propia)

Si se acepta, se abrirá la pantalla con la descripción de la skill en la que aparece un botón con el texto ”permitir su uso”. Al pulsar este botón se accede a los permisos de la aplicación. Si se quiere tener la posibilidad de crear recordatorios y que Alexa use el nombre del usuario, deben aceptarse ambos permisos.

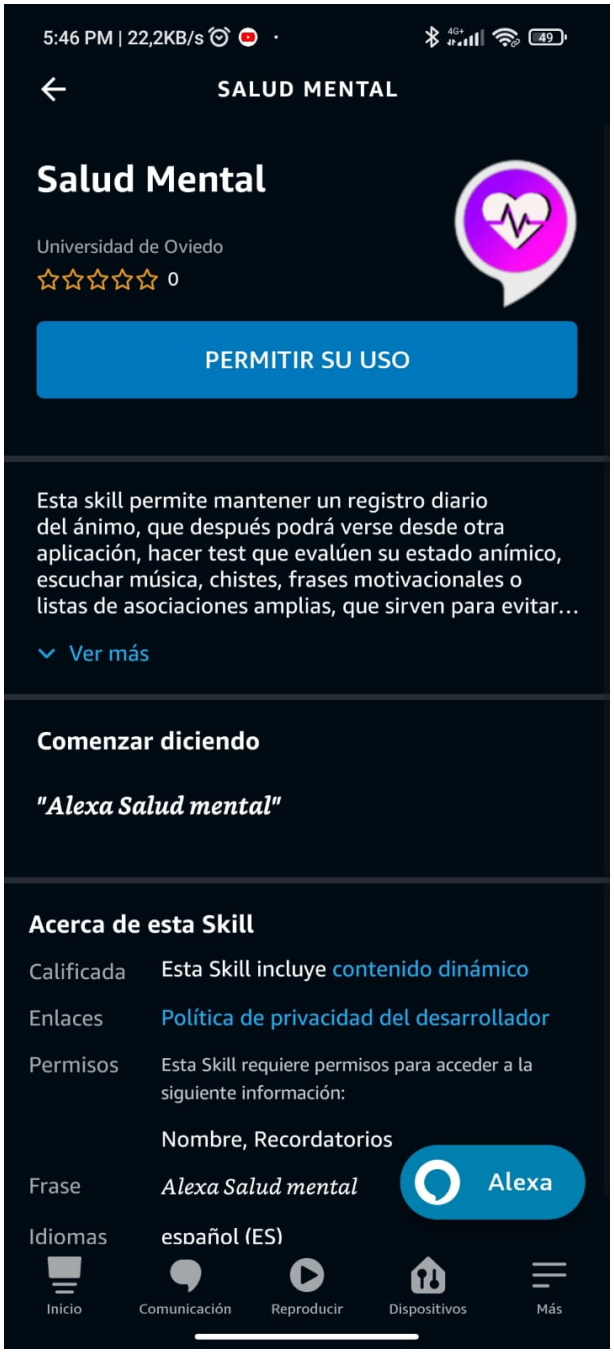


Figura 7.6.- App (Fuente propia)

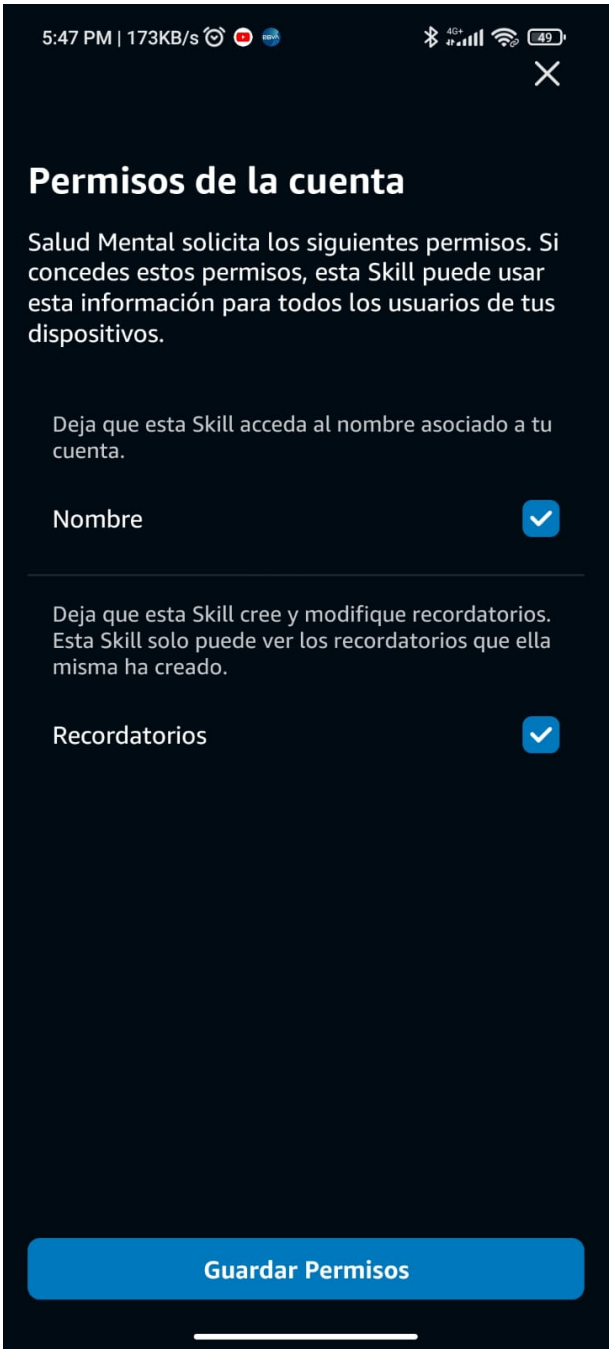


Figura 7.7.- Permisos (Fuente propia)

Una vez guardado, se vuelve a la pantalla con la descripción de la skill, donde ahora aparecerá un botón de configuración desde el que se puede volver a acceder a los permisos y retirarlos si se desea.

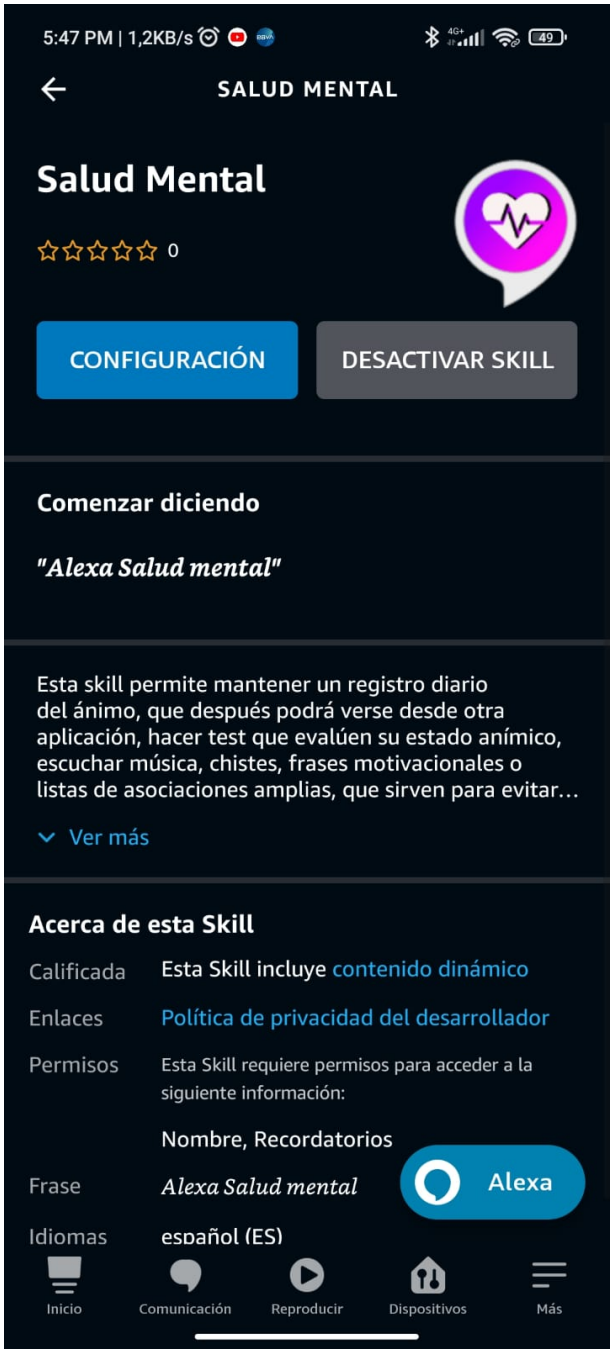


Figura 7.8.- App (Fuente propia)

La aplicación se puede usar desde el propio móvil, simplemente dándole al botón de abajo .ªlexaz diciendo el comando para iniciarla ”Salud mental”.

8. Conclusiones

Han surgido ciertos inconvenientes al realizar este proyecto, ya que en un principio se pretendía usar inteligencia artificial para detectar las emociones en la voz del usuario. Esto no pudo hacerse por los términos de privacidad de Amazon, que impiden que se grabe la voz del usuario para analizarla. Para compensarlo, se optó por usar los test descritos anteriormente, aunque sean menos precisos, sirven para dar cierta noción del ánimo del usuario, y que este mantenga un registro diario de su estado ánimo es un ejercicio útil para ayudar en su mejoría. También hubo que añadir ciertas modificaciones en los test, ya que estos están diseñados para que se lean en un papel y, al ser leídos en alto por Alexa, la longitud del enunciado se hacía demasiado pesada, así que se trató de reducir intentando no cambiarlo demasiado.

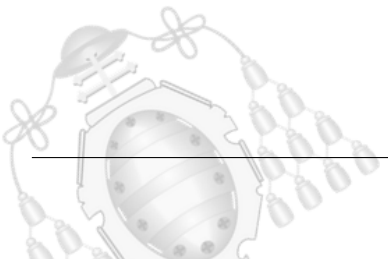
Además, para añadirle contenido y ayudar a los usuarios a mejorar su ánimo, se decidió incluir ciertos consejos que, según las investigaciones, pueden ser útiles para aumentar su humor, como escuchar música relajante o meditar. También se han añadido listas de asociaciones amplias, un método no muy conocido, pero que, según los estudios ya citados, ayudan a evitar un estado mental rumiativo, muy relacionado con los síntomas depresivos.

Como la aplicación mantiene un registro tanto del estado anímico del usuario como de las veces que ha realizado cada consejo, podría hacerse un estudio de la eficacia de estos consejos, si los usuarios permitieran que se usaran sus datos.

Quitando el inconveniente inicial, se han cumplido los objetivos planteados al inicio del proyecto. Se ha creado una aplicación por interfaz de voz que realiza un seguimiento del estado anímico del usuario, además de ofrecerle consejos para mejorarlo, y una plataforma donde el usuario puede ver sus resultados. Además, se han analizado diversos métodos de evaluación de la depresión para escoger el más adecuado teniendo en cuenta las limitaciones del entorno y se han investigado métodos para mantener una actitud positiva que el usuario pueda aplicar fácilmente.

Los tutoriales y documentación proporcionada por Amazon para el desarrollo de una skill de Alexa han ayudado a comprender mejor esta tecnología y aprender a aprovechar todas las funciones que ofrece.

Como trabajo futuro, podrían considerarse otros dispositivos complementarios, como un reloj inteligente que haga un seguimiento del estado físico del usuario: sus horas de sueño, calidad de sueño o el ritmo cardíaco, y le ayude a mantener una rutina de ejercicio y descanso saludable. También se podrían registrar estos datos para evaluar su impacto en el estado anímico.



Bibliografía

[1] *Amazon*. Amazon. URL: https://www.amazon.es/b?ie=UTF8%5C&node=13944662031 (visitado 29-03-2023).

[2] *OPS*. Organización Panamericana de la Salud. URL: https://www.paho.org/es/temas/depresion%20%5C#:~:text=La%5C%20depresi%5C%C3%5C%B3n%5C%20es%5C%20una%5C%20enfermedad%5C%20que%5C%20se%20%5C%20caracteriza%5C%20por%5C%20una,durante%5C%20al%5C%20menos%5C%20dos%5C%20semanas (visitado 29-03-2023).

[3] *GBD Results*. Institute for Health Metrics y Evaluation. URL: https://vizhub.healthdata.org/gbd-results/?params=gbd-api-2019-public/6272860a1e71390f9d8c81131d79623 (visitado 29-03-2023).

[4] *Boletin informativo del INE*. Instituto Nacional de estadística. URL: https://www.ine.es/ss/Satellite?L=es%5C_ES%5C%%C2%A1%5C&c=INECifrasINE%20%5C_C%5C%%C2%A1%5C&cid=1259953225445%5C&p=1254735116567%5C&pagename=ProductosYServicios%5C%%202FINECifrasINE%5C_C%5C%2FPYSDetalleCifrasINE%5C#:~:text=La%5C%20depresi%5C%%20C3%5C%B3n%5C%20grave%5C%20afecta%5C%20a%5C%20230.000%5C%20personas%5C&text=En%5C%%20202020%5C%2C%5C%20se%5C%20cifra%5C%20en,sintomatolog%5C%C3%5C%ADa%5C%2C%5C%20230.000%20%5C%20se%5C%20consideran%5C%20graves.%20Accedido%2029%20de%20marzo%20de%202023 (visitado 29-03-2023).

[5] J. Iglesia et al. «Versión española del cuestionario de Yesavage abreviado (GDS) para el despistaje de depresión en mayores de 65 años: Adaptación y validación». En: *Medifam* 12 (dic. de 2002). DOI: 10.4321/S1131-57682002001000003.

[6] *Inventario de Beck*. Facultad de psicología - UBA. URL: https://www.psi.uba.ar/academica/carrerasdegrado/psicologia/sitios_catedras/obligatorias/070_psicoterapias1/material/inventario_beck.pdf (visitado 29-03-2023).

[7] Martin Solis. «The dilemma of combining positive and negative items in scales». En: *Psicothema* 27 (mayo de 2015), págs. 192-200. DOI: 10.7334/psicothema2014.266.

[8] José Tomás et al. «Method effects associated with negatively worded items Vs. Negative items». En: *Revista Mexicana de Psicologia* 29 (jul. de 2012), págs. 105-115.

[9] *Inventario de Depresión Infantil*. TEA Ediciones. URL: https://web.teaediciones.com/CDI--INVENTARIO-DE-DEPRESION-INFANTIL.aspx (visitado 2023).

[10] Eiran Harel et al. «Linking major depression and the neural substrates of associative processing». En: *Cognitive, Affective, & Behavioral Neuroscience* 16 (ago. de 2016). DOI: 10.3758/s13415-016-0449-9.

[11] Wendy Treynor y Richard Gonzalez. «Rumination Reconsidered: A Psychometric Analysis». En: *Cognitive Therapy and Research* 27 (jun. de 2003). DOI: 10.1023/A:1023910315561.

[12] Malia Mason y Moshe Bar. «The Effect of Mental Progression on Mood». En: *Journal of experimental psychology. General* 141 (ago. de 2011), págs. 217-21. DOI: 10.1037/a0025035.

[13] *Meditacion*. Universidad de Jaen. URL: https://tauja.ujaen.es/bitstream/10953.1/15909/4/Martnez%5C_Del%5C_Cubo%5C_Nuria%5C_TFG%5C_Psicologa.pdf (visitado 25-03-2023).

[14] *La intervención de Kundalini Yoga aumenta el volumen del hipocampo en adultos mayores*. URL: https://kundaliniresearchinstitute.org/es/la-intervencion-de-kundalini-yoga-aumenta-el-volumen-del-hipocampo-en-adultos-mayores-un-ensayo-piloto-controlado-y-aleatorizado/ (visitado 25-03-2023).

[15] Diego Calderón M. «Epidemiología de la depresió en el adulto mayor». es. En: *Revista Medica Herediana* 29 (jul. de 2018), págs. 182-191. ISSN: 1018-130X. DOI: https://doi.org/10.20453/rmh.v29i3.3408. URL: http://www.scielo.org.

pe/scielo.php?script=sci_arttext&pid=S1018-130X2018000300009&nrm=iso.

[16] Sánchez-González VJ. Ortega-Díaz DI Orozco-Barajas M. «Entrenamiento cognitivo: efectos en la cognición, depresión y actividades de la vida diaria en sujetos institucionalizados». En: *Salud Jalisco* 7 (jun. de 2020), págs. 26-31.

[17] Liliana Chazenbalk. «El valor del humor en el proceso psicoterapéutico». En: *Psicodebate* 6 (dic. de 2006), págs. 73-84. DOI: 10.18682/pd.v6i0.442.

[18] *"cómo me encuentro"*. Amazon. URL: https://www.amazon.es/Yasyt-C%C3%B3mo-meencuentro/dp/B08R8QVMN1/ref=sr%5C_1%5C_1%5C_%5C_mk%5C_es%5C_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91%5C&crid=38M17ZHL50TX7%5C&keywords=c%C3%B3mo+me+encuentro%5C&qid=1650379846%5C≈s=digital-skills%5C&sprefix=c%C3%B3mo+me+encuentro,alexa-skills,75%5C&sr=1-1 (visitado 2023).

[19] *Prueba Calm*. Calm. URL: https://www.calm.com/es (visitado 29-03-2023).

[20] *Track Your Happiness*. URL: https://go.trackyourhappiness.org/ (visitado 29-03-2023).

[21] *What's Up? - Mental Health App*. Google play. URL: https://play.google.com/store/apps/details?id=com.jacksontempra.apps.whatsup%5C&hl=es (visitado 29-03-2023).

[22] *Do you feel like shit?* URL: https://philome.la/jace%5C_harr/you-feel-like-shit-an-interactive-self-care-guide/play/index.html (visitado 29-03-2023).

[23] «*Alexa Skills Kit Official Site: Build Skills for Voice*». Amazon (Alexa). URL: https://developer.amazon.com/es-ES/alexa/alexa-skills-kit.html (visitado 29-03-2023).

[24] «*Node.js*». Node.js. URL: https://nodejs.org/es (visitado 29-03-2023).

[25] *nvm para windows*. GitHub. URL: https://github.com/coreybutler/nvm-windows#installation--upgrades (visitado 28-04-2023).

[26] «*Alexa Skills Kit Official Site: Build Skills for Voice*». Amazon (Alexa). URL: https://developer.amazon.com/es-ES/alexa/alexa-skills-kit.html (visitado 29-03-2023).

[27] *SQL*. Microsoft. URL: https://support.microsoft.com/es-es/office/access-sql-conceptos-b%5C%C3%5C%A1sicos-vocabulario-y-sintaxis-444d0303-cde1-424e-9a74-e8dc3e460671 (visitado 29-03-2023).

[28] *Angular*. Google. URL: https://angular.io/ (visitado 19-04-2023).

[29] *Graficos - Angular, cmurestudillos*. GitHub. URL: https://github.com/cmurestudillos/graficos-angular (visitado 21-04-2023).

[30] «*Alexa y Google Assistants con LUIS*». *Piensa en software, desarrolla en colores*. URL: https://blogs.encamina.com/piensa-en-software-desarrolla-en-colores/alexa-y-google-assistants-con-luis/ (visitado 31-03-2023).

[31] *Zero To Hero, Alexa developers*. Amazon. URL: https://www.youtube.com/watch?v=CzTKDu7Qgjs&list=PLdZn93YfA_1ZP1WFkz6bm08v3zFfWwfGW&index=1 (visitado 23-04-2023).

[32] *Git*. URL: https://git-scm.com/ (visitado 23-04-2023).

[33] *OverLeaf*. OverLeaf. URL: https://es.overleaf.com/ (visitado 28-04-2023).

[34] *Documentación de OverLeaf*. OverLeaf. URL: https://www.overleaf.com/learn (visitado 28-04-2023).

[35] *PlantUMplantUM*. URL: https://plantuml.com/es/ (visitado 28-04-2023).

[36] *MySQL*. Oracle Corporation. URL: https://dev.mysql.com/doc/ (visitado 28-04-2023).

Anexos

Anexo A. El valor del humor en el proceso psicoterapéutico

Algunos beneficios del buen humor y la risa son:

1. Libera las hormonas endorfinas: cuando la glándula pituitaria recibe un estímulo generado por nuestra sonrisa voluntaria y consciente, reacciona liberando endorfinas, las que además de ser el analgésico natural del cuerpo, producen, al ser liberadas, una sensación de bienestar.
2. Disminuye la hormona suprarrenal cortisol: el estrés crónico provoca cambios fisiológicos adversos, mientras que la risa es su antídoto. Esta hace descender el nivel de cortisol que se produce en la sangre ante una situación de estrés. La risa y el humor son escapes al sufrimiento que resulta de la diferencia entre las aspiraciones humanas y la realidad que a uno le toca vivir.El Dr. Daniel Rossetti, (2001) indica que "durante la risa hay una reducción y/o normalización de aquellas funciones orgánicas que se alteran durante el estrés.Es así mismo como se ha determinado que la risa disminuye la presión arterial, la frecuencia cardiaca, la tensión muscular y decrece la activación del estrés."
3. Relaja el sistema muscular: Actúa sobre el sistema neurovegetativo, que es regulado por el sistema límbico, el núcleo de las emociones. En los episodios de risa se pone en marcha en primer lugar el sistema nervioso simpático, para dar pronto paso al parasimpático, cuya acción es más duradera. La tensión arterial baja y regulariza la respiración y la digestión, pues al descender el diafragma se produce un efecto masaje sobre el hígado y la vesícula biliar. Con relación a la respiración, se produce una mejor oxigenación de todos los tejidos.
4. Algunas investigaciones médicas y especialmente de la psico-neuro-inmuno-endocrinología determinaron que la glándula timo no se atrofia en la adolescencia. Pero sí cambia de tamaño y disminuye, de este

modo, en beneficio del organismo.Es la glándula más importante del sistema inmunológico, en ella maduran los linfocitos T. Si realmente se atrofiaran, no existiría la inmunidad del organismo. El humor, ayuda a mantenerla en funcionamiento.

Anexo B. Escala de depresión geriátrica de Yesavage

¿En el fondo está satisfecho con su vida?	Sí	No*
¿Ha abandonado muchas de sus actividades y pasatiempos?	Sí*	No
¿Siente que su vida está vacía?	Sí*	No
¿Se aburre con frecuencia?	Sí*	No
¿Tiene esperanza en el futuro?	Sí	No*
¿Le preocupan ideas que no pueda quitar de su cabeza?	Sí*	No
¿Se encuentra de buen ánimo la mayor parte del tiempo?	Sí	No*
¿Teme que algo malo pueda sucederle?	Sí*	No
¿Se siente feliz la mayor parte del tiempo?	Sí	No*
¿Se siente desamparado con frecuencia?	Sí*	No
¿Con frecuencia se siente desvelado y nervioso?	Sí*	No
¿Prefiere quedarse en casa a salir y realizar cosas nuevas?	Sí*	No
¿Se preocupa con frecuencia por el futuro?	Sí*	No
¿Piensa que tiene más problemas de memoria que las demás personas?	Sí*	No
¿Piensa que es bueno estar vivo hoy?	Sí	No*
¿Se siente triste y desanimado con frecuencia?	Sí*	No
¿Se siente inútil en su estado actual?	Sí*	No
¿Se preocupa mucho por el pasado?	Sí*	No
¿Le parece que la vida es algo apasionante?	Sí	No*
¿Le cuesta mucho emprender nuevos proyectos?	Sí*	No

Continúa en la siguiente página

¿Se siente con energías?	Sí	No*
¿Piensa que su situación no tiene arreglo?	Sí*	No
¿Piensa que la mayor parte de la gente está mejor que usted?	Sí*	No
¿Se disgusta con frecuencia por cosas sin importancia?	Sí*	No
¿Siente ganas de llorar frecuentemente?	Sí*	No
¿Tiene dificultad para concentrarse?	Sí*	No
¿Disfruta al levantarse de mañana?	Sí	No*
¿Prefiere evitar las reuniones sociales?	Sí*	No
¿Le resulta fácil tomar decisiones?	Sí	No*
¿Siente su mente tan despejada como antes?	Sí*	No

Cuadro 1.- Test GDS

* = Asignar un punto en cada respuesta marcada con el asterisco. Aplicar sin los asteriscos.

Anexo C. Inventario de depresión de Beck (BDI-2)

Este cuestionario consta de 21 grupos de afirmaciones. Por favor, lea con atención cada uno de ellos cuidadosamente. Luego elija uno de cada grupo, el que mejor describa el modo como se ha sentido las últimas dos semanas, incluyendo el día de hoy. Marque con un círculo el número correspondiente al enunciado elegido Si varios enunciados de un mismo grupo le parecen igualmente apropiados, marque el número más alto. Verifique que no haya elegido más de uno por grupo, incluyendo el ítem 16 (cambios en los hábitos de Sueño) y el ítem 18 (cambios en el apetito)

- Tristeza

1. No me siento triste.
2. Me siento triste gran parte del tiempo.

3. Me siento triste todo el tiempo.
4. Me siento tan triste o soy tan infeliz que no puedo soportarlo.

- Pesimismo
  1. No estoy desalentado respecto del mi futuro.
  2. Me siento más desalentado respecto de mi futuro que lo que solía estarlo.
  3. No espero que las cosas funcionen para mí.
  4. Siento que no hay esperanza para mi futuro y que sólo puede empeorar.
- Fracaso
  1. No me siento como un fracasado.
  2. He fracasado más de lo que hubiera debido.
  3. Cuando miro hacia atrás, veo muchos fracasos.
  4. Siento que como persona soy un fracaso total.
- Pérdida de placer
  1. Obtengo tanto placer como siempre por las cosas de las que disfruto.
  2. No disfruto tanto de las cosas como solía hacerlo
  3. Obtengo muy poco placer de las cosas que solía disfrutar.
  4. No puedo obtener ningún placer de las cosas de las que solía disfrutar.
- Sentimientos de Culpa
  1. No me siento particularmente culpable.
  2. Me siento culpable respecto de varias cosas que he hecho o que debería haber hecho.
  3. Me siento bastante culpable la mayor parte del tiempo.
  4. Me siento culpable todo el tiempo.
- Sentimientos de Castigo
  1. No siento que este siendo castigado.
  2. Siento que tal vez pueda ser castigado.
  3. Espero ser castigado.
  4. Siento que estoy siendo castigado.

- Disconformidad con uno mismo.
    1. Siento acerca de mi lo mismo que siempre.
    2. He perdido la confianza en mí mismo.
    3. Obtengo muy poco placer de las cosas que solía disfrutar.
    4. No puedo obtener ningún placer de las cosas de las que solía disfrutar.
- Autocrítica
    1. No me critico ni me culpo más de lo habitual.
    2. Estoy más crítico conmigo mismo de lo que solía estarlo.
    3. Me critico a mí mismo por todos mis errores.
    4. Me culpo a mí mismo por todo lo malo que sucede.
- Pensamientos o Deseos Suicidas
    1. No tengo ningún pensamiento de matarme.
    2. He tenido pensamientos de matarme, pero no lo haría.
    3. Querría matarme.
    4. Me mataría si tuviera la oportunidad de hacerlo.
- Llanto
    1. No lloro más de lo que solía hacerlo.
    2. Lloro más de lo que solía hacerlo.
    3. Lloro por cualquier pequeñez.
    4. Siento ganas de llorar pero no puedo.
- Agitación
    1. No estoy más inquieto o tenso que lo habitual.
    2. Me siento más inquieto o tenso que lo habitual.
    3. Estoy tan inquieto o agitado que me es difícil quedarme quieto.
    4. Estoy tan inquieto o agitado que tengo que estar siempre en movimiento o haciendo algo.
- Pérdida de Interés
    1. No he perdido el interés en otras actividades o personas.

2. Estoy menos interesado que antes en otras personas o cosas.
3. He perdido casi todo el interés en otras personas o cosas.
4. Me es difícil interesarme por algo.

- Indecisión
  1. Tomo mis propias decisiones tan bien como siempre.
  2. Me resulta más difícil que de costumbre tomar decisiones.
  3. Encuentro mucha más dificultad que antes para tomar decisiones.
  4. Tengo problemas para tomar cualquier decisión.
- Desvalorización
  1. No siento que yo no sea valioso.
  2. No me considero a mi mismo tan valioso y útil como solía considerarme.
  3. Me siento menos valioso cuando me comparo con otros.
  4. Siento que no valgo nada.
- Pérdida de Energía
  1. Tengo tanta energía como siempre.
  2. Tengo menos energía que la que solía tener.
  3. No tengo suficiente energía para hacer demasiado.
  4. No tengo energía suficiente para hacer nada.
- Cambios en los Hábitos de Sueño
  1. No he experimentado ningún cambio en mis hábitos de sueño.
  2. Duermo un poco más que lo habitual.
  3. Duermo un poco menos que lo habitual.
  4. Duermo mucho más que lo habitual.
  5. Duermo mucho menos que lo habitual.
  6. Duermo la mayor parte del día.
  7. Me despierto 1-2 horas más temprano y no puedo volver a dormirme.
- Irritabilidad
  1. No estoy tan irritable que lo habitual.

2. Estoy más irritable que lo habitual.
3. Estoy mucho más irritable que lo habitual.
4. Estoy irritable todo el tiempo.

- Cambios en el Apetito

1. No he experimentado ningún cambio en mi apetito.
2. Mi apetito es un poco menor que lo habitual.
3. Mi apetito es un poco mayor que lo habitual.
4. Mi apetito es mucho menor que antes.
5. Mi apetito es mucho mayor que lo habitual.
6. No tengo apetito en absoluto.
7. Quiero comer todo el día.

- Dificultad de Concentración

1. Puedo concentrarme tan bien como siempre.
2. No puedo concentrarme tan bien como habitualmente.
3. Me es difícil mantener la mente en algo por mucho tiempo.
4. Encuentro que no puedo concentrarme en nada.

- Cansancio o Fatiga

1. No estoy más cansado o fatigado que lo habitual.
2. Me fatigo o me canso más fácilmente que lo habitual.
3. Estoy demasiado fatigado o cansado para hacer muchas de las cosas que solía hacer.
4. Estoy demasiado fatigado o cansado para hacer la mayoría de las cosas que solía hacer.

- Pérdida de Interés en el Sexo

1. No he notado ningún cambio reciente en mi interés por el sexo.
2. Estoy menos interesado en el sexo de lo que solía estarlo.
3. Estoy mucho menos interesado en el sexo.
4. He perdido completamente el interés en el sexo.

Anexo D. Cuestionario de Depresión Infantil (CDI)

1.
   - Estoy triste de vez en cuando.
   - Estoy triste muchas veces.
   - Estoy triste siempre.
2.
   - Nunca me saldrá nada bien.
   - No estoy seguro de si las cosas me saldrán bien.
   - Las cosas me saldrán bien.
3.
   - Hago bien la mayoría de las cosas.
   - Hago mal muchas cosas.
   - Todo lo hago mal.
4.
   - Me divierten muchas cosas.
   - Me divierten algunas cosas.
   - Nada me divierte.
5.
   - Soy malo siempre.
   - Soy malo muchas veces.
   - Soy malo algunas veces.
6.
   - A veces pienso que me pueden ocurrir cosas malas.
   - Me preocupa que me ocurran cosas malas.
   - Estoy seguro de que me van a ocurrir cosas terribles.
7.
   - Me odio.
   - No me gusta como soy.
   - Me gusta como soy.
8.
   - Todas las cosas malas son culpa mía.
   - Muchas cosas malas son culpa mía.
   - Generalmente no tengo la culpa de que ocurran cosas malas.
9.
   - No pienso en matarme.
   - Pienso en matarme pero no lo haría.

- Quiero matarme.

10.
- Tengo ganas de llorar todos los días.
- Tengo ganas de llorar muchos días.
- Tengo ganas de llorar de cuando en cuando.

11.
- Las cosas me preocupan siempre.
- Las cosas me preocupan muchas veces.
- Las cosas me preocupan de cuando en cuando.

12.
- Me gusta estar con la gente.
- Muy a menudo no me gusta estar con la gente.
- No quiero en absoluto estar con la gente.

13.
- No puedo decidirme.
- Me cuesta decidirme.
- Me decido fácilmente.

14.
- Tengo buen aspecto.
- Hay algunas cosas de mi aspecto que no me gustan.
- Soy feo.

15.
- Siempre me cuesta ponerme a hacer los deberes.
- Muchas veces me cuesta ponerme a hacer los deberes.
- No me cuesta ponerme a hacer los deberes.

16.
- Todas las noches me cuesta dormirme.
- Muchas noches me cuesta dormirme.
- Duermo muy bien.

17.
- Estoy cansado de cuando en cuando.
- Estoy cansado muchos días.
- Estoy cansado siempre.

18.
- La mayoría de los días no tengo ganas de comer.
- Muchos días no tengo ganas de comer.
- Como muy bien.

19.
- No me preocupa el dolor ni la enfermedad.
- Muchas veces me preocupa el dolor y la enfermedad.

- Siempre me preocupa el dolor y la enfermedad.

20.
- Nunca me siento solo.
- Me siento solo muchas veces.
- Me siento solo siempre.

21.
- Nunca me divierto en el colegio.
- Me divierto en el colegio sólo de vez en cuando.
- Me divierto en el colegio muchas veces.

22.
- Tengo muchos amigos.
- Tengo muchos amigos pero me gustaría tener más.
- No tengo amigos.

23.
- Mi trabajo en el colegio es bueno.
- Mi trabajo en el colegio no es tan bueno como antes.
- Llevo muy mal las asignaturas que antes llevaba bien.

24.
- Nunca podré ser tan bueno como otros niños.
- Si quiero puedo ser tan bueno como otros niños.
- Soy tan bueno como otros niños.

25.
- Nadie me quiere.
- No estoy seguro de que alguien me quiera.
- Estoy seguro de que alguien me quiere.

26.
- Generalmente hago lo que me dicen.
- Muchas veces no hago lo que me dicen.
- Nunca hago lo que me dicen.

27.
- Me llevo bien con la gente.
- Me peleo muchas veces.
- Me peleo siempre.

Anexo E. Escala de Respuestas Rumiativas

1. Piensas en lo solo que te sientes.

2. Piensas: no podré hacer mi trabajo si no salgo de este estado.
3. Piensas en tu sensación de fatiga y malestar.
4. Piensas en lo difícil que te resulta concentrarte.
5. Piensas, ¿qué hago yo para merecer esto?
6. Piensas en lo pasivo y poco motivado que te sientes.
7. Analizas acontecimientos recientes e intentas comprender por qué estás deprimido.
8. Piensas que ya nada te resulta estimulante.
9. Piensas: ¿por qué no puedo seguir adelante?
10. Piensas: ¿por qué siempre reacciono así?
11. Te aíslas y piensas por qué te sientes así.
12. Escribes lo que piensas y lo analizas.
13. Piensas en una situación reciente, deseando que hubiera ido mejor.
14. Piensas: no seré capaz de concentrarme si me sigo sintiendo así.
15. Piensas: ¿Por qué tengo problemas que los demás no tienen?
16. Piensas: ¿por qué no soy capaz de gestionar mejor mi situación?
17. Piensas en lo triste que te sientes.
18. Piensas en todos tus defectos, errores, fallos, carencias.
19. Piensas en que no te sientes con ganas de hacer nada.
20. Analizas tu personalidad para intentar comprender por qué estás deprimido.
21. Vas solo a algún lugar para pensar en tus sentimientos.
22. Piensas en lo enfadado que estás contigo mismo.

Anexo F. Graficos-Angular, licencia

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the ”Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED .ªS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Anexo G. Código Skill

```
1  const Alexa = require("ask-sdk-core");
2  const AWS = require("aws-sdk");
3  const ddbAdapter = require("ask-sdk-dynamodb-persistence-adapter");
4  const util = require("./util"); // utility functions
5  const questionsGDS = require("./JSON/preguntasGDS.json");
6  const questionsBeck = require("./JSON/preguntasBeck.json");
7  const questionsCDI = require("./JSON/CDI.json");
8  const interceptors = require("./interceptors");
9  const questionsRumiacion = require("./JSON/preguntasRumiacion.json");
10 const asociacionesAmplias = require("./JSON/asociacionesAmplias.json");
11 const preguntas = require("./JSON/preguntas.json");
12 const chistes = require("./JSON/chistes.json");
13 const frases = require("./JSON/frasesMotivacion.json");
14 var preguntasCount;
15 var answerCount = 0;
16 var currentQuestion = 0;
```

```
17 var countSi;
18 var test = "";
19 const estados = { EMPEZANDO: 0, PREGUNTANDO: 1 };
20 var estadoActual = estados.EMPEZANDO;
21 const timezone = "Europe/Madrid";
22 var hoy = new Date();
23 var fecha =
24   hoy.getDate() + "-" + (hoy.getMonth() + 1) + "-" + hoy.getFullYear();
25 const moment = require("moment-timezone");
26 const dias = [
27   "domingo",
28   "lunes",
29   "martes",
30   "miercoles",
31   "jueves",
32   "viernes",
33   "sabado",
34 ];
35 const numeroDia = hoy.getDay();
36 const nombreDia = dias[numeroDia];
37 const hora = moment().tz(timezone).format("HH:mm");
38
39 const LaunchRequestHandler = {
40   canHandle(handlerInput) {
41     return (
42       Alexa.getRequestType(handlerInput.requestEnvelope) === "LaunchRequest"
43     );
44   },
45   handle(handlerInput) {
46     const { attributesManager, serviceClientFactory, requestEnvelope } =
47       handlerInput;
48     const sessionAttributes =
49       handlerInput.attributesManager.getSessionAttributes();
50     let speakOutput = "";
51     const timezone = 'Europe/Madrid';
52     let hora = moment().tz(timezone).format("HH:mm");
53     const name = sessionAttributes["name"] || "";
```

```
54      if (sessionAttributes["edad"] === null || sessionAttributes["edad"] ===
    undefined) {
55        test = "edad";
56        speakOutput = `Bienvenido, ${name}. Para empezar, necesito saber tu edad,
    cuantos aos tienes?`;
57      } else if (!sessionAttributes["reminderId"]) {
58        speakOutput = `Bienvenido, ${name}. Son las ${hora} del ${nombreDia}
    ${fecha}. Empecemos estableciendo unos recordatorios para registrar tu estado
    de animo.`;
59        return RecordatoriosIntentHandler.handle(handlerInput);
60      } else {
61        speakOutput = `Hola, ${name}. Son las ${hora} del ${nombreDia} ${fecha}.
    En una escala del uno al diez, como te encuentras?`;
62        test = "animo";
63      }
64
65      util.addAPLTexto(`Hola, ${name}!`, "", handlerInput);
66
67      return handlerInput.responseBuilder
68        .speak(speakOutput)
69        .reprompt(speakOutput)
70        .getResponse();
71    },
72  };
73
74
75  const RecordatoriosIntentHandler = {
76    canHandle(handlerInput) {
77      return (
78        Alexa.getRequestType(handlerInput.requestEnvelope) === "IntentRequest" &&
79        Alexa.getIntentName(handlerInput.requestEnvelope) ===
80          "RecordatoriosIntent"
81      );
82    },
83    async handle(handlerInput) {
84      const { attributesManager, serviceClientFactory, requestEnvelope } =
85        handlerInput;
86      const sessionAttributes = attributesManager.getSessionAttributes();
```

```
87      const { intent } = requestEnvelope.request;
88      //let timezone = sessionAttributes['timezone'];
89      let timezone = "Europe/Madrid"; // so it works on the simulator, you should
    uncomment this line, replace with your time zone and comment sentence below
90      const name = sessionAttributes["name"] || "";
91      let speechText = "";
92      if (!timezone) {
93        return handlerInput.responseBuilder
94          .speak(
95            handlerInput.t(
96              "No he podido determinar tu zona horaria. Verifica la configuracion
    de tu dispositivo, abre otraa vez la skill e intentalo otra vez."
97            )
98          )
99          .getResponse();
100     }
101     try {
102       const { permissions } = requestEnvelope.context.System.user;
103       if (!(permissions && permissions.consentToken))
104         throw { statusCode: 401, message: "No permissions available" };
105       const reminderServiceClient =
106         serviceClientFactory.getReminderManagementServiceClient();
107       const remindersList = await reminderServiceClient.getReminders();
108       console.log("Current reminders: " + JSON.stringify(remindersList));
109       const previousReminder = sessionAttributes["reminderId"];
110       if (previousReminder) {
111         try {
112           if (remindersList.totalCount !== "0") {
113             await reminderServiceClient.deleteReminder(previousReminder);
114             delete sessionAttributes["reminderId"];
115             console.log("Deleted previous reminder token: " + previousReminder);
116           }
117         } catch (error) {
118           console.log(
119             "Failed to delete reminder: " +
120               previousReminder +
121               " via " +
122               JSON.stringify(error)
```

```
123         );
124       }
125     }
126     const locale = Alexa.getLocale(requestEnvelope);
127     const message = `Hola, ${name}, recuerda registrar tu estado de animo`;
128     moment.locale(locale);
129     const createdMoment = moment().tz(timezone);
130     let hora =
131       handlerInput.requestEnvelope.request.intent.slots.recordatorio.value;
132     let separado = hora.split(":");
133     let triggerMoment = moment()
134       .tz(timezone)
135       .set({ hour: "separado[0]", minute: "separado[1]" });
136     const reminder = util.createReminder(
137       createdMoment,
138       triggerMoment,
139       timezone,
140       locale,
141       message
142     );
143     const reminderResponse = await reminderServiceClient.createReminder(
144       reminder
145     );
146     sessionAttributes["reminderId"] = reminderResponse.alertToken;
147     console.log(
148       "Reminder created with token: " + reminderResponse.alertToken
149     );
150     speechText = `${name} tu recordatorio se ha creado correctamente. Del uno
    al diez, como te encuentras de animo?`;
151   } catch (error) {
152     console.log(JSON.stringify(error));
153     switch (error.statusCode) {
154       case 401:
155         handlerInput.responseBuilder.withAskForPermissionsConsentCard([
156           "alexa::alerts:reminders:skill:readwrite",
157         ]);
158         speechText =
```

```
159           "Parece que no has autorizado el envio de recordatorios. Te he
    enviado una tarjeta a la app Alexa para que lo habilites. ";
160         break;
161       case 403:
162         speechText = ‘Este dispositivo no soporta la operacion que estas
    intentando realizar. ‘;
163         break;
164       default:
165         speechText = ‘Perdona, ha habido un error al crear el recordatorio. ‘;
166     }
167     speechText +=
168       "Si no sabes como continuar intenta pedir ayuda. Si quieres salir solo
    di para. Que quieres hacer? ";
169   }
170
171   return handlerInput.responseBuilder
172     .speak(speechText)
173     .reprompt(speechText)
174     .getResponse();
175 },
176 };
177
178 const RegistrarEdadIntentHandler = {
179  canHandle(handlerInput) {
180   return (
181     Alexa.getRequestType(handlerInput.requestEnvelope) === "IntentRequest" &&
182     (Alexa.getIntentName(handlerInput.requestEnvelope) ===
183       "RegistrarEdadIntent" ||
184       Alexa.getIntentName(handlerInput.requestEnvelope) ===
185         "RespuestaIntent") &&
186     test === "edad"
187   );
188 },
189
190  handle(handlerInput) {
191   test = "edad";
192   const { attributesManager, serviceClientFactory, requestEnvelope } =
193     handlerInput;
```

```
194     const sessionAttributes = attributesManager.getSessionAttributes();
195     var anios;
196     if (handlerInput.requestEnvelope.request.intent.slots.anios) {
197       anios = parseInt(
198         handlerInput.requestEnvelope.request.intent.slots.anios.value,
199         10
200       );
201     } else {
202       anios = parseInt(
203         handlerInput.requestEnvelope.request.intent.slots.numero.value,
204         10
205       );
206     }
207     sessionAttributes["edad"] = anios;
208     let speakOutput = `Tienes ${anios} aos, es correcto?`;
209     util.addAPLTexto(`Tienes ${anios} aos?`, "", handlerInput);
210
211     return handlerInput.responseBuilder
212       .speak(speakOutput)
213       .reprompt()
214       .getResponse();
215   },
216 };
217
218 const RegistrarAnimoIntentHandler = {
219   async canHandle(handlerInput) {
220     return (
221       Alexa.getRequestType(handlerInput.requestEnvelope) === "IntentRequest" &&
222       Alexa.getIntentName(handlerInput.requestEnvelope) === "RespuestaIntent" &&
223       test === "animo"
224     );
225   },
226
227   handle(handlerInput) {
228     const { attributesManager, serviceClientFactory, requestEnvelope } =
229       handlerInput;
230     const sessionAttributes = attributesManager.getSessionAttributes();
231     var numero = parseInt(
```

```
232       handlerInput.requestEnvelope.request.intent.slots.numero.value,
233       10
234     );
235     const name = sessionAttributes["name"] || "";
236     let speakOutput = "";
237     const payload = {
238             idSkill: "d344f58284064bf80c2851dfe224c8ee",
239             usuario: name,
240             enunciado: 'Test animo',
241             respuesta:
    handlerInput.requestEnvelope.request.intent.slots.numero.value,
242             tipoPregunta: '1'
243         };
244     speakOutput = util.peticiones(payload, true)
245     if (numero === 0 || numero === 1) {
246       speakOutput = `si estas muy deprimido o tienes pensamientos suicidas, te
    recomiendo llamar al 024,
247       Linea de atencion a la conducta suicida o al 717 003 717, telefono de la
    esperanza. Quieres escuchar un consejo?`;
248       util.addAPLTexto("Quieres escuchar un consejo?", " telefono de la
    esperanza: 717 003 717", handlerInput);
249     } else {
250       speakOutput += "Quieres escuchar un consejo?";
251       countSi = 0;
252           util.addAPLTexto("Quieres escuchar un consejo?", " ", handlerInput);
253     }
254
255     return handlerInput.responseBuilder
256       .speak(speakOutput)
257       .reprompt()
258       .getResponse();
259   },
260 };
261
262 const RumiacionIntentHandler = {
263   canHandle(handlerInput) {
264     return (
265       Alexa.getRequestType(handlerInput.requestEnvelope) === "IntentRequest" &&
```

```
266       Alexa.getIntentName(handlerInput.requestEnvelope) === "RumiacionIntent"
267     );
268   },
269   handle(handlerInput) {
270     const sessionAttributes =
271       handlerInput.attributesManager.getSessionAttributes();
272     test = "Rumiacion";
273     currentQuestion = 0;
274     answerCount = 0;
275     const speakOutput = `contesta uno, dos, tres o cuatro segun lo identificado
    que te sientas con las afirmaciones.
    ${questionsRumiacion[currentQuestion].text}`;
276     util.addAPLTexto(questionsRumiacion[currentQuestion].text, "contesta 1, 2,
    3 o 4", handlerInput);
277
278     return handlerInput.responseBuilder
279       .speak(speakOutput)
280       .reprompt(
281       )
282       .getResponse();
283   },
284 };
285
286 const AsociacionesIntentHandler = {
287   canHandle(handlerInput) {
288     return (
289       Alexa.getRequestType(handlerInput.requestEnvelope) === "IntentRequest" &&
290       Alexa.getIntentName(handlerInput.requestEnvelope) === "AsociacionesIntent"
291     );
292   },
293   handle(handlerInput) {
294     const { attributesManager, serviceClientFactory, requestEnvelope } =
295       handlerInput;
296     const sessionAttributes =
297       handlerInput.attributesManager.getSessionAttributes();
298     const name = sessionAttributes["name"] || "";
299     const payload = {
300           idSkill: "d344f58284064bf80c2851dfe224c8ee",
```

```
301             usuario: name,
302             enunciado: 'Asociaciones',
303             respuesta: '',
304             tipoPregunta: '1'
305         };
306     util.peticiones(payload, false);
307     let randomNum = Math.floor(Math.random() * 7) + 1;
308     var speakOutput = `hola, ${name}, aqui tienes una lista de asociaciones
    amplias: `;
309     speakOutput += asociacionesAmplias[randomNum].text;
310     const lista = asociacionesAmplias[randomNum].text;
311
312     util.addAPLTexto(lista, "", handlerInput);
313
314     return (
315       handlerInput.responseBuilder
316         .speak(speakOutput)
317         .reprompt()
318         .getResponse()
319     );
320   },
321 };
322
323 const MusicaIntentHandler = {
324   canHandle(handlerInput) {
325     return (
326       Alexa.getRequestType(handlerInput.requestEnvelope) === "IntentRequest" &&
327       Alexa.getIntentName(handlerInput.requestEnvelope) === "MusicaIntent"
328     );
329   },
330   handle(handlerInput) {
331     const { attributesManager, serviceClientFactory, requestEnvelope } =
332       handlerInput;
333     const sessionAttributes =
334       handlerInput.attributesManager.getSessionAttributes();
335     const name = sessionAttributes["name"] || "";
336     const payload = {
337             idSkill: "d344f58284064bf80c2851dfe224c8ee",
```

```
338             usuario: name,
339             enunciado: 'Musica',
340             respuesta: '',
341             tipoPregunta: '1'
342         };
343     util.peticiones(payload, false);
344     util.addAPLMusica(handlerInput);
345 
346     return (
347       handlerInput.responseBuilder
348         //.speak(speakOutput)
349         .reprompt("")
350         .getResponse()
351     );
352   },
353 };
354 const TestIntentHandler = {
355   canHandle(handlerInput) {
356     return (
357       Alexa.getRequestType(handlerInput.requestEnvelope) === "IntentRequest" &&
358       Alexa.getIntentName(handlerInput.requestEnvelope) === "TestIntent"
359     );
360   },
361   handle(handlerInput) {
362     const sessionAttributes =
363       handlerInput.attributesManager.getSessionAttributes();
364     let speakOutput = "";
365     if (sessionAttributes["edad"] > 65) {
366       test = "GDS";
367       estadoActual = estados.EMPEZANDO;
368       currentQuestion = 0;
369       answerCount = 0;
370       speakOutput+=
371         "voy a hacerte una serie de preguntas a las que debes contestar con si o
    no, entendido?";
372     } else if (sessionAttributes["edad"] < 15) {
373       test = "CDI";
374       currentQuestion = 0;
```

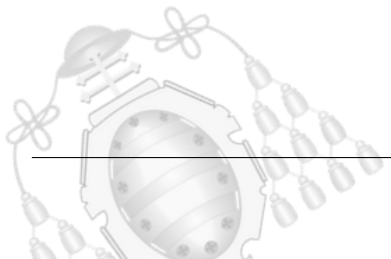
```
375       answerCount = 0;
376       speakOutput = `contesta uno, dos o tres segun lo identificado que te
    sientas con las afirmaciones. ${questionsCDI[currentQuestion].text}`;
377       util.addAPLTexto(questionsCDI[currentQuestion].text, "contesta 1, 2 o 3",
    handlerInput);
378     }
379     else {
380       test = "Beck";
381       currentQuestion = 0;
382       answerCount = 0;
383       speakOutput = `contesta uno, dos, tres o cuatro segun lo identificado que
    te sientas con las afirmaciones. ${questionsBeck[currentQuestion].text}`;
384       util.addAPLTexto(questionsBeck[currentQuestion].text, "contesta 1, 2, 3 o
    4", handlerInput);
385     }
386     return handlerInput.responseBuilder
387       .speak(speakOutput)
388       .reprompt(
389       )
390       .getResponse();
391   },
392 };
393
394 const RespuestaIntentHandler = {
395   canHandle(handlerInput) {
396     return (
397       Alexa.getRequestType(handlerInput.requestEnvelope) === "IntentRequest" &&
398       Alexa.getIntentName(handlerInput.requestEnvelope) === "RespuestaIntent" &&
399       (test === "Beck" || test === "Rumiacion" || test === "CDI")
400     );
401   },
402   handle(handlerInput) {
403     const { attributesManager, serviceClientFactory, requestEnvelope } =
404       handlerInput;
405     const sessionAttributes = attributesManager.getSessionAttributes();
406     var numero = parseInt(
407       handlerInput.requestEnvelope.request.intent.slots.numero.value,
408       10
```

```
409     );
410     var speakOutput = "";
411     const name = sessionAttributes["name"] || "";
412     var nivel = "";
413     if (numero > 4) {
414       speakOutput = ‘por favor, responde 1, 2, 3 o 4.‘;
415     } else {
416       answerCount += numero - 1;
417       currentQuestion++;
418       if (test === "Beck") {
419         if (currentQuestion < questionsBeck.length) {
420           speakOutput = questionsBeck[currentQuestion].text;
421            util.addAPLTexto(speakOutput, "contesta 1, 2, 3 o 4", handlerInput);
422         } else {
423             const payload = {
424             idSkill: "d344f58284064bf80c2851dfe224c8ee",
425             usuario: name,
426             enunciado: ’Test depresion’,
427             respuesta: answerCount.toString(),
428             tipoPregunta: ’1’
429         };
430             speakOutput = util.peticiones(payload, true);
431             if (answerCount > 20){
432                 test = "";
433                 countSi = 0;
434                 speakOutput += ‘ Parece que tu animo es bajo, quieres que te diga
    algun consejo para mejorarlo?‘;
435             }
436             else
437                 speakOutput += ‘ Parece que tu animo es bueno, que quieres hacer?
    Puedes pedir ayuda para saber las opciones.‘;
438             answerCount = 0;
439         }
440       } else if (test === "Rumiacion") {
441         if (currentQuestion < questionsRumiacion.length) {
442           speakOutput = questionsRumiacion[currentQuestion].text;
443            util.addAPLTexto(speakOutput, "contesta 1, 2, 3 o 4", handlerInput);
444         } else {
```

```
445             const payload = {
446             idSkill: "d344f58284064bf80c2851dfe224c8ee",
447             usuario: name,
448             enunciado: 'Test Rumiacion',
449             respuesta: answerCount.toString(),
450             tipoPregunta: '1'
451         };
452             speakOutput = util.peticiones(payload, true);
453           if (answerCount > 33) {
454             speakOutput += `Parece que te encuentras en un estado mental
    rumiativo. Una forma de cambiar esto es escuchar listas de asociaciones
    amplias, como la que sigue.`;
455             const randomNum = Math.floor(Math.random() * 7) + 1;
456             speakOutput += asociacionesAmplias[randomNum].text;
457             speakOutput +=
458               'Prueba a decir: "asociaciones amplias" si quieres escuchar mas';
459           } else {
460             speakOutput += `Parece que tu estado mental no es rumiativo. No
    obstante, si te encuentras repimido, te recomiendo acudir a un profesional`;
461           }
462           answerCount = 0;
463         }
464       } else if (test === "CDI") {
465         if (numero > 3) {
466           speakOutput = `por favor, responde 1, 2 o 3.`;
467         } else {
468           if (currentQuestion < questionsCDI.length) {
469             speakOutput = questionsCDI[currentQuestion].text;
470             util.addAPLTexto(speakOutput, "contesta 1, 2 o 3", handlerInput);
471           } else {
472             const payload = {
473             idSkill: "d344f58284064bf80c2851dfe224c8ee",
474             usuario: name,
475             enunciado: 'Test depresion',
476             respuesta: answerCount.toString(),
477             tipoPregunta: '1'
478         };
479             speakOutput = util.peticiones(payload, true);
```

```
480           if (answerCount > 30){
481               test = "";
482               countSi = 0;
483               speakOutput += ‘Parece que tu animo es bajo, quieres que te diga
    algun consejo para mejorarlo?‘;
484           }
485           else
486               speakOutput += ‘Parece que tu animo es bueno, que quieres hacer?
    Puedes pedir ayuda para saber las opciones.‘;
487           answerCount = 0;
488         }
489       }
490     }
491   }
492   return handlerInput.responseBuilder
493     .speak(speakOutput)
494     .reprompt(
495     )
496     .getResponse();
497  },
498 };
499
500 const SiIntentHandler = {
501  canHandle(handlerInput) {
502    return (
503      Alexa.getRequestType(handlerInput.requestEnvelope) === "IntentRequest" &&
504      Alexa.getIntentName(handlerInput.requestEnvelope) === "AMAZON.YesIntent"
505    );
506  },
507  handle(handlerInput) {
508    const { attributesManager, serviceClientFactory, requestEnvelope } =
509      handlerInput;
510    const sessionAttributes = attributesManager.getSessionAttributes();
511    const name = sessionAttributes["name"] || "";
512    var speakOutput = "";
513    if (test === "GDS") {
514      switch (estadoActual) {
515        case estados.EMPEZANDO:
```

```
516           speakOutput = questionsGDS[currentQuestion].text;
517           estadoActual = estados.PREGUNTANDO;
518           break;
519         case estados.PREGUNTANDO:
520           if (questionsGDS[currentQuestion].yes === true) answerCount++;
521
522           currentQuestion++;
523           if (currentQuestion < questionsGDS.length) {
524             speakOutput = questionsGDS[currentQuestion].text;
525           } else{
526             const payload = {
527             idSkill: "d344f58284064bf80c2851dfe224c8ee",
528             usuario: name,
529             enunciado: 'Test GDS',
530             respuesta: answerCount.toString(),
531             tipoPregunta: '1'
532         };
533             speakOutput = util.peticiones(payload, true);
534                 if (answerCount > 15){
535                     test = "";
536                     countSi = 0;
537                     speakOutput += `Parece que tu animo es bajo, quieres que te
    diga algun consejo para mejorarlo?`;
538                 }
539                 else
540                     speakOutput += `Parece que tu animo es bueno, que quieres
    hacer? Puedes pedir ayuda para saber las opciones.`;
541           }
542           break;
543         default:
544           speakOutput = "ha habido un error";
545       }
546       util.addAPLTexto(speakOutput, "Test GDS", handlerInput);
547     } else if(test === "edad"){
548     if (sessionAttributes["reminderId"] === null ||
    sessionAttributes["reminderId"] === undefined) {
```



```
549      speakOutput = ‘Tu edad ha sido registrada correctamente. Ahora vamos a
    crear unos recordatorios para que no te olvides de registrar tu animo. Por
    favor, di: crear recordatorio.‘;
550    } else {
551      speakOutput =
552        "Tu edad ha sido registrada correctamente. Del uno al diez, como te
    encuentras hoy?";
553      test = "animo";
554    }
555    }
556    else{
557      if (countSi === 0) {
558        preguntasCount = Math.floor(Math.random() * 8) + 1
559        speakOutput = preguntas[preguntasCount].text;
560        countSi++;
561      } else {
562        switch (preguntasCount) {
563          case 4:
564            util.addAPLMusica(handlerInput);
565            break;
566          case 5:
567            return AsociacionesIntentHandler.handle(handlerInput);
568          case 6:
569            return ChistesIntentHandler.handle(handlerInput);
570          case 7:
571            return FrasesIntentHandler.handle(handlerInput);
572          case 8:
573            return TestIntentHandler.handle(handlerInput);
574          default:
575            let speakOutput = "Genial, nos vemos luego!";
576            const payload = {
577                idSkill: "d344f58284064bf80c2851dfe224c8ee",
578                usuario: name,
579                enunciado: preguntas[preguntasCount].text,
580                respuesta: ’si’,
581                tipoPregunta: ’1’
582            };
583            util.peticiones(payload, false);
```

```
584             return handlerInput.responseBuilder
585               .speak(speakOutput)
586               .getResponse();
587           }
588       }
589       util.addAPLTexto(speakOutput, "", handlerInput);
590     }
591
592     return handlerInput.responseBuilder
593       .speak(speakOutput)
594       .reprompt(speakOutput)
595       .getResponse();
596   },
597 };
598
599 const NoIntentHandler = {
600   canHandle(handlerInput) {
601     return (
602       Alexa.getRequestType(handlerInput.requestEnvelope) === "IntentRequest" &&
603       Alexa.getIntentName(handlerInput.requestEnvelope) === "AMAZON.NoIntent"
604     );
605   },
606   handle(handlerInput) {
607     const { attributesManager, serviceClientFactory, requestEnvelope } =
608       handlerInput;
609     const sessionAttributes = attributesManager.getSessionAttributes();
610     var speakOutput = "";
611     if (test === "GDS") {
612     if (estadoActual === estados.EMPEZANDO){
613         return TestIntentHandler.handle(handlerInput);
614     }
615     else{
616       if (questionsGDS[currentQuestion].yes === false) answerCount++;
617
618       currentQuestion++;
619       if (currentQuestion < questionsGDS.length) {
620         speakOutput = questionsGDS[currentQuestion].text;
621       } else{
```

```
622             const name = sessionAttributes["name"] || "";
623             const payload = {
624             idSkill: "d344f58284064bf80c2851dfe224c8ee",
625             usuario: name,
626             enunciado: 'Test GDS',
627             respuesta: answerCount.toString(),
628             tipoPregunta: '1'
629         };
630             if (answerCount > 15){
631                 test = "";
632                 countSi = 0;
633                 speakOutput = `Parece que tu animo es bajo, quieres que te diga
    algun consejo para mejorarlo?`;
634             }
635             else
636                 speakOutput = `Parece que tu animo es bueno, que quieres hacer?
    Puedes pedir ayuda para saber las opciones.`;
637       }
638
639       util.addAPLTexto(speakOutput, "Test GDS", handlerInput);}
640     } else if(test === "edad"){
641         speakOutput = "cuantos  aos  tienes?"
642     }
643     else{
644         return HelpIntentHandler.handle(handlerInput);
645     }
646
647     return handlerInput.responseBuilder
648       .speak(speakOutput)
649       .reprompt(speakOutput)
650       .getResponse();
651   },
652 };
653
654 const ChistesIntentHandler = {
655   canHandle(handlerInput) {
656     return (
657       Alexa.getRequestType(handlerInput.requestEnvelope) === "IntentRequest" &&
```

```
658       Alexa.getIntentName(handlerInput.requestEnvelope) === "ChistesIntent"
659     );
660   },
661   handle(handlerInput) {
662     let randomNum = Math.floor(Math.random() * 33) + 1;
663     const speakOutput = chistes[randomNum].text;
664     const { attributesManager, serviceClientFactory, requestEnvelope } =
    handlerInput;
665     const sessionAttributes = attributesManager.getSessionAttributes();
666     const name = sessionAttributes["name"] || "";
667     const payload = {
668             idSkill: "d344f58284064bf80c2851dfe224c8ee",
669             usuario: name,
670             enunciado: 'Chistes',
671             respuesta: '',
672             tipoPregunta: '1'
673         };
674     util.peticiones(payload, false);
675
676     return (
677       handlerInput.responseBuilder
678         .speak(speakOutput)
679         .reprompt()
680         .getResponse()
681     );
682   },
683 };
684
685 const FrasesIntentHandler = {
686   canHandle(handlerInput) {
687     return (
688       Alexa.getRequestType(handlerInput.requestEnvelope) === "IntentRequest" &&
689       Alexa.getIntentName(handlerInput.requestEnvelope) === "FrasesIntent"
690     );
691   },
692   handle(handlerInput) {
693     let randomNum = Math.floor(Math.random() * 41) + 1;
694     const speakOutput = frases[randomNum].text;
```

```
695         const { attributesManager, serviceClientFactory, requestEnvelope } =
    handlerInput;
696     const sessionAttributes = attributesManager.getSessionAttributes();
697     const name = sessionAttributes["name"] || "";
698     const payload = {
699             idSkill: "d344f58284064bf80c2851dfe224c8ee",
700             usuario: name,
701             enunciado: 'Frases',
702             respuesta: '',
703             tipoPregunta: '1'
704         };
705     util.peticiones(payload, false);
706 
707     return (
708       handlerInput.responseBuilder
709         .speak(speakOutput)
710         .reprompt(speakOutput)
711         .getResponse()
712     );
713   },
714 };
715 
716 const ReiniciarHandler = {
717   canHandle(handlerInput) {
718     return (
719       Alexa.getRequestType(handlerInput.requestEnvelope) === "IntentRequest" &&
720       Alexa.getIntentName(handlerInput.requestEnvelope) === "ReiniciarIntent"
721     );
722   },
723   handle(handlerInput) {
724     const speakOutput = "Se ha reiniciado el test";
725 
726     return (
727       handlerInput.responseBuilder
728         .speak(speakOutput)
729         .reprompt()
730         .getResponse()
731     );
```

```
732   },
733 };
734 const HelpIntentHandler = {
735   canHandle(handlerInput) {
736     return (
737       Alexa.getRequestType(handlerInput.requestEnvelope) === "IntentRequest" &&
738       Alexa.getIntentName(handlerInput.requestEnvelope) === "AMAZON.HelpIntent"
739     );
740   },
741   handle(handlerInput) {
742     const speakOutput =
743       "Puedes hacer un test para medir tu estado de animo o rumiacion, escuchar
    musica relajante, un chiste, una frase motivacional o alguna de mis listas de
    asociaciones amplias que quieres hacer?";
744     util.addAPLTexto("Que quieres hacer?", "", handlerInput);
745     return (
746       handlerInput.responseBuilder
747         .speak(speakOutput)
748         .reprompt(speakOutput)
749         .getResponse()
750     );
751   },
752 };
753
754 const CancelAndStopIntentHandler = {
755   canHandle(handlerInput) {
756     return (
757       Alexa.getRequestType(handlerInput.requestEnvelope) === "IntentRequest" &&
758       (Alexa.getIntentName(handlerInput.requestEnvelope) ===
759         "AMAZON.CancelIntent" ||
760         Alexa.getIntentName(handlerInput.requestEnvelope) ===
761           "AMAZON.StopIntent")
762     );
763   },
764   handle(handlerInput) {
765     const sessionAttributes =
766       handlerInput.attributesManager.getSessionAttributes();
767     const name = sessionAttributes["name"] || "";
```

```
768     const speakOutput = `Hasta luego, ${name}!`;
769
770     return handlerInput.responseBuilder
771       .speak(speakOutput)
772       .withShouldEndSession(true) // session can remain open if APL doc was
    rendered
773       .getResponse();
774   },
775 };
776 const FallbackIntentHandler = {
777   canHandle(handlerInput) {
778     return (
779       Alexa.getRequestType(handlerInput.requestEnvelope) === "IntentRequest" &&
780       Alexa.getIntentName(handlerInput.requestEnvelope) ===
781         "AMAZON.FallbackIntent"
782     );
783   },
784   handle(handlerInput) {
785     const speakOutput = "Sorry, I don't know about that. Please try again.";
786
787     return handlerInput.responseBuilder
788       .speak(speakOutput)
789       .reprompt(speakOutput)
790       .getResponse();
791   },
792 };
793
794 const SessionEndedRequestHandler = {
795   canHandle(handlerInput) {
796     return (
797       Alexa.getRequestType(handlerInput.requestEnvelope) ===
798       "SessionEndedRequest"
799     );
800   },
801   handle(handlerInput) {
802     console.log(
803       `~~~~ Session ended: ${JSON.stringify(handlerInput.requestEnvelope)}`
804     );
```

```
805    // Any cleanup logic goes here.
806    return handlerInput.responseBuilder.getResponse(); // notice we send an
    empty response
807   },
808 };
809
810 const IntentReflectorHandler = {
811   canHandle(handlerInput) {
812     return (
813       Alexa.getRequestType(handlerInput.requestEnvelope) === "IntentRequest"
814     );
815   },
816   handle(handlerInput) {
817     const intentName = Alexa.getIntentName(handlerInput.requestEnvelope);
818     const speakOutput = `You just triggered ${intentName}`;
819
820     return (
821       handlerInput.responseBuilder
822         .speak(speakOutput)
823         //.reprompt('add a reprompt if you want to keep the session open for the
    user to respond')
824         .getResponse()
825     );
826   },
827 };
828
829 const ErrorHandler = {
830   canHandle() {
831     return true;
832   },
833   handle(handlerInput, error) {
834     const speakOutput =
835       "Sorry, I had trouble doing what you asked. Please try again.";
836     console.log(`~~~~ Error handled: ${JSON.stringify(error)}`);
837
838     return handlerInput.responseBuilder
839       .speak(speakOutput)
840       .reprompt(speakOutput)
```

```
841       .getResponse();
842   },
843 };
844
845 exports.handler = Alexa.SkillBuilders.custom()
846   .addRequestHandlers(
847     ChistesIntentHandler,
848     FrasesIntentHandler,
849     LaunchRequestHandler,
850     AsociacionesIntentHandler,
851     SiIntentHandler,
852     NoIntentHandler,
853     MusicaIntentHandler,
854     TestIntentHandler,
855     RumiacionIntentHandler,
856     RespuestaIntentHandler,
857     RecordatoriosIntentHandler,
858     RegistrarAnimoIntentHandler,
859     RegistrarEdadIntentHandler,
860     ReiniciarHandler,
861     HelpIntentHandler,
862     CancelAndStopIntentHandler,
863     FallbackIntentHandler,
864     SessionEndedRequestHandler,
865     IntentReflectorHandler
866   )
867   .addRequestInterceptors(
868     interceptors.LoggingRequestInterceptor,
869     interceptors.LoadNameRequestInterceptor,
870     interceptors.LoadAttributesRequestInterceptor,
871     interceptors.LoadTimezoneRequestInterceptor
872   )
873   .addResponseInterceptors(
874     interceptors.LoggingResponseInterceptor,
875     interceptors.SaveAttributesResponseInterceptor
876   )
877   .withPersistenceAdapter(
878     new ddbAdapter.DynamoDbPersistenceAdapter({
```

```
879       tableName: process.env.DYNAMODB_PERSISTENCE_TABLE_NAME,
880       createTable: false,
881       dynamoDBClient: new AWS.DynamoDB({
882         apiVersion: "latest",
883         region: process.env.DYNAMODB_PERSISTENCE_REGION,
884       }),
885     })
886   )
887   .withApiClient(new Alexa.DefaultApiClient())
888   .addErrorHandlers(ErrorHandler)
889   .withCustomUserAgent("sample/hello-world/v1.2")
890   .lambda();
891
```

Anexo H. Código aplicación web

.1.- Componente del gráfico de líneas

```
1  import { Component, ViewChild } from "@angular/core";
2  import { ChartDataSets, ChartOptions } from "chart.js";
3  import { Label, Color, BaseChartDirective } from "ng2-charts";
4  import { GetDataService } from "../shared/get-data.service";
5  import { dataResponse, datos, restultadoTest} from "../shared/lineaClases";
6  import * as moment from "moment";
7
8  @Component({
9    selector: "app-linea",
10   templateUrl: "./linea.component.html",
11   styleUrls: ["./linea.component.css"],
12 })
13 export class LineaComponent {
14   more: boolean = false;
15
16   public lineChartData: ChartDataSets[] = [
17     { data: [], label: "Nivel de nimo", yAxisID: "y-axis-1" },
18     { data: [], label: "Test depresin" },
19     { data: [], label: "Test rumiacin" },
```

```
20  ];
21  public testRumiacion: restultadoTest[] = [];
22  public testDepresion: restultadoTest[] = [];
23  public testAnimo: restultadoTest[] = [];
24  public datosAnimo: datos[] = [];
25  public datosDep: datos[] = [];
26  public datosRum: datos[] = [];
27
28  //public lineChartData: ChartDataSets[] = []
29  public lineChartLabels: Label[] = [];
30
31  public lineChartOptions: ChartOptions & { annotation: any } = {
32    responsive: true,
33    scales: {
34      // We use this empty structure as a placeholder for dynamic theming.
35      xAxes: [{}],
36      yAxes: [
37        {
38          id: "y-axis-0",
39          position: "left",
40        },
41        {
42          id: "y-axis-1",
43          position: "right",
44          gridLines: {
45            color: "rgba(255,0,0,0.3)",
46          },
47          ticks: {
48            fontColor: "red",
49          },
50        },
51      ],
52    },
53    annotation: {
54      annotations: [
55        {
56          type: "line",
57          mode: "vertical",
```

```
58          scaleID: "x-axis-0",
59          value: "March",
60          borderColor: "orange",
61          borderWidth: 2,
62          label: {
63            enabled: true,
64            fontColor: "orange",
65            content: "LineAnno",
66          },
67        },
68      ],
69    },
70  };
71
72  public lineChartColors: Color[] = [
73    {
74      // grey
75      backgroundColor: "rgba(0, 255, 0, 0.3)",
76      borderColor: "green",
77      pointBackgroundColor: "rgba(148,159,177,1)",
78      pointBorderColor: "#fff",
79      pointHoverBackgroundColor: "#fff",
80      pointHoverBorderColor: "rgba(148,159,177,0.8)",
81    },
82    {
83      // dark grey
84      backgroundColor: "rgba(77,83,96,0.2)",
85      borderColor: "rgba(77,83,96,1)",
86      pointBackgroundColor: "rgba(77,83,96,1)",
87      pointBorderColor: "#fff",
88      pointHoverBackgroundColor: "#fff",
89      pointHoverBorderColor: "rgba(77,83,96,1)",
90    },
91    {
92      // red
93      backgroundColor: "rgba(255,0,0,0.3)",
94      borderColor: "red",
95      pointBackgroundColor: "rgba(148,159,177,1)",
```

```
96      pointBorderColor: "#fff",
97      pointHoverBackgroundColor: "#fff",
98      pointHoverBorderColor: "rgba(148,159,177,0.8)",
99    },
100 ];
101 public lineChartLegend = true;
102 public lineChartType = "line";
103 private user;
104 private fechaActual = new Date();
105
106 @ViewChild(BaseChartDirective, { static: true }) chart: BaseChartDirective;
107
108 constructor(private getDataService: GetDataService) {}
109
110 ngOnInit() {
111   this.user = JSON.parse(localStorage.getItem('usuario'));
112   this.sieteDias();
113 }
114
115 // events
116 public chartClicked({
117   event,
118   active,
119 }: {
120   event: MouseEvent;
121   active: {}[];
122 }): void {
123   console.log(event, active);
124 }
125
126 public chartHovered({
127   event,
128   active,
129 }: {
130   event: MouseEvent;
131   active: {}[];
132 }): void {
133   console.log(event, active);
```

```
134 }
135
136 public sieteDias() {
137   this.getChartLabel(7);
138   let fechaInicio = new Date(this.fechaActual);
139   fechaInicio.setDate(this.fechaActual.getDate() - 7);
140
141   let fechaFormateada = this.getDateFormat(fechaInicio);
142   this.request(fechaFormateada);
143 }
144
145 public quinceDias() {
146   let fechaInicio = new Date(this.fechaActual);
147   fechaInicio.setDate(this.fechaActual.getDate() - 15);
148
149   let fechaFormateada = this.getDateFormat(fechaInicio);
150   this.request(fechaFormateada);
151   this.getChartLabel(15);
152 }
153
154 public treintaDias() {
155   let fechaInicio = new Date(this.fechaActual);
156   fechaInicio.setDate(this.fechaActual.getDate() - 30);
157
158   let fechaFormateada = this.getDateFormat(fechaInicio);
159   this.request(fechaFormateada);
160   this.getChartLabel(30);
161 }
162
163 public getDateFormat(date){
164   let year = date.getFullYear();
165   let month = (date.getMonth() + 1).toString().padStart(2, '0');
166   let day = date.getDate().toString().padStart(2, '0');
167
168   let fechaFormateada = `${year}-${month}-${day}`;
169   return fechaFormateada;
170 }
171
```

```
172 public getChartLabel(days){
173   this.lineChartLabels = [];
174   let fechaInicio = new Date(this.fechaActual);
175   for(let i=(days-1);i>=0;i--){
176     fechaInicio = new Date(this.fechaActual);
177     fechaInicio.setDate(this.fechaActual.getDate() - i);
178     let fechaFormateada = this.getDateFormat(fechaInicio);
179     this.lineChartLabels.push(fechaFormateada);
180   }
181 }
182
183 public calculaMedia(array: datos[]){
184   array.forEach(e => {
185     let acum = e.valor.reduce((acumulador, valor) => acumulador + valor, 0);
186     e.media = acum/e.valor.length;
187   });
188 }
189
190 public rellenaDatos(){
191   this.lineChartData = [
192     { data: [], label: "Nivel de nimo", yAxisID: "y-axis-1" },
193     { data: [], label: "Test depresin" },
194     { data: [], label: "Test rumiacin" },
195   ];
196   this.lineChartLabels.forEach(day => {
197     let datoA = this.datosAnimo.find(dato => dato.dia === day);
198     if(datoA){
199       this.lineChartData[0].data.push(datoA.media)
200     }else{
201       this.lineChartData[0].data.push(0);
202     }
203     let datoD = this.datosDep.find(dato => dato.dia === day);
204     if(datoD){
205       this.lineChartData[1].data.push(datoD.media)
206     }else{
207       this.lineChartData[1].data.push(0);
208     }
209     let datoR = this.datosRum.find(dato => dato.dia === day);
```

```
210     if(datoR){
211       this.lineChartData[2].data.push(datoR.media)
212     }else{
213       this.lineChartData[2].data.push(0);
214     }
215   })
216 }
217
218 public request(fechaFormateada){
219   this.datosAnimo = [];
220   this.datosDep = [];
221   this.datosRum = [];
222   this.getDataService.makeHttpGetRequest(this.user, fechaFormateada).subscribe(
223     (response: dataResponse) => {
224       for (let clave in response) {
225         if (response[clave].enunciado === 'Test nimo') {
226           let valorEncontrado = false;
227           this.datosAnimo.forEach(e =>{
228             if(e.dia === response[clave].fecha.split(" ")[0]){
229               e.valor.push(parseInt(response[clave].respuesta));
230               valorEncontrado = true;
231             }
232           });
233           if(valorEncontrado === false){
234             let dia = response[clave].fecha.split(" ")[0];
235             let valor = [];
236             valor.push(parseInt(response[clave].respuesta));
237             this.datosAnimo.push({dia: dia, valor: valor});
238           }
239         }else if (response[clave].enunciado === 'Test depresin') {
240           let valorEncontrado = false;
241           this.datosDep.forEach(e =>{
242             if(e.dia === response[clave].fecha.split(" ")[0]){
243               e.valor.push(parseInt(response[clave].respuesta));
244               valorEncontrado = true;
245             }
246           });
247           if(valorEncontrado === false){
```

```
248             let dia = response[clave].fecha.split(" ")[0];
249             let valor = [];
250             valor.push(parseInt(response[clave].respuesta));
251             this.datosDep.push({dia: dia, valor: valor});
252           }
253         }else if (response[clave].enunciado === 'Test Rumiacin') {
254           let valorEncontrado = false;
255           this.datosRum.forEach(e =>{
256             if(e.dia === response[clave].fecha.split(" ")[0]){
257               e.valor.push(parseInt(response[clave].respuesta));
258               valorEncontrado = true;
259             }
260           });
261           if(valorEncontrado === false){
262             let dia = response[clave].fecha.split(" ")[0];
263             let valor = [];
264             valor.push(parseInt(response[clave].respuesta));
265             this.datosRum.push({dia: dia, valor: valor});
266           }
267         }
268       }
269       this.calculaMedia(this.datosAnimo);
270       this.calculaMedia(this.datosDep);
271       this.calculaMedia(this.datosRum);
272       this.rellenaDatos();
273     },
274     (error) => {
275       console.error("Error:", error);
276     }
277   );
278 }
279}
280
```

.2.- Vista del gráfico de líneas

```
1 <h3 class="m-3">Grfico de Lneas</h3>
2 <hr />
```

```
3 <div class="row">
4   <div class="col-sm-6 mx-auto">
5     <div>
6       <canvas
7         baseChart
8         width="400"
9         height="400"
10        [datasets]="lineChartData"
11        [labels]="lineChartLabels"
12        [options]="lineChartOptions"
13        [colors]="lineChartColors"
14        [legend]="lineChartLegend"
15        [chartType]="lineChartType"
16        (chartHover)="chartHovered($event)"
17        (chartClick)="chartClicked($event)"
18      ></canvas>
19    </div>
20 </div>
21 <div class="col-sm-6" *ngIf="more">
22   <table class="table table-responsive table-condensed">
23     <tr>
24       <th *ngFor="let label of lineChartLabels">{{ label }}</th>
25     </tr>
26     <tr *ngFor="let d of lineChartData; let i = index" [class]="'line-' + i">
27       <td *ngFor="let label of lineChartLabels; let j = index">
28         {{ d && d.data[j] }}
29       </td>
30     </tr>
31   </table>
32   <button mat-button mat-raised-button color="primary" (click)="sieteDias()">
33     ltimos  7  das
34   </button>
35   <button mat-button mat-raised-button color="primary" (click)="quinceDias()">
36     ltimos  15  das
37   </button>
38   <button
39     mat-button
40     mat-raised-button
```

```
41      color="primary"
42      (click)="treintaDias()"
43    >
44       ltimo  mes
45    </button>
46
47    <p></p>
48    <button
49      mat-button
50      mat-raised-button
51      class="btn btn-4 btn-block mx-auto w-25"
52      (click)="more = false"
53    >
54      Cerrar
55    </button>
56  </div>
57  <div class="mas" *ngIf="more === false">
58    <button
59      mat-button
60      mat-raised-button
61      class="btn btn-4 btn-block"
62      (click)="more = true"
63    >
64      Mostar  ms
65    </button>
66  </div>
67</div>
68
```

.3.- Componente del gráfico de barras

```
1  import { Component, OnInit } from '@angular/core';
2  import { ChartOptions, ChartType, ChartDataSets } from 'chart.js';
3  import { Label } from 'ng2-charts';
4  import { GetDataService } from '../shared/get-data.service';
5  import { dataResponse } from '../shared/lineaClases';
6
7  @Component({
```

```
8   selector: 'app-barra',
9   templateUrl: './barra.component.html',
10  styleUrls: ['./barra.component.css']
11 })
12 export class BarraComponent implements OnInit{
13
14  constructor(private getDataService: GetDataService){};
15  private user;
16  private fechaActual = new Date();
17
18  ngOnInit(): void {
19    this.user = JSON.parse(localStorage.getItem('usuario'));
20    this.getValues7days();
21    this.getValues15days();
22    this.getValues30days();
23  }
24
25  public barChartOptions: ChartOptions = {
26    responsive: true,
27    // We use these empty structures as placeholders for dynamic theming.
28    scales: { xAxes: [{}], yAxes: [{}] },
29    plugins: {
30      datalabels: {
31        anchor: 'end',
32        align: 'end',
33      }
34    }
35  };
36 // public acum: number[] = [0, 0, 0, 0, 0, 0, 0];
37  public barChartLabels: Label[] = ['salir', 'leer', 'hablar', 'meditar',
    'msica', 'listas', 'test'];
38  public barChartType: ChartType = 'bar';
39  public barChartLegend = true;
40
41  public barChartData: ChartDataSets[] = [
42    { data: [], label: '7 das' },
43    { data: [], label: '15 das' },
44    { data: [], label: '30 das' }
```

```
45  ];
46
47  // events
48  public chartClicked({ event, active }: { event: MouseEvent, active: {}[] }):
    void {
49    console.log(event, active);
50  }
51
52  public chartHovered({ event, active }: { event: MouseEvent, active: {}[] }):
    void {
53    console.log(event, active);
54  }
55
56  public randomize(): void {
57    // Only Change 3 values
58    const data = [
59      Math.round(Math.random() * 100),
60      59,
61      80,
62      (Math.random() * 100),
63      56,
64      (Math.random() * 100),
65      40];
66    this.barChartData[0].data = data;
67  }
68
69  public async getValues7days(){
70    let fechaInicio = new Date(this.fechaActual);
71    fechaInicio.setDate(this.fechaActual.getDate() - 7);
72    let year = fechaInicio.getFullYear();
73    let month = (fechaInicio.getMonth() + 1).toString().padStart(2, '0');
74    let day = fechaInicio.getDate().toString().padStart(2, '0');
75    let fechaFormateada = `${year}-${month}-${day}`;
76    this.getData(fechaFormateada, 7);
77  }
78
79  public async getValues15days(){
80    let fechaInicio = new Date(this.fechaActual);
```

```
81    fechaInicio.setDate(this.fechaActual.getDate() - 15);
82    let year = fechaInicio.getFullYear();
83    let month = (fechaInicio.getMonth() + 1).toString().padStart(2, '0');
84    let day = fechaInicio.getDate().toString().padStart(2, '0');
85    let fechaFormateada = `${year}-${month}-${day}`;
86    this.getData(fechaFormateada, 15);
87  }
88
89  public async getValues30days(){
90    let fechaInicio = new Date(this.fechaActual);
91    fechaInicio.setDate(this.fechaActual.getDate() - 30);
92    let year = fechaInicio.getFullYear();
93    let month = (fechaInicio.getMonth() + 1).toString().padStart(2, '0');
94    let day = fechaInicio.getDate().toString().padStart(2, '0');
95    let fechaFormateada = `${year}-${month}-${day}`;
96    this.getData(fechaFormateada, 30);
97  }
98
99  public getData(fecha, dias){
100   this.getDataService.makeHttpGetRequest(this.user, fecha).subscribe(
101     (response: dataResponse) => {
102       let acum: number[] = [0, 0, 0, 0, 0, 0, 0];
103       for (let clave in response) {
104         switch (response[clave].enunciado) {
105           case 'Quieres salir a dar un paseo?':
106             acum[0]++;
107             break;
108           case 'Quieres leer algn libro que te guste?':
109             acum[1]++;
110             break;
111           case 'Quieres hablar con algn amigo o familiar?':
112             acum[2]++;
113             break;
114           case 'Quieres meditar o hacer ejercicios de respiracin?':
115             acum[3]++;
116             break;
117           case 'Msica':
118             acum[4]++;
```

```
119             break;
120           case 'Asociaciones':
121             acum[5]++;
122             break;
123           case 'Test depresin':
124           case 'Rumiacin':
125             acum[6]++;
126             break;
127         }
128       }
129       switch(dias){
130         case 7:
131         this.barChartData[0].data = acum;
132         break;
133         case 15:
134         this.barChartData[1].data = acum;
135         break;
136         case 30:
137         this.barChartData[2].data = acum;
138         break;
139       }
140     },
141     (error) => {
142       console.error("Error:", error);
143     }
144   );
145 }
146}
147
```

.4.- Vista del gráfico de barras

```
1 <h3 class="m-3">Grafico de Barras</h3>
2 <hr />
3 <div class="row">
4   <div class="col-sm-6 mx-auto">
5     <div>
6       <canvas
```

```
7         baseChart
8         width="400"
9         height="400"
10        [datasets]="barChartData"
11        [labels]="barChartLabels"
12        [options]="barChartOptions"
13        [legend]="barChartLegend"
14        [chartType]="barChartType"
15      ></canvas>
16    </div>
17    <button
18      mat-button
19      mat-raised-button
20      class="btn btn-4 btn-block mx-auto w-25"
21      (click)="randomize()"
22    >
23      Aleatorio
24    </button>
25  </div>
26 </div>
27
```