DESARROLLO DE UN PROTOCOLO DE ADHERENCIA A PARTIR DE UN ASISTENTE DE VOZ VIRTUAL PARA PERSONAS DE LA TERCERA EDAD

Sebastian Palomares Cabrera

Práctica profesional

Tutora

MSc. Sandra Liliana Cancino Suarez

UNIVERSIDAD DEL ROSARIO
ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO
PROGRAMA DE INGENIERÍA BIOMÉDICA
BOGOTÁ D.C
2019

AGRADECIMIENTOS

En primera instancia, agradezco a mi madre por confiar plenamente en mis capacidades como persona y del mismo modo, que haya sido el pilar fundamental de apoyo en cada una de las etapas como estudiante durante el periodo universitario. También agradezco al señor Ali Vargas Sanchez, esposo de mi señora madre, por todo el apoyo y amor incondicional que nos ha brindado a lo largo de nuestra vida.

Agradezco infinitamente a mis compañeras de carrera Hillary y Samantha, por haberme acompañado durante toda nuestra carrera universitaria, convirtiéndose en las compañeras de trabajo y secuaces que cualquier persona quisiera tener.

A la profesora Sandra Liliana Cancino, tutora del proyecto final de carrera durante mis prácticas profesionales, por todo su acompañamiento durante el proceso de elaboración del documento.

Finalmente, agradecer a la Escuela Colombiana de Ingeniería Julio Garavito y a la Universidad del Rosario junto con el programa de Ingeniería Biomédica y todos los docentes que hacen parte de la misma, por haberme brindado todo el conocimiento necesario para estar hoy presentando mi proyecto final de carrera.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE ANEXOS.

1. INTRODUCCIÓN

Según la OMS, en países desarrollados sólo el 50% de la población con algún tipo de patología crónica, cumple de forma eficaz con el tratamiento asignado por el médico. Dependiendo del tipo de circunstancia en la cual se encuentre la persona, las cifras se alteran, provocando que la adherencia al tratamiento se convierta en una preocupación crítica a nivel mundial [1]. Este término, definido como el comportamiento de una persona a las recomendaciones establecidas por un profesional de la salud en la prescripción e ingesta de medicamentos, abarca una población muy grande, incluyendo personas en las diferentes distribuciones demográficas de edad [1]. Según la revista "Información Terapéutica del Sistema Nacional de Salud" en España, al menos el 17% de la población, que son personas mayores a 65 años, son los que consumen el 30% de los medicamentos que residen en el país, teniendo una distribución de 2 y 3 medicamentos por paciente, y con un 75% de estos que presentan un suministro crónico [1].

Existen diversos factores que pueden afectar los problemas en la adherencia, tales como instrucciones insuficientes, fallos en la relación médico-paciente, desacuerdo en el tratamiento por parte del paciente, y particularmente en personas de la tercera edad, mala memoria [2]. Para estos factores, se han desarrollado diferentes tipos de estrategias por parte los profesionales de la salud, sin embargo, las personas siguen incurriendo en malas prácticas en su medicación. Muchas veces el paciente toma decisiones que a su parecer son correctas, considerando factores personales que van relacionados con las creencias que poseen sobre la salud, tales como la percepción de la causa de su enfermedad o la forma en que se le debe dar frente a la situación [3]. Por otro lado, el problema no reside por decisión del paciente, sino que simplemente por la avanzada edad de los mismos, olvidan por completo el horario adecuado de ingesta o se les hace "inútil" seguir con un plan determinado de medicación, considerando que "desde que se haga el consumo del medicamento, no va a ocurrir nada" [4].

Implementación de la Inteligencia Artificial (IA)

El sector salud es uno de los principales objetivos para la implementación de nuevas tecnologías, dando solución a problemas en diferentes escalas de complejidad para los diferentes campos de aplicación. La inteligencia artificial es una de las propuestas que está promoviendo un futuro más ágil y próspero para la medicina. Por ejemplo, los desarrollos de sistemas computacionales que presentan comportamiento inteligente han tenido un gran impacto en el sector de diagnóstico [4]. Por otro lado, los asistentes de voz virtual están dando frutos muy positivos para la asignación de tareas en salas de cirugía o quirófanos, reduciendo de este modo el estrés de los cirujanos y generando un ambiente más ordenado durante los procedimientos quirúrgicos [5].

Las repercusiones a futuro que presentan este tipo de problemas tienen un área de afectación tanto para los pacientes como para el gobierno. Las incorrectas tomas de medicamentos prescritas por los médicos provocan pronósticos negativos para la salud de los pacientes y al mismo tiempo un incremento exponencial en costos de hospitalización y sanidad para el gobierno, convirtiéndose esto en un problema crítico y prioritario en el sector de la salud pública [4].

Figura 1. Logo de AWS (Amazon Web Services) [11]

Todo el desarrollo que conlleva la implementación de IA en los diferentes campos, tiene ligado una cantidad de trabajo enorme. Grandes compañías como Amazon, se han dado a la tarea de crear y ofrecer a las empresas de desarrollo de software herramientas y servicios de toda índole, dentro de las cuales se incluyen la inteligencia artificial y el aprendizaje automático.

Alexa Voice Services (AVS) es un conjunto de servicios de Amazon que está construido en torno a un asistente de inteligencia artificial controlado por voz [6]. El desarrollo que permite este servicio se ha llevado a diferentes niveles de implementación; desde interacciones de la vida cotidiana como hacer consultas del clima, la hora, noticias, pedir órdenes de comida, entre otras, como aplicaciones a nivel de investigación y dedicadas a campos mucho más sofisticados como la medicina o economía.

La figura 2 muestra el diagrama de funcionamiento básico del servicio de Alexa. A partir de esto, se pueden identificar cinco momentos necesarios para que una orden sea cumplida por el asistente de voz virtual. En primera instancia se tiene el proceso de activación del asistente con una palabra clave, que particularmente es “Alexa” y la solicitud deseada. Cabe mencionar que esta puede ser modificable con la alteración del código base del SDK (Kit de desarrollo de software) del AVS [6]. Posterior a esto, las siguientes tres instancias se llevan a cabo en el servicio de la nube, donde se recibe la solicitud, se accede a las habilidades desarrolladas de Alexa y finalmente se emite la respuesta hacia la nube, que finalmente se verá ejecutada nuevamente en el entorno del solicitante [6].

Figura 2. Funcionamiento de AVS (Alexa Voice Service)[9]

El éxito de AVS está en que estas implementaciones se encuentran disponibles a diferentes niveles de complejidad, incluyendo ambientes que permiten desarrollar habilidades personalizadas sin tener un conocimiento avanzado de programación, hasta inclusiones de bajo nivel donde se puede interactuar con el código base del funcionamiento de Alexa.

El gran impacto que ha generado este asistente de voz, ha permitido que personas se interesen en la adquisición de dispositivos inteligentes para hacer de su vida algo más dinámico y entretenido. La cantidad de habilidades que posee Alexa tanto predeterminadas como personalizadas ha desenvuelto proyectos de implementación en campos de negocios propicios para las industrias.

AVS para problemas de adherencia a tratamientos

La implementación de asistentes virtuales en el campo de la adherencia al tratamiento no ha sido de los puntos más fuertes como una funcionalidad de este servicio. Considerar la personalización de un agente para dar solución a este tipo de problema resulta algo complejo debido a que no todos los futuros usuarios presentan un mismo comportamiento. Es posible que algunos rechacen la idea de que una máquina les dé órdenes, o que simplemente, no tengan interés alguno en hablar con un "aparato". [2].

Por esta razón, se ha decido realizar un acotamiento de público para el desarrollo de un asistente virtual que abarque la población de personas de la tercera edad con el objetivo de darle una mejora a los procesos de medicación por parte de los pacientes. La tarea identificada es permitirle a una persona poder agendar su medicación, ya sea parcial o definitiva, en un asistente de voz basado en inteligencia artificial aplicando métodos de interacción humano-máquina, donde se implementa el estudio de cómo las personas interactúan con sistemas informáticos y su respectiva reacción e influencia de las mismas sobre los usuarios. De este modo, las personas tienen un apoyo adicional que vaya en pro a las correctas prácticas de suministro de medicamentos. Es importante mencionar que el entorno de interacción paciente-máquina debe ser orientado al trato con personas mayores, es decir, intentar que el ambiente de comunicación con el agente de inteligencia artificial sea lo más ameno posible.

Entonces, dentro del documento se encontrarán cada uno de los procedimientos que se llevaron a cabo para el desarrollo del protocolo de adherencia, con toda la configuración y adecuación del entorno de Alexa en una computadora de placa reducida (Raspberry Pi 3). Se mostrarán los protocolos diseñados como diagramas de flujo, evidenciando cada uno de los pasos requeridos para un agendamiento eficaz. Finalmente, se mostrarán y explicarán los resultados obtenidos en consola del AVS, argumentando cada situación presentada durante la ejecución de las pruebas.

RESEÑA DE LA EMPRESA

Work.r es una compañía creada en el 2018, como fruto de una evolución de otras empresas que trabajan actualmente con el desarrollo y sistematización de software de alto nivel. El propósito de esta empresa fue encontrar nuevos caminos de desarrollo con la implementación de tecnologías revolucionarias como lo son el aprendizaje automático y la inteligencia artificial.

Figura 3. Logotipo de la Empresa [10]

El departamento de Work.r Labs (W.Labs) (Ver Anexo 1) es el lugar donde se realizan todos los procedimientos de desarrollo de software. Los objetivos que pretende esta compañía presentan dos direccionamientos,

1. Hacer de las máquinas un servicio para los humanos y no al revés. A partir de la inteligencia artificial y la automatización transformar drásticamente la naturaleza del trabajo permitiendo la realización de las personas en el contexto profesional.

2. Hacer que no existan límites para brindar una experiencia Premium a los usuarios finales y clientes. Usualmente ese límite se impone por el costo de perfiles especializados realizando tareas de valor marginal, por lo cual, que la tecnología implementada elimine ese límite y permita una experiencia Premium a cualquier usuario.

El trabajo realizado en este departamento abarca muchas áreas de desarrollo. La generalidad principal de la labor es apoyar todos los procedimientos que contengan implementación de aprendizaje automático a partir de servicios proporcionado por AWS. Por otro lado, está la labor de QA (Quality Assurance) la cual implica el testeo y revisión de software creados por los agentes en cada una las etapas de desarrollo, identificando errores de ejecución, problemas con la interacción de las interfaces, entre otros. Finalmente, la incursión en la dirección y planeación de proyectos. La empresa posee grupos Junior a los cuales se les asignan proyectos de pequeña escala para que los ejecuten completamente, incluyendo planificación de tareas, entrevistas con clientes, identificación de flujos de trabajo y sin dejar atrás, el desarrollo de software.

Información obtenida de [8]

2. OBJETIVOS

2.1. General

1. Desarrollar un protocolo de adherencia a partir de un asistente de voz virtual basado en inteligencia artificial, que permita a las personas de la tercera edad realizar un agendamiento de medicación parcial o definitivo en un entorno de interacción humano-máquina agradable, con una lectura previa de alguna receta médica.

2.2. Específicos

1. Implementar el servicio de *Alexa Voice Service* (AVS) como librería de desarrollo para algoritmo que contiene el protocolo de agendamiento.

2. Utilizar el dispositivo Raspberry Pi 3 como una herramienta para la ejecución del entorno de desarrollo del algoritmo.

3. Medir el desempeño del sistema al generar nombres personalizados para la activación del asistente de voz virtual.

4. Incluir un servicio para la ejecución de llamadas de emergencia como elemento adicional del asistente de voz virtual.

5. Evaluar el sistema con pruebas de desempeño y experiencia del algoritmo con usuarios pertenecientes a la tercera edad.

3. METODOLOGÍA

Para alcanzar los objetivos planteados en el presente proyecto, se presenta la metodología, en la cual se incluyen toda la organización cronológica de las actividades puntuales realizadas, entrando en detalle en cada uno de los procedimientos paso a paso.

3.1. Problema a solucionar

Teniendo en cuenta los objetivos instaurados para el desarrollo de este proyecto, y, junto con una breve realimentación del porqué o para qué; identificada como la creación de una habilidad personalizada en un asistente de voz virtual para mejorar los procesos de medicación en pacientes de la tercera edad, se establecen cinco actividades clave que tienen como objetivo llevar un orden estructural de cada uno de los pasos a seguir.

Actividades

1. Implementación e instalación del entorno de programación de Alexa en un dispositivo que involucra un lenguaje de bajo nivel, para este caso, el Raspberry Pi 3.

2. Generar el nombre de invocación "WakeWord" personalizado del asistente de voz virtual.

3. Implementar el servicio de Alexa Built Skill para la construcción y entrenamiento de cada una de las instrucciones que recibe del usuario a cargo.

4. Inclusión de un asistente de llamadas para casos de emergencia.

5. Pruebas de evaluación de desempeño y satisfacción del algoritmo en usuarios objetivos.

La elección de la Raspberry Pi 3 como dispositivo es debido a que éste fue el material proporcionado por la empresa para llevar a cabo todo el desarrollo. Por otro lado, como se mencionó en la reseña de la empresa, ésta sostiene su entorno de trabajo sobre los servicios de Amazon, por lo tanto, *Alexa Voice Service* fue, del mismo modo, el asistente de voz virtual indicado por la empresa para la creación del protocolo de agendamiento.

INSTALACIÓN E IMPLEMENTACIÓN DEL ENTORNO DE ALEXA EN LA RASPBERRY PI 3

Para realizar la instalación e implementación de AVS en la Raspberry, se tuvieron en cuenta algunos procedimientos previos que son requisitos para poder tener acceso a los servicios de Amazon. Con esto, se tiene lo siguiente,

- Creación y configuración de una cuenta de AWS como desarrollador.

Cada uno de los pasos que implica el procedimiento mencionado, traen consigo el registro de los respectivos productos que se utilizaron en la consola de AVS, para el caso

en particular, la Raspberry Pi 3. Por otro lado, se establecieron perfiles de seguridad requeridos por Amazon para asegurar una conexión única entre el computador de placa reducida implementado y el servicio de Alexa.

Como parte de hardware adicional utilizado en el desarrollo, se utilizaron los siguientes elementos

- Tarjeta de memoria MicroSD
- Micrófono USB
- Dispositivo para salida de audio
- Fuente de alimentación
- Periféricos (Teclado, Ratón, Cable HDMI)

Es importante mencionar que estos elementos son completamente necesarios para hacer uso de la Raspberry, generando de este modo, una limitación en posibles diseños de hardware. El único elemento que puede hacer la diferencia entre un entorno con mejor desempeño que otro, es el micrófono. La calidad del mismo puede provocar inconsistencias a la hora de que el asistente de voz virtual reciba e interprete bien o no la instrucción emitida por el usuario. Por esta razón, para asegurar un posible porcentaje de errores de interpretación por parte del agente, se utilizó el micrófono de referencia Audio-Technica ATR2500. (Ver figura 4), ya que presenta un funcionamiento bastante bueno a la hora del funcionamiento.

Figura 4. Micrófono utilizado para el desarrollo del proyecto [12]

Preparación del SO (Sistema Operativo)

Para la instalación del SO en la Raspberry, se opta por el RaspbianOS, sistema operativo propio de los creadores de estas computadoras de placa reducida, en su versión Stretch. Esto debido a que para la implementación del AVS, hasta el momento, esta versión es la más actualizada y la única que soporta una conectividad estable con los servicios de Amazon.

Ensamblar la Raspberry Pi

Para el ensamble del dispositivo, se tuvo en cuenta solamente un aspecto,

- Todos los elementos hardware utilizados para el montaje deben estar conectados a la hora de cada proceso de instalación. Este fue un aspecto muy

importante, debido a que, si alguno fallaba durante cualquier procedimiento, el sistema presentaría errores en cuanto a la detección de los mismos.

Configuración de las credenciales AVS

Para realizar la configuración de las credenciales, se utilizó el terminal que posee el software, mediante el cual es posible ejecutar cualquier tipo de petición en relación con el SO. Como medida principal, se actualizaron todas las librerías presentes en el sistema operativo actual del momento. Para esto se utiliza el comando,

```
sudo apt-get upgrade
```

La función de esta línea es hacer una actualización completa de todo el contenido que haya dentro del sistema operativo previamente instalado, concediendo permisos tanto para la lectura como escritura de la nueva información por instalar.

Luego de haber terminado la actualización, se obtuvieron desde el repositorio de Alexa, los archivos necesarios para la ejecución del servicio en el dispositivo. Para esto, en el terminal, se ejecutó el comando wget, que funciona como herramienta para obtener contenido desde servidores web.

```
wget https://raw.githubusercontent.com/alexa/avs-device-
sdk/master/tools/Install/setup.sh \
wget https://raw.githubusercontent.com/alexa/avs-device-
sdk/master/tools/Install/genConfig.sh \
wget https://raw.githubusercontent.com/alexa/avs-device-
sdk/master/tools/Install/pi.sh
```

Estos archivos poseen cada una de las instancias necesarias para llevar a cabo toda la instalación del servicio de Alexa, incluyendo sus librerías, las credenciales que permiten la única comunicación entre el dispositivo y los servidores de AVS, entre otros.

Configuración del SDK de AVS en el dispositivo

La ejecución de la configuración del SDK de Alexa, asegura que todos los protocolos de comunicación entre el dispositivo y el servidor funcionen de manera correcta. Este procedimiento también instala todas las dependencias necesarias para el funcionamiento, incluyendo la configuración del nombre de invocación “Wake Word”, el cual se encarga de la activación del asistente una vez se mencione la palabra identificada, para este caso está por defecto, “Alexa”. Para llevar a cabo este proceso, se ejecutó en el terminal el siguiente comando,

```
sudo bash setup.sh config.json [-s 1234]
```

Obtención del TOKEN de autenticación

La obtención del token de autenticación se realiza con el objetivo de garantizar una sesión abierta con AVS de forma permanente. Básicamente, se hace uso de este proceso

para evitar estar configurando las credenciales de AVS cada vez que se fuera a utilizar el servicio de AVS en el dispositivo Raspberry Pi 3.

PERSONALIZACIÓN DEL NOMBRE DE INVACIÓN "*Wake Word* (WW)"

Para proceder a la personalización del WW, es necesario presentar el agente que hace posible realizar este cambio.

Snowboy, es un mecanismo de alta precisión el cual permite detectar palabras claves totalmente personalizables. Este sistema funciona en un entorno de tiempo real donde siempre está atento a cualquier intención de invocación. El fundamento de este sistema se basa en un algoritmo conocido como ASR (Automatic Speech Recognition), el cual es el encargado de toda la detección y procesamiento de la señal de entrada, la cual es la que posee el WW.

Con esto, se procede a la implementación del mecanismo haciendo uso del repositorio disponible. En primera instancia, se clonó el repositorio de Snowboy desde el terminal para tener acceso a toda la información que posee el mecanismo desde una ruta local. Luego, se realizaron los cambios necesarios dentro de los archivos fuente del AVS instalado previamente. De este modo, el WW puede ser alterado por fuentes externas a las que provee el servicio de Amazon.

Para la creación del WW personalizado, el procedimiento consistió en la asignación de un nombre con un idioma determinado. Esta característica permite que el asistente de voz sea capaz de diferenciar los intentos de invocación en diferentes formas de entonación y acento. Generalmente se requieren unas 2000 muestras para tener una palabra de invocación precisa, por esta razón se opta por tomar una palabra ya entrenada, la cual es "Jarvis".

Cabe resaltar que la palabra de invocación escogida, no es la mejor si se desea apuntar a una población de la tercera edad. Sin embargo, la posibilidad de personalización siempre estará disponible a la necesidad que se pueda presentar. Poder cambiar el nombre del asistente de voz genera mucha más confianza entre la comodidad que pueda sentir un usuario al interactuar con el mismo.

CREACIÓN E IMPLEMENTACIÓN DE LA SKILL PERSONALIZADA

Para la creación e implementación de una habilidad personalizada, AVS permite un desarrollo en dos lenguajes de programación de alto nivel, NodeJS y Python. Para este caso, se escogió trabajar con NodeJS por motivos de manejos previos con el lenguaje y por la alta cantidad de documentación que se encuentra en internet.

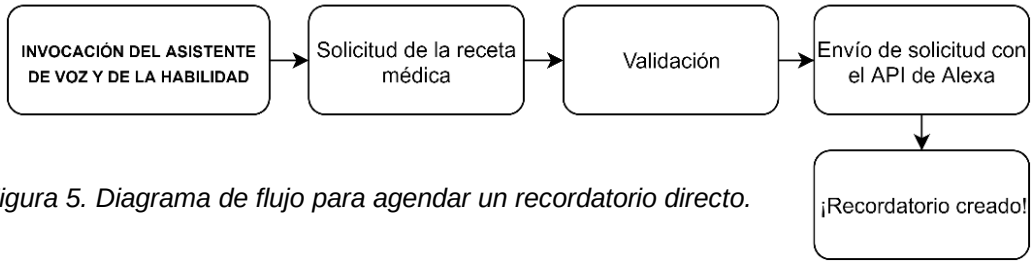


Figura 5. Diagrama de flujo para agendar un recordatorio directo.

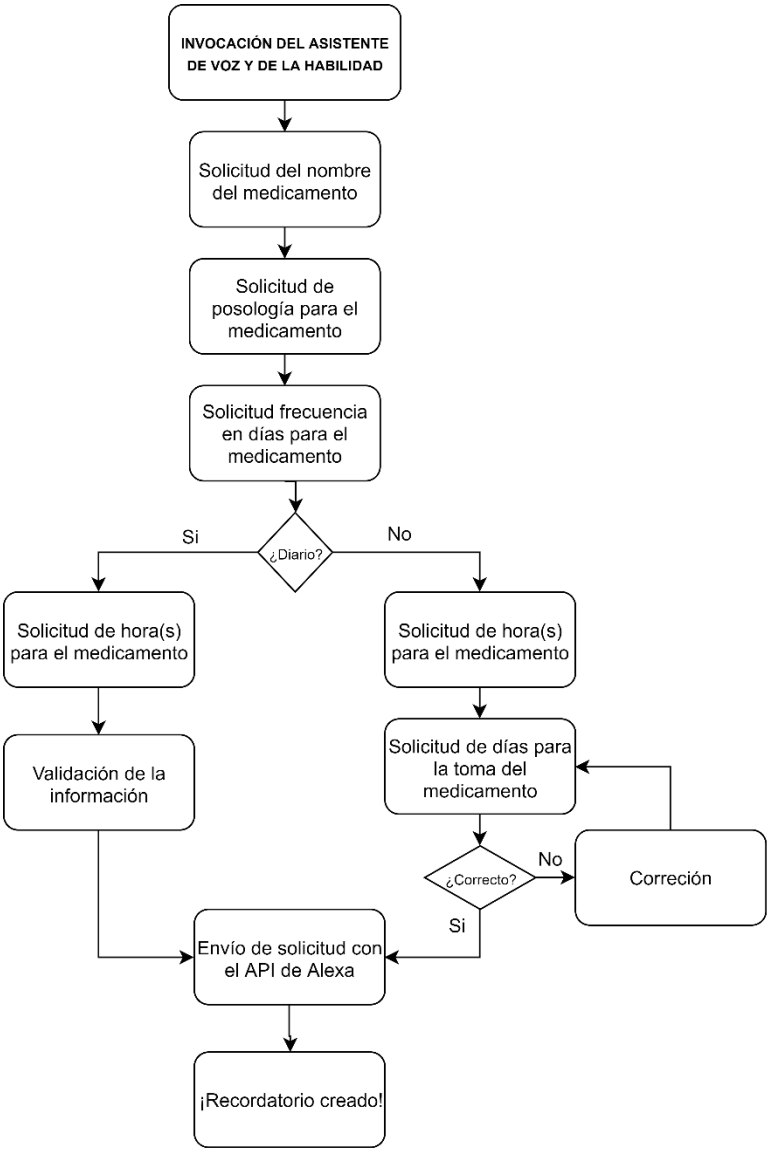


Figura 6. Diagrama de flujo para agendar un recordatorio no directo.

Para el desarrollo de la habilidad, se tuvieron en cuenta dos instancias de programación, conocidas como FRONT-END (FE) y BACK-END (BE). En resumidas palabras, lo que implica el FE es toda la interacción que tiene el desarrollador con las diferentes opciones de programación para definir las intenciones y/o variables de asignación. Por otro lado, el BE abarca todo el código implementado para seguir una secuencia dentro de la habilidad diseñada.

Dado esto, para la creación de la habilidad, se tuvieron en cuenta dos factores. En principio, que la complejidad del protocolo a seguir por un usuario resultara lo más sencillo posible, pues para las personas de la tercera edad es importante que el proceso de agendamiento sea algo fácil de realizar.

Por otro lado, se consideró la implementación de dos protocolos de agendamiento diferentes. El objetivo era poder ofrecer a los usuarios la posibilidad de realizar agendamientos mucho más dinámicos. Con esto, se crearon dos procesos de agendamiento, en donde en el primero, la persona es capaz de agendar un medicamento con tan solo la invocación del asistente y la respectiva instrucción, mencionando características como nombre, dosificación, frecuencia del medicamento, entre otras (Ver figura 5).

Con el otro protocolo, se consideraron aquellas personas que necesitan seguir un proceso más detallado en el agendamiento de los medicamentos. Por esta razón, se diseñó una secuencia de pasos en donde el agente interactúa con el usuario, solicitando en cada instancia, una petición determinada, ya sea el nombre del medicamento, dosificación, frecuencia, hora de ingesta, entre otras (Ver figura 6). Como se mencionó en un principio, el principal objetivo de la creación de estos protocolos es que la persona que esté

interactuando con el asistente de voz se sienta lo más cómodo posible, y que su experiencia durante el proceso sea positiva.

Una vez creadas las habilidades del asistente de voz, se procede a la construcción y entrenamiento del algoritmo. Con esto, se busca que dichas habilidades queden disponibles para su uso en la nube de AVS, y así, poder acceder a las mismas desde el dispositivo hardware (Raspberry Pi 3)

3.2. Fases del proyecto

Para el desglose de las fases del proyecto se recurrió a plasmar cada una de las actividades con sus respectivos tiempos de duración en un diagrama de Gantt. Para la elaboración del mismo, se tuvieron en cuenta tres fases principales, dentro de las cuales se tienen las siguientes

- **Fase 1**

  Esta fase incluye toda la parte de investigación de los temas los cuales cimientan toda la ejecución del proyecto. Partiendo de un estado del arte para el problema encontrado, revisiones bibliográficas sobre desarrollo que se encuentren en la actualidad y como actividad principal, la revisión de documentación que presenta *Alexa Voice Service* para la implementación de sus librerías en el entorno de desarrollo.

- **Fase 2**

  Esta fase es la más extensa de toda la ejecución del proyecto. Aquí se encuentran todos los procedimientos y desarrollos que se realizan para llevar a cabo la elaboración del algoritmo. Partiendo de pruebas generales utilizando codificación propuesta por la documentación encontrada de AVS, implementando toda la configuración inicial de hardware y software, pruebas pilotos con algoritmos ya preestablecidos, configuración de los dispositivos, asignación de nombres de activación, seguimientos parciales del funcionamiento del algoritmo, entre otros. De este modo, también se diseña cada protocolo implementado para el desarrollo de la habilidad, buscando la mejor forma de utilizar AVS con personas de la tercera edad.

- **Fase 3**

  En esta etapa se buscó realizar pruebas de evaluación y experiencia del algoritmo desarrollado con usuarios en casos de la vida real.

Finalmente, la asignación de tiempos para la ejecución de todo el proyecto fue de 9.5 semanas aproximadamente, en donde para cada intervalo de tiempo, se esperaba realizar diferentes actividades en pro al desarrollo de la habilidad. Finalmente, el diagrama de Gantt se adjunta como archivo Anexo al final del documento, donde se especifica cada actividad con su respectiva duración.

4. RESULTADOS

Los resultados que se muestran, abarcan todo el detalle de la programación realizada para la obtención de la *skill* final que se implementa con el servicio de AVS, explicando cada una de las fases con las dos perspectivas de programación, FE y BE.

Partiendo de la documentación que propone AVS para la creación de nuevas *skills*, se tiene la identificación de diferentes parámetros requeridos para una iniciación o apertura del protocolo estándar en el desarrollo de habilidades personalizadas de Alexa. La arquitectura que se implementó, funciona de la siguiente manera:

- Se declara una función madre la cual posee tanto las variables de entrada y de salida, que particularmente son archivos en formato json que se generan una vez se recibe y se entrega la solicitud.

- Se establece un nombre de invocación, tanto para el asistente de voz como para la habilidad personalizada. Esto implica que el usuario debe mencionar tanto el nombre del asistente que escogió como la palabra que acciona la habilidad. Para este caso se escogió “Jarvis, quiero agendar un medicamento”.

- Se declaran cada una de las intenciones que respectan al protocolo de activación de cada solicitud. Esto hace referencia a un proceso en un orden estructurado en cuanto “qué debería hacer el asistente después”.

- Para algunos casos, se declaran variables tipo slot, las cuales permiten almacenar palabras o respuestas en particular, para que puedan ser usadas luego.

- Finalmente, existen funciones necesarias para un desarrollo completo, tales como intenciones de cancelación, de ayuda, de detención y de afirmación.

Dentro de las intenciones que se crearon, se encuentran las siguientes:

- **PreAgendar:** Su función es solicitar el nombre del medicamento.

- **AgendarMedicamento:** Su función es interpretar el nombre que el usuario mencionó y a su vez solicitar la posología del mismo.

- **CantidadVeces:** Su función es guardar la cantidad de veces que debe tomar el medicamento por día y a su vez, solicitar la frecuencia de la ingesta, ya sea diaria o personalizada (No todos los días).

- **Diario**: El objetivo de esta intención es inicializar el proceso de agendamiento diario de pastillas. También solicita los horarios de toma.

- **noDiario:** El objetivo de esta intención es inicializar el proceso de agendamiento del medicamento en ciertos días en específicos, donde de igual forma, solicita los respectivos horarios

- **daysRequest:** Esta intención se encarga de recolectar los días en específico que mencione el usuario.

- **StopdaysRequest:** Esta intención detiene la recolección de días.

- **corregirRequest:** Permite al usuario corregir los días escogidos para el agendamiento.

- **horaRequest**: Hace el procesamiento de las horas con el objetivo de que puedan ser ingresadas en la creación del recordatorio.

- **CreacionReminder:** Ejecuta todo el proceso necesario para agendar un recordatorio.

A partir de estas intenciones, se genera todo el protocolo de recolección de datos necesarios para el agendamiento de un medicamento. Las siguientes figuras muestran el resultado en consola de la interacción del asistente de voz con un usuario en particular.

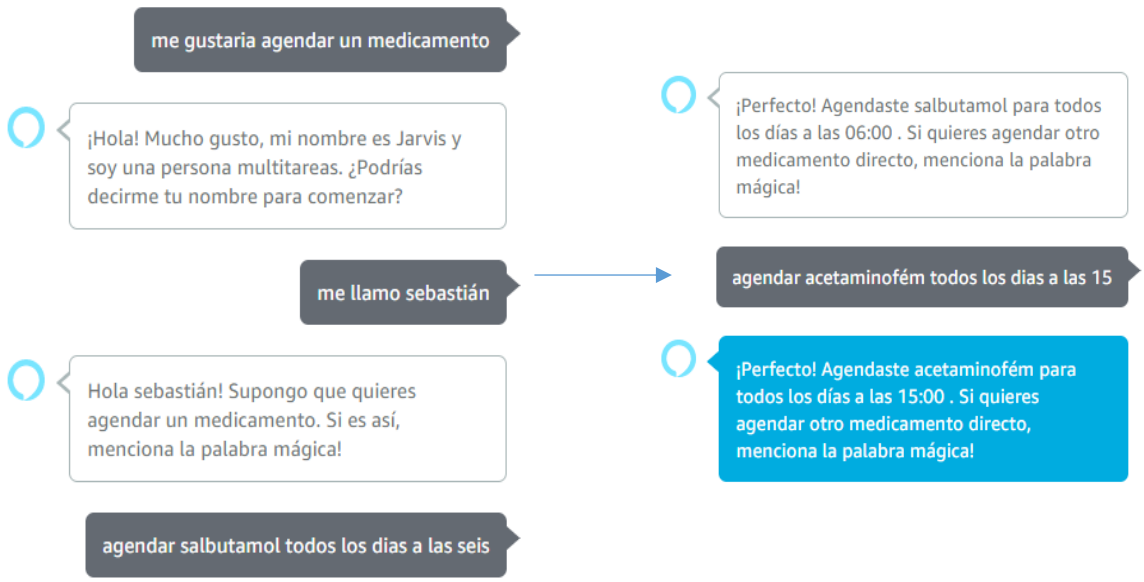


Figura 7. Interacción usuario-asistente en un agendamiento diario de forma directa.

Figura 8. Interacción usuario-asistente en un agendamiento no directo con dos ingestas al día.

Figura 9. Interacción usuario-asistente en un agendamiento específico de días con una ingesta por día de forma no directa.

Las figuras 7, 8 y 9 muestran el resultado de cada protocolo ejecutado a nivel de consola. Como se puede observar, cada escenario muestra el paso a paso de las instrucciones que tiene programado el asistente de voz virtual, estableciendo una comunicación agradable con el usuario.

Es importante mencionar que cada respuesta que tiene el asistente de voz, es programable, por lo cual, el gran potencial que tiene este sistema en comparación con otros asistentes de voz, es la capacidad de personalización de acuerdo a las condiciones que presente el usuario a aplicar. Para efectos de pruebas, se crearon respuestas basados en una persona de la tercera edad que esté cómoda con frases que involucren un ambiente mucho más amigable. Sin embargo, habrá personas que prefieran que su asistente de voz se limite a responder lo que el usuario busque, sin tonalidades ni formas de respuesta amables.

Como procedimientos de evaluación del sistema, se midieron a partir de pruebas iterativas el desempeño del asistente en tres factores diferentes.

- En primera instancia, se realizaron 50 intentos de agendamiento de la forma directa, es decir, mencionando el nombre del medicamento, dosificación, frecuencia y hora(s) de la ingesta. A continuación, se muestra el resultado de los procedimientos completados sin ningún error.

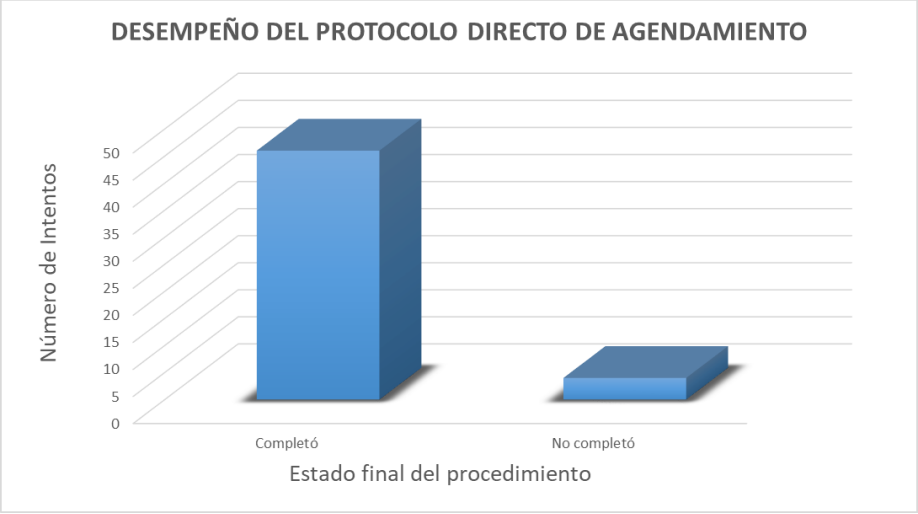


Figura 10. Gráfica de desempeño del agendamiento directo en función del número de procedimientos realizados vs el estado final.

- Por otro lado, evaluó el desempeño del agendamiento no directo, realizando variaciones en las instrucciones como los días de ingesta, dosificación y horario al día de la toma de los medicamentos. Del mismo modo, se realizaron 50 intentos. A continuación, se muestra el resultado.

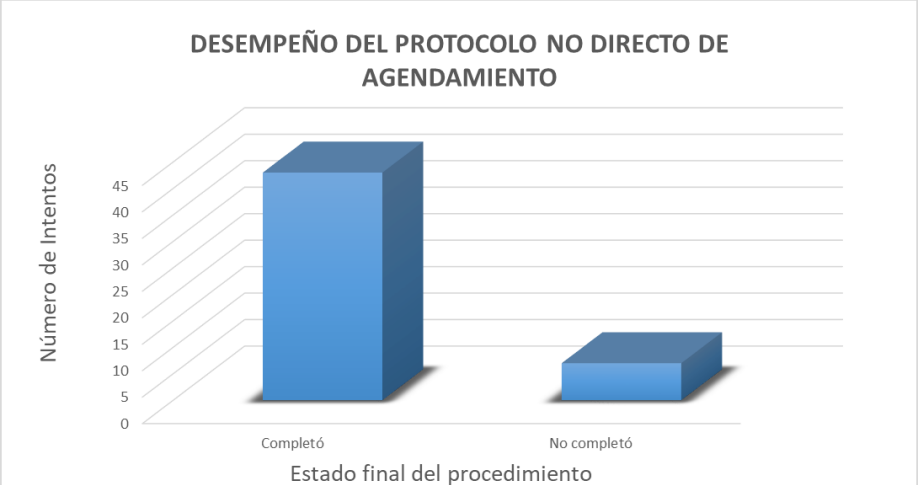


Figura 11. Gráfica de desempeño del agendamiento no directo en función del número de procedimientos realizados vs el estado final.

- Finalmente, como prueba final se realizó la evaluación de desempeño al nombre de invocación personalizado. Como se dijo anteriormente, para hacer uso de una palabra de invocación personalizada es recomendado hacer un entrenamiento previo de al menos 1500 muestras, obteniendo grabaciones de voz de diferentes tipos, mujeres, hombres, niños, etc. Para medir el desempeño, se realizaron 50 intentos de invocación al asistente una vez modificado el WW. Los resultados se muestran a continuación.

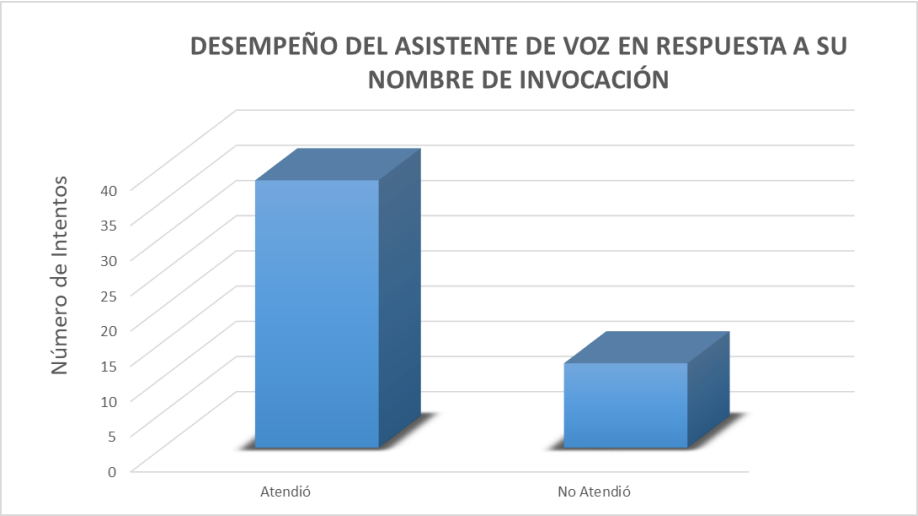


Figura 12. Gráfica de desempeño del asistente de voz en respuesta al nombre de invocación asignado manualmente.

5. DISCUSIÓN

Partiendo con el proceso de instalación e implementación de la Raspberry Pi 3, se pueden discutir diferentes cosas. En principio, se debe recalcar que hacer uso de un dispositivo como estos permite de alguna forma, una facilidad al momento de realizar pruebas en bruto con cada uno de los procesos que presenta el protocolo, es decir, cada que se efectuaba un cambio en el código, se podía comprobar de una manera sencilla, sin necesidad de poseer un dispositivo especial para las pruebas, como lo puede ser un Echo Dot de Amazon u otros prototipos de asistentes de voz inteligentes. Por otro lado, el fácil acceso y manipulación de los archivos que presenta un desarrollador al trabajar con un sistema operativo basado en Linux, permite una gran libertad de poder modificar archivos que podrían ser los causantes en los problemas de compilación de código. Por ejemplo, para la implementación del cambio del WW, se tuvieron que alterar los archivos fuente debido a una inconsistencia en la compilación.

Por otro lado, se tiene la programación de cada una de las intenciones que maneja el protocolo de adherencia. En primera instancia, resultó muy útil la amplia gama de librerías que presenta el SDK de AVS, con los cuales se pudieron implementar diferentes funcionalidades que permitieron un desarrollo mucho más limpio y claro. Sin embargo, dentro de los problemas que se encontraron, está la capacidad de interpretación de instrucciones del asistente. Cabe mencionar que a pesar de que los resultados de las evaluaciones de desempeño fueron muy buenos, pues la gran mayoría de intentos fueron exitosos, siempre queda el imperfecto de que el asistente en ocasiones no logra procesar bien las instrucciones generadas por el usuario. Esto puede llegar a ser un problema en una persona que tenga su tono de voz muy bajo, pues difícilmente el asistente será capaz de interpretar la orden.

Dado a lo anterior, se buscaron formas alternativas de comunicación con el asistente para lograr cumplir con el objetivo de que el protocolo estuviera en buenas condiciones de uso. Por esta razón, se utilizó la consola de desarrollo de AVS, la cual permite interactuar con el asistente de manera escrita. Gracias a esto, se pudo comprobar el correcto funcionamiento del protocolo. Como se muestra en las figuras 7, 8 y 9, la habilidad desarrollada es capaz de agendar recordatorios para la toma de medicamentos en diferentes modalidades, variando la posología y la frecuencia de ingesta de los mismos. Cabe resaltar que cada una de las intenciones programadas funcionaron correctamente, implicando una respuesta deseada de acuerdo a la respectiva solicitud del usuario.

Por otro lado, el reconocimiento de caracteres especiales y la estricta necesidad de solicitar las intenciones resultó más eficiente de lo que se esperaba (Ver figuras 7, 8 y 9). Es importante mencionar que estos asistentes de voz virtual trabajan bajo el criterio del aprendizaje automático, por lo cual, basados en la capacidad de compresión auditiva, este tipo máquinas tienden a entrenarse cada vez más. De este modo, cuando la intención de activación u orden estaba levemente incorrecta, aun así, el asistente entendía la solicitud y ejecutaba la siguiente instrucción. Con esto se hace referencia a que el asistente de voz, en ocasiones no es capaz de responder si el usuario solicita la intención de una forma diferente a la predefinida en el código. Por ejemplo, al momento de pedir la hora, el usuario debería responder explícitamente “a las NN y a las NN” para que el asistente entienda perfectamente. Sin embargo, si el usuario responde “me gustaría que quedara a las NN”,

gracias al ML, el asistente era capaz de entender la instrucción con la misma intención. Los problemas que existen entorno a la capacidad de entendimiento pueden solucionarse generando una programación mucho más estructural, es decir, incluir las diferentes formas de instrucción la cual una persona puede realizar la orden.

Ahora, considerando el tema de la inclusión de un asistente capaz de realizar llamadas en caso de emergencia, esta funcionalidad no fue posible agregarla por problemas internos presentados en la empresa. El proyecto que en un principio se solicitó, llevaba consigo unos objetivos preestablecidos, dentro de los cuales estaba incluido el propósito de esta herramienta funcional. Sin embargo, la implementación de un número de teléfono en el asistente, el cual tuviese la capacidad de generar llamadas presentó un costo muy elevado, por lo cual, la entidad interesada en el proyecto del asistente virtual afirmó que no le interesaba la adquisición de la misma. Por esta razón, a pesar de que se tenían las mejores expectativas de funcionamiento para este desarrollo, no se llevó a cabo.

Finalmente, la evaluación de desempeño y experiencia por parte de los usuarios fue otro de los objetivos que no se lograron. Una vez desarrollada la habilidad con sus diferentes modos de funcionamiento, fue presentada a la empresa. Desafortunadamente, esta entidad decidió no seguir con el proyecto por razones ajenas a las de la empresa. Por esta razón, se tomó la decisión de suspender con totalidad el proyecto el cual se había estado llevando a cabo. Sin embargo, dentro de un rango de pruebas piloto durante el desarrollo del algoritmo, la habilidad mostró buenos resultados en cuanto a la capacidad de entendimiento de instrucciones y seguimiento ordenado del protocolo en cuestión.

6. RECOMENDACIONES Y TRABAJOS FUTUROS

Para establecer posibles recomendaciones en el proyecto desarrollado, es necesario tener en cuenta los objetivos logrados y a su vez los problemas que se presentaron durante el desarrollo.

Es importante mencionar de nuevo el objetivo principal de este proyecto, el cual consistía en el desarrollo de un protocolo de adherencia a partir de un asistente de voz virtual, el cual estaba dirigido a un trato con personas de la tercera edad, permitiendo agendar medicamentos de forma agradable y rápida, incluyendo su posterior recordatorio.

A partir de esto y con los resultados obtenidos, es importante que, para trabajos futuros, se establezcan procedimiento más concretos y simples a la hora de implementar el desarrollo a nivel de código. Cabe resaltar que el funcionamiento del código implementado en el algoritmo fue muy bueno, ya que con este fue capaz de realizar agendamientos en las diferentes modalidades sin gran problema alguno. Sin embargo, siempre es bueno hacer uso de mejores prácticas de programación, con esto, evitar códigos tan extensos como se puede evidenciar en los anexos.

Por otro lado, es importante mencionar que la implementación de una Raspberry Pi 3 facilita de alguna forma la programación del algoritmo, ya que se puede tener fácil acceso a permisos que quizá en otro entorno de desarrollo no existan o no sea posible acceder. No obstante, se debe considerar un ambiente real, donde una persona de la tercera edad, con sus respectivas condiciones de vida, al estar en un trato con un sistema tan complejo, puede resultar un poco complicado. Por esta razón, con el objetivo de ofrecer una experiencia de usuario mucho más satisfactoria, se recomienda hacer uso de dispositivos inteligentes como Echo Dots que posean el entorno de *Alexa Voice Service*.

Otro aspecto importante a tener en cuenta, es la evaluación de satisfacción que debe existir para este producto. Como ya se explicó, debido a problemas ajenos a la empresa, no fue posible la ejecución de los mismos. Sin embargo, en todo proyecto que presente un usuario final, es necesaria la existencia de métricas de desempeño. Con esto, poder medir que tan efectivo fue el desarrollo implementado y si los usuarios finales se sienten a gusto con el mismo. Como posible trabajo a futuro, se recomienda la ejecución de estas evaluaciones. Cabe mencionar que dentro de las pruebas pilotos parciales realizadas durante el desarrollo curso, el algoritmo no presentó gran problema a la hora de probar su funcionamiento. No obstante, la experiencia de terceros son pruebas contundentes para la evaluación de un producto.

Finalmente, este proyecto tiene una proyección con bastante alcance. Implementando una mejora en las prácticas de programación y con una evaluación de satisfacción del algoritmo para los usuarios, es posible llevar a este prototipo a una gran escala, pudiendo resultar en conexiones con terminales de salud como farmacias, hospitales, entre otros. También podría reconsiderarse la adición de un servicio de teléfono virtual el cual sea capaz de realizar llamadas en ocasiones de emergencia; cabe mencionar de nuevo que estos servicios son de pago, por lo cual los costos pueden resultar elevados. Al final, se busca que la tecnología sea lo más completa posible para ofrecer a los usuarios servicios con experiencias inigualables.

7. CONCLUSIONES

A partir de todo el desarrollo que conllevó el proyecto y junto con los resultados positivos que se obtuvieron, se llegó a las siguientes conclusiones.

En primera instancia, la eficiencia que ofrece llevar a cabo un desarrollo de bajo nivel en dispositivos como la Raspberry Pi 3, se vio reflejado en este proyecto. Hacer uso de un ambiente de programación dedicado, permite un acceso total a todas las funcionalidades del entorno de desarrollo, pudiendo realizar modificaciones si son necesarias, cosa que quizá en otros entornos no permitirían. Pese a la limitación de uso de estos hardware en casos de la vida real, toda la experiencia de desarrollo dentro de este entorno dejó mucho conocimiento por delante y despertó intereses que pueden expandir la idea inicial a propuestas como mucho más nivel de sofisticación.

Por otro lado, evaluando el funcionamiento parcial del prototipo desarrollado con los servicios de AVS, el resultado final superó las expectativas que se tenían en un principio. Hay que considerar que implementar un asistente de voz virtual para todos los posibles casos de instrucciones en primera instancia, es complejo. Sin embargo, la capacidad de entendimiento del asistente virtual mejoraba cada vez más, atendiendo solicitudes de respuesta con versiones de instrucción diferentes.

Finalmente, la gran utilidad que representa hacer uso de asistentes de voz virtuales basados en inteligencia artificial, al momento de desarrollar una propuesta de solución a un problema tan robusto como lo es la adherencia a los medicamentos; posee altas expectativas de éxito. La implementación de la inteligencia artificial como herramienta funcional de desarrollo permite un amplio pool de posibilidades para la creación de servicios basados en agentes inteligentes [5]. La proyección de éxito de este tipo de productos en un mercado orientado a un público en particular, posee un potencial con gran alcance, por lo cual, si se ejecutan protocolos de mejora como el uso de números de teléfono para llamadas de emergencia, todo el sistema será más atractivo para los usuarios.

REFERENCIAS

[1] María Luisa Peralta,* Patricia Carbajal Pruneda*, “Adherencia Al Tratamiento”. Rev Cent Dermatol Pascua • Vol. 17, Núm. 3 • Sep-Dic 2008, pg 1.

[2] F. Martínez Mir and V. Palop Larrea, “Adherencia al tratamiento en el paciente anciano,” Inf. Ter. del Sist. Nac. Salud, vol. 28, no. 5, pp. 113–120, 2004.

[3] N. V. Bello Escamilla and P. A. Montoya Cáceres, “Adherencia al tratamiento farmacológico en adultos mayores diabéticos tipo 2 y sus factores asociados,” Gerokomos, vol. 28, no. 2, pp. 73–77, 2017.

[4] S. Vicente-Sánchez, R. Olmos-Jiménez, C. Ramírez-Roig, M. J. García-Sánchez, M. Valderrey-Pulido, and A. De La Rubia-Nieto, “Treatment adherence in patients more than 65 years who experience early readmissions,” Farm. Hosp., vol. 42, no. 4, pp. 147–151, 2018.

[5] A. Torres Pombert, “Acimed.,” Acimed, vol. 11, no. 3, pp. 7–8, 2003.

[6] “Referencia general de AWS Guía de referencia.”

[7] T. H. E. Official, “Raspberry Pi Beginner ’ s Guide,” 2018.

[8] Workr Labs. (2019). Como trabaja Work.r. [online] Available at: https://www.workr.com.co/labs [Accessed 16 Sep. 2019].

Archivos Multimedia

[9] Tiwari, S. (2019). Quora. [online] How does Alexa voice service works?. Available at: https://www.quora.com/How-does-Alexa-voice-service-works [Accessed 16 Sep. 2019].

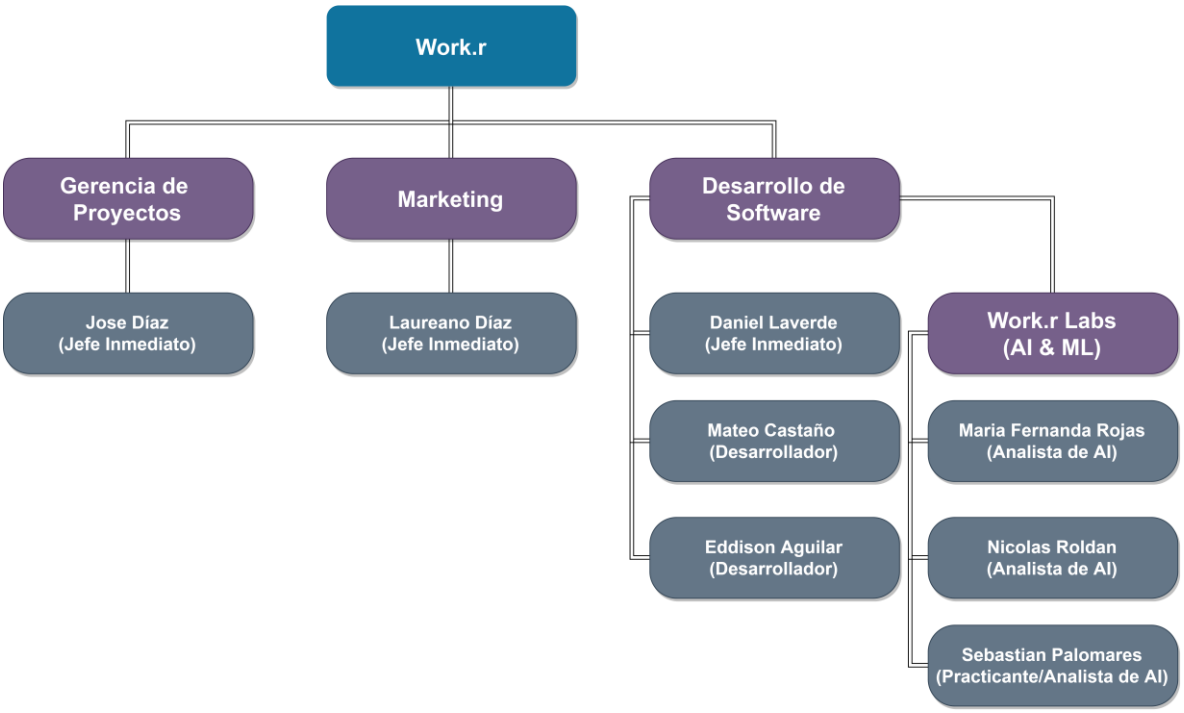
[10] Workr Labs. (2019). Como trabaja Work.r. [online] Available at: https://www.workr.com.co/labs [Accessed 16 Sep. 2019].

[11] Amazon Web Services, (2019) [onlines] “AWS” Available at: https://aws.amazon.com/blogs/opensource/recruiting-open-source-contributorsaws/ [Accessed 16 Sep. 2019].

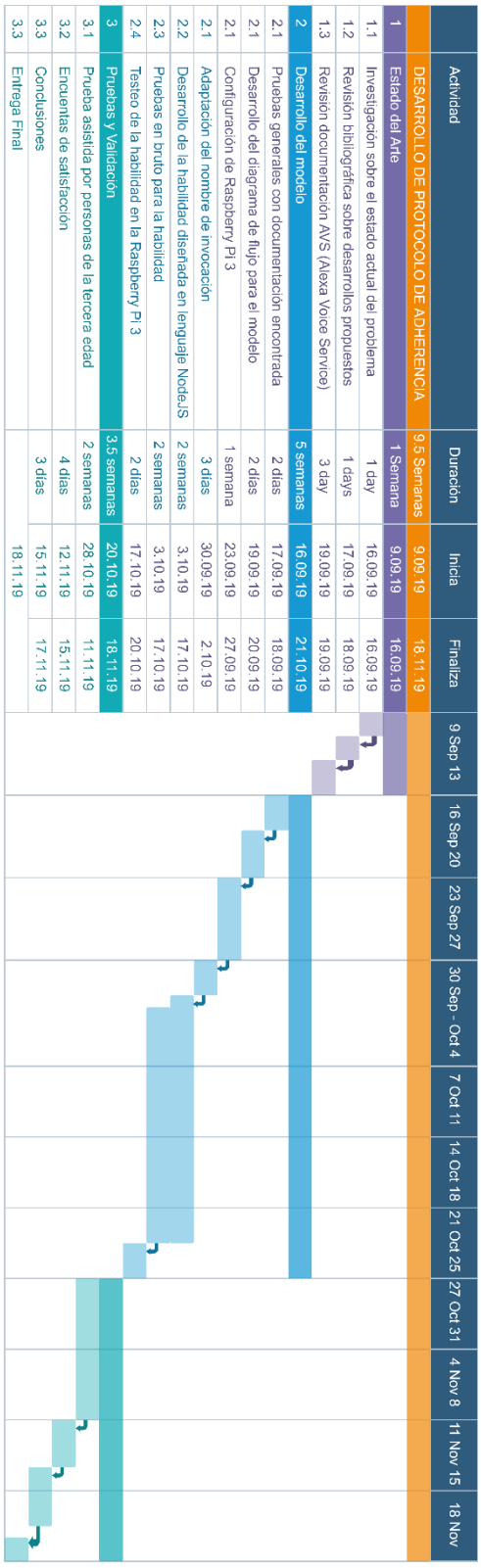
[12] Audio Technica, (2019) [onlines] “ATR2500” Available at: https://www.audio-technica.com/cms/wired_mics/84ed38489e23aa1e/index.html [Accessed 07 Dic. 2019].

ANEXOS

Anexo 1. Organigrama de la Empresa



Anexo 2. Diagrama de Gantt



Anexo 3. Código de programación

```
const Alexa = require('ask-sdk-core');
const moment = require('moment-timezone');

//Variables temporales para la asignación de slots en cada escenario.

var cantidadV = ''; //Cantidad de veces por día para la toma del medicamento
var nombrePropio = ''; // Nombre del medicamento
var nombrePersona = '';//Nombre de la persona a tratar
var hora1T = ''; // Primera y/o única hora de toma del medicamento
var hora2T = ''; // Segunda hora de toma del medicamento
var hora1Sp = ''; // Primera y/o única hora de toma del medicamento separada (SPLIT)
var hora2Sp = ''; // Segunda hora de toma del medicamento separada (SPLIT)
var dia = ''; var lunes = '';
var martes = ''; var miercoles = '';
var jueves = ''; var viernes = ''; var sabado = ''; var domingo = '';
var diasescogidos = [];

//Mensajes de ALEXA

const intro = 'Hola! Mucho gusto, mi nombre es ALEXA y soy una persona multitareas.
¿Podrías decirme tu nombre para comenzar?' const bienvenida2 = 'Muy buenas ' + nombrePersona + '! Supongo que quieres agendar un medicamento. Si es así, menciona la palabra mágica!'
//Construcción de la habilidad
const LaunchRequestHandler = {
canHandle(handlerInput) {
return Alexa.getRequestType(handlerInput.requestE nvelope) === 'LaunchRequest';
},
handle(handlerInput) {
const speakOutput = intro;
return handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse();
}
};
//Solicita el nombre del medicamento const preguntarNombreIntent = {
canHandle(handlerInput) {
return Alexa.getRequestType(handlerInput.requestE nvelope) === 'IntentRequest'
&& Alexa.getIntentName(handlerInput.requestEn velope) === 'preguntarNombre';
},
handle(handlerInput){
nombrePersona = handlerInput.requestEnvelope.request.intent.s lots.nombrep.value;
const speakOutput = 'Muy buenas ' + nombrePersona + '! Supongo que quieres agendar un medicamento. Si es así, menciona la palabra mágica!';
return handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse()
}
};

const agendarDirectoIntent = {
canHandle(handlerInput) {

return Alexa.getRequestType(handlerInput.requestE nvelope) === 'IntentRequest'
&& Alexa.getIntentName(handlerInput.requestEn velope) === 'agendarDirecto';
},
handle(handlerInput){
var med = handlerInput.requestEnvelope.request.intent.s lots.nombrem.value;
var horilla = handlerInput.requestEnvelope.request.intent.s lots.horilla.value;
const speakOutput = 'Perfecto! Agendaste' + med + ' para todos los días a las ' + horilla + ' . Si quieres agendar otro medicamento directo, menciona la palabra mágica!'
return handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse()
}
};

const PreAgendarIntent = {
canHandle(handlerInput) {
return Alexa.getRequestType(handlerInput.requestE nvelope) === 'IntentRequest'
&& Alexa.getIntentName(handlerInput.requestEn velope) === 'PreAgendar';
},
handle(handlerInput){
const speakOutput = 'Has mencionado la palabra mágica! Ahora dime el nombre del medicamento que quieres agendar para anotarlo en mi libreta.';
return handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse()
}
};

//Solicita la cantidad de veces por día del medicamento.

const NMedicamentoIntent = {
canHandle(handlerInput) {
return Alexa.getRequestType(handlerInput.requestE nvelope) === 'IntentRequest'
&& Alexa.getIntentName(handlerInput.requestEn velope) === 'AgendarMedicamento';
},
handle(handlerInput){
nombrePropio = handlerInput.requestEnvelope.request.intent.s lots.nombre.value;
const speakOutput = 'Perfecto! Ahora por favor dime cuantas veces debes tomarte el ' + ' ' + nombrePropio;
return handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse()
}
};

// Solicita la frecuencia de toma del medicamento.

const CantidadVIntent = {
canHandle(handlerInput) {
return Alexa.getRequestType(handlerInput.requestE nvelope) === 'IntentRequest'
&& Alexa.getIntentName(handlerInput.requestEn velope) === 'CantidadVeces';
```

```
},
handle(handlerInput){

cantidadV=
handlerInput.requestEnvelope.req
uest.intent.s lots.cantidad.value;
const speakOutput = 'Listo! Con
esto, vas a tomar' + ' ' + cantidadV
+ ' ' + 'veces el medicamento' + ' ' +
nombrePropio + ' ' + 'cada día. Sin
embargo, me queda una duda. ¿El
medicamento debe tomarse todos
los días? ó solamente algunos días
a la semana.'
return
handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse()
}
};

//Procedimiento para un
agendamiento diario del
medicamento.

const DiarioIntent = {
canHandle(handlerInput) {
return
Alexa.getRequestType(handlerInp
ut.requestE nvelope) ===
'IntentRequest'
&&
Alexa.getIntentName(handlerInput
.requestEn velope) === 'Diario';
},
handle(handlerInput){
if (cantidadV === "1"){
const speakOutput = 'Que bien!
Ahora,
¿A que horas te gustaría tomarte la
medicina?'; return
handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse();
}else if (cantidadV === "2"){
const speakOutput = 'Que bien!
Pero,
¿Cuales son los horarios para la
toma de los medicamentos?'
return
handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse()
}else{
return
handlerInput.responseBuilder
.speak('No está permitido el
agendamiento de medicamentos
tres veces al día. Porfavor repita su
intención teniendo en cuenta la
ingesta de una o dos veces al día.')
.reprompt('No está permitido el
agendamiento de medicamentos
tres veces al día. Porfavor repita su
intención teniendo en cuenta la
ingesta de una o dos veces al día.')
.getResponse()
}

}
};
// Agendamiento para días en
específico

const noDiarioIntent = {
canHandle(handlerInput) {
return
Alexa.getRequestType(handlerInp
ut.requestE nvelope) ===
'IntentRequest'
&&
Alexa.getIntentName(handlerInput
.requestEn velope) === 'noDiario';
},
handle(handlerInput){
const speakOutput = 'Genial! A
continuación voy a tomar nota de
los días que desees agendar el
medicamento. Por favor menciona
un día de la semana:'
return
handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse()
}
};



const dayRequestIntent = {
canHandle(handlerInput) {



return
Alexa.getRequestType(handlerInp
ut.requestE nvelope) ===
'IntentRequest'
&&
Alexa.getIntentName(handlerInput
.requestEn velope) ===
'daysRequest';
},
handle(handlerInput){ if(dia === ''){
dia =
handlerInput.requestEnvelope.req
uest.intent.s lots.day.value;
if(dia.toLowerCase() === "lunes"){
lunes = dia
diasescogidos.push('MO')
const speakOutput = 'Usted ha
escogido el día ' + dia + '. Que otro
día quieres agendar el
medicamento?'
return
handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse()
}else if (dia.toLowerCase() ===
"martes"){
martes = dia
diasescogidos.push('TU')
const speakOutput = 'Usted ha
escogido el día ' + dia + '. Que otro
día quieres agendar el
medicamento?'
return
handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse()
}else if (dia === "miércoles"){
miercoles = dia
diasescogidos.push('WE')
const speakOutput = 'Usted ha
escogido el día ' + dia + '. Que otro
día quieres agendar el
medicamento?'
return
handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse()
}else if (dia.toLowerCase() ===
"jueves"){
jueves = dia
diasescogidos.push('TH')
const speakOutput = 'Usted ha
escogido el día ' + dia + '. Que otro
día quieres agendar el
medicamento?'
return
handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse()
}else if (dia.toLowerCase() ===
"viernes"){
viernes = dia
diasescogidos.push('FR')
const speakOutput = 'Usted ha
escogido el día ' + dia + '. Que otro
día quieres agendar el
medicamento?'
return
handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse()
}else if (dia === "sábado"){ sabado
= dia diasescogidos.push('SA')
const speakOutput = 'Usted ha
escogido el día ' + dia + '. Que otro
día quieres agendar el
medicamento?'
return
handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse()
}else if (dia.toLowerCase() ===
"domingo"){
domingo = dia
diasescogidos.push('SU')
```

```
const speakOutput = 'Usted ha escogido el día ' + dia + '. Que otro día quieres agendar el medicamento?'
return
handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse()
}else{
dia = ""

const speakOutput = "Hubo un error en la obtención del día, vuelva a mencionarlo porfavor."
return
handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse()
}



}else{
dia = handlerInput.requestEnvelope.request.intent.s lots.day.value;
if(dia.toLowerCase() === "lunes"){
lunes = dia diasescogidos.push('MO')
const speakOutput = 'Escogiste el día ' + dia + '. Si deseas añadir otro día menciónalo. De lo contrario di NO MÁS!'
return
handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse()
}else if (dia.toLowerCase() === "martes"){
martes = dia diasescogidos.push('TU')
const speakOutput = 'Escogiste el día ' + dia + '. Si deseas añadir otro día menciónalo. De lo contrario di NO MÁS!'
return
handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse()
}else if (dia === "miércoles"){
miercoles = dia diasescogidos.push('WE')
const speakOutput = 'Escogiste el día ' + dia + '. Si deseas añadir otro día menciónalo. De lo contrario di NO MÁS!'
return
handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse()
}else if (dia.toLowerCase() === "jueves"){
jueves = dia diasescogidos.push('TH')
const speakOutput = 'Escogiste el día ' + dia + '. Si deseas añadir otro día menciónalo. De lo contrario di NO MÁS!'
return
handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse()
}else if (dia.toLowerCase() === "viernes"){
viernes = dia diasescogidos.push('FR')
const speakOutput = 'Escogiste el día ' + dia + '. Si deseas añadir otro día menciónalo. De lo contrario di NO MÁS!'
return
handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse()
}else if (dia === "sábado"){ sabado = dia diasescogidos.push('SA')
const speakOutput = 'Escogiste el día ' + dia + '. Si deseas añadir otro día menciónalo. De lo contrario di NO MÁS!'
return
handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse()
}else if (dia.toLowerCase() === "domingo"){
domingo = dia diasescogidos.push('SU')
const speakOutput = 'Escogiste el día ' + dia + '. Si deseas añadir otro día menciónalo. De lo contrario di NO MÁS!'
return
handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse()
}else{

const speakOutput = "Hubo un error en la obtención del día, vuelva a mencionarlo porfavor."
return
handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse()
}



}

}
};

const StopdaysRequestIntent = {
canHandle(handlerInput) {
return
Alexa.getRequestType(handlerInput.requestE nvelope) === 'IntentRequest'
&&
Alexa.getIntentName(handlerInput.requestEn velope) === 'StopdaysRequest';
},
handle(handlerInput){ if(cantidadV === '1'){
const speakOutput = 'Perfecto! Los días que escogiste fueron ' + diasescogidos + '. Si quieres corregir los días que mencionanste, di la palabra CORREGIR' +
'. De lo contrario, ¿A que horas debes tomarte el medicamento?'
return
handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse()
}else if (cantidadV === "2"){
const speakOutput = 'Perfecto! Los días que escogiste fueron ' + diasescogidos + '. Si quieres corregir los días que mencionanste, di la palabra CORREGIR' +
'. De lo contrario, menciona los horarios en los cuales debes tomarte los medicamentos.'
return
handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse()
}else{
return
handlerInput.responseBuilder
.speak('No está permitido el agendamiento de medicamentos tres veces al día. Porfavor repita su intención teniendo en cuenta la ingesta de una o dos veces al día.')
.reprompt('No está permitido el agendamiento de medicamentos tres veces al día. Porfavor repita su intención teniendo en cuenta la ingesta de una o dos veces al día.')
.getResponse()
}
}
};

const corregirRequestIntent = {
canHandle(handlerInput) {
return
Alexa.getRequestType(handlerInput.requestE nvelope) === 'IntentRequest'
&&
Alexa.getIntentName(handlerInput
```

```
.requestEn velope) === 'corregirRequest';
},
handle(handlerInput){
const speakOutput = 'Menciona que días quieres agendar tu medicamento de nuevo.'
dia = '' lunes = '' martes = '' miercoles = '' jueves = '' viernes = '' sabado = '' domingo = ''
diasescogidos = []
return handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse()
}
};



//Proceso para agendar dos momentos del día diferentes

const sHoraRequestIntent = {
canHandle(handlerInput) {
return Alexa.getRequestType(handlerInput.requestE nvelope) === 'IntentRequest'
&& Alexa.getIntentName(handlerInput.requestEn velope) === 'sHoraRequest';
},
handle(handlerInput) {
hora1T = handlerInput.requestEnvelope.request.intent.s lots.horauno.value;
hora2T = handlerInput.requestEnvelope.request.intent.s lots.horados.value;
hora1Sp = hora1T.split(":");
hora2Sp = hora2T.split(":");
const speakOutput = 'La toma del medicamento ' + nombrePropio + ' se agendará para todos los días, ' + cantidadV + ' veces al día, a las ' + hora1T + ' y a las ' + hora2T + '. Si esta información es correcta, por favor decir la palabra CREAR'
return handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse();
}
};

}
if(dia === ''){
const enviarReminder = {} if (cantidadV === "1"){
const currentDataTime = moment().tz('America/Bogota'),
enviarReminder = { requestTime: currentDataTime.format('YYYY-MM- DDTHH:mm:ss'),
trigger: { type: 'SCHEDULED_ABSOLUTE',
scheduledTime: currentDataTime.set({
hour: hora1Sp[0], minute: hora1Sp[1], second: '00'
}).format('YYYY-MM-DDTHH:mm:ss'),
timeZoneId: 'America/Bogota',
recurrence: { freq: 'DAILY',
}
},
alertInfo: { spokenInfo: {
content: [{
locale: 'es-MX',
text: nombrePersona
+ ' Ya es hora de tomarse el ' + nombrePropio
+ '. Recuerde tomar agua para no atorarse!',
}],

const enviarReminder2 = {
requestTime: currentDataTime.format('YYYY-MM- DDTHH:mm:ss'),
trigger: { type: 'SCHEDULED_ABSOLUTE',
scheduledTime: currentDataTime.set({
hour: hora2Sp[0], minute: hora2Sp[1], second: '00'
}).format('YYYY-MM-DDTHH:mm:ss'),
timeZoneId: 'America/Bogota',
recurrence: { freq: 'DAILY'
}
},
alertInfo: { spokenInfo: {
content: [{
locale: 'es-MX',
text: nombrePersona
+ ' Ya es hora de tomarse el ' + nombrePropio
+ '. Recuerde tomar agua para no atorarse!',
}],
},
},
pushNotification: { status: 'ENABLED',
}
}

//Proceso para agendar el medicamento para la ingesta única al día.
},
},


const HoraRequestIntent = {
canHandle(handlerInput) {
return Alexa.getRequestType(handlerInput.requestE nvelope) === 'IntentRequest'
&& Alexa.getIntentName(handlerInput.requestEn velope) === 'HoraRequest';
},
handle(handlerInput) {
hora1T = handlerInput.requestEnvelope.request.intent.s lots.phora.value;
hora1Sp = hora1T.split(":");
const speakOutput = 'La toma del medicamento ' + nombrePropio + ' se agendará para todos los días, ' + cantidadV + ' veces al día, a las ' + hora1T + '. Si esta información es correcta, por favor decir la palabra CREAR'
return handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse();
}
};
//Proceso para crear el recordatorio DIARIO. const CreacionReminderIntent = {
canHandle(handlerInput) { return Alexa.getRequestType(handlerInput.requestE nvelope) === 'IntentRequest'
&& Alexa.getIntentName(handlerInput.requestEn velope) === 'CreacionReminder';
},
async handle(handlerInput){
const reminderApiClient = handlerInput.serviceClientFactory.getRemind erManagementServiceClient(),
{ permissions } = handlerInput.requestEnvelope.context.System .user
if (!permissions){
return handlerInput.responseBuilder
.speak("Primero habilite los permisos desde el app de Alexa")

.withAskForPermissionsConsentCard(['alexa:: alerts:reminders:skill:readwrite'])
.getResponse()

pushNotification: { status: 'ENABLED',
}
}


reminderApiClient.createReminder(enviarRem inder)
```

```
const speakOutput = 'Listo! El agendamiento se ha creado satisfactoriamente.'
return handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse()
}else{
const currentDataTime = moment().tz('America/Bogota'),
enviarReminder = { requestTime: currentDataTime.format('YYYY-MM- DDTHH:mm:ss'),
trigger: { type: 'SCHEDULED_ABSOLUTE',
scheduledTime: currentDataTime.set({
hour: hora1Sp[0], minute: hora1Sp[1], second: '00'
}).format('YYYY-MM-DDTHH:mm:ss'),
timeZoneId: 'America/Bogota',
recurrence: { freq: 'DAILY'
}
},
alertInfo: { spokenInfo: {
content: [{
locale: 'es-MX',
text: nombrePersona
+ ' Ya es hora de tomarse el ' + nombrePropio
+ '. Recuerde tomar agua para no atorarse!',
}],
},
},
pushNotification: { status: 'ENABLED',
}
}

reminderApiClient.createReminder(enviarRem inder)

reminderApiClient.createReminder(enviarRem inder2)

const speakOutput = 'Listo! El agendamiento se ha creado satisfactoriamente.'
return handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse()
}
}else if(dia !== ''){
const enviarReminder = {} if (cantidadV === "1"){
const currentDataTime = moment().tz('America/Bogota'),
enviarReminder = { requestTime: currentDataTime.format('YYYY-MM- DDTHH:mm:ss'),
trigger: { type: 'SCHEDULED_ABSOLUTE',
scheduledTime: currentDataTime.set({
hour: hora1Sp[0], minute: hora1Sp[1], second: '00'
}).format('YYYY-MM-DDTHH:mm:ss'),
timeZoneId: 'America/Bogota',
recurrence: {
freq: 'WEEKLY',
byDay: diasescogidos
}
},
alertInfo: { spokenInfo: {
content: [{
locale: 'es-MX',
text: nombrePersona
+ ' Ya es hora de tomarse el ' + nombrePropio
+ '. Recuerde tomar agua para no atorarse!',
}],
},
},
pushNotification: { status: 'ENABLED',
}

}

reminderApiClient.createReminder(enviarRem inder)

const speakOutput = 'Listo! El agendamiento se ha creado satisfactoriamente.'
return handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse()
}else{
const currentDataTime = moment().tz('America/Bogota'),
enviarReminder = { requestTime: currentDataTime.format('YYYY-MM- DDTHH:mm:ss'),
trigger: { type: 'SCHEDULED_ABSOLUTE',
scheduledTime: currentDataTime.set({
hour: hora1Sp[0], minute: hora1Sp[1], second: '00'
}).format('YYYY-MM-DDTHH:mm:ss'),
timeZoneId: 'America/Bogota',
recurrence: {
freq: 'WEEKLY',
byDay: diasescogidos
}
},
alertInfo: { spokenInfo: {
content: [{
locale: 'es-MX',
text: 'Es hora de tu pastilla',
}],
},
},
pushNotification: { status: 'ENABLED',
}
}
const enviarReminder2 = {
requestTime: currentDataTime.format('YYYY-MM- DDTHH:mm:ss'),
trigger: { type: 'SCHEDULED_ABSOLUTE',
scheduledTime: currentDataTime.set({
hour: hora2Sp[0], minute: hora2Sp[1], second: '00'
}).format('YYYY-MM-DDTHH:mm:ss'),
timeZoneId: 'America/Bogota',
recurrence: {
freq: 'WEEKLY',
byDay: diasescogidos
}
},
alertInfo: { spokenInfo: {
content: [{
locale: 'es-MX',
text: 'Es hora de tu pastilla',
}],
},
},
pushNotification: { status: 'ENABLED',
}
}
reminderApiClient.createReminder(enviarRem inder)

reminderApiClient.createReminder(enviarRem inder2)

const speakOutput = 'Listo! El agendamiento se ha creado satisfactoriamente.'
return handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse()
}
}
}
};
```

```
//Procesos para agendar en días específicos

//Procesos Default que ALEXA requiere para trabajar.

const HelpIntentHandler = {
canHandle(handlerInput) {
return Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest'
&& Alexa.getIntentName(handlerInput.requestEnvelope) === 'AMAZON.HelpIntent';
},
handle(handlerInput) {
const speakOutput = 'You can say hello to me! How can I help?';
return handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse();
}
};

const YesIntentHandler = {
canHandle(handlerInput) {
return Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest'
&& Alexa.getIntentName(handlerInput.requestEnvelope) === 'AMAZON.YesIntent';
},
handle(handlerInput) {
const speakOutput = 'Perfecto!';
return handlerInput.responseBuilder
.speak(speakOutput)
.reprompt(speakOutput)
.getResponse();
}
};

const CancelAndStopIntentHandler = {
canHandle(handlerInput) {
return Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest'
&& (Alexa.getIntentName(handlerInput.requestEnvelope) === 'AMAZON.CancelIntent'
|| Alexa.getIntentName(handlerInput.requestEnvelope) === 'AMAZON.StopIntent');
},
handle(handlerInput) {
const speakOutput = 'Hasta luego!';
return handlerInput.responseBuilder
.speak(speakOutput)
.getResponse();
}
};

const SessionEndedRequestHandler = {
canHandle(handlerInput) {
return Alexa.getRequestType(handlerInput.requestEnvelope) === 'SessionEndedRequest';
},
handle(handlerInput) { return handlerInput.responseBuilder.getResponse();
}
};


const IntentReflectorHandler = {
canHandle(handlerInput) {
return Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest';
},
handle(handlerInput) {
const intentName = Alexa.getIntentName(handlerInput.requestEnvelope);
const speakOutput = `You just triggered ${intentName}`;

return handlerInput.responseBuilder
.speak(speakOutput)
.getResponse();
}
};

const ErrorHandler = { canHandle() {
return true;
},
handle(handlerInput, error) {
console.log(`~~~~ Error handled: ${error.stack}`);
const speakOutput = `Sorry, I had trouble doing what you asked. Please try again.`;

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

exports.handler = Alexa.SkillBuilders.custom()
.addRequestHandlers(
LaunchRequestHandler,
CreacionReminderIntent,
PreAgendarIntent,
NMedicamentoIntent,
preguntarNombreIntent,
HoraRequestIntent,
sHoraRequestIntent,
CantidadVIntent, noDiarioIntent,
DiarioIntent,
StopdaysRequestIntent,
corregirRequestIntent,
agendarDirectoIntent,
dayRequestIntent,
YesIntentHandler,
HelpIntentHandler,
CancelAndStopIntentHandler,
SessionEndedRequestHandler,
IntentReflectorHandler)
.addErrorHandlers( ErrorHandler)
.withApiClient(new Alexa.DefaultApiClient())
.lambda();
```