



**“Diseño e implementación de un sistema Cloud Robotics basado en la plataforma humanoide NAO, para robótica social”**

Gia Díaz, Janis Michelle

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica, Automatización y Control

Trabajo de titulación, previo a la obtención del título de  
Ingeniera en Electrónica, Automatización y Control

Ing. Orozco Brito, Luis Alberto MSc.

22 de julio de 2021



## Urkund Analysis Result

Analysed Document: GiaJ\_TrabajoTitulación\_SinAnexos.pdf (D110768131)  
 Submitted: 7/26/2021 4:58:00 PM  
 Submitted By: laorozco@espe.edu.ec  
 Significance: 4 %



LUIS ALBERTO  
 OROZCO BRITO

### Sources included in the report:

Tesis Dayanna Silva.pdf (D78255919)  
 Implementación y evaluación de patrones y buenas prácticas para mejorar el rendimiento en aplicaciones Serverless.pdf (D67753238)  
<http://docplayer.es/208703448-Departamento-de-electrica-y-electronica.html>  
<https://dspace.ups.edu.ec/handle/123456789/14031>  
<https://blog.usejournal.com/introduction-to-json-and-restful-apis-coding-bootcamp-series-7ad6aa294c89>  
<https://link.springer.com/article/10.1007/s12369-016-0351-1>  
<https://ieeexplore.ieee.org/document/8102694>  
<https://dspace.ups.edu.ec/handle/123456789/15012>  
<https://dspace.ups.edu.ec/handle/123456789/14020?locale=es>  
<https://citelia.es/blog/que-es-cloud-computing-y-como-funciona/>  
<https://www.datacentric.es/blog/business/procesamiento-lenguaje-natural-revolucion-futuro/>  
<https://www.devdungeon.com/content/install-gcc-compiler-windows-msys2-cc>  
<https://www.elcomercio.com/guaifai/nuka-foca-robot-mejora-salud.html>  
<https://blog.makeitreal.camp/el-protocolo-http/>  
<https://www.euronews.com/2016/03/28/meet-little-casper-a-robot-designed-to-help-children-suffering-from-cancer>  
<https://dspace.ups.edu.ec/handle/123456789/15287>  
<https://dspace.ups.edu.ec/handle/123456789/13083>  
<http://www.dspace.uce.edu.ec/bitstream/25000/1965/1/T-UCE-0010-307.pdf>  
<https://news.gallup.com/poll/309203/increased-low-contact-services-may-prove-permanent.aspx>  
<https://intellipaat.com/blog/aws-vs-azure-vs-google-cloud/>  
<https://blog.hubspot.com/website/what-is-rest-api>  
<https://docplayer.es/6438559-Servicios-web-antecedentes-y-justificacion-soap-wsdl-uddi-utilizacion-de-servicios-web-creacion-de-servicios-web.html>  
<https://www8.gsb.columbia.edu/articles/brand-talk/pandemic-and-smarter-world-future-robots>  
<https://www.therobotreport.com/pandemic-proves-utility-wide-range-service-robots/>  
<https://azure.microsoft.com/es-es/overview/what-is-paas/>  
<https://azure.microsoft.com/es-es/overview/what-is-saas/>  
<https://dspace.ups.edu.ec/handle/123456789/14159>



**DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y  
TELECOMUNICACIONES**

**CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y  
CONTROL**

**CERTIFICACIÓN**

Certifico que el trabajo de titulación, “**Diseño e implementación de un sistema Cloud Robotics basado en la plataforma humanoide NAO, para robótica social**” fue realizado por la señorita **Gia Díaz, Janis Michelle** el cual ha sido revisado y analizado en su totalidad por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Sangolquí, 16 de julio de 2021

Firma:



Firmado electrónicamente por  
**LUIS ALBERTO  
OROZCO BRITO**

**Ing. Orozco Brito, Luis Alberto MSc.**

C. C.: 171044380-3



**DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y  
TELECOMUNICACIONES**

**CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y  
CONTROL**

**RESPONSABILIDAD DE AUTORÍA**

Yo, **Gia Díaz, Janis Michelle**, con cédula de ciudadanía n°175008120-8, declaro que el contenido, ideas y criterios del trabajo de titulación: **“Diseño e implementación de un sistema Cloud Robotics basado en la plataforma humanoide NAO, para robótica social”** es de mi autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 16 de julio de 2021

Firma:

**Gia Díaz, Janis Michelle**

C.C.: 175008120-8



**DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y  
TELECOMUNICACIONES**

**CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y  
CONTROL**

**AUTORIZACIÓN DE PUBLICACIÓN**

Yo **Gia Díaz, Janis Michelle**, con cédula de ciudadanía n°175008120-8 ,autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: **“Diseño e implementación de un sistema Cloud Robotics basado en la plataforma humanoide NAO, para robótica social”** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Sangolquí, 16 de julio de 2021

Firma:

**Gia Díaz, Janis Michelle**

C.C.: 175008120-8

## Dedicatoria

Dedico este trabajo de titulación, primero a Dios por darme fortaleza para culminar exitosamente tan maravillosa carrera.

A mi madre, María Díaz, por ser ese apoyo incansable y constante. Te amo madre mía, sabes que todo esto lo hago por ti. Eres lo más importante en mi vida, mi razón de ser. Seguiré trabajando día a día para hacerte feliz y lograr todos nuestros sueños y nuestras metas. Culmina un capítulo, y empieza uno nuevo para nosotras, que estará lleno de felicidad y éxitos.

A mis padres, Rubén Narváez y Jhimy Gia que me brindaron siempre palabras de aliento para seguir adelante. Me siento muy afortunada de tenerlos en mi vida. No podría pedir más.

A mi familia entera por creer en mí aun cuando todo parecía perdido. Gracias por ser la maravillosa familia que son, tan unidos, tan amorosos.

A mi abuelita, Elsa Quintana, que ahora se encuentra en el cielo. Lamento que ya no estés aquí para presenciarlo, pero espero de todo corazón que te sientas orgullosa del legado de grandes mujeres y hombres que has dejado en nuestra familia. Te extrañamos cada día.

## **Agradecimiento**

Agradezco, en primer lugar, a mis padres por haberme acompañado a lo largo de este duro camino, por siempre confiar en mí y levantarme cuando he desfallecido. No hay suficientes palabras para agradecerles todo lo que han hecho por mí, pero tengan por seguro que recompensaré cada esfuerzo que han invertido en mi persona, para que lograra llegar hasta aquí.

Agradezco a la persona que ha estado a mi lado los últimos 5 años, brindándome cariño y apoyo, en las buenas y las malas. Luis, hoy empieza una nueva etapa en mi vida, y espero con ansias compartirla contigo. Gracias por tu apoyo incondicional en el trabajo, la familia, las pruebas del proyecto, en todo.

Gracias a los maravillosos amigos y compañeros a lo largo de la carrera, guardo con cariño todas y cada una de las experiencias vividas, alegrías, tristezas, éxitos y fracasos. Gracias por compartir conmigo un pedacito de sus vidas.

Agradezco a los compañeros e ingenieros que me han brindado una oportunidad para demostrar mi capacidad, mis talentos y mis habilidades en el ámbito de la ingeniería, que han depositado su confianza en mí y me han abierto varias puertas a experiencias laborales únicas.

Un agradecimiento especial al Ing. Luis Alberto Orozco Brito, mi director de proyecto, por su paciencia y ayuda en la ejecución y culminación exitosa de este trabajo de titulación.

Sin más que decir...Gracias Totales...

**Janis Michelle Gia Diaz**

## Índice de Contenido

Hoja de resultados de la herramienta Urkund.....	2
Certificación de trabajo de titulación .....	3
Responsabilidad de Autoría.....	4
Autorización de publicación .....	5
Dedicatoria .....	6
Agradecimiento .....	7
Índice de Contenido.....	8
Índice de Tablas .....	13
Índice de Figuras.....	15
Resumen.....	22
Abstract .....	23
Capítulo I.....	24
Planteamiento del problema de investigación.....	24
Antecedentes .....	24
Justificación e importancia .....	26
Alcance del proyecto.....	28
Objetivos.....	29
Objetivo General .....	29
Objetivos Específicos .....	30



	9
Capítulo II.....	31
Fundamentación Teórica.....	31
Robótica Social.....	31
Interacción Humano-Robot.....	32
Clasificación de la Interacción Humano - Robot.....	33
Robot Humanoide NAO.....	35
Características Técnicas Robot Humanoide Nao.....	35
NAOqi.....	38
Choregraphe.....	38
Servicios Web.....	41
RESTFUL API.....	43
Interfaz de Programación de Aplicaciones (API).....	43
Transferencia De Estado Representacional (REST).....	43
Protocolo HTTP.....	45
Mensajes HTTP.....	46
Métodos HTTP.....	48
JSON.....	48
Servicios en la nube.....	51
Cloud Robotics.....	52
Capitulo III.....	54

	10
Diseño del sistema Cloud Robotics para desarrollo de aplicativos de robótica social ....	54
Selección de plataforma de servicios en la nube.....	54
Proceso de análisis jerárquico (AHP) y la toma de decisiones multicriterio ....	55
Diseño de arquitectura para sistema Cloud Robotics .....	60
Diseño de aplicativos para robótica social.....	63
Aplicativo para educación .....	63
Planteamiento del problema. ....	63
Diseño de la aplicación.....	65
Aplicativo para cuidados de la salud .....	67
Planteamiento del problema. ....	67
Diseño de la aplicación.....	68
Aplicativo para atención al cliente .....	71
Planteamiento del problema. ....	71
Diseño de la aplicación.....	73
Capitulo IV.....	76
Implementación, pruebas y resultados .....	76
Implementación de aplicaciones orientadas a la robótica social.....	76
Diseño de algoritmo de programación para la utilización de servicios en la nube .....	76
Librerías y variables. ....	76

	11
Datos a enviar .....	77
Petición. ....	79
Respuesta. ....	80
Diseño de bloques de programación. ....	82
Implementación del algoritmo de Python en Choregraphe. ....	84
Adición de bloques complementarios. ....	86
Creación de un bloque de programación principal. ....	89
Aplicativo para educación del idioma inglés .....	92
Reconocimiento del usuario. ....	92
Nivel 1 – Relación imagen/palabra y práctica de pronunciación. ....	94
Nivel 2 – Listening y writing. ....	98
Almacenamiento en base de datos.....	101
Aplicativo para cuidados de la salud .....	105
Autenticación OAuth2.0.....	106
Agendamiento de Eventos. ....	113
Consulta de Eventos. ....	115
Aplicativo para atención al cliente .....	117
Creación Chatbot en Dialogflow. ....	117
Almacenamiento de Datos en Firebase.....	125
Autorización de cuenta de servicio sin OAuth.....	129

	12
Implementación en Choregraphe. ....	133
Desarrollo de laboratorios teórico/prácticos .....	138
Pruebas .....	141
Aplicativo para educación .....	141
Aplicativo para cuidados de la salud .....	145
Aplicativo para atención al cliente .....	148
Resultados.....	150
Aplicativo para educación .....	150
Aplicativo para cuidados de la salud .....	170
Aplicativo para atención al cliente .....	180
Capítulo V .....	190
Conclusiones y Recomendaciones.....	190
Conclusiones .....	190
Recomendaciones .....	194
Trabajos Futuros.....	195
Referencias Bibliográficas .....	196
Anexos .....	214

## Índice de Tablas

<b>Tabla 1</b> Sensores y actuadores robot NAO .....	35
<b>Tabla 2</b> Criterios y valores de las plataformas de servicios en la nube.....	56
<b>Tabla 3</b> Escala de importancia relativa de Saaty.....	57
<b>Tabla 4</b> Compilación de vectores promedios.....	59
<b>Tabla 5</b> Matriz de Razón de Consistencia de los criterios evaluados .....	60
<b>Tabla 6</b> Ejemplos de códigos de estado de respuesta HTTP .....	62
<b>Tabla 7</b> Nombre de entidades y tipo de datos a obtener en el chatbot .....	122
<b>Tabla 8</b> Pacientes para pruebas del aplicativo para cuidados de la salud.....	146
<b>Tabla 9</b> Resultados reconocimiento de imágenes en escala de grises.....	151
<b>Tabla 10</b> Resultados reconocimiento de imágenes a color.....	152
<b>Tabla 11</b> Resultados reconocimiento de imágenes con tarjetas formato A5.....	154
<b>Tabla 12</b> Resultados reconocimiento de imágenes con tarjetas formato A7.....	155
<b>Tabla 13</b> Resultados reconocimiento de imágenes con luz puntual .....	157
<b>Tabla 14</b> Resultados reconocimiento de imágenes con luz ambiental.....	159
<b>Tabla 15</b> Tabulación del número de aciertos obtenido del servicio Cloud Vision API .	161
<b>Tabla 16</b> Tabulación del número de aciertos obtenidos por el usuario .....	163
<b>Tabla 17</b> Cuota gratuita de servicios en la nube utilizados en el aplicativo de educación .....	165
<b>Tabla 18</b> Tabulación de servicios en una sesión del aplicativo de educación.....	166
<b>Tabla 19</b> Tabulación del uso gratuito del aplicativo de educación .....	166
<b>Tabla 20</b> Cuadro comparativo de solución offline y online para el reconocimiento óptico de caracteres.....	168
<b>Tabla 21</b> Tabulación de aciertos y errores en la transcripción de nombres de pacientes y medicamentos .....	171

<b>Tabla 22</b> Tabulación del número de eventos obtenidos en diferentes horas de consulta .....	175
<b>Tabla 23</b> Cuota gratuita de servicios en la nube utilizados en el aplicativo de cuidados de la salud.....	177
<b>Tabla 24</b> Tabulación de servicios en una sesión del aplicativo de cuidados de la salud .....	177
<b>Tabla 25</b> Tabulación del uso gratuito del aplicativo de cuidados de la salud .....	178
<b>Tabla 26</b> Cuadro comparativo de la solución offline y online en el almacenamiento de datos .....	179
<b>Tabla 27</b> Resultados reconocimiento de voz a diferentes distancias .....	181
<b>Tabla 28</b> Tabulación del número de aciertos obtenido del servicio SpeechToText API .....	184
<b>Tabla 29</b> Cuota gratuita de servicios en la nube utilizados en el aplicativo de atención al cliente.....	185
<b>Tabla 30</b> Tabulación de servicios en una sesión del aplicativo de atención al cliente.	186
<b>Tabla 31</b> Tabulación del uso gratuito del aplicativo de atención al cliente .....	186
<b>Tabla 32</b> Cuadro comparativo de la solución offline y online en la creación de diálogos .....	189

## Índice de Figuras

<b>Figura 1</b> Robot Little Casper en Instituto Portugués de Oncología en Lisbon.....	32
<b>Figura 2</b> Framework NAOqi .....	38
<b>Figura 3</b> Software Choregraphe.....	39
<b>Figura 4</b> Bloque de programación: A. Entradas - B. Salidas - C. Parámetros.....	39
<b>Figura 5</b> Plantillas de los tipos de bloques.....	40
<b>Figura 6</b> Arquitectura Servicio Web SOAP.....	42
<b>Figura 7</b> Arquitectura Servicio Web REST.....	42
<b>Figura 8</b> Funcionamiento de una API.....	43
<b>Figura 9</b> Arquitectura básica del protocolo HTTP.....	46
<b>Figura 10</b> Partes de los mensajes HTTP .....	47
<b>Figura 11</b> Sintaxis objeto JSON.....	48
<b>Figura 12</b> Sintaxis de un array JSON.....	49
<b>Figura 13</b> Petición y respuesta a una API REST.....	50
<b>Figura 14</b> Tipos de servicios en la nube.....	52
<b>Figura 15</b> Arquitectura básica Cloud Robotics .....	53
<b>Figura 16</b> Arquitectura de sistema Cloud Robotics para robótica social, utilizando el autómata NAO.....	61
<b>Figura 17</b> Puntajes EF EPI de los países de América Latina en 2019.....	63
<b>Figura 18</b> Flashcards .....	64
<b>Figura 19</b> Arquitectura de sistema Cloud Robotics para la enseñanza del idioma inglés mediante flashcards .....	65
<b>Figura 20</b> Diagrama de flujo del aplicativo de educación .....	66
<b>Figura 21</b> Robot Healthbot siendo utilizado por una persona mayor .....	68

<b>Figura 22</b> Arquitectura de sistema Cloud Robotics para el recordatorio de ingesta de medicamentos.....	69
<b>Figura 23</b> Diagrama de flujo del aplicativo de salud.....	70
<b>Figura 24</b> Robot Loweobot guiando al cliente a través de la tienda Lowe.....	72
<b>Figura 25</b> Robot Tally en revisión de inventario en tiempo real.....	73
<b>Figura 26</b> Arquitectura de sistema Cloud Robotics para la aplicación del chatbot en atención al cliente.....	74
<b>Figura 27</b> Diagrama de flujo de aplicativo de atención al cliente.....	75
<b>Figura 28</b> Código fuente del programa de librerías y variables para el uso del API Speech To Text.....	77
<b>Figura 29</b> Sintaxis de un objeto diccionario en Python.....	78
<b>Figura 30</b> Código fuente del programa del cuerpo de la solicitud para el uso del API Speech To Text.....	78
<b>Figura 31</b> Sintaxis de petición HTTP utilizando el módulo requests.....	79
<b>Figura 32</b> Código fuente del programa de petición HTTP para el uso del API Speech To Text.....	79
<b>Figura 33</b> Sintaxis de petición HTTP utilizando el método post.....	80
<b>Figura 34</b> Código fuente del programa de la respuesta obtenida del API Speech To Text.....	81
<b>Figura 35</b> Impresión de la variable data con la respuesta a la petición HTTP en Python.....	82
<b>Figura 36</b> Texto obtenido del audio procesado por el API Speech To Text.....	82
<b>Figura 37</b> Bloque de programación en Choregraphe para código en Python.....	83
<b>Figura 38</b> Script base del bloque de programación Python en Choregraphe.....	83
<b>Figura 39</b> Bloque de programación para el uso del API Speech To Text, en Choregraphe.....	84



<b>Figura 40</b> Configuración del bloque de programación para el uso del API Speech To Text.....	85
<b>Figura 41</b> Bloque de programación para grabación de audio en Choregraphe .....	86
<b>Figura 42</b> Configuración del bloque de programación para grabación de audio en Choregraphe .....	86
<b>Figura 43</b> Bloque de programación para reproducción de sonidos, en Choregraphe ....	87
<b>Figura 44</b> Configuración del bloque de programación para reproducción de audio en Choregraphe .....	88
<b>Figura 45</b> Conexión de bloques de programación para el uso del API Speech To Text	89
<b>Figura 46</b> Bloque de programación madre para el uso del API Speech To Text en el robot NAO .....	90
<b>Figura 47</b> Conexión de bloques, dentro del bloque madre, del API Speech To Text.....	90
<b>Figura 48</b> Diagrama de flujo del funcionamiento del bloque SpeechToText API .....	91
<b>Figura 49</b> Diagrama de flujo del reconocimiento de usuario para aplicativo de educación.....	93
<b>Figura 50</b> Diagrama de bloques del reconocimiento del usuario para aplicativo de educación.....	94
<b>Figura 51</b> Bloque de programación para el uso del API Cloud Vision en el robot NAO .	95
<b>Figura 52</b> Bloque de programación para el uso del Cloud Vision API para reconocimiento de imágenes, en Choregraphe .....	95
<b>Figura 53</b> Diagrama de flujo del funcionamiento del bloque Cloud Vision API .....	96
<b>Figura 54</b> Diagrama de flujo del Nivel 1 del aplicativo de educación.....	97
<b>Figura 55</b> Bloque de programación para el uso del Cloud Vision API para reconocimiento de imágenes, en Choregraphe .....	99
<b>Figura 56</b> Diagrama de flujo del Nivel 2 del aplicativo de educación.....	100
<b>Figura 57</b> Bloque de programación para el envío y almacenamiento de datos en la nube, en Python .....	101

<b>Figura 58</b> Diagrama de flujo del almacenamiento de datos en Firebase .....	102
<b>Figura 59</b> Información almacenada en la base de datos en tiempo real de Firebase ..	103
<b>Figura 60</b> Página web para la consulta de puntajes por usuario para el aplicativo de educación.....	104
<b>Figura 61</b> Tabla de puntajes del usuario de nombre James – aplicativo de educación .....	105
<b>Figura 62</b> Diagrama de secuencia del proceso de autenticación para aplicaciones en dispositivos de entrada limitada.....	106
<b>Figura 63</b> Bloque de programación para la realización del paso 1 y 2 de la autenticación OAuth 2.0.....	108
<b>Figura 64</b> Ingreso a URL de verificación desde dispositivo externo .....	109
<b>Figura 65</b> Ingreso de código de usuario dictado por el robot NAO .....	109
<b>Figura 66</b> Inicio sesión en cuenta de Gmail para el acceso a Google Calendar.....	110
<b>Figura 67</b> Pantalla de consentimiento para el acceso de Google Calendar API a la cuenta del usuario .....	110
<b>Figura 68</b> Pantalla de confirmación de conexión exitosa.....	111
<b>Figura 69</b> Bloque de programación para la realización de los pasos 1, 2 y 3 de la autenticación OAuth 2.0 .....	111
<b>Figura 70</b> Bloque de programación para la realización del paso 4 de la autenticación OAuth 2.0.....	112
<b>Figura 71</b> Diagrama de flujo del funcionamiento del bloque de agendamiento de eventos.....	114
<b>Figura 72</b> Diagrama de flujo del funcionamiento del bloque de consulta de eventos..	116
<b>Figura 73</b> Creación de agente en Dialogflow .....	118
<b>Figura 74</b> Flujo básico para la coincidencia de intents .....	119
<b>Figura 75</b> Intents creados para el chatbot en Dialogflow.....	120
<b>Figura 76</b> Frases de entrenamiento del intent ReservaRobot .....	120

<b>Figura 77</b> <i>Parámetro tipo de robot del intent ReservaRobot .....</i>	121
<b>Figura 78</b> <i>Creación de entidad robot en Dialogflow.....</i>	122
<b>Figura 79</b> <i>Diagrama de flujo del funcionamiento del chatbot para la reserva de robots fijos y móviles.....</i>	123
<b>Figura 80</b> <i>Diagrama de flujo del funcionamiento del chatbot para la reserva de robots fijos y móviles.....</i>	124
<b>Figura 81</b> <i>Diagrama de flujo del funcionamiento del chatbot para la reserva de robots fijos y móviles.....</i>	125
<b>Figura 82</b> <i>Inline Editor en Dialogflow.....</i>	126
<b>Figura 83</b> <i>Habilitación de Webhook para el intent GuardarDatos.....</i>	127
<b>Figura 84</b> <i>Diagrama de flujo del almacenamiento de datos en Firebase y funciones creadas en Inline Editor.....</i>	128
<b>Figura 85</b> <i>Diagrama de secuencia del proceso de autenticación para aplicaciones de servidor a servidor.....</i>	129
<b>Figura 86</b> <i>Código fuente del programa para la generación del token JWT en Python</i>	131
<b>Figura 87</b> <i>Diagrama de flujo del aplicativo para la generación del token JWT en Python .....</i>	132
<b>Figura 88</b> <i>Funcionamiento de la aplicación de generación de JWT .....</i>	133
<b>Figura 89</b> <i>Bloque de programación para la lectura del token de la base de datos en la nube .....</i>	134
<b>Figura 90</b> <i>Diagrama de flujo para la lectura del token de la base de datos en la nube .....</i>	135
<b>Figura 91</b> <i>Bloque de programación para la utilización de API de Dialogflow.....</i>	135
<b>Figura 92</b> <i>Diagrama de flujo para la utilización de API de Dialogflow.....</i>	136
<b>Figura 93</b> <i>Diagrama de flujo del programa de la aplicación para atención al cliente ..</i>	137
<b>Figura 94</b> <i>Laboratorio #1 - Instalación software Choregraphe.....</i>	138
<b>Figura 95</b> <i>Laboratorio #2 - Instalación software Webots con Choregraphe .....</i>	139

<b>Figura 96</b> Laboratorio #3 - Utilización de servicios en la nube (SpeechToText API) en Choregraphe .....	140
<b>Figura 97</b> Inducción del usuario al aplicativo de educación.....	142
<b>Figura 98</b> Tarjetas utilizadas en escala de grises y a color .....	142
<b>Figura 99</b> Tarjetas utilizadas en formato A5 y A7.....	143
<b>Figura 100</b> Luz ambiental y luz puntual.....	144
<b>Figura 101</b> Inducción del usuario a la aplicación de cuidados de la salud.....	145
<b>Figura 102</b> Página de Google Calendar de la cuenta autorizada .....	147
<b>Figura 103</b> Inducción del usuario a la aplicación de atención al cliente.....	148
<b>Figura 104</b> Pruebas de funcionamiento del aplicativo de educación .....	150
<b>Figura 105</b> Diagrama de barras del número de aciertos obtenido del servicio Cloud Vision API.....	161
<b>Figura 106</b> Resultados del usuario en el juego del aplicativo educación.....	162
<b>Figura 107</b> Gráficas de pastel, por nivel, del número de aciertos obtenidos por el usuario .....	163
<b>Figura 108</b> Resultados obtenidos de la participación del usuario en el aplicativo de educación.....	164
<b>Figura 109</b> Evaluación al usuario de las palabras aprendidas en inglés .....	164
<b>Figura 110</b> Resultados solución offline y online para el reconocimiento óptico de caracteres .....	169
<b>Figura 111</b> Pruebas de funcionamiento del aplicativo de cuidados de la salud.....	170
<b>Figura 112</b> Consulta de pacientes registrados en la aplicación cuidados de la salud.....	170
<b>Figura 113</b> Porcentaje de efectividad del SpeechToText API para realizar las transcripciones .....	172
<b>Figura 114</b> Agendamiento de eventos en Google Calendar.....	173
<b>Figura 115</b> Ejecución de la consulta de eventos en Google Calendar.....	174

<b>Figura 116</b> Eventos almacenados en Google Calendar .....	175
<b>Figura 117</b> Resultado del número de eventos obtenidos en diferentes horas de consulta.....	176
<b>Figura 118</b> Pruebas de funcionamiento del aplicativo de atención al cliente .....	180
<b>Figura 119</b> Reservaciones realizadas del robot KUKA número 3.....	182
<b>Figura 120</b> Reservaciones realizadas del robot KUKA número 5.....	182
<b>Figura 121</b> Diagrama de barras del número de aciertos obtenido del servicio SpeechToText API .....	184

## Resumen

En la actualidad, el internet se ha convertido en una pieza fundamental de la sociedad, gracias a su carácter multidisciplinario, cuyos contenidos pueden ser utilizados en varios campos como: educación, comunicación, salud, investigación, economía, industria, entre otros. Brinda un sinnúmero de herramientas y servicios que permiten, al ser humano, trascender en cómo maneja, percibe y recibe el conocimiento, dotándole de la habilidad de acceder desde cualquier parte y en cualquier momento, a información y recursos tecnológicos valiosos. Por ello, en el presente trabajo de titulación, se fusionan los conceptos de Cloud Computing y la robótica social, para adentrarse en el campo del Cloud Robotics, mediante el cual se busca desarrollar e implementar aplicativos avanzados, en las áreas de educación, cuidados de la salud y atención al cliente, que permitan demostrar la importancia del trabajo conjunto entre los recursos locales del robot humanoide NAO y los servicios en la nube. Esto, con el fin de presenciar su impacto en la mejora de la ejecución de tareas y la incorporación de nuevas funcionalidades. Entre las actividades desarrolladas, se mencionan el análisis comparativo de las principales plataformas de servicios, el planteamiento de problemas reales donde la implementación de los aplicativos represente soluciones tecnológicas viables, el diseño de arquitecturas Cloud Robotics, el análisis de costos de cada aplicativo y una comparativa con proyectos previos, para verificar la mejora del desempeño del robot, frente a soluciones offline implementadas.

### Palabras Clave:

- **ROBÓTICA SOCIAL**
- **CLOUD ROBOTICS**
- **SERVICIOS EN LA NUBE**

## **Abstract**

Nowadays, the internet has become a fundamental piece of society, thanks to its multidisciplinary nature, whose contents can be used in various fields such as: education, communication, health, research, economy, industry, among others. It provides countless tools and services that allow human beings to transcend how they handle, perceive and receive knowledge, providing them with the ability to access valuable information and technological resources from anywhere and at any time. Therefore, in the present degree work, the concepts of Cloud Computing and social robotics are merged, to enter the field of Cloud Robotics, through which it seeks to develop and implement advanced applications, in the areas of education, healthcare and customer service, which allow demonstrating the importance of joint work between the local resources of the NAO humanoid robot and cloud services. This, in order to witness its impact on the improvement of the execution of tasks and the incorporation of new functionalities. Among the activities developed, mention is made of the comparative analysis of the main service platforms, the approach of real problems where the implementation of the applications represents viable technological solutions, the design of Cloud Robotics architectures, the cost analysis of each application and a comparison with previous projects, to verify the improvement of the robot's performance, compared to implemented offline solutions.

### **Keywords:**

- **SOCIAL ROBOTICS**
- **CLOUD ROBOTICS**
- **CLOUD SERVICES**

## Capítulo I

### Planteamiento del problema de investigación

#### Antecedentes

Gracias a los avances tecnológicos, dentro del campo de la inteligencia artificial, la robótica social se introduce a un ritmo moderado en la sociedad humana, formando parte de la cotidianidad del día a día. (Pérez, Castro-González, & Castillo, 2017) Esto ha generado un incremento considerable de la interacción humano-robot, así como la necesidad de desarrollar nuevas técnicas de diseño y programación de robots que permitan una inclusión y aceptación completa por parte del hombre.

En el año 1990, MIT crea Kismet (Wikipedia, 2018), uno de los primeros robots sociales encargado de reconocer y simular las emociones. Solo seis años después, Honda crea los robots antropomórficos P1, P2, P3, P4, y en el año 2000, ASIMO (Honda, 2018), siendo este último, utilizado para investigar y profundizar sobre los robots de compañía, en los cuales se considera no solo la interacción con el humano, sino también con su entorno.

Con el paso del tiempo, la lista de robots sociales es cada vez más larga, a medida que se los considera como proyectos comerciales. Uno de los casos más exitosos es el robot Pepper, creado por SoftBank Robotics (SoftBank Robotics, 2018).

Su principal cualidad es la de percibir las emociones y adaptar su comportamiento al estado anímico del usuario. En Japón, este robot ya se está utilizando para recibir, informar y entretener a los clientes. Entre otros robots se encuentran el HRP-4C (Wikipedia, 2018), Nuka (El Comercio, 2016), Jibo (Jibo, Inc., 2019) o Sota (Siegel, 2015) que abarcan el campo de asistenciales.



En el Ecuador, ya se han desarrollado diversos trabajos en el área de la robótica social, siendo protagonista el robot NAO (SoftBank Robotics, 2006), cuya edición académica fue creada para las universidades y laboratorios con fines de investigación y educación, gracias a la gran cantidad de sensores, actuadores y funcionalidades que posee. (Wikipedia, 2019)

La Universidad Politécnica Salesiana (UPS) ha generado un considerable número de trabajos de titulación, utilizando este robot, centrados en la ejecución de tareas como: rehabilitación física de adultos mayores mediante visión artificial (Calvopiña Iglesias & Valladares Romero, 2017), comandos de voz (Achig Ortiz & Lasluisa Naranjo, 2017), suministro de medicamentos a través de lectura de código de barras (Navas Reascos & Rivera Bonifaz, 2017), clasificación de piezas, con visión artificial, para la industria mecánica (Cadena Castro & Heredia López, 2018), diseño de sistema de redes neuronales orientadas a la expresión oral (Montalvo López, 2017), el área neuro-computacional (Gómez Vega & Guerrero Rivera, 2016) y la teleoperación de movimientos (Galarza Guerrero & Llumiquinga Pumisacho, 2018).

En la Universidad de las Fuerzas Armadas – ESPE, ya existe un primer acercamiento a un enfoque más especializado de interacción hombre-robot, obtenido mediante el desarrollo de un sistema básico de robótica de entretenimiento persuasivo, que permite conocer las capacidades del mismo para relacionarse con las personas (Vilatuña Salguero, 2018).

Sin embargo, este proyecto se ve limitado al uso de los recursos disponibles preinstalados en el robot, los cuales pueden resultar ineficientes a la hora de crear aplicativos más complejos, que se adapten a las crecientes necesidades de los usuarios y a los avances de la tecnología.

Con el fin de trascender las limitaciones de hardware y software de los autómatas comerciales y aumentar las capacidades de procesamiento, se ha profundizado en la aplicación de servicios en la nube o Cloud Computing, dando paso a una nueva rama de investigación llamada Robótica en la Nube o Cloud Robotics. (Guizzo, 2011)

Este campo de la robótica es relativamente nuevo, sin embargo, ya se han realizado trabajos de investigación orientados a la mejora de funciones de robots comerciales como NAO (Bouziane, Terrisa, & Brethe, 2017), y robots de asistencia social (Bonaccorsi, Fiorini, Cavallo, Alessandro, & Dario, 2016).

### **Justificación e importancia**

Debido a la disponibilidad del robot humanoide NAO, en los Laboratorios del DEEL de la Universidad de las Fuerzas Armadas – ESPE, es posible impulsar la investigación y el trabajo continuo en el desarrollo de aplicativos, y funcionalidades que puedan ser utilizadas en el campo de la robótica social, con el fin de interactuar con las personas de manera natural e interpersonal, a menudo para lograr objetivos socioemocionales en diversas aplicaciones como: educación, salud, calidad de vida, entretenimiento, comunicación y colaboración. (Breazeal, Takanishi, & Kobayashi, Social Robots that Interact with People, 2008)

Sin embargo, la baja capacidad de procesamiento disponible, dado por el CPU Intel ATOM Z530 1.6 GHz (Aldebaran, 2018) perteneciente al robot, así como la inviabilidad de la instalación de programas y librerías en el mismo, se presentan como un limitante al momento de implementar tareas más complejas como procesamiento de lenguaje natural, visión artificial avanzada, entre otras.

Es por ello que se propone, como solución, la utilización de servicios en la nube o Cloud Computing, que permiten el acceso a hardware y software proporcionado como un servicio, de otra empresa, a través de Internet, por lo general de una manera completamente transparente. (Citelia, 2020)

Los estudios realizados en el país (Calvopiña Iglesias & Valladares Romero, 2017) (Achig Ortiz & Lasluisa Naranjo, 2017) (Navas Reascos & Rivera Bonifaz, 2017) (Cadena Castro & Heredia López, 2018) (Montalvo López, 2017) (Gómez Vega & Guerrero Rivera, 2016) (Galarza Guerrero & Llumiquinga Pumisacho, 2018) están orientados a la utilización de robots humanoides solo como herramientas para trabajos específicos.

Por lo cual, mediante la ejecución del presente trabajo de titulación, se desea sentar un precedente en el desarrollo de la robótica social, en el Ecuador, para impulsar a estudiantes, docentes e investigadores a la creación de soluciones tecnológicas que faciliten la interacción continua del robot, con el usuario y su entorno, y a su vez, permitir una mejor integración de los autómatas a la sociedad, en el futuro cercano.

La importancia de este proyecto reside en el máximo aprovechamiento de las características de hardware disponibles del robot NAO combinado a la amplia oferta de servicios existentes en la nube, como, por ejemplo, procesamiento de lenguaje natural, visión artificial, etc., con el fin de desarrollar aplicativos, dentro del área de robótica social, que puedan resultar complejos de implementar, utilizando solo los recursos locales disponibles.

## **Alcance del proyecto**

El presente proyecto de titulación propone la utilización de servicios en la nube, mediante el manejo de APIs en lenguaje Python, que permitan expandir las funcionalidades del robot NAO, en el área de la robótica social, las cuales se ven limitadas a los módulos, funciones y librerías preinstaladas en el software principal que controla al robot, llamado NAOqi (Aldebaran, 2018).

Con esto, se pretende incursionar en el campo del Cloud Robotics, mediante la conexión a internet del robot humanoide NAO, a fin de beneficiar al mismo de grandes recursos computacionales, de almacenamiento y comunicaciones. En este caso, el Cloud Robotics se ve dirigido hacia la transferencia de tareas computacionales pesadas hacia la nube, a través del uso de servicios cloud, para suplementar los recursos locales limitados.

Una vez implementados los algoritmos de programación para el acceso a los servicios en la nube, se unificarán estos, en aplicaciones basadas en los diferentes sectores de la robótica social tales como educación, cuidados de la salud y atención al cliente. Por lo cual, este proyecto tiene como alcance desarrollar rutinas y tareas específicas que permitan demostrar la utilidad de los servicios en la nube, en el desempeño del robot, dentro de los sectores previamente mencionados.

Así, para ilustrar la utilización de robots humanoides en la educación, se plantea la realización de un juego para la enseñanza del idioma inglés donde el usuario mostrará una tarjeta al robot y éste reconocerá la imagen y pedirá al usuario que le indique el nombre, de manera oral y escrita. El autómata verificará si la información entregada es correcta, mediante visión artificial y lenguaje natural, retroalimentando al usuario con frases positivas. El grupo objetivo son personas mayores a 18 años.

En el área de cuidados de salud, el robot podrá crear y consultar recordatorios en Google Calendar a petición del usuario, para la ingesta de medicamentos en horas definidas. El grupo objetivo abarca a personas mayores de 18 años.

Finalmente, en el campo de atención al cliente, el robot permitirá la reserva de robots móviles y fijos pertenecientes al Laboratorio de Robótica de la Universidad de las Fuerzas Armadas – ESPE, mediante la utilización de un chatbot para una comunicación oral más fluida y el almacenamiento de datos en una base de datos alojada en la nube. El grupo objetivo abarca a personas mayores de 18 años.

Con el fin de generar interés y desarrollar habilidades, en los estudiantes de la carrera de Ingeniería en Electrónica y Automatización, el proyecto propone la realización de dos prácticas de laboratorio, para el aprendizaje y utilización del robot humanoide.

Al terminar con los diferentes aplicativos, se realizarán pruebas y evaluaciones del comportamiento mejorado del robot, comparando su desempeño actual (aplicativos offline recopilados de trabajos de investigación previos) con el desempeño obtenido (aplicativos online presentados en este trabajo).

## **Objetivos**

### ***Objetivo General***

Desarrollar módulos que permitan la conexión del robot humanoide NAO con servicios de computación en la nube (Cloud Robotics), con el fin de implementar aplicativos en los sectores de robótica social de educación, salud y atención al cliente.

***Objetivos Específicos***

- Identificar la plataforma y servicios en la nube adecuadas para el desarrollo del presente proyecto.
- Desarrollar algoritmos que permitan utilizar los servicios en la nube escogidos, en el robot NAO, mediante lenguaje de programación libre.
- Diseñar e implementar aplicativos de robótica social, utilizando los servicios en la nube necesarios.
- Evaluar el desempeño del robot NAO, mediante la comparación de los módulos creados con soluciones offline.
- Diseñar laboratorios teóricos/prácticos para la enseñanza de la utilización del robot NAO

## Capítulo II

### Fundamentación Teórica

#### Robótica Social

Según el International Journal of Social Robotics, la robótica social se define como el estudio de robots que pueden interactuar y comunicarse entre sí, con los humanos y con el medio ambiente, dentro de la estructura social y cultural asociada a su función. (Sam Ge, 2009)

A lo largo de la historia, y gracias a los avances tecnológicos, los robots sociales han evolucionado a pasos agigantados, desde las primeras creaciones donde se buscaba reproducir, de forma artificial, las interacciones de las comunidades biológicas, hasta la creación y comercialización de robots humanoides utilizados para mejorar la calidad de vida de las personas. (Pérez, Castro-González, & Castillo, 2017)

En la actualidad, existen siete sectores, en los cuales, los robots sociales se desempeñan activamente: información y comunicación, salud y trabajo social, educación, arte y entretenimiento, cuidado de la salud, comercio al por menor, alojamiento y servicio de comida. (KPMG, 2016)

Estos sectores involucran una mayor interacción entre el usuario y el robot, lo cual impulsa a los investigadores, a desarrollar nuevas tecnologías y sistemas más complejos que permitan una mejora continua del autómata en su aspecto físico, expresión corporal, procesos cognitivos, intercomunicación, etc., con el fin de alcanzar su completa integración en la vida diaria del ser humano.

## Interacción Humano-Robot

Según Goodrich y Schultz, la interacción humano – robot (HRI) es un campo de estudio dedicado a comprender, diseñar y evaluar sistemas robóticos para su uso por o con humanos (Goodrich & Schultz, 2007). Posee contribuciones de la interacción humano-computadora, inteligencia artificial, robótica, comprensión del lenguaje natural, diseño y ciencias sociales.

Es relativamente nuevo y va ganando territorio a medida que los robots trascienden más allá de simples herramientas autónomas que, muchas veces, son dirigidas de manera remota por un supervisor humano, para convertirse en asistentes personales (IDMind, 2016), robots de compañía (Jibo, Inc., 2019), tutores (Dautenhahn, 1999), etc. Así, en la Figura 1 se evidencia la utilización de un robot en el área de enfermería.

### Figura 1

*Robot Little Casper en Instituto Portugués de Oncología en Lisbon*



*Nota.* Adaptado de Meet Little Casper, a robot designed to help children suffering from cancer, de (Euronews, 2016)

En el campo de la interacción humano-computador (HCI), los autores Reeves y Nass demostraron que las personas pueden llegar a tratar a las máquinas como a



cualquier otra persona siempre que ésta manifieste comportamientos sociales competentes (Reeves & Nass, 1996).

Si se analiza esta premisa, aplicada a la robótica social y su situación actual, se puede observar una fuerte influencia de ésta, en el desarrollo de nuevas técnicas de comunicación, donde se busca obtener un desenvolvimiento más natural por parte del robot, para una aceptación e integración más rápida y sencilla por parte de la sociedad.

Es importante recalcar que las técnicas dentro de HCI han sentado precedentes para el desarrollo del HRI en sí, sin embargo, por sí solas no son suficientes para satisfacer las exigencias de dicha interacción, ya que el comportamiento del robot no debe ser evaluado solamente desde la perspectiva del ser humano, sino desde la del robot per se, frente a todo lo que le rodea (Breazeal, 2004).

Por ello, al diseñar sistemas HRI, es necesario tomar en consideración cinco aspectos que, combinados entre sí, pueden dar paso a una interacción satisfactoria, (Goodrich & Schultz, 2007): nivel de autonomía y comportamiento, intercambio natural de información, mecanismo de interacción, adaptación, aprendizaje y entrenamiento de las personas y los robots, capacidad de cumplir tareas.

### ***Clasificación de la Interacción Humano - Robot***

Debido a la gran cantidad de formas de comunicación e interacción que puede utilizar un robot frente a aplicaciones y/o situaciones que involucran a personas, su entorno e incluso otros robots, es posible clasificar la HRI en múltiples formas.

Así, por ejemplo, Cynthia Breazeal lo divide en cuatro paradigmas de interacción, basados en el modelo mental que un humano tiene del robot cuando interactúa con él (Breazeal, 2004):

1. **Robot como herramienta:** El ser humano percibe al robot como una herramienta para cumplir una tarea, por ejemplo, una aspiradora automática.
2. **Robot como una extensión cibernética:** El robot se fusiona físicamente con la persona y esta lo acepta como parte integral de su cuerpo, por ejemplo, la utilización de prótesis robóticas o exoesqueletos.
3. **Robot como avatar:** La persona se proyecta a sí misma, a través del robot con el fin de comunicarse con otro ser lejos de él, proporcionando una sensación de presencia física durante la interacción.
4. **Robot como un compañero sociable:** La persona logra interactuar con el robot, como si interactuara con cualquier otra criatura socialmente receptiva que coopera como un compañero.

Otro ejemplo de clasificación de HRI se basa en el rol que desempeña el robot en la interacción, así, Takayuki Kanda propone lo siguiente (Kanda & Ishiguro, 2013):

1. **Interacción Pasiva:** El robot se presenta al humano, pero no recibe órdenes de él / ella.
2. **Interacción Pasivo-Social:** Los robots se comunican entre sí y se presentan al humano. El compañero humano es pasivo, significa que él / ella no da órdenes a los robots.
3. **Interacción Interactiva:** Un robot está interactuando con un compañero humano. La interacción es bidireccional.
4. **Interacción Interactivo-Social:** Dos o más robots se comunican entre sí. Además de su comunicación, están interactuando con un compañero humano.

Es importante identificar en qué paradigma o tipo de interacción recae el robot con el que se va a trabajar, ya que es evidente que las necesidades de comunicación

difieren, por ejemplo; entre un robot-herramienta, el cual tiene una interacción limitada con el individuo y un robot-compañero que recae en una interacción de tipo interactiva, utilizando sistemas de reconocimiento facial, de voz, lenguaje natural, etc., para lograr establecer un vínculo con la persona a cargo.

### Robot Humanoide NAO

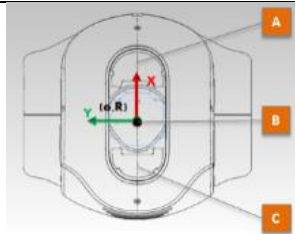
NAO es un robot humanoide autónomo programable, creado por la compañía Softbank Robotics (antes Aldebaran Robotics subsidiaria de la compañía Softbank). Es utilizado como una poderosa herramienta en el campo de la educación e investigación y como asistente para compañías o centros de salud donde recibe, informa y entretiene a visitantes. (SoftBank Robotics, 2020)


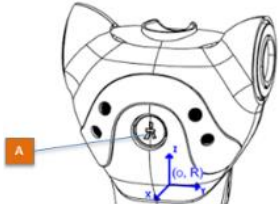
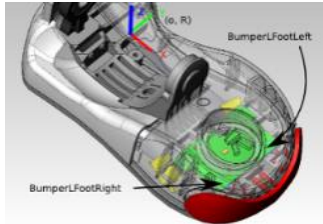
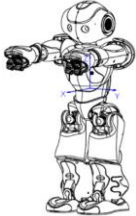
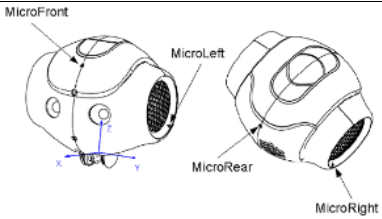
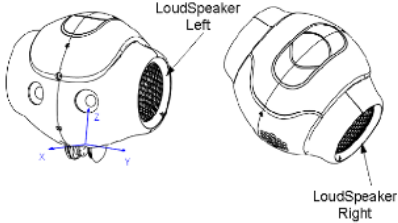
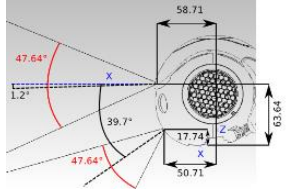
#### **Características Técnicas Robot Humanoide Nao**

El robot NAO tiene una altura de 573mm, ancho de 275mm y profundidad de 311mm. Posee un procesador ATOM Z530 de 1.6GHz, con 1GB de RAM, 2GB de memoria Flash y 8GB Micro SDHC. Además, cuenta con sensores y actuadores que le permiten una interacción completa con su entorno, los cuales se mencionan en la Tabla 1 :

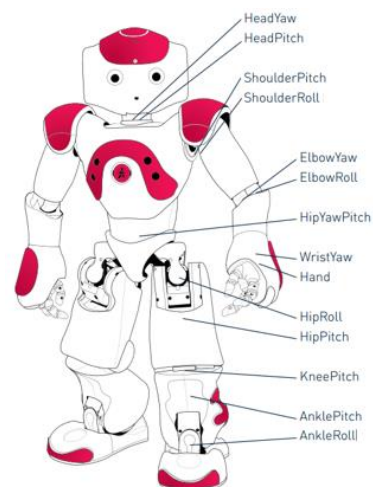
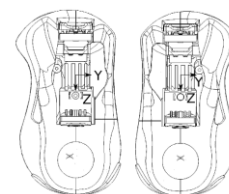
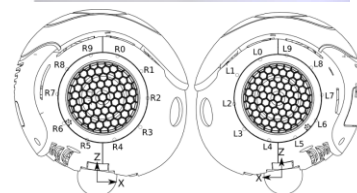
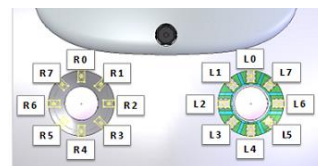
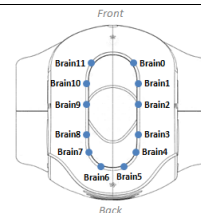
**Tabla 1**

*Sensores y actuadores robot NAO*

Sensor/Actuador	Ubicación	Imagen Referencial
Sensores Touch <sup>a</sup>	Cabeza Frontal Medio Posterior	

Sensor/Actuador	Ubicación		Imagen Referencial
Sensores Touch <sup>a</sup>	Mano Izquierda	Izquierda Derecha Atrás	
	Mano Derecha	Izquierda Derecha Atrás	
Interruptores <sup>a</sup>	Pecho		
	Pie Izquierdo	Izquierda Derecha	
Pie Derecho	Izquierda Derecha		
Giroscopio <sup>b</sup>	Torso		
Acelerómetro <sup>b</sup>			
Micrófonos <sup>c</sup>	Frontales	Izquierdo Derecho	
	Posteriores	Izquierdo Derecho	
Parlantes <sup>d</sup>	Cabeza	Izquierdo Derecho	
Cámaras <sup>e</sup>	Frente	Superior Inferior	

Sensor/Actuador	Ubicación	Imagen Referencial
LEDS <sup>f</sup>	Cabeza	Izquierda Derecha
	Ojos	Izquierda Derecha
	Oídos	Izquierda Derecha
	Pies	Izquierda Derecha
Motores <sup>g</sup>	Cabeza	Pitch Yaw
	Hombros	Pitch Roll
	Codo	Yaw Roll
	Cadera	Roll Pitch Yaw
	Muñeca/Mano	Yaw
	Rodilla	Pitch
	Tobillo	Pitch Yaw



Nota. <sup>a</sup> (Contact and Tactile Sensors, 2014). <sup>b</sup> (Inertial Unit, 2014). <sup>c</sup> (Microphones, 2014). <sup>d</sup> (Loudspeakers, 2014). <sup>e</sup> (Video Camera, 2014). <sup>f</sup> (LEDs, 2014). <sup>g</sup> (Motors, 2014).

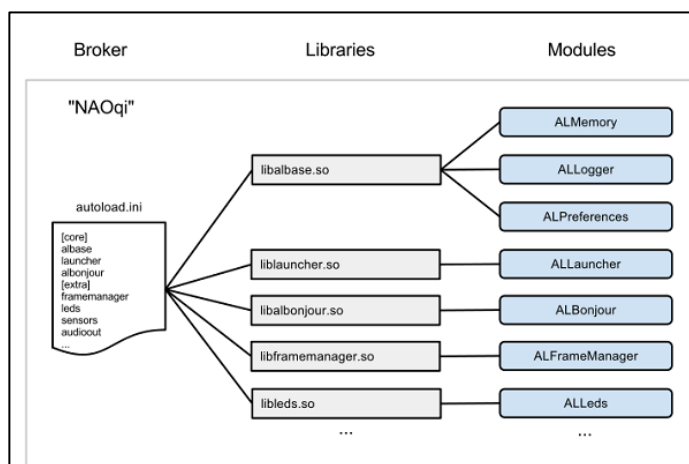
## NAOqi

NAOqi es el software principal que funciona en el robot y permite su control. Por otro lado, el framework NAOqi es el entorno de trabajo para programar al robot, el cual permite la ejecución de APIs tanto en lenguaje C++ como en Python.

El ejecutable NAOqi es un broker, un objeto que permite encontrar módulos y métodos (servicios de directorio) y el llamado de los mismos desde fuera del proceso (acceso a la red). Cuando se inicia, carga un archivo de preferencias llamado `autoload.ini` que define qué bibliotecas debe cargar, como se observa en la Figura 2. Cada biblioteca contiene uno o más módulos que usan el broker para anunciar sus métodos. (Aldebaran Robotics, 2011)

**Figura 2**

*Framework NAOqi*



*Nota.* Obtenido de *The NAOqi process*, de (Aldebaran Robotics, 2011)

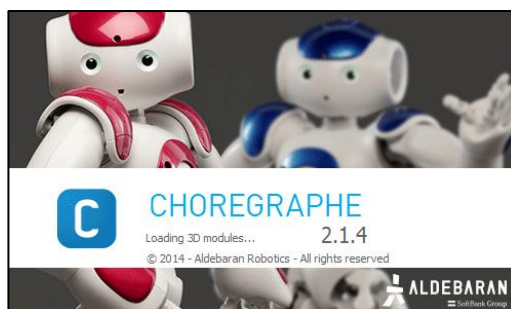
## Choregraphe

Choregraphe es una aplicación multiplataforma para PC que permite crear animaciones, comportamientos, diálogos y programarlos en el robot humanoide NAO.

Además, facilita la simulación de un robot para pruebas, así como el monitoreo y control del real (Softbank Robotics, 2020). En la Figura 3 se muestra la imagen inicio, desplegada al momento de acceder al programa.

### Figura 3

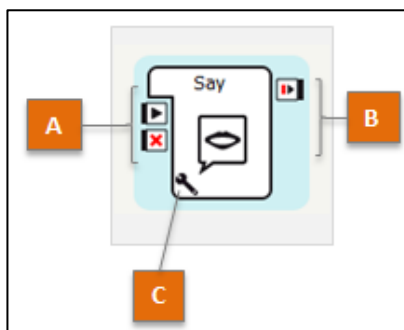
*Software Choregraphe*



Este software utiliza un lenguaje de programación estructurado por bloques, donde el usuario puede crear desde simples hasta complejas aplicaciones, sin necesidad de digitar las líneas de código. Los bloques prediseñados, existentes en la librería de bloques, están conformados por entradas, salidas, parámetros configurables y algoritmos que desempeñan funciones específicas como se observa en la Figura 4.

### Figura 4

*Bloque de programación: A. Entradas - B. Salidas - C. Parámetros*



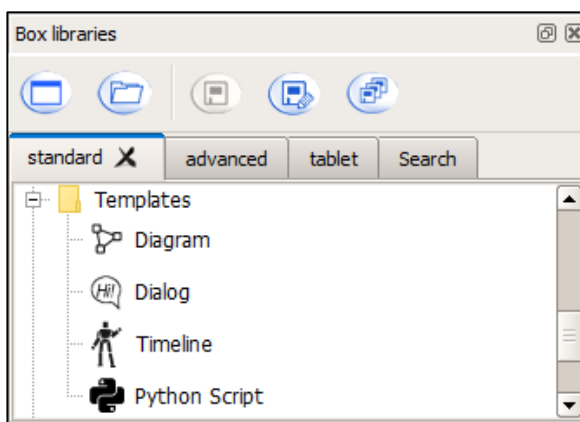
Existen cuatro tipos de bloques:

- **Bloque Python:** contiene un script de Python que puede ser prediseñado o diseñado por el usuario desde cero, utilizando las librerías de funciones.
- **Bloque de diagrama de flujo (Flow diagram):** contiene un grupo de bloques conectados entre sí mediante sus entradas y salidas.
- **Bloque de Línea de Tiempo (Timeline):** contiene una capa de movimiento donde se almacena los movimientos del NAO en un lapso de tiempo. Cada fotograma clave de movimiento en la capa de movimiento corresponde a una posición del robot o una parte de su cuerpo.
- **Bloque de Diálogo:** contiene scripts configurables que permiten el diálogo entre el robot y un agente externo.

En la Figura 5 se muestran las distintas plantillas de los tipos de bloques previamente descritos. Para su utilización, es necesario arrastrar el elemento hacia el área de trabajo.

### Figura 5

*Plantillas de los tipos de bloques*





## **Servicios Web**

Según el consorcio World Wide Web (W3C), un servicio web es "una aplicación de software identificada por un Identificador de Recursos Uniforme (URI), cuyas interfaces y enlaces pueden definirse, describirse y descubrirse como artefactos XML. Un servicio web admite interacciones directas con otros agentes de software que utilizan mensajes basados en XML intercambiados a través de protocolos basados en Internet" (Alonso, Casati, Kuno, & Machiraju, 2004).

Es importante recalcar que, el acceso a los servicios web puede realizarse desde cualquier lugar de Internet y es completamente independiente tanto a la plataforma escogida para ello como al lenguaje de programación en el que se haya implementado internamente el servicio (Universidad de Alicante, 2014).

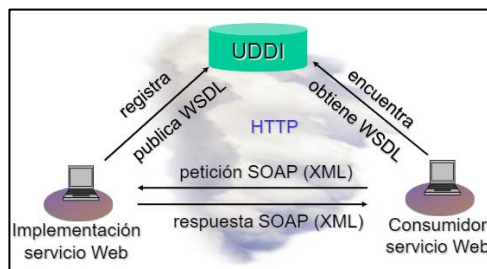
Existen dos tipos de servicios web, cuyas diferencias recaen en la complejidad del acceso/entrega de información por parte del cliente/servidor. Así se tiene (Programación Colombia, 2018):

- **Servicios Web SOAP**

Son aquellos que utilizan mensajes XML para comunicarse y el protocolo de comunicación SOAP para definir la arquitectura y formato de los mensajes. Además, utiliza documentos WSDL (Web Services Description Language) para exponer los métodos y detalles técnicos. SOAP soporta únicamente información en formato XML. Su arquitectura se muestra a detalle en la Figura 6.

**Figura 6**

*Arquitectura Servicio Web SOAP*



*Nota. Obtenido de Servicios Web. Antecedentes y Justificación SOAP, WSDL, UDDI*

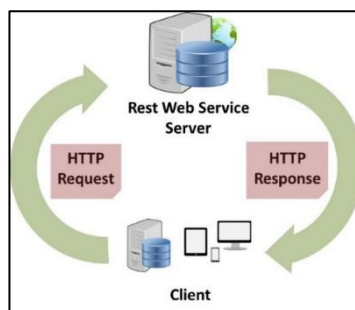
*Utilización de Servicios Web Creación de Servicios Web, de (Labra Gayo, 2016)*

#### - **Servicios Web REST**

Son aquellos que usan protocolos como HTTP o similares para la comunicación, restringiendo a la interfaz a usar operaciones estándar como GET, POST, PUT, DELETE. Utiliza URIs para exponer los recursos a utilizar. REST permite el intercambio de datos en formato JSON, XML, texto plano, entre otros. Su arquitectura se muestra a detalle en la Figura 7

**Figura 7**

*Arquitectura Servicio Web REST*



*Nota. Obtenido de Mastering REST Architecture — REST Architecture Details, de (Özlu, 2018)*

## RESTFUL API

Para entender qué es una RESTful API, es importante, primero, definir estos conceptos individualmente y conocer sus respectivas funciones y características.

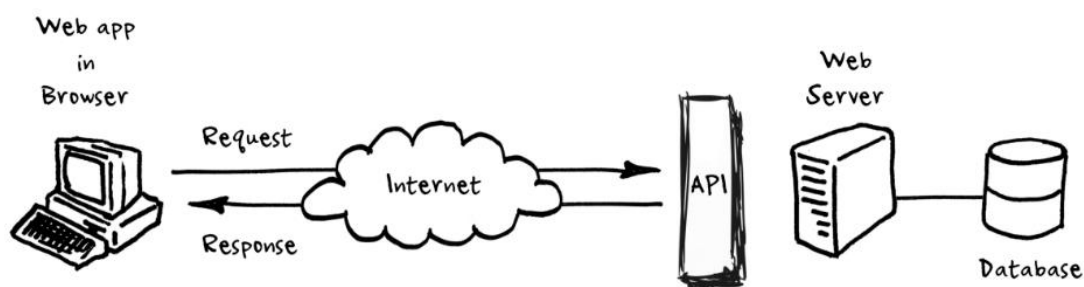
### ***Interfaz de Programación de Aplicaciones (API)***

La interfaz de programación de aplicaciones (API) es el conjunto de definiciones y protocolos que se utilizan para el desarrollo e integración de aplicaciones (Red Hat, 2019), permitiendo la comunicación entre productos de software.

En pocas palabras, es el mensajero (o software intermediario) que se encarga de obtener la solicitud por parte del cliente, entregarla al servidor para su procesamiento y devolver el resultado al solicitante (MuleSoft, 2015), como se observa en la Figura 8.

### **Figura 8**

#### *Funcionamiento de una API*



*Nota.* Obtenido de *What exactly is an API?*, de (Eising, 2017)

### ***Transferencia De Estado Representacional (REST)***

Transferencia de estado representacional (REST) es un estilo de arquitectura para el diseño de aplicaciones en red. Provee estándares entre

sistemas informáticos interconectados, facilitando la comunicación entre estos, mediante la utilización del protocolo HTTP. (Codecademy, 2020)

A su vez, es un conjunto de restricciones que aseguran un sistema escalable, tolerante a fallas y fácilmente extensible. Cabe mencionar que en REST es posible usar no solo HTTP, sino otros protocolos de transferencia como SNMP, SMTP e incluso SOAP. (Thijssen, 2016)

Está basada en la arquitectura cliente – servidor, donde cada uno desempeña su papel de manera independiente: los clientes envían solicitudes para recuperar o modificar recursos, y los servidores envían respuestas a estas solicitudes.

Así, el servidor almacena y/o manipula la información, poniéndola a disposición del usuario de manera eficiente mientras que el cliente toma esa información y se la muestra al usuario, para posteriormente ser utilizada. (Hoguin, 2018) Las siguientes seis restricciones identifican una arquitectura REST (Rouse, Bedell, Hannan, & Wilson, 2019):

- **Uso de una interfaz uniforme (UI):** Los recursos deben ser identificables de manera única a través de una única URL, y solo se puede manipular un recurso mediante el uso de los métodos subyacentes del protocolo de red, como DELETE, PUT y GET con HTTP.
- **Basado en cliente-servidor:** Debe haber una delimitación clara entre el cliente y el servidor.
- **Operaciones apátridas:** Todas las operaciones cliente-servidor deben ser sin estado, y cualquier gestión de estado que se requiera debe llevarse a cabo en el cliente, no en el servidor.

- **Caché de recursos RESTful:** Todos los recursos deben permitir el almacenamiento en caché a menos que se indique explícitamente que el almacenamiento en caché no es posible.
- **Sistema de capas:** REST permite una arquitectura compuesta de múltiples capas de servidores.
- **Código bajo demanda:** La mayoría de las veces, un servidor enviará representaciones estáticas de recursos en forma de XML o JSON. Sin embargo, cuando sea necesario, los servidores pueden enviar código ejecutable al cliente.

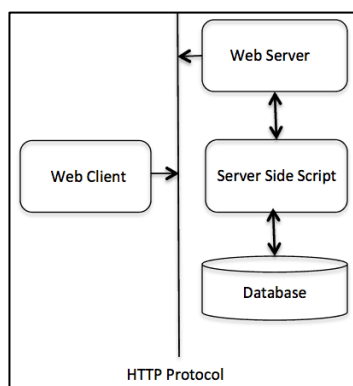
### ***Protocolo HTTP***

El Protocolo de Transferencia de Hipertexto (HTTP) es un protocolo de comunicación cliente-servidor basado en TCP / IP, utilizado para enviar y recibir información a través de la red (Ahlawat, 2020).

Es decir, especifica cómo se construirán y enviarán los datos de solicitud de los clientes al servidor, y a su vez, cómo los servidores responderán a estas solicitudes. Su arquitectura se muestra en la Figura 9.

## Figura 9

### Arquitectura básica del protocolo HTTP



Nota. Obtenido de *HTTP - Overview*, de (Tutorials Point, 2020)

Las características principales de este protocolo son (Tutorials Point, 2020):

- **No tiene conexión:** La interacción cliente/servidor solo existe durante la solicitud y la respuesta actuales. Luego ambos se olvidan.
- **Es independiente de los medios:** HTTP puede enviar cualquier tipo de datos, siempre que el cliente y servidor sepan cómo manejar el contenido de los mismos.
- **No tiene estado:** Ni el cliente ni el servidor pueden retener información entre diferentes solicitudes. Cada petición es independiente de las demás.

**Mensajes HTTP.** Un mensaje HTTP se compone, generalmente, de tres partes (Escobar, 2017):

- **La primera línea:** En la solicitud o petición, la primera línea está conformada por el verbo o método, el recurso a trabajar y la versión de HTTP. En cambio, en la respuesta se tienen la versión de HTTP y un código de 3 dígitos, que indica al cliente cómo interpretar el resultado de la petición.

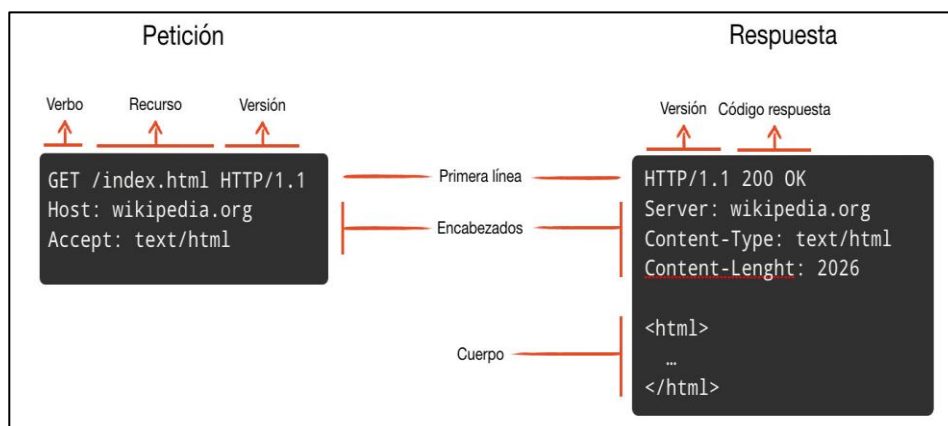
Estos códigos se dividen en 5 categorías, dependiendo del dígito con el que inician: 1XX – Información, 2XX – Éxito, 3XX – Redirección, 4XX - Error en el cliente, 5XX - Error en el servidor

- **Encabezados:** Brindan información adicional sobre la petición o la respuesta, entre los más comunes se encuentran:
  - o **Content-Type:** refiere al tipo de contenido que se está enviando en el cuerpo de un mensaje de petición, por ejemplo, text/html.
  - o **Accept:** refiere al tipo de contenido que el cliente está esperando.
  - o **User-Agent:** refiere al tipo de navegador que está haciendo la petición.
- **Cuerpo (opcional):** Lleva los datos, archivos o documentos asociados con la petición o la respuesta.

En la Figura 10 se muestra un ejemplo de las partes de los mensajes HTTP.

**Figura 10**

*Partes de los mensajes HTTP*



*Nota.* Obtenido de *El protocolo HTTP*, de (Escobar, 2017)

**Métodos HTTP.** Los métodos (también llamados verbos) definen la acción que se desea realizar sobre el recurso descrito en la primera línea de la petición (Escobar, 2017). Los más comunes se enlistan a continuación (Ahlawat, 2020):

- **GET:** se utiliza para solicitar un recurso del servidor. La solicitud no posee un cuerpo de mensaje.
- **POST:** se utiliza para publicar un recurso o enviar los datos al servidor, que se encuentran en el cuerpo del mensaje de petición.
- **PUT:** se utiliza para reemplazar, actualizar o crear información que se encuentre o no presente en el servidor, con los datos que se envían en el cuerpo del mensaje.
- **DELETE:** se utiliza para eliminar un recurso en el servidor.

## JSON

JavaScript Object Notation (JSON) es un formato liviano para almacenar y transportar datos. Cabe mencionar que el código para leer y generar datos JSON se puede escribir en cualquier lenguaje de programación (w3schools.com, 2019). En la Figura 11 se detalla un ejemplo básico de la sintaxis de un objeto JSON.

### Figura 11

*Sintaxis objeto JSON*

```
//JSON Object
{
  "employee": {
    "id": 1,
    "name": "Admin",
    "location": "USA"
  }
}
```

*Nota.* Obtenido de *Introduction to JSON*, de (Gupta, 2019)



Es el principal formato utilizado por las APIs, para enviar y recibir información. En su estructura, los datos se organizan en pares clave/valor separados por comas, donde el objeto está envuelto por llaves y las matrices por corchetes (Bernardo, 2019). En la Figura 12 se muestra un array JSON, que contiene 3 objetos con los mismos campos y diferente información entre sí.

## Figura 12

*Sintaxis de un array JSON*

```
//JSON Array
{
  "employees": [
    {
      "id": 1,
      "name": "Admin",
      "location": "USA"
    },
    {
      "id": 2,
      "name": "User",
      "location": "USA"
    },
    {
      "id": 3,
      "name": "User2",
      "location": "USA"
    }
  ]
}
```

*Nota.* Obtenido de *Introduction to JSON*, de (Gupta, 2019)

Una vez establecidos los conceptos previamente mencionados, se puede definir a una API REST como una interfaz de programación de aplicaciones que utiliza peticiones HTTP para el acceso y manipulación de información entre cliente y servidor.

En otras palabras, Las API REST manejan las solicitudes realizadas a un recurso y devuelven toda la información relevante sobre el mismo, traducida a un formato que los clientes pueden interpretar fácilmente. Así, los clientes pueden agregar, modificar o eliminar elementos en el servidor, a través de una API REST. (Juviler, 2020)

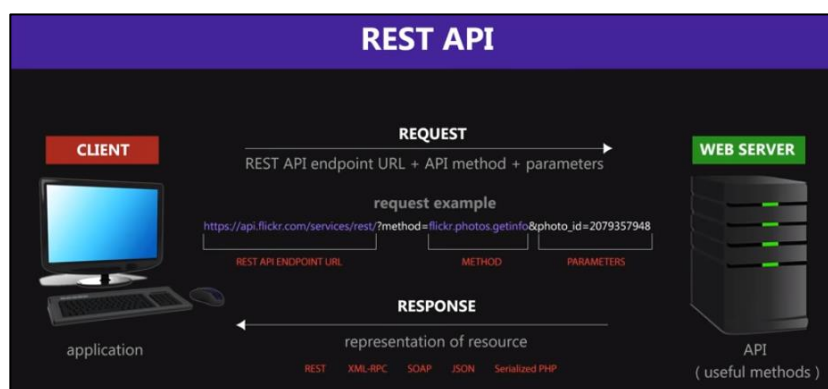
La estructura de petición a una API REST es similar a la mencionada anteriormente en HTTP, donde se tiene un método o verbo, encabezados y cuerpo. Sin

embargo, se adiciona un elemento llamado *API REST endpoint URL*, que describe la ubicación donde las API envían solicitudes y dónde vive el recurso requerido. Para las API, un endpoint puede incluir una URL de un servidor o servicio (SmartBear, 2020). En la Figura 13 se muestra un ejemplo de la sintaxis de solicitud de una API REST y el formato de respuesta de la misma.

Cabe mencionar que, la estructura de la respuesta entregada por la API REST es similar a su versión en HTTP, tomando en cuenta que, en este caso, el cuerpo del mensaje (datos) enviados al cliente estará en formato JSON.

### Figura 13

*Petición y respuesta a una API REST*



*Nota.* Obtenido de *REST API & RESTful Web Services Explained | Web Services Tutorial*, de (Clever Techie, 2017)

Si analizamos el concepto de Servicio Web REST y API REST encontraremos similitudes, entre ellas, el uso del protocolo y métodos HTTP para peticiones/respuestas sobre la red, así como el envío/recepción de datos en formato JSON, por lo tanto, se puede afirmar que una API REST es un tipo de servicio web.

Sin embargo, es importante recalcar que: *“Todos los servicios web son APIs, pero no todas las APIs son servicios web”*. Esto, debido a que no todas las API son accesibles a través de Internet, mientras que los servicios web siempre deben ser accedidos a través de una red. (Samer, 2017)

### **Servicios en la nube**

Los servicios en la nube o cloud services son infraestructuras, plataformas o sistemas de software que alojan los proveedores externos y que se ponen a disposición de los usuarios a través de Internet (RedHat, 2019). Las empresas encargadas de establecer nubes públicas, manejar nubes privadas y ofrecer distintos componentes de cloud computing son llamados proveedores de servicios en la nube.

Existen tres principales tipos de servicios en la nube, los cuales se enlistan a continuación (Leading Edge, 2015):

- **IaaS (infraestructura como servicio)**

Ofrece la infraestructura fundamental de servidores virtuales, redes, sistemas operativos y unidades de almacenamiento de datos. Es un servicio de pago por uso totalmente subcontratado y está disponible como infraestructura pública, privada o híbrida.

- **PaaS (plataforma como servicio)**

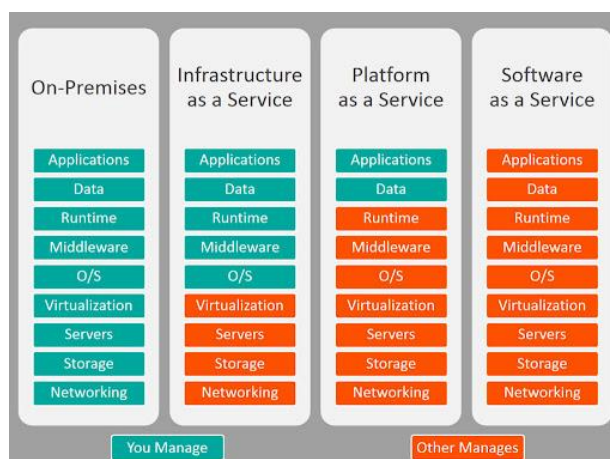
Ofrece las mismas prestaciones que IaaS, añadiendo middleware, herramientas de desarrollo, servicios de inteligencia empresarial (BI), sistemas de administración de bases de datos, etc. PaaS está diseñado para sustentar el ciclo de vida completo de las aplicaciones web: compilación, pruebas, implementación, administración y actualización (Microsoft Azure, 2015).

## - SaaS (software como servicio)

Permite a los usuarios conectarse a aplicaciones basadas en la nube a través de Internet y usarlas. Ofrece una solución de software integral que se adquiere de un proveedor de servicios en la nube mediante un modelo de pago por uso (Microsoft Azure, 2015).

### Figura 14

*Tipos de servicios en la nube*



*Nota.* Obtenido de *Diferencias entre IaaS, PaaS y SaaS*, de (RedHat, 2020)

## Cloud Robotics

El término Cloud Robotics fue introducido por el investigador de Google James Kuffner durante la Conferencia Internacional de Robots Humanoides de la IEEE realizada en Nashville en diciembre de 2010, para referirse a la posibilidad de un nuevo campo de la robótica que permita el trabajo conjunto entre autómatas y servicios en la nube. (Guizzo, 2011).

Esto, con el fin de eliminar las limitaciones dadas por la capacidad de procesamiento local, memoria o programación. A su vez, plantea un impacto

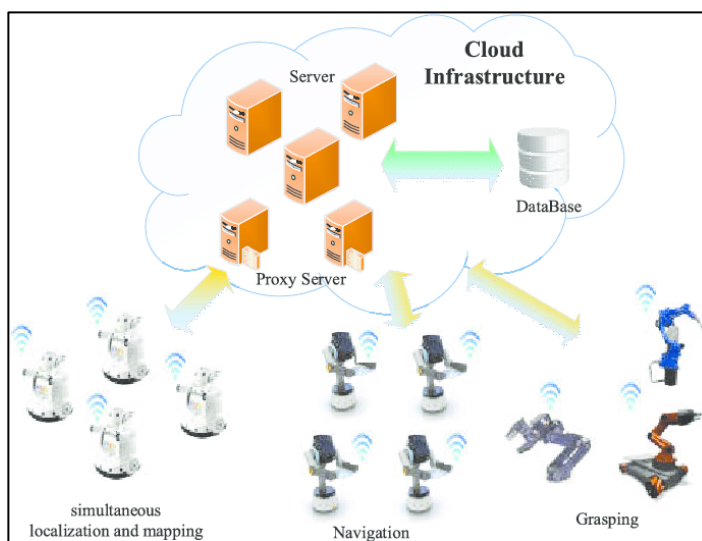
significativo de la integración de la nube en el diseño de robots, haciéndolos “más ligeros, más baratos y más inteligentes” (Koken & Gyula, 2015)

Para ello, Cloud Robotics se basa en la utilización de computación y almacenamiento en la nube, Big Data y otras tecnologías del internet, centradas en los beneficios de la infraestructura convergente y los servicios compartidos. Esto, permite a los robots beneficiarse de los poderosos recursos computacionales, de almacenamiento y de comunicaciones de los centros de datos modernos y a su vez, elimina gastos generales de mantenimiento y actualizaciones (RoboEarth, 2016).

En la Figura 15 se muestra una arquitectura básica de Cloud Robotics, donde distintos tipos de robots se conectan a la nube, de donde reciben comandos o información relevante para su funcionamiento como: localización y mapeo simultáneo, navegación, manipulación de objetos, etc.

### Figura 15

Arquitectura básica Cloud Robotics



*Nota.* Obtenido de *Cloud Robotics: Current Status and Open Issues*, de (Wan, 2016)

## Capítulo III

### Diseño del sistema Cloud Robotics para desarrollo de aplicativos de robótica social

#### Selección de plataforma de servicios en la nube

Para la implementación del presente trabajo de investigación, es necesario elegir una plataforma de servicios en la nube adecuada, que cumpla con las necesidades del proyecto y el cliente. Una vez finalizado el proceso de selección, se revisan los servicios disponibles y su documentación, con el fin de plantear el diseño de los aplicativos de robótica social.

En el mercado de los servicios en la nube, las plataformas más populares son Amazon Web Services (AWS), Microsoft Azure y Google Cloud Platform (GCP). Por ello, se consideran estas tres alternativas para el contraste de sus características mediante el proceso de análisis jerárquico (AHP) y la toma de decisiones multicriterio. A continuación, se describen las plataformas previamente mencionadas:

- **Amazon Web Services (AWS)**

Lanzada en 2006, es una subsidiaria de amazon.com, que proporciona una plataforma de Cloud Computing a particulares, empresas y gobiernos (Seligman, 2017).

- **Microsoft Azure**

Es un conjunto completo y en expansión constante de servicios de informática en la nube lanzado en 2010 con la intención de proporcionar una plataforma competente para empresas (Microsoft Azure, 2020).

- **Google Cloud Platform (GCP)**

Es un conjunto de servicios de Cloud Computing, que se ejecuta en la misma infraestructura que Google usa internamente para sus productos de usuario final, como el motor de búsqueda de Google, YouTube y más (Sabharwal & Gupta Edward, 2019).

### **Proceso de análisis jerárquico (AHP) y la toma de decisiones multicriterio**

Para realizar la comparación entre plataformas, es necesario establecer criterios o factores comunes de los que se puedan obtener información, así se tiene:

- **Criterio 1 - Año de lanzamiento:** Permite conocer cuánto tiempo se encuentra en funcionamiento la plataforma y relacionarlo con la experiencia obtenida.
- **Criterio 2 - Disponibilidad en regiones:** Número de regiones, alrededor del mundo, que están habilitadas para el acceso a la plataforma.
- **Criterio 3 - Acciones de mercado:** Cuota de mercado que abarca la plataforma.
- **Criterio 4 - Tasa de crecimiento:** Tasa de crecimiento en el mercado desde el año 2018 al año 2019.
- **Criterio 5 - Servicios:** Cantidad de servicios disponibles en cada plataforma.
- **Criterio 6 - Modelo de precios:** Permite identificar si el costo por el uso de servicios es alto o bajo.
- **Criterio 7 - Trabajos con robot NAO:** Ejemplos de trabajos de investigación realizados utilizando cada una de las plataformas junto al robot humanoide NAO.

Cabe mencionar que los servicios ofrecidos son muy similares entre sí, así como el tiempo de prueba gratuito del servicio por un año, por lo cual no se toman en cuenta dentro de los factores. Una vez recopilada la información necesaria, se realiza la siguiente tabla resumen comparativa:

**Tabla 2**

*Criterios y valores de las plataformas de servicios en la nube*

Selección de Plataforma de Servicios en la Nube							
Plataforma de Servicios	Lanzamiento	Disponibilidad	Acciones mercado	Tasa de crecimiento	Servicios	Modelos de precios	Trabajos con robot NAO
<b>Amazon Web Services</b>	Año 2006	66 regiones	30%	41%	200+	Pago por minuto	NAO Chatbot con AWS LEX (Ibáñez Romero, 2017)
<b>Microsoft Azure</b>	Año 2010	54 regiones	16%	75%	100+	Pago por minuto	Robotic Calculations and Inference Agent (Kumar, Chahal, & Hosurmath, 2017) A.I.B. August: Nao the robot on Google Cloud & CenterNet (Petreanu & Spinu, 2019)
<b>Google Cloud Platform</b>	Año 2011	20 regiones	10%	82%	60+	Pago por segundo	The NAO robot as a personal assistant (Zhou, 2016)

*Nota.* Obtenido de AWS vs Azure vs Google - Detailed Cloud Comparison, de (IntelliPaat, 2019)

Establecidos los valores para cada criterio previamente seleccionado, se realiza una comparación entre pares, colocando un valor cuantitativo que refleje el nivel de preferencia de las opciones en la primera columna versus las opciones en la primera fila. Para ello, se utiliza la escala de Saaty de importancia relativa donde estos valores



varían del 1 al 9, como se observa en la Tabla 3. Así, por ejemplo, en el Anexo I., Tabla 33 se observa que, dentro del criterio *lanzamiento*, la opción AWS es nueve veces mejor que la opción GPC, esto debido a sus años de inicio en 2006 y 2011 respectivamente. Se infiere que, a mayor antigüedad, mejor es la plataforma. Cabe mencionar que las ponderaciones son subjetivas y están estrechamente relacionadas a la opinión del usuario.

**Tabla 3**

*Escala de importancia relativa de Saaty*

<b>Escala</b>	<b>Valoración Numérica</b>	<b>Definición</b>
Extremadamente Preferido	<b>9</b>	La evidencia a favor de uno sobre el otro es de la mayor validez posible.
Muy fuerte a Extremadamente	<b>8</b>	Valores Intermedios
Muy fuerte preferido	<b>7</b>	La experiencia y el juicio favorecen fuertemente uno sobre el otro. Su importancia se demuestra en la práctica.
Fuerte a Muy Fuerte	<b>6</b>	Valores Intermedios
Muy Preferido	<b>5</b>	La experiencia y el juicio favorecen fuertemente uno sobre el otro.
Moderadamente a Fuertemente	<b>4</b>	Valores Intermedios
Moderadamente preferido	<b>3</b>	La experiencia y el juicio favorecen ligeramente uno sobre el otro.
Igual a Moderadamente	<b>2</b>	Valores Intermedios
Igualmente Preferido	<b>1</b>	Dos factores contribuyen igualmente al objetivo

Nota. Obtenido de Theory and Applications of the Analytic Network Process: Decision Making with Benefits, Opportunities, Costs, and Risks, de (Saaty, 1990)

Posteriormente se calcula el valor de la comparación inversa, es decir GCP vs. AWS, dividiendo la unidad para el puntaje previamente mencionado. Se repite este proceso para cada par hasta finalizar la tabla. Luego, se calcula una matriz normativa al dividir el valor de cada celda para la suma acumulativa de su respectiva columna. Finalmente se calcula un valor promedio de los valores de cada fila de la matriz normalizada. Se muestran los resultados obtenidos para cada criterio en el Anexo I.

Además, se realiza la ponderación de criterios, con el fin de establecer el porcentaje de influencia de cada uno de estos en la elección de la alternativa. En Anexo I., Tabla 40 se observa que los puntajes más altos se han asignado a los criterios de: número de servicios, modelos de precios y número de trabajos con el robot NAO.

Al finalizar el análisis, las columnas *Vector Promedio* se agrupan en una nueva matriz como se muestra en la Tabla 4. Posteriormente se realiza el producto de la fila de cada plataforma con la fila de ponderación y se suman los valores resultantes para obtener un puntaje individual total para AWS, Microsoft Azure y GCP que determinará la elección final.

Así, en la Tabla 4 se observa que la plataforma **Google Cloud Platform** consigue el mayor puntaje entre las tres opciones planteadas. Esto, debido a que, dentro de los criterios más importantes, GCP posee una buena cantidad de servicios que pueden ser utilizables en el presente trabajo y los precios son asequibles, incluso una vez finalizado el periodo de prueba de un año o \$300 de saldo.

Adicional, los distintos trabajos realizados con el robot humanoide NAO (y con diferentes tipos de robots) son un fuerte antecedente que aporta confiabilidad en el uso de esta alternativa.

**Tabla 4***Compilación de vectores promedios*

Selección de Plataforma de Servicios en la Nube								
Plataforma de Servicios	Lanzamiento	Disponibilidad	Acciones de mercado	Tasa de crecimiento	Servicios	Modelos de precios	Trabajos con robot NAO	Total
Amazon Web Services	0,75	0,65	0,72	0,07	0,75	0,07	0,06	<b>0,25</b>
Microsoft Azure	0,18	0,29	0,19	0,28	0,18	0,18	0,27	<b>0,22</b>
Google Cloud Platform	<b>0,07</b>	<b>0,06</b>	<b>0,08</b>	<b>0,64</b>	<b>0,07</b>	<b>0,75</b>	<b>0,67</b>	<b>0,53</b>
Ponderación	0,03	0,03	0,05	0,09	0,16	0,32	0,32	

Para verificar si la respuesta obtenida es correcta, es necesario calcular la razón de consistencia, la cual valida que no se haya producido contradicciones en los juicios. Un valor de este coeficiente inferior a 0.10 es considerado aceptable. Por el contrario, si es mayor a 0.10, los juicios no son confiables porque están demasiado cerca de la aleatoriedad y en ese caso, el ejercicio no tiene valor o debe repetirse (Paraskevopoulos). Para calcularlo, se aplica la siguiente fórmula (Hoyos, 2015):

$$CR = \frac{CI}{RI} = \frac{\text{Índice de consistencia de } A}{\text{Consistencia aleatoria de } A}$$

Donde:

$$CI = \frac{n_{max} - n}{n - 1} \quad ; \quad RI = \frac{1.98(n - 2)}{n}$$

En la Tabla 5 se recopila la razón de consistencia de cada una de los criterios evaluados, donde se observa que los valores obtenidos son menores o iguales a 0.10, demostrando así que tanto los puntajes asignados como el resultado final son confiables y válidos.

**Tabla 5**

*Matriz de Razón de Consistencia de los criterios evaluados*

	Matriz de Razón de Consistencia							
	Lanzamiento	Disponibilidad	Acciones de mercado	Tasa de crecimiento	Servicios	Modelos de precios	Trabajos con robot NAO	Comparación Criterios
<b>Razón de Consistencia</b>	0,04	0,09	0,08	0,07	0,04	0,04	0,03	0,10

### **Diseño de arquitectura para sistema Cloud Robotics**

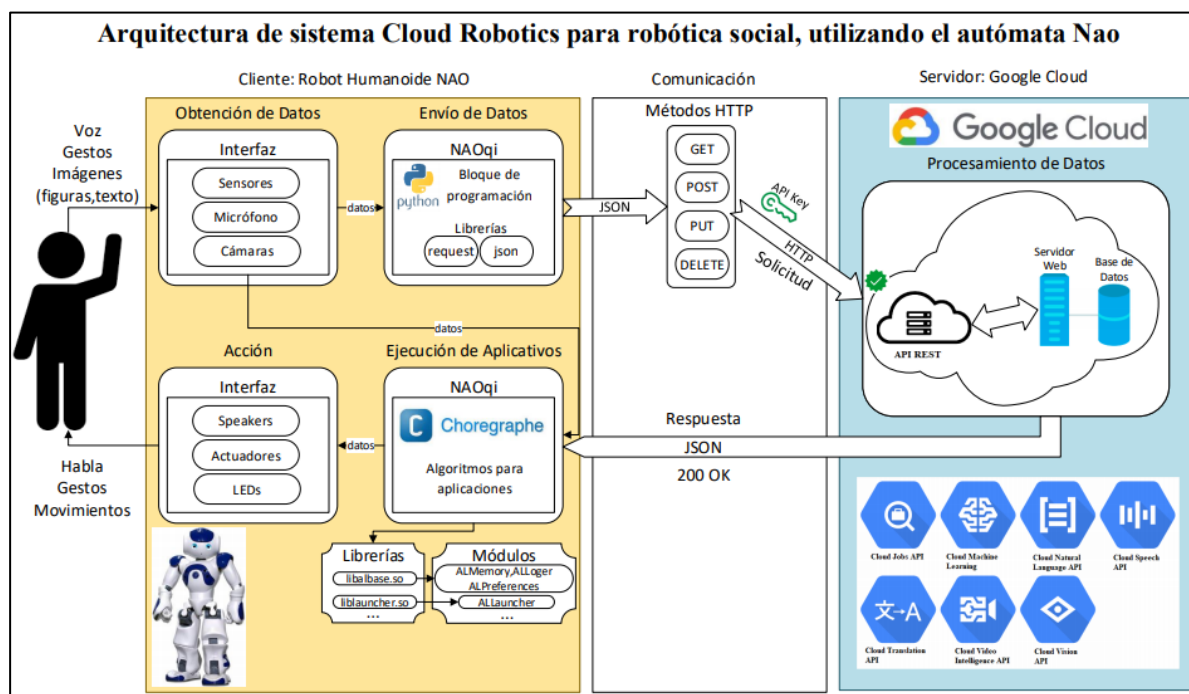
El presente sistema Cloud Robotics se centra en la conexión del robot humanoide NAO a Internet, con el fin de utilizar las diferentes APIs disponibles en una plataforma de servicios en la nube previamente seleccionada, y adicionar nuevas funcionalidades a la biblioteca de módulos interna del robot.

A su vez, disminuir la carga de procesamiento y la utilización de recursos de software, delegando estas tareas a la nube donde algoritmos avanzados y bases de datos extensas se integran para el funcionamiento de estas APIs.

Esto es factible gracias a que, dentro de las posibilidades de programación, NAO permite crear código para enriquecer las librerías de Choregraphe (software de programación por bloques) mediante el lenguaje de programación Python. En la Figura 16 se muestra la propuesta de arquitectura del sistema Cloud Robotics:

Figura 16

Arquitectura de sistema Cloud Robotics para robótica social, utilizando el autómeta NAO



El sistema inicia con la obtención de datos como voz, gestos, imágenes de figuras o texto entregados por el usuario, mediante los sensores, micrófono y cámaras integradas al robot. Posteriormente esta información será enviada a la nube mediante algoritmos de programación realizados en Python donde se utilizan librerías para convertir los datos en formato JSON y enviar solicitudes a la nube mediante métodos HTTP. Antes de acceder a los servicios, es necesario generar un código o API Key que autentifica al usuario y rastrea las solicitudes a las APIs utilizadas para conocer la cuota y facturación. Es único e intransferible.

Una vez que la información ha llegado, el servicio elegido procesa la misma y devuelve el resultado, en formato JSON, junto con un código que permite conocer el estado de respuesta HTTP. Si se ha realizado correctamente, su valor será de 200,

caso contrario será necesario obtener su valor con el fin de identificar el posible error generado en las tablas de códigos preestablecidas. En la Tabla 6 se muestra el tipo de códigos existentes y su interpretación.

**Tabla 6**

*Ejemplos de códigos de estado de respuesta HTTP*

Código	Rango	Tipo	Ejemplos
1xx	100 – 199	Mensaje Informativo	100 Continue
			101 Switching Protocol
			102 Processing
2xx	200 – 299	Respuestas Exitosas	200 OK
			201 Created
			202 Accepted
			300 Multiple Choice
3xx	300 – 399	Redirecciones	301 Moved Permanently
			302 Found
			400 Bad Request
4xx	400 – 499	Errores del Cliente	401 Unauthorized
			403 Forbidden
			500 Internal Server Error
5xx	500 – 599	Errores del Servidor	501 Not Implemented
			502 Bad Gateway

*Nota.* Obtenido de HTTP response status codes, de (Mozilla, 2020)

Los datos obtenidos de la nube se utilizan, en conjunto con los módulos preexistentes en el NAO, para la ejecución de aplicativos de robótica social donde el usuario interactúa activamente con el robot mediante speakers, actuadores y LEDs integrados en el mismo.

## Diseño de aplicativos para robótica social

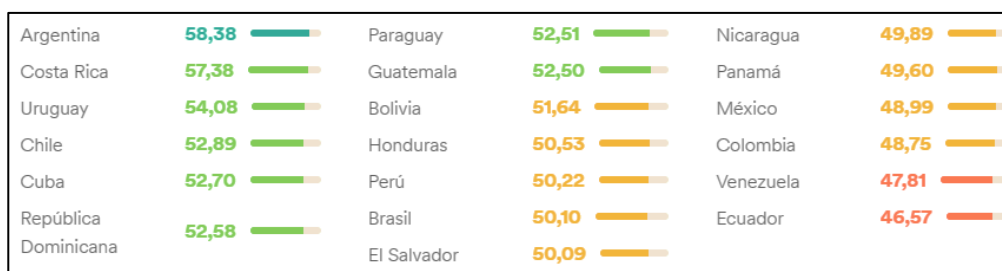
### *Aplicativo para educación*

**Planteamiento del problema.** Desde 2011 la empresa Education First realiza una evaluación de idioma a miles de personas (jóvenes y adultos) alrededor del mundo. Esto, con el fin de identificar el índice de inglés en cada país participante, y realizar un ranking mundial que se divide en cinco categorías, de acuerdo al nivel de aptitud: muy alto, alto, medio, bajo y muy bajo (Education First, 2011).

En 2019, Ecuador obtuvo el porcentaje más bajo en América Latina con una puntuación de 46.57/100 como se observa en la Figura 17 colocándolo en la posición #19/19 dentro de la región, y en la posición #81/100 a nivel mundial (Education First, 2015).

**Figura 17**

*Puntajes EF EPI de los países de América Latina en 2019*



*Nota.* Obtenido de *Índice del EF English Proficiency*, de (Education First, 2015)

Es por ello que, en pro de la enseñanza del idioma inglés, se plantea la realización de un aplicativo orientado al refuerzo de vocabulario, usando como técnica de enseñanza tarjetas de aprendizaje o flashcards.

Estas tarjetas (físicas o virtuales) contienen palabras, imágenes, símbolos o números en uno o ambos lados. Se utilizan para adquirir conocimientos, en diferentes campos de interés, al memorizar su contenido mediante el repaso espaciado y repetitivo de las mismas (Velasco Pumasunta, 2017).

### Figura 18

#### Flashcards



*Nota.* Obtenido de *Dear Zoo - Play ideas and printables for preschool*, de (YouCleverMonkey.com, 2015)

Existen diversos estudios que validan la efectividad de estas tarjetas en la docencia del idioma Inglés, por ejemplo, una evaluación a 29 estudiantes de sexto grado, cuyo puntaje inicial promedio era de 66.21/100, mejoró su desempeño en la comprensión de vocabulario obteniendo un puntaje de 79.55/100, después de utilizar flashcards durante todo el período académico 2017-2018 (Titin & Rizkilillah, 2018).

Esto debido a que son una ayuda visual muy simple que permite al profesor involucrar activamente al estudiante, a través de juegos, durante el proceso de enseñanza/aprendizaje. Una gran ventaja que presentan este tipo de tarjetas es su versatilidad ya que pueden combinarse con un sinnúmero de actividades complementarias, como lo expresa el autor Eduardo Gonzáles (2013): “El uso de

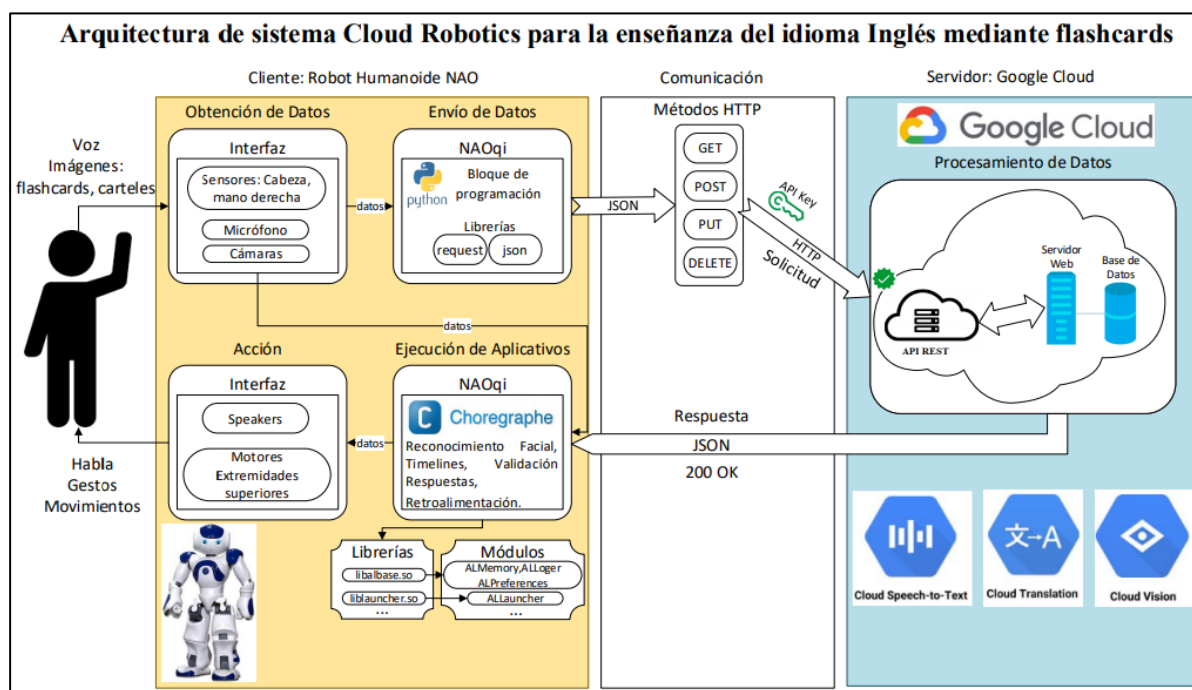


sonidos acompañados de flashcards, ayudan al desarrollo del léxico del inglés, ya que la familiaridad de la voz del hablante, con la ortografía de las palabras son un factor de influencia en el reconocimiento cognitivo de los estudiantes.”

**Diseño de la aplicación.** Para el desarrollo de esta aplicación, se propone el empleo de los servicios de la nube enfocados en visión artificial para el reconocimiento de objetos y texto en una imagen, así como Speech To Text para la validación de respuestas auditivas. En la Figura 19 se muestra la arquitectura planteada, especificando las interfaces de entrada/salida de datos, servicios de la nube y bloques de función propias de Choregraphe a utilizar.

**Figura 19**

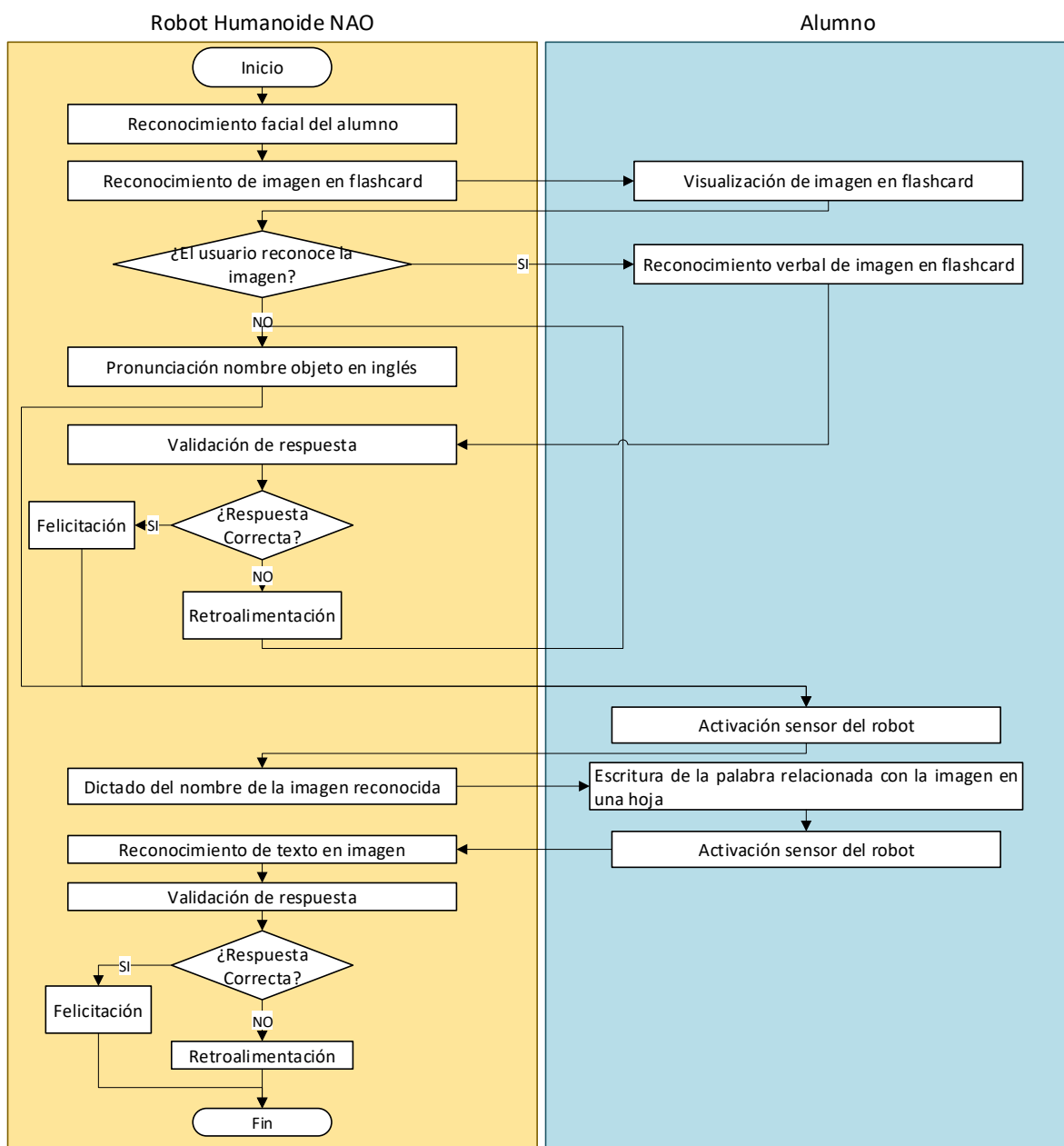
*Arquitectura de sistema Cloud Robotics para la enseñanza del idioma inglés mediante flashcards*



Así, con la información recopilada, se plantea un juego de cartas dividido en las siguientes etapas, como se muestra en la Figura 20.

**Figura 20**

*Diagrama de flujo del aplicativo de educación*



### ***Aplicativo para cuidados de la salud***

**Planteamiento del problema.** Según la Organización Mundial de la Salud (OMS), la adherencia al tratamiento farmacológico se define como la medida en la que el comportamiento del paciente, al ingerir su medicación, seguir una dieta y/o realizar cambios en su estilo de vida, corresponde con las recomendaciones acordadas con su médico. (World Health Organization, 2003)

En un sinnúmero de casos, la falta de adherencia al tratamiento se presenta como una gran barrera en el cuidado de la salud, impidiendo que el paciente logre sobrellevar, de manera exitosa, una enfermedad crónica.

Este hecho se ve reflejado en los datos obtenidos por el Dr. Javier Soto, del Departamento de Farmacoeconomía de Pfizer, quien indica que “la mala adherencia es responsable de entre el 5 y 10% de los ingresos hospitalarios, de 2,5 millones de urgencias médicas y 125.000 fallecimientos al año en Estados Unidos” (Pfizer, 2009).

Por ello, es importante involucrar al paciente y hacerlo consciente de su rol en la efectividad de un tratamiento farmacológico, al utilizar herramientas que le ayuden a recordar constantemente la toma de medicamentos. Entre las favoritas se encuentran las cajas para píldoras, recordatorios a través de mensajes de texto y calendarios para llenar con pegatinas (Rife, y otros, 2012).

Sin embargo, y gracias a los avances tecnológicos, se han creado nuevas soluciones dentro de este campo, utilizando interfaces y dispositivos más amigables e interactivos para el usuario. Así se tienen aplicaciones de recordatorios para celulares (Stawarz, Cox, & Blandford, 2014), recordatorios mediante diálogos sintéticos (Wolters, Johnson, Campbell, DePlacido, & McKinstry, 2014) e incluso la utilización de asistentes

robóticos para el recordatorio y entrega de medicamentos, como se muestra en la Figura 21 (Jayawardena, Kuo, Aleksandar, & Datta, 2011).

### **Figura 21**

*Robot Healthbot siendo utilizado por una persona mayor*



*Nota.* Obtenido de *Feasibility study of a robotic medication assistant for the elderly*, de (Jayawardena, Kuo, Aleksandar, & Datta, 2011)

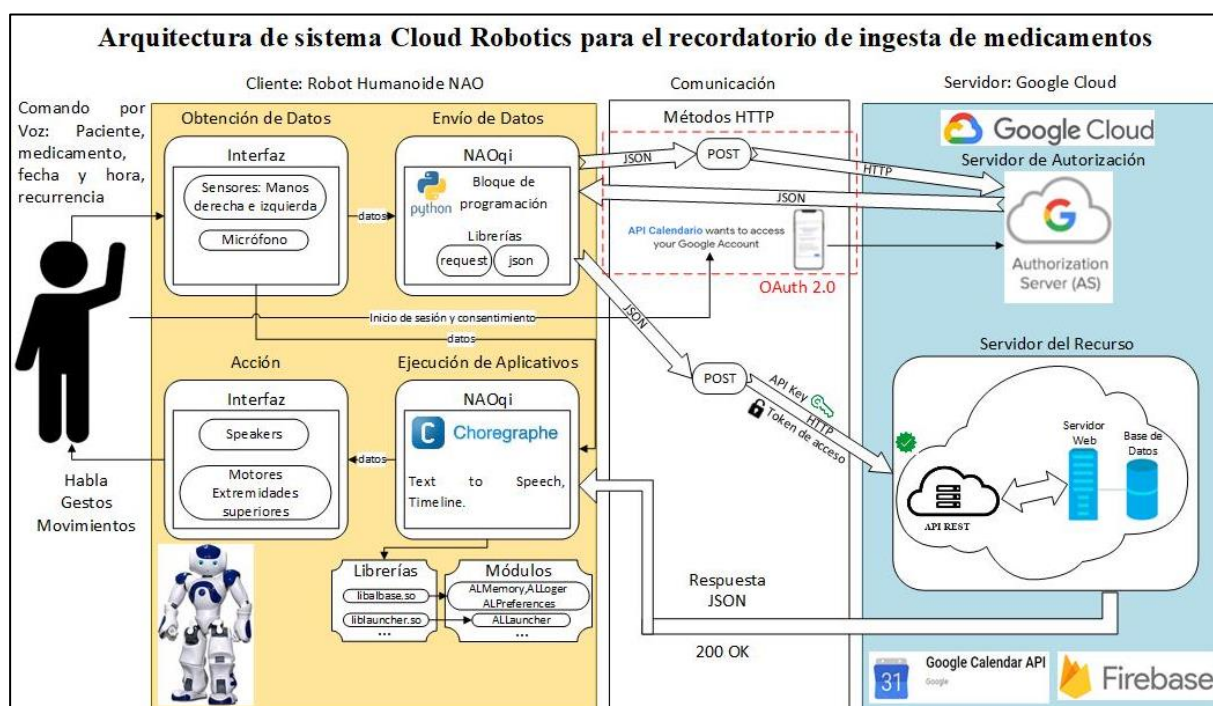
Este último ha logrado resultados satisfactorios utilizando comandos simples para la interacción con el usuario, mediante una pantalla táctil y una aplicación conformada por diferentes pantallas, condicionadas por las respuestas dadas a preguntas preestablecidas.

**Diseño de la aplicación.** La clave del éxito se basa en la facilidad del manejo de estas herramientas, las cuales no demandan mayor esfuerzo físico o intelectual para su funcionamiento. Por ello, se plantea un aplicativo orientado para el agendamiento y recordatorio de la ingesta de medicamentos mediante comandos de voz, utilizando el servicio Google Calendar API.

En la Figura 22 se muestra la arquitectura planteada, especificando las interfaces de entrada/salida de datos, servicios de la nube y bloques de función propias de Choregraphe a utilizar.

## Figura 22

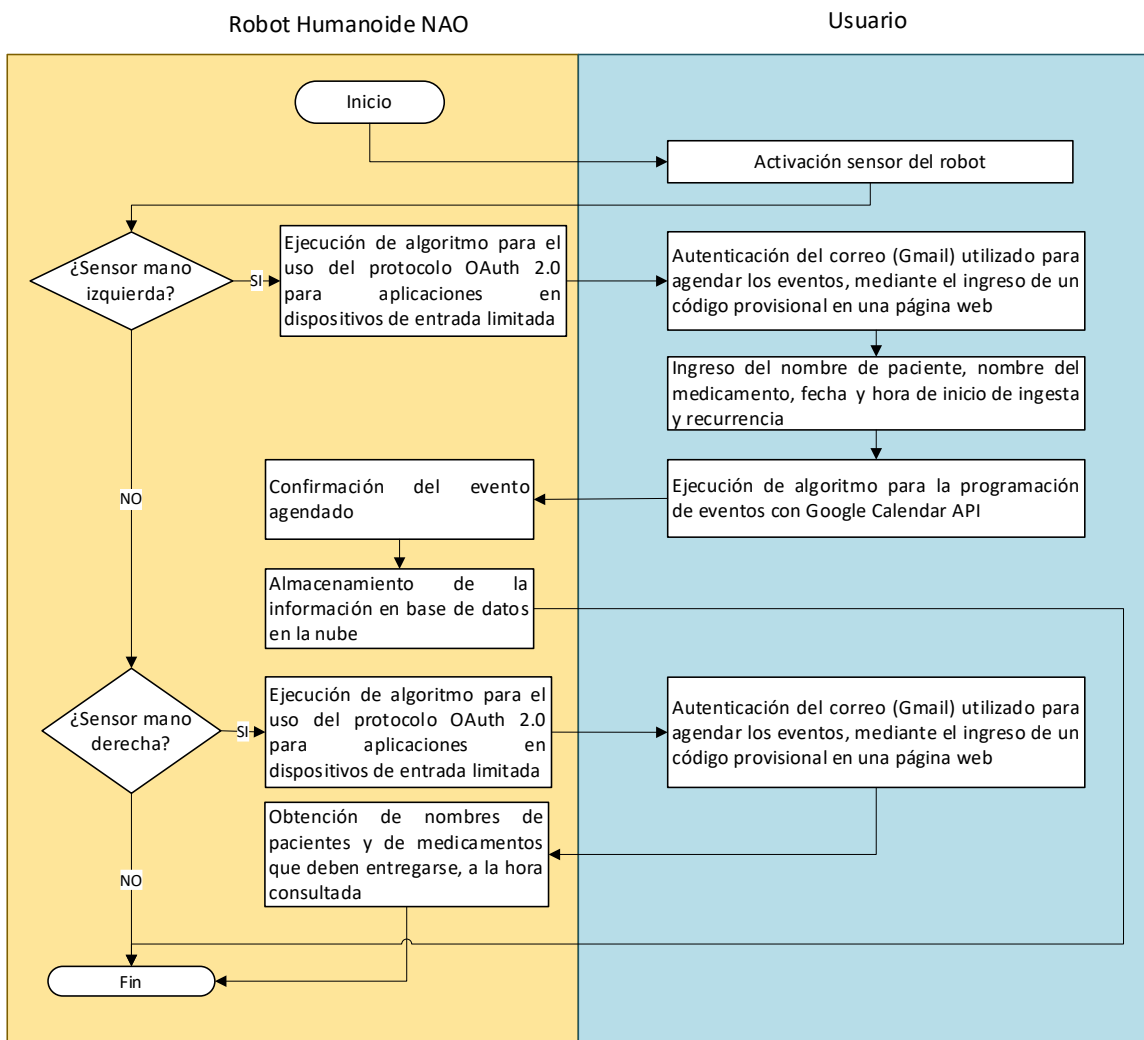
*Arquitectura de sistema Cloud Robotics para el recordatorio de ingesta de medicamentos*



Adicional, se muestra en la Figura 23 el diagrama de flujo de la dinámica del aplicativo de salud.

Figura 23

Diagrama de flujo del aplicativo de salud



### ***Aplicativo para atención al cliente***

**Planteamiento del problema.** En el año 2020, la pandemia de COVID-19 se extendió por todo el mundo, obligando a los países a tomar medidas de aislamiento parcial y total durante todo este año, impidiendo el ejercicio de toda actividad económica donde la interacción humana es indispensable.

Sin embargo, a finales de este año, e inicios del 2021 se ha evidenciado una reintegración de las personas a sus actividades diarias, bajo el estricto control de medidas de bioseguridad implementadas como: distanciamiento social, uso de mascarillas, lavado de manos, la prohibición de eventos sociales y aglomeración de personas en espacios cerrados, entre otros.

Debido a esto, se ha planteado la posibilidad de utilizar robots, en el campo de atención al cliente, para brindar un servicio de calidad sin contacto directo con el usuario. La actual crisis pandémica está demostrando la utilidad y la expansión de los roles de los robots en una amplia gama de servicios e industrias.

Por ejemplo, los robots de servicio pueden aliviar la carga de trabajo de los socorristas en los hospitales mientras los mantienen a salvo del contagio, realizar tareas para que los humanos puedan practicar el distanciamiento social, ejecutar procedimientos médicos delicados, etc. (Matthews, 2020).

En el pasado, las principales razones para usar robots eran: aumentar la eficiencia y precisión, acortar el tiempo de servicio y realizar tareas repetitivas e inseguras, siendo esta última orientada más hacia los robots industriales, pero la pandemia actual ha creado una normalidad completamente nueva donde el uso de

aplicaciones y robots de servicio se volvieron frecuentes para la seguridad de las personas ante el virus (Hrynowski, 2020).

Así la pandemia, en realidad, puede haber creado un punto de inflexión para la adopción de robots de servicio en áreas como la atención médica, la entrega de alimentos y la seguridad pública (Lew, 2020). Algunos ejemplos de robots utilizados actualmente en el área de servicio al cliente son (Underwood, 2020):

- **Lowebot:** desarrollado por Fellow Robots e introducido en 2016 a las tiendas Lowe en San Francisco, este robot se encarga de ayudar al cliente a encontrar un producto dentro de la tienda, responder preguntas básicas, entre otras.

#### **Figura 24**

*Robot Lowebot guiando al cliente a través de la tienda Lowe*



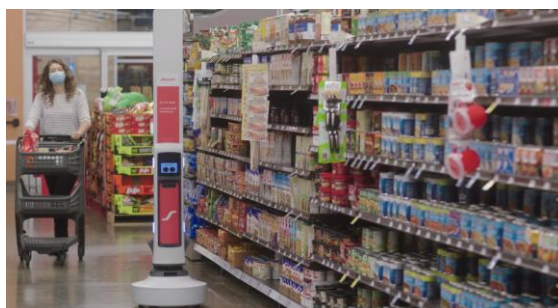
*Nota.* Obtenido de *A Helping Hand*, de (Lowe's Innovation Labs, 2016)

- **Tally:** desarrollado por Simbe Robotics e introducido en 2016 a la cadena de supermercados Target, este robot está específicamente orientado al monitoreo de productos y precios en perchas dentro del establecimiento, mediante software de reconocimiento visual.



**Figura 25**

*Robot Tally en revisión de inventario en tiempo real*



*Nota.* Obtenido de *Say hello to Tally 3.0*, de (Simbe Robotics, 2018)

**Diseño de la aplicación.** El pleno uso de robots en el área de atención al cliente ha sido un tema difícil de tratar a lo largo de los años, debido a la baja aceptación por parte de los clientes y los aspectos técnicos que conllevan a su ejecución.

Sin embargo, y debido a la pandemia de COVID-19, ha surgido un cambio de pensamiento, donde ahora los robots son vistos como herramientas valiosas en pro de realizar tareas cotidianas con el fin de disminuir el número de contagios entre personas.

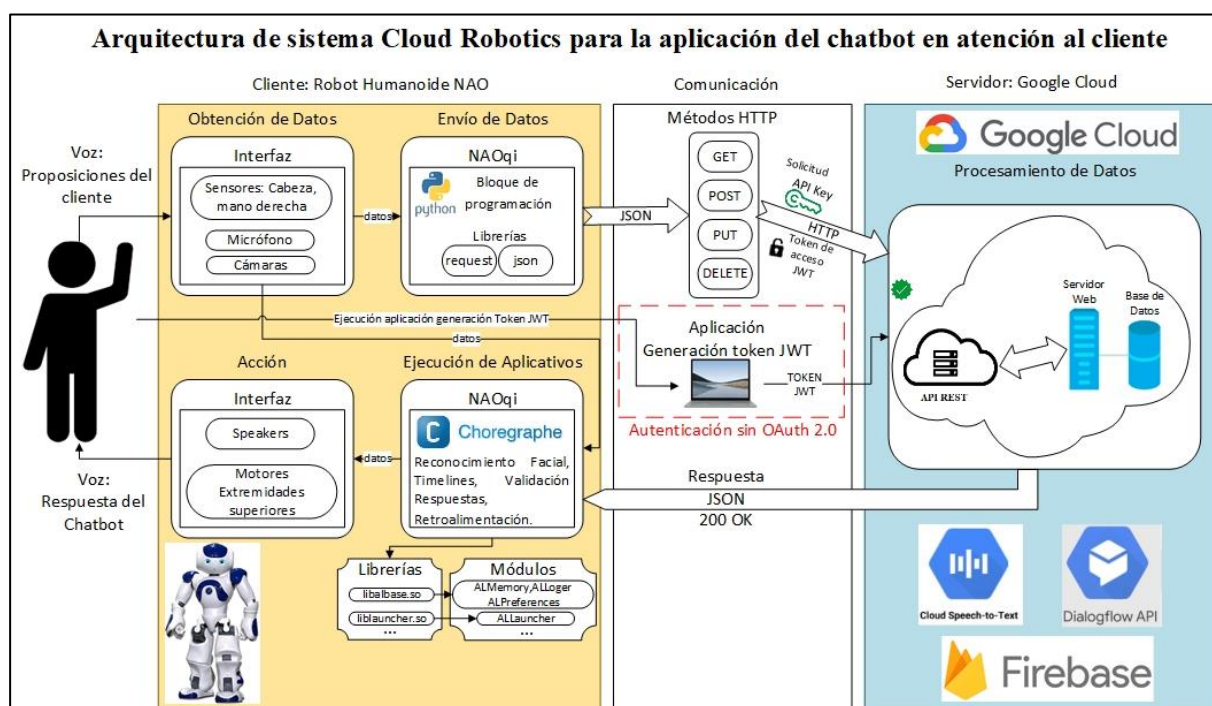
Si bien es cierto que, es una tarea extremadamente compleja el diseño de sistemas robóticos que puedan brindar una experiencia de compra completa, gracias a las nuevas tecnologías es posible crear aplicaciones orientadas a la ejecución de tareas básicas pero significativas para el cliente, en una cooperación continua entre el personal humano y el robótico.

A continuación, se plantea un aplicativo dirigido a la reserva de robots fijos y móviles disponibles en el Laboratorio de Robótica de la Universidad de las Fuerzas Armadas - ESPE, mediante la utilización de un chatbot en línea vinculado al robot humanoide NAO. Adicional, se integra una base de datos en la nube para el

almacenamiento de la información entregada por el usuario. En la Figura 26 se muestra la arquitectura planteada, especificando las interfaces de entrada/salida de datos, servicios de la nube y bloques de función propias de Choregraphe a utilizar.

**Figura 26**

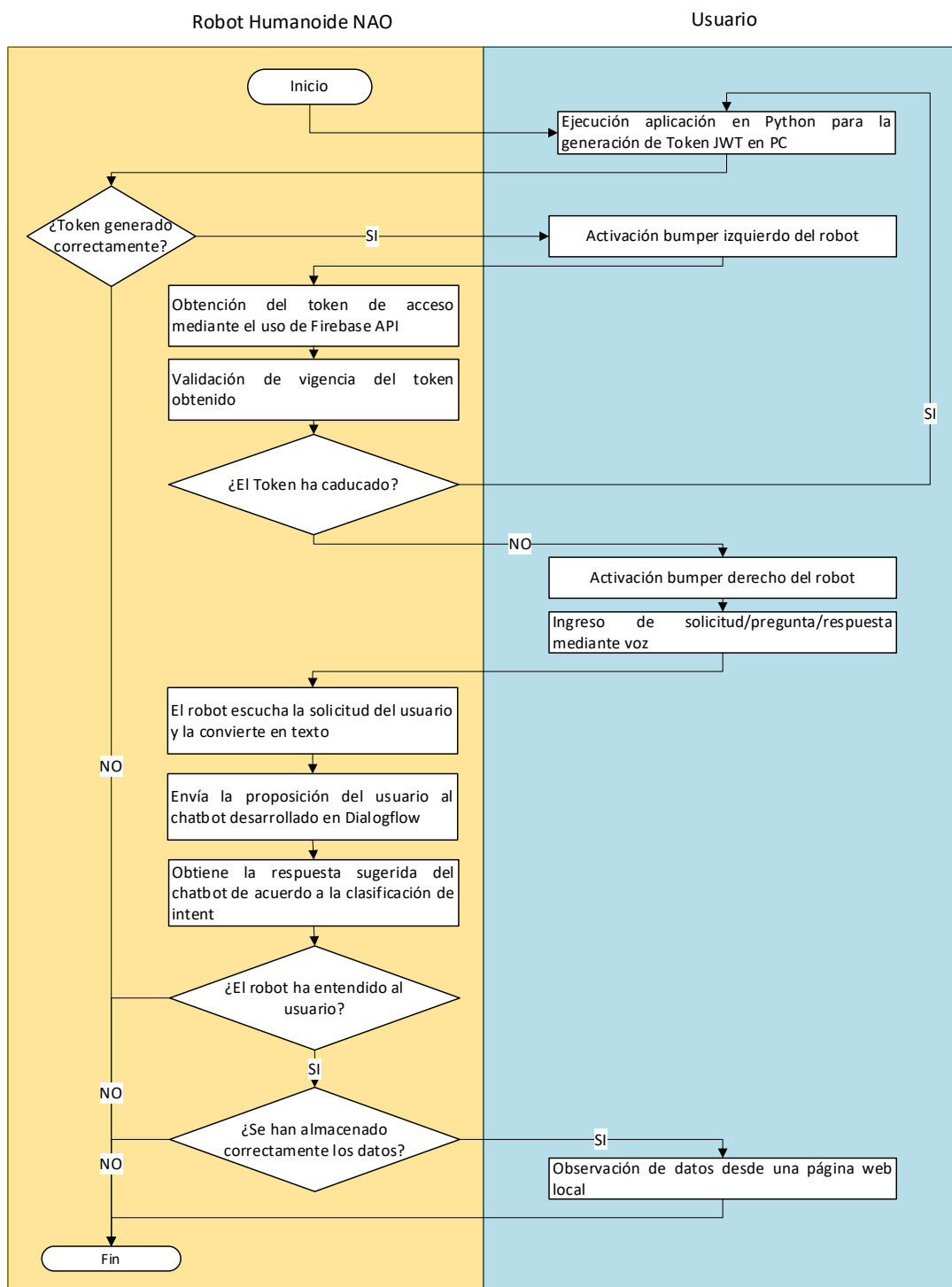
*Arquitectura de sistema Cloud Robotics para la aplicación del chatbot en atención al cliente*



Adicional, se muestra en la Figura 27 el diagrama de flujo de la dinámica del aplicativo de atención al cliente.

Figura 27

Diagrama de flujo de aplicativo de atención al cliente



## Capítulo IV

### Implementación, pruebas y resultados

#### Implementación de aplicaciones orientadas a la robótica social

Una vez planteada la problemática a tratar y la arquitectura de cada uno de los aplicativos, se procede a su implementación donde se aplicó la combinación de funcionalidades propias del robot en conjunto con los servicios en la nube de Google Cloud. Esto, con el fin de mostrar la influencia del cloud computing en el mejoramiento de sus capacidades y la adquisición de nuevas habilidades.

#### *Diseño de algoritmo de programación para la utilización de servicios en la nube*

Para el uso de las APIs que ofrece Google Cloud, es necesario desarrollar un algoritmo de programación en Python, mediante el cual se enviarán peticiones HTTP y se recibirán objetos de respuesta con la información resultante del cómputo en la nube.

Cabe mencionar que, estos algoritmos difieren para cada API en: recurso a utilizar, APIKEY e información a enviar. Sin embargo, todos siguen una estructura similar, por lo cual es reproducible para cada uno de los casos, pero deberá ser adaptable a las necesidades de cada aplicación. A continuación, se muestra un esquema general del programa diseñado para el llamado de cada API, en lenguaje Python, utilizando el SpeechToText API como ejemplo:

**Librerías y variables.** Para iniciar, se importan principalmente las siguientes librerías:

- **Request:** Permite enviar peticiones HTTP de una manera fácil.
- **Json:** Permite convertir diccionarios y listas de Python en cadenas JSON.

Posteriormente se declaran las siguientes variables que varían entre cada API:

- **APIKey:** Es un conjunto de caracteres único que permite identificar al desarrollador o programa que consume una API. Éste es comúnmente utilizado para registrar tasas de consumo y facturación. El desarrollador deberá generar una APIKey, diferente para cada API, en la consola de Google Cloud.
- **Endpoint:** Es la URL a la que se envían las solicitudes y donde reside el recurso que se desea utilizar. Para obtener esta información, es necesario consultar la documentación de la API correspondiente. En la Figura 28 se muestra el código implementado:

### Figura 28

*Código fuente del programa de librerías y variables para el uso del API Speech To Text*

```
#Importar librerías
import requests, base64, json, os

#Declaración de Variables
APIKEY="XXXXXXXXXXXX"
GOOGLE_SPEECH_URL=
"https://speech.googleapis.com/v1/speech:recognize?key={0}".format(APIKEY)
```

**Datos a enviar.** La información que será enviada y recibida al momento de consumir una API REST se encuentra en el formato JSON, cuyas características se mencionan en el Capítulo II, Fundamentación Teórica, en el apartado JSON.

Sin embargo, en el programa, los datos que serán enviados y procesados por la API inicialmente se escribirán como diccionarios, es decir, como una colección de pares clave-valor separados por comas dentro de un par de llaves. A continuación, en la Figura 29, se muestra la sintaxis de un diccionario en Python.

**Figura 29**

*Sintaxis de un objeto diccionario en Python*

```
d = { <clave>: <valor>,
      <clave>: <valor>,
      .
      .
      <clave>: <valor>}
```

Posteriormente, al momento de realizar la petición, este objeto será convertido a JSON con ayuda del módulo *json* importado al inicio del programa. Cabe mencionar que las claves y valores a colocar están definidos por el tipo de recurso de la API a utilizar, por ello es importante consultar su documentación respectiva, referente a la estructura del cuerpo de la solicitud. En la Figura 30 se muestra el código implementado:

**Figura 30**

*Código fuente del programa del cuerpo de la solicitud para el uso del API Speech To*

*Text*

```
#Conversión FLAC del audio
FLAC_CONV = 'flac -f'
filename = "C:\Users\grabacion.wav"
del_flac = False
if 'flac' not in filename:
    del_flac = True
    os.system(FLAC_CONV + ' ' + filename)
    filename = filename.split('.')[0] + '.flac'
#Conversión base64
f = open(filename, 'rb')
flac_cont = f.read()
base64_data = base64.b64encode(flac_cont)
f.close()
#Datos a enviar - diccionario Python
body = {
    'config': {
        'encoding': 'FLAC',
        'sampleRateHertz': '16000',
        'languageCode': 'es-ES'
    },
    'audio': {
        'content': base64_data
    }
}
```

**Petición.** Una vez definida la información a enviar, se procede a realizar la petición a la API, utilizando los métodos mencionados en el Capítulo II, Fundamentación Teórica, en el apartado Protocolo HTTP: get, post, put, delete, los cuales indican la acción que se desea realizar para un recurso determinado.

Nuevamente, el método a aplicar está dado por el tipo de recurso de la API a utilizar, por lo que es necesario revisar la documentación respectiva. Como se mencionó anteriormente, para la realización de peticiones HTTP, Python utiliza la librería o módulo llamado *requests*, cuya sintaxis es la siguiente:

### Figura 31

*Sintaxis de petición HTTP utilizando el módulo requests*

```
requests.methodname (params)
```

Donde *methodname* corresponde a los diferentes métodos HTTP disponibles para enviar solicitudes. Cabe mencionar que la sintaxis de la solicitud difiere entre métodos, por lo que es necesario verificarla para cada caso. En la Figura 32 se muestra el código implementado:

### Figura 32

*Código fuente del programa de petición HTTP para el uso del API Speech To Text*

```
#Petición HTTP a la API SpeechToText de Google Cloud  
res = requests.post(GOOGLE_SPEECH_URL, data=json.dumps(body), verify=False)
```

Para utilizar el recurso *speech.recognize* del API SpeechToText, se utiliza el método **POST**, para el cual corresponde la siguiente sintaxis:

### Figura 33

*Sintaxis de petición HTTP utilizando el método post*

```
requests.post(url, data={key: value}, args)
```

Donde:

- *url*: Es la URL a la que se enviará la petición.
- *data*: Es la información que se enviará junto con la petición.
- *args*: Son argumentos opcionales que pueden añadirse a la petición.

Si comparamos con la Figura 32, se concluye lo siguiente:

- **GOOGLE\_SPEECH\_URL**: Es la variable definida al inicio del programa, conocida como endpoint.
- **data**: Se especifica la información a enviar a la API. Utilizando la función “*json.dumps()*”, se transforma de un objeto tipo diccionario a uno con formato JSON.
- **verify**: Permite especificar si se desea verificar o no el certificado TLS de los servidores. En este caso se coloca en *false* para evitar errores.

**Respuesta.** Una vez realizada la petición, el API responderá con un objeto el cual contiene, entre otras cosas, el código de status de la respuesta y la información resultante de la operación ejecutada.

Para continuar, es importante verificar que el código de status obtenido sea **200**, el cual significa que la solicitud ha tenido éxito. Si éste es diferente, es necesario consultar su significado ya que, dependiendo de su valor, arrojará información sobre errores provenientes del cliente o del servidor.



Para conseguir los datos resultantes, se debe extraer el objeto JSON de la respuesta, utilizando el método “*json()*” e identificar el campo al que corresponde. Cabe mencionar que, en la documentación del recurso utilizado, dentro del apartado *cuerpo de respuesta* se muestra la estructura que presentará dicha información. En la Figura 34 se muestra el código implementado:

### Figura 34

*Código fuente del programa de la respuesta obtenida del API Speech To Text*

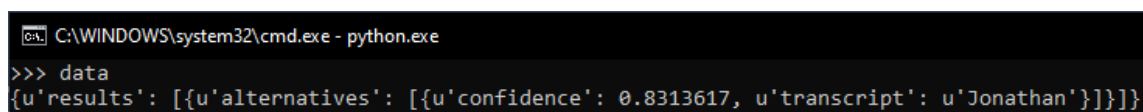
```
if res.status_code == 200:
    data = res.json()
    result = data['results']
    if len(result) > 0:
        if len(result[0]['alternatives']) > 0:
            text = result[0]['alternatives'][0]['transcript'].encode('utf')
            print("Texto")
            print(text)
else:
    print(res.status_code)
    print(res.text.encode('utf8'))
```

Ejecutando el programa en una PC con Python, se observa lo siguiente:

- Al imprimir la variable *data*, se obtiene un objeto JSON con un único campo llamado **results**. Si el tamaño de este objeto es mayor a 0, se concluye que se ha obtenido, al menos, una transcripción del audio enviado.
- A continuación, se obtiene otro objeto dentro de *results*, llamado **alternatives**. Si el tamaño de este objeto es mayor a 0, se concluye que se ha obtenido, una o más hipótesis de reconocimiento.
- Finalmente, dentro del objeto *alternatives*, se extrae la primera alternativa ya que será la más probable, según la clasificación del reconocedor, y se accede al campo **transcript**, que contiene el texto deseado.

### Figura 35

*Impresión de la variable data con la respuesta a la petición HTTP en Python*

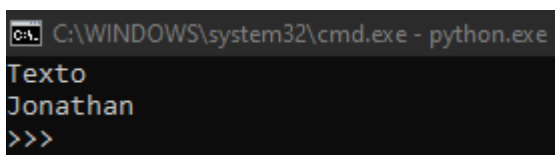


```
C:\WINDOWS\system32\cmd.exe - python.exe
>>> data
{'results': [{'alternatives': [{'confidence': 0.8313617, u'transcript': u'Jonathan'}]}]}
```

En este caso, el texto obtenido del audio enviado al SpeechToText API de Google Cloud es “Jonathan”

### Figura 36

*Texto obtenido del audio procesado por el API Speech To Text*



```
C:\WINDOWS\system32\cmd.exe - python.exe
Texto
Jonathan
>>>
```

Una vez obtenida la información resultante de las operaciones ejecutadas en la nube, ésta puede ser integrada y utilizada en otras aplicaciones o programas. Para continuar con el siguiente paso, es necesario que el código diseñado haya sido probado en la versión de Python utilizada por el robot humanoide NAO.

Esto, con el fin de descartar errores de compatibilidad al momento de su implementación en los bloques de programación.

**Diseño de bloques de programación.** Para la implementación del código, previamente diseñado, para el consumo de APIs de Google Cloud en el robot NAO se utilizará el bloque *Python Script* de Choregraphe, el cual se muestra en la Figura 37.

**Figura 37**

*Bloque de programación en Choregraphe para código en Python*



El bloque, por defecto, está compuesto por dos entradas y una salida:

- **onStart Input:** El comportamiento del bloque comienza cuando se recibe una señal en esta entrada.
- **onStop Input:** El comportamiento del bloque se detiene cuando se recibe una señal en esta entrada.
- **onStopped Output:** Envía una señal cuando finaliza el comportamiento del bloque.

Cabe mencionar que se pueden agregar/eliminar entradas y salidas, de acuerdo a la necesidad del código a ejecutar. Al hacer doble clic en el bloque, se abrirá la ventana que se observa en la Figura 38 con un script base.

**Figura 38**

*Script base del bloque de programación Python en Choregraphe*

```

Script editor
Python Script X
1 class MyClass(GeneratedClass):
2     def __init__(self):
3         GeneratedClass.__init__(self)
4
5     def onUnload(self):
6         #put initialization code here
7         pass
8
9     def onReload(self):
10        #put clean-up code here
11        pass
12
13    def onInput_onStart(self):
14        #self.onStopped() #activate the output of
the box
15        pass
16
17    def onInput_onStop(self):
18        self.onUnload() #it is recommended to reuse
the clean-up as the box is stopped
19        self.onStopped() #activate the output of the
box
20
Find

```

Como se mencionó anteriormente, los algoritmos para el consumo de diferentes APIs son similares en estructura, pero guardan diferencias en los datos enviados, recibidos, entre otras características, y esto también se ve reflejado en el diseño del bloque de programación.

Adicional, es necesario utilizar bloques complementarios para el correcto funcionamiento del algoritmo. Continuando con el ejemplo del SpeechToText API, se tiene lo siguiente:

**Implementación del algoritmo de Python en Choregraphe.** Una vez el código se encuentre depurado y comprobado, se lo ingresa al bloque de Python, adaptando la sintaxis de algunas funciones, a las utilizadas en Choregraphe. Posteriormente se modifican las entradas y salidas del bloque, tomando en consideración los datos de entrada (enviados a la API) y salida (obtenidos de la API).

En la Figura 39 se muestra el bloque de programación para el consumo del SpeechToText API de Google Cloud, mediante peticiones HTTP. Se observa que tanto la primera entrada como la salida son de color celeste, esto refiere a que son de tipo STRING, es decir, representa un evento que lleva datos. Estos datos son una cadena de caracteres o una matriz de cadenas de caracteres. El código fuente del programa se muestra en Anexo II., Figura 122.

### Figura 39

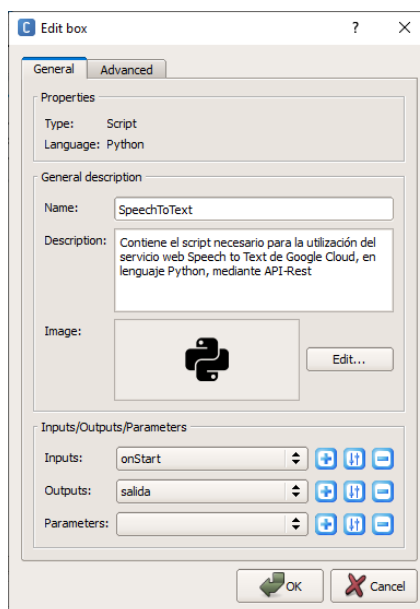
*Bloque de programación para el uso del API Speech To Text, en Choregraphe*



En la primera entrada ingresa la ruta de almacenamiento interna de la imagen capturada por una de las cámaras del robot, así el algoritmo podrá acceder a ella para transformarla a un formato compatible con la API a utilizar. En cambio, la salida envía la transcripción de audio a texto, obtenida como resultado del procesamiento de la API. En la Figura 40 se muestra la configuración del bloque, donde se describen: nombre, descripción, imagen del bloque, entradas, salidas y parámetros.

### Figura 40

*Configuración del bloque de programación para el uso del API Speech To Text*



Cabe mencionar que, la implementación de este bloque es importante debido a la inexistencia de operaciones donde el robot NAO realice la conversión de audio a texto. Lo más similar que se puede encontrar localmente, es un bloque de reconocimiento de voz en el cual es necesario preestablecer una lista de palabras que, al ser mencionadas por el usuario, activarán la señal de salida del mismo.

**Adición de bloques complementarios.** Como se mencionó anteriormente, es necesaria la adición de bloques complementarios para el correcto funcionamiento del algoritmo. En este caso, los bloques añadidos son los siguientes:

- **Record Sound**

**Figura 41**

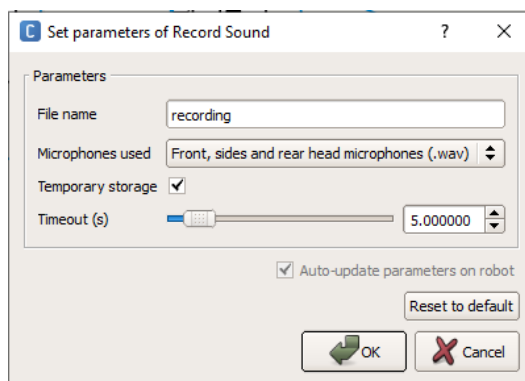
*Bloque de programación para grabación de audio en Choregraphe*



Graba sonidos en el robot, tanto en formato “.ogg” como en formato “.wav”. Al finalizar la grabación y, como salida, devuelve la ruta absoluta del archivo de sonido almacenado dentro del robot. En la Figura 42 se muestra la ventana de configuración de parámetros del bloque.

**Figura 42**

*Configuración del bloque de programación para grabación de audio en Choregraphe*



- **Nombre Archivo:** Nombre con el que se guarda el archivo de audio grabado.

- **Micrófono usado:** Permite elegir grabar audio con el micrófono de cabeza frontal, y lo guarda en formato ogg, o con cuatro micrófonos situados en la cabeza, en la parte frontal, laterales y posteriores, en formato wav.
- **Almacenamiento Temporal:** Si este campo no está marcado, el archivo de sonido de salida se guardará en "~ / recordings / microphones / <Nombre de archivo>", de lo contrario, se guardará en un directorio temporal.
- **Timeout (s):** Tiempo de duración de la grabación en segundos.

En este caso, este bloque se utiliza para grabar el archivo de audio del que se quiere extraer el texto. Es importante establecer la duración de audio, dependiendo de las respuestas que se grabarán del usuario. Por ejemplo, para la obtención de un nombre y un apellido, un timeout de 5 segundos es suficiente.

- **Play Sound**

### Figura 43

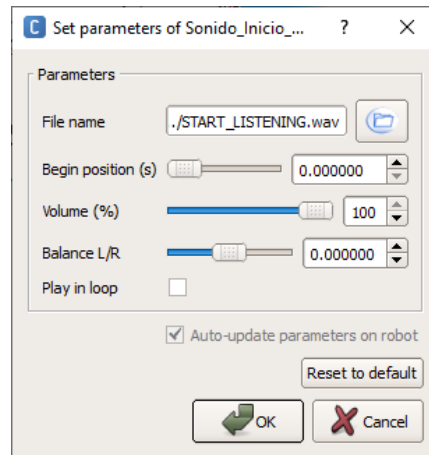
*Bloque de programación para reproducción de sonidos, en Choregraphe*



Reproduce un archivo de sonido específico. El formato del archivo puede ser ".wav", ".ogg" y ".mp3". Para seleccionar el archivo, es necesario importarlo previamente a la carpeta del proyecto. En la Figura 44 se muestra la ventana de configuración de parámetros del bloque.

**Figura 44**

*Configuración del bloque de programación para reproducción de audio en Choregraphe*



- **Nombre Archivo:** Ruta donde se encuentra importado el archivo a reproducir.
- **Posición de inicio:** Tiempo de inicio en segundos
- **Volumen:** Permite configurar el volumen con el que se reproducirá el archivo de audio
- **Balance L/R:** Permite balancear si el audio se reproducirá por la bocina izquierda o derecha.
- **Reproducir en loop:** Si está marcado, permite habilitar que el audio se reproduzca en bucle.

La utilización de este bloque no influye en el funcionamiento de la operación SpeechToText, sino que, funciona como un complemento para facilitar la interacción entre el humano y el robot ya que, notificará el inicio y fin de la grabación de audio mediante la reproducción de sonidos cortos (beeps).

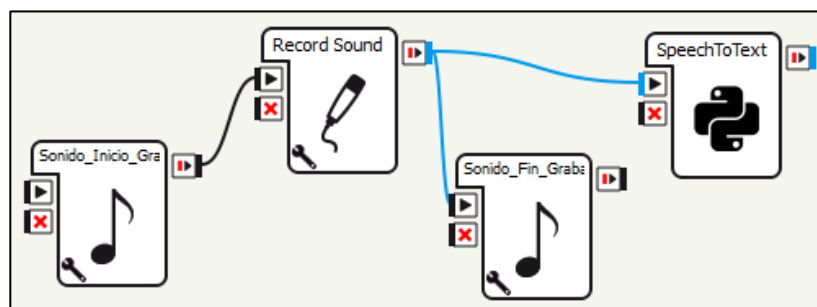


**Creación de un bloque de programación principal.** Una vez colocados, ordenados e interconectados los bloques de programación, se los unifica en un bloque principal con el fin de manejar un solo elemento y facilitar su réplica en diferentes aplicaciones o diferentes segmentos de un mismo programa.

Para ello, es importante identificar qué bloque se encuentra primero y último, debido a que, las entradas del primero y las salidas del último deberán ser conectadas a las entradas y salidas del bloque madre. En la Figura 45 se muestran todos los bloques descritos anteriormente, que serán unificados en un bloque madre.

**Figura 45**

*Conexión de bloques de programación para el uso del API Speech To Text*

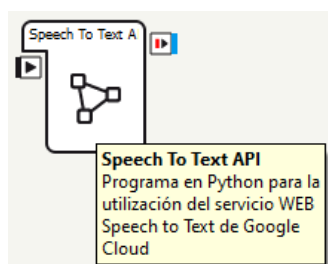


Para ello, es necesario seleccionar todos los bloques, hacer clic derecho y seleccionar la opción *Convert to box*. Aparecerá una nueva ventana, ingresar el nombre del nuevo bloque y, si se desea, una descripción breve del mismo. Hacer clic en el botón *OK*.

A continuación, aparecerá un solo bloque en la pantalla como el mostrado en la Figura 46 y, al hacer doble clic, aparecerán dentro de este, los bloques previamente seleccionados.

**Figura 46**

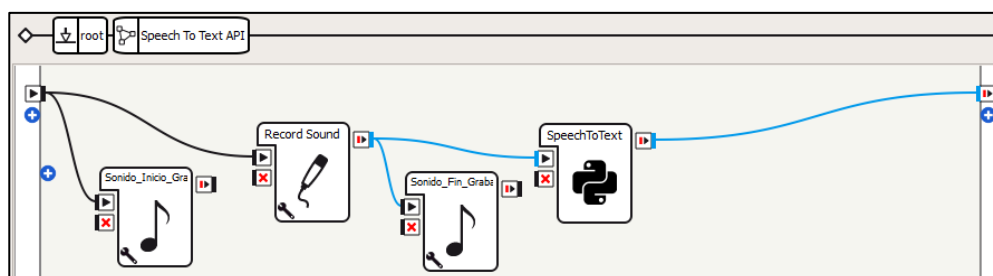
*Bloque de programación madre para el uso del API Speech To Text en el robot NAO*



Para finalizar, es necesario configurar y conectar las entradas y salidas. En la Figura 47 se muestra el bloque creado. En la Figura 48 se muestra el diagrama de flujo correspondiente al funcionamiento del bloque.

**Figura 47**

*Conexión de bloques, dentro del bloque madre, del API Speech To Text*



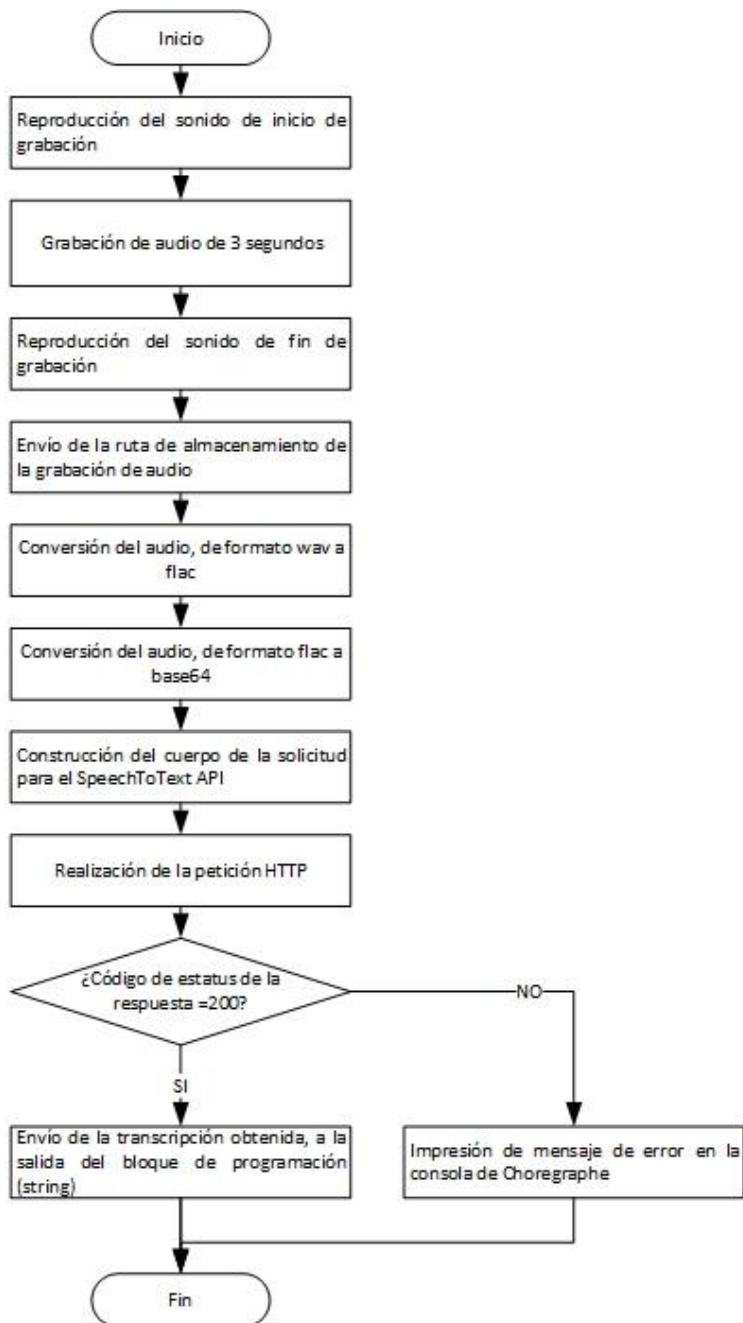
Antes de copiar el bloque y utilizarlo para otras aplicaciones, es importante realizar pruebas de funcionamiento para verificar que no existan errores en su ejecución. Ya comprobado, solo será necesario copiar y pegar el bloque madre en las partes del programa donde se necesite transformar audio a texto.

Este proceso se repetirá para los diferentes servicios en la nube a utilizar, en cada uno de los aplicativos diseñados. Esto dará como resultado, bloques de programación fáciles de reproducir e implementar por desarrolladores futuros quienes,

no necesitan tener una comprensión total del funcionamiento interno del bloque, sino solo conocer la información a ingresar y obtener de las entradas y salidas del mismo.

**Figura 48**

*Diagrama de flujo del funcionamiento del bloque SpeechToText API*



### ***Aplicativo para educación del idioma inglés***

Para el desarrollo del aplicativo orientado a la enseñanza oral y escrita de palabras en el idioma inglés, se ha planteado la siguiente estructura:

**Reconocimiento del usuario.** Para iniciar con este aplicativo, en forma de juego, es importante relacionar al robot con el usuario, de una manera más personal, mediante el aprendizaje y reconocimiento de su rostro ligado a su nombre. Para ello, el robot utiliza la función *Face Reco*, el cual detecta la cara de las personas y reconoce las conocidas por el robot. Este bloque puede arrojar dos posibles resultados, los cuales determinarán los escenarios:

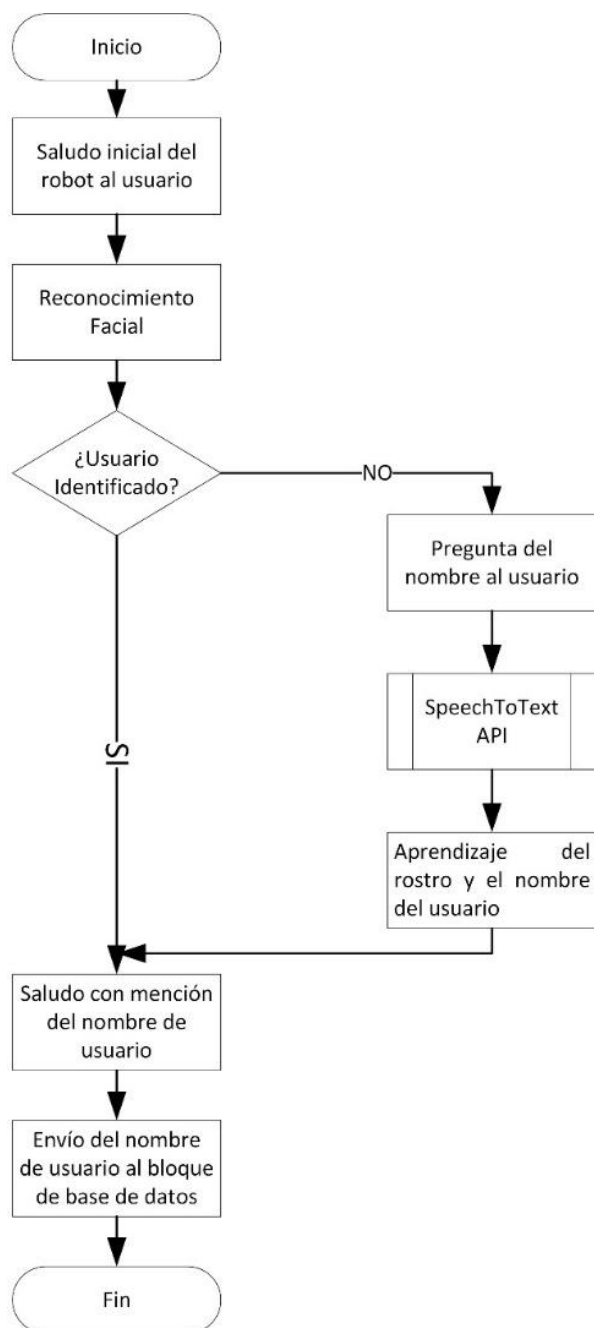
1. Si el rostro del usuario ha sido reconocido, el robot realizará un saludo y el nombre ligado al rostro se enviará a otro bloque de programación para el almacenamiento de puntajes.
2. Si el rostro del usuario no ha sido reconocido, el robot necesitará aprender la cara con el bloque *Learn Face* y asociarlo a su nombre.

Para el aprendizaje del nombre del usuario, se ha implementado un bloque de función, que alberga un algoritmo en Python, que permite la utilización del API *SpeechToText* de Google Cloud con el cual transforma archivos de audio en texto.

Una vez finalizado el proceso de aprendizaje, el robot realizará un saludo y el nombre ligado al rostro se enviará a otro bloque de programación para el almacenamiento de puntajes. En la Figura 49 se muestra el diagrama de flujo de este proceso, y en la Figura 50 se muestra el diagrama de bloques realizado en *Choregraphe*.

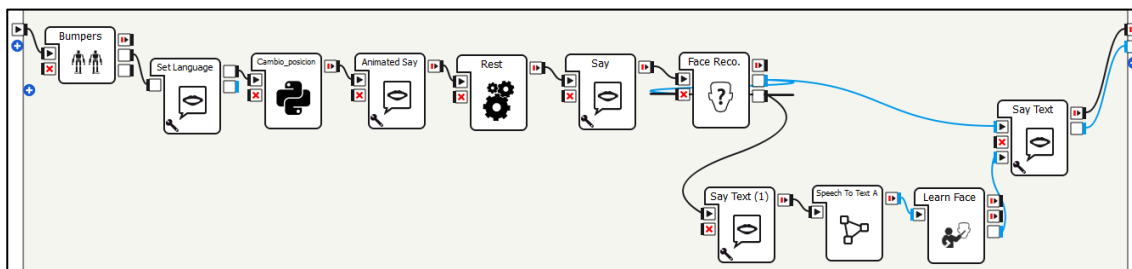
**Figura 49**

*Diagrama de flujo del reconocimiento de usuario para aplicativo de educación*



**Figura 50**

*Diagrama de bloques del reconocimiento del usuario para aplicativo de educación*



### **Nivel 1 – Relación imagen/palabra y práctica de pronunciación. A**

continuación, el usuario podrá empezar con el primer nivel del juego, donde el robot realizará el reconocimiento de imagen de una flashcard y preguntará al usuario si conoce la palabra correspondiente a dicha imagen en inglés. Existen dos posibles escenarios:

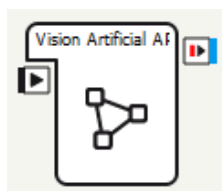
1. Si el usuario identifica al objeto en inglés, el robot solicitará la pronunciación de dicha palabra y validará si es correcta o no, mediante el análisis de audio.
2. Si el usuario no identifica la imagen o no conoce la palabra en inglés, el robot reproducirá dicha palabra con la pronunciación correcta.

Cabe mencionar que, si el usuario ha identificado el objeto en inglés de manera correcta, obtendrá un puntaje positivo y, a su vez, tendrá acceso al segundo nivel del juego. Caso contrario, obtendrá un puntaje de cero y no podrá pasar al segundo nivel, en cambio, deberá repetir la dinámica del primer nivel, hasta responder de manera correcta.

En este nivel, se utiliza el servicio en la nube *Cloud Vision API*, para el cual se ha creado un bloque de programación madre que se puede observar en la Figura 51.

### Figura 51

*Bloque de programación para el uso del API Cloud Vision en el robot NAO*



En éste, la entrada recibe un impulso para iniciar el comportamiento del bloque y la salida entrega el nombre del objeto reconocido en la imagen, como una cadena de caracteres (string). En la Figura 52 se muestra el bloque de programación para el uso del Cloud Vision API para el reconocimiento de imágenes y el código fuente del programa se muestra en Anexo II., Figura 123.

### Figura 52

*Bloque de programación para el uso del Cloud Vision API para reconocimiento de imágenes, en Choregraphe*



En la Figura 53 se muestra el diagrama de flujo del funcionamiento del bloque madre, creado para el uso del API Cloud Vision en el robot NAO. En la Figura 54 se muestra el diagrama de flujo correspondiente al programa del nivel 1. En la sección Anexo II., Figura 124 se puede encontrar el diagrama de bloques realizado en Choregraphe, junto con Python.

**Figura 53**

*Diagrama de flujo del funcionamiento del bloque Cloud Vision API*

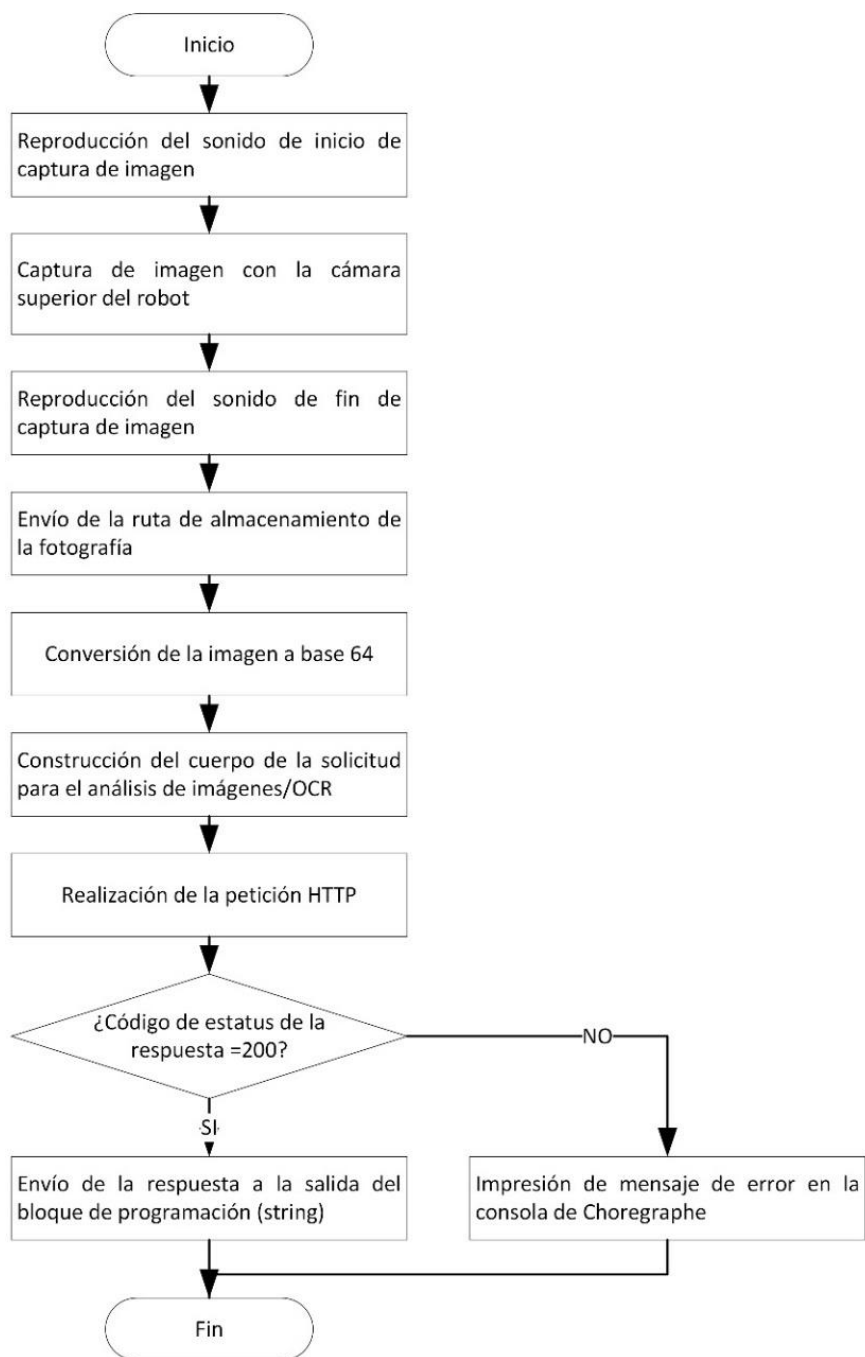
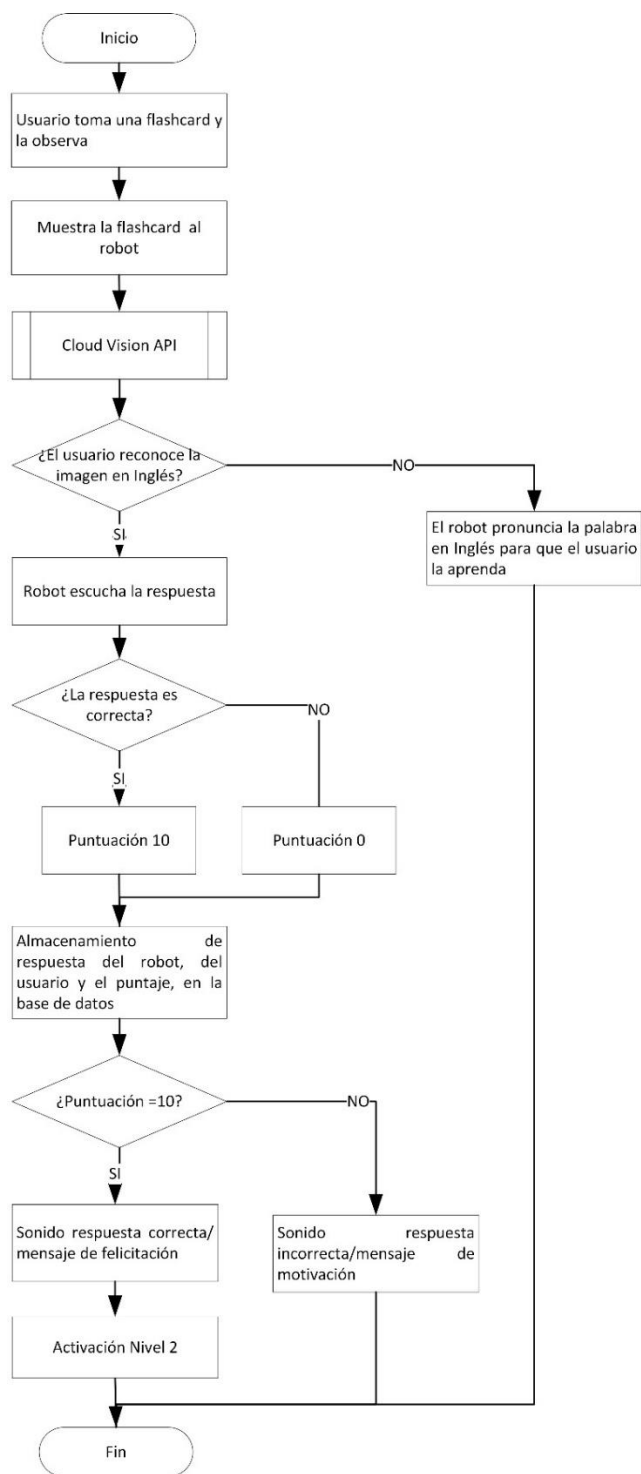




Figura 54

Diagrama de flujo del Nivel 1 del aplicativo de educación



**Nivel 2 – Listening y writing.** Una vez superado el nivel 1, el usuario podrá elegir si continuar al siguiente nivel o seguir practicando en el anterior. Existen dos posibles escenarios:

- Si el usuario acepta continuar, el nivel 2 se activará y comenzará la nueva rutina
- Si el usuario no acepta continuar, el robot pronunciará una frase de ánimo: *Entiendo. Entonces ¡sigamos practicando!*

Ya dentro del nivel 2, el robot procederá a realizar un dictado, letra por letra, de la palabra en inglés donde el usuario deberá copiar en una hoja de papel con esfero o marcador, con letra imprenta y fácilmente legible. Cabe mencionar que, para el correcto desempeño del usuario en el nivel 2, es necesario que el mismo tenga conocimientos previos sobre la pronunciación de las letras del alfabeto.

Una vez finalizado el dictado, el usuario deberá tocar la mano derecha del robot y mostrar la hoja de papel para que éste realice una foto de la palabra, extraiga el texto de la misma y valide si se ha escrito correctamente.

Para la evaluación, tanto el texto obtenido del reconocimiento de la imagen de la tarjeta realizado en el nivel 1, como el texto obtenido al reconocer la palabra en el dictado del nivel 2 ingresan a un bloque de programación, donde se comparan las cadenas de caracteres.

Es importante recordar que, el juego está diseñado para que el usuario solo pueda ingresar a este nivel, si se ha superado el nivel anterior. Por ello, si el usuario ha cometido errores en el dictado, además de obtener un puntaje de cero, tendrá que repetir la ronda con la misma tarjeta, desde el nivel 1.

En este nivel, se utiliza el servicio en la nube *Cloud Vision API*, para el cual se ha insertado el bloque de programación madre mencionado en el nivel anterior, pero con modificaciones que permitan realizar un reconocimiento óptico de caracteres (OCR).

En la Figura 55 se muestra el bloque de programación para el uso del Cloud Vision API para el reconocimiento de imágenes y el código fuente del programa se muestra en Anexo II., Figura 125.

### Figura 55

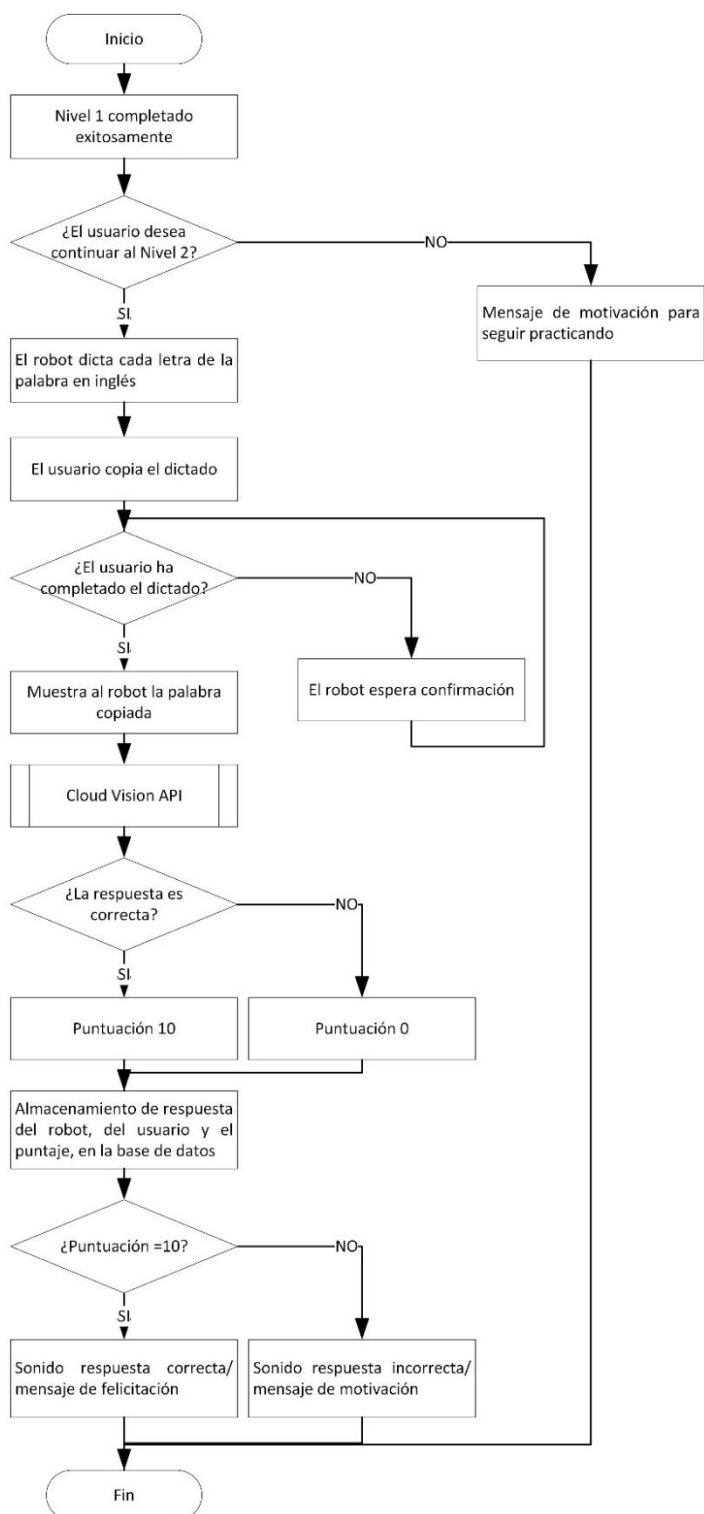
*Bloque de programación para el uso del Cloud Vision API para reconocimiento de imágenes, en Choregraphe*



En la Figura 56 se muestra el diagrama de flujo correspondiente al programa del nivel 2. En la sección Anexo II., Figura 126 se puede encontrar el diagrama de bloques realizado en Choregraphe, junto con Python.

Figura 56

Diagrama de flujo del Nivel 2 del aplicativo de educación

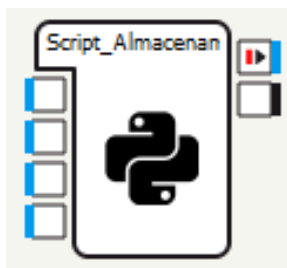


**Almacenamiento en base de datos.** Al final de cada una de las actividades previamente mencionadas, los datos obtenidos del usuario como: nombre, puntajes de cada nivel, respuesta dada por el robot y respuesta obtenida del usuario se almacenarán en una base de datos alojada en la nube llamada Firebase Realtime Database.

La escritura de datos en la base mencionada anteriormente se realiza gracias a la utilización del API REST de Firebase Database, mediante algoritmos de programación en lenguaje Python. En la Figura 57 se muestra el bloque creado. En la sección Anexo II., Figura 127, se puede encontrar el código fuente del programa.

#### **Figura 57**

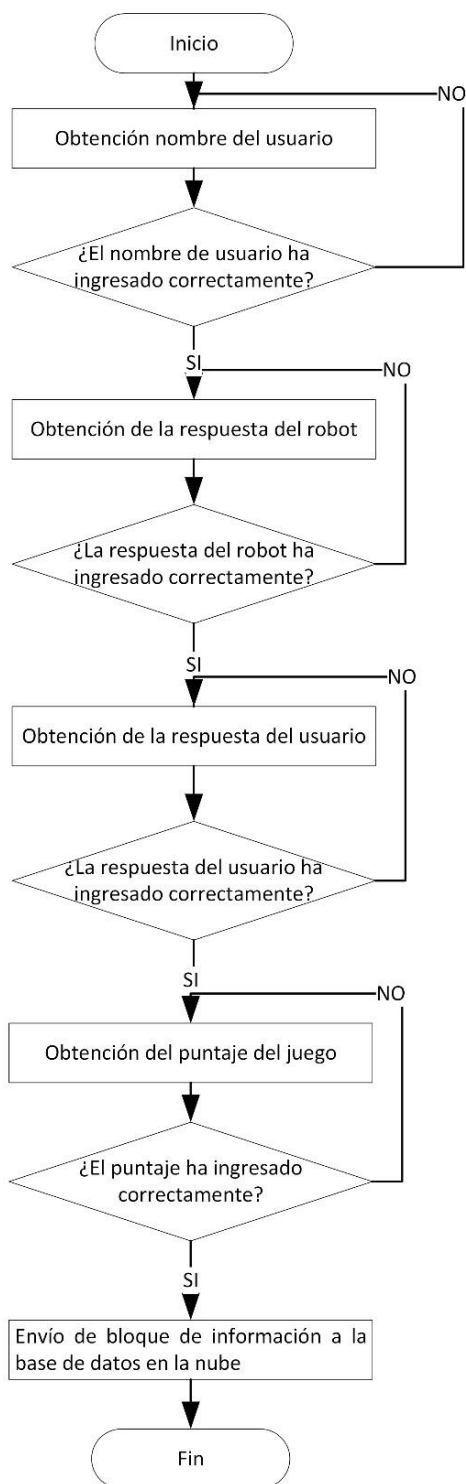
*Bloque de programación para el envío y almacenamiento de datos en la nube, en Python*



Este bloque posee cuatro entradas de tipo STRING, donde ingresan los datos a enviar a la base, en tiempo real, mencionados anteriormente. En la Figura 58 se muestra el diagrama de flujo del proceso de almacenamiento de datos. En la Figura 59 se muestra la información almacenada en la nube.

**Figura 58**

*Diagrama de flujo del almacenamiento de datos en Firebase*



**Figura 59**

*Información almacenada en la base de datos en tiempo real de Firebase*



Es importante aclarar que:

- Todos los datos relacionados al aplicativo para enseñanza de palabras en inglés se guardan en un solo campo llamado *juego1*
- Bajo este objeto se crean hijos que poseen como nombre el correspondiente al usuario, que se obtiene en la primera interacción entre la persona y el robot (reconocimiento de rostro)
- Posteriormente, dentro de cada usuario se almacenará un objeto con ID único, el cual contendrá el nombre, puntajes de cada nivel, respuesta dada por el robot y respuesta obtenida del usuario.
- Solo el desarrollador tendrá acceso a la vista mostrada en la Figura 59 , ya que se encuentra vinculada a la cuenta Gmail del mismo.

Para resolver este último punto, se ha creado una página web offline donde se mostrarán todos los datos almacenados, permitiendo al usuario buscarlos por su

nombre. En la Figura 60 se muestra el diseño de la página para el aplicativo de educación.

## Figura 60

*Página web para la consulta de puntajes por usuario para el aplicativo de educación*

**Aprende Ingles con NAO**

NAO te invita a jugar con cartas que incluyen dibujos de distintas categorías: frutas, verduras, cosas, animales, etc, con el fin de incrementar tu vocabulario de una manera fácil y entretenida.

El juego consta de dos niveles:

- 1. Reconocimiento de objetos:** En el primer nivel deberás interactuar con el robot, mostrándole una carta al azar. Él te preguntará si sabes o no qué es. Si lo sabes, tendrás la oportunidad de decirse lo y él te evaluará. Caso contrario, te lo dirá.
- 2. Dictado:** Una vez termines el primer nivel, de manera correcta, podrás realizar un dictado junto a NAO. Te mencionará, una a una, las letras en inglés, para que tú puedas anotarlo en tu cuaderno. Al finalizar, muéstralo a NAO para su evaluación.

Cada actividad tiene un puntaje, ¡Consultalo Aquí!

**Puntajes**

Haz clic en el siguiente botón para consultar los puntajes obtenidos

Ingrese su nombre:

**Puntaje Total**

**MENU PRINCIPAL**

- INICIO
- APLICATIVO EDUCACION
- APLICATIVO SALUD
- APLICATIVO RESERVACIONES
- ACCEDER

**LINKS DE INFORMACION**

- Documentacion NAO**  
Todo sobre el robot NAO y su software Choregraphe
- Google Cloud**  
Conoce los servicios en la nube que ofrece Google
- Firebase**  
Conoce los servicios para el desarrollo de tus aplicaciones

Esto, con el fin de llevar un registro de avance, por usuario, donde el docente podrá analizar el número de intentos realizados por sesión, el número de respuestas correctas e incorrectas y la complejidad de las palabras aprendidas. En la Figura 61 se muestra la tabla de datos obtenida a partir del nombre de usuario “James”.

La lectura, escritura y eliminación de datos, almacenados en la nube, se realizan mediante scripts en lenguaje de programación JavaScript, utilizando el Firebase Realtime Database API. En la sección Anexo II., Figura 128 se puede encontrar el código fuente del programa.



**Figura 61**

*Tabla de puntajes del usuario de nombre James – aplicativo de educación*

**Puntajes**

Haz clic en el siguiente botón para consultar los puntajes obtenidos

Se ha cargado con éxito.

Ingrese su nombre:

**Puntaje Total**  
30



Usuario	Respuesta Usuario	Respuesta Robot	Puntaje
James	lion	lion	10
James	lion	lion	10
James	lion	lion	10
James	watch	what	0
James	apple	open	0

### ***Aplicativo para cuidados de la salud***

En el área de la salud, se ha creado un aplicativo orientado al agendamiento y recordatorio de eventos, mediante la interacción continua del usuario con el robot NAO utilizando comandos de voz y sensores táctiles, utilizando Google Calendar. En este caso, los eventos han sido relacionados a la ingesta de medicamentos a largo plazo, por ejemplo, para pacientes crónicos que padecen de Diabetes, Enfermedades Cardiovasculares, Osteoporosis Artritis, Hipotiroidismo, Hipertiroidismo, etc.

Sin embargo, los eventos pueden modificarse de acuerdo a las necesidades del usuario, siendo estos de tipo social, empresarial, académicos, culturales, deportivos, etc., y con ello, el desarrollador tiene la oportunidad de definir nuevas acciones a realizar por parte del robot humanoide.

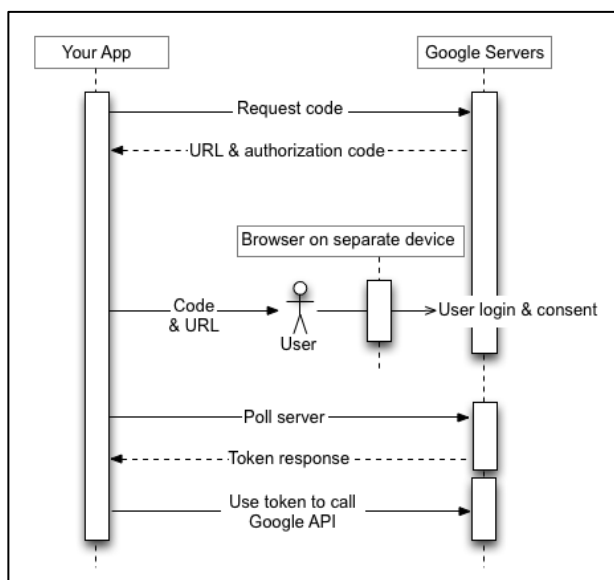
**Autenticación OAuth2.0.** Antes de iniciar, cabe resaltar que, cada solicitud que envía el robot a la API de Google Calendar, debe incluir un token de autorización ya que es necesario el acceso a los datos y cuenta del usuario de Google para crear, consultar y eliminar eventos, entre otras acciones.

Para ello, se utiliza el protocolo OAuth 2.0 cuya implementación dependerá de los siguientes escenarios: aplicaciones de servidor web, aplicaciones instaladas, aplicaciones del lado del cliente (JavaScript), cuentas de servicio y aplicaciones en dispositivos de entrada limitada. Esta última será adaptada y utilizada para la aplicación ejecutada en el robot NAO.

En la Figura 62 se muestra el diagrama de secuencia del proceso de autenticación para aplicaciones en dispositivos de entrada limitada como: consolas de juego, cámaras de video, impresoras, etc.

### **Figura 62**

*Diagrama de secuencia del proceso de autenticación para aplicaciones en dispositivos de entrada limitada*



A continuación, se describe el diagrama mostrado en la Figura 62, incluyendo los bloques de programación creados en Choregraphe, para el uso del protocolo OAuth2.0 en el robot:

### 1. Paso 1: Solicitud de códigos de dispositivo y usuario

La secuencia comienza cuando el usuario presiona el bumper ubicado en el pie izquierdo del robot, activando el primer bloque que, mediante un algoritmo de Python creado, realiza una solicitud HTTP POST al servidor de autorización de Google para obtener una respuesta válida, con información relevante, para continuar con el proceso.

### 2. Paso 2: Manejo de la respuesta del servidor de autorización

Si la solicitud se ha realizado con éxito, el objeto JSON de respuesta contendrá los siguientes parámetros:

- **device\_code**: Un valor que Google asigna, de forma única, para identificar el dispositivo que ejecuta la aplicación que solicita autorización
- **user\_code**: Un valor que identifica a Google los ámbitos a los que la aplicación solicita acceso.
- **verification\_url**: Una URL a la que el usuario debe navegar, en un dispositivo separado, para ingresar el **user\_code** y otorgar o denegar el acceso a su aplicación.
- **interval**: El tiempo, en segundos, que su dispositivo debe esperar entre solicitudes de sondeo.
- **expires\_in**: El período de tiempo, en segundos, que el **device\_code** y el **user\_code** son válidos. Si, en ese tiempo, el usuario no completa el flujo de autorización y su dispositivo tampoco realiza una encuesta para recuperar

información sobre la decisión del usuario, es posible que deba reiniciar este proceso desde el paso 1.

Una vez obtenida esta información, el bloque mostrado en la Figura 63 enviará, a través de sus salidas tipo String, el *user\_code* y el *verification\_url*, los cuales serán procesados para ser transmitidos al usuario en forma auditiva.

Adicional, se enviará el *device\_code* a un segundo bloque para proseguir con el paso 4. El código fuente del programa se muestra en Anexo III., Figura 129.

### **Figura 63**

*Bloque de programación para la realización del paso 1 y 2 de la autenticación OAuth 2.0*

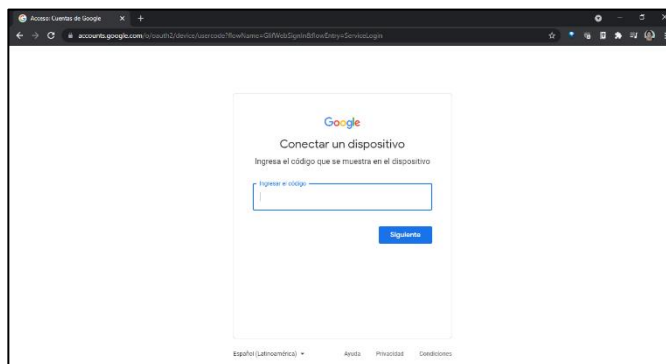


### **3. Paso 3: Entrega del código de usuario y URL de verificación al usuario**

Una vez obtenida la información, el robot solicitará, al usuario, el acceso a la URL de verificación mediante un navegador en un dispositivo externo como un teléfono celular, una tablet o una computadora. La pantalla mostrada se muestra en la Figura 64.

**Figura 64**

*Ingreso a URL de verificación desde dispositivo externo*



A continuación, realizará un dictado, letra por letra, del código de usuario que deberá ingresar en la barra con el texto *“Ingresar el código”*. Hacer clic en el botón *Siguiente* para continuar. Se muestra un ejemplo en la Figura 65.

**Figura 65**

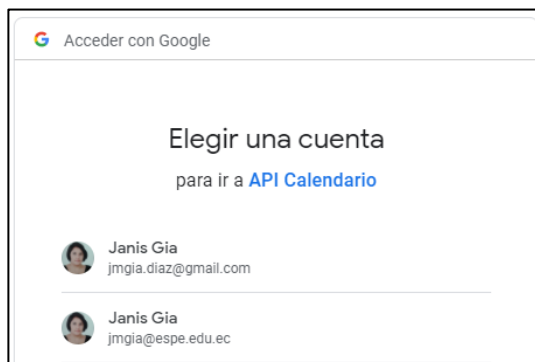
*Ingreso de código de usuario dictado por el robot NAO*



Como se muestra en la Figura 66, si el usuario aún no lo ha hecho, deberá iniciar sesión con la cuenta de Gmail en la que se desea crear los eventos.

**Figura 66**

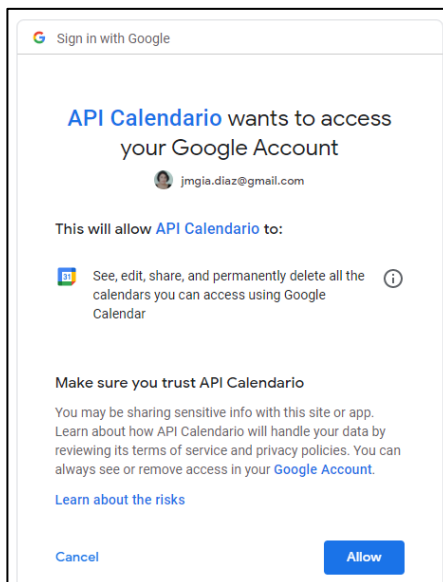
*Inicio sesión en cuenta de Gmail para el acceso a Google Calendar*



El usuario verá la pantalla de consentimiento que se muestra en la Figura 67, donde deberá aceptar o rechazar el acceso de la aplicación a su Google Calendar personal, para ver, editar, compartir y borrar permanentemente los calendarios a los que se pueda acceder.

**Figura 67**

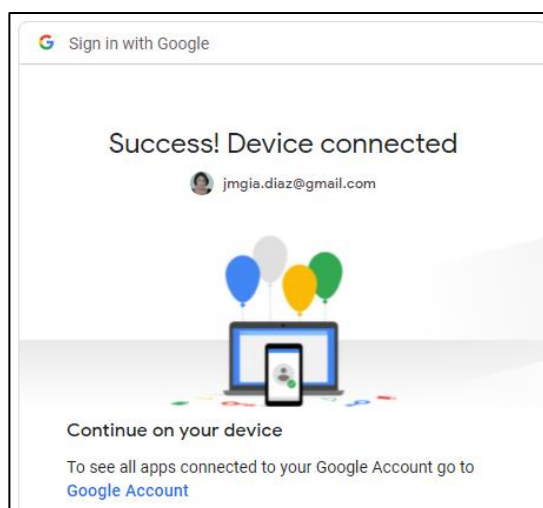
*Pantalla de consentimiento para el acceso de Google Calendar API a la cuenta del usuario*



Si el usuario ha dado autorización al acceso de la cuenta, aparecerá una pantalla como la que se muestra en la Figura 68 donde se confirma que el dispositivo en el que se ejecuta la aplicación se ha conectado con éxito.

### Figura 68

*Pantalla de confirmación de conexión exitosa*

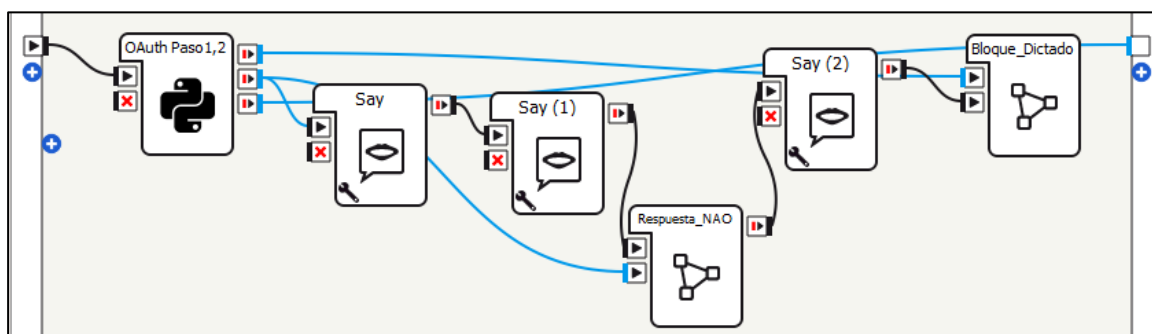


En la Figura 69 se muestra el bloque de programación creado para los tres primeros pasos, previamente realizados.

### Figura 69

*Bloque de programación para la realización de los pasos 1, 2 y 3 de la autenticación*

*OAuth 2.0*



#### 4. Paso 4: Consulta al servidor de autorización de Google desde el robot

Una vez iniciada la sesión y otorgados los permisos, el usuario deberá presionar el bumper del pie derecho del robot para continuar. Con ello, se activará un segundo bloque de programación, que se muestra en la Figura 70, con el fin de realizar una nueva solicitud HTTP POST para consultar al servidor de Google si el usuario ya ha aprobado el acceso.

La respuesta del servidor de Google contiene un token de acceso, que será utilizado para acceder a la API de Google, y un token de actualización que deberá ser almacenado para su uso futuro. Una vez que expire el token de acceso, la aplicación usará el token de actualización para obtener uno nuevo. Ya finalizado el proceso de autenticación, el robot reproducirá un sonido que indique al usuario que se ha completado correctamente.

#### Figura 70

*Bloque de programación para la realización del paso 4 de la autenticación OAuth 2.0*



La Figura 70 muestra el bloque de programación para la realización del paso 4 de la autenticación OAuth 2.0. Posee dos entradas: la primera tipo pulso para iniciar el funcionamiento del bloque y la segunda tipo string por donde ingresa el *device\_code* obtenido en el paso 2.

Adicional posee dos salidas: una, tipo pulso, que envía una señal una vez terminada la ejecución del bloque y otra, tipo string, para el envío del token de acceso



hacia el bloque de ejecución de la API. El código fuente del programa se muestra en Anexo III., Figura 130

**Agendamiento de Eventos.** Ya generado el token de acceso, se puede utilizar libremente el Google Calendar API durante el tiempo de validez del mismo. Cabe recordar que, si este llegara a expirar, será necesario utilizar el token de actualización para generar uno nuevo.

Para el agendamiento de eventos, se ha creado un bloque de programación que permite, mediante el robot humanoide NAO, crear eventos con información específica de: paciente, medicamento y hora de ingesta, que posteriormente se verá reflejada en la aplicación Google Calendar ligada al correo electrónico de la cuenta previamente utilizada para la autenticación.

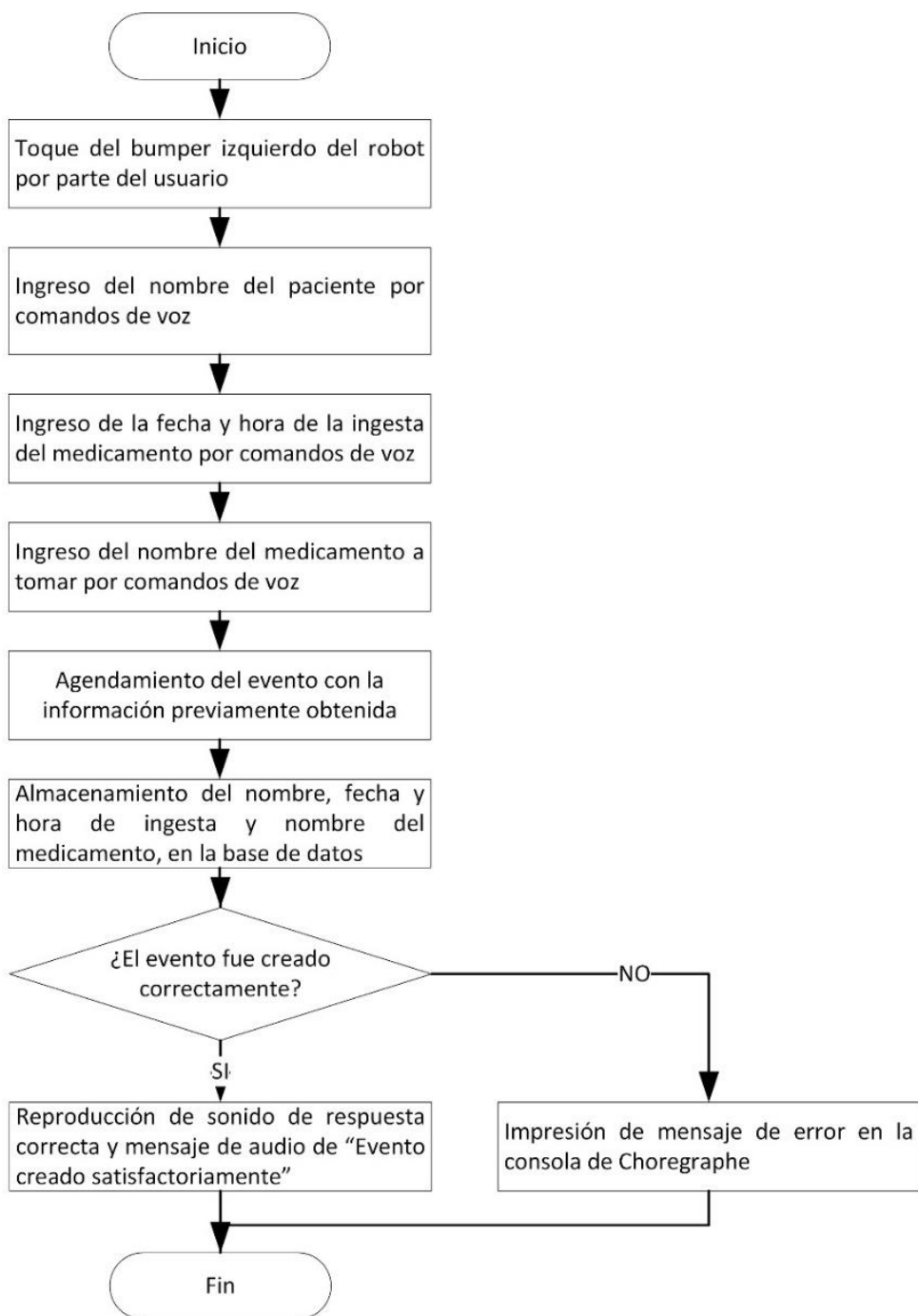
Los datos serán ingresados mediante comandos de voz, como respuestas a cada una de las preguntas que el robot realiza en el siguiente orden: nombre del paciente, hora de ingesta del medicamento, nombre del medicamento.

Adicional, los datos del evento, que permiten su creación, serán almacenados en una base de datos en línea para que, la persona encargada de la creación de los eventos, pueda consultarlos mediante una página web previamente creada.

En la Figura 71 se muestra el diagrama de flujo de este proceso. En la sección Anexo III., Figura 131 se puede encontrar el código fuente del programa y en la Figura 132 el diagrama de bloques realizado en Choregraphe, en conjunto con Python.

**Figura 71**

*Diagrama de flujo del funcionamiento del bloque de agendamiento de eventos*



**Consulta de Eventos.** Para la consulta de eventos preprogramados en Google Calendar, se realiza un bloque de programación similar al de agendamiento de eventos, donde se replicará la fase de autenticación. Ya generado el token de acceso, se puede utilizar libremente el Google Calendar API durante el tiempo de validez del mismo. Cabe recordar que, si este llegara a expirar, será necesario utilizar el token de actualización para generar uno nuevo.

Para esta función, solo es necesaria la activación de bumpers/sensores para ejecutar los bloques de programación en Choregraphe que se encargan de hacer una búsqueda de todos los eventos que coincidan con la fecha y hora actual de activación del mismo.

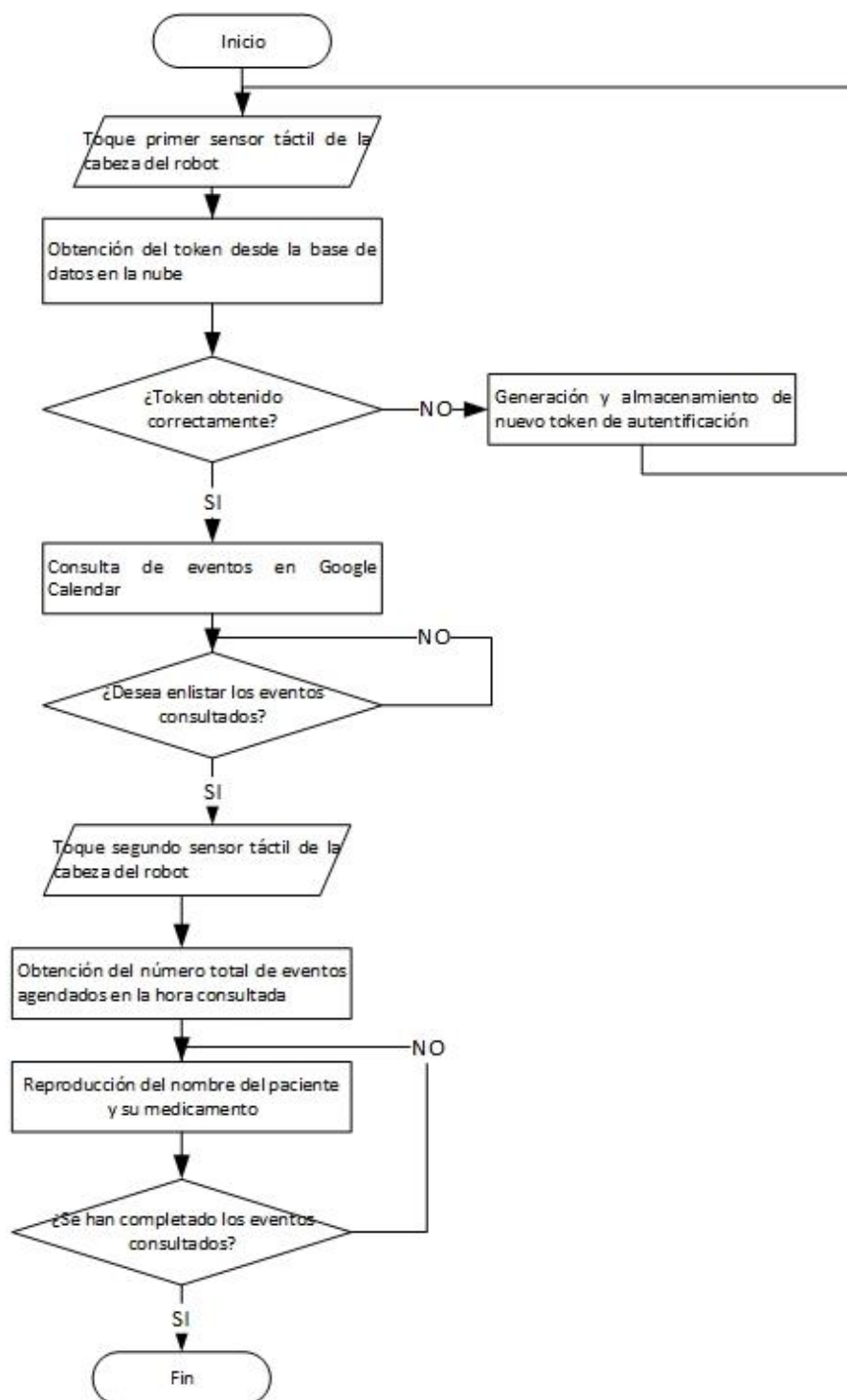
Una vez finalizada la recuperación de eventos, el robot esperará la confirmación del usuario para, a continuación, indicar el número total de eventos agendados y realizar un dictado de los nombres de pacientes con su respectivo medicamento.

Así, por ejemplo, si se activa la aplicación el día 10 de Julio a las 14:30, el robot realizará una búsqueda exhaustiva de todos los eventos que se encuentren agendados el día 10 de Julio, desde las 14:00 hasta las 14:59:59.

En la Figura 72 se muestra el diagrama de flujo de este proceso. En la sección Anexo III, Figura 133 se puede encontrar el código fuente del programa y en la Figura 134, el diagrama de bloques realizado en Choregraphe, en conjunto con Python.

Figura 72

Diagrama de flujo del funcionamiento del bloque de consulta de eventos



### ***Aplicativo para atención al cliente***

Para el último aplicativo, encaminado a la atención al cliente, se ha planteado un escenario donde el robot humanoide NAO sea utilizado como un facilitador de servicios para la reserva de maquinaria mediante comandos de voz. Por ello, es necesario que la comunicación oral entre el usuario y el robot sea más fluida y natural, que permita la retroalimentación continua en cada interacción y facilite la extracción de datos importantes provenientes de las respuestas del usuario.

Estas funcionalidades logran conseguirse gracias al diseño y uso de un chatbot en línea, desarrollado en Dialogflow, una herramienta de creación de chatbots capaz de entender el lenguaje natural y proveer una infraestructura para recrear conversaciones y construir diálogos. Adicional, éste admite la lectura/escritura de base de datos en tiempo real creadas en Firebase, un backend-as-a-Service (Baas) que incluye servicios para crear, probar y administrar aplicaciones.

Así, el presente aplicativo tiene como objetivo realizar la reserva de robots fijos y móviles disponibles en el Laboratorio de Robótica de la Universidad de las Fuerzas Armadas - ESPE, utilizando el robot humanoide NAO, mediante la obtención de información del usuario como: nombre del solicitante, tipo y número de robot, horario y cantidad de personas a ingresar al laboratorio. Al mismo tiempo, estos datos serán almacenados en una base de datos en la nube para ser accesibles mediante página web.

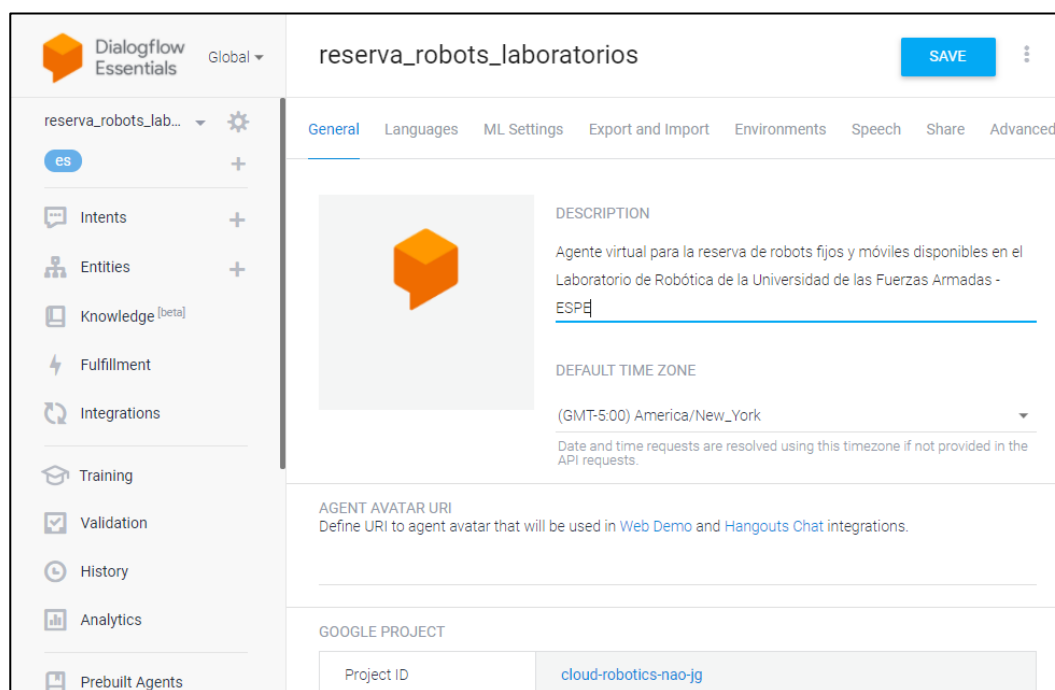
**Creación Chatbot en Dialogflow.** Para la creación del chatbot para reserva de maquinaria, es necesario revisar ciertos conceptos sobre Dialogflow:

## - Agente

Un agente es un módulo de comprensión de lenguaje natural que se encarga de manejar conversaciones con los usuarios finales, mediante un entrenamiento continuo y la definición de reglas que permitan una interacción fluida sobre un tema específico. Para este aplicativo, se ha creado un nuevo agente en Dialogflow Essentials llamado *reserva\_robots\_laboratorios* como se muestra en la Figura 73.

### Figura 73

#### Creación de agente en Dialogflow



## - Intents

Un intent refiere a los intentos de respuestas que pueden provenir del usuario final en cada turno de conversación y en los cuales se definen las posibles réplicas del chatbot, los parámetros que pueden extraerse del usuario, las acciones a realizar y las

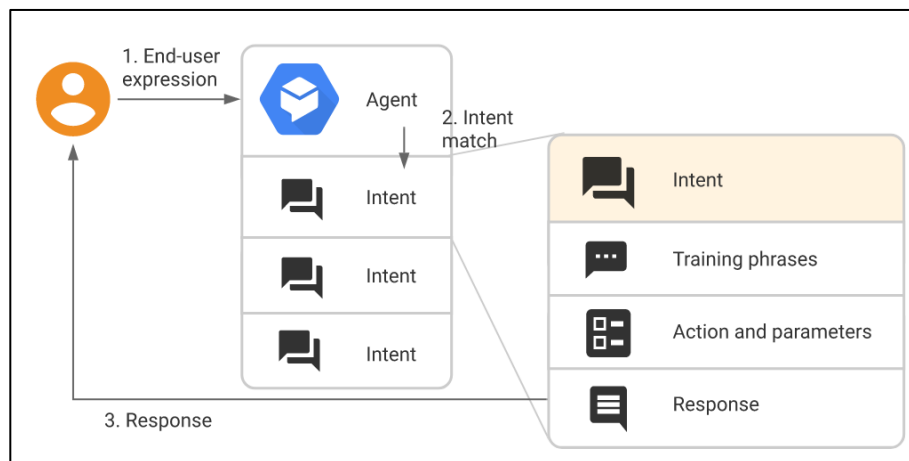
frases de entrenamiento que ayuden al agente virtual a clasificar una expresión en el intent adecuado.

Cabe mencionar que, al crear un nuevo agente, dos intents son creados por defecto: default welcome intent que es el primer mensaje al abrir el chatbot (usualmente un saludo) y default fallback intent, que aparece cuando el chatbot no ha logrado coincidir la expresión del usuario con ningún intent disponible.

Estos pueden ser editados y configurados para ajustarse a las necesidades de la aplicación. En la Figura 74 se muestra el flujo básico para la coincidencia de intents y la respuesta al usuario final.

### Figura 74

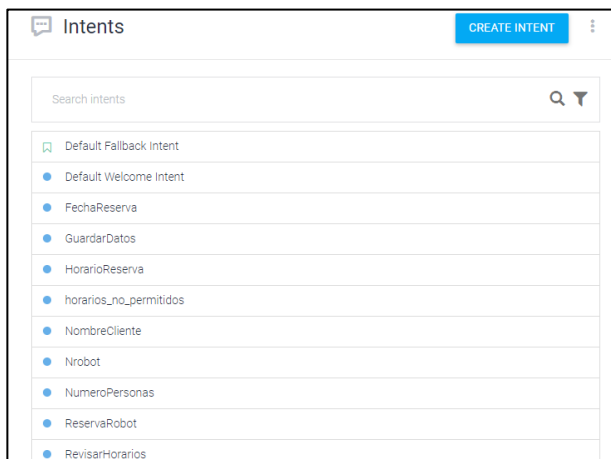
*Flujo básico para la coincidencia de intents*



Dentro del agente *reserva\_robots\_laboratorios*, se han agregado once intents que estructuran la conversación para la reserva de robots fijos y móviles, mediante los cuales se obtienen los siguientes datos: nombre del usuario, número de personas, fecha, hora, tipo y número de robot a reservar. En la Figura 75 se muestran los intents mencionados.

## Figura 75

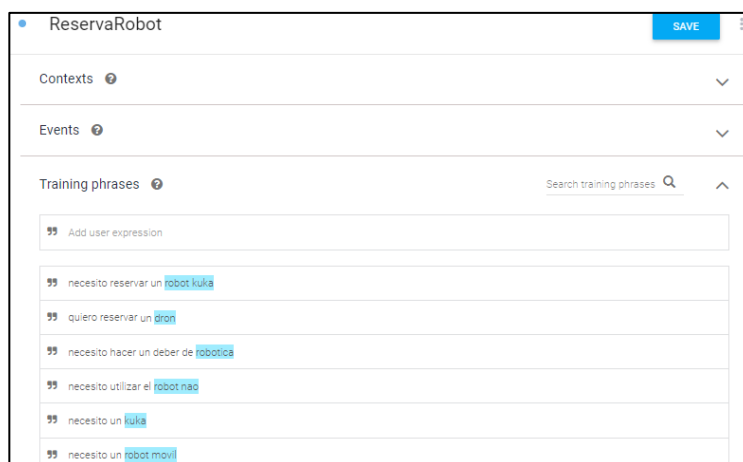
*Intents creados para el chatbot en Dialogflow*



En la Figura 76 se muestra el intent *ReservaRobot*, donde una de las frases de entrenamiento es: *“necesito un kuka”*. Es importante colocar varias frases de entrenamiento, pero no todas las posibilidades de respuesta por parte del usuario. Esto, debido a que, con cada interacción, el chatbot se enriquece con más frases que deben ser validadas por el desarrollador, en el apartado *Training* de Dialogflow.

## Figura 76

*Frases de entrenamiento del intent ReservaRobot*

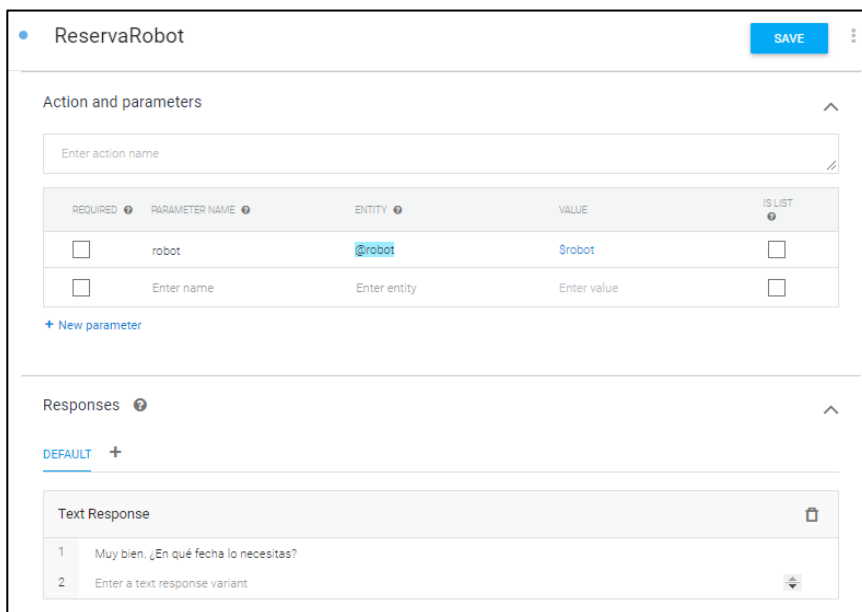




En la Figura 77 se muestra el parámetro *tipo de robot* que se desea obtener del usuario y las posibles respuestas que serán dadas por el chatbot.

### Figura 77

*Parámetro tipo de robot del intent ReservaRobot*



The screenshot shows the configuration interface for the 'ReservaRobot' intent. It is divided into two main sections: 'Action and parameters' and 'Responses'.

**Action and parameters:** This section contains a text input field for 'Enter action name'. Below it is a table with columns: 'REQUIRED', 'PARAMETER NAME', 'ENTITY', 'VALUE', and 'IS LIST'. The first row shows a parameter named 'robot' with the entity '@robot' and value '\$robot'. The second row shows a parameter named 'Enter name' with the entity 'Enter entity' and value 'Enter value'. A '+ New parameter' link is located below the table.

REQUIRED	PARAMETER NAME	ENTITY	VALUE	IS LIST
<input type="checkbox"/>	robot	@robot	\$robot	<input type="checkbox"/>
<input type="checkbox"/>	Enter name	Enter entity	Enter value	<input type="checkbox"/>

**Responses:** This section is titled 'Responses' and has a 'DEFAULT' tab selected. It contains a 'Text Response' section with two variants:

- 1 Muy bien. ¿En qué fecha lo necesitas?
- 2 Enter a text response variant

#### - Entidades

Las entidades son etiquetas o categorías que permiten identificar y extraer datos relevantes de las entradas del usuario. Existen entidades predefinidas para obtener datos como: fechas, horas, colores, números, etc., sin embargo, el desarrollador puede crear sus propias entidades, si así lo requiere, en las cuales deberá agregar un conjunto de palabras o frases que se consideren equivalentes entre sí, llamadas entradas de entidad.

Como se mencionó anteriormente, el propósito del chatbot es extraer datos de una conversación con el usuario para la reserva de maquinaria, por lo que es necesario utilizar entidades para que éste pueda reconocer y obtener la información que será

posteriormente almacenada en una base de datos en la nube. En la Tabla 7 se muestran las entidades correspondientes a los datos obtenidos.

**Tabla 7**

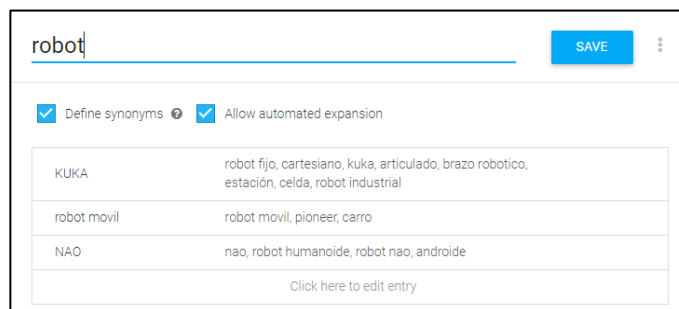
*Nombre de entidades y tipo de datos a obtener en el chatbot*

Datos a obtener	Entidad
Nombre del usuario	@sys.given-name
Apellido del usuario	@sys.last-name
Tipo de robot	@robot
Número de robot	@sys.number
Fecha de reserva	@sys.date
Hora de reserva	@sys.time

En la Figura 78 se muestra una de las entidades creadas para obtener el tipo de robot que se desea reservar. Cabe mencionar que, las palabras mostradas a la derecha, son todas aquellas que el usuario puede utilizar y que serán relacionadas con las que se encuentran a la izquierda de la imagen. Por ejemplo, si se desea un robot NAO, el usuario podrá solicitarlo también como robot humanoide o simplemente NAO.

**Figura 78**

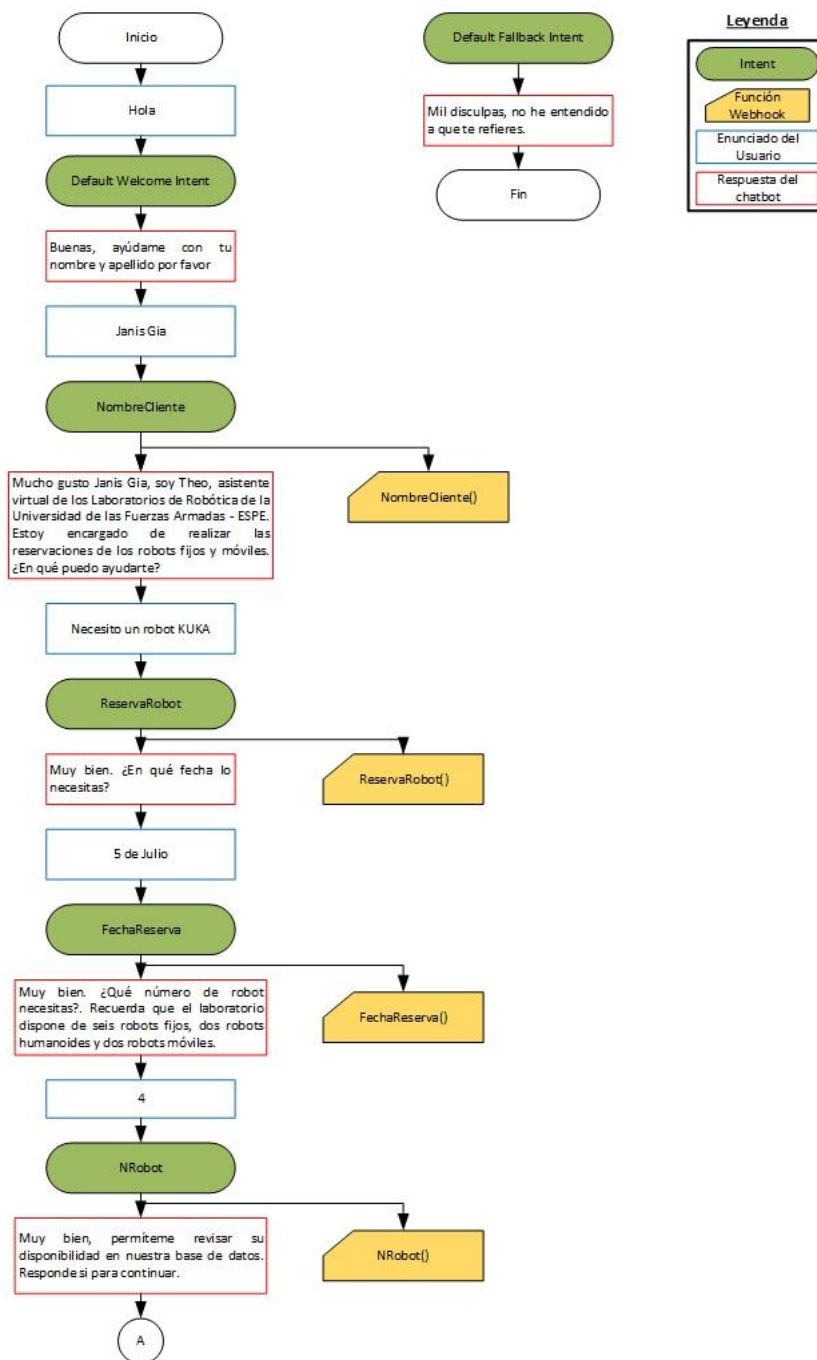
*Creación de entidad robot en Dialogflow*



Una vez aclarados los conceptos, en la Figura 79 se muestra el diagrama de flujo perteneciente al chatbot de reserva de robots creado, así como su respectiva leyenda.

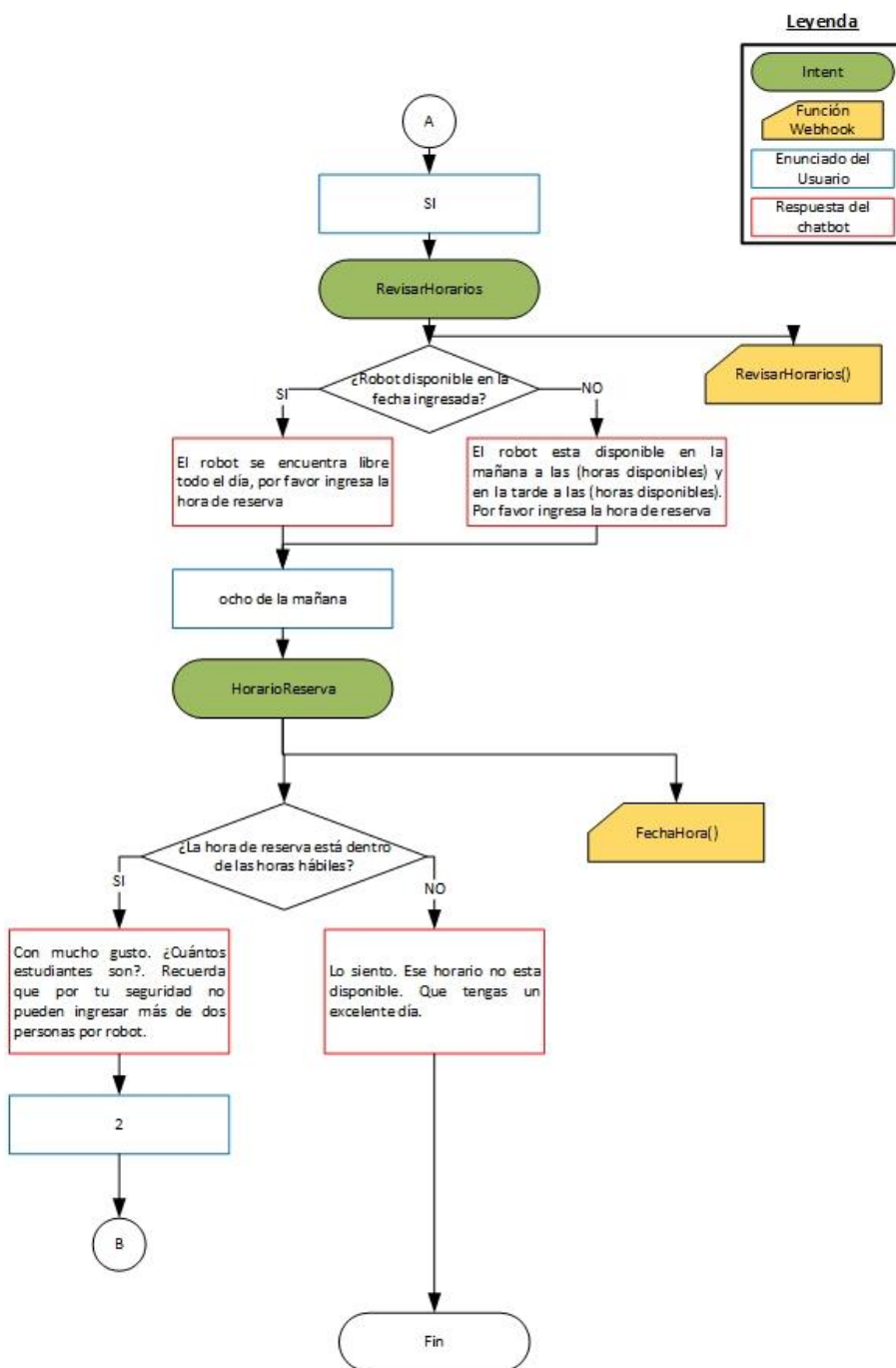
Figura 79

Diagrama de flujo del funcionamiento del chatbot para la reserva de robots fijos y móviles



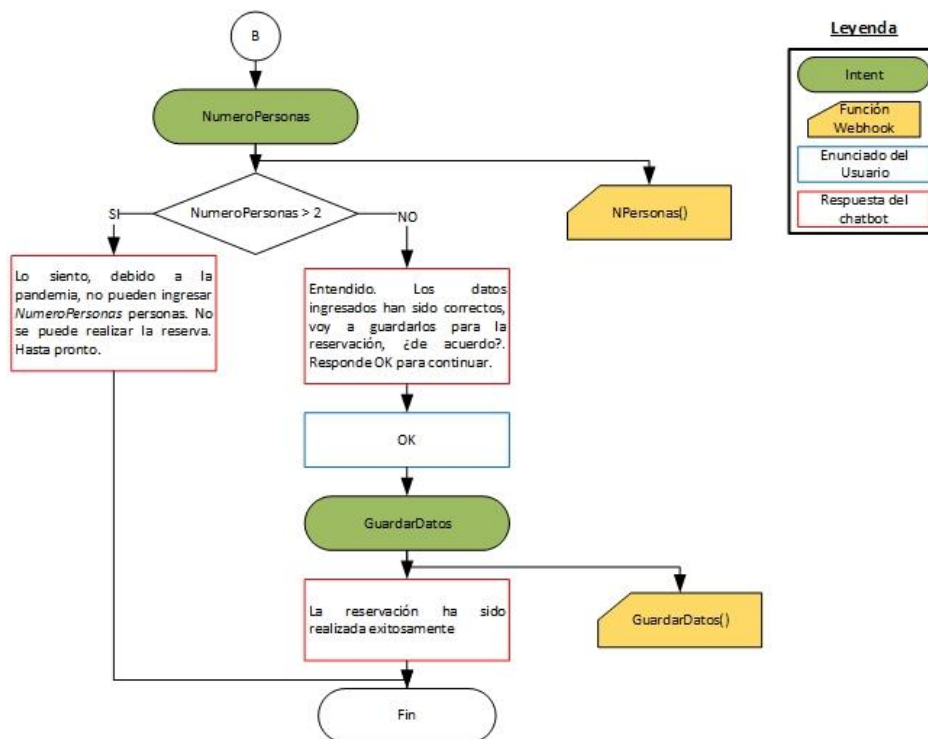
**Figura 80**

*Diagrama de flujo del funcionamiento del chatbot para la reserva de robots fijos y móviles*



**Figura 81**

*Diagrama de flujo del funcionamiento del chatbot para la reserva de robots fijos y móviles*



**Almacenamiento de Datos en Firebase.** Para la reserva de robots fijos y móviles, se ha planteado el almacenamiento de los datos obtenidos mediante el chatbot, en una base de datos en tiempo real que se encuentra albergada en la nube, dentro de la plataforma llamada Firebase. Esto puede realizarse de dos maneras diferentes: desde el robot como se describió en aplicaciones anteriores o desde el mismo Dialogflow, mediante programación.

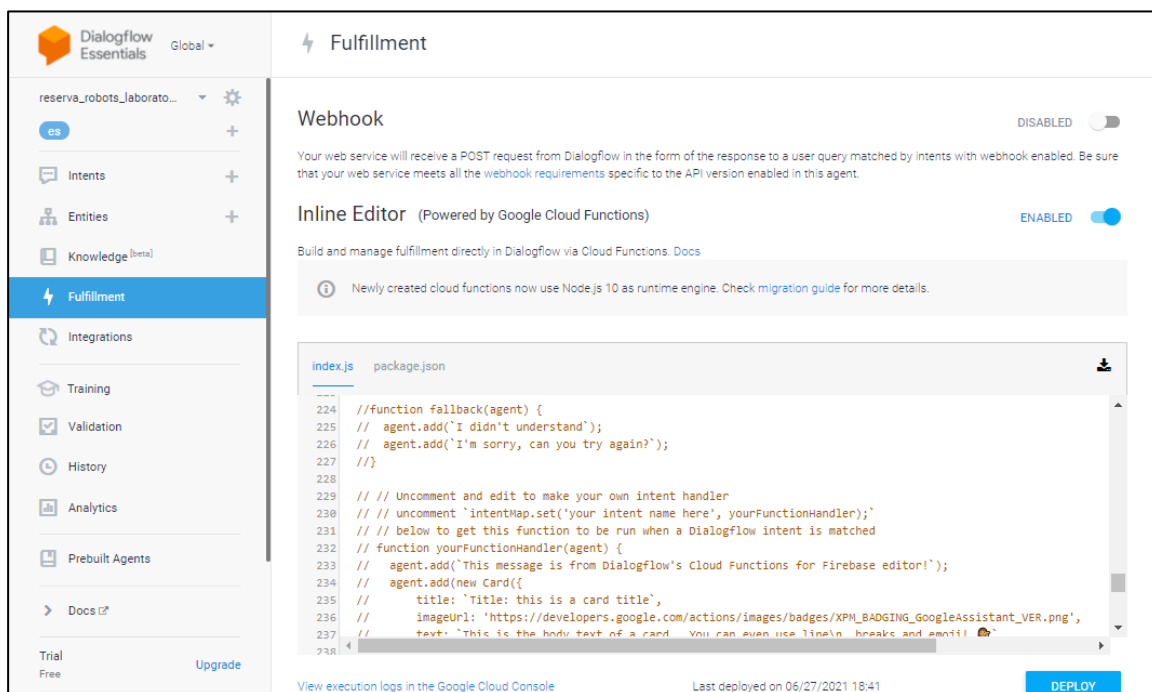
En este caso se realiza la lectura/escritura de datos dentro de Dialogflow, mediante fulfillment, código que se implementa como un webhook que permite, dentro de un intent, definir respuestas dinámicas o realizar el llamado a un servicio externo. Cabe mencionar que, se pueden utilizar dos tipos de “Fulfillment”: uno externo en el que

se tiene un servicio web que cumpla con la API de Fulfillment de Dialogflow, llamado Webhook, y otro utilizando el editor inline de Dialogflow, que desplegará un Cloud Function en node.js.

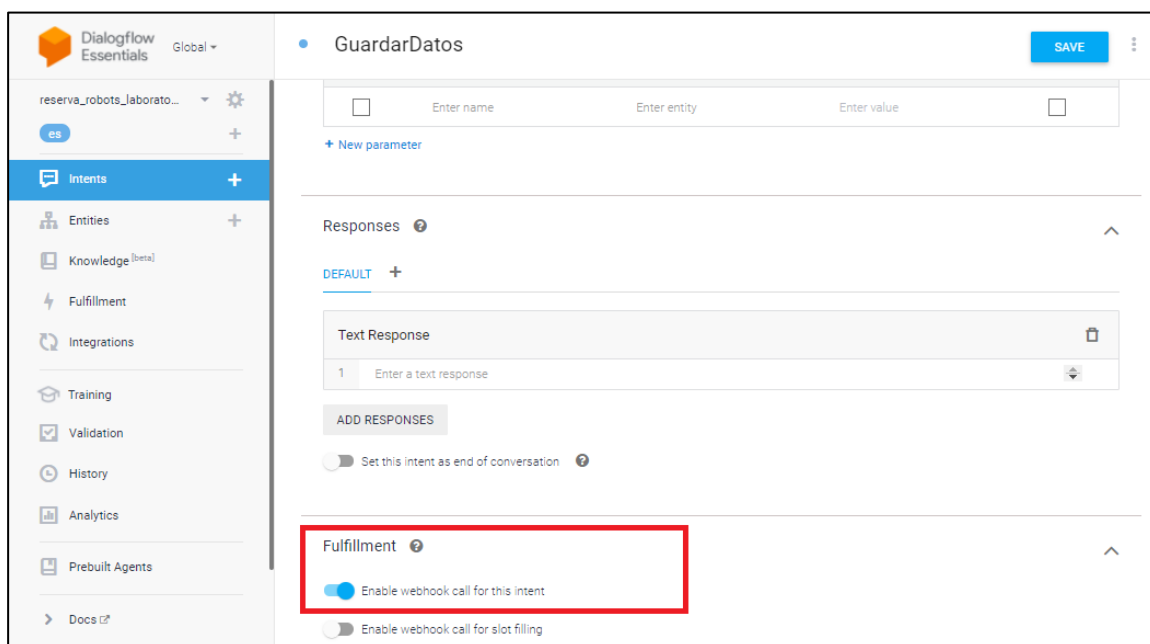
Al ser ésta una aplicación sencilla, es posible utilizar el editor Inline, propio de Dialogflow para realizar el llamado a las funciones de Firebase. Se puede acceder a esta ventana haciendo clic en el menú lateral izquierdo de la consola, en la opción *Fulfillment*. Para utilizarlo, es necesario activar la opción *Inline Editor* como se muestra en la Figura 82

**Figura 82**

### *Inline Editor en Dialogflow*



Adicional, es necesario activar la opción *Enable webhook call for this intent* en cada uno de los intents de los que se desea obtener datos, generar respuestas dinámicas o hacer llamados a los servicios en la nube. Esto se muestra en la Figura 83.

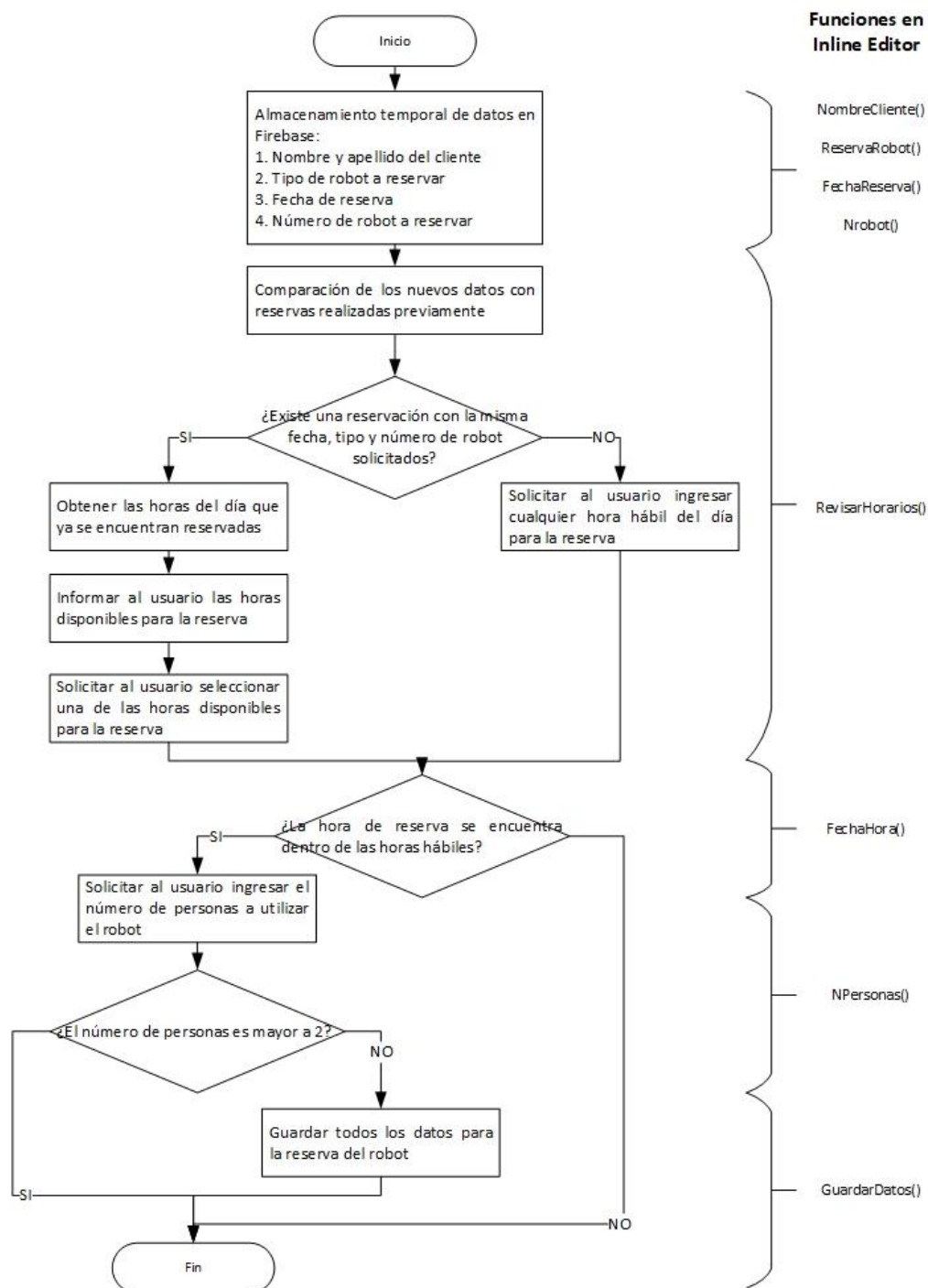
**Figura 83***Habilitación de Webhook para el intent GuardarDatos*

Como se observó en el diagrama de flujo de la Figura 79, se han implementado bloques de función en cada uno de los intents creados, esto es debido al proceso de almacenamiento y validación de datos que se debe realizar para generar correctamente la reservación de robots. En la Figura 84, se detallan los pasos de este procedimiento, así como el nombre de las funciones implementadas en el Inline Editor.

Adicional, cabe mencionar que, los datos para la reserva de los robots serán almacenados de manera definitiva si y solo si la información ha sido validada correctamente. En el caso de que la fecha, hora, tipo y número de robot ya estén ocupados, que la hora ingresada se encuentre fuera de las horas hábiles y que el número de personas exceda de dos, la reserva no será realizada, los datos no serán almacenados de manera permanente en la base de datos y el usuario deberá repetir el proceso de reserva desde el principio.

Figura 84

Diagrama de flujo del almacenamiento de datos en Firebase y funciones creadas en Inline Editor



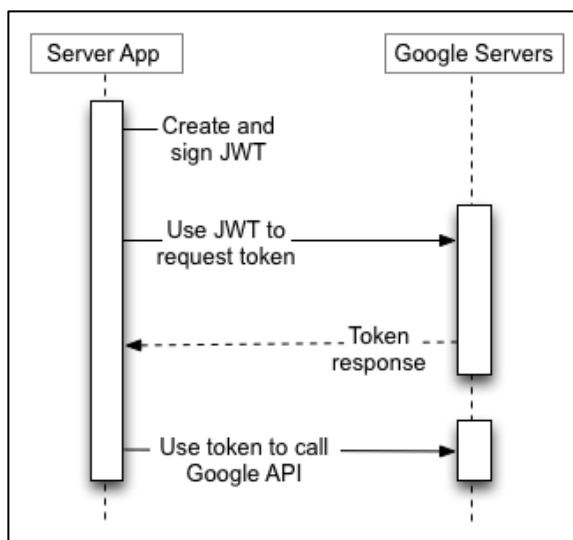


**Autorización de cuenta de servicio sin OAuth.** Una vez probado y depurado tanto el chatbot en Dialogflow, como el almacenamiento de datos en Firebase, se procede a realizar su implementación en Choregraphe, mediante la utilización del Dialogflow API, para lo cual es necesario realizar un proceso de autenticación ya que, a la aplicación cliente, se le debe otorgar permisos para el acceso a los recursos solicitados.

En este caso, se aplica un método de autenticación utilizando cuentas de servicio debido a que, proporcionan credenciales para aplicaciones, permitiendo las llamadas de las API de Google desde la cuenta creada, evitando que los usuarios finales se involucren directamente. En la Figura 85 se muestra el diagrama de secuencia del proceso de autenticación para aplicaciones de servidor a servidor.

### Figura 85

*Diagrama de secuencia del proceso de autenticación para aplicaciones de servidor a servidor*



En este proceso, existen cuatro pasos a seguir:

1. La aplicación crea y firma un JWT (JSON Web Token), que es una cadena de texto que tiene tres partes (un encabezado, un conjunto de reclamos y una firma.) codificadas en Base64, cada una de ellas separadas por un punto.
2. Una vez formado el JWT, se envía una solicitud HTTP POST con esta información al servidor de autorización de Google para obtener un token de acceso.
3. Si el JWT y la solicitud del token de acceso están formados correctamente y la cuenta de servicio tiene permiso para realizar la operación, la respuesta JSON del servidor de autorización incluye un token de acceso.
4. Una vez que la aplicación obtiene el token de acceso, puede usarlo para realizar llamadas a una API de Google en nombre de la cuenta de servicio previamente creada.

Sin embargo, para algunas API de Google, la aplicación puede realizar llamadas API autorizadas utilizando un JWT firmado directamente como un token de portador, en lugar de un token de acceso OAuth 2.0., es decir, se pueden omitir los pasos 2 y 3 anteriormente mencionados. Para ello, se realizan dos únicos pasos que se detallan a continuación:

1. Crear una cuenta de servicio y conservar el archivo JSON que se obtiene cuando se crea el KEY de la cuenta.
2. Usando cualquier biblioteca JWT estándar, crear y firmar un JWT utilizando la información obtenida del archivo JSON previamente mencionado.

En el caso del lenguaje de programación Python, para la ejecución del segundo paso, se sugiere utilizar la librería *PyJWT* que permite codificar y decodificar JSON Web Tokens (JWT). En la Figura 86 se muestra el código a implementar.

**Figura 86**

*Código fuente del programa para la generación del token JWT en Python*

```

iat = time.time()
exp = iat + 3600
payload = {'iss': '123456-compute@developer.gserviceaccount.com',
          'sub': '123456-compute@developer.gserviceaccount.com',
          'aud': 'https://firestore.googleapis.com/',
          'iat': iat,
          'exp': exp}
additional_headers = {'kid': PRIVATE_KEY_ID_FROM_JSON}
signed_jwt = jwt.encode(payload, PRIVATE_KEY_FROM_JSON,
headers=additional_headers, algorithm='RS256')

```

Donde:

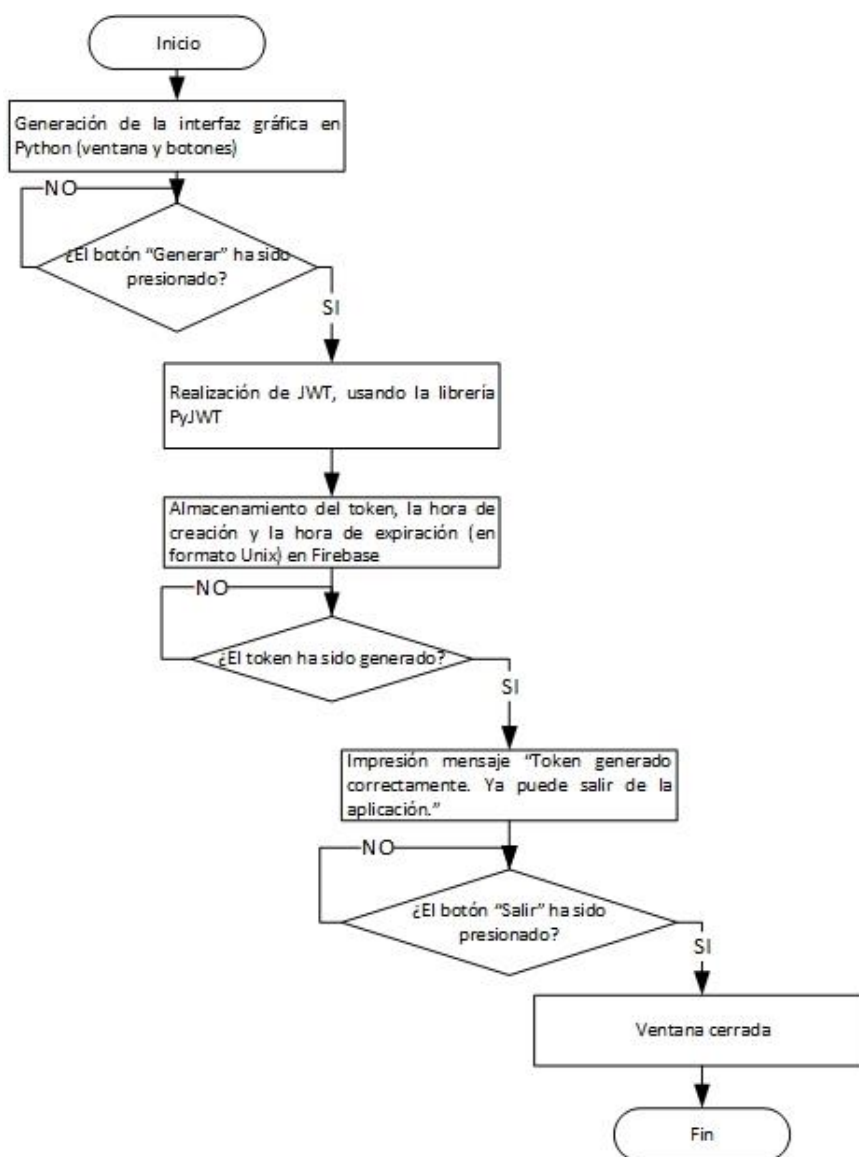
- **kid:** es el ID de la clave privada de la cuenta de servicio. Puede encontrar este valor en el campo *private\_key\_id* del archivo JSON de la cuenta de servicio.
- **iss y sub:** es la dirección de correo electrónico de la cuenta de servicio. Puede encontrar este valor en el campo *client\_email* del archivo JSON de la cuenta de servicio.
- **aud:** es el endpoint de la API, es decir, es la URL que responde a una petición. En el caso de Dialogflow sería: 'https://dialogflow.googleapis.com/'
- **iat:** es la hora actual de Unix.
- **exp:** es la hora actual exactamente 3600 segundos después, cuando el JWT caducará.

Cabe mencionar que, la librería PyJWT solo está disponible para versiones de Python mayores a 3.6, por lo cual no es posible utilizarla en el robot humanoide NAO, debido a que éste maneja la versión de Python 2.7. Por ello, se diseña un aplicativo, ejecutable en computadora, para la generación del token de acceso JWT utilizando

Python 3.9. Posteriormente, el JWT se almacena en la base de datos en la nube, mediante el API de Firebase, para que, por este medio, el NAO pueda consultarlo y utilizarlo en la llamada de la API de Dialogflow. En la Figura 87 se muestra el diagrama de flujo del aplicativo previamente mencionado. El código fuente del programa se muestra en Anexo IV., Figura 135.

### Figura 87

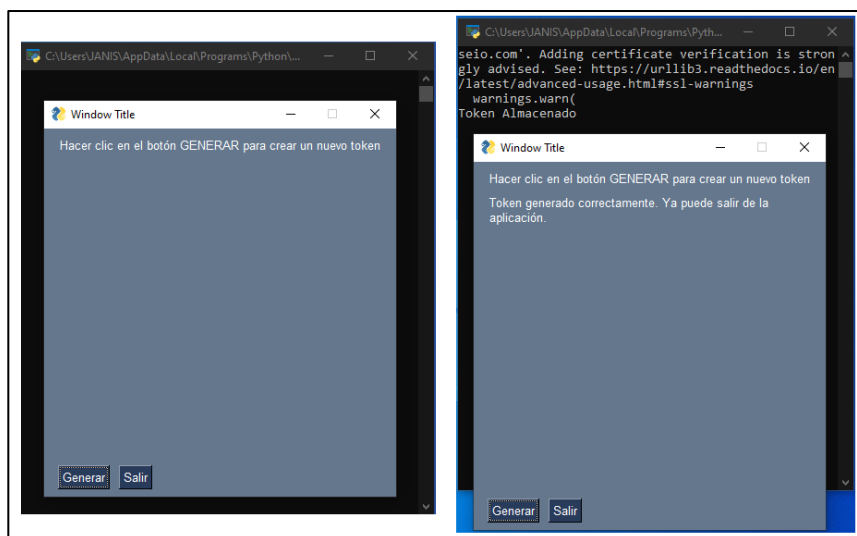
*Diagrama de flujo del aplicativo para la generación del token JWT en Python*



En la Figura 88 se muestra el funcionamiento de la aplicación de generación de JWT. Es importante recordar que esta aplicación debe utilizarse cada vez que el robot indique que el token ha expirado.

### Figura 88

*Funcionamiento de la aplicación de generación de JWT*



**Implementación en Choregraphe.** Para la implementación del chatbot en el robot humanoide NAO, se crean varios bloques de programación en Python.

El primer bloque corresponde a la lectura de la base de datos en la nube, para la obtención del token de acceso, que será utilizado en cada llamada al Dialogflow API, así como la validación del tiempo de duración del token (de una hora) para solicitar al usuario que genere uno nuevo, a través de la aplicación previamente mencionada, cuando el actual expira. En la Figura 89 se muestra el bloque creado. El código fuente del programa se muestra en Anexo IV., Figura 136.

**Figura 89**

*Bloque de programación para la lectura del token de la base de datos en la nube*



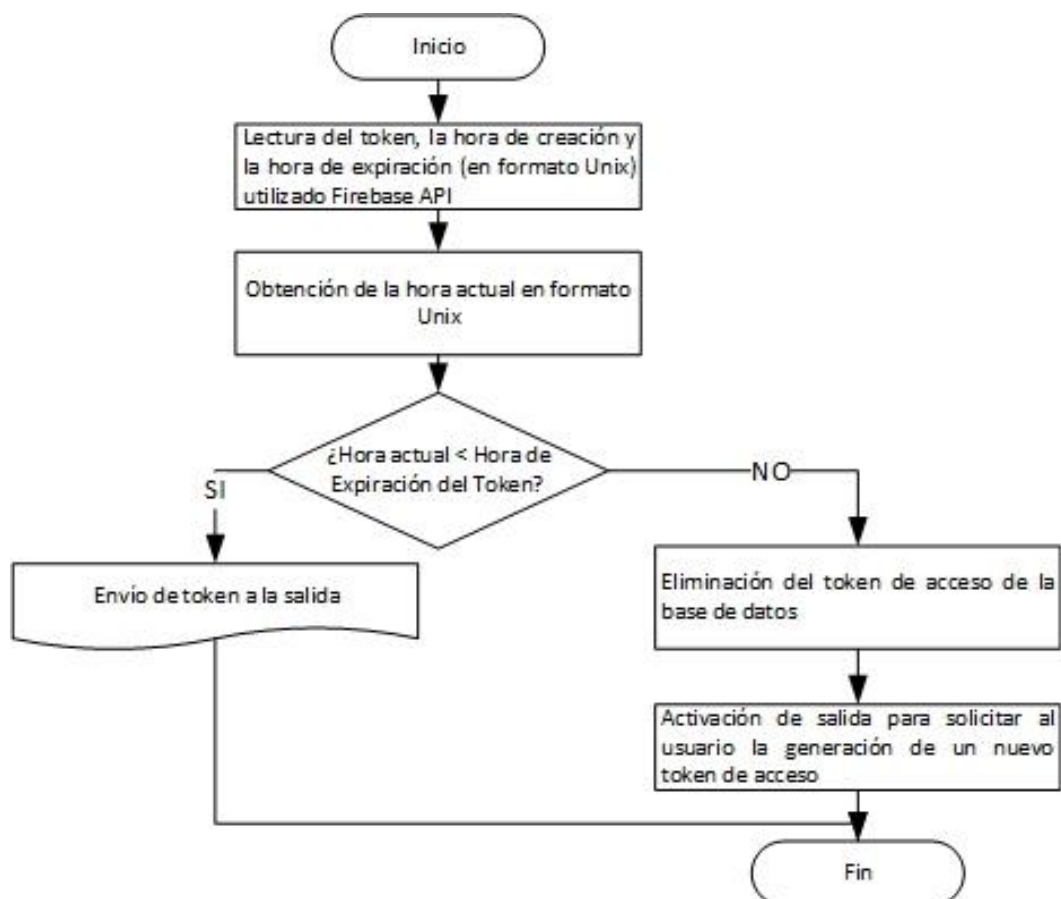
Este bloque posee una entrada de tipo BANG con la cual inicia la ejecución del bloque, una salida tipo STRING por donde se envía el token de acceso obtenido hacia el bloque de Dialogflow API y una salida de tipo BANG que envía una señal para que el robot solicite al usuario generar un nuevo token, una vez expire. En la Figura 90 se muestra el diagrama de flujo del bloque mencionado.

Un segundo bloque de programación corresponde a la utilización de API de Dialogflow para enviar las proposiciones del usuario y recibir las respuestas del chatbot que reproducirá el robot para entablar una conversación fluida.

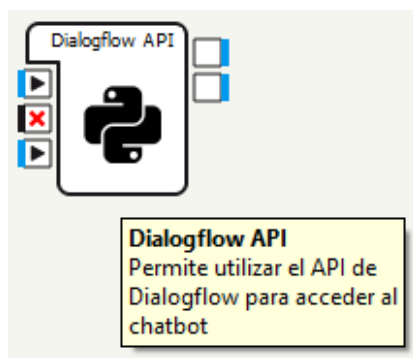
En la Figura 91 se muestra el bloque creado. Este bloque posee dos entradas tipo STRING por donde ingresarán la proposición del usuario y el token de acceso, y dos salidas de tipo STRING por donde se envían la respuesta del chatbot en texto para ser reproducida por el robot, y el nombre del intent en el que se encuentra actualmente, para validar cuándo la conversación debe terminar.

**Figura 90**

*Diagrama de flujo para la lectura del token de la base de datos en la nube*

**Figura 91**

*Bloque de programación para la utilización de API de Dialogflow*



En la Figura 92 se muestra el diagrama de flujo del bloque descrito. El código fuente del programa se muestra en el Anexo IV., Figura 137. Cabe mencionar que, para obtener las respuestas dadas por el usuario, se reutiliza el bloque de programación SpeechToText API, desarrollado previamente para la aplicación de educación.

En el Anexo IV., Figura 138 se muestra los bloques del programa creado en Choregraphe para la aplicación de atención al cliente. En la Figura 93 se muestra el diagrama de flujo de funcionamiento del programa mencionado.

### Figura 92

*Diagrama de flujo para la utilización de API de Dialogflow*

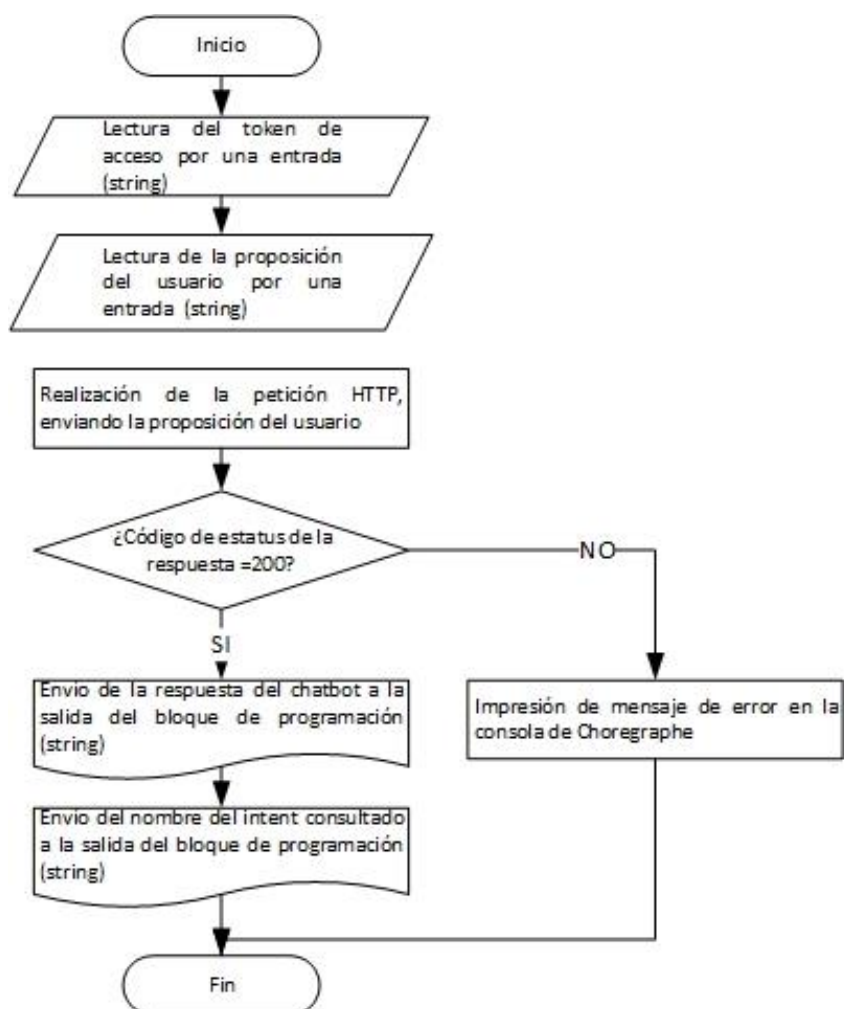
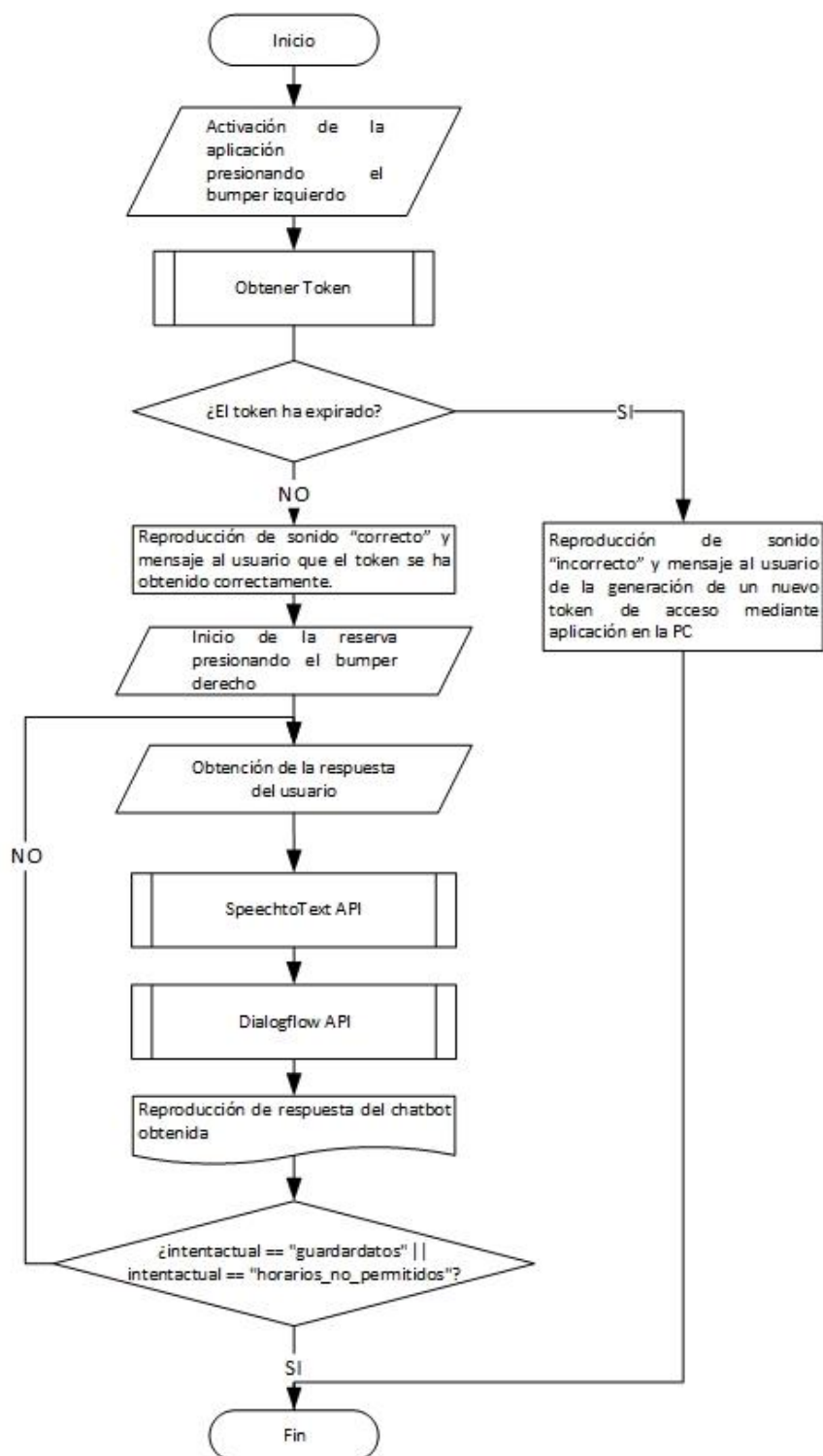




Figura 93

Diagrama de flujo del programa de la aplicación para atención al cliente



## Desarrollo de laboratorios teórico/prácticos

Una vez finalizada la implementación de los aplicativos, se desarrollan tres laboratorios teórico/prácticos, orientados a la enseñanza del manejo del robot humanoide NAO, así como su software principal, Choregraphe.


Se plantean los siguientes temas:


### 1. La instalación del software Choregraphe para la programación del robot NAO

Para el primer laboratorio, mostrado en la Figura 94, se cimentan conocimientos básicos para el manejo del robot NAO como el encendido, conexión y configuración, así como una introducción a Choregraphe mediante la realización de un programa básico, utilizando los bloques de programación disponibles en la librería.


#### Figura 94

*Laboratorio #1 - Instalación software Choregraphe*





**ESPE**  
UNIVERSIDAD DE LAS FUERZAS ARMADAS  
INNOVACIÓN PARA LA EXCELENCIA



**PRÁCTICA DE LABORATORIO # 1**

**I. TEMA:**

**INSTALACIÓN SOFTWARE CHOREGRAPHE**

**II. OBJETIVOS:**

**Objetivo General:**

Instalar el software Choregraphe para la utilización del robot humanoide NAO

**Objetivos Específicos:**


- Instalar el software Choregraphe
- Conectar el robot NAO al software
- Verificar el funcionamiento del programa


## 2. La instalación del software Webots para la simulación del robot NAO

Para el segundo laboratorio, mostrado en la Figura 95, se introduce un nuevo software que desempeñará un papel clave en pruebas de funcionamiento de los programas desarrollados, siempre y cuando no necesiten de los periféricos para su funcionamiento. Webots, permite la simulación, no solo del robot NAO, sino de una amplia variedad de robots, de distintas marcas. Para este caso, será vinculado a Choregraphe, con el fin de ejecutar las acciones programadas en el mismo.


### Figura 95

*Laboratorio #2 - Instalación software Webots con Choregraphe*





**ESPE**  
UNIVERSIDAD DE LAS FUERZAS ARMADAS  
INNOVACIÓN PARA LA EXCELENCIA



**PRÁCTICA DE LABORATORIO # 2**

**I. TEMA:**

**INSTALACIÓN SOFTWARE WEBOTS CON CHOREGRAPHE**

**II. OBJETIVOS:**

**Objetivo General:**

Simular el robot humanoide NAO utilizando el software Webots y programarlo mediante Choregraphe

**Objetivos Específicos:**

- Instalar el software Webots con el controlador naoqsim para una simulación completa del robot
- Conectar el software Webots con el programa Choregraphe para realizar pruebas de programación
- Verificar el funcionamiento del software con un programa básico.


## 3. La utilización del servicio SpeechToText API en el robot NAO

Finalmente, en el tercer laboratorio mostrado en la Figura 96, se exploran los temas mencionados en el presente trabajo de investigación, ya que está orientado a la

utilización del servicio en la nube SpeechToText API. Así, en su desarrollo, se mencionan temas como: la creación de una cuenta en Google Cloud, la creación de API Keys y la implementación del código fuente, para la utilización del servicio, en bloques de programación Python, en Choregraphe.

## Figura 96

*Laboratorio #3 - Utilización de servicios en la nube (SpeechToText API) en Choregraphe*



**PRÁCTICA DE LABORATORIO # 3**

**I. TEMA:**

**UTILIZACIÓN DE SERVICIOS EN LA NUBE (SPEECH TO TEXT) EN CHOREGRAPHE**

**II. OBJETIVOS:**

**Objetivo General:**

Utilizar el servicio en la nube Speech To Text API, de Google Cloud, mediante programación en Python

**Objetivos Específicos:**

- Crear una cuenta en Google Cloud
- Crear de API KEY para la utilización de servicios en la nube
- Implementar código fuente en Python, para la utilización del servicio en la nube Speech To Text API
- Verificar el funcionamiento del programa realizado

Todos los documentos mencionados anteriormente se encuentran adjuntos en la sección Anexo V.

## **Pruebas**

Para verificar el desempeño de los aplicativos implementados, se realizan cinco rondas de prueba a cada una, donde el usuario tendrá total interacción con el robot.

Cabe mencionar que, las actividades se han desarrollado dentro de las instalaciones del Laboratorio de Robótica de la Universidad de las Fuerzas Armadas – ESPE, donde se involucra a un solo sujeto de prueba de 27 años de edad, con el fin de cumplir las medidas de bioseguridad y distanciamiento social.

Previo a su manejo, se ha realizado una breve inducción sobre el robot NAO y una capacitación sobre la utilización de los aplicativos, donde se establecen los pasos a seguir para su correcto funcionamiento, ya que el usuario será el encargado de manipular al robot durante cada prueba. No obstante, el evaluador se encontrará cerca para aclarar cualquier duda e intervenir si es necesario.

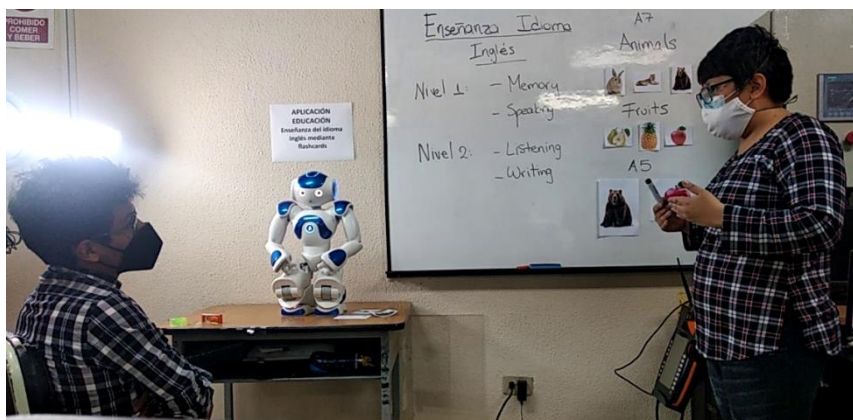
### ***Aplicativo para educación***

Para el primer aplicativo, se han realizado tres rondas de pruebas, en las cuales se utilizan diferentes juegos de tarjetas o flashcards, con las mismas imágenes: tres de animales (oso, conejo, tigre) y tres de frutas (manzana, pera, piña).

Adicional, se entrega al usuario un cuaderno cuadriculado y un esfero color azul para el desarrollo del segundo nivel de la aplicación. En la Figura 97 se muestra la inducción del aplicativo de educación, brindada al usuario.

**Figura 97**

*Inducción del usuario al aplicativo de educación*



Los parámetros de evaluación son:

- **Color de las imágenes**

Para evaluar el funcionamiento del servicio de reconocimiento de imágenes, el primer criterio refiere al desempeño del mismo en los casos donde las imágenes mostradas al robot sean a escala de grises y a color, para confirmar cuál de las alternativas es la más óptima. Se realiza una ronda de prueba para cada juego de tarjetas. En la Figura 98 se muestran las tarjetas utilizadas.

**Figura 98**

*Tarjetas utilizadas en escala de grises y a color*



- **Tamaño de las tarjetas**

El segundo criterio de evaluación del servicio de reconocimiento de imágenes está relacionado con el tamaño de las tarjetas, por lo cual se plantea la utilización de dos juegos a color, uno en formato A5 (148 x 210 mm) y otra en tamaño A7 (74 x 105 mm). Se realiza una ronda de prueba para cada juego de tarjetas. En la Figura 99 se muestran las tarjetas utilizadas.

**Figura 99**

*Tarjetas utilizadas en formato A5 y A7*



- **Presencia de luz puntual o focalizada**

El tercer criterio de evaluación es sobre la presencia/ausencia de luz puntual y su influencia en el correcto reconocimiento de las imágenes. Se realiza dos rondas de prueba, una con luz general (luminarias del Laboratorio de Robótica) y otra utilizando luz puntual cerca del robot y el usuario (lámpara LED). En la Figura 100 se muestra la calidad de imagen ante la presencia/ausencia de luz puntual.

**Figura 100**

*Luz ambiental y luz puntual*



- **Número de aciertos del usuario**

Para verificar el funcionamiento de la aplicación y su influencia en el usuario, se realiza una tabulación del número de intentos, aciertos y puntaje del usuario tanto en el nivel 1, donde se evalúa la retentiva y la pronunciación de palabras en inglés por parte del usuario, como en el nivel 2, donde se evalúan las habilidades de lectura y escritura en inglés. Esta información será obtenida desde la página web que permite la lectura de la base de datos en la nube.

- **Análisis de Costos**

Considerando que, los servicios en la nube utilizados en el presente aplicativo facturan un costo por solicitud, se realiza un análisis de costo para conocer cuántas interacciones puede realizar el usuario, al mes, para no superar las cuotas gratuitas que ofrece Google Cloud Platform.



### - Comparativa con soluciones locales

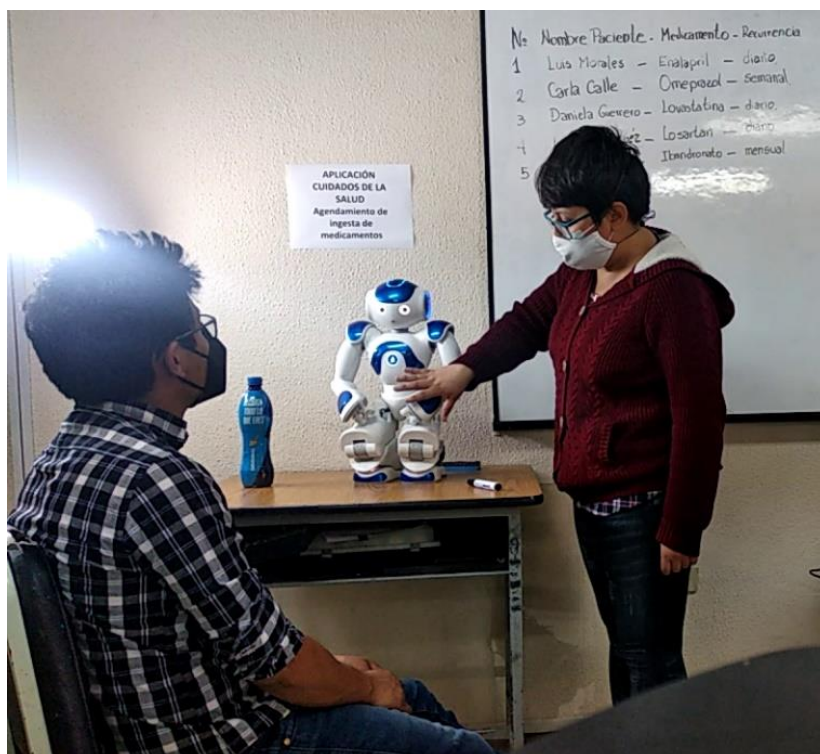
Como análisis adicional, se ha realizado una comparativa de los servicios utilizados en este aplicativo, versus soluciones offline desarrollados en trabajos de investigación previos.

#### ***Aplicativo para cuidados de la salud***

Para el segundo aplicativo orientado al agendamiento y consulta de eventos de ingesta de medicamentos en Google Calendar mediante el robot NAO, se realizan el ingreso de cinco pacientes como se muestra en la Tabla 8, los cuales se indican en la introducción a la aplicación, ejecutada por el evaluador al usuario.

#### **Figura 101**

*Inducción del usuario a la aplicación de cuidados de la salud*



**Tabla 8**

*Pacientes para pruebas del aplicativo para cuidados de la salud*

#	Nombre de Paciente	Nombre de Medicamento	Recurrencia de ingesta
1	Luis Morales	Enalapril	Diario
2	Carla Calle	Omeprazol	Semanal
3	Daniel Guerrero	Lovastatina	Diario
4	Luis Rodríguez	Losartán	Diario
5	María López	Ibandronato	Mensual

En la fecha de reserva se establece el día de la ejecución de las pruebas, es decir, 8 de julio de 2021. El apartado de hora de ingesta queda a criterio del usuario.

Los parámetros de evaluación son:

- **Precisión en la transcripción de audio a texto**

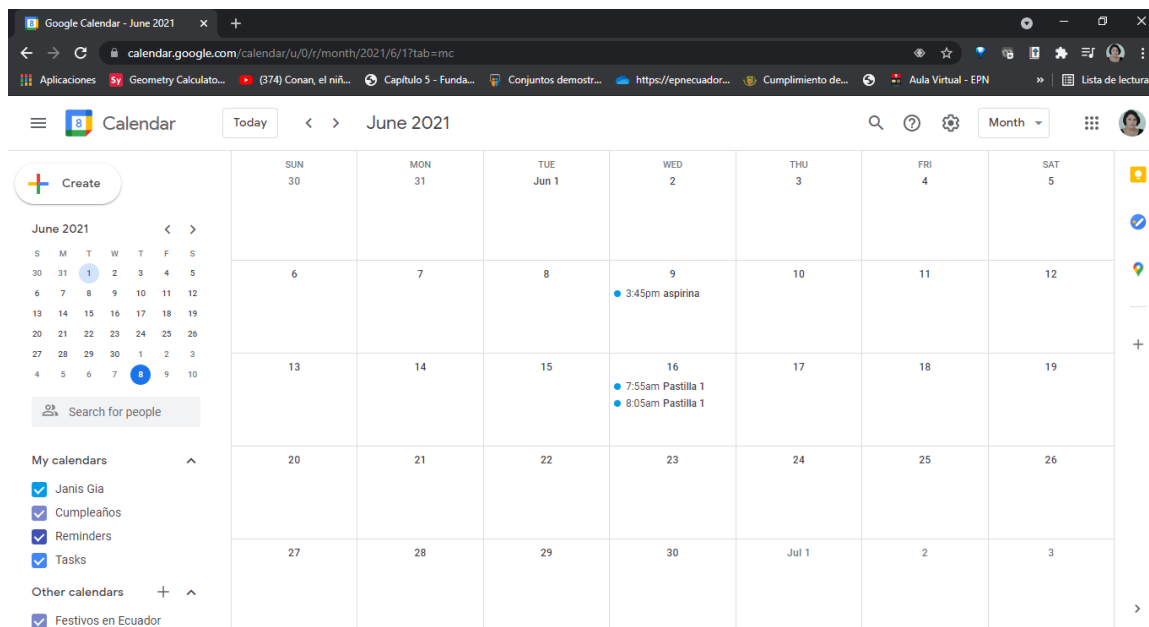
Para evaluar el funcionamiento del servicio de Speech to Text, se verifica en la página web, la precisión de los nombres de pacientes y medicamentos transcritos de audio a texto y almacenados en la base de datos en la nube.

- **Ejecución de agendamiento de eventos en Google Calendar**

Para verificar el correcto funcionamiento del aplicativo, se realiza la revisión de Google Calendar de la cuenta autorizada para el agendamiento de eventos. Se confirman que los datos ingresados sean correctos y que los eventos se repitan de acuerdo a la recurrencia predeterminada.

## Figura 102

Página de Google Calendar de la cuenta autorizada



### - Ejecución de consulta de eventos en Google Calendar

En la consulta de eventos, se verifica que el robot NAO recupere correctamente todos los pacientes, cuyas horas de ingesta se encuentren dentro del rango de tiempo de activación de la función.

### - Análisis de Costos

Considerando que, los servicios en la nube utilizados en el presente aplicativo facturan un costo por solicitud, se realiza un análisis de costo para conocer cuántas interacciones puede realizar el usuario, al mes, para no superar las cuotas gratuitas que ofrece Google Cloud Platform.

### - Comparativa con soluciones locales

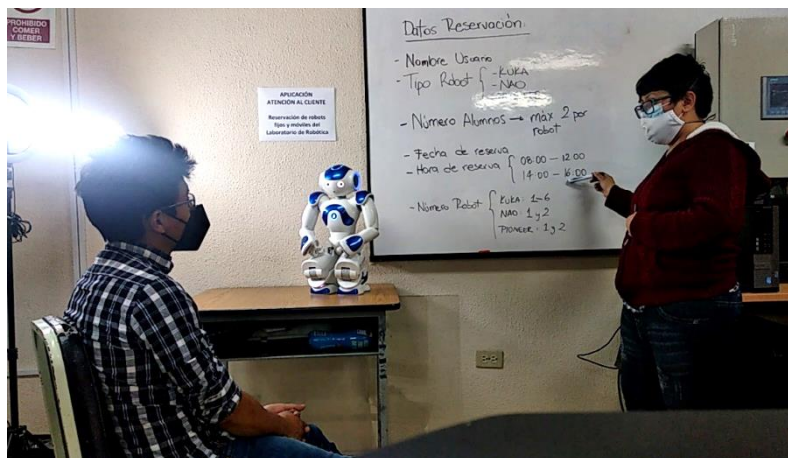
Como análisis adicional, se ha realizado una comparativa de los servicios utilizados en este aplicativo, versus soluciones offline desarrollados en trabajos de investigación previos.

### ***Aplicativo para atención al cliente***

Para el tercer aplicativo, donde el robot humanoide NAO actúa como asistente virtual para la reserva de robots fijos y móviles, se realiza el ingreso de cinco estudiantes. En la inducción al usuario sobre el aplicativo, se especifican los campos de información a ser completados por el mismo, para concretar cada reservación.

### **Figura 103**

*Inducción del usuario a la aplicación de atención al cliente*



Los parámetros de evaluación son:

### - Distancia entre el usuario y el robot

Para el primer criterio, se realizan tres reservaciones con distintas distancias entre el usuario y el robot para comprobar la opción adecuada en la que el robot pueda

captar, de mejor manera, las palabras del usuario y lograr realizar una transcripción exitosa.

- **Coincidencias correctas de las respuestas del usuario**

Para verificar el correcto funcionamiento del asistente virtual y su procesamiento de lenguaje natural, se realiza una tabulación de los casos exitosos en los que el chatbot reconoce las respuestas del usuario.

- **Presencia de ruido**

El tercer criterio de evaluación es sobre la presencia/ausencia de ruido en el ambiente y su influencia en el correcto reconocimientos de palabras en audio. Se realiza dos rondas de prueba, una sin ruido y otra con una fuente de ruido (celular con sonidos de ciudad) cerca al usuario y al robot.

- **Análisis de Costos**

Considerando que, los servicios en la nube utilizados en el presente aplicativo facturan un costo por solicitud, se realiza un análisis de costo para conocer cuántas interacciones puede realizar el usuario, al mes, para no superar las cuotas gratuitas que ofrece Google Cloud Platform.

- **Comparativa con soluciones locales**

Como análisis adicional, se ha realizado una comparativa de los servicios utilizados en este aplicativo, versus soluciones offline desarrollados en trabajos de investigación previos.

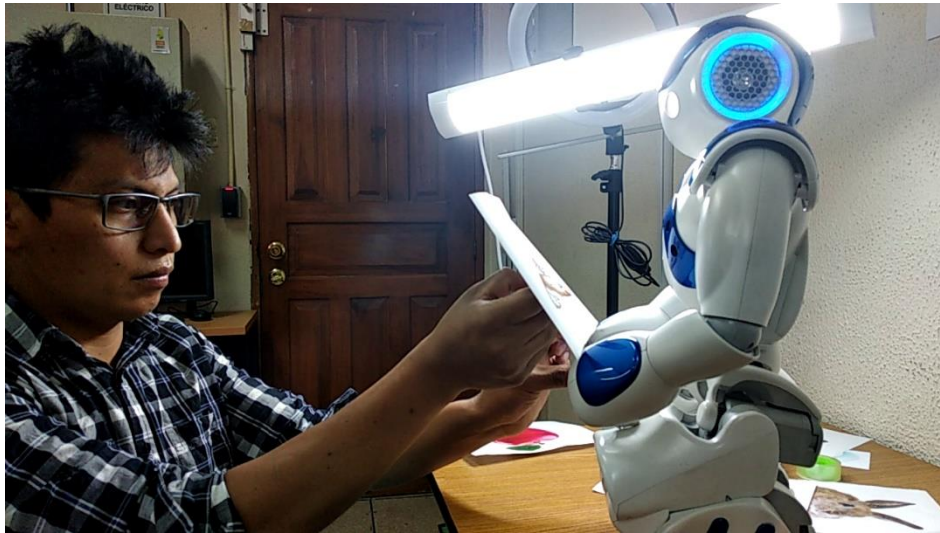
## Resultados

A continuación, se muestran los resultados obtenidos de las pruebas realizadas, para cada uno de los aplicativos desarrollados.

### *Aplicativo para educación*

#### Figura 104

*Pruebas de funcionamiento del aplicativo de educación*


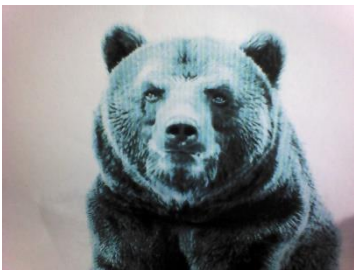






#### - **Color de las imágenes**

Para las pruebas de color de las imágenes mostradas al robot NAO, se han utilizado tarjetas de tamaño A5 tanto de animales como de frutas. Se han realizado seis rondas de pruebas de la aplicación, donde el usuario solo ha jugado en el nivel 1. A su vez, se obtiene la imagen capturada del robot para verificar la calidad de la misma antes de ser enviada al Vision API en la nube. En la Tabla 9 y Tabla 10 se muestran los resultados obtenidos.

Tabla 9

Resultados reconocimiento de imágenes en escala de grises




Descripción	Imagen capturada por el robot NAO	Resultados obtenidos de Vision API	Estado del reconocimiento de imagen
Tigre		tiger	Correcto
		tiger	
		tiger	
Oso		luggage & bags	Incorrecto
		luggage & bags	
		luggage & bags	
Conejo		animal	Incorrecto
		rabbit	
		animal	
Manzana		hat	Incorrecto
		fruit	
		packaged goods	
Pera		packaged goods	Incorrecto
		packaged goods	
		packaged goods	

Descripción	Imagen capturada por el robot NAO	Resultados obtenido de Vision API	Estado del reconocimiento de imagen
Piña		packaged goods packaged goods packaged goods	Incorrecto

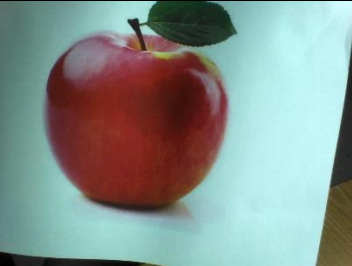


*Nota:* Las imágenes mostradas corresponden a las fotografías capturadas por la cámara del robot NAO.

**Tabla 10**

*Resultados reconocimiento de imágenes a color*

Descripción	Imagen capturada por el robot NAO	Resultado obtenido de Vision API	Estado del reconocimiento de imagen
Tigre		Tiger	Correcto
Oso		Brown Bear	Correcto
Conejo		Rabbit	Correcto



Descripción	Imagen capturada por el robot NAO	Resultado obtenido de Vision API	Estado del reconocimiento de imagen
Manzana		Apple	Correcto
Pera		Pear	Correcto
Piña		-	Incorrecto

*Nota:* Las imágenes mostradas corresponden a las fotografías capturadas por la cámara del robot NAO.

Una vez realizadas las pruebas, se verifica que la única imagen a color no reconocida por el robot es la piña, por lo cual es importante, previo a la utilización de la aplicación, realizar una ronda de prueba con las imágenes escogidas para comprobar que sean aptas para la misma. En el caso del criterio color, la presencia del mismo es clave para el reconocimiento de imágenes, ya que, como puede observarse en la Tabla 9, la mayoría de las imágenes mostradas al robot, en escala de grises, no pudieron ser reconocidas correctamente, incluso tomando en cuenta que se realizaron tres intentos con cada tarjeta.




### - Tamaño de las tarjetas

Para las pruebas de tamaño de las tarjetas mostradas al robot NAO, se han impreso las mismas en formato A5 y A7, a color. Se han realizado seis rondas de pruebas de la aplicación, donde el usuario solo ha jugado en el nivel 1. A su vez, se obtiene la imagen capturada del robot para verificar la calidad de la misma antes de ser enviada al Vision API en la nube. En la Tabla 11 y Tabla 12 se muestran los resultados obtenidos.

**Tabla 11**

*Resultados reconocimiento de imágenes con tarjetas formato A5*


Descripción	Imagen capturada por el robot NAO	Resultado obtenido de Vision API	Estado del reconocimiento de imagen
Tigre		Tiger	Correcto
Oso		Brown Bear	Correcto
Conejo		Rabbit	Correcto





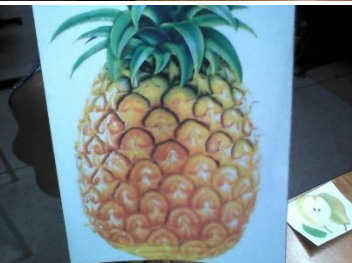
Descripción	Imagen capturada por el robot NAO	Resultado obtenido de Vision API	Estado del reconocimiento de imagen
Manzana		Apple	Correcto
Pera		Pear	Correcto
Piña		-	Incorrecto

*Nota:* Las imágenes mostradas corresponden a las fotografías capturadas por la cámara del robot NAO.

**Tabla 12**

*Resultados reconocimiento de imágenes con tarjetas formato A7*

Descripción	Imagen capturada por el robot NAO	Resultado obtenido de Vision API	Estado del reconocimiento de imagen
Tigre		Tiger	Correcto

Descripción	Imagen capturada por el robot NAO	Resultado obtenido de Vision API	Estado del reconocimiento de imagen
Oso		Brown Bear	Correcto
Conejo		Rabbit	Correcto
Manzana		Apple	Correcto
Pera		Pear	Correcto
Piña		-	Incorrecto

*Nota:* Las imágenes mostradas corresponden a las fotografías capturadas por la cámara del robot NAO.

Mediante las pruebas realizadas con los dos diferentes tamaños de tarjetas, se observa que no existe problemas de funcionamiento ya que se reconocen las imágenes mostradas, a excepción de la piña que tampoco se reconoció en la anterior prueba. Sin embargo, surge un nuevo inconveniente que está relacionado a la comodidad del usuario y el manejo de las tarjetas.


Así, el usuario indica que, mientras más pequeña sea la tarjeta, será más complicado centrarla en la cámara y mostrarla al robot humanoide NAO, o no podrá sostenerla de manera correcta sin cubrir alguna zona de la imagen. Por ello, se concluye que el tamaño correcto es en formato A5, para comodidad del usuario y funcionalidad del servicio Vision API.






- **Presencia de luz puntual o focalizada**

Para esta prueba, se ha realizado dos rondas de prueba, de seis imágenes cada uno, las cuales serán mostradas con presencia de luz puntual y con ausencia de la misma. Se utilizan tarjetas en formato A7, a color. En la Tabla 13 y Tabla 14 se muestran los resultados obtenidos.

**Tabla 13**

*Resultados reconocimiento de imágenes con luz puntual*






Descripción	Imagen capturada por el robot NAO	Resultado obtenido de Vision API	Estado del reconocimiento de imagen
Tigre		Tiger	Correcto


Descripción	Imagen capturada por el robot NAO	Resultado obtenido de Vision API	Estado del reconocimiento de imagen
Oso		Brown Bear	Correcto
Conejo		Rabbit	Correcto
Manzana		Apple	Correcto
Pera		Pear	Correcto
Piña		-	Incorrecto

*Nota:* Las imágenes mostradas corresponden a las fotografías capturadas por la cámara del robot NAO.

Tabla 14

*Resultados reconocimiento de imágenes con luz ambiental*

Descripción	Imagen capturada por el robot NAO	Resultado obtenido de Vision API	Estado del reconocimiento de imagen
Tigre		Tiger	Correcto
Oso		Brown Bear	Correcto
Conejo		Rabbit	Correcto
Manzana		Apple	Correcto
Pera		Pear	Correcto

Descripción	Imagen capturada por el robot NAO	Resultado obtenido de Vision API	Estado del reconocimiento de imagen
Piña		-	Incorrecto

*Nota:* Las imágenes mostradas corresponden a las fotografías capturadas por la cámara del robot NAO.

Al comparar las tablas anteriores, se puede observar que, el robot ha identificado correctamente las imágenes tanto con luz puntual, como luz ambiental del Laboratorio de Robótica. Sin embargo, es completamente recomendable que una fuente de luz focalizada sea colocada junto al robot ya que, en pruebas adicionales existieron algunos fallos de reconocimiento de imágenes, con gran hincapié en el reconocimiento óptico de caracteres, en el nivel 2 del juego.

Esto, debido a que sombras o zonas oscuras generadas en la imagen por una mala iluminación, al momento de ser analizadas por un algoritmo OCR pueden transformarse en caracteres (letras o números) aleatorios, que pueden infiltrarse en la transcripción y estropear el resultado obtenido.

Si realizamos un análisis comparativo entre las tres primeras pruebas realizadas, mediante el conteo del número de aciertos del robot en el reconocimiento de imágenes, bajo los criterios previamente mencionados, obtenemos la información que se muestra en la Tabla 15.



**Tabla 15**

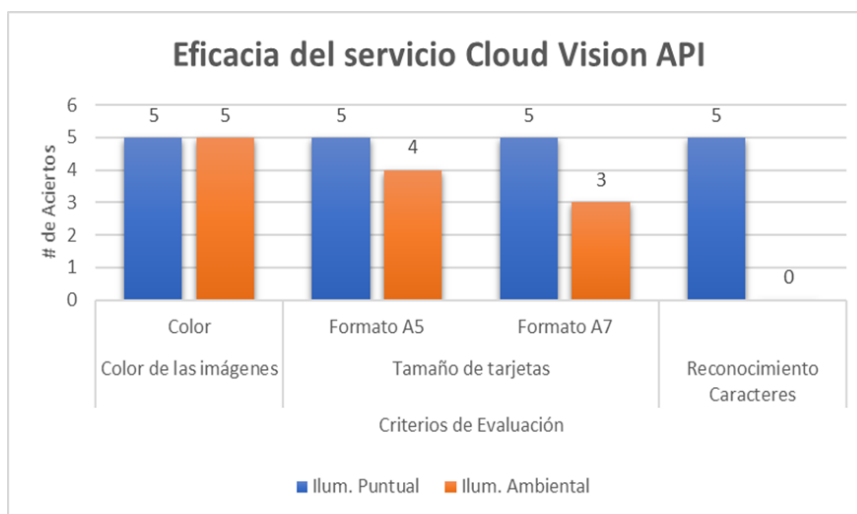
*Tabulación del número de aciertos obtenido del servicio Cloud Vision API*

Tipo de Iluminación	Color de las imágenes		Tamaño de tarjetas		Reconocimiento Caracteres
	Color	Escala de Grises	Formato A5	Formato A7	
<b>Ilum. Puntual</b>	5	1	5	5	5
<b>Ilum. Ambiental</b>	5	0	4	3	0

En la Figura 105, se observa que, la iluminación puntual y el color de las tarjetas son fundamentales para el correcto funcionamiento del servicio, tanto en el reconocimiento de imágenes como en el reconocimiento óptico de caracteres ya que, bajo estos parámetros se obtiene el mayor puntaje de aciertos. Cabe mencionar que, los gráficos de animales o frutas colocados en las tarjetas deben ser de los objetos reales e impresos a una muy buena calidad, para que el aplicativo se ejecute con normalidad.

**Figura 105**

*Diagrama de barras del número de aciertos obtenido del servicio Cloud Vision API*



### - Número de aciertos del usuario

Para la presente prueba, se realiza una ronda de juego con las seis imágenes, donde el usuario tendrá que superar tanto el nivel 1 como el nivel 2 del aplicativo de educación. Posteriormente se realizará una evaluación al usuario para comprobar el aprendizaje de las palabras. Los resultados del puntaje obtenido se visualizarán en la página web diseñada.

### Figura 106

*Resultados del usuario en el juego del aplicativo educación*

**Puntajes**

Haz clic en el siguiente botón para consultar los puntajes obtenidos

Se ha cargado con éxito.

Ingrese su nombre:

**Puntaje Total**

**210**



Usuario	Nivel	Respuesta Usuario	Respuesta Robot	Puntaje
Luis morales	1	tiger	tiger	10
Luis morales	1	tiger	tiger	10
Luis morales	1	tiger	Seguin	0
Luis morales	1	tiger	tiger	10
Luis morales	1	tiger	Tigard	0
Luis morales	1	tiger	tiger	10
Luis morales	1	tiger	tiger	10
Luis morales	2	tiger	ige	0
Luis morales	1	tiger	tiger	10
Luis morales	2	tiger	tiger	10
Luis morales	1	rabbit	right	0

En la Figura 106 se muestra una captura de la página web que permite generar una tabla con los datos del juego, tanto en el nivel 1 como el nivel 2. Con esta información, se realiza la tabulación de los intentos realizados y se muestran en la Tabla 16.

**Tabla 16**

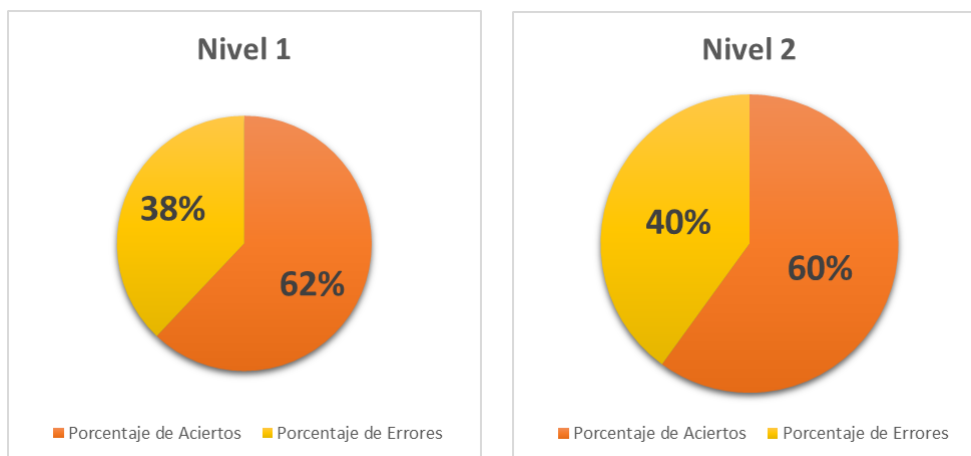
*Tabulación del número de aciertos obtenidos por el usuario*

Niveles	Total de Intentos	Número de Aciertos	Porcentaje de Aciertos	Porcentaje de Errores
Nivel 1	29	18	62,07	37,93
Nivel 2	5	3	60,00	40,00

En la Figura 107 se muestran las gráficas de pastel de cada nivel ejecutado. Se observa que, el usuario ha realizado más actividades en el nivel 1 que en el nivel 2 y, a su vez, ha obtenido una mayor cantidad de aciertos gracias a una práctica continua del vocabulario. Cabe mencionar que, el porcentaje de errores, es debido a la falta de costumbre del usuario, en la pronunciación de palabras en inglés.

**Figura 107**

*Gráficas de pastel, por nivel, del número de aciertos obtenidos por el usuario*



Adicional, se observa que, en la primera etapa del juego que consiste en memoria y pronunciación en inglés existen algunos inconvenientes con el usuario, ya que se le dificulta la entonación de las palabras.

Sin embargo, en el nivel 2, el usuario logra desenvolverse más fácilmente ya que tiene habilidades de escucha y escritura para el idioma inglés. En la Figura 108 se muestran las tarjetas utilizadas por el usuario y sus respuestas en la actividad del nivel 2. Posteriormente se realiza una evaluación para verificar lo aprendido y en la Figura 109 se muestran los resultados.

### Figura 108

*Resultados obtenidos de la participación del usuario en el aplicativo de educación*



### Figura 109

*Evaluación al usuario de las palabras aprendidas en inglés*



Al revisar las respuestas colocadas por el usuario, se comprueba que, el aprendizaje se ha logrado correctamente, gracias a varias sesiones de juego realizadas. En palabras del usuario, los ejercicios de pronunciación son los de mayor dificultad para el mismo, sin embargo, su ejecución repetitiva le permiten practicar constantemente para familiarizarse y perfeccionar sus habilidades en el idioma.

#### - Análisis de Costos

Para el análisis de costos, es necesario enlistar los servicios utilizados en el aplicativo, en este caso, corresponden: SpeechToText API (conversión de audio a texto), Cloud Vision API (reconocimiento de imágenes y reconocimiento óptico de caracteres), Firebase API (almacenamiento de datos en la nube).

Posteriormente se consulta la cuota mensual que ofrece Google Cloud Platform a sus usuarios, dentro de su programa gratuito. Sin embargo, cabe mencionar que, la cobertura del nivel gratuito varía según el servicio y que no todos los servicios de Google Cloud ofrecen esta opción. En la Tabla 17 se muestran las cuotas gratuitas.

**Tabla 17**

*Cuota gratuita de servicios en la nube utilizados en el aplicativo de educación*

Cloud	Cuota gratuita por mes			Cuota gratuita por día		
	Cloud Vision		Lectura de datos	Firestore		
Speech To Text	Reconocimiento de imágenes	Reconocimiento óptico de caracteres		Escritura de datos	Eliminación de datos	Datos almacenados
60 minutos	1.000	1.000	50.000	20.000	20.000	1GB

*Nota:* Información obtenida de Programa gratuito de Google Cloud (Google Cloud, 2020)

Posteriormente, se realiza un conteo del número de veces en que se utilizan los servicios previamente mencionados, en una sesión del aplicativo de educación. En la Tabla 18 se muestran los resultados obtenidos.

**Tabla 18**

*Tabulación de servicios en una sesión del aplicativo de educación*

<b>Servicio en la Nube</b>	<b># Veces utilizado</b>	<b>Descripción</b>
<b>Cloud Speech To Text</b>	15 segundos	Ingreso nombre usuario
	15 segundos	Ingreso pronunciación palabra en inglés
<b>Reconocimiento de imágenes</b>	1	Nivel 1
<b>Reconocimiento óptico de caracteres</b>	1	Nivel 2
<b>Escritura de datos</b>	1	Puntaje Nivel 2
	1	Puntaje Nivel 1
<b>Datos almacenados</b>	0,75 KB	Puntaje Nivel 2
	0,75KB	Puntaje Nivel 1

*Nota:* Valores relativos del uso de los servicios en la aplicación de educación.

Realizando un cálculo entre la Tabla 17 y la Tabla 18, se obtienen el número de veces en las que se puede utilizar el aplicativo de manera gratuita. En la Tabla 19 se muestran los resultados obtenidos.

**Tabla 19**

*Tabulación del uso gratuito del aplicativo de educación*

<b>Servicio en la Nube</b>	<b># Veces utilizado</b>	<b>Descripción</b>
<b>Cloud Speech To Text</b>	30 segundos	120 veces mensuales
<b>Reconocimiento de imágenes</b>	1	120 veces mensuales
<b>Reconocimiento óptico de caracteres</b>	1	120 veces mensuales
<b>Escritura de datos</b>	2	240 veces mensuales
<b>Datos almacenados</b>	1.5 KB	180 KB mensuales

*Nota:* Valores relativos del uso de los servicios en la aplicación de educación.

Se observa que, el número total de veces que puede utilizarse el aplicativo de educación, tomando en consideración que cada ronda consta de la ejecución tanto del nivel 1 como del nivel 2, es de 120 veces al mes. Esto, debido al límite gratuito de Cloud Speech To Text, ya que, sin importar que la duración de las grabaciones enviadas a la nube es de 3 segundos, para la facturación, Google Cloud las considera de mínimo 15 segundos.

Así, si se desea seguir utilizando la aplicación por sobre la cuota gratuita, Google Cloud deberá solo facturar lo correspondiente al servicio de Cloud Speech To Text, ya que las cuotas de los otros servicios no supera a los límites mostrados en la Tabla 17.

- **Comparativa con soluciones locales**

Considerando que, esta aplicación utiliza Cloud Vision API para el reconocimiento óptico de caracteres, se realiza un análisis comparativo con el trabajo titulado "*Implementación de un sistema basado en el robot NAO para lectura de textos en aplicaciones de Human Robot Interaction*", donde se afirma lo siguiente:

Como ya se indicó anteriormente, el procesamiento de la imagen se realiza en la Raspberry, pero esta es tomada por el robot NAO y se guarda en sus archivos en el directorio principal. Por este motivo se generó un nuevo problema ¿Cómo enviar de manera automática dicha fotografía para reconocer el texto del cuento?

Como solución se seleccionó a la Raspberry para obtener la imagen, procesarla y enviar nuevamente al robot el texto para usar la función TextToSpeech.

(Valarezo Correa, 2019, pág. 56)

En este texto, se manifiesta la utilización de hardware adicional al disponible en el robot humanoide NAO para el procesamiento de imágenes y el reconocimiento de caracteres mediante Python, lo que implica una mayor cantidad de recursos invertidos.

Mientras que, mediante la utilización del servicio en la nube Google Cloud Vision API, tanto el hardware como el software propio del robot son suficientes debido a que, tanto el procesamiento como el reconocimiento de imágenes quedan a cargo de la nube y el cliente es el receptor de la información final. En la Tabla 20 se muestran las diferencias entre las dos metodologías empleadas para el reconocimiento de caracteres.

#### **Tabla 20**

*Cuadro comparativo de solución offline y online para el reconocimiento óptico de caracteres*

<b>Criterios de Evaluación</b>	<b>Solución Offline</b>	<b>Solución Online</b>
<b>Hardware Adicional</b>	Raspberry Pi	-
<b>Software Adicional</b>	Motor de OCR Tesseract	Cloud Vision API
<b>Tiempo de Reconocimiento</b> (136 caracteres)	12,05s	3,97s
<b>Costo</b>	\$70	\$0*

*Nota:* \* El servicio Cloud Vision API es gratuito para 1000 solicitudes mensuales.

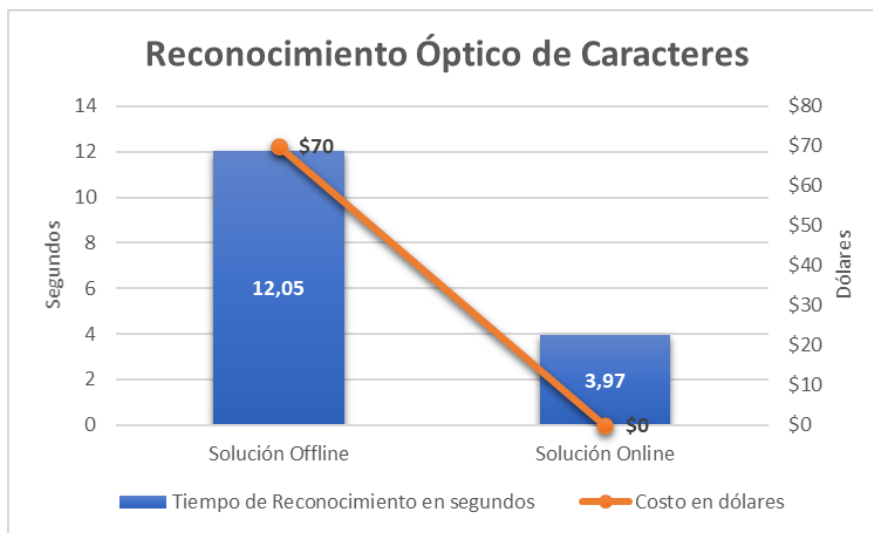
Con ayuda de la Figura 110 podemos concluir que, la solución offline no solo utiliza más recursos, sino que es más costosa y menos eficiente que la solución online ya que, realizando un cálculo en el reconocimiento de 136 caracteres en una imagen, se



obtiene que, el servicio Cloud Vision es **3 veces más rápido** que el motor OCR Tesseract ejecutado en una Raspberry PI.

### Figura 110

*Resultados solución offline y online para el reconocimiento óptico de caracteres*



Finalmente, se observa una mejora significativa en la ejecución de este tipo de tareas, donde es necesaria la obtención de texto de una imagen. Adicional, cabe resaltar que, si bien los servicios en la nube implican un costo monetario, si la cuota de solicitudes mensual es menor a un cierto límite específico, su utilización es gratuita.

## Aplicativo para cuidados de la salud

**Figura 111**

*Pruebas de funcionamiento del aplicativo de cuidados de la salud*



### - Precisión en la transcripción de audio a texto

Para el presente criterio, se realiza una comparativa entre los datos almacenados en la base de datos en la nube desplegados en la Figura 112 con los mostrados en la Tabla 8.

**Figura 112**

*Consulta de pacientes registrados en la aplicación cuidados de la salud*

Pacientes Registrados				
Haz clic en el siguiente botón para consultar los usuarios disponibles				
Se ha cargado con éxito.				
<input type="button" value="Consultar"/>				
Nombre Paciente	Nombre Medicamento	Datetime Inicio	Datetime Fin	Recurrencia
daniel Lacalle	aspirina	2021-7-7T15:30:00	2021-7-7T15:35:00	SEMANAL
Juan Pérez	aspirina	2021-12-15T12:30:00	2021-12-15T12:35:00	DIARIO
Carlos Páez	aspirina	2021-6-9T15:45:00	2021-6-9T15:50:00	TRIMESTRAL
Luis morales	enalapril	2021-7-8T14:45:00	2021-7-8T14:50:00	DIARIO
Carla calle	omeprazol	2021-7-8T15:30:00	2021-7-8T15:35:00	SEMANAL
Daniela Guerrero	lovastatina	2021-7-8T15:30:00	2021-7-8T15:35:00	DIARIO
Luis Rodríguez	losartán	2021-7-8T15:0:00	2021-7-8T15:5:00	DIARIO
María López	ibandronato	2021-7-8T15:15:00	2021-7-8T15:20:00	MENSUAL
Juan Pérez	Dayamineral	2021-7-8T15:30:00	2021-7-8T15:35:00	SEMANAL
Luis calle	aspirina	2021-7-8T16:0:00	2021-7-8T16:5:00	DIARIO
Juan Caixa	aspirina	2021-7-8T15:15:00	2021-7-8T15:20:00	DIARIO

En la Tabla 21 se realiza la tabulación de caracteres de nombre de pacientes y medicamentos, y se realiza una comparativa entre el texto entregado al usuario y el obtenido de la transcripción realizada por el SpeechToText API, con el fin de contabilizar el número de aciertos y errores.

**Tabla 21**

*Tabulación de aciertos y errores en la transcripción de nombres de pacientes y medicamentos*

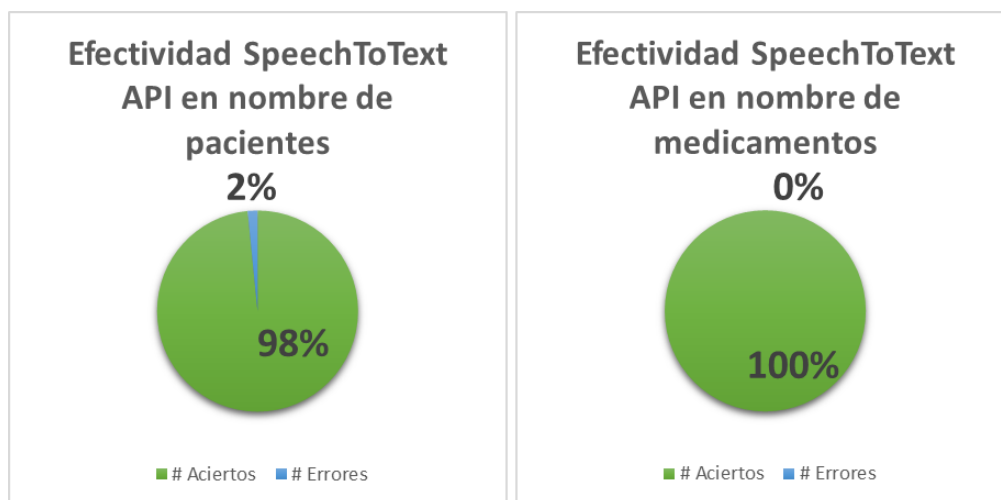
<b>Nombre paciente entregado</b>	<b>Nombre paciente transcrito</b>	<b># Caracteres</b>	<b># Aciertos</b>	<b># Errores</b>
Luis Morales	Luís morales	12	11	1
Carla Calle	Carla calle	11	11	0
Daniel Guerrero	Daniela Guerrero	15	15	0
Luis Rodríguez	Luis Rodríguez	14	14	0
María López	María López	11	11	0
<b>Nombre medicamento entregado</b>	<b>Nombre medicamento transcrito</b>	<b># Caracteres</b>	<b># Aciertos</b>	<b># Errores</b>
Enalapril	enalapril	9	9	0
Omeprazol	omeprazol	9	9	0
Lovastatina	lovastatina	11	11	0
Losartán	losartán	8	8	0
Ibandronato	ibandronato	11	11	0

En la Figura 113 se muestran las gráficas de pastel que evidencian la efectividad del SpeechToText API para realizar las transcripciones. Al realizar la comparativa, se puede observar que los datos ingresados al robot, por voz, se han obtenido correctamente.

Esto, gracias al servicio en la nube SpeechToText. Sin embargo, es importante recalcar que, el usuario deberá pronunciar, de manera clara y sin mascarilla, el nombre de los medicamentos para evitar posibles errores.

**Figura 113**

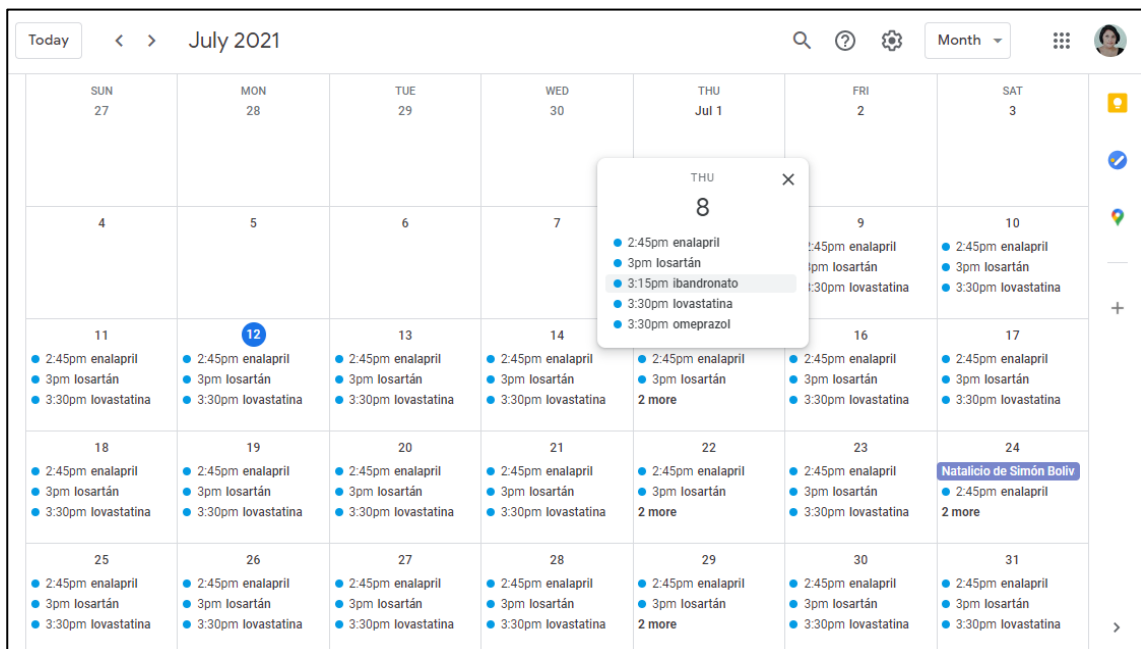
*Porcentaje de efectividad del SpeechToText API para realizar las transcripciones*



#### - Ejecución de agendamiento de eventos en Google Calendar

En la Figura 114, se observan los eventos mostrados en la Tabla 8 agendados en el Google Calendar de la cuenta utilizada, en el proceso de autenticación, de la aplicación. Cabe mencionar que, los eventos cuya recurrencia de ingesta es diaria, se repetirán todos los días hasta el 7 de julio del 2023.

Si la ingesta es semanal, los eventos serán recordados cada semana, el mismo día agendado. En este caso, la fecha 8 de julio de 2021 cae un día jueves, por lo tanto se repetirá cada jueves hasta el 28 de junio del 2035. Finalmente, si la ingesta es mensual, el evento se repetirá el mismo día agendado, cada mes. En este caso, la fecha de agendamiento es 8 de julio de 2021, por lo tanto se repetirá cada día 8 del mes, hasta el 8 de abril del 2082.

**Figura 114***Agendamiento de eventos en Google Calendar*

Cabe mencionar que, todos los eventos recurrentes pueden eliminarse completamente ya que aparecerá una ventana donde se muestran las opciones de borrar: solo un evento en la fecha seleccionada, los eventos de la fecha escogida en adelante, o todos los eventos agendados.

Adicional, cada evento agendado será recordado mediante un correo electrónico, generado automáticamente a la cuenta autenticada, cinco minutos antes de su ejecución.

#### - Ejecución de consulta de eventos en Google Calendar

Para este criterio, se realiza la consulta de eventos en Google Calendar mediante el robot humanoide NAO. En la Figura 115 se muestra la acción ejecutada por el usuario, para activar esta función. Se realizó la primera consulta a las 14:55 del 8 de

julio de 2021, donde el robot NAO indicó al usuario solo un evento agendado: paciente Luis Morales y medicamento Enalapril.

### **Figura 115**

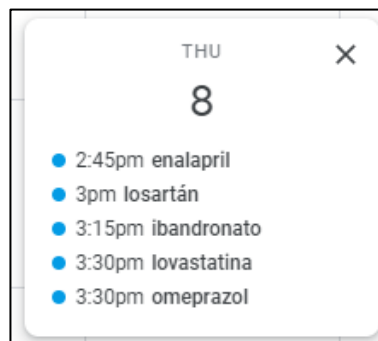
*Ejecución de la consulta de eventos en Google Calendar*



Posteriormente se realizó una nueva consulta a las 15:21 horas, donde el robot comunicó al usuario la existencia de cuatro pacientes agendados, correspondientes a los siguientes: paciente Luis Rodríguez y medicamento Losartán, paciente María López y medicamento Ibandronato, paciente Daniela Guerrero y medicamento Lovastatina, paciente Carla Calle y medicamento Omeprazol, los cuales corresponden a los datos entregados en la Tabla 8 y a los eventos almacenados en Google Calendar, mostrados en la Figura 116.

**Figura 116**

*Eventos almacenados en Google Calendar*



La consulta de eventos se repite en las horas mostradas en la Tabla 22, para confirmar que, todos los eventos previamente agendados dentro del rango de tiempo de consulta sean recuperados.

**Tabla 22**

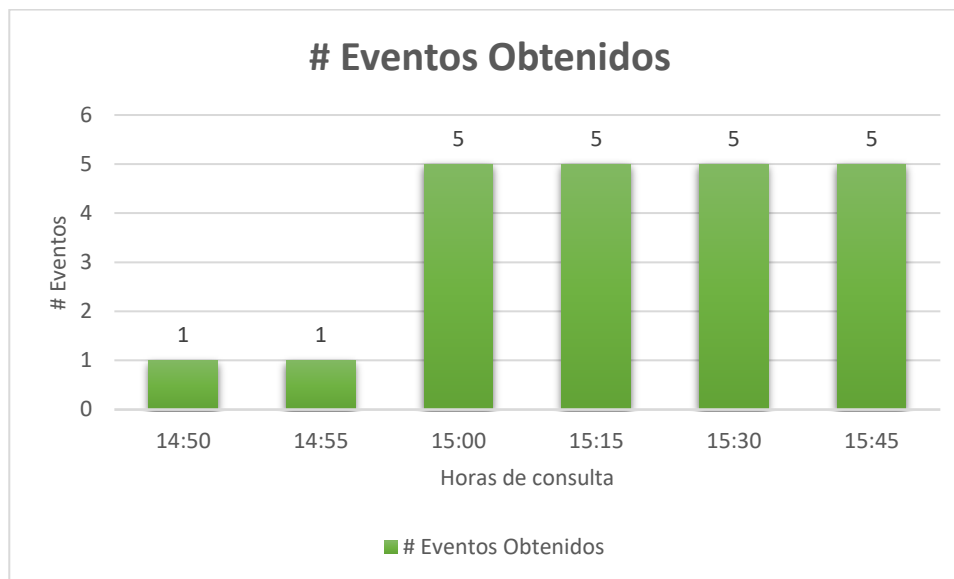
*Tabulación del número de eventos obtenidos en diferentes horas de consulta*

<b>Horas de Consulta</b>	<b># Eventos Obtenidos</b>
<b>14:50</b>	1
<b>14:55</b>	1
<b>15:00</b>	5
<b>15:15</b>	5
<b>15:30</b>	5
<b>15:45</b>	5

En la Figura 117 se observa que, el 100% de eventos agendados han sido consultados de manera exitosa. Así, se comprueba que, sin importar la hora de consulta, NAO recuperará los eventos de Google Calendar que se encuentren dentro del rango de la hora de reservación. En este caso, si la consulta se realiza a las 15:03 o 15:55 horas, el aplicativo mencionará todos los eventos que se encuentren agendados entre las 15:00 y 15:59 horas.

**Figura 117**

*Resultado del número de eventos obtenidos en diferentes horas de consulta*



#### - **Análisis de Costos**

Para el análisis de costos, es necesario enlistar los servicios utilizados en el aplicativo, en este caso, corresponden: SpeechToText API (conversión de audio a texto), Google Calendar API (agendamiento y consulta de eventos en Google Calendar), Firebase API (almacenamiento de datos en la nube).

Posteriormente se consulta la cuota mensual que ofrece Google Cloud Platform a sus usuarios, dentro de su programa gratuito. En la Tabla 23 se muestran las cuotas gratuitas.



**Tabla 23**

*Cuota gratuita de servicios en la nube utilizados en el aplicativo de cuidados de la salud*

Cuota gratuita por mes		Cuota gratuita por día			
Cloud Speech To Text	Google Calendar	Firestore			
	Creación eventos	Lectura de datos	Escritura de datos	Eliminación de datos	Datos almacenados
60 minutos	100.000	50.000	20.000	20.000	1GB

*Nota:* Información obtenida de Programa gratuito de Google Cloud (Google Cloud, 2020)

Posteriormente, se realiza un conteo del número de veces en que se utilizan los servicios previamente mencionados, en una sesión del aplicativo de cuidados de salud.

En la Tabla 24 se muestran los resultados obtenidos.

**Tabla 24**

*Tabulación de servicios en una sesión del aplicativo de cuidados de la salud*

Servicio en la Nube	# Veces utilizado	Descripción
Cloud Speech To Text	15 segundos	Ingreso nombre paciente
	15 segundos	Ingreso nombre medicamento
Agendamiento de eventos	1	Agendamiento de eventos
Consulta de eventos	1	Consulta de eventos
Escritura de datos	1	Agendamiento de eventos
Datos almacenados	0,75 KB	Almacenamiento de datos de pacientes

*Nota:* Valores relativos del uso de los servicios en la aplicación de cuidados de la salud

Realizando un cálculo entre la Tabla 23 y Tabla 24, se obtienen el número de veces en las que se puede utilizar el aplicativo de manera gratuita. En la Tabla 25 se muestran los resultados obtenidos.

**Tabla 25**

*Tabulación del uso gratuito del aplicativo de cuidados de la salud*

<b>Servicio en la Nube</b>	<b># Veces utilizado</b>	<b>Descripción</b>
<b>Cloud Speech To Text</b>	30 segundos	120 veces mensuales
<b>Agendamiento de eventos</b>	1	120 veces mensuales
<b>Consulta de eventos</b>	1	120 veces mensuales
<b>Escritura de datos</b>	1	120 veces mensuales
<b>Datos almacenados</b>	0.75 KB	90 KB mensuales

*Nota:* Valores relativos del uso de los servicios en la aplicación de cuidados de la salud.

Se observa que, el número total de veces que puede utilizarse el aplicativo de cuidados de la salud, específicamente el agendamiento de eventos, es de 120 veces al mes. Esto, nuevamente debido al límite gratuito de Cloud Speech To Text, ya que, sin importar que la duración de las grabaciones enviadas a la nube es de 3 segundos, para la facturación, Google Cloud las considera de mínimo 15 segundos.

Así, si se desea seguir utilizando la aplicación por sobre la cuota gratuita, Google Cloud deberá solo facturar lo correspondiente al servicio de Cloud Speech To Text, ya que las cuotas de los otros servicios no supera a los límites mostrados en la Tabla 23. Por otro lado, en la actividad de consultar los eventos no existe un rubro netamente facturado ya que la utilización del API de Google Calendar es gratuita, solo no debe superarse los límites para evitar posibles errores de funcionamiento.

#### **- Comparativa con soluciones locales**

Dado que, esta aplicación utiliza Firebase API para el almacenamiento de datos en la nube, en tiempo real, se realiza un análisis comparativo con el trabajo titulado *“Desarrollo de aplicaciones interactivas mediante robótica persuasiva para adultos*

mayores utilizando el robot humanoide NAO”, donde en el apartado *Trabajos Futuros* se afirma lo siguiente:

De ser posible la implementación de una base de datos en la que se almacene la información de evolución del usuario y que la aplicación de dificultad de los ejercicios dependa del uso de esta información complementaria de manera positiva al sistema. (Silva Freire, 2020, pág. 143)

En las aplicaciones orientadas a juegos interactivos o asistentes personales, la integración de la base de datos en la nube es primordial para almacenar información relevante que permita una mejora continua en el desempeño de las aplicaciones y una retroalimentación de respuestas para dar seguimiento a los usuarios. Mediante Firebase API es posible realizar varias tareas sobre una base de datos que, se actualiza en tiempo real y que permite su acceso desde cualquier dispositivo, web o móvil.

En este caso, los datos almacenados en la nube, son posteriormente recuperados desde una página web local, mediante programación de scripts en lenguaje JavaScript, así con la información obtenida, es posible realizar cálculos, validar datos, desplegar tablas de puntajes, generar informes y gráficas, entre otras. En la Tabla 26 se resumen las diferencias entre las dos metodologías para el almacenamiento de datos.

**Tabla 26**

*Cuadro comparativo de la solución offline y online en el almacenamiento de datos*

<b>Características</b>	<b>Solución Offline</b>	<b>Solución Online</b>
<b>Acceso</b>	Local (en el robot)	Remoto (desde cualquier lugar)
<b>Almacenamiento</b>	Archivo de texto plano	Base de datos en la nube
<b>Costo</b>	Gratis	Cuota diaria gratis

## ***Aplicativo para atención al cliente***

### **Figura 118**

*Pruebas de funcionamiento del aplicativo de atención al cliente*






#### **- Distancia entre el usuario y el robot**

En el presente criterio, se realizaron tres reservaciones de robots, a tres distancias distintas para verificar la distancia máxima que debe existir entre el robot y el usuario, para que pueda escucharlo de manera clara y precisa. Cabe mencionar que, las pruebas se realizaron con el usuario sin mascarilla, ya que, sin importar la distancia, éste impide que el robot pueda obtener un correcto audio. En la Tabla 27 se muestran los resultados obtenidos.

La distancia es un factor clave para el correcto funcionamiento del servicio en la nube SpeechToText, ya que, como se observa en la Tabla 27, a menor distancia, el robot será capaz de realizar una grabación de voz clara, de la cual podrá extraerse la información como texto. A su vez, si la distancia es grande (150 cm), el robot captará apenas ciertos ruidos que, al ejecutarse el aplicativo, darán error en su funcionamiento ya que el servicio en la nube no podrá extraer ninguna información.

**Tabla 27***Resultados reconocimiento de voz a diferentes distancias*

Distancia	Evidencia Fotográfica	Resultados
50 cm		El robot logra captar las respuestas del usuario al primer intento, sin levantar la voz
100 cm		El robot logra captar las respuestas del usuario, si el usuario levanta levemente la voz.
150 cm		El robot no logra captar las respuestas del usuario en ningún intento.

**- Coincidencias correctas de las respuestas del usuario**

Para este criterio, se han realizado cinco reservaciones seguidas, donde el usuario decidirá qué información ingresar. Para verificar su funcionamiento, se ingresa a la página web donde se realiza una búsqueda de los robots y número de robots reservados. En la Figura 119 y Figura 120 se muestran los resultados.

**Figura 119**

*Reservaciones realizadas del robot KUKA número 3*

### Reserva robots fijos y móviles con NAO

NAO es tu asistente virtual y te permite realizar la reserva de robots fijos y móviles disponibles en el Laboratorio de Robótica de la Universidad de las Fuerzas Armadas - ESPE.

¡Consulta Aquí tu reserva!

## Reservaciones

Haz clic en el siguiente botón para consultar los robots reservados

Se ha cargado con éxito.

Seleccione el tipo de robot:  Seleccione el número de robot:

## Reservas Realizadas

Nombre Usuario	Tipo Robot	Número Robot	Fecha Reserva	Hora Reserva	Número Personas
Daniel Calle	KUKA	3	2021-07-10	10:00:00	1
María Díaz	KUKA	3	2021-07-04	09:00:00	1

**Figura 120**

*Reservaciones realizadas del robot KUKA número 5*

## Reservaciones

Haz clic en el siguiente botón para consultar los robots reservados

Se ha cargado con éxito.

Seleccione el tipo de robot:  Seleccione el número de robot:

## Reservas Realizadas

Nombre Usuario	Tipo Robot	Número Robot	Fecha Reserva	Hora Reserva	Número Personas
Gabriel Morales	KUKA	5	2021-08-09	09:00:00	2
Gabriel Morales	KUKA	5	2021-08-12	08:00:00	2
María Díaz	KUKA	5	2021-08-10	10:00:00	1

Como se puede observar, las reservaciones se han realizado con éxito en las fechas y horas preestablecidas. Cabe mencionar que, al momento de ejecutar el aplicativo, éste se encarga de realizar una revisión completa de la base de datos para que, si el usuario ha escogido un robot, del mismo número, y en la misma fecha que un estudiante anterior, NAO mencione las horas disponibles para no crear conflictos de horarios.

Adicional, si el robot no es capaz de distinguir un nombre o un apellido, por no ser fácilmente identificables, el desarrollador deberá almacenarlo en las frases de entrenamiento del chatbot en Dialogflow, para que en próximas reservaciones, el robot pueda identificarlo y almacenarlo.

- **Presencia de ruido**

En este criterio, se han realizado dos rondas de prueba, en las cuales se ha agregado una fuente de ruido con ayuda de un celular y una computadora portátil, cerca del robot NAO. Esto, para verificar la influencia del mismo en la obtención de respuestas claras. Las dos pruebas se realizaron con el usuario a 50 cm de distancia del robot.

En este caso, el ruido no influyó de manera negativa en la grabación del audio, esto debido a que el usuario se encuentra lo suficientemente cerca de los micrófonos para que su respuesta sobresalga dentro de la grabación y además, en presencia de ruido, es natural que el usuario reaccione, levantando la voz al hablar con el robot.

Adicional, es sumamente importante que, en todo momento, el usuario realice una correcta pronunciación de palabras y que mantenga un volumen continuó en la voz hasta finalizar la respuesta.

Si realizamos un análisis comparativo entre las tres pruebas realizadas, mediante el conteo del número de aciertos del robot en la transcripción de audio a texto, bajo los criterios previamente mencionados, obtenemos la información que se muestra en la Tabla 28.

**Tabla 28**

*Tabulación del número de aciertos obtenido del servicio SpeechToText API*

Ambiente	Distancia		
	50cm	100cm	150 cm
<b>Sin Ruido</b>	5	5	2
<b>Con Ruido</b>	5	3	0

En la Figura 121, se observa que la distancia entre el usuario y el robot es un condicionante para el correcto funcionamiento del servicio ya que, a menor distancia entre los dos, el audio grabado de la voz del usuario será más claro, independientemente de la existencia de ruido ambiental.

**Figura 121**

*Diagrama de barras del número de aciertos obtenido del servicio SpeechToText API*





### - Análisis de Costos

Para el análisis de costos, es necesario enlistar los servicios utilizados en el aplicativo, en este caso, corresponden: SpeechToText API (conversión de audio a texto), Dialogflow API (diseño y ejecución de asistentes virtuales), Firebase API (almacenamiento de datos en la nube).

Posteriormente se consulta la cuota mensual que ofrece Google Cloud Platform a sus usuarios, dentro de su programa gratuito. En la Tabla 29 se muestran las cuotas gratuitas.

**Tabla 29**

*Cuota gratuita de servicios en la nube utilizados en el aplicativo de atención al cliente*

Cuota gratuita por mes		Cuota gratuita por día			
Cloud Speech To Text	Dialogflow API	Firestore			
	Chatbot	Lectura de datos	Escritura de datos	Eliminación de datos	Datos almacenados
60 minutos	-	50.000	20.000	20.000	1GB

*Nota:* Información obtenida de Programa gratuito de Google Cloud (Google Cloud, 2020)

A continuación, se realiza un conteo del número de veces en que se utilizan los servicios previamente mencionados, en una sesión del aplicativo de atención al cliente. En la Tabla 30 se muestran los resultados obtenidos.

**Tabla 30**

*Tabulación de servicios en una sesión del aplicativo de atención al cliente*

<b>Servicio en la Nube</b>	<b># Veces utilizado</b>	<b>Descripción</b>
<b>Cloud Speech To Text</b>	120 segundos	Ingreso datos para reserva
<b>Chatbot</b>	1	Agendamiento de eventos
<b>Escritura de datos</b>	7	Almacenamiento de datos para reserva
<b>Datos almacenados</b>	0,75 KB	Almacenamiento de datos para reserva

*Nota:* Valores relativos del uso de los servicios en la aplicación de atención al cliente

Realizando un cálculo entre la Tabla 29 y Tabla 30, se obtienen el número de veces en las que se puede utilizar el aplicativo de manera gratuita. En la Tabla 31 se muestran los resultados obtenidos.

**Tabla 31**

*Tabulación del uso gratuito del aplicativo de atención al cliente*

<b>Servicio en la Nube</b>	<b># Veces utilizado</b>	<b>Descripción</b>
<b>Cloud Speech To Text</b>	120 segundos	30 veces mensuales
<b>Chatbot</b>	1	30 veces mensuales
<b>Escritura de datos</b>	7	210 veces mensuales
<b>Datos almacenados</b>	0.75 KB	157.5 KB mensuales

*Nota:* Valores relativos del uso de los servicios en la aplicación de atención al cliente.

Se observa que, el número total de veces que puede utilizarse el aplicativo de atención al cliente para la reserva de robots es de 30 veces al mes. Esto, nuevamente debido al límite gratuito de Cloud Speech To Text, ya que, sin importar que la duración de las grabaciones enviadas a la nube es de 3 segundos, para la facturación, Google Cloud las considera de mínimo 15 segundos.

Así, si se desea seguir utilizando la aplicación por sobre la cuota gratuita, Google Cloud deberá solo facturar lo correspondiente al servicio de Cloud Speech To Text, ya que las cuotas de los otros servicios no supera a los límites mostrados en la Tabla 31.

Sin embargo, ya que se utiliza una versión gratuita de Dialogflow Essentials, no tiene ningún costo, pero este factor se ve opacado gracias a las limitaciones de funciones que puede desempeñar este tipo de suscripción.

Por ello, Dialogflow Essentials Trial Edition es recomendada para experimentar con Dialogflow ya que ofrece una cuota limitada y asistencia de la comunidad y mediante correo electrónico. Si se necesita realizar un proyecto más grande, será necesario actualizar la edición de la cuenta a Dialogflow ES Edition o Dialogflow CX Edition las cuales son totalmente pagadas.

En este caso, Dialogflow Essentials Trial Edition es adecuado para la aplicación desarrollada ya que ofrece las características necesarias del agente de Essentials y funcionalidades básicas de Firebase para el almacenamiento de datos.

#### - **Comparativa con soluciones locales**

Considerando que, esta aplicación utiliza Dialogflow API para el desarrollo de asistentes virtuales que utilizan inteligencia artificial y procesamiento de lenguaje natural (chatbot), se realiza un análisis comparativo con el trabajo titulado “*Desarrollo de un sistema básico de robótica de entretenimiento persuasivo basado en el sistema humanoide NAO*”, donde en el apartado *Trabajos futuros* se afirma lo siguiente:

La visión de este proyecto a futuro se enfoca en el desarrollo de un sistema de diálogos más avanzado que contenga tanto una mayor cantidad de vocabulario,

terminologías, conceptos, reglas, y expresiones idiomáticas que permitan ampliar los rangos de respuestas que se den por parte de un robot e indexarlo a internet con el objetivo de obtener mayor cantidad de información y ante preguntas que no se contemplen dentro del sistema de diálogo, buscar e interpretar de manera autónoma posibles respuestas. (Vilatuña Salguero, 2018, pág. 87)

Al utilizar el servicio Dialogflow API, se ha cumplido con las expectativas planteadas ya que, gracias a este, la comunicación entre el usuario y el robot se ha vuelto más fluida, debido a la capacidad de la plataforma en la nube, de implementar, interpretar y procesar el lenguaje natural, en más de 30 idiomas y variantes compatibles.

Adicional la plataforma permite una retroalimentación continua de expresiones entregadas por el usuario, mediante la herramienta de entrenamiento y proporciona entidades del sistema predefinidas para extraer datos de categorías comunes como: colores, nombres, apellidos, fechas y horas, números, entre otras, lo que facilita el desarrollo del asistente virtual.

Es decir, mientras más interacciones del chatbot con diversos clientes, Dialogflow usa esos datos de entrenamiento a fin de compilar modelos de aprendizaje automático específicamente para el agente creado. Estos datos de entrenamiento consisten principalmente en intents, frases de entrenamiento y entidades a las que se hace referencia en un agente, que se usan con eficacia como etiquetas de datos de aprendizaje automático (Google Cloud, 2019).

En la Tabla 32 se resumen las diferencias entre las dos metodologías para la creación de diálogos en el robot.

**Tabla 32**

*Cuadro comparativo de la solución offline y online en la creación de diálogos*

	<b>Solución Offline</b>	<b>Solución Online</b>
<b>Acceso</b>	Local (en el robot)	Remoto (desde cualquier lugar)
<b>Obtención de información</b>	Implementación compleja	Fácil implementación
<b>Entrenamiento</b>	No	Si
<b>Utiliza procesamiento del lenguaje natural</b>	No	Si
<b>Costo</b>	Gratis	Gratis*

*Nota:* \* El servicio Dialogflow API es gratuito mientras se utilice la versión Dialogflow

Trial Edition.

## Capítulo V

### Conclusiones y Recomendaciones

#### Conclusiones

En el presente trabajo de titulación, se ha logrado trascender las limitaciones de hardware y software del robot NAO, así como la implementación de aplicativos avanzados orientados a las áreas de robótica social de educación, salud y atención al cliente, mediante la utilización de servicios en la nube. Esto, ha permitido dotar al robot humanoide NAO de nuevas funcionalidades como: la conversión de audio a texto, reconocimiento de imágenes, reconocimiento óptico de caracteres, agendamiento de eventos en Google Calendar, almacenamiento de información en base de datos en la nube, entre otras.

Para la selección de plataformas de servicios en la nube, se aplicó el Proceso de Análisis Jerárquico (AHP), el cual facilita la toma de decisiones mediante la evaluación de diferentes criterios. En este caso, los más relevantes fueron: cantidad de servicios, precios y número de trabajos realizados con el robot NAO. Finalmente, el proveedor seleccionado fue Google Cloud Platform, el cual ha dado buenos resultados en el desarrollo de las aplicaciones planteadas gracias a las herramientas brindadas por la consola y la cantidad de información existente sobre la plataforma

Los algoritmos para la utilización de los servicios en la nube han sido plenamente desarrollados en Python, un lenguaje de programación libre, abierto y gratuito, disponible para insertar como bloque dentro de los programas de Choregraphe. Para el correcto funcionamiento de las aplicaciones, es importante verificar la versión de Python preinstalada en el robot ya que, dependiendo de esto, podrán o no surgir ciertas trabas al

momento de integrar los diferentes servicios en la nube disponibles en Google Cloud Platform.

Por ejemplo, en este caso, la versión del Python utilizada por el robot NAO (V 2.7.0) ha llegado al final de su vida útil el 1 de enero de 2020 lo que implica que ya no recibe soporte, no se crean nuevas librerías ni se realizan nuevos informes de errores, correcciones o cambios, lo que ha generado problemas al momento de instalar la librería *PyJWT*, compatible solo con versiones de Python mayores a 3.6. Es por ello que, para la generación del token JWT utilizado en la tercera aplicación fue necesario el desarrollo de un programa independiente, ejecutable en una PC, en Python 3.9.

Sin embargo, a pesar de estos inconvenientes relacionados con los procesos de autenticación para la ejecución de ciertos servicios en la nube, el desempeño general de las aplicaciones ha sido favorable, lo que demuestra la importancia y eficiencia de la integración de APIs Restful de Google Cloud, mediante métodos HTTP, la utilización de endpoints (URI) como identificador único de cada recurso y el formato de intercambio de datos JSON.

Desde un inicio, se han planteado los servicios en la nube a ser utilizados para cada aplicativo, sin embargo, gracias a las diferentes opciones disponibles en Google Cloud Platform, se han podido agregar adicionales como: base de datos en la nube en tiempo real (Firebase) o asistentes virtuales (Dialogflow), que permitan complementar la funcionalidad de los aplicativos, facilitando su uso y el acceso de la información recopilada a los potenciales usuarios.

Para conocer el impacto de los servicios en la nube, en la ejecución de tareas específicas del robot, se realizó una comparativa con trabajos de investigación previamente desarrollados en la Universidad de las Fuerzas Armadas – ESPE, en los

cuales se implementaron soluciones offline que utilizan los recursos locales disponibles del robot o necesitan de hardware adicional para su ejecución. Gracias a esto, se ha concluido que, los servicios en la nube permiten la optimización de recursos y cumplen con las expectativas planteadas por los autores, en el apartado de trabajos futuros.

No obstante, si bien las soluciones online mostradas en este documento han brindado mejores resultados que aquellas realizadas con los recursos locales del robot, previo a su utilización es importante evaluar las necesidades de las aplicaciones a desarrollar ya que, ambas opciones brindan tanto ventajas como desventajas, las cuales deberán ser consideradas por parte del desarrollador.

Adicional, se manifiesta que, la utilización de servicios no busca reemplazar ni demeritar los aplicativos propios del robot, sino está encaminado al trabajo conjunto entre estos dos, con el fin de facilitar la programación de rutinas en el robot y dotarlo de un gran número de nuevas habilidades.

Gracias al desarrollo de laboratorios teórico/prácticos, orientados los docentes y alumnos de la Universidad de las Fuerzas Armadas – ESPE, en el presente proyecto, se establecen bases de conocimiento valiosas para las futuras generaciones, interesadas en el mundo de la robótica en general. Esto, con el fin de facilitar el proceso de aprendizaje sobre el robot humanoide NAO, su funcionamiento, y capacidad en la ejecución de tareas.

Analizando y clasificando los servicios involucrados en cada aplicativo, se ha comprobado que, los tipos más óptimos para la robótica social son: software como servicio (SaaS) por medio de Dialogflow y Google Calendar, backend como servicio (BaaS) mediante Firebase y machine learning como servicio (MLAAS) a través de Cloud Vision y Speech-to-Text, gracias a su facilidad de implementación ya que, el proveedor



se encarga de gestionar toda la infraestructura, el mantenimiento y los problemas de seguridad que pudieran surgir con las aplicaciones, y el desarrollador solo se encarga de consumir dichos servicios, bajo un modelo “on demand” de pago por consumo.

Cabe resaltar que, una de las ventajas principales de trabajar con API REST, es la independencia de plataformas. Es decir, el cliente puede consumirlas sin necesidad de una preinstalación de software dedicado o SDKs, e incluso, se adapta a todo tipo de sintaxis o lenguajes de programación con los que se desee trabajar, siempre y cuando se respete el lenguaje de intercambio de información (JSON) tanto en peticiones como en respuestas.

En el aplicativo de educación, gracias a las pruebas realizadas, se concluye que, para un correcto funcionamiento del servicio Cloud Vision API, las tarjetas a utilizar deben ser impresas a color, en alta calidad. Adicional, la ejecución del aplicativo debe realizarse utilizando luz puntual cerca del robot y el usuario, por ejemplo, una luminaria LED.

En el aplicativo de cuidados de la salud, se concluye que, el servicio SpeechToText API es una ventaja en el desempeño del robot NAO, esto debido a que, incluso en el ingreso de nombres complicados como los de medicamentos, la transcripción obtenida no contiene errores.

En el aplicativo de atención al cliente, se comprueba que, cuando existe una corta distancia entre el robot y el usuario, las grabaciones de voz realizadas captan correctamente la información que se desea transcribir, así el servicio SpeechToText API puede funcionar de manera óptima, independientemente de la presencia o ausencia de ruido ambiental.

## Recomendaciones

Debido a la emergencia sanitaria, no fue posible realizar pruebas con personas pertenecientes a los grupos de estudio previamente planteados, por lo cual se recomienda la ejecución de pruebas de los aplicativos diseñados, en un futuro, con un mayor número de participantes.

En el caso de los bloques de programación de Python, se recomienda la instalación de este programa en una PC, con la versión vigente que se encuentre en el robot NAO a trabajar (en este caso V 2.7.0), esto con el fin de realizar pruebas de funcionamiento y corrección de algoritmo, para obtener un código 100% funcional, evitando así posibles errores de compatibilidad al momento de integrarlo al robot.

En lo que a costos refiere, es importante consultar las cuotas mensuales gratuitas de los servicios en la nube a utilizar y monitorear que el número de solicitudes realizadas no sobrepasen este límite. Adicional, se recomienda utilizar la calculadora de precios de Google Cloud para cotizar el costo final de las aplicaciones, de acuerdo a su frecuencia de uso.

En los servicios de Cloud Vision y el bloque de programación de reconocimiento facial, se recomienda utilizar una buena iluminación para obtener imágenes más claras y facilitar el procesamiento de las mismas.

Para la utilización de servicios en la nube, es importante contar con una red sin restricciones en puertos o recursos ya que esto puede impedir el correcto funcionamiento de los algoritmos creados.

Es importante recordar que, los bloques de programación para el uso de las APIs de Google Cloud contienen información restringida solo para el desarrollador ya

que involucra claves de acceso encriptadas y cuentas de usuario que permiten asociar las solicitudes a la API con el proyecto, para las cuotas y facturación. Por ello, si se pretende exponer al público los algoritmos de programación, es necesario retirar esta información e instruir a otros desarrolladores en la creación de una cuenta personal en la plataforma y la generación de API Keys.

### **Trabajos Futuros**

Gracias a la gran cantidad de servicios disponibles en la plataforma Google Cloud, existe una incontable cantidad de aplicaciones que pueden desarrollarse, en conjunto con el robot, siempre y cuando, se tenga en consideración las capacidades propias de los recursos disponibles en el mismo.

Adicional, se propone el desarrollo de trabajos de investigación similares al presentado, utilizando otras plataformas como Amazon Web Services (AWS) y Microsoft Azure que permitan realizar una extensa comparativa entre los tres principales proveedores e identificar los servicios más relevantes y completos, a aplicarse en el campo de la robótica social.

En el caso de los aplicativos planteados, se pueden agregar bloques de programación orientados al movimiento autónomo del robot que complementen la ejecución de los mismos, de una manera más natural, aportando así una mejora significativa en la experiencia de usuario y la interacción humano-robot. Por ejemplo, en la aplicación de salud, desarrollar un sistema de clasificación, reconocimiento y entrega de medicamentos basado en la información recopilada de los eventos agendados por el personal y los recordatorios obtenidos por el robot de Google Calendar.

### Referencias Bibliográficas

- Achig Ortiz, Á. O., & Lasluisa Naranjo, J. C. (Marzo de 2017). *Desarrollo de algoritmos para modelado de rutinas de ejercicios para rehabilitación física en mejora de la calidad y esperanza de vida de adultos mayores implementados en robot humanoide Nao*. Recuperado el 12 de Junio de 2019, de <https://dspace.ups.edu.ec/handle/123456789/14031>
- Ahlawat, A. (2020). *Understanding HTTP: The Backbone of REST*. Recuperado el 4 de Abril de 2020, de Study Tonight: <https://www.studytonight.com/rest-web-service/understanding-http>
- Aldebaran. (2018). *Motherboard*. Recuperado el 26 de Mayo de 2019, de Aldebaran Documentation: [http://doc.aldebaran.com/2-1/family/robots/motherboard\\_robot.html](http://doc.aldebaran.com/2-1/family/robots/motherboard_robot.html)
- Aldebaran. (2018). *NAOqi Framework*. Recuperado el 31 de Julio de 2019, de <http://doc.aldebaran.com/1-14/dev/naoqi/index.html#naoqi-framework-overview>
- Aldebaran Robotics. (9 de Diciembre de 2011). *NAOqi Framework*. Recuperado el 24 de Abril de 2020, de NAO Software 1.14.5 documentation: <http://doc.aldebaran.com/1-14/dev/naoqi/index.html>
- Aldebaran Robotics. (5 de Junio de 2014). *Box connectors and parameters*. Recuperado el 21 de Abril de 2020, de Box: <http://doc.aldebaran.com/2-1/software/choregraphe/objects/box.html>
- Aldebaran Robotics. (6 de Octubre de 2014). *Contact and Tactile Sensors*. Recuperado el 18 de Junio de 2020, de NAO Software 1.14.5 documentation: [http://doc.aldebaran.com/1-14/family/robots/contact-sensors\\_robot.html](http://doc.aldebaran.com/1-14/family/robots/contact-sensors_robot.html)

Aldebaran Robotics. (6 de Octubre de 2014). *Inertial Unit*. Recuperado el 19 de Junio de 2020, de NAO Software 1.14.5 documentation: [http://doc.aldebaran.com/1-14/family/robots/inertial\\_robot.html](http://doc.aldebaran.com/1-14/family/robots/inertial_robot.html)

Aldebaran Robotics. (6 de Octubre de 2014). *LEDs*. Recuperado el 19 de Junio de 2020, de NAO Software 1.14.5 documentation: [http://doc.aldebaran.com/1-14/family/robots/leds\\_robot.html](http://doc.aldebaran.com/1-14/family/robots/leds_robot.html)

Aldebaran Robotics. (6 de Octubre de 2014). *Loudspeakers*. Recuperado el 19 de Junio de 2020, de NAO Software 1.14.5 documentation: [http://doc.aldebaran.com/1-14/family/robots/loudspeaker\\_robot.html](http://doc.aldebaran.com/1-14/family/robots/loudspeaker_robot.html)

Aldebaran Robotics. (6 de Octubre de 2014). *Microphones*. Recuperado el 19 de Junio de 2020, de NAO Software 1.14.5 documentation: [http://doc.aldebaran.com/1-14/family/robots/microphone\\_robot.html](http://doc.aldebaran.com/1-14/family/robots/microphone_robot.html)

Aldebaran Robotics. (6 de Octubre de 2014). *Motors*. Recuperado el 19 de Junio de 2020, de NAO Software 1.14.5 documentation: [http://doc.aldebaran.com/1-14/family/robots/motors\\_robot.html](http://doc.aldebaran.com/1-14/family/robots/motors_robot.html)

Aldebaran Robotics. (6 de Octubre de 2014). *Video Camera*. Recuperado el 19 de Junio de 2020, de NAO Software 1.14.5 documentation: [http://doc.aldebaran.com/1-14/family/robots/video\\_robot.html](http://doc.aldebaran.com/1-14/family/robots/video_robot.html)

Alonso, G., Casati, F., Kuno, H., & Machiraju, V. (2004). *Web Services*. Recuperado el 6 de Abril de 2020, de SpringerLink: [https://link.springer.com/chapter/10.1007/978-3-662-10876-5\\_5](https://link.springer.com/chapter/10.1007/978-3-662-10876-5_5)

Bernardo, D. (28 de Junio de 2019). *Introduction to JSON and RESTful APIs (coding bootcamp series)*. Recuperado el 6 de Abril de 2020, de Noteworthy - The

Journal Blog: <https://blog.usejournal.com/introduction-to-json-and-restful-apis-coding-bootcamp-series-7ad6aa294c89>

- Bonaccorsi, M., Fiorini, L., Cavallo, F., Alessandro, S., & Dario, P. (2016). A Cloud Robotics Solution to Improve Social Assistive Robots for Active and Healthy Aging. *International Journal of Social Robotics*, 8(3), 393-408. Obtenido de <https://link.springer.com/article/10.1007/s12369-016-0351-1>
- Bouziane, R., Terrisa, L. S., & Brethe, J.-F. (2017). A Web services based solution for the NAO Robotin Cloud Robotics environment. *4th International Conference on Control, Decision and Information Technologies (CoDIT'17)*. Barcelona. Obtenido de <https://ieeexplore.ieee.org/document/8102694>
- Breazeal, C. (4 de Mayo de 2004). Social Interactions in HRI: The Robot View. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 34(2). Recuperado el 10 de Octubre de 2019, de <https://ieeexplore.ieee.org/document/1291665>
- Breazeal, C., Takanishi, A., & Kobayashi, T. (2008). *Social Robots that Interact with People*. Recuperado el 29 de Agosto de 2019, de [https://link.springer.com/referenceworkentry/10.1007%2F978-3-540-30301-5\\_59](https://link.springer.com/referenceworkentry/10.1007%2F978-3-540-30301-5_59)
- Cadena Castro, L. M., & Heredia López, J. A. (Enero de 2018). *Sistema inteligente con visión artificial para el reconocimiento de piezas mecánicas en el Robot NAO*. Recuperado el 12 de Junio de 2019, de <https://dspace.ups.edu.ec/handle/123456789/15012>
- Calvopiña Iglesias, F. R., & Valladares Romero, P. E. (Febrero de 2017). *Interpretación de expresiones faciales en adultos mayores utilizando la visión artificial del robot*

*humanoide NAO*. Recuperado el 12 de Junio de 2019, de  
<https://dspace.ups.edu.ec/handle/123456789/14020?locale=es>

Citelia. (26 de Marzo de 2020). *¿Qué es y para qué sirve el Cloud Computing?*  
Recuperado el 13 de Junio de 2019, de <https://citelia.es/blog/que-es-cloud-computing-y-como-funciona/>

Clever Techie. (14 de Febrero de 2017). *REST API & RESTful Web Services Explained | Web Services Tutorial*. Recuperado el 6 de Abril de 2020, de Youtube:  
[https://www.youtube.com/watch?v=LooL6\\_chvN4](https://www.youtube.com/watch?v=LooL6_chvN4)

Codecademy. (2020). *What is REST?* Recuperado el 2020 de Abril de 2, de  
<https://www.codecademy.com/articles/what-is-rest>

Cyberbotics. (25 de Marzo de 2019). *NAOqi enabled controller for simulated NAO robots in Webots*. Recuperado el 25 de Febrero de 2020, de GitHub:  
<https://github.com/cyberbotics/naoqisim>

Data Centric. (2 de Enero de 2019). *¿Qué es el Procesamiento del Lenguaje Natural y por qué es la revolución del futuro?* Recuperado el 13 de Junio de 2019, de  
<https://www.datacentric.es/blog/business/procesamiento-lenguaje-natural-revolucion-futuro/>

Dautenhahn, K. (1999). Robots as social actors: Aurora and the case of autism. *Proc. 3rd Int. Cognitive Technology Conf.*, (págs. 359-374). San Francisco. Obtenido de <https://www.semanticscholar.org/paper/ROBOTS-AS-SOCIAL-ACTORS%3A-AURORA-AND-THE-CASE-OF-Dautenhahn/e61957d9da1e63e8c2f3ad4044b56e56734eebf1>

Dev Dungeon. (10 de Julio de 2018). *Install gcc compiler on Windows with MSYS2 for C/C++*. Recuperado el 25 de Febrero de 2020, de Dev Dungeon:

<https://www.devdungeon.com/content/install-gcc-compiler-windows-msys2-cc>

Education First. (30 de Marzo de 2011). *EF - EPI Índice del EF English Proficiency*.

Recuperado el 5 de Marzo de 2020, de Education First:

<https://www.ef.com.ec/epi/>

Education First. (30 de Octubre de 2015). *Índice del EF English Proficiency en Ecuador*.

Recuperado el 5 de Marzo de 2020, de Education First:

<https://www.ef.com.ec/epi/regions/latin-america/ecuador/>

Eising, P. (7 de Diciembre de 2017). *What exactly is an API?* Recuperado el 2 de Abril

de 2020, de [https://medium.com/@perrysetgo/what-exactly-is-an-api-](https://medium.com/@perrysetgo/what-exactly-is-an-api-69f36968a41f)

[69f36968a41f](https://medium.com/@perrysetgo/what-exactly-is-an-api-69f36968a41f)

El Comercio. (29 de Enero de 2016). *"Nuka", la foca robot que ayuda a mejorar la salud*.

Recuperado el 12 de Junio de 2019, de El Comercio:

<https://www.elcomercio.com/guaifai/nuka-foca-robot-mejora-salud.html>

Escobar, G. (9 de Agosto de 2017). *El protocolo HTTP*. Recuperado el 5 de Abril de

2020, de Make it real: <https://blog.makeitreal.camp/el-protocolo-http/>

Euronews. (28 de Marzo de 2016). *Meet Little Casper, a robot designed to help children*

*suffering from cancer*. Recuperado el 10 de Octubre de 2019, de Euronews:

<https://www.euronews.com/2016/03/28/meet-little-casper-a-robot-designed-to-help-children-suffering-from-cancer>

Galarza Guerrero, D. S., & Llumiquinga Pumisacho, C. M. (Marzo de 2018).

*Teleoperación del robot Nao para la ejecución de tareas en espacios reducidos*



*usando el dispositivo kinect v2*. Recuperado el 12 de Junio de 2019, de  
<https://dspace.ups.edu.ec/handle/123456789/15287>

Gómez Vega, A. S., & Guerrero Rivera, O. F. (Marzo de 2016). *Análisis comparativo de algoritmos neuro computacionales biológicos para proceso cognitivo de la memoria y su aplicación en el robot Nao*. Recuperado el 12 de Junio de 2019, de  
<https://dspace.ups.edu.ec/handle/123456789/13083>

González Paredes, E. M. (Abril de 2013). *Repositorio Digital - Universidad Central del Ecuador*. Recuperado el 10 de Marzo de 2020, de  
<http://www.dspace.uce.edu.ec/bitstream/25000/1965/1/T-UCE-0010-307.pdf>

Goodrich, M., & Schultz, A. (2007). Human–Robot Interaction: A Survey. *Foundations and Trends in Human–Computer Interaction*, 1(3), 203-275. Recuperado el 10 de Octubre de 2019, de  
[https://www.researchgate.net/publication/220613473\\_Human-Robot\\_Interaction\\_A\\_Survey](https://www.researchgate.net/publication/220613473_Human-Robot_Interaction_A_Survey)

Google Cloud. (2019). *Calendars and Events*. Recuperado el 8 de Noviembre de 2019, de Google Calendar API:  
<https://developers.google.com/calendar/v3/reference/events>

Google Cloud. (2019). *Overview of the Calendar API*. Recuperado el 8 de Noviembre de 2019, de Google Calendar API: <https://developers.google.com/calendar/concepts>

Google Cloud. (30 de Enero de 2019). *Training*. Recuperado el 1 de Julio de 2021, de Dialogflow: <https://cloud.google.com/dialogflow/es/docs/training>

- Google Cloud. (2 de Agosto de 2019). *Usa claves de API*. Recuperado el 21 de Octubre de 2019, de Documentación Google Cloud:  
<https://cloud.google.com/docs/authentication/api-keys>
- Google Cloud. (16 de Septiembre de 2020). *Programa gratuito de Google Cloud*. Recuperado el 13 de Julio de 2021, de Google Cloud:  
<https://cloud.google.com/free/docs/gcp-free-tier#free-tier-usage-limits>
- Guizzo, E. (24 de Febrero de 2011). *Robots with their heads in the clouds*. Recuperado el 7 de Abril de 2020, de IEEE:  
<https://ieeexplore.ieee.org/abstract/document/5719709>
- Gupta, L. (2019). *Introduction to JSON*. Recuperado el 6 de Abril de 2020, de REST API Tutorial: <https://restfulapi.net/introduction-to-json/>
- Hoguin, L. (2018). *REST principles*. Recuperado el 4 de Abril de 2020, de NineNines:  
[https://ninenines.eu/docs/en/cowboy/2.7/guide/rest\\_principles/](https://ninenines.eu/docs/en/cowboy/2.7/guide/rest_principles/)
- Honda. (2018). *Asimo: El robot humanoide más avanzado del mundo*. Recuperado el 12 de Junio de 2019, de <https://www.honda.mx/asimo/>
- Hoyos, S. (14 de Noviembre de 2015). *Análisis decisión multicriterio: Razón de Consistencia Método AHP*. Recuperado el 15 de Mayo de 2020, de Youtube:  
[https://www.youtube.com/watch?v=qRr4swHmlOo&list=PLewZAippGPI23v0CFjGVDYBC\\_Tf9XUXJu&index=9](https://www.youtube.com/watch?v=qRr4swHmlOo&list=PLewZAippGPI23v0CFjGVDYBC_Tf9XUXJu&index=9)
- Hrynowski, Z. (23 de Abril de 2020). *Increased Use of Low-Contact Services May Prove Permanent*. Recuperado el 17 de Diciembre de 2020, de Gallup:  
<https://news.gallup.com/poll/309203/increased-low-contact-services-may-prove-permanent.aspx>

- Ibáñez Romero, G. (28 de Junio de 2017). *Chatbots on Cloud*. Recuperado el 2 de Mayo de 2020, de [http://cybercom\\_ep10.episerverhosting.com/innovation-zone/blogs-innovation-zone/innovation-zone-blog/chatbots-on-cloud/](http://cybercom_ep10.episerverhosting.com/innovation-zone/blogs-innovation-zone/innovation-zone-blog/chatbots-on-cloud/)
- IDMind. (Enero de 2016). *MOnarCH*. Recuperado el 10 de Octubre de 2019, de IDMind: <https://www.idmind.pt/research/concludedprojects/>
- IntelliPaat. (25 de Junio de 2019). *AWS vs Azure vs Google - Detailed Cloud Comparison*. Recuperado el 15 de Agosto de 2020, de IntelliPaat: <https://intellipaas.com/blog/aws-vs-azure-vs-google-cloud/>
- Jayawardena, C., Kuo, I., Aleksandar, I., & Datta, C. (2011). *Feasibility study of a robotic medication assistant for the elderly*. Auckland: University of Auckland. Recuperado el 14 de Marzo de 2020, de <https://dl.acm.org/doi/10.5555/2460616.2460623>
- Jibo, Inc. (2019). *Jibo*. Recuperado el 10 de Octubre de 2019, de Jibo: <https://www.jibo.com/>
- Juviler, J. (27 de Julio de 2020). *REST APIs: How They Work and What You Need to Know*. Recuperado el 21 de Julio de 2021, de HubSpot: <https://blog.hubspot.com/website/what-is-rest-api>
- Kanda, T., & Ishiguro, H. (2013). *Human-Robot Interaction in Social Robotics*. CRC Express. Recuperado el 17 de Octubre de 2019, de [https://www.researchgate.net/publication/305491476\\_Human-Robot\\_Interaction\\_in\\_Social\\_Robotics](https://www.researchgate.net/publication/305491476_Human-Robot_Interaction_in_Social_Robotics)
- Koken, B., & Gyula, M. (2015). *The evolution of Cloud Robotics: A survey*. ResearchGate. Obtenido de

[https://www.researchgate.net/publication/333089539\\_THE\\_EVOLUTION\\_OF\\_CLOUD\\_ROBOTICS\\_A\\_SURVEY](https://www.researchgate.net/publication/333089539_THE_EVOLUTION_OF_CLOUD_ROBOTICS_A_SURVEY)

KPMG. (2016). *Social Robots*. Países Bajos: KPMG. Obtenido de <https://assets.kpmg/content/dam/kpmg/pdf/2016/06/social-robots.pdf>

Kumar, S., Chahal, V., & Hosurmath, M. (26 de Octubre de 2017). *Robotic Calculations and Inference Agent*. Recuperado el 2 de Mayo de 2020, de IBM Developer: <https://developer.ibm.com/patterns/robotic-calculations-and-inference-agent/>

Labra Gayo, J. E. (2016). *Servicios Web. Antecedentes y Justificación SOAP, WSDL, UDDI Utilización de Servicios Web Creación de Servicios Web*. Recuperado el 6 de Abril de 2020, de DocPlayer: <https://docplayer.es/6438559-Servicios-web-antecedentes-y-justificacion-soap-wsdl-uddi-utilizacion-de-servicios-web-creacion-de-servicios-web.html>

Leading Edge. (20 de Septiembre de 2015). *What are the Types of Cloud Computing?* Recuperado el 21 de Julio de 2021, de <https://www.leadingedgetech.co.uk/it-services/it-consultancy-services/cloud-computing/what-are-the-types-of-cloud-computing/>

Lew, E. (5 de Mayo de 2020). *Pandemic and the Smarter World: A Future of Robots?* Recuperado el 19 de Diciembre de 2020, de Columbia Business School: <https://www8.gsb.columbia.edu/articles/brand-talk/pandemic-and-smarter-world-future-robots>

Lowe's Innovation Labs. (30 de Agosto de 2016). *A Helping Hand*. Recuperado el 28 de Diciembre de 2020, de Lowe's Innovation Labs: <https://www.lowesinnovationlabs.com/lowebot/>

Matthews, K. (26 de Abril de 2020). *Pandemic proves utility of a wide range of service robots*. Recuperado el 17 de Diciembre de 2020, de The robot report:

<https://www.therobotreport.com/pandemic-proves-utility-wide-range-service-robots/>

Microsoft. (18 de Febrero de 2018). *Windows 10 | Variables de entorno windows 10 version 1709*. Recuperado el 25 de Febrero de 2020, de

<https://answers.microsoft.com/es-es/windows/forum/all/windows-10-variables-de-entorno-windows-10-version/703ea5fa-1db4-46da-8ff7-6261140bf58b>

Microsoft Azure. (21 de Noviembre de 2015). *¿Qué es PaaS? - Plataforma como servicio*. Recuperado el 21 de Julio de 2021, de <https://azure.microsoft.com/es-es/overview/what-is-paas/>

Microsoft Azure. (21 de Noviembre de 2015). *¿Qué es SaaS? - Software como servicio*. Recuperado el 21 de Julio de 2021, de <https://azure.microsoft.com/es-es/overview/what-is-saas/>

Microsoft Azure. (2020). *¿Qué es Azure?* Recuperado el 2 de Mayo de 2020, de Microsoft Azure: <https://azure.microsoft.com/es-mx/overview/#intro-to-cloud-computing>

Montalvo López, M. E. (Abril de 2017). *Programación por demostración del robot Aldebaran NAO H25 de la Universidad Politécnica Salesiana para la enseñanza y preservación de tradiciones, leyendas y expresiones orales del Ecuador*.

Recuperado el 12 de Junio de 2019, de

<https://dspace.ups.edu.ec/handle/123456789/14159>

- Mozilla. (23 de Febrero de 2020). *HTTP response status codes*. Recuperado el 23 de Marzo de 2020, de <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>
- MuleSoft. (19 de Junio de 2015). *What is an API? (Application Programming Interface)*. Recuperado el 5 de Abril de 2020, de MuleSoft:  
<https://www.mulesoft.com/resources/api/what-is-an-api>
- Navas Reascos, G. E., & Rivera Bonifaz, C. E. (Agosto de 2017). *Desarrollo de un sistema para administrar medicamentos a adultos mayores a través del reconocimiento de código de barras y su implementación en el robot Nao*. Recuperado el 12 de Junio de 2019, de  
<https://dspace.ups.edu.ec/handle/123456789/14534>
- NixCraft. (26 de Marzo de 2017). *How to unzip a zip file using the Linux and Unix bash shell terminal*. Recuperado el 25 de Febrero de 2020, de  
<https://www.cyberciti.biz/faq/how-to-unzip-a-zip-file-using-the-linux-and-unix-bash-shell-terminal/>
- Özlü, A. (29 de Julio de 2018). *Mastering REST Architecture — REST Architecture Details*. Recuperado el 3 de Abril de 2020, de Medium:  
<https://medium.com/@ahmetozlu93/mastering-rest-architecture-rest-architecture-details-e47ec659f6bc>
- Paraskevopoulos, K. (s.f.). *The Analytic Hierarchy Process*. Recuperado el 15 de Mayo de 2020, de International Hellenic University:  
[http://rad.ihu.edu.gr/fileadmin/labsfiles/decision\\_support\\_systems/lessons/ahp/AHP\\_Lesson\\_1.pdf](http://rad.ihu.edu.gr/fileadmin/labsfiles/decision_support_systems/lessons/ahp/AHP_Lesson_1.pdf)

- Pérez, A., Castro-González, Á. A.-M., & Castillo, J. C. (Septiembre de 2017). *Evolución de la robótica social y nuevas tendencias*. Recuperado el 6 de Junio de 2019, de [https://www.researchgate.net/publication/321332806\\_Evolucion\\_de\\_la\\_robotica\\_social\\_y\\_nuevas\\_tendencias](https://www.researchgate.net/publication/321332806_Evolucion_de_la_robotica_social_y_nuevas_tendencias)
- Petreanu, A., & Spinu, V. (1 de Agosto de 2019). *A.I.B. August: Nao the robot on Google Cloud & CenterNet*. Recuperado el 2 de Mayo de 2020, de Meetup: <https://www.meetup.com/es/Artificial-Intelligence-Bucharest/events/263005898/>
- Pfizer. (2009). La adherencia al tratamiento: Cumplimiento y constancia para mejorar la calidad de vida. *III Foro Diálogos Pfizer-Pacientes* (pág. 2). Madrid: Pfizer. Recuperado el 14 de Marzo de 2020, de [https://www.pfizer.es/docs/pdf/asociaciones\\_pacientes/2009/FOROpfizer\\_2009.pdf](https://www.pfizer.es/docs/pdf/asociaciones_pacientes/2009/FOROpfizer_2009.pdf)
- PortalEducativo. (2018). *Lenguaje verbal, no verbal y paraverbal*. Recuperado el 17 de Octubre de 2019, de PortalEducativo: <https://www.portaleducativo.net/octavo-basico/179/Lenguaje-verbal-no-verbal-paraverbal>
- Programación Colombia. (4 de Noviembre de 2018). *Introducción: Diferencias entre servicios REST y SOAP, Tutorial en español*. Recuperado el 6 de Abril de 2020, de Youtube: <https://www.youtube.com/watch?v=VCPluX3VZpU>
- Red Hat. (2019). *¿Qué es una API?* Recuperado el 26 de Octubre de 2019, de Red Hat: <https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces>

- RedHat. (13 de Octubre de 2019). *¿Qué son los servicios de nube?* Recuperado el 21 de Julio de 2021, de RedHat: <https://www.redhat.com/es/topics/cloud-computing/what-are-cloud-services>
- RedHat. (3 de Julio de 2020). *Diferencias entre IaaS, PaaS y SaaS.* Recuperado el 21 de Julio de 2021, de <https://www.redhat.com/es/topics/cloud-computing/iaas-vs-paas-vs-saas>
- Reeves, B., & Nass, C. I. (1996). *The media equation: How people treat computers, television, and new media like real people and places.* Chicago, IL: Center for the Study of Language and Information. Recuperado el 10 de Octubre de 2019, de [https://www.researchgate.net/publication/37705092\\_The\\_Media\\_Equation\\_How\\_People\\_Treat\\_Computers\\_Television\\_and\\_New\\_Media\\_Like\\_Real\\_People\\_and\\_Pla](https://www.researchgate.net/publication/37705092_The_Media_Equation_How_People_Treat_Computers_Television_and_New_Media_Like_Real_People_and_Pla)
- Rife, K. M., Ginty, S. E., Hohner, E. M., Stamper, H. R., Sobota, K. F., & Bright, D. R. (2012). *Remember Your MEDS: Medication Education Delivers Success.* Minnesota: Libraries Publishing. Recuperado el 14 de Marzo de 2020, de <https://pubs.lib.umn.edu/index.php/innovations/article/view/250/244>
- RoboEarth. (6 de Septiembre de 2016). *What is Cloud Robotics?* Recuperado el 21 de Abril de 2020, de [http://roboearth.ethz.ch/cloud\\_robotics/index.html](http://roboearth.ethz.ch/cloud_robotics/index.html)
- Rouse, M., Bedell, C., Hannan, E., & Wilson, S. (Junio de 2019). *RESTful API (REST API).* Recuperado el 6 de Abril de 2020, de Search App Architecture: <https://searchapparchitecture.techtarget.com/definition/RESTful-API>



- Saaty, T. (1990). *How to make a decision: The analytic hierarchy process*. Recuperado el 15 de Mayo de 2020, de <https://www.sciencedirect.com/science/article/abs/pii/0377221790900571>
- Sabharwal, N., & Gupta Edward, S. (2019). *Hands On Google Cloud SQL and Cloud Spanner: Deployment, Administration and Use Cases with Python*. India: Apress. Recuperado el 2 de Mayo de 2020, de <https://books.google.com.ec/books?id=67JEDwAAQBAJ&printsec=frontcover#v=onepage&q&f=false>
- Sam Ge, S. (2009). *International Journal of Social Robotics*. Springer. Obtenido de <https://link.springer.com/journal/12369>
- Samer, A. (26 de Agosto de 2017). *Difference Between API and Web Service*. Recuperado el 6 de Abril de 2020, de Medium: <https://medium.com/@programmerasi/difference-between-api-and-web-service-73c873573c9d>
- Seligman, J. (2017). *Artificial Intelligence / Machine Learning In Marketing*. Hampshire, Inglaterra: McGraw-Hill. Recuperado el 2 de Mayo de 2020, de <https://books.google.com.ec/books?id=l6zRDwAAQBAJ&printsec=frontcover#v=onepage&q&f=false>
- Siegel, H. (29 de Julio de 2015). *Japanese telcos vie for share in consumer robot-as-a-service business*. Recuperado el 12 de Junio de 2019, de <https://robohub.org/tag/sota/>
- Silva Freire, K. D. (2020). *Desarrollo de aplicaciones interactivas mediante robótica persuasiva para adultos mayores utilizando el robot humanoide NAO*. Sangolquí.

Recuperado el 2 de Julio de 2021, de  
<http://repositorio.espe.edu.ec:8080/handle/21000/22395>

Simbe Robotics. (30 de Octubre de 2018). *Say hello to Tally 3.0*. Recuperado el 28 de Diciembre de 2020, de Simbe Robotics:  
<https://www.simberobotics.com/platform/tally/>

SmartBear. (2020). *API Endpoints - What Are They? Why Do They Matter?* Recuperado el 6 de Abril de 2020, de AlertSite: <https://smartbear.com/learn/performance-monitoring/api-endpoints/>

SoftBank Robotics. (2006). *NAO*. Recuperado el 12 de Junio de 2019, de  
<https://www.softbankrobotics.com/emea/en/nao>

SoftBank Robotics. (2018). *Pepper*. Recuperado el 12 de Junio de 2019, de  
<https://www.softbankrobotics.com/emea/en/pepper>

SoftBank Robotics. (2020). *NAO*. Recuperado el 26 de Febrero de 2020, de  
<https://www.softbankrobotics.com/emea/en/nao>

Softbank Robotics. (2020). *What is Choregraphe*. Recuperado el 21 de Abril de 2020, de  
SOFTBANK ROBOTICS DOCUMENTATION: [http://doc.aldebaran.com/2-5/software/choregraphe/choregraphe\\_overview.html](http://doc.aldebaran.com/2-5/software/choregraphe/choregraphe_overview.html)

Stawarz, K., Cox, A. L., & Blandford, A. (2014). *Don't Forget Your Pill! Designing Effective Medication Reminder Apps That Support Users' Daily Routines*. Toronto. Recuperado el 14 de Marzo de 2020, de  
<https://dl.acm.org/doi/abs/10.1145/2556288.2557079>

Thijssen, J. (2016). *Are REST and HTTP the same thing?* Recuperado el 6 de Abril de 2020, de Rest Cookbook: <http://restcookbook.com/Miscellaneous/rest-and-http/>

Titin, S., & Rizkilillah, V. P. (4 de Julio de 2018). *Teaching vocabulary using flashcard*.

Recuperado el 10 de Marzo de 2020, de

[https://www.researchgate.net/publication/337251628\\_TEACHING\\_VOCABULARY\\_USING\\_FLASHCARD](https://www.researchgate.net/publication/337251628_TEACHING_VOCABULARY_USING_FLASHCARD)

Tutorials Point. (2020). *HTTP - Overview*. Recuperado el 5 de Abril de 2020, de

Tutorials Point: [https://www.tutorialspoint.com/http/http\\_overview.htm](https://www.tutorialspoint.com/http/http_overview.htm)

Underwood, C. (11 de Marzo de 2020). *Robots in Retail – Examples of Real Industry*

*Applications*. Recuperado el 19 de Diciembre de 2020, de Emerj:

<https://emerj.com/ai-sector-overviews/robots-in-retail-examples/>

Universidad de Alicante. (26 de Junio de 2014). *Introducción a los Servicios Web*.

*Invocación de servicios web SOAP*. Recuperado el 6 de Abril de 2020, de

Universidad de Alicante: <http://www.jtech.ua.es/j2ee/publico/servc-web-2012-13/sesion01-apuntes.html#%C2%BFQu%C3%A9+es+un+Servicio+Web%3F>

Valarezo Correa, J. E. (2019). Implementación de un sistema basado en el robot NAO

para lectura de textos en aplicaciones De Human Robot Interaction., (pág. 56).

Sangolquí. Recuperado el 1 de Julio de 2021, de

<http://repositorio.espe.edu.ec/handle/21000/20796>

Velasco Pumasunta, W. E. (Agosto de 2017). *Repositorio Digital Universidad Central del*

*Ecuador*. Recuperado el 10 de Marzo de 2020, de

<http://www.dspace.uce.edu.ec/bitstream/25000/13215/1/T-UCE-0010-045-2017.pdf>

- Vilatuña Salguero, F. S. (2018). *Desarrollo de un sistema básico de robótica de entretenimiento persuasivo basado en el sistema humanoide NAO*. Recuperado el 12 de Junio de 2019, de <https://repositorio.espe.edu.ec/handle/21000/14911>
- w3schools.com. (2019). *What is JSON?* Recuperado el 7 de Noviembre de 2019, de [https://www.w3schools.com/whatis/whatis\\_json.asp](https://www.w3schools.com/whatis/whatis_json.asp)
- Wan, J. (Enero de 2016). *Cloud Robotics: Current Status and Open Issues*. Recuperado el 24 de Abril de 2020, de ResearchGate: [https://www.researchgate.net/figure/System-architecture-of-cloud-robotics\\_fig1\\_303779070](https://www.researchgate.net/figure/System-architecture-of-cloud-robotics_fig1_303779070)
- Wikipedia. (9 de Enero de 2018). *HRP-4C*. Recuperado el 12 de Junio de 2019, de <https://es.wikipedia.org/wiki/HRP-4C>
- Wikipedia. (15 de Noviembre de 2018). *Kismet*. Recuperado el 12 de Junio de 2019, de [https://es.wikipedia.org/wiki/Robot\\_Kismet](https://es.wikipedia.org/wiki/Robot_Kismet)
- Wikipedia. (30 de Mayo de 2019). *Nao (robot)*. Recuperado el 12 de Junio de 2019, de [https://es.wikipedia.org/wiki/Nao\\_\(robot\)](https://es.wikipedia.org/wiki/Nao_(robot))
- Wikipedia. (14 de Enero de 2020). *Webots*. Recuperado el 25 de Febrero de 2020, de Wikipedia: <https://en.wikipedia.org/wiki/Webots>
- Wolters, M. K., Johnson, C., Campbell, P. E., DePlacido, C. G., & McKinstry, B. (2014). *Can older people remember medication reminders presented using synthetic speech?* United Kingdom: Oxford University Press. Recuperado el 14 de Marzo de 2020, de <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4433370/>

World Health Organization. (2003). *Adherence to Long-Term Therapies: Evidence for action*. Suiza: World Health Organization. Recuperado el 14 de Marzo de 2020, de <https://apps.who.int/iris/bitstream/handle/10665/42682/9241545992.pdf>

YouCleverMonkey.com. (2015). *Dear Zoo - Play ideas and printables for preschool*. Recuperado el 14 de Mazro de 2020, de You Clever Monkey: <https://www.youclevermonkey.com/2014/03/dear-zoo.html>

Zhou, Z. (2016). *The NAO robot as a personal assistant*. Recuperado el 2 de Mayo de 2020, de University of Applied Sciences Vaasan Ammattikorkeakoulu: [https://www.theseus.fi/bitstream/handle/10024/109398/Zhou\\_Ziye.pdf?sequence=1&isAllowed=y](https://www.theseus.fi/bitstream/handle/10024/109398/Zhou_Ziye.pdf?sequence=1&isAllowed=y)

**Anexos**