Universidad Autónoma de Querétaro

Facultad de Informática

Maestría en Sistema Computacionales

Interfaz de lenguaje natural a base de datos usando la API de reconocimiento de voz y conversión a texto DialogFlow

Tesis

Que como parte de los requisitos para obtener el Grado de
Maestro en Sistemas Computacionales

Presenta
Julio Alejandro Villeda Maldonado

Dirigido por:
M.C. José Arturo Gaona Cuadra

M.C. José Arturo Gaona Cuadra Presidente	_____ Firma
Dr. Alberto Lara Guevara Secretario	_____ Firma
M.S.I. Diego Octavio Ibarra Corona Vocal	_____ Firma
Dra. Ma. Teresa García Ramírez Suplente	_____ Firma
Dr. Ricardo Chaparro Sánchez Suplente	_____ Firma

Centro Universitario, Querétaro, Qro.
Fecha de aprobación por el Consejo Universitario (Octubre, 2020)
México

RESUMEN

En el presente trabajo se presenta una interfaz de lenguaje natural a base de datos utilizando la API de reconocimiento de voz y conversión a texto DialogFlow. El lenguaje natural es el lenguaje que las personas utilizan para comunicarse diariamente, tales como el español o el inglés. La interfaz de lenguaje natural provee los medios para interactuar con las computadoras utilizando el lenguaje natural. En este trabajo se explora una de las áreas más activos en el campo de la investigación de las ciencias computacionales. El procesamiento de lenguaje natural está teniendo mayor auge en los últimos años para manejar la interacción entre las bases de datos y los usuarios. La capacidad de los usuarios para recuperar información almacenada en bases de datos está limitada por la falta de entendimiento de la sentencia SELECT y todas las cláusulas que la pueden componer. Si un usuario no tiene el conocimiento suficiente sobre esta sentencia no puede recuperar información de la base de datos. Es por esto que surge la necesidad de brindar medios a los usuarios que les facilite la interacción y obtención de datos con las bases de datos relacionales utilizando el lenguaje natural. Los usuarios sin experiencia en lenguajes de bases de datos se habilitarán para recuperar información con esta interfaz; en tanto que los usuarios con experiencia reducirán el tiempo de creación de consultas. La interfaz que aquí se presenta fue desarrollada bajo la premisa de que el usuario se exprese en cualquier forma que sea natural para él y la interfaz le dé sentido a su sentencia de entrada. La arquitectura con la que se desarrolló esta interfaz es el emparejamiento de patrones, donde algunas reglas prestablecidas se definieron usando DialogFlow. Posteriormente, se emparejó la consulta que el usuario realizó en lenguaje natural con dichas reglas. La interfaz habilita dos formas de ingresar la consulta en lenguaje natural, con voz y texto. La interfaz tuvo un 72% de confiabilidad utilizando simplemente la voz, y alcanzó el 100% de confiabilidad en modo texto.

(**Palabras clave**: interfaz, base de datos, lenguaje natural, aprendizaje automatizado)

SUMMARY

A natural language interface to database using the DialogFlow voice recognition and text conversion API is presented in this work. Natural language is the language that people use to communicate daily such as Spanish or English. The natural language interface provides people the means to interact with computers using natural language. This paper explores one of the most active fields of the computational science research, which is having a major boom in recent years, Natural language processing handles the interaction between databases and users. The user's ability to retrieve information stored in databases is limited by the lack of understanding of the SELECT statement and the clauses that can make it up. If a user does not have sufficient knowledge about this statement, he is unable to obtain information from the database. This is why the need arises to provide a mechanism to users that facilitates the interaction and data acquisition with the databases using a natural language. Users without experience in database languages will be enabled to retrieve information with this interface; while experienced users will reduce query creation time. The interface presented here was developed under the premise that the user expresses himself in any way that is natural to him and the interface makes sense of his entry statement. This interface was developed using the architecture pairing of patterns, where some preset rules were defined using DialogFlow. Subsequently, the query that the user made in natural language was matched with those rules. The interface enables two ways to enter the query in natural language, with voice and text. The interface had 72% reliability by simply using the voice, and reached 100% reliability in text mode.

(**Key words:** interface, database, natural language, machine learning)

DEDICATORIA

Dedico este trabajo a mis padres por haberme forjado como la persona que soy en la actualidad. Muchos de mis logros se los debo a ellos, entre los que se incluye éste. Me formaron con reglas y algunas libertades, pero al final, me brindaron una motivación constante para alcanzar mis metas.

AGRADECIMIENTOS

En primera instancia, quiero extender un agradecimiento a los académicos que participaron en la realización de esta tesis, entre ellos, la Dra. Ma. Teresa García Ramírez, quien durante los dos primeros semestre estuvo dando seguimiento constante a la entrega de documentos y avance de tesis. Al profesor Ricardo Chaparro Sánchez quien durante segundo semestre me impulsó de manera significativa para lograr el desarrollo del artículo que deriva de esta tesis, así como al Dr. Alberto Lara Guevara que a inicios del tercer semestre me brindó su tiempo y apoyo para concretar el envío del artículo a un congreso internacional de IEEE.

Quiero extender un agradecimiento especial a mi asesor el M.C. José Arturo Gaona Cuadra, quien durante cuatro semestres estuvo apoyándome en la realización de este trabajo. En diferentes sesiones me aportó ideas de gran valor para desarrollar dicho proyecto.

TABLA DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

En los últimos años el campo de las Tecnologías de Información (TI) ha crecido a un ritmo acelerado y sus áreas de aplicaciones se han vuelto cada vez más amplias, estableciendo la base de todo tipo de aplicaciones, tanto de carácter básico como de naturaleza crucial o de alto impacto. Cada uno de estos sistemas tienen sus propios principios de interacción entre sus componentes, pero casi todos ellos comparten como punto crucial el almacenamiento de datos. Sin embargo, a través de los años se ha identificado una carencia en la recuperación de información derivado de la falta de entendimiento del Lenguaje de Consultas Estructurado (SQL del inglés *Structured Query Language*) por lo que en este trabajo se presenta una interfaz que permite tanto a usuarios con experiencia como a usuarios sin experiencia en la sentencia SELECT de SQL, poder interactuar con los datos almacenados en bases de datos relacionales, particularmente para recuperar información empleando el lenguaje natural sin necesidad de que los usuarios tengan dominio del lenguaje de consultas SQL.

En este trabajo se explora una alternativa a las Interfaces de Lenguaje Natural a Base de Datos (ILNBD) que existen al día de hoy. La interfaz que aquí se presenta está realizada con la API de reconocimiento de voz y conversión a texto DialogFlow. Con esta API la interfaz ofrece un amplio Entendimiento del Lenguaje Natural (ELN) o NLU por sus siglas en inglés (*Natural Language Understanding*), siendo incluso multilenguaje. El motor procesa y entiende sentencias de entradas en lenguaje natural, y al ser un analizador robusto, entiende y procesa las variantes del lenguaje natural. A través de entrenamiento y el uso de *Machine Learning* (ML), la interfaz cubre las carencias que han presentado otras interfaces en el pasado, particularmente las interfaces orientadas al dominio y la dificultad de mantener el diccionario de la base de datos. Con la interfaz que aquí se presenta, se genera un modelo de lenguaje más variado que empareja mejor la sentencia de entrada del usuario.

En este trabajo primero se hizo una breve revisión de los trabajos existentes en el área de las ILNBD, particularmente de los últimos cinco años, después se hizo una descripción del problema a desarrollar para presentar la fundamentación teórica que respalda el trabajo realizado. En seguida, se muestran los pasos realizados para el desarrollo de este proyecto y, por último, se presentan los resultados, conclusiones y trabajo futuro.

2. ANTECEDENTES

En esta sección se presenta un análisis de las ILNBD desarrolladas en los últimos cinco años, en donde se observa las interfaces propuestas, las técnicas empleadas, los resultados obtenidos y el trabajo por realizar en cada una de ellas.

Kumar & Dua (2014). propusieron un sistema para recuperar datos desde una base de datos usando el lenguaje hindú, integrando un analizador morfológico y un analizador de grupo de palabras para obtener las palabras principales de una consulta de entrada en leguaje hindú. En el trabajo se empleó la técnica *pattern matching* para encontrar las palabras clave y se empleó una interfaz del lenguaje hindú controlado con algunas características de consultas sugeridas para reducir la ambigüedad y reducir el tiempo de entrada de la consulta hindú. En este sistema se empleó un diccionario orientado al dominio para determinar el tipo de palabras claves usadas en una consulta de entrada y se presentó una importante limitación, la cual radicó en la dificultad de mantener la base de datos y el diccionario orientado al dominio cuando la interfaz se aplicaba a base de datos muy grandes. Dentro de los trabajos a futuro de esta investigación se buscó el desarrollo de una interfaz independiente de su dominio y además que dicha interfaz fuera multilenguaje.

Posevkin & Bessmertny (2015) desarrollaron un prototipo de interface de usuario de lenguaje natural a consultas SQL para base de datos. Este prototipo requiere la estructura de datos describiendo la base de datos, en particular, requiere información acerca de las tablas y los nombres de los campos. La interfaz tiene algunas restricciones, tal como no usar pronombres personales ni demostrativos y se recomienda empezar las consultas con palabras especiales tales como: “obtener”, “mostrar”, “enumerar”, entre otras.

Mohite & Bhojane (2015) discutieron el diseño y la implementación de un sistema que utiliza el método de la matriz de co-ocurrencia de palabras modificadas para proveer acceso a una base de datos utilizando el idioma inglés. En este trabajo se emplearon varias técnicas, tales como: análisis sintáctico, extracción de palabras

clave, detener la eliminación de palabras, generación de la matriz de co-ocurrencia, el uso de *WordNet*, algoritmo de derivación y mapeo semántico. Este sistema que se desarrolló da respuestas correctas a consultas sencillas, consultas con condiciones lógicas y funciones de agregado, sin embargo, el sistema no soporta todas las formas de las consultas de SQL.

Del Rio & Castillo (2015) realizaron un traductor de lenguaje natural a SQL, integrando tanto el esquema de la base de datos como información del dominio, la cual se ingresa por medio de un archivo CSV (*comma-separated values*). El sistema propuesto utiliza las preposiciones, los artículos y principales adverbios del lenguaje español como parte de su análisis y se utiliza un algoritmo de aproximación, que de acuerdo con las palabras que se entienden de la entrada del usuario, se busca generar una consulta SQL. En dicha investigación se enfocaron en resolver la dificultad que tienen los usuarios no técnicos para acceder a la base de datos, implementando para ello una interfaz del lenguaje natural. Como resultado de las pruebas que se realizaron en dicha interfaz, se comprobó que el prototipo es de utilidad, pero es necesario mejorarlo enfocándose en incrementar la capacidad de generación de código SQL utilizando sentencias del tipo “GROUP BY”-“HAVING”, con funciones de SQL y consultas anidadas, incluso siendo deseable brindar una retroalimentación al usuario indicándole qué partes de su consulta no son precisas.

Bais, Machkour, & Koutti (2017) presentaron la arquitectura de una interfaz genérica para consultar bases de datos usando lenguaje arábigo. La interfaz que propusieron funciona independientemente del dominio de la base de datos y tiene la capacidad para mejorar a través de la experiencia su base de conocimiento. Dicha interfaz estaba basada en ML y Procesamiento de Lenguaje Natural (PLN). Este sistema ofreció algunas ventajas, primero que el sistema era independiente del lenguaje, dominio y modelo de la base de datos, además de que es capaz de mejorar automáticamente su base de conocimiento a través de la experiencia.

Uma, Sneha, Sneha, Bhuvana, & Bharathi (2019) presentaron la propuesta de una interfaz integrando el PLN. En dicha interfaz, se integran distintos pasos, tales como: generación de *tokens*, lematización, partes de etiquetado de voz, análisis sintáctico, análisis semántico y mapeo. Está interfaz logró un 98.89% de confiablidad, aunque sólo realiza el proceso mediante texto, todavía es necesario incorporar un sistema de entrada mediante voz, además de que hace falta probar aún más la interfaz con sentencias de entrada más complejas.

Zhong, Xiong, & Socher (2017).propusieron un sistema Seq2SQL, una red neuronal profunda para traducir preguntas del lenguaje natural a su correspondiente consulta en SQL. El modelo que se presentó aprovecha la estructura de las consultas de SQL para reducir significativamente el espacio de salida de las consultas generadas. Además, el modelo integra un sistema de recompensas sobre la base de datos para aprender una política para generar partes no ordenadas de la consulta. Este modelo es menos adecuado para la optimización a través de la pérdida de entropía cruzada. Es importante destacar que, como parte de este trabajo, se publicó *WikiSQL*, el cual es un conjunto de datos de 87,726 ejemplos anotados a mano de preguntas y consultas de SQL distribuidas a través de 26,375 tablas de Wikipedia. Este conjunto de datos fue utilizado para entrenar el modelo y mejoró la confiabilidad de ejecución del 35.9% al 60.3%, así como la confiabilidad de forma lógica del 23.4% al 49.2%, respecto a interfaces similares diseñadas previamente

Después de revisar los trabajos previos se identificó que las ILNBD aún no son 100% confiables. Algunas de ellas encuentran su principal limitante en el dominio de la base de datos, aunque, por otro lado, las investigaciones en el área de inteligencia artificial, tales como: redes neuronales, “*Pattern Matching*” y ML son bastantes promisorias, por lo que en este trabajo se busca explorar una alternativa en estas áreas con la API de interfaz conversacional DialogFlow.

3. OBJETIVOS

3.1 Objetivo General

Implementar una ILNBD usando la API de reconocimiento de voz y conversión de texto DialogFlow.

3.2 Objetivos Particulares

- Implementar una interfaz de usuario conversacional con DialogFlow que convierta las consultas de lenguaje natural a texto.
- Integrar los algoritmos de inteligencia artificial disponibles en DialogFlow tales como ML para procesar las sentencias de entrada y generar como salida consultas estructuradas SQL para Base de Datos Relacionales (BDR).
- Ejecutar las consultas generadas en DialogFlow en la base de datos de datos en SQL server y determinar el nivel de fiabilidad de dichas consultas.

4. MARCO TEÓRICO.

El marco teórico, que se desarrolla a continuación, presenta los conceptos básicos necesarios para el entendimiento del presente trabajo. Primero se parte de la definición y descripción de las bases de datos, el lenguaje de consultas estructurado SQL y el uso de la cláusula SELECT; después, se exploran los conceptos de PLN, las Interfaces de lenguaje natural, las ILNBD, y, por último, algunos ejemplos de estas interfaces para conocer las diferentes arquitecturas y sus componentes de cada una de ellas.

4.1 Bases de datos

Una base de datos es un conjunto de datos almacenados en un soporte informático y que puede ser accedido por distintos usuarios y aplicaciones de manera simultánea. Existen diferentes tipos de bases de datos (Wada, Watanabe, Syoubu, Sawamoto, & Katoh, 2010):

1) **De tipo jerárquico**: Este tipo de bases de datos se organizan como un árbol con un nivel de jerarquía y de punteros entre registros.
2) **De datos en red**: Utilizan un poco el modelo jerárquico, y permiten navegar entre los elementos y no sólo de forma descendente.
3) **No relacionales**: Este tipo de base de datos son aquéllas que no usan el esquema tabular de filas y columnas. En su lugar usan un modelo de almacenamiento que está optimizado para los requisitos específicos del tipo de datos que se almacena.
4) **Relacionales**: Los datos se organizan en tablas diferentes sin nivel de jerarquía, no hay punteros, pero los datos de las tablas permiten realizar vínculos entre las mismas. Este trabajo particularmente se enfoca en las BDR.

Según Rendón (2019), las bases de datos se componen de

1) **Datos**: Los datos son el componente principal y fundamental de una base de datos, los cuales deben estar estructurados y almacenados independientemente de las aplicaciones que la utilizan. Estos datos se encuentran interrelacionados entre sí, evitando redundancia. Los datos por sí mismos no aportan conocimiento, hay que procesarlos y transformarlos.
2) **Software Sistema de Gestión de Base de Datos (SGBD):** Un SGBD es un software o conjunto de programas que permiten crear y mantener una base de datos. El SGBD actúa como interfaz entre los programas de aplicación, los usuarios y el sistema operativo. El objetivo principal de un SGBD es proporcionar un entorno eficiente para almacenar y recuperar la información de la base de datos. Este software facilita el proceso de definir, construir y manipular base de datos para diversas aplicaciones.

   Definir una base de datos consiste en especificar los tipos de los datos, las estructuras de los datos y las restricciones de los datos.

   Construir una base de datos es el proceso de almacenar los datos en algún medio de almacenamiento controlado por el SGBD, una vez definida la base de datos.

   Manipular la base de datos es:

   - Consultar los datos para obtener cierta información.
   - Actualizar la base de datos (modificar, eliminar o introducir nuevos datos).
   - Generar reportes a partir de los datos almacenados.

3) **Usuarios:** Otro componente de una base de datos son los usuarios. Existen diferentes tipos de usuarios:
    a. Programadores de aplicación: Escriben programas de aplicación de alto nivel que acceden a la base de datos.
    b. Usuarios sin experiencia: Usuarios pocos experimentados que utilizan las aplicaciones escritas previamente.
    c. Usuarios con experiencia: Utilizan consultas formuladas en un lenguaje de consultas SQL de base de datos.
    d. Administrador de la base de datos (DBA por sus siglas en inglés *Data Base Administrator*): Es la persona o grupo de personas encargadas del control de sistema.

    Este trabajo se enfoca particularmente hacia los usuarios sin experiencia, así como a los usuarios con experiencia.

4.2 BDR

Este tipo de bases de datos se empezaron a utilizar a principios de los años 1980's. Están basadas en el trabajo de Edgard Codd, quien desarrolló un modelo relacional para IBM una década antes (Wei & Zhang, 2009).

Las BDR se basan en la organización de la información en partes pequeñas que se integran mediante identificadores filas, columnas, registros y se almacenan en tablas. Este tipo de bases de datos generalmente son más robustas, ya que tienen mayor capacidad de almacenamiento, y son menos vulnerables ante fallas. (Rendón, 2019)

El concepto de "relación" es el fundamento del modelo relacional. Dicho concepto permite relacionar los dominios de acuerdo a los criterios especificados, gestionando tres elementos: el objeto principal, la relación y un objeto secundario asociado.

A cada relación se aplican algunas reglas para respetar las restricciones vinculadas:

1) Coherencia: Cualquier valor que toma un atributo debe pertenecer al dominio sobre el que se ha definido.
2) Unicidad: Todos los elementos de una relación deben ser únicos e irrepetibles.
3) Identificador: Puede ser un atributo o un conjunto de atributos que permitirán dotar de una característica única a cada elemento de la relación.
4) Clave primaria: Es el identificador mínimo de una relación.
5) Claves secundarias: Pueden ser otros identificadores de la relación.
6) Integridad referencial: Esta regla impone que un atributo o un conjunto de atributos de una relación aparezcan como clave primaria en otra relación.
7) Clave foránea: Atributo o conjunto de atributos que cumplen la regla de integridad referencial.

8) Valor nulo: En el modelo relacional se admite el valor nulo, el cual es un valor que representa una información desconocida o inaplicable en alguna columna.
9) Restricción de entidad: Cualquier valor que sea parte de una llave primaria no puede ser un valor null.

4.3 SQL

SQL es la interfaz estándar de programa de usuario y aplicación a una BDR. Los comandos de este lenguaje se utilizan para varios objetivos, entre ellos, realizar consultas interactivas para obtener información almacenada en la base de datos, así como la recopilación de información para la generación de reportes. (Rendón, 2019)

SQL fue creado a principios de los años 70. Una empresa pequeña que recién iniciaba llamada Relational Software produjo la primera versión comercial en 1979. La primera versión normalizada de SQL por ANSI fue en 1986, actualmente la norma SQL2 o SQL92 es la más importante, sin embargo, la mayoría de los proveedores de manejadores de bases de datos han implementado a su manera el lenguaje SQL y además ha agregado sus propias extensiones. En este trabajo, se utilizó SQL server en su versión 2017.

Según Godoc (2014), SQL está dividido en varios subconjuntos:

1) Lenguaje de Definición de Datos (DDL por sus siglas en inglés *Data Definition Language*): Agrupa todos los comandos utilizados para crear, modificar o eliminar la estructura de la base de datos. Estos comandos interactúan con los objectos de la base de datos tales como: tablas, índices, vistas, funciones, desencadenadores, entre otros. Se trata principalmente de los comandos:
    - CREATE: Para crear nuevos objetos.
    - ALTER: Para modificar objetos existentes y,
    - DROP: Para eliminar objetos existentes.

2) Lenguaje de Control de Datos (DCL por sus siglas en ingles *Data Control Language*): Agrupa los comandos utilizados para administrar la seguridad de acceso a los datos. Se trata principalmente de los siguientes comandos:
    - GRANT: Para conceder permisos y,
    - REVOKE: Para retirar permisos.
3) Lenguaje de Control de Transacciones (TCL por sus siglas en inglés *Transaction Control Language*): Este subconjunto agrupa los comandos utilizados para administrar la confirmación o no de actualizaciones realizadas sobre la base de datos. Se trata principalmente de los siguientes comandos:
    - COMMIT y,
    - ROLLBACK
4) Lenguaje de Manipulación de Datos (DML por sus siglas en ingles *Data Manipulation Language*): Agrupa los comandos utilizados para manipular los datos contenidos en la base de datos, e incluye aquellas sentencias que sirven para manipular o procesar los datos, tales como:
    - INSERT: Para agregar nuevos datos
    - DELETE: Para borrar datos existentes
    - UPDATE: Para actualizar datos existentes y,
    - SELECT: Para recuperar datos almacenados previamente en las tablas. De estas cuatro sentencias, la que se estará abordando a detalle en este trabajo es la sentencia SELECT.

4.4 Algebra relacional

SQL está basado en el álgebra relacional y representa el estándar en lo que se refiere a gestión de datos. El algebra relacional es un método de extracción que permite la manipulación de tablas y columnas. Se basa en la creación de nuevas tablas, que se pueden denominar tablas resultantes, a partir de tablas que ya existen. Estas nuevas tablas se convierten en objetos que se utilizan de manera inmediata. Los operadores del álgebra relacional que permiten crear las tablas resultantes se basan en la teoría de conjuntos:

1) Unión: La unión entre dos tablas genera una nueva tabla que combina la estructura de las dos tablas originales y que tiene como elementos el conjunto de los diferentes elementos de las dos relaciones iniciales.
2) Intersección: La intersección entre dos relaciones de igual estructura da una tabla resultante de igual estructura que tiene como elementos el conjunto de elementos comunes a las dos relaciones iniciales.
3) Diferencia: Es el conjunto de elementos que están en el primer conjunto de elementos, pero no están presentes en el segundo conjunto.
4) División: Es el conjunto de elementos que cumplen para toda tupla en el primer conjunto y que existe su correspondiente en el segundo conjunto.
5) Restricción: La restricción se basa en una condición. A partir de una relación, se crea otra con la misma estructura que solo tenga los elementos de la relación inicial que responda a la condición.
6) Proyección: La proyección de una relación sobre un grupo de atributos da una relación que tiene como estructura sólo estos atributos, y como elementos n-tupla diferentes compuestos por los valores asociados de estos atributos.
7) Producto cartesiano: El producto cartesiano entre dos relaciones crea una relación que tiene como estructura todos los atributos de las dos relaciones existentes y como elementos la asociación de cada fila de la primera tabla con cada fila de la segunda tabla.

8) *Join*: La unión entre dos relaciones se produce por la restricción sobre el producto cartesiano.
9) Cálculos elementales: Son proyecciones que se realizan sobre una relación asociada a un cálculo en cada fila para crear uno o varios nuevos atributos. Las operaciones pueden ser:
    a. Una operación aritmética
    b. Una función matemática
    c. Una función sobre una cadena.
10) Cálculos agregados: Proyección sobre una relación asociada a uno o más cálculos estadísticos sobre un atributo para todos los elementos de la relación o del agrupamiento vinculado a la proyección para crear uno o más atributos nuevos. Las principales funciones estadísticas son:
    a. COUNT(*): Número total de filas
    b. COUNT(Atributo): Número total de valores no nulos.
    c. SUM(Atributo): Suma de los valores no nulos.
    d. AVG(Atributo): Promedio de los valores no nulos.
    e. MAX(Atributo): Valor máximo (no nulo).
    f. MIN(Atributo): Valor mínimo (no nulo)

4.5 Cláusula SELECT

Como se mencionó previamente, recuperar información de las BDR es una tarea que requiere por lo menos del dominio de la sentencia SELECT del SQL, además de tener un claro entendimiento del algebra relacional. En la sección anterior se presentó algunos conceptos del álgebra relacional, y en esta sección se define la estructura de la sentencia SELECT. La sintaxis completa de esta sentencia es compleja, pero se puede resumir de la siguiente manera:

SELECT select_list

[FROM table_source]

[WHERE search_condition]

[GROUP BY group_by_expression]

[HAVING search_condition]

[ORDER BY order_expression [ASC|DESC]]

Debido a la complejidad de la sentencia SELECT, se describe cada una de las cláusulas que la componen, incluyendo algunas de sus variantes:

1) **SELECT**: Determina el número y nombre de las columnas a incluir en la consulta. A continuación, se pueden observar las variantes de la cláusula SELECT:

   - **DISTINCT**: Específica que el conjunto de resultados solo puede incluir filas únicas. Los valores nulos se consideran iguales desde el punto de vista de esta palabra reservada.

   - **TOP(Expresión)**: Indica que el conjunto de resultados de la consulta solamente deberá incluir un primer conjunto o porcentaje de filas

especificado. Por lo tanto, “expresión” puede ser un número o un porcentaje del total de las filas.

- **“*” (Asterisco)**: Específica que se deben devolver todas las columnas de todas las tablas del origen de datos. Las columnas serán mostradas en el orden en el que se encuentren en el origen de datos.

- **<SELECT LIST>**: Se específica una o varias columnas a mostrarse como parte del conjunto de datos contenidos en las tablas o vistas seleccionadas como origen de datos. Esta lista de columnas está separada por comas y máximo se podrán seleccionar 4,096 columnas.

2) **FROM**: Identifica las tablas de origen de los datos, puede ser una sola tabla o múltiples tablas. Generalmente, está cláusula es requerida en la instrucción SELECT; sin embargo, cuando no hay columnas de tablas enumeradas, y los únicos elementos que se muestran son las literales, variables o expresiones aritméticas, se omite la cláusula FROM. Al usar la cláusula FROM se puede usar la cláusula JOIN para indicar que se va a ejecutar la operación de combinación especificada entre los orígenes de la tabla o vistas indicadas y tener como resultado una tabla combinada. Se pueden tener diferentes tipos de combinaciones tales como INNER, FULL, LEFT, RIGHT y CROSS.

3) **WHERE**: Restringe el número de filas por una o varias condiciones. Algunas de las variantes de esta cláusula se mencionan a continuación:

   - Restringir filas con una igualdad simple.

   - Restringir filas que contienen un valor como parte de una cadena.

   - Buscar filas utilizando operadores de comparación, tales como mayor que (>), menor que (<), mayor o igual que (>=), y menor o igual que (<=).

   - Buscar filas utilizando operadores lógicos tales como OR donde al menos una condición debe cumplirse o AND donde todas las condiciones deben de cumplirse.

   - Buscar filas que están dentro de una lista de valores usando la cláusula IN, o la cláusula ANY.

   - Buscar filas que estén comprendidas entre un rango de valores con la sentencia BETWEEN.

4) **GROUP BY**: Agrupas filas por valores de columnas comunes. Especifica una columna o un cálculo no agregado en una columna. Esta columna puede pertenecer a una tabla, una tabla derivada o una vista. La columna debe aparecer en la cláusula FROM de la instrucción SELECT, pero no es necesario que aparezca en la lista SELECT. Aun así, deben incluirse en la lista GROUP BY todas las columnas de la tabla o la vista de cualquier expresión no agregada de la lista de <select>.
5) **HAVING**: Restringe el número de filas cuando se usa agrupación de filas. Especifica una condición de búsqueda para un grupo o agregado. HAVING sólo se puede utilizar con la instrucción SELECT. Normalmente, HAVING se usa con una cláusula GROUP BY. Cuando no se usa GROUP BY, hay un solo grupo implícito agregado.

6) **ORDER BY**: Ordena el resultado por una o más columnas. No hay ningún límite en cuanto al número de columnas de la cláusula ORDER BY.

Se puede ver que la cláusula SELECT debido a su naturaleza implica mayor aprendizaje de cada uno de los diferentes componentes que la pueden integrar, por lo que si un usuario no tiene este conocimiento no puede obtener información de la base de datos.

4.6 PLN

Derivado de la falta de capacidad para recuperar información de BDR tanto para usuarios con experiencia como usuarios sin experiencia, es que se busca aplicar el PLN para la obtención de información de base de datos. PLN es un área que estudia la interacción entre una computadora y un usuario. Tanto el entendimiento del lenguaje natural como el procesamiento del lenguaje mismo son partes fundamentales del PLN. Cuando se índica “lenguaje natural” se refiere al lenguaje que es usado para comunicarse diariamente por los humanos, lenguajes tales como el español o el inglés. Una de las principales dificultades de los lenguajes naturales es que han evolucionado conforme han pasado de generación a generación y son difíciles de precisar con reglas explícitas (Bird, Klein, & Loper, 2009).

En este trabajo se emplea el término PLN para cubrir cualquier tipo de procesamiento o manipulación computacional del lenguaje natural, por ejemplo, la traducción automatizada (*machine translation*), que se enfoca a convertir texto de un lenguaje humano a otro automáticamente. Analizar preguntas del lenguaje natural a SQL puede ser visto como una forma especial de traducción automatizada (Koehn, 2010).

En este sentido, hacer preguntas a bases de datos en lenguaje natural es un método fácil y muy conveniente de acceder a los datos, especialmente para usuarios que no entienden lenguajes de consultas como SQL. PLN y traducción automatizada es el proceso por el cual la consulta en lenguaje natural de un usuario es convertida a una consulta SQL basado en la sentencia ingresada por el usuario (Ghosh, 2014).

4.7 ILNBD

Dado que las computadoras no pueden entender el lenguaje natural, es necesario desarrollar interfaces. Una interfaz es una conexión que puede ser física o lógica, entre una computadora, un dispositivo periférico o un enlace de comunicaciones. Las Interfaces de Lenguaje Natural (ILN) o NLI por sus siglas en inglés (*Natural Language Interfaces*) son interfaces que proveen los medios al usuario para interactuar con la computadora a través del lenguaje natural creando una conexión lógica entre las dos partes (Mohite & Bhojane, 2015).

Según Tennant, Ross, & Thompson (2003), en general, la tarea de escribir una ILN involucra una gramática del lenguaje natural y una correspondiente estructura léxica, así como un conjunto de traducciones semánticas para el sistema objetivo. Las ILN pueden ser clasificadas en dos categorías:

1) Aquéllas basadas en un **seudo-lenguaje**, donde un usuario deberá aprender un lenguaje de comandos que sea similar al lenguaje natural.
2) Aquéllas basadas en la premisa de que un usuario debe poder expresarse en cualquier forma que sea natural para él y el sistema le dará sentido a su expresión de entrada.

Las ILNBD son un campo del PLN donde los datos de BDR son recuperados usando preguntas en el lenguaje natural. En la Figura **1** se muestra un esquema de la traducción del lenguaje natural a una consulta SQL.

Figura 1 Conversión de lenguaje natural a consulta SQL
Fuente: elaboración propia

Si se supone que se quiere obtener toda la información acerca de los estudiantes almacenados en la tabla ESTUDIANTE, para ello se utilizaría una consulta SQL tal como:

“SELECT * FROM ESTUDIANTE;”

Una persona que no conoce SQL y la estructura de la sentencia SELECT no tiene forma de obtener la información. Al usar una ILNDB, la consulta que se ejemplificó arriba puede ser rescrita como:

“Dame toda la información de la tabla ESTUDIANTE”

Ambas sentencias regresarán el mismo resultado, la gran diferencia es que con el uso de una ILNDB, una persona sin conocimiento de SQL podrá recuperar los datos almacenados en la base de datos (Ghosh, 2014).

4.8 Componentes de ILNBD

Según Reshma & Remya (2018), las ILNBD tienen dos componentes principalmente:

1) **Componente lingüístico**: Produce una consulta lógica intermedia de la consulta de lenguaje natural ingresada. El componente lingüístico realiza análisis morfológico, sintáctico y semántico. Como resultado produce una consulta lógica.
2) **Componente base de datos**: Este componente realiza las funciones tradicionales de administración de base de datos. En esta etapa se produce una consulta de base de datos al hacer una correspondencia de las palabras de entrada en el lenguaje natural a las cláusulas correspondientes en la consulta de base de datos. La generación de la consulta está dividida en cuatro fases, donde cada etapa manipulará únicamente una parte específica de la consulta de base de datos.

    a. Esta fase trata con la parte de la consulta lógica, que corresponde a los nombres de los campos para construir la cláusula SELECT.

    b. En esta etapa se construye la cláusula FROM al seleccionar la parte de la consulta lógica que corresponde al nombre de la tabla o grupo de tablas.

    c. En esta etapa se construye la cláusula WHERE al identificar cualquier condición que esté presente en la consulta del lenguaje natural.

    d. En la última etapa, se selecciona la porción de la consulta lógica que corresponde al ordenamiento de la presentación de datos recuperados.

4.9 Arquitecturas de ILNBD

Según Reshma & Remya (2018), existen principalmente cuatro arquitecturas de ILNBD que se describen a continuación:

1) **Emparejamiento de patrones**: Los patrones son definidos usando algunas reglas establecidas manualmente, posteriormente la consulta en lenguaje natural es mapeada a esas reglas. Este tipo de interfaces solo pueden construir una consulta cuando un patrón se iguala con una entrada de usuario dada. La interfaz presentada en este trabajo, se basa en el emparejamiento de patrones.
2) **Interfaces basadas en sintaxis**: La consulta del lenguaje natural es analizada sintácticamente y entonces se mapea el árbol de sintaxis en la estructura de cualquier lenguaje de consultas formal. El árbol de análisis contiene toda la información acerca de la estructura de la consulta y es directamente mapeada a cualquier lenguaje de consulta de base de datos y no hay análisis semántico. Los enfoques basados en sintaxis proveen información detallada acerca de la estructura de una sentencia lo cual es una de las mayores ventajas de usar este tipo de interfaces. Un árbol de análisis da la información detallada acerca de la estructura de una sentencia, pero el problema es que no se pueden mapear todos los nodos, ya que algunos nodos son dejados fuera dado que no agregan ningún significado semántico, y no siempre es claro cuáles nodos deben ser mapeados y cuáles no.
3) **Interfaces de gramática semántica**: En este tipo de interfaces la consulta de la base de datos es generada al analizar la consulta de entrada y generar un árbol de análisis de la consulta del lenguaje natural y también agregando algo de significado lógico al árbol de análisis. Entonces se mapea este árbol de análisis a una consulta de base de datos. La entrada del usuario es analizada sin conceptos sintácticos y la gramática semántica es mejor situada para sistemas de dominio específico, por lo tanto, es difícil exportar a otro dominio.

4) **Interfaces de representación intermedia**: Estas interfaces transforman la consulta con significado interno descrito por una pregunta formulada en lenguaje natural. Actualmente la mayoría de ILNBD son de este tipo porque esta arquitectura convierte primero la consulta de lenguaje natural en una consulta lógica intermedia al pasar por diferentes análisis: semánticos, sintácticos y morfológicos. La consulta lógica es entonces trasladada a una expresión en el lenguaje de consultas de la base de datos, y evaluada contra la base de datos. En el enfoque del lenguaje de representación intermedia, la interfaz puede ser dividida en dos partes:
    a. Una parte empieza desde una sentencia hasta la generación de una consulta lógica.
    b. La segunda parte empieza desde una consulta lógica hasta la generación de una consulta de base de datos.

Además, la consulta lógica intermedia es independiente del dominio de la base de datos, de este modo, la consulta puede ser exportada a diferentes lenguajes de base de datos, así como de dominios tales como sistemas expertos y sistemas operativos.

4.10 Ejemplos de ILNBD

En esta sección se describen algunos ejemplos de ILNDB que se han desarrollado y destacado de manera importante en el ramo de la investigación de sistemas de PLN. A continuación, se muestran cuatro ejemplos:

1) **LIFER/LADDER** Según Hendrix, Sacerdoti, Sagalowicz, & Slocum (1978): Fue uno de los primeros sistemas PLN, fue diseñado como una ILNBD de información acerca de los barcos de la marina de Estados Unidos. Esta interfaz usaba una gramática semántica para analizar preguntas y consultar una base de datos distribuida. Este sistema podía únicamente soportar consultas a una tabla, o consultas a múltiples tablas con condiciones de unión sencillas.
2) **QUESTION-ANSWERING SYSTEM** Dang, Thi, & Tuyen (2009) propusieron un método para construir una interfaz de respuestas-preguntas, el cual es integrado con un sistema de búsqueda para libros electrónicos en una librería. Los usuarios pueden usar simples preguntas en inglés para buscar en la biblioteca información acerca de los libros electrónicos, tales como título, autor, lenguaje, categoría, publicador. En este proyecto de investigación, el foco principal está sobre problemas fundamentales en consultas del PLN, tales como enfoques de análisis de sintaxis, representación de sintaxis, representación semántica y, reglas de transformación para estructura de sintaxis de estructura semántica.
3) **NaLIX - Natural Language Interface for an XML-Database** Según Li, Yang, & Jagadish (2005), es una ILN interactiva a lenguaje XML y consiste de un clasificador de árbol de análisis, validador y un traductor, el cual es requerido para la traducción de consultas de lenguaje natural a consultas XML, que consiste en tres pasos:
    a. Clasificador de árbol de análisis: El cual identifica la palabra clave o frase en la consulta de lenguaje natural que puede ser traducida en su consulta XML correspondiente.

b. Validador del árbol de análisis: Checa el árbol de análisis y checa si el árbol obtenido puede ser mapeado en su correspondiente consulta XML. En esta etapa también se asegura que las palabras claves, atributos y valores numéricos estén en la base de datos.
c. Traductor: Produce una salida efectiva.

4) **WASP - Word Alignment-based Semantic Parsing** Según Wong & Mooney (2006), es una ILN desarrollada recientemente para la representación de consultas. Esta interfaz tiene la intención de lograr el objetivo más amplio, construir una representación completa, formal, simbólica y significativa de una sentencia en lenguaje natural.

4.11 Reconocimiento de voz

Los sistemas de reconocimiento de voz han llegado a ser aplicaciones muy importantes para las tecnologías de reconocimiento del habla. El primer componente del reconocimiento de voz, es naturalmente, la voz, la cual debe ser convertida de un sonido físico a una señal eléctrica empleando un micrófono, y después convertirla a un dato digital con un convertidor análogo a digital. Una vez digitalizada la señal puede ser usada para transcribir el audio a texto. (Yeo, Al-Haddad, & Ng, 2011)

La mayoría de sistemas de reconocimiento de voz modernos están basados en lo que se conoce como "*Hidden Markov Model*" (HMM). En un sistema HMM típico, en donde la señal de voz es dividida en fragmentos de 10 milisegundos. El espectro de poder de cada fragmento, que es esencialmente una trama del poder de la señal como una función de frecuencia, es comparada contra un vector de números reales conocido como coeficientes Cepstral. La dimensión de dicho vector generalmente es pequeña, algunas veces no mayor de 10, aunque los sistemas más confiables tienen dimensiones de 32 o más. La salida final del sistema HMM es una secuencia de esos vectores (Andics et al., 2010).

Para decodificar la voz en texto, los grupos de vectores son empatados a uno o más fonemas (unidad fundamental del habla). Este cálculo requiere entrenamiento, dado que el fonema de una persona varía de persona a persona, e incluso varía de una declaración dicha por la misma persona de un momento a otro. Un algoritmo especial es entonces aplicado para determinar las palabras más probables que produce la secuencia dada de fonemas.

En muchos sistemas modernos, las redes neuronales son utilizadas para simplificar la señal de voz usando técnicas para transformación de características y reducción de dimensionalidad antes del reconocimiento HMM. Detectores de Actividad de Voz (VAD por sus siglas en inglés *Voice Activity Detectors*) también son usados para reducir una señal de audio únicamente a las porciones que son

probables que contengan voz. Esto previene que el reconocedor desperdicie tiempo analizando partes innecesarias de la señal.

5. DESARROLLO

En este trabajo se desarrolló una ILNBD, cuya premisa es que un usuario se exprese en cualquier forma que sea natural para él y la interfaz le dé sentido a su sentencia de entrada. La arquitectura que esta interfaz tiene es el emparejamiento de patrones (Reshma & Remya, 2018). Para declarar los patrones, se definen algunas reglas preestablecidas usando DialogFlow, la cual es una plataforma en donde se pueden construir interfaces conversacionales, y posteriormente se empareja la consulta en lenguaje natural con esas reglas.

En las siguientes secciones se muestran los pasos que se siguieron para la realización de la ILNBD utilizando DialogFlow.

5.1 Crear una cuenta de DialogFlow

Para registrar una cuenta de DialogFlow es necesario contar con una cuenta de Google e ir a https://dialogflow.com/. Automáticamente se habilitan dos herramientas de Google: "*Machine Learning* de Google" y "*Google Cloud Speech-To-Text*", mismas que son empleadas en este trabajo. En la Figura 2 se puede observar la página inicial de DialogFlow:

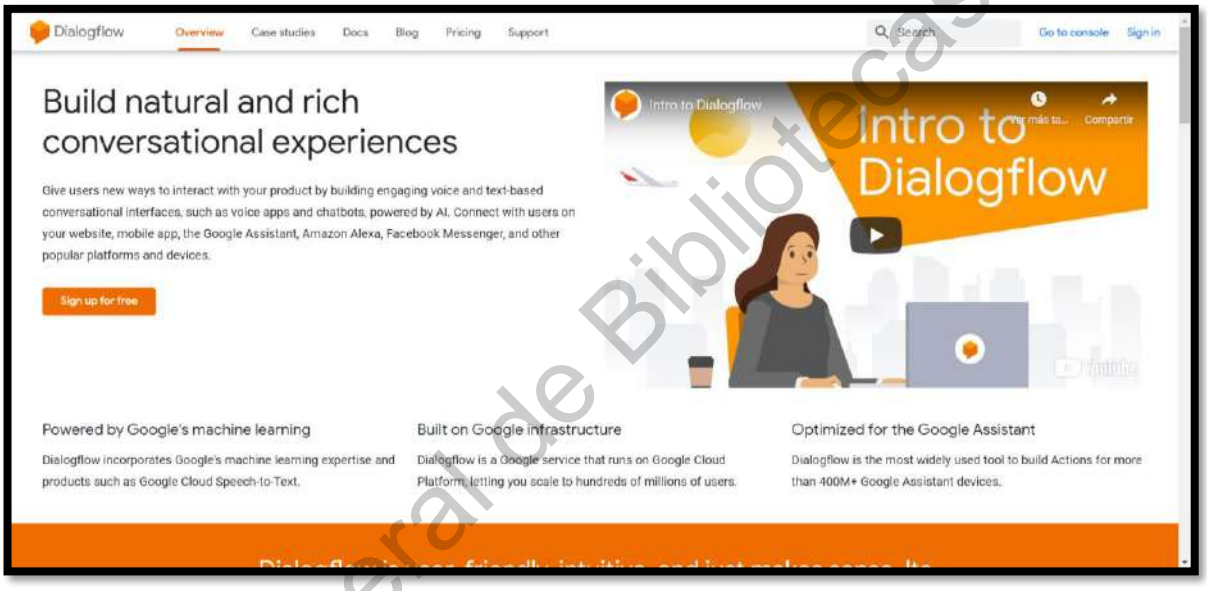

Figura 2 Página inicial DialogFlow

Para iniciar a utilizar DialogFlow es necesario dar clic sobre el botón “Sign In” ubicado en la esquina superior derecha de la página de inicio, y esto direccionará a la consola de DialogFlow, la cual es una interfaz de usuario web que se usa para crear, compilar y probar agentes. En la Figura 3 se muestra la consola de DialogFlow:

Figura 3 Consola de DialogFlow

La consola se usa para crear, compilar, administrar, configurar y probar los agentes. Con ella, se pueden realizar las siguientes tareas:

1) Crear **agentes** para definir la interfaz conversacional
2) Crear **intenciones** que asignan respuestas a las entradas del usuario
3) Crear **entidades** para extraer datos útiles de las entradas del usuario
4) Controlar **rutas de conversación** con contextos
5) Realizar integraciones en otras plataformas de conversación e implementar API’s para conectar los servicios cuando se usa integraciones.
6) Analizar el rendimiento de los agentes.
7) Probar el agente a través del simulador.

5.2 Crear un agente de DialogFlow

DialogFlow permite de manera fácil lograr una experiencia conversacional al controlar el ELN. Con DialogFlow es necesario crear agentes virtuales, los cuales manejan las conversaciones con los usuarios finales, y entienden las variantes del lenguaje humano y traducen eso a un significado estructurado y estándar que las aplicaciones y servicios pueden entender.

Lo primero que se realizó fue crear un nuevo agente. Para esto, es necesario ir al panel de la izquierda de la consola de DialogFlow y desplegar las opciones. Es necesario dar clic en la opción "*Create new agent*", como se muestra en la Figura 4.

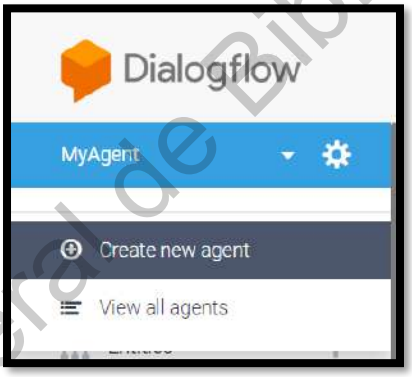

Figura 4 Crear nuevo agente de DialogFlow

Un agente procesa las entradas de usuario y las convierte en datos estructurados que se usan para regresar una respuesta apropiada. Esto se define dentro de una o varias "intenciones", las cuales definen cómo la entrada del usuario es mapeada a su respuesta correspondiente.

En la Figura 5 se ilustra la pantalla principal para la creación de agentes en DialogFlow:

Figura 5 Formulario de agentes en DialogFlow

Esta pantalla está dividida en cinco partes principalmente:

1) Nombre del agente: Aquí se define el nombre del agente.
2) Lenguaje predeterminado: Aquí se selecciona el lenguaje en el que estará operando el agente. El agente que se definió en este trabajo está configurado en el lenguaje “English – en”, esto con la finalidad de probar las variantes multilenguaje. Es importante destacar que este parámetro no puede ser cambiado una vez que el agente ya ha sido creado.
3) Zona horaria predeterminada: Para el agente que se definió para este trabajo se configure la zona horaria GMT -6:00 America/Chicago.
4) Proyecto de Google: Si se requiere habilitar las funciones de la nube, acciones y administración de permisos en Google, es necesario activar está opción. Para este proyecto no es necesario.

5) Tipo de Agente: Si se requiere combinar múltiples agentes en uno solo (mega agente), se deberá activar esta opción. Para este agente no es necesario.

El agente sirve como un contenedor para la configuración y los datos:

- **Configuración del agente,** donde se definen opciones de lenguaje, configuración de aprendizaje automático y otras opciones que controlan el comportamiento del agente.
- **Intenciones** a fin de clasificar las intenciones del usuario final para cada momento donde el usuario final interactúa con la interfaz.
- **Entidades** que sirven para identificar y extraer datos específicos de las expresiones de usuario final.
- **Conocimiento** para analizar documentos y encontrar respuestas automáticas
- **Integraciones para aplicaciones** que se ejecutan en dispositivos o servicios que manejan directamente las interacciones del usuario final en el nombre del usuario, en este caso el asistente de Google.

En la Figura 6 se puede observar el agente que se ha creado para este proyecto llamado “MyAgent”

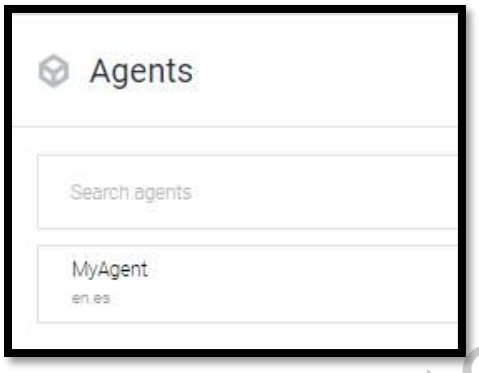

Figura 6 Lista de Agentes

La configuración de un Agente está dividida en 8 pestañas:

- General (*General*)
- Lenguajes (*Languages*)
- Configuraciones ML (*ML Settings*)
- Exportar e Importar (*Export and Import*)
- Ambientes (*Environments*)
- Habla (*Speech*)
- Compartir (*Share*)
- Avanzado (*Advanced*)

5.2.1 Configuración general

En la Figura **7** se aprecia la configuración de la pestaña “General”, establecida para el agente diseñado para este proyecto.

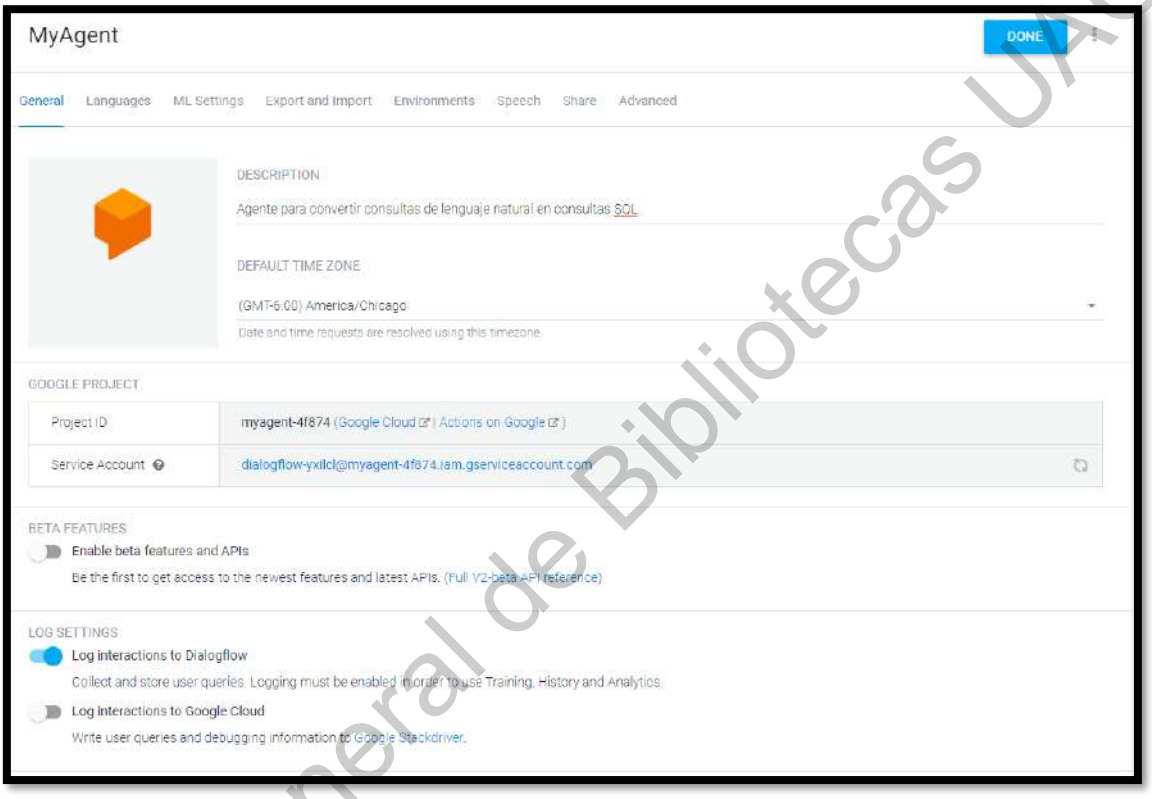

Figura 7 Configuración de agente "MyAgent" – General

En esta pestaña se configura los siguientes parámetros:

- **Descripción**: Aquí se indica el propósito de este agente, y si se desea, puede agregarse una descripción más amplia.
- **Zona horaria predeterminada**: La zona horaria predeterminada del agente es GMT-6:00 América/Chicago.
- **Proyecto de Google**:
    - **ID del proyecto**: el proyecto de GCP vinculado al agente. En este caso myagent-4f874.

    - **Cuenta de servicio**: la cuenta de servicio que utiliza Dialogflow para la integración del Sistema. Para este proyecto es *dialogflow-yxilcl@myagent-4f874.iam.gserviceaccount.com*
- **Funciones Beta**: Es un interruptor para habilitar o inhabilitar las funciones Beta del agente. Para este proyecto no es necesario.
- **Configuración de registros**:
    - Registrar interacciones en Dialogflow (*Log interactions to Dialogflow*): Para este proyecto es necesario habilitarlo para recolectar y almacenar las entradas de los usuarios, de modo que se pueda emplear “entrenamiento”, “históricos” y “analíticos”.
    - Registrar interacciones en Google Cloud (*Log interactions to Google Cloud*): Esta opción sólo está disponible si “Registrar interacciones en Dialogflow” (*Log interactions to Dialogflow*) está habilitada. Si se inhabilita el registro de Dialogflow, también se inhabilitará esta configuración. Esta opción permite escribir las entradas de usuario e información de depuración en “Google Stackdriver. Para este proyecto no es necesario.
- **Borrar agente**: Esta opción borra por completo el agente y no se puede deshacer. Si el agente se comparte con otros usuarios, debes quitarlos del agente antes de borrarlo. Esta opción debe utilizarse con extremo cuidado, ya que una vez que se elimina el agente, no hay manera de revertir esta acción.

5.2.2 Configuración de idiomas

En la Figura **8**, se puede ver la segunda pestaña de la configuración llamada “lenguajes”. En esta pestaña se puede observar el idioma predeterminado y se pueden agregar varios idiomas adicionales, obteniendo de este modo un agente multilenguaje. Para este proyecto, en el agente se usaron dos lenguajes:

a. English – en (predeterminado)
b. Spanish - es

Figura 8 Configuración de agente "MyAgent" – Languages

Es importante mencionar que existen dos tipos de lenguajes:

- Idiomas raíz: Estos son idiomas como el inglés (en), que no especifican una configuración regional.
- Idiomas específicos de la configuración regional: Estos son idiomas como el inglés de EE.UU. (en-US), que especifican una configuración regional, como un área o un país específico.

Algunos idiomas pertenecen a ambas categorías: actúan como raíz y como específicos de la configuración regional. Existen diferencias significativas entre las configuraciones regionales para estos idiomas, por lo que las configuraciones regionales no pueden compartir un mismo idioma raíz. En algunos idiomas raíz, también se pueden agregar una o más configuraciones regionales.

5.2.3 Configuración aprendizaje automático

La siguiente pestaña es configuración Aprendizaje Automático (AA). Los agentes de Dialogflow usan algoritmos de AA para comprender las expresiones del usuario final, detectar coincidencias con las intenciones y extraer datos estructurados. Un agente aprende de las frases de entrenamiento que le suministras y de los modelos de lenguaje integrados en Dialogflow. A partir de estos datos, compila un modelo para tomar decisiones sobre qué intención debe asignarse a determinada expresión del usuario final. Ese modelo es único para cada agente, y cada desarrollador debe generar su propio modelo.

Según la configuración predeterminada, Dialogflow actualiza el modelo de AA del agente cada vez que se modifican las intenciones y las entidades, así como cuando se importa, restablece o entrena un agente.

Se puede configurar los siguientes parámetros:

- **Modo de coincidencia**: Esta configuración define los algoritmos que deben usarse para la coincidencia de todas las intenciones en los que el AA esté habilitado. Se puede seleccionar uno de los siguientes dos modos:
    - **Híbrido:** Este modo primero intenta la coincidencia gramatical basada en reglas. Si no obtiene resultados, pasa al algoritmo de coincidencia por AA.
    - **Sólo AA**: Este modo solo busca coincidencias por AA.

- **Umbral de clasificación del AA**: A fin de filtrar los falsos positivos y obtener variedad en las entradas de lenguaje natural coincidentes para cada agente, se puede ajustar el umbral de clasificación de AA. Esta configuración controla la confianza mínima de detección de intenciones requerida para una coincidencia de intenciones. Para este proyecto se determinó un valor de 0.85.
- **Corrección ortográfica automática**: Si esta función está habilitada y la entrada del usuario tiene un error ortográfico o gramatical, se buscará una intención coincidente como si la entrada estuviera escrita correctamente. La respuesta de la detección de intención contendrá la entrada del usuario corregida. Por ejemplo, si un usuario escribe "Quiero una mansana", la frase se procesará como si hubiese escrito "Quiero una manzana". Esto también se aplica a las coincidencias que involucran entidades personalizadas y del sistema. La corrección ortográfica está disponible para todos los idiomas admitidos en Dialogflow. Para este proyecto, esta opción se encuentra habilitada.
- **Entrenamiento automático:** Habilita o inhabilita el entrenamiento automático del agente cada vez que éste sufra algún cambio. Este valor no se recomienda para agentes grandes, pero dado que el agente del proyecto es pequeño, se activó esta opción para re-entrenar al agente cada que se realice algún cambio.
- **Validación del agente**: Para el agente del proyecto está opción está habilitada, lo cual permitirá que el agente sea validado automáticamente cuando el entrenamiento del agente se esté llevando a cabo.

En la Figura **9** se puede observar la pestaña de “ML Settings” configurada para el agente que se presenta en este trabajo.

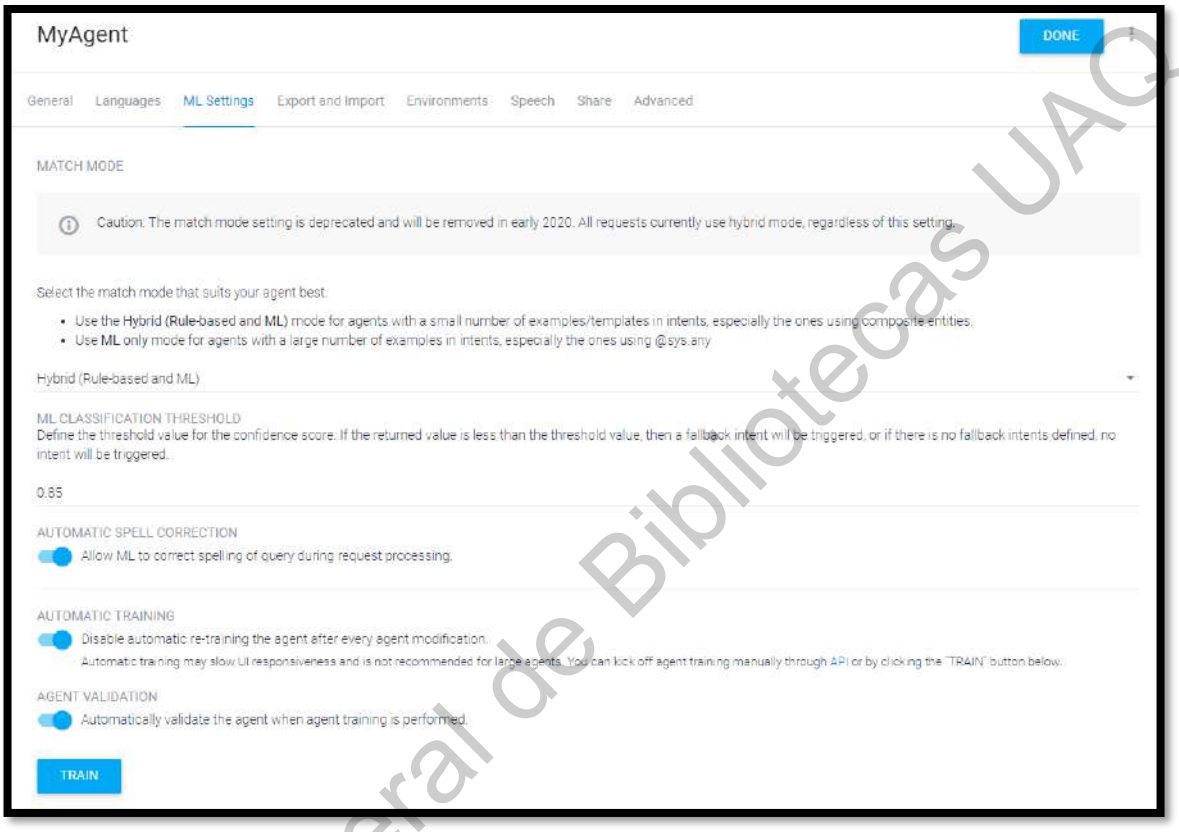

Figura 9 Configuración de agente "MyAgent" – ML Settings

5.2.4 Exportar e importar

Esta función permite exportar o importar un agente en un archivo ZIP para crear copias de seguridad de los agentes o transferirlos de una cuenta a otra. Están disponibles las siguientes opciones:

- Exportar como ZIP: Exporta el agente como un archivo zip.
- Restablecer desde ZIP: Reemplaza el agente actual por el archivo zip proporcionado.
- Importar desde ZIP: Agrega intenciones y entidades al agente actual desde el archivo zip proporcionado. Si alguna de las intenciones o las entidades existentes tienen el mismo nombre que los del archivo zip, se les reemplazará.

En la Figura **10**, se puede ver la pestaña de configuración “Export and Import”.

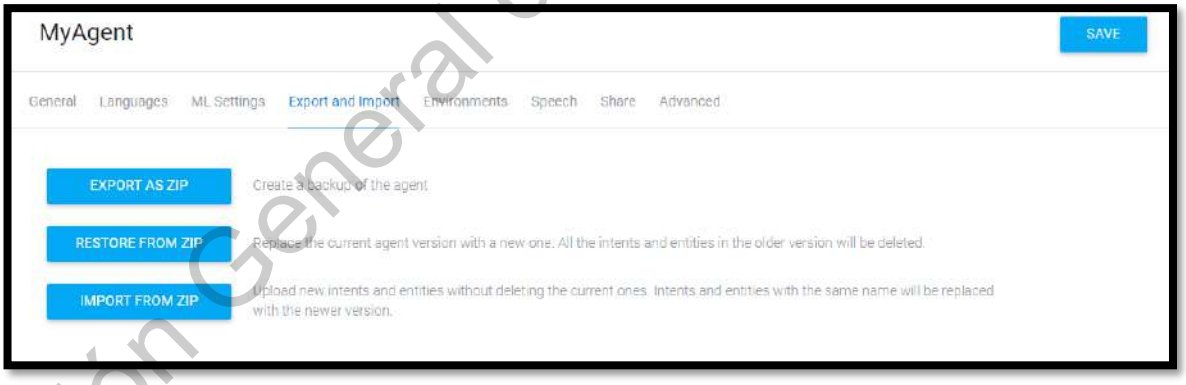

Figura 10 Configuración de agente "MyAgent" – Export and Import

Una vez configurado debidamente el agente, fue necesario realizar el proceso de “Export as ZIP” para generar un respaldo.

5.2.5 Configuración de ambientes

Las versiones y los entornos permiten implementar varias versiones de los agentes en entornos separados y personalizables. Se pueden crear varias versiones de cada agente y publicarlas en entornos separados.

Cuando se edita un agente, lo que se modifica es el agente de borrador, y cuando se requiera se puede guardar el agente de borrador como una versión de agente, que es una instantánea inmutable de éste.

Cuando se guarda el agente de borrador, éste se publica en el entorno predeterminado. Cuando se crean versiones del agente, se pueden publicar en los entornos personalizados. Se pueden crear diversos entornos personalizados para lo siguiente:

- Pruebas
- Desarrollo
- Producción

En este proyecto únicamente se definieron dos ambientes, uno para el desarrollo, y el segundo para producción, pero se pueden crear tantos ambientes como se requieran.

En la Figura **11** se puede observar la pestaña de configuración "*Environments*", y los ambientes que se definieron:

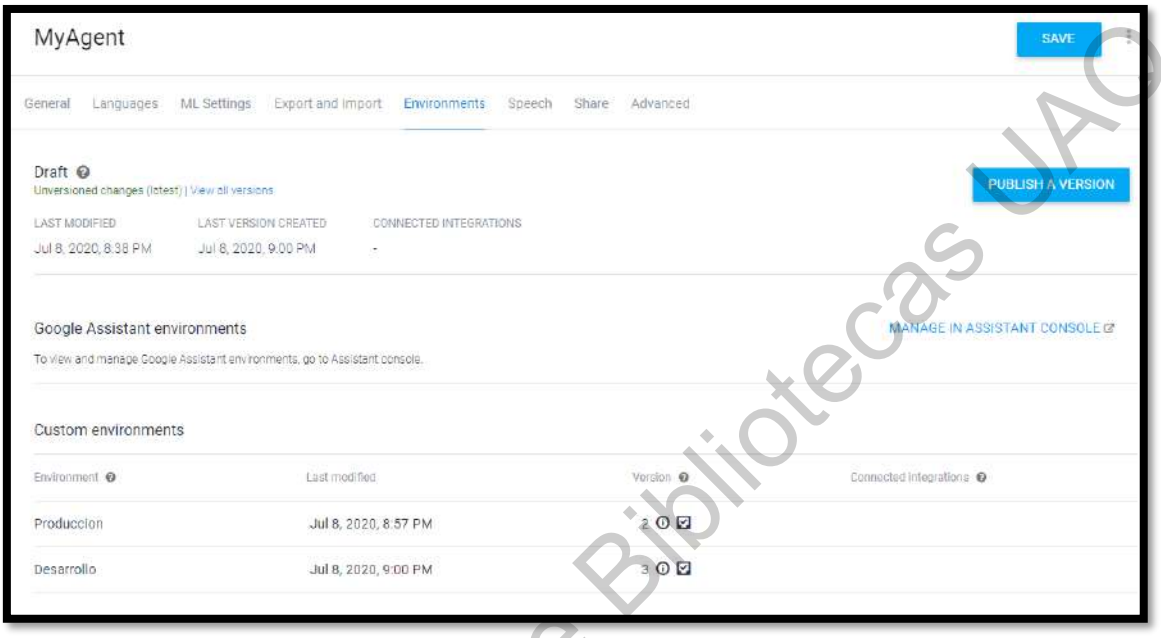

Figura 11 Configuración de agente "MyAgent" - Environments

5.2.6 Habla

En esta pestaña se definen algunos parámetros relacionados al reconocimiento de voz. Para empezar, hay dos opciones para mejorar la calidad del reconocimiento del habla:

1) Habilitar modelos de habla mejorados y bitácora de datos: Está opción solo está disponible para cuentas de tipo *Enterprise.* Para este proyecto, está opción se encuentra deshabilitada.
2) Habilitar la adaptación automática del habla: Al habilitar esta opción se usará la información de agentes de DialogFlow, tales como intenciones o entidades para mejorar automáticamente la calidad de reconocimiento del habla. El agente que se definió para este proyecto tiene habilitada esta opción.

Además, en esta pestaña se puede configurar algunos parámetros para el reconocimiento de voz, como el lenguaje de voz del agente, el tipo de voz, la frecuencia de voz, el volumen, entre otros parámetros.

En la Figura **12**, se puede ver la configuración de la pestaña “Speech” para el agente, donde se habilitó la adaptación para mejorar la calidad del reconocimiento de voz.

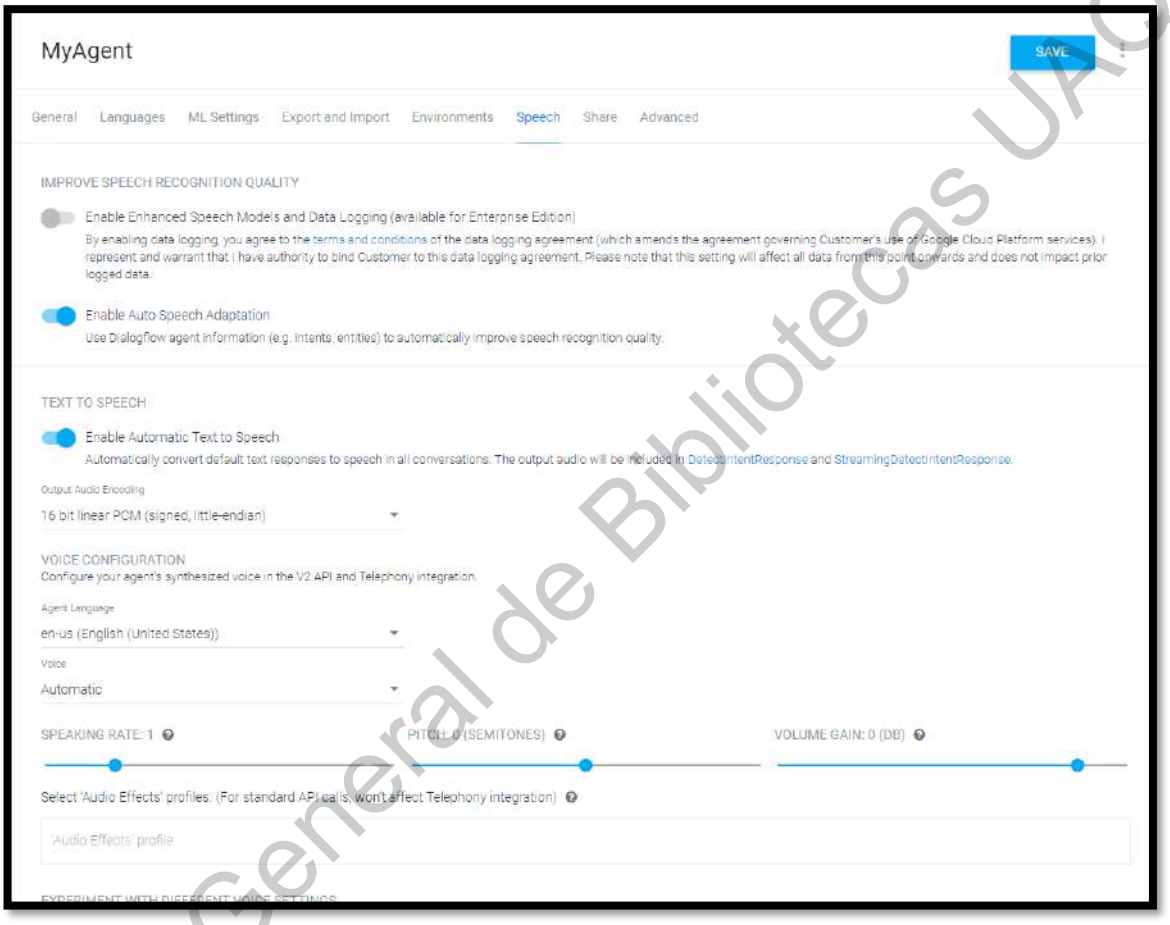

Figura 12 Configuración de agente "MyAgent" – Speech

5.2.7 Compartir

En esta pestaña se puede definir si se desea compartir el agente con otros usuarios de DialogFlow, concediendo o revocando los accesos. Para este agente sólo se permite el acceso al usuario administrador, como se puede observar en la Figura **13**.

Figura 13 Configuración de agente "MyAgent" – Share

5.2.8 Configuración avanzada

En esta pestaña se puede habilitar la opción “Sentiment Analysis”, que da una puntuación para cada entrada de usuario. Para el agente que se configuró no se habilitó, como se muestra en la Figura **14**:

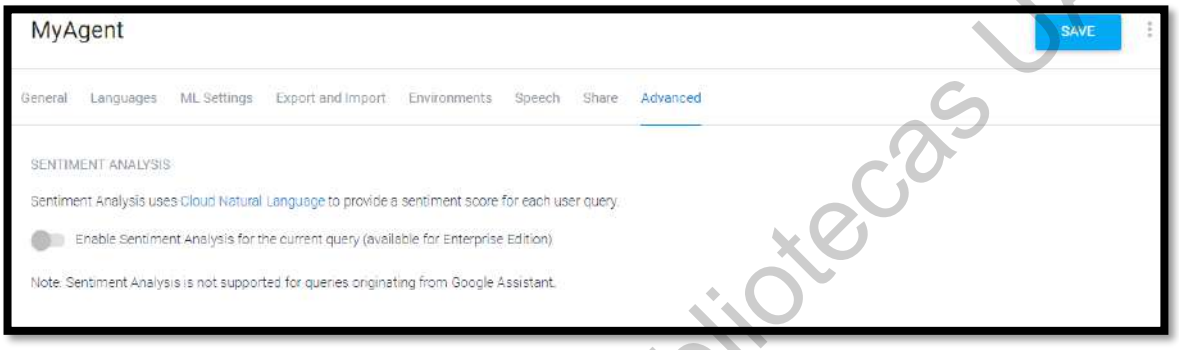

Figura 14 Configuración de agente "MyAgent" – Advanced

Una vez que ya se ha creado el agente, el siguiente paso será diseñar y definir en la siguiente sección las intenciones que tendrá el agente.

5.3 Crear intenciones

Como se mencionó anteriormente, una intención clasifica la intención del usuario final para un turno de conversación. En general, es necesario definir varias intenciones para cada agente; así, al combinarlos se logra establecer una conversación completa con el usuario final. En la Figura 15 se puede observar la pantalla para crear intenciones en DialogFlow:

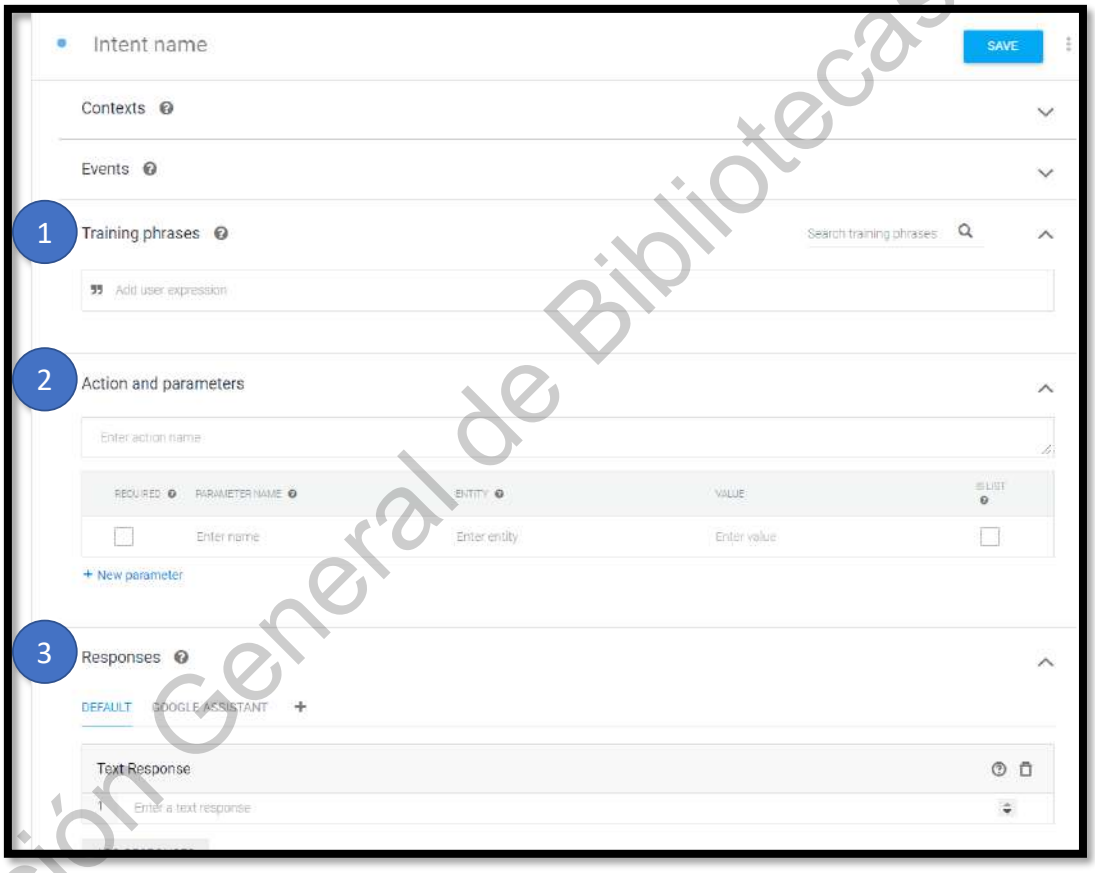

Figura 15 Crear intenciones

Una intención básica está compuesta de los siguientes componentes, como se muestra en la Figura **15**:

1) **Frases de entrenamiento**: Son frases de ejemplo de lo que los usuarios puedan decir. DialogFlow usa estas frases de entrenamiento y las expande a muchas más frases similares para construir un modelo de lenguaje que de forma más precisa empareje la entrada del usuario.
2) **Acciones y parámetros**: Para mejorar un modelo de lenguaje de una intención, se puede estructurar las frases de entrenamiento con “entidades”, las cuales son categorías de datos con las que se quiere que DialogFlow empareje. Esto le dice a DialogFlow que se requiere un particular tipo de dato y no sólo emparejar de manera literal la entrada del usuario. DialogFlow extrae las entidades que encuentra como parámetros de las frases de entrenamiento. Se puede procesar estos parámetros en una lógica llamada “cumplimiento” para personalizar una respuesta del usuario. En la siguiente sección, se muestra las entidades generadas para este agente.
3) **Respuestas:** Define un texto, diálogo, o respuesta visual para el usuario. Para enviar respuestas, se puede usar respuestas propias de DialogFlow o llamar el modo “cumplimiento” para procesar los datos extraídos y regresar una respuesta a DialogFlow.

En la Figura **16**, se observa como los componentes interactúan entre sí:

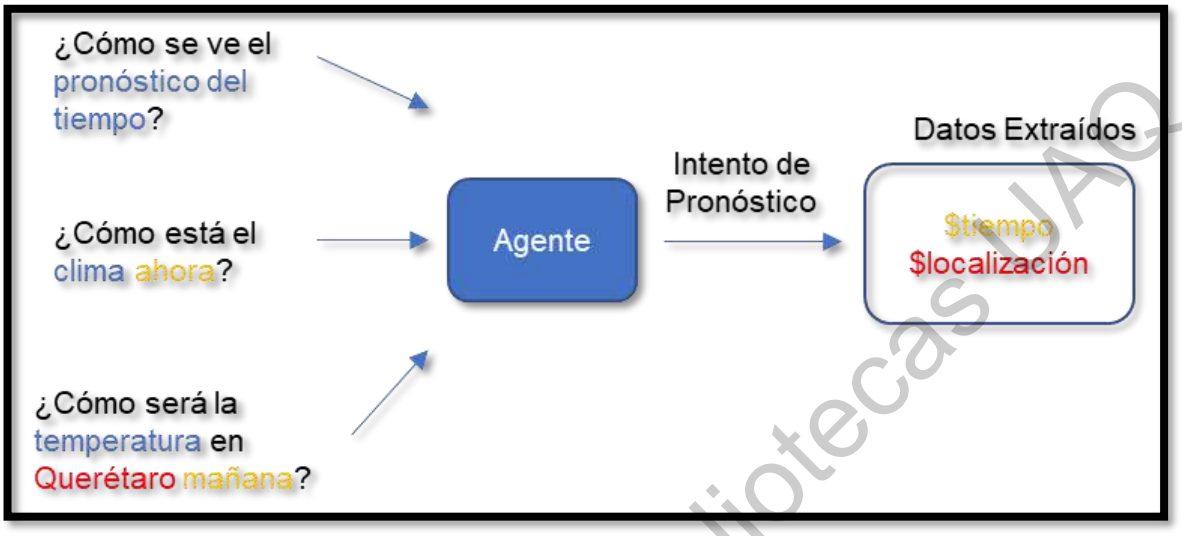

Figura 16 Intercomunicación de los componentes de una intención. Fuente (Google, 2019a)

Cuando los usuarios dicen algo conocido como una expresión, el agente identifica la expresión con una intención que coincida, también conocido como clasificación de intención. Se concuerda una intención si el modelo de lenguaje para esa intención puede coincidir de manera cercana o exacta con la expresión del usuario. El modelo de lenguaje se define especificando frases de entrenamiento o ejemplos de cosas que los usuarios quieran decir. DialogFlow toma estas frases de entrenamiento y crea el modelo de lenguaje del intento.

Una vez que el agente coincide con una intención, extrae los parámetros. Puede ser un color, nombre, fecha o una serie de otras categorías de datos llamados “entidades”. DialogFlow define un conjunto grande de entidades que categorizan los parámetros extraídos, o el usuario puede crear los suyos propios. También define qué extraer en sus frases de entrenamiento, anotando partes específicas de las frases de entrenamiento para especificar qué parámetros desea extraer.

Luego, envía una respuesta para finalizar la conversación. DialogFlow incluye un controlador de respuestas fácil de usar para devolver respuestas simples, generalmente estáticas. Si desea devolver respuestas más completas y dinámicas, puede usar la lógica llamada “cumplimiento” para procesar los parámetros extraídos y devolver una respuesta que sea más útil.

Cada intención tiene un controlador de respuesta incorporado que puede devolver respuestas después de que la intención coincida. Sin embargo, esta característica sólo le permite construir respuestas que son estáticas o que tienen una lógica mínima. La mayoría de las veces se usa el “cumplimiento” para procesar la intención primero y luego devolver una respuesta más inteligente o útil. El “cumplimiento” es una lógica personalizada que se implementa como un *webhook*, que los servicios solicitan, procesan y devuelven respuestas.

El agente normalmente utiliza una combinación de los dos tipos de respuesta, utilizando el controlador de respuesta incorporado de Dialogflow para devolver respuestas simples o estáticas, y luego delegar en el cumplimiento para generar respuestas más adecuadas, variadas o adecuadas.

A continuación, se describe como trabaja un simple agente.

1) El agente hace coincidir la expresión de un usuario con una intención.
2) El agente extrae parámetros de la declaración del usuario y llama a su Cumplimiento con una carga JSON que contiene los parámetros junto con una gran cantidad de otra información útil sobre la intención.
3) El cumplimiento procesa cualquier información necesaria que necesite desde el JSON, como llamar a otra API REST con los parámetros extraídos.
4) El Cumplimiento construye una respuesta y la devuelve a Dialogflow para representarla al usuario. Esta respuesta puede ser un texto simple o una respuesta enriquecida, como una tarjeta con una imagen. Dialogflow devuelve automáticamente una respuesta utilizando el controlador de respuesta incorporado de la intención como una alternativa. A veces es útil

definir una respuesta en el agente y en cumplimiento para aprovechar esta función.

En la Figura **17**, se muestra el flujo básico para la coincidencia de intenciones y respuesta al usuario final:

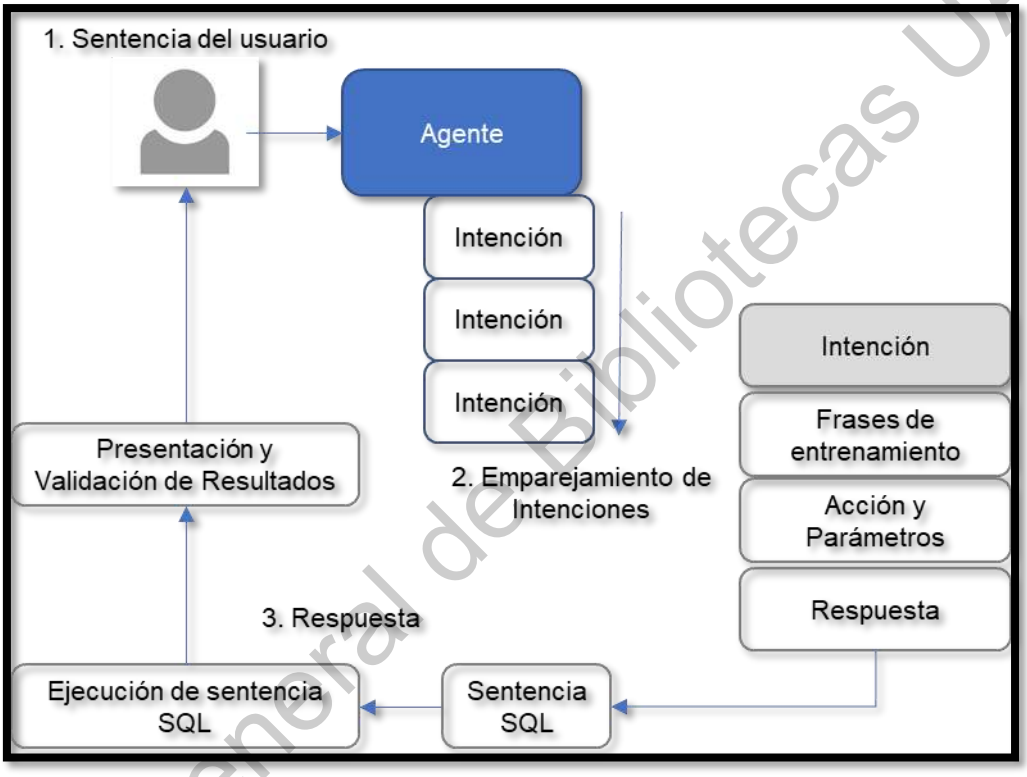

Figura 17 Flujo básico para la coincidencia de intenciones

Para este trabajo se definieron tres intenciones, como se muestra en la Figura 18, más otras dos que se generan de manera automática:

Figura 18 Intenciones de MyAgent

En la Figura **18**, se pueden observar las siguientes intenciones:

1) ***Default Fallback Intent***: A *fallback intent* será lanzado si la entrada de usuario no empareja por ninguna otra intención regular o si empata con alguna de las frases de entrenamiento que en esta intención se definan.
2) ***Default Welcome Intent***: Un *default welcome intent* será lanzado al inicio de cualquier conversación. Generalmente un saludo es lo que lanzará esta intención.

3) SELECT COUNT TABLES: Esta intención que se muestra en la Figura 19 será lanzada cuando el usuario quiera conocer el número total de tablas dentro de una base de datos determinada.

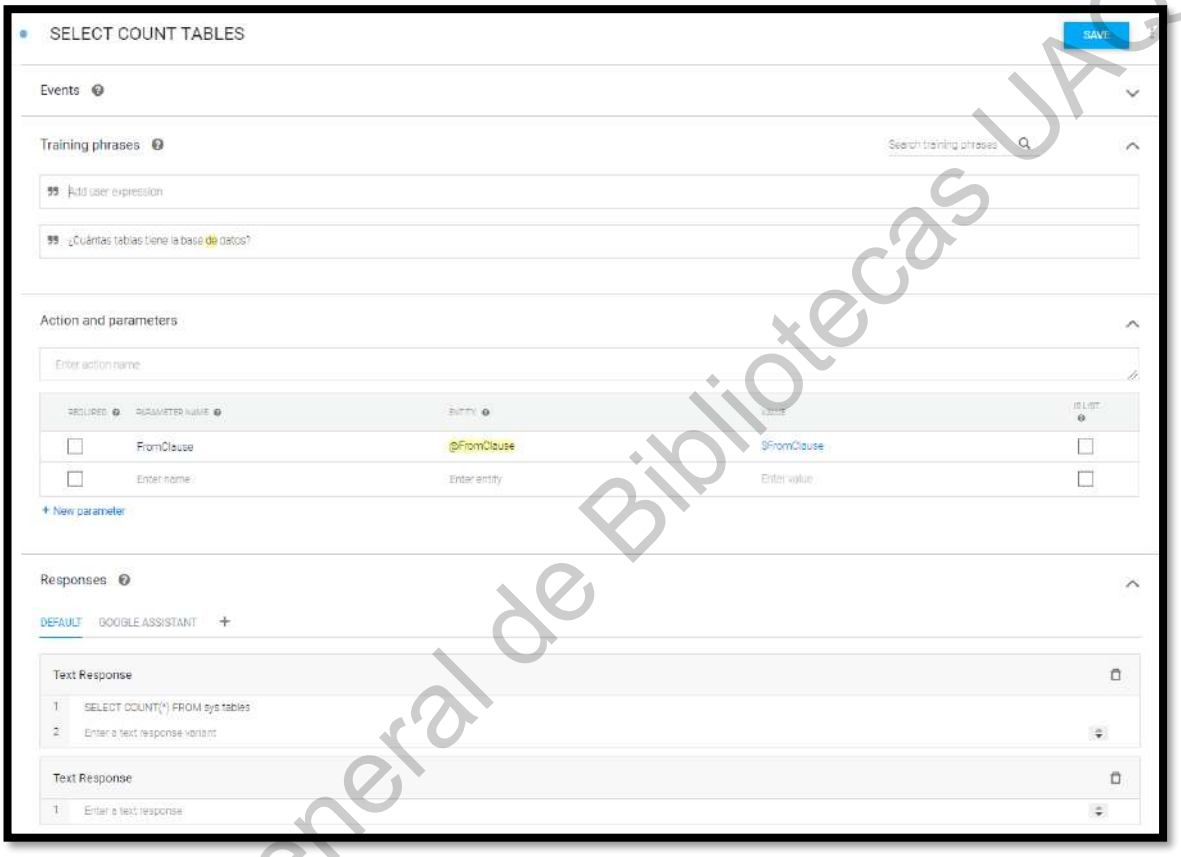

Figura 19 Intención SELECT COUNT TABLES

En esta intención se definió únicamente una frase de entrenamiento: “¿Cuántas tablas tiene la base de datos?” y se añadió un parámetro llamado “FromClause”, que en la sección de entidades se describe qué representa. Al final, esta intención devolverá el texto de respuesta:

SELECT COUNT(*) FROM sys.tables;

El cual es un comando listo para ser ejecutado en SQL y devolverá el número total de tablas que contiene una base de datos determinada.

4) SELECT COUNT {TABLE}: Esta intención que se muestra en la Figura 20 será lanzada cuando el usuario quiera conocer el número total de registros en una tabla determinada.

Figura 20 Intención SELECT COUNT {TABLE}

En esta intención se definieron 3 frases de entrenamiento:

- ¿Cuántos alumnos hay?
- ¿Cuántas personas hay?
- ¿Cuántos registros tiene la tabla de personas?

Se definieron 2 parámetros:

- TablesName: Este parámetro identificará el nombre de la tabla que se desea consultar.

- Hay: Identificará los sinónimos de la palabra “Hay” de las consultas de entrada del usuario final, entendiendo como sinónimos “existen”, “tiene”, entre otros.

Esta intención devolverá como resultado de emparejamiento el texto:

SELECT COUNT(*) FROM *$TablesName*;

$TablesName es un parámetro dinámico y será obtenido de la consulta de entrada. Se identifica como parte de una entidad y es regresado en la respuesta de texto. Por lo cual, está intención permite hacer la misma consulta para todas las tablas que se tengan definidas en la base de datos y que se haya definido en las entidades como parte del diccionario de datos.

5) SELECT SIMPLE: Esta intención que se muestra en la Figura 21 será lanzada cuando el usuario quiera obtener todos los registros de una simple tabla especificada.

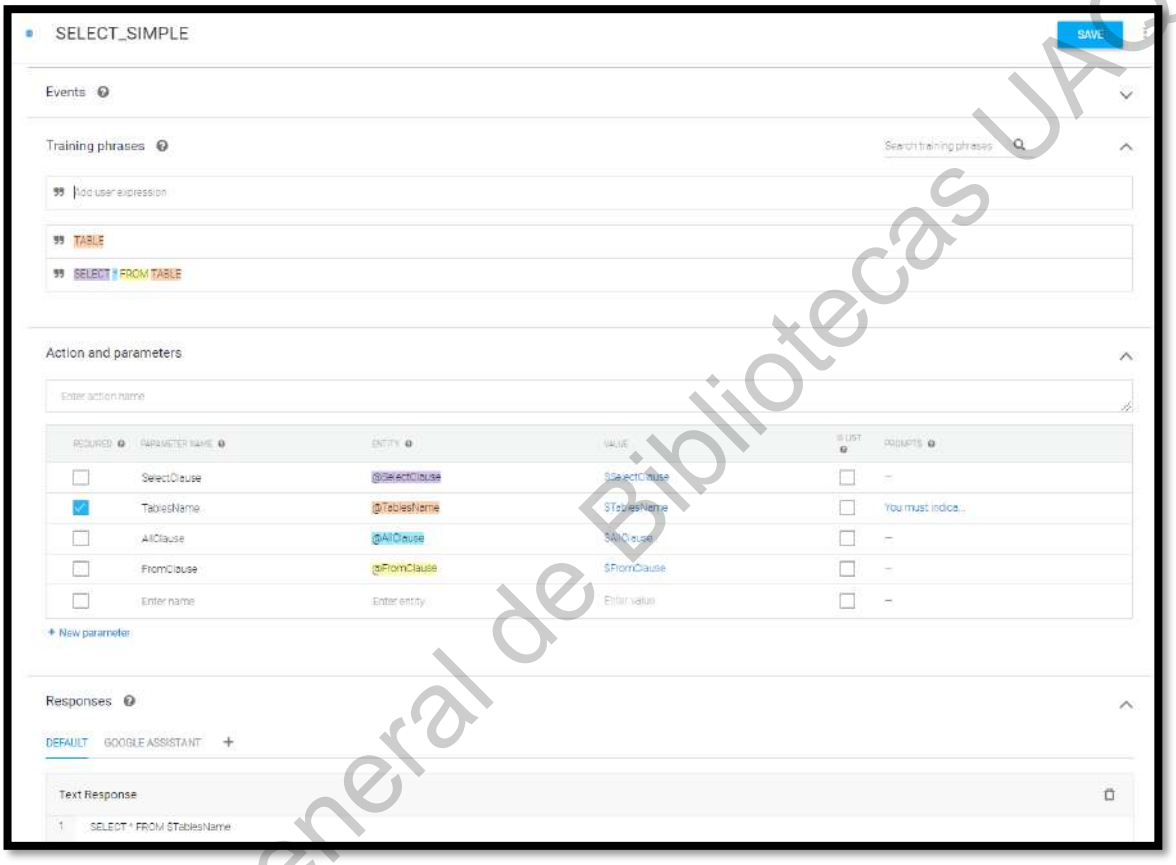

Figura 21 Intención SELECT SIMPLE

Para esta intención se definieron sólo dos frases de entrenamiento:

- TABLE: Se usa cuando el usuario sólo mencione el nombre de una tabla.
- SELECT * FROM TABLE: Esta intención está compuesta de cuatro entidades:
    - SELECT: Entiende todos los sinónimos de esta palabra clave.
    - * (asterisco): Entiende todas las variantes para indicar que se quiere obtener todo el contenido.
- FROM: Entiende todas las variantes para indicar el origen de la consulta.

- TABLE: Entiende el nombre de la tabla de la que se quiere obtener la información.

Esta intención devuelve como resultado de emparejamiento el texto:

SELECT * FROM *$TablesName*;

$TablesName es un parámetro dinámico y será obtenido de la consulta de entrada. Se identifica como parte de una entidad y es regresado en la respuesta de texto. Por lo cual, está intención permite hacer la misma consulta para todas las tablas que se tengan definidas en la base de datos y que se haya definido en las entidades como parte del diccionario de datos.

5.4 Crear entidades

Las Entidades son mecanismos de DialogFlow para identificar y extraer datos útiles de entradas del lenguaje natural. Define cómo los datos serán extraídos de cada sentencia. Las entidades categorizan partes importantes de las sentencias de los usuarios. Con las entidades se empareja una categoría en lugar de una sentencia específica, lo cual brinda mayor flexibilidad. Representa el diccionario de datos, tablas y campos presentes en la base de datos.

Mientras que las intenciones le permiten al agente entender la motivación detrás de una entrada de usuario en particular, las entidades se usan para seleccionar información específica que mencionan los usuarios, nombres de tablas o nombres de campos. Cualquier dato importante que se requiera obtener de la solicitud del usuario tendrá una entidad correspondiente.

1) **Tipo de entidad**: Define el tipo de información que desea extraer de la entrada del usuario. Por ejemplo, “Tabla” es el nombre de un tipo de entidad.

2**) Entrada de entidad**: Para cada tipo de entidad, hay muchas entradas de entidad, las cuales proporcionan un conjunto de palabras o frases que se consideran equivalentes. Por ejemplo, si “Tabla” es un tipo de entidad, se definen estas tres entradas de entidad: (1) Empleados, (2) Alumnos y, (3) Profesores.

Hay cuatro tipos de Entidades que se describen a continuación:

1) **Entidades del sistema:** Dialogflow incluye numerosas entidades del sistema, que permiten a los agentes extraer información sobre una amplia gama de conceptos sin ninguna configuración adicional. Por ejemplo, para extraer fechas, horas y ubicaciones de las entradas de lenguaje natural.

2) **Entidades desarrolladas**: Si se necesita extraer información sobre conceptos más allá de los cubiertos por las entidades del sistema de Dialogflow, puede definir sus propios tipos de entidades de desarrollador, como es el caso de la entidad “Tabla”.

3) **Entidades de sesión:** También es posible definir tipos de entidades que se aplican sólo a una conversación específica. Por ejemplo, se puede crear un tipo de entidad para representar las opciones sensibles al tiempo disponibles para un usuario en particular al realizar una reserva. Estos son llamados tipos de entidad de sesión.

4) **Gestionar entidades**: Dialogflow también proporciona herramientas para administrar entidades, incluidos mecanismos para exportar y cargar datos de entidades y modificar entidades mediante API.

En la Figura **22** se puede ver de manera gráfica la interacción entre los componentes descritos:

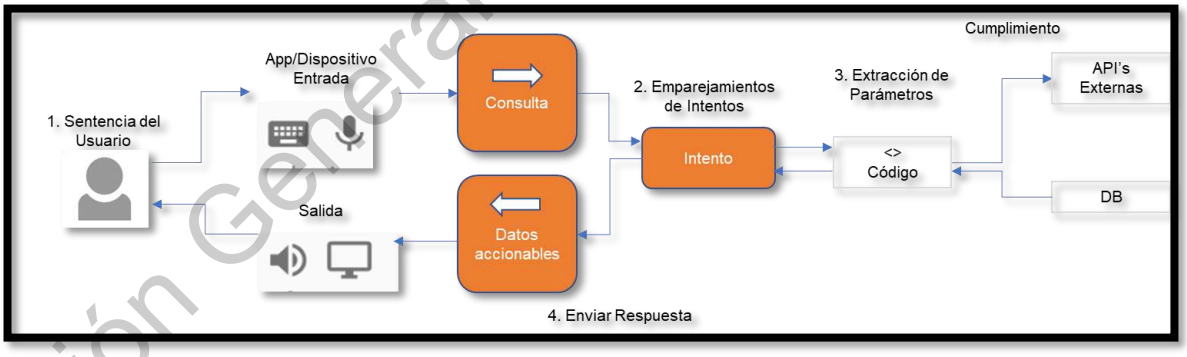

Figura 22 Diagrama de un agente simple. Fuente: (Google, 2019b)

Para este proyecto se definieron cinco entidades que se pueden ver en la Figura **23**.

Figura 23 Lista de entidades

1) **AllClause**: Esta entidad que se muestra en la Figura 24 identificará cuando el usuario final requiera saber todos los elementos en una consulta, incluye algunos sinónimos de la clausula *ALL,* tales como: "*****", *ALL*, *ALL FIELDS*, *EVERYTHING*, Todas, y Todo.

Figura 24 Entidad AllClause

2) **FromClause**: Esta entidad identificará cuando el usuario final se exprese para definir una referencia hacia una tabla. Incluye los sinónimos de la cláusula FROM, tales como De, FROM, Las, Los, *Of* y *Using*. En la Figura **25** se puede observar la configuración de esta entidad.

Figura 25 Entidad FromClause

3) **Hay**: Esta entidad identificará cuando el usuario se refiera a la existencia de un elemento. Incluye sinónimos tales como: Hay o Tenemos. En la Figura **26** se puede observar la configuración de esta entidad.

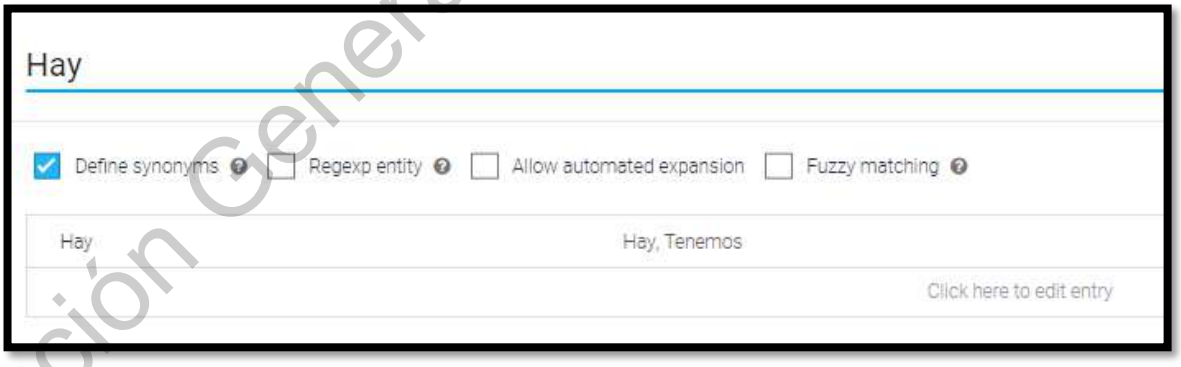

Figura 26 Entidad Hay

4) **SelectClause**: Esta entidad identifica cuando el usuario exprese una solicitud para recuperar información desde una tabla. Define algunos sinónimos como: ¿Cuáles son?, ¿Cuáles son los nombres?, Da, Di, *Give*, Muestra, *Read*, Regresa, *SELECT*, selecciona, y *Tell*. Todos estos son algunas variantes de cómo las personas, requieren información en lenguaje natural. En la Figura **27** se puede observar la configuración de esta entidad.

Figura 27 Entidad SelectClause

5) **TablesName**: Esta entidad identifica a qué tabla se está refiriendo el usuario cuando exprese una solicitud en lenguaje natural. En esta entidad se definen las diferentes tablas que existen en la base de datos, y todas las formas como se puede llamar a cada una de las tablas. En la Figura 28 se puede observar la configuración de esta entidad.

Figura 28 Entidad TablesName

5.5 Implementación

A continuación, se describe el proceso general de cómo se integró DialogFlow en la ILNDB, mismo que se resume en la Figura **17**.

1) **Generación de consulta**: Creación de la sentencia de usuario usando voz en lenguaje natural y conversión a texto con DialogFlow.

2) **Emparejamiento de intenciones**: Empareja la sentencia del usuario para entregar una respuesta.

a. Frases de entrenamiento: Valida con que intención tiene mayor porcentaje de emparejamiento y determina la intención que sea más confiable de acuerdo a las frases de entrenamiento.

b. Acción y parámetros: Extrae las tablas para estructurar la consulta SELECT.

c. Respuesta: Es la consulta SQL que será ejecutada en el motor de base de datos SQL server.

3) **Ejecución de consulta generada**: DialogFlow genera la consulta lista para ser ejecutada en SQL server.

4) **Presentación de resultados**: Después de generar la consulta SQL generada en DialogFlow se presentan los resultados generados.

5) **Evaluación de resultados**: Se valida que los resultados obtenidos vayan de acuerdo a la sentencia emitida por el usuario. En caso de que no sea así, deberá refinar la consulta, así como las frases de entrenamiento e intenciones predefinidas.

5.6 Integración con el asistente de Google

La integración de Dialog Flow con el asistente de Google permite llegar a los usuarios a través de altavoces activados por voz como Google Home, teléfonos, iPhones y Android elegibles, y en el futuro, todos los dispositivos donde el asistente de Google esté disponible.

Para probar el agente en dispositivos compatibles, sólo es necesario iniciar sesión en el dispositivo con la misma cuenta de Google que utilizó para iniciar sesión en DialogFlow y las acciones en la consola de desarrollador de Google.

A continuación, se describen los pasos para implementar el agente con el asistente de Google. Para empezar, es necesario ir al menú “Integrations”, ubicado en el panel lateral izquierdo, y dar clic en la opción “Integration Settings”, como se muestra en la Figura **29**.

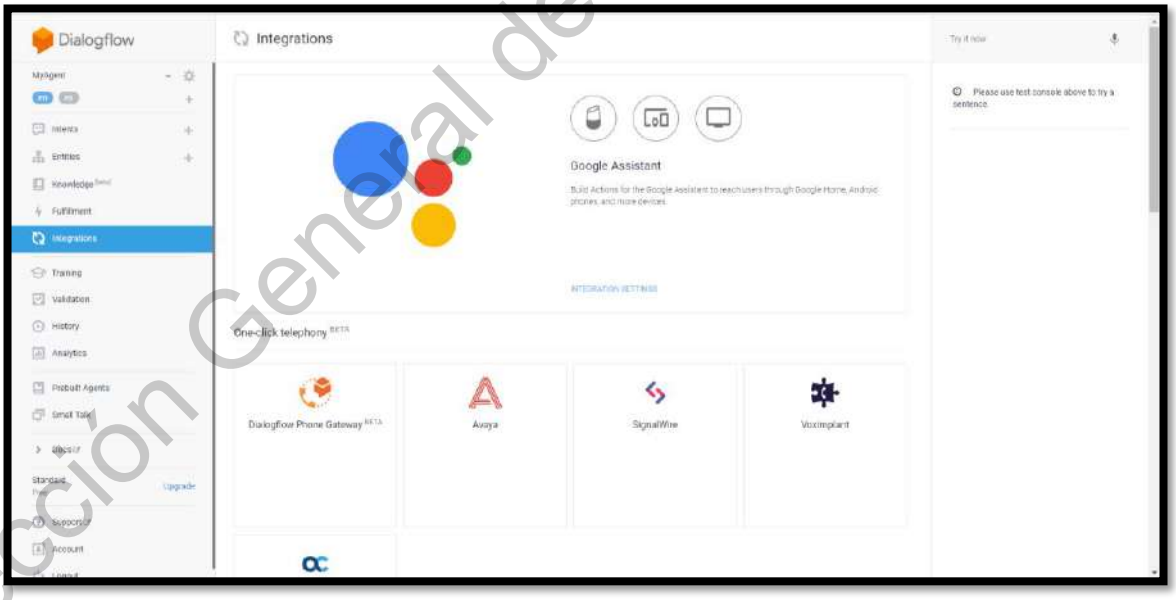

Figura 29 Integración con el asistente de Google

Al iniciar el asistente de integraciones se deben seleccionar las intenciones que serán publicadas en el asistente de Google. Se puede ver en la Figura **30** que se han seleccionado las tres intenciones desarrolladas en este proyecto.

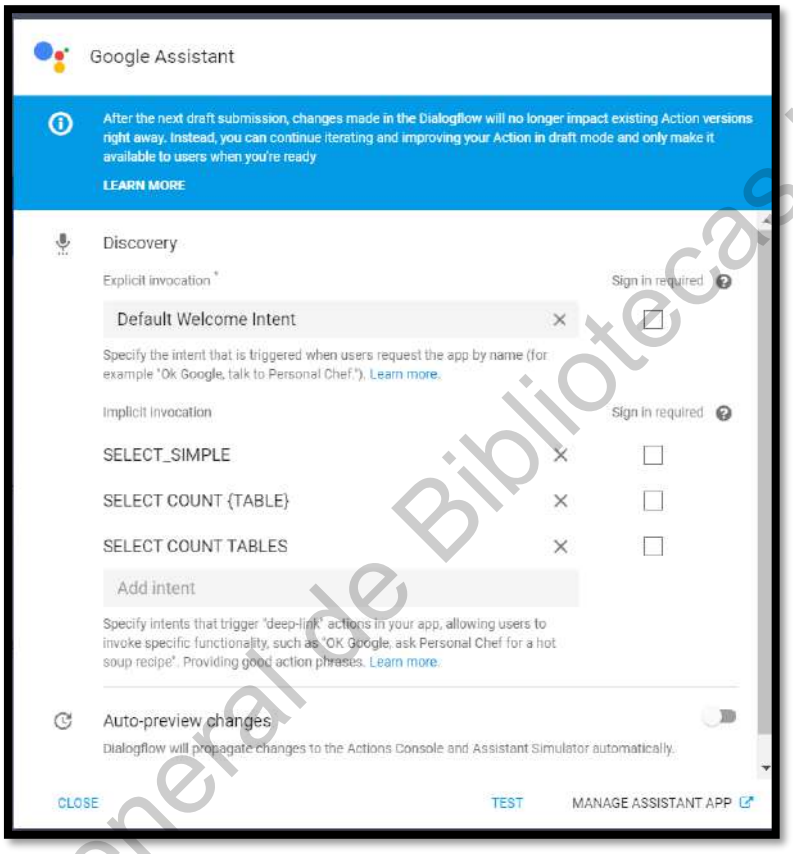

Figura 30 Intenciones en el asistente de Google

En la Figura **31** se presenta la “consola de acciones”, donde se puede observar el proyecto “MyAgent” que fue publicado en el asistente de Google. En esta consola se configuran algunos parámetros para el agente.

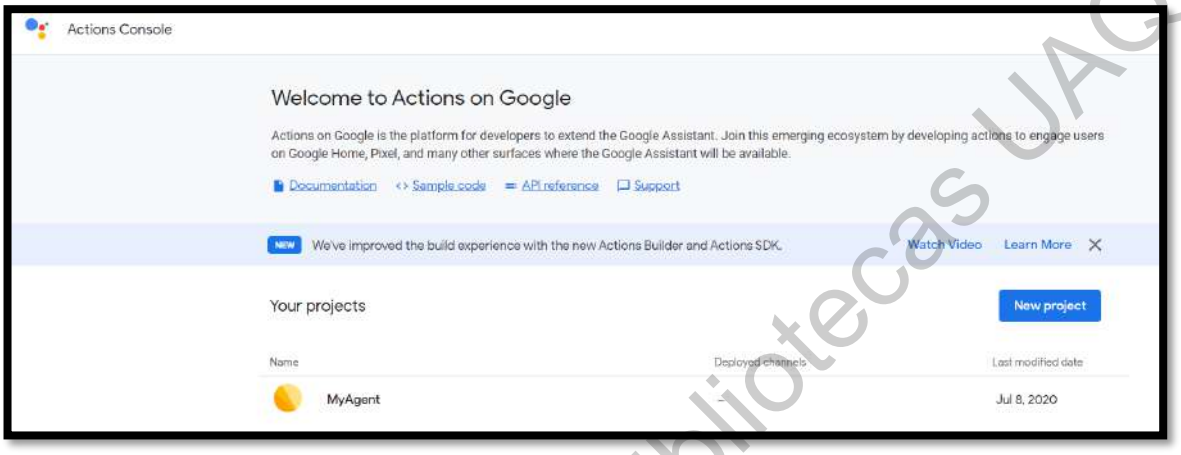

Figura 31 Acciones en Google

La consola de acciones, está divida en cinco configuraciones:

1) Overview: En esta pestaña mostrada en la Figura 32, por cada idioma que se haya definido se debe configurar cómo será invocado el agente, y se construirán las acciones para probarlas y liberarlas.

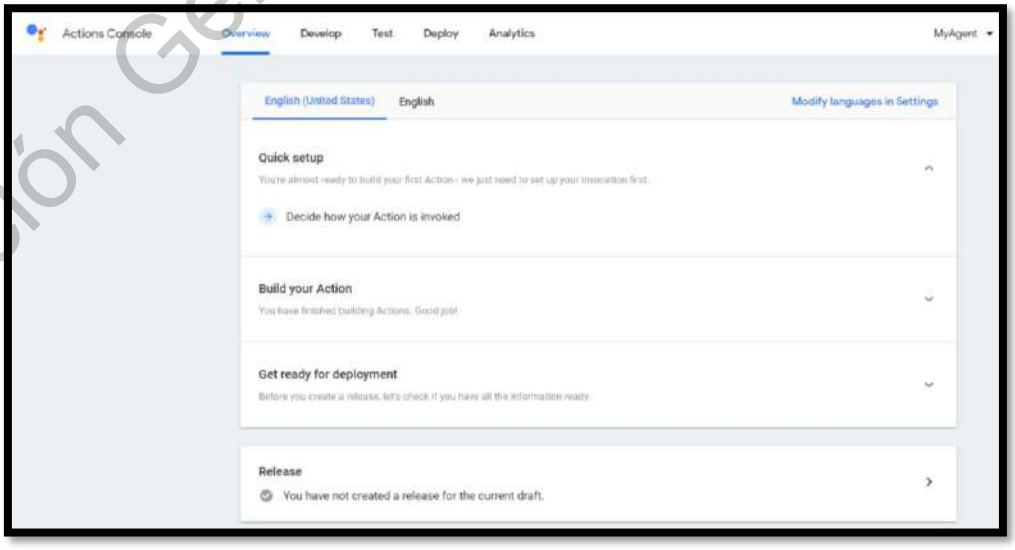

Figura 32 Consola de acciones - *Overview*

2) Develop: En esta pestaña se definen las formas de invocar al agente, así como determinar las acciones que estarán liberadas para pruebas y para la implementación. El agente se activa con el nombre “My Agent” como se puede observar en la Figura **33**.

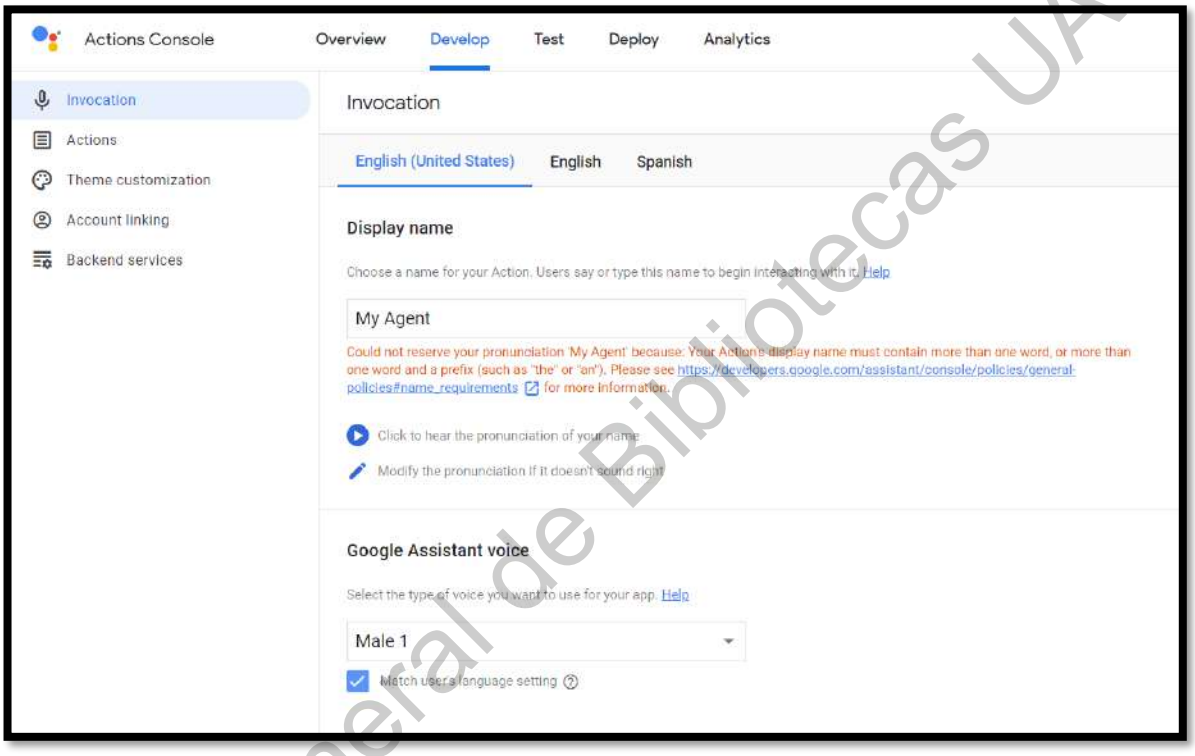

Figura 33 Consola de acciones – *Invocation*

3) Test: En esta pestaña mostrada en la Figura 34, se prueba el agente. Se puede interactuar con el agente usando voz, o tecleando directamente en texto las consultas que se requieren.

Figura 34 Consola de acciones - *Test*

4) Deploy: En la Figura 35 se definen los paramétros para la publicación.

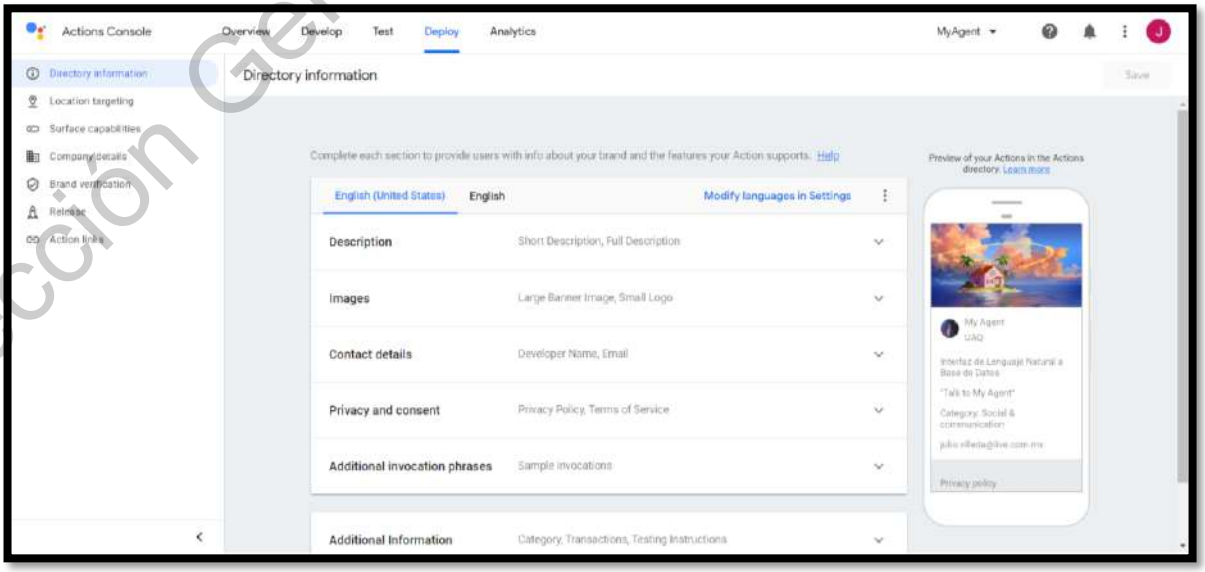

Figura 35 Consola de acciones - *Deploy*

5) Analytics: Una vez que se ha publicado el agente, en esta pestaña se puede explorar algunos indicadores relacionados al rendimiento del agente, como se muestra en la Figura 36.

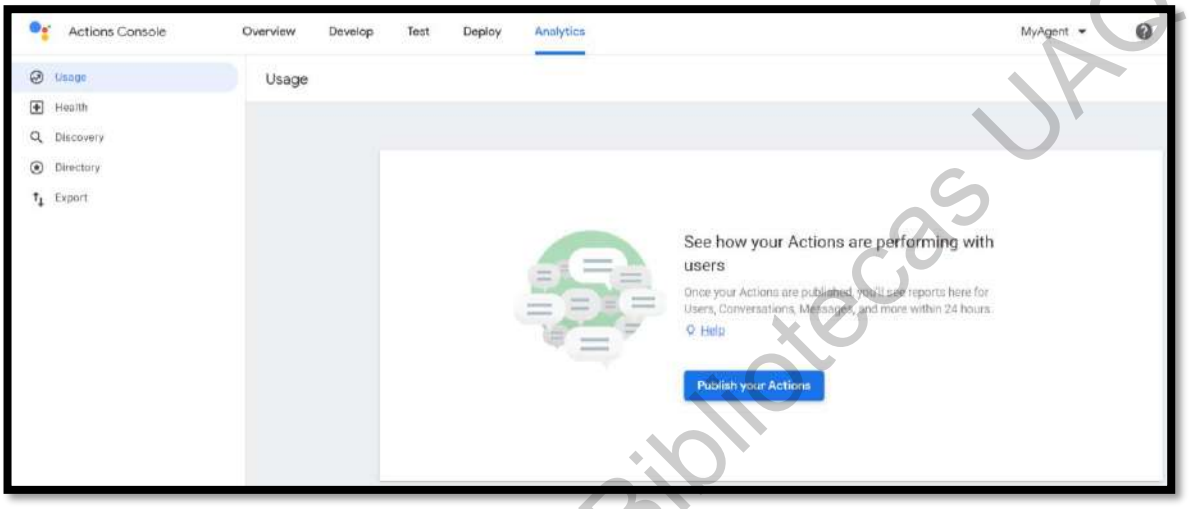

Figura 36 Consola de acciones - *Analytics*

6. RESULTADOS

Para la evaluación de la ILNBD se utilizó el emulador de DialogFlow, donde se pueden ingresar consultas de entrada utilizando voz o texto, y se evaluará la consulta, empatando con la mejor intención, como se muestra en la Figura **37**.

Figura 37 Simulador de DialogFlow

Una vez que se ha seleccionado la intención, se procede a evaluar la consulta directamente en el SGBD SQL server 2017.

Para esta evaluación se diseñó una base de datos conteniendo cuatro tablas. Es importante destacar que la definición de las tablas y sus campos están en idioma inglés.

A continuación, se muestran las cuatro tablas utilizadas en la base de datos:

1) **T_Person**: Tabla que incluye los datos personales de cualquier persona, puede ser empleado o estudiante.

2) **T_Employee**: Define los datos específicos para un empleado y está relacionada a la tabla T_Person.

3) **T_Student**: Define los datos específicos para un estudiante y también está relacionada a la tabla T_Person

4) **T_Sex**: Incluye los registros masculino o femenino.

En la Figura **38** se muestra el diagrama de base de datos.

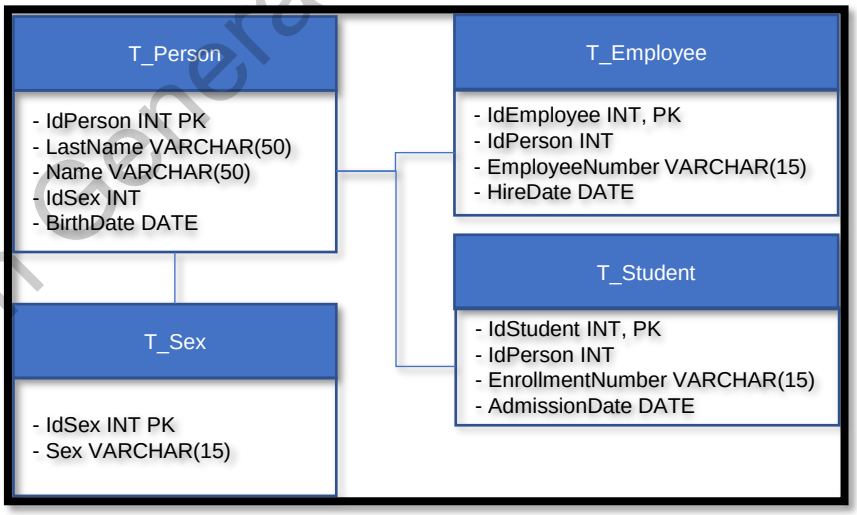

Figura 38 Diagrama de Base de Datos (Fuente: Propia)

Para evaluar la confiabilidad de la interfaz se diseñaron treinta y una preguntas en español, divididas en tres grupos:

Se realizaron dos tipos de pruebas:

1) **Con reconocimiento de voz**: Al realizar las 31 preguntas utilizando el micrófono y el lenguaje español solamente **74.19%**, 23 de 31 de las pruebas fueron correctas. En los casos erróneos, la API de reconocimiento de voz tuvo algunas deficiencias reconociendo palabras diferentes.

2) **Sin reconocimiento de voz, consultas en texto**: DialogFlow permite introducir la consulta directamente en texto, como se formularía en lenguaje natural. En este caso, los resultados fueron del **100%**, es decir, las 31 consultas fueron procesadas correctamente.

Las 31 consultas fueron divididas en 3 grupos:

1) Cinco consultas SELECT ALL simples mostradas en la Tabla **1**.

Tabla 1 Consultas SELECT ALL SIMPLE

Consulta en lenguaje natural	**Consulta generada en SQL**
¿Cuáles son los nombres de las tablas en la base de datos?	SELECT * FROM sys.tables
Muestra todas las tablas	
Dame todas las personas	SELECT * FROM T_Person
Regresa todos los empleados	SELECT * FROM T_Employee
Muestra todos los estudiantes	SELECT * FROM T_Student

2) Quince consultas SELECT simple empleando variaciones de sinónimos, plural y singular mostradas en la Tabla **2**.

Tabla 2 Consultas SELECT ALL SIMPLE utilizando sinónimos y variaciones en singular y plural

Consulta en lenguaje natural	**Consulta generada en SQL**
Persona / Personas	SELECT * FROM T_Person
Gente	
Individuo / Individuos	
Empleado / Empleados	SELECT * FROM T_Employee
Colaborador / Colaboradores	
Trabajador / Trabajadores	
Estudiante / Estudiantes	SELECT * FROM T_Student
Alumno / Alumnos	

3) Once consultas SELECT COUNT(ALL) mostrados en laTabla **3**.

Tabla 3 Consultas SELECT COUNT(ALL)

Consulta en lenguaje natural	**Consulta generada en SQL**
¿Cuántas tablas tiene la base de datos?	SELECT COUNT(*) FROM sys.tables
¿Cuántos registros tiene la tabla de Personas / Gente / Individuos?	SELECT COUNT(*) FROM T_Person
¿Cuántos registros tiene la tabla de Empleados / Colaboradores / Trabajadores?	SELECT COUNT(*) FROM T_Employee
¿Cuántos registros tiene la tabla de Estudiantes / Alumnos?	SELECT COUNT(*) FROM T_Student
¿Cuántas Personas hay? / ¿Cuánta gente hay?	SELECT COUNT(*) FROM T_Person

7. CONCLUSIONES

La API de reconocimiento de voz y conversión a texto DialogFlow ha mostrado resultados muy interesantes y promisorios para adaptarla a una ILNBD, aunque al momento sólo se han explorado tres escenarios con consultas simples. Los resultados obtenidos al momento permiten indicar que con esta API se podrá generar una ILNBD integral que habilitará a los usuarios sin experiencia en SQL a poder recuperar información de BDR. A los usuarios con experiencia les permitirá generar consultas en tiempos más cortos.

El AA de DialogFlow a través de entrenamiento permite que la interfaz sea bastante flexible respecto al dominio de la base de datos, ya que el mantenimiento del diccionario es muy ágil respecto a los nombres de las tablas y los campos, aprendiendo todas las variaciones que puedan tener los nombres de los objetos; por ejemplo, sinónimos, plurales, singulares, e incluso en el modo texto, aprender y corregir palabras escritas parcialmente correctas.

Es necesario seguir trabajando con la interfaz para integrar consultas que requieran uniones entre dos o más tablas y todas las combinaciones que se puedan tener entre ellas, tales como INNER, FULL, RIGHT, LEFT y CROSS. Asimismo, es necesario probar sentencias que contengan requerimientos de agrupación con las sentencias GROUP BY, HAVING y operaciones de grupo.

Por último, como parte del trabajo a futuro, es necesario explorar el asistente de voz integrado con el asistente de Google, para validar que el reconocimiento de voz mejore, sobre todo en pruebas realizadas directamente sobre dispositivos móviles, ya que al momento las consultas con reconocimiento de voz y DialogFlow no ofrecieron los resultados esperados. En algunos casos, la conversión de voz a texto interpretó palabras distintas a las que el usuario había indicado verbalmente.

8. REFERENCIAS BIBLIOGRÁFICAS

Andics, A., McQueen, J. M., Petersson, K. M., Gál, V., Rudas, G., & Vidnyánszky, Z. (2010). Neural mechanisms for voice recognition. *NeuroImage*. https://doi.org/10.1016/j.neuroimage.2010.05.048

Bais, H., Machkour, M., & Koutti, L. (2017). An independent-domain natural language interface for relational database: Case Arabic language. *Proceedings of IEEE/ACS International Conference on Computer Systems and Applications, AICCSA*. https://doi.org/10.1109/AICCSA.2016.7945786

Bird, S., Klein, E., & Loper, E. (2009). Language Processing and Python. *Computing*.

Dang, N. T., Thi, D., & Tuyen, T. (2009). Document retrieval based on question answering system. *2009 2nd International Conference on Information and Computing Science, ICIC 2009*. https://doi.org/10.1109/ICIC.2009.53

del Rio, M. M., & Castillo, S. J. L. (2015). *Implementation of prototype of natural language interface to database*. https://doi.org/10.1109/cisti.2015.7170474

Ghosh, P. K. (2014). Automatic SQL Query Formation from Natural Language Query. *IInternational Journal Of Computer Application*.

Godoc, E. (2014). *SQL Los fundamentos del lenguaje*.

Google. (2019a). Agents overview. Retrieved from https://dialogflow.com/docs/intro/agents?authuser=1

Google. (2019b). How does fulfillment work? Retrieved from https://dialogflow.com/docs/intro/fulfillment?authuser=1

Hendrix, G. G., Sacerdoti, E. D., Sagalowicz, D., & Slocum, J. (1978). Developing a natural language interface to complex data. *ACM Transactions on Database Systems*. https://doi.org/10.1145/320251.320253

Koehn, P. (2010). Enabling Monolingual Translators : Post-Editing vs . Options.

Human Language Technologies The 2010 Annual Conference of the North American Chapter of the ACL.

Kumar, R., & Dua, M. (2014). Translating controlled natural language query into SQL query using pattern matching technique. *2014 International Conference for Convergence of Technology, I2CT 2014*. https://doi.org/10.1109/I2CT.2014.7092161

Li, Y., Yang, H., & Jagadish, H. V. (2005). NaLIX: an interactive natural language interface for querying XML. *SIGMOD*. https://doi.org/10.1145/1066157.1066281

Mohite, A., & Bhojane, V. (2015). Natural language interface to database using modified co-occurrence matrix technique. *2015 International Conference on Pervasive Computing: Advance Communication Technology and Application for Society, ICPC 2015*. https://doi.org/10.1109/PERVASIVE.2015.7087045

Posevkin, R., & Bessmertny, I. (2015). Translation of natural language queries to structured data sources. *9th International Conference on Application of Information and Communication Technologies, AICT 2015 - Proceedings*. https://doi.org/10.1109/ICAICT.2015.7338516

Rendón, Y. A. (2019). Bases de datos relacionales vs. no relacionales. Retrieved June 4, 2020, from https://www.pragma.com.co/academia/lecciones/bases-de-datos-relacionales-vs.-no-relacionales

Reshma, E. U., & Remya, P. C. (2018). A review of different approaches in natural language interfaces to databases. *Proceedings of the International Conference on Intelligent Sustainable Systems, ICISS 2017*. https://doi.org/10.1109/ISS1.2017.8389287

Tennant, H. R., Ross, K. M., & Thompson, C. W. (2003). *Usable natural language interfaces through menu-based natural language understanding*.

https://doi.org/10.1145/800045.801601

Uma, M., Sneha, V., Sneha, G., Bhuvana, J., & Bharathi, B. (2019). Formation of SQL from natural language query using NLP. *ICCIDS 2019 - 2nd International Conference on Computational Intelligence in Data Science, Proceedings*. https://doi.org/10.1109/ICCIDS.2019.8862080

Wada, Y., Watanabe, Y., Syoubu, K., Sawamoto, J., & Katoh, T. (2010). Virtual database technology for distributed database. *24th IEEE International Conference on Advanced Information Networking and Applications Workshops, WAINA 2010*. https://doi.org/10.1109/WAINA.2010.38

Wei, L. L., & Zhang, W. (2009). A method for rough relational database transformed into relational database. *Proceedings - 2009 IITA International Conference on Services Science, Management and Engineering, SSME 2009*. https://doi.org/10.1109/SSME.2009.79

Wong, Y. W., & Mooney, R. J. (2006). Learning for semantic parsing with statistical machine translation. *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics* -. https://doi.org/10.3115/1220835.1220891

Yeo, C. Y., Al-Haddad, S. A. R., & Ng, C. K. (2011). Animal voice recognition for identification (ID) detection system. *Proceedings - 2011 IEEE 7th International Colloquium on Signal Processing and Its Applications, CSPA 2011*. https://doi.org/10.1109/CSPA.2011.5759872

Zhong, V., Xiong, C., & Socher, R. (2017). *Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning*. Retrieved from http://arxiv.org/abs/1709.00103