Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Grado en Ingeniería Informática

ETSIIBot: Asistente Conversacional para la Escuela Superior de Ingeniería Informática

ETSIIBot: Conversation Assistant

Kevin Díaz Marrero

La Laguna, 10 de junio de 2019

Dña. **María Elena Sánchez Nielsen**, con N.I.F. 42.848.599-J profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutora.

C E R T I F I C A

Que la presente memoria titulada:

"ETSIIBot: Asistente Conversacional para la Escuela Superior de Ingeniería Informática"

ha sido realizada bajo su dirección por D. **Kevin Díaz Marrero**,

con N.I.F. 43.389.751-K.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 10 de junio de 2019.

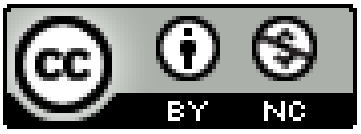
Agradecimientos

En primer lugar, agradecer a mi tutora el apoyo aportado durante el desarrollo del proyecto. Tanto a la hora de resolver dudas, como a la hora de aportar ideas que pudiera añadir.

También agradecer a compañeros, familia y amigos su indudable apoyo a lo largo de estos años, aportando la fuerza necesaria para continuar.

Muchas gracias a todos.

Licencia



Resumen

El objetivo de este trabajo ha sido desarrollar un asistente capaz de facilitar información a los estudiantes de Ingeniería Informática, mediante el filtrado de información de la página web de la Universidad de La Laguna, permitiendo realizar la consulta tanto por voz como de forma escrita.

Para el desarrollo de este proyecto se ha hecho uso de una plataforma Dialogflow, que mediante técnicas de reconocimiento del lenguaje natural y Machine Learning permite procesar las solicitudes del usuario y responder en consecuencia a lo que nos indique.

Finalmente, cuando nuestro asistente funcione correctamente podemos hacer uso de APIs para poder utilizarlo en varias plataformas y redes sociales, pero en este caso nos centraremos en Google Assistant, que nos permite interactuar con nuestro asistente mediante voz. Para hacer uso de esta API utilizaremos el lenguaje de programación JavaScript.

Palabras clave: Dialogflow, reconocimiento del lenguaje natural, Machine Learning, JavaScript, API, Google Assistant.

Abstract

The main goal of this project implies the development of an assistant capable of supply information to the Computer Engineering students, by filtering information from the website of the University of La Laguna, allowing the query by voice or written form.

For the development of this project we have used Dialogflow platform, that by means of techniques of recognition of the natural language and Machine Learning allows to process the user's requests and respond accordingly to what the user says.

When our assistant finally work correctly, we can use APIs in order to use it in many platforms and social social networks, but in this case we will focus on Google Assistant, that allow us to interact with aur assistant by voice. To use this API we will use the programming language JavaScript.

Keywords: Dialogflow, Natural Lenguaje Processing, Machine Learning, JavaScript, API, Google Assistant.

Índice general

Introducción **10**
- Antecedentes y estado del arte 10
  - Metodologías utilizadas 10
  - Plataformas de desarrollo de chatbot 11
- Funcionamiento del proyecto 12
- Objetivos 13

Plan de trabajo y tecnologías utilizadas **14**
- Plan de trabajo 14
- Cronograma 16

Dialogflow **17**
- Primeros pasos 17
  - Prioridad 18
  - Intent name 18
  - Context 18
  - Events 19
  - Training phrases 19
  - Actions and parameters 19
  - Response 19
  - Fulfillment 20
- Entities 20
- Fulfillment 20
  - Webhook 20
  - Inline editor 21
- Training 21
- History 21

Google Assistant **22**
Nombre de invocación 22
Información adicional 23
Uso del asistente en dispositivos móviles 24
Problemas encontrados **27**
Autenticación 27
Versión 2 de la API 27
Asignación de parámetros 27
Formato de las guías docentes 31
Disponibilidad de las salas 32
Consultas simples 33
Existencia de una API 36
Formato universal de guías docentes 37
Summary and Conclusions **38**
Existence of an api 38
Universal format of teaching guides 38
Presupuesto **39**

Índice de figuras

Figura 1: Funcionamiento del proyecto 12
Figura 2: Diagrama de flujo de conversación 15
Figura 3: Menú principal de Dialogflow 17
Figura 4: Prioridades de las intenciones 18
Figura 5: Ejemplo de contexto 18
Figura 6: Entidad curso 20
Figura 7: Validar respuesta 21
Figura 8: Nombre de Google Assistant 22
Figura 9: Descripción del asistente 23
Figura 10: Nombres de invocación 23
Figura 11: Visualización en móvil 24
Figura 12: Asistente en móvil simulado 25
Figura 13: Visualización de respuesta en el simulador 25
Figura 14: Visualización de respuesta en el móvil 26
Figura 15: Migración a versión 2 24
Figura 16: Entrenamiento con parámetro 27
Figura 17: Asignación nombre de parámetro 27
Figura 18: Asignación correcta del valor al parámetro 28
Figura 19: Código de ejemplo para la versión 1 29
Figura 20: Código versión 2 30
Figura 21: Función de cuatrimestre 30
Figura 22: Función del día de la semana 31
Figura 23: Tarjetas de servicios 32
Figura 24: Ejemplo de formato de tutorías 29
Figura 25: Estructuras para almacenar las tutorías 34
Figura 26: Asignación de información de tutorías 35
Figura 27: Mostrar tutorías 35
Figura 28: Diagrama de flujo mejorado 37

Índice de tablas

Tabla 1: Cronograma del proyecto 16
Tabla 2: Presupuesto del proyecto 39

Capítulo 1 Introducción

1.1 Antecedentes y estado del arte

En la actualidad la inteligencia artificial está viviendo la expansión de la tecnología de los asistentes virtuales, lo que implica una mejora tanto de los servicios digitales como de la interacción entre los usuarios y los sistemas de información. Estos asistentes deben ser capaces de entender el lenguaje natural y por ese motivo es necesario utilizar algunas de las siguientes metodologías.

1.1.1 Metodologías utilizadas

A la hora de desarrollar un asistente debemos tener en cuenta el proceso por el que pasa la información para poder ser interpretada correctamente. El primer paso es interpretar la información que nos transmite el usuario, dándole significado a las expresiones utilizadas, diferenciando datos y relacionando la información con los servicios que presta, a este proceso lo llamamos análisis semántico. Además, es necesario realizar también un análisis de sentimientos si queremos comprender mejor a cada usuario individualmente, este método es el encargado de diferenciar y clasificar las intenciones y peticiones del usuario, un ejemplo de plataforma que se encarga de este tipo de análisis es MonkeyLearn.

Una vez realizados estos análisis, se le aplican diferentes algoritmos de inteligencia artificial para potenciar la comprensión de nuestro chatbot, ejemplos de ello son las siguientes [1]:

- **Procesamiento del lenguaje natural:** es la unión de ciencias computacionales con logran la lingüística aplicada con el fin de permitir a un programa comprender y procesar la información transmitida por el usuario para tareas concretas como traducción y análisis de opinión. Las dificultades de este proceso las encontramos con ambigüedades a nivel léxico (palabras polisémicas o en diferente contexto), referencias a información previamente mencionada y uso de entidades lingüísticas como pueden ser las metáforas.
- **Comprensión del lenguaje natural:** es una rama perteneciente a procesamiento del lenguaje natural encargada de interpretar y analizar la información proporcionada por el usuario, consiguiendo extraer conceptos, relaciones, entidades, categorías, palabras clave, pero esto se simplifica si únicamente requerimos terminología de un ámbito en concreto.
- **Machine Learning:** se basa en la localización de patrones de conducta generados por los usuarios para poder discernir los diferentes grupos de usuarios posibles, todo ello aprendiendo de la información suministrada de antemano, pudiendo posteriormente corregir y enseñarle de forma manual a base de entrenamiento. Podemos esta técnica en tres grupos:

- **Aprendizaje supervisado:** este tipo de aprendizaje se basa en entrenar al sistema suministrando una cantidad de datos clasificándolos como deseamos que lo haga. Una vez se le ha dado la cantidad de datos suficiente, el sistema debería ser capaz de clasificar datos nuevos debido a los patrones que haya encontrado mientras entrenaba.
- **Aprendizaje no supervisado o Deep Learning [2]:** este algoritmo de aprendizaje se basa en el uso de redes neuronales, para optimizar tareas como reconocimiento de imagen, voz, o incluso procesamiento del lenguaje natural. Esta red neuronal está compuesta por diferentes niveles jerárquicos, incrementando la complejidad al avanzar en ella. A los primeros nodos de esta red se les proporciona la información a procesar, y tras operar con ella se le comunica el resultado a la siguiente capa, y así hasta llegar a la última proporcionandonos una respuesta. A diferencia del anterior, este aprendizaje no requiere tanta supervisión, pero a cambio requiere una gran cantidad de tiempo para ir evolucionando.
- **Aprendizaje por refuerzo:** esta técnica utiliza un sistema de valores que penaliza o premia en función de las decisiones de nuestro sistema, permitiendo así aprender basándose en la experiencia. Básicamente es utiliza prueba y error pero priorizando aquellas decisiones más acertadas.

1.1.2 Plataformas de desarrollo de chatbot

Cada vez son más las empresas que se animan a desarrollar su propia interfaz conversacional, permitiendo a muchos desarrolladores comenzar sus proyectos con una mejor base debido a la infraestructura ya generada, ejemplos de estas plataformas son:

- **Microsoft:** contenido en Azure (su plataforma en la nube) ofrecen una herramienta para desarrollar chatbots, además de la tecnología llamada *Luis* (Language Understanding Intelligence Service), que permite entrenar a nuestro agente en conversaciones modeladas utilizando servicios de procesamiento de lenguaje natural y análisis lingüístico.
- **Google:** ofreciendo la API Natural Language sobre su plataforma Cloud, permite analizar la estructura y significado mediante Machine Learning. Esta API REST se utiliza para contrastar la información contenida en documentos, artículos o publicaciones. Por último, en 2016 API.ai fue comprada por Google (renombrando esta plataforma como DialogFlow), una plataforma para el desarrollo de chatbots, que ha tenido gran impacto debido a su capacidad de procesamiento del lenguaje natural.
- **Amazon:** desarrollando Lex, un servicio diseñado para la creación de conversaciones basado en Deep Learning aplicado al reconocimiento automático del habla, convirtiendo mensajes dictados en texto y comprensión del lenguaje natural, pudiendo interpretar el texto finalmente. Para el desarrollo de chatbots se creó Alexa, permitiendo a los desarrolladores utilizar esta tecnología.

1.2 Funcionamiento del proyecto

Para comprender el desarrollo del proyecto es necesario diferenciar las partes que lo componen, podemos dividirlo en las siguientes tres etapas.

1.2.1 Interfaz de usuario

Es la encargada procesar y comunicar la información introducida por el usuario a la plataforma en la que se procese el mensaje, en este caso la interfaz de usuario puede ser un móvil ejecutando Google Assistant, el usuario introduce por escrito o la dice en voz alta, ese audio o texto es enviado a una plataforma que posteriormente responderá con la información que se ha solicitado.

1.2.2 Procesamiento de la información

La información introducida se analiza y clasifica conforme a los servicios que nuestro asistente ofrece, y una vez localizada la intención de la petición del usuario se realizan conexiones con aplicaciones externas si fuera necesario (por ejemplo si el usuario quiere hacer una reserva, se debe ejecutar el código que realice esa reserva).

1.2.3 Conexión con aplicaciones externas

Una vez reconocida la petición solicitada por el usuario, se realizarán las acciones necesarias para llevar a cabo su solicitud, enviando y/o recibiendo información de una plataforma externa, concluyendo finalmente con una respuesta al usuario con el resultado.

Podemos ver lo explicado anteriormente en la siguiente imagen:

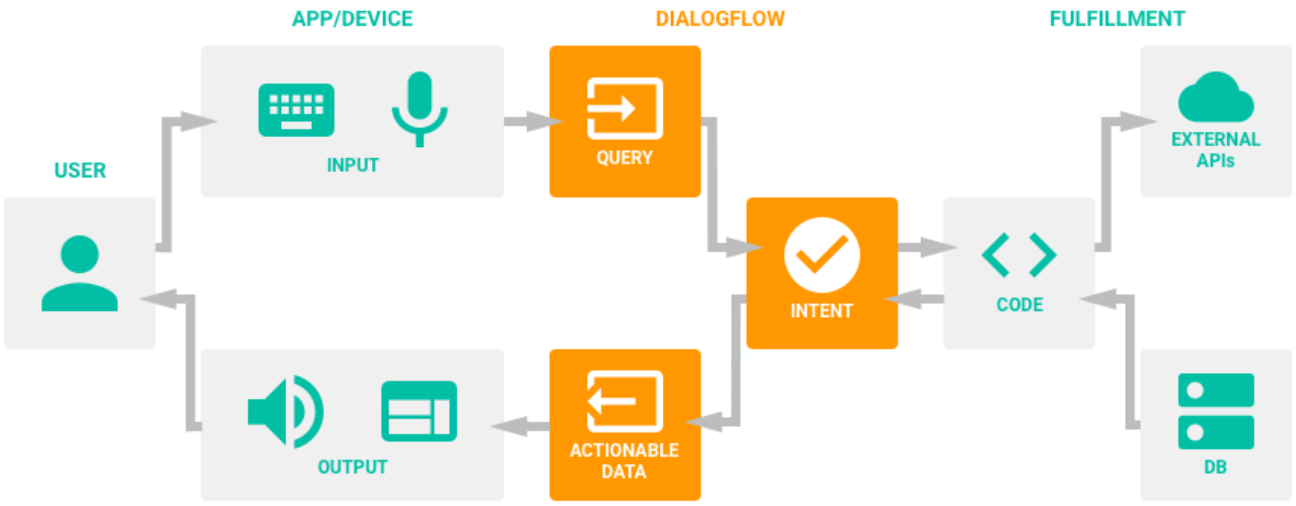


Figura 1: Funcionamiento del proyecto. [3]

1.3 Objetivos

La finalidad del presente proyecto es el empleo de las herramientas y plataformas existentes para el desarrollo e implementación de un bot conversacional capaz de interpretar el lenguaje natural y proporcionar, automáticamente, al usuario la información que solicite sobre servicios básicos, en el contexto de la Escuela Superior de Ingeniería Informática. Podemos dividir este proyecto en las diferentes etapas:

1. Seleccionar los servicios que podrá manejar nuestro asistente.
2. Desarrollar el diagrama de flujo de la conversación.
3. Implementar las diferentes respuestas que podrá proporcionar nuestro asistente.
4. Corregir aquellas respuestas que no correspondan con la intención real del usuario.
5. Enlazar nuestro asistente con la API de Google Assistant.

Capítulo 2 Plan de trabajo y tecnologías utilizadas

2.1 Plan de trabajo

Elegimos Dialogflow como plataforma de desarrollo ya que está optimizada para la integración con Google Assistant, pero a su vez nos ofrece un amplio abanico de posibilidades de integración, pero aún no habíamos definido la información que va a manejar nuestro asistente, así que después de revisar la página de la universidad y también la de la facultad, realizamos una lista con los servicios utilizados con más frecuencia por los alumnos. Una vez conseguida esta lista pensamos cómo podría procesar la información nuestro asistente y a su vez, una forma de filtrar y responder únicamente con la parte más útil. Eso finalmente dio lugar al siguiente diagrama de flujo:

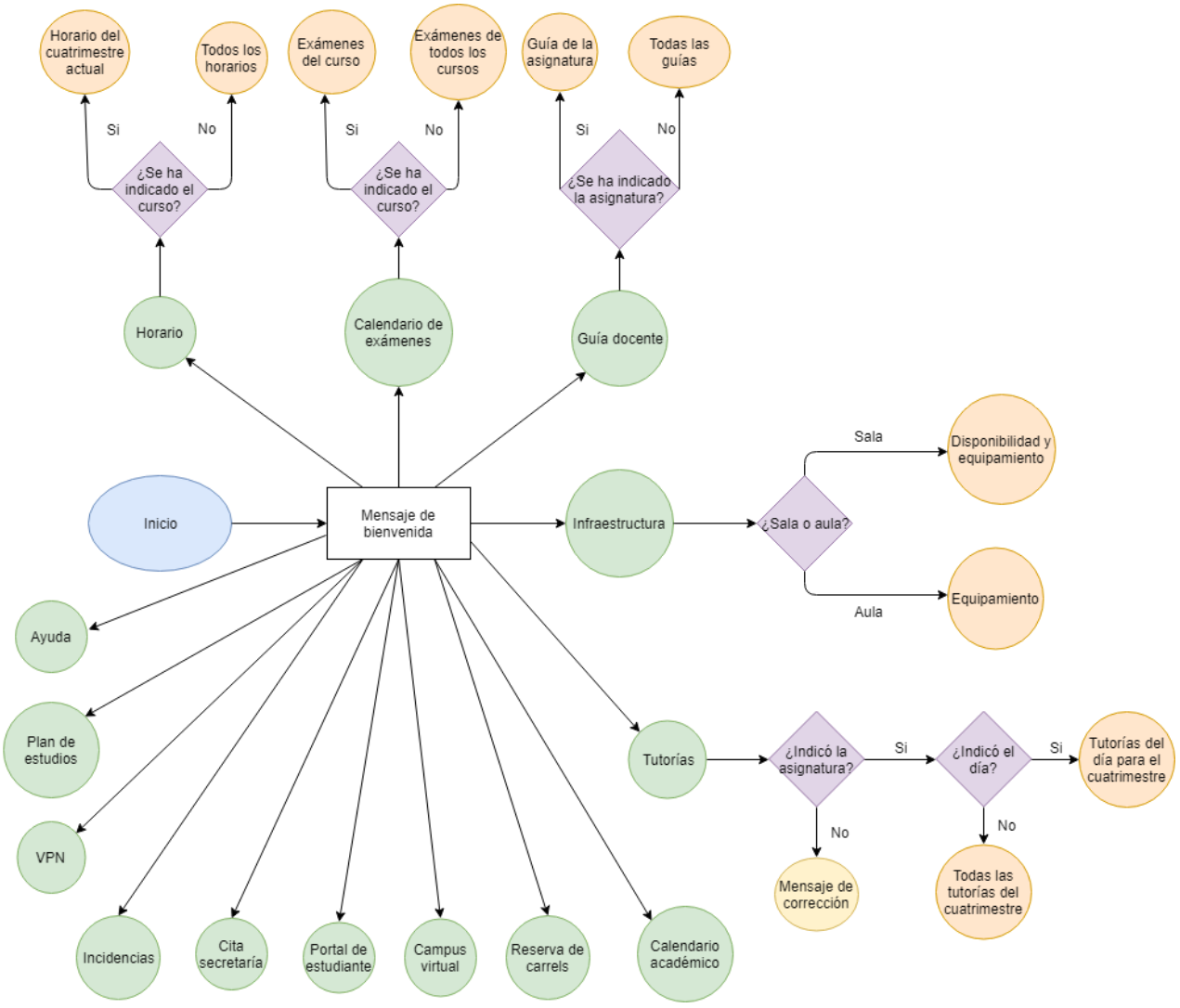


Figura 2: Diagrama de flujo de conversación

Una vez realizado el diagrama de flujo debemos ver las diferentes formas en las que el usuario puede solicitar cada uno de los servicios, surgiendo variaciones de respuesta en función de la información que se nos proporciona, y una vez nuestro agente esté entrenado podemos integrar nuestro proyecto en Google Assistant.

2.2 Cronograma

El cronograma final del proyecto fue el siguiente:

Inicio	Fin	Actividad
12-12-2018	28-01-2019	Selección de servicios que proporcionará nuestro asistente.
01-02-2019	15-02-2019	Desarrollar el diagrama de flujo que seguirá el asistente.
16-02-2019	02-04-2019	Comienzo del desarrollo del asistente.
03-04-2019	16-04-2019	Entrenamiento del asistente, mejoras y corrección de errores.
18-04-2019	15-05-2019	Integración con Google Assistant
16-05-2019	05-06-2019	Evaluación final del asistente y realización de memoria.

Tabla 1: Cronograma del proyecto.

Capítulo 3 Dialogflow

3.1 Primeros pasos

[4] Dialogflow es una plataforma perteneciente a Google, esta nos permite desarrollar chatbots, inicialmente solo deberemos indicar el nombre del proyecto, el lenguaje que utilizará el usuario y la zona horaria. Una vez creado nos saldrá un menú como el siguiente:

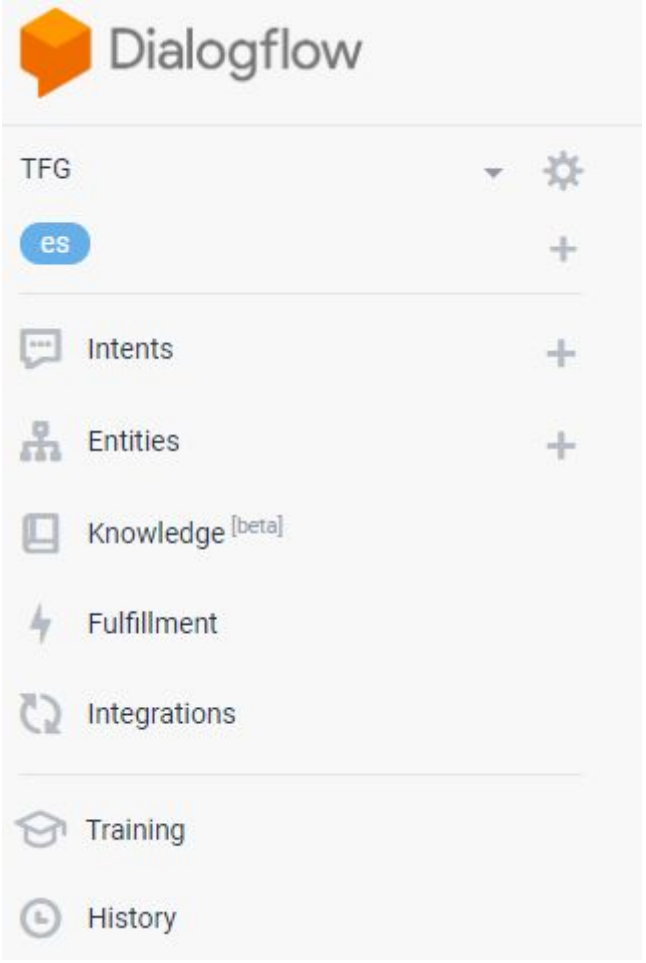


Figura 3: Menú principal de Dialogflow.

3.2 Intents

En este apartado podemos diferenciar las posibles peticiones que nos enviarán los usuarios, en la figura 1 vimos las diferentes intenciones representadas con un círculo verde. Una vez tengamos clara la lista de servicios que dará nuestro asistente deberemos

rellenar varios campos necesarios.

3.2.1 Prioridad

Dentro de nuestras intenciones podemos hacer que unas se apliquen con mayor prioridad que otras, para ello seguiremos el siguiente código de colores:

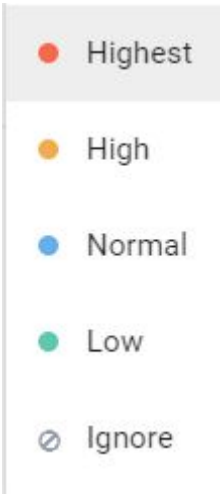


Figura 4: Prioridades de las intenciones.

3.2.2 Intent name

Es el nombre con el que podemos hacer referencia a esta intención en concreto, este nombre será necesario a la hora de desarrollar el código en el apartado *Fullfilment* (apartado 3.5).

3.2.3 Context

Los contextos son el campo que nos indica que es lo que ha realizado el usuario previamente, cada intención creada puede tener contextos de entrada, lo que significa que para llegar hasta esa intención, anteriormente ha tenido que pasar por una intención que le sitúe en dicho contexto. Por ejemplo:

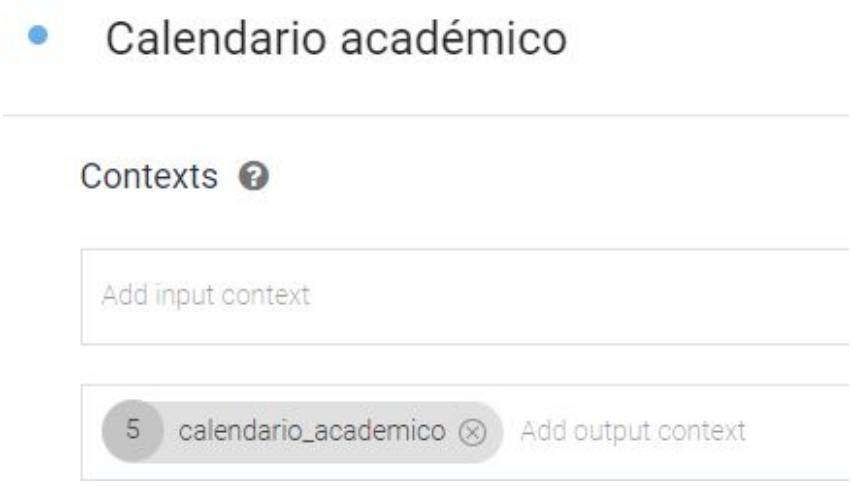


Figura 5: Ejemplo de contexto.

Aquí podemos ver que para la intención de “Calendario académico” no se requiere ningún contexto en particular, pero como salida nos situaremos en el contexto “calendario_academico”. Si realizamos una consulta a nuestro asistente podemos ver que esto también se refleja en su respuesta.

3.2.4 Events

En este apartado podemos definir que al activar esta intención se desencadene una llamada a una función para conectar nuestro chatbot a un servidor externo que posteriormente responderá con otra información que podemos asociar a otra intención. Este apartado es útil si nuestro asistente necesita consultar información externamente.

3.2.5 Training phrases

Aquí podemos indicar que debe decir el usuario para ser asociado a esta intención, podemos definir una lista de posibilidades y se entrenará al asistente en base a esas entradas indicadas. Por ejemplo, para la intención de “Campus virtual” hemos utilizado las siguientes frases:

- aula virtual
- moodle
- aula virtual
- campus virtual
- campus

Estas frases son insensibles entre mayúsculas, minúsculas, tildes, he incluso en ocasiones pueden estar mal escritas y nuestro bot podrá entenderlas igual.

3.2.6 Actions and parameters

En función de las frases indicadas podemos definir partes de esa entrada como parámetros de nuestra intención, los cuales tendrán tipología por defecto, aunque posteriormente explicaremos como crear nuestros propios tipos de parámetros. Estos parámetros podrán de carácter obligatorio si así lo deseamos y tendrán asignado un nombre. Un ejemplo de esto podría ser el caso del horario, teniendo la posibilidad de parametrizar el curso para poder transmitir una información filtrada.

3.2.7 Response

[5] Esta es una de las partes vitales de nuestro asistente, la respuesta que recibirá el usuario, se nos permite indicar una lista de respuestas posibles para que nuestro asistente no resulte tan repetitivo en el caso de algunas consultas. También se puede activar que el chatbot cierre la conversación, esta opción es bastante práctica si queremos que una vez realizada una acción nuestro asistente de por concluida la interacción.

3.2.8 Fulfillment

[6] Este último apartado es el que nos proporciona la oportunidad de enlazar esta intención con código desarrollado por nosotros y así poder integrar nuestro asistente en alguna de las plataformas disponibles, es opcional activarlo, pero en nuestro caso lo hemos activado en todas las intenciones para poder tener respuestas utilizando Google Assistant.

3.3 Entities

[7] Las entidades son aquellos datos que nos facilita el usuario y nos permiten filtrar la información que le facilitamos o nos ayuda a diferenciarla. Podemos crear un tipo de dato y decir los valores asignables a este nuevo tipo, además de añadir sinónimos que serán interpretables por un mismo valor posteriormente. Otra opción a destacar es la de permitir expandirse automáticamente, en base a los valores de entrada que le asigne el usuario podrá definir más valores de los que declaremos inicialmente.

Un ejemplo de entidad en nuestro caso podría ser el curso en el caso de los horarios:

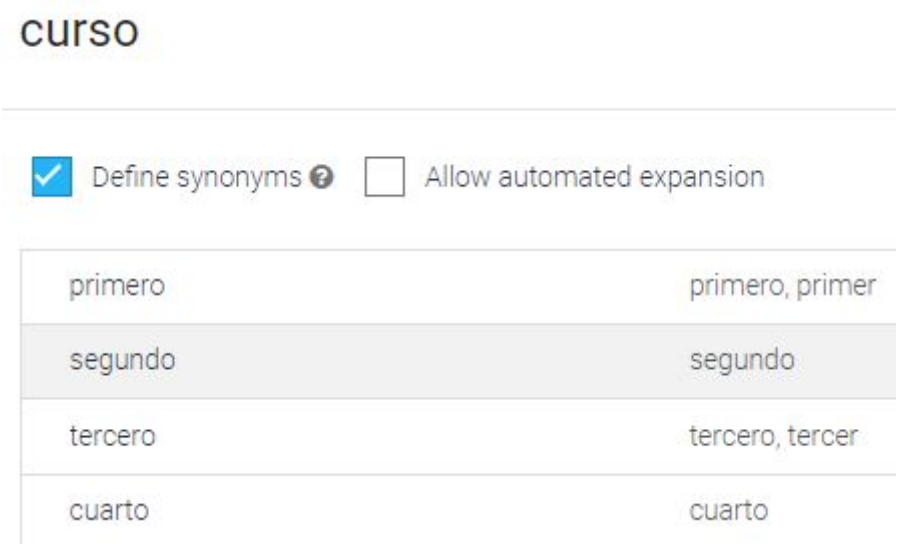


Figura 6: Entidad curso.

3.4 Fulfillment

[8] Hasta ahora solo habíamos explicado cómo reconocer intenciones, tipos de datos, parámetros, pero para que nuestro bot pueda filtrar, conectar con APIs, realizar peticiones y sobretodo conseguir integrarse en plataformas es necesario este apartado, que consta de dos opciones, las cuales no se pueden activar de forma simultáneamente.

3.4.1 Webhook

Si lo que necesitamos es que nuestro asistente únicamente realice peticiones POST, esta es la opción que nos conviene activar, nos solicitará la URL a la que realizar la petición y algunos parámetros necesarios para realizarla.

3.4.2 Inline editor

[9] Esta opción es la que nos permite utilizar una API para poder programar diferentes comportamientos en función de la intención en la que estemos y a su vez nos da la opción de personalizar respuestas para nuestra plataforma de integración (en este caso Google Assistant). Se nos proporciona también un código de ejemplo para que podamos comprender el funcionamiento de las funciones existentes además de un fichero JSON con las dependencias necesarias para comenzar nuestro proyecto.

En caso de que nuestro chatbot se cierre inesperadamente podemos revisar la consola con los logs de ejecución para depurar nuestro código con facilidad.

3.5 Training

Cuando nuestro asistente se equivoque o no sepa qué intención asignar, podemos hacer uso de este apartado, que nos permite revisar el histórico de las conversaciones con nuestro asistente y reasignar la respuesta para que posteriormente lo realice correctamente, es bastante útil cuando varias intenciones son similares y dan lugar a ambigüedad. De igual forma, podríamos crear un fichero con las entradas, subirlo a la plataforma y que nuestro bot aprenda en base a esas entradas en lugar de introducirlas de forma manual una a una. La interfaz para validar las respuestas es la siguiente:

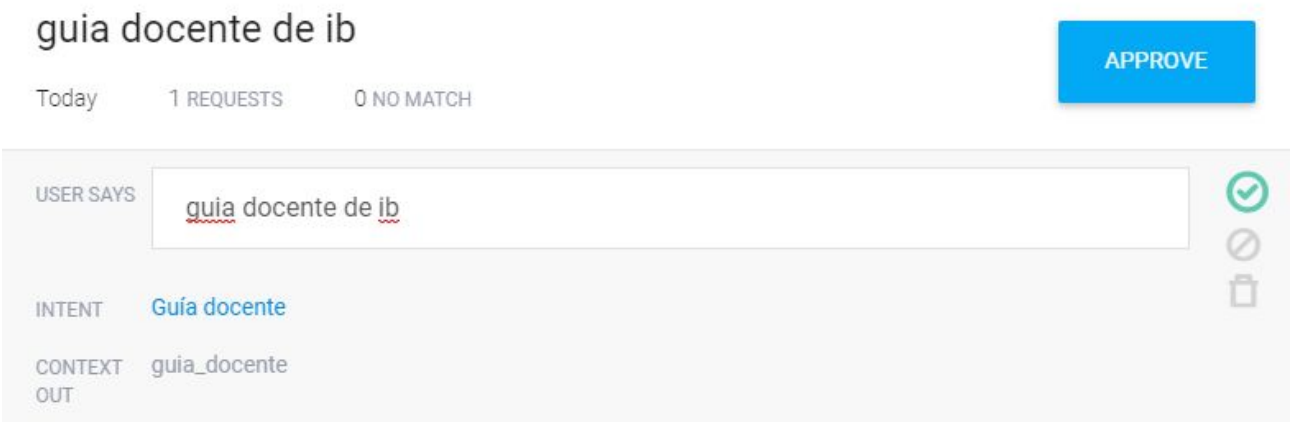


Figura 7: Validar respuesta.

En la parte derecha podemos elegir entre aprobar, rechazar o eliminar del entrenamiento en caso de que hayamos modificado algún comportamiento y queramos que ello no influya en futuras comparaciones.

3.6 History

Este campo es similar al explicado anteriormente con la diferencia de permitirnos analizar los eventos activados en cada parte de la conversación y además permitirnos filtrar por plataforma utilizada.

Capítulo 4 Google Assistant

4.1 Nombre de invocación

Una vez desarrollado nuestro chatbot lo enlazaremos con Google Assistant para poder usarlo en cualquier dispositivo compatible, para ello deberemos registrar nuestro proyecto en esta plataforma. Lo primero que se nos pide es que se le asigne un nombre, este será el nombre que tendrá nuestro chatbot en la plataforma, en nuestro caso lo hemos llamado ETSII Bot, y posteriormente nos permite elegir el tipo de voz con el que nos atenderá, podemos elegir entre dos voces masculinas y dos voces femeninas. Tras pedir la opinión de algunos alumnos nos decantamos por asignarle un tipo de voz en concreto, aquí podemos observar la personalización de lo comentado anteriormente:

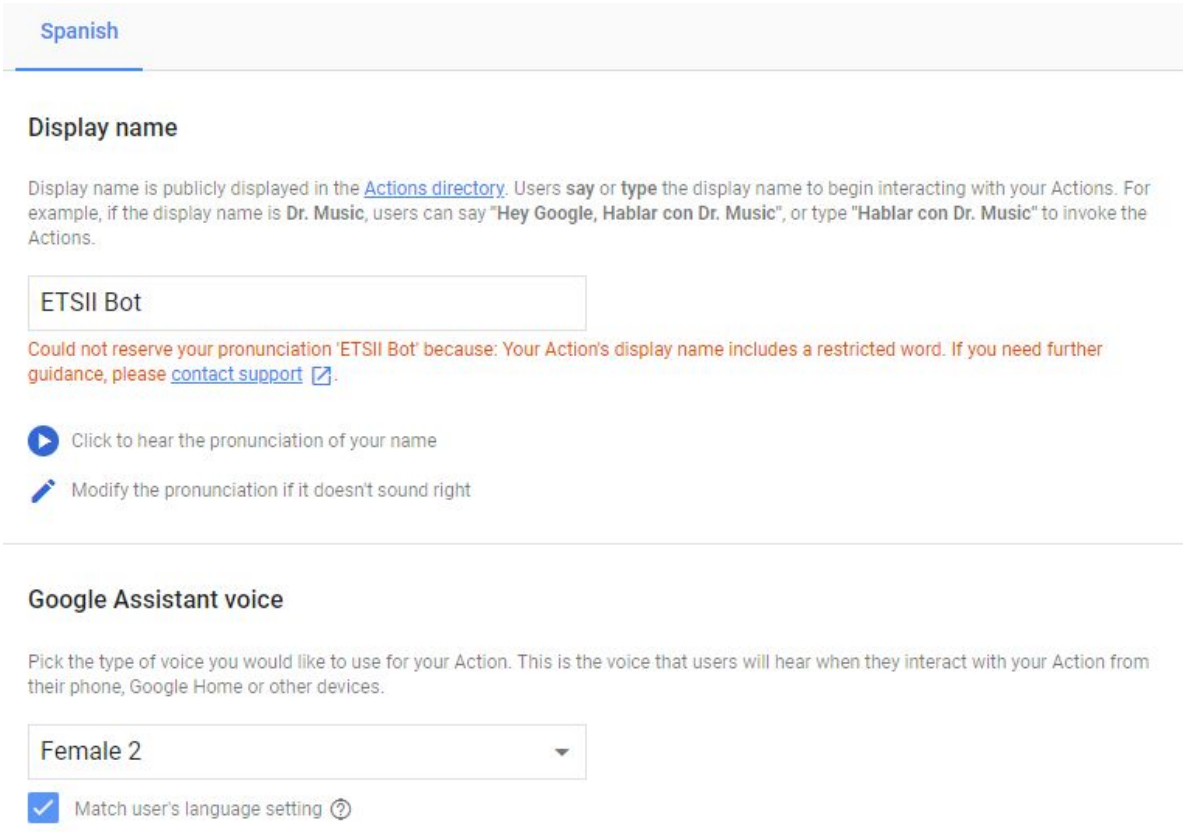


Figura 8: Nombre de Google Assistant.

4.2 Información adicional

Se nos solicita una descripción de los tareas a realizar por nuestro asistente, en nuestro caso es un chatbot de ayuda para estudiantes de Ingeniería Informática, hemos puesto lo siguiente:

Figura 9: Descripción del asistente.

Una vez rellenos los campos de descripción se nos solicita que asignemos bajo que nombres podrá ser invocado nuestro asistente, en nuestro caso hemos puesto las diferentes formas de referirnos al grado o a la escuela por comodidad para el usuario:

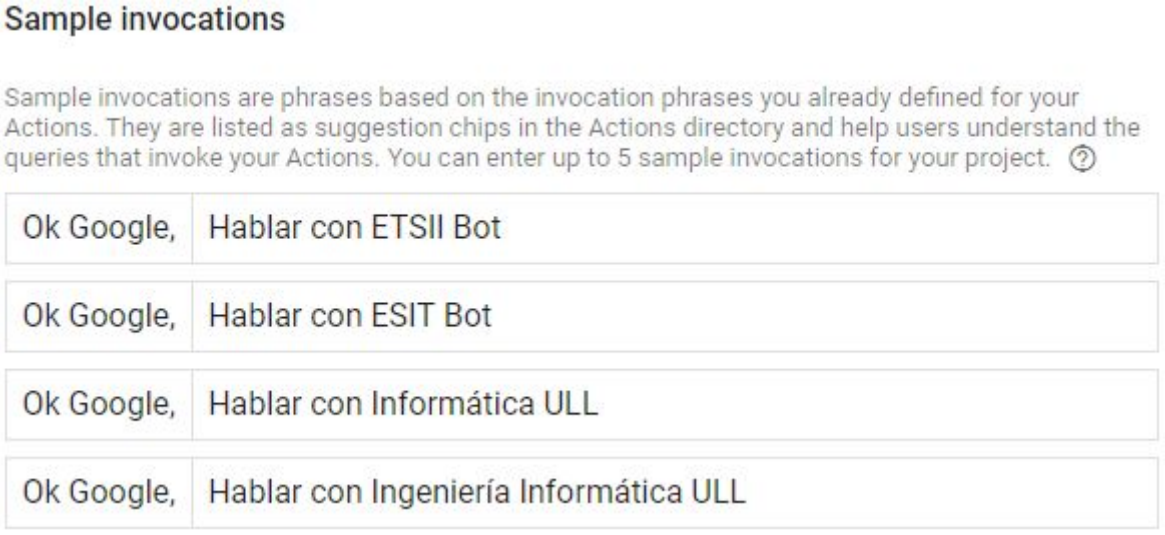


Figura 10: Nombres de invocación

Tras definir los nombres de invocación podemos poner un logo y un banner, en nuestro caso hemos utilizado el logo de la Universidad de La Laguna. Por último se especifican los datos del desarrollador y la categoría a la que pertenece el asistente, en este caso hemos asignado la categoría de educación. Finalmente nos aparecerá una vista de cómo luciría nuestro asistente en un teléfono móvil.

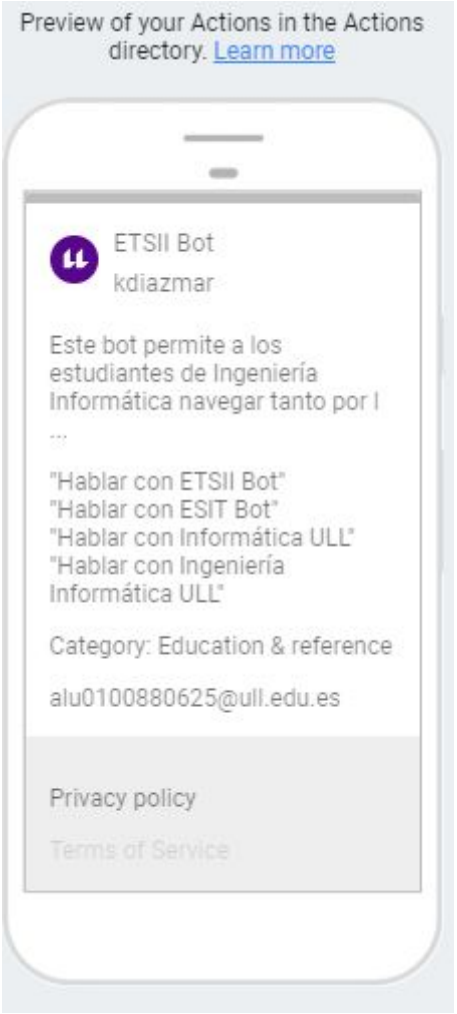


Figura 11: Visualización en móvil

4.3 Uso del asistente en dispositivos móviles

Existe un simulador para poder utilizar nuestro asistente en el ordenador previsualizando como se vería en Google Assistant para hacer comprobaciones sin necesidad de utilizar el móvil, es bastante práctico cuando aún estamos realizando comprobaciones.

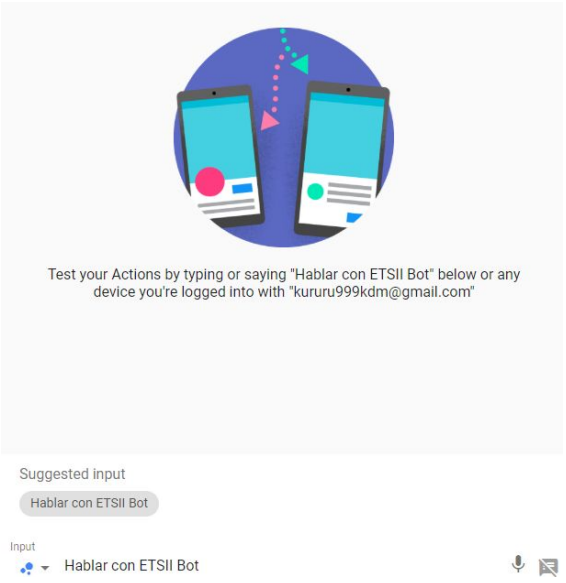


Figura 12: Asistente en móvil simulado.

Una vez invoquemos a nuestro chatbot podemos ver una respuesta en esa misma zona de la pantalla pero a su vez podemos verlo a la derecha como quedaría en el móvil (figura 13), es recomendable fijarse únicamente en cómo se visualiza en la parte de la derecha ya que es la que hace uso de la API de Google Assistant y puede haber pequeños errores en la parte principal.

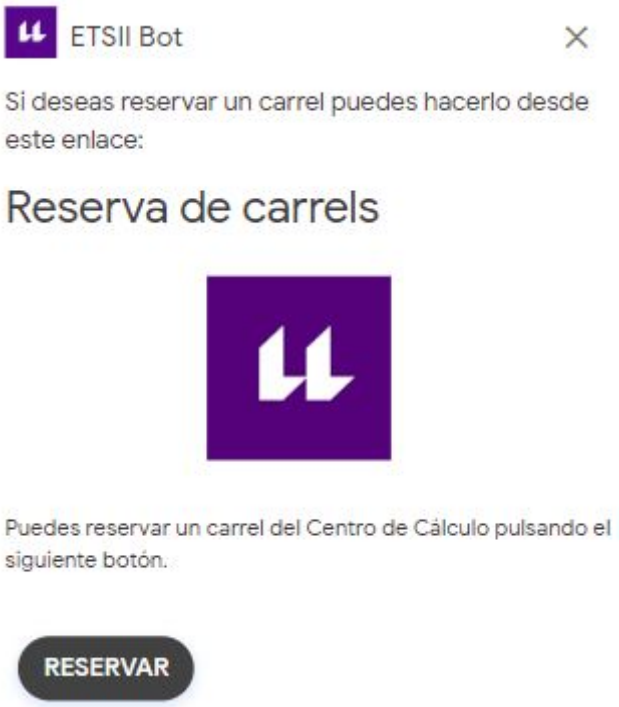


Figura 13: Visualización de respuesta en el simulador.

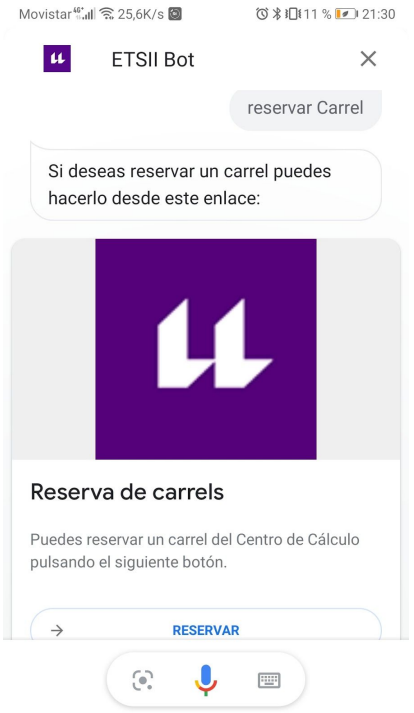


Figura 14: Visualización de respuesta en el móvil.

Capítulo 5 Problemas encontrados

5.1 Autenticación

Una de las principales limitaciones de nuestro asistente es no poder autenticarse para poder realizar consultas más personalizadas, debido a esto no se puede dar información específica del portal o campus virtual, ni tampoco reservar una sala de estudio o solicitar cita en secretaría, por lo que su funcionalidad de ve reducida.

5.2 Versión 2 de la API

Al comenzar este proyecto pudimos ver que muchas de las funciones que encontrábamos en ejemplos no funcionaban correctamente o directamente no se reconocían, a pesar de tener comentarios bastante positivos. Posteriormente nos dimos cuenta que esos proyectos que observábamos en tutoriales y documentación pertenecían a la versión anterior de DialogFlow, la cual próximamente quedará obsoleta y dejará de tener soporte, por lo tanto no era una versión viable. Algunos de los tutoriales disponibles de la página oficial redirigen a páginas inexistentes en esa versión, con lo que eran de poca ayuda y tuvimos que ir probando hasta hallar los métodos que estaban disponibles en la versión 2.

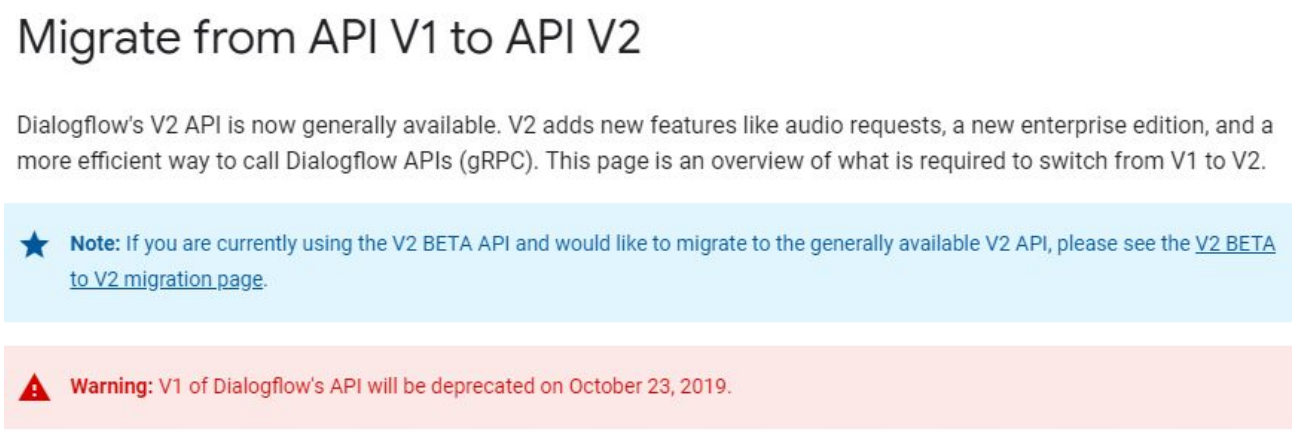


Figura 15: Migración a versión 2. [11]

5.2.1 Asignación de parámetros

Tras configurar correctamente la primera entidad de este proyecto (curso), configuramos la intención de horario con esta entidad como parámetro y nos dio los resultados esperados.

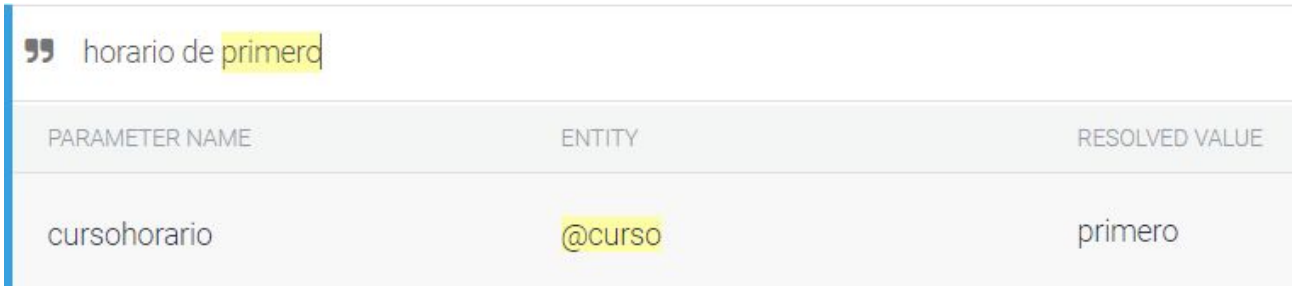


Figura 16: Entrenamiento con parámetro.

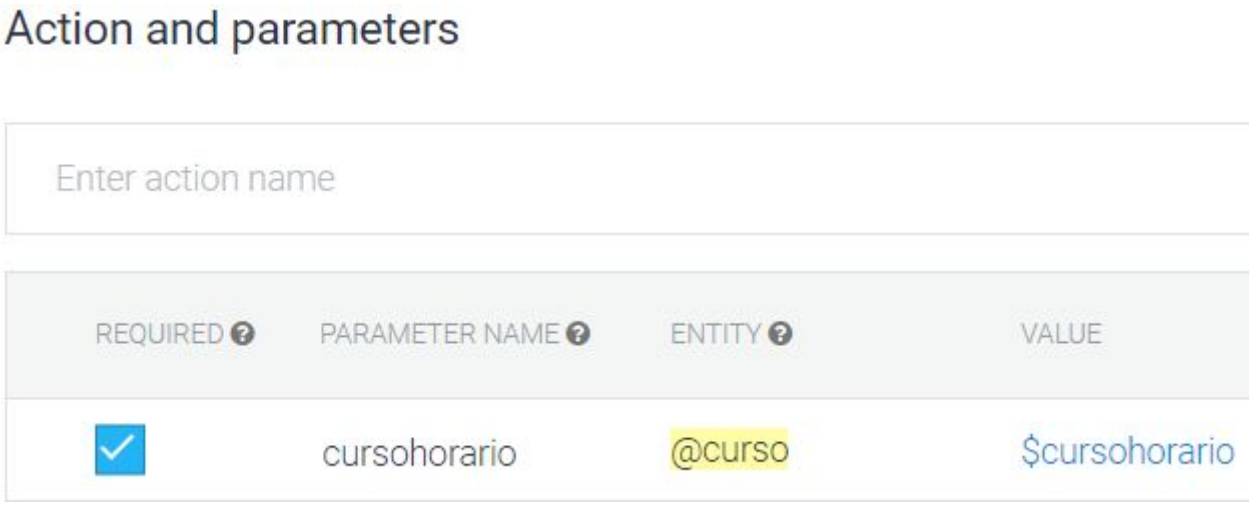


Figura 17: Asignación nombre de parámetro.

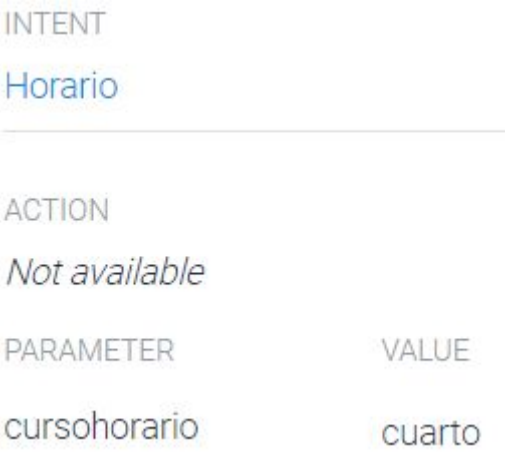


Figura 18: Asignación correcta del valor al parámetro.

Pero al realizar nuestro código javascript y enlazarlo para poder filtrar en función del curso indicado, el código desarrollado no logró ejecutarse nunca, mostrando siempre la respuesta que le asignamos por defecto dentro de la propia intención definida. El código de ejemplo era el siguiente:

```
const functions = require('firebase-functions');
const {dialogflow} = require('actions-on-google');

const HORARIO_INTENT = 'Horario';
const HORARIO_TYPE_ENTITY = 'curso';

const app = dialogflow();

app.intent(HORARIO_INTENT, (conv) => {
  const horario_type = conv.parameters[HORARIO_TYPE_ENTITY].toLowerCase();
  if(horario_type == "primero")
    conv.ask("1");
  else if(horario_type == "segundo")
    conv.ask("2");
  else if(horario_type == "tercero")
    conv.ask("3");
  else if(horario_type == "cuarto")
    conv.ask("4");
  else
    conv.ask("curso inexistente");
});

exports.dialogflowFirebaseFulfillment = functions.https.onRequest(app);
```

Figura 19: Código de ejemplo para la versión 1.

Esta función nos asignaría como respuesta el número correspondiente al curso que le indiquemos, pero como ya mencioné en la versión 2 no se ejecutaba, simplemente se omitía. Tras investigar la documentación encontré pequeños fragmentos que poco a poco le añadían funcionalidad e iban alterando el comportamiento del asistente, consiguiendo así desarrollar un código similar al siguiente:

```
'use strict';

const functions = require('firebase-functions');
const {WebhookClient, Card, Suggestion, SimpleResponse} = require('dialogflow-fulfillment');
process.env.DEBUG = 'dialogflow:debug'; // enables lib debugging statements

exports.dialogflowFirebaseFulfillment = functions.https.onRequest((request, response) => {
  const agent = new WebhookClient({ request, response });
  console.log('Dialogflow Request headers: ' + JSON.stringify(request.headers));
  console.log('Dialogflow Request body: ' + JSON.stringify(request.body));

  function horario(agent) {
    agent.add("Ejecutando función de horario.");
  }

  let intentMap = new Map();
  intentMap.set('Horario', horario);
  agent.handleRequest(intentMap);
});
```

Figura 20: Código versión 2.

Una vez conseguido este gran avance, implementamos funciones que en dependiendo de la fecha en la que estemos pueda decirnos si estamos en el primer o en el segundo cuatrimestre del curso (utilizada para filtrar los horarios de tutorías y horario de clase):

```
function getCuatrimestreActual()
{
  var cuatrimestre;
  switch((new Date()).getMonth())
    {
      case 1: //febrero
      case 2: //marzo
      case 3: //abril
      case 4: //mayo
      case 5: //junio
      case 6: //julio
      case 7: //agosto
      case 8: //septiembre (?)
        cuatrimestre = 1;
        //segundo cuatrimestre
        break;
      case 0:
      case 9:
      case 10:
      case 11:
      case 12:
        cuatrimestre = 0;
        break;
    };
  return cuatrimestre;
};
```

Figura 21: Función de cuatrimestre.

Y una función que nos retorne el día de la semana en el que estamos (para saber si hay o no tutorías ese día):

```
function getDiaHoy()
{
    return (new Date()).getDay() - 1;
};
```

Figura 22: Función del día de la semana.

El último problema con el que nos encontramos en este apartado fue conseguir que nuestro asistente nos diera la opción de hacer click en un link para dirigirnos a la página del servicio, que resolvimos con un código similar al siguiente:

```
const LOGO = 'https://www.ull.es/portal/marca/wp-content/themes/base-ull/as

function campusvirtual(agent)
{
  agent.add("Para acceder al campus virtual pulsa el siguiente botón:");
  agent.add(new Card({
    title: 'Campus virtual de la ULL',
    imageUrl: LOGO,
    text: 'Puedes acceder al campus virtual pulsando el siguiente botón.',
    buttonText: 'Abrir campus virtual',
    buttonUrl: 'https://campusvirtual.ull.es/login/index.php'
  }));
};
```

Figura 23: Tarjetas de servicios. [10]

5.3 Formato de las guías docentes

A la hora de filtrar el horario de las tutorías se nos ocurrió hacer scraping a las guías docentes y así poder automatizar la obtención de información, pero al analizar las primeras de la lista encontramos demasiados tipos de formatos diferentes, por lo que no podíamos establecer un patrón de filtrado, el siguiente ejemplo es uno de los que nos impedía continuar con esta idea:

Figura 24: Ejemplo de formato de tutorías.

Como podemos observar este formato es poco comprensible, por lo que sería una buena propuesta utilizar un formato único en las guías docentes, no solo por este proyecto, sino por la propia comprensión por parte del alumnado.

5.4 Disponibilidad de las salas

Decidimos implementar también la función de poder preguntarle al asistente qué salas del Centro de Cálculo se encontraban libres, para que los alumnos puedan organizar donde desarrollar sus prácticas cada día, pero al intentar conectar con la API de Google Calendar, se requiere una clave autorizada, ya que la API está pensada para consultar y modificar el calendario con el asistente. Debido a que no disponemos de estos permisos tampoco ha sido posible añadir esta funcionalidad.

Capítulo 6 Análisis de rendimiento y funcionamiento

Los servicios ofertados por nuestro asistente son de dos tipos, el primer tipo es una búsqueda de información simple, sin filtrar (este es el caso de servicios como el calendario académico o las incidencias), en este caso el chatbot nos mostrará información sobre el servicio solicitado y nos adjuntará un enlace para realizar el trámite o simplemente acceder a la información solicitada. El segundo tipo de servicio posee una etapa de consulta (que en nuestro caso al no poseer API, esta consulta se realiza sobre una estructura de datos creada manualmente). Para simplificar nos dirigiremos a estos dos tipos como consultas simples y avanzadas respectivamente.

6.1 Consultas simples

Las consultas simples nos permiten visualizar información de forma instantánea como si utilizaramos un navegador, pero la ventaja que aportan es el número de pasos a dar para obtener la misma información. Para reservar un carrel del Centro de Cálculo por ejemplo, los alumnos suelen buscar algo similar a “reserva carrels cc etsii ull”, pero al hacer esto encuentran una página en la que se ofrece ese servicio entre otros, una vez encontrado el apartado de “Reserva de Carrels”, al acceder se le redirige a otra página donde se da la opción de acceder a la aplicación de reservas y así finalmente llegar al destino de nuestra consulta, tras interactuar con tres páginas. Si realizamos la misma consulta con nuestro asistente, obtendremos el mismo resultado final, pero sin recorrer todas esas páginas. Un caso más básico sería un alumno consultando el horario, independientemente de que el curso y el cuatrimestre esté especificado en su búsqueda, siempre deberá interactuar con la página al menos dos veces, una para elegir el curso y la segunda para elegir el cuatrimestre, sin embargo nuestro asistente nos permite especificar el curso, por lo tanto la primera interacción deja de ser necesaria, y además al detectar la fecha en la que se realiza la consulta podemos obviar el cuatrimestre solicitado, con lo que volvemos a obtener la misma información con un tercio de las interacciones.

6.2 Consultas avanzadas

Estas consultas cuentan con algún parámetro que ayuda a filtrar de forma eficiente la información consultado, el ejemplo más claro es el servicio de tutorías, el proceso habitual de un alumno consultando tutorías para un día concreto conlleva al menos seis interacciones con diferentes páginas (sin contar con el posterior proceso de filtrado de horarios), sin embargo nuestro asistente nos permite en una sola consulta conocer las tutorías disponibles para la asignatura elegida en un día concreto. Estas consultas suelen ser más sensibles a las palabras utilizadas.

6.3 Comprensión del lenguaje natural

A la hora de comprobar el funcionamiento del asistente nos dimos cuenta que al principio le costaba más procesar los nombres de las asignaturas. Por ejemplo con las guías docentes, si únicamente decíamos “guía”, seguido del nombre de la asignatura, en

ocasiones no encontraba la asignatura, sin embargo en consultas más simples como el servicio VPN, independientemente de las palabras utilizadas siempre conseguía entendernos y responder adecuadamente. Para mayor comodidad del usuario, decidimos añadir los nombres de las asignaturas abreviados, las siglas y su escritura sin tildes, aunque esta última sólo es útil si se realizan consultas escritas. Cualquier consulta se puede simplificar a la hora de realizarla, pero en algunos casos genera conflictos, como por ejemplo la consulta del calendario, nuestro asistente podría responder tanto con el calendario académico como con el calendario de exámenes debido a que ambas respuestas tendrían cierto sentido, pero por comodidad hemos decidido darle mayor prioridad al calendario académico para evitar incertidumbre. Finalmente pusimos a prueba si era capaz de responder consultas en las que faltaran algunas palabras o se diera demasiada información, cuanta más larga sea la consulta, más fácil es que se pierda nuestro asistente, por ejemplo si le decimos “quiero reservar un carrel del Centro del Cálculo”, inicialmente el asistente no sabía el servicio solicitado, pero tras entrenarlo con diferentes frases y corregirlo, ha ido mejorando la comprensión hasta diferenciar frases como esa. Lo mismo pasaba con el servicio de guías docentes, si únicamente decíamos “guía” seguido de un nombre muy largo, no respondía, pero al decir “guía docente de”, nunca fallaba. Para mejorar su comprensión ha sido necesaria la ayuda de algunos alumnos para analizar cómo interactúan con un asistente de este tipo e ir corrigiendo los fallos.

6.4 Obtención de información

Debido a que la información sobre los servicios ofertados por el asistente no está en un formato único (los horarios y calendarios de exámenes están en Google Docs, los horarios de disponibilidad de las salas en Google Calendar, las guías docentes en el portal e-guía…) no se podía buscar una única forma de acceder a toda la información, además de que en algunos casos la información no se encontraba en un formato único (en el caso de las guías docentes por ejemplo), así que nos vimos en la obligación de utilizar la información que fuera menos propensa a cambios. Los nombres de las asignaturas por ejemplo, no debería cambiar, las guías docentes se encuentran cada una en su link y son actualizadas cada año, los servicios como VPN y reserva de carrels son enlaces que no cambian tampoco. Para aquella información que si cambia de año en año simplemente utilizamos estructuras de datos que en un futuro podrían estar conectadas a una API (es el caso del filtrado de tutorías).

```
var asignaturaEncontrada = false, nombreAsignaturaReal = "";
var tutoriasCuatrimestre = ["", ""];
var tutoriasDia = [["", "", "", "", ""], ["", "", "", "", ""]];
```

Figura 25: Estructuras para almacenar las tutorías.

La primera variable que encontramos en este código nos indicará si la asignatura que hemos indicado es o no reconocida, seguida de la variable que indicará el nombre de la asignatura (esta variable existe para poder considerar abreviaturas del nombre). El array situado en la segunda línea indicará todos los horarios de tutorías, situando los del primer cuatrimestre en la posición cero y los del segundo cuatrimestre en la posición uno. Por

último tenemos un array bidimensional que contendrá la misma información que el anterior pero separándose por días de la semana también. Al carecer de API a la que consultar, hemos extraído y separado la información manualmente para asignarla de la siguiente forma:

```
case "álgebra":
case "algebra":
  tutoriasCuatrimestre[0] = "IRENE MARQUEZ CORBELLA Lunes y Miércoles de 16:00 a 19:00 (Despacho 66 - Tercera Planta, Edificio de Matemáticas y Física), " +
                            "IGNACIO GARCIA MARCO Lunes de 16:00 a 18:00 y Miércoles y Jueves de 9:30 a 11:30 (Despacho 70 - Tercera Planta, Edificio de Matemáticas y Física)";
  tutoriasCuatrimestre[1] = "IRENE MARQUEZ CORBELLA Lunes y Miércoles de 16:00 a 19:00 (Despacho 66 - Tercera Planta, Edificio de Matemáticas y Física), " +
                            "IGNACIO GARCIA MARCO Lunes de 16:00 a 18:00 y Miércoles y Jueves de 9:30 a 11:30 (Despacho 70 - Tercera Planta, Edificio de Matemáticas y Física)";

  tutoriasDia[0][LUNES] = "IRENE MARQUEZ CORBELLA de 16:00 a 19:00 (Despacho 66 - Tercera Planta, Edificio de Matemáticas y Física)," +
                          "IGNACIO GARCIA MARCO de 16:00 a 18:00 (Despacho 70 - Tercera Planta, Edificio de Matemáticas y Física)";
  tutoriasDia[0][MARTES] = "";
  tutoriasDia[0][MIERCOLES] = "IRENE MARQUEZ CORBELLA de 16:00 a 19:00 (Despacho 66 - Tercera Planta, Edificio de Matemáticas y Física), " +
                              "IGNACIO GARCIA MARCO de 9:30 a 11:30 (Despacho 70 - Tercera Planta, Edificio de Matemáticas y Física)";
  tutoriasDia[0][JUEVES] = "";
  tutoriasDia[0][VIERNES] = "";

  tutoriasDia[1][LUNES] = "IRENE MARQUEZ CORBELLA de 16:00 a 19:00 (Despacho 66 - Tercera Planta, Edificio de Matemáticas y Física), " +
                          "IGNACIO GARCIA MARCO de 16:00 a 18:00 (Despacho 70 - Tercera Planta, Edificio de Matemáticas y Física)";
  tutoriasDia[1][MARTES] = "";
  tutoriasDia[1][MIERCOLES] = "IRENE MARQUEZ CORBELLA de 16:00 a 19:00 (Despacho 66 - Tercera Planta, Edificio de Matemáticas y Física), " +
                              "IGNACIO GARCIA MARCO de 9:30 a 11:30 (Despacho 70 - Tercera Planta, Edificio de Matemáticas y Física)";
  tutoriasDia[1][JUEVES] = "";
  tutoriasDia[1][VIERNES] = "";

  nombreAsignaturaReal = "Álgebra";
  break;
```

Figura 26: Asignación de información de tutorías.

Este bloque de código se encuentra dentro de un switch que tiene un case diferente por asignatura. Por facilitar la lectura del código se han declarado constantes que representan los diferentes días de la semana. Una vez asignada la información correspondiente, concluimos con el siguiente código de respuesta:

```
if(asignaturaEncontrada)
{
  if(dia != NO_DEFINIDO)
  {
    if(tutoriasDia[getCuatrimestreActual()][dia] == "")
      agent.add("No hay tutorías de " + nombreAsignatura + " para el día indicado.");
    else
      agent.add(tutoriasDia[getCuatrimestreActual()][dia]);
  }
  else
    agent.add(tutoriasCuatrimestre[getCuatrimestreActual()]);
}
else
  agent.add("No se ha encontrado la asignatura.");
```

Figura 27: Mostrar tutorías.

En este fragmento de código comprobamos si se encuentra o no la asignatura, en caso afirmativo comprobamos si ha sido especificado el día y en función de este parámetro filtraremos las tutorías del día o mostraremos todas las del cuatrimestre.

Capítulo 7 Conclusiones y líneas futuras

Este proyecto es bastante prometedor, no solo para los futuros estudiantes de la facultad, se podría también generalizar a estudiantes de todas las facultades, que en ciertas ocasiones no encuentran la información que necesitan y se pierden en la página de la universidad, pero también al personal, ya que ellos también realizan consultas e interactúan con ciertos servicios como podrían ser las reservas de salas para realizar exámenes. Para ampliar el alcance de nuestro proyecto sería necesario:

7.1 Existencia de una API

Esta API permite consultar datos como por ejemplo, el grupo al que pertenece cada alumno (esta información es pública para todos los estudiantes del mismo grado, ya que se envía un correo y solo se comprueba que el lector tenga un correo del dominio de la universidad), lo que podría añadir la función de que nuestro asistente pueda decir que horario tiene el usuario cada día, además de poder consultar las tareas pendientes en el campus virtual o incluso realizar trámites simples como pedir cita en secretaría o reservar una sala de estudio. Los servicios se verían mejorados como en el siguiente diagrama:

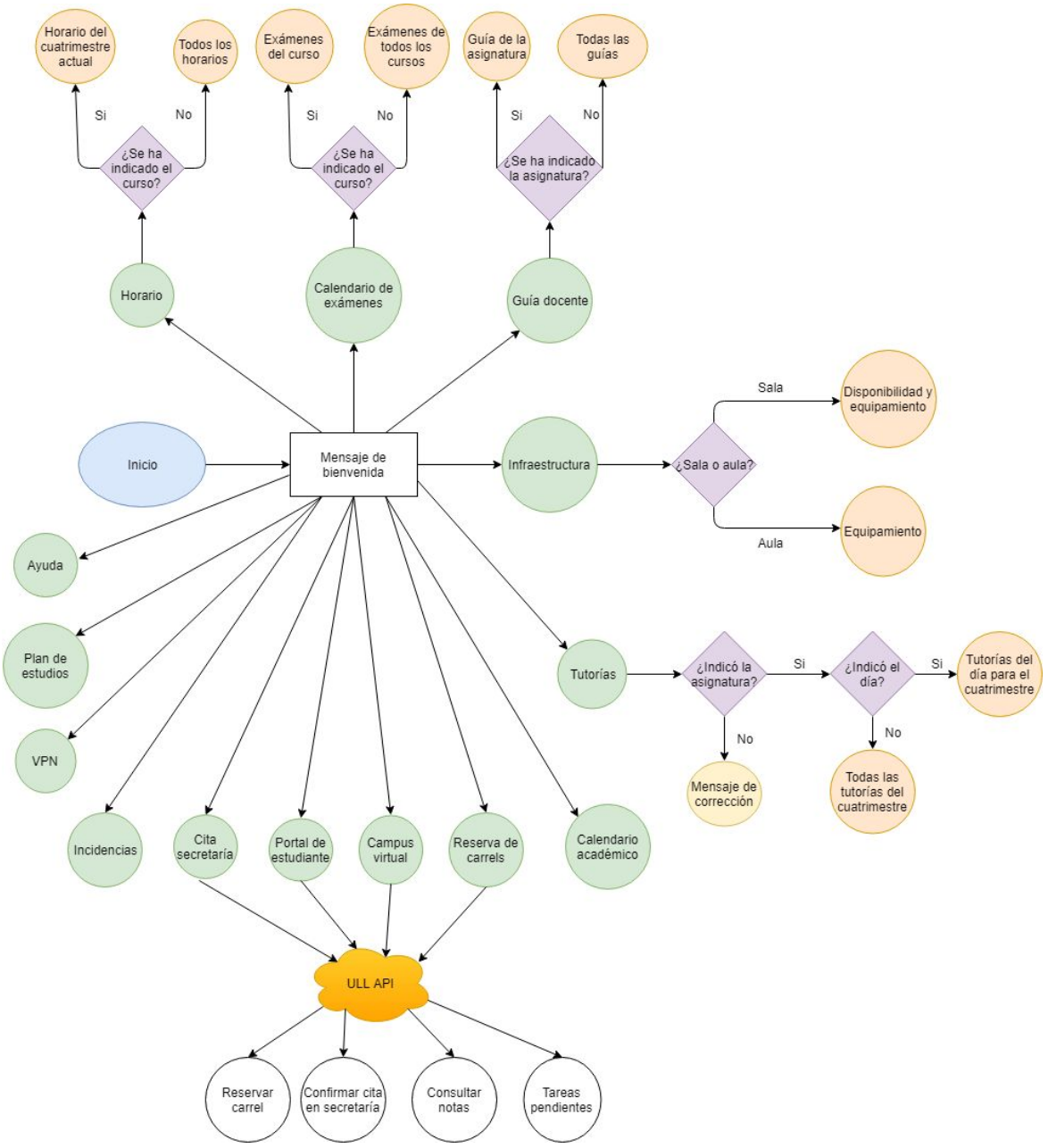


Figura 28: Diagrama de flujo mejorado.

7.2 Formato universal de guías docentes

Si conseguimos que las guías docentes tengan un único formato, podríamos lograr que el asistente pueda recopilar información de todas las guías sin necesidad de separarla manualmente, y al ser un enlace que nunca cambia, todos los años esta información se actualizará automáticamente.

Capítulo 8 Summary and Conclusions

This project is quite hopeful, not only for future students of the school, it could be generalized for all the students of all schools, that sometimes does not find the information you need and is lost on the page of the university, but also to the staff, since they also consult and interact with certain services, such as reservations for exam rooms. To expand the scope of our project it would be necessary:

8.1 Existence of an api

This API allows you to consult data such as the group to which each student belongs (this information is public for all the students of the same grade, since an email is sent and it is only verified that the reader has an email from the domain of the university), which could add the function that our assistant can tell what class schedule the user has every day, besides being able to consult the pending tasks in the virtual campus or even perform simples procedures like make an appointment or book a study room. The services would be improved as in figure 28.

8.2 Universal format of teaching guides

If we get the teaching guides to have a single format, we could get the assistant to gather information from all the guides without the need to separate it manually, and being a link that never changes, every year this information will be updated automatically.

Capítulo 9 Presupuesto

Elemento	Duración (h)	Coste (€)
Análisis de servicios	40h	800
Diseño	40h	800
Implementación	200h	4000
Total	**270h**	**5600**

Tabla 2: Presupuesto del proyecto.

Bibliografía

[1]https://futurizable.com/chatbot/
[2]https://www.bbvaopenmind.com/tecnologia/mundo-digital/que-es-el-aprendijaze-profundo/
[3]https://vicampuzano.com/dialogflow/
[4]https://planetachatbot.com/primeros-pasos-en-dialogflow-7e0510ac3591
[5]https://developers.google.com/actions/assistant/responses
[6]https://blog.dialogflow.com/post/use-slot-filling-in-fulfillment/
[7]https://planetachatbot.com/entendiendo-dialogflow-de-manera-definitiva-parte-2-entidades-927cffd8f4f4
[8]https://cloud.google.com/dialogflow-enterprise/docs/tutorials/build-an-agent/create-fulfillment-using-webhook
[9]https://dialogflow.com/docs/getting-started/integrate-services
[10]https://blog.dialogflow.com/post/fulfillment-library-beta/
[11]https://dialogflow.com/docs/reference/v1-v2-migration-guide