
ESCUELA SUPERIOR DE INGENIERÍA
Y TECNOLOGÍA

# Trabajo de Fin de Grado

Grado en Ingeniería Informática

# Bot Conversacional Detector de Sentimientos

*Chatbot - Sentiment Analytics*

SOFÍA PIZARRO ARBELO

La Laguna, 15 de marzo de 2019

D. **Jesús Miguel Torres Jorge**, con N.I.F.: 43.826.207-Y profesor Contratado Doctor adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

**C E R T I F I C A (N)**

Que la presente memoria titulada:

*“Bot conversacional detector de sentimientos”*

ha sido realizada bajo su dirección por D. **Sofía Pizarro Arbelo**,

con N.I.F. 78.633.072-M.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 15 de marzo de 2019

# Agradecimientos

Profesores, compañeros y demás personal de la ULL. A mi familia, que ha confiado siempre en mí, al pequeño Limón por darme ese rayo de luz. A todos aquellos que me han facilitado el camino durante todos estos años. A ti, por tu ayuda y paciencia infinita. A mi tutor, por creer en este proyecto. A todos, solo me queda decirles ¡Gracias! Sin ustedes nada hubiera sido igual.

**Resumen**

*El objetivo de este trabajo es desarrollar un sistema conversacional interactivo que sea capaz de analizar los sentimientos del usuario mediante diferentes escenarios de conversación. Este proyecto plantea una propuesta de ayuda a los alumnos de la ETSII mediante la orientación y, a su vez, proponer jugar o escuchar música cuando detecte sentimientos negativos o de tristeza. Del mismo modo, al detectar sentimientos de alegría, miedo, frustración, etc. responderá en base a eso e intentará guiarle por distintas opciones de resolución de conflictos o ideas.*

*Dado que es un tema delicado que podría requerir a expertos en temas sociales se tratará, en este caso, desde un punto más superficial y desenfadado, al menos inicialmente.*

*Además, se trata de un sistema de autoaprendizaje que va absorbiendo información del usuario y aprendiendo de ello, de esta manera su cerebro irá «in crescendo».*

*En este proyecto se pretende elaborar una herramienta, como es un asistente virtual, de cara a orientar a los alumnos que se encuentran perdidos en sus carreras, ayudándoles en la rama por la que pueden ir en informática teniendo en cuenta en todo momento sus gustos o sus asignaturas de preferencia y que, a su vez, sea capaz de entender cómo se sienten. Algo tan esencial como trabajar con las emociones y los sentimientos para que este asistente sea lo más similar posible al comportamiento humano.*

*Este proyecto ofrece una alternativa a los alumnos de manera que no tengan que asistir a un orientador humano si no se sienten cómodos, pudiendo resolver sus dudas con este asistente virtual.*

**Palabras clave:** orientador académico, estudiantes ETSII, análisis de sentimientos, detector de emociones, asistente virtual, bot conversacional, entendimientos del lenguaje natural, analizador de tono, redes neuronales, machine learning, Watson.

**Abstract**

*The purpose of this project is to develop an interactive conversational system that will be able to analyze the user's feelings through different conversation scenarios. This project offers a proposal to help the ETSII students through guidance and, also, propose to play or listen to music when it detects negative feelings or sadness. The same by detecting feelings of joy, fear, frustration, etc. will respond based on that and will try to guide them through different options in order to resolve conflicts or ideas.*

*Since it is a sensitive topic, it could require experts in psychology or pedagogy or in any social topic. Then, in this case, this will be studied from a more superficial and casual point of view, at least at the beginning.*

*In addition, it is a self-learning system that absorbs information from the user and learns from it, in this way the brain will go "in crescendo".*

*This project aims to develop a tool, such as a virtual assistant, in order to guide students who are confused in their degrees, helping them in the Career specialization that they have to choose into computing engineering studies taking into account during the chat their favorites subjects, hobbies or preferences. In addition, it will be able to understand how they feel. Something as essential as working with emotions and feelings so that this assistant is as similar as possible to human behavior.*

*This project offers an alternative to the students so they do not have to attend a human counselor if they do not feel comfortable, being able to solve their doubts with this virtual assistant.*

**Keywords:** academic counselor, ETSII students, sentiment analysis, emotions analysis, virtual assistant, chatbot, natural language understanding, tone analyzer, neural network, machine learning, Watson.

# Índice de Contenidos

# Índice de figuras

# Índice de tablas

# Capítulo 1
# Introducción

Aristóteles decía que somos seres sociales, es decir, necesitamos vivir en comunidad y rodeados de otros seres humanos. Es pues, desde la Antigua Grecia, cuando el ser humano comprende y valora la importancia de la comunicación. Llegados al siglo XXI... ¿por qué no reformularlo en forma de *chatbot*?

Siguiendo la definición oficial, un *chatbot* es un software de Inteligencia Artificial (I.A.) diseñado para realizar una serie de tareas de manera independiente y sin la ayuda de un humano. Por ejemplo, los *bots* podrían hacer una reserva en un hotel o marcar una fecha en el calendario de nuestro smartphone. Básicamente, sirven para automatizar procesos que se ejecutan sin la necesidad de una intervención humana, siendo capaces de procesar el lenguaje natural.

Realmente los *bots* llevan existiendo más de 50 años. Alan Turing, matemático británico, comenzó en 1950 a desarrollar una hipótesis con el fin de comprobar si los ordenadores podían mantener una conversación con humanos. Esto es lo que hoy en día conocemos como Test de Turing, donde no solo comprobó esto, sino que además hizo creer al usuario que estaba hablando con un humano y no con un ordenador.

Valiéndonos de estas valiosas demostraciones y quedando justificada la importancia de la comunicación más allá de lo humano, procedo a explicar la estructura de mi trabajo sobre el Bot Conversacional. Consta de los siguientes apartados:

En primer lugar, encontraremos un capítulo dedicado a los objetivos del proyecto: "qué pretendo conseguir con este trabajo".

Un segundo capítulo dedicado a los *Chatbots* y sus antecedentes: situación actual, problemáticas y trabajos relacionados.

Un tercer bloque donde encontramos la metodología y plan de trabajo: ¿cómo voy a conseguir los objetivos planteados?

Un cuarto gran bloque en el que expongo de forma detallada el desarrollo en la elaboración del *chatbot,* investigación previa, lenguajes y requisitos, servicios utilizados y diseño.

El quinto bloque es el último peldaño de este trabajo. En él explicaré qué resultados he obtenido y qué retos quedan por delante para futuras investigaciones. Todo ello, quedará completado por las conclusiones, la bibliografía empleada y el presupuesto general.

Ya decía Saramago en su famoso “Ensayo sobre la ceguera” que la comunicación va más allá de los ojos y trasciende a los sentidos, siendo una experiencia ultra sensorial. ¿Serán los c*hatbots* un ejemplo de ello? Comencemos a investigarlo.

## 1.1 Objetivos

El objetivo de este trabajo consiste en el desarrollo de una herramienta capaz de interactuar con los alumnos de manera que les pueda orientar académicamente y, a su vez, sea capaz de entender y analizar sus emociones y sentimientos, en tiempo real y de forma natural. Para ello, intenta averiguar mediante algunas preguntas si está o no contento en la carrera y si tiene claro qué rama o especialidad elegir, proponiéndoles, según preferencias, las que considera más acertadas según el nivel de confianza que obtiene en cada iteración de resultados. Siguiendo esta línea, es capaz de analizar de manera continua cómo se encuentra el usuario a nivel emocional y según como se encuentre redirige la conversación por un flujo u otro.

Así, partiendo de herramientas ya existentes como es la de IBM Watson y, por lo tanto, explotando todo su potencial como servicio en la nube, se busca desarrollar un proyecto que aporte mejoras a los servicios existentes en el panorama actual como son los expuestos a continuación:

- Integrar diversos servicios que hacen que un *bot* conversacional sea aún más inteligente, como son el asistente de conversación, el analizador de tonos y el servicio de comprensión de lenguaje natural (NLU).
- Reconocimiento de emociones y sentimientos cuando la entrada del usuario es en español. Este servicio está limitado en *IBM Watson* a inglés y francés. Por lo tanto, se amplía con un traductor de manera que esta detección sea incluso posible en nuestro idioma.
- Hacerlo accesible a todos los alumnos tanto por servicio web como por aplicación móvil, usando para ello un servicio tan utilizado como es *Telegram*.
- Suministrar a la aplicación una cadena de herramientas unificada (*Eclipse*, *github*, entrega y despliegue de forma repetible con la mínima intervención humana), capaz de automatizar las compilaciones, las pruebas y que controle la calidad de los resultados.

Otro objetivo, en este momento secundario, pero no menos importante, es la integración del *machine learning* (Ennis, s.f.) que nos permita crear y entrenar modelos de I.A. y preparar y supervisar datos en un entorno integrado como es el *Watson Studio*. De este modo, la aplicación será capaz de aprender durante toda la conversación, de manera supervisada, almacenando como modelo el tipo de alumnado y preferencias para así dar un resultado más fino y preciso a medida que va aprendiendo de los anteriores.

# Capítulo 2
# Estado del arte

Desde siempre, y más en los últimos tiempos, vivimos en un mundo que exige, cada vez más, nuevas tecnologías y una sociedad más preparada y capacitada para su desarrollo. Es por ello por lo que no paran de surgir nuevos sistemas inteligentes que participan de manera activa con el humano, bien como potentes sistemas de ayuda o como mero entretenimiento. En este caso vamos a centrarnos en los asistentes virtuales que, como bien dice su nombre, nos ayudan de forma virtual a resolver dudas o a aprender sobre diversos temas para los que haya sido programado.

Un bot conversacional o *chatbot* puede definirse como un asistente virtual que emula el comportamiento de una persona, animal u otra criatura y que permite mantener conversaciones con humanos, ya sea o bien escritas o bien por voz.

Desde que estos sistemas han sido dotados de Inteligencia Artificial, que se aproxima de manera fiel el comportamiento humano, su capacidad y fama han aumentado. Como era de esperar la variedad dentro de este ámbito es grandísima, quizás los más representativos sean: consulta de productos, comerciales, asistentes de ayuda o entretenimiento. Es en estos dos últimos campos, ayuda y entretenimiento, en los que basaremos este proyecto, usando los servicios de IBM Watson para modelar el conocimiento y la lógica de la conversación.

## 2.1 Problema que se plantea

Aunque ya existen trabajos previos con los *chatbots*, apenas podemos encontrarlos en español. A su vez, los trabajos al respecto apenas se centran en *chatbots* capaces de tomar decisiones según el estado de ánimo de la persona.

En definitiva, la principal motivación de este proyecto es poder explotar este hueco encontrado en este tipo de bots conversacionales, es decir, interacción humano-computador lo más cercano posible a la realidad.

## 2.2 Trabajos Relacionados

El concepto de asistente virtual es relativamente nuevo, pero desde hace mucho tiempo se ha venido trabajando en sistemas informáticos capaces de mantener una conversación. A continuación, se presentan algunos ejemplos relacionados con este proyecto, no tanto a nivel de orientación académica, pero sí otros tipos de asistentes y, sobre todo, algunos ejemplos en inglés donde se realiza el análisis de sentimientos y emociones.

### 2.2.1 Food Coach (@IBM, Play with the Food Coach chatbot, s.f.)

Esta aplicación es un ejemplo de *Tone Analyzer* integrado con *Watson Assistant*. En esta aplicación, dependiendo del tipo de comida que el usuario comió y cómo se siente al respecto, el agente automatizado proporciona una respuesta de entrenamiento adecuada para alentar al usuario a tomar decisiones saludables.

*Imagen 2-1. Trabajos relacionados - Food Coach*

### 2.2.2 Cognitive Car Dashboard Application (@IBM, Tutorial: Building a cognitive car dashboard dialog, s.f.)

Esta aplicación muestra cómo integrar el servicio de comprensión de lenguaje natural (*NLU*) con el servicio de conversación de *Watson* para extraer entidades genéricas y pasarlas al servicio de conversación. Esta aplicación es una extensión de la aplicación de *Watson assistant* donde agrega integración a otro servicio de *Watson*, como es, *NLU*, así como a una API de terceros, en este caso la API de *Weather Underground.*

*Imagen 2-2. Trabajos relacionados - Cognitive Car Dashboard*

### 2.2.3 Watson banking chatbot (@IBM, Create a cognitive banking chatbot, s.f.)

En esta aplicación veremos un *chatbot* utilizando *Node.js* y *Watson Assistant*. El flujo del Asistente se mejorará utilizando el entendimiento del lenguaje natural (*NLU*) para identificar entidades y el analizador de tonos para detectar las emociones de los clientes. Para preguntas frecuentes, llamará al servicio *Discovery* para obtener respuestas de una colección de documentos.


*Imagen 2-3.Trabajos relacionados - Watson Banking Chatbot*

### 2.2.4 A.L.I.C.E (Wallace, s.f.) y Eliza (Weizenbaum, s.f.)

*Eliza* fue uno de los primeros sistemas informáticos que contaba con la capacidad de procesar el lenguaje natural humano para llevar a cabo una conversación mediante texto de forma coherente con una persona. Funcionaba buscando palabras claves dentro de una frase que recibía del usuario. Si alguna de las palabras que leía pertenecía a su base de datos, contestaba alguna de las respuestas previamente programadas para esa clave. Fue diseñado y desarrollado en el Instituto Técnico de Massachusetts (MIT) por Joseph Weizenbaum entre 1964 y 1966. Debido a la baja complejidad que presenta, este programa ha sido implementado en una gran variedad de lenguajes de programación.

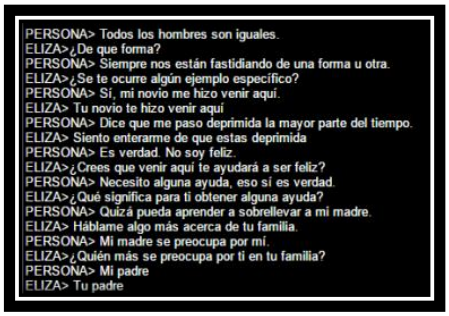



*Imagen 2-4. Trabajos relacionados - Eliza*

*A.L.I.C.E* (Artificial Linguistic Internet Computer Entity) está inspirada en *Eliza*, año 1995, y se trata de un *bot* muy básico, en este caso el lenguaje que utiliza está basado en *AIML* (*Artificial intelligence Markup Language*) para especificar las reglas heurísticas de la conversación. Fue diseñada en *Java* por el Dr. Richard S. Wallace. Nace como parte del llamado Proyecto Pandora, un proyecto de internet que consiste en la creación de *chatbots* especializados en diferentes temáticas. Simula una conversación inteligente y desarrollada capaz de engañar a un usuario haciéndole pensar que está hablando con otro ser humano.


*Imagen 2-5. Trabajos relacionados - A.L.I.C.E.*

### 2.2.5 Siri, Alexa, Google Assistant y Cortana (Nepal, s.f.)

Llegamos con los ejemplos más actuales y usados hoy en día: *Siri.* Este es, quizás, el asistente por voz más conocido del mundo. Disponible solo para los usuarios de *iPhone*, se ha convertido en uno de los iconos del teléfono de *Apple*, que este año cumple su undécimo aniversario. *Siri* hace cualquier cosa, desde cálculos matemáticos hasta informar del tiempo, hacer llamadas o conocer el saldo de tus cuentas bancarias; también es capaz de reconocer qué canción suena o mostrarte las fotos de un determinado lugar en el que has estado si se lo pides. En resumen, no hay casi nada que no pueda hacer.

*Imagen 2-6. Trabajos relacionados – Siri*

Grandes competidores son *Google Assistant*, integrado para dispositivos *Android* y sus altavoces, *Alexa* de *Amazon* y *Cortana* de *Windows*, todos ellos, quizás *Cortana* está más limitado, capaces de hacer tantas cosas como *Siri*, además *Google assistant* y *Alexa* han abierto su SDK para que otros fabricantes y desarrolladores puedan empezar a trabajar sobre ellos. Las opciones son infinitas.


*Imagen 2-7. Trabajos Relacionadoos -Alexa - Google Assistant – Cortana*

## 2.3 Actualidad de los asistentes virtuales

Actualmente, el crecimiento de la mensajería en tiempo real ha producido un cambio fundamental en el modo en que los usuarios prefieren conectar con las empresas. Donde hay una aplicación de mensajería instantánea, habrá un *chatbot*. Esto beneficia mayoritariamente a las empresas ya que permite tener conversaciones instantáneas, en cualquier momento, en cualquier lugar y a cualquier hora, ni horarios, ni vacaciones, simplemente están siempre disponibles para atendernos en cuánto les requieras.

Para ser sinceros, no hay nada más tedioso que llamar por teléfono y darte cuenta de que al otro lado hay un sistema de voz atendiéndote con unas opciones y respuestas limitadas, incluso incapaz de entender lo que queremos decir y entrando en un ciclo sin fin. ¡Es desesperante! Gracias a los nuevos avances tecnológicos, los sistemas son capaces de reconocer nuestra voz e incluso el tono de la conversación, bromas, frustración o cualquier otro estado de ánimo. Ya no nos comunicamos igual, no disponemos de tanto tiempo y mientras más rápido y cómodo sea mejor. Es un hecho, el presente y el futuro de la asistencia tiene nombre propio y se llama Bot.

Pese a que todavía a nivel de desarrollo algunos *chatbots* son algo rudimentarios, la I.A. se ha posicionado como la gran aliada de atención al cliente, una reinvención constante al igual que esta sociedad. Hay cinco sectores especialmente sensibles a esta evolución digital: banca, e-comercio, turismo, salud y gestión empresarial.

# Capítulo 3
# Metodología

La metodología escogida para el desarrollo del proyecto es la incremental (mariCh, s.f.), una de las metodologías clásicas. De este modo, el desarrollo se sucede de manera que incrementemos las funcionalidades del programa, aplicando secuencias lineales de forma escalonada a la par que va progresando el tiempo en el calendario. Cada secuencia lineal produce un incremento en el programa, mostrando en cada incremento una evolución con respecto a la fecha anterior, nunca debe ser igual. Por lo tanto, generando versiones incompletas hasta llegar al producto final, basando cada incremento en el anterior, pero con su propio ciclo de vida. A continuación, veremos estos pasos de manera detallada.

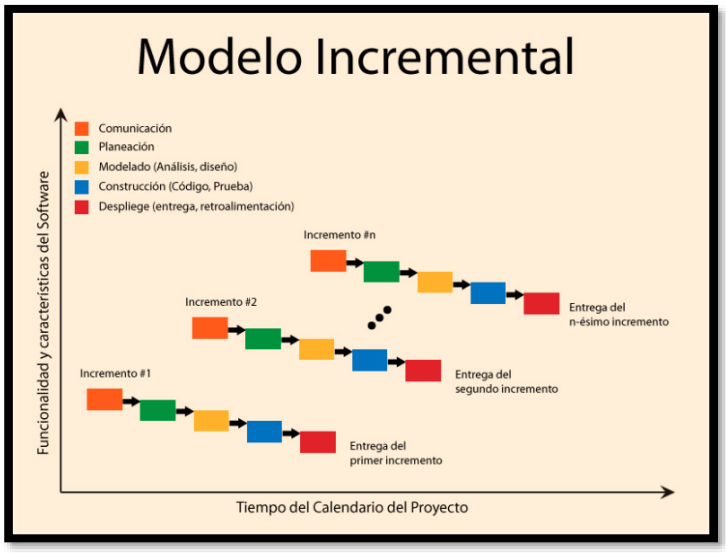



*Imagen 3-1 - Metodología Modelo Incremental*

Se ha optado por esta metodología por la cantidad de servicios que contiene la aplicación, así como la investigación y cambios en el desarrollo de esta. De esta manera, podemos obtener un estudio en profundidad de los servicios ofrecidos y sus limitaciones, así como la cantidad de posibles diseños y su programación.

## 3.1 Plan de trabajo

El plan de trabajo se divide en tres grandes bloques. El primer bloque se basa en la investigación de las tecnologías a utilizar y el diseño del flujo de la conversación mediante un diagrama de flujo. El segundo bloque constituirá el desarrollo de la aplicación mediante los servicios ofrecidos por IBM Watson. Por último, el tercer bloque de trabajo está compuesto por los testeos de la aplicación y el análisis de resultados.

### 3.1.1 Primer Bloque

| TAREA | Tiempo/Horas |
|---|---|
| **ANÁLISIS DEL PROBLEMA** | 20 |
| **INVESTIGACIÓN DE LAS HERRAMIENTAS DISPONIBLES** | 15 |
| **DESARROLLO DEL FLUJO DE CONVERSACIÓN** | 4 |

*Tabla 3-1. Tareas del primer bloque*

### 3.1.2 Segundo Bloque

| TAREA | Tiempo/Horas |
|---|---|
| **DESARROLLO DEL ASISTENTE CONVERSACIONAL** | 30 |
| **CREACIÓN DE LA CONVERSACIÓN** | 80 |
| **INTEGRANDO TONE ANALYZER** | 10 |
| **INTEGRANDO NATURAL LANGUAGE UNDERSTANDING** | 10 |
| **INTEGRANDO EL TRADUCTOR DE IDIOMAS** | 3 |
| **PROTOTIPO DE FUNCIONAMIENTO BÁSICO** | 15 |
| **CADENA DE HERRAMIENTAS (ENTREGA CONTÍNUA Y DESPLIEGUE)** | 10 |
| **CONECTANDO A TELEGRAM** | 5 |
| **CONECTANDO A SERVICIOS EXTERNOS (SPOTIFY)** | 2 |
| **CREANDO MODELO MACHINE/DEEP LEARNING** | 2 |
| **PROTOTIPO FINAL** | 40 |

*Tabla 3-2. Tareas del segundo bloque*

### 3.1.3 Tercer Bloque

| TAREA | Tiempo/Horas |
|---|---|
| **TESTEO DE LA INTEGRACIÓN/ CONVERSACIÓN** | 15 |
| **EVALUACIÓN** | 3 |
| **ANÁLISIS DE RESULTADOS** | 10 |

*Tabla 3-3. Tareas del tercer bloque*

# Capítulo 4
# Diseño y Desarrollo

En esta sección detallaremos varios temas importantes para el diseño y desarrollo del proyecto: tema en el que basaremos la conversación, proceso de detección de sentimientos y evaluación y el plan de desarrollo seguido. También se van a describir los distintos servicios que han sido necesarios para construir el *chatbot*, así como la forma de integración entre estos, para más adelante describir cómo se ha implementado el sistema.

## 4.1 Elección del tema de conversación

Para la conversación se ha decidido centrar el flujo del diálogo en la orientación académica de los alumnos dentro de la ETSII en el grado de informática. La idea era, inicialmente, que el *chatbot* pudiera hablar de diversos temas de carácter general, como son la política, geografía, estudios y aficiones, etc. A causa de algunos inconvenientes durante el proceso de desarrollo, que serán explicados en las conclusiones, se ha optado por reducirlo a tema específico como es la orientación académica.

Este asistente será capaz de orientar y ayudar a los alumnos a decidirse por una rama de la carrera según sus preferencias de asignaturas o incluso hobbies. Además, si nota que el alumno no está feliz con la carrera intenta averiguar el por qué y que rama debería elegir o si realmente debiera cambiar de carrera. En cualquier caso, cuando se trate de temas más complejos o delicados le propondrá asistir al orientador del centro. Durante todo el proceso irá analizando el tono de la conversación y extrayendo las emociones como ya hemos dicho previamente.

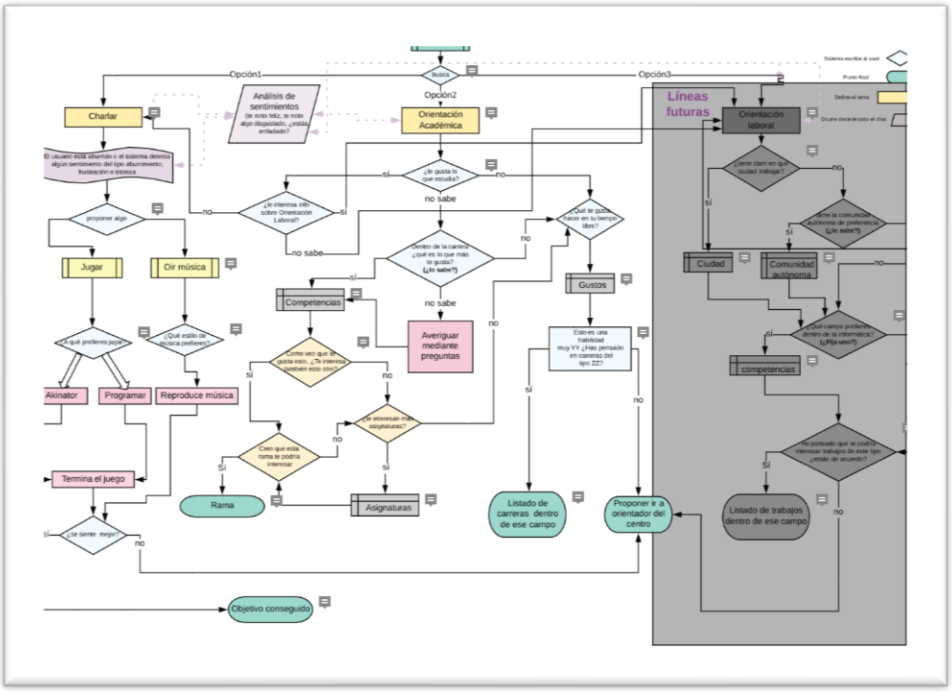



*Imagen 4-1. Diseño - Diagrama de flujo de la conversación*

## 4.2 Módulos de la aplicación

El proyecto se encuentra dividido en dos partes bien diferenciadas: Aplicación (servicios externos) y servicios en la nube.

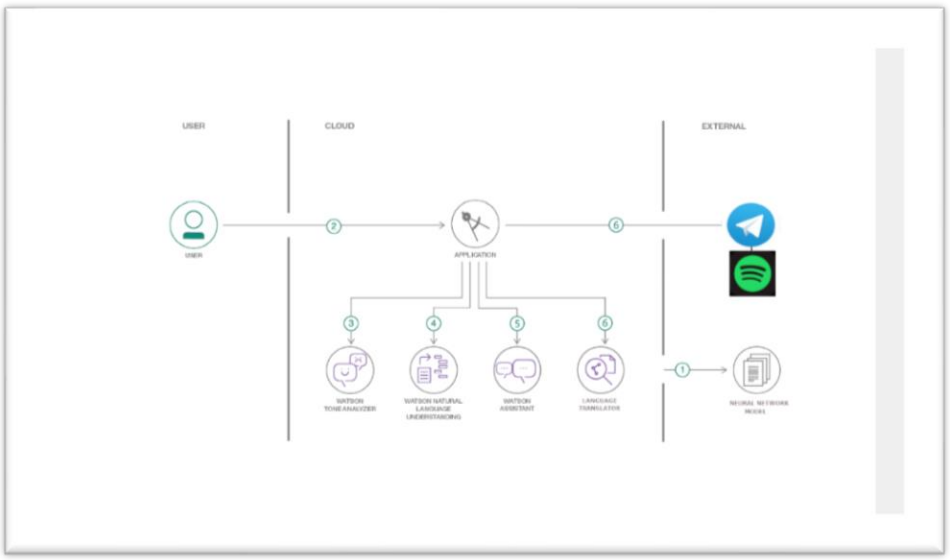



*Imagen 4-2. Diseño - Módulos*

La **Aplicación** es la parte del software que interactúa con los usuarios del sistema, recolectando los datos de entrada para enviarlos al **Servicio en la nube** que se encarga de procesar esa entrada y producir una respuesta que la aplicación recibe de vuelta y devuelve al usuario a través de la misma aplicación.

Este tipo de esquema se ha elegido por la flexibilidad que ofrece a la hora de desarrollar cada una de las partes de manera independiente. No es necesario para la interfaz saber cómo está implementada la conversación en la parte del servicio, solamente debe tener autorización de acceso para recibir el texto de entrada y saber en qué modo devolverá el mensaje. Gracias a esto tendremos un sistema más potente, robusto y más sencillo a la hora de identificar errores. A su vez, ambas partes se dividen en submódulos para realizar de una manera más eficiente y óptima las tareas que corresponden a cada uno de ellos. Veamos ahora en profundidad cada una de las partes.

### 4.2.1 Módulo Servicio en la nube

Este módulo será el encargado de la parte lógica del *chatbot.* Proporcionará todas las funciones necesarias para proveer de cada uno de los servicios de manera adecuada para que la conversación sea fluida y natural. Para ello, cada vez que se reciba un texto por parte del usuario pasará por los cuatro submódulos de manera individual ofreciendo cada uno de ellos la función para el que ha sido creado, esto es lo que conocemos como algoritmo "*Divide y vencerás*". Expliquemos esto con un ejemplo:

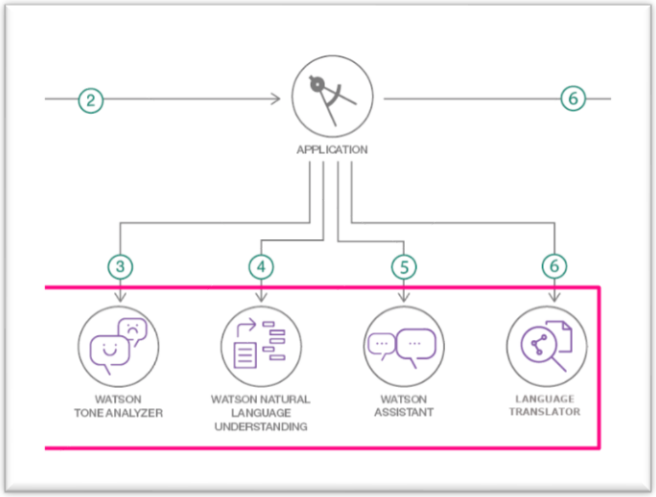



*Imagen 4-3. Diseño - Submódulos Servicios*

Recibimos la entrada del usuario a través de la aplicación, procesamiento de entrada hacia *Watson Assistant* alojado en *IBM Bluemix* mediante la API de *Telegram*.

A su vez, este texto pasará por el *Language translator* para que traduzca al inglés el texto del usuario. A continuación, el procesamiento del lenguaje natural *(NLU)* se encargará de extraer la información que necesita sobre sentimientos dentro de la carrera y del mismo modo extraerá las emociones del submódulo *Tone Analyzer* para dirigir la conversación en una dirección u otra según el estado de ánimo. Finalmente se procesará el mensaje de salida, mediante *Telegram*, usando *Watson Assistant* de nuevo, alojado también en *IBM Bluemix*.

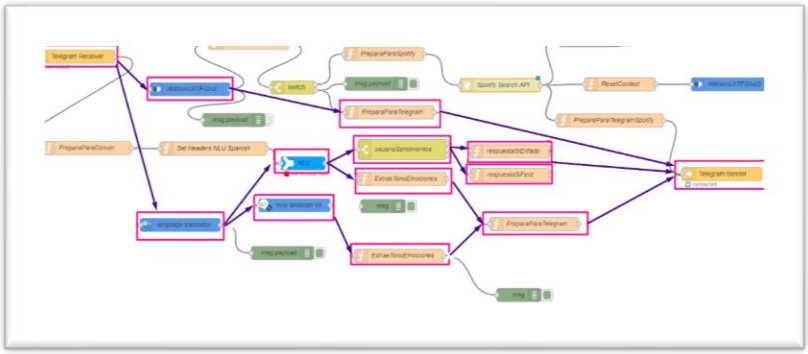


*Imagen 4-4. Diseño – Flujo de servicios en la nube*

La plataforma Bluemix es una plataforma desarrollada por IBM con servicio

PaaS (*Platform as a Service*), que ofrece servicios *cloud* y proporciona una plataforma y un entorno que permite a los desarrolladores crear aplicaciones y servicios que funcionen a través de la red. Soporta varios lenguajes de programación (*Java, Node.js, Python, .Net, etc.*) y servicios cadenas de herramientas que permiten crear, ejecutar, desplegar y gestionar aplicaciones en la nube. En la misma se pueden utilizar diferentes servicios y tecnologías en forma de alquiler. (Pagamos por uso), pero dispone de una cuenta gratuita que ofrece un espacio limitado y número de llamadas al mes.

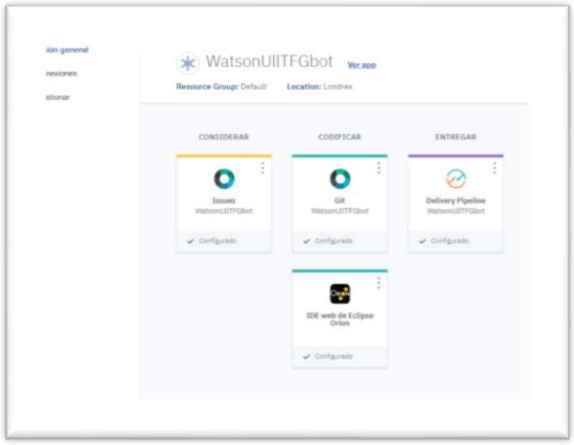



*Imagen 4-5. Diseño - Cadena de herramientas*

Todos estos servicios estarán alojados en el *Cloud Foundry* de IBM, se trata de un espacio en la nube para alojar y desplegar los servicios creados, pudiendo conectarlos todos entre sí a través de una aplicación que funcionará como paraguas de todas las demás. Lo estudiaremos en detalle en el siguiente capítulo.

### 4.2.2 Módulo aplicación

En este módulo se explica la parte visual del proyecto. ¿Cómo el usuario puede comunicarse con nuestro asistente? Pues tras varias pruebas de interfaz como son *Slack*, o aplicación *HTML* en el localhost y del propio espacio de trabajo de *Watson Assistant*, se ha decidido optar por la aplicación de mensajería instantánea y social *Telegram* ya que es una aplicación cercana al usuario, accesible, sencilla y gratuita. La podemos usar tanto desde móvil como desde el propio PC sin hacer ningún esfuerzo, por ello, se convirtió en la mejor opción para implementar nuestro asistente.

Pero ¿cómo podemos conectar *Telegram* con los servicios de *Watson?* Pues usando la herramienta ***Node-red***, que explicaremos con detalle en otro apartado. Pero de manera general, esta herramienta genera un flujo de funcionalidades que nos permite esta integración de una manera totalmente intuitiva y usando una cantidad de código pequeñísima, evitando así muchos de los errores de programación tan temidos como pueden ser la incompatibilidad de versiones, los errores de conectividad, etc.

Este paquete contiene un receptor y un nodo emisor que actúan como un *bot* de *Telegram.*

El nodo de entrada recibe mensajes del *bot* y envía un objeto de mensaje con el siguiente diseño:

- **msg.payload** contiene los detalles del mensaje
    - **chatId:** identificador único del chat. Este valor necesita ser pasado al nodo de salida cuando se envía la respuesta en el mismo chat.
    - **type:** el tipo de mensaje recibido: mensaje, foto, audio, ubicación, video, voz, contacto.
    - **content**: el contenido del mensaje recibido: string o file_id, u objeto con datos completos (ubicación, contacto)
- **msg.originalMessage** contiene el objeto del mensaje original del nodo subyacente telegram-bot-api lib.

El nodo de salida envía el contenido al chat especificado. Un flujo simple de entrada-salida se vería así:


*Imagen 4-6. Diseño - Node-RED*

Para ***Telegram*** basta con crear un bot en el servicio de mensajería a través del @BotFather, también será explicado en el siguiente apartado. Y luciría tal que así:

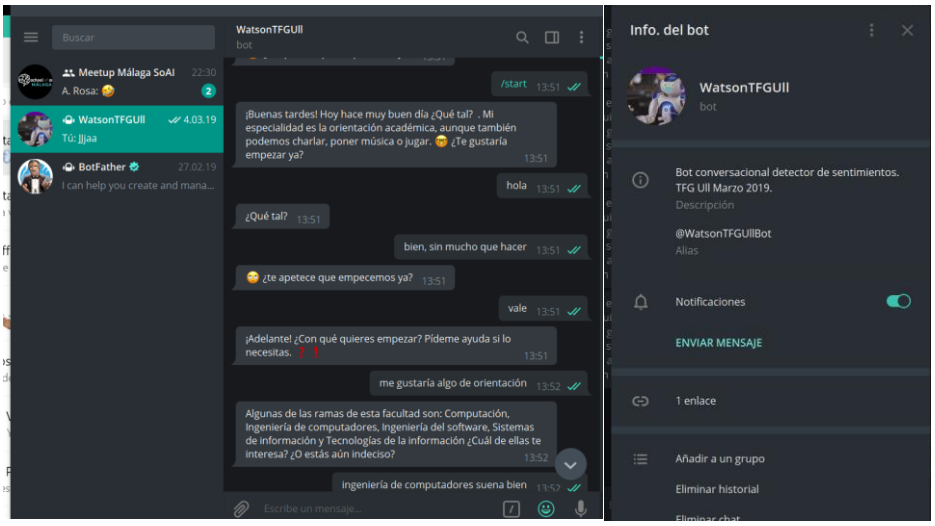



*Imagen 4-7. Diseño - Telegram*

Lo único que debemos hacer es conectarlo a *Node-RED* con el token de identificación de nuestro *bot* en *telegram* y listo.

## 4.3 Tecnologías utilizadas

El objetivo es conseguir mantener una conversación entre un usuario y el *chatbot* de la manera más natural posible, al igual que lo hacemos los humanos. A la vez, durante todo el proceso de la conversación, el objetivo será analizar el tono de la conversación y los sentimientos del alumno. Para este resultado de sentimientos y emociones tendremos en cuenta, únicamente, cuando hable sobre algo relacionado con la carrera. Para conseguir esto se ha optado por la solución que propone IBM.

### 4.3.1 Watson Assistant (@IBM, s.f.)

Utilizado como asistente conversacional, comprende la entrada en lenguaje natural y utiliza el aprendizaje de máquina para responder a los usuarios, de modo que simula una conversación entre humanos. La conversación se implementa a partir de, por un lado, las intenciones, que son los objetivos que tendrá el usuario al interactuar con el servicio; y, por otro lado, entidades, que representan los objetos que proporciona el contexto a una intención. Por ejemplo, una entidad puede ser el nombre de una asignatura que ayuda a distinguir durante el diálogo las preferencias del alumno dentro de la carrera.

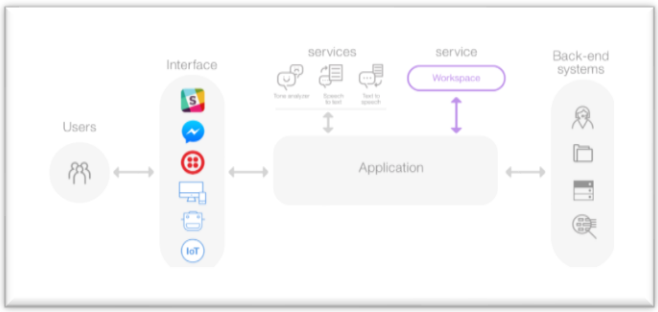



*Imagen 4-8. Tecnologías - Watson Assistant*

junto a algunos servicios complementarios:

### 4.3.2 Tone analyzer (@IBM, IBM Cloud, s.f.)

Para poder crear estrategias de diálogo que se ajustan al tono de la conversación extrae las emociones que detecta en las palabras del usuario donde los tonos son más fuertes, como pueden ser la alegría, el miedo, la tristeza, la ira, los tonos analíticos, de confianza o dudosos. Por ejemplo, puede responder ante un tono triste con el mensaje "¡Vaya! Te noto algo disgustado, ¿quieres que hagamos algo como escuchar música o jugar?”" o ante un tono de satisfacción con el mensaje "Te noto feliz. ¡Eso me gusta!".

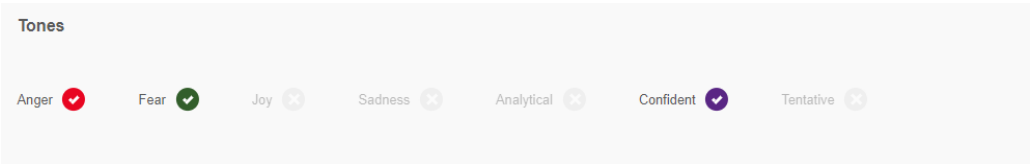



*Imagen 4-9. Tecnologías - Tone Analyzer*

### 4.3.3 Natural Language Understanding (@IBM, IBM Cloud, s.f.)

Este servicio es el encargado de analizar el texto extrayendo los metadatos del contenido como, por ejemplo, en nuestro caso, las entidades y palabras claves para poder hacer el análisis de sentimientos, positivos o negativos, devolviendo un porcentaje de confianza de entre 0 y 1.

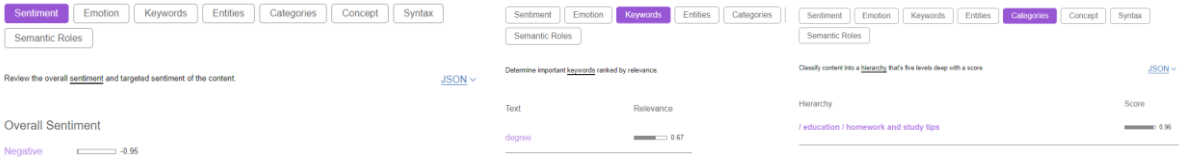



*Imagen 4-10. Tecnologías - NLU*

### 4.3.4 Language Translator (@IBM, IBM Cloud, s.f.)

Necesario integrar este traductor ya que los servicios de Watson no son capaces de detectar las emociones si el idioma de entrada es el español, es solo capaz de extraer el sentimiento, si es positivo o negativo. Por lo tanto, se encarga de traducir la entrada al inglés y cuando reconoce algo sobre la carrera el *NLU* se encarga de extraer la emoción y el *Tone Analyzer* el tono de la oración.


*Imagen 4-11. Tecnologías - Soporte de idiomas*

Todo ello con el objetivo de generar la conversación principal y detectar los sentimientos/emociones cuando el usuario hable de las entidades de la carrera. Podemos confinar entonces que, con estas capacidades, *Watson* permite satisfacer y gestionar de forma automatizada todos los requisitos del usuario mediante una interfaz conversacional, pudiéndose utilizar dentro de un sitio web, aplicación móvil, aplicación de mensajería o red social o incluso un robot capaz de gesticular.

Como servicios externos invocados a través de la API de Watson, se usarán:

### 4.3.5 Spotify

Para proponer algunas canciones al alumno en caso de que quiera escuchar algo de música. El alumno pide una canción, especifica el género musical y el *bot* le propone una de manera aleatoria mediante la API de *Spotify*.

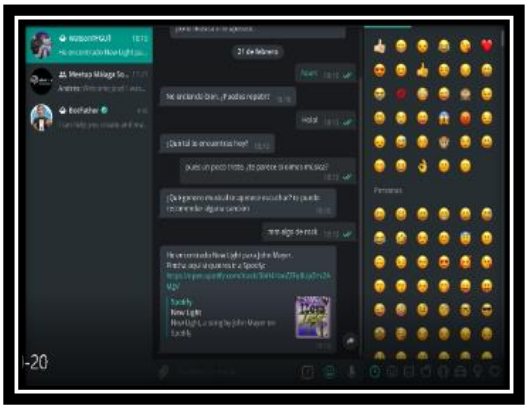



*Imagen 4-12. Tecnologías - Spotify*

### 4.3.6 Telegram

Este ha sido el servicio principal seleccionado para mantener la conversación con el alumno ya que podemos hablar con el asistente integrando todos los servicios de Watson en una misma aplicación. Y, además, es de código abierto, es rápido de implementar, gratuito y además está a la mano de cualquiera mediante la aplicación móvil o de escritorio.

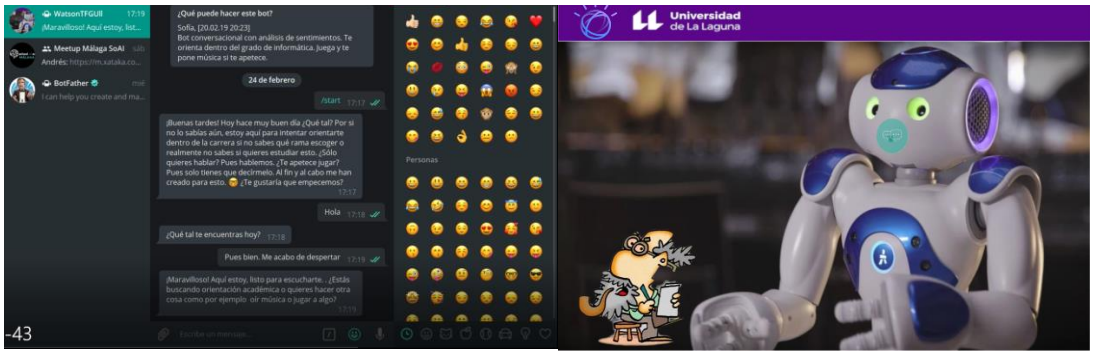



*Imagen 4-13. Tecnologías – Telegram*

### 4.3.7 Slack (Slack, s.f.)

Una de las interfaces del módulo de aplicación que se pensaba utilizar inicialmente, pero debido a sus limitaciones y lo poco accesible de su uso se descartó para usar como interfaz el *telegram*.

Se trata de una herramienta de colaboración en la que poder trabajar en equipo

para llevar a cabo los proyectos, permite abarcar desde los primeros pasos de un proyecto hasta conversaciones de presupuestos. Se integrará *Watson Assistant* en esta herramienta y creará un canal de conversación donde poder hablar con nuestra *bot* de manera sencilla. No se ha elegido para ser la herramienta principal ya que no permitía inicialmente trabajar con todos los servicios de *Watson* demanera simultánea (*NLU, Tone, Assistant, Translator*). Pero como ventaja he de decir que permite visualizar botones como método de opciones de la conversación y usar código *HTML*, que en el caso de *Telegram* no es posible.

Para visualizar nuestro asistente en slack, solamente debemos integrar las habilidades con el canal de Slack, mediante las credenciales del servicio. Este nos permitirá conversar con nuestra app en cualquier momento.

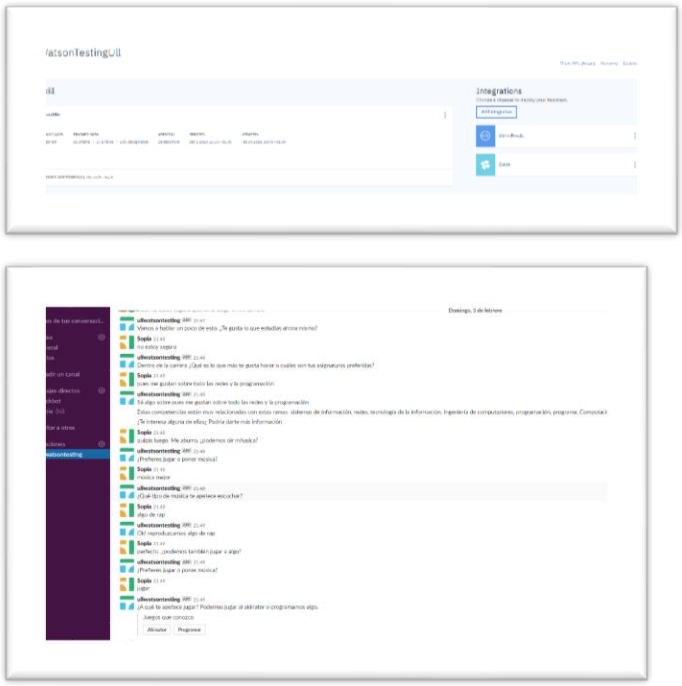


*Imagen 4-14. Tecnologías - Slack*

### 4.3.8 Node-RED (Foundation., s.f.)

Es una herramienta para conectar el *Internet de las cosas* de forma sencilla y gráfica incluyendo dispositivos de hardware, *API's* y servicios en la nube. Está construido sobre *Node.js* y aprovecha el enorme catálogo de módulos de nodo para proporcionar una herramienta que es capaz de integrar muchos sistemas diferentes. Centrándonos en los servicios de *Watson*, esta instancia se ejecuta como una aplicación *IBM Cloud*, que le da acceso a la amplia gama de servicios disponibles en la plataforma. La idea es crear un flujo de aplicación a través de nodos que pueden ser tanto nodos de entrada como de salida (*hardware, http response*, cámara*, websocket*, etc.), nodos de función o de plantillas, de unión o intercambio, nodos que enlacen con aplicaciones de terceros como email, *Twitter*, el tiempo, *IBM Watson* o nodos que conecten con bases de datos y multitud de funcionalidades más. Una aplicación realmente completa y potente con una interfaz gráfica simple y amigable.

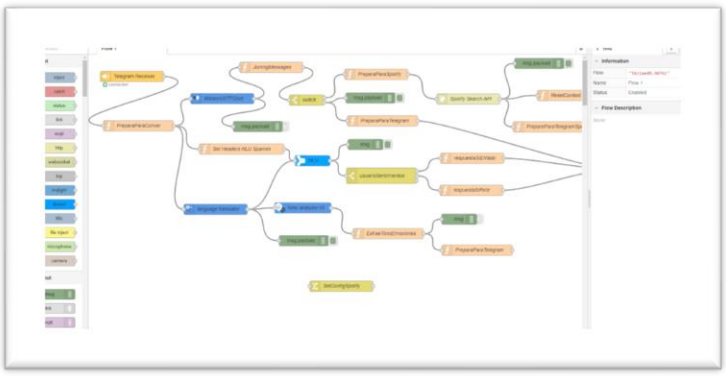


*Imagen 4-15. Tecnologías - Node-RED*

### 4.3.9 Cloud Foundry (@IBM, Cloud Foundry Documentation, s.f.)

Es el espacio donde desplegar y escalar nuestra aplicación sin necesidad de configurar y gestionar los servidores de forma manual, usaremos *Cloud Foundry*, garantizando que los aspectos de la compilación y el despliegue de código permanezcan perfectamente coordinados con cualquier servicio adjunto, ofreciendo una iteración de las aplicaciones rápida, consistente y fiable, aquí haremos la conexión de todos los servicios utilizados.

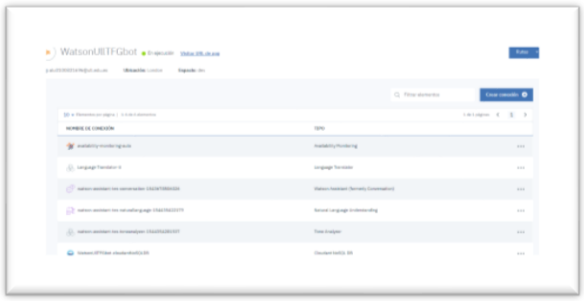


*Imagen 4-16. Diseño - Servicios Conectados*

Una vez hemos creado la aplicación, a la que hemos llamado WatsonUllTFGbot, realizamos todas las conexiones con los servicios que utilizaremos. Tal y como vemos en la imagen disponemos de casi 2GB de memoria para esta instancia, lo que nos permite hacer bastantes llamadas y uso del servicio. Como paquete de compilación hemos elegido la SDK de Node.js.

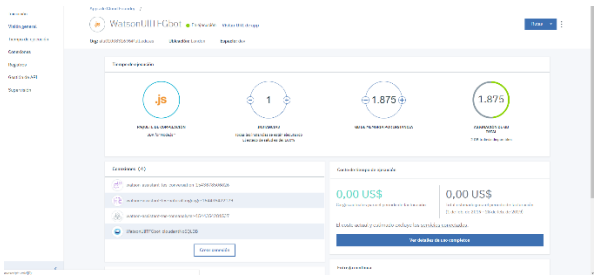


*Tabla 4-1. Diseño - App de Cloud Foundry*

Desde el panel de control de IBM cloud veremos las aplicaciones que hemos creado además de todos los servicios locales y los desplegados en la nube.

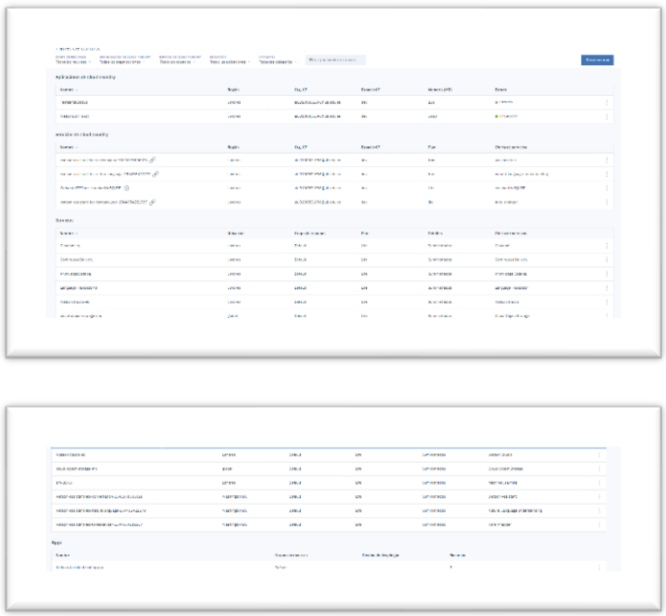


*Tabla 4-2. Diseño - Apps y Servicios en el Panel de control*

### 4.3.10 API en el localhost con NPM y Node.js (@IBM, GitHub, s.f.)

Cuando se comenzó a desarrollar el proyecto, las pruebas de conectividad entre los servicios de *Watson* y la interfaz se hacía mediante la API de *IBM Watson* con *Node.js*, esta API era la encargada de hacer la comunicación con el usuario. A través de la cadena de herramientas del propio *cloud foundry* de *Watson* (*eclipse orion y github*) se creó una interfaz con *HTML, CSS y javascript* para poder interactuar con el asistente.

El procedimiento era quizás un poco más tedioso que con la herramienta de Node.js ya que esta vez teníamos que trabajar con código, además de tener que utilizar varios paquetes y librerías, entre ellas dos paquetes de *NPM: Express,* para montar nuestro servidor *HTTP* y manipular las peticiones del usuario y el *Watson SDK API Node.js* que es la biblioteca responsable de tener acceso a los servicios de *IBM Watson.*

Inicialmente creamos el servidor local *http://localhost: 3000/asistente/mensaje* que podíamos ejecutar a través de nuestra línea de comandos.

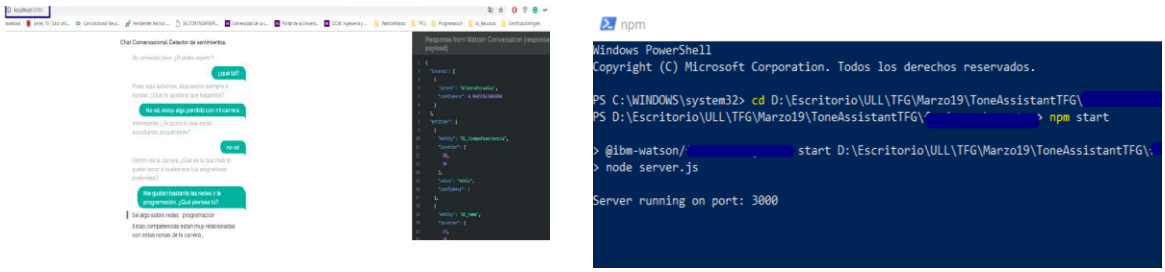



*Imagen 4-17. Diseño - NPM API*

## 4.4 Desarrollo de la aplicación

### 4.4.1 Watson Tone analyzer

El servicio *IBM Watson Tone Analyzer* utiliza el análisis lingüístico para detectar tonos emocionales y de idioma en un texto escrito. Para que una conversación sea lo más natural posible es lógico pensar que la conversación está incompleta si no es capaz de tener en cuenta el tono de ésta. De ahí, la importancia de este servicio. Este servicio no se usa solo para asistentes conversaciones, sino también en empresas de banca, logística, comerciales, etc. que quieren conocer las emociones de sus clientes a través de cómo se expresan en twitter, por ejemplo. Podemos incluso analizar el tono del mensaje de un email antes de enviarlo o conocer si un cliente de operador móvil está contento o descontento con el servicio que esa empresa ofrece.

Podemos acceder a las prestaciones del servicio a través de *API REST HTTP* o a través de SDK simplificando el desarrollo de la aplicación en diversos lenguajes y entornos ampliamente utilizados, que incluyen Node.js*, Java, Python y Apple iOS*.

Para poder analizar el tono del texto de un individuo, debe pasar el texto de entrada al servicio mediante *GET* o *POST /v3/tone_chat.* En ambos casos, el servicio devuelve el análisis en formato *JSON*. Todos los SDK permiten la autenticación mediante sus credenciales de servicio o mediante una señal de autenticación.

*Imagen 4-18 - Desarrollo - SDK Node.js*

Para ejecutar *tone analyzer* en nuestro pc necesitamos tener instalado *Node.js* y *NPM* o en nuestro caso *Node-RED*. Resultado de la ejecución de este último es

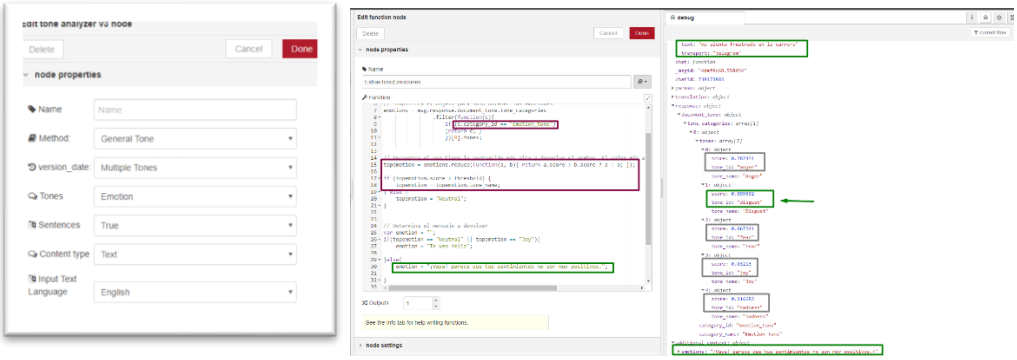


*Imagen 4-19. Desarrollo - Tone Analyzer .RED*

Podemos ver, en la imagen, que *Watson* dio resultados basados en el tono general de la oración. *Enfado 0.787 Disgusto 0.09 Miedo 0.06 Feliz 0.03 tristeza 0.21*. Por lo tanto, el sentimiento de enfado es el tono más fuerte de la oración

Los idiomas soportados de entrada son inglés y francés, y de salida acepta variedad de lenguajes, incluyendo el español. Los tonos que analiza son emociones, lenguaje y social.

El flujo de la conversación hacia el *tone analyzer* es de la siguiente manera:

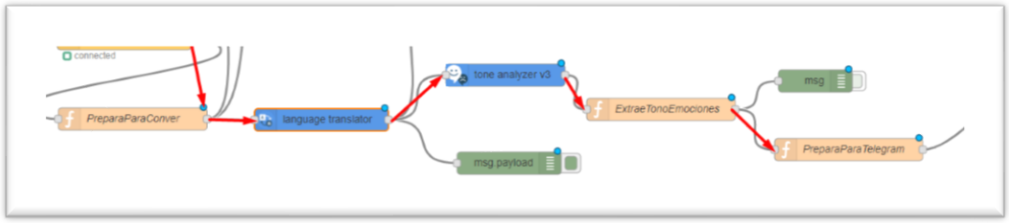
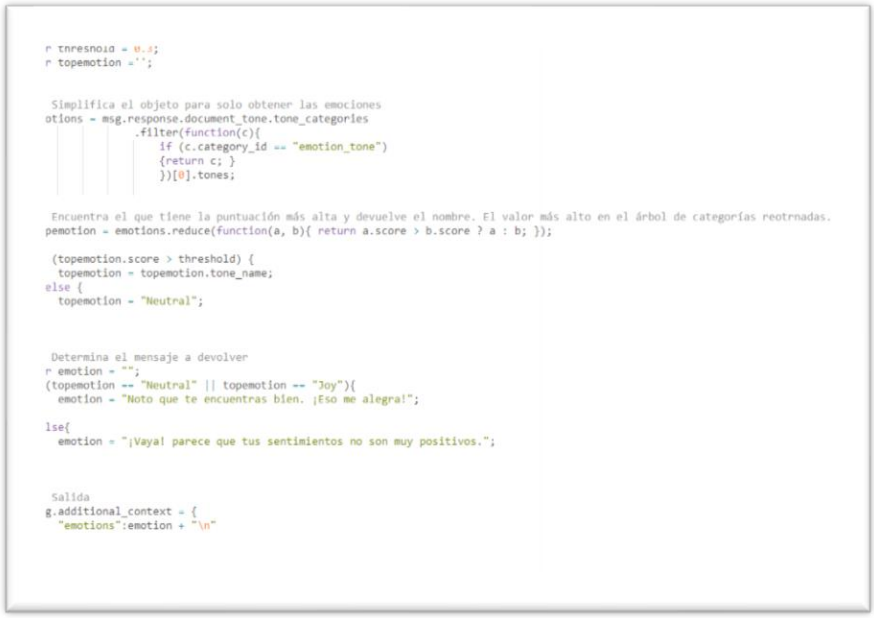



*Imagen 4-20. Desarrollo - Flujo Tone*

Recibe el texto de entrada, pasa a ser traducido por *Language Translator* de modo que sea capaz de detectar de qué emoción se trata, lo envía a una función que extraerá el tono de las emociones detectadas y envía la respuesta a la salida, si la emoción es de alegría o felicidad dirigirá la conversación hacia el nodo correspondiente y en caso contrario la dirigirá hacia otro nodo de la conversación, del mismo modo que lo hace el *NLU.*



```
r threshold = 0.6;
r topemotion ='';


 Simplifica el objeto para solo obtener las emociones
otions = msg.response.document_tone.tone_categories
            .filter(function(c){
                if (c.category_id == "emotion_tone")
                {return c; }
                })[0].tones;


 Encuentra el que tiene la puntuación más alta y devuelve el nombre. El valor más alto en el árbol de categorías reotrnadas.
pemotion = emotions.reduce(function(a, b){ return a.score > b.score ? a : b; });

 (topemotion.score > threshold) {
  topemotion = topemotion.tone_name;
else {
  topemotion = "Neutral";


 Determina el mensaje a devolver
r emotion = "";
(topemotion == "Neutral" || topemotion == "Joy"){
  emotion = "Noto que te encuentras bien. ¡Eso me alegra!";

lse{
  emotion = "¡Vaya! parece que tus sentimientos no son muy positivos.";


 Salida
g.additional_context = {
  "emotions":emotion + "\n"
```

*Imagen 4-21. Desarrollo - Extrae emociones*

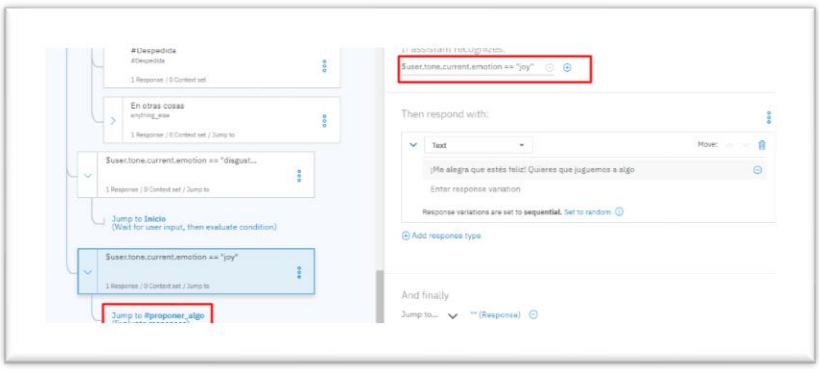



*Imagen 4-22. Desarrollo - Tone Analyzer Assistant*

### 4.4.2 Natural Language understanding

La función más importante de un *chatbot* es su capacidad para entender el lenguaje natural, lenguaje humano, y en base a eso, mantener una conversación fluida y con sentido dentro del contexto de la conversación. Podemos decir que el *NLU* es la capacidad de la máquina para comprender el texto o el habla humanos, extrayendo el significado correcto a pesar de los problemas de acentos, faltas de ortografía, pronunciación o errores gramaticales.

En este sentido cabe destacar de este servicio la capacidad de reconocer y extraer entidades (categorías de palabras, números o fechas), la normalización (obviando errores ortográficos y gramaticales), el análisis sintáctico (identificando sustantivo, verbo y adjetivos para comprender la estructura de la oración y saber cómo afectará al significado) y el análisis de dependencias (identificando sujeto, objetos o acciones para encontrar fases independientes).

Comúnmente se le debe añadir un corpus de palabras o diccionario para que puedan entender las palabras y frases. Sin embargo, hoy en día, los *NLU* son capaces de aprender a medida que avanza la conversación. Y por último también incluye el análisis de sentimiento, detectando el tono de las oraciones y, en nuestro caso, solamente cuando detecta algunas palabras claves.

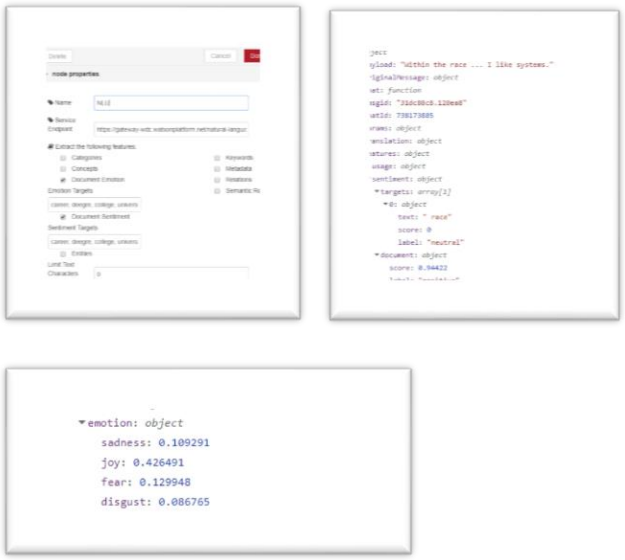



*Imagen 4-23. Desarrollo - NLU .RED*

El flujo de la conversación hacia el NLU es de la siguiente manera:

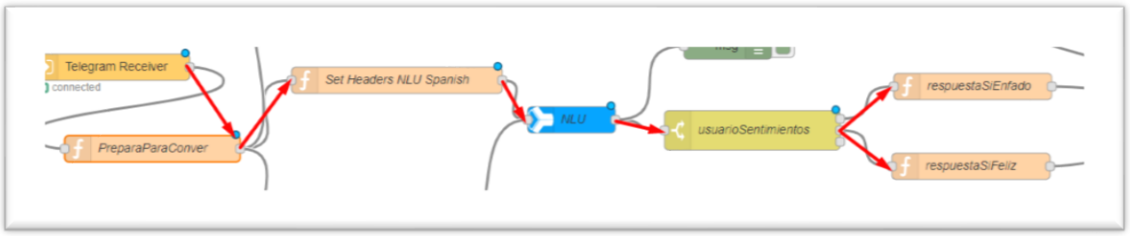



*Imagen 4-24. Desarrollo - NLU Flujo*

Recibe el texto de entrada, pasa a ser analizado por el servicio de NLU y si el sentimiento es mayor que 0 entonces es un sentimiento positivo en cualquier otro caso es negativo y entonces dirigirá el flujo de la conversación hacia el nodo correspondiente.

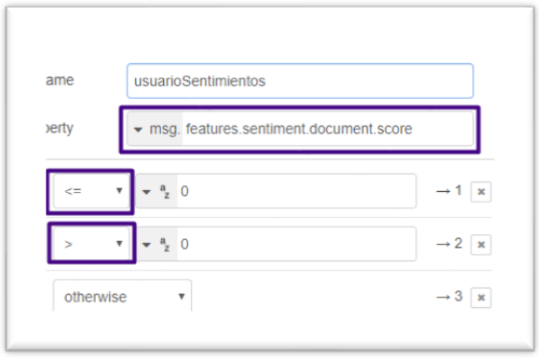



*Imagen 4-25. Desarrollo NLU*

## 4.4.3 Language translator

Como he mencionado en apartados anteriores, se ha decidido integrar como servicio el *Language translator* también como servicio de *IBM Watson* pues es la única solución encontrada para extraer las emociones de la conversación.

El traductor toma la entrada de texto, lo traduce del español al inglés y envía esto al *tone analyzer* para que pueda trabajar.

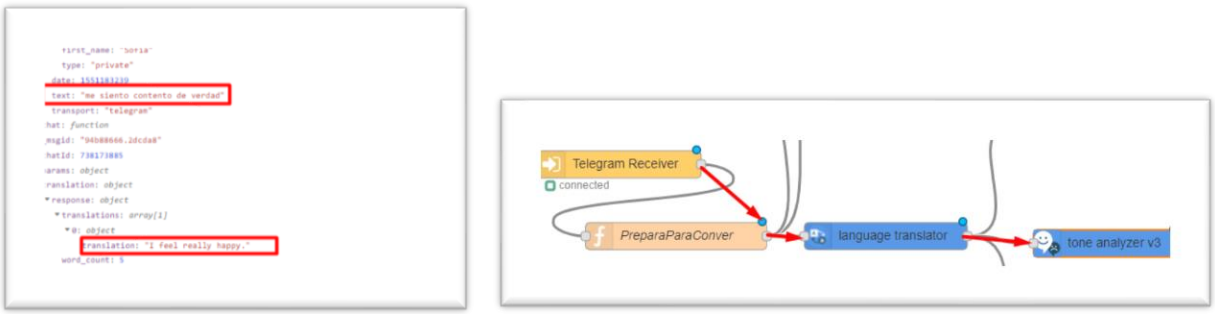



*Imagen 4-26. Desarrollo – Translator*

### 4.4.4 Watson Assistant (@IBM, s.f.) (@IBM, s.f.)

Como muchas de estas tecnologías funciona a través del procesamiento de lenguaje natural. El motor es entrenado con un banco de datos de consultas que vamos creando en el espacio de trabajo habilitado para ello. Las definiciones que la componen se basan en propósitos, intenciones (*intents)*, entidades (*entities)* y diálogo (*dialog)*. El propósito es asignar algo que ha escrito el usuario a una acción determinada.

Para definir estos propósitos se hará a través del *workspace* de *IBM Watson Assistant:*

- Creamos las habilidades (*Skills NLU*) y se habilitará un espacio con identificador único y credenciales de acceso.

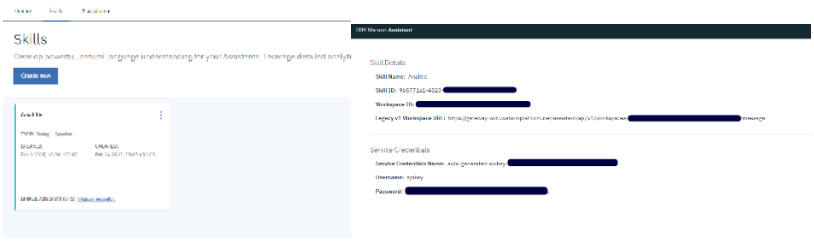



*Tabla 4-3. Desarrollo – Skills*

Podremos descargar el *JSON* en cualquier momento de manera que podamos integrar las habilidades del asistente en cualquier otra aplicación que use *Watson assistant.*

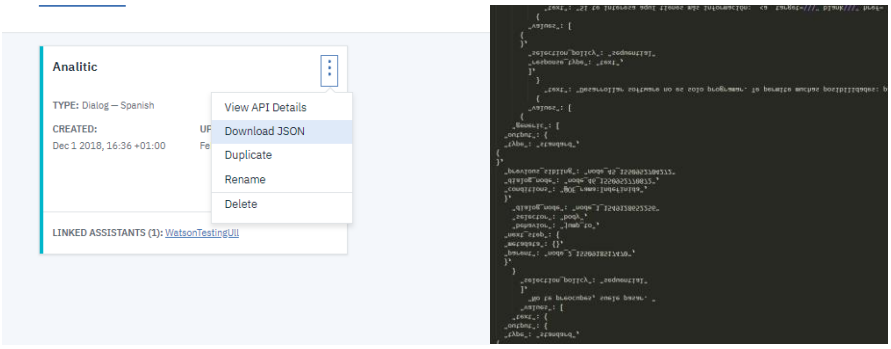



*Tabla 4-4. Desarrollo - JSON skills*

***Intenciones***

El primer paso para la conversación es que exista una intención, por lo tanto, lo primero que espera la aplicación es detectar esa intención. Para ello crearemos un conjunto de ejemplos que representará lo que el usuario podría decir. Es importante mencionar que no es necesario poner cada ejemplo que se nos ocurre ya que la NLU usará nuestros ejemplos para entrenarse y normalizar las entradas del usuario para entender entradas similares.

Podemos definir dos tipos de intenciones:

- De ámbito específico. Son aquellas que se centran en el tema de la conversación, en el caso de esta aplicación, la orientación laboral.

- De ámbito general. Acciones que puede solicitar el usuario independientemente del tema principal de la conversación, como saludo, agradecimientos o despedida.

Una vez hemos creado las intenciones, el siguiente paso será incluir ejemplos de usuario, cuantos más ejemplos se añadan, más posibilidad de que se reconozca la intención. Cada vez que se añade un nuevo ejemplo, Watson se entrena para reconocer el nuevo dato.

Algunos de ejemplos de ellos pueden ser:

- Estoy intentado decidirme por una rama
- No sé si estoy haciendo lo que me gusta
- ¿Cómo te encuentras?
- Me aburro. Hagamos algo

Ya que el alumno podría solicitar múltiples intenciones en un mismo mensaje, es posible asignar prioridades de proceso, así como regresiones.

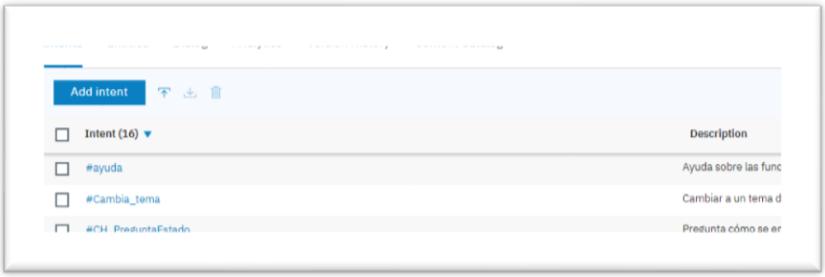



*Imagen 4-27. Desarrollo – Intenciones*

La opción *content catalog* contiene algunas intenciones del sistema para diferentes tipos de asistentes, entre ellos: los bancarios, control del bot, atención, utilidades, etc. Si nuestro asistente estuviera relacionado con alguno de los ejemplos disponibles nos facilitaría mucho el trabajo, sobre todo en la rapidez de desarrollo.

Un ejemplo del catálogo bancario lo mostramos en esta imagen.

*Imagen 4-28. Desarrollo - Content Catalog*

En el Apéndice A se puede ver las intenciones definidas para el proyecto.

### *Entidades*

Las entidades no son obligatorias, pero ayudan a crear un mejor flujo de la conversación cuando van ligadas a las intenciones. Son objetos del mundo real, sustantivos ligados a una acción. Las NLU pueden extraer los valores de los parámetros de la solicitud del usuario buscando entidades, ya sean las propias del sistema (sys-time, sys-data, etc.) como las definidas por nosotros durante el desarrollo.

Es recomendable crear varios grupos de entidades que apunten de manera única a una intención.

Al tratarse de sustantivos tenemos la posibilidad de asociar cada uno de los valores creados con infinitos sinónimos lo que nos permite tener una base de conocimientos muy amplia.

Algunos ejemplos de entidades son @OA_competencias (bases de datos, sistemas, programación), @OE_rama (Computación, sistemas de la información, etc.)

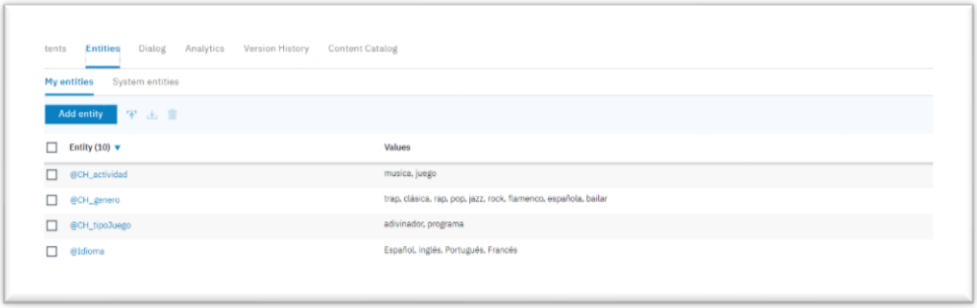



*Imagen 4-29.Desarrollo - Entidades*

En el Apéndice B se puede ver las entidades definidas para el proyecto.

### *Flujo de diálogo*

El flujo de diálogo es el área de la lógica de la conversación, dando la sensación de que el diálogo es natural y lo más similar a la conversación humana.

Una vez se han definido las intenciones y las entidades podemos pasar a crear la conversación. Es el cerebro de la aplicación, sin no se crea de manera correcta el diálogo no va a fluir por lo tanto hablar con el *chatbot* sería tedioso.

Para crear este diálogo he seguido el diagrama de flujo de la conversación, siendo cada nodo del diagrama un nodo en el diálogo. Se tienen en cuenta todas las entradas posibles del usuario y a medida que se van generando se va testeando para así ir creando simultáneamente las condiciones de salto y la evaluación de las intenciones definidas.

Cada salto y cada condición cuenta como un nodo, o sub-nodo, y ya que la versión gratuita solo nos permite crear 100, este es el número de nodos que tenemos.

Cada nodo debe contener una respuesta o condición de salto hacia otro nodo y en algunos de los casos acciones más complejas como, por ejemplo, analizar el tono de la conversación o lanzar Spotify para reproducir una canción. Para este tipo de acciones más complejas debemos ir guardando las variables, en lo que se conoce en

*assistant* como *context*, para no perder el tema del que estuviéramos hablando.

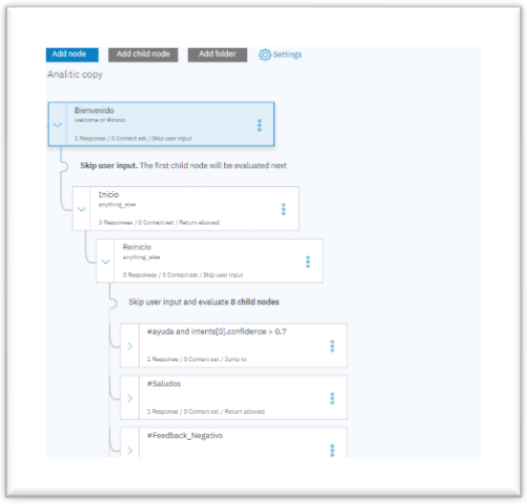



*Imagen 4-30. Desarrollo - Diálogo*

Tenemos la posibilidad de crear nodos de respuesta simple o de múltiples respuestas, en este caso, cada respuesta puede estar asociado a una condición distinta. Además, podemos activar los slots, esto permite que el asistente no salga del nodo hasta obtener el valor que está solicitando.

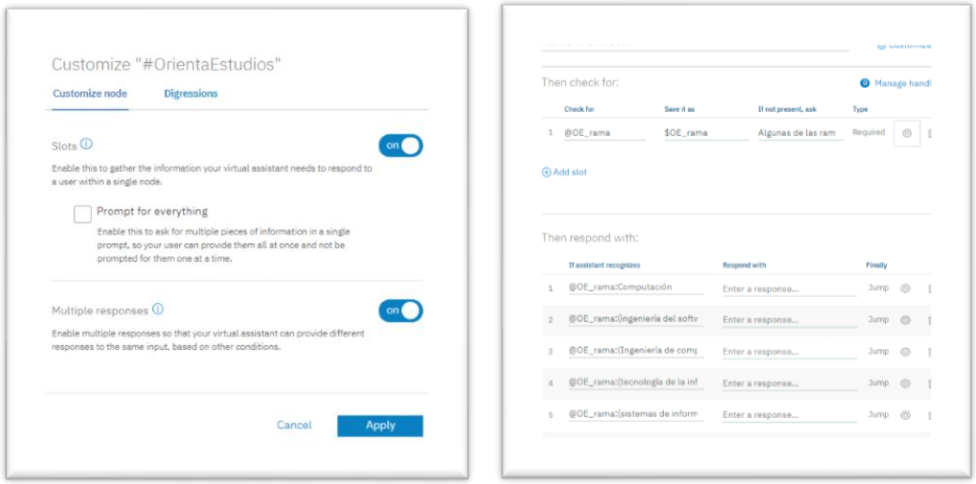



*Imagen 4-31. Desarrollo – Assistant Edición Simple*

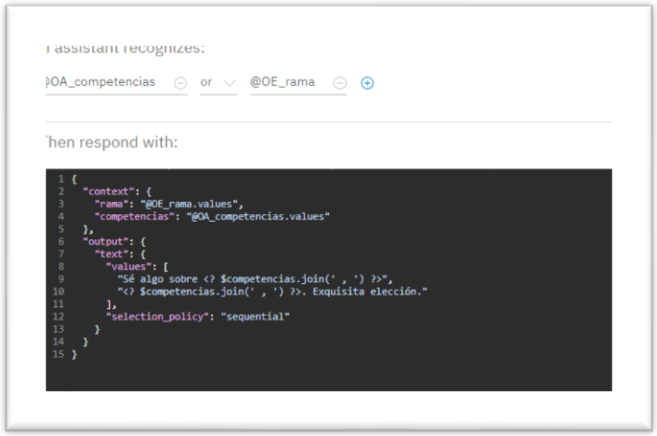



*Imagen 4-32. Desarrollo – Assistant Edición Avanzada*

Otra de las opciones realmente útiles que ofrece la conversación es la función de digresión, esto da la posibilidad de aun desviándonos del tema durante la conversación permita al alumno regresar al punto de la conversación en donde quedó. Por ejemplo, el alumno está hablando sobre sus gustos personales y de repente pide escuchar música, entonces en este momento pondrá la música que le está solicitando y a continuación regresará al punto dónde estaba peguntando sobre sus gustos personales.

Como nodos principales me gustaría destacar:

- Bienvenida: Ya que iniciará la conversación cada vez que se inicie el asistente.

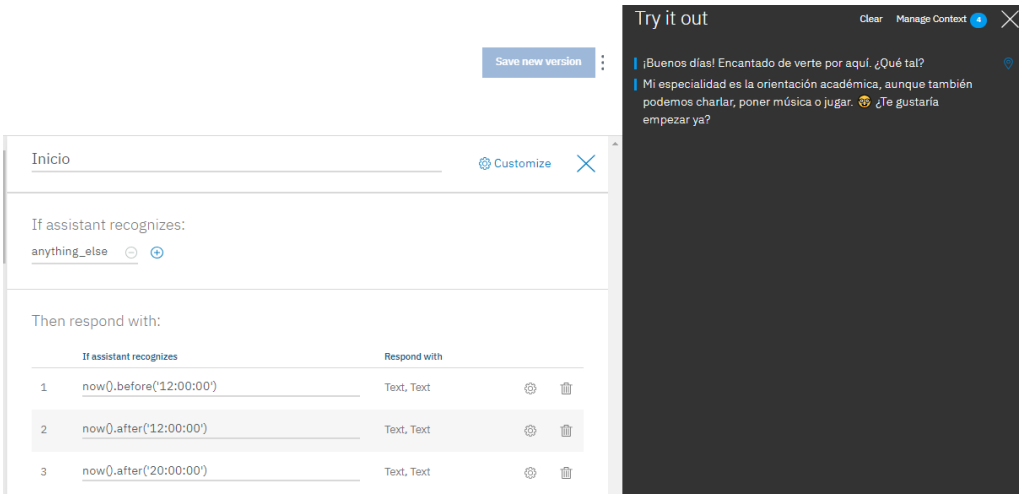



*Imagen 4-33- Desarrollo - Nodo Bienvenida*

- OrientaEstudios: Nodo principal del asistente. Preguntará al usuario si le gusta lo que estudia y según si la respuesta es afirmativa o negativa se moverá en la dirección señalada. Por ejemplo, en caso de que esté en duda y conozco sus asignaturas favoritas le dirá con qué competencias están relacionadas y a que rama pertenece. Si quieres más información le hablará sobre esta, si no está de acuerdo intentará averiguar mediante sus gustos qué rama o carrera le podría interesar.

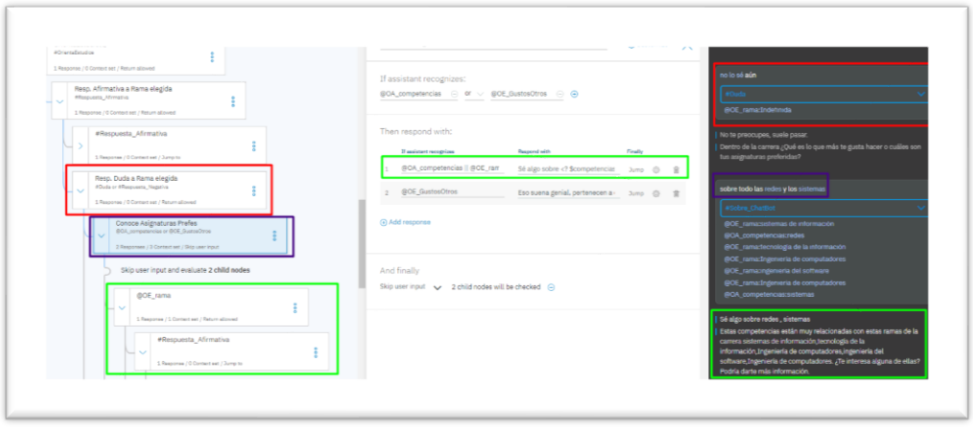


*Imagen 4-34. Desarrollo - Nodo Orientación*

- ProponerAlgo: Este nodo será el encargado de proponer una canción a través de Spotify con el género que el usuario ha elegido o proponer dos juegos, akinator y trivial de programación. A este nodo llega cuando el usuario se siente aburrido o frustrado, e incluso cuando se siente algo triste y no quiero hablar de los estudios.

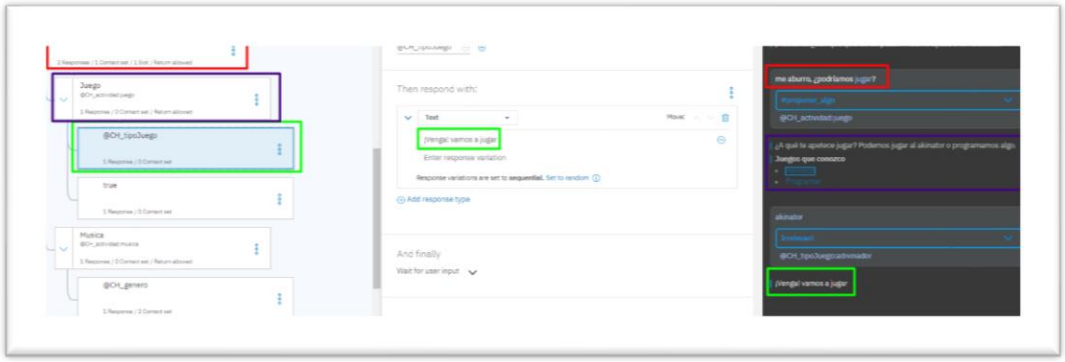


*Imagen 4-35. Desarrollo - Nodo ProponerAlgo*

- Entre otras cosas: A este nodo se llegará siempre que las opciones que proponga el usuario no sean reconocidas. Reiniciará la conversación proponiendo opciones las disponibles del asistente.

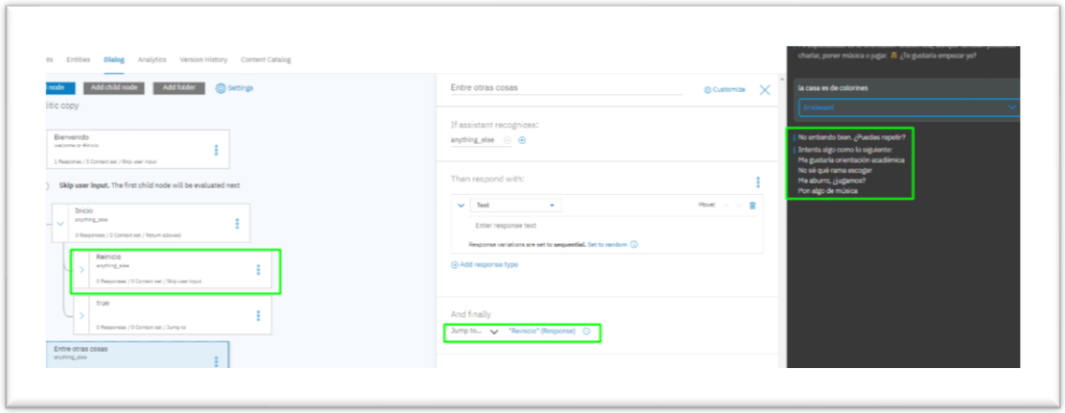
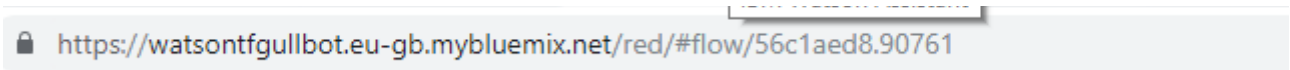


*Imagen 4-36. Desarrollo - Nodo Entre otras cosas*

El resto del diálogo la podemos encontrar en el Apéndice C.

### 4.4.5 Node- RED

Node-RED como ha sido explicado anteriormente, es una herramienta que nos permite conectar muchos sistemas y servicios de manera gráfica. La creación de una instancia de la aplicación en Bluemix permite el despliegue de la herramienta en un servidor gestionado por IBM, al que se accede mediante una dirección URL proporcionada por la plataforma.



Se crean tres flujos de conversación:

- **Watson Assistant**:

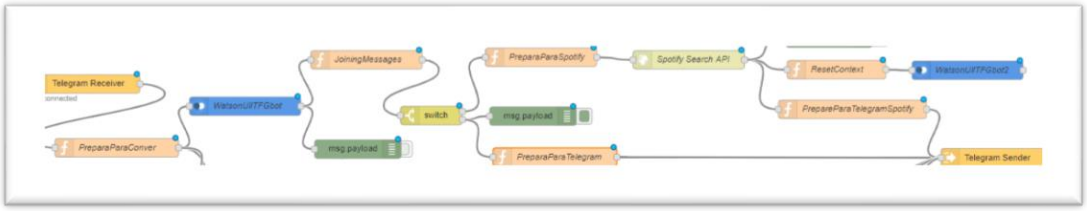



*Imagen 4-37. Desarrollo - Flujo Assistant*

- Entrada de texto desde *telegram*
- Preparación para la conversación, crea las variables para ello (*ChatID, type, content*)

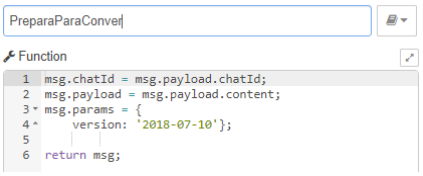



*Imagen 4-38. Desarrollo - Flujo Assistant*

- WatsonAssistantTFGULL, configuramos en este nodo las credenciales de acceso a nuestro workspace.

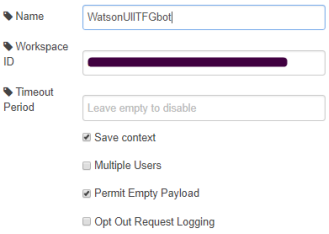



*Imagen 4-39. Desarrollo - Flujo Assistant*

- Joinin Messages, Une todas las posibles respuestas en un solo mensaje para enviar de vuelta a telegram.

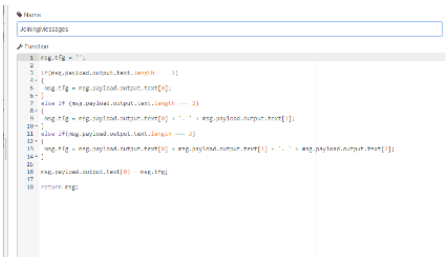


*Imagen 4-40. Desarrollo - Flujo Assistant*

- Switch
  - si la acción que recibe es escuchar música: Dirige el flujo a una app externa, *API de Spotify,* en la primera función establecemos las credenciales de acceso, en una segunda función se establece el

modo de acceso y el modo de devolver el mensaje. Finalmente, se envía esta información a *telegram* y podemos continuar con la conversación mientras suena la música.

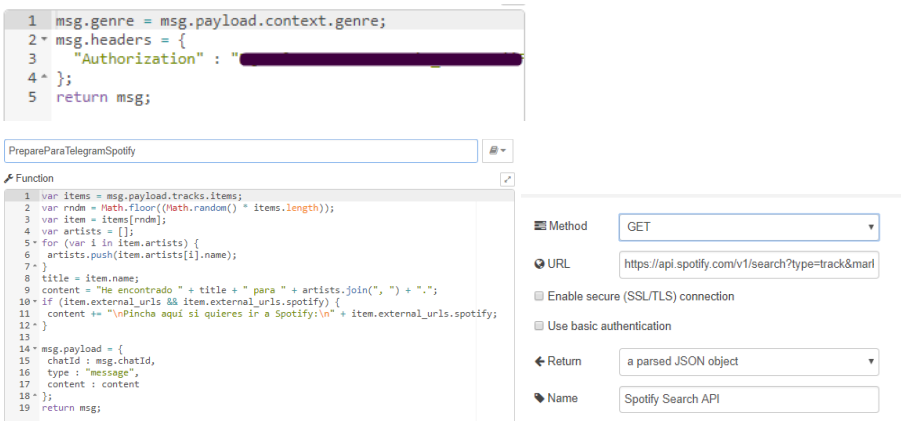



*Imagen 4-41. Desarrollo - Flujo Assistant*

- en cualquier otro caso devuelve el mensaje a telegram para continuar con la conversación.

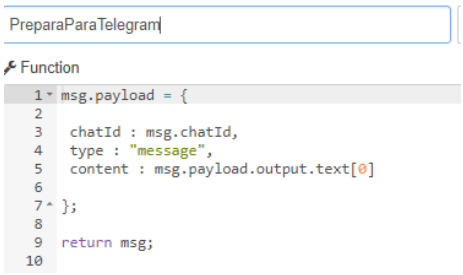



*Imagen 4-42. Desarrollo - Flujo Assistant*

- **Language translator + NLU**:

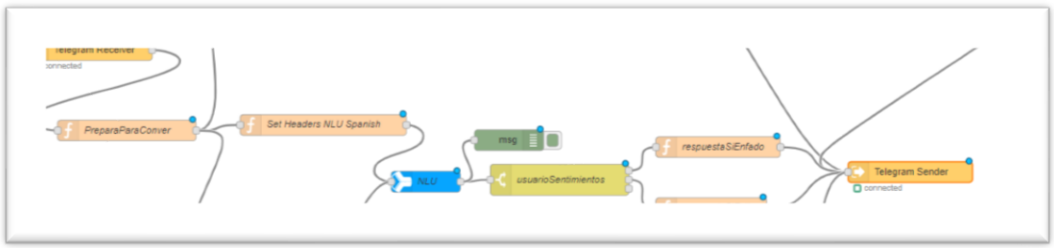



*Imagen 4-43. Desarrollo - Flujo NLU*

- Recibe la entrada de texto.
- Establece la cabecera del NLU

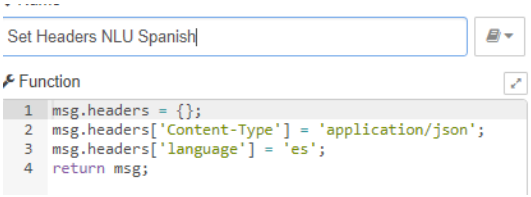



*Imagen 4-44. Desarrollo - Flujo NLU*

- Definimos en el nodo de NLU qué queremos que detecte, en este caso emociones y sentimientos filtrados por algunas palabras claves, relacionadas con la carrera.

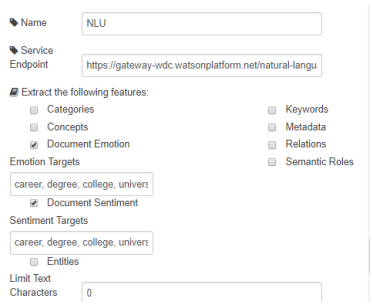



*Imagen 4-45. Desarrollo - Flujo NLU*

- Es el momento de identificar si los sentimientos con respecto a las palabras claves del paso anterior, son positivos o negativos, en ambos casos envía el valor a telegram pero con diferentes resultados.


*Imagen 4-46. Desarrollo - Flujo NLU*

- **Language Translator + Tone Analyzer**:

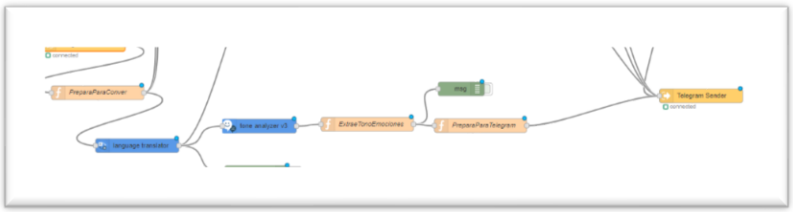


*Imagen 4-47. Desarrollo - Flujo Tone + Translator*

- Recibe el texto del usuario
- Traduce del español al inglés para extraer las emociones con el *tone analyzer*

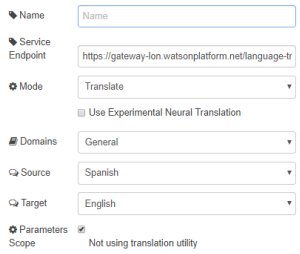



*Imagen 4-48. Desarrollo - Flujo Tone + Translator*

- Establece los parámetros del *tone analyzer* y las credenciales

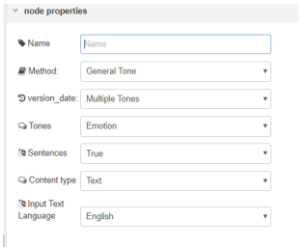



*Imagen 4-49. Desarrollo - Flujo Tone + Translator*

- Una vez tenemos esto pasamos a la función que extraerá estas emociones.

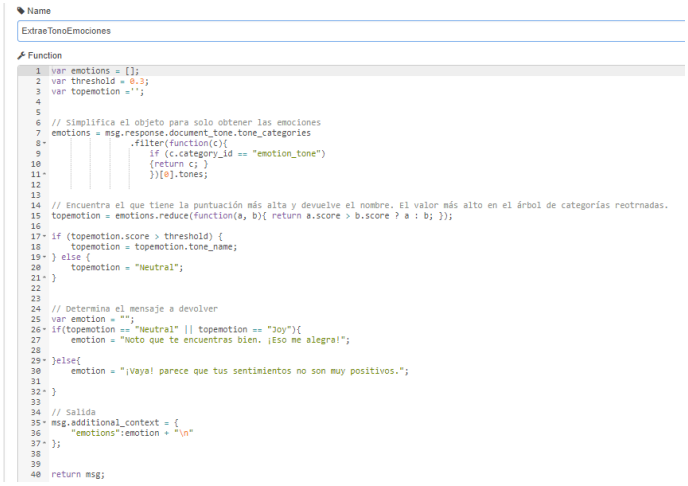



*Imagen 4-50. Desarrollo - Flujo Tone + Translator*

- Y finalmente devuelve este valor a IBM para que haga el análisis y tenga en cuenta que nodo del diálogo es el más acertado.

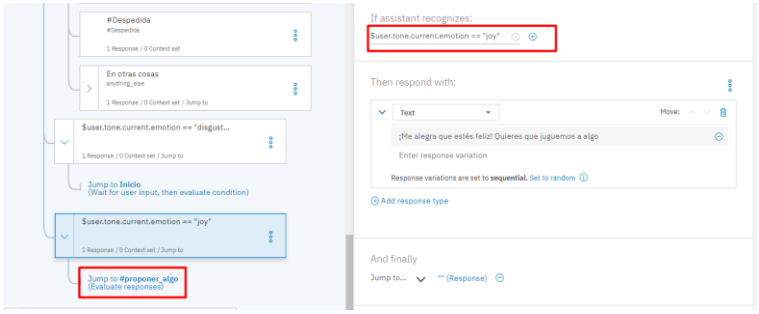



*Imagen 4-51. Desarrollo - Flujo Tone + Translator*

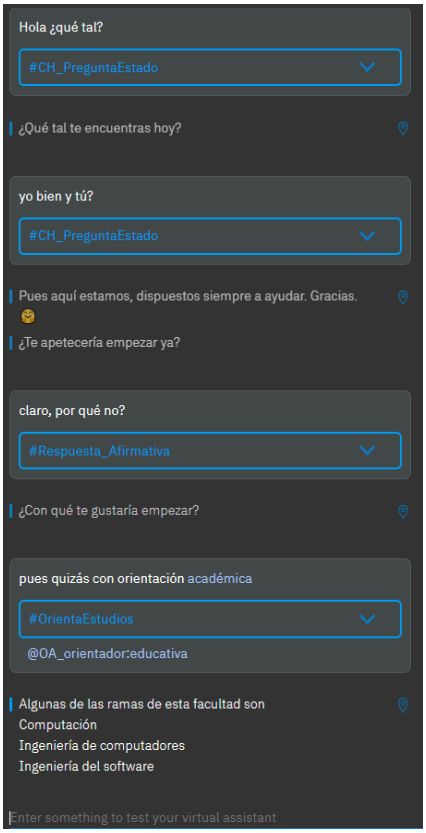
Todos los nodos en verde son los nodos de *Debug*, para identificar rápidamente de dónde vienen los errores. Puede imprimir el contenido completo de la traza o solo el mensaje que estamos recibiendo.

## 4.5 Análisis de resultados

Ya que el modelo usado para el trabajo ha sido el incremental nos ha facilitado el proceso de testeos ya que a cada módulo implementado podíamos probar que funcionara, de manera que al integrarlos entre ellos reconociéramos rápidamente de dónde venía el error.

### 4.5.1 Pruebas sobre servicios IBM Watson

El Watson *Assistant* ofrece la facilidad de ir testeando a medida que se va desarrollando la conversación mediante intenciones, entidades o diálogo, todo esto desde el propio *workspace* de *bluemix.* Mientras que el resto de los servicios de Watson: *language translator, NLU* y *tone analyzer* se fueron testeando con la la SDK de Node .js en el servidor local.




El panel de prueba de Watson nos permite ir viendo en tiempo real lo que vamos programando, siempre y cuando haya una entrada de texto. Como podemos ver en la imagen de la izquierda, el panel muestra en color blanco el **texto introducido** por el usuario, justo debajo enmarcado en azul se encuentra la **intención reconocida**, y bajo esta, si nos fijamos en la última entrada, aparece también la **entidad reconocida**. Finalmente, en color blanco y acompañado de una muesca azul encontramos el **texto de salida**, la respuesta del *chatbot*.

Además, el icono  nos permite activar o desactivar la opción de visualizar las entidades reconocidas.

En caso de que la intención reconocida no sea la que esperamos, tenemos la opción de modificarlo mediante el seleccionador disponible a la derecha del recuadro. Si aun así no se adecúa a ninguna de las existentes podemos marcarlo como irrelevante o crear una nueva.

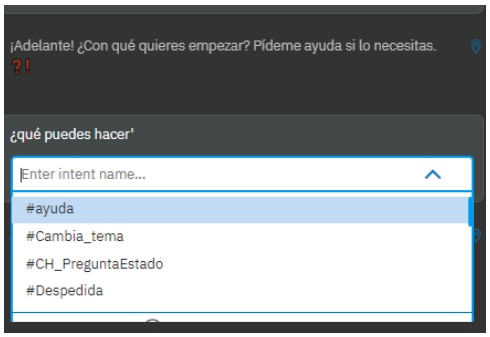



*Imagen 4-52. Pruebas - Assistant Bluemix*

Otra de las opciones que nos ofrece *Bluemix* es poder ver si los valores de las variables están siendo guardados de manera correcta o incluso crear algunas nuevas y darle un valor inicial o borrar las existentes a través del *Manage Context*.

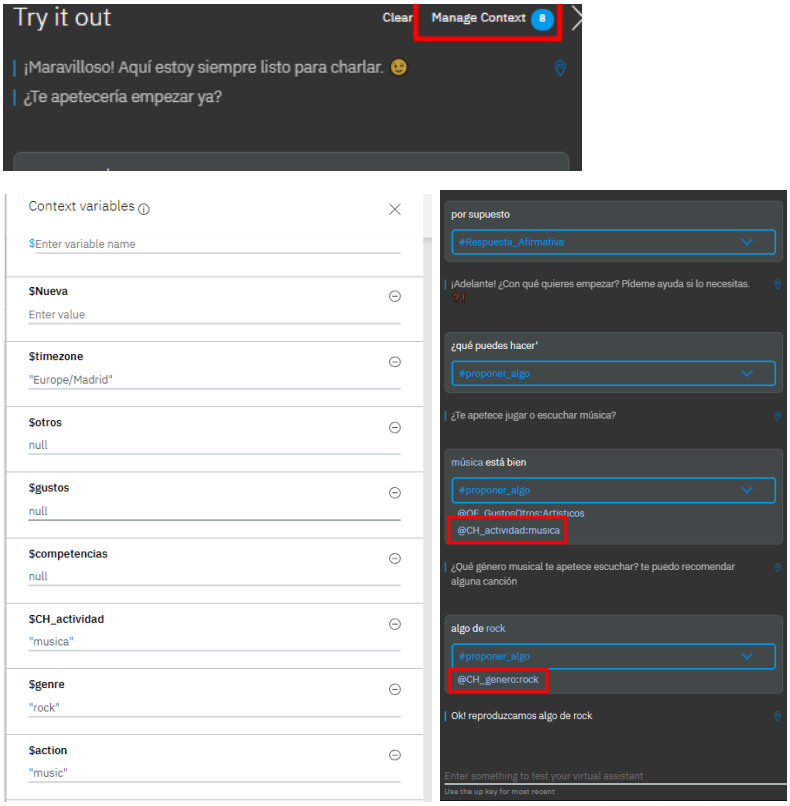



*Imagen 4-53. Pruebas - Assistant Bluemix*

Además, en el menú *analytics* podemos ver de manera gráfica cómo se comporta el *chatbot*. A través de métricas que ayudan a entender mejor a los usuarios. Entre estas métricas tenemos: total de mensajes, mensajes por conversación, total de usuarios y conversaciones por usuario.

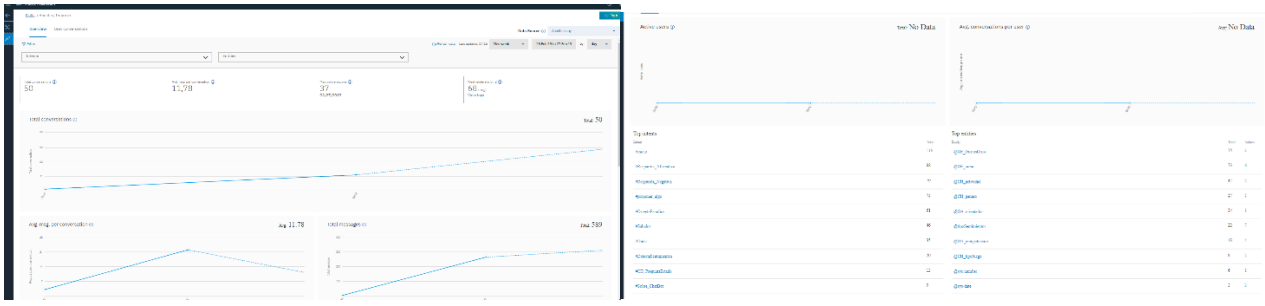


*Imagen 4-54. Pruebas - Assistant Analytics*

En el menú principal de la aplicación, *Cloud Foundry* nos permite la supervisión mediante entrega continua, de manera que es posible ver un resumen de pruebas y su estado.

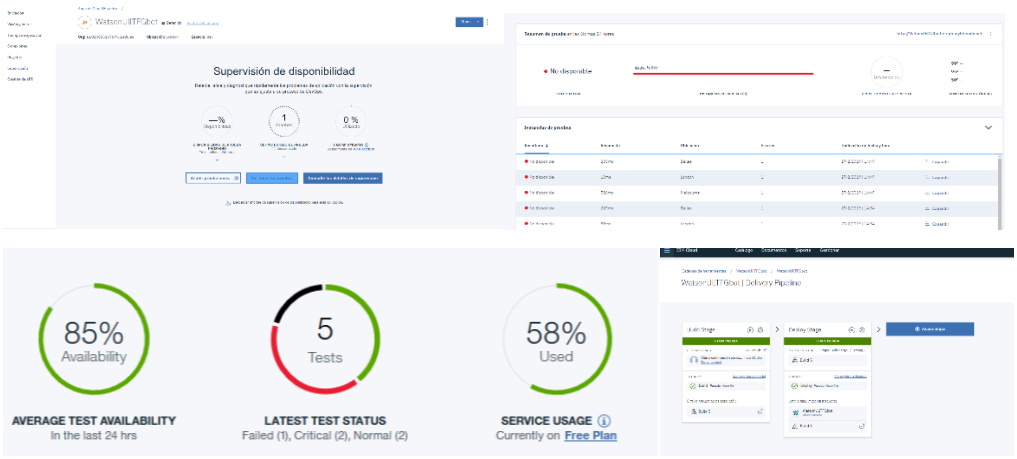
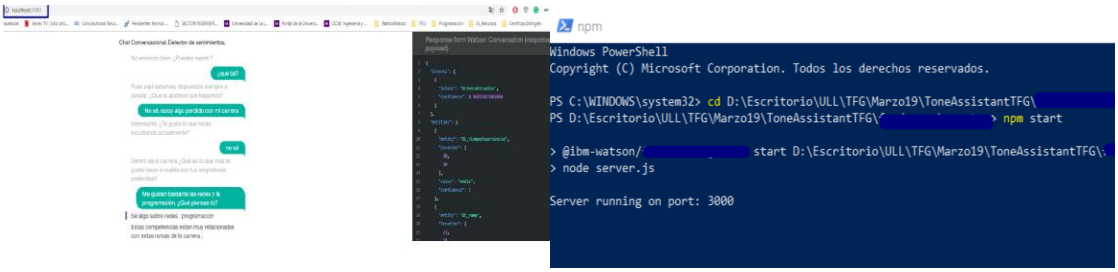



*Imagen 4-55. Pruebas - Supervisión Despliegue*

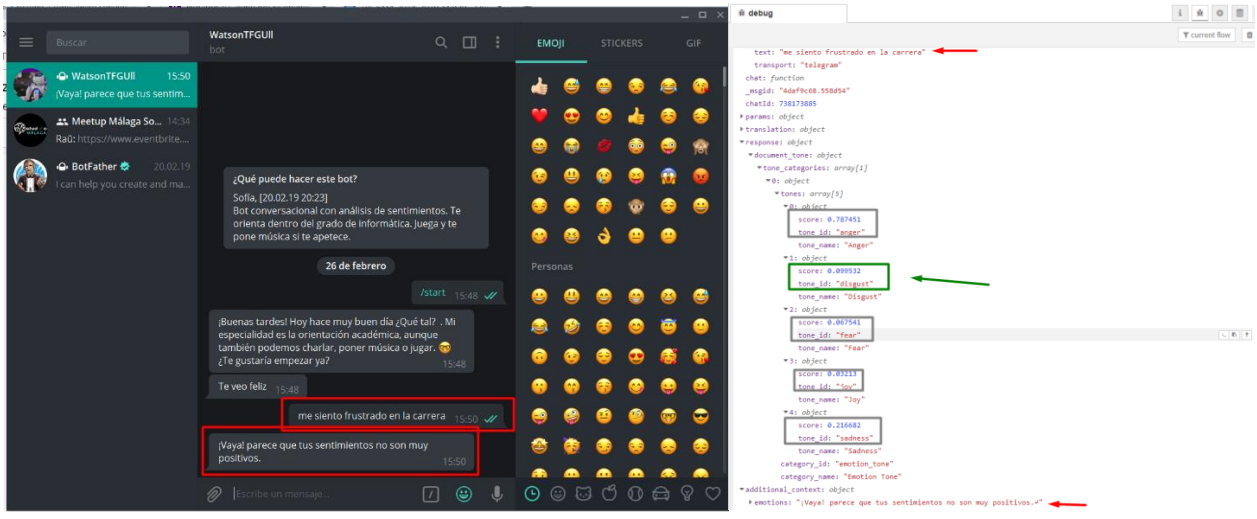
## 4.5.2 Pruebas en entorno real

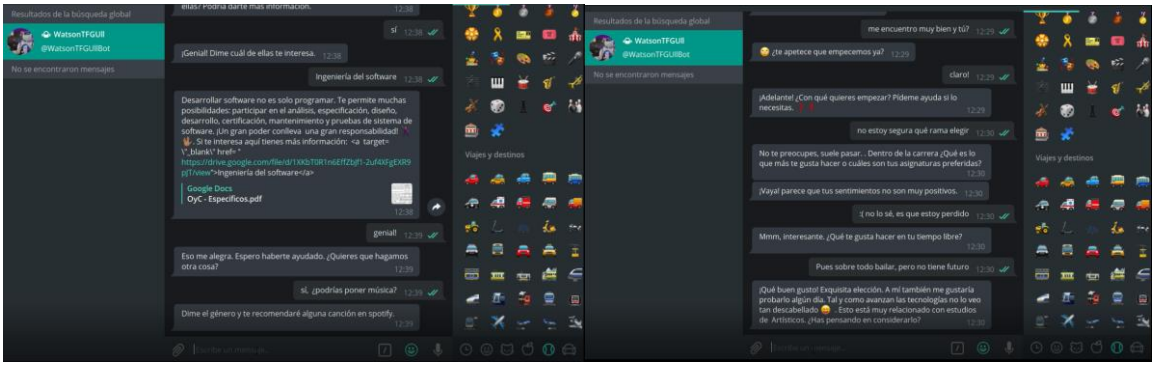
Para las pruebas de la integración de servicios se utilizó inicialmente la SDK de Node.js como explicamos en uno de los apartados del diseño. Pero era más costoso y el tiempo jugaba en mi contra.

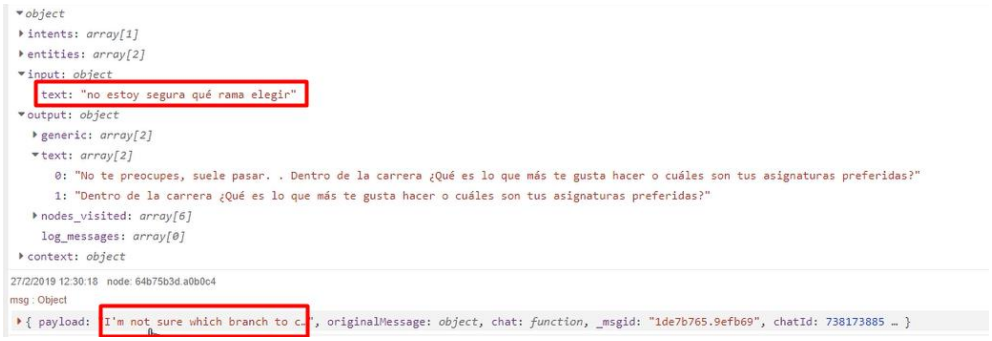
Los resultados que obteníamos se podían ver a medida que fluía la conversación en el panel lateral derecho de la aplicación.




Finalmente, y lo que ha ocupado la mayoría del testeo es el que ha sido creado integrando los servicios con *Node-RED* y *telegram*. De manera que confirmar que los servicios de terceros (Spotify) y el análisis de sentimientos trabajaran de manera conjunta con la aplicación. Y los resultados fueron un éxito, aquí algunas capturas de las conversaciones:





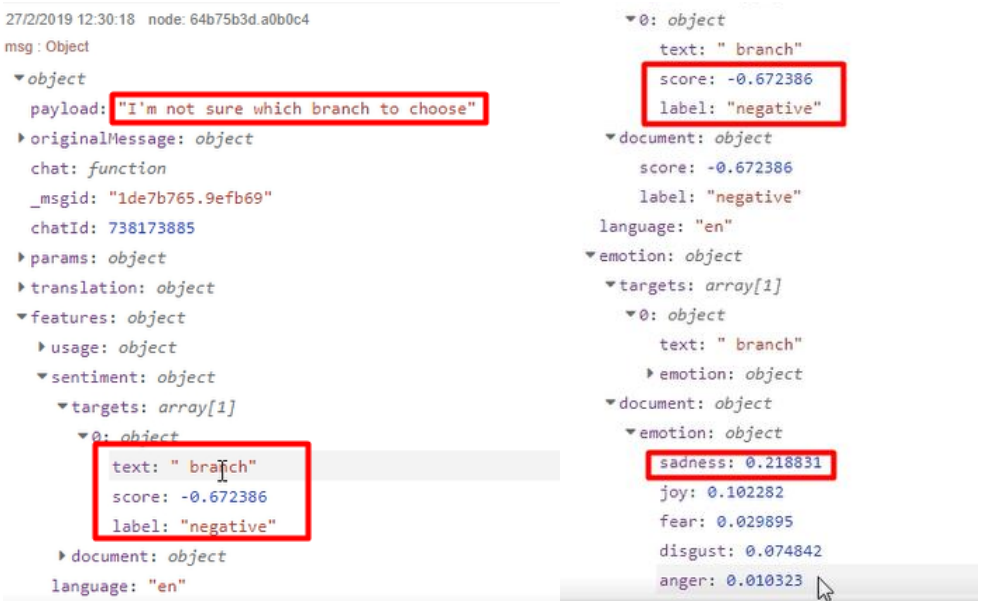




*Imagen 4-56. Pruebas -Tiempo real*

# Capítulo 5
# Conclusiones y líneas futuras

## 5.1 Conclusiones

Este proyecto desde su planteamiento inicial, pasando por su etapa de desarrollo hasta su finalización ha sufrido muchos cambios: empezando por el lenguaje de programación, herramientas a utilizar, plan de trabajo y evaluación e incluso el tema de conversación de este. Pero aun así ha mantenido su esencia de *bot* conversacional detector de sentimientos. Por lo tanto, ha sufrido cambios, pero siempre para conseguir un producto mejor. Es accesible ya que trabaja a través de una aplicación de mensajería social y gratuita que está disponible para cualquiera, en cualquier lugar del mundo y a cualquier hora. Por ende, potente.

Su cerebro nace de una aplicación tan impetuosa y robusta como es *IBM Watson*, con posibilidades de trabajo, diseño, desarrollo y despliegue que superan con creces la idea inicial del proyecto. ¿Por qué inventar la rueda si podemos mejorarla?

Durante el proceso de desarrollo ha sido fundamental el estudio de muchísimas herramientas que mejoraban lo ya desarrollado, pero sin tener que iniciar un nuevo proceso para ello. Se ha convertido en un proceso de investigación, aprendizaje e, incluso ilusión por seguir trabajando en este campo y creciendo de manera continua dentro de la I.A.

Al finalizar este TFG, una de las conclusiones principales es la importancia hoy en día y de cara al futuro de estos asistentes virtuales, pues la mayoría de las industrias, ya sean pequeñas o grandes empresas, se podrían quedar atrás si no ofrecen un servicio en la nube 24x7 y que sea lo más similar posible al comportamiento humano en cuanto al trato y la fluidez. Además, que el poder extraer de una conversación, ya sea escrita o hablada, el tono, los intereses y los gustos de una persona amplían infinitamente las posibilidades de crecer tanto de manera económica, industrial como personalmente.

Aunque este *chatbot* se ha desarrollado para un grupo específico de usuarios, puede servir como base para el diseño de cualquier bot conversacional que nos planteemos, debido a la flexibilidad que este sistema presenta.

El poder utilizar servicios que ya existen para crear nuevas ideas y que además sea extensible a cualquier entorno hace de este proyecto algo realmente útil en este campo de estudio.

## 5.2 Problemas encontrados

Uno de los mayores problemas encontrados durante el desarrollo de este proyecto ha sido el tiempo y las limitaciones del servicio de *IBM Watson*. En referencia al tiempo debido a que IBM ha decidido parar su servicio gratuito de *workspace* y migrarlo a otra herramienta. En su momento no se sabía si sería compatible con el trabajo que ya se había realizado o, por el contrario, se perdería todo lo implementado

hasta entonces. Por lo tanto, se ha tenido que trabajar de manera un poco atropellada durante la programación de este y grabar en todo momento lo que se iba haciendo y las conversaciones que se mantenían con el *bot*.

Con respecto a las limitaciones, al tratarse de un servicio gratuito, la cantidad de memoria que ofrecía para la instancia era realmente bajo y en cuanto pasaba un mes de uso su ciclo de vida se quedaba al 0%. Fue entonces cuando recordé que existía un registro para estudiantes que de manera también gratuita ofrecía más de el doble de memoria y un número de llamadas al servicio casi ilimitado. Esto me permitió continuar con el proyecto con un ciclo de vida del 100%.

Y cabe destacar también, que el número de nodos que permite el *workspace* del asistente es de 100, no era suficiente para crear una conversación totalmente completa ni óptima, por lo tanto, se pospuso algunas partes de la conversación como la orientación laboral a un desarrollo en líneas futuras.

Otro de los problemas que han aparecido durante la programación fue el idioma aceptado, no era posibles extraer las emociones cuando el idioma de entrada era el español. De ahí la solución de usar un traductor de español a inglés para poder beneficiarnos de los servicios de análisis de sentimientos y emociones sin tener que volver a cambiar el planteamiento del proyecto.

A grandes rasgos, los problemas no han sido muchos, pero el no tener conocimientos previos sobre la construcción de *bot* conversacionales ha ralentizado, en gran medida, el desarrollo de este.

## 5.3 Líneas futuras

Este sistema permite incluir una gran cantidad de mejoras, como podría ser:

- Ampliar su base de conocimientos, su conjunto de capacidades y diálogo sin las limitaciones del servicio gratuito, ampliando el flujo de la conversación y mejorando las ya existentes para asemejarlo aún más al comportamiento humano.
- Permitir que la entrada de texto sea en varios idiomas y el *bot* sea capaz de responder de manera coherente.
- Enriquecer la aplicación ofreciendo un servicio de orientación más específico y especializado. Y por supuesto, que permita que la entrada y salida no sea solo de texto sino también de voz.
- Y lo más importante y que también se ha estudiado durante el desarrollo, es la implementación de redes neuronales (*machine learning/Deep learning*). De manera que podamos crear modelos que sean capaces de predecir, mediante los datos que va obteniendo de otros usuarios, las necesidades y gustos de los alumnos que la utilizan. A través de un aprendizaje iterativo antes del despliegue, y que además después de este aprenda de los datos que recibe.

# Capítulo 6
# Summary and Conclusions

## 6.1 Summary

This project consists in the development of a tool capable of interacting with the students in a way that can guide them academically and, at the same time, be able to understand and analyze their emotions and feelings, in real time and in a natural way. To do this, try to find out by some questions whether or not they are happy in the degree and if they are clear about which specialization should choose. According to preferences will propose which is considerably more accurate according to the level of confidence it gets in each results iteration. Following this way is able to continuously analyze how the user is at an emotional level and depending on how he is then will redirect the conversation by one flow or another one.

Thus, starting from existing tools such as IBM Watson and, therefore, exploiting its full potential as a service in the cloud, it seeks to develop a project that provides improvements to existing services in the current scenario as they are exposed below:

- Integrate several services that make a conversational bot even more intelligent, such as the conversation assistant, the tone analyzer and the natural language comprehension service (NLU).

- Recognition of emotions and feelings when the user's input is in Spanish. This service is limited in IBM Watson to English and French. Therefore, it is extended with a translator so that this detection is even possible in our language.

- Do it accessible to each student, both by web service and by mobile application, using a known service as Telegram.

- Provide the application with a unified toolchain (Eclipse, GitHub, delivery and deployment in a continuous delivered with minimal handled tasks), being able to compile automatically, tests and controlling the quality of results.

Additionally, but less important right now, is the integration of machine learning that allows us to create and train models of I.A. and prepare and monitor data in an integrated environment such as the Watson Studio. In this way, the application will be able to learn throughout the conversation, in a supervised way, storing as a model the type of students and preferences to give a more accurate result learning from the previous ones.

## 6.2 Future research lines

This system allows to include several improvements, such as:

- Expanding its knowledge base, its skills sets, and dialogue without the limitations of free service, expanding the flow of the conversation and improving existing ones to make it even more similar to human behavior.

- Allowing the entry of text in several languages and the bot will be able to respond in a consistent way.

- Application enhancement by offering a more specific and specialized guidance service. And of course, that allows the input and output not only text but also voice.

- And the most important thing that has also been studied during development, is the implementation of neural networks (machine learning / Deep learning). So that we can create models that are able to predict, through the data that is obtained from other users, the needs and the student's preferences who use it. Through iterative learning before deployment, and also after this learn the data its receive.

## 6.3 Conclussions

This project from the beginning, also during the development stage to its completion has undergone many changes: starting with the programming language, tools to be used, work plan and evaluation and even the topic of chat of this. But still, it has maintained its essence of chatbot - Sentiment Analytics. Therefore, it has undergone some changes, but always to get a better product. It is accessible because it works through a social and free instant messenger application that is available to anyone, anywhere in the world and at any time. Therefore, a powerful tool.

His brain is born from a robust, effective and usable tool, like IBM Watson, with possibilities for design and deployment that far exceed the initial idea of the project. Why invent the wheel if we can improve it?

During the development process has been fundamental the study of many tools that improved the already developed, but without having to start a new process for it. It has become a process of research, learning and even hopes to continue working in this field and growing continuously within the I.A.

At the end of this TFG, one of the main conclusions is the importance nowadays and for the future of these virtual assistants, since most of the industries, be they small or big companies, could be left behind if they do not offer a service in the cloud 24x7 and being as similar as possible to human behavior in terms of treatment and fluency. In addition, the power to extract from a chat, whether written or spoken, the tone, interests, and hobbies of a person make the possibilities of growing economically, industrially and personally.

Although this chatbot has been developed for a specific users, it can use as a basis for any chatbot design that we consider, due to the flexibility that this system presents, being able to use services that already exist to create new ideas and that is also extensible to any environment makes this project really useful in this field of study.

# Capítulo 7
# Presupuesto

La duración de todas estas fases es estimada y puede sufrir variaciones, especialmente si hay cambios en diseño estructurales o alguna del servicio de terceros con cambios críticos que afecten a la gestión de recursos (memoria, almacenamiento, interrupción del servicio, etc.). El presupuesto será por horas trabajadas, estableciendo la hora de trabajo a 25€/hora.

| Primer bloque | | |
|---|---|---|
| **Tarea** | **Tiempo/Horas** | |
| **Análisis del problema** | 20 | 500€ |
| **Investigación de las herramientas disponibles** | 15 | 375€ |
| **Desarrollo del flujo de conversación** | 4 | 100€ |
| **Total** | **39 horas** | **975€** |

*Tabla 7-1. Presupuesto – Primer bloque*

| segundo bloque | | |
|---|---|---|
| **Tarea** | **Tiempo/Horas** | |
| **Desarrollo del asistente conversacional** | 30 | 750€ |
| **Creación de la conversación** | 80 | 2000€ |
| **integrando tone analyzer** | 10 | 250€ |
| **integrando natural language understanding** | 6 | 150€ |
| **integrando el traductor de idiomas** | 3 | 75€ |
| **prototipo de funcionamiento básico** | 15 | 375€ |
| **cadena de herramientas (entrega contínua y despliegue)** | 10 | 250€ |
| **conectando a telegram** | 5 | 125€ |
| **conectando a servicios externos (spotify)** | 2 | 50€ |
| **creando modelo machine/deep learning** | 2 | 50€ |
| **prototipo final** | 0 | NA |
| | 20 | 500€ |
| **TOTAL** | **183 horas** | **4575€** |

*Tabla 7-2. Presupuesto – Segundo bloque*

| tercer bloque | | |
|---|---|---|
| **Tarea** | **Tiempo/Horas** | |
| **testeo de la integración/ conversación** | 15 | 375€ |
| **evaluación** | 3 | 75€ |
| **análisis de resultados** | 10 | 250€ |
| **TOTAL** | **28 horas** | **700€** |

*Tabla 7-3. Presupuesto – Tercer bloque*

| nº Bloque | Total H | Total € |
|---|---|---|
| Primer bloque | 39 | 975 |
| Segundo bloque | 183 | 4575 |
| Tercer bloque | 28 | 700 |
| TOTAL | **250 horas** | **6250€** |

*Tabla 7-4. Presupuesto - Bloque total*

# Apéndice A. Intenciones

| INTENTS | DESCRIPCIÓN | Nº DE EJEMPLOS | EJEMPLO |
|---|---|---|---|
| **#AYUDA** | Ayuda sobre las funciones del bot | 31 | ¿Con qué me puedes ayudar? |
| **#CAMBIA_TEMA** | Cambiar a un tema diferente | 21 | Cambiemos de tema. |
| **#CH_PREGUNTAESTADO** | Pregunta cómo se encuentra | 15 | ¿Cómo estás? |
| **#DESPEDIDA** | Fin de la conversación | 28 | Hablamos luego. |
| **#DETECTASENTIMIENTOS** | Detecta cómo te sientes | 50 | me encuentro frustado |
| **#DUDA** | Usuario tiene dudas | 36 | No entiendo nada |
| **#FEEDBACK_NEGATIVO** | Comentarios desfavorables | 31 | Creo que te equivocas |
| **#FEEDBACK_POSITIVO** | Sentimientos de gratitud o positivo | 25 | Gracias infinitas por tu ayuda |
| **#INICIO** | Reinicar | 2 | empecemos |
| **#ORIENTAESTUDIOS** | Orientador Académico | 27 | no estoy segura en que especializarme |
| **#PROPONER_ALGO** | Proponer hacer algo | 29 | ¿Me propones algo divertido? |
| **#RESPUESTA_AFIRMATIVA** | Respuesta afirmativa | 33 | De acuerdo. |
| **#RESPUESTA_INCOHERENTE** | Respuesta incoherente | 26 | No entiendo qué quieres decir |
| **#RESPUESTA_NEGATIVA** | Respuesta negativa | 32 | No quiero |
| **#SALUDOS** | Saludos | 20 | Hola |
| **#SOBRE_CHATBOT** | Sobre el bot | 26 | ¿Quién eres? |

*Tabla 7-5. Apéndice A - Intenciones*

# Apéndice B. Entidades

| ENTITIES | VALORES EJEMPLO | SINÓNIMOS |
|---|---|---|
| **@CH_ACTIVIDAD** | musica, juego | radio, spotify |
| **@CH_GENERO** | clásica, rap, pop, jazz, rock | ---- |
| **@CH_TIPO DE JUEGO** | adivinador, programa | Akinator, programar |
| **@IDIOMA** | Español, inglés, francés | English, spanish |
| **@OA_COMPETENCIAS** | bases de datos, redes, programación, diseño | Datos, minería |
| **@OA_ORIENTADOR** | educativa, laboral | Académica, estudios |
| **@OE_GUSTOSOTROS** | Humanidades, Artísticos, Actividades agrarias | Mecánica, coches, bailar |
| **@OE_RAMA** | Computación, Ingeniería de computadores, Indefinida | Software, IA |
| **@SOBREBOTTIPOS** | tipo, nombre, interes | Intereses, estilo |
| **@TIPOSENTIMIENTOS** | miedo, tristeza, alegría, ira, esperanza | Susto, euforia |

*Tabla 7-6. Apéndice B - Entidades*

# Apéndice C. Dialog

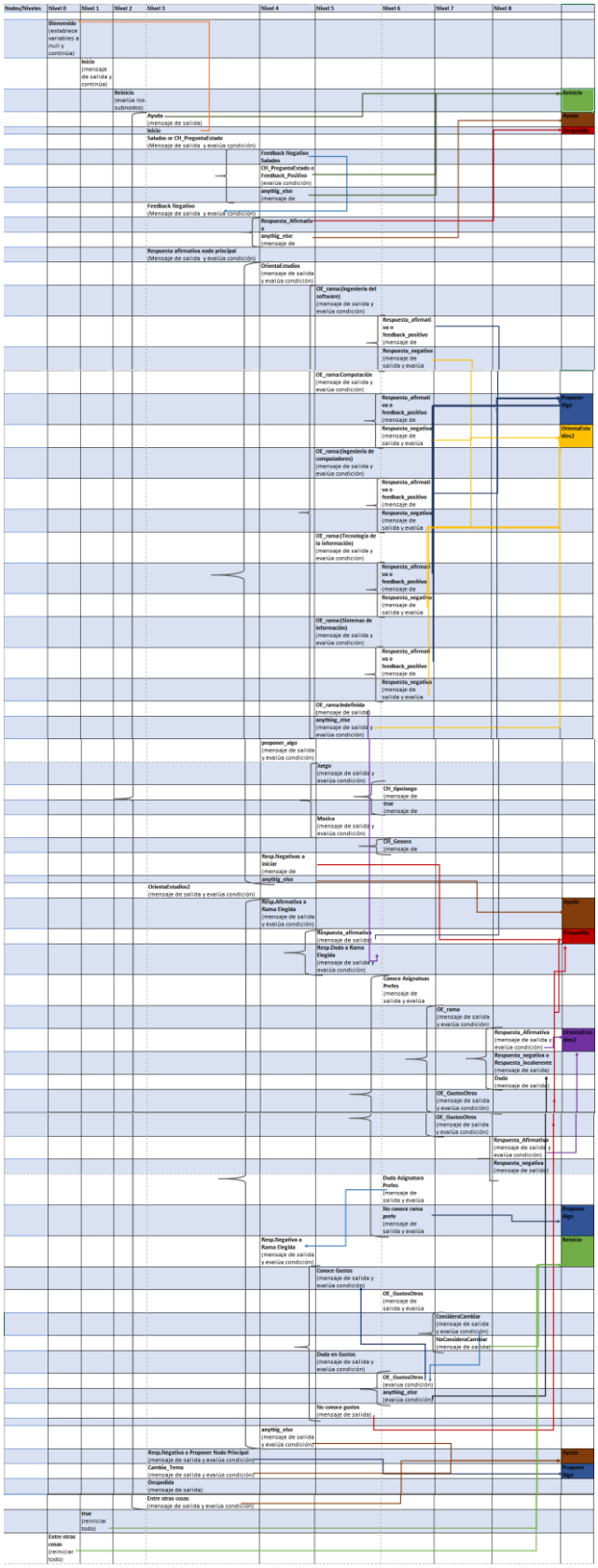


*Tabla 7-7. Apéndice C - Diálogo*

# Capítulo 8
# Bibliografía


@IBM, O. S. (s.f.). Obtenido de https://console.bluemix.net/docs/services/assistant/getting-started.html#gettingstarted

@IBM, O. S. (s.f.). Obtenido de https://developer.ibm.com/courses-center/play-food-coach-chatbot/

@IBM, O. S. (s.f.). Obtenido de https://www.ibm.com/blogs/bluemix/2017/06/building-chatbots-tips-tricks/

@IBM, O. S. (s.f.). Obtenido de https://cloud.ibm.com/docs/services/assistant?topic=assistant-expresiones-para-acceder-a-objetos#expresiones-para-acceder-a-objetos

@IBM, O. S. (2019). *Virtual Agent Watson*. Obtenido de https://www.ibm.com/watson/developercloud/doc/virtual-agent/toc-es.html

@IBM, O. S. (s.f.). *Cloud Foundry Documentation*. Obtenido de Getting Started with Cloud Foundry: https://docs.cloudfoundry.org/

@IBM, O. S. (s.f.). *Create a cognitive banking chatbot*. Obtenido de https://developer.ibm.com/patterns/create-cognitive-banking-chatbot/

@IBM, O. S. (s.f.). *GitHub*. Obtenido de watson-developer-cloud/node-sdk: https://github.com/watson-developer-cloud/node-sdk

@IBM, O. S. (s.f.). *IBM Cloud*. Obtenido de Tone Analyzer: https://cloud.ibm.com/apidocs/tone-analyzer

@IBM, O. S. (s.f.). *IBM Cloud*. Obtenido de NLU: https://cloud.ibm.com/apidocs/natural-language-understanding

@IBM, O. S. (s.f.). *IBM Cloud*. Obtenido de Language Translator: https://console.bluemix.net/docs/services/language-translator/index.html

@IBM, O. S. (s.f.). *Play with the Food Coach chatbot*. Obtenido de https://developer.ibm.com/courses-center/play-food-coach-chatbot/

@IBM, O. S. (s.f.). *Tutorial: Building a cognitive car dashboard dialog*. Obtenido de https://console.bluemix.net/docs/services/assistant-icp/tutorial-car-dashboard.html

Ennis, S. (s.f.). Obtenido de https://github.com/samcennis/Watson-Conversation-ML-Demo

Foundation., J. (s.f.). *Node-RED*. Obtenido de Documentation: https://nodered.org/docs/

Hurwitz, J., Kirsch, D., & edition, I. l. (2018). *Machine Learninf for dummies.* John Wiley & Sons.

Kristian Hammond, P. (2015). *Practical Artificial Intelligence for DUMMIES A Wiley Brand.* John Wiley $ Sons.

mariCh. (s.f.). *BLOG DE MARICHELO*. Obtenido de Modelo incremental: http://marich.blogspot.es/1459223366/modelo-incremental/

Nepal, M. (s.f.). *FreschchatBlog*. Obtenido de Siri, Alexa, or chatbots—what is most useful for your business?: https://www.freshworks.com/live-chat-software/the-best-chatbots-vs-siri-alexa-showdown-blog/

Slack. (s.f.). *Slack api*. Obtenido de Build: https://api.slack.com/

Wallace, R. (s.f.). *pandorabots*. Obtenido de http://www.pandorabots.com/pandora/talk?botid=f5d922d97e345aa1

Weizenbaum, J. (s.f.). *xataka*. Obtenido de Así era ELIZA, el primer bot conversacional de la historia: https://www.xataka.com/historia-tecnologica/asi-era-eliza-el-primer-bot-conversacional-de-la-historia


Narrative Science Edition – Practical Artificial Intelligence for Dummies A Wiley Brand *(Kristian Hammond, 2015)*

Machine Learning for dummies a Wiley Brand (Hurwitz, Kirsch, & edition, 2018)

Watson Virtual Agent Documentación *(@IBM, Virtual Agent Watson, 2019)*