

# CHATBOT

**Nombre Estudiante:**

Alberto Silla del Ser  
05.629 TFG - Inteligencia artificial

**Nombre Consultor/a:**

Dr. David Isern Alarcón

**Nombre Profesor/a responsable de la asignatura:**

Dr. Carles Ventura Royo

**Data de entrega**

24/06/2019



Esta obra está sujeta a una licencia de [Reconeixement-NoComercial-SenseObraDerivada 3.0 Espanya de Creative Commons](https://creativecommons.org/licenses/by-nc-nd/3.0/es/)

## FICHA DEL TRABAJO FINAL

<b>Título del trabajo:</b>	<i>ChatBot</i>
<b>Nombre del autor:</b>	<i>Alberto Silla del Ser</i>
<b>Nombre del consultor/a:</b>	<i>Dr. David Isern Alarcón</i>
<b>Nombre del PRA:</b>	<i>Dr. Carles Ventura Royo</i>
<b>Data de entrega (mm/aaaa):</b>	<i>06/2019</i>
<b>Titulación o programa:</b>	<i>Grau en Enginyeria Informàtica</i>
<b>Área del Trabajo Final:</b>	<i>05.629 TFG - Inteligencia Artificial</i>
<b>Idioma del trabajo:</b>	<i>Castellano</i>
<b>Palabras clave</b>	<i>BBDD, LUIS i C#.</i>
<p><b>Resum del Treball (màxim 250 paraules):</b> <i>Amb la finalitat, context d'aplicació, metodologia, resultats i conclusions del treball</i></p>	
<p>La inteligencia artificial (IA) permite, a partir de algoritmos, crear recursos que simulen la capacidad del ser humano. La creación de un chatbot no es más que un sistema que piensa racionalmente dentro de lo debido, pero la pregunta es, ¿Es posible crear un sistema racional escalable?</p> <p>Pues bien, comprobamos en este proyecto que el aprendizaje profundo/automático nos permite a través de una base de datos dinámica, crear un código capaz de gestionar un sistema que pueda pasar de 10 a 500 procesos sin sufrir ninguna alteración de peso en este.</p> <p>LUIS es nuestro aprendizaje profundo/automático y C# es el lenguaje que nos permite crear las funciones para darle un recorrido lógico a las tablas dinámicas de nuestra base de datos. A partir de todos estos conceptos el resultado refleja un recurso estable y capaz de adquirir conocimiento en grandes cantidades, destinado a ofrecer la información que almacena de manera lógica.</p> <p>Al final, nuestro sistema escalable (chatbot) evita errores de gestión, acelera los procesos, aumenta la eficacia del beneficiario y permite la posibilidad de crecer acorde con su funcionalidad.</p>	

**Abstract (in English, 250 words or less):**

Artificial intelligence (AI) allows, from algorithms, to create resources that simulate the capacity of the human being. The creation of a chatbot is nothing more than a system that thinks rationally within what is due, but the question is, is it possible to create a rational scalable system?

Well, we verify in this project that the deep / automatic learning allows us through a dynamic database, to create a code capable of managing a system that can go from 10 to 500 processes without undergoing any alteration of weight in it.

LUIS is our deep / automatic learning and C # is the language that allows us to create the functions to give a logical route to the dynamic tables of our database. Based on all these concepts, the result reflects a stable resource capable of acquiring knowledge in large quantities, designed to offer the information it stores in a logical manner.

In the end, our scalable system (chatbot) avoids management errors, accelerates processes, increases beneficiary efficiency and allows the possibility of growing according to its functionality.

# Índice

1. Introducción.....	1
1.1 Contexto y justificación del Trabajo .....	1
1.2 Objetivos del Trabajo.....	2
1.3 Núcleo y método de seguimiento .....	2
1.4 Planificación del Trabajo .....	4
1.5 Productos obtenidos.....	5
1.6 Breve explicación de los otros capítulos de la memoria .....	5
2. Desarrollo del Trabajo .....	6
2.1 Base de datos .....	12
2.2 Introducción al LUIS/Machine Learning.....	16
2.3 Código C# Visual Studio.....	22
2.4 Microsoft Bot Framework Emulator .....	29
2.5 Actividades no previstas .....	42
2.6 Objetivos principales del proyecto.....	43
3. Conclusiones.....	44
4. Glosario.....	46
5. Bibliografía .....	47
5. Apéndice .....	48

# 1. Introducción

## 1.1 Contexto y justificación del Trabajo

De acuerdo con el concepto chatbot, partimos de una base muy extensa dentro de la inteligencia artificial. El chatbot no es más que una herramienta de automatización en el campo del RPA (Robotic Process Automation), con lo cual el abanico de salida es amplio, sin embargo la *rama* del chatbot está aún en desarrollo cuando se trata de darle un fin empresarial interno.

Con este trabajo quiero obtener una gestión automatizada con la finalidad de facilitar, agilizar y disminuir errores en cualquier proceso de los trabajadores (dentro de los datos que abarque nuestro chatbot).

La idea principal dejando a un lado el desarrollo técnico, es crear una base de datos capaz de abarcar los convenios posibles que pueda tener un banco con los diferentes clientes que pueda tener. Esta idea es muy importante ya que se puede extrapolar a cualquier tipo de situación laboral, y sobretodo empresarial.

## 1.2 Objetivos del Trabajo

La meta primordial es lograr escalabilidad. Crear un proyecto escalable es crear una solución óptima para el fin de un chatbot, de lo contrario lo único que obtendremos es una base de datos sin posibilidades, inmóvil y sin rango de expansión.

Un concepto que va ligado a escalabilidad es el dinamismo, es necesario que el código esté enfocado a una BBDD dinámica, de esta manera las relaciones nos permitirán siempre gestionar cambios.

Estos dos objetivos nos advierten de crear un código más escueto con una BBDD generosa.

## 1.3 Núcleo y método de seguimiento

El método a seguir es partir de los modelos/plantillas de chatbot que los desarrolladores nos ofrecen de manera pública i gratuita, ver su funcionamiento y analizar el enfoque técnico que le daremos al programa. De acuerdo con esto, elaboraremos un producto adaptado al concepto de escalabilidad, ya que hay proyectos en internet como este:

[8]<https://github.com/Gr8z/ChatBotProject>

Donde encontraremos un código C# sencillo con muy pocas funciones que nos permitirá probar las siguientes tareas:

- Respuestas a diferentes preguntas y tiene la habilidad de llevar una conversación normal.
- Miles de plantillas en su repositorio para amplio conocimiento genérico.

- Tiene un conocimiento de actualidad elevado.
- Dotado de sentido del humor.
- Interfaz de usuario agradable.
- Tiene reconocimiento de voz (Microsoft.Speech).

Como podemos ver la totalidad de sus funciones son un trabajo tedioso y costoso en cuanto a volumen, ya que está enfocado a la diversidad ante la que el chatbot se va a enfrentar, el único punto interesante aquí es la introducción del reconocimiento de voz en el chat.

Después de ver este pequeño proyecto que circula por la red, de entre tantos, y buscando a conciencia llegué a la conclusión de que la estrategia para tener una demanda incuestionable y un éxito asegurado en este apartado se basaba en la capacidad de evolucionar a gran escala del chat.

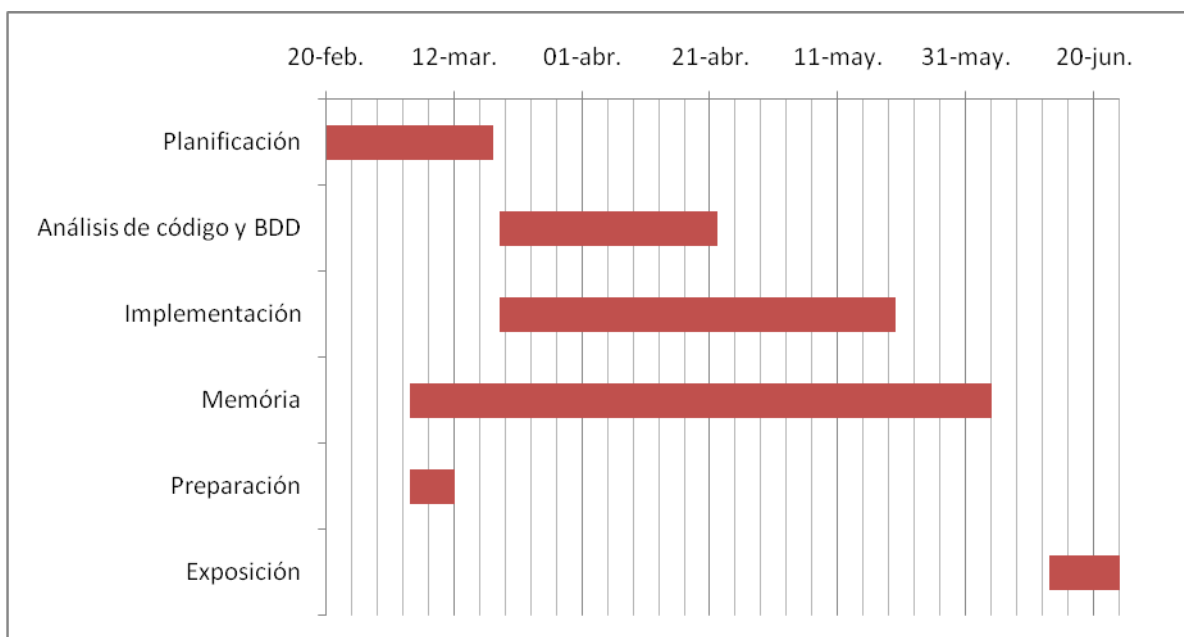


## 1.4 Planificación del Trabajo

De manera gratuita los recursos esenciales van a ser;

- BBDD Microsoft Azure, SQL (temporal)
- Visual Studio en código C#
- LUIS BOT
- Microsoft BOT Framework Emulator

Diagrama de Gantt, líneas temporales del proyecto



## 1.5 Productos obtenidos

Microsoft BOT Framework Emulator

Microsoft Visual Studio

LUIS BOT web

Microsoft Azure (temporal)

## 1.6 Breve explicación de los otros capítulos de la memoria

A grandes rasgos la memoria se basa en 2 capítulos principales y varios puntos finales.

El primer capítulo del cual forma parte este pequeño punto se basa en la introducción al proyecto, donde explicamos el motivo, la finalidad, tiempos y los componentes que se desglosarán en el siguiente capítulo.

En el segundo capítulo estaremos enfatizando en el desarrollo del proyecto, es decir, en la implementación del código, la construcción/estructura de la base de datos, la creación del LUIS y la correcta explicación de todo el funcionamiento en conjunto con demostraciones a partir de un emulador. Así mismo, trataremos las fisuras y las soluciones que se requieran.

Para finalizar, los puntos finales son, el mayor grosor, información acerca de puntos clave del proyecto, ya que el glosario, la bibliografía y el apéndice tratan de explicarnos tecnicismos y nos concretan fuentes. Las conclusiones forman parte de este último apartado y se analiza de manera exacta el propósito.

## 2. Desarrollo del Trabajo

Siguiendo lo establecido anteriormente, es necesario crear un contenido para el desarrollo del proyecto, a continuación explicaré en que se basa la BBDD;

De manera introductoria, ICO (Instituto de Crédito Oficial), es una opción para conseguir financiación en la situación de dificultad de acceso por parte de los autónomos y pymes al crédito bancario. Las líneas ICO 2019 se pueden tramitar en las principales entidades financieras.

Hay diferentes opciones;

- Principales líneas ICO para autónomos, pymes y emprendedores:

- Línea ICO Empresas y Emprendedores 2019
- Línea ICO Garantía SGR 2019
- Línea ICO Crédito Comercial 2019

Línea ICO Canal Internacional 2019

- Línea ICO Internacional 2019
- Línea ICO Exportadores 2019

Enlace pertinente [10]: <https://infoautonomos.eleconomista.es/financiacion-autonomos-empresas/lineas-del-instituto-de-credito-oficial-ico/>

En nuestro proyecto encontraremos información para los empleados bancarios acerca de; amortizaciones, cancelaciones y entregas de préstamos.

En esta entrega **adjunto un documento Excel** con 28 ejemplos de productos para posibles convenios o sub-convenios, disponen de un ID, la fecha en la que se formaliza, la autorización ICO/BCE y el del tipo de sub-convenio.

De manera aleatoria y por necesidad me decidí a escoger una entidad bancaria, en este caso no es de Cataluña y sin ningún tipo de representación específica he creado toda la base de datos y el chat como una entidad de Cajamar.

**Cajamar** Caja Rural es la entidad financiera resultante de la fusión de cajas rurales de Andalucía Oriental, Murcia, Madrid, Castilla y León, Baleares, Comunidad Valenciana, Canarias y Caixa Albalat.

He querido darle más realidad al chat y no inventarme un “nombre”, ya que con ello tendría que crear un logo exclusivo, entre otras cosas, y no se trata de crear una entidad bancaria sino crear una solución interna para este sector y posiblemente cualquier empresa que esté necesitada de una herramienta automatizada como esta.

La decisión de crear una base de datos lógica y capaz en este sector bancario me ha llevado a tener que crear mucho y con sentido los datos que usamos para consultar en este chat;

- Recibos
- Subvenciones
- Productos
- Ayudas
- Explotaciones
- Anticipos
- Cancelaciones
- Amortizaciones
- Entregas parciales y totales

(He preferido un desglose visual más que una lista porque son conceptos “importantes” en la base de datos)

El pilar de la base de datos se basa en 5 comunidades autónomas y 2 SubConvenios más, ya que cada comunidad no deja de ser un convenio:

- Fomento
  
- Generalitat Valenciana
  
- Junta de Castilla y León
  
- Región de Murcia
  
- Generalitat de Catalunya
  
- Junta de Andalucía
  
- MAGRAMA

A partir del tipo de convenio el chat esta guiado con respuestas para que el usuario sepa que tiene que contestar. Por ejemplo si accedemos al convenio Fomento tendremos una serie de opciones las cuales no disponen los demás, ya sea el año del convenio o de si dispone de una subvención o ayuda AEDE.

De manera genérica he introducido un tipo parcial o total de amortización en cada convenio para establecer algo en común, y de manera más profunda algunos productos de los cuales solo disponen algunos convenios.

Para acabar he introducido preguntas de información para la propia empresa, si ha sido útil el “recorrido” para el usuario, y también unas respuestas finales que concluyen la consulta del usuario con una meta informativa, veáse el siguiente ejemplo:

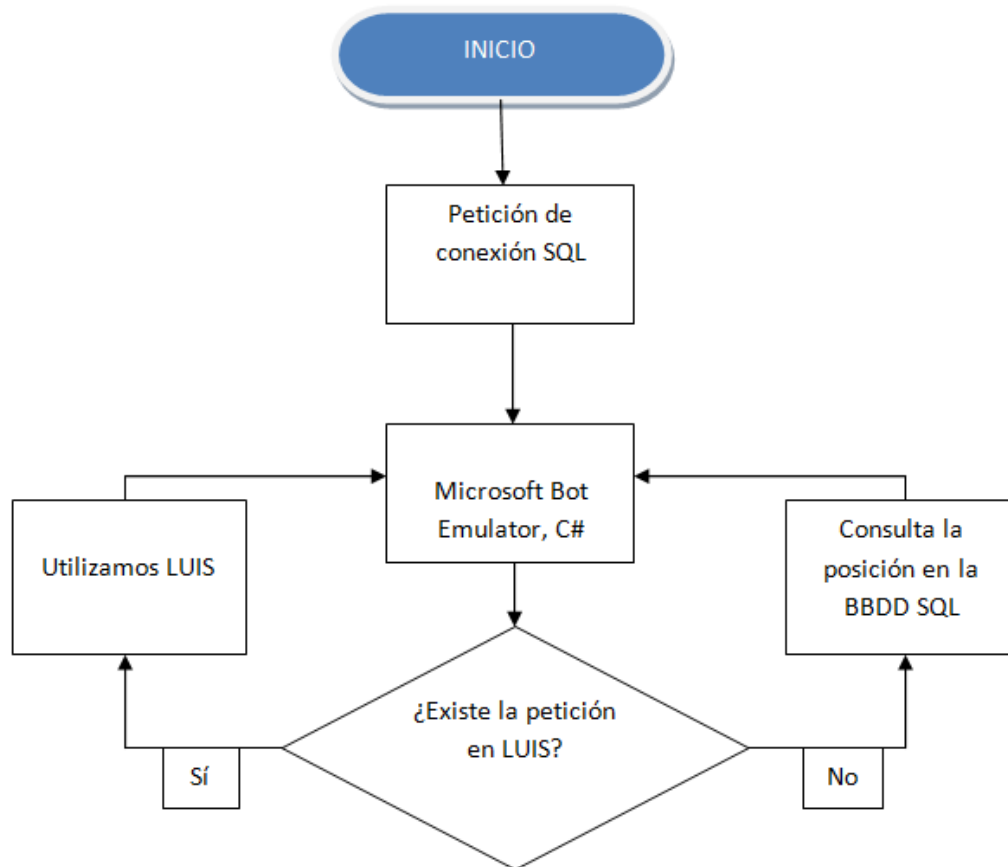
Ha seleccionado el convenio 1998-2001 o anteriores. Indicarles que para realizar entrega parcial (el importe de la entrega debe ser igual o superior a 1.202 euros, y a reparto proporcional) de la operación deberán reportarnos [ENLACE CONVENIOS], con el motivo 00068 ENTREGAS A PRESTAMOS-CONVENIOS.

Las entregas tienen que coincidir con la fecha de liquidación (indicar fecha de aplicación en el enlace). En caso de existir recibos pendientes deberán de regularizarlos en oficina antes de reportarnos el enlace.

Todos estos conceptos bancarios van a estar involucrados en la base de datos, ya que para darle forma a algo tiene que tener un sentido, no podía crear un proyecto así sin tener un concepto elaborado y lógico.

Para este cometido, los tecnicismos, las respuestas en los textos, etc, me ha ayudado mucho el conocimiento de un amigo de la infancia que trabaja actualmente y desde hace años en La Caixa, su nombre es **Daniel Talavera Pérez** ([danieltalaveraperez@gmail.com](mailto:danieltalaveraperez@gmail.com)) y está abierto a cualquier consulta referente a este trabajo si fuese necesario.

He visto más adecuado crear un diagrama de flujo que un diagrama de arquitectura ya que puedo comprender todos los conceptos de este último mostrando el recorrido continuo que tiene el chatbot.



Como podemos ver en el diagrama no he querido implementar el terminador fin ya que el chat esta continuamente funcionando y reiniciando su registro para poder establecer nuevos caminos.



## 2.1 Base de datos

La lógica principal del chat esta creada en las tablas dinámicas SQL de la DB (data base/base de datos) y su recorrido entre dos de ellas, Preguntas y Respuestas. Las tablas son las siguientes y vienen **adjuntas en forma de BACKUP (.txt)** en esta entrega;

- USUARIOS
- MENSAJES BIENVENIDA
- MENSAJES BASICOS
- MENSAJES NONE
- LUIS\_PREGUNTAS
- LUIS\_RESPUESTAS

La tabla usuarios la necesitamos para identificar y poder usar de esta manera el usuario actual, ya que si no controlásemos esto la sesión del chatbot sería siempre la misma y el proceso de preguntas y respuestas que tengamos se basaría en una sesión ya iniciada.

De manera global trabajaremos a partir de convenios y sub-convenios, con lo cual siempre identificaremos los procesos a partir de números de expedientes.

Los mensajes básicos y de bienvenida sirven para crear varias opciones RANDOM (aleatorias) de las cuales responder al usuario, controlado a partir del código C# y gestionados en LUIS.

Los mensajes NONE nos sirven para dar respuesta cuando LUIS no detecta ninguna cadena de texto válida, entonces creamos una opción capaz de guiar al usuario para darnos una orden funcional.

A continuación vamos a ver las dos tablas más importantes (LUIS PREGUNTAS, LUIS RESPUESTAS) las veremos gráficamente para apreciar la lógica del proyecto.

## DB - LUIS Preguntas

id_LUIS_pregu...	identificador	nombre	accion_anterior	accion_siguien...	pregunta
1	1	entrega total o ...	4000	4;2	¿Supone una entrega parcial o total del ...
2	2	entrega t/p No	1	0	Si la baja de la finca no supone una can...
3	3	entrega total o ...	4000	4;2	¿Supone una entrega parcial o total del ...
4	4	Convenio	0	100;200;300;400...	Escribe o indica el convenio o producto ...
5	100	Fomento/MOPU	4	101;102	Has seleccionado el convenio de Fome...
6	200	Generalitat Vale...	4	201;202	Has seleccionado el convenio de Gener...
7	300	Junta Castilla y ...	4	301;302	Has seleccionado el convenio de Junta ...
8	400	Región de Mur...	4	401;402	Has seleccionado el convenio de Regió...
9	500	Generalitat Cat...	1100	504;505	Has seleccionado el convenio de Gener...
10	600	Junta de Andal...	4	601;602;612	Has seleccionado el convenio de Junta ...
11	700	MAGRAMA	4	701;702	Has seleccionado el convenio de MAGR...
12	101	Fomento 5 año...	100	103;104	Por favor, indica si la amortización es to...
13	102	Fomento 5 año...	100	109;110	¿Has recibido subvención o ayuda AEDE?
14	103	Fomento total	101	0	Para realizar cancelación de la operació...
15	104	Fomento parcial	101	105;106;107;108	Por favor indica al sub-convenio que p...
16	105	Fomento parcia...	104	0	Has seleccionado el convenio 1998-200...
17	106	Fomento parcia...	104	0	Has seleccionado el convenio 2002-200...
18	107	Fomento parcia...	104	0	Has seleccionado el convenio 2005-200...

## DB - LUIS Respuestas

id_LUIS_respue...	info	LUIS	precedente	siguiente	clickable
1	Que operación quiere realizar?	Cancelación/Amortizaci...	0	4	True
4	Cual es el convenio?	Fomento/MOPU	0	100	False
5	Cual es el convenio?	Generalitat Valenciana	0	200	False
6	Cual es el convenio?	Junta de Castilla y León	0	300	False
7	Cual es el convenio?	Región de Murcia	0	400	False
8	Cual es el convenio?	Generalitat de Catalunya	0	500	False
10	Cual es el convenio?	MAGRAMA	0	700	False
11	Fomento, han pasado más de 5 añ...	Sí	100	101	False
12	Fomento, han pasado más de 5 añ...	No	100	102	False
13	Fomento, tipo?	Total	101	103	True
14	Fomento, tipo?	Parcial	101	104	True
15	Fomento, si 5 años, parcial, año del...	Convenio 1998 a 2001 o ...	104	105	True
16	Fomento, si 5 años, parcial, año del...	Convenio 2002 a 2005	104	106	True
17	Fomento, si 5 años, parcial, año del...	Convenio 2005 a 2008	104	107	True
18	Fomento, si 5 años, parcial, año del...	Convenio 2009 a 2012	104	108	True
19	Fomento, no 5 años, subvención o ...	Sí	102	109	False
20	Fomento, no 5 años, subvención o ...	No	102	110	False
21	Fomento, no 5 años, si ayuda, ha d...	Sí	109	111	False

Como podemos observar he creado la lógica en la base de datos, el *workflow* es el siguiente:

Cuando llega un mensaje nuevo primero se comprueba en USUARIOS, si el usuario no existe se le asigna el valor de inicio con **precedente** = 0, acto seguido se consulta al LUIS Respuestas directamente el mensaje que ha escrito el usuario, ya que comprueba si se ha escrito la palabra **exacta** que hay en la columna LUIS (los intentos de LUIS).

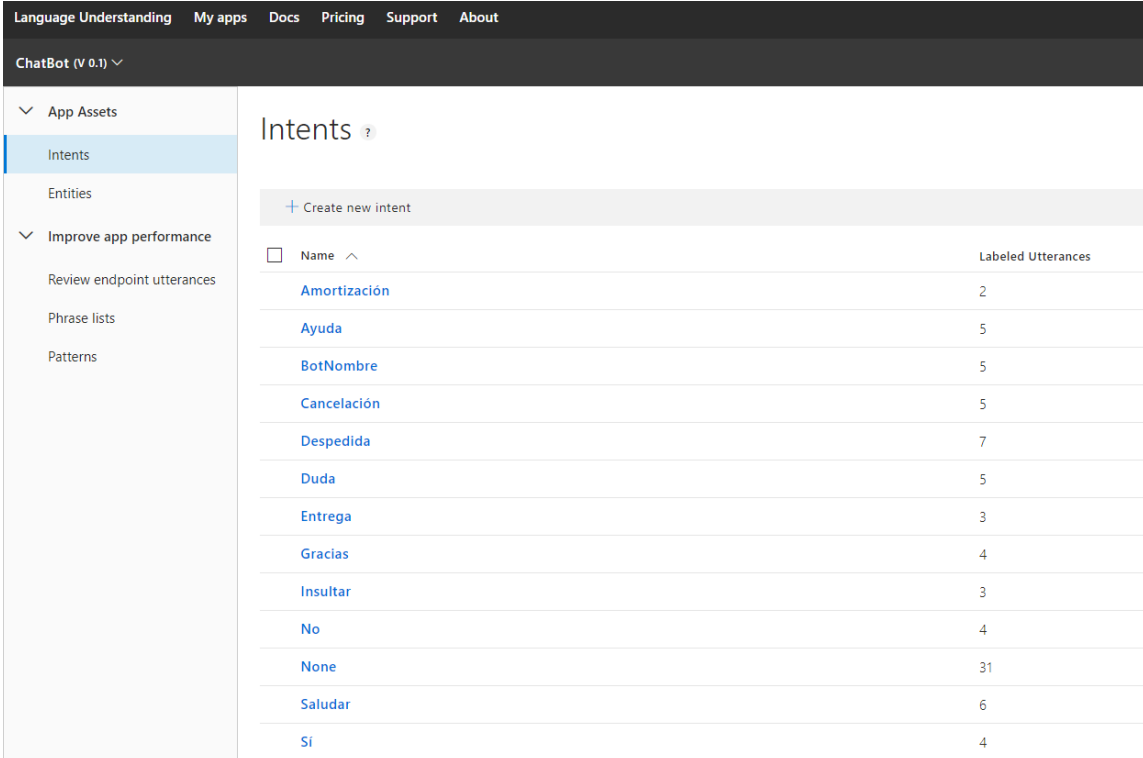
Si encuentra una fila exacta pasa al **LUIS ESPECIFICO**, pero si no encuentra ninguna exacta llama al LUIS donde retorna un **Intento/INTENT**, el mas probable, y después comprueba **Mensajes Basicos**.

Si encuentra algo en Mensajes Basicos entonces devuelve el mensaje que toca, en el caso que no encuentre nada en Mensajes Basicos pasa a LUIS ESPECIFICO. En el caso de no encontrar nada en esta etapa devuelve Mensajes NONE, pero si encuentra algo en LUIS Respuestas entonces viaja a **LUIS Preguntas** donde el identificador sea el precedente y devuelve la pregunta. A partir de aquí se actualiza el nuevo precedente al siguiente del LUIS Preguntas.

## 2.2 Introducción al LUIS/Machine Learning

El Machine Learning es un método de análisis de datos que automatiza la construcción de modelos analíticos, en este caso LUIS.

Url [7]: <https://www.luis.ai/>



The screenshot shows the LUIS web interface. At the top, there is a navigation bar with links for 'Language Understanding', 'My apps', 'Docs', 'Pricing', 'Support', and 'About'. Below this, the current application is identified as 'ChatBot (v 0.1)'. A left-hand sidebar contains a menu with options: 'App Assets', 'Intents' (which is currently selected), 'Entities', 'Improve app performance', 'Review endpoint utterances', 'Phrase lists', and 'Patterns'. The main content area is titled 'Intents' and features a '+ Create new intent' button. Below the button is a table listing existing intents and their corresponding labeled utterances.

Name	Labeled Utterances
Amortización	2
Ayuda	5
BotNombre	5
Cancelación	5
Despedida	7
Duda	5
Entrega	3
Gracias	4
Insultar	3
No	4
None	31
Saludar	6
Si	4

En esta primera imagen podemos ver la interfaz del LUIS, el servicio que usaremos para nuestro proyecto y se compone de dos partes básicas;

La primera parte, Intents/Intentos nos permite crear como bien dice la palabra una sección donde poder “entrenar” a la aplicación para que reconozca la opción que estamos creando. Veamos de que se trata, por ejemplo en el intento de Amortización;

## Amortización

Type about 5 examples of what a user might say and hit Enter

---

Utterance

---

quiero realizar una **Amortizar**

---

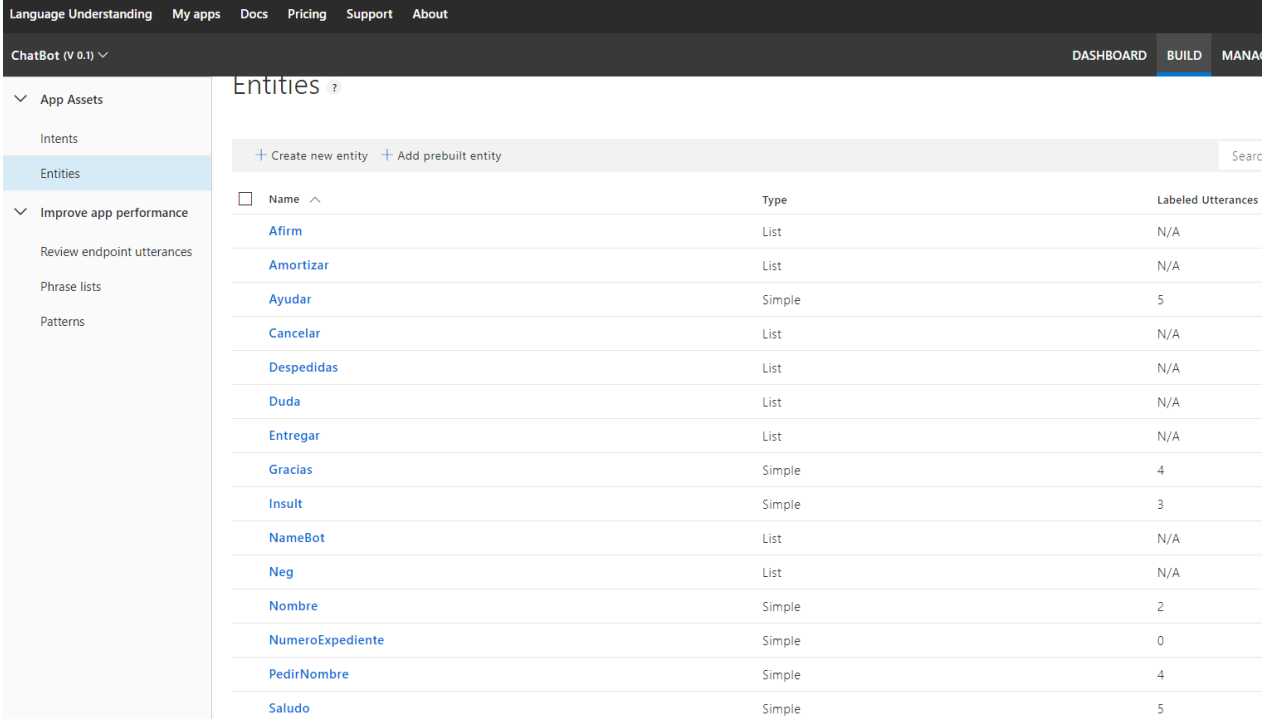
**Amortizar**

---

Dentro del Intento Amortización queremos que se reconozca a partir de las Utterances/Declaraciones, el intento en cuestión, en este caso amortización.

Las **Utterances/Declaraciones** no son más que frases o sentencias que nosotros construimos con diferentes variables.

Segunda parte, veamos cómo se construye una entidad:



<input type="checkbox"/>	Name ^	Type	Labeled Utterances
	<a href="#">Afirm</a>	List	N/A
	<a href="#">Amortizar</a>	List	N/A
	<a href="#">Ayudar</a>	Simple	5
	<a href="#">Cancelar</a>	List	N/A
	<a href="#">Despedidas</a>	List	N/A
	<a href="#">Duda</a>	List	N/A
	<a href="#">Entregar</a>	List	N/A
	<a href="#">Gracias</a>	Simple	4
	<a href="#">Insult</a>	Simple	3
	<a href="#">NameBot</a>	List	N/A
	<a href="#">Neg</a>	List	N/A
	<a href="#">Nombre</a>	Simple	2
	<a href="#">NumeroExpediente</a>	Simple	0
	<a href="#">PedirNombre</a>	Simple	4
	<a href="#">Saludo</a>	Simple	5

Vemos que no hay mucha diferencia con los intentos, al contrario que estos las entidades se definen por un conjunto de palabras, ya pueden ser listas de diferentes palabras sueltas o simplemente variaciones de una misma palabra escrita de diferentes formas (a prueba de fallo del usuario).

Las entidades podemos agruparlas en diferentes tipos:

Lo más práctico y común cuando queremos crear una entidad que nos controle una lista muy específica y amplia, es crear una de tipo **Simple**, esto quiere decir que iremos al apartado *phrase lists* y crearemos nuestra propia lista de la siguiente manera;

### Edit Phrase list

Name (Required)

Value (Required)

These values are interchangeable

[Done](#) [Cancel](#)

Por otro lado podemos crear una entidad de tipo *List*, la cual se basa en sinónimos o variaciones basadas en el error del usuario, como por ejemplo en la entidad “Amortizar”;

## Amortizar

Entity type: List

### Values

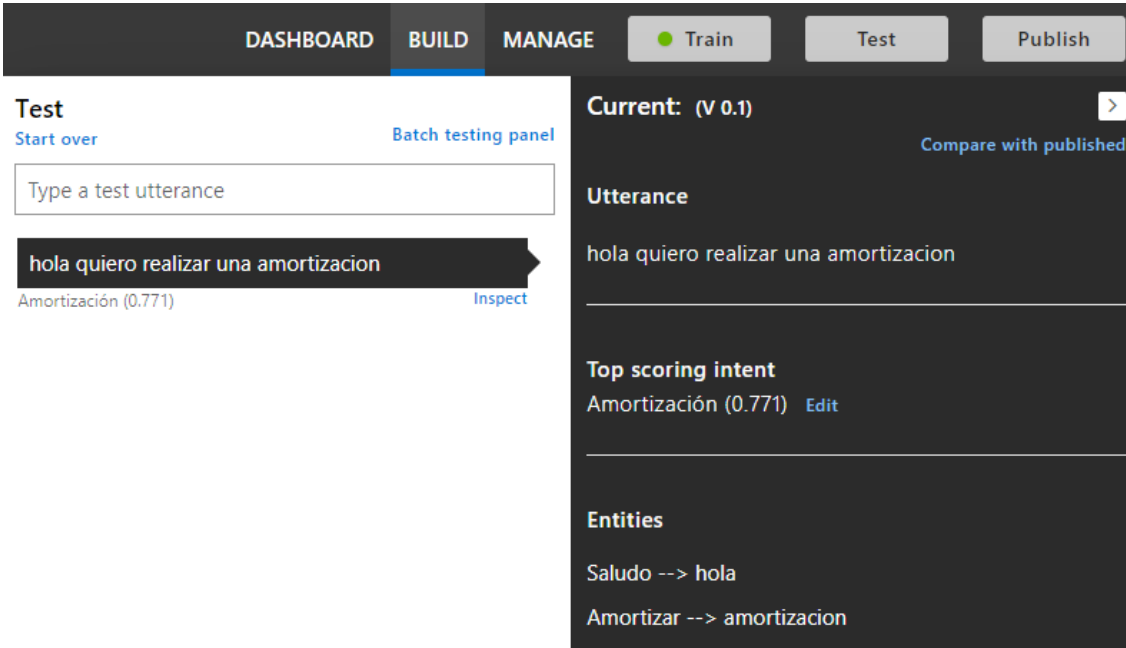
<input type="checkbox"/> Normalized Value	Synonyms
amortizar	<a href="#">Click here to start adding values</a>
amortizo	<a href="#">Click here to start adding values</a>
amortiza	<a href="#">Click here to start adding values</a>
amortización	<a href="#">Click here to start adding values</a>
amortizacion	<a href="#">Click here to start adding values</a>
amortizaciones	<a href="#">Click here to start adding values</a>



## El archivo Chatbot.json, es el que pertenece a la exportación del LUIS.

LUIS como servicio te permite ejecutar tus propios tests para deducir si estamos creando una aplicación con sentido o si realmente no estamos invirtiendo los recursos necesarios en cada intento o entidad.

Veamos una prueba:



The screenshot displays the LUIS interface with the following components:

- Navigation:** DASHBOARD, BUILD (active), MANAGE, Train, Test, Publish.
- Test Panel (Left):**
  - Buttons: Start over, Batch testing panel.
  - Input field: Type a test utterance.
  - Utterance: hola quiero realizar una amortizacion
  - Score: Amortización (0.771)
  - Action: Inspect
- Results Panel (Right):**
  - Current: (v 0.1) Compare with published
  - Utterance: hola quiero realizar una amortizacion
  - Top scoring intent: Amortización (0.771) Edit
  - Entities:
    - Saludo --> hola
    - Amortizar --> amortizacion

Como podemos ver en la imagen anterior, al escribir “hola quiero realizar una amortizacion” en la parte izquierda de la interfaz donde dice “type a test utterance”, nuestra aplicación ha deducido que queremos llegar al proceso Amortización a partir de un 77% (0.771). LUIS ha detectado dos posibles entidades, “hola” que forma parte de la entidad Saludo y “amortizacion” que forma parte de la entidad Amortizar.

Esto quiere decir que tenemos que entrenar al servicio para que nuestro porcentaje final en las sentencias más relevantes que el usuario pueda hacer sean de una deducción de 1, lo que equivale a un 100% de acierto y sin margen de error, y deducimos entonces que en el momento de este ejemplo nuestro LUIS no estaba lo suficientemente entrenado.

*\*Detalle importante;* he querido mostrar un ejemplo donde “amortizacion” no lleve acento en la declaración de prueba, ya que de esta manera demostramos cuán bien construida esta una entidad. Si una entidad está bien “entrenada” nos reconocerá el camino ya sea con acento o sin, porque de eso nos habremos encargado en el proceso de construcción, creando utterances y entidades con acento y sin acento.

## 2.3 Código C# Visual Studio

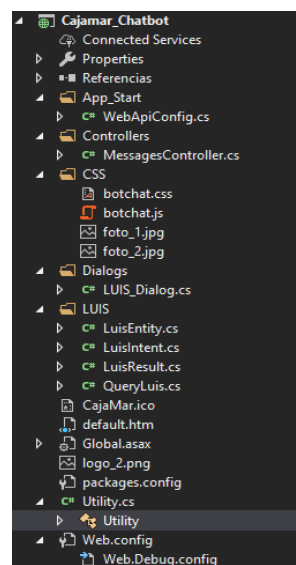
En el proceso de este proyecto he tenido que implementar un código mucho más escueto al principal, en la primera entrega del desarrollo se dispone de una solución de visual studio donde el código alcanzaba más de 4.000 líneas ya que se buscaba todo el trabajo en este.

La solución a la escalabilidad más adecuada, personalmente, fue crear una base de datos con una lógica capaz de permitirme escribir el código en menos de 700 líneas.

Quiero aclarar que no todo el código que se ha “creado” tanto en esta etapa final de desarrollo como en la inicial, está hecho de 0.

C# es un lenguaje muy diverso, amplio y complicado, está orientado a objetos y tiene infinidad de funciones y variantes. En mi caso he tenido que crear más en la base de datos y pensar cuáles de las tantas funciones que existen eran necesarias en mi código.

Durante todo el código me he dedicado a crear secciones y comentarios explicativos sobre lo más relevante, a continuación veremos de qué se compone el código.



Como podemos observar partimos de varias secciones dentro de nuestra solución, veremos las más importantes ya que es donde yo he tenido que trabajar:

## VS - MessagesController

Este apartado aunque poco extenso es relevante ya que se compone de la interacción con nuestra tabla de Usuarios y Logs.

Aquí controlamos de manera inicial con esta API el idioma del chat y su actividad para la respuesta del chat:

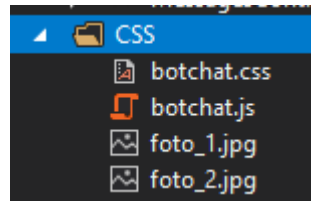
```
public async Task<HttpResponseMessage> Post([FromBody]Activity activity)
{
    activity.Locales = "es-ES";
    try
    {
        await HandleActivity(activity);
    }
    catch (Exception ex)
    {
        System.Diagnostics.Trace.WriteLine(ex.ToString());
    }

    return Request.CreateResponse(HttpStatusCode.Accepted);
}
```

Las dos bases de este apartado son `HandleActivity(Activity activity)` y `ActivityTypes.ConversationUpdate`.

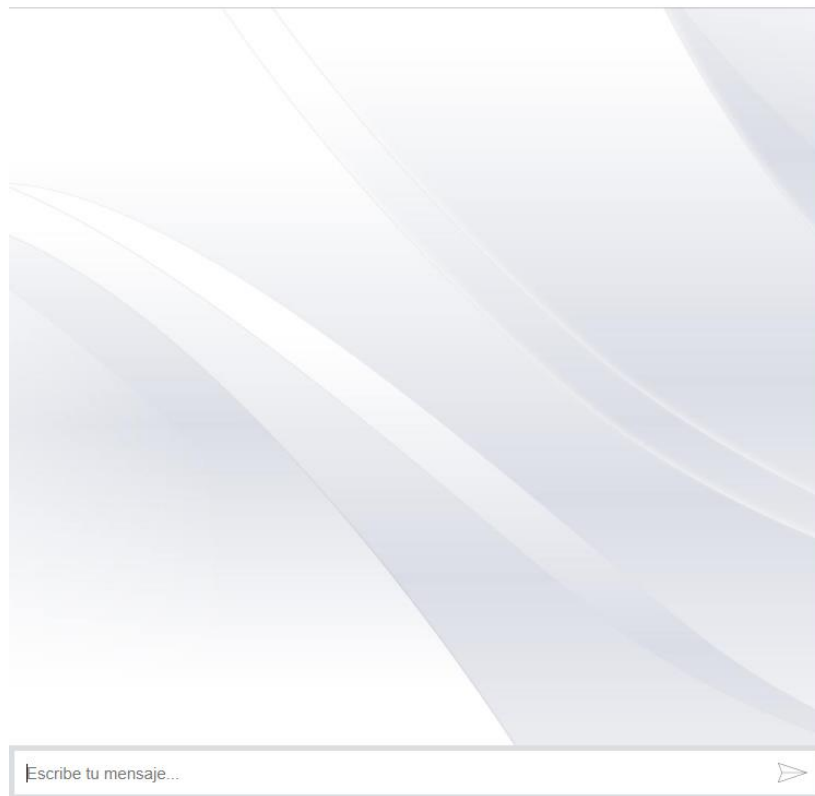
En estas dos funciones se controla la actividad que tiene el usuario en la conversación y su registro en la Base de Datos.

## VS - CSS



Como ya sabremos este apartado se basa en las hojas de estilo en cascada, o de manera más resumida, hojas de estilo.

Aquí tocamos colores, posicionamiento web, insertamos iconos, y todo lo que tenga que ver con la interfaz de nuestro chat, que se basa en la siguiente ventana final;



## VS - Utility

```

namespace LUIS_Dialog
{
    40 referencias
    public static class Utility
    {
        public static void IfIs<TDerived>(this object baseObject, Action<TDerived> doIfDerived) where TDerived : TBase
        {
            if (baseObject is TDerived)
            {
                doIfDerived((TDerived)baseObject);
            }
        }

        public static TResult IfIs<TDerived, TResult>(this object baseObject, Func<TDerived, TResult> doIfDerived) where TDerived : TBase
        {
            if (baseObject is TDerived)
            {
                return doIfDerived((TDerived)baseObject);
            }
            return default;
        }

        public static void ArgumentNotNull<TArg>(TArg arg, string name)
        {
            if (arg == null)
            {
                throw new ArgumentNullException(name);
            }
        }

        //Esta función crea la carta donde iran todos los botones
        public static Activity ShowButtons(IDialogContext context, string strText, IList<string> ListaBotones)
        {
            return ShowButtons(context, strText, ListaBotones, true);
        }

        //Esta función crea los nombres de los botones y el valor que cogen
        public static List<CardAction> CreateButtons(IList<string> ListaBotones, int NumeroBotones)
        {
            List<CardAction> listaBotones = new List<CardAction>();
            for (int i = 0; i < NumeroBotones; i++)
            {
                listaBotones.Add(new CardAction() { Title = ListaBotones[i], Value = i });
            }
            return listaBotones;
        }

        //Esta función crea la carta donde iran todos los botones que desaparecen al clicar
        public static Activity ShowButtons_DeletedAnswer(IDialogContext context, string strText, IList<string> ListaBotones)
        {
            return ShowButtons(context, strText, ListaBotones, false);
        }

        //Esta función muestra un enlace para descargar un URL
        public static Activity Show_URL_Card(IDialogContext context, string strText, string URL_image)
        {
            return Show_URL_Card(context, strText, URL_image, true);
        }

        //Esta función establece la conexión a la base de datos
        public static DataSet Ejecutar(string cmd)
        {
            DataSet ds = new DataSet();
            try
            {
                ds = Ejecutar(cmd);
            }
            catch { }
            return ds;
        }
    }
}
    
```

En el apartado *utility* tenemos funciones muy importantes entre ellas la conexión de la base de datos. Esta región específicamente la voy a plasmar en

el documento directamente para ver los diferentes requerimientos de la conexión.

```
public static DataSet Ejecutar(string cmd)
{
    SqlConnection Con = new SqlConnection("Data Source = DIRECCION DE LA
    BD; Initial Catalog = NOMBRE DE LA DB; Persist Security Info = True; User ID = ID
    DEL USUARIO DE LA BD; Password = CONTRASEÑA DEL USUARIO DELA DB");
    Con.Open();
}
```

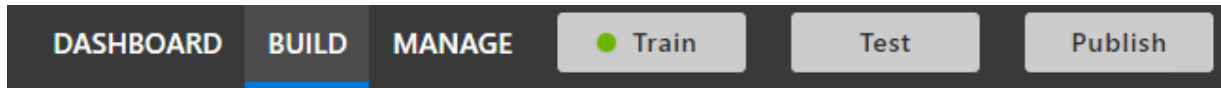
Como podemos ver en la función de la conexión he expresado los términos necesarios para establecer el vínculo entre el SQL y el código.

## VS - Web.config

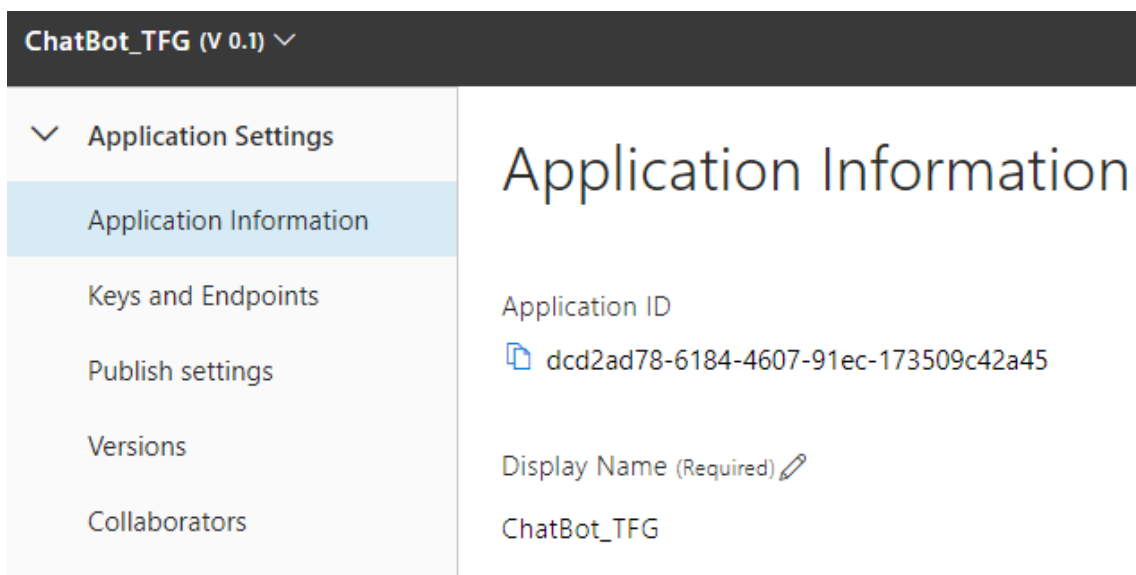
```
<appSettings>
  <!-- update these with your Microsoft App Id and your Microsoft App Password-->
  <add key="MicrosoftAppId" value=""/>
  <add key="MicrosoftAppPassword" value=""/>
  <add key="LUISID" value="0235f7ba-b94b-4d34-8a57-6b66e5cc7870"/>
  <add key="LUISKey" value="5a9cae1f114b4b7f80418e255589e309"/>
</appSettings>
```

En este apartado hay 4 *values/valores*, dos de ellos referentes a Microsoft, los cuales son necesarios para la ejecución en línea del chat y se obtienen a partir de Azure.

Los otros dos son los que obtenemos en LUIS, a partir de nuestro proyecto en el siguiente apartado de publicación:

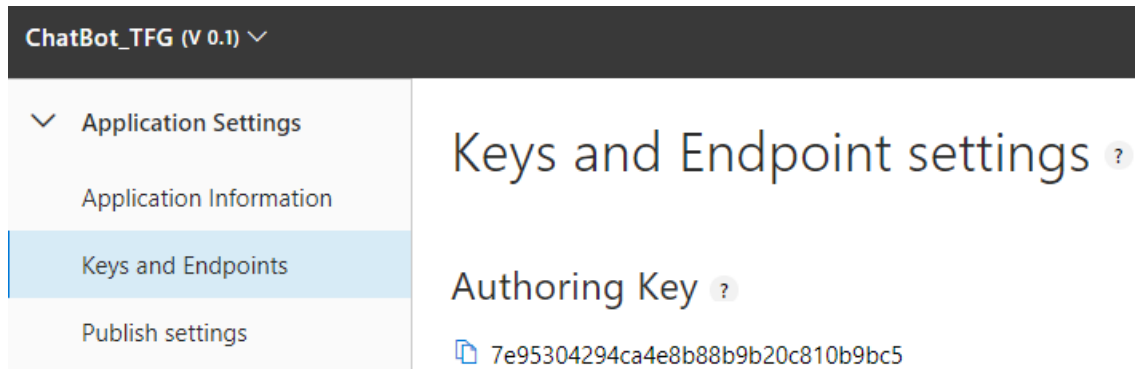


De aquí sacamos el valor ID que nos piden:





Y de aquí obtenemos el valor *key* que nos piden:



**Esta parte es importante** explicarla al detalle ya que Azure es un servicio de Microsoft pensado para grandes empresas que se puedan costear sus servicios. Yo disponía de una cuenta limitada gracias a mi anterior empleo, pero es una cuenta a la cual ya no tengo acceso y no me puedo permitir pagar una de manera particular.

El fin de este proyecto nunca se ha basado en tener activo un servicio de chat automatizado, sino en crear un chat automatizado escalable, así que lo único que pierdo es el no poder tener este servicio activo. La solución a este problema es sencilla ya que solo requiere de una empresa que esté dispuesta a pagar por él.

De todos modos, el proyecto en su totalidad lo he probado en el Emulador de Microsoft, y es lo que he plasmado en esta memoria, el acceso al servidor SQL con la base de datos que he creado ha sido temporal pero aún así toda la base de datos esta en forma de *back up* en lenguaje SQL ya que es como fui creando todas las tablas.

Con esto quiero decir que el proyecto está completo, la conexión SQL a las tablas no está disponible por el presupuesto, y la publicación del proyecto de manera activa/continua también depende de poseer una cuenta Azure de pago.

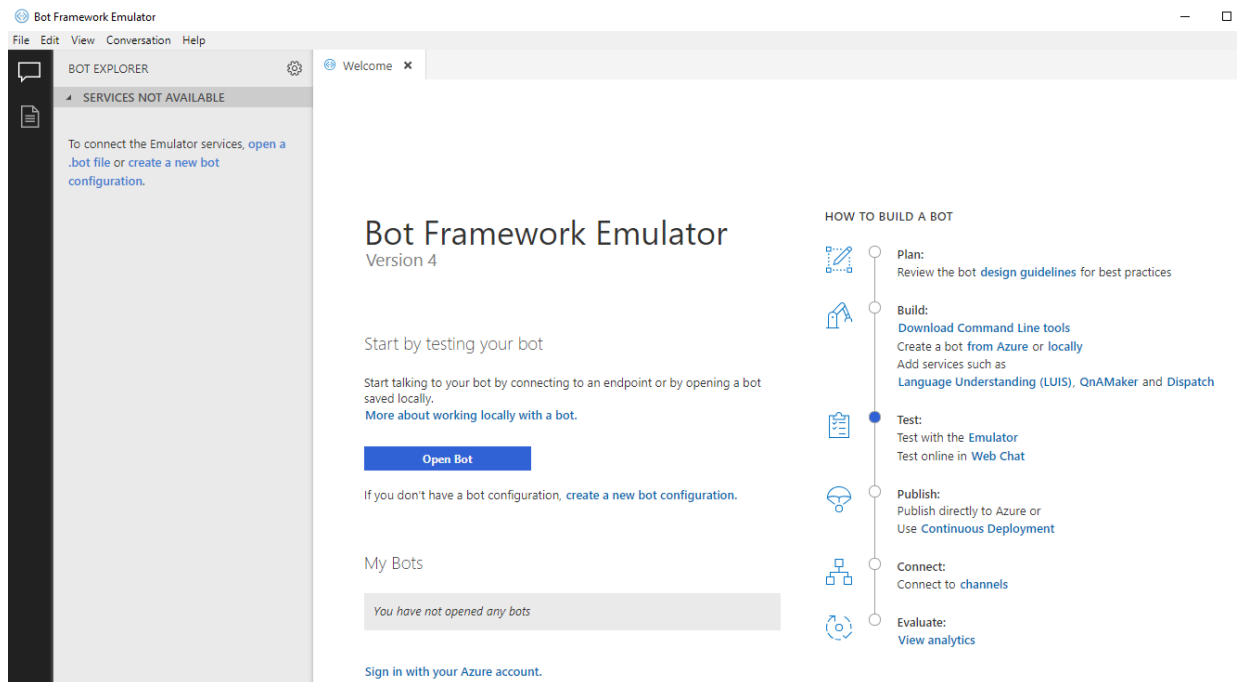
## 2.4 Microsoft Bot Framework Emulator

Esta aplicación nos ayudará a probar de manera local el chat que estamos desarrollando y así depurarlo con más facilidad ya que no disponemos del presupuesto de empresa necesario para establecer un chatbot en línea con un servidor SQL.

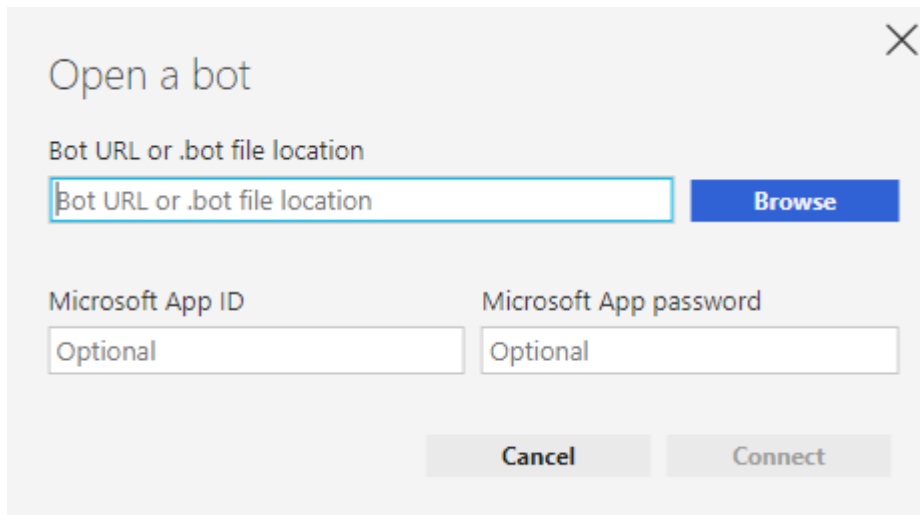
Se trata de una aplicación gratuita desarrollada por Microsoft la cual podemos encontrar en el siguiente enlace [6]:

<https://github.com/Microsoft/BotFramework-Emulator/releases/tag/v4.4.1>

Solo tenemos que elegir la plataforma en la que usaremos dicha aplicación.



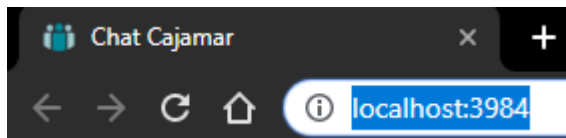
Esta es la interfaz principal de esta aplicación y nos dirigiremos a Open Bot para crear una emulación de cero.



Nos pedirá 3 apartados, como ya hemos explicado con anterioridad los Microsoft ID/Password no los vamos a necesitar. Debemos o bien abrir un bot ya creado con anterioridad o introducir la url del bot.

Que quiere decir con la url del bot y donde podemos encontrarla?

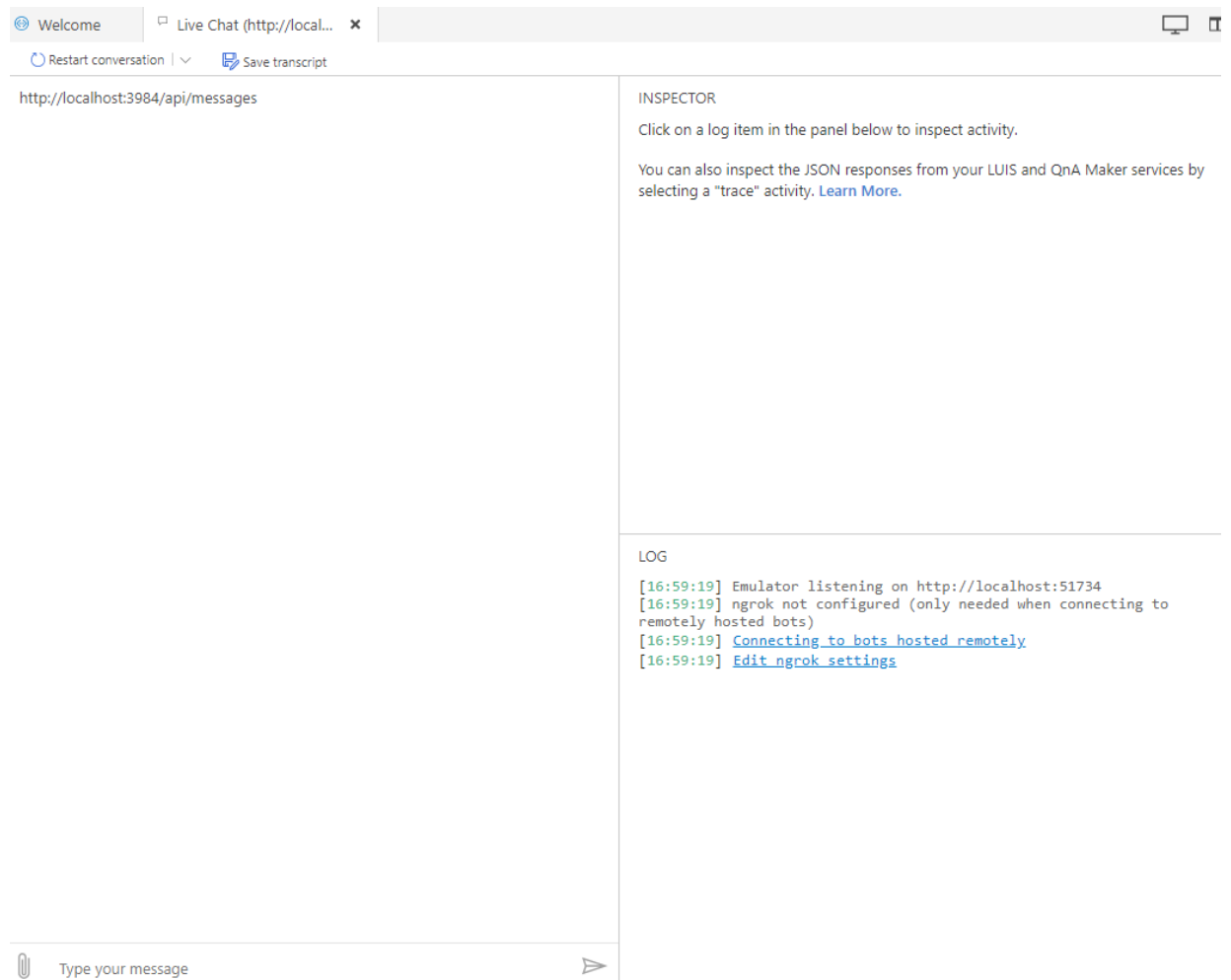
Muy sencillo, se trata de la url local que crea nuestro código C#, cuando ejecutamos el proyecto nos abre el navegador con una dirección como la siguiente:



Cierto es, y ya nos lo indicará si no lo ponemos nosotros, que debemos introducir "/api/messages" a continuación de dicha dirección:

<http://localhost:3984/api/messages>

Una vez ejecutado estaremos en la siguiente ventana para hacer las pruebas y depuraciones que veamos pertinentes en nuestro proyecto.



La imagen anterior es la ventana principal para emular el chatbot.

Ejemplo visual de cómo se ven los desplegables en este emulador, vemos que al haber varias opciones podemos utilizar una flecha a la izquierda y derecha.

Ahora mismo puedo ayudarte con la cuestión: ¿Qué operación quieres realizar?

Elija una de las siguientes opciones:

11 minutes ago

Cancelación/Amortización/Entrega

Just now

De acuerdo. Escribe o indica el convenio o producto al que pertenece.

Elija una de las siguientes opciones:

Just now

< Fomento/MOPU    Generalitat Valenciana    Junta de Castilla y León >

En las siguientes imágenes vemos un ejemplo de un recorrido correcto.

Bienvenido al chat del grupo de Cajamar. Esta conversación es totalmente confidencial. ¿En qué puedo ayudarte?

Just now

Me gustaría cancelar

Just now

De acuerdo. Escribe o indica el convenio o producto al que pertenece.

Elija una de las siguientes opciones:

Just now

Fomento/MOPU    Generalitat Valenciana    Junta de Castilla y León

Elija una de las siguientes opciones:

Just now

Junta de Castilla y León

Just now

Vale. Has seleccionado el convenio de Junta Castilla y León.  
Por favor, indícame el sub-convenio al que corresponde la operación.

Elija una de las siguientes opciones:

Just now

Vivienda Rural    Creación de Empleo para Autónomos    Otros

Elija una de las siguientes opciones:

Just now

Vivienda Rural

Just now

Entendido. Has seleccionado Vivienda Rural. Por favor, indica si es Total o Parcial.

Elija una de las siguientes opciones:

Just now

Total

Parcial

Total

Just now

La amortización se tiene que realizar coincidiendo con la fecha de liquidación del recibo y previa autorización del Organismo. Esta autorización debe estar realizada con un plazo máximo de dos meses anteriores a la fecha de liquidación del préstamo.  
Por tanto, si no tiene autorización del Organismo deberán solicitar escrito al cliente en el que autorice la solicitud de entrega e importe al Organismo y comprobar que resta menos de 2 meses para la liquidación del recibo. Una vez obtenga escrito y compruebe que quedan menos de 2 meses para la siguiente liquidación deberán poner petición vía remedy en la categoría: Inversión Gestionada/Convenios/Entregas a Cuenta/Cancelaciones con la carta para solicitar al Organismo autorización.

¿Te ha sido útil la información proporcionada?

Just now

Sí

Just now

Vale. ¿Podrías decir cómo podría mejorar?


Just now

Todo correcto.

Just now

Gracias por tu sinceridad. Si necesitas más ayuda no dudes en contactar conmigo.

Just now

 type your message



Ahora veamos que pasa si escribimos opciones que el LUIS no reconoce como válidas en los niveles correspondientes. En el siguiente caso hablamos de la fase inicial, la cual nos requiere una Cancelación/Amortización/Entrega, si el cliente/usuario escribe una consulta fuera de lugar, en este caso “Convenio”, vemos como el chat nos va a decir que no nos ha entendido pero que elijamos una de las opciones válidas de las que dispone para ese nivel.

Welcome | Live Chat (http://local... x

Restart conversation | Save transcript

http://localhost:3984/api/messages

Bienvenido al chat del grupo de Cajamar. Esta conversación es totalmente confidencial. ¿En qué puedo ayudarte?

9 minutes ago

Hola

9 minutes ago

Buenos días Sr/Sra. ¿En qué puedo ayudarte?

9 minutes ago

Convenio

9 minutes ago

No te he entendido.

Ahora mismo puedo ayudarte con la cuestión: ¿Qué operación quieres realizar?

Elija una de las siguientes opciones:

9 minutes ago

Cancelación/Amortización/Entrega

9 minutes ago



Un ejemplo de cómo podemos acabar la conversación con la detección del **adiós en cualquier nivel/fase de la conversación**, el cual se reconoce como una **despedida** y se **reinicia** el registro de la conversación, con lo que volveríamos a las opciones iniciales.

The screenshot displays a chatbot interface with the following sequence of messages and user actions:

- Bot:** "Hola Sr/Sra. ¿En qué puedo ayudarte?" (A minute ago)
- User:** "amortizacion" (A minute ago)
- Bot:** "De acuerdo. Escribe o indica el convenio o producto al que pertenece." (A minute ago)
- Bot:** "Elija una de las siguientes opciones:" (A minute ago)
- User:** "adios" (Just now)
- Bot:** "Adiós! No dudes en contactarme si necesitas mi ayuda." (Just now)
- User:** "Hola" (Just now)
- Bot:** "Saludos Sr/Sra. ¿En qué puedo ayudarte?" (Just now)
- User:** "Amortizar" (Just now)
- Bot:** "Entendido. Escribe o indica el convenio o producto al que pertenece." (Just now)
- Bot:** "Elija una de las siguientes opciones:" (Just now)

At the bottom of the chat window, there is a horizontal menu with five buttons: "Fomento/MOPU", "Generalitat Valenciana", "Junta de Castilla y León", "Región de Murcia", and "MAGRAMA".

En la imagen anterior podemos apreciar la detección del LUIS cuando escribimos "Amortizar", ya nos muestra el desplegable de Cancelación/Amortización/Entrega, directamente lo reconoce y pasa a la siguiente fase/nivel.

Otro ejemplo de la detección del LUIS, saltamos directamente al paso de los convenios por el hecho de escribir “Deseo cancelar”, ya que tenemos al LUIS *entrenado* para que detecte “cancelar” (entre muchas otras variantes) como opción válida en esa primer/a nivel/fase.

Bienvenido al chat del grupo de Cajamar. Esta conversación es totalmente confidencial. ¿En qué puedo ayudarte?

Just now

Hola mi nombre es Alberto Silla

Just now

Bienvenido/a Sr/Sra. ¿En qué puedo ayudarte?

Just now

Deseo cancelar

Just now

Entendido. Escribe o indica el convenio o producto al que pertenece.

Elija una de las siguientes opciones:

Just now

Fomento/MOPU    Generalitat Valenciana    Junta de Castilla y León

📎 | type your message    ➤

En este caso al haber reiniciado el registro de la conversación con nuestro “Adiós” como despedida, el chat no entiende el siguiente “Ok” como una opción válida dentro de la fase inicial, y directamente nos guía a los niveles que hay establecidos como camino.

The screenshot shows a chat interface with the following elements:

- A blue button labeled "Adios" with a timestamp "9 minutes ago" below it.
- A grey message box containing the text: "Gracias por tu sinceridad. Si necesitas más ayuda no dudes en contactar conmigo." with a timestamp "9 minutes ago" below it.
- A blue button labeled "Ok" with a timestamp "9 minutes ago" below it.
- A grey message box containing the text: "Disculpa, pero no he entendido lo que has dicho."
- A grey message box containing the text: "Ahora mismo puedo ayudarte con la cuestión: ¿Qué operación quieres realizar?"
- A grey message box containing the text: "Elija una de las siguientes opciones:" with a timestamp "9 minutes ago" below it.
- A blue button labeled "Cancelación/Amortización/Entrega" at the bottom right.

Este ejemplo de Generalitat Valenciana/Orden Marzo 2008 Sequía 2014, el chat nos pide que contestemos si hemos recibido o no una subvención, la respuesta es sencilla pero si le damos una opción que el LUIS no ha entrenado (lógicamente), el chat nos guía directamente dándonos las dos posibles opciones, en este caso “Nolose”.

The screenshot shows a chat interface with three main interaction points:

- Initial Selection:** A blue button labeled "Generalitat Valenciana" is selected. Below it, a grey box contains the text: "Vale. Has seleccionado el convenio de Generalitat Valenciana. Por favor, indicame el sub-convenio al que corresponde la operación. Elija una de las siguientes opciones:". A "Just now" timestamp is visible.
- Second Selection:** A blue button labeled "Orden Marzo 2008 Sequia 2014" is selected. Below it, a grey box contains the text: "Entendido. Has seleccionado Orden Marzo 2008 Sequia 2014 (PRODUCTO PZ122 o PZ123). Por favor, indique si el préstamo ha recibido subvención o no.". A "Just now" timestamp is visible.
- Clarification and Final Selection:** A grey box contains the text: "Disculpa, pero no he entendido lo que has dicho. Ahora mismo puedo ayudarte con la cuestión: Has seleccionado Orden Marzo 2008 Sequia 2014 (PRODUCTO PZ122 o PZ123). Por favor, indique si el préstamo ha recibido subvención o no. Elija una de las siguientes opciones:". Below this, two buttons labeled "Sí" and "No" are visible. To the right, a blue button labeled "Nolose" is selected, with a "Just now" timestamp below it.

Otro ejemplo de respuesta guiada cuando no se reconocen las opciones válidas. Con estos ejemplos vemos como el chatbot por una parte dispone de la inteligencia artificial del LUIS y por otra parte le hemos dado una forma y recursos en la base de datos capaz de lidiar con cualquier conversación, ya que siempre que el cliente/usuario no escriba una opción válida o adecuada para el nivel actual que se encuentre, le guiaremos, de todas formas, con un recorrido de opciones hasta la meta deseada.

Indicarles que deberán interponer petición a CONVENIOS, Elemento: entregas a cuenta/cancelaciones; indicando en dicha petición número de operación, importe y tipo de entrega que se quiere realizar.

¿Te ha sido útil la información proporcionada?

Just now

Nolose

Just now

Disculpa, pero no he entendido lo que has dicho.

Ahora mismo puedo ayudarte con la cuestión: ¿Te ha sido útil la información proporcionada?

Elija una de las siguientes opciones:

Just now

Sí

No

Aquí se puede apreciar el final de la conversación y como se reinicia el registro de esta. Vemos que la última pregunta del chat se basa en como se podría mejorar el servicio que el chat está dando, una vez el usuario/cliente contesta, el chat registra esa última respuesta y devuelve el mensaje final, con lo cual y tal y como hemos indicado en el código, la conversación vuelve al principio, por eso después del "Ok" por parte del cliente/usuario el chat devuelve la primera parte del recorrido con sus respectivas opciones válidas iniciales.

The screenshot displays a chat interface with the following elements:

- User Message:** "De acuerdo. ¿Podrías decir cómo podría mejorar?" (Just now)
- Bot Response:** "Gracias por tu sinceridad. Si necesitas más ayuda no dudes en contactar conmigo." (Just now)
- User Message:** "No te he entendido." (Just now)
- Bot Response:** "Ahora mismo puedo ayudarte con la cuestión: ¿Qué operación quieres realizar?" (Just now)
- User Message:** "Elija una de las siguientes opciones:" (Just now)
- Bot Response:** "Cancelación/Amortización/Entrega" (Just now)
- User Action:** A blue button labeled "Es posible" (Just now) is visible on the right side of the chat.
- User Action:** A blue button labeled "Ok" (Just now) is visible on the right side of the chat.

## 2.5 Actividades no previstas

### **Botones clickables**

He tenido que invertir varias horas en la investigación del funcionamiento de los “cards buttons” (PrompStyle), no es nada más que el formato en el que se “expresan” las opciones que le damos al usuario en el chat, los clickables.

### **Mensaje principal del chat**

Así mismo he dedicado varias horas en averiguar cómo funciona el mensaje principal (webChat-Directline) que aparece cuando iniciamos sesión en la ventana de diálogo.

### **Detección de idioma**

He tenido complicaciones para identificar el idioma de la consola que usaremos en el chat, pero después de una buena búsqueda y consultas con Microsoft he encontrado la solución, que para mi sorpresa era muy trivial ya que se basa en una línea de código:

```
session.preferredLocale('es', function(err){
  if (!err) {
    console.log("Locale: es");
  } else {
    console.log(err);
  }
}
```

Si cambiamos `Locale: es`, por `Locale: en`, estaremos cambiando la detección del idioma al inglés.

### **Cambio/Reestructuración de la implementación del código C#**

## 2.6 Objetivos principales del proyecto

- Establecer una BBDD.
- Crear una estructura en LUIS capaz de ofrecer un servicio eficaz.
- Crear una estructura en C# dedicada a la escalabilidad de la Base de Datos.
- Establecer conexiones SQL y comprobar si la lógica es la adecuada.
- Desarrollar las pruebas necesarias para el correcto funcionamiento del chat.
- Desarrollar una BBDD bancaria.

*Más adelante pude comprobar que había que hacer una implementación mucho más escueta en el código pero más elaborada, para tener toda la lógica del chat en la base de datos creando tablas dinámicas con un recorrido inteligente.*



### 3. Conclusiones

La presente investigación se ha dedicado al desarrollo de una característica indispensable para proyectos en crecimiento, la escalabilidad. Se ha utilizado como herramienta de implementación, software totalmente gratuito salvo en una ocasión.

El desarrollo de la base de datos, el lenguaje más óptimo y sus correspondientes fuentes, ha sido creado a partir de una cuenta limitada de Microsoft Azure, la cual ya no está disponible más que haciendo pagos mensuales o anuales. De esta parte se concluye positivamente, que el objetivo principal se puede lograr de forma prematura en una base de datos tipo económicamente factible, véase gratuita o con un coste muy reducido. Como lado negativo se concluye que la elección de este apartado ha sido la mejor a largo plazo para grandes proyectos y la peor a corto plazo para investigaciones económicamente limitadas o proyectos de menor escala.

El proceso de implementación del código se ha tratado desde un breve periodo inicial con un enfoque erróneo, se estaba estipulando una base de datos más simple con un código que implementaba todo el contenido. La corrección de esta estrategia ha permitido establecer un código escueto pero versátil, obteniendo lógica a partir de la conexión con las tablas dinámicas en las cuales reside el grosor a escalar.

Del análisis de los ejemplos mostrados en este trabajo de investigación se concluye que la conexión entre el aprendizaje profundo/automático (LUIS), la base de datos lógica y el código, resulta un chatbot capaz de tener un balance eficaz entre crecimiento potencial y funcionalidad.

Del estudio en la emulación se comprueba la garantía del producto, se aprovecha la inteligencia artificial para crear un resultado más ligero combinado con un método guiado, garantizando así la satisfacción del consumidor.

En este trabajo se ha elaborado un producto final deseado, cumpliendo los objetivos principales y añadiendo funciones adicionales para lograr su propósito. Se puede concluir que la futura exploración de este proyecto no es más que la correcta adaptación a servidores que permitan dar un servicio en línea continuo con soporte técnico y un almacenamiento adecuado.

## 4. Glosario

**LUIS:** Language Understanding Intelligence Service, básicamente Machine Learning y el servicio que usamos en el proyecto.

**Machine Learning:** Es un método de análisis de datos que automatiza la construcción de modelos analíticos.

**BBDD o DB:** Base de datos o por lo contrario en inglés Data Base.

**Azure:** Plataforma de Microsoft para gestionar proyectos.

**BOT:** Nos referimos a robot, utilizado para cualquier tarea automatizada.

**ChatBot:** Chat robotizado/automatizado.

**C#:** Lenguaje de programación orientado a objetos, la almohadilla se pronuncia Sharp.

**Cajamar:** Cajamar Caja Rural es la entidad financiera resultante de la fusión de cajas rurales de Andalucía Oriental, Murcia, Madrid, Castilla y León, Baleares, Comunidad Valenciana y Canarias y Caixa Albalat.

**Utterance:** Sentencia.

**Test:** Prueba.

**Phrase List:** Un tipo de lista en LUIS.

**Value:** Valor.

**Workflow:** Funcionamiento.

**Intent:** Intento en este caso para LUIS

**Entitie:** Entidad en este caso para LUIS

## 5. Bibliografía

<https://github.com/microsoft/botframework-sdk> - Recursos para Bots de Microsoft [1]

<https://www.connectionstrings.com/sql-server/> - Información acerca de las conexiones a un servidor SQL [2]

<https://github.com/sunilpottumuttu/Useful-Urls-For-Developers> - Información y recursos para desarrolladores[3]

<https://azure.microsoft.com/en-gb/blog/conversational-bots-deep-dive-what-s-new-with-the-general-availability-of-azure-bot-service-and-language-understanding/> - Información útil acerca de la conversación del bot[4]

<https://github.com/Microsoft/computerscience/blob/master/Labs/AI%20and%20Machine%20Learning/Azure%20Bot%20Service/Azure%20Bot%20Service.md> – Primeros pasos sobre la construcción de un bot[5]

<https://github.com/Microsoft/BotFramework-Emulator/releases/tag/v4.4.1> - Descarga del Emulador Microsoft Bot Framework[6]

<https://www.luis.ai/> - Gestión de LUIS [7]

<https://github.com/Gr8z/ChatBotProject> - Proyecto de chatbot aleatorio[8]

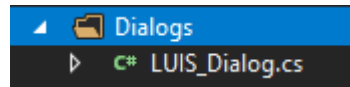
<https://visualstudio.microsoft.com/es/> - Descarga de VisualStudio[9]

<https://infoautonomos.eleconomista.es/financiacion-autonomos-empresas/lineas-del-instituto-de-credito-oficial-ico/> - Información sobre productos bancarios [10]

[https://azure.microsoft.com/es-es/free/search/?&OCID=AID719820\\_SEM\\_VAab2G2A&Inkd=Google\\_Azure\\_Brand&dclid=CPWO2ZayquICFUoC0wodmbMGYg](https://azure.microsoft.com/es-es/free/search/?&OCID=AID719820_SEM_VAab2G2A&Inkd=Google_Azure_Brand&dclid=CPWO2ZayquICFUoC0wodmbMGYg) – Gestión de Microsoft Azure[11]

## 5. Apéndice

### VS - LUIS\_Dialog



La esencia principal del código está en este apartado, explicado con comentarios **//VERDE**.

```
using Luis;
using Microsoft.Bot.Builder.Dialogs;
using Microsoft.Bot.Connector;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using System.Data;

namespace LUIS_Dialog.Runtime
{
    [Serializable]
    public class LUIS_Dialog : IDialog<object>
    {
        string LUIS_ID =
System.Configuration.ConfigurationManager.AppSettings["LUISID"];
        string LUIS_subscriptionKey =
System.Configuration.ConfigurationManager.AppSettings["LUISKey"];
        private const string EntityProductos = "Productos";

        public async Task StartAsync(IDialogContext context)
        {
            context.Wait(MessageReceivedAsync);
        }

        public async Task MessageReceivedAsync(IDialogContext context,
IAwaitable<IMessageActivity> argument)
        {
            var message = await argument;

            Activity reply;
            Activity message_url;

            string preced = "0";

            string mensaje_usuario = "";
            string mensaje_bot = "";
            string mensaje_error = "";
            string mensaje_LUIS = "";
            string Abrir_Peticion = "0";
            string Num_Consultas = "0";
            string Oficina = "Oficina_D";
            double Score_Intent = 0;
            Int32 intents_none = 0;
            int LUIS_especifico = 0;
            var LU = (Luis.LuisResult)null;
            string Intent_LUIS = null;
            bool Call_Luis = false;
            bool Call_Basicos = false;

```

```

string ID_User = message.From.Id; //Guardar ID del usuario

//Coje el valor de la hora actual de la zona "+1" y la guarda en una sting
DateTime now = DateTime.Now;
TimeZoneInfo LocalTimeZone =
TimeZoneInfo.FindSystemTimeZoneById("Romance Standard Time");
DateTime BarcelonaDateTime = TimeZoneInfo.ConvertTime(now,
TimeZoneInfo.Local, LocalTimeZone);
string BarcelonaDateTime_FormatSQL =
BarcelonaDateTime.ToString("yyyy-MM-dd HH:mm:ss");
string BarcelonaDateTime_string = BarcelonaDateTime.ToString();

//Se substituye la comilla para que no salga un error al gurdar en SQL los LOGS
string message_string = message.Text.ToString().Replace("'", "
comilla ");
mensaje_usuario = message_string;

//COMPROBAR USUARIO EN LA BASE DE DATOS, mira si existe el usuario y actualiza el
preced, en caso contrario, crea un usuario con el preced 0.
try
{
string CMD_ID = string.Format("SELECT preced, intent_none FROM
Usuarios WHERE id_usuario='{0}'", ID_User);
DataSet ds_ID = Utility.Ejecutar(CMD_ID);
DataRow[] dr_ID = ds_ID.Tables[0].Select();
int Length_ID = dr_ID.Length;
//Mira si existe con la longitud del select

if (Length_ID == 0)
//Si el usuario no existe, lo crea con el preced 0 y con intentos_fallados 0
{
preced = "0";
intents_none = 0;
string CMD_Preced = $"INSERT INTO Usuarios(id_usuario,
preced, intent_none, fecha_string, fecha) VALUES('{ID_User}', '{preced}',
{intents_none}, '{BarcelonaDateTime_string}', '{BarcelonaDateTime_FormatSQL}')";
Utility.Ejecutar(CMD_Preced);
}
else
//Si el usuario existe guarda los valores de preced i intents nones de manera
local
{
preced = ds_ID.Tables[0].Rows[0]["preced"].ToString().Trim();
intents_none = Convert.ToInt32(dr_ID[0].ItemArray[1]);
}
}
catch (Exception error)
//Si hay algun error en acceder con SQL a la base de datos anterior, recoge el
error
{
string message_error_3 = "USUARIO ERROR: Error con la
identificación del usuario con el chatbot. Contacte con un administrador.";
mensaje_bot = mensaje_bot + message_error_3;
mensaje_error = mensaje_error + error.ToString();
await context.PostAsync(message_error_3);
}
}

```

```

//COMPROBAR SI SE NECESITA LLAMAR A LA API LUIS, COMPROBANDO SI EL MENSAJE DEL
USUARIO SE CORRESPONDE EN LA BASE DE DATOS
    try//Exactamente este try sirve para no tener que llamar a la API de
LUIS si seleccionan un boton porque el valor se correspondera en la BDD
    {
        string CMD_LUIS_Especifico_Inicial = string.Format("SELECT COUNT
(id_LUIS_respuestas) as COUNT_ID FROM LUIS_Respuestas WHERE LUIS='{0}' AND
precedente='{1}'", message_string, preced);
        DataSet ds_LUIS_Especifico_Inicial =
Utility.Ejecutar(CMD_LUIS_Especifico_Inicial);
        string COUNT_ID =
ds_LUIS_Especifico_Inicial.Tables[0].Rows[0]["COUNT_ID"].ToString().Trim();

        if (COUNT_ID.Equals("0")) //Si no encuentra el mensaje en la base
de datos teniendo en cuenta el precedente
        {
            Call_Luis = true; //Llamara la API de LUIS
        }
        else
        {
            Call_Luis = false; //No llamara la API de LUIS
            mensaje_LUIS = message_string;
        }
    }
    catch (Exception error) //Controlar prosible error
    {
        string message_error_3 = "Error inesperado. Contacte con un
administrador.";
        mensaje_bot = mensaje_bot + message_error_3;
        mensaje_error = mensaje_error + error.ToString();
        await context.PostAsync(message_error_3);
    }

//Si el mensaje es valido (texto) y no ha detectado el mensaje directamente en la
base de datos (try anterior), llama la API de LUIS
    if ((message.Text != null)&(Call_Luis))
    {
        try
        {
            LU = await QueryLuis.GetIntentAndEntitiesFromLuis(LUIS_ID,
LUIS_subscriptionKey, message_string); //Recoge el resultado del LUIS
(LUIS_Result)
            Intent_LUIS = ((LuisIntent[])LU.Intents)[0].Intent;
//Se guarda el intento con mas probabilidad
            Score_Intent = ((LuisIntent[])LU.Intents)[0].Score;
//Se recoge el score del intento con mas probabilidad
            string Intent_Producto = "";

//Si la prob del intento es menor al 10%, autoasignamos intent NONE (Para depurar
posibles errores en el LUIS)
            if ((Score_Intent < 0.1)|(preced == "1002"))
//preced = 1002 cuando recogemos la información de las posibles mejoras (caso
especifico)
            {
                Intent_LUIS = "None";
            }
        }
    }

```

```

else
{
    foreach (Luis.LuisEntity Ent in LU.Entities)
    {
        if (Ent.Type.ToString() == EntityProductos)
        {
            Intent_Producto = Ent.Entity.ToString();
        }
    }
}

string CMD_LUIS_Especifico_Inicial = string.Format("SELECT
COUNT(id_LUIS_respuestas) as COUNT_ID FROM LUIS_Respuestas WHERE LUIS='{0}' AND
precedente='{1}'", Intent_Producto, preced);
DataSet ds_LUIS_Especifico_Inicial =
Utility.Ejecutar(CMD_LUIS_Especifico_Inicial);
string COUNT_ID =
ds_LUIS_Especifico_Inicial.Tables[0].Rows[0]["COUNT_ID"].ToString().Trim();

if (COUNT_ID.Equals("0")) //Si no encuentra el producto en la
base de datos teniendo en cuenta el precedente
{
    Intent_LUIS = Intent_LUIS; //No hacemos ningun cambio
}
else
{
    Intent_LUIS = Intent_Producto; //Asignar caso como el
valor del producto
}

}
catch (Exception error_log) // recoger posible error
{
    string message_error_1 = "LUIS ERROR: Error con la licencia
de LUIS. Contacte con un administrador.";
    Intent_LUIS = "None";
    mensaje_bot = mensaje_bot + message_error_1;
    mensaje_error = mensaje_error + error_log.ToString();
    await context.PostAsync(message_error_1);
}

//COMPROBAR SI SE NECESITA LLAMAR A MENSAJES BASICOS (POR SI NO SABE EN CONVENIO)
try //Se comprueba por el caso "Desconocido/no lo se) porque hay
un caso especifico que el LUIS avanza con este intento y lo comprobamos en
respuestas
{
    string CMD_LUIS_Especifico_Inicial = string.Format("SELECT
LUIS_stats FROM LUIS_Respuestas WHERE LUIS='{0}' AND precedente='{1}'",
Intent_LUIS, preced);
DataSet ds_LUIS_Especifico_Inicial =
Utility.Ejecutar(CMD_LUIS_Especifico_Inicial);
DataRow[] dr_LUIS_Especifico_Inicial =
ds_LUIS_Especifico_Inicial.Tables[0].Select();
int Length_LUIS_Especifico_Inicial =
dr_LUIS_Especifico_Inicial.Length;

if (Length_LUIS_Especifico_Inicial == 0)
//Si no encuentra el "No lo se" en LUIS_Respuestas significa que tiene que llamar
a la tabla de mensajes básicos
{
    Call_Basicos = true; //Llamar mensajes_Basicos
}
}

```



```

        mensaje_LUIS = Intent_LUIS;
    }
    else
    {
        mensaje_LUIS =
ds_LUIS_Especifico_Inicial.Tables[0].Rows[0]["LUIS_stats"].ToString().Trim();
        Call_Basicos = false;
//No llamar mensajes basicos y pasar al LUIS_Especifico con el siguiente
precedente
        LUIS_especifico = 1;
    }
}
catch (Exception error) //Controlar posibles errores
{
    string message_error_3 = "Error inesperado 2. Contacte con un
administrador.";
    Call_Basicos = false;
    mensaje_bot = mensaje_bot + message_error_3;
    mensaje_error = mensaje_error + error.ToString();
    await context.PostAsync(message_error_3);
}

}
else //En caso que no tengamos que llamar al LUIS
{
    if(message.Text != null)
//Asignamos directamente el Intent_LUIS como el message del usuario (porque
anteriormente ya hemos comprobado que coincidía en la BDD)
    {
        Intent_LUIS = message_string;
        Call_Basicos = false;
        LUIS_especifico = 1;
    }
    else //Formato incorrecto (foto, video etc...)
    {
        string message_error_2 = "Lo siento, no acepto este formato.
Introduce solo texto.";
        mensaje_bot = mensaje_bot + message_error_2;
        await context.PostAsync(message_error_2);
        Intent_LUIS = "None";
        Call_Basicos = false;
        LUIS_especifico = 0;
    }
}

if (Call_Basicos)
{

//DETECTAR LUIS BASICOS(SALUDAR, DESPEDIDA, INSULTAR, ETC...) mensajes generales.
try
{

//Intenta encontrar el intento en la tabla de "mensajes_Basicos" (Saludar,
despedida...)
        string CMD_LUIS_Basico = string.Format("SELECT mensaje FROM
Mensajes_Basicos WHERE LUIS='{0}'", Intent_LUIS);
        DataSet ds_LUIS_Basico = Utility.Ejecutar(CMD_LUIS_Basico);
        DataRow[] dataRow_LUIS_Basico =
ds_LUIS_Basico.Tables[0].Select();
        int Length_LUIS_Basico = dataRow_LUIS_Basico.Length;

```

```

        if (Length_LUIS_Basico == 0)
//Si no reconoce ningun mensaje basico, pasa a la siguiente tabla de Pregutnas y
respuestas
        {
            LUIS_especifico = 1;
        }
        else //En caso de encontrarlo
        {
            List<string> Mensaje_LUIS_Basico = new List<string>();
            int i = 0;

            foreach (DataRow element in dataRow_LUIS_Basico)
            {
Mensaje_LUIS_Basico.Add(dataRow_LUIS_Basico[i].ItemArray[0].ToString());
//Guarda el mensaje en una string
                i++;
            }
            Random random = new Random();
            int Random_Basico = random.Next(0,
Mensaje_LUIS_Basico.Count);

            if (Intent_LUIS == "Despedida")
            {
                string message_general_despedida =
Mensaje_LUIS_Basico[Random_Basico];
                mensaje_bot = mensaje_bot +
message_general_despedida;
                await context.PostAsync(message_general_despedida);

                preced = "0";
                intents_none = 0;
                string CMD_Insert = string.Format("UPDATE Usuarios
SET preced = '{0}', intent_none = {1} WHERE id_usuario='{2}'", preced,
intents_none, ID_User);
                Utility.Ejecutar(CMD_Insert);
            }
            else if (Intent_LUIS == "Desconocido")
            {
                string message_general_duda =
Mensaje_LUIS_Basico[Random_Basico];
                mensaje_bot = mensaje_bot + message_general_duda;
                await context.PostAsync(message_general_duda);

                string consulta = "";
                consulta = preced;
                string CMD_Pregunta_Duda = string.Format("SELECT
pregunta FROM LUIS_Preguntas WHERE identificador='{0}'", consulta);
                DataSet ds_CMD_Pregunta_Duda =
Utility.Ejecutar(CMD_Pregunta_Duda);
                string pregunta_duda =
ds_CMD_Pregunta_Duda.Tables[0].Rows[0]["pregunta"].ToString().Trim();

                string CMD_Opciones_Duda = string.Format("SELECT
LUIS, clickable_especial FROM LUIS_Respuestas WHERE precedente='{0}'", consulta);
                DataSet ds_CMD_Opciones_Duda =
Utility.Ejecutar(CMD_Opciones_Duda);
                DataRow[] dr_Opciones_Duda =
ds_CMD_Opciones_Duda.Tables[0].Select();

                List<string> Clickable_Botton_Duda = new
List<string>();

```

```

        string opciones_coma_duda = "";
        int k = 0;
        int m = 0;
        foreach (DataRow element in dr_Opciones_Duda)
        {
            if(dr_Opciones_Duda[k].ItemArray[1].ToString() ==
"True")
                {
                    Clickable_Botton_Duda.Add(dr_Opciones_Duda[k].ItemArray[0].ToString());
                    opciones_coma_duda = opciones_coma_duda +
dr_Opciones_Duda[k].ItemArray[0].ToString() + ",";
                    m++;
                }
                k++;
        }

        if (Clickable_Botton_Duda.Count == 0)
        {
            string message_duda = $"Ahora mismo puedo
ayudarte con la cuestión: {pregunta_duda}";
            mensaje_bot = mensaje_bot + message_duda;
            await context.PostAsync(message_duda);
        }
        else
        {
            string message_duda = $"Ahora mismo puedo
ayudarte con la cuestión: {pregunta_duda}";
            string message_logs_duda = message_duda + ":" +
opciones_coma_duda;

            mensaje_bot = mensaje_bot + message_logs_duda;
            await context.PostAsync(message_duda);

            reply =
Utility.ShowButtons_DeletedAnswer(context, "Elija una de las siguientes
opciones:", Clickable_Botton_Duda, m);
            await context.PostAsync(reply);
        }
    }
    else if (Intent_LUIS == "Ayuda")
    {
        string consulta = "";
        consulta = preced;
        string CMD_Pregunta_Ayuda = string.Format("SELECT
estado, pregunta FROM LUIS_Preguntas WHERE identificador='{0}'", consulta);
        DataSet ds_CMD_Pregunta_Ayuda =
Utility.Ejecutar(CMD_Pregunta_Ayuda);
        string estado_ayuda =
ds_CMD_Pregunta_Ayuda.Tables[0].Rows[0]["estado"].ToString().Trim();
        string preg_ayuda =
ds_CMD_Pregunta_Ayuda.Tables[0].Rows[0]["pregunta"].ToString().Trim();

        string CMD_Opciones_Ayuda = string.Format("SELECT
LUIS, clickable_especial FROM LUIS_Respuestas WHERE precedente='{0}'", consulta);
        DataSet ds_CMD_Opciones_Ayuda =
Utility.Ejecutar(CMD_Opciones_Ayuda);
        DataRow[] dr_Opciones_Ayuda =
ds_CMD_Opciones_Ayuda.Tables[0].Select();

        string estado_consulta = $"Estado: {estado_ayuda}\n";

```

```

        string salir = "Salir: En cualquier momento puede
salir y reiniciar la consulta con tan solo despedirse.\n";
        string ayuda = "Ayuda: Puede pedir ayuda y se le
mostrara esta información con las preguntas anteriores y la actual.\n";
        string pregunta = $"Pregunta actual: {preg_ayuda}\n";

        List<string> Clickable_Botton_Ayuda = new
List<string>();

        string opciones_coma_ayuda = "";
        int k = 0;
        int m = 0;
        foreach (DataRow element in dr_Opciones_Ayuda)
        {
            if (dr_Opciones_Ayuda[k].ItemArray[1].ToString()
== "True")
                {
                    Clickable_Botton_Ayuda.Add(dr_Opciones_Ayuda[k].ItemArray[0].ToString());
                    opciones_coma_ayuda = opciones_coma_ayuda +
dr_Opciones_Ayuda[k].ItemArray[0].ToString() + ",";
                    m++;
                }
            k++;
        }

        if (Clickable_Botton_Ayuda.Count == 0)
        {
            string respuestas = "Respuestas posibles: No
definidas.";

            string message_ayuda = estado_consulta +
Environment.NewLine + salir + Environment.NewLine + ayuda + Environment.NewLine +
pregunta + Environment.NewLine + respuestas;
            mensaje_bot = mensaje_bot + message_ayuda;
            await context.PostAsync(message_ayuda);
        }
        else if (Clickable_Botton_Ayuda[0] == "None")
        {
            string respuestas = "Respuestas posibles: Todas
aceptadas.";

            string message_ayuda = estado_consulta +
Environment.NewLine + salir + Environment.NewLine + ayuda + Environment.NewLine +
pregunta + Environment.NewLine + respuestas;
            mensaje_bot = mensaje_bot + message_ayuda;
            await context.PostAsync(message_ayuda);
        }
        else
        {
            string respuestas = "Respuestas posibles:";
            string message_ayuda = estado_consulta +
Environment.NewLine + salir + Environment.NewLine + ayuda + Environment.NewLine +
pregunta + Environment.NewLine + respuestas;
            string message_ayuda_log = message_ayuda + ":" +
opciones_coma_ayuda;

            mensaje_bot = mensaje_bot + message_ayuda_log;
            await context.PostAsync(message_ayuda);

            reply =
Utility.ShowButtons_DeletedAnswer(context, "Elija una de las siguientes
opciones:", Clickable_Botton_Ayuda, m);
            await context.PostAsync(reply);
        }
    }

```

```

    }
    else
    {
        string message_general =
Mensaje_LUIS_Basico[Random_Basico];
        mensaje_bot = mensaje_bot + message_general;
        await context.PostAsync(message_general);
    }
    LUIS_especifico = 0;
}
}
catch (Exception error)
{
    LUIS_especifico = 0;
    string message_error = "ERROR: Error con la tabla de Mensajes
Básicos. Contacte con un administrador.";
    mensaje_bot = mensaje_bot + message_error;
    mensaje_error = mensaje_error + error.ToString();
    await context.PostAsync(message_error);
}
}

//DETECTAR LUIS ESPECIFICOS (CANCELAR, ENTREGAR, ETC)
if (LUIS_especifico == 1)
{
    try
    {
        string CMD_LUIS_Especifico = string.Format("SELECT
precedente, siguiente, clickable FROM LUIS_Respuestas WHERE LUIS='{0}' AND
precedente='{1}'", Intent_LUIS, preced);
        DataSet ds_LUIS_Especifico =
Utility.Ejecutar(CMD_LUIS_Especifico);
        DataRow[] dr_LUIS_Especifico =
ds_LUIS_Especifico.Tables[0].Select();
        int Length_LUIS_Especifico = dr_LUIS_Especifico.Length;

        if (Length_LUIS_Especifico == 0)
        {
            intents_none += 1;
            if (intents_none >= 4)
            {
                preced = "0";
                intents_none = 0;

                string errores_maximos = "None_MAX";
                string CMD_None_MAX = string.Format("SELECT mensaje
FROM Mensajes_Basicos WHERE LUIS='{0}'", errores_maximos);
                DataSet ds_None_MAX = Utility.Ejecutar(CMD_None_MAX);
                DataRow[] dataRow_None_MAX =
ds_None_MAX.Tables[0].Select();

                List<string> Mensaje_None_MAX = new List<string>();
                int y = 0;

                foreach (DataRow element in dataRow_None_MAX)
                {
Mensaje_None_MAX.Add(dataRow_None_MAX[y].ItemArray[0].ToString());
                    y++;
                }
                Random random = new Random();

```

```

        int Random_None_MAX = random.Next(0,
Mensaje_None_MAX.Count);

        string message_error_max =
Mensaje_None_MAX[Random_None_MAX];
mensaje_bot = mensaje_bot + message_error_max;
await context.PostAsync(message_error_max);

        string CMD_Insert_reset = string.Format("UPDATE
Usuarios SET preced ='{0}', intent_none ={1} WHERE id_usuario='{2}'", preced,
intents_none, ID_User);
        Utility.Ejecutar(CMD_Insert_reset);
    }

else
    {
        string Consulta_None = string.Format("SELECT
respuestas FROM Mensajes_None");
        DataSet ds_None = Utility.Ejecutar(Consulta_None);
        DataRow[] dataRow_None = ds_None.Tables[0].Select();

        List<string> Mensajes_None = new List<string>();
        int i = 0;

        foreach (DataRow element in dataRow_None)
        {
Mensaje_None.Add(dataRow_None[i].ItemArray[0].ToString());
            i++;
        }
        Random random = new Random();
        int Random_None = random.Next(0,
Mensaje_None.Count);

        string mensaje_none_general =
Mensaje_None[Random_None];
mensaje_bot = mensaje_bot + mensaje_none_general;
await context.PostAsync(mensaje_none_general);

        string CMD_Pregunta_None = string.Format("SELECT
pregunta FROM LUIS_Preguntas WHERE identificador='{0}'", preced);
        DataSet ds_CMD_Pregunta_None =
Utility.Ejecutar(CMD_Pregunta_None);
        string preg_2 =
ds_CMD_Pregunta_None.Tables[0].Rows[0]["pregunta"].ToString().Trim();

        string CMD_LUIS_Especifico_3 = string.Format("SELECT
LUIS, clickable_especial FROM LUIS_Respuestas WHERE precedente='{0}'", preced);
        DataSet ds_LUIS_Especifico_3 =
Utility.Ejecutar(CMD_LUIS_Especifico_3);
        DataRow[] dr_Convenios_1 =
ds_LUIS_Especifico_3.Tables[0].Select();

        List<string> Clickable_Botton_None = new
List<string>();

        string opciones_coma = "";
        i = 0;
        int m = 0;
        foreach (DataRow element in dr_Convenios_1)
        {
            if (dr_Convenios_1[i].ItemArray[1].ToString() ==
"True")

```



```

        string precedente =
ds_LUIS_Especifico.Tables[0].Rows[0]["precedente"].ToString().Trim();
        string siguiente =
ds_LUIS_Especifico.Tables[0].Rows[0]["siguiente"].ToString().Trim();
        string click =
ds_LUIS_Especifico.Tables[0].Rows[0]["clickable"].ToString().Trim();

        string CMD_LUIS_Especifico_1 = string.Format("Select
identificador, accion_siguiente, pregunta, nombre_adjunto, documento_adjunto,
abrir_peticion FROM LUIS_Preguntas WHERE identificador= '{0}'",
ds_LUIS_Especifico.Tables[0].Rows[0]["siguiente"].ToString().Trim());
        DataSet ds_LUIS_Especifico_1 =
Utility.Ejecutar(CMD_LUIS_Especifico_1);

        DataRow[] dr_LUIS_Especifico_1 =
ds_LUIS_Especifico_1.Tables[0].Select();

        string identificador =
ds_LUIS_Especifico_1.Tables[0].Rows[0]["identificador"].ToString().Trim();
        string accion_siguiente =
ds_LUIS_Especifico_1.Tables[0].Rows[0]["accion_siguiente"].ToString().Trim();
        string Pregunta_Siguiente =
ds_LUIS_Especifico_1.Tables[0].Rows[0]["pregunta"].ToString().Trim();
        string Nombre_adjunto =
ds_LUIS_Especifico_1.Tables[0].Rows[0]["nombre_adjunto"].ToString().Trim();
        string Documento_adjunto =
ds_LUIS_Especifico_1.Tables[0].Rows[0]["documento_adjunto"].ToString().Trim();
        Abrir_Peticion
ds_LUIS_Especifico_1.Tables[0].Rows[0]["abrir_peticion"].ToString().Trim(); =

if (accion_siguiente == "0")
{
    mensaje_bot = mensaje_bot + Pregunta_Siguiente;
    await context.PostAsync(Pregunta_Siguiente);
    intents_none = 0;
    preced = "1000";

    string Chatbot_Util = string.Format("SELECT pregunta
FROM LUIS_Preguntas WHERE identificador='{0}'", preced);
    DataSet ds_Chatbot_Util =
Utility.Ejecutar(Chatbot_Util);
    string pregunta_Chatbot_Util =
ds_Chatbot_Util.Tables[0].Rows[0]["pregunta"].ToString().Trim();

    if (Nombre_adjunto == "")
    {
        mensaje_bot = mensaje_bot +
pregunta_Chatbot_Util;
        await context.PostAsync(pregunta_Chatbot_Util);
    }
    else
    {
        message_url = Utility.Show_URL_Card(context,
Nombre_adjunto, Documento_adjunto);
        mensaje_bot = mensaje_bot + Documento_adjunto;
        await context.PostAsync(message_url);

        mensaje_bot = mensaje_bot +
pregunta_Chatbot_Util;
        await context.PostAsync(pregunta_Chatbot_Util);
    }
}

```



```

        string CMD_Insert = string.Format("UPDATE Usuarios
SET preced='{0}', intent_none={1} WHERE id_usuario='{2}'", preced,
intents_none, ID_User);
        Utility.Ejecutar(CMD_Insert);
    }
    else if (accion_siguiente == "00")
    {
        mensaje_bot = mensaje_bot + Pregunta_Siguiente;
        await context.PostAsync(Pregunta_Siguiente);

        preced = "0";
        intents_none = 0;

        string CMD_Insert = string.Format("UPDATE Usuarios
SET preced='{0}', intent_none={1} WHERE id_usuario='{2}'", preced,
intents_none, ID_User);
        Utility.Ejecutar(CMD_Insert);
    }
    else if (identificador == siguiente)
    {
        preced = siguiente;
        string CMD_LUIS_Especifico_2 = string.Format("SELECT
LUIS, clickable FROM LUIS_Respuestas WHERE precedente='{0}'", preced);
        DataSet ds_LUIS_Especifico_2 =
Utility.Ejecutar(CMD_LUIS_Especifico_2);
        DataRow[] dr_Convenios =
ds_LUIS_Especifico_2.Tables[0].Select();

        string CMD_LUIS_Educado = string.Format("SELECT
mensaje FROM Mensajes_Basicos WHERE LUIS='{0}'", "Educación");
        DataSet ds_LUIS_Educado =
Utility.Ejecutar(CMD_LUIS_Educado);
        DataRow[] dataRow_LUIS_Educado =
ds_LUIS_Educado.Tables[0].Select();

        List<string> Mensaje_LUIS_Educado = new
List<string>();

        int i = 0;
        foreach (DataRow element in dataRow_LUIS_Educado)
        {
            Mensaje_LUIS_Educado.Add(dataRow_LUIS_Educado[i].ItemArray[0].ToString());
            i++;
        }
        Random random = new Random();
        int Random_Basico = random.Next(0,
Mensaje_LUIS_Educado.Count);
        string message_educado =
Mensaje_LUIS_Educado[Random_Basico];

        string texto_pregunta = message_educado +
Pregunta_Siguiente;

        await context.PostAsync(texto_pregunta);

        List<string> Clickable_Botton = new List<string>();
        i = 0;
        string opciones_coma = "";
        int m = 0;
        foreach (DataRow element in dr_Convenios)
        {

```

```

        if (dr_Convenios[i].ItemArray[1].ToString() ==
"True")
        {
Clickable_Botton.Add(dr_Convenios[i].ItemArray[0].ToString());
        opciones_coma = opciones_coma +
dr_Convenios[i].ItemArray[0].ToString() + ",";
        m++;
        }
        i++;
    }
    if (Clickable_Botton.Count == 0)
    {
        mensaje_bot = mensaje_bot + texto_pregunta;
    }
    else
    {
        string guardar_log = texto_pregunta + ":" +
opciones_coma;
        mensaje_bot = mensaje_bot + guardar_log;

        reply =
Utility.ShowButtons_DeletedAnswer(context, "Elija una de las siguientes
opciones:", Clickable_Botton, m);
        await context.PostAsync(reply);
    }

    string CMD_Insert = string.Format("UPDATE Usuarios
SET preced='{0}' WHERE id_usuario='{1}'", preced, ID_User);
    Utility.Ejecutar(CMD_Insert);
    }
}
}
catch (Exception error)
{
    string error_final = "FATAL ERROR: Error con el LUIS
Especifico. Contacte con un administrador.";
    mensaje_bot = mensaje_bot + error_final;
    mensaje_error = mensaje_error + error;
    await context.PostAsync(error_final);
}
}

try
{
    string CMD_Insert_log = string.Format("INSERT INTO
LOGS_Cajamar(id_usuario, oficina, mensaje_usuario, mensaje_bot, mensaje_error,
mensaje_LUIS, abrir_peticion, num_consulta, fecha_string, fecha) VALUES('{0}',
'{1}', '{2}', '{3}', '{4}', '{5}', '{6}', '{7}', '{8}', '{9}']", ID_User,
Oficina, mensaje_usuario, mensaje_bot, mensaje_error, mensaje_LUIS,
Abrir_Peticion, Num_Consultas, BarcelonaDateTime_string,
BarcelonaDateTime_FormatSQL);
    Utility.Ejecutar(CMD_Insert_log);
}
catch (Exception ex)
{
    string error_final = "";
    try
    {

```

```

        string CMD_Insert_error = string.Format("INSERT INTO
Errores_LOGS(id_usuario, mensaje_error, fecha) VALUES('{0}', '{1}', '{2}'),
ID_User, ex.ToString(), BarcelonaDateTime_FormatSQL);
        Utility.Ejecutar(CMD_Insert_error);

        error_final = "LOGS ERROR_0: Error al insertar LOGS a la base
de datos. Contacte con un administrador.";
    }
    catch (Exception error_2)
    {
        error_final = "LOGS ERROR_1: Error al insertar LOGS a la base
de datos. Contacte con un administrador.";
    }
    await context.PostAsync(error_final);
}

//string Proba_DT_2 = Prueba.ToString("yyyy-MM-dd HH:mm:ss");

//string CMD_LUIS_Especifico_2 = string.Format("SELECT fecha_datetime2 FROM
LOGS_Cajamar WHERE fecha_datetime2 < '{0}'", Proba_DT_2);
context.Wait(MessageReceivedAsync);
    }
}
}

```

## DB - MENSAJES NONE

DEFINICION:

```
CREATE TABLE [dbo].[Mensajes_None] (  
  
    [id] INT NOT NULL,  
  
    [respuestas] NVARCHAR (MAX) NULL,  
  
    CONSTRAINT [PK_Mensajes_None] PRIMARY KEY CLUSTERED ([id] ASC)  
  
);
```

DATOS:

```
INSERT INTO [dbo].[Mensajes_None] ([id], [respuestas]) VALUES (1, N'Lo siento, no te he entendido.  
Intenta ser más preciso por favor.')
```

```
INSERT INTO [dbo].[Mensajes_None] ([id], [respuestas]) VALUES (2, N' No te he entendido. ')
```

```
INSERT INTO [dbo].[Mensajes_None] ([id], [respuestas]) VALUES (3, N'Disculpa, pero no he entendido lo  
que has dicho.')
```

## DB - MENSAJES BIENVENIDA

DEFINICION:

```
CREATE TABLE [dbo].[mensaje_bienvenida] (  
  
    [id] INT IDENTITY (1, 1) NOT NULL,  
  
    [mensaje] NVARCHAR (MAX) NULL,  
  
    CONSTRAINT [PK_mensaje_bienvenida] PRIMARY KEY CLUSTERED ([id] ASC)  
  
);
```

DATOS:

```
SET IDENTITY_INSERT [dbo].[mensaje_bienvenida] ON
```

```
INSERT INTO [dbo].[mensaje_bienvenida] ([id], [mensaje]) VALUES (1, N'Bienvenido al chat del grupo de  
Cajamar.')
```

```
INSERT INTO [dbo].[mensaje_bienvenida] ([id], [mensaje]) VALUES (2, N'Esta conversación es totalmente confidencial.')
```

```
INSERT INTO [dbo].[mensaje_bienvenida] ([id], [mensaje]) VALUES (3, N'¿En qué puedo ayudarte?')
```

```
SET IDENTITY_INSERT [dbo].[mensaje_bienvenida] OFF
```

## **DB - USUARIOS I LOGS**

DEFINICION LOGS:

```
CREATE TABLE [dbo].[LOGS_Cajamar] (  
    [id] INT IDENTITY (1, 1) NOT NULL,  
    [id_usuario] NVARCHAR (50) NOT NULL,  
    [id_conversación] NVARCHAR (50) NULL,  
    [mensaje_usuario] NVARCHAR (MAX) NULL,  
    [mensaje_bot] NVARCHAR (MAX) NULL,  
    [mensaje_error] NVARCHAR (MAX) NULL,  
    [fecha_actual] NVARCHAR (50) NULL  
);
```

DEFINICION USUARIOS:

```
CREATE TABLE [dbo].[Usuarios] (  
    [id] INT IDENTITY (1, 1) NOT NULL,  
    [id_usuario] NVARCHAR (MAX) NOT NULL,  
    [preced] NVARCHAR (50) NOT NULL,  
    [intent_none] INT NOT NULL,  
    [fecha_actual] NVARCHAR (MAX) NULL);
```

## **DB - MENSAJES BÁSICOS**

DEFINICIÓN:

```

CREATE TABLE [dbo].[Mensajes_Basicos] (

    [id_mensaje] INT          IDENTITY (1, 1) NOT NULL,

    [mensaje] NVARCHAR (MAX) NULL,

    [LUIS] NVARCHAR (50) NULL,

    CONSTRAINT [PK_Mensajes_Basicos] PRIMARY KEY CLUSTERED ([id_mensaje] ASC)

);

DATOS:

SET IDENTITY_INSERT [dbo].[Mensajes_Basicos] ON

INSERT INTO [dbo].[Mensajes_Basicos] ([id_mensaje], [mensaje], [LUIS]) VALUES (1, N'Buenos días Sr/Sra. ¿En qué puedo ayudarte?', N'Saludar')

INSERT INTO [dbo].[Mensajes_Basicos] ([id_mensaje], [mensaje], [LUIS]) VALUES (2, N'Saludos Sr/Sra. ¿En qué puedo ayudarte?', N'Saludar')

INSERT INTO [dbo].[Mensajes_Basicos] ([id_mensaje], [mensaje], [LUIS]) VALUES (3, N'Bienvenido/a Sr/Sra. ¿En qué puedo ayudarte?', N'Saludar')

INSERT INTO [dbo].[Mensajes_Basicos] ([id_mensaje], [mensaje], [LUIS]) VALUES (4, N'Hola Sr/Sra. ¿En qué puedo ayudarte?', N'Saludar')

INSERT INTO [dbo].[Mensajes_Basicos] ([id_mensaje], [mensaje], [LUIS]) VALUES (5, N'Puedes llamarme Luis! ¿En qué puedo ayudarte?', N'BotNombre')

INSERT INTO [dbo].[Mensajes_Basicos] ([id_mensaje], [mensaje], [LUIS]) VALUES (6, N'Mi nombre es Luis! ¿Que puedo hacer por ti?', N'BotNombre')

INSERT INTO [dbo].[Mensajes_Basicos] ([id_mensaje], [mensaje], [LUIS]) VALUES (7, N'Un placer ayudarte, hasta otra!', N'Despedita')

INSERT INTO [dbo].[Mensajes_Basicos] ([id_mensaje], [mensaje], [LUIS]) VALUES (8, N'Que tengas un buen día!', N'Despedita')

INSERT INTO [dbo].[Mensajes_Basicos] ([id_mensaje], [mensaje], [LUIS]) VALUES (9, N'Adiós! No dudes en contactarme si necesitas mi ayuda.', N'Despedita')

INSERT INTO [dbo].[Mensajes_Basicos] ([id_mensaje], [mensaje], [LUIS]) VALUES (10, N'Puedo ayudarte con la cancelación, amortización y entrega de préstamos.', N'Ayuda')

```

INSERT INTO [dbo].[Mensajes\_Basicos] ([id\_mensaje], [mensaje], [LUIS]) VALUES (16, N'No hay de qué! Ya sabes donde encontrarme.', N'Gracias')

INSERT INTO [dbo].[Mensajes\_Basicos] ([id\_mensaje], [mensaje], [LUIS]) VALUES (17, N'Gracias a ti, solo hago mi trabajo.', N'Gracias')

INSERT INTO [dbo].[Mensajes\_Basicos] ([id\_mensaje], [mensaje], [LUIS]) VALUES (18, N'Muy amable por tu parte, no dudes en contactarme para futuras consultas.', N'Gracias')

INSERT INTO [dbo].[Mensajes\_Basicos] ([id\_mensaje], [mensaje], [LUIS]) VALUES (21, N'Si tienes alguna duda, consulta tu préstamo para seguir adelante.', N'Duda')

INSERT INTO [dbo].[Mensajes\_Basicos] ([id\_mensaje], [mensaje], [LUIS]) VALUES (22, N'Me han dicho cosas peores y gente mucho mejor que tú.', N'Insultar')

INSERT INTO [dbo].[Mensajes\_Basicos] ([id\_mensaje], [mensaje], [LUIS]) VALUES (23, N'¡Increíble! Realmente estás dando lo mejor de ti.', N'Insultar')

INSERT INTO [dbo].[Mensajes\_Basicos] ([id\_mensaje], [mensaje], [LUIS]) VALUES (24, N'Esperaba una conversación racional, pero parece que no hay nadie alrededor para tenerla.', N'Insultar')

INSERT INTO [dbo].[Mensajes\_Basicos] ([id\_mensaje], [mensaje], [LUIS]) VALUES (25, N'Creo que la conversación está desvariando...', N'Insultar')

INSERT INTO [dbo].[Mensajes\_Basicos] ([id\_mensaje], [mensaje], [LUIS]) VALUES (26, N'Voy a obviar esto, sigamos con el proceso.', N'Insultar')

INSERT INTO [dbo].[Mensajes\_Basicos] ([id\_mensaje], [mensaje], [LUIS]) VALUES (30, N'Cuida ese vocabulario o tendré que lavarte la boca con jabón.', N'Insultar')

INSERT INTO [dbo].[Mensajes\_Basicos] ([id\_mensaje], [mensaje], [LUIS]) VALUES (31, N'Gracias. ', N'Educación')

INSERT INTO [dbo].[Mensajes\_Basicos] ([id\_mensaje], [mensaje], [LUIS]) VALUES (32, N'Muchas gracias. ', N'Educación')

INSERT INTO [dbo].[Mensajes\_Basicos] ([id\_mensaje], [mensaje], [LUIS]) VALUES (33, N'De acuerdo. ', N'Educación')

INSERT INTO [dbo].[Mensajes\_Basicos] ([id\_mensaje], [mensaje], [LUIS]) VALUES (34, N'Entendido. ', N'Educación')

INSERT INTO [dbo].[Mensajes\_Basicos] ([id\_mensaje], [mensaje], [LUIS]) VALUES (35, N'Perfecto. ', N'Educación')

```
INSERT INTO [dbo].[Mensajes_Basicos] ([id_mensaje], [mensaje], [LUIS]) VALUES (37, N'Has realizado demasiados intentos erróneos seguidos, ponte en contacto con un administrador, yo vuelvo al inicio.', N'None_MAX')
```

```
INSERT INTO [dbo].[Mensajes_Basicos] ([id_mensaje], [mensaje], [LUIS]) VALUES (39, N'En el caso que no sepa responder a la pregunta, consulta tu préstamo correctamente para poder seguir adelante.', N'Duda')
```

```
INSERT INTO [dbo].[Mensajes_Basicos] ([id_mensaje], [mensaje], [LUIS]) VALUES (40, N'Encantado de conocerte. ¿Te puedo ayudar en algo?', N'Saludar')
```

```
INSERT INTO [dbo].[Mensajes_Basicos] ([id_mensaje], [mensaje], [LUIS]) VALUES (41, N'Para poder responder a la pregunta es necesario que consultes correctamente el préstamo.', N'Duda')
```

```
INSERT INTO [dbo].[Mensajes_Basicos] ([id_mensaje], [mensaje], [LUIS]) VALUES (42, N'Para seguir adelante con el proceso, debes responder la siguiente pregunta, de lo contrario, tendrás que consultar el préstamo para seguir adelante.', N'Duda')
```

```
INSERT INTO [dbo].[Mensajes_Basicos] ([id_mensaje], [mensaje], [LUIS]) VALUES (43, N'Gracias a ti por confiar en mí! Si tienes alguna otra consulta no dudes en contactar conmigo.', N'Gracias')
```

```
INSERT INTO [dbo].[Mensajes_Basicos] ([id_mensaje], [mensaje], [LUIS]) VALUES (44, N'Okey. ', N'Educación')
```

```
INSERT INTO [dbo].[Mensajes_Basicos] ([id_mensaje], [mensaje], [LUIS]) VALUES (45, N'Genial. ', N'Educación')
```

```
SET IDENTITY_INSERT [dbo].[Mensajes_Basicos] OFF
```