UNIVERSITAT POLITÈCNICA DE VALÈNCIA

MIARFID
Máster en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Máster en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital
Universitat Politècnica de València

Desarrollo de un chatbot para comprar entradas de cine o reservar pistas de bolos

Reconocimiento Automático del Habla

Autor: Jacobo López Fernández
Juan Antonio López Ramírez

Curso 2019-2020

Índice general

CAPÍTULO 1
Introducción

El objetivo ha consistido en ampliar las prácticas que realizamos los integrantes de este trabajo. Para ello, hemos juntado los dos dominios que se realizaron en su momento (Compra de entradas de cine y reserva de pista de bolos) y se le han añadido unas funcionalidades para que el cliente pueda hacer ambas acciones.

De esta forma, se facilita la interacción entre el usuario y el bot para mejorar su experiencia con el servicio ofrecido.

Aunque las prácticas se hicieron con la herramienta *Dialogflow* de Google, decidimos implementar nuestro nuevo chatbot con una consola alternativa proporcionada por la empresa **Aunoa Software**, debido a las comodidades que presenta en comparación con *Dialogflow*.

Figura 1.1: Logo de Aunoa Software.

CAPÍTULO 2
Diseño y desarrollo del chatbot

Nuestro bot va a recibir peticiones del cliente, que tendrá que satisfacer mediante la ejecución de lo que se conoce como *Intents* (en nuestra consola también adoptan la denominación de estados) o reacciones del bot ante una determinada entrada.

El flujo de diálogo, a modo de prototipo, se puede apreciar en la figura 2.1. Como podemos observar, el bot empezará la conversación preguntando al usuario qué desea hacer, si ir al cine a ver una película o jugar una partida de bolos. En función de lo que le conteste, el bot tomará un camino u otro, haciendo distinción entre los dos dominios.

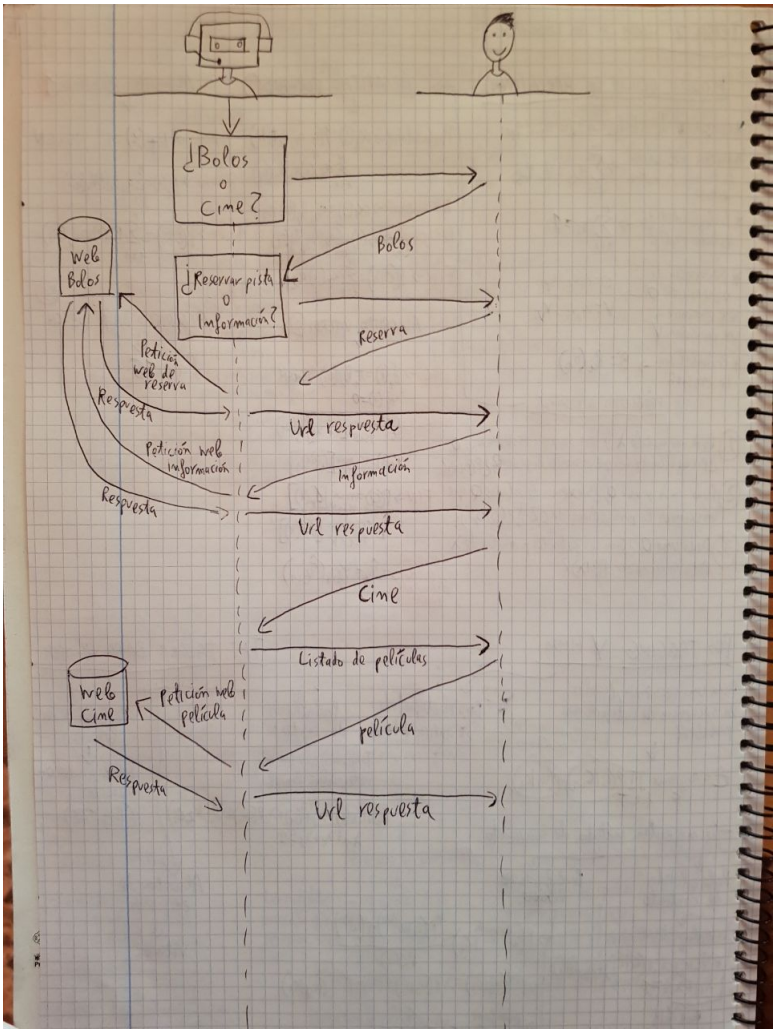

Figura 2.1: Prototipo del diagrama de flujo del chatbot.

Si el usuario escoge ver una película, el bot le mostrará una lista de las películas que se encuentran en la cartelera de los cines de la empresa Cinesa (concretamente de Aldaia), y quedará a elección del cliente la película a seleccionar. En este caso, el flujo de diálogo del bot será dirigido, y el usuario deberá de pulsar en los botones que el sistema le proporcione en función de la película que quiera ver.

Si, por otro lado, el usuario decide jugar a los bolos, el bot le mostrará dos opciones:

- **Reservar pista**. En la que se enviará al usuario a la página web que se encarga de reservar una pista de bolos.
- **Información**. En la que el bot mandará al cliente a la web que recopila información acerca de esta bolera: Horarios disponibles, precios, ubicación, etc.

Es importante aclarar que, si el usuario responde al bot con una de las dos opciones en lugar de pulsar sus respectivos botones, el bot se encargará de mostrarle al cliente el camino a seguir para informarse del sitio o reservar una partida. En ambos casos, el bot le redirigirá a la página web de Bowling Chamartín, la bolera que se ha seleccionado.

2.1 Flujo de diálogo

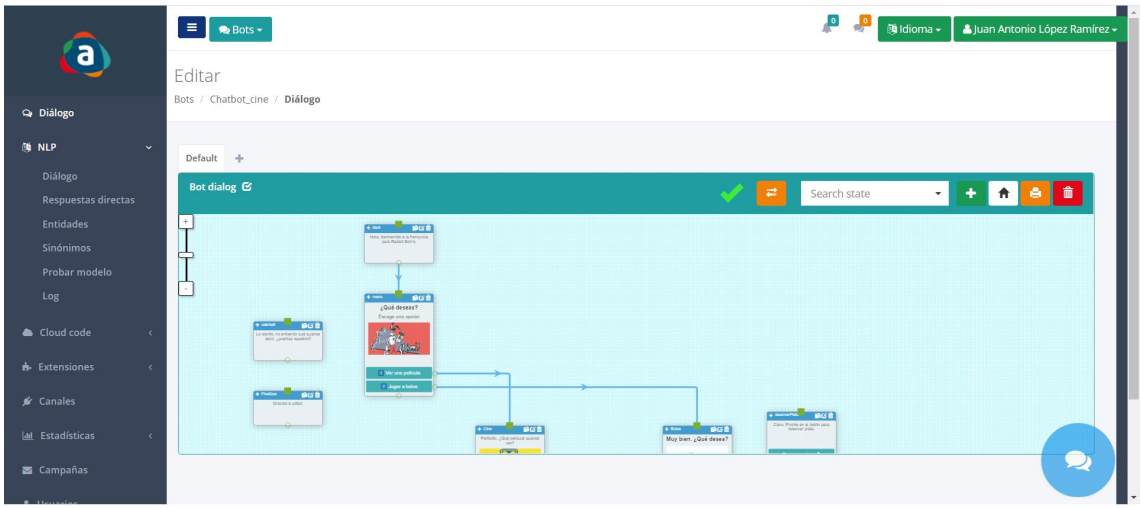

Figura 2.2: Flujo de diálogo.

Para añadir al sistema los *Intents* o estados, se emplea el *Bot Dialog* que proporciona la consola de Aunoa (véase la figura 2.2). De esta manera, podemos observar gráficamente el flujo de diálogo de nuestro bot, una ventaja con la que no se contaba en *Dialogflow*.

Como se observa en la figura 2.3, podemos configurar el estado a través de las cuatro pestañas que proporciona:

- **Diseño**. Establecemos el nombre del *Intent*, su tipo, las variables (si es que queremos añadir alguna) y el estado que se ejecutará después del actual. En *Contenido* se define la respuesta que el bot proporcionará al usuario, mientras que en *Acciones* se especificará la tarea que llevará a cabo el bot.
- **Quicks**. Es similar a las *Acciones* del Diseño, pero aquí hay un límite de 11, mientras que en la otra solo se podían añadir 3.

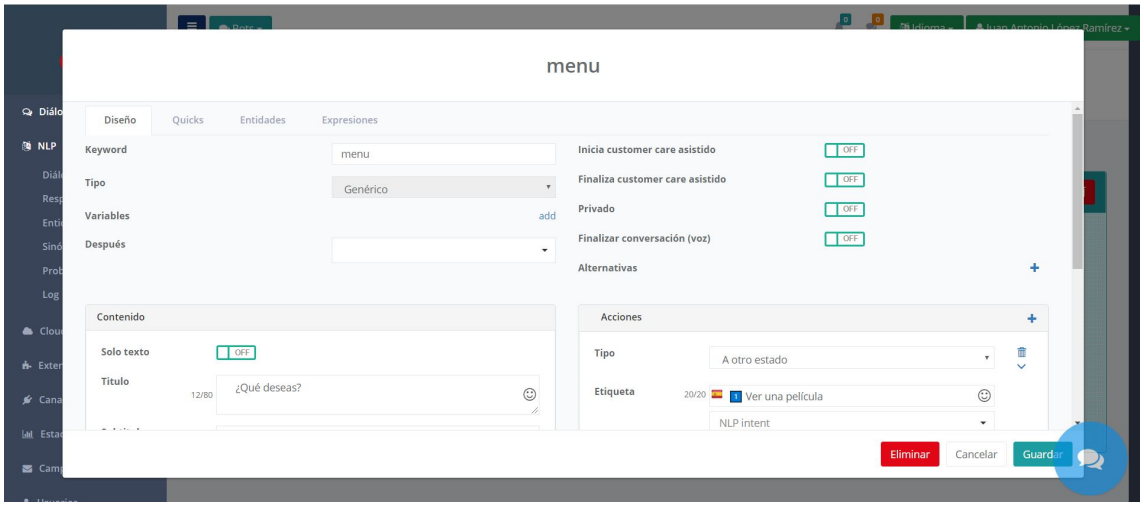

Figura 2.3: Ejemplo de estado donde se puede observar la pestaña de Diseño.

- **Entidades**. Las entidades que el *Intent* ha de detectar de la respuesta del usuario.
- **Expresiones**. Respuestas que el cliente proporciona al bot para que se ejecute el estado actual. Como veremos más adelante, también se pueden modificar en la parte de NLP.

A grandes rasgos, nos hemos centrado en la parte de Diseño y Expresiones cuando creamos y definimos los *Intents*.

Se han creado 7 estados:

- **/start**. El inicial. Se activa por defecto al principio del diálogo, o cuando el usuario saluda. Está configurado para que, inmediatamente después, se ejecute el *Intent* de menú. Su nombre no es casual, ya que de esta forma es como se activa el diálogo del bot al exportarlo a Telegram, pero de eso hablaremos más adelante.
- **menu**. Se activa justo después de /start, y en él el bot da a elegir al usuario si quiere ir al cine o jugar a los bolos. Se le ha añadido una imagen, a modo de decoración, con el nombre de la franquicia ficticia que tiene los cines y la bolera.
- **Cine**. Se activa si el usuario ha seleccionado la opción correspondiente en menú, o si le ha escrito al bot que su intención es ir a ver una película. Aquí se pregunta al cliente cuál quiere ver, y después se activa el *Intent* de Cartelera.
- **Cartelera**. Es un estado especial, puesto que se trata de un *Cloud Function*. Este tipo de *Intents* se caracterizan por realizar una función predefinida en Javascript. En nuestro caso, se realiza un scraping de la página web de *Filmaffinity* que contiene los horarios y la cartelera de las películas que se estén proyectando en Cinesa Aldaia. A este estado se puede acceder preguntándole al bot por la cartelera actual.
- **Bolos**. Este *Intent* introduce al usuario en el dominio de los bolos. También, de forma similar al de Cine, el usuario puede acceder si le escribe que quiere jugar a bolos, en lugar de pulsar el botón en el menú. Aquí hay dos botones: Reservar Pista e Información, este último para que la web a la que se redirija al usuario le informe de los horarios, precios, ubicación de la bolera, etc. Se ha añadido una imagen de decoración con el logo de Bowling Chamartín.
- **catchall**. Es el estado de error. Se lleva a cabo cuando el bot no entiende lo que le ha dicho el cliente. El bot le pide al usuario si puede repetir su consulta.

- **Finalizar**. Tiene un papel casi insignificante, pues solo se activa si el usuario agradece el servicio del bot.

Además de estos 7 *Intents*, hay definidos otros 2 auxiliares (1 para reservar pista y otro para información de la bolera) para el caso en que el usuario le escriba las correspondientes peticiones que el bot no puede clasificar de forma individual desde Bolos.

En la figura 2.4 se puede apreciar el flujo de diálogo final implementado para nuestro bot.

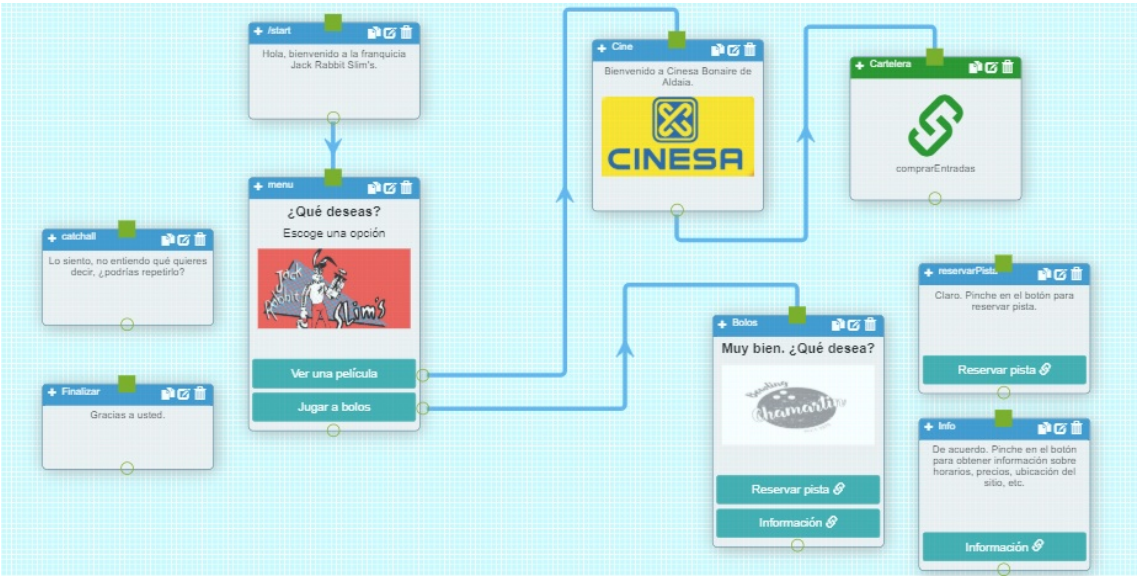

Figura 2.4: Flujo de diálogo final.

2.2 NLP

Esta es la parte donde entrenamos nuestro bot. Para ello, debemos determinar qué estado responde nuestro bot en función de qué dice el usuario.

Podemos apreciar en la figura 2.5 que disponemos de 6 apartados dentro de esta parte (Aunque principalmente nos hemos centrado en Diálogo, Probar modelo y Log):

- **Diálogo**. Asignamos las expresiones que dice el usuario con los estados que se deberán ejecutar. En la figura 2.5 vemos algunos ejemplos de expresiones que se han empleado para ciertos estados. Estas expresiones son las mismas que se encuentran en la parte del Flujo de diálogo.

- **Respuestas directas**. Es similar al anterior, pero aquí el bot no pasa a ningún estado, simplemente contesta una respuesta predefinida. A modo de curiosidad, se ha implementado que el bot conteste una respuesta soez a cualquier insulto que pueda recibir por parte del usuario, o que responda con su nombre si se lo han pedido.

- **Entidades**. Cumple la misma función que los *Entities* de Dialogflow. En nuestro caso, nos hemos necesitado definir ninguna entidad.

- **Sinónimos**. Definimos una palabra y una lista de sus sinónimos. Si el usuario responde con uno de estos sinónimos, el bot lo entenderá como si fuese la palabra original.

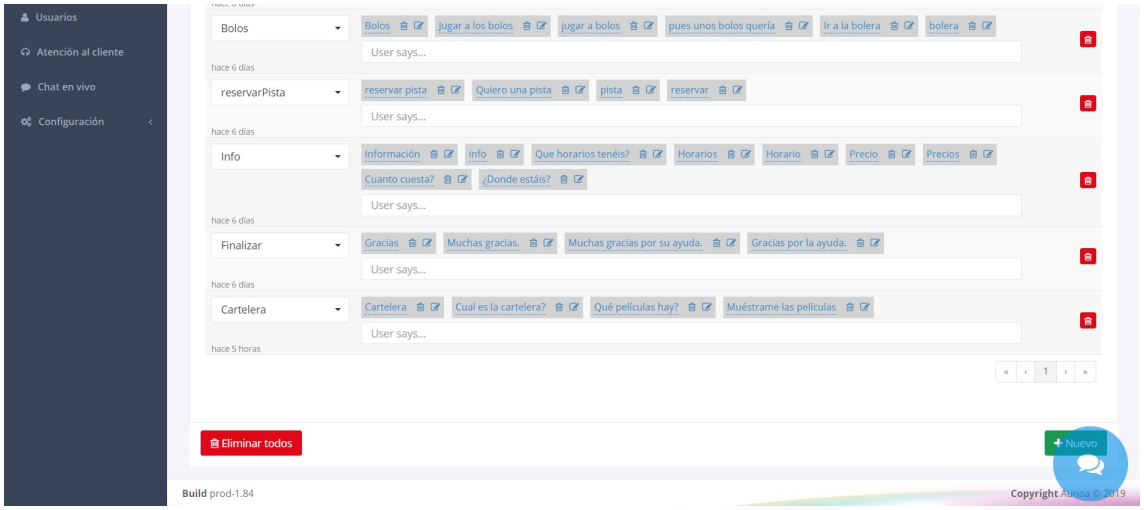

Figura 2.5: Ejemplos de expresiones utilizadas para la clasificación en *Intents* de nuestro bot.

- **Probar modelo**. En este apartado escribimos una frase y el sistema devolverá el estado en el que, con un cierto porcentaje de confianza, entrará el bot. Como se puede apreciar en la figura 2.6, hemos probado la frase 'Pues desearía información sobre la bolera' y el sistema lo clasifica en el *Intent* Info con un 55.322 % de confianza, lo cuál es correcto. Podemos añadir esta frase a las expresiones de ese estado pulsando en 'Añadir al modelo'. Si pulsamos sobre 'Validar modelo', el sistema irá probando, de manera automática, frases para comprobar la calidad de las respuestas del bot (si el lenguaje es el correcto y si el *Intent* al que se clasifica una determinada expresión no es el correcto). Una prueba de esta validación se puede observar en la figura 2.8.

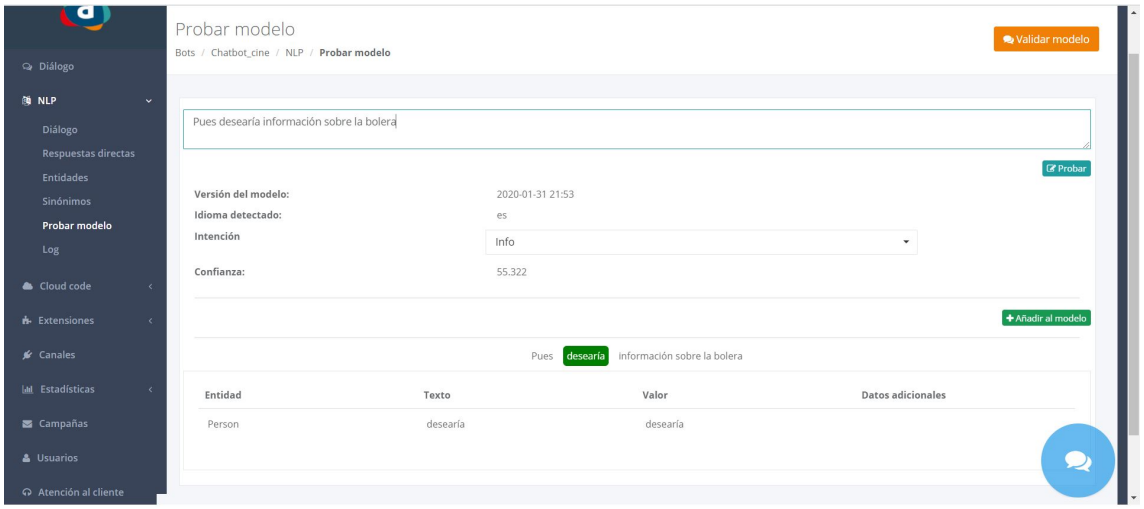

Figura 2.6: Probar modelo.

- **Log**. En este apartado se muestran los resultados del último entrenamiento del modelo. Podemos escoger el estado en el que se han clasificado las expresiones y el nivel de confianza de esta clasificación. En el ejemplo de la figura **??**, se pueden apreciar las expresiones que, para el estado catchall, han obtenido una confianza del 0 %, que han sido las de 'ver una peli' y 'ver una película', lo cuál es incorrecto pues se tendría que haber clasificado en el *Intent* de Cine. Esto se puede cambiar en el desplegable que hay debajo de 'Responder con'.

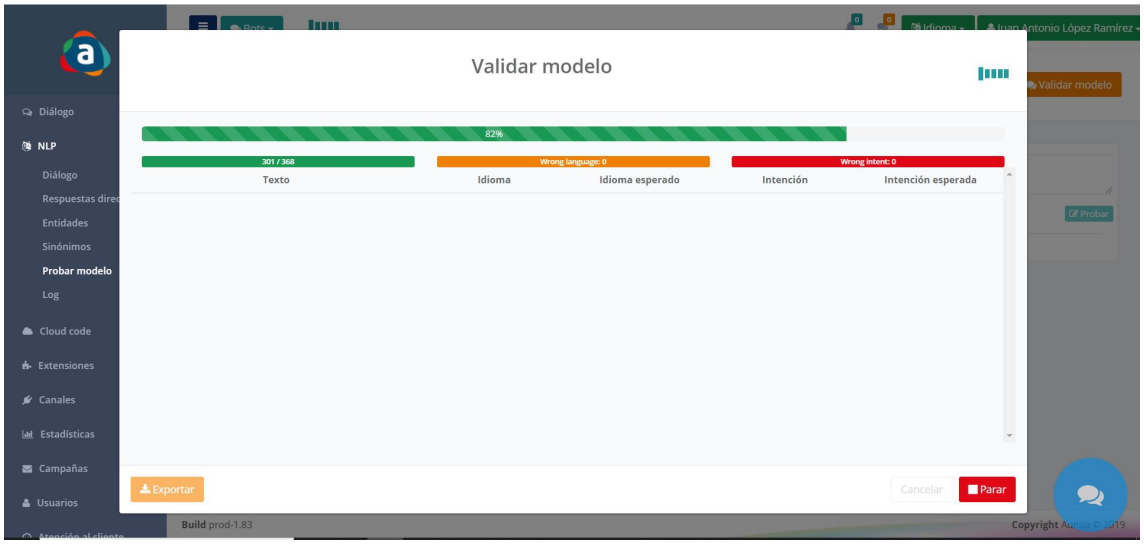

Figura 2.7: Validación del modelo.

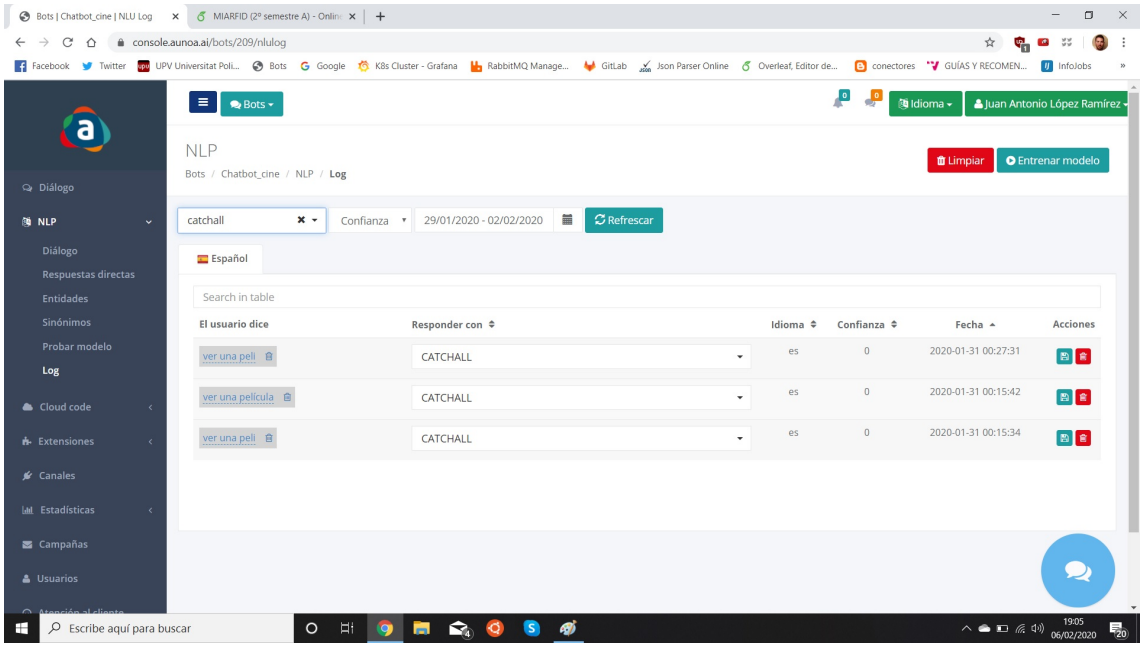

Figura 2.8: Ejemplo de resultado del log para el estado catchall.

2.3 Cloud code

En esta parte se definen las funciones en Javascript que pueden llegar a ser útiles en el diálogo del bot con el cliente. En nuestro caso, se ha implementado una función **comprarEntradas**, como se puede apreciar en la figura 2.9 y el código completo de esta es:

```
1    const fetch = require('node-fetch');
2    const cheerio = require('cheerio');
3    var data=await fetch("https://www.filmaffinity.com/es/theater-showtimes.php
         ?id=291");
4    var html=await data.text();
5    const $ = cheerio.load(html);
6    var total_items=$('li.fa-shadow > a');
```

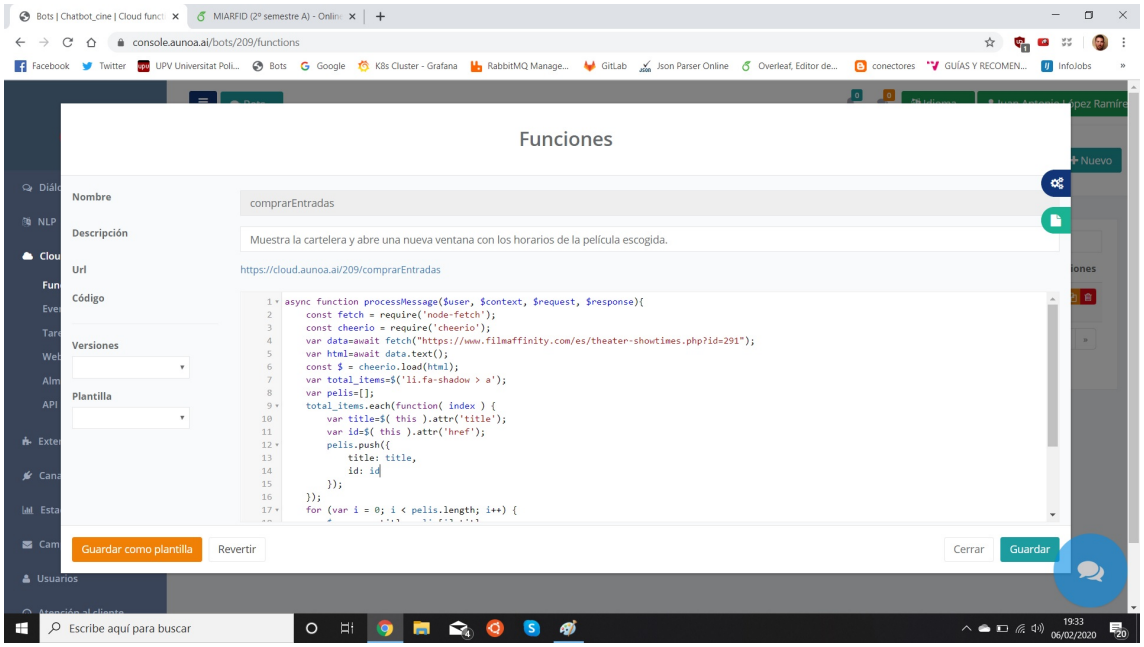

Figura 2.9: Función de Cloud code.

```
    var pelis=[];
    total_items.each(function( index ) {
        var title=$( this ).attr('title');
        var id=$( this ).attr('href');
        pelis.push({
            title: title,
            id: id
        });
    });
    for (var i = 0; i < pelis.length; i++) {
        $response.title=pelis[i].title;
        var a=new Action();
        a.label="Ver horarios";
        a.link="https://www.filmaffinity.com/es/theater-showtimes.php?id=291"+(
            pelis[i].id);
        $response.addAction(a);
        $response.send(1*i);
}
```

El objetivo de esta función es mostrarle al usuario la cartelera actual de Cinesa de Aldaia. Para ello, hemos hecho uso de la API Fetch, que proporciona una interfaz JavaScript para acceder y manipular partes del canal HTTP. La hemos aprovechado para recuperar el código en HTML de la web de *Filmaffinity*. Además de esta, hemos utilizado la API Cheerio para hacer un Web Scraping y obtener, del HTML recuperado, las películas de la cartelera, junto con un identificador que permite obtener la parte de la página web donde se especifican los horarios de la película en cuestión.

Cuando ya tenemos todas las películas, le enviamos al usuario (con una diferencia de un segundo) una respuesta con los títulos de la películas y un botón que contiene el enlace a los horarios de cada una de ellas, gracias a los identificadores que previamente habíamos guardado.

Finalmente, esta función la incluimos dentro del *Intent* Cartelera, en la parte de Flujo de diálogo, como se puede observar en la figura 2.10.

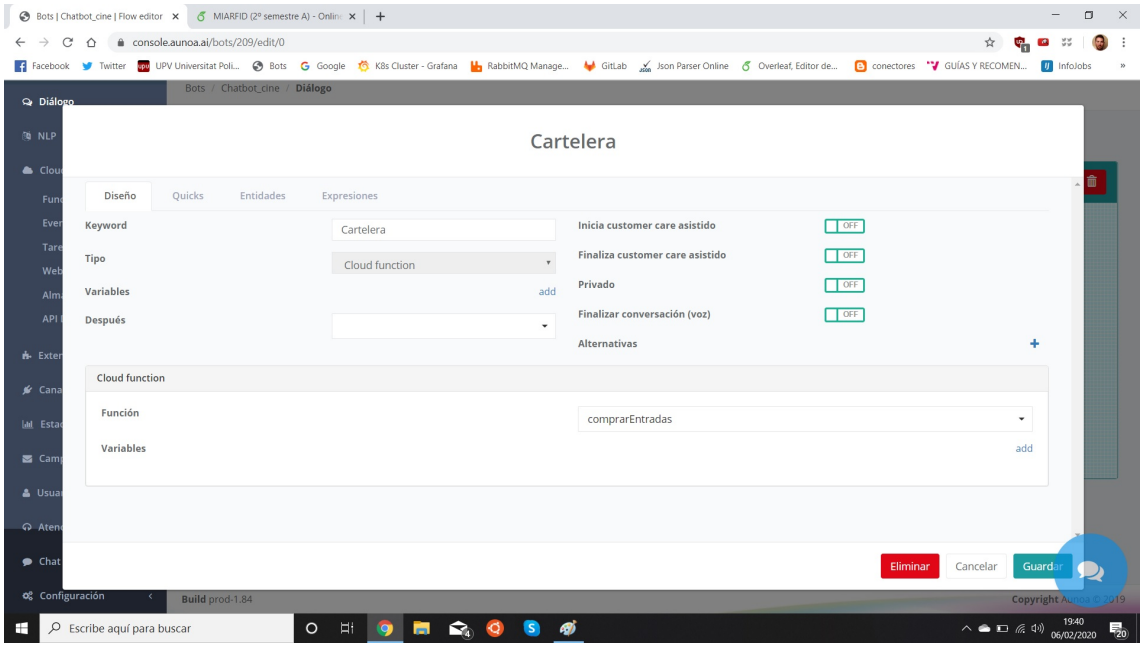

Figura 2.10: Estado de Cartelera como una *Cloud function*, que implementa la función de comprarEntradas.

CAPÍTULO 3
Despliegue y Pruebas

Para iniciar los servicios de nuestro bot y probar su funcionamiento, primeramente hemos de seleccionar un canal de los que tenemos disponibles (se pueden observar en la figura 3.1).

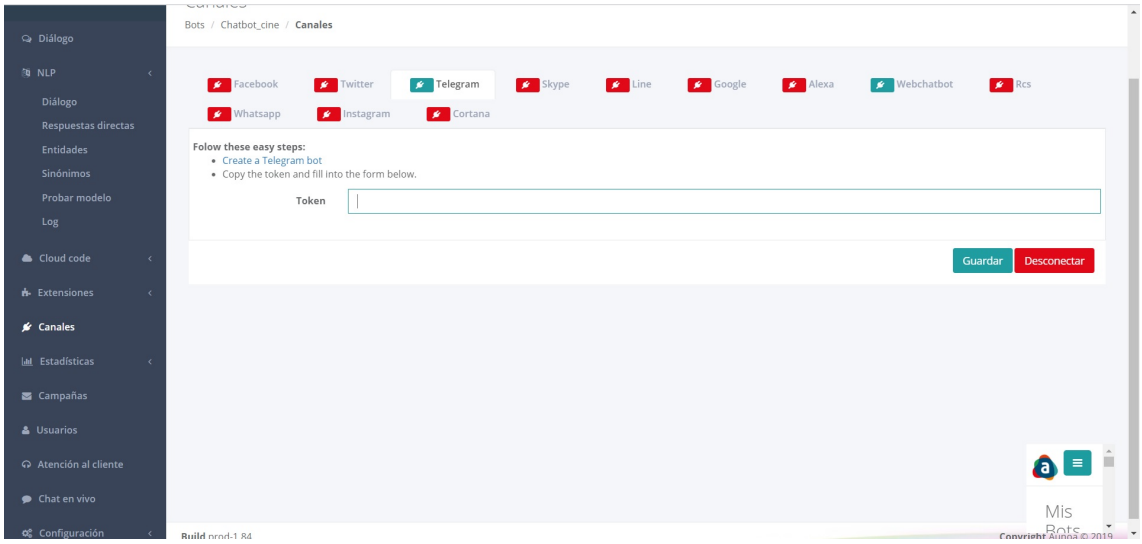

Figura 3.1: Canales disponibles para desplegar nuestro bot. Concretamente, hemos seleccionado Telegram.

En nuestro caso, hemos decidido desplegarlo en Telegram. Para ello, antes que nada debemos crear un bot en Telegram mediante **@BotFather**, una cuenta que no deja de ser más que un chatbot administrador.

A este le contestamos con un /newbot, nos pedirá un nombre para el bot (en nuestro caso le hemos llamado **Bot_cine_bolos** con alias **@Cinebolosbot**) y nos proporcionará su token, que debemos colocar en el campo vacío de la figura 3.1 y luego guardar. Este proceso se puede apreciar en la figura 3.2.

Una vez desplegado, se han realizado pruebas de funcionamiento para demostrar que el bot funciona como debe. Iniciamos la conversación con /start y recibimos el mensajes de bienvenida con instrucciones de uso (figura 3.3).

A partir de aquí podemos separar las pruebas en los dos dominios que tenemos.

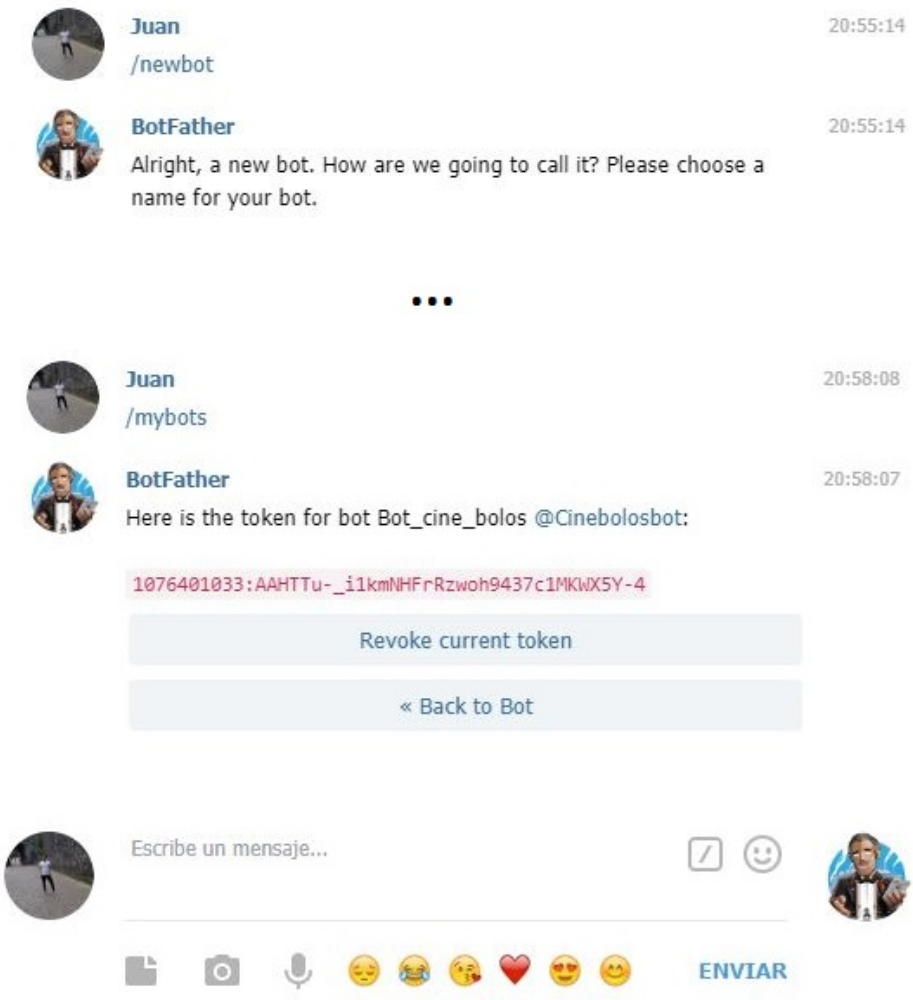

Figura 3.2: Proceso de creación de nuestro chatbot en Telegram.

3.1 Pruebas en el dominio de cine

Hemos probado a que se nos muestre la cartelera por dos vías: Pulsando el botón de 'Ver una película' y con texto (figura 3.4 y 3.5).

Si pinchamos en 'Ver horarios' de una película nos sale el mensaje de confirmación de la figura 3.6 y se nos abre una pestaña con los horarios de la película tal y como se aprecia en la figura 3.7. Pinchando en uno de ellos accedemos a comprar las entradas.

3.2 Pruebas en el dominio de bolos

Al seleccionar 'Jugar a bolos' (figura 3.8) o escribiéndolo (figura 3.9), el bot entiende que el usuario desea, o bien reservar pista, o bien información sobre la bolera.

Se ha probado el *Intent* auxiliar reservarPista con texto, el bot lo ha reconocido sin problemas y muestra un mensaje para poder acceder a la página web pertinente (figura 3.10 y 3.11).

De la misma forma, se ha hecho lo propio con Info, obteniendo resultados igualmente satisfactorios (figura 3.12 y 3.13).

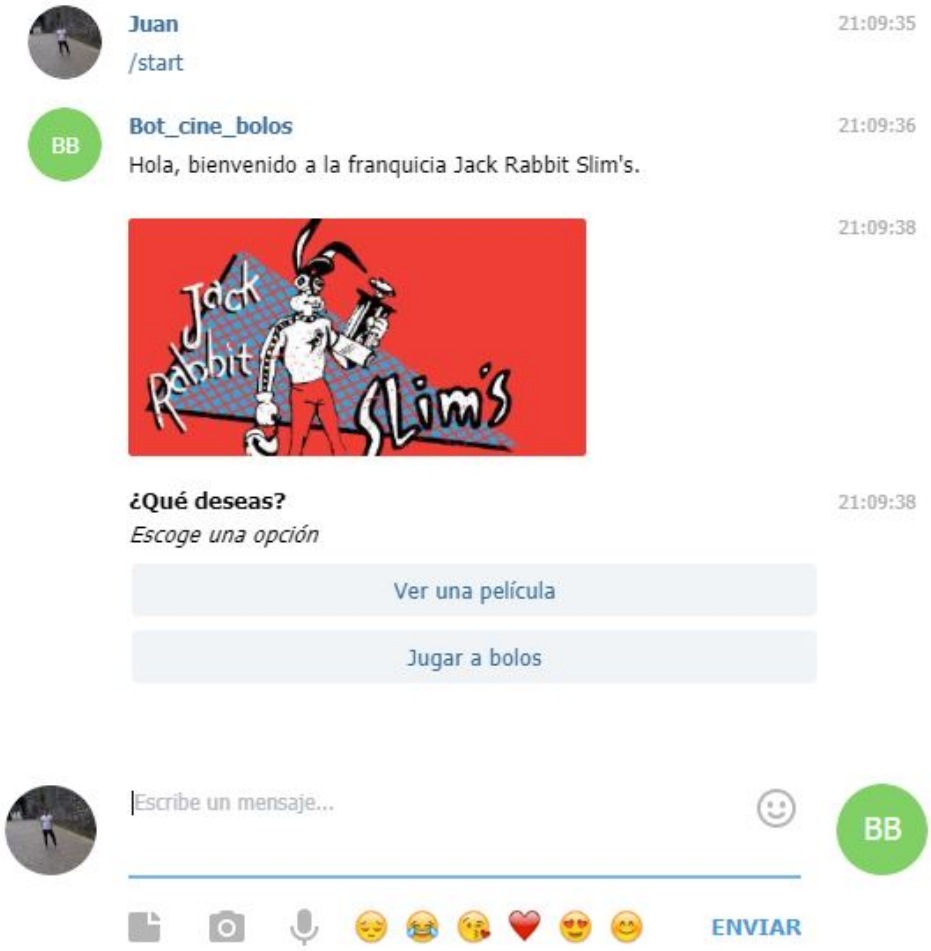

Figura 3.3: Prueba /start.

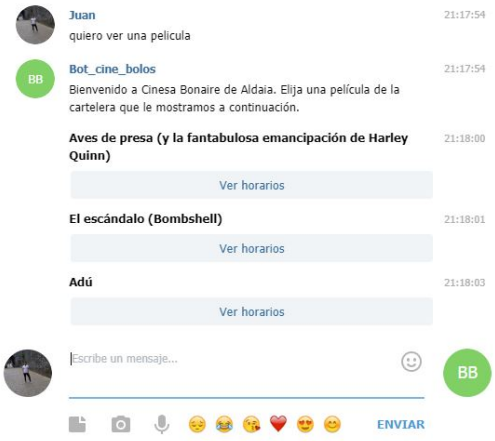

Figura 3.4: Prueba del estado Cine con texto.

Figura 3.5: Prueba del estado Cartelera con texto.

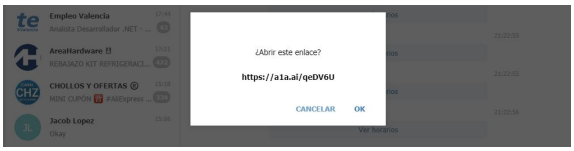

Figura 3.6: Mensaje de confirmación para abrir el enlace en una pestaña nueva.

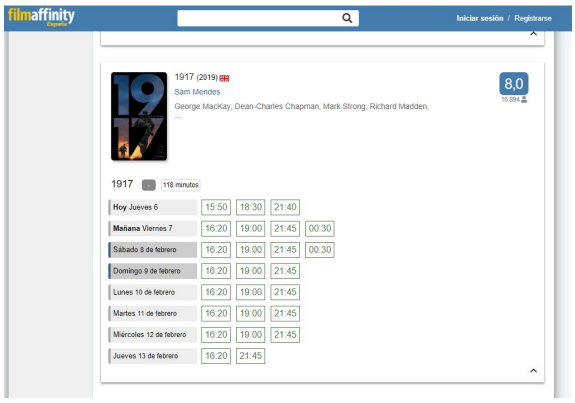

Figura 3.7: Página web con los horarios de la película seleccionada. Pinchando en uno de ellos accedemos a comprar las entradas.

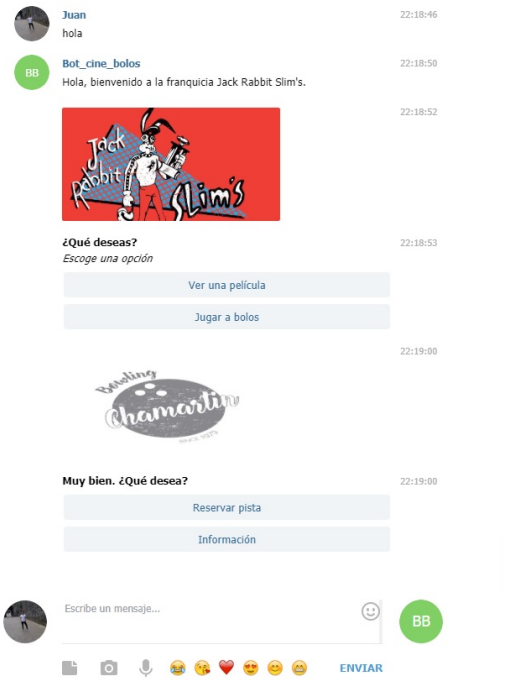

Figura 3.8: Prueba del estado Bolos con el botón.

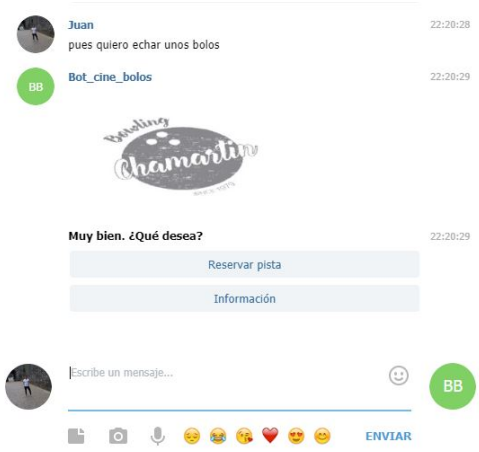

Figura 3.9: Prueba del estado Bolos con texto.

Figura 3.10: Prueba del estado reservarPista.

Figura 3.11: Página web de Bowling Chamartín para reservar.

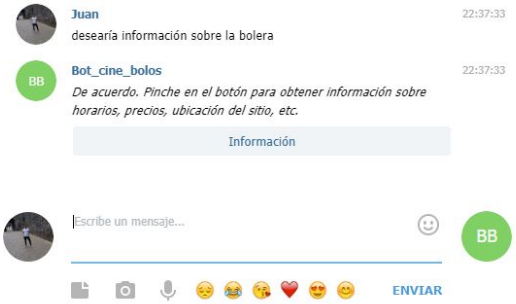

Figura 3.12: Prueba del estado Info.

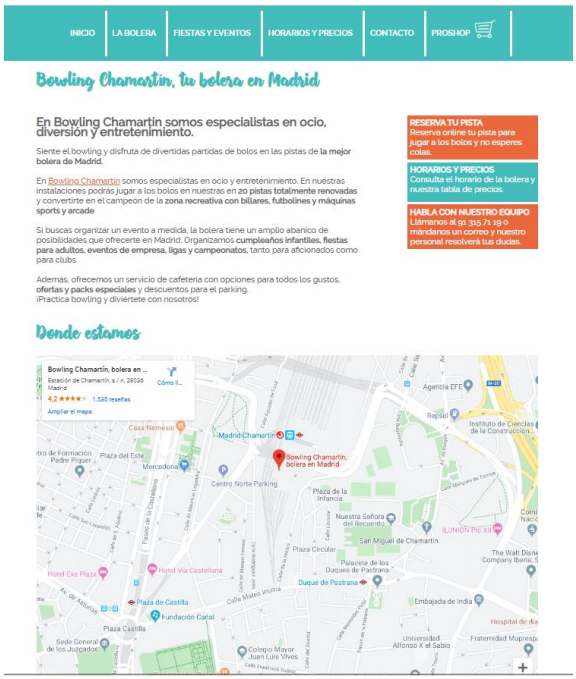

Figura 3.13: Página web de Bowling Chamartín para información.

CAPÍTULO 4
Conclusiones

Los bots conversacionales es una tecnología que ahora mismo se encuentra en alza. Con la mejora constante en el campo de la inteligencia artificial (en este caso del procesamiento del lenguaje natural) y el uso masivo de internet y de aplicaciones de teléfonos móviles, probablemente, cada vez sea más común interactuar con un bot.

Nuestro bot Bot_cine_Bolos entra en esta etapa como un servicio de compra de entradas para el cine y/o reserva de pistas de la bolera totalmente a mano y fácil de utilizar, cumpliendo los humildes objetivos que se determinaron antes de su desarrollo.

Mediante la realización de este proyecto se han adquirido muchos conocimientos técnicos, tanto por las tecnologías utilizadas, como por la labor informativa necesaria para la elaboración de este documento.