Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN

Escuela Politécnica Superior de Jaén

Trabajo Fin de Grado

DESARROLLO DE UNA APLICACIÓN PARA ALEXA: ATENCIÓN AL ALUMNO DE LA UJA

Alumno: Luis Montecatine Beltrán

Tutores: Prof. D. Miguel Ángel García Cumbreras

Prof. Dña. Salud María Jiménez Zafra

Dpto: Departamento de Informática

Septiembre, 2019

UNIVERSIDAD DE JAÉN

Miguel Ángel García Cumbreras y Salud María Jiménez Zafra, tutores del Trabajo Fin de Grado del Grado en Ingeniería Informática, titulado Desarrollo de una aplicación para Alexa: atención al alumno de la UJA", que presenta Luis Montecatine Beltrán, autoriza su presentación para defensa y evaluación.

Jaén, a 5 de septiembre de 2019

El alumno/a

MONTECATINE BELTRAN LUIS - 77366066T

Firmado digitalmente por MONTECATINE BELTRAN LUIS - 77366066T
Fecha: 2019.09.05 09:51:06 +02'00'

Luis Montecatine Beltrán

Los tutores

GARCIA CUMBRERAS MIGUEL ANGEL - 52871272E

JIMENEZ ZAFRA SALUD MARIA - 77370897R

Miguel Ángel García Cumbreras
Salud María Jiménez Zafra

Índice

Resumen

Este proyecto describe el desarrollo una aplicación, o Skill, para el asistente virtual Alexa, de Amazon.

La aplicación se ha desarrollado con la idea de poder responder verbalmente a las preguntas relacionados con el ámbito de la secretaría de la Universidad de Jaén.

Más concretamente está preparada para responder a las preguntas frecuentes relacionadas con las secciones de Acceso y Matrícula.

Capítulo 1

Introducción

1. Introducción

1.1. Motivación

Este proyecto nace del actual auge de la tecnología ubicua. Sobre todo, si nos centramos en los asistentes virtuales, comprobamos que no solo están cada día en más casas, si no, que la mayoría de la población, ya lleva alguno preinstalado en su dispositivo móvil.

Debido al potencial que presentan y la versatilidad que le aportan a todo tipo de usuarios, se convierte una tecnología realmente competitiva. Solo con fijarnos en las grandes empresas del mercado, vemos que muchas tienen ya en desarrollo su propio proyecto de asistente virtual.

Pese a lo que nos presentan a día de hoy, se trata de una tecnología que está lejos de realmente de mostrar su potencial real y que seguro nos irán sorprendiendo día a día.

1.2. Propósito

El desarrollo de este proyecto plantea un acercamiento a los asistentes para ver que son capaces de hacer a día de hoy.

Además, podremos comprobar el potencial que pueden desarrollar si se utilizan de forma conjunta con otras tecnologías, como en este caso, con un sistema de recuperación de información.

Además, profundizando más en la temática escogida, se nos presenta la posibilidad de aprovechar las opciones que nos brinda la informática para agilizar el trabajo humano. Pudiendo incluso imaginar una ventanilla en secretaría que fuera capaz de estar gestionada únicamente por un asistente virtual, siendo capaz de solucionar dudas del alumnado, disminuyendo así, los tiempos de espera.

1.3. Objetivos

Los objetivos principales de este proyecto son:

- Realizar el proceso completo de desarrollo de un proyecto de Ingeniería Informática.
- Conocer el mundo de los asistentes virtuales.
- Descubrir cómo desarrollar una aplicación con Amazon Alexa.
- Comprender cómo funciona un sistema de recuperación de información y aprender a implementarlo en Python.
- Aplicar un sistema de recuperación a un asistente virtual.
- Documentar todo el proceso llevado a cabo, en una memoria de TFG.

1.4. Resultados esperados

Los resultados esperados tras el desarrollo de este proyecto son:

- Conseguir una comunicación fluida con el usuario.
- Conseguir un sistema usable e intuitivo para todo tipo de usuarios.
- Entender cómo desarrollar Skills para Amazon Alexa.
- Implementar un sistema de recuperación de información con la eficiencia apropiada para esta aplicación.
- Comprobar si es efectivo montar un sistema de recuperación de información con modelo vectorial sobre un asistente.

1.5. Estructura del proyecto

La estructura de la memoria de este proyecto se divide en 8 capítulos, la bibliografía y 3 anexos. Describiendo brevemente cada capítulo:

Capítulo 1. Introducción al proyecto. Se explica la motivación, propósitos, objetivos y resultados esperados del proyecto.

Capítulo 2. Descripción del mundo de los asistentes virtuales, veremos qué son y para qué sirven, qué funcionalidades presentan y cuáles son los más punteros del mercado.

Capítulo 3. Capítulo introductorio para conocer qué es y cómo funciona un sistema de recuperación de información.

Capítulo 4. En este capítulo realizaremos un primer análisis del problema, descubriremos la solución planteada y estimaremos los costes de desarrollo.

Capítulo 5. Se realizará un diseño general de la solución escogida y diseñaremos cómo debemos plantear el proyecto para su posterior desarrollo.

Capítulo 6. Apartado dedicado a explicar cómo se ha desarrollado e implementado la solución propuesta.

Capítulo 7. Capítulo referente a las pruebas realizadas, tanto de usabilidad del sistema, como del rendimiento obtenido durante todo el desarrollo del proyecto.

Capítulo 8. En esta parte de la memoria se explicarán los resultados obtenidos y se sacan conclusiones de si ha sido una buena o mala decisión abarcarlo con esta metodología. Además se plantea lo necesario para la puesta en marcha del sistema.

BIBLIOGRAFÍA. Aquí se recogen todas las referencias y recursos empleados en el desarrollo del proyecto.

ANEXO I. En este primer anexo se describe todo el proceso necesario para poner en marcha una Skill para Amazon Alexa, utilizando Python 3.6, y apoyándome sobre un ejemplo para que los conceptos sean más fáciles de comprender.

ANEXO II. En este apartado se recogen todas las preguntas de la colección. De esta forma pueden probarse para obtener unas respuestas fiables.

ANEXO III. Se describe la puesta en marcha del proyecto a partir de los archivos entregados junto a la memoria.

Capítulo 2

Asistentes Virtuales

2. Asistentes Virtuales

Desde su primera aplicación real en 1994, por el primer teléfono inteligente de IBM, los asistentes virtuales son una fuente de innovación con alto potencial.

Un asistente virtual a día de hoy es un agente de software capaz de interaccionar entre una persona y una máquina, propiciando una comunicación fluida entre ambos, con el fin de simplificar tareas humanas.

2.1. Aplicación

La aplicación de estos asistentes a día de hoy tiene un número muy reducido de dispositivos, que son:

- Dispositivos móviles.
- Ordenadores.
- Relojes inteligentes.
- Pequeños dispositivos distribuidos por casa u oficina.

Además, los asistentes permiten varios métodos diferentes de interacción:

- Texto
- Voz
- Imágenes

El futuro de los asistentes presenta una gran expansión en el ámbito de internet de las cosas, previendo que cada vez sea más habitual encontrarlos en objetos cotidianos.

2.2. Funcionalidades

Las funcionalidades varían ligeramente según el sistema que elijamos, pero por lo general el ámbito de todos ellos es común.

Los métodos de interacción de estos dispositivos son mediante; voz, texto o fotos.

Según un estudio[1] de Adobe Digital Insights en 2018, el 70% los usuarios de altavoces inteligentes lo utilizan para reproducir música, y un 64% para pedir la meteorología. El mismo estudio prevé que en este 2019 uno de cada 4 usuarios de internet en China y Estados Unidos contarán con un altavoz inteligente. Los sectores que más se prevé que crezcan serán los juegos grupales y los servicios de comida a domicilio.

Aunque cada uno tiene sus propias funcionalidades y aplicaciones, aquí listo una muestra de lo que son capaces de hacer:

- Trabajar con aplicaciones de música, audiolibros, vídeos.
- Responder a preguntas simples, como el tiempo, la hora.
- Búsqueda de respuestas.
- Gestionar eventos de agendas, recordatorios y alarmas.
- Controlar un hogar inteligente, gestionando luces.
- Gestionar tareas domésticas de tipo poner temporizador o por ejemplo gestionar cestas de la compra.

[1] https://www.nobbot.com/futuro/asistentes-virtuales-2019/

2.3. Amazon Alexa

Amazon en 2014 lanzó su asistente virtual Alexa, siendo este un servicio de voz en la nube de Amazon disponible en todos los dispositivos que posteriormente ha ido sacando al mercado Amazon, estando entre ellos, por ejemplo, los altavoces cada día más comunes; Dot y Echo.

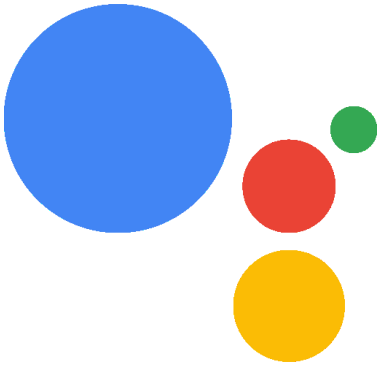
En el presente Amazon Alexa cuenta con una gran expansión en el sector de la domótica contando con la colaboración de empresas como Phillips o TP-Link, las cuales lanzan al mercado multitud de dispositivos con funcionalidades compatibles con el asistente.

2.4. Competidores actuales

Asistente	Desarrollador	Lanzamiento	Dispositivo
Alexa	Amazon	noviembre de 2014	Amazon Echo/Dot
Bixby	Samsung	abril de 2017	Samsung Galaxy S8
Cortana	Microsoft	abril de 2014	Invoke
Google Assistant	Google	mayo de 2016	Google Home
Siri	Apple	octubre de 2011	HomePod / iOS

Tabla 2-1. Lista de los asistentes más importantes actualmente

2.4.1. Google Assistant.



Google Assistant[2] aparece de la evolución de Google Now en 2016, introduciendo como gran diferencia la inclusión de diálogos bidireccionales con el usuario. Nació siendo una funcionalidad de los smartphones Pixel, de Google, para posteriormente liberarse al sistema operativo Android a principios del año 2017 y pasar a ser uno de los asistentes más usados en el presente.

Su dominio de aplicación incluye soporte para smartphones, altavoces inteligentes, wearables, televisores y pantallas inteligentes.

Actualmente se encuentra junto con Alexa en la carrera tecnológica por hacerse con el puesto del mejor asistente del mercado. Lo que supone para el usuario un lanzamiento constante de nuevas funcionalidades y servicios.

A día de hoy cuenta con una gran comunidad de desarrolladores y multitud de aplicaciones de terceros. Si a esto, además le añadimos los precios cada día más competitivos de sus altavoces, podemos prever que es una muy buena opción para iniciarse en el mundo de los asistentes.

2.4.2. Siri

Siri[3] pese a ser un asistente y competir con el resto en muchos aspectos, no sigue del todo los mismos estándares.

Este asistente viene integrado en los dispositivos de Apple, por tanto, cuenta con un amplio despliegue en cuanto

[2] https://assistant.google.com/intl/es_es/
[3] https://www.apple.com/es/siri/

a número de dispositivos hablamos. Además, en 2018 Apple lanzó su altavoz bajo el nombre de HomePod con un precio de salida, bastante superior al de sus competidores, de 350€.

Pese a ser uno de los asistentes más potentes y versátiles del mercado, si lo comparamos con el resto, observamos que en cuanto a lo que prestaciones y funcionalidades no presenta nada innovador, lo que no propicia que se haga un hueco competitivo en el mercado actual que cada día innova más en el internet de las cosas.

2.4.3. Cortana

Cortana[4] es el nombre que recibe el asistente desarrollado por Microsoft. Este asistente está presente en todos los ordenadores con Windows 10 y los móviles y tablets con el sistema operativo Windows phone.

Las herramientas que implementa son:

- Gestión recordatorios
- Avisos según localización
- Avisos o recordatorios con el uso de imágenes.
- Búsqueda de respuestas
- Abrir o cerrar aplicaciones

Como observamos el dominio de Cortana es bastante más limitado que el de sus competidores ya que ni siquiera utiliza aplicaciones de desarrolladores externos.

[4] https://www.microsoft.com/es-es/windows/cortana

2.4.4. Bixby

Bixby[5] es la alternativa lanzada por Samsung, orienta su funcionamiento en sus móviles que al menos usen Android 7.0 “Nougat”, televisores y neveras. Pusieron a la venta un altavoz inteligente, pero no tuvo nada de éxito.

Una gran apuesta de Samsung a partir del Galaxy 8 fue incluir un botón físico dedicado específicamente a lanzar el asistente.

Nació con la idea de poder realizar las mismas funciones que actualmente se hacen tocando, se puedan realizar en lenguaje natural. Pero la realidad es que actualmente están lejos de conseguir algo siquiera parecido a eso.

El único punto fuerte que Samsung explotó en su inicio fue el potencial de usar un sistema de recuperación de información sobre las fotos que realizamos, con lo que por ejemplo puedes conectar directamente con la compra de elementos en Amazon. Pero la realidad es que rápidamente la competencia implementó esta funcionalidad.

En este 2019, podemos afirmar que Bixby está muy por detrás del resto de sus competidores, pero aun así logra ser competente.

[5] https://www.samsung.com/es/apps/bixby/

2.5. ¿Por qué Alexa?

Pese a que Siri es el claro competidor a batir en cuanto hablamos del número de dispositivos del que dispone en el mercado, los dispositivos con Alexa, en el pasado 2018, se han hecho un hueco muy importante, posicionando sus ventas en cifras similares a las de Google Home. Contando Amazon con 2.2 millones de altavoces inteligentes vendidos frente a los 2.3 millones de dispositivos Google Home.

Google Home es realmente el competidor directo de Amazon Alexa, ya que ofrece los servicios más parecidos.

Viendo los números observamos que realmente ambos son asistentes punteros y además cuentan con el respaldo de dos de los gigantes de la tecnología, así que no es fácil decantarse por uno.

El factor por el que me decanté al inicio de este proyecto, fue que Amazon hacía dos meses que había lanzado el soporte para el idioma en español, y por tanto el mercado apenas tenía desarrolladores ni contenido para su dispositivo en los países de habla hispana.

Esta me parecía una buena razón para escoger a Alexa, ya que teniendo en cuenta el potencial que tiene, conocer una tecnología puntera y que poca gente estudia podría abrirme muchas puertas en el futuro.

Capítulo 3

Sistemas de Recuperación de Información

3. Sistemas de recuperación de información

Los sistemas de recuperación de información a lo largo de su historia han sido definidos de múltiples formas, dos de las más famosas son:

"La Recuperación de Información trata con la representación de información, el almacenamiento, la organización y el acceso a ítems de información", Baeza-Yates (1999).

"La Recuperación de la Información tiene que ver con la representación, almacenamiento, organización y acceso a los items de información", Salton (1983)

Existen múltiples modelos, pero los más extendidos son:

- Booleano
    - Fue el primero y es aún bastante usado para determinados problemas.
    - Está basado en la concordancia exacta de los términos de la consulta.
    - La consulta y los documentos se tratan como un conjunto de términos.
    - La recuperación se basa en si los documentos contienen o no los términos de la consulta.

- Vectorial
    - Sigue una metodología más avanzada que el booleano.
    - Ordena los documentos según la similitud con la consulta.

  - Es dependiente también de encontrar los términos de la consulta en los documentos.
  - Se le pueden aplicar mejoras en el tratamiento de la consulta y la colección para mejorar su efectividad y hacerlo realmente competente.

- Probabilístico
  - Ordena según la probabilidad de importancia de los documentos.
  - Evita la comparación exacta de términos de la consulta en los documentos de la colección.
  - Su implementación es mucho más compleja que la de los otros modelos.

A la hora de decidir por qué modelo decantarme y teniendo en cuenta lo estudiado en la asignatura del grado, Sistemas de recuperación de información.

Decidí que quizás lo más adecuado era intentar aplicar un modelo probabilístico.

La contra de este, es que la implementación iba a suponer una carga demasiado grande de trabajo y para no opacar el resto del proyecto, ya que el desarrollo del sistema de recuperación no es el tema central del proyecto, decidí optar por la opción de aplicar el modelo vectorial.

Este modelo fue al que más tiempo le dedicamos en la asignatura antes mencionada y en la cual, pudimos comprobar que realmente daba un buen rendimiento a cambio de dar un tiempo de desarrollo alto, pero no excesivo, lo que me pareció lo más adecuado para las dimensiones de este proyecto.

Este modelo está basado en el álgebra vectorial y permite un sistema de ranking.

Utiliza un sistema de pesos calculados a partir de dos valores:

- TF (Term Frequency): Frecuencia del término.
- IDF (Inverse Document Frequency): Inversa de la frecuencia documental.

La metodología a seguir será el cálculo de pesos de documentos para posteriormente calcular la similitud de cada documento con una consulta dada.

En la siguiente imagen se describe el esquema de funcionamiento básico de un sistema de recuperación de información.

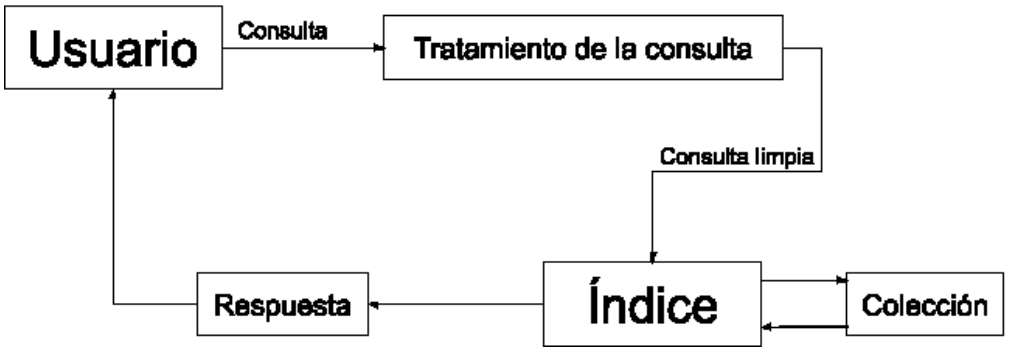


Ilustración 3-1. Esquema básico de funcionamiento de un SRI

Capítulo 4

Análisis detallado y solución planteada

4. Análisis detallado y solución planteada

4.1. Perfiles de usuario

Todo estudiante universitario antes o después se ha cuestionado alguna duda de carácter administrativo.

El alumnado en este caso puede verse en el problema de tener que ir a secretaría a resolver su duda.

El verdadero problema reside cuando descubre que el horario de atención es solo durante la mañana, lo que puede ser un gran inconveniente si trabaja, vive en otra localidad o tiene clases durante dicho horario.

La solución a esto puede ser buscar dicha información en la página de la universidad, pero... ¿Por qué no preguntarle directamente en lenguaje natural a tu dispositivo móvil u ordenador?

Bajo esa premisa, nace el proyecto, con la idea de unificar las preguntas del alumnado y poder obtener una respuesta a consultas en lenguaje natural.

Dado el reciente desarrollo de los asistentes virtuales, se nos plantea la posibilidad de llevarlo a cabo.

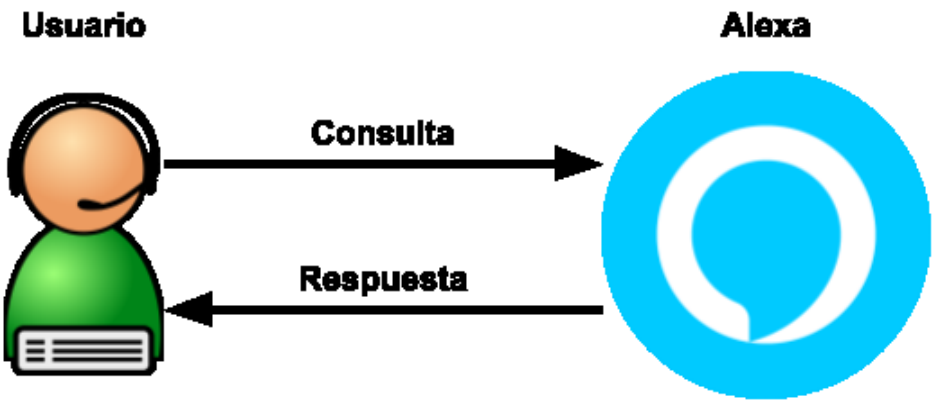


Ilustración 4-1. Modelo mental de la interacción

4.2. Estimación temporal

Tarea	Duración	Comienzo	Finalización
Estudio de los Asistentes virtuales	10	04/02/19	13/02/19
Estudio del desarrollo de Skills	15	16/02/19	03/03/19
Estudio de los sistemas de recuperación de información	9	01/03/19	09/03/19
Generación de la colección	8	10/03/19	17/03/19
Desarrollo de la Skill	56	18/03/19	12/05/19
Desarrollo del sistema de recuperación de información	68	02/04/19	16/06/19
Pruebas y mejoras	55	05/05/19	08/08/19
Generación de manuales	13	17/07/19	13/08/19
Generación de la memoria	185	10/03/19	01/09/19
Total	200	04/02/19	01/09/19

Tabla 4-1. Estimación temporal del proyecto completo

Diagrama de Gantt:

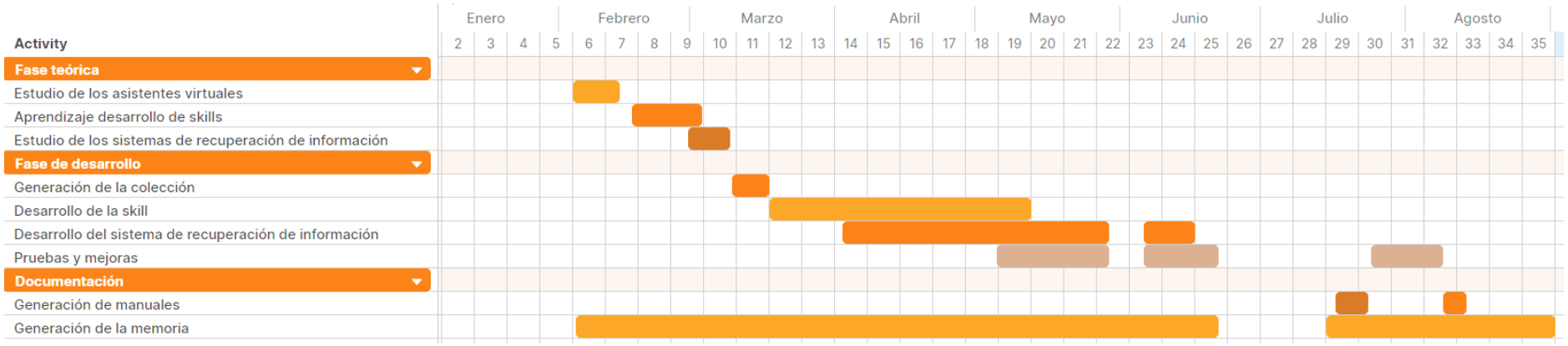


Ilustración 4-2 Diagrama de Gantt realizado con la herramienta Tom's Planner

Como comentario adicional, añadir a este diagrama que las etapas sin desarrollo corresponden a las épocas de exámenes y de vacaciones.

Estos días no están contabilizados en el coste temporal de desarrollo.

4.3. Estimación de coste

Para la realización de este proyecto solo estimo que se necesite una persona que haga de analista y de programador.

Con un salario medio de unos 1000€ al mes, a jornada parcial.

Considerando que se desarrolle una buena colección. El tiempo de desarrollo completo del proyecto de forma continua lo estimo en unos 6 meses.

En cuanto a material, se necesita un ordenador, que no necesita excesiva potencia y sería conveniente al menos 2 altavoces Alexa (150€), uno para el desarrollador y otro para implantarlo y que recopile datos.

Por lo tanto, la estimación del coste económico de estudio y desarrollo del proyecto será de unos 6300€.

4.4. Colección

Para determinar la colección de documentos a utilizar buscaré las páginas de preguntas frecuentes de las que dispone la universidad de Jaén.

La más poblada es la del servicio técnico de informática, aunque considero que las preguntas son en su mayoría de fallos concretos y las respuestas para las preguntas no son triviales para ser locutadas por un asistente.

La otra página web interesante sobre la que se puede obtener una colección es la de la sección de Acceso y matrícula, que se encuentra en la siguiente dirección:

https://www.ujaen.es/estudios/acceso-y-matricula/preguntas-frecuentes?title=&term_node_tid_depth=All&field_categoria_faq_target_id=All

Esta página será la elegida para extraer las preguntas y las respuestas sobre las que aplicaremos las búsquedas.

La siguiente figura muestra un ejemplo de la web informativa que tiene la Universidad de Jaén.

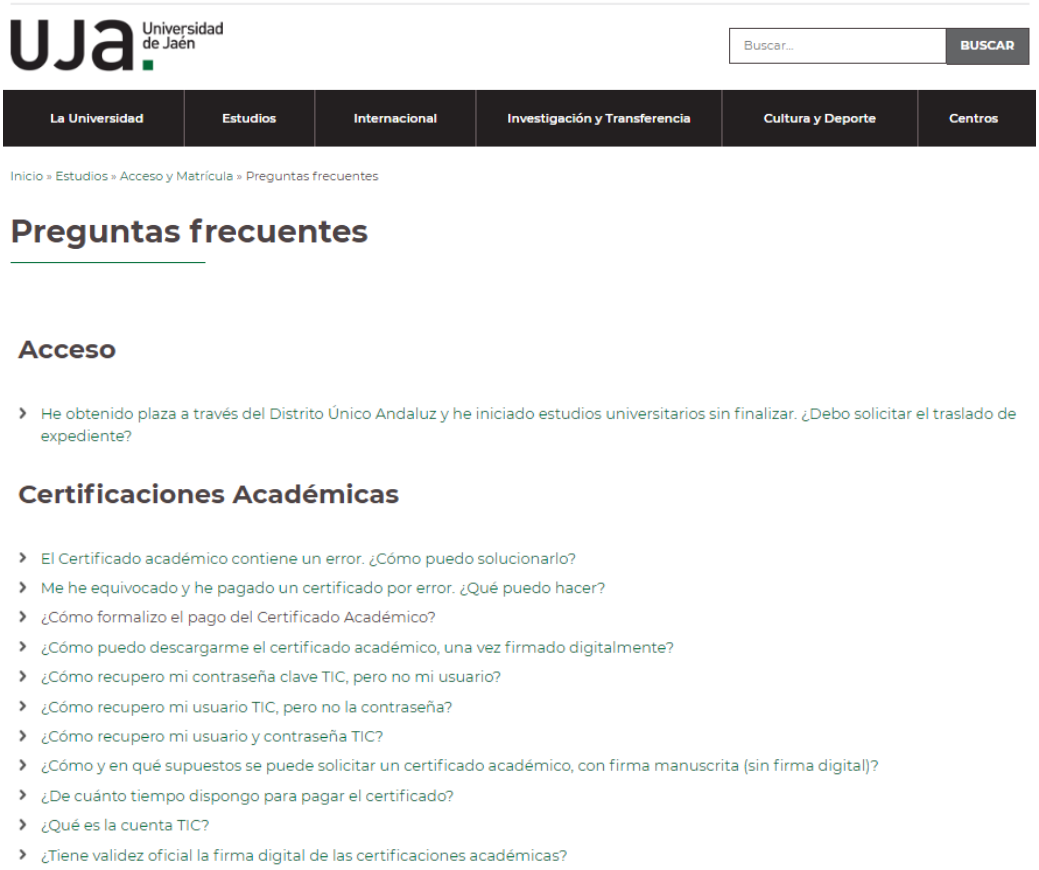


Ilustración 4-3. Página web de preguntas frecuentes de la universidad

4.5. Índice de la colección textual

La discusión sobre usar una estructura de datos u otra en este caso se presenta a la hora de crear un sistema de recuperación de información competente.

La motivación es buscar una estructura que tenga el menor tiempo de consulta posible.

Las estructuras más indicadas para montar un sistema de este perfil, son las que tengan un tiempo de consulta muy bajo, sacrificando a cambio que tengan un tiempo de precálculo algo mayor pero soportable en tiempo de cómputo.

Las estructuras más adecuadas para esto es un índice y los hay de dos tipos:

- directo
- inverso

Índice directo

El índice directo parte de la idea de tener una fila por documento y en cada fila tener parejas de término peso.

$$\begin{array}{cccccc} d_1 & w_{11}t_{11} & w_{12}t_{12} & \cdots & w_{1\mathrm{d}_1}t_{1\mathrm{d}_1} \\ d_2 & w_{21}t_{21} & w_{22}t_{22} & \cdots & w_{2\mathrm{d}_2}t_{2\mathrm{d}_2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_m & w_{m1}t_{m1} & w_{m2}t_{m2} & \cdots & w_{md_m}t_{md_m} \end{array}$$

Ilustración 4-4. Estructura del índice directo

Este índice directo tiene una complejidad de:

- Creación: O (NºDocumentos * Media términos)
  - Soportable
- Consulta: O (NºDocumentos * Log (Media de términos))
  - No soportable

Índice inverso

Usa la estructura de 1 fila por término y en cada fila parejas de documento-peso

$$\begin{array}{cccccc} t_1 & w_{11}d_{11} & w_{12}d_{12} & \cdots & w_{1t_1}d_{1t_1} \\ t_2 & w_{21}d_{21} & w_{22}d_{22} & \cdots & w_{2t_2}d_{2t_2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_m & w_{n1}d_{n1} & w_{n2}d_{n2} & \cdots & w_{nt_n}d_{nt_n} \end{array}$$

Ilustración 4-5. Estructura del índice inverso

Estudiando la complejidad:

- Creación: O (NºDocumentos * Media palabras)
  - Soportable
- Consulta: O (NºDocumentos * NºPalabras de la consulta)
  - Mejor que el directo

Capítulo 5

Diseño

5. Diseño

5.1. Diagrama general

A continuación se muestra el diagrama general de funcionamiento, que describe una primera idea de lo que será el sistema. Habrá que definir cómo funciona internamente el sistema de recuperación y como se trata la colección.

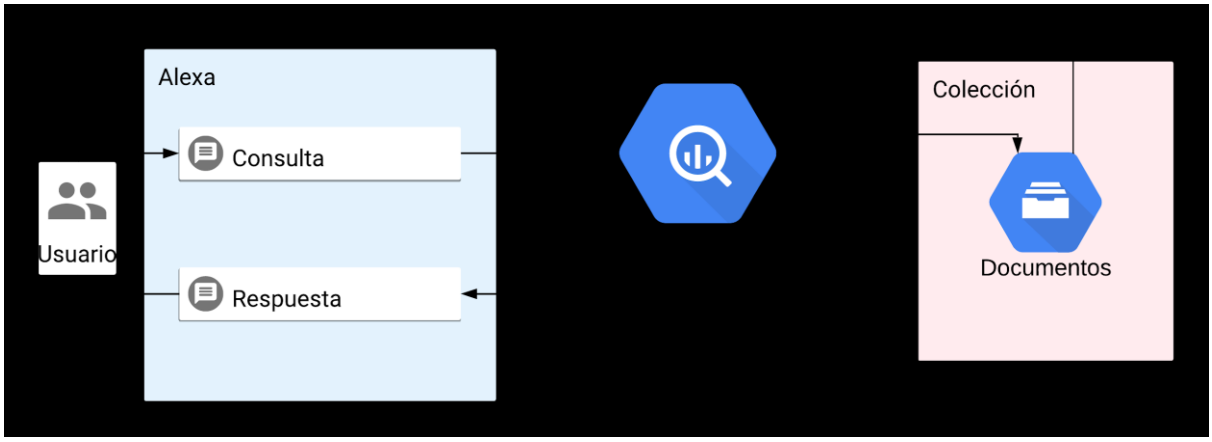


Ilustración 5-1 Diagrama general del funcionamiento del sistema

Cada consulta del usuario seguirá el siguiente esquema:

1. Usuario se comunica con Alexa y le expresa su consulta.
2. La consulta se lleva al sistema de recuperación de información.
3. Este, busca en la colección si hay algún documento que resulte relevante para la consulta realizada.
4. Se devuelve el documento relevante o si no se encuentra ninguno se le comunica al usuario

5.2. Storyboard

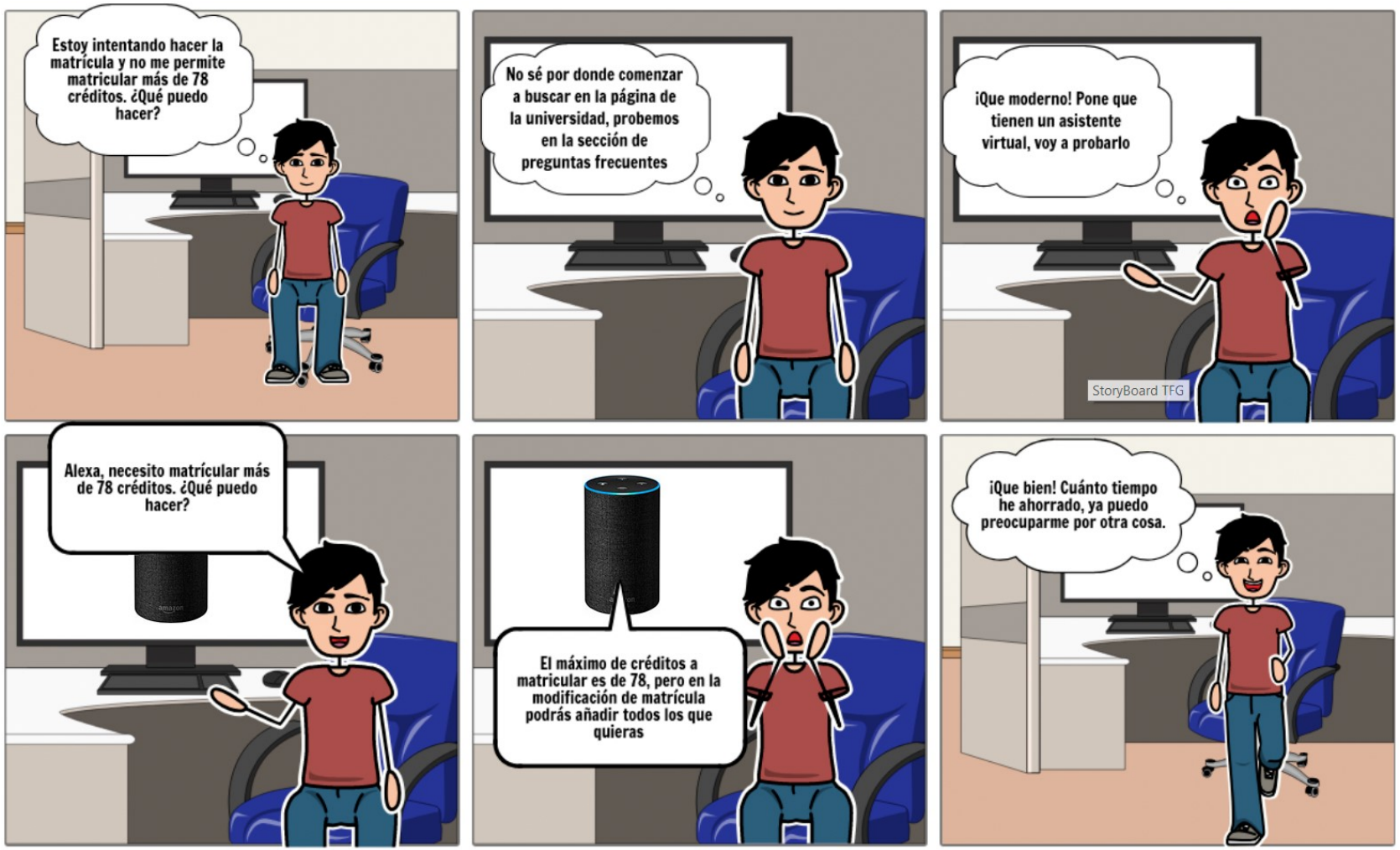


Ilustración 5-2. Storyboard creado con la herramienta online storyboardthat

Capítulo 6

Desarrollo

6. Desarrollo

6.1. Descripción de la colección

La colección sobre la que se trabaja se ha extraído mediante un script propio en Python, utilizando para ello el módulo urllib [6]de Python

Como ya se ha comentado antes, la colección ha sido extraída de la página de preguntas frecuentes sobre Acceso y Matrícula de la Universidad de Jaén.

Una vez extraídas las preguntas y las respuestas manualmente hay que adaptar cada una de ellas para conseguir una frase locutable por el asistente intentando ceñirse a lo que el usuario pregunta.

Una vez el sistema decida cuál es el documento más relevante, podrá leer la respuesta correspondiente. Estas respuestas locutables las voy a guardar en el archivo llamado “data.py”. En este archivo es donde irán todas las líneas de texto del programa, para así en caso de encontrar fallos o añadir nuevos documentos, esté todo en un único archivo.

Como comentario adicional el nombre de cada documento de la colección es el comienzo del texto de la pregunta a la que responde el documento en la faq de la universidad de Jaén.

Además, se han eliminado todos los símbolos de puntuación y diacríticos, de los nombres de los documentos, para su correcto funcionamiento.

Un ejemplo de como se organiza la página:

[6] Biblioteca estándar de Python que nos ayuda a extraer contenido de páginas web.

Ilustración 6-1. Ejemplo de pregunta extraída para la colección

De esta página obtendré la pregunta y la respuesta proporcionada por la universidad.

Con esos dos elementos alimentaré mi colección de documentos.

6.2. Diagrama general de interacción:

A continuación, se describe el funcionamiento del sistema planteado para la solución de este TFG.

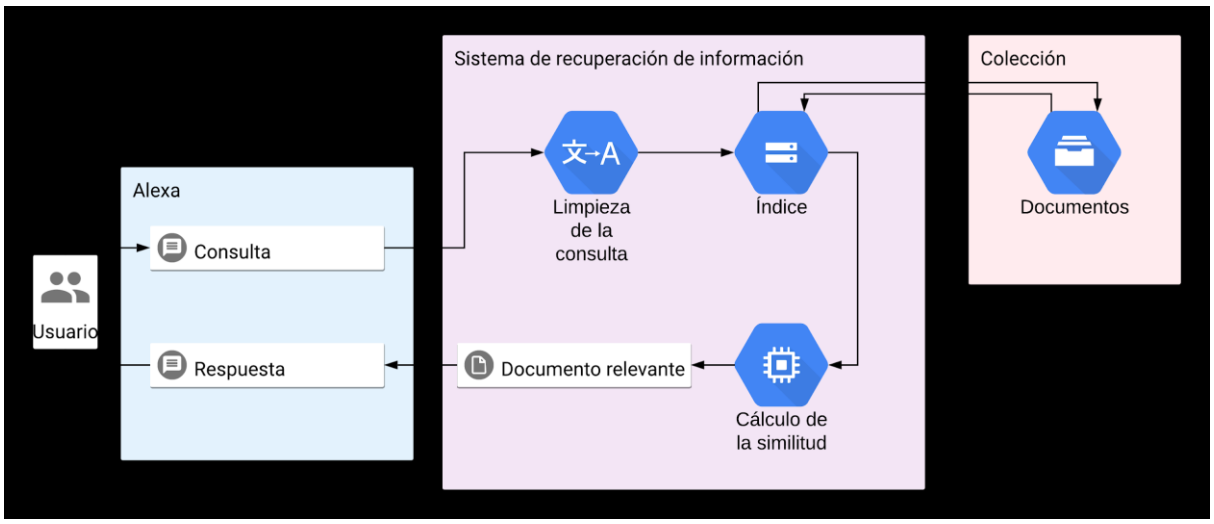


Ilustración 6-2. Esquema del funcionamiento del sistema.

Tal como se puede ver en la figura 6-1 el funcionamiento del sistema será el siguiente:

1. Un usuario realiza una consulta a Alexa mediante voz o texto escrito.
2. Alexa, comprenderá la voz del usuario y capturará la consulta.
3. La consulta se lanzará a la dirección del AWS Lambda donde tenemos almacenado el SRI.
4. El SRI, aplicará técnicas a la consulta para mejorar su rendimiento y efectividad.
5. Con los términos que nos queden de la consulta, calcularemos la similitud con los documentos de la colección previante indexados en local.
6. Una vez recuperemos el documento relevante que contenga nuestra respuesta, se la pasaremos a Alexa de nuevo para que la locute al usuario.

6.3. Detalles de implementación.

Este proyecto cuenta con 2 partes diferenciadas:

1. Desarrollo de la Skill
2. Desarrollo del sistema de recuperación de información

6.3.1. Desarrollo de la Skill

Todo desarrollo de una Skill a su vez se fundamenta en 2 partes:

6.3.1.1. Alexa Developer console

Nombre de invocación: secretaría ujaen

Intents:

En este caso el funcionamiento de la Skill será capturar la consulta del usuario en un único Intent, esto no es lo normal, pero en este caso se realiza así para recoger la frase del usuario y llevarlo directamente al sistema de recuperación de información.

El nombre escogido para nuestro Intent es “Consulta”.

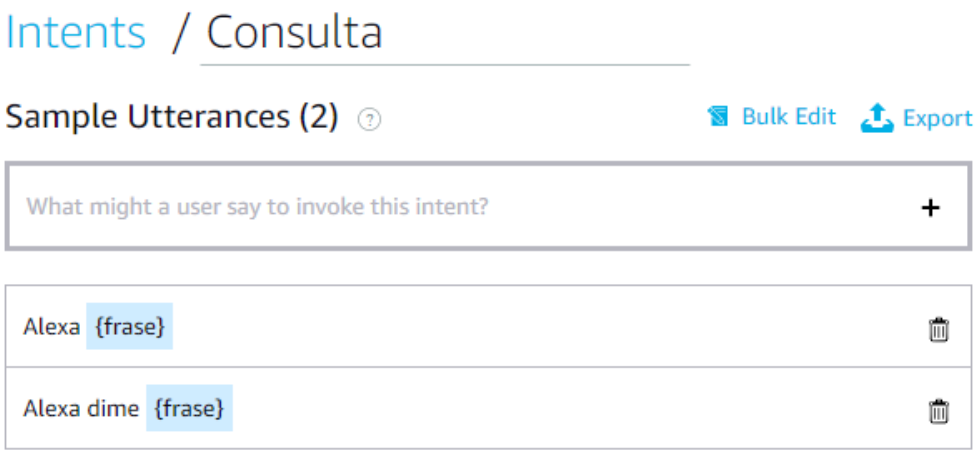


Ilustración 6-3 Frases que controlará el Intent Consulta

Funcionamiento de este Intent:

Cómo podemos observar contamos con dos frases, en las cuales contamos con el Slot {frase}.

Para hacerlo funcional debemos asignar un Slot al concepto que nosotros usamos en la oración bajo el nombre de “frase”. En este caso recibe el nombre de frase_slot.

Ilustración 6-4. Como indicar el slot que usará la variable frase

Frase_slot será un Slot creado simplemente para luego recoger el valor de lo que se le diga a Alexa. El valor o valores que introduzcamos en la creación del Slot será irrelevante ya que en este caso no se usarán.

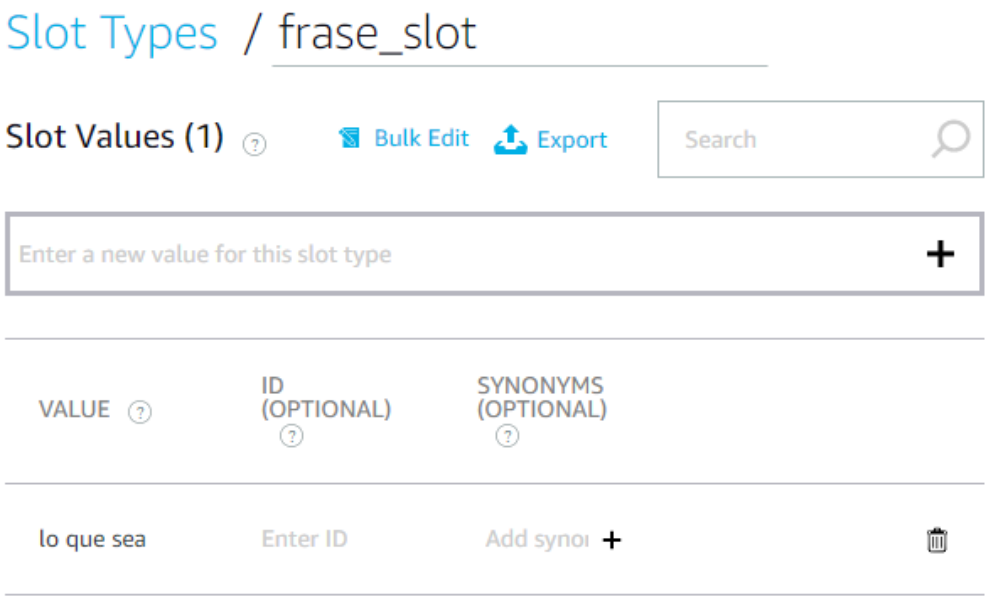


Ilustración 6-5 Definición del Slot, frase_slot

Cómo podemos observar tenemos el Intent correctamente configurado y el funcionamiento será muy simple. Recoger todas las frases que comiencen por las expresiones “Alexa ...” o “Alexa dime...”.

El resto de la frase que se recoja pasará a formar parte del Slot “frase” el cual recogeremos para el sistema de recuperación de información.

En la consola solo nos falta indicar en el apartado endpoint la dirección del AWS Lambda donde almacenaremos nuestra Skill.

6.3.1.2. AWS Lambda

Para esta parte del desarrollo deberemos crear un proyecto básico como se muestra en el Anexo I: “Desarrollo desde 0 de una Skill con Python”.

Intents por defecto:

- LaunchRequestHandler
  - Se lanza cada vez que se inicia la Skill.
  - Se usa a modo de saludo.
- HelpIntentHandler
  - Se activa cuando Alexa entiende una expresión de ayuda recogida en su Slot AMAZON.HelpIntent.
- CancelOrStopIntentHandler.
  - Se activa cuando alexa entiende una expresión de cancelación o de parada de sus Slots AMAZON.CancelIntent o AMAZON.StopIntent.
- SessionEndedRequestHandler
  - Se dispara cuando la petición al asistente es de tipo SessionEndedRequest.
  - Es decir, esta solo se dispara al cerrar la Skill.
  - Se usa para lanzar un mensaje de despedida al usuario.
- CatchAllExceptionHandler
  - Recoge todas las excepciones y fallos de la Skill.
  - También se activa si ningún otro Intent es disparado.

Todas estas funciones, siguen el mismo estilo, los mensajes locutados se cargan del archivo “data”, el cual contiene todos los diálogos que se usan en la Skill. En todos los casos se carga la lista correspondiente y se elige un diálogo al azar.

El archivo “data” sigue el siguiente estilo:

```
NOMBRE_SKILL = "Secretaria ujaen"

MENSAJE_BIENVENIDA = [  "bienvenido al sistema de secretaría. ¿Qué puedo hacer por ti?",
                        "bienvenido al sistema de secretaría. ¿En qué puedo ayudarte?",
                        "bienvenido al sistema de secretaría. ¿Qué deseas?"]

QUIERES_ALGO_MAS = [" ¿puedo ayudarle en algo más?",
                    " ¿puedo hacer algo más por ti?",
                    " ¿qué más puedo hacer por ti?",
                    " ¿querría preguntarme algo más?",
                    " Si desea algo más, estoy a su entera disposición"]
```

Ilustración 6-6 Estructura del archivo data.py

Un ejemplo de estos intents:

```
class CancelOrStopIntentHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        return (is_intent_name("AMAZON.CancelIntent")(handler_input) or
                is_intent_name("AMAZON.StopIntent")(handler_input))

    def handle(self, handler_input):
        logger.info("En CancelOrStopIntentHandler")

        tmp = getattr(data, "STOP_MESSAGE", None)
        speech = tmp[randint( 0, len(tmp)-1)]

        tmp = getattr(data, "QUIERES_ALGO_MAS", None)
        ayuda = tmp[randint( 0, len(tmp)-1)]


        handler_input.response_builder.speak(speech).ask(ayuda).set_should_end_session(False)
        return handler_input.response_builder.response
```

Ilustración 6-7 Código del Intent de una orden de parada o cancelación

Las función logger.info(), es una función usada para sacar por consola una cadena de texto, no realiza nada más.

Para devolver la respuesta utilizamos el objeto “handler_input.response_builder”, con la función “speak()”, especificamos el texto a locutar.

Además Alexa también cuenta la función “ask()”. La cual recibe también un parámetro de tipo string, y se utiliza también para locutar, pero en este caso, el texto solo se locuta cuando ya se ha locutado la de la función “speak()” y ha pasado cierto tiempo sin obtener respuesta del usuario.

Su funcionalidad por tanto es la de realizar preguntas del tipo, “¿Puedo ayudarte en algo más?”.

Además si no se recibe ninguna respuesta a la función “ask()” lanza una SessionEndedRequest y se cierra la Skill.

Intent “Consulta”:

```python
class Consulta (AbstractRequestHandler):
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return is_intent_name("Consulta")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        logger.info("En Consulta")

        #bool. controla que la consulta que Alexa recibe sea válida
        falloLeer = False

        try:
            #se extrae la consulta de Alexa
            tmp = handler_input.request_envelope.request.to_dict()
            #se guarda el contenido del Slot "frase"
            consulta = tmp['intent']['slots']['frase']['value']
        except Exception:
            logger.info("fallo al leer algo")
            falloLeer = True

        tmp = getattr(data, "QUIERES_ALGO_MAS", None)
        ayuda = tmp[randint( 0, len(tmp)-1)]

        # Si algo la consulta no es válida se comunica al usuario y no se lanza la búsqueda
        if falloLeer or consulta=='':
            no_entiendo = getattr(data, "NO_ENTIENDO", None)
            speech = no_entiendo[randint( 0, len(no_entiendo)-1)]

            handler_input.response_builder.speak(speech).ask(ayuda).set_should_end_session(False)
            return handler_input.response_builder.response

        logger.info("pre busqueda")

        busqueda = Busqueda()

        try:
            #se lanza la búsqueda
            speech = busqueda.ejecuta(consulta,logger)
        except Exception:
            no_entiendo = getattr(data, "NO_ENTIENDO", None)
            speech = no_entiendo[randint( 0, len(no_entiendo)-1)]

        logger.info("fin de la búsqueda")

        handler_input.response_builder.speak(speech).ask(ayuda).set_should_end_session(False)
        return handler_input.response_builder.response
```

Ilustración 6-8 Código del Intent Consulta

Consulta es el intent pensado para recoger todas preguntas del usuario, como vimos antes en la consola de desarrolladores recogerá todas las frases que sean del tipo “Alexa {frase}”.

Como podemos suponer nos interesa extraer la frase del usuario para realizar la recuperación de información.

Esto es exactamente lo que realiza el intent “Consulta”, en primer lugar almacena la frase del usuario, si no es válida locuta un mensaje de error y termina su ejecución. Si es válida la frase la pasa como argumento a la función ejecuta() del objeto Búsqueda().

De esta función se recoge directamente la cadena que contiene el diálogo que debe locutar Alexa, y se le pasa a la función response_builder del objeto handler_input, el cual es el encargado de indicarle a Alexa la cadena que debe locutar.

6.3.1.3. Sistema de recuperación de información

El esquema de funcionamiento del sistema de recuperación de información planteado es el siguiente:

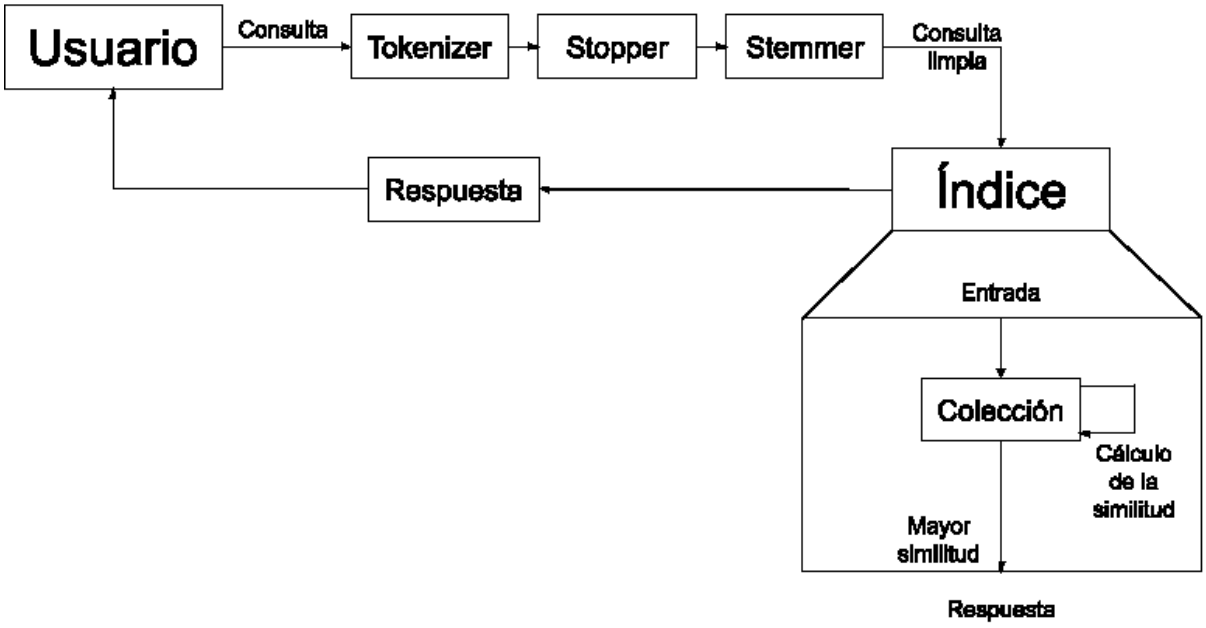


Ilustración 6-9. Esquema de funcionamiento del Sri desarrollado

Una vez contamos con la colección, debemos indexar los documentos.

Para ello por cada documento primero aplicaremos una limpieza del texto.

A. Limpieza del texto

Para realizar una limpieza del texto usaremos 3 elementos:

- Tokenizer
- Stopper
- Stemmer

Tokenizer[7]:

Esta clase se usa para dividir una cadena de texto en subcadenas. De esta forma dividiremos las cadenas de los documentos para conseguir extraer todas las palabras que hay en el documento y almacenar cada una en una posición diferente de una lista.

Stopper[8]:

Un Stopper es el elemento usado para eliminar las Stopwords o palabras vacías.

En el conjunto de las Stopwords se incluyen todas las palabras que no aportan significado semántico en una oración, es decir, que carecen de información.

Estas palabras se eliminan en los sistemas de recuperación para mejorar el rendimiento en tiempo de cómputo y en porcentaje de acierto.

[7] https://docs.python.org/3/library/tokenize.html
[8] https://pythonspot.com/nltk-stop-words/

Si no se eliminan las palabras vacías de la colección podremos dar lugar a falsos positivos.

Por ejemplo si la consulta incluye artículos, estos van a aumentar el peso de documentos con dichos artículos, y cómo podemos prever esto causa errores graves en nuestro sistema.

En este caso voy a usar las Stopwords del módulo de nltk de python, pero voy a realizar una pequeña modificación y voy a eliminar de la lista negra ciertas palabras que para nuestro problema si suponen un peso como son; “dónde”, “cómo” y “cuándo”.

Stemmer[9]:

Un Stemmer es la herramienta usada para extraer el lema de una palabra.

Un lema es la unidad mínima de significado.

Al aplicar esta transformación en las palabras de la colección conseguiremos que palabras derivadas, que comúnmente tienen relación, sean tratadas de la misma forma.

Con esto ganamos flexibilidad en la comprensión de las consultas y mejoramos el porcentaje de acierto del sistema en la mayoría de los casos.

El Stemmer escogido será el Snowball de la librería nltk de Python.

[9] https://www.nltk.org/api/nltk.stem.html

B. Elección de la estructura de datos para el índice.

Como hemos visto en el Error! Reference source not found. a la hora de implentar el Sri debemos elegir sobre si usar un índice directo o un índice inverso.

Como para nuestro problema lo más importante es conseguir un tiempo de consulta bajo con un tiempo de indexación razonable la mejor opción es utilizar un índice inverso.

Una vez esté montado lo serializaremos haciendo uso del módulo preinstalado de Python, Pickle[10].

C. Detalles de implementación y metodología del funcionamiento:

Una vez completada la limpieza de un documento se calculan los pesos de cada término, en los documentos de la colección.

Una vez completamos el proceso, lo almacenamos en un índice inverso y lo serializamos con el módulo Pickle de Python. Este índice se implementa sobre un diccionario de Python siguiendo la estructura:

{‘término’: [‘documento1-peso’,’documento2-peso’]}

Como se muestra en la siguiente imagen:

[10] https://docs.python.org/3/library/pickle.html

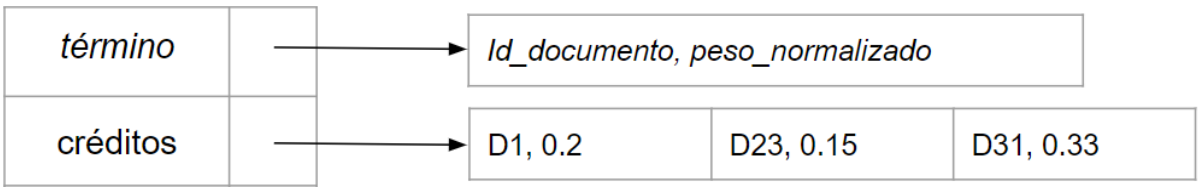


Ilustración 6-10 Esquema de la estructura del índice inverso

De esta forma tendremos todas las palabras de la colección y para cada palabra almacenamos los documentos en los que se encuentra y el peso de ese término en cada documento.

Respecto a los índices es necesario que se actualicen en local ya que dentro de nuestra función en AWS Lambda no se nos permite abrir archivos, con lo que no podremos leer el contenido de nuestra colección.

Una vez contamos con el índice serializado podemos realizar búsquedas.

La búsqueda recibe la consulta, aplica Tokenizer, Stopper y Stemmer. Posteriormente se recorren los documentos en los que salen esos términos y se calcula la similitud.

El documento que tenga la mayor similitud con la consulta será el elegido por el sistema. En caso de no encontrar ninguno o no superar un umbral, manda una salida para que Alexa locute un mensaje indicando que no ha sido posible encontrar una respuesta a su consulta.

D. Estructura de la colección

En primer lugar, tendremos en una carpeta todos los documentos que forman nuestra colección.

El nombre de estos documentos serán las primeras letras de la pregunta encontrada en la faq.

En estos documentos encontramos en la primera línea la pregunta completa, tal cual la encontramos en la faq.

En las posteriores líneas encontraremos la respuesta que la página de la universidad da a cada pregunta.

Como suplemento en algunos documentos se han añadido las palabras clave de ese documento para propiciar que el sistema de recuperación los recupere cuando es debido.

En cuanto a las locuciones que realizará alexa, las encontramos todas en el archivo “data.py” cuya ruta es “/alexa/data.py”. Su uso se explicará posteriormente en el apartado de desarrollo.

6.4. Estructura de ficheros

Los archivos que forman este proyecto se dividen en 2 partes:

- Un archivo JSON el cual define el modelo de interacción que sigue la Skill, que debe definirse en la Alexa Developer Console.
- Un archivo .Zip, que se sube tal cual a AWS Lambda sobre Python 3.6.

En el JSON quedarán reflejado los Intents, Slots y el nombre de invocación de la Skill.

En el Zip tendremos todo el funcionamiento de la Skill.

En una primera parte contamos con todos los módulos de Python necesarios para su ejecución.

De todas las carpetas solo debemos fijarnos en una carpeta y 2 scripts de Python.

Scripts:

- **prueba.py**

Este script es un script auxiliar que no afecta al funcionamiento de la Skill, pero si sirve para realizar pruebas en local. Este resulta bastante útil ya que sin él cada vez que queramos probar algún cambio en el sistema deberemos comprimir toda la carpeta otra vez y subirla al AWS Lambda.

- **main.py**

Este es el script principal desde donde se define los Handlers y el SkillBuilder que nuestra Skill buscará cada vez que lance una petición. De esta forma deberemos especificarle al SkillBuilder todas las clases que queramos que use.

Desde este archivo es desde donde se controla en funcionamiento de la Skill, y se define qué sucede si un usuario activa un Intent.

Contiene los Intents por defecto que necesita toda aplicación y el intent único que usaremos para gestionar las preguntas.

Una vez se active este último cogeremos la consulta del usuario, llamaremos al sistema de recuperación de información y esperaremos una respuesta.

Carpeta "/alexa":

El nombre de la carpeta importante es "alexa".

Dentro de esta carpeta encontramos:

- El script **data.py**.
    - En este Script encontramos todas las frases locutables que utiliza la Skill. Desde la frase de bienvenida, hasta cada una de las respuestas que debe dar si se devuelve un documento relevante al usuario.
- La carpeta **sri** donde se encuentra todo nuestro sistema de recuperación de información.

Carpeta "/alexa/sri"

Aquí encontramos los scripts:

- **Tokenizer**
  - Contiene la clase Tokenizer que transforma los documentos de texto en listas de términos
- **Stopper**
  - Contiene la la clase Stopper, la cual define lista de palabras vacías que usaremos en este proyecto.
- **Stemmer**
  - Define la clase Stemmer, para la extracción de raíces.
- **Índice**
  - Define la clase Índice
  - Contiene la función indexa, la cual recibe un booleano con el que definimos si queremos actualizar el índice con solo las preguntas o con el documento completo.
- **ActualizaÍndices**
  - Define un objeto de tipo Índice y actualiza ambos índices.
- **Búsqueda**
  - Define la clase búsqueda y la función que llamaremos para buscar desde el intent Consulta.
  - La función “ejecuta”, realiza todos los cálculos de pesos y similitudes y devuelve directamente la respuesta que debe locutar alexa.

Y las carpetas:

- **Índice**
  - Almacena los índices serializados.
- **Colección**
  - Almacena toda la colección de documentos.

Capítulo 7

Pruebas y evaluación

7. Pruebas y evaluación

A. Pruebas de usabilidad

Tras una serie de pruebas de usabilidad de la aplicación comprobé que la efectividad de la aplicación no era la esperada en un principio en todos los casos, pero aún así funcionaba bastante bien.

En algunos casos el sistema de recuperación de información devolvía por error un documento que no respondía a la pregunta realizada.

Pese a que este es uno de los temas más difíciles de corregir, ya casi siempre está presente en los problemas de procesamiento del lenguaje natural, decidí intentar buscar alternativas para ganar algo de usabilidad.

Primera alternativa:

La primera alternativa probada fue aplicar ciertos cambios en la forma en la que trataba la colección.

En primera instancia realicé pruebas comprobando si realmente merecía la pena aplicar o no el Stemmer. (Recordemos que el Stemmer es la clase usada para extraer las raíces de las palabras).

Tras no aplicarlo, el sistema funcionaba aún peor así que la idea fue descartada.

La segunda idea fue observar el rendimiento al aplicar el Stopper, que sirve para eliminar palabras vacías.

En este caso observé que como el sistema se basa en preguntas frecuentes de la gente sería conveniente eliminar ciertas palabras de la lista negra del Stopper, ya que en este caso palabras como “Qué”, “Cuando” o “Cuánto” podrían suponer un peso muy alto en el sentido de la frase.

Con esto, estas palabras ya no eran suprimidas del índice ni de la consulta y se pudo ver una clara mejora en el acierto del sistema.

Segunda alternativa:

La alternativa que mejor resultado me ha dado ha sido la inclusión de un segundo índice serializado al sistema.

Este segundo índice indexa de la colección de documentos solo la pregunta correspondiente a ese documento.

Este segundo índice ahora pasa a ser el primer índice donde se calculará la similitud de los documentos con la consulta.

Al tratarse de solo un índice con las preguntas, el proceso es realmente rápido y consigue que, si se realiza una consulta y se encuentra un alto valor de similitud con un documento, se devuelve aún más rápido (En torno a 6 milisegundos de respuesta).

En el caso de no encontrar similitudes o que las encontradas sean muy poco fiables (valor de similitud bajo), se vuelve a lanzar la búsqueda sobre el índice donde tendremos los documentos completos indexados.

De esta forma mejoramos la probabilidad de acierto del sistema además de la velocidad en la mayoría de los casos.

B. Pruebas de rendimiento

I. Eficiencia

Las pruebas de rendimiento se han realizado sobre mi equipo personal, que cuenta con un disco SSD y un procesador AMD Ryzen 5 2600.

La ejecución la he realizado ejecutando desde el IDE Spyder.

Primera prueba

En esta primera prueba lanzaremos una consulta sobre el sistema de recuperación y discutiremos los resultados obtenidos.

La consulta escogida de manera aleatoria ha sido:

"¿qué es el reconocimiento de créditos?"

1.Sobre el índice de solo preguntas:

La salida devuelta por el sistema de recuperación es la siguiente:

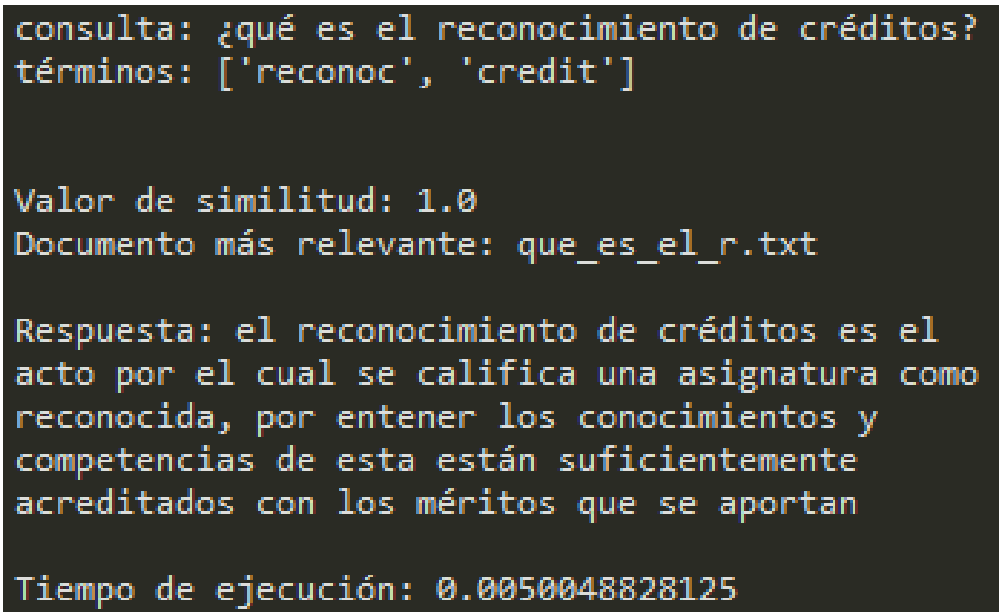


Ilustración 7-1 Salida del sistema para la búsqueda de una consulta sobre el índice de solo preguntas

Descripción:

1. Se muestra la consulta.
2. Se muestran los términos de la consulta que quedan después de aplicar el Stopper y el Stemmer.
3. A continuación se muestra el valor de similitud obtenido.
4. Se muestra el documento recuperado más relevante
5. Se muestra la respuesta que locutará alexa.
6. Se muestra el tiempo de ejecución.

Análisis de resultados:

- El valor de similitud como era de esperar es 1.0, ya que se ha buscado la consulta tal cual está en la faq. Este valor es el máximo posible ya que se trabaja con valores normalizados
- El tiempo de ejecución es 0.005 segundos (5 milisegundos) . Realmente bajo, vemos que en cuanto a velocidad no tenemos problema a priori.
- Debemos tener en cuenta que este tiempo será proporcional a la cantidad de términos que nos queden después de normalizar la consulta.

2. Sobre el índice del contenido completo de los documentos:

Lanzamos la misma consulta de forma similar y la salida obtenida es:

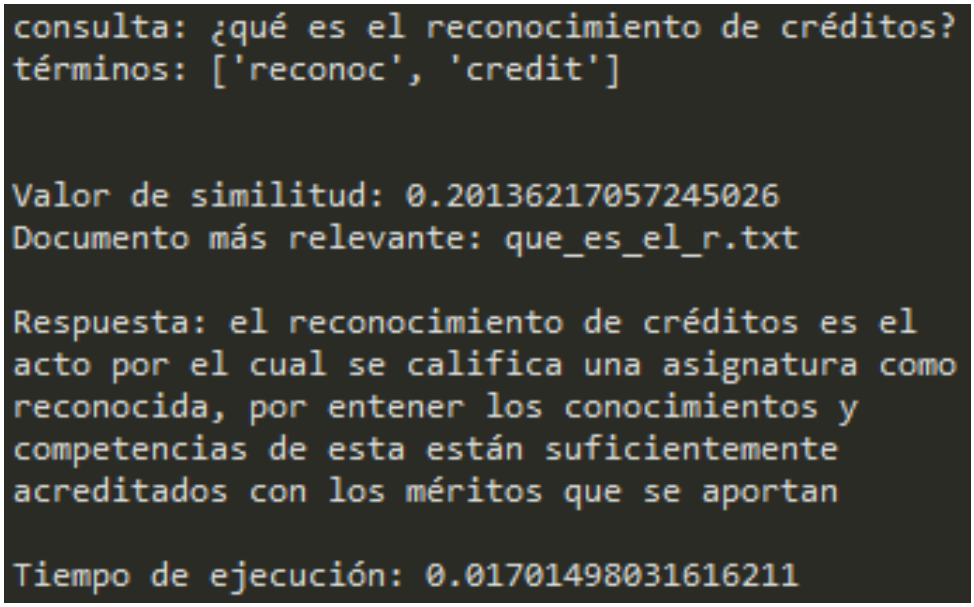


Ilustración 7-2 Salida del sistema para la búsqueda de una consulta sobre el índice completo

Las diferencias que observamos son:

- El valor de similitud de la consulta, observamos un valor bastante más bajo.
- El tiempo de ejecución sube hasta 0.017 segundos que sigue siendo un tiempo bueno.

Análisis:

Observamos que tenemos que tener cuidado con el umbral soportado a la hora de decidir si un documento es relevante o no en este caso, por lo que basta con disminuir ese umbral si lanzamos la búsqueda a este índice.

Segunda prueba

Para la segunda prueba vamos a lanzar cada una de las consultas soportadas por el sistema una tras una.

1. **Sobre el índice de solo preguntas**

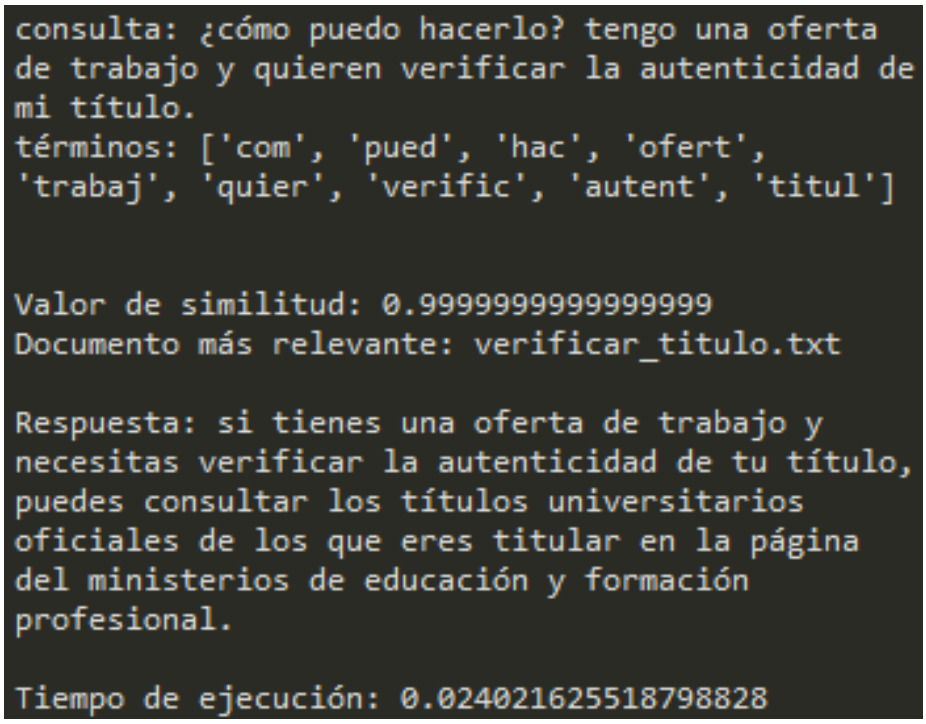


Ilustración 7-3. Salida del sistema para la búsqueda de una consulta sobre el índice de solo preguntas

Análisis:

Lo único que podemos comentar es el tiempo de ejecución, ya que los valores de similitud en todos los casos vuelven a ser 1.0 o un valor muy cercano. (0.999999).

Aun lanzando las 47 consultas observamos que sobre este índice la velocidad es bastante buena.

2. **Sobre el índice del contenido completo de los documentos:**

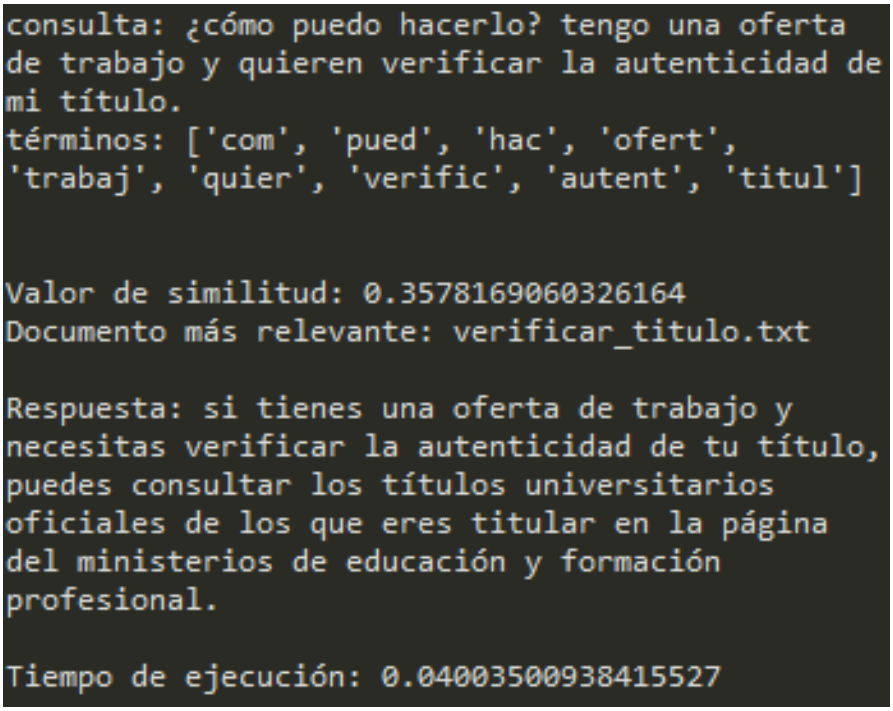


Ilustración 7-4 Salida del sistema para la búsqueda de una consulta sobre el índice completo

Análisis:

El tiempo de ejecución vemos que sigue siendo realmente bueno.

Cabe destacar que, en la versión final de este sistema, el tiempo de ejecución de esta misma operación sería en torno a lo descrito en la siguiente imagen:

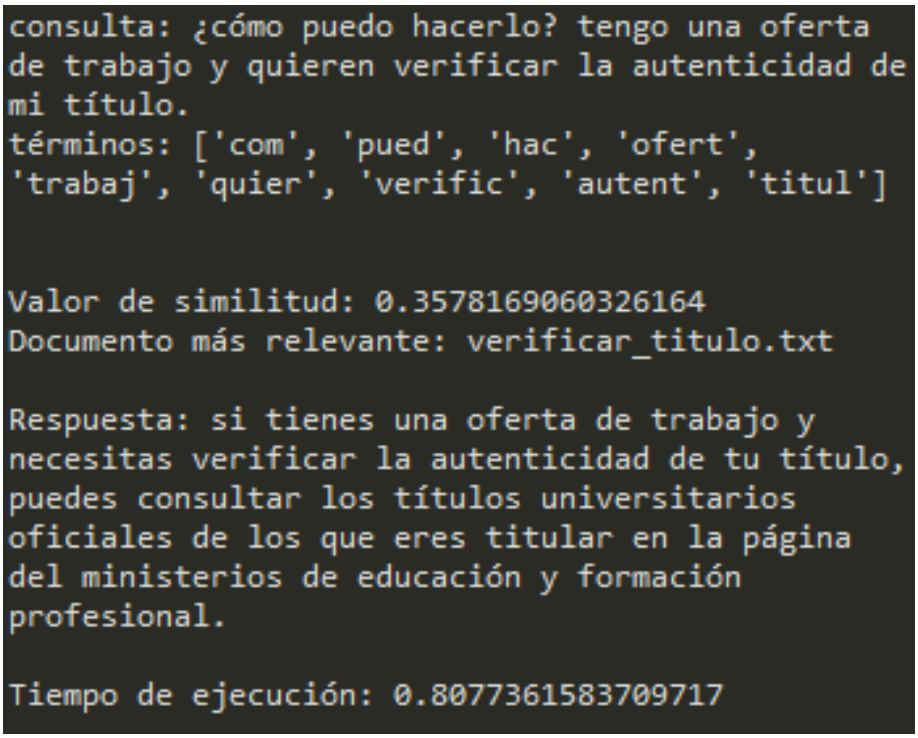


Ilustración 7-5 Salida al ejecutar cargando el índice en cada consulta

Esta diferencia se debe a que en la primera imagen se ha realizado un cambio en el código de la clase búsqueda para que no se tenga que cargar para cada documento el índice, y este quede siempre cargado en memoria.

Con esto podemos ver lo que realmente tarda en realizar la búsqueda sobre las 47 consultas.

La segunda imagen sería lo que tarda en ejecutar cargando para cada documento el índice de nuevo.

II. Efectividad/Acierto

Cómo podemos deducir no podremos realizar las mismas pruebas para este apartado, ya que por ejemplo si nos fijamos en la probabilidad de acierto de lanzar una consulta tal cual está en el índice donde solo tenemos las consultas, la efectividad va a ser del 100% siempre.

Si nos saltamos esa parte podemos mirar la efectividad resultante de lanzar todas las consultas sobre el índice donde tenemos indexados los documentos completos.

En este encontramos 6 fallos de las 47 consultas realizadas, lo que sería un 87’23% de probabilidad de acierto.

Este porcentaje a priori no es para nada bien, pero analizando los fallos a fondo, he podido comprobar que por ejemplo se producen errores como el siguiente:

Para la consulta “pago de la matrícula”

Se devuelve el documento “donde debo realizar el ingreso de las tasas”.

En este caso observamos que ambos documentos responden a la misma pregunta, así que el fallo recae sobre la colección utilizada.

Con esto podemos afirmar que el sistema será realmente dependiente de la colección con la que contemos y por desgracia en este caso la colección es pobre, dado que la base de preguntas frecuentes contiene muy pocas preguntas.

El desencadenante de esto será una efectividad baja, muy fácil de mejorar si poblamos la colección.

Conclusión sobre la efectividad

Teniendo en cuenta el funcionamiento del sistema, si lanzamos la consulta y se encuentra un documento relevante al calcular similitudes con el índice donde solo tenemos las preguntas. La efectividad será realmente alta.

La contrapartida de esto es que usamos sinónimos en vez de las palabras que se encuentran en la consulta, no va a recuperar ningún elemento.

Cuando esto sucede el sistema calcula similitudes sobre el índice donde tenemos indexados los documentos completos. En este caso puede ocurrir que encuentre el documento, que devuelva otro por error, o que no devuelva ninguno.

El factor real para saber cuál de los 3 casos sucederá es muy dependiente de la consulta realizada en cada caso, así que lo más adecuado en estos casos será mandarle al usuario un mensaje, comunicándole, que vuelva a realizar su consulta de otra forma, para así intentar encontrar una respuesta válida y correcta.

Capítulo 8

Conclusiones y trabajo futuro

8. Conclusiones y trabajo futuro

Las grandes conclusiones obtenidas en este proyecto son:

- Se han alcanzado todos los objetivos fijados para este TFG, y en algunos casos incluso mejorado la funcionalidad o rendimiento.
- Con una buena colección se aumentará la efectividad del sistema notablemente.
- El tiempo de ejecución de las búsquedas es muy bueno, ya que necesitábamos un tiempo muy bajo para que la comunicación con el usuario fuera fluida.
- El sistema devolverá buenas respuestas si el usuario utiliza las palabras adecuadas en cada caso. Esto lo convierte en un sistema algo dependiente del vocabulario empleado.

Y como trabajo futuro son múltiples las posibilidades de mejora, teniendo en cuenta que una implantación real de este tipo de sistemas se debería hacer cuando se cumplan unas garantías mínimas de éxito.

Implantación física

El proyecto se ideó con una clara pretensión de conseguir un sistema competente con posibilidad de ser presentado a la universidad como una línea de innovación tecnológica en sus secretarías.

La posible implantación apenas supondría coste económico ya que solo precisa de un altavoz inteligente y el coste de mantenimiento y actualización de la Skill. Y teniendo en cuenta el potencial que tiene con una buena colección detrás, pienso que es una de las líneas a seguir para el futuro y que para el presente supone una buena ayuda al trabajo humano, ya que podría disminuir la carga de

trabajo de forma notable, tanto en el sentido físico de poner un altavoz a disposición del alumnado, tanto como para poner la Skill en la tienda de aplicaciones y que el alumnado no tuviera que ir físicamente a realizar sus consultas.

Nuevas funcionalidades

Planteando posibles mejoras o funcionalidades, lo mejora más importante en este proyecto sería conseguir una buena colección.

Contando con ella, el sistema dará cada vez mejores resultados y mejorará la experiencia de uso.

También me parece una buena idea poner un altavoz con el sistema a disposición del alumnado que quiera usarlo, y apuntar todas las preguntas que se le realizaran, para posteriormente añadirlas a la colección y de esta forma poder tener un sistema muy bueno sin suponer apenas coste.

Referencias

G. Salton, A. Wong, and C. S. Yang (1975), "A Vector Space Model for Automatic Indexing," Communications of the ACM, vol. 18, nr. 11, páginas 613–620.

Baeza-Yates, Ricardo; Ribeiro-Neto, Berthier (1999), “Modern Information Retrieval”

N. Jardine, C.J. van Rijsbergen (December 1971). "The use of hierarchic clustering in information retrieval". Information Storage and Retrieval. 7 (5): 217–240

Tutorial oficial para desarrollo de aplicaciones para Alexa con Python.

https://developer.amazon.com/en-US/alexa/alexa-skills-kit/alexa-skill-python-tutorial

Diapositivas de la asignatura Sistemas de Recuperación de Información.

Anexo I:

Puesta en marcha de una Skill con Python 3.6

Anexo I: Puesta en marcha de una Skill con Python 3.6

La Skill de ejemplo que voy a enseñar estará pensada para saludar a la persona que le hable.

Todo desarrollo de una Skill para Alexa tiene 2 partes.

- Parte de Amazon Developer Console
- Parte de AWS Lambda

1. Alexa Developer Console

La primera parte del desarrollo se desarrollará en la consola de desarrolladores de Amazon Alexa.

La url es la siguiente:

https://developer.amazon.com/alexa/console/ask

En esta parte del proyecto tendremos que crear lo que será nuestra skill y definir el modelo de interacción con el usuario.

Una vez contemos con una cuenta debemos ir a crear nuestra Skill.

La crearemos con el nombre que más nos guste, con modelo personalizado y para almacenar el funcionamiento de nuestra aplicación, usaremos la opción de alojarlo nosotros mismos, ya que voy a trabajar con Python 3.6, y la opción para que te lo monte automáticamente Amazon es para desarrollar Skills sobre Node.js.

Una vez creado, comenzamos a definir todo lo necesario:

El nombre de invocación será “saludador”.

Por tanto, para iniciar la Skill simplemente tendremos que decir: “Alexa, abre saludador”.

Una vez digamos esto, nuestro asistente entrará en el intent “LaunchIntentRequest”, un intent por defecto que se activa cada vez que se abre la Skill.

Este intent lo usaremos para que Alexa locute un saludo y le pregunte el nombre al usuario, de la forma:

“Hola, bienvenido al sistema, por favor dime tu nombre”.

Ahora Alexa estará esperando una respuesta del usuario.

Debemos pensar cómo respondería un usuario cualquiera a dicha pregunta.

Y debemos definir un intent para capturar esas respuestas.

Este intent se llamará: “DiMiNombre”

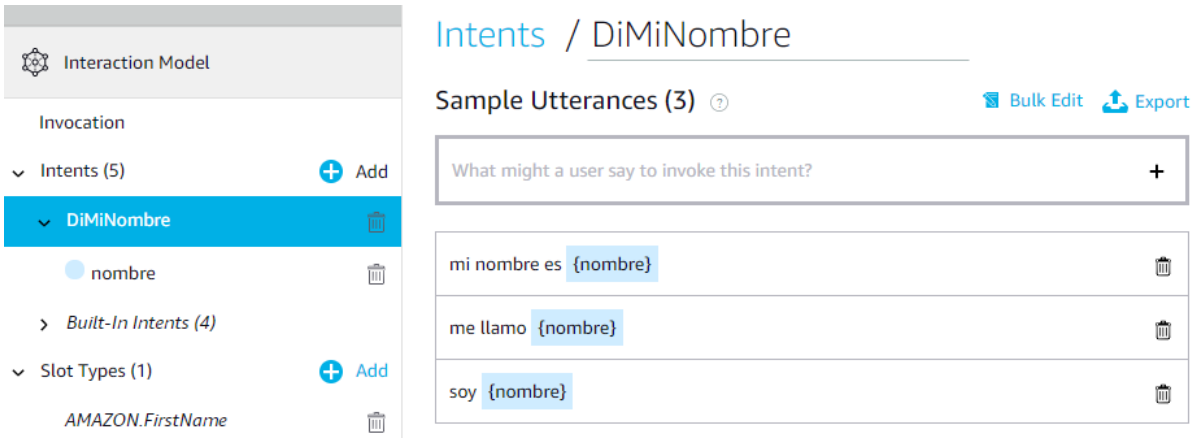


Ilustración 0-1. Ejemplo intent

Con el intent “DiMiNombre” vamos a capturar cualquier frase que contenga las frases que sigan las estructuras que definamos. (“Soy {nombre}” entre otras)

La palabra “{nombre}”, que aparece entre llaves denota que se trata de un Slot.

Como ya hemos definido un slot actúa de forma similar a una variable.

Existen 2 tipos, los proporcionados por Amazon o los personalizados que puedes crear a tu gusto.

Los proporcionados por Amazon son una serie de conjuntos de palabras de un mismo tipo, por ejemplo, he definido que nuestro Slot “{nombre}” sea de tipo AMAZON.FirstName. Como podemos deducir, este conjunto, contiene una gran lista de nombres personales.

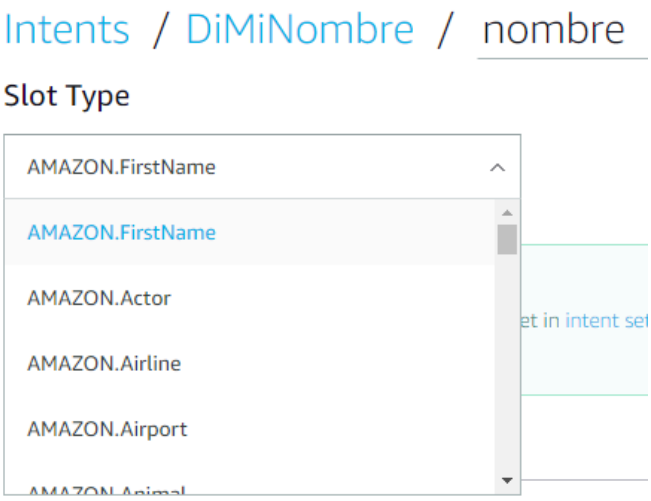


Ilustración 0-2. Ejemplos de conjuntos de Slot predefinidos de Amazon

Podemos comprobar todos los Slots Types de Amazon en el siguiente enlace:

https://developer.amazon.com/docs/custom-skills/slot-type-reference.html

Para mostrar cómo se define un Slot voy a cambiar el tipo de Slot de AMAZON.FirstName a “nombres_Slot”, un nuevo Slot personalizado en el que añadiré los nombres que quiera.

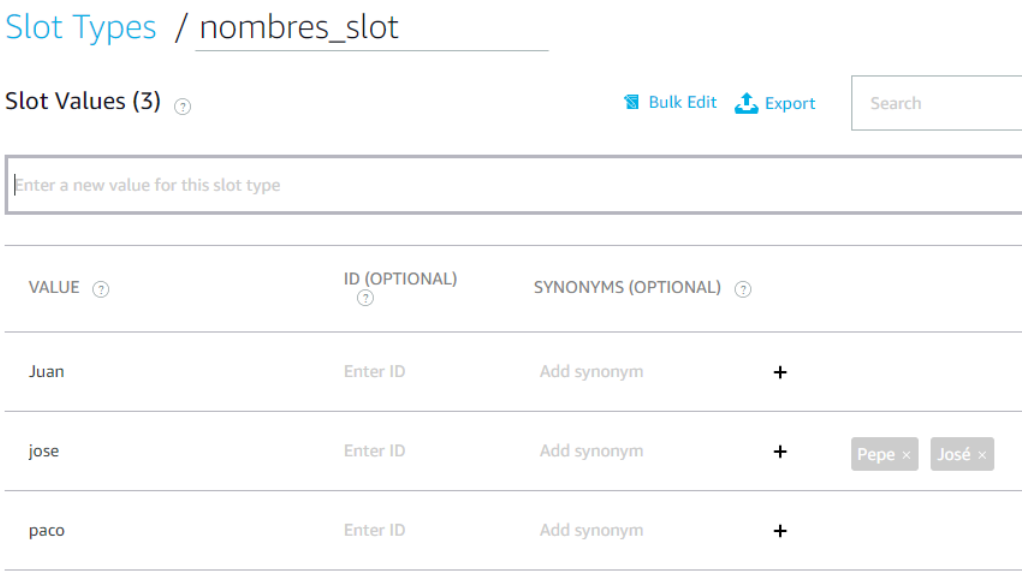


Ilustración 0-3. Valores para el Slot nombres_slot

Este es el Slot “nombres_slot”, y la idea es que en cada hueco del intent donde ponga {nombre}, se pueda insertar cualquiera de las opciones contempladas en él Slot.

Como detalle decir que también contempla el uso de sinónimos, por ejemplo, de José, Pepe. En este caso no lo utilizaré, pero es bastante útil en otros muchos casos.

Importante ir guardando los cambios y llegado a este punto debemos pulsar en el botón Build Model.

En lo que respecta a la parte de la consola de desarrollador ya solo nos faltaría definir en el apartado “Endpoint” la dirección donde cargaremos el código de nuestra Skill. Para eso usaremos AWS Lambda.

2. AWS Lambda

AWS Lambda es el servicio de Amazon que usaremos para tener el código de nuestra función en la nube.

Actualmente en Europa pese a haber varios servidores, el único que funciona es el de Irlanda así que trabajamos sobre este en todo momento.

Para este ejemplo trabajaremos sobre una función con Python 3.6.

La dirección de AWS es la siguiente:

https://eu-west-1.console.aws.amazon.com/console/home?region=eu-west-1#

Vamos al servicio Lambda y creamos una nueva función.

La creamos con la opción “Author from scratch” y con el lenguaje, Python 3.6.

Ahora debemos crear el proyecto en local desde un ordenador.

Para ello necesitaremos tener instalada la versión de Python que usemos y la orden pip.

Proceso:

1. Abrir cmd
2. *“Python --version”* (nos aseguramos de que está Python instalado)
3. *“pip install virtualenv”*

4. Creamos una carpeta donde más nos guste
   4.1. “mkdir Skill”
   4.2. “cd Skill”
5. “virtualenv skill_env”
6. “skill_env\Scripts\activate”
7. Aquí ahora en la consola nos saldrá (skill_env) $
   7.1. “pip install ask-sdk”

Ya tendremos la carpeta con la base. ahora nos vamos a la carpeta “Skill\skill_env\Lib\site-packages” y creamos un archivo Python con el nombre que más nos guste, por ejemplo saludador.py

Trabajando en ese archivo añadiremos los imports necesarios e iniciamos los objetos SkillBuilder y Logger. También necesarios.

```
# -*- coding: utf-8 -*-
import logging

from ask_sdk_core.skill_builder import SkillBuilder
from ask_sdk_core.dispatch_components import AbstractRequestHandler
from ask_sdk_core.dispatch_components import AbstractExceptionHandler
from ask_sdk_core.utils import is_request_type, is_intent_name
from ask_sdk_core.handler_input import HandlerInput

from ask_sdk_model.ui import SimpleCard
from ask_sdk_model import Response

sb = SkillBuilder()

logger = logging.getLogger(__name__)
logger.setLevel(logging.INFO)
```

Ilustración 0-4. Imports necesarios y definición del objeto SkillBuilder

Lo primero será crear la clase que será la encargada de recoger el intent anteriormente creado como “DiMiNombre”.

```python
class Saluda(AbstractRequestHandler):
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return is_intent_name("DiMiNombre")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response

        #se extrae la consulta de Alexa
        aux = handler_input.request_envelope.request.to_dict()
        #se guarda el contenido del Slot "nombre"
        nombre = aux['intent']['slots']['nombre']['value']

        speech = "Hola " + str(nombre) + ", encantado de conocerte"

        handler_input.response_builder.speak(speech).set_should_end_session(False)
        return handler_input.response_builder.response
```

Ilustración 0-5. Definición de la clase Saluda

Cada clase se tiene que definir de la misma forma, tendremos “can_handle” que será la encargada de determinar si la frase que ha dicho el usuario corresponde con el intent.

En este caso he indicado que entre en la clase Saluda si el intent capturado es del tipo “DiMiNombre”.

Y por otro lado tenemos la parte “handle” que se ejecuta si se cumple el “can_handle”.

Una vez dentro, podremos capturar el nombre que ha dicho el usuario y devolver un saludo locutado por Alexa.

Además de las funciones que queramos definir para nuestro programa deberemos definir en todo proyecto de Alexa las siguientes clases:

- LaunchRequestHandler
  - Se lanza cada vez que se inicia la Skill.

```python
class LaunchRequestHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        return is_request_type("LaunchRequest")(handler_input)

    def handle(self, handler_input):
        logger.info("En LaunchRequestHandler")

        speech = "Hola, bienvenido al sistema, por favor, dime tu nombre"

        handler_input.response_builder.speak(speech).set_should_end_session(False)
        return handler_input.response_builder.response
```

Ilustración 0-6. Definición de la clase LaunchRequestHandler

- HelpIntentHandler
  - Se activa cuando Alexa entiende una expresión de ayuda, recogida en su Slot AMAZON.HelpIntent.

```python
class HelpIntentHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        return is_intent_name("AMAZON.HelpIntent")(handler_input)
    def handle(self, handler_input):
        logger.info("En HelpIntentHandler")

        speech = "Si necesitas ayuda, prueba a decir como te llamas"

        handler_input.response_builder.speak(speech).set_should_end_session(False)
        return handler_input.response_builder.response
```

Ilustración 0-7. Definición de la clase HelpIntentHandler

- CancelOrStopIntentHandler.
  - Se activa cuando Alexa entiende una expresión de cancelación o de parada. Estas expresiones están contenidas en sus Slots propios AMAZON.CancelIntent y AMAZON.StopIntent.

```python
class CancelOrStopIntentHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        return (is_intent_name("AMAZON.CancelIntent")(handler_input) or
                is_intent_name("AMAZON.StopIntent")(handler_input))

    def handle(self, handler_input):
        logger.info("En CancelOrStopIntentHandler")

        speech = "Vale, cancelo lo que estaba haciendo. Te escucho"

        handler_input.response_builder.speak(speech).set_should_end_session(False)
        return handler_input.response_builder.response
```

Ilustración 0-8. Definición de la clase CancelOrStopIntent

- SessionEndedRequestHandler
  - Se dispara cuando la petición al asistente es de tipo SessionEndedRequest. Es decir, esta solo se dispara al cerrar la Skill.

```python
class SessionEndedRequestHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        return is_request_type("SessionEndedRequest")(handler_input)

    def handle(self, handler_input):
        logger.info("En SessionEndedRequest")

        speech = "Hasta luego, vuelve pronto"

        handler_input.response_builder.speak(speech).set_should_end_session(True)
        return handler_input.response_builder.response
```

Ilustración 0-9. Definición de la clase SessionEndendRequest

- CatchAllExceptionHandler
  - Recoge todas las excepciones y fallos de la Skill.
  - También se activa si ningún otro Intent es disparado.

```
class CatchAllExceptionHandler(AbstractExceptionHandler):
    def can_handle(self, handler_input, exception):
        return True

    def handle(self, handler_input, exception):
        logger.info("En CogeTodosFallos")

        logger.error(exception, exc_info=True)

        speech = "Algo ha salido mal"

        handler_input.response_builder.speak(speech).set_should_end_session(False)
        return handler_input.response_builder.response
```

Ilustración 0-10. Definición de la clase CatchAllExceptionHandler

Por último, debemos añadir todas las clases al objeto SkillBuilder

```
sb.add_request_handler(Saluda())

sb.add_request_handler(LaunchRequestHandler())
sb.add_request_handler(HelpIntentHandler())
sb.add_request_handler(CancelOrStopIntentHandler())
sb.add_request_handler(SessionEndedRequestHandler())
sb.add_exception_handler(CatchAllExceptionHandler())

handler = sb.lambda_handler()
```

Ilustración 0-11. Especificación al SkillBuilder de las clases a usar

Una vez hecho esto ya tendremos nuestra Skill lista.

Nos dirigimos a AWS Lambda y entramos en la función que habíamos creado.

En el primer apartado de “Designer” debemos desplegar la ventana y pulsar en “Add trigger”.

Debemos añadir el “Alexa Skills Kit”. Nos saldrá una pantalla para introducir nuestro Skill Id. Este Id lo podemos encontrar en la dashboard de la Alexa developer console. Es decir, en el menú donde tenemos todas nuestras Skills. Para una

aplicación sencilla también podemos optar por no ponerlo ya que no afecta al funcionamiento.

Una vez añadido nos quedará tal que así:

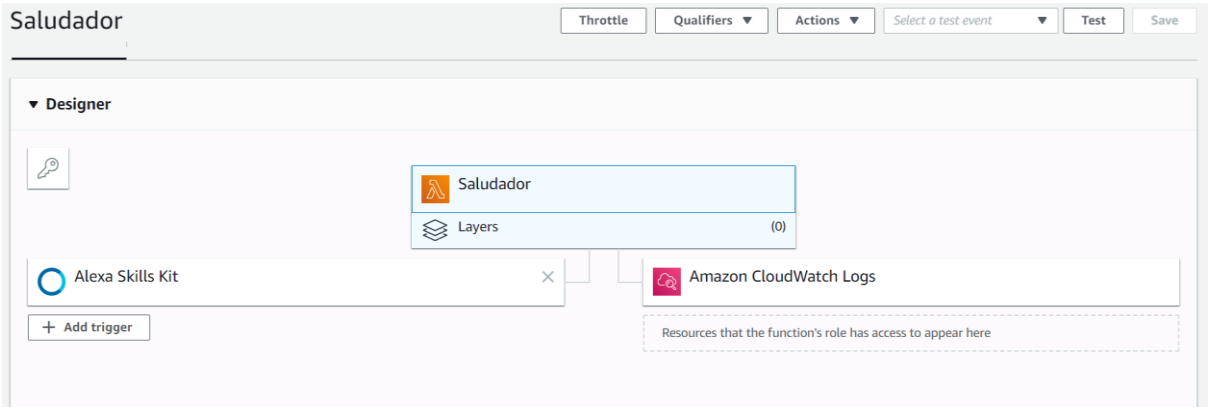


Ilustración 0-12. Muestra de la pantalla de AWS Lambda

Volvemos a Saludador y ahora vamos a la pestaña de “Function code” que está justo debajo.

Ahí vamos a cambiar “Code entry type” a “upload a .zip file”

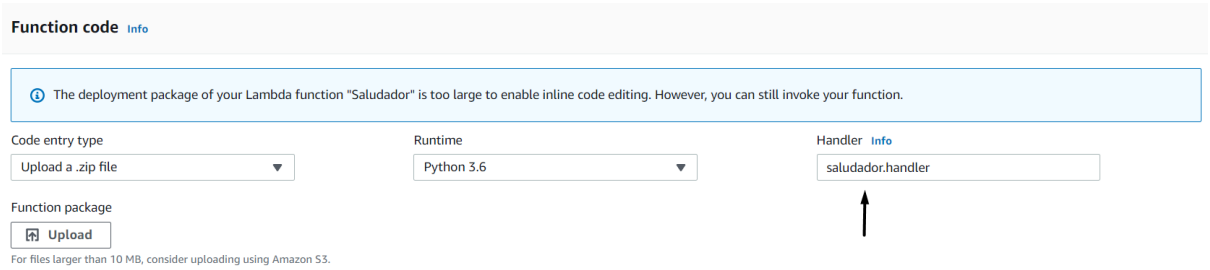


Ilustración 0-13. Muestra de la configuración correcta.

En el apartado handler debemos especificar el nombre de nuestro archivo.py previamente creado, que en mi caso era saludador.py y añadirle al final “.handler”.

Ahora debemos ir a nuestro proyecto local, ir a la carpeta donde pusimos nuestro saludador.py que era “Skill\skill_env\Lib\site-packages” y debemos comprimir en un archivo Zip todo el contenido de la carpeta. Recalco, debes copiar todo el contenido de la carpeta, no la carpeta en sí.

Una vez tenemos el .Zip lo podemos subir a Lambda pulsando en “Upload”.

Por último, tenemos que ir a la parte superior derecha de la pantalla y copiar el texto que pone en **ARN**.

En este punto podemos probar si hemos cargado bien nuestro proyecto en Lambda creando un evento de prueba. Esta opción está en la parte superior derecha de la pantalla.

Lo creamos bajo la plantilla Amazon Alexa Start Session. Esta plantilla lo que hará es lanzar un LaunchIntentRequest. Pulsamos en Test y si está todo correcto debería mostrar el mensaje de bienvenida a la Skill.

Ahora volvemos a la Alexa Developer console, al apartado Endpoint y pegamos la dirección **ARN** en la región por defecto.

Guardamos los cambios y ya tenemos la Skill lista.

Volvemos a pulsar el Build Model para asegurarnos que todo está listo y podemos pasar a probarla.

Para probarla nos vamos a la pantalla Test, la ponemos en modo desarrollador e introduciendo la orden de invocación deberá abrirla. En mi caso “abre saludador”.

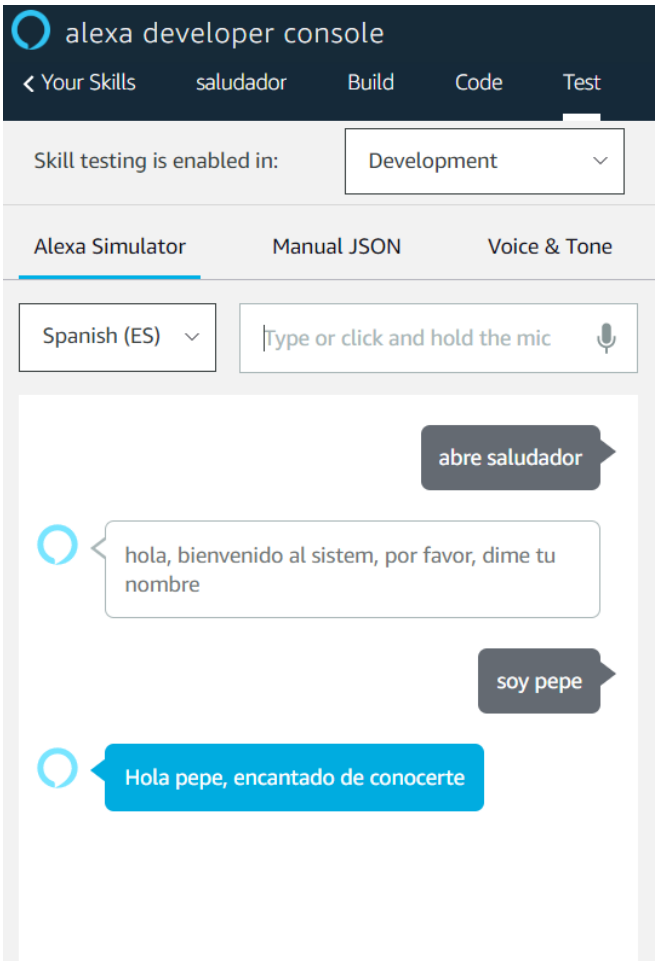


Ilustración 0-14. Ejemplo de interacción

Anexo II:

Preguntas soportadas.

Anexo II: Preguntas soportadas.

Este es el listado de preguntas soportadas por el sistema.

Han sido extraídas de la faq de la Universidad de la sección de Estudios / Acceso y Matrícula.

Puedes verlas en el siguiente enlace:

https://www.ujaen.es/estudios/acceso-y-matricula/preguntas-frecuentes?title=&term_node_tid_depth=All&field_categoria_faq_target_id=All

Preguntas de la colección

- ¿cómo consigo los programas de asignaturas o las guías docentes?
- ¿cómo formalizo el pago del certificado académico?
- ¿cómo puedo descargarme el certificado académico, una vez firmado digitalmente?
- ¿cómo puedo recoger el suplemento europeo al título?
- ¿cómo puedo saber que mi suplemento europeo al título está disponible?
- ¿cómo puedo solicitar la mención o la especialidad de mi titulación para que aparezca en el título?
- ¿cómo recupero mi usuario y contraseña tic?
- ¿cómo sabré que mi título oficial está disponible?
- ¿cómo se solicita el reconocimiento de créditos?
- ¿cómo solicitar la certificación supletoria del título (pretítulo)?
- ¿cómo y dónde puedo realizar la apostilla de la haya?
- ¿cómo y en qué supuestos se puede solicitar un certificado académico, con firma manuscrita (sin firma digital)?
- ¿cuáles son los requisitos para la retirada del pretítulo?

- ¿cuál es el coste de precios públicos para la expedición del título oficial?
- ¿cuál es el horario de atención al público de la sección de títulos oficiales y lugar de recogida?
- ¿cuándo puedo solicitar el título oficial?
- ¿cuándo tengo que renovar mi título de familia numerosa?
- ¿cuándo y cómo recibiré contestación del reconocimiento solicitado?
- ¿cuánto tarda la expedición del título?
- ¿de cuánto tiempo dispongo para pagar el certificado?
- ¿dónde debo hacer el ingreso de las tasas?
- ¿dónde puedo consultar si existen precedentes?
- ¿dónde realizo el reconocimiento de firmas de mis documentos (títulos, suplemento europeo al título, certificación académica, certificación supletoria del título, emitidos por la uja).
- ¿cómo puedo obtener un duplicado de mi suplemento europeo al título?
- el certificado académico contiene un error. ¿cómo puedo solucionarlo?
- ¿es necesario matricular la asignatura para solicitar el reconocimiento de créditos?
- ¿he extraviado el título, cómo puedo obtener un duplicado?
- he obtenido plaza a través del distrito único andaluz y he iniciado estudios universitarios sin finalizar. ¿debo solicitar el traslado de expediente?
- he olvidado indicar en la matrícula una bonificación (becario, familia numerosa, discapacidad etc.) ¿qué tengo que hacer para indicarla en mi matrícula?
- he participado en el concurso de olimpiadas de economía, física, geología, matemáticas y química en el presente año académico (abril 2019), ¿tengo derecho a la exención de precios públicos?
- ¿la certificación supletoria provisional (pretítulo) tiene los mismos efectos legales que el título?
- me he equivocado y he pagado un certificado por error. ¿qué puedo hacer?
- ¿mi título está disponible? he recibido un sms
- necesito matricular más de 78 créditos
- pago de la matrícula

- ¿puede recoger mi título oficial otra persona en mi lugar presentando una autorización firmada por mí con mi dni?
- ¿qué bonificaciones puedo obtener? he obtenido premio extraordinario de mi titulación
- ¿qué es el reconocimiento de créditos?
- ¿qué es el suplemento europeo al título?
- ¿qué es la cuenta tic?
- ¿qué recurso / reclamación y dónde puedo presentarlo, una vez que me han comunicado el resultado del reconocimiento (resolución)?
- ¿qué tienen que hacer los estudiantes que quieran pasar de una titulación conjunta a una de las simples que la forman?
- ¿qué tipos de reconocimientos hay?
- ¿se puede enviar el título oficial a nuestro domicilio?
- ¿tengo que estar matriculado para solicitar la compensación curricular?
- ¿tiene validez oficial la firma digital de las certificaciones académicas?
- ¿cómo puedo hacerlo? tengo una oferta de trabajo y quieren verificar la autenticidad de mi título.

Anexo III:

Puesta en marcha del proyecto

Anexo III: Puesta en marcha del proyecto

Para la replicación de la skill desarollada en este TFG, necesitaremos los archivos incluidos en la carpeta “código” entregada junto a esta memoria.

Para una primera parte deberemos entrar en la Alexa Developer Console y contar con una cuenta.

Una vez ahí deberemos crear una nueva Skill con el nombre que queramos, de tipo custom y con el backend gestionado por nosotros mismos. Tal cual se muestra en esta imagen:

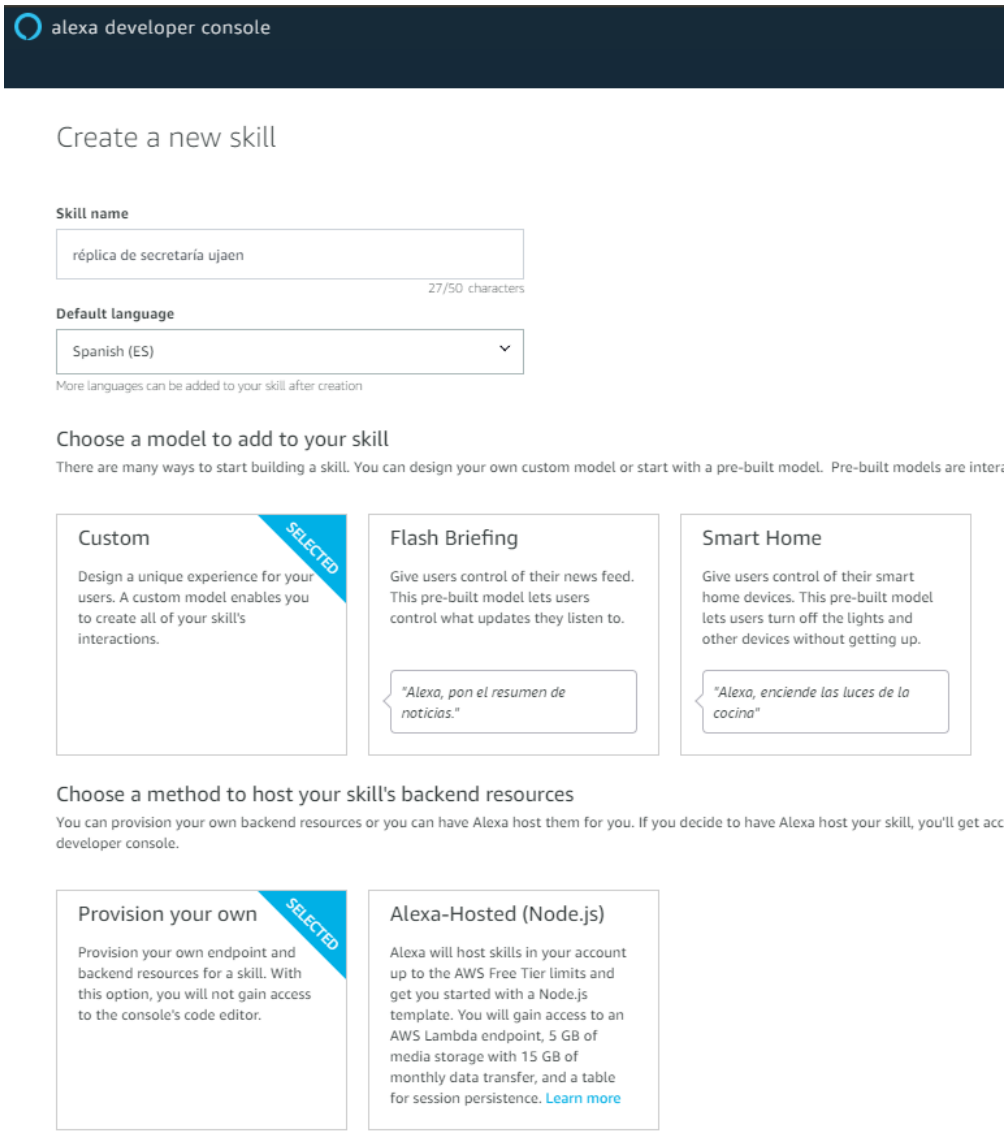


Ilustración 0-1. Creación de la Skill

Una vez aquí nos situaremos en el menú de la parte izquierda de la página.

Iremos a la pestaña de Interaction Model y luego a JSON Editor, como se muestra en la imagen:

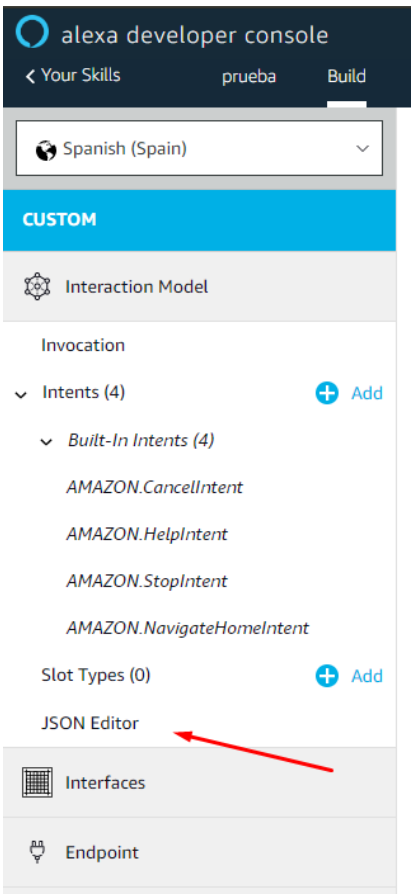


Ilustración 0-2. Muestra del Menú

En JSON Editor debemos importar el JSON que he incluido en la carpeta “código”.

Acto seguido nos diriguiremos a la pestaña EndPoint del menú de la izquierda.

Dejaremos esta pestaña abierta para seguir más adelante.

Nos diriguimos al servicio AWS y posteriormente a Lambda de Amazon, (nos registramos en caso de no tener una cuenta,) y nos aseguramos de que estamos en el servidor de Irlanda, ya que es el único que funciona en Europa actualmente. Para saber eso, basta con fijarnos en la parte derecha superior de la pantalla.

Una vez en AWS Lambda, debemos crear una nueva funcion, con las opciones:

- Crear desde cero
- El nombre que queramos
- En tiempo de ejecución debemos poner Python 3.6, ya que mi sistema está montado sobre este lenguaje.

Los permisos los podemos dejar predeterminados asi que pulsamos en Crear función.

Una vez estemos en la función, en la parte de diseño deberemos añadir un desencadenador, pulsamos esta opción y añadimos el “Alexa Skills Kit”

Ilustración 0-3. Muestra del desencadenador a añadir

Nos saldrá una opción de si queremos activar la verificación del ID de habilidad.

Esta parte es opcional, no afecta al funcionamiento, puedes hacerla o seleccionar “Deshabilitar”.

Para saber cual es el ID debemos ir a la Alexa Developer Console, abrir la página donde hemos creado la Skill y debajo de nuestra Skill creada nos aparecerá algo parecido a esto:

Ilustración 0-4. Imagen de donde se encuentra el ID

Simplemente copiamos y pegamos en el AWS.

Una vez añadido el desencadenador pulsamos en la parte donde pone el nombre de nuestra Skill, y nos aparecerá debajo el apartado código de la función tal que así:

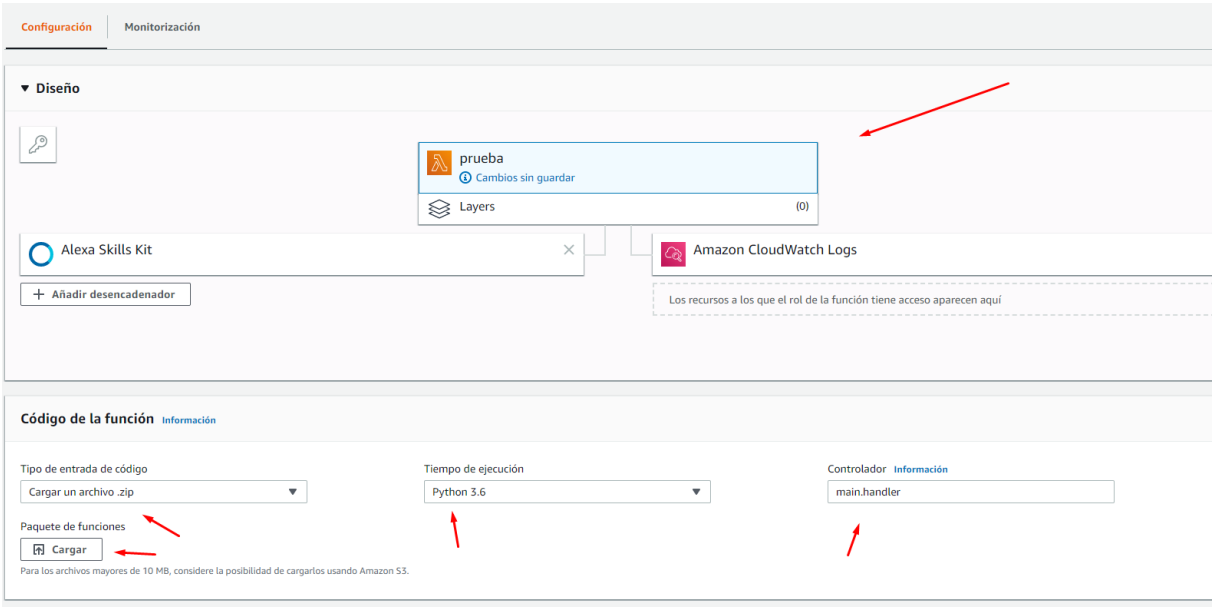


Ilustración 0-5. Configuración del AWS

En código de la función debemos cambiar el tiempo de ejecución a python 3.6 y debemos cambiar el tipo de entrada de código a “cargar un archivo zip”.

Una vez hecho cargaremos el archivo “main.zip” de mi carpeta “código”.

Y por último en controlador pondremos “main.handler”. En este caso se pone main, dado que mi script con los Intents y demás, se llama main.py.

Una vez listo, Guardamos con el botón de la parte superior derecha y justo encima vemos un apartado llamado ARN, pulsamos en el botón de justo al lado para copiar esa dirección.

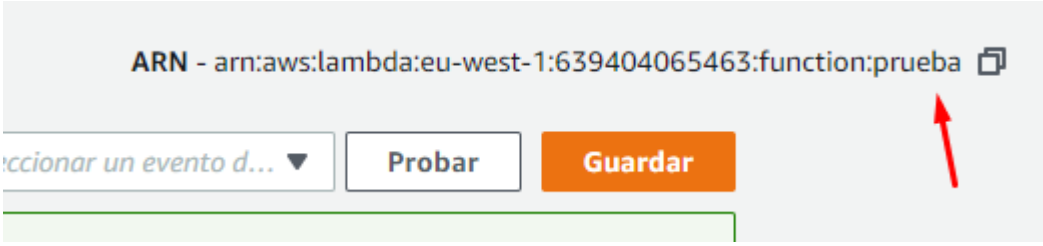


Ilustración 0-6. ARN

Ahora debemos volver a la Alexa Developer Console y en la sección endPoint antes mencionada añadimos nuestra ARN.

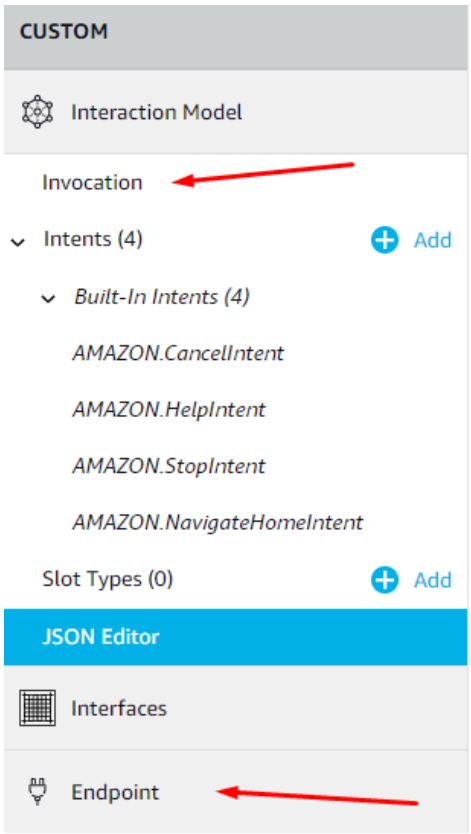


Ilustración 0-7. Menú izquierdo

Guardamos los cambios y Build Model.

Con esto ya tenemos lista la Skill. Podemos ir a la pestaña Test donde seleccionando el modo desarrollador ya podremos probar nuestra Skill.

Para invocar a la Skill deberemos decir o escribir “abre secretaría ujaen” y formular nuestra consulta.

Un ejemplo de interacción sería el siguiente:

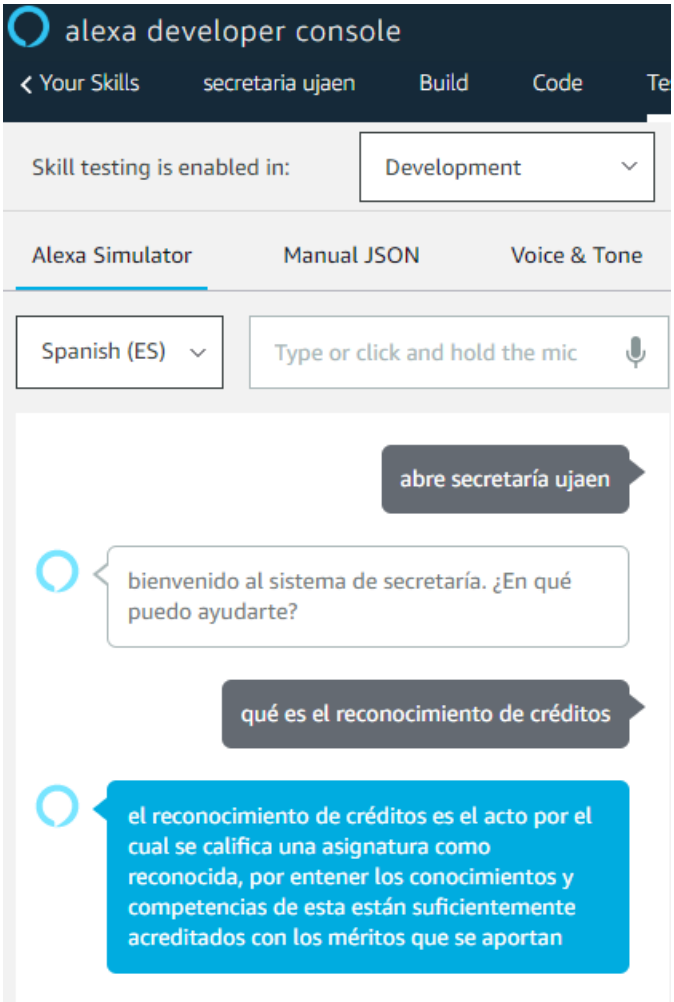


Ilustración 0-8. Ejemplo de interacción

Índice de ilustraciones

Anexo I:

Anexo III:

Índice de tablas: