Departamento de Ciencias da Computación e
Tecnoloxías da Información
Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN ENXEÑARÍA DO SOFTWARE

Asistente Virtual para sitios web

Estudante: Miguel Anxo Pérez Vila
Dirección: Paula López Otero
Javier Parapar López

A Coruña, setembro de 2019.

A mis padres

Agradecimientos

En primer lugar me gustaría agradecer a mis directores de proyecto, a Paula, por mostrarme su atención siempre que lo he requerido incluso estando lejos. A Javi, por su infinita dedicación y la ayuda que ha supuesto tanto profesional, como sobre todo, humana, yendo mucho más allá de sus obligaciones.

Quiero dar las gracias a mi familia. A mis padres, por notar su apoyo y aguantarme en todo momento, más aún en momentos que no han sido tan fáciles. A mi hermana, suponiendo un ejemplo para mí, superando sus metas y logrando vencer todos sus miedos. Mención especial para mi abuela, por notar su ilusión en cada una de las llamadas que compartimos, cargándome de energía y enseñándome a terminar las cosas de la mejor manera, incluso cuando las condiciones no son las ideales.

A los grandes amigos que he conocido durante la carrera, Pablo y David, por acompañarme en todas las clases y haber hecho una amistad única. A Josefina, gracias por ver ese potencial en mí, que muchas veces ni yo creía posible.

A mis compañeros de laboratorio, por aguantarme estos meses y ponerme las pilas cada vez que me distraía un poco. Alfonso y Manu, muchas gracias por vuestro ánimo.

Realmente siento que a partir de ahora tengo las cosas mucho más claras, es una verdadera suerte contar con vosotros, **gracias**.

Resumen

El objetivo de este trabajo fin de grado es el desarrollo e implementación de un asistente virtual para el sitio web de la Facultad de Informática de la Universidad de A Coruña (UDC). Los usuarios serán capaces de interactuar con el asistente, tanto por entrada de texto como a través de voz, y le podrán pedir información sobre cualquier apartado de la web. Para promover el uso generalizado del asistente, se encontrará disponible tanto en gallego como en español. El proyecto seguirá una metodología que incluye todos los ciclos que corresponden a un proyecto software, desde un análisis previo para conocer los objetivos principales del proyecto, planificación, pasando por el diseño, implementación y pruebas. Con este trabajo se ha conseguido un sistema que implementa y demuestra las funcionalidades de un sistema conversacional.

Abstract

The objective of this end-of-degree project is to develop and implement a virtual assistant for the website of the College of Informatic Engineers of the University of A Coruña (UDC). Users will be able to interact with the assistant, both by keyboard inputs and voice, so they will be able to ask for any information of the different sections of the website. In accordance with the promotion of the generalized use of the assistant, its use will be available for both Galician and Spanish. The project will follow a methodology that includes all the software steps that correspond to a project with this characteristics, from a analysis that allows to know the main objectives of the projet, planning, through the design, implementation and finally testing. Most of the development was carried out following a framework that was planned in the initial phase of the project. With this work, it was achieved a system that implements and demonstrated the functionalities of a question answering process with an virtual assistant as its interface.

Palabras clave:

- Sistema conversacional
- Chatbot
- Búsqueda de respuestas
- Rastreador web
- Recuperación de la información

Keywords:

- Conversational system
- Chatbot
- Question Classifier
- Crawling
- Information Retrieval

Índice general

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

1.1 Motivación

Gartner[1], empresa consultora y de investigación estadounidense con sede en Stanford, predijo que, en el año 2020, en un día normal, una persona podrá llegar a tener más conversaciones con *Chatbots* que con su cónyuge [1]. Es un buen ejemplo de cómo los sistemas conversacionales se están asentando cada vez más en nuestro día a día.

Kleiner Perkins[2], empresa americana especializada en inversión de *start-ups*, ha revelado que el 80% del tiempo que dedicamos a los móviles va destinado a solo tres aplicaciones, Facebook Messenger, WhatsApp y el uso del navegador web, además de que este porcentaje es mucho mayor en los usuarios más jóvenes. Las interfaces conversacionales están presentes en dos de estos tres servicios, mediante *Facebook Messenger* y sistemas de atención al cliente vía Whatsapp [2]. Los sistemas de interacción por voz se han convertido en una funcionalidad principal en todos los dispositivos tecnológicos actuales, desde los ordenadores hasta los móviles y tablets [3].

Los gigantes tecnológicos llevan años mejorando sus asistentes software inteligentes para la gran mayoría de sus productos. Google[3] proporciona su propio asistente para todos sus productos, *Google Assistant*. Otro famoso ejemplo es Cortana, elaborado por Microsoft[4] como ayuda en el sistema operativo Windows; Apple[5] también ha desarrollado un asistente por voz para sus móviles y ordenadores, Siri; también se encuentran en Amazon[6] desarrollando su propio asistente *Amazon Alexa*. No solo los encontramos en dispositivos, si no que tienen gran presencia en importantes sitios web, presentes en infinidad de páginas web de empresas, proporcionando diferentes servicios a los clientes.

[1] https://www.gartner.com/smarterwithgartner/
[2] https://www.kleinerperkins.com/
[3] https://www.google.com/
[4] https://www.microsoft.com/
[5] https://www.apple.com/
[6] https://www.amazon.com/

Todos estos productos son capaces de responder a una gran cantidad de preguntas y están integrados con diferentes aplicaciones para tener la capacidad de reproducir música, crear listas de comprobación, establecer alarmas, reproducir audiolibros, proveer información en tiempo real del clima, tráfico o noticias. La compañía de investigación de mercado eMarketer[7] estableció que casi el 20% de los ciudadanos estadounidenses usarán un asistente inteligente al menos una vez al mes durante este año, unos datos muy positivos para una tecnología que lleva presente tan poco tiempo [4].

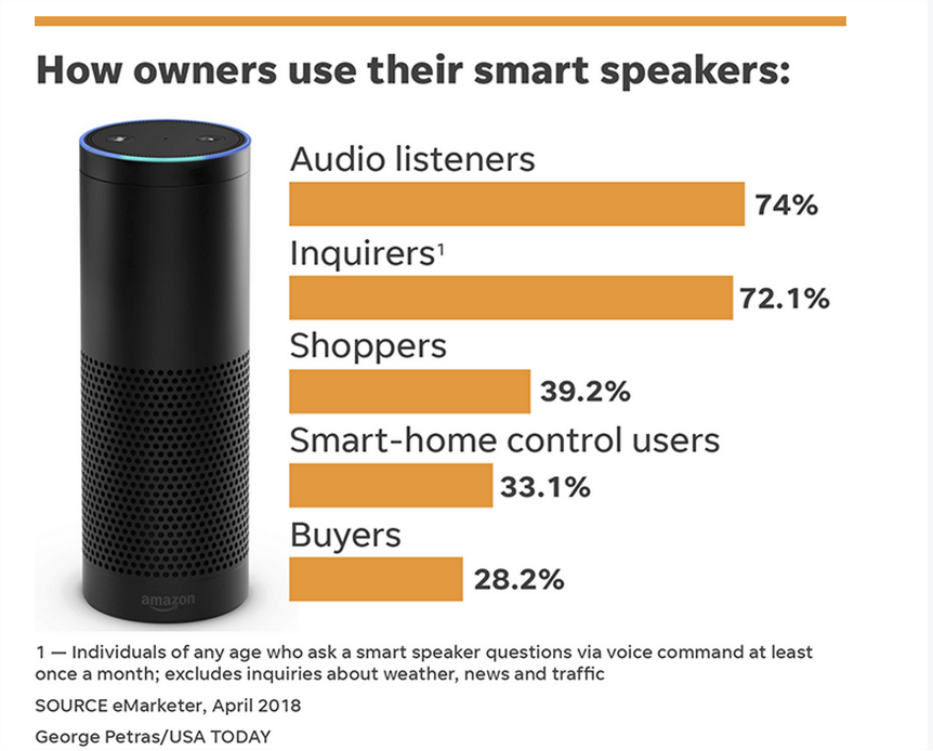

Figura 1.1: Estadística de eMarketer sobre el uso de los asistentes por voz inteligentes

Existe una tendencia real a moverse hacia los controles por audio y alejarse de las interfaces táctiles, lo que está orientando a las empresas a invertir más recursos en el aspecto auditivo que en el aspecto visual. Sin ir más lejos, Amazon tiene más de 10.000 personas trabajando para su asistente inteligente [5], trabajando en departamentos de bases de datos, técnicas de aprendizaje automático o técnicas de mejora de su habilidad conversacional [6].

Este tipo de servicios han supuesto un medio para las compañías de conocer mucho mejor a los usuarios que interaccionan con sus productos. Al interactuar de manera directa con los clientes, las empresas tienen a su disposición una gran cantidad de datos sobre las inquietudades y preferencias de los usuarios, es por esto que los asistentes virtuales también están en

[7] https://www.emarketer.com/

lo alto de las nuevas estrategias de mercado. Son una herramienta única para analizar oportunidades de negocio, mejorar la experiencia de sus usuarios y ser conscientes de cómo enfocar su negocio. Debido a toda esta información resulta totalmente lógico el auge de los asistentes virtuales y el interés de las empresas en integrar este tipo de servicios. También es fundamental destacar que estas herramientas son verdaderamente útiles para personas con diversidad funcional, ya que les facilitan el accesso a información comparado con la tecnología táctil a la que suelen estar habituados.

Una de las principales virtudes de este proyecto y que lo hace diferente al desarrollo de la mayoría de los asistentes actuales es el hecho de que se emplean técnicas de recuperación de la información para encontrar respuestas a las preguntas. La mayor parte de los asistentes que se pueden encontrar en sitios web están construidos con herramientas tipo *IBM Watson* o *DialogFlow*. Esto significa que usan aprendizaje automático, analizan tus datos y van aprendiendo a medida que el usuario introduce preguntas; como es el caso de *Anna* para la compañía sueca Ikea[8] o el asistente de la empresa ferroviaria Renfe[9]. Sin embargo, este asistente es 100% no supervisado, no necesita un aprendizaje para devolver respuestas óptimas.

Todos estos motivos, junto con la curiosidad personal por la integración de los procesos de transcripción de audio, han sido unas de las motivaciones principales a la hora de decidirnos por este trabajo.

1.2 Objetivos

El objetivo principal de este proyecto es desarrollar un asistente virtual que resuelva las dudas de los usuarios en la página web de la Facultad de Infórmatica de la UDC. Dentro de las numerosas funcionalidades que permitiría realizar la aplicación, se ha considerado como objetivo básico permitir la resolución de cualquier tipo de información que los usuarios requieran y que se encuentre dentro del dominio de la facultad.

Una de las finalidades del trabajo es que el asistente sea capaz de comprender tanto el gallego como el castellano. Las respuestas han de ser devueltas en tiempo real y de la manera más completa y relevante posible. Los usuarios tienen la opción de preguntar tanto por texto como por voz. Dependiendo del tipo de cuestión, el asistente deberá decidir cuál es el mejor tipo de respuesta a enviar, desde una simple frase o el enlace hacia un sitio web, hasta una fecha, como podrían ser cuestiones relacionadas con horarios de exámenes, una charla que se impartirá y que se encuentra promocionada por la facultad o la hora de un examen. En general, el sistema estará preparado para el análisis de distintos tipos de preguntas.

En relación con la recepción de la voz, un objetivo marcado es la correcta gestión del ruido que pueda llegar a existir en las entradas; en el proyecto siempre se han tenido en cuenta

[8] https://www.ikea.com/ms/en_JP/customer_service/splash.html
[9] https://consulta.renfe.com/irene/main

posibles complicaciones en el entendimiento de ciertos audios y se ha desarrollado la lógica del proyecto en base al tratamiento de este tipo de errores.

1.3 Estructura de la memoria

La memoria del actual proyecto contiene la siguiente estructura:

- **Introducción** Contextualiza y explica el entorno en el que está situado el proyecto. Se describen las razones que motivaron a su construcción, además de mencionar los objetivos generales del mismo. Contiene la estructura de la memoria y el plan de trabajo que se ha realizado.

- **Fundamentos y conceptos principales** Comprende un resumen de la base teórica sobre la que se sustenta el trabajo. Se hace una pequeña explicación de las técnicas que se han utilizado en el posterior desarrollo, en el que se pueden ver los aportes al conocimiento que han surgido a partir del aprendizaje de estas técnicas.

- **Herramientas y tecnología** En este capítulo se describen las tecnologías y herramientas utilizadas en este proyecto, la mayoría de ellas novedosas para el autor, puesto que no se habían visto durante la carrera, incluyendo una justificación de su uso describiendo lo que nos han aportado al proyecto.

- **Planificación y metodología** Se describen las tareas necesarias para la consecución de los objetivos del proyecto, además de los recursos que han sido necesarios para hacerlo, incluyendo los costes y la gestión de riesgos. También incluye una descripción de la metodología que se ha seguido durante el desarrollo y su adaptación al proyecto.

- **Análisis** Descripción general de los requisitos del proyecto, comprendiendo tanto los funcionales como los no funcionales en diferentes niveles de abstracción. Se incluye una descripción de alto nivel de la arquitectura del sistema.

- **Desarrollo** Detalla el proceso seguido durante la implementación del proyecto, con una pequeña explicación de cada *Sprint*.

- **Conclusiones y trabajo futuro** Describe el grado de cumplimiento de los objetivos propuestos al empezar el proyecto y la utilidad de la herramienta en un plano más práctico. Se detallan los aspectos tanto positivos como negativos adquiridos durante la implementación del proyecto y su relación con la carrera. Por último, se mencionan posibles ampliaciones del proyecto en un futuro.

- **Apéndices:**
  -*Lista de acrónimos y glosario*: Términos y acrónimos técnicos usados en la memoria del proyecto

  -*Bibliografía*: Documentación bibliográfica utilizada en el proyecto.

1.4 Plan de trabajo

Se han seguido distintas fases relacionadas con la metodología empleada:

- Estudio previos sobre distintos conceptos necesarios para la ejecución del proyecto:
  - Web Crawling.
  - Sistemas de búsqueda de la información.
  - Búsqueda de respuestas *Question Answering*
  - Sistemas conversacionales.
  - Tratamiento de voz : ASR y TTS.
  - Búsqueda de información degradada.
- Como se ha seguido una metodología ágil e incremental, en cada de las iteraciones se ha seguido el siguiente procedimiento.
  - Administrar la realización de las tareas correspondientes a cada iteración.
  - Desarrollo de las mismas.
  - Pruebas y análisis del trabajo realizado.

Capítulo 2

Fundamentos y conceptos principales

En este capítulo se explican los conceptos relacionados con la elaboración de este proyecto. El autor ha necesitado realizar un estudio sobre los mismos y obtener así el conocimiento necesario para su posterior desarrollo. Se menciona cada uno de ellos, junto con una pequeña presentación. De esta manera, el lector estará familiarizado con estos fundamentos y le facilitará la lectura del resto de la documentación.

2.1 Recuperación de la información

2.1.1 Búsqueda de respuestas - Question Answering

La recuperación de la información (IR) es la ciencia que trata la búsqueda de información en cualquier tipo de colección de documentos digital, incluyendo la búsqueda dentro de los mismos. Su objetivo es satisfacer las necesidades de información de los usuarios, permitiendo a estos acceder de una manera sencilla a la información requerida [7]. Utiliza una gran variedad de disciplinas, como arquitectura de la información, diseño, inteligencia artificial, entre otros. Los motores de búsqueda web son probablemente el producto más popular de aplicaciones dentro del campo de la recuperación de la información, como *Google*, *Bing* o *Baidu*.

Búsqueda de respuestas, más conocido por su término inglés Question Answering (QA) es un tipo de recuperación de la información. Su función consiste en que, dada una cantidad (normalmente grande) de documentos, como puede ser la World Wide Web, poder ser capaz de recuperar respuestas a preguntas planteadas en lenguaje natural. La complejidad radica en que, para encontrar la respuesta exacta, no solo es una búsqueda de documentos, ya que es necesario buscar en el propio contenido de estos documentos, en el segmento donde se encuentra la información. Debido al análisis de preguntas, *Question Answering* requiere una

tecnología de procesamiento de lenguaje natural compleja y es considerada como un paso por delante de la tecnología del buscador.

Una de las dificultades de *Question Answering* es la gran variedad de cuestiones que pueden ser planteadas. El dominio sobre el que realizar estas búsquedas puede ser muy variable, en algunos sistemas simplemente son pequeñas colecciones de documentos, en otras pueden llegar a ser una cantidad inmensa de páginas web. Se suelen distinguir dos tipos de QA:

Dominio cerrado Los sistemas responden a preguntas sobre un dominio concreto, como es el caso del sistema de búsquedas del proyecto. Solo se responde a preguntas sobre temas relacionados con la Facultad de Informática. Algunos sistemas de este tipo también mantienen restricciones sobre el tipo de preguntas a formular.

Dominio abierto Los sistemas responden a preguntas sobre cualquier tema. Estos sistemas tienen una base de información mucho mayor, ya que abarcan un mayor número de tipos de preguntas.

Muchos de los primeros programas de Inteligencia Artificial eran en realidad sistemas *Question Answering*. Uno de los más destacados es el LUNAR [8], que fue creado en los años 60. Este sistema respondía a preguntas sobre el análisis geológico de las rocas conservadas en las misiones del Apolo a la Luna. Consiguió responder al 90% de las preguntas. El éxito de este proyecto fue debido a que era un dominio muy específico y contenía reglas simples para establecer en una máquina. Posteriormente se crearían otros sistemas de QA, pero todos con un dominio bastante restringido.

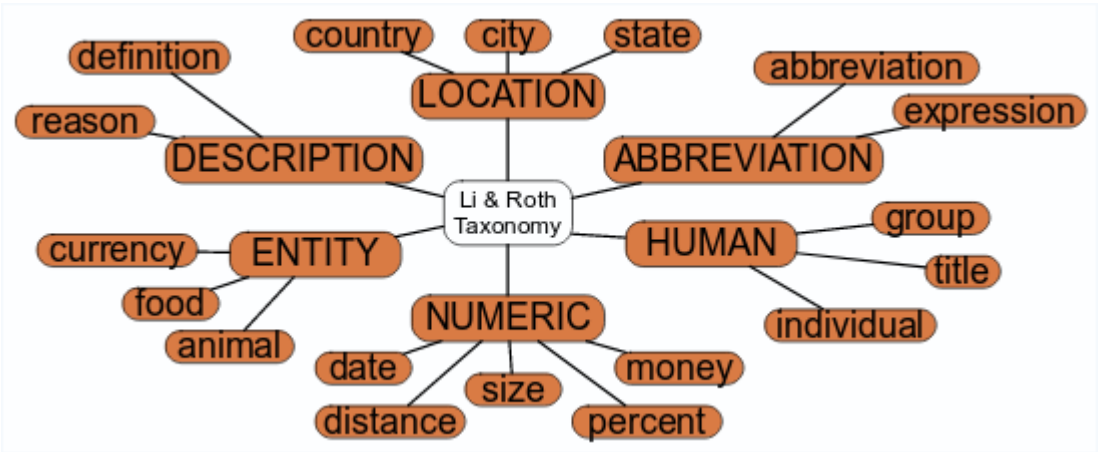

Figura 2.1: Ejemplo de un conjunto de clasificación de preguntas en un sistema QA [9]

Los sistemas de QA fueron evolucionando en los 70 y 80, mejorando cada vez más las respuestas, ampliando y abarcando diferentes dominios. La principal diferencia con los sistemas

actuales es que estos programas seguían consultando sobre estructuras de datos conocidas y organizadas. A partir del 2001, estos sistemas solían incluir un clasificador de preguntas *Question classifier* en su arquitectura, tal como el de la figura 2.1. Siguiendo una arquitectura multiagente, donde cada dominio a tratar es representado por un agente que intenta contestar preguntas utilizando sus datos específicos. Mientras tanto, un meta-agente sería el encargado de organizar las respuestas de los agentes y elegir la de mayor relevancia o puntuación [10].

En la actualidad, el sistema de inteligencia artificial más famoso capaz de responder preguntas formuladas en lenguaje natural es el superordenador *Watson*, desarrollado por la empresa estadounidense IBM[1]. Su proceso de búsqueda se realiza sobre una base de datos local, que contiene información de multitud de fuentes, como diccionarios, artículos de noticias, libros, enciclopedias, entre otros. Para probar su capacidad práctica, en 2011 participó en un concurso de televisión respondiendo preguntas frente a dos rivales humanos especialistas en este tipo de concursos, por lo que tenía que tener gran acierto y sobre todo rapidez a la hora de contestar las preguntas. Derrotó a ambos sin muchas dificultades.

Los sistemas actuales deben cubrir repositorios muy grandes de documentos, por lo que resulta razonable que el rendimiento de un sistema de *question-answering* sea muy dependiente de un buen motor de búsqueda, ya que es el encargado de seleccionar los documentos. A la hora de encontrar la mejor respuesta posible, es lógico que cuanto mayor sea la colección de documentos, mayores son las probabilidades de encontrar la respuesta idónea. Esto puede suponer que exista algo de redundancia, que en este caso significaría que la respuesta esté en varios sitios distintos, lo que supone dos ventajas:

- Al aparecer en varios lugares, las respuestas correctas se pueden filtrar y así minimizar el riesgo de falsos positivos.
- Reducir la carga en los sistemas de procesamiento de lenguaje.

2.1.2 Web Crawler

Un Web Crawler, también conocido como araña y denominado comúnmente como *Crawler*, término inglés, es un programa para la inspección sistemática de la World Wide Web, normalmente utilizado para alimentar al indexador de un sistema de búsqueda con los contenidos de sitios web.

El término *Crawler* proviene del primer motor de búsqueda de Internet, el Web Crawler. Actualmente, el crawler más famoso es Googlebot, diseñado por Google para recolectar documentos desde la Web, lo que construye la base de datos para su motor de búsqueda.Es un hecho que en los últimos años el crecimiento de Internet es incontrolable, tanto que hasta se

[1]https://www.ibm.com/es-es

podría considerar que el número de sitios web es infinito. Además, uno de los principales rasgos de Internet es su ausencia de organización centralizada. Estas dos características son los mayores inconvenientes a la hora de querer localizar información específica en ella. Debido a este crecimiento un crawler no puede ser capaz de analizar todos los sitios web existentes. Es por esto que en un buen crawler resulta crucial una elaboración eficiente de los filtros para detectar las páginas con mayor relevancia durante el recorrido, al igual que la información existente en estas. En este proyecto se ha hecho uso de la tecnología de Web Crawling para analizar y almacenar la información de la página de la facultad. En la arquitectura de nuestro asistente virtual, comentada en su respectiva sección, se hace uso de un almacén de datos donde se buscan las respuestas a enviar. Se ha desarrollado un Crawler para almacenar esta base de datos de una manera estructurada, y permitir realizar el proceso de búsqueda de una manera eficiente.

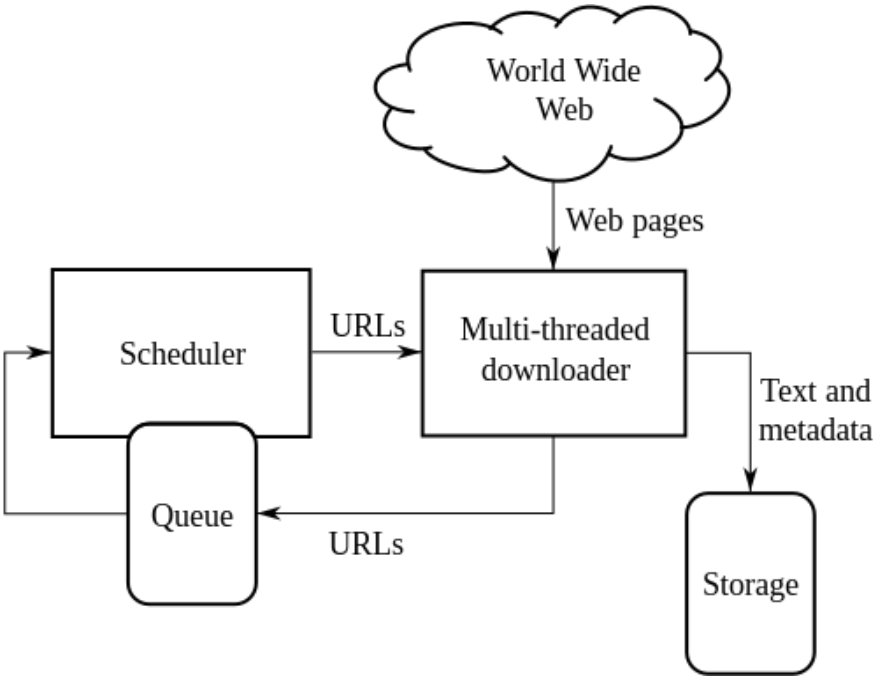

Figura 2.2: Arquitectura típica de un Web Crawler

Normalmente los *Crawler* parten de un conjunto inicial de URL's, conocidas como semillas. En este caso, se ha partido de la dirección *https://www.fic.udc.es/*. En la lógica de la inspección de estos sitios web, es importante indicar qué tipo de información se quiere almacenar [11]. En la sección de desarrollo se explica con mayor profundidad los campos creados y que información almacenan.

El crawler identifica los hiperenlaces en las páginas visitadas y los añade a la lista de URL's a inspeccionar de manera recurrente, este proceso se hace de acuerdo a una serie de reglas.

Estas reglas, definidas por el desarrollador, sirven para limitar qué dominios web visitar (en el caso del Crawler realizado, solo hemos permitido análisis de el dominio *fic.udc.es*), a qué tipo de información acceder y como almacenarla.

2.1.3 Procesos de búsqueda con información degradada

Los modelos tradicionales de búsqueda de consultas en documentos se basan en la comparación entre la propia consulta y los términos de los documentos. Este tipo de comparación simple, no sería del todo efectiva en el contexto de nuestro proyecto, ya que los usuarios tienen la opción de preguntar mediante su micrófono, por lo que el reconomiento de voz mediante ASR puede generar diversas incorrecciones a la hora de resolver la búsqueda. Esto es debido a que el proceso de transcripción puede llegar a fallar en ciertas palabras, bien porque ha sido un error del sistema o que el usuario no ha pronunciado de manera correcta, lo que da lugar a que algunas palabras no se guarden exactamente como se escriben.

Los errores en las transcripciones de consultas conducen a un rendimiento de búsqueda degradado al comparar consultas habladas y escritas. El uso de transcribir las consultas en sub-palabras es muy común para afrontar los problemas ocasionados por el ruido; también es usado en otros ámbitos como en la extracción de información en páginas web o para detectar plagio en artículos [12]. Existen distintas aproximaciones al problema; en este proyecto se ha tokenizado todos los campos en n-gramas, un n-grama basicamente es una subsecuencia de n elementos de una secuencia dada:

Full-text-search. Se busca el término sin ningún tipo de variación, exceptuando un filtro para que no sea *case-sensitive*; de esta manera el sistema considera la palabra ”Horario” y ”horario” como términos iguales.

Mediante n-gramas. Cada campo del índice se tokeniza en n-gramas. El texto ”Clases de herramientas” quedaría tokenizado de manera {cla, las, ase , ses, de, her, err, rra, ram , ami , mie , ien , ent, nta , tas} para los 3-gramas.

En el propio proyecto, el análisis de los documentos se ha realizado mediante ambas técnicas. En busca de una mejora de la relevancia en las respuestas, las consultas han combinado tanto la búsqueda de n-gramas de tres y cuatro caracteres de tamaño, como de las palabras en su totalidad. De esta manera, si el sistema ha transcrito *”Closes de herramentas”* debido a errores relacionados con el ruido, la división en n-gramas permitirá reducir el impacto de estos errores, y la consulta devolverá resultados similares a si hubiese preguntado *”Clases de herramientas”*.

2.2 Sistemas conversacionales

Un robot conversacional es un tipo de *ChatBot* que posibilita realizar una conversación con él de forma textual o mediante lenguaje natural [13]. La mayoría son empleados como asistentes virtuales para proporcionar servicios, como servicios de atención al cliente o como forma de obtención de información sobre algún sitio web.

Algunos de estos sistemas emplean procesamientos de lenguaje natural sofisticados pero muchos de ellos simplemente examinan las palabras de entrada y las analizan con un simple patrón en su base de datos, conservando una lógica simple para el procesamiento de la respuesta. Hoy en día existen un gran número de categorías para clasificar los asistentes virtuales, puesto que abarcan muchos ámbitos.

En 1950, Alan Turing publicó el famoso artículo "Computing Machinery and Intelligence", en el que se cuestionaba si las máquinas podían llegar a pensar [14]. A partir de esta publicación ganó una gran fama el llamado *Test de Turing*, hasta el punto que existe un premio anual para el sistema que mejor cumple este Test[2]. Es una prueba sencilla de habilidad en la que se mide la capacidad de estos sistemas de simular un comportamiento idéntico al de un ser humano. Un juez se enfrenta a dos pantallas de ordenador, una de ellas que se encuentra bajo el control de una persona y la otra controlada por el programa. El juez plantea preguntas a ambos,y basándose en sus respuestas, debe decidir qué pantalla está controlada por el ser humano y cuál por el programa.

Tal y como mencionamos en la sección de *Question Answering*, el sistema con mayor capacidad de responder a preguntas sobre cualquier tipo de ámbito es el superordenador *Watson*. Según IBM, el uso en el futuro de *Watson* será facilitar que los dispositivos como ordenadores y móviles comiencen a interactuar de forma natural con humanos con mucha mayor frecuencia, a través de una amplia gama de aplicaciones, dando respuestas a los seres humanos que puedan comprenden y satisfacer sus servicios [15]. Desde el punto de vista empresarial, la integración de sistemas capaces de interactuar con lenguaje natural y responder a un gran dominio de preguntas presentan numerosas ventajas:

Reducción de costes Eliminan la obligación de cualquier presencia de trabajadores durante la interacción. Sobre todo en aquellos tipos de sitios web donde suelen repetirse las mismas preguntas.

Atención 24/7 Una vez instalado está accesible en cualquier momento, sin necesidad de establecer horarios de atención al cliente.

Factor humano Proporcionan situaciones similares a como si estuvieses hablando con una persona, lo que los hace más intuitivos. Además es mucho más veloz y menos tedioso

[2] http://www.aisb.org.uk/events/loebner-prize

que, por ejemplo, tener que llamar al teléfono de atención al cliente.

Ayuda personalizada Al poder acceder a los datos del propio cliente que está preguntando, en muchos negocios es una ventaja para proporcionarles información relevante y adaptada a ellos.

El proceso que sigue un sistema conversacional para generar sus mensajes, no difiere tanto al proceso de comunicación con otra persona. Por ejemplo, si un amigo nos pide que le vayamos a comprar una rueda de bici, nuestra primera pregunta será ¿De cuánta anchura?, ¿De qué material? En cuanto tenemos claros los detalles, tendremos que informarnos sobre qué tiendas venden el tipo específico de ruedas que estamos buscando. Una vez buscadas, vamos a la tienda y le comentamos a nuestro amigo lo que hemos encontrado.

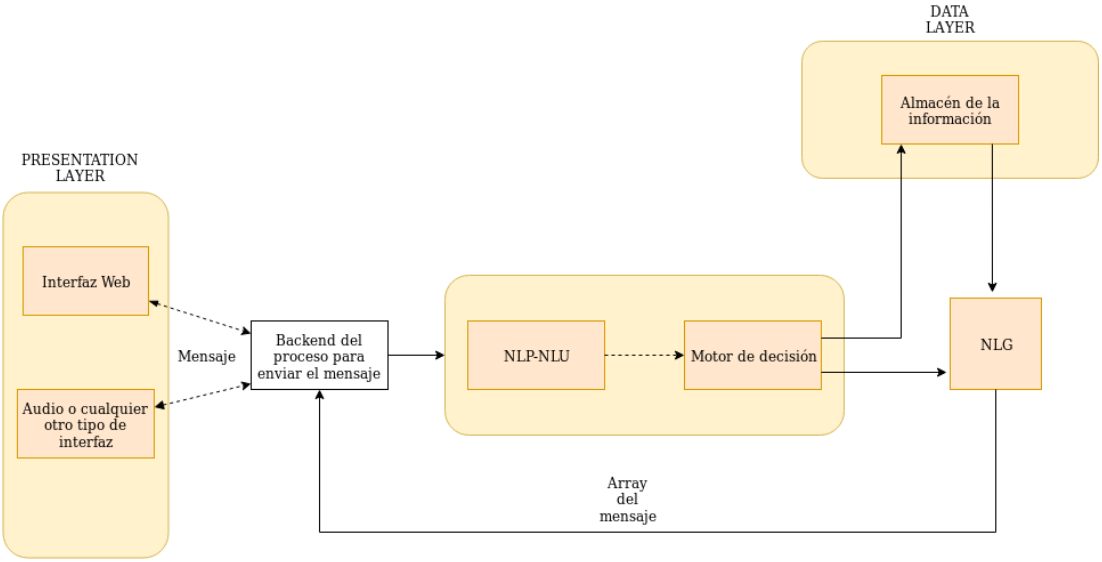

Figura 2.3: Arquitectura típica de un ChatBot

Palabras clave:

- *NLP (Natural Language Processing)* Describe la capacidad del sistema de recibir lo que se dice, descomponerlo para su entendimiento y determinar la acción más adecuada.
- *NLU (Natural Language Understanding)* Parte del *NLP* que se encarga de manejar las entradas no estructuradas.
- *NLG (Natural Language Generation)* Proceso de escribir el lenguaje por el propio sistema.

Siguiendo el ejemplo anterior, el ChatBot recogerá la frase de comprar una rueda de bicicleta. Mediante el proceso *NLU* convertirá el texto en datos estructurados, de manera que

el sistema ya comprenderá la información requerida. Analizará esta información el motor de decisión, que en nuestro ejemplo, sería similar a plantearse las preguntas ¿De cuánta anchura?, ¿De qué material?, mediante el *NLG*, pueden ser capaces de pedirte información adicional. Una vez tiene todos los datos necesarios, el motor buscará las tiendas más cercanas que disponen del producto requerido con las características más similares. En este punto, hasta te podrá redirigir hacia el sitio web de la tienda para procesar el pago de la rueda.

Evidentemente hay muchos matices y diferencias entre ambos casos. Sin embargo, cada vez se están realizando más avances y las máquinas entienden cada vez mejor las peticiones de los humanos.

2.3 Sistemas de tratamiento de voz

2.3.1 Motores ASR

ASR (*Automatic Speech Recognition*) son las siglas en inglés para referirse a los sistemas de reconocimiento automático de voz. También conocidos como sistemas RAH (*Reconocimiento automático del habla*).

El reconocimiento automático del habla es una disciplina de la inteligencia artificial que tiene como objetivo permitir la comunicación hablada entre seres humanos y computadoras. La labor de un sistema de reconocimiento de voz es llegar a obtener una interpretación adecuada del mensaje acústico recibido, llegando a minimizar el número de errores y ambigüedades en la transcripción.

Alrededor de 1950 apareció el primer sistema de reconocimiento de números, un dispositivo completamente cableado, antes de que los sistemas de reconocimiento se convirtieran en programas implementados por ordenadores. El reconocimiento automático de voz da lugar hoy en día a un importante conjunto de funciones de naturaleza y dificultad muy variadas, utilizadas diariamente en millones de aplicaciones por personas de todo el mundo [16].

Los motores ASR contienen dos características para maximizar su eficacia. La primera es la presencia de identificadores que detectan el sonido de la voz de una persona, la segunda es la presencia de minimizadores de ruido, para eliminar cualquier sonido innecesario. Ciertos sistemas de reconocimiento también requieren de procesos de entrenamiento. En estos, una persona lee un texto o vocabulario específico para añadirlo al sistema, el sistema lo analiza y ajusta mejor el reconocimiento del habla de esa persona, lo que da lugar a un incremento de la precisión.

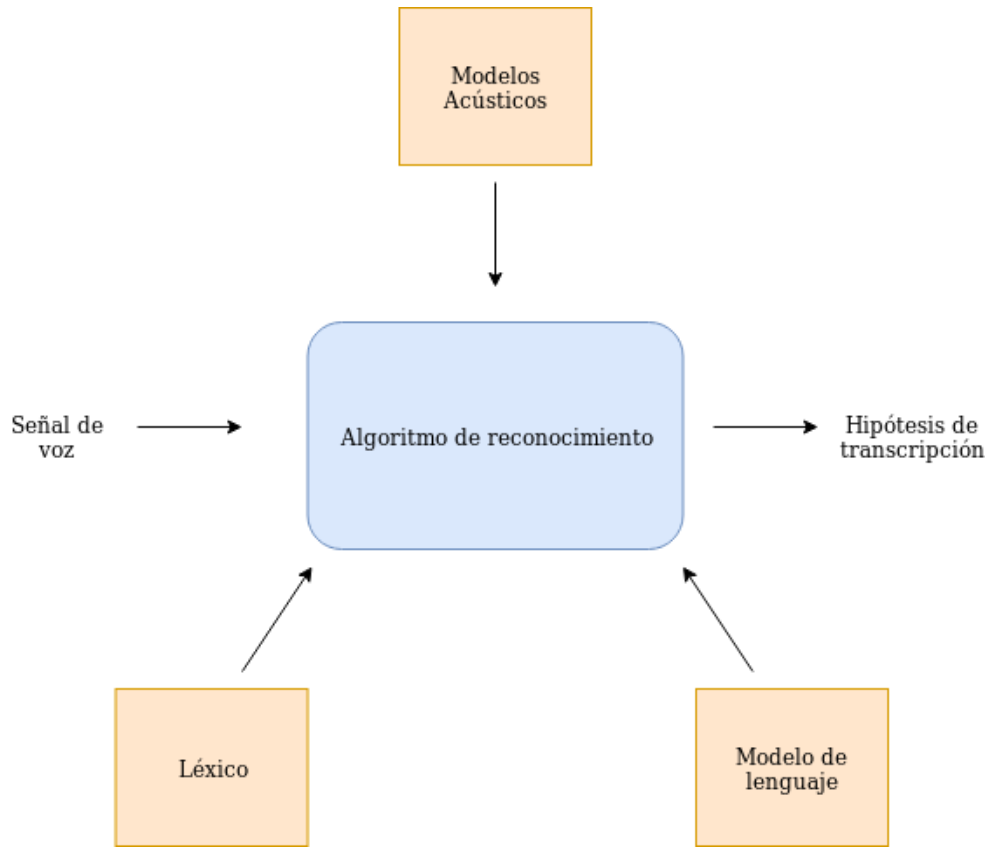

Figura 2.4: Arquitectura base de los sistemas ASR
[17]

En el caso del motor ASR usado para el proyecto es necesario ensayar los modelos con los datos de entrenamiento. El proceso de entrenamiento comienza ordenando los datos de audio transcritos en un orden específico, de acuerdo con la documentación[3].
Cuando ya se encuentran ordenados, hay que elaborar un diccionario. Los diccionarios contienen la transcripción fonética de las palabras que conforman el vocabulario, es decir, determinan como son pronunciadas. Un ejemplo de las palabras por las que está compuesto un diccionario serían:

horario -> *o'rario*

clases -> *'klases*

aula -> *'aula*

informática -> *iNfoR'matika*

[3]http://kaldi-asr.org/doc/data_prep.html

Una vez completado el diccionario, el modelo ya está preparado para someterlo a entrenamiento. Los modelos de lenguaje junto con los acústicos serán usados en el algoritmo de reconocimiento que compone el núcleo del sistema, además del léxico o vocabulario. Los ficheros de audios serán las entradas de este proceso y el sistema devolverá la hipótesis en formato texto de lo que ha conseguido transcribir.

2.3.2 Motores TTS

Un sistema *text-to-speech* (TTS) convierte un texto en habla, es decir, se obtiene artificialmente habla sintetizada. Los sistemas TTS incorporan conocimiento de lingüística y procesado de señal.

La calidad de la voz generada mediante un sistema *text-to-speech* depende de diversos factores, como su intenligibilidad, naturalidad del habla, en definitiva, que sea fácil su comprensión por parte de los oyentes [18]. Los programas text-to-speech son de gran ayuda para las personas con problemas visuales o con algún tipo de discapacidad que les impida leer o ver correctamente; es por eso que la mayoría de dispositivos en la actualidad cuentan con funcionalidades para convertir texto a voz.

Capítulo 3

Herramientas y tecnología

Este capítulo contiene la justificación de las elecciones tecnológicas y la descripción de las herramientas más importantes utilizadas en el desarrollo del sistema. Gracias a la utilización de estas herramientas se ha llevado a cabo la construcción de las distintas iteraciones del proyecto.

3.1 Lenguajes de programación

La elección de los lenguajes de programación en los que se elaborado el desarrollo viene justificado por el tipo de aplicación que se ha desarrollado. A continuación, se mencionarán los lenguajes utilizados durante el proyecto.

Python

Python es un lenguaje de programación interpretado. Está basado en una sintaxis que favorece un código legible y fácil de entender. Se trata de un lenguaje multiparadigma, pensado para dar soporte a orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, de tipado fuerte y dinámico, utilizado en una gran cantidad de proyectos.

Python ha sido empleado en dos diferentes fases del proyecto. El principal motivo de su uso ha sido por la gran cantidad de librerías que aporta y la facilidad de su empleo. Python contiene soporte a librerías que han servido para elaborar un *crawling* completo que rastree el dominio de la facultad. Gracias a estas librerías se ha permitido almacenar la información que contiene la página web de la facultad, para posteriormente indexarla y buscar en ella[1].

[1] https://www.python.org/about/apps/

JavaScript, HTML, AJAX, CSS

JavaScript[2] y AJAX[3] se emplean para construir el cliente. Se lee el audio o texto por parte del usuario y se hace una petición al servidor enviando los datos, esta comunicación desde el navegador hacia el servidor es mantenida gracias a AJAX, de esta forma nos permite realizar cambios en el asistente sin necesidad de recargar la página en ningún momento. Ha posibilitado mejorar la velocidad y usabilidad de la aplicación. También se ha usado CSS[4] para el diseño de la ventana del asistente.

3.2 Librerías y componentes

Scrapy

Scrapy es un framework utilizado para el web-crawling escrito en Python, aunque también se usa para extraer información usando API's. Su arquitectura está construida entorno a Spiders. Un Spider se podría considerar como un rastreador automático de sitios web que recibe y ejecuta un conjunto de instrucciones.

Se ha importado su librería en código python para extraer información estructurada de los enlaces de la facultad. Nos ha permitido organizar los datos en diversos campos, facilitando el almacenamiento de la información [5].

BeautifulSoup

BeautifulSoup es una biblioteca de Python para analizar documentos HTML. Su función es crear un árbol con todos los elementos del documento HTML y a partir de él extraer su información. Ha sido la biblioteca utilizada para extraer el texto en cada una de las páginas de la facultad. Gracias a su uso nos ha permitido obtener el texto e implementar las modificaciones correspondientes para facilitar su lectura[6].

Goose3

Es otra biblioteca de Python pensada para extraer información sobre noticias en medios digitales. No solo extrae campos importantes como el propio texto del artículo, sino también sus imágenes principales, cualquier vídeo embebido, así como metadatos de los sitios web. A pesar de que el contenido de la página de la facultad no contiene artículos, esta biblioteca ha sido aprovechada para obtener de manera correcta el título de cada página, así como un campo

[2] https://www.javascript.com/
[3] https://www.w3schools.com/xml/ajax_intro.asp
[4] https://www.w3.org/Style/CSS/
[5] http://docs.scrapy.org/en/latest/intro/overview.html
[6] https://www.crummy.com/software/BeautifulSoup/bs4/doc/

de texto filtrado. Este campo optimiza la información y no guarda información redudante o en blanco y guarda las frases que considera realmente relevantes respecto a los temas tratados en el artículo[7].

BotUI

BotUI[8] es una API de Javascript muy intuititiva para interfaces conversacionales la cual nos permite añadir mensajes y proporcionar al usuario realizar distintas acciones. Formada por tres conceptos principales, *mensaje*, *acción* y *Uso del "then"*, ha sido la API sobre la que se ha basado la interfaz del proyecto[9].

Figura 3.1: Ejemplo de interfaz usando la API de BotUI

3.3 Herramientas de Desarrollo

Servicio Rest con Spring Boot

El servicio REST se ha desarrollado con la herramienta Spring. Spring[10] es un framework de código abierto que proporciona herramientas y facilidades al programador de cara a la construcción de aplicaciones Java, en cualquier tipo de plataforma.

Es un framework muy completo, se divide en diferentes módulos, y cada uno de estos cubren diferentes funcionalidades. El núcleo de Spring realiza la inversión de control, más específicamente la inyección de dependencias. Estimula el uso de buenas prácticas, de manera que busca maximizar tanto la eficacia como la eficiencia de los proyectos desarrollados.

[7] https://github.com/goose3/goose3
[8] https://docs.botui.org/
[9] https://docs.botui.org/concepts.html
[10] https://spring.io/projects/spring-framework

Además ha permitido al autor centrarse exclusivamente en el desarrollo de la aplicación, olvidándose de configuraciones complejas.

El asistente está creado como una anotación SpringBootApplication, que incluye 3 principales anotaciones:

- *@Configuration* Se habilita la configuración basada en Java
- *@ComponentScan* Habilita el escaneo de componentes.
- *@EnableAutoConfiguration* Habilita la función de configuración automática propia de Spring Boot.

Trabajar con la herramienta Spring ha permitido ahorrar tiempo de programación y conseguir un código mucho más limpio, profesional y estructurado. Además, se han profundizado los conocimientos sobre esta herramienta adquiridos durante los estudios del grado.

Kaldi

Kaldi [11] es una librería de ASR. Kaldi es una herramienta para el reconocimiento de voz escrito en C++ y con licencia de Apache. Está destinado a ser utilizado para investigar sobre el reconocimiento de voz, aunque también es usado para numerosas aplicaciones comerciales. Ha sido la tecnología escogida debida que hoy en día es la herramienta de ASR más potente y además es de código abierto [19]. Permite crear un ASR propio, generando modelos con los datos de entrenamiento.

Cotovia

Cotovia [12] es un sistema TTS, ha sido utilizado debido a que actualmente es el único TTS de código abierto válido tanto para gallego como para castellano. Cotovia es una herramienta de conversión de texto a voz desarrollado por el Grupo de Tecnologías Multimedia de la Universidad de Vigo y el Centro Ramón Piñeiro para la Investigación en Humanidades. Se encuentra disponible tanto para la conversión en español como en gallego [20].

ElasticSearch

ElasticSearch es un servidor de búsqueda basado en Lucene. Su función es hacer búsquedas y análisis *RESTful* distribuido que almacena de manera centralizada sus datos para que pueda buscar, indexar y analizar datos de todas las formas y tamaños y es utilizada por un gran número de compañías. La creación de índices del proyecto como su posterior recorrido han sido mantenidos por Elastic[13].

[11] https://kaldi-asr.org/doc/about.html
[12] http://gtm.uvigo.es/cotovia
[13] https://www.elastic.co/products/elastic-stack

ElasticSearch puede ser usado para buscar cualquier tipo de documento, propociona una búsqueda escalable y con múltiples instancias, con una velocidad en tiempo real. Tal y como se ha comentado anteriormente, se basa en Lucene e intenta que todas sus funcionalidades se encuentren disponibles a través de la API de Java. El hecho de que siempre envía objetos JSON como respuestas, ha facilitado en gran medida su comprensión. A la hora de realizar las consultas, también se realizan mediante una interfaz JSON, por lo que han sido muy fáciles de formular y leer. Los tipos de consultas más utilizados en el proyecto han sido:

- *Match all query*. Devuelve todos los objetos indexados, la más simple.
- *Match query*. Se busca por un término de un campo en concreto, aunque también permite realizarla sobre distintos campos.
- *Range query*. Se busca por término de un atributo específico.

```
  {
"query": {
    "bool": {
        "must": [ ],
        "must_not": [ ],
        "should": [
           {
            "query_string": {
            "default_field": "subjectcodengrams",
            "query": "horario clases prácticas de herramientas de
    desarrollo"
           }
        }
        ,
        {
            "query_string": {
            "default_field": "subjectcodewords",
            "query": "horario clases prácticas de herramientas de
    desarrollo"
            }
        }
      ]
    }
},
"from": 0,
"size": 10,
"sort": [ ],
"aggs": { }
}
```

Listado 3.1: Ejemplo de una consulta del proyecto en formato JSON

La gran variedad de filtros y sus combinaciones permiten un abanico de consultas realmente amplio.

ElasticSearch Head

Para manejar los documentos índices y la elaboración de sentencias se ha usado el plugin *ElasticSearch Head*[14] disponible para Google Chrome.

Figura 3.2: Interfaz del plugin ElasticSearch Head

Se encuentra dentro del menú de extensiones de herramientas para desarrolladores, ofrece una interfaz muy intuitiva y sencilla de usar, ha sido utilizada durante toda la etapa de desarrollo del motor de búsqueda.

Tal y como se muestra en el menú de la Figura 3.3, cuenta con un gran número de funcionalidades. Esto ha permitido ver los índices creados, así como sus documentos indexados y el contenido de cada uno de sus campos. También contiene soporte para la creación de consultas con todo tipo de filtros, con sus respectivas respuestas encontradas y su puntuación en tiempo real, cuenta con un historial de las mismas. Ha sido realmente útil para la comprobación de la correcta indexación de los documentos, así como para elaborar consultas de prueba y analizar los resultados. Esto ha optimizado con creces el tiempo de depuración de errores, ya

[14] https://chrome.google.com/webstore/detail/elasticsearch-head/ffmkiejjmecolpfloofpjologoblkegm

que ha servido para darse cuenta al momento si el motor de búsqueda devolvía los resultados esperados.

Postman

Se trata de una herramienta enfocada a desarrolladores web. Su versión gratuita ha sido utilizada en el proyecto para enviar distintos tipos de peticiones HTTP al servidor de Elastic y al servicio Rest desarrollado.

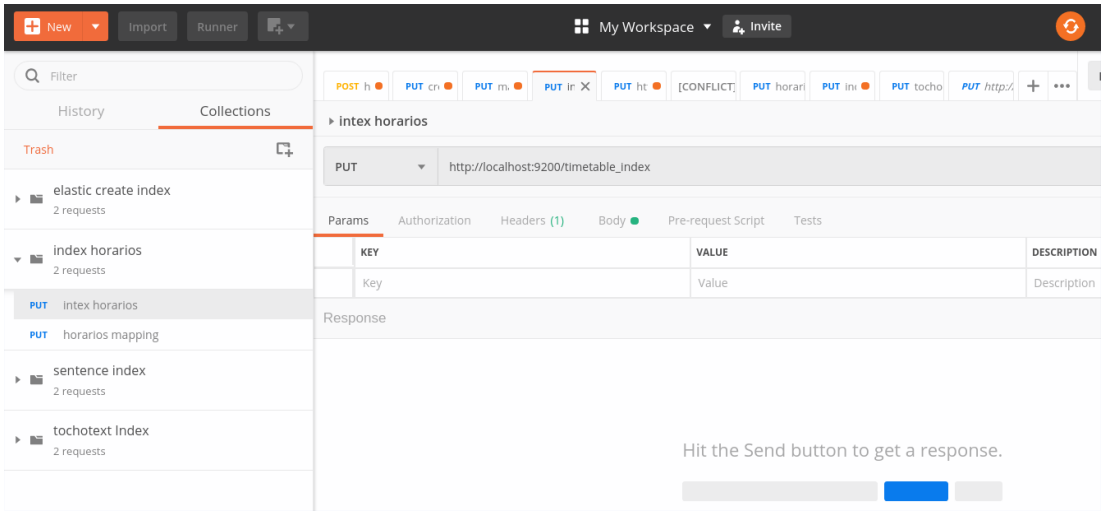

Figura 3.3: Menú de la herramienta Postman.

Con soporte para más de 100 millones de APIS y utilizado por más de 300.000 compañías, es una de las herramientas más utilizadas por los desarrolladores dentro del mundo API REST[15]. Nos ha permitido guardar y agrupar colecciones, de manera que se ha creado una por cada índice creado, tal y como se muestra en el menú de la izquierda en la Figura 3.4.

3.4 Herramientas de soporte

A continuación se hará una descripción de las distintas herramientas que han servido como soporte para el proyecto.

Ejecución del proyecto

En primer lugar, para la ejecución del proyecto se han utilizado los navegadores web Google Chrome[16] y Firefox[17], en sus versiones de escritorio. Ambos navegadores incluyen herra-

[15] https://www.getpostman.com/
[16] https://www.google.com/intl/es/chrome/
[17] https://www.mozilla.org/es-ES/firefox/features/

mientas de inspección y de navegación, que han permitido al autor revisar el código HTML y CSS, además de hacer uso de su herramienta de depuración para la resolución de errores.

Gestión de paquetes

Es conveniente el uso de una herramienta de gestión de paquetes que permita la inclusión de librerías necesarias de una manera sencilla buscando el ahorro de tiempos. Dado que hemos empleado librerías no incluídas en distribuciones estándar ha sido preciso la instalación de las mismas.

- *pip* Es un sistema de gestión de paquetes utilizado para instalar y administrar paquetes de software escritos en Python. Estos paquetes son encontrados en el *Python Package Index*[18], repositorio público de sofware abierto para su uso en Python[19], de los gestores más utilizados por los desarrolladores.

- *npm* Sistema de gestión de paquetes para JavaScript que facilitó la importación de librerías en el desarrollo del asistente[20].

- *virtualenv* Herramienta para la creación de entornos Python, con el objetivo de aislar librerías o entornos de ejecución. Contiene una documentación sencilla[21].

- *Maven* Apache Maven[22] es una herramienta de gestión y comprensión de proyectos software. Su principal objetivo es permitir al programador ver en cualquier momento el estado completo del esfuerzo de desarrollo. Utiliza un *Project Object Model* (POM) para describir el proyecto que se está construyendo, cargar las dependencias de otros módulos o proyectos externos, y el orden de su construcción. En este trabajo se han incluido dependencias de software externo como *ElasticSearch*.

  Maven busca mejorar la calidad del producto, acelerar el proceso de compilación y tener un historial de versiones para así reducir el número de incidencias. Debido a que ha sido la herramienta de gestión que el autor ha visto durante el curso académico, se ha optado por usar Maven en vez de otras herramientas como Gradle.

[18] https://pypi.org/
[19] https://pip.pypa.io/en/latest/news/
[20] https://www.npmjs.com/
[21] https://virtualenv.pypa.io/en/latest/
[22] https://maven.apache.org/

```
<!-- https://mvnrepository.com/artifact/com.github.axet/jssrc -->
  <!-- LIBRARY FOR RESAMPLING AUDIO -->
  <dependency>
    <groupId>com.github.axet</groupId>
    <artifactId>jssrc</artifactId>
    <version>1.0.2-2</version>
  </dependency>

  <!-- https://mvnrepository.com/artifact/commons-io/commons-io
  -->
  <!-- READING CONTEXT OF TXT FILE -->
  <dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
    <version>2.6</version>
  </dependency>

  <!-- ELASTIC SEARCH REPOSITORY -->
  <dependency>
    <groupId>org.springframework.data</groupId>
    <artifactId>spring-data-elasticsearch</artifactId>
    <version>3.1.9.RELEASE</version>
  </dependency>
```

Listado 3.2: Carga de algunas dependencias usadas en la aplicación

Eclipse

Eclipse[23] es la plataforma de software en la que se ha desarrollado la totalidad del código Java del proyecto. Es un Integrated Development Environment (IDE) multilenguaje con un entorno de trabajo con la opción de personalizarlo. Es el IDE más utilizado por los desarrolladores para lenguaje Java [21].

Cuenta con una gran cantidad de plugins con numerosas funcionalidades. A su vez, es muy completo en cuanto a *refactoring* y tiene disponible un gran número de atajos de teclado. Contiene soporte para integraciones de versiones, como la usada en el proyecto, Git. También incluye un sistema de depuración muy útil para seguir el recorrido del código e inspeccionarlo. Debido a su popularidad y a que es el IDE que el autor más ha manejado durante el curso académico, ha sido el elegido para llevar a cabo el proyecto.

[23]https://www.eclipse.org/

Visual Code

Visual Studio[24] es el nombre del entorno de desarrollo integrado utilizado para el desarrollo en Python, Javascript, HTML y CSS. En la actualidad es el IDE más popular para los programadores, con bastante diferencia respecto al resto [22]. Incluye compiladores, herramientas de finalización de código, diseñadores gráficos y muchas más características que facilitan el proceso de desarrollo software.

Al igual que en el caso de Eclipse, ha sido elegido para el proyecto debido a que era el que más sintonía tenía para el autor, puesto que había sido empleado en varias asignaturas durante el curso académico.

Gestión del proyecto: Taiga

Taiga[25] es una herramienta de software libre y código abierto, creada para gestionar y colaborar en proyectos ágiles, principalmente aquellos que utilizan metodología Scrum. Su principal funcionalidad es la gestión de tareas, pudiendo asignarlas a cada Sprint y así facilitar ampliamente su seguimiento, pudiendo asignar puntos de historia y de este modo analizar tanto los puntos completados como restantes para la finalización del mismo. Es muy simple e intuitivo y con la opción de extender sus funcionalidades gracias a sus módulos y poder integrarlas con otras herramientas como *GitLab* o *Github*. Debido a que cuenta con un tablero diseñado exlusivamente para Scrum junto a su facilidad de uso han sido los motivos para su uso como plataforma de gestión del proyecto.

Elaboración de prototipos : Balsamiq

Balsamiq[26] es una herramienta con una interfaz gráfica para la elaboración de *mockups*, lo que permite la construcción de la estructura de un sitio web. Un *mockup* es un modelo a escala de un diseño o de un dispositivo, utilizado para le demostración y evaluación de algún tipo de producto. Contiene una versión web y es la que se ha utilizado para elaborar los prototipos del proyecto. Se ha usado su versión gratuita de 30 días de duración. La elección de la herramienta ha estado influenciada por la experiencia que se tiene con ella al ser usada para diversas materias durante el curso.

Control de versiones: Git

A la hora de llevar a cabo el proceso software ha sido de vital importancia realizar un seguimiento de los cambios realizados durante el proyecto.

[24] https://visualstudio.microsoft.com/
[25] https://taiga.io/
[26] https://balsamiq.com/

Para organizar el registro de los cambios que han ido surgiendo se ha usado el sistema de control de versiones Git[27]. Pensando en la eficiencia y confiabilidad, es el software más usado a día de hoy para la gestión de la mayoría de proyectos nuevos. Su arquitectura está pensada para la creación de diferentes ramas de desarrollo, de manera que se puede trabajar en distintas funcionalidades del mismo proyecto, sin que los cambios interfieran unos con otros.

Git le da a cada programador una copia local del desarrollo en su totalidad, y los cambios se propagan entre los repositorios locales, por lo que podemos trabajar con *Git* sin ninguna necesidad de tener acceso a la red. Es por lo que para este proyecto se ha escogido Git frente a otros sistemas de control visto durante la carrera como SVN[28]. Además, está integrado en Eclipse, por lo que supuso una gran comodidad tener el historial dentro del mismo IDE.

A pesar de que Git ha sido construido para dar soporte a proyectos en grupo, y en este caso solo existía un desarrollador, ha supuesto una buena práctica y el autor ha aprendido a manejar mejor esta herramienta gracias a su correcto uso durante el proceso de desarrollo de la aplicación.

[27] https://git-scm.com/
[28] https://subversion.apache.org/

Capítulo 4

Metodología y planificación

Este apartado comprende las características principales sobre la metodología usada para desarrollar este proyecto y cómo se ha adaptado a las condiciones del mismo. Se comentarán las adaptaciones realizadas sobre la metodología con sus respectivas justificaciones. También se describirá cómo se ha elaborado la planificación del proyecto, incluyendo los costes y la gestión de riesgos.

4.1 Metodología de desarrollo

Para la construcción del proyecto se ha decidido seguir la metodología ágil Scrum. Para este tipo de desarrollo se ha considerado que seguir una metodología incremental sería lo que mejor se adaptaba. Se sigue una línea de trabajo dividida en iteraciones. De esta manera hemos podido asegurar que al final de cada período tengamos un producto operativo y entregable. Cada iteración representa una fase típica en cualquier desarrollo de proyecto software (análisis, diseño, implementación y pruebas).

A lo largo del trabajo ha sido necesario modificar y refinar los objetivos ante diferentes imprevistos que han ido surgiendo, es por eso que utilizar una metodología ágil ha permitido detectar problemas de manera rápida y eficaz [23]. A su vez, al ser un marco adaptativo nos ha posibilitado tomar decisiones en base a experiencias pasadas surgidas en el propio proyecto.

Scrum [24] es un marco de trabajo ligero y muy sencillo de entender. Situado dentro de las metodologías ágiles, nos ha permitido obtener grandes resultados en un tiempo mínimo. Nos ha garantizado entregar productos de calidad al final de cada iteración, reduciendo el tiempo dedicado a errores y gestión del mismo. Los tres pilares sobre los que se asienta Scrum son la transparencia, adaptación e inspección. Estos concetos se han intentando mantener en todas las etapas del proyecto.

Ya que en este proyecto no han intervenido el número de personas ideales según las directrices de Scrum (de 5 a 9 personas), en algunos elementos se ha tenido que adaptar la metodo-

logía. A continuación se describirán los principales elementos de Scrum con sus respectivas modificaciones para este proyecto.

4.1.1 Roles

Scrum está formado por un grupo auto-organizado en el cual todas las tareas pueden ser realizadas por las personas implicadas en el proyecto, los tres roles principales a tener en cuenta han sido:

- **Product Owner**
  Es el propietario del producto, representa a los clientes y sus necesidades. Es el encargado de comunicarse con el equipo. Sus decisiones deben ser respetadas por el equipo y es el encargado de organizar y priorizar el contenido del Product Backlog. En este proyecto el rol de Product Owner fue desarrollado por los directores Javier Parapar y Paula López, ayudando en la obtención de requisitos y priorizando las tareas de la pila del producto.

- **Scrum Master**
  Es el responsable de hacer entendible el trabajo a realizar y cómo ejecutarlo. El resto del equipo ha de ceñirse a las prácticas establecidas y seguir las directrices marcadas por el Scrum Master, su objetivo es aumentar lo máximo posible la productividad de todo el equipo. Debe promover que se sigan las directrices de Scrum en todas las fases del desarrollo. No debe confundirse con director del proyecto, puesto que en Scrum este concepto carece de valor. En el proyecto la figura de Scrum Master fue desarrollada por el propio estudiante, ya que ha sido el encargado de incrementar la efectividad del trabajo y plasmar soluciones a los distintos problemas que iban surgiendo.

- **Equipo de desarrollo**
  Grupo de trabajo encargado de producir los productos entregables a partir de los requisitos establecidos por el Product Owner. No disponen de roles específicos como en otras metodologías, se asegura de gozar de todas las habilidades necesarias para realizar los incrementos en el producto. El equipo de desarrollo está formado únicamente por el autor, puesto que es el único que ha participado en la implementación.

  Además de los mencionados, en Scrum también se contemplan diversos roles como los Stakeholders, managers o los destinatarios finales que usarán el producto.

4.1.2 Eventos

Scrum identifica diferentes eventos a la hora de gestionar un proyecto, están caracterizados por tener un bloque de tiempo predefinidos con una duración máxima, su labor principal es la de ayudar a la inspección y adaptación del proyecto, en las que se busca hablar de los temas más relevantes sin dar lugar a discusiones banalesen busca de una mejora de los tiempos de entrega.

- **Sprint**
  El evento principal de Scrum, que contiene al resto. Es un time-boxing de menos de un mes de duración máxima en el que se debe realizar un incremento del producto. En este caso se ha decidido dividir en seis Sprints con una duración de cuatro semanas cada uno, seis meses en total en los que se han llevado a cabo las distintas iteraciones. Al finalizar cada Sprint se comienza el siguiente de inmediato. Es muy recomendable no añadir nuevas tareas al Sprint a no ser que estas sean totalmente indispensables para la realización del mismo, así como llevar una duración constante en todos los Sprints.

- **Sprint planning**
  Es la reunión previa a cada Sprint, es una reunión en la que se planifica el trabajo a realizar durante su tiempo de duración, es decir, se marcan las historias de usuario que se llevarán a cabo, con su respectiva estimación de complejidad y el resultado esperado a su cierre.

- **Revisión del Sprint**
  Al contrario que el Sprint planning, se realiza al término de cada Sprint. Como su nombre indica es la reunión encargada de revisar e inspeccionar el producto realizado durante el Sprint. Se ha intentado detectar los fallos y tener en cuenta este aprendizaje para el futuro Sprint.

  Con el objetivo de optimizar el tiempo, tanto el *Sprint planning* como la *Revisión del Sprint* han ido seguidas. Es decir, en cuanto se acababa un Sprint se tenía una reunión en la que se inspeccionaba y se discutían con los Product Owner los aspectos positivos y negativos del mismo. Acto seguido, se enlazaba la reunión con la planificación del siguiente Sprint.

- **Scrum diario**
  Reunión diaria en la que el grupo de desarrollo pone en común el trabajo realizado en el último día, su planificación para el día actual y si ha aparecido algún obstáculo durante el proceso. Debido a que el grupo de desarrollo está formado únicamente por el autor, estas reuniones han sido obviadas en este proyecto.

- **Retrospectiva del Sprint**
  Reunión posterior a la revisión del Sprint y previa a la planificación del siguiente Sprint. En ella se valora el trabajo de todo el equipo que conforma el proyecto y se hace una inspección sobre las mejoras a introducir en el siguiente, localizar las debilidades y los puntos fuertes para mejorar en la siguiente tarea basándose en la experiencia previa. Es por esto este proyecto está dentro de un marco de trabajo adaptativo.

4.1.3 Artefactos

Son aquellos elementos que representan trabajo o valor para así propocionar oportunidades de inspección y adaptación. Diseñados para maximizar el entendimiento de información clave y que todo el grupo de trabajo tenga la misma percepción.

- **Pila del Producto**
  Es una lista con prioridades formada por los requisitos necesarios para el desarrollo del producto, suele estar representada por las historias de usuario. Esta lista enumera todas las funcionalidades y requisitos que deben ser empleados sobre el producto en interacciones futuras. Debido a la característica adaptativa del proyecto, está en continuo cambio y han aparecido requisitos nuevos en diversos momentos. El Product Owner es el encargado de llevar su disponibilidad, orden y contenido; a medida que avanza el proyecto la pila se hace más extensa y contendrá mayor carga de trabajo. En esta lista se han ido mencionando los diferentes tareas durante los 6 Sprints que contiene el proyecto.

- **Pila del Sprint**
  Es el subconjunto de elementos de la Pila del Producto que han sido seleccionados para el desarrollo de un determinado Sprint. Al acabar este subconjunto se obtiene como resultado una nueva interacción. Se ha intentado que el total de puntos de historia de cada Pila de Sprint sea constante.

- **Incremento**
  El incremento después de un Sprint, contiene todas las funcionalidades del producto de ese Sprint junto con la de los anteriores.

- **Evolución gráfica del proyecto**
  Es habitual incluir una gráfica que pueda hacer visible el progreso seguido por el proyecto. Muestra los requisitos de la Pila que se estiman realizar al comienzo de cada Sprint y dibuja una línea que conecta los puntos de los Sprints completados.

4.2 Planificación y seguimiento

En esta sección se comentan las estimaciones del tiempo y costes realizadas al principio del proyecto, además de enumerar los recursos utilizados para llevarlas a cabo.

4.2.1 Recursos

Se pueden dividir en tres los tipos de recursos empleados para un proyecto con estas características.

Recursos humanos

Tal y como se ha explicado en la Sección 4.1.1, se ha tenido la ayuda de un equipo de tres personas para la elaboración de este proyecto. Disponemos de un grupo de trabajo de tamaño reducido. Los directores Javier Parapar y Paula López han ejercido la función de *Product Owner*, colaborando con la planificación y definición de requisitos. El resto de roles serán ejercidos por el autor, tanto el de *Scrum Master*, asegurando que se cumpla el trabajo de acuerdo a las directrices de la metodología, como el de *Equipo de desarrollo*, ejerciendo de analista, diseñador y desarrollador de todo el trabajo planificado.

Recursos materiales

Como recursos materiales se incluye el ordenador personal del autor, configurado con las herramientas necesarias. Además, se ha hecho uso de un servidor GPU para entrenar los modelos de lenguaje.

Recursos software

Todos los mencionados en el capítulo de tecnologías 3. Todas las herramientas han contado con licencia de estudiante o han sido proporcionadas por los directores.

4.2.2 Gestión y planificación del proyecto

En las etapas iniciales del proyecto se ha llevado a cabo un análisis del proyecto en la cual se decidieron los siguientes Sprints:

- Sprint 1: Elaboración del proceso de Crawling para el dominio de la facultad.
- Sprint 2: Construcción del servicio de transmisión de audio.
- Sprint 3: Servicios de gestión de audio.
- Sprint 4: Indexación con Elastic.

- Sprint 5: Elaboración de las búsquedas con Elastic.
- Sprint 6: Desarrollo de la interfaz del asistente.

En la planificación, se ha establecido una duración de Sprints de un mes, en busca de un equilibrio entre la cantidad y calidad del trabajo. Se había considerado reducir la duración de los Sprints a tres semanas y aumentar el número de estos, pero creímos que se adaptaría algo peor a las condiciones del proyecto, ya que en ciertos Sprints no habría suficiente trabajo ejecutado para evaluar.

Las estimaciones resultaron en 15 horas semanales de desarrollo, lo que resulta en un total 60 horas por Sprint dedicadas al desarrollo. Los Sprints tienen un ajuste inicial de 45 puntos historia, por lo que cada punto historia representa 1 hora y 20 minutos de trabajo.

Siempre se ha intentando mantener la constancia de duración de los Sprints, priorizando la optimización de recursos y cumplir según el plan acordado. Sin embargo, algunos Sprints han sufrido algunas variaciones de esfuerzo a lo planificado al comienzo del mismo. La principal razón ha sido por las circunstancia personales y académicas del autor, además de la aparición de complicaciones durante el desarrollo que no se habían planificado en las etapas iniciales. Estas desviaciones se han intentado cubrir de la mejor manera posible para intentar no desequilibrar el ritmo de trabajo.

4.2.3 Costes

En esta sección se encuentran los costes estimados al desarrollo del sistema:

Equipo	Director	Analista	Desarrollador
Miguel Anxo Pérez	-	26€	20.8€
Paula López	45.5€	-	-
Javier Parapar	45.5€	-	-

Tabla 4.1: Costes por hora para los recursos humanos del proyecto estimados.

Hardware	Vida útil	Uso	Total
Servidor GPU	4 años	1 mes	93.75€

Tabla 4.2: Costes por hora para los recursos materiales del proyecto estimados.

El coste de los salarios ha sido basado en el informe *Estudio salarial - Sector TIC en Galicia 2015-2016* [25]. En la base del cálculo de salario de los recursos humanos viene incluído los costes de Seguridad Social. Asimismo, el precio del servidor GPU es de 4.500€.

Para entender cómo se ha contabilizado el proyecto se deben tener en cuenta las siguientes características:

- Los *Product Owner* han dedicado 4 horas al proyecto en cada Sprint
- El equipo de desarrollo ha dedicado 15 horas semanales al desarrollo, lo que resulta un total de 360 horas totales de desarrollo.

Los datos anteriores contabilizan los siguientes costes:

Rol	Tiempo(h/Sprint)	Nº Sprints	Coste(€h)	Total(€)
Desarrollador	60	6	20.8€	7488€
Analista	5	6	26€	780€
Director	8 (Ambos)	6	45€	2184€
Servidor GPU	-	-	-	93.75€
Total	-	-	-	10545.75€

Tabla 4.3: Desglose total de los costes.

4.2.4 Gestión de riesgos

En la planificación del proyecto, tiene gran relevancia hacer una correcta identificación de los riesgos a los que está expuesto. Una identificación oportuna permitirá evitar posibles problemas para el proyecto, y minimizar sus consecuencias, en caso de que estos ocurran. A continuación se mencionan los principales riesgos presentes en el desarrollo del proyecto:

- Falta de conocimiento y no llegar a cumplir todos los requisitos no funcionales establecidos., la mayor parte de los componentes y librerías nunca habían usado con anterioridad por el estudiante. Además, nunca se había desarrollado en lenguaje Python. Debido a la falta de experiencia del estudiante a la hora de implementar este tipo de sistemas, resulta probable que la implementación inicial carezca de eficiencia. Para paliar esto, se han analizado de manera exhaustiva los resultados obtenidos, incluyendo pruebas de validación y usabilidad ante usuarios inexpertos una vez el trabajo ha estado terminado.
- Abordar un trabajo fin de grado multidisciplinar con un equipo de desarrollo unipersonal, puede verse plasmado en una incorrecta planificación o un diseño insuficiente. Para hacer frente a estos riesgos se ha contado con la labor de los directores, con amplios conocimientos tanto en el campo de la recuperación de la información para las búsquedas como en el procesamiento de señales para la gestión del audio.

- Integración de un gran número de tecnologías que intervienen conjuntamente en la aplicación del asistente, que suma complejidad a la implementación. Debido al uso de una metodología ágil e incremental, se ha podido realizar entregas parciales y regulares del producto, por lo que se ha podido tener en cuenta este riesgo en todo momento.

Capítulo 5

Análisis

El análisis de requisitos se ha llevado a cabo en las etapas iniciales del proyecto. Es una de las fases más importantes dentro del desarrollo software. Se han discutido las funcionalidades necesarias que debe cumplir nuestra aplicación. Precede a la etapa de diseño y durante este tiempo se ha realizado un análisis en profundidad de la arquitectura del sistema.

5.1 Análisis de requisitos

Para poder construir un asistente que resuelva satisfactoriamente las preguntas de los usuarios, se establecieron una serie de requisitos que la aplicación debe cumplir.

La primera tarea que se ha realizado ha sido mantener una reunión con los Product Owner para definir los requisitos funcionales y no funcionales con un nivel alto de abstracción. Para plasmar esto de una manera más transparente, ha servido de ayuda utilizar el concepto de actores y la elaboración de prototipos. Aunque en este proyecto concreto la interfaz es bastante sencilla, siempre es buena práctica en cualquier desarrollo software para tener facilidades a la hora de captar requisitos en fases tempranas del mismo, y tener una visión global del proyecto desde los inicios.

Posteriormente se han ido puliendo estos requisitos y se han descrito con mucho más detalle utilizando historias de usuario dentro de cada Sprint.

5.1.1 Actores

Solo se presenta un único actor, aquel que accede a la página web de la FIC. Tiene permisos para acceder a todas las funcionalidades del asistente, no existe ningún tipo de restricción especial ni usuarios con más ventajas que otros.

5.1.2 Requisitos funcionales

Los requisitos funcionales comprenden las características que debe tener el sistema. Tal y como hemos mencionado anteriormente, al principio se plasmaron en alto nivel después de una primera reunión, y después vienen recogidos como historias de usuario. Los requisitos funcionales establecidos al comienzo fueron los siguientes:

Construcción de un almacén de datos El asistente debe contar con una capa de datos donde buscar la información. Esta base de datos tendrá una organización estructurada. De estos datos vendrá la información enviada en las respuestas por parte del asistente, por lo que su búsqueda debe ser rápida y efectiva.

Elaboración de un proceso *question answering* efectivo El asistente debe contar con una lógica que permita redirigir la búsqueda dependiendo de cual sea el tipo de pregunta, bien información, una página web o localizaciones, entre otros.

Respuestas concisas y efectivas El asistente debe proporcionar respuestas breves, el objetivo es contestar las preguntas sin añadir información irrelevante de por medio.

Gestión de los errores El asistente debe estar implementado de manera que sepa reaccionar cuando no ha entendido la pregunta o no haya conseguido encontrar la información.

Respuestas coherentes e idénticas ante entradas de audio o texto Los usuarios tienen la posibilidad de escuchar la respuesta mediante voz, esta ha de ser entendida e idéntica a la respuesta por escrito.

5.1.3 Requisitos no funcionales

También fueron establecidos en la etapa inicial del proyecto. Se han establecido diversas características que el asistente debe abarcar. El concepto visual, y la efectividad de las respuestas ha sido lo que más se ha tenido en cuenta a la hora de describir los requisitos no funcionales. Esto ha condicionado la decisión de utilizar ciertas herramientas y tecnologías por encima de otras. Los requisitos no funcionales recogidos han sido los siguientes:

Velocidad en las respuestas El principal servicio que debe proporcionar el asistente a los estudiantes es ahorrarles tiempo, que sea más comodo preguntarle que tener que buscar por los enlaces de la web. Es de vital importancia que la elaboración de la respuesta sea en tiempo real.

Facilidad de uso Las preguntas se deben poder hacer de un modo veloz e intuitivo, tanto por teclado como por voz.

Interfaz atractiva La interfaz debe integrarse de manera atractiva con la página web.

5.1.4 Prototipos

Después de definir los requisitos de alto nivel, se decidió crear una serie de bocetos, sin mucho nivel de detalle, de como sería la interfaz de la aplicación. El objetivo de estas ilustraciones es plasmar la idea de que fuese una interfaz simple e intuitiva.

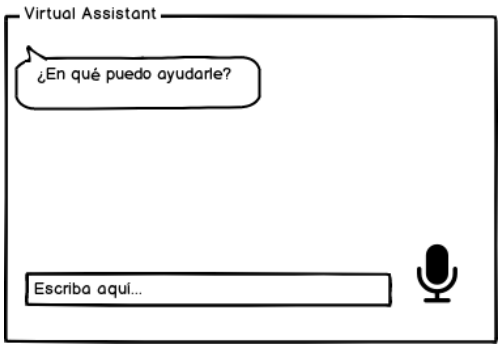

Figura 5.1: Maqueta de la interfaz del asistente

La página de la facultad es bastante dinámica y contiene muchas secciones diferentes, por lo que se ha estipulado como primordial que la adaptación del asistente sea visualmente agradable.

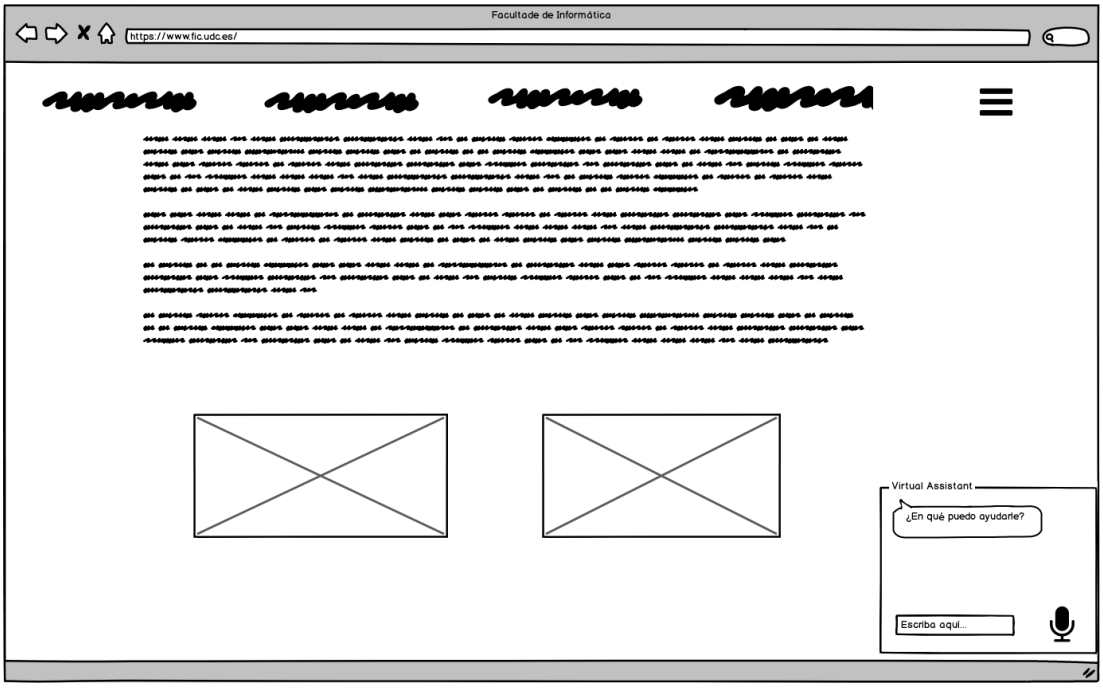

Figura 5.2: Maqueta de la página web de la FIC integrada con el asistente

5.1.5 Historias de Usuario y Tareas

En el momento de comenzar a desarrollar la aplicación, se tomaron como base los requisitos funcionales de alto nivel para crear las principales tareas a realizar. Esta transformación fue lo que ha generado las historias de usuario, que es de lo que está compuesto la pila del producto. A continuación se muestran los Sprints y sus respectivas historias:

T1 Elaboración del proceso de crawling para el dominio de la facultad.

T1.1 Configuración del entorno.
T1.2 Estudio y formación sobre las técnicas de crawling.
T1.3 Elaboración del crawling de *https://www.fic.udc.es/*.
T1.4 Depuración del Spider que ejecuta el crawling.

T2 Desarrollo del servicio de transmisión de audio.

T2.1 Construcción del servicio *REST* para la recepción, envío y tratamiento de los datos.
T2.2 Desarrollo de métodos para la captura y el envío de la información de audio en la interfaz.
T2.3 Implementación de una interfaz útil para la recepción de audio. (Mediante JS y HTML)
T2.4 Gestión de los archivos que se generan y la organización de su almacenamiento.

T3 Servicios de gestión y transcripción de audio.

T3.1 Estudio y formación de las distintas alternativas para conseguir la transcripción del audio.
T3.2 Clase *FileUtils* para gestionar los archivos de preguntas y respuestas.
T3.3 Entrenamiento del modelo Kaldi tanto para gallego como para español.
T3.4 Funcionalidad de poder escuchar y tener el texto de las respuestas en tiempo real.
T3.5 Creación de servidor *docker* para Kaldi.
T3.6 Uso de Cotovia.

T4 Indexación de documentos con Elastic.

T4.1 Aprendizaje de la lógica Elastic y funcionamiento de los índices de documentos.
T4.2 Construcción de los índices, diviendo los campos en texto y n-gramas. Elaboración del mapeado sobre los campos de estos mismos índices.

T4.3 Recorrido sobre la información del dominio de la FIC para almacenarla en diferentes campos usando los índices ya creados.
T4.4 Discusión y realización de un tipo horario para poder permitir a los usuarios acceder a información sobre las clases, como sus horas, aulas, entre otros.

T5 Realización de búsquedas con ElasticSearch.

T5.1 Desarrollo de las entidades sobre las que se va a realizar la inspección. Generar los campos en cada una de ellas para mapearlos con los índices de Elastic. Construcción y generación de las repositorios para desarrollar el método de búsqueda tanto por texto como por n-gramas.
T5.2 Gestión de las búsquedas. Estudio de las diferentes posibilidades de respuestas y diseño de la lógica sobre el orden de búsqueda sobre los índices. Tratamiento de los errores.
T5.3 Modificar la configuración del campo Score del API de Elastic.

T6 Diseño de la interfaz BOTUI.

T6.1 Estudio de las diferentes alternativas para la generación de la interfaz que se va anclar a la página como asistente virtual.
T6.2 Elaboración de la interfaz de una manera simple e intuitiva.
T6.3 Poder ver las respuestas a mis cuestiones por pantalla.
T6.4 Poder escuchar las respuestas en tiempo real.
T6.5 Integración de la aplicación en la página de la facultad.

Las tareas se han organizado en seis Sprints, como se ve reflejado en la tabla 5.1. Su duración total es desde el 4 de Febrero hasta el 4 de Agosto. A la hora de plasmar las tareas como historias de usuario en *Scrum*, se han descrito las siguientes:

ID	Historia de usuario	Puntos
1	Como usuario quiero tener disponible la información del dominio web de la facultad de manera offline 20	
2	Como usuario quiero ser capaz de recibir respuestas provenientes de cualquier espacio de la página de la FIC	25
3	Como usuario quiero poder ser capaz de enviarle mis preguntas al asistente	15
4	Como usario quiero poder escuchar mi propia pregunta reproducida en el navegador	20

5	Como usuario quiero ver mi propia pregunta en el interfaz cuando escribo en formato texto	7
6	Como usuario quiero poder escuchar mi propia pregunta hecha por audio con voz artificial reproducida en el navegador	20
7	Como usuario quiero poder ver por pantalla mi pregunta por voz en formato texto	18
8	Como usuario quiero poder escuchar mi propia pregunta hecha por texto con voz artificial reproducida en el navegador	3
9	Como usuario quiero poder acceder a los documentos indexados en mi consulta	25
10	Como usuario quiero poder acceder a los distintos tipos de documentos diferenciados por tipo de pregunta	14
11	Como usuario quiero poder obtener respuestas satisfactorias	20
12	Como usuario quiero poder obtener distintos tipos de respuestas dependiendo de mi pregunta	12
13	Como usuario quiero poder reformular la pregunta en caso de que el asistente no me haya entendido	5
14	Como usuario quiero una interfaz agradable y facil de manejar	28
15	Como usuario quiero poder tener un asistente adaptable mientras navego por la web	7

Tabla 5.1: Historias de usuario y su estimación.

Sprints	**Tareas**	**Historias**
1	T1.1, T1.2, T1.3, T1.4	1, 2
2	T2.1, T2.2, T2.3, T2.4	3, 4, 5
3	T3.1, T3.2, T3.3, T3.4, T3.5, T3.6	6, 7, 8
4	T4.1, T4.2, T4.3, T4.4	9, 10
5	T5.1, T5.2, T5.3	11, 12, 13
6	T6.1, T6.2, T6.3, T6.4, T6.5	14, 15

Tabla 5.2: Organización de las tareasy su relación con los sprints e historias.

5.2 Arquitectura

En esta sección se tratará la arquitectura que ha seguido el sistema. Se propone una arquitectura básica cliente-servidor compuesta por los principales componentes:

- *Interfaz cliente del asistente*
- *Servidor REST del sistema*

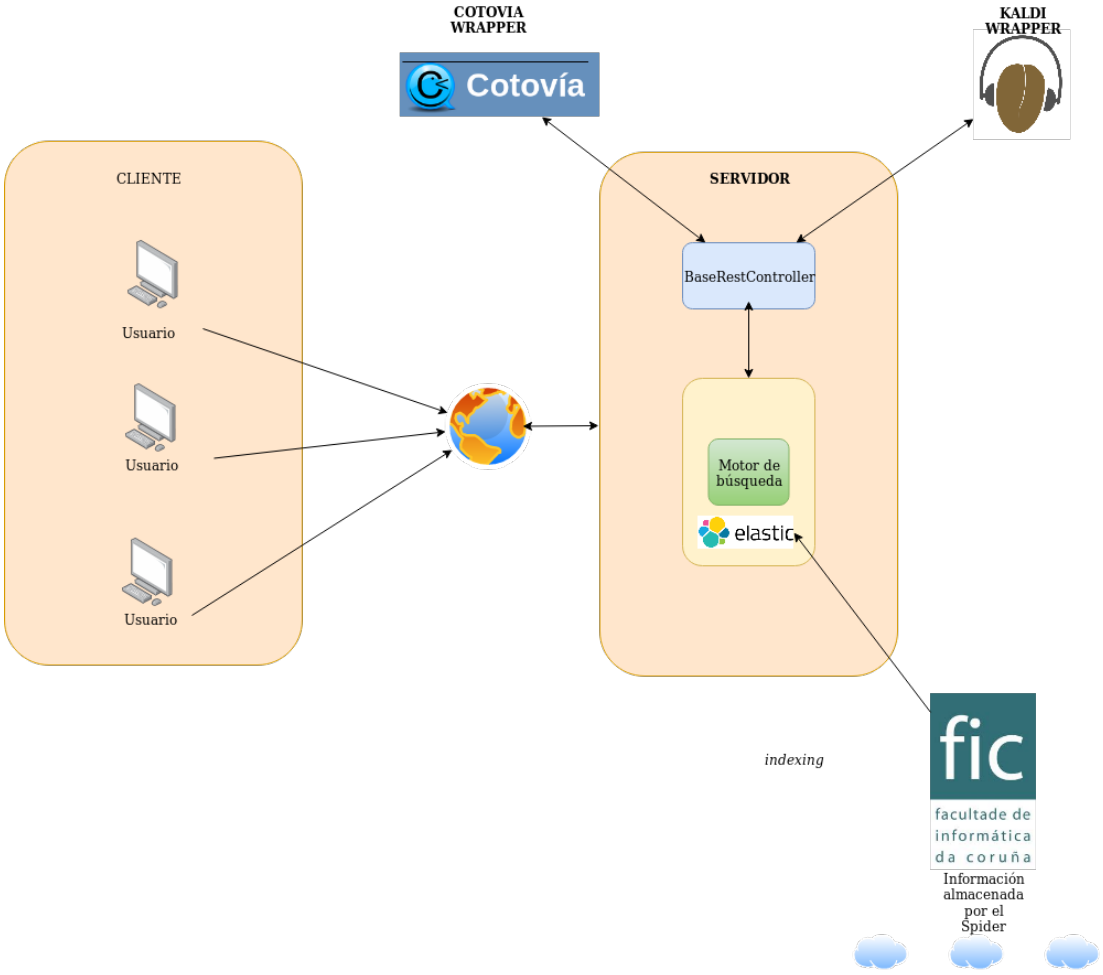

Figura 5.3: Arquitectura general de alto nivel del proyecto

5.2.1 Cliente

El cliente está formado por la interfaz gráfica que compone la parte visual del asistente virtual. Es el encargado de responder ante las distintas interacciones del cliente, enviando al servidor las consultas de los usuarios. Para esta comunicación con la parte *back-end* se ha hecho uso de las tecnologías mencionada en la sección 3.1. Además, para su diseño, se ha hecho uso de la API *BotUI*, mencionada en la misma sección.

5.2.2 Servidor

En la parte del servidor, la tecnología escogida para su desarrollo ha sido Spring Boot, aprovechando la gran cantidad de funcionalidades que proporciona junto con el lenguaje Java. Contiene la implementación de una interfaz *REST* necesaria para la gestión y tratamiento de los distintos archivos de audio y texto, con las siguientes operaciones (en este caso, peticiones GET Y POST):

- **POST *question/*** Construye la comunicación de la pregunta en audio para su almacenamiento en el servidor.
- **POST *questionkeyboard/*** Construye la comunicación de la pregunta cuando el usuario la introduce mediante el teclado para su almacenamiento en el servidor.
- **GET *response/sessionId/*** Maneja la respuesta encontrada en audio y la envía al cliente.
- **GET *response_text/sessionId/*** Maneja la respuesta encontrada en texto y le envía al cliente.
- **GET *question_text/sessionId/*** Maneja la pregunta hecha por el usuario y la muestra por escrito en la conversación del asistente.
- **GET *response_keyboard/sessionId/*** Cuando la entrada es por teclado, maneja la respuesta encontrada vía archivo de audio y la envía al cliente.
- **GET *response_keyboardtext/sessionId/*** Cuando la entrada es por teclado, maneja la respuesta encontrada en texto y le envía al cliente.

Todas las operaciones anteriores hacen uso de las herramientas Kaldi y Cotovia, tanto para tratar las preguntas como para enviar de vuelta las respuestas ya obtenidas. Una vez está disponible la pregunta en formato texto, el servidor comienza el proceso correspondiente a la búsqueda de la información.

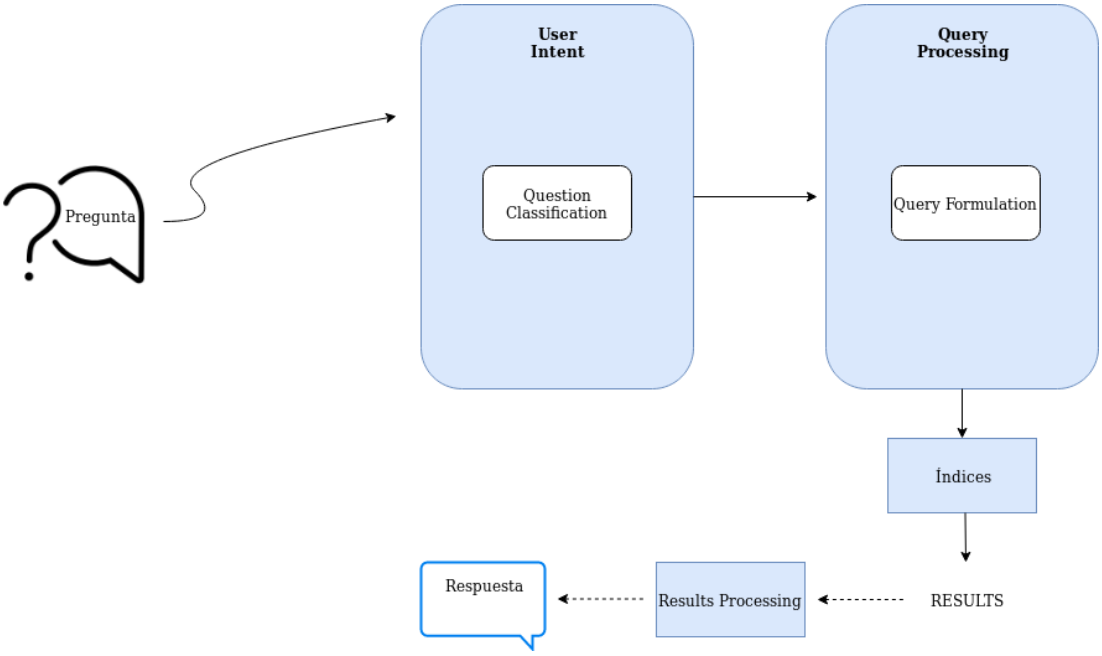

Figura 5.4: Diagrama de flujo de alto nivel del proceso de búsqueda

En el diagrama de la figura 5.4 se puede ver el recorrido que utiliza el sistema conversacional para generar las respuestas.

El sistema se ha implementado de acuerdo a tres tipos de tareas básicas, siendo facilmente extendible:

- **Búsqueda web general** Las cuales contienen información básica sobre una web, junto con un enlace a la misma.
- **Búsqueda de respuestas** Respuestas cortas las cuales se consideran suficientes para satisfacer la pregunta.
- **Tareas ad-hoc** Tareas específicas correspondientes al dominio web de una página de universidad. Calendario de exámenes, horarios, aulas; entre otras.

Cada una de estas tareas representará una entidad en el desarrollo. Estas entidades contienen sus respectivos repositorios para la formulación de las consultas con los campos requeridos para cada una.

Cuando el sistema identifica la pregunta, esta llega a la clase *User Intentions*, que representaría el *Question Processing*. Esta clase es la encargada de clasificarla y así obtener el tipo de tarea que se va a abordar. En cuanto se conoce la entidad sobre la cual se va a realizar la búsqueda, el sistema elabora la consulta. La consulta es ejecutada sobre los índices de documentos de la página de la Facultad, almacenados previamente. Los distintos resultados obtenidos por

las consultas devolverán una puntuación. A la hora de escoger la respuesta candidata, la lógica del sistema incluye una inspección en el caso de que las puntuaciones no sean lo suciente-mente relevantes, por lo que serán rechazadas. En el caso de que esto ocurra, el *Question Processing* cambiará a un tipo distinto de pregunta, y se volverá a lanzar la consulta con la entidad elegida. Una vez se tenga una puntuación satisfactoria, se selecciona la respuesta y será la mostrada a los usuarios por el asistente.

Capítulo 6

Desarrollo

Este capítulo contiene la descripción del desarrollo que se ha llevado a cabo para construir la aplicación. Se detallarán las tareas mencionadas anteriormente, explicando con mayor profundidad lo que se ha realizado en las historias de usuario.

La estimación total de las tareas que pertenecen a la Pila del Producto ha sido de 270 puntos de historia. El desarrollo está dividido en seis Sprints, por lo que hemos intentado que cada Sprint abarcara 45 puntos de historia.

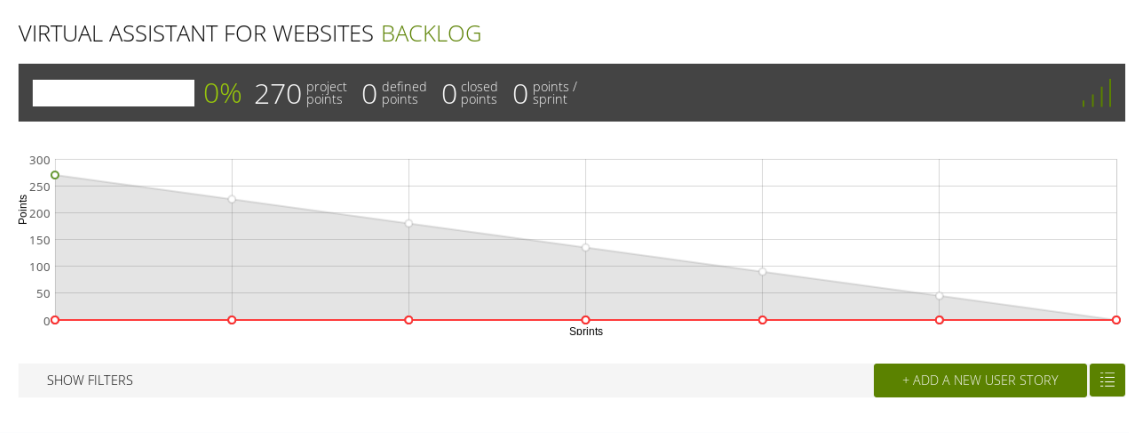

Figura 6.1: Planificación del esfuerzo inicial

En la figura 6.1 se pueden ver las siguientes referencias :

- *Project points* : Puntos totales asignados al proyecto
- *Defined points* : Puntos asignados a las historias del desarrollador
- *Closed points* : Aquellos puntos que se se han completado

- *Points/sprint* : Corresponde a la media de puntos que se han logrado en cada Sprint. En este proyecto hemos intentado que la media se mantuviese en los 45 puntos, tal como se ha estipulado al inicio del proyecto.

6.0.1 Sprint 1 : Elaboración del proceso de crawling para el dominio de la facultad

Después de discutir la captación de requirimientos y estudiar las funcionalidades que permitiría el asistente, dio comienzo el desarrollo. El primer paso de este primer sprint ha sido la configuración del entorno:

- Construcción del proyecto en Eclipse, siguiendo un arquetipo básico.
- Integración de la herramienta de control de versiones Git en el flujo de desarrollo.
- Integración de la herramientas de gestión del proyecto Taiga.

Posteriormente , como el autor no tenía conocimientos previos sobre *Crawling*, se ha dedicado un tiempo al aprendizaje de la tecnología necesaria para desarrollar este trabajo. Además, se estudiaron las diferentes opciones para realizar el crawling, con la decisión de usar la librería Scrapy, disponible para Python.

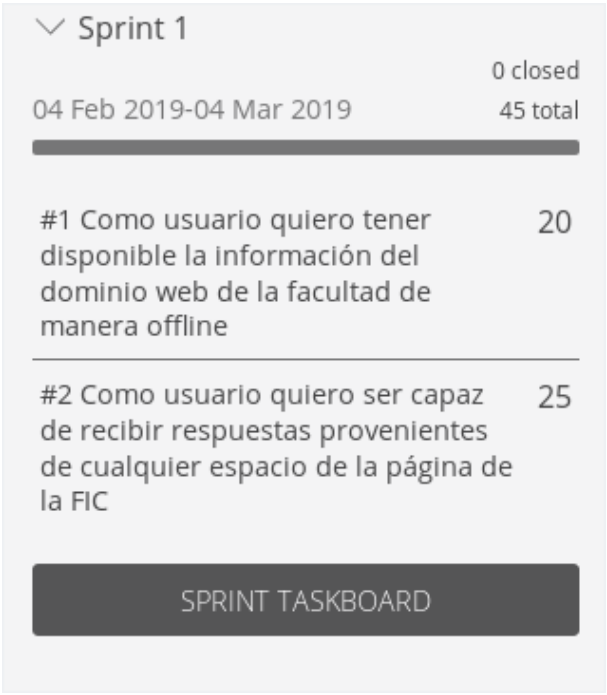

Figura 6.2: Historias de usuario y duración estimadas del primer sprint

En cuanto al desarrollo, este Sprint comprende la creación del Spider que generará los archivos con la información de las páginas del dominio de la facultad. Se ha permitido que recorra todos los dominios que sean "fic.udc.es", de manera que cuando haya un enlace a otro dominio diferente, no lo analice, como pueden ser enlaces a Twitter, Youtube o incluso otras universidades. Para esto se ha hecho uso de diferentes librerías para Python 3.2, siendo la principal Scrapy.

```
1 class FicPage(scrapy.Item):
2     url = scrapy.Field()
3     title = scrapy.Field()
4     texto = scrapy.Field()
5     bigtext = scrapy.Field()
```

Listado 6.1: Campos que componen cada instancia de la clase Ficpage

Al finalizar el análisis, se ha obtenido un archivo *webFicItems* en formato Json Lines [1], en el que cada línea representa un valor JSON. Cada una de estas líneas recoge la información de cada página del dominio, con la información optimizada en cada uno de sus campos. El proceso de Crawling ha durado más de 24 horas, con un tamaño final de 11.221 líneas.

Una vez inspeccionado el resultado, se comprobó que el campo texto obtenía demasiada información irrelevante. A veces se podía llegar a encontrar duplicada, o era la misma información pero cambiando entre español y gallego. Debido a esto, la solución planteada ha sido dividir el campo texto en dos campos diferentes. El valor texto obtendrá la información que Scrapy considere relevante, y el campo bigtext simplemente obtendrá toda la información del texto de la página.

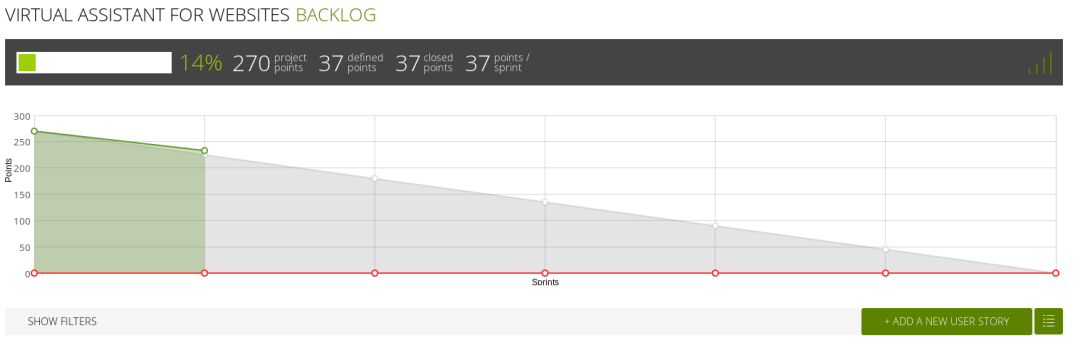

Figura 6.3: Progreso del proyecto finalizado el primer sprint

Como se puede apreciar en el progreso de la anterior figura, hubo un error de estimación y ha sobreestimado el primer Sprint. Este motivo es debido al desconocimiento inicial

[1] http://jsonlines.org/

sobre tecnologías de *Crawler*. Su desarrollo ha resultado más afable de lo previsto, por lo que el tiempo y esfuerzo ha sido algo inferior a lo planeado para este Sprint. En lugar de los 45 estimados se ha acabado completando 37 puntos de historia.

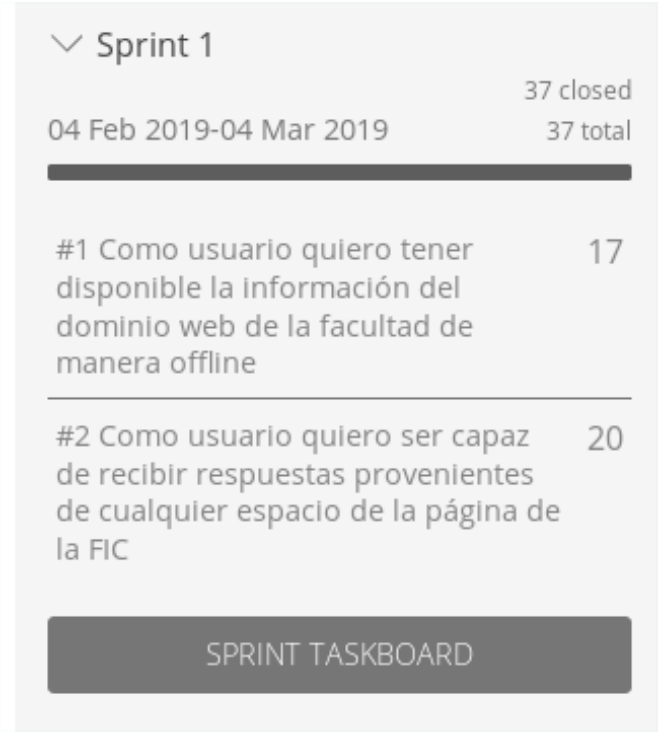

Figura 6.4: Progreso del proyecto finalizado el primer sprint

6.0.2 Sprint 2 : Construcción del Servicio REST de transmisión de audio

En este Sprint se comienza el desarrollo del servicio *REST* encargado de la comunicación con el asistente. Se ha creado una Spring Application con un controlador encargado de contener los *endpoints* para así recibir y enviar los datos al cliente.

Al mismo tiempo, se ha empezado con la base del asistente de manera muy simple, de manera que la interfaz simplemente tiene un botón para grabar audio procedente del usuario, y ver como se reproducía de vuelta en el navegador web. A la hora de elaborar el desarrollo, la comunicación se ha hecho mediante una petición POST en el cliente, con el audio que envía el usuario grabado por el cliente. Esta llamada se ha hecho utilizando AJAX con *jQuery*. El cuerpo de los datos va codificado como un *form-data*. En la página siguiente se encuentra una parte del código encargado de la recepción del audio una vez la comunicación con el servidor haya tenido éxito.

```
public class UploadForm {
    /*used to keep user session id*/
    private String session;
    /*dealing the audio with as a MultiPartFile*/
    private MultipartFile[] files;
    /*in case the input is from the keyboard*/
    private String keyboardText;
    .
    .
```

Listado 6.2: Campos de la clase UploadForm

```
public static String saveUploadedFiles(MultipartFile[] files,
    String session) throws IOException {
        StringBuilder sb = new StringBuilder();
    /*dealing with the audio as a MultiPart*/
    for (MultipartFile file : files) {
            if (file.isEmpty()) {
        continue;
      }
      /*Converting it to normal File*/
      File FileAProcesar = convert(file);
      AudioInputStream audioInputStream = null;
      try {
        audioInputStream =
    AudioSystem.getAudioInputStream(FileAProcesar);
      } catch (UnsupportedAudioFileException e) {
        e.printStackTrace();
      }
      /*need to change the rate to 16KHz for ASR system*/
      AudioFormat sourceFormat = audioInputStream.getFormat();
      AudioFormat targetFormat = new
    AudioFormat(AudioFormat.Encoding.PCM_SIGNED, WAV_SAMPLE,
          sourceFormat.getSampleSizeInBits(),
    sourceFormat.getChannels(), sourceFormat.getFrameSize(),
          sourceFormat.getFrameRate(), sourceFormat.isBigEndian());
            AudioInputStream inputStream =
    AudioSystem.getAudioInputStream(targetFormat, audioInputStream);
      AudioSystem.write(inputStream, AudioFileFormat.Type.WAVE, new
    File(getQuestionAudioFile(session)));
        }
    return sb.toString();
  }
```

Listado 6.3: Método empleado para la recepción del audio

Figura 6.5: Historias de usuario y duración del segundo sprint

Además del propio contenido del audio, se enviará un identificador de sesión al servidor. Tener un identificador para cada audio es necesario, pensando en diferenciar cada usuario. De manera que cuando se reciban diversas peticiones al asistente, poder redirigirlas hacia el usuario correcto. En este período también se creó una clase *FileUtils* para organizar el almacenamiento de los archivos. Cada archivo llevará en su nombre la propia *session* del usuario que ha generado el audio. La idea establecida es que cada pregunta originase los siguientes archivos, tomando como ejemplo el identificador de sesión *f7mztsxzl*:

- **_f7mztsxzl_question.wav** Archivo que contiene el audio de la pregunta original formulada por el usuario.
- **_f7mztsxzl_question.txt** Archivo que contendrá el texto de la pregunta original formulada por el usuario. Se mostrará en la conversación con el asistente.
- **_f7mztsxzl_response.wav** Archivo que contiene el audio de la respuesta obtenida por el sistema. Se reproducirá en el navegador del usuario.
- **_f7mztsxzl_response.txt** Archivo que contiene el texto de la respuesta obtenida por el sistema. Se mostrará en la conversación con el asistente.

A pesar de que hubo ciertas complicaciones sobre la manera de como tratar los archivos de

audio, el sprint resultó en 42 puntos de historia, quedando la media de los sprints en algunos puntos por debajo de lo estimado.

Figura 6.6: Interfaz del proyecto en esta etapa

Al finalizar esta fase, el producto permitía enviar audios al servidor. Como hasta el momento no había ningún tipo de desarrollo de la transcripción del audio, simplementa se hacían las pruebas devolviendo el mismo archivo.

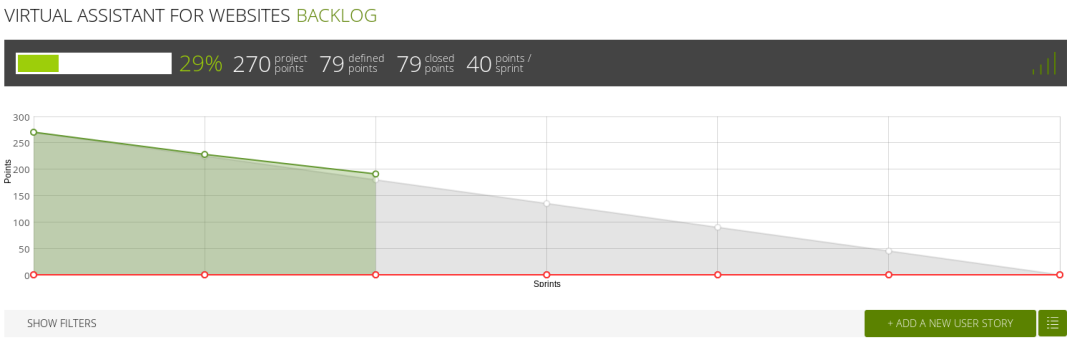

Figura 6.7: Progreso del proyecto finalizado el segundo sprint

6.0.3 Sprint 3 : Servicios de gestión del audio: Kaldi y Cotovia

En el inicio de este Sprint, la aplicación todavía no manejaba ningún tipo de transcripción de audio. Después de un estudio de las tecnologías disponibles para el desarrollo de este Sprint,

se ha decidido usar el *ASR Kaldi*[2] y el *TTS Cotovia*[3]. En el caso del motor ASR usado para el proyecto, ha sido necesario ejecutar los modelos con los datos de entrenamiento. Para la construcción de este servicio se ha seguida la guía que Kaldi proporciona en su página web[4]. Con el modelo de lenguaje sometida a un correcto entrenamiento, los modelos acústicos que proporciona Kaldi, junto con el léxico de la transcripción fonética construido en el diccionario, se ha podido generar el reconocimiento de voz [19]. El algoritmo de reconocimiento hace uso de estos tres componentes y posteriormente envía la hipótesis obtenida.

A la hora de comprobar su funcionamiento, se han hecho unas pruebas clásicas. Simplemente se usaba el micrófono en la aplicación. Una vez el audio ha sido enviado, se analizaba el contenido del archivo en texto, devuelto por el proceso de Kaldi, comparando que este texto realmente correspondía lo que había dicho el usuario.

Figura 6.8: Planificación del tercer Sprint

[2]http://kaldi-asr.org/
[3]http://gtm.uvigo.es/cotovia
[4]http://kaldi-asr.org/doc/

Una vez terminado el proceso de conversión a texto, hemos observado un cierto retraso en la transcipción a texto, ya que tenía una duración de veinte segundos. Esta espera suponía un grave error en los requisitos no funcionales puesto que las respuestas tardaban demasiado en llegar al usuario.

```
./run_decode.sh /tmp/virtualassistant/audios/_n0e7v3nrp_question.wav es | ts
12:09:37 ./steps/online/nnet2/decode.sh --config conf/decode.config --cmd run.pl

--mem 4G --nj 1 --skip-scoring true --per-utt true exp/nnet2_online/nnet_a_online/graph_es

/tmp/7017 exp/nnet2_online/nnet_a_online/decode_es
lattice-to-nbest --acoustic-scale=0.1 --n=10 'ark:gunzip -c exp/nnet2_online

/nnet_a_online/decode_es/lat.*.gz|' ark:exp/nnet2_online/nnet_a_online/decode_es/nbest.lat
LOG (lattice-to-nbest[5.5.222~1-991a7]:main():lattice-to-nbest.cc:125)

Done applying N-best algorithm to 1 lattices with n = 10, average actual #paths is 10
nbest-to-linear ark,t:exp/nnet2_online/nnet_a_online/decode_es/nbest.lat ark,t:exp

/nnet2_online/nnet_a_online/decode_es/nbest.ali ark,t:exp/nnet2_online/nnet_a_online/decode_es/nbest.words ark,t:exp/nnet2_online
/nnet_a_online/decode_es/nbest.lmscore ark,t:exp/nnet2_online

/nnet_a_online/decode_es/nbest.acscore
LOG (nbest-to-linear[5.5.222~1-991a7]:main():nbest-to-linear.cc:89)

Done 10 n-best entries, 0 had errors.
12:09:52 Done getting n-best
12:09:52 /tmp/virtualassistant/textos/_n0e7v3nrp_question.txt
```

Figura 6.9: Log con los tiempos de procesado del motor ASR.

Cada vez que se hacía una llamada se creaba un proceso, se observó que realizaba la carga de todos los modelos y datos necesarios, y ese proceso moría cuando se acababa la transcripción. El proceso de carga de los modelos de lenguaje, diccionarios, tomaba la mayoría del tiempo.

Como solución propuesta, se ha ejecutado Kaldi como servicio. Gracias a esto, la carga se realiza solo una vez y el proceso perduraría como *daemon* (proceso no interactivo, es decir, se ejecuta en segundo plano en vez de ser controlado directamente por el usuario) permaneciendo a la espera de recibir peticiones en un socket. De esta manera, se utilizaría un servicio para elaborar el proceso de decoder online. Al cargar solamente una vez los modelos necesarios para la transcripción, se obtuvieron unos resultados mucho más rápidos.

Para reproducir la respuesta, se hizo uso del sistema *text-to-speech* Cotovia. Se utilizaron los modelos entrenados para generar síntesis de voz. En esta fase todavía no se había implementado el motor de búsqueda, por lo que las pruebas se hicieron devolviendo la misma pregunta realizada por el usuario, pero mediante la voz artificial.

Esta transcripción se ha hecho mediante la creación de un proceso en el código Java, que realiza una llamada al sistema usando el módulo Cotovia, el módulo admite distintos parametros, como la opción entre transcribir en gallego o en español, sintetizar una llamada con voz

femenina o masculina, entre otros.

```
Runtime rt = Runtime.getRuntime();
try {
  Process process = rt.exec("cotovia -i " +
responseTextFilePath + " --lang=es -V sabela -O
"+FileUtils.getAudioFolder()+" -w " + responseAudioFilePath);
  String line;
  BufferedReader input = new BufferedReader(new
InputStreamReader(process.getErrorStream()));
  while ((line = input.readLine()) != null) {

  }
  input.close();

  if (process.waitFor(20, TimeUnit.SECONDS)) {
    logger.info("TTS ok");
  }
} catch (IOException | InterruptedException e) {

  e.printStackTrace();
}
```

Listado 6.4: Llamada al sistema del proceso Cotovia

Después de la llamada al sistema, se almacena el archivo con la respuesta en voz. Este archivo será enviado al cliente para que se reproduzca en el dispotisivo de los usuarios y sean capaces de escucharla.

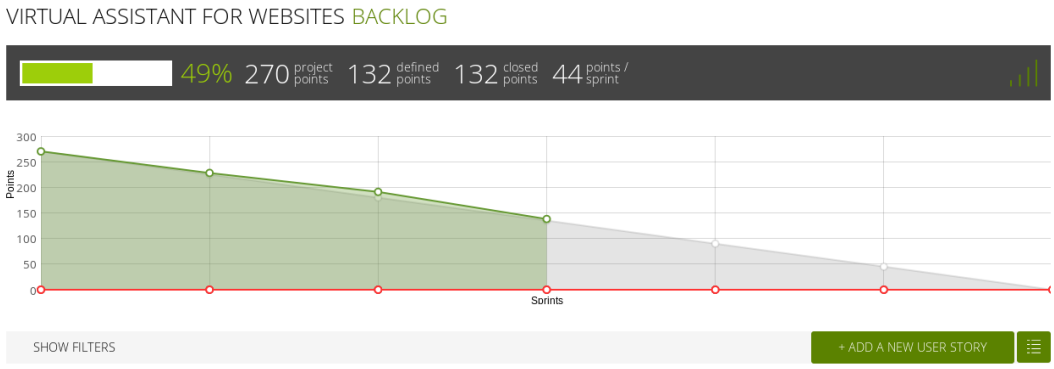

Figura 6.10: Progreso del proyecto finalizado el tercer sprint.

Una vez finalizado este Sprint, el trabajo ya era capaz de generar y transcribir todos los archivos de audio y texto necesarios para su funcionamiento. Este Sprint ha contado con más puntos historia de lo planificado inicialmente. Esto es debido a la necesidad de tener una solución para aumentar la rapidez del procesamiento de Kaldi, como hemos comentado anteriormente. Además, entrenar y probar los modelos de datos ha resultado más tedioso de lo pensado en etapas iniciales. Tal y como se ha fijado en los objetivos, conviene destacar que el proceso de entrenamiento se ha tenido en cuenta tanto para el castellano como para el gallego, de manera que será posible realizar consultas en ambos idiomas.

Figura 6.11: Plan del Sprint con las historias de usuario modificadas

6.0.4 Sprint 4 : Indexación con ElasticSearch

Finalizada la construcción del proceso de tratamiento de audio, dió comienzo el desarrollo del motor de búsqueda.

Al inicio de este Sprint se investigó como crear el motor de búsqueda haciendo uso de las funcionalidades que nos proporciona la herramienta ElasticSearch.

La totalidad de las entradas van a ser preguntas en lenguaje natural, considerada como la frase en sí, en lugar de un conjunto de palabras. Tomando como ejemplo : *"¿A qué hora es la clase de Marcos de Desarrollo"* Esta frase será transformada en una consulta sobre los índices de documentos almacenados.

Figura 6.12: Historias de usuario y duración del cuarto sprint

Instalado y funcionando el servidor de ElasticSearch, se llevó a cabo la construcción de los índices. Mediante Postman 3.3, se envió un método put hacia el servidor de Elastic con la configuración y el mapeado de los distintos índices.

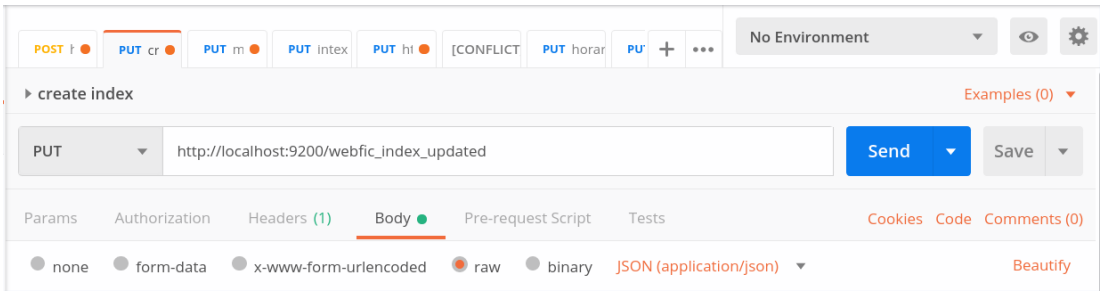

Figura 6.13: HTTP PUT de la configuración del índice *webfic_index_updated*

A la hora de elaborar la configuración de los índices, se han creado dos analizadores por cada índice.

- Analizador por defecto, cuya única excepción son la inclusión de dos filtros. Uno de ellos para cambiar toda la información a minúsculas y que el índice no sea *case sensitive*. El segundo filtro es *asciifolding*, lo que significa que convierte todos los símbolos que no estén en los primeros 127 ASCII caracteres en sus equivalentes, por lo que el sistema obvia signos de puntuación y otros símbolos especiales, como las *ñ's* y acentos en español.

- Un analizador por n-gramas, tal y como se ha mencionado en la sección 2.1.3. El analizador divide las palabras tanto en tres como en cuatro n-gramas.

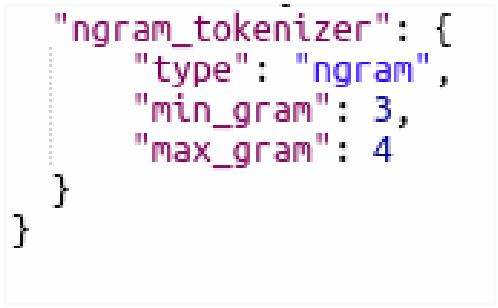

Figura 6.14: Partición en n-gramas en la configuración del índice

Después de tener los índices creados se ha desarrollado el *mapping* de los mismos. También ha consistido en un método PUT al servidor Elastic mediante *Postman*. En el mapping se ha creado un tipo de dato con los campos que se querían incluir en cada índice. Aquellos campos en los que se analice puro texto, llevarán el analizador por defecto. En los que se quiera inspeccionar por n-gramas, llevarán el analizador por n-gramas generado anteriormente. Se han generado los siguientes índices:

- ***webfic_index*** Índice para analizar las páginas web. Tanto su título, como su contenido más relevante, como el contenido al completo (sin ningún tipo de filtro).
- ***webfic_index_sentences*** Índice para analizar dividiendo en frases el texto más relevante de cada sitio web. Creará un documento por cada frase que exista en este texto.
- ***webfic_index_bigtext_sentences*** Índice para analizar dividido en frases todo el texto al completo que contenga cada sitio web. También creará un documento por cada frase.
- ***timetable_index*** Índice para analizar los horarios de las distintas asignaturas de la facultad, además del aula donde se imparten.

A la hora de separar el texto en frases, se ha hecho uso de la librería de Python *SentenceSplitter*[5]. Se ha considerado el final de una frase cuando hay un signo de puntuación como el punto seguido o punto y aparte. De esta manera el índice *webfic_index_bigtext* contiene 1.136.553 documentos, lo que nos dice que la información de las páginas web se encuentra dividido totalmente.

[5]https://pypi.org/project/sentence-splitter/

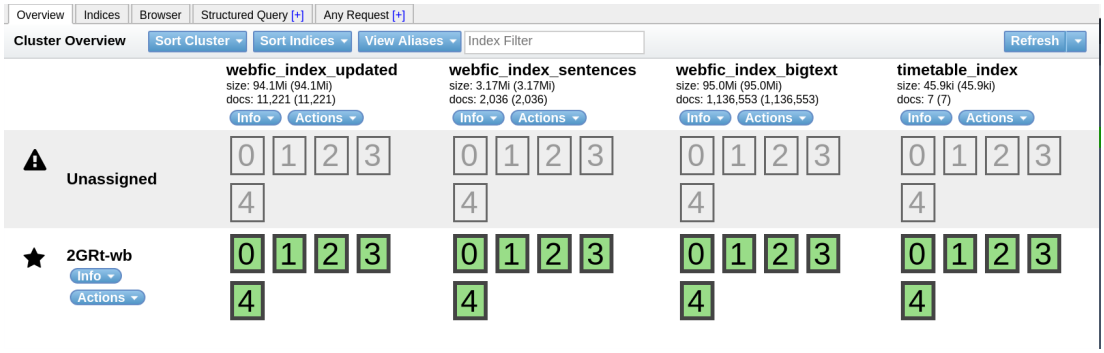

Figura 6.15: Ejemplo de los índices creados y sus documentos.

En la figura 6.15 se pueden ver cada tipo de índice creado, junto con el número de documentos que lo componen, junto con su tamaño total.

El hecho de considerar como entradas el propio lenguaje natural hace al sistema *user-friendly*, más entendible para los estudiantes. A su vez, supone mayor complejidad para su desarrollo, ya que, al existir distintos tipos de preguntas, el sistema tendrá que identificar el que mejor se adapta para enviar la mejor respuesta posible. Asignar correctamente el tipo de pregunta es una tarea crucial, puesto que de él depende la totalidad del proceso de búsqueda sobre los documentos.

Dependiendo del tipo de pregunta, se ha elaborado una estrategia de construcción de respuesta distinto. Para ello usará un NLU *(Natural Language Understanding)* encargado de recibir las entradas no estructuradas, y estructurarlas para permitir que el sistema entienda la pregunta.

La extracción de palabras es el primer paso a la hora de hacer que el sistema llegue a identificar el tipo de cuestión. En ocasiones el lenguaje puede dejar muchas ambigüedades. Sin embargo, es lógico que algunas palabras sean un gran indicador del tipo de respuesta que espera el usuario. Tomando como ejemplo la consulta planteada anteriormente, al existir la palabra *"Hora"*, es un buen indicador para clasificar la entrada como un tipo de pregunta *"Horario"*. Es por eso que el NLU de el asistente formado por plantillas, con palabras clave para reconocer qué es lo que pregunta el usuario. Sistemas más complejos emplean otro tipo de técnicas sintácticas.

Una vez asignada la clafisicación de tipo de pregunta, el sistema empieza el proceso de rastreo de los documentos en busca de las palabras clave correctas.

Los asistentes virtuales responden a tareas ad-hoc, es decir, formulan ciertas soluciones específicamente elaboradas para un problema o fin preciso. Un claro ejemplo sería cuando a estos se les pregunta por el resultado de un partido de fútbol. En este caso, manejar la consulta de horarios es totalmente equivalente a dicho ejemplo. Siendo una de las principales inquietudes en los alumnos, se ha considerado que era necesario procesar, estructurar y almacenar

la información respectiva a los horarios.

El dominio web de la facultad solo proporciona un PDF con los horarios de todos los cursos, además de otro con los exámenes para todas las convocatorias. Debido a la dificultad de obtener información directamente del PDF por parte del crawling, la solución obtenida ha sido crear un índice específico para horarios. Es un ejemplo del conjunto total de tareas que debe soportar una asistente con estas características. Se ha diseñado una arquitectura flexible para la inclusión de tantas tareas como sea necesario, para añadirlas sin demasiada dificultad. Este índice analizará los documentos de la fuente que almacena la información sobre las horas en formato JSON.

A la hora de realizar el tratamiento sobre los índices, se ha importado la librería Elastic-Search[6], también en Python. Todos los índices se han construido con la información obtenida por el *Crawling* del primer sprint. En el campo del índice de horarios, se ha hecho sobre el archivo JSON mencionado anteriormente. A cada uno de los índices se les ha añadido un identificador, así cada documento tendrá un campo *id* único como identificación.

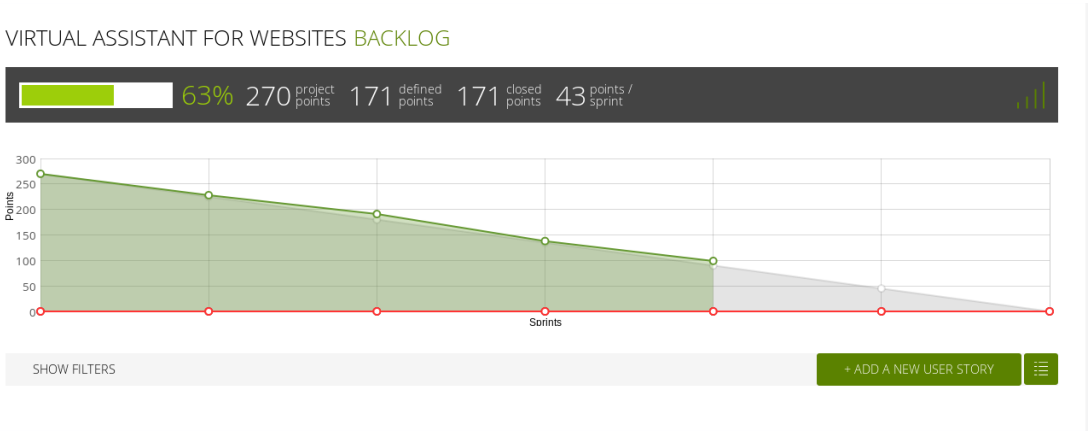

Figura 6.16: Progreso finalizado el cuarto sprint.

Este Sprint ha comprendido un total de 39 puntos de historia, coincidiendo con una etapa en la que se tuvo que dedicar menos tiempo al proyecto. La construcción de los índices consumió más tiempo de lo planificado al tener que hacer diversas modificaciones en ellos, en especial con el índice de los horarios.

Como el esfuerzo de este Sprint ha resultado algo inferior a los 45 puntos correspondientes, no ha tenido demasiada influencia en la desviación del proyecto, ya que la media sigue siendo cercana a 45. En este punto ya se han cubierto algo más de la mitad de puntos de historia totales, quedando 100 todavía por completar entre los dos últimos Sprints planeados.

[6]https://elasticsearch-py.readthedocs.io/en/master/

6.0.5 Sprint 5 : Búsquedas con ElasticSearch

Una vez desarrollado y recorrido los índices en donde se va a buscar la información, el siguiente paso consiste en construir el motor de búsqueda y su lógica.

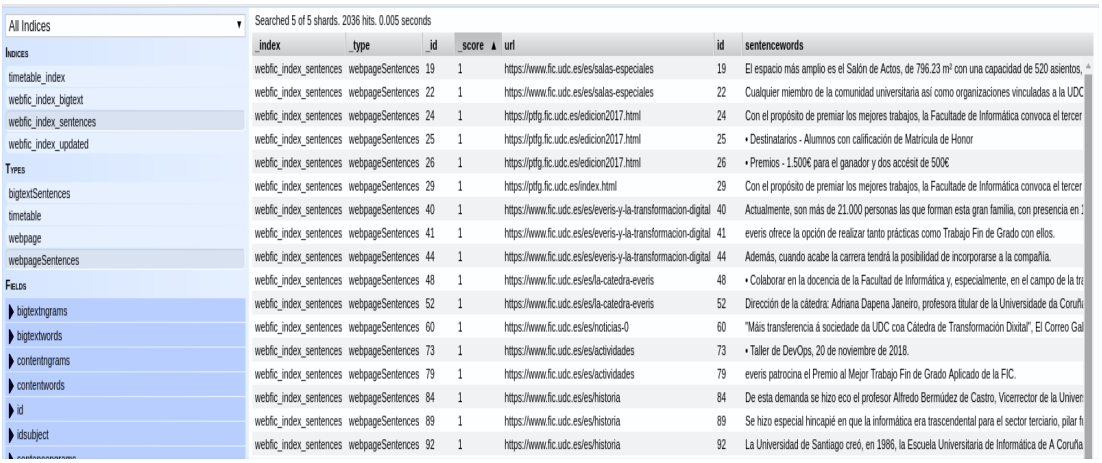

Figura 6.17: Ejemplos de algunos documentos del índice *webfic_index_sentences*

Para organizar la búsqueda, se ha creado una clase *UserIntentions*, la cual recibe como entrada las cuestiones de los usuarios al asistente, y se encarga de clasificarla y elegir la estrategia de construcción de respuesta adecuada. Se ha discutido los tipos de respuestas que suelen surgir ante la mayoría de preguntas que se realizarán al asistente, destacando tres tipos de respuestas principales:

- *Webpage*. En este tipo de consultas el asistente devolverá un enlace a una web, bien de la propia facultad o un dominio externo. Esta búsqueda por ejemplo se ejecuta cuando el usuario dice palabras como ”url”, ”página”, ”enlace”, ”sitio”, ”web”; entre otras.
- *TimeTable*. En este tipo de consultas el asistente devolverá un horario. Correspondiente a alguna asignatura de la facultad, puede ser un examen, horas de una clase teoría, el aula de unas prácticas, como ejemplos. Esta búsqueda se ejecuta cuando por ejemplo detecta palabras clave como ”horario” o ”hora”.
- *Answer*. Cuando el reconocedor no detecta ninguna equivalencia aproximada con ninguno de los otros dos tipos de pregunta, se reconduce hacia una búsqueda de respuestas en las frases del índice correspondiente. Si la puntuación de la respuesta es suficientemente alta se toma como la salida elegida. Si no lo fuera se hace un proceso ante fallos de entendimiento, y le pedirá al usuario que vuelva a introducir la pregunta.

Figura 6.18: Historias de usuario y duración del quinto sprint

Cada tipo de respuesta contiene su propio tipo de documento de ElasticSearch en la lógica del motor de búsqueda. A su vez, cada entidad tiene todos sus campos mapeados con los campos que componen el índice, incluidos los que están por n-gramas.

```
@Document(indexName = "timetable_index",type = "timetable", shards
    = 1, replicas = 0, refreshInterval = "-1")
public class TimeTable implements Scoreable{
    .
    .
    .
    }

```

Listado 6.5: Mapeado del tipo TimeTable con el índice de horarios

Cada una de las entidades contiene su propio repositorio. En estos repositorios se implementa el método de búsqueda. Este método recibe por parámetro la consulta, realiza la operación con los campos necesarios. Viene heredado de la interfaz que proporciona ElasticSearch. Devolverá una lista con el tipo de respuestas candidatas, ordenadas por ranking, en la cual nos quedaremos con la que tenga mayor puntuación.

La lógica también contempla la gestión de los errores. Si la consulta no alcanza un Score específico, se redirige hacia las demás entidades. Si ninguna alcanza un Score considerado suficientemente satisfactorio, se le pide al usuario que vuelva a introducir la pregunta. Se ha intentado que esto ocurra el menor número posible de veces, con el objetivo de mejorar la experiencia de usuario.

En este sprint ha sido necesario cambiar la configuración correspondiente al *ScoreResultsMapper* del repositorio de Elastic para poder obtener el Score. En la versión actual de Spring-Data junto con Elastic no es posible obtener directamente el Score de una consulta. Como solución se sobreescribió la clase *ScoreResultsMapper* y se le añadió un campo Score a las entidades de respuesta.

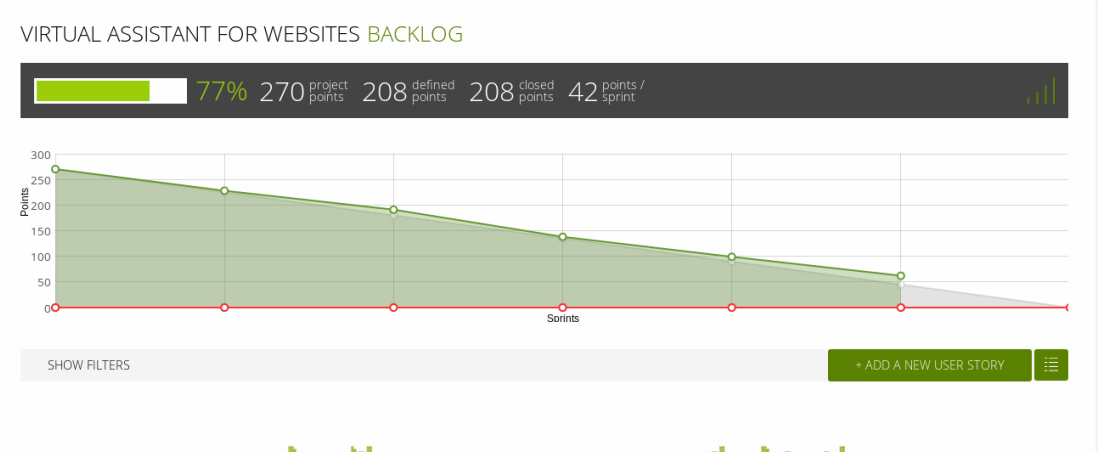

Figura 6.19: Progreso a falta de un sprint

En esta etapa se ha realizado el 77% del proyecto. Durante los Sprints anteriores, la esfuerzo real comprendido por diversas historias de usuario ha resultado inferior a lo planificado. Es por esto que para el último Sprint refleja la falta de 62 puntos de historia, cuando lo ideal deberían ser 45 puntos.

6.0.6 Sprint 6 : Interfaz asistente

El último sprint ha sido dedicado a la construcción de la interfaz que contendrá al asistente. En este Sprint se ha empezado con un estudio previo de las distintas alternativas a la hora de realizar el desarrollo de la interfaz, teniendo como base los requisitos no funcionales establecidos en la planificación 5.1.3.

Conviene mencionar que a la hora de desarrollar la comunicación del cliente con el servicio, el interfaz también actúa de controlador por medio de su JavaScript.

Figura 6.20: Planificación del último sprint

Se ha decidido usar *BotUI*, una API para JavaScript de código abierto, y ha sido necesario su modificación para añadir toda la funcionalidad y control de la comunicación por audio.

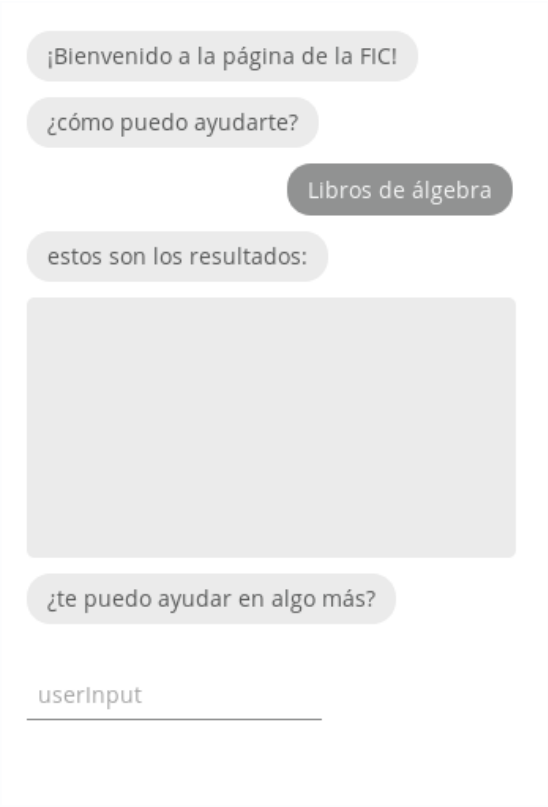

Figura 6.21: Primer diseño de la interfaz en la fase inicial, sin la integración de búsquedas

Los tres conceptos principales en los que se basa el API son:

- *Mensaje*
  Como su nombre indica, corresponde al mensaje que aparece en la conversación. Dependiendo del emisor, se puede generar a la derecha o a la izquierda de la conversación, entre otras acciones disponibles.

- *Acción*
  Acciones que puede realizar el usuario. Pueden ser desde escribir algo en la conversación hasta pulsar un botón.

- *Lógica de control*
  Su función es ir encadenando mensajes y las acciones que estos conllevan, de manera que sigue una serie de reglas dependiendo de la entrada por parte de los usuarios.

A la hora del diseño, se ha intentado proporcionar una interfaz simple y con las funcionalides básicas de un asistente.

- Garantizar la opción de formular la pregunta tanto por texto como por voz. La interfaz contendrá un espacio de texto si desea introducir las preguntas por teclado. Además de proporcionar un botón de audio para el caso de formularlas por voz.

- Las preguntas de los estudiantes se registrarán en la conversación por texto. En la parte derecha podrá verse su pregunta, tanto la haya hecho por teclado o hablando.

- Las respuestas obtenidas por el motor de búsqueda siempre estarán disponibles a la izquierda de la conversación, al mismo tiempo, se enviará un audio con la misma información.

Una vez establecido el diseño, ha sido necesario integrarlo dentro de la página web de la facultad. Conviene recalcar que esta integración se ha lanzado en una versión *offline* propia de la página al no tener acceso a la aplicación que controla la web en producción.

- El asistente debe mostrarse una vez se cargue el contenido de la página en el navegador, situado debajo en la parte derecha de la misma.

- Tendrá opción a cerrarse la ventana en el caso de que los usuarios no necesiten sus servicios para aumentar su comodidad.

- Que la conversación siga visible pese al posible *scrolleado* de los clientes sobre el sitio web.

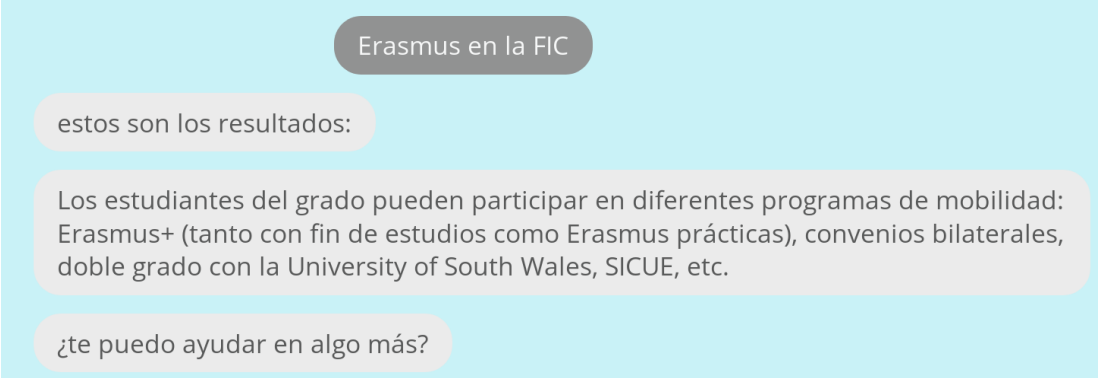

Figura 6.22: Ejemplo de consulta en un diseño en pruebas

Esta fase estuvo caracterizada por los constantes cambios y adaptaciones en el diseño del asistente, buscando un tipo de interfaz que sea lo más accesible para el uso de cualquier clase de usuario.

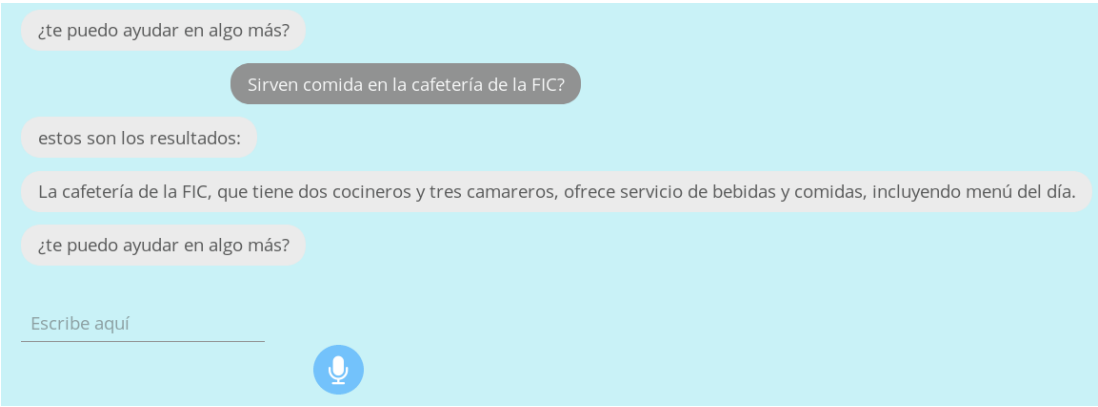

Figura 6.23: Captura de una consulta dentro de la conversación con el asistente

A la hora de medir cuanto de fácil es entender la aplicación y su interfaz gráfica, se ha realizado su validación ante una serie de usuarios inexpertos. De esta forma, se ha garantizado su usabilidad para confirmar el uso sencillo e intuitivo del trabajo, viendo la manera en que se han cumplido los requisitos no funcionales estipulados. Esta validación ha tenido lugar tanto en los navegadores web *Google Chrome* como *Mozilla Firefox*, recogiendo las experiencias de los usuarios después de un tiempo haciendo uso del asistente.

Se ha validado la aplicación tanto en idioma gallego como en castellano.

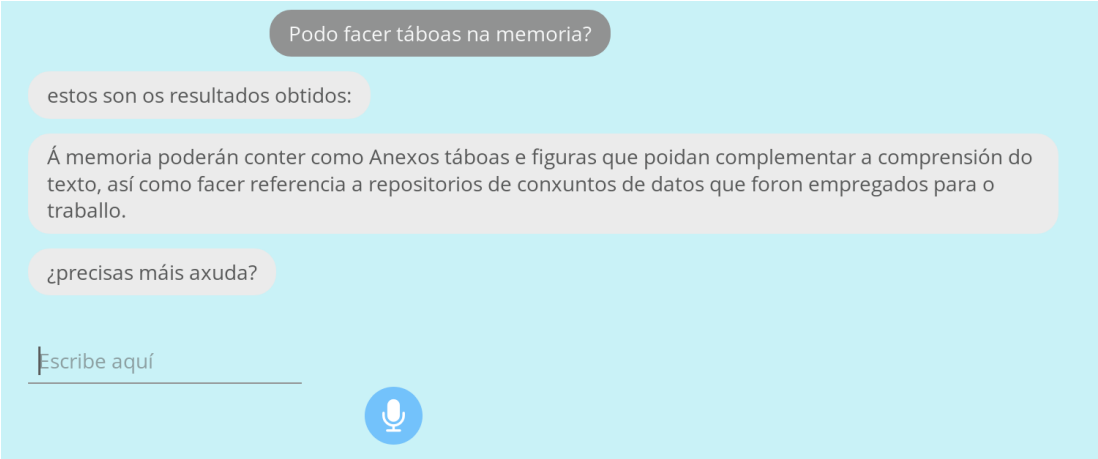

Figura 6.24: Ejemplo de consulta realizada en gallego

Figura 6.25: Diseño ejemplo de como quedaría la integración del asistente en la págin

6.0.7 Observación de los resultados

Terminado el desarrollo se elaboró una conclusión sobre el progreso del trabajo. Todos los datos correspondientes a la visión del progreso en el desarrollo han sido obtenidos gracias a la herramienta Taiga. El trabajo medio por sprint ha resultado en 42 puntos de historia, ligeramente inferior a lo establecido inicialmente.

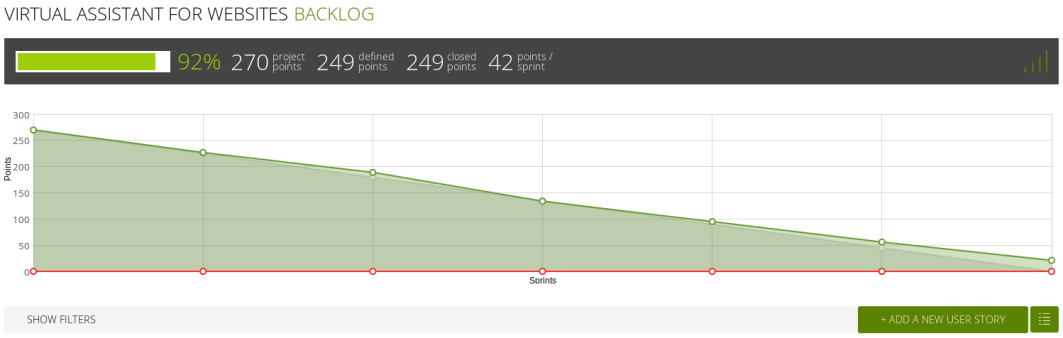

Figura 6.26: Progreso finalizado el desarrollo

En la planificación inicial, ha habido una sobrestimación de los puntos totales, ya que se ha terminado el proyecto con 249 puntos, cuando se había estipulado un esfuerzo total de 270 puntos. Debido a motivos académicas y personales, la planificación inicial sufrió diversas variaciones en algunas etapas del proyecto. Sin embargo, se han conseguido compensar, tanto con una carga mayor de trabajo en ciertos sprints como por que algunos sprints no han abarcado el esfuerzo inicial planteado, por lo que se ha logrado mantener un ritmo constante para todos los sprints. A continuación se mostrará una tabla con los puntos de historia de reales para cada una de las historias de usuario, en comparación con la creada en la tabla 5.1 correspondiente al análisis.

ID	**Historia de usuario**	**Puntos**
1	Como usuario quiero tener disponible la información del dominio web de la facultad de manera offline	20 -> **17**
2	Como usuario quiero ser capaz de recibir respuestas provenientes de cualquier espacio de la página de la FIC	25 -> **20**
3	Como usuario quiero poder ser capaz de enviarle mis preguntas al asistente	15
4	Como usario quiero poder escuchar mi propia pregunta reproducida en el navegador	20
5	Como usuario quiero ver mi propia pregunta en el interfaz cuando escribo en formato texto	7
6	Como usuario quiero poder escuchar mi propia pregunta hecha por audio con voz artificial reproducida en el navegador	20 -> **32**
7	Como usuario quiero poder ver por pantalla mi pregunta por voz en formato texto	18

8	Como usuario quiero poder escuchar mi propia pregunta hecha por texto con voz artificial reproducida en el navegador	3
9	Como usuario quiero poder acceder a los documentos indexados en mi consulta	25
10	Como usuario quiero poder acceder a los distintos tipos de documentos diferenciados por tipo de pregunta	14
11	Como usuario quiero poder obtener respuestas satisfactorias	20
12	Como usuario quiero poder obtener distintos tipos de respuestas dependiendo de mi pregunta	12
13	Como usuario quiero poder reformular la pregunta en caso de que el asistente no me haya entendido	5
14	Como usuario quiero una interfaz agradable y facil de manejar	28
15	Como usuario quiero poder tener un asistente adaptable mientras navego por la web	7

Tabla 6.1: Historias de usuario con su esfuerzo real

Capítulo 7

Conclusiones y trabajo futuro

En este último capítulo de la memoria, se hará una evaluación sobre el trabajo realizado, junto con directrices a modo de mejora y enfocadas hacia la continuación del proyecto.

7.0.1 Conclusiones

El producto conseguido al final de este proyecto, cumple con los objetivos expuestos en la primera parte de este documento. Se ha hecho un estudio en profundidad de las tecnologías necesarias para un buen desarrollo de este tipo de sistemas. El autor ha obtenido aprendizajes valiosos en conceptos de crawling, recuperación de la información, clasificadores de preguntas y procesamiento de lenguaje, en sistemas de transcripción y tratamiento de audio, además de aprender sobre la creación de índices de documentos y como recorrerlos de manera eficiente. Con este trabajo se ha conseguido un sistema que implementa y demuestra las funcionalidades de un sistema conversacional.

- Se ha construido un asistente virtual que permite su interacción con los usuarios por medio de una conversación por chat, respondiendo a las cuestiones de los mismos.
- El asistente hace uso de métodos de transcripción de audio a texto y viceversa, enviando las respuestas por escrito y por voz.
- La lógica del proceso de búsqueda de la información se ha desarrollado de acuerdo a las tecnologías planteadas, con lo que las respuestas generadas por el sistema atienden a los criterios de calidad y eficiencia planteados.
- El asistente permite a los usuarios tener una experiencia agradable. La interfaz es muy intuitiva, proporcionando una buena integración con la página de la Facultad de Informática de la UDC y de uso sencillo.

Gracias a los conceptos mencionados anteriormente, y junto con los conocimientos previos adquiridos a lo largo de la carrera, ha sido posible desarrollar este trabajo tal y como se

había planificado. Conviene destacar los siguientes conocimiento adquiridos:

- Conocimientos sobre planificación, organización y métodos de trabajo para proyectos software. Se ha profundizado sobre la metodología Scrum, que ha ayudado a resolver los problemas de manera eficaz y experimentar de primera mano como se trabaja con ella.
- Conocimientos tecnológicos, se ha ampliado la experiencia en programación Java y Spring. Especial atención en Python, nunca visto anteriormente a la ejecución del framework. Además, también se han adquirido conocimientos nuevos en desarrollo *frontend* mediante el diseño y desarrollo del cliente y su interfaz.
- Experiencia en herramientas nuevas tal y como se han introducido en la sección 3. Además de profundizar en las ya conocidas durante la carrera, puesto que se han utilizado de manera más especializada.
- Profundizar en técnicas de análisis y diseño. Construcción y creación de prototipos. Además de la gestión de los costes y la gestión en riesgos del proyecto.

Una de las cosas más útiles que ha aprendido el autor con el desarrollo de este proyecto ha sido la capacidad de tomar decisiones rápidamente y de focalizarse en un objetivo concreto. Por encima de todo, tener la capacidad de resolver problemas de manera eficaz usando las herramientas y conocimientos adquiridos.

7.0.2 Trabajo futuro

En todo momento la lógica del sistema se ha implementado intentando que facilitar su posible extensión. De esta manera se abren nuevas líneas de continuación del trabajo:

- Adaptación del modelo de lenguaje del ASR al dominio de aplicación para minimizar la cantidad de palabras fuera de vocabulario que pueden aparecer. De manera que el asistente llegue a comprender las palabras específicas de los dominios en los que se mueve la página de la facultad.
- Se ha utilizado Cotovia como motor de TTS porque, es la única herramienta de código abierto disponible para TTS tanto en castellano como en gallego. Sería interesante el desarrollo de un sistema TTS en estos idiomas empleando tecnologías más actuales que las de Cotovía.
- Mejorar la calidad del proceso de búsqueda. Aumentar el rango de tipo de preguntas a identificar, su clasificación y gestión ante los errores generados por el ruido, para mejorar así la experiencia de los usuarios.

- Extender el sistema a diferentes centros con una estructura similar. De manera que solo habría que entrenar los modelos para otras páginas y sería relativamente sencillo adaptarlo a otras facultades, institutos , entre otros.
- Con la aplicación en uso e integrada, sería posible obtener datos de entrada relativas a la interacción de los usuarios con el asistente. Así se podrían entrenar los procesos NLU y NLG para implementar un aprendizaje automático en el sistema.

Apéndices

Lista de acrónimos

TTS *Text-to-speech.*

ASR *Speech-to-text.*

QA *Question Answering.*

IR *Information Retrieval.*

Glosario

API Application Programming Interface: Conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como capa de abstracción.

JSON JavaScript Object Notation: JSON es un formato de texto sencillo para el intercambio de datos. Se trata de un subconjunto de la notación literal de objetos de JavaScript.

NLP Natural Language Processing: Término utilizado para definir la capacidad de una máquina para ingerir lo que se le dice, comprender su significado y determinar la acción a realizar.

NLU Natural Language Understanding: Término utilizado para definir la capacidad de una máquina para manejar las entradas no estructuradas y estructurarlas para llegar a comprender lo que se le dice.

NLG Natural Language Generation: Término utilizado para definir la capacidad de una máquina para generar texto a partir de una informacón estructurada de datos.

Bibliografía

[1] "Gartner Predicts a Virtual World of Exponential Change," Accedido: 05-04-2019. [En línea]. Disponible en: https://www.gartner.com/smarterwithgartner/gartner-predicts-a-virtual-world-of-exponential-change/

[2] "By 2020, the average person will have more conversations with bots than with their spouse," Accedido: 05-04-2019. [En línea]. Disponible en: https://www.linkedin.com/pulse/2020-average-person-have-more-conversations-bots-than-schotsborg

[3] "GlobalMe - Innovative Voice Controlled Devices on the Market Today," 2019, Accedido: 06-04-2019. [En línea]. Disponible en: https://www.globalme.net/blog/best-voice-controlled-devices-market

[4] "Amazon Echo is slowly losing ground to Google in smart speaker market," Accedido: 06-04-2019. [En línea]. Disponible en: https://eu.usatoday.com/story/tech/2018/05/02/almost-20-percent-americans-use-smart-speakers-such-amazon-echo-google-home/573555002/

[5] "Amazon has 10,000 employees dedicated to Alexa," Accedido: 07-04-2019. [En línea]. Disponible en: https://www.businessinsider.es/

[6] B. Brown, "Amazon confirms it has 10,000 employees working on Amazon Alexa," Accedido: 07-04-2019. [En línea]. Disponible en: https://www.digitaltrends.com/home/10000-amazon-employees-work-on-alexa/

[7] P. C.D Manning and H.Schütze, *Introduction to Information Retrieval.* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

[8] R. Woods, William A; Kaplan, *Lunar rocks in natural English: Explorations in natural language question answering*, 5th ed., 1977.

[9] D. R. Xin Li, "Learning Question Classifiers," Accedido: 15-04-2019. [En línea]. Disponible en: https://aclweb.org/anthology/C02-1150

[10] B. Galitsky and R. Pampapathi, "Can many agents answer questions better than one?" Accedido: 16-04-2019. [En línea]. Disponible en: https://firstmonday.org/ojs/index.php/fm/article/view/1204/1124

[11] C. Castillo, "Effective Web Crawling," Accedido: 24-01-2019. [En línea]. Disponible en: http://chato.cl/papers/crawling_thesis/effective_web_crawling.pdf

[12] P. Lopez-Otero, J. Parapar, and A. Barreiro, "Efficient query-by-example spoken document retrieval combining phone multigram representation and dynamic time warping," *Information Processing Management*, vol. 56, no. 1, pp. 43 – 60, 2019. [En línea]. Disponible en: http://www.sciencedirect.com/science/article/pii/S0306457318301614

[13] M. Rouse, "Im bot," Accedido: 09-05-2019. [En línea]. Disponible en: https://searchdomino.techtarget.com/definition/IM-bot

[14] A.M.Turing, "Computing machinery and Intelligence," Accedido: 03-05-2019. [En línea]. Disponible en: https://www.csee.umbc.edu/courses/471/papers/turing.pdf

[15] "IBM's Jeopardy-playing machine can now beat human contestants," Accedido: 03-05-2019. [En línea]. Disponible en: https://web.archive.org/web/20130603034018/http://www.networkworld.com/news/2010/021010-ibm-jeopardy-game.html?hpg1=bn

[16] B. Juang and R.Rabiner, "Automatic speech recognition – a brief history of the technology development," 2004. [En línea]. Disponible en: https://web.ece.ucsb.edu/Faculty/Rabiner/ece259/Reprints/354_LALI-ASRHistory-final-10-8.pdf

[17] J.-P. Haton, *La parole Numérique*, 1st ed. L'Academie En Poche, 2006.

[18] Y.-Y. Chang, "Evaluation of tts systems in intelligibility and comprehension tasks." [En línea]. Disponible en: https://www.aclweb.org/anthology/O11-1004

[19] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The kaldi speech recognition toolkit," 2011. [En línea]. Disponible en: https://publications.idiap.ch/downloads/papers/2012/Povey_ASRU2011_2011.pdf

[20] E. R. Banga, C. G. Mateo, F. J. M. Pazó, M. G. González, and C. Magariños, "Cotovia: an open source TTS for Galician and Spanish," in *"IberSPEECH 2012:""VII Jornadas en Tecnología del Habla"" and III Iberian SLTech Workshop"*, November 2012.

[21] "Most popular IDES," Accedido: 23-01-2019. [En línea]. Disponible en: https://www.infoworld.com/article/3217008/the-most-popular-ides-visual-studio-and-eclipse.html

[22] "Most Popular Development Environments. Stack Overflow 2019 Survey," Accedido: 23-01-2019 . [En línea]. Disponible en: https://insights.stackoverflow.com/survey/2019

[23] J. Highsmith, *Agile Project Management*, 2nd ed. Pearson Education, 2010.

[24] J. K. Schwaber, "The Scrum Guide," 2017, Accedido: 27-01-2019. [En línea]. Disponible en: https://www.scrumguides.org/scrum-guide.html

[25] "Guía Salarial Sector TI en Galicia 2015-2016." [En línea]. Disponible en: https://www.scribd.com/document/288511179/Guia-Salarial-Sector-TI-Galicia-2015-2016