UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Grado en Ingeniería Informática en Ingeniería del Software

Trabajo de Fin de Grado

Diseño y desarrollo de API y servicios web RESTful para la gestión de Chatbot multiplataforma: la UEx como caso de estudio

Raquel Sánchez Salas

Julio, 2022

UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Grado en Ingeniería Informática en Ingeniería del Software

Trabajo de Fin de Grado

Diseño y desarrollo de API y servicios web RESTful para la gestión de Chatbot multiplataforma: la UEx como caso de estudio

Autor: Raquel Sánchez Salas

Fdo:

Directores: Juan Hernández Núñez y Elena Jurado Málaga

Fdo:

Tribunal Calificador

Presidente:

Fdo:

Secretario:

Fdo:

Vocal:

Fdo:

Agradecimientos

A mi familia por su apoyo incondicional, en especial a mis padres por creer siempre en mí, a mi hermano por recordarme que puedo lograr todo lo que me proponga, a mis abuelos y a Miryam. Gracias por confiar en mí más que yo misma.

A Patri, a Marta y a Javi, por acompañarme, animarme y comprenderme, por formar parte de mi vida desde pequeñita.

A mis amigas, porque incluso en la distancia siempre están conmigo.

A mis compañeros de la Roso de Luna por convertir la rutina en una aventura, en especial a mis niñas del Chalet II por acompañarme y alegrarme cada día.

A mis compañeros de la carrera, por todo lo que he crecido y aprendido junto a ellos y por haber puesto una nota de humor hasta en los momentos más duros. De todos me llevo un buen recuerdo.

A mi grupo de *A Golpe de Calcetín* y compañía, por haber compartido conmigo tantos ratos, unos buenos y otros no tanto, este último curso, y a la *Friki Piña*, en especial a Sole y a Belén, por haber sido compañía incondicional desde el primer día de carrera.

A Juan y a Elena por darme la oportunidad de realizar este proyecto, por su dedicación y por guiar mi trabajo estos últimos meses.

A todos los integrantes del grupo de investigación *Quercus*, por acogerme y hacerme sentir una más desde el primer día. Especialmente, debo dar las gracias a Javier y a Jaime, por su ayuda y todo lo que me han enseñado durante el desarrollo de este trabajo.

Gracias.

Resumen

En la actualidad, cada vez son más las empresas e instituciones que apuestan por el uso de asistentes virtuales o *chatbots* para automatizar los procesos de atención a usuarios y la resolución de preguntas frecuentes. Además, el uso de múltiples dispositivos por parte de los usuarios hace necesario su desarrollo multiplataforma, aunque esto aumente su coste y complejidad.

Siendo conscientes de que, en la mayor parte de los casos es necesaria una actualización periódica de las preguntas y respuestas de un chatbot, resulta de gran utilidad la existencia de un sistema o plataforma que facilite la gestión de estos datos por parte de personal, no informático, de las empresas e instituciones.

En presente Trabajo de Fin de Grado se implementa y desarrolla una API Restful y una plataforma web para la gestión de los datos con los que trabaja Lyx, asistente virtual de la Universidad de Extremadura. La plataforma web, a través de una sencilla interfaz, pretende facilitar el manejo de los datos por parte del personal de la Sección de Información y Atención al Alumnado (SIAA) de la UEX, de manera que sea posible añadir y modificar preguntas y respuestas de manera sencilla.

Por otro lado, con la implementación de la API, se pretende alcanzar un mayor nivel de abstracción en el funcionamiento de la aplicación, pudiendo ser utilizada tanto por el chatbot en sí como por la plataforma web.

Abstract

Nowadays, more and more companies and institutions are betting on the use of virtual assistants or chatbots to automate user service processes and the resolution of frequently asked questions. In addition, the use of multiple devices by users makes its multiplatform development necessary, although this increases its cost and complexity.

Being aware that, in most cases, it is necessary to regularly update the questions and answers of a chatbot, the existence of a system or platform that facilitates the management of this data by companies and institutions staff, is very useful.

In this Final Thesis, a Restful API and a web platform are developed to manage the data used by Lyx, the virtual assistant at the University of Extremadura. The web platform, through a simple interface, aims to facilitate the handling of data by the staff of UEX Student Information and Assistance Section, so that it is possible to add and modify questions and answers in a simple way.

Moreover, by means of the API implementation, it is intended to reach a higher level of abstraction in application performance, being able to be used by both, by the chatbot itself and by the web platform.

Índice general

1. Introducción **1**

1.1. Contextualización 1

1.2. Objetivos 3

1.3. Motivación 5

1.4. Estructura del documento 7

2. Estado del arte **9**

2.1. Chatbots y su mundo 9

2.1.1. Concepto de bot y chatbot 10

2.1.2. Taxonomía de los chatbots 11

2.2. Antecedentes y sistemas actuales 13

2.2.1. Arquitectura actual 13

2.2.2. Nueva arquitectura 19

2.3. Posibles tecnologías para el desarrollo 21

2.3.1. Tecnologías para el almacenamiento de datos 22

2.3.2. Bases de datos relacionales 22

2.3.3. Bases de datos No SQL 30

2.3.4. Tecnologías para la implementación del BackEnd 36

2.3.5. Tecnologías para la implementación del FrontEnd 40

2.3.6. Justificación de las Tecnologías Elegidas 45

3. Propuesta de la solución **47**

3.1. Descripción general de la solución 47

3.2. Arquitectura de la solución 48
3.2.1. Arquitectura inicial. Mejoras propuestas 48
3.2.2. Nueva arquitectura propuesta 51
3.2.3. Base de datos: contextos y preguntas 58
3.2.4. API REST 60
3.2.5. Aplicación Web 66
3.2.6. Integración de los nuevos componentes en la arquitectura. Mejoras derivadas. 71

4. Validación **74**
4.1. Validación de la API REST 74
4.2. Pruebas sobre la aplicación web 76
4.3. Validación del sistema completo 77

5. Gestión del Proyecto **79**
5.1. Resolución de incidencias en Jira 80

6. Conclusiones y trabajos futuros **82**
6.1. Trabajos Futuros 83

Anexo I: Prototipo inicial o wireframe de la aplicación **86**

Anexo II: Manual de usuario de la aplicación **92**

Anexos **86**

Bibliografía **104**

Índice de figuras

2.1. Arquitectura general de un chatbot 14
2.2. Arquitectura real del framework 15
2.3. Arquitectura real del framework con tecnologías empleadas 15
2.4. Componentes de la Lógica del Bot 16
2.5. Nueva arquitectura general del framework 19
2.6. Diagrama del componente Answers 21

3.1. Arquitectura inicial del framework 49
3.2. Arquitectura inicial del framework con tecnologías empleadas 49
3.3. Arquitectura de los nuevos componentes 52
3.4. Nueva arquitectura del framework 54
3.5. Componentes modificados y añadidos al framework 55
3.6. Diagrama de tecnologías y nuevos componentes de la arquitectura 56
3.7. Diagrama de tecnologías y componentes de la arquitectura integrada . . . 56
3.8. Diagrama de tecnologías del componente Answers 57
3.9. Fragmento de la tabla Contextos 59
3.10. Fragmento de la tabla Preguntas 60
3.11. Casos de Uso de la Aplicación 69

4.1. Pruebas de la API en Postman 76

5.1. Diagrama de flujo acumulado del proyecto 80

1. Página de bienvenida 86

2. Página de inicio de sesión 87
3. Página principal de la aplicación 87
4. Página que muestra la lista de contextos 88
5. Página para editar/añadir un nuevo contexto 88
6. Página que muestra la lista de preguntas 89
7. Página que muestra la lista de preguntas por contexto 89
8. Página para editar/añadir una nueva pregunta 90
9. Página que muestra los sinónimos de un intent 90
10. Página para editar/añadir una nueva pregunta 91
11. Página para confirmar el borrado 91

12. Página de bienvenida de la aplicación 92
13. Ventana de inicio de sesión 93
14. Página de menú inicial de la aplicación 94
15. Página de visualización y edición de contextos 94
16. Página mediante la cual se añade un contexto 95
17. Página de edición de contexto 96
18. Ventana de confirmación para borrar contexto 96
19. Ventana que informa de que no se puede eliminar un contexto 97
20. Página que facilita la gestión de los intents de un contexto 97
21. Página mediante la cual se añade un intent 98
22. Página de edición de intents 98
23. Página que muestra los sinónimos de un intent 99
24. Página que contiene el formulario para añadir un sinónimo 100
25. Ventana modal de confirmación para eliminar un sinónimo 100
26. Opción ”Finalizar”tras realizar modificaciones en los sinónimos 101
27. Página de entrenamiento de la aplicación 102
28. Ventana de alerta mostrada al añadir sinónimo 102
29. Pantalla de consulta de intents 103

Capítulo 1

Introducción

En este primer capítulo, se pretende introducir al lector en el proyecto, estableciendo el contexto, los objetivos, la motivación y la estructura del presente documento.

1.1. Contextualización

En la actualidad, la tecnología se encuentra cada vez más presente en el día a día de las personas sin que apenas se percaten de ello. Esto se observa en el creciente uso de dispositivos, aplicaciones y demás herramientas mejoradas por la tecnología que ha significado un ahorro considerable de tiempo y esfuerzo de trabajo, haciendo las tareas cotidianas más sencillas y logrando un estilo de vida más cómodo [1].

Además, a causa de la pandemia de COVID-19, la sociedad y el mercado se hallan en una situación forzada a la digitalización. Esto ha derivado en que ciertas aplicaciones, cuyo uso ya estaba ampliamente extendido antes de la pandemia, como son plataformas de mensajería, vídeo conferencias, asistentes virtuales o chatbots, se encuentren ahora en pleno apogeo. Numerosos procesos que antes se realizaban de manera presencial tienen que llevarse a cabo a distancia, jugando la tecnología un papel primordial [2, 3].

En esta digitalización, las empresas e instituciones están reformulando el modo en que atienden a sus usuarios y clientes, adoptando para ello sistemas como los bots conversacionales e integrando en sus redes sociales diversos tipos de bots orientados a ofrecer servicio y ayuda a los usuarios [4].

Los chatbots constituyen una realidad y han llegado para quedarse. Cada vez son más las instituciones y empresas que están adoptando este tipo de sistemas para gestionar la atención al usuario. Tanto grandes como pequeñas empresas los incluyen en sus procesos de negocio para tareas como atención al cliente o comercio electrónico [5].

La elección y adopción de chatbots por parte de empresas e instituciones se debe a la escalabilidad y versatilidad de este tipo de sistemas, que pueden ser desplegados en diferentes entornos como sitios web o redes sociales, y así servir a un gran número de usuarios al mismo tiempo, tarea que un humano no podría llevar a cabo. Todo ello reduce considerablemente los tiempos de espera y aumenta la calidad del servicio al cliente, mejorando así los comentarios y la satisfacción de los usuarios [6].

Las instituciones educativas también se están sumando a esta iniciativa con el objetivo de ofrecer una atención continua a sus alumnos [7]. Un excelente ejemplo es la Universidad de Extremadura que, con el fin de que los estudiantes estén siempre totalmente informados y atendidos, ofrece un original servicio: el asistente virtual bautizado como Lyx. Este chatbot ayuda a los estudiantes proporcionándoles información de interés además de recordarles tareas, entregas y plazos [8].

En el caso de resolución de preguntas frecuentes, resulta óptimo el uso de chatbots sensibles al contexto, es decir, que agilicen los diálogos teniendo en cuenta las preguntas que haya realizado previamente el usuario, evitando así que las conversaciones sean tediosas [9, 10]. Lyx es capaz de llevar a cabo esta tarea, pudiendo responder a la misma pregunta de forma distinta dependiendo del contexto en el que se encuentre.

Lyx está disponible en dos plataformas, Telegram [11] y Alexa [12], siendo el estudiante el público principal al que va dirigido. El presente Trabajo de Fin de Grado se

centra en ampliar y mejorar el framework empleado por Lyx, facilitando la gestión de los datos que utiliza para su funcionamiento, modificando y añadiendo nuevos elementos a la arquitectura para incrementar su eficiencia y facilitar su integración con diversas plataformas.

1.2. Objetivos

El objetivo y alcance principal de este Trabajo de Fin de grado consiste en ampliar el framework para el desarrollo de chatbots que emplea Lyx, asistente virtual de la Universidad de Extremadura. Esta ampliación reside en llevar a cabo la implementación y desarrollo de una API REST y una plataforma web para la gestión de los datos con los que trabaja el asistente.

La plataforma web pretende, gracias a una interfaz simple y orientada al usuario, facilitar el manejo de los datos por parte del personal del Servicio de Información y Atención al Alumnado (SIAA) de la UEx, de manera que sea posible añadir y modificar preguntas y respuestas de manera sencilla.

Por otro lado, con la implementación de la API se pretende alcanzar un mayor nivel de abstracción en el acceso a los datos que requiere el chatbot para su funcionamiento. Esta API podría ser utilizada tanto por el asistente en sí como por la aplicación web.

Teniendo en cuenta la finalidad principal, es posible establecer los objetivos específicos que se desean abordar. Estos se presentan a continuación:

- Profundizar en los conceptos relevantes del mundo de los chatbots.
- Comprender y analizar la arquitectura actual del framework que emplea Lyx para establecer las ampliaciones que se pretenden llevar a a cabo.
- Estudiar las diferentes tecnologías alternativas que se podrían emplear para el desarrollo del proyecto así como seleccionar aquellas más adecuadas.

- Establecer de manera clara las ampliaciones que se van a realizar y destacar aquellos componentes añadidos que diferencian la nueva arquitectura de la inicial, estos son la aplicación web y la API. Clarificar toda la estructura mediante el uso de diagramas

- Migrar los datos con los que trabaja el asistente desde la fuente actual, archivos CSV, a una base de datos. A esta base de datos es a la que accederá la API.

- Lograr una mayor eficencia en la gestión de los diálogos que es capaz de mantener el asistente gracias al uso de nuevos atributos que se añaden a las tablas de la base de datos.

- Desarrollar e integrar la API REST y la aplicación web fijadas como objetivos principales con los elementos que componen la arquitectura inicial, de manera que los elementos existentes no tengan que verse modificados o, si es necesario realizar cambios en ellos, que estos sean mínimos.

- Lograr que la API REST ofrezca una capa de abstracción para la gestión de datos siendo capaz de gestionar de manera simultánea todas las fuentes de datos que emplea el sistema, estas son la base de datos y la plataforma de Procesamiento de Lenguaje Natural (NLP).

- Satisfacer mediante la creación de la aplicación web la necesidad de los usuarios de ser capaces de actualizar, de manera sencilla y sin intermediarios, la información que emplea el asistente virtual para su funcionamiento. Estos usuarios son el personal de la Sección de Información y Atención al Alumnado (SIAA) de la universidad.

- Desplegar la arquitectura desarrollada para poder validarla mediante la realización de pruebas y ejecuciones correspondientes.

- Validar el sistema completo mediante el diseño y desarrollo de distintas pruebas sobre la API y la aplicación web.

1.3. Motivación

Como ya se ha comentado en las secciones anteriores, en la actualidad, cada vez son más las empresas e instituciones que apuestan por el uso de asistentes virtuales o chatbots para automatizar los procesos de atención a usuarios y la resolución de preguntas frecuentes. Su éxito reside, en gran medida, en la versatilidad y escalabilidad que ofrecen, suponiendo además un ahorro considerable de tiempo y dinero [13, 14].

Asimismo, los chatbots son capaces de atender las dudas y demandas de numerosos usuarios a la vez en cualquier momento, lo que una persona no puede hacer, ofreciendo un trato personalizado en tiempo real, disminuyendo el tiempo de espera de los usuarios para ser atendidos y mejorando la calidad de servicio. Este hecho aumenta la satisfacción de los usuarios y muestra a las empresas e instituciones más accesibles al público [6, 15].

Es por todos los motivos descritos que las principales empresas multinacionales de tecnología han invertido en el desarrollo de asistentes virtuales, como el caso de Apple con Siri [16], DialogFlow de Google [17] y Alexa de Amazon [12]. Estos asistentes son capaces de atender a miles de usuarios de todo el mundo y, en el caso de Alexa, de integrarse con cientos de dispositivos IoT para acercarse lo máximo posible a los usuarios. Otro ejemplo notable de apuesta por este tipo de sistemas es el de Microsoft, no solo con Cortana en Windows [18], sino debido a la adquisición de varias empresas del sector en los últimos años [19, 20, 21]. Asimismo, estas grandes empresas no sólo ofrecen asistentes virtuales propios, sino que también brindan herramientas para el desarrollo de aplicaciones con ellos.

No obstante, no son solo empresas las que están interesadas en este tipo de sistemas, sino que existen proyectos centrados en investigar la aplicación de chatbots en ámbitos diferentes al de atención al cliente o comercio electrónico. En esta línea, ciertas instituciones, entre otras las educativas como universidades, también se están sumando a la iniciativa de adoptar asistentes virtuales con el fin de ofrecer atención continua a sus alumnos, manteniéndoles siempre informados [22].

Un caso destacable es el de la Universidad de Extremadura que, a través de su asistente virtual Lyx resuelve las cuestiones de los estudiantes, proporcionándoles información de interés y recordándoles sus tareas, entregas y plazos [23].

El uso de múltiples dispositivos y plataformas por parte de los usuarios hace necesario su despliegue multiplataforma, uno de los grandes problemas al que se enfrentan los desarrolladores de chatbots. Los potenciales usuarios se encuentran en multitud de plataformas de información, principalmente redes sociales, como Facebook, Instagram o Twitter; sistemas de mensajería, como WhatsApp y Telegram; y páginas web. Por tanto, si las empresas e instituciones desean llegar hasta los usuarios y no quedarse atrás, deben buscar soluciones innovadoras para poder integrar sus asistentes en diversos canales, aunque esto suponga un mayor coste y complejidad de desarrollo [15].

Además de admitir diversas plataformas que sirvan de canal de comunicación con los usuarios, es conveniente que estos sistemas toleren más de una fuente de datos y que estas puedan modificarse y sustituirse por otras sin que se vean afectados el resto de elementos del sistema. He aquí la importancia de la existencia de una capa de abstracción para el acceso a datos y conexión a entidades externas, de forma que posibles cambios en ellas no influyan en la lógica del bot ni otros elementos que lo componen.

Siendo conscientes de que, en la mayor parte de los casos es necesaria una actualización periódica de las preguntas y respuestas que requiere el chatbot para funcionar, otro problema es la búsqueda de una forma de manejar estos datos de manera sencilla. Para ello, resulta de gran utilidad la existencia de un sistema o plataforma que facilite la gestión de estos datos por parte de personal, no informático, de las empresas e instituciones.

Volviendo al ejemplo de Lyx, cada vez que se produce un cambio en fechas clave, como es EBAU, plazos de matrículas y becas, los miembros del SIAA de la UEX se tienen que poner en contacto con los administradores y desarrolladores del chatbot para que actualicen la información y el asistente proporcione información verídica y no obsoleta. Es aquí donde se pone de manifiesto la necesidad de una plataforma o sistema sencillo que permita gestionar los datos de manera directa al personal del SIAA. El tipo de plataforma

que mejor se adapta es una página web.

Para el desarrollo del presente proyecto, se toma como caso de estudio el asistente virtual Lyx, cuyo framework de desarrollo aspira a ofrecer una solución para que sea posible integrar varias plataformas unificadas en un solo sistema [15], ahorrando así costes de mantenimiento y desarrollo. Sin embargo, tal como se ha descrito previamente, este framework no facilita una gestión simple de los datos ni cuenta con una capa de abstracción para el acceso a los mismos y a la conexión con otros sistemas externos, por lo que estos serán los aspectos que incentiven al desarrollo de este trabajo.

1.4. Estructura del documento

Una vez descritos los objetivos propuestos y la motivación que impulsa la decisión de desarrollar este Trabajo de Fin de Grado, se procede a explicar la estructura de la presente memoria. Esta se divide en diferentes capítulos organizados de la manera siguiente:

En el Capítulo 1, el actual, se presenta la introducción al trabajo, estableciendo el contexto, los objetivos y la motivación para realizarlo.

En el Capítulo 2 se incluye el estudio del Estado del Arte, con el que se pretende profundizar en los conceptos relevantes del mundo de los chatbots y analizar la arquitectura actual del framework que utiliza el asistente Lyx. Además, se realiza una descripción general de las ampliaciones que se plantean llevar a cabo y se estudian las distintas tecnologías alternativas que se podrían emplear para materializar estos cambios, seleccionando al final las más adecuadas.

En el Capítulo 3 se detalla la propuesta de la solución, describiendo los nuevos componentes que se añaden a la arquitectura original de manera independiente, explicando su desarrollo y función, además de las modificaciones que se realizan para lograr una integración óptima en el sistema.

En el Capítulo 4 se explican las tareas llevadas a cabo en el proceso de validación, comentando las pruebas realizadas sobre los componentes de manera individual y sobre la arquitectura completa tras su integración. Así se verifica el correcto funcionamiento del sistema en su totalidad.

En el Capítulo 5 se presenta la gestión del proyecto, explicando cómo se ha organizado el desarrollo del proyecto, la metodología seguida y la división temporal del trabajo.

Finalmente, en el Capítulo 6, establecen las conclusiones del proyecto y trabajos futuros con los que se pueda ampliar el mismo.

Capítulo 2

Estado del arte

En el presente estudio del estado del arte, en primer lugar, se expone el concepto de chatbot y sus tipos. Seguidamente se comentan antecedentes y sistemas actuales que dan solución a problemáticas similares a las que se enfrenta este proyecto, es decir, sistemas empleados en la gestión de chatbots multiplataforma. En este análisis se hace especial hincapié en el caso de estudio del proyecto, el chatbot empleado en la UEX, estudiando su arquitectura actual y destacando los cambios que se plantean realizar en ella. Asimismo, se investigan las posibles tecnologías útiles para el desarrollo de cada uno de los elementos que componen la arquitectura del proyecto, enumerando aquellas que se emplearán finalmente en la propuesta de la solución.

2.1. Chatbots y su mundo

En esta sección se pretende introducir el concepto de bot, chatbot y términos fundamentales asociados a ellos.

2.1.1. Concepto de bot y chatbot

Se define bot como un programa informático o software que es capaz de efectuar de manera automática tareas repetitivas o ejecutar una secuencia de comandos o funciones autónomas tras recibir una entrada. Posee capacidad de interacción, cambiando de estado para responder a un estímulo, e intenta imitar la conducta humana. Su principal utilidad actualmente reside en la automatización de tareas en diversos ámbitos [24].

Existen diferentes tipos de bots, como los **chatbots**, el tipo más común, diseñados para mantener conversaciones con los usuarios; los **bots informativos** y **crawlers** o **rastreadores web**, cuya finalidad es recabar y gestionar información de diversos tipos encontrada en la web para, por ejemplo, mostrar resultados coherentes al realizar una búsqueda [25]. Existen también **bots transaccionales**, orientados a la automatización de procesos empresariales que requieren de la interacción con sistemas externos [26]; los denominados **spambots**, diseñados para enviar propaganda y correos no deseados; y los **bots de vídeojuegos**, diseñados para emular el comportamiento de un jugador real compitiendo e interactuando con otros [27, 28].

Concretamente, un **chatbot** o **bot conversacional** puede entenderse como un software o servicio programado para mantener una conversación con el usuario, que interactúa con él mediante un chat o asistente de voz. Son capaces de proveer respuestas automáticas a las cuestiones planteadas de manera útil y concisa. Los chatbots poseen multitud de usos y funcionalidades actualmente y se encuentran presentes en diversidad de plataformas [29, 30].

Algunos de los principales usos de los chatbots son los siguientes [31]:

- Atención al usuario o cliente.
- Respuesta a preguntas frecuentes.
- Asistentes personales.

- Encuestación.
- Venta directa.
- Generación y cualificación de oportunidades de venta o leads.
- Gestión de servicios.

2.1.2. Taxonomía de los chatbots

Los chatbots pueden ser clasificados según distintos criterios, como su utilidad, tecnología o usuarios a los que va dirigido. En esta taxonomía se distinguen y explican brevemente distintos tipos de chatbots en función de su tecnología, tipo de interacción y canal empleado [32].

Según su tecnología

En función de la tecnología o "inteligencia" que empleen, los chatbots se pueden clasificar en:

- **Dumb chatbots:** Chatbots de lógica secuencial que implementan comandos simples. Solo responden ante cuestiones muy específicas. Perfectos para responder a preguntas frecuentes.
- **Con tecnología "word-spotting:"** Chatbots capaces de reconocer ciertas palabras clave al generar las respuestas.
- **Con Inteligencia Artificial:** Son los chatbots más actuales y avanzados, logrando ir más lejos en la comprensión de usuarios y descubrimiento de sus necesidades. Gracias a técnicas de Machine Learning son capaces de interpretar el lenguaje humano para desvelar la intención del usuario y el contexto de la conversación, dando respuestas más precisas. Es en este grupo donde se ubica el chatbot que cosntituye el caso de estudio de este proyecto.

Según el modo de interacción

Según el modo en el que los usuarios interactúan con el chatbot, se puede establecer la clasificación siguiente:

- **Chatbots textuales:** Interactúan con el usuario a través de mensajes escritos.
- **Chatbots multimedia:** Combinan mensajes de texto con elementos multimedia, como GIFs, imágenes y vídeos, provocando que la conversación sea más dinámica.
- **Chatbots de voz:** Los usuarios se expresan de forma oral, de modo que el chatbot debe interpretar las peticiones habladas respondiendo mediante un altavoz. Es el caso de asistentes personales como Siri o Alexa.

Según el canal empleado

Para interactuar con los chatbots se pueden emplear distintos canales, que originan los siguientes grupos:

- **De sitio web:** Chatbots integrados en páginas web para ofrecer atención a los usuarios visitantes.
- **En redes sociales:** Chatbots integrados en redes sociales para asegurar el compromiso con los usuarios, pudiendo responder a sus cuestiones en cualquier momento.
- **Mensajería instantánea:** Chatbots integrados en plataformas de mensajería instantánea como Whatsapp o Telegram que resuelven preguntas de usuarios mediante mensajes de manera simple.
- **Chatbots omnicanal y multiplataforma:** Chatbots capaces de ser integrados en diversidad de canales y plataformas, ofreciendo una experiencia de usuario sin fricciones a través del medio que el usuario escoja.

2.2. Antecedentes y sistemas actuales

En esta sección se pretende realizar un análisis de los antecedentes del proyecto y modo de funcionamiento del sistema actual, para que así el lector se haga una idea de la base de la que parte el trabajo.

2.2.1. Arquitectura actual

La arquitectura de la que se parte para el desarrollo de este Trabajo de Fin de Grado consiste en el diseño de un framework para el desarrollo y despliegue de bots multiplataforma. Concretamente, el caso de estudio es el sistema que conforma el chatbot empleado por la Universidad de Extremadura para responder a las preguntas frecuentes realizadas por los usuarios [15].

Este marco consta de diferentes componentes, imprescindibles para la implementación de chatbots. La integración de estos componentes da como resultado un sistema final capaz de mantener conversaciones teniendo en cuenta el contexto de cada una de ellas, así como los usuarios que interactúan con el chatbot en cada momento.

El sistema debe responder correctamente a cada una de las consultas lanzadas por usuarios, estén o no contemplados en el sistema, debiendo además ser capaz de soportar la conexión o conexiones con sistemas externos necesarios para el funcionamiento óptimo del chatbot.

Estos componentes también permiten la conexión con diferentes fuentes de información para actualizar u obtener las respuestas a cada uno de los *intents*, es decir, preguntas que plantea el usuario. De este modo, se simplifica el desarrollo y mantenimiento del proyecto.

El esquema de la figura 2.1 muestra cómo es la arquitectura general de un chatbot y el proceso que siguen las conversaciones con los usuarios:

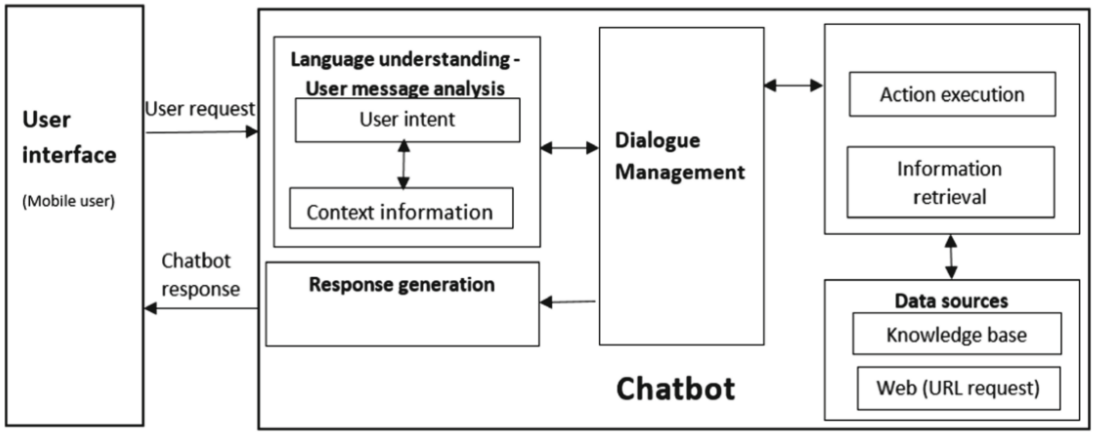

Figura 2.1: Arquitectura general de un chatbot

La arquitectura del framework empleada en el chatbot de la UEX se basa en la anterior. Concretamente, la arquitectura real se muestra en la figura 2.2.

En este esquema se muestra el modo en el que los clientes se conectan con el framework a través de su API, que transfiere la entrada del usuario a la lógica del bot. A su vez, esta lógica se conecta a la plataforma de procesamiento de lenguaje natural (NPL) para procesar la entrada y obtener la respuesta a la pregunta desde el repositorio de respuestas. También puede conectarse a un sistema externo para ello, o con el fin de realizar cualquier otra operación.

Si se sustituye en el esquema cada componente por la tecnología realmente empleada, el esquema resultante es el que se presenta en la figura 2.3.

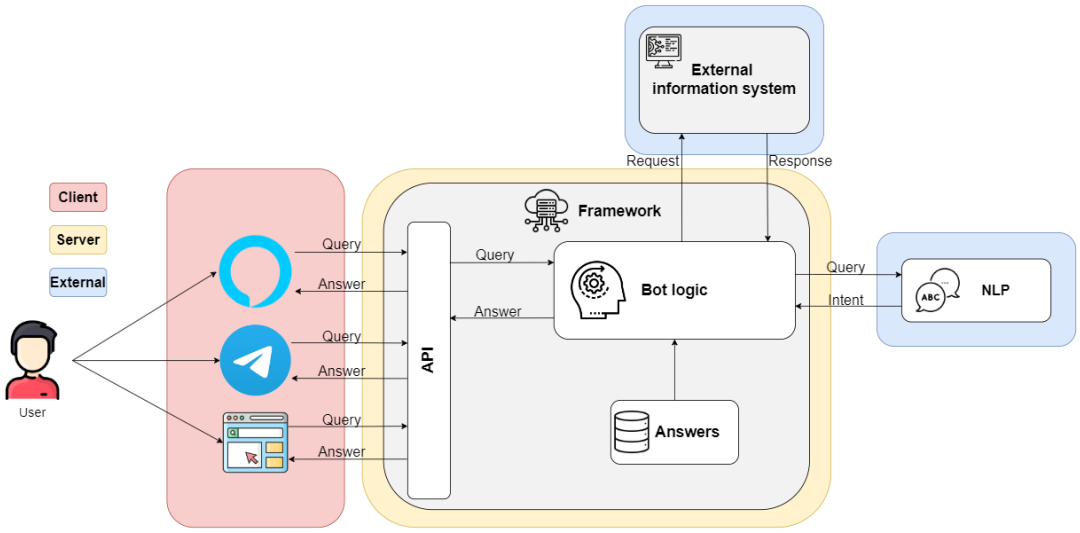

Figura 2.2: Arquitectura real del framework
[33]

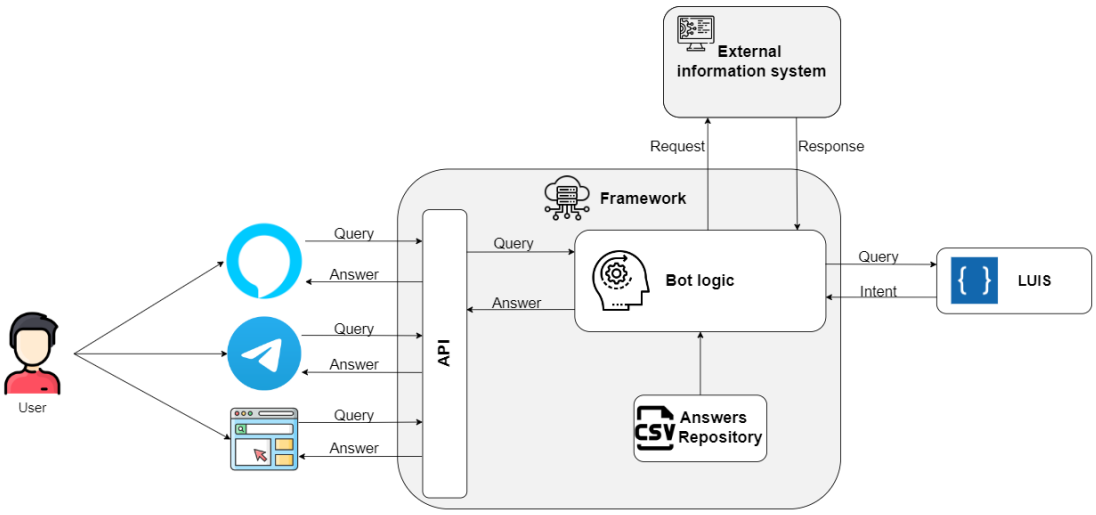

Figura 2.3: Arquitectura real del framework con tecnologías empleadas
[15]

A continuación, se describe brevemente el funcionamiento de los diferentes componentes:

Lógica del bot

Este componente está formado por diferentes elementos, cuya integración da como resultado el funcionamiento óptimo de la lógica del bot.

Su funcionamiento consiste en, una vez leída la entrada del usuario, conectarse con la plataforma de Procesamiento de Lenguaje Natural (NLP) para obtener la intención, extraer su contexto teniendo en cuenta los diálogos establecidos y, si es necesario, emplea las funciones definidas en utilidades para realizar alguna acción. Por último, el diálogo maneja la intención y genera la respuesta que se envía al usuario.

En la figura 2.4 se pueden observar los elementos que conforman este componente:

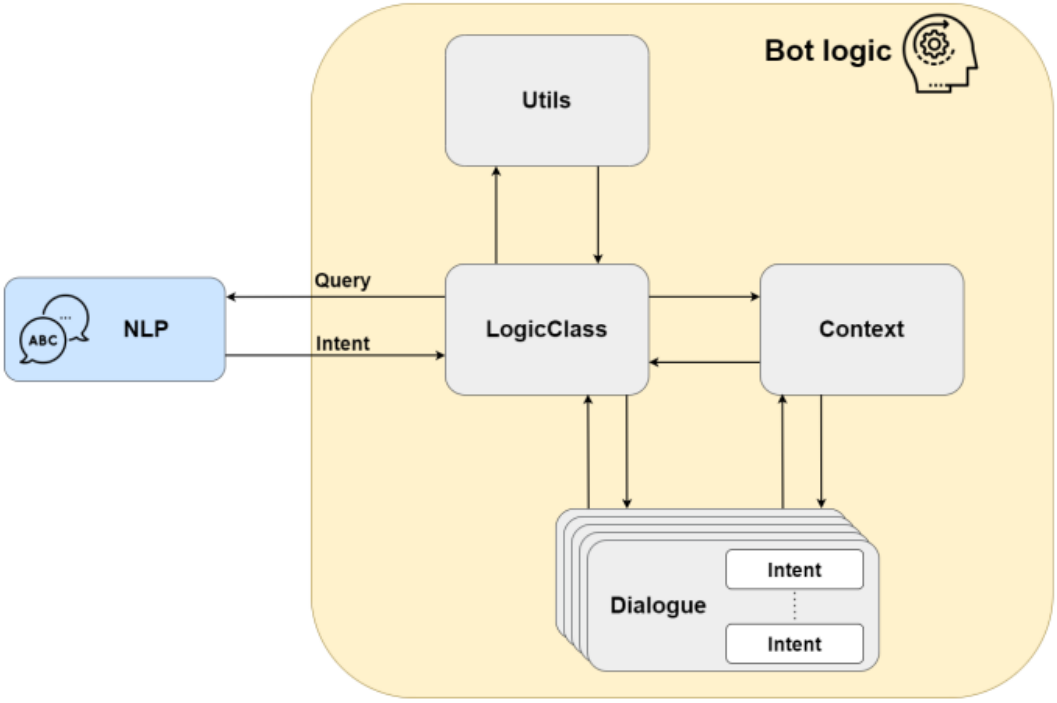

Figura 2.4: Componentes de la Lógica del Bot

[15]

El elemento **LogicClass** permite la definición de los diccionarios que incluyen las respuestas que puede dar el bot. Además, se conecta con la plataforma NPL, verifica

la intención, entidad y contexto devueltos por esta tras procesar la consulta del usuario y procesa el contexto para instanciar el diálogo adecuado encargado de generar la respuesta.

La finalidad del elemento **Contexto** es mantener el contexto de la conversación para cada usuario que interactúa con el bot. Para ello emplea atributos como la intención (intent), el diálogo que responde la pregunta asociada al intent, el paso o momento de la conversación en el que se encuentra y las entidades, como nombres de grados, ciudades, etc.

El elemento **Diálogos** implementa cada una de las clases que corresponden a los diálogos que puede mantener el bot. Estos se emplean para mantener el contexto de la conversación, dependiendo de las preguntas previas que haya realizado un usuario. Para ello, el elemento LogicClass comprueba el contexto de la intención que ha sido reconocido por el sistema NLP. Si el contexto es Dependiente, la conversación se redirige al diálogo, almacenándolo en el contexto del usuario.

El componente **Utilidades** incluye diferentes funciones útiles para el desarrollo del sistema, como obtener la duración de intervalos de tiempo o escritura en ficheros de log.

API

La API es el elemento clave para la conexión con tantos canales como se deseen. Está diseñada para que cada canal funcione de manera independiente, pero teniendo su propia ruta dentro de cada punto de acceso.

Además, se debe definir una ruta para cada plataforma de acuerdo a determinados requisitos y, por tanto, el formato de respuesta se puede adaptar según ciertos requerimientos.

Funciona mediante SSL, que permite HTTPS, estándar de seguridad usado en estos sistemas,

En la API se define cada uno de los manejadores, denominados *endpoints* que se

conectan con cada canal de información.

Se puede emplear cualquier plataforma que ofrezca una API o SDK para desarrolladores, de manera que implementando una determinada lógica para el bot sería posible conectar varios canales siendo más eficiente.

Los canales de conexión al chatbot ofrecidos actualmente son Alexa, Telegram y vía Web.

Repositorio de respuestas

Este componente es el encargado de almacenar los datos necesarios para el funcionamiento del chatbot, siendo los más representativos las preguntas junto a sus respuestas. El sistema está diseñado para que se pueda integrar cualquier tipo de fuente de datos; bases de datos o archivo de almacenamiento, es decir, es posible utilizar una base de datos relacional o no relacional, o archivos de datos como CSV o JSON.

Concretamente, el chatbot de la Universidad de Extremadura emplea actualmente ficheros de almacenamiento cuyo formato es CSV.

Sistemas externos

La lógica del bot requiere de sistemas externos para su óptimo funcionamiento. Estos, a excepción de la plataforma NLP, pretenden complementar las funciones del chatbot.

La existencia de la plataforma de Procesamiento de Lenguaje Natural es imprescindible, ya que sin ella sería imposible reconocer la intención de los usuarios y responder a sus cuestiones. En este caso, se hace uso de la plataforma LUIS de Azure, que ofrece una API donde se pueden enviar solicitudes para predecir la intención y las entidades que contienen las consultas de los usuarios.

Gracias a la implementación del sistema, es posible el entrenamiento de modelos de

aprendizaje automático con algunas bibliotecas o sistemas en la nube. Asimismo, facilita el envío de datos de monitorización a sistemas como AWS Cloudwatch, o incluso permite obtener los datos necesarios para generar respuestas de los usuarios desde los sistemas LMS como Moodle u OpenEDX.

2.2.2. Nueva arquitectura

La nueva arquitectura que se propone pretende mantener las funcionalidades actuales del chatbot optimizando su funcionamiento y agregando nuevos elementos que permitan la gestión de los datos de manera más simple, eficaz y eficiente.

El esquema de esta nueva arquitectura es el que se muestra en la figura 2.5:

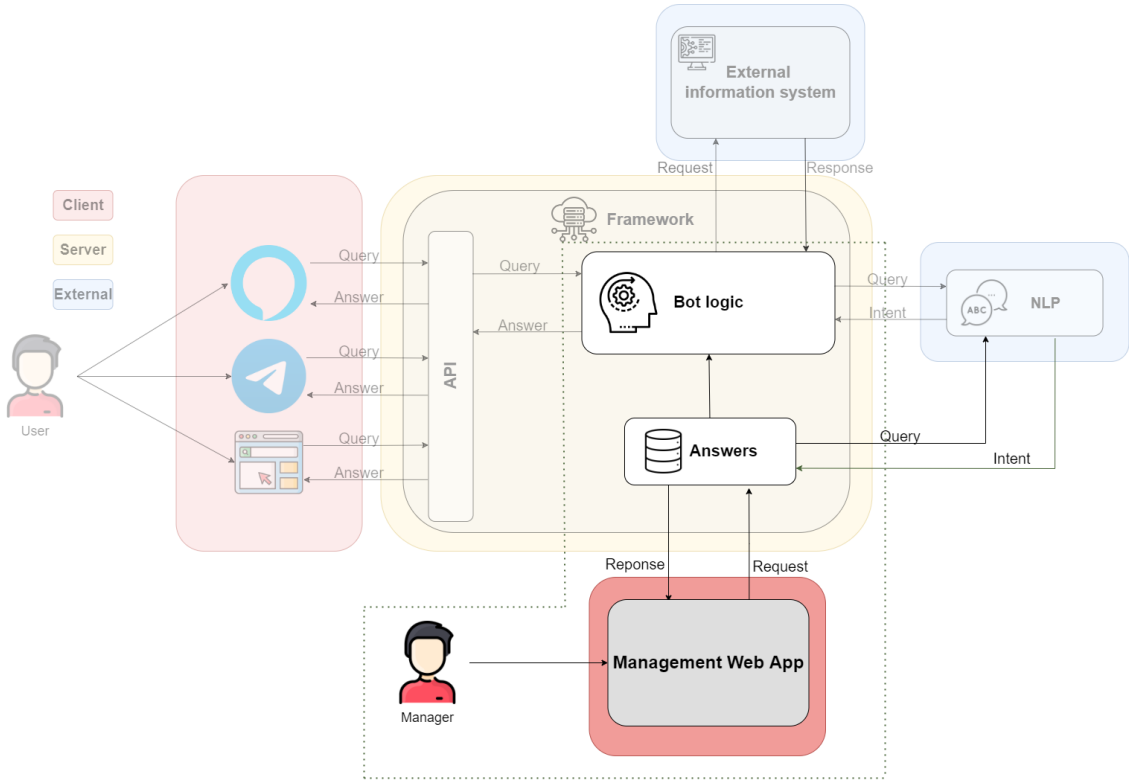

Figura 2.5: Nueva arquitectura general del framework

En la figura anterior (2.5) aparecen resaltados los nuevos componentes que se añaden a la arquitectura original y aquellos que se modifican y amplían. Se puede comprobar que

los cambios se centran en el modo en el que la lógica del bot obtiene los datos, en la forma de almacenarlos y gestionarlos.

Tal como se puede observar, la nueva arquitectura mantiene los componentes que presenta la arquitectura previa, esquematizada en la figura 2.2, añadiendo la interfaz de usuario que facilita la gestión de los datos con los que trabaja el chatbot y ampliando el componente Answers.

La ampliación se centra en la creación de una API que ofrezca una capa de abstracción en el acceso a datos desde la interfaz y desde la Lógica del bot. Esta API facilita la realización de operaciones CRUD sobre los datos, siendo estos los contextos, intents que puede plantear el usuario, sus sinónimos, estos son, distintas formas de realizar una misma pregunta, y las respuestas correspondientes.

Asimismo, se pretenden sustituir parte de los ficheros CSV en los que se almacenan los datos por una base de datos.

Un esquema de los diferentes elementos que forman el componente Answers es el que aparece en la figura 2.6.

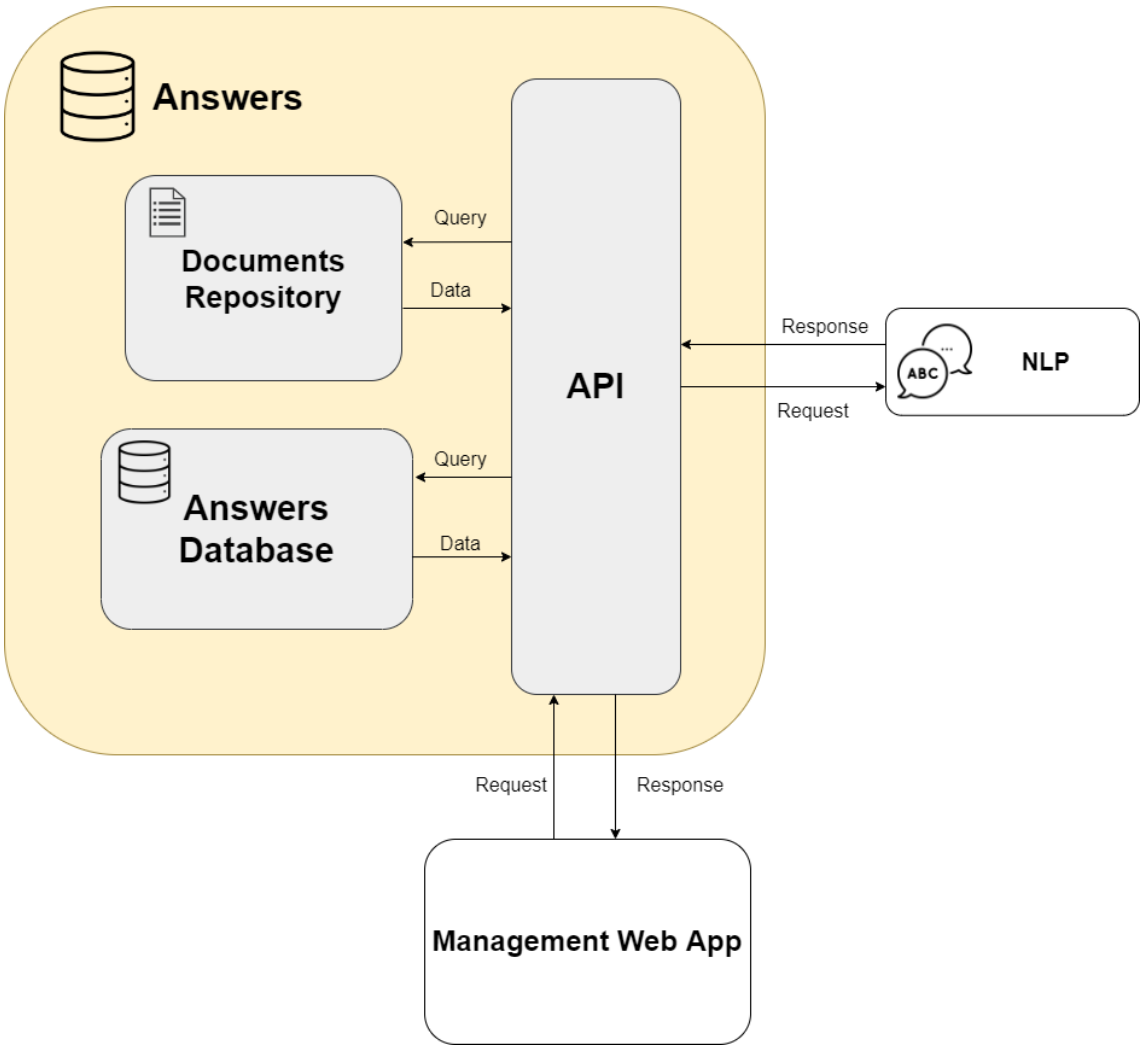

Figura 2.6: Diagrama del componente Answers

2.3. Posibles tecnologías para el desarrollo

Tras analizar la arquitectura existente de la que se parte y a la que se pretende llegar, se deben encontrar las tecnologías necesarias para materializar los cambios descritos, implementando y dando soporte a los componentes que conforman la nueva solución. En los siguientes puntos de este apartado se evalúan las diversas tecnologías alternativas para el desarrollo de la solución, describiéndolas y analizando, para cada una, sus ventajas e inconvenientes.

2.3.1. Tecnologías para el almacenamiento de datos

Los datos necesarios para el funcionamiento del chatbot, que serán gestionados desde la interfaz de la aplicación web, deben ser almacenados. Para ello se requiere de tecnologías que faciliten el guardado de datos.

Como ya se ha mencionado, estos datos se componen de las preguntas reconocidas por el chatbot y los contextos a los que pertenecen. En el sistema del que se parte estos datos se guardan en ficheros CSV, pero la nueva arquitectura requiere del uso de tecnologías más confiables y fácilmente actualizables, como son los sistemas gestores de bases de datos.

En este subapartado se estudian dos grandes grupos de sistemas gestores de bases de datos, relacionales (SQL) y No SQL, valorando qué ofrecen para el desarrollo de la nueva arquitectura. [34]

2.3.2. Bases de datos relacionales

Las bases de datos relacionales SQL han sido las más empleadas desde su invención hasta la actualidad. Estas hacen uso del conocido como lenguaje SQL, siglas de Structured Query Language, esto es, lenguaje de consulta estructurado [35].

El componente fundamental de las bases de datos relacionales son las tablas, formadas por filas y columnas. Las filas contienen los distintos registros de la tabla, mientras que las columnas corresponden a los campos. Las bases de datos que usan el lenguaje SQL se consideran el estándar según el American National Standards Institute (ANSI), en español, autoridad americana oficial de estándares [36].

A continuación, se analizan las características, ventajas e inconvenientes de las bases de datos relacionales más importantes y que fueron candidatas a utilizarse como almacén de datos para el proyecto.

SQL Server

SQL Server constituye un sistema gestor de bases de datos relacional, cuya empresa desarrolladora es Microsoft. El lenguaje que emplea es Transact-SQL, implementación del estándar ANSI del lenguaje SQL. Este se usa con el fin de crear tablas, definir relaciones entre ellas, manipular y recuperar datos. Seguidamente, se describen las ventajas e inconvenientes de esta base de datos [37].

Las principales características de este SGDB se describen a continuación:

Características

- Eficaz a la hora de manejar y obtener datos de Internet.
- Soporta transacciones.
- Ofrece un alto rendimiento.
- Permite el uso de procedimientos almacenados.
- Es estable, segura y fácilmente escalable.
- Consta de entorno gráfico que facilita la ejecución de comandos DDL y DML.
- Permite administrar información de otros servidores de datos y trabajar en modo cliente-servidor.

Las ventajas e inconvenientes derivados de las características anteriores se sintetizan en la tabla siguiente:

Ventajas	Inconvenientes
	La mayoría de las versiones de SQL Server son de pago.
Alto rendimiento.	En la versión gratuita las funciones son muy reducidas e insuficientes para aplicaciones más avanzadas basadas en datos.
Admite procedimientos almacenados.	
Es estable, segura y fácilmente escalable.	Presenta compatibilidad restringida, lo que obliga a la compra e instalación de software de Microsoft.
Permite trabajar en modo cliente-servidor.	
Ofrece alta disponibilidad y recuperación rápida de datos en caso de fallo.	Presenta restricciones de hardware.
	Versiones más actuales de SQL Server requieren de avanzadas tecnologías para funcionar.

Oracle Database

Oracle Database constituye el sistema gestor de base de datos objeto-relacional empleado con mayor frecuencia a nivel mundial. Este motor desarrollado por Oracle Corporation busca mejorar la gestión de bases de datos de gran tamaño aumentando también su nivel de seguridad. A continuación, se describen las ventajas e inconvenientes de esta base de datos. Una base de datos Oracle ofrece las siguientes características para cumplir con los requisitos de una administración de base de datos poderosa [38]. Las principales características de este SGDB se describen a continuación:

Características

- **Escalabilidad y rendimiento:** las funciones ofrecidas hacen que una base de datos de Oracle sea escalable según el uso. En una base de datos multiusuario se requiere

controlar la consistencia y concurrencia de los datos contemplados.

- **Disponibilidad:** las aplicaciones en tiempo real requieren una alta disponibilidad de datos. Los entornos informáticos de alto rendimiento están configurados para proporcionar disponibilidad de datos en todo momento. Los datos están disponibles durante el tiempo de inactividad y fallos planificados o no planificados.
- **Copia de seguridad y recuperación:** su diseño incorpora funciones de recuperación para salvar los datos de casi todo tipo de fallos. En caso de error, la base de datos debe recuperarse en poco tiempo para lograr una alta disponibilidad. Las partes no afectadas de los datos están disponibles mientras se recuperan las afectadas.
- **Seguridad:** Proteger los datos es siempre la máxima prioridad. Oracle proporciona mecanismos para controlar el acceso y el uso de datos. La implementación de acciones de autorización y edición del usuario puede evitar el acceso no autorizado y permitir un acceso seguro a los usuarios.

Las ventajas e inconvenientes que presenta esta base de datos se sintetizan en la siguiente tabla:

Ventajas	Inconvenientes
Es posible ejecutarla desde multitud de plataformas y sistemas operativos. Soporta diversidad de funciones, por ejemplo, el uso de PL/SQL, triggers y procedimientos almacenados. Admite el uso de particiones para ejecutar consultas, realizar informes y análisis de datos.	Presenta una curva de aprendizaje pronunciada. Presenta algunas versiones con fallos, que se han ido corrigiendo mediante actualizaciones. Requiere de pago de licencias de Oracle, además de costes adicionales para obtener documentación técnica. Las funciones sin coste son limitadas.

SQLite

SQLite constituye un sistema para la gestión de bases de datos relacionales compatible con ACID. Su implementación se encuentra contenida en una pequeña biblioteca escrita en código C. SQLite se enmarca en un proyecto de dominio público ideado por D. Richard Hipp. SQLite se orienta al manejo de diversos tipos de datos de forma intuitiva y simple. Funciona sin servidor, es independiente y confiable [39]. Las principales características de este SGDB se describen a continuación [40]:

Características

- Las transacciones son atómicas, consistentes, aisladas y duraderas (ACID) incluso después de fallos del sistema y fallos de energía.
- No se requiere de configuración ni administración.
- Gestión completa mediante lenguaje SQL con capacidades avanzadas como índices parciales, índices en expresiones, JSON, expresiones de tablas comunes y funciones de ventana.

- Una base de datos completa se almacena en un único archivo de disco multiplataforma, ideal para usar como formato de archivo de aplicación.
- Admite bases de datos del tamaño de un terabyte, cadenas y blobs del tamaño de un gigabyte.
- Consta de interfaz simple y fácil de usar.
- Rápido: en algunos casos, SQLite es más rápido que la E/S directa del sistema de archivos
- Disponible como un solo archivo de código fuente ANSI-C que es fácil de compilar y, por lo tanto, fácil de agregar a un proyecto más grande.
- No requiere de dependencias externas.
- Es multiplataforma y portable a otros sistemas.
- Las fuentes son de dominio público y pueden ser usadas para cualquier propósito.
- Ofrece un cliente con interfaz de línea de comandos (CLI) independiente que se puede usar para administrar bases de datos SQLite.

Las ventajas e inconvenientes derivados de las características anteriores se sintetizan en la siguiente tabla:

Ventajas	**Inconvenientes**
No requiere de instalaciones complejas. Esto conlleva una integración directa con la aplicación.	
Muy adecuada para desarrollo básico de aplicaciones y ejecución de pruebas.	No es recomendable para grandes volúmenes datos.
Constituye un sistema de bases de datos estable y multiplataforma.	Restringido respecto a las extensiones de archivo aceptadas.
Alberga la base de datos en un único archivo fácil de exportar, lo que hace de este un sistema ligero en comparación con otros.	No es fácilmente escalable.
Código y uso de dominio público y sin coste.	Carece de sistema de administración de usuarios
Recomendable cuando es necesario leer y escribir directamente desde disco.	

MySQL

MySQL constituye uno de los sistemas gestores de bases de datos relacionales más extendidos actualmente. Consta de una doble licencia, ya que en su origen fue creada por MySQLAB bajo una licencia de código abierto, pero fue adquirida por la compañía Oracle, que gestiona su versión comercial. De sus principales características se debe destacar que presenta una arquitectura Cliente-Servidor y su forma de procesar las tablas, acción que no efectúa directamente sino mediante procedimientos almacenados [41]. Seguida-

mente, se describen las principales características de este SGDB:

Características

- MySQL es de código abierto, lo que significa que cualquiera puede descargar, usar y modificar este software. Es de uso gratuito y fácil de comprender.
- Es rápido y confiable dado que MySQL almacena datos de manera eficiente en memoria, asegurando que los datos sean consistentes y no redundantes. Por lo tanto, el acceso y la manipulación de datos rápida.
- El servidor MySQL fue desarrollado para trabajar con grandes bases de datos y está preparado para ser escalable.
- Contiene diversos tipos de datos y admite diferentes conjuntos de caracteres.
- Proporciona una interfaz segura ya que tiene un sistema de contraseñas flexible y que garantiza la verificación en función del host antes de acceder a la base de datos. La contraseña se cifra mientras se conecta al servidor.
- Ofrece soporte para grandes bases de datos.
- El servidor MySQL incluye programas de utilidad, como programas de línea de comandos y programas gráficos.

Las ventajas e inconvenientes derivados de las características anteriores se sintetizan en la siguiente tabla:

Ventajas	Inconvenientes
Existencia de versión de código abierto sin coste.	
Es fácilmente escalable y recomendable para albergar grandes volúmenes de datos.	Presenta una mayor complejidad técnica en su instalación y configuración.
Permite ejecutar operaciones de manera rápida, lo que provoca que su rendimiento sea alto y garantiza la eficiencia cuando exista tráfico denso de datos.	La sintaxis para el manejo de los datos es levemente distinta al SQL clásico, lo que implica una curva de aprendizaje más pronunciada.
Presenta una gestión de usuarios eficaz proporcionando múltiples controles de acceso.	La mayoría de las funcionalidades avanzadas no son accesibles desde la versión libre.
Las características de este SGDB garantizan su seguridad y confiabilidad.	Presenta deficiencias al trabajar con bases de datos demasiado grandes.
Se considera óptimo para el desarrollo de aplicaciones web.	

2.3.3. Bases de datos No SQL

La segunda alternativa que existe para el almacenamiento de datos son las bases de datos No SQL, término que se refiere a la denominación que reciben este tipo de bases de datos en inglés, Not Only SQL.

Las bases de datos No SQL constituyen sistemas de almacenamiento que, por lo general, no emplean las estructuras de datos tabulares características de los sistemas relacionales para el guardado de la información. Tampoco siguen el esquema entidad-relación clásico. En su lugar, ofrecen modelos de datos flexibles y adaptables a datos específicos, requisitos y necesidades de las aplicaciones actuales. Algunos de estos modelos son clave-valor, orientación a columnas o grafos [42].

Por tanto, este tipo de bases de datos buscan resolver los problemas que originan las bases de datos relacionales en ciertas situaciones, sobre todo en relación con cuestiones de escalabilidad y rendimiento cuando se da un número alto de usuarios conectados de

manera concurrente y se deben ejecutar numerosas consultas diarias [43].

Existen multitud de bases de datos No SQL. A continuación, se describen aquellas consideradas más adecuadas para almacenar los datos con los que trabaja el bot y la aplicación. Estas implementan el modelo Clave-Valor.

MongoDB

MongoDB consiste en un sistema de base de datos No SQL multiplataforma dirigido a documentos y de código abierto. El origen de su nombre proviene del término inglés humongous, cuyo significado es "enorme", haciendo referencia al gran volumen de datos que puede albergar [44].

El sistema de base de datos MongoDB fue ideado por la empresa MongoDB Inc. Está escrito en C++, de modo que es bastante rápido a la hora de ejecutar tareas. Se creó bajo una licencia GNU AGPL 3.0, una licencia de software libre. Sus características principales se describen a continuación:

Características

- MongoDB se clasifica en el modelo de base de datos clave-valor, lo que quiere decir que los datos se guardan en forma de elementos con una clave o nombre de la columna y su valor.
- En MongoDB cada conjunto de datos o registro se denomina documento. Estos registros se pueden reunir en colecciones. Como su esquema es libre, cada documento se puede organizar en un esquema de datos diferente, sin obligatoriedad de repetir los atributos de un registro a otro, lo que no resultaba posible en las bases de datos relacionales.
- Este sistema de bases de datos posibilita la creación de índices de manera sencilla para ciertos atributos de los documentos, de forma que MongoDB mantiene una estructura interna eficiente para el acceso a datos.

- En lugar de almacenar los datos en tablas, MongoDB guarda los datos dentro de estructuras de datos BSON, cuyo esquema es dinámico, provocando que la integración de los datos en algunas aplicaciones resulte más fácil y rápida.
- BSON o Binary JSON consiste en una representación binaria de JSON que facilita búsquedas rápidas de datos. Para que el manejo de los datos resulte más sencillo, no se visualiza el formato real en el que se almacena la información, sino que se trabaja siempre sobre JSON.
- MongoDB constituye una base de datos distribuida, esto es, tiene la capacidad de ser ejecutada simultáneamente en múltiples servidores, por lo que soporta balanceo de carga y replicación de datos, conservando así copias en distintos nodos o servidores.
- Mantiene una alta disponibilidad y tolerancia a particiones, es decir, el sistema sigue funcionando y sobrevive a fallos en nodos parciales de la base de datos.

Las ventajas e inconvenientes derivados de las características anteriores se sintetizan en la siguiente tabla [45]:

Ventajas	Inconvenientes
No necesita que los datos sigan una estructura fija. Utiliza el formato JSON, sencillo de comprender. Admite replicación de datos en distintos servidores mediante la arquitectura maestro-esclavo. Diseñada para desarrollo de aplicaciones web actuales. Tolera grandes volúmenes de datos y es simple de escalar horizontalmente. Ofrece soporte para transacciones, manteniendo la integridad de los datos. No tiene coste.	No admite el uso de triggers ni JOINS. En algunas versiones desfasadas de MongoDB no existe manera de comprobar que los campos de los documentos que se insertan cumplan con ciertos requisitos en sus campos.

Cassandra

Apache Cassandra constituye un sistema de gestión de bases de datos (SGDB) No SQL de código abierto escrito en Java. Su propiedad más notable es que está destinada a almacenamiento de bases de datos estructuradas de gran tamaño. Cassandra no se encuentra unida a un único servidor, sino que presenta una arquitectura distribuida, lo que facilita la gestión de grandes volúmenes de datos. Sigue el modelo de almacenamiento clave-valor y está orientada al almacenamiento en columnas [46]. A continuación se describen sus principales características:

Características

- Cassandra se basa en un sistema distribuido donde todos los nodos que forman parte de la base de datos poseen las mismas funciones. Concretamente, presenta arquitectura P2P (Peer-to-Peer), lo que significa que todos los nodos pueden comunicarse con el resto de los nodos sin necesidad de intermediarios y cualquiera de ellos puede tomar el rol de coordinador.

- El protocolo empleado por los nodos de Cassandra para comunicarse se denomina Gossip.
- La estructura de Cassandra es clave garantizar disponibilidad continua, haciendo que el sistema sea tolerante a fallos.
- Al funcionar cada nodo de forma independiente, si uno de ellos cae los servicios no se interrumpen gracias al resto de nodos, que cubren el fallo. Esto contrasta con la arquitectura maestro-esclavo en la que la caída del maestro provoca también la baja del servicio.
- Cassandra pretende ser una base de datos linealmente escalable, es decir, conseguir que el número de nodos sea linealmente dependiente del número de operaciones que se puedan procesar.

Las ventajas e inconvenientes derivados de las características anteriores se sintetizan en la siguiente tabla [45]:

Ventajas	**Inconvenientes**
Presenta una gran tolerancia a fallos. Los datos se encuentran replicados entre diferentes nodos de un clúster. Ofrece nivel de consistencia configurable para lecturas y escrituras. Consta de gran cantidad de recursos disponibles. Soporta uso de triggers. Fácil de escalar horizontalmente. No tiene coste.	No presenta soporte para transacciones. Requiere de máquina virtual de Java para ser ejecutada. No permite uso de JOINS en las consultas. Presenta dificultades en el momento de agregar nuevos nodos al cúster.

Redis

Redis constituye una base de datos que sigue el esquema clave-valor. Un punto importante es que trabaja en memoria a una velocidad alta. Creada en 2009 con el objetivo de aumentar la escalabilidad de la empresa en la que trabaja su autor, Salvatore Sanfilippo. Su nombre significa Remote Directory Server [47].

Redis destaca por ser un sistema que puede ejercer simultáneamente de almacén de datos y caché. Esto se debe a que, dado su diseño, los datos se leen y modifican desde la memoria principal del ordenador en el que se ejecuta. Estos se escriben ocasionalmente en la memoria persistente utilizando un formato pensado para ser reconstruidos en memoria principal y que no es apto para el acceso aleatorio [48].

Redis destaca en situaciones en las que el flujo de los datos es elevado y se necesita una baja latencia. Se usa comúnmente en sistemas en tiempo real que proporcionan muchos datos a velocidades altas, caching de operaciones complejas y costosas, caching de estado de sesiones o almacenamiento de consultas frecuentes a otras bases de datos.

Las ventajas e inconvenientes de esta base de datos se sintetizan en la tabla siguiente [45]:

Ventajas	**Inconvenientes**
El almacenamiento en memoria conlleva una velocidad muy elevada en comparación con otros sistemas de su tipo.	
La persistencia de datos en disco permite la recuperación rápida ante fallos.	Redis requiere de una enorme capacidad en la memoria principal, ya que alberga los datos en memoria.
Es fácilmente configurable.	No es recomendable para conjuntos de datos complejos que requieran de búsquedas elaboradas.
Presenta una alta disponibilidad.	
Su curva de aprendizaje poco pronunciada.	
Admite variedad de tipos de datos.	

2.3.4. Tecnologías para la implementación del BackEnd

El siguiente elemento que requiere el desarrollo de la arquitectura es aquel que permita conectar y comunicar el resto de los elementos que la componen. Este elemento clave consiste en una API REST, esto es, una arquitectura para el desarrollo web apoyada en el protocolo HTTP cuya función principal es recibir peticiones de un recurso concreto y devolver una respuesta en un formato de texto organizado. El objetivo de una API REST consiste en ofrecer acceso a información sobre un sistema de forma controlada definiendo restricciones, permitiendo así a terceros la posibilidad de desarrollar aplicaciones para que interactúen con el sistema.

Una API debe ser creada teniendo en cuenta los requisitos y funciones que debe brindar el sistema. Esta sección tiene como objetivo analizar diversas tecnologías que facilitan el desarrollo de APIs REST [49].

Express

La primera tecnología estudiada es Express, que se puede definir como un marco de desarrollo de aplicaciones web sencillo y flexible para NodeJS. Este framework, basado en Javascript, ha servido de inspiración para el desarrollo de muchos otros marcos de trabajo existentes en mercado [50].

Sus características minimalistas unidas a su velocidad de respuesta convierten a Express en un fuerte candidato para el desarrollo de API Rest.

NodeJS contribuye de forma significativa a su rapidez debido a su modelo de entrada y salida asíncronas que minimiza el tiempo de respuesta al continuar ejecutándose sin esperar por el retorno de operaciones externas.

Si se desea implementar un sistema completo con esta tecnología, su concepto simplista puede resultar un inconveniente; sin embargo, se convierte en una opción ideal en el desarrollo de APIs que pretenden proporcionar acceso a sistemas sencillos.

Para crear los puntos de entrada de la API (*endpoints*) no es necesario hacer uso de funcionalidades ni librerías externas.

Las ventajas e inconvenientes de este framework se sintetizan en la siguiente tabla:

Ventajas	**Inconvenientes**
El gestor de paquetes de NodeJS o NPM facilita el acceso a una variedad grande de paquetes reutilizables que favorecen el desarrollo e implementación de las APIs.	Mantener una estructura y estilo limpio en el desarrollo puede resultar complejo.
Su uso es simple al basarse en código JavaScript.	La creación de la API se complica si no se comprenden las funciones de devolución de llamada y los bloqueos.
Ofrece una completa documentación y se pueden encontrar ejemplos de uso en la red.	

Flask

Flask consiste en un micro framework centrado en el rendimiento y de diseño minimalista. Flask cuenta con la versatilidad que ofrece el lenguaje Python, permitiendo crear APIs y aplicaciones web rápidamente en un reducido número de líneas de código [51].

Se define como micro framework debido a que, una vez instalado, ofrece todo lo necesario para crear una web funcional, permitiendo la instalación de extensiones, plugins y complementos si es necesario. Se puede combinar con diferentes herramientas para favorecer su funcionalidad.

De sus características se debe destacar que implementa la arquitectura Modelo – Vista – Controlador y consta de un servidor de desarrollo, un depurador y soporte para pruebas unitarias. Además, se adapta con facilidad al tipo de sintaxis y a la plataforma con la que

se trabaje.

Las ventajas e inconvenientes de este framework se sintetizan en la siguiente tabla:

Ventajas	**Inconvenientes**
Es fácilmente adaptable a la plataforma y sintaxis empleada. Posibilita la instalación de extensiones y complementos si se necesitan en función de la modalidad de proyecto que se desee desarrollar. Brinda variedad de plugins desarrollados por la comunidad, siendo posible combinarlos con diversas herramientas externas para mejorar su rendimiento.	La implementación pruebas unitarias puede resultar compleja. Las migraciones pueden causar problemas. El sistema de autenticación para usuarios es básico y poco seguro.

FastApi

FastApi se trata de un framework definido por sus creadores como una herramienta de alto rendimiento, curva de aprendizaje poco pronunciada, sencilla de programar y preparada para producción. Emplea el lenguaje de programación Python [52].

Este marco de desarrollo ofrece un sistema intuitivo con soporte en los IDE y editores más simples. Asimismo, su diseño está basado en los estándares abiertos para API, de manera que reduce la duplicación del código y facilita el ahorro de tiempo en lectura de documentación.

Constituye un framework muy rápido, con tiempos comparables a NodeJS, y es capaz de genera la documentación interactiva de la API de manera automática.

Las ventajas e inconvenientes de este framework se sintetizan en la siguiente tabla:

Ventajas	**Inconvenientes**
Ejecuta las operaciones de manera muy veloz.	
Presenta soporte de código asíncrono.	El archivo principal del programa puede saturarse con facilidad.
Tiempo de desarrollo reducido.	
Permite la implementación de pruebas de manera fácil.	Existe más de un caso de inyección dependiente, no soporta instancias singleton pero sí una por solicitud.
El manejo de excepciones es eficaz.	
Cuenta con una excelente documentación.	La validación de solicitudes puede acarrear problemas.
Ofrece facilidad en el despliegue.	

Django

El marco de desarrollo Django Rest Framework, basado en Python, proporciona un entorno de desarrollo de software libre y código abierto que facilita la vinculación entre la API y las entidades que forman el modelo de datos. Está dirigido al desarrollo web escalable y de alta calidad [53].

Django brinda un conjunto de librerías con una amplia gama de funciones y métodos que permiten el desarrollo de APIs REST estables, haciendo posible la definición, gestión y control de los recursos.

Desarrollar una API empleando esta tecnología supone la generación automáticamente de una web desde donde es posible hacer peticiones a los recursos definidos y consultar

los resultados de manera visual.

Respeta el patrón de diseño MVC (Modelo–Vista–Controlador) y cuenta con un panel de administración que facilita la gestión de bases de datos.

Las ventajas e inconvenientes de este framework se sintetizan en la siguiente tabla:

Ventajas	**Inconvenientes**
Proporciona una serie de librerías que facilitan el desarrollo de APIs REST estables. Las librerías cuentan con diversidad de métodos y funciones parala gestión de recursos. Ofrece un panel visual para la administración de bases de datos.	Su documentación es demasiado densa y puede resultar confusa. Presenta cierto grado de dificultad a la hora poner en producción la API.

2.3.5. Tecnologías para la implementación del FrontEnd

Una vez estudiadas las herramientas que permiten almacenar datos, conectar el resto de los componentes y crear servicios sobre ellos, es el momento de analizar las alternativas que facilitan la creación de una interfaz web que permita la interacción con el sistema de forma simple e intuitiva. Esta interfaz debe conseguir que usuarios y administradores realicen tareas sobre el sistema sin que sea necesario manipular ficheros de configuración o bases de datos de forma directa [54].

La primera cuestión planteada es la de cuál es la mejor plataforma para la implementación de esta aplicación web: Android, IOS, o web.

Teniendo en cuenta la naturaleza y finalidad del sistema que se va a desarrollar, gestión de los datos con los que trabaja un chatbot, así como sus potenciales usuarios, personal de la administración, se decide que la mejor opción es una página web a la que se pueda

acceder desde diferentes dispositivos, por lo que debe ser responsive.

A continuación, se analizan diferentes tecnologías que facilitan la creación de aplicaciones web dinámicas.

Angular

Angular constituye un framework para el desarrollo aplicaciones web que emplea el lenguaje en TypeScript. De código abierto y sostenido por Google, se emplea con el fin de crear y mantener aplicaciones web SPA, esto es, de una sola página [55].

Su principal objetivo consiste en simplificar el desarrollo y pruebas de las aplicaciones basadas en web, extendiéndolas a su vez con arquitectura Modeo-Vista-Controlador [56].

Asimismo, proporciona un mapeo de datos sencillo, ya sean estos obtenidos desde una API o de la lógica interna de la aplicación, para mostrarlos en la vista o presentación de la aplicación. Para conseguir esto, hace uso de etiquetas específicas sobre HTML.

Las aplicaciones basadas en Angular se compilan a partir de componentes. Un componente combina plantillas HTML con objetos especiales para controlar una parte de la página.

El trabajo con componentes y plantillas en vez de con elementos DOM, provoca que las aplicaciones desarrolladas en Angular funcionen en un nivel de abstracción más elevado y empleando menos código en general que las aplicaciones escritas solo en JavaScript.

Angular también impone, por convención, una estructura de carpetas común para organizar los archivos de script en el lado del cliente: scripts de módulos y componentes se ubican en la carpeta de aplicación y archivos de compilación y pruebas a nivel superior [57].

Sus ventajas y desventajas quedan resumidas en la siguiente tabla:

Ventajas	Inconvenientes
Facilita el uso de librerías npm junto a librerías propias. El mapeo de los datos hacia la vista es muy sencillo. El lenguaje que utiliza es TypeScript, un lenguaje que consiste en JavaScript enriquecido. Cuenta con un servidor de pruebas. El concepto de componente y servicio conlleva que el código sea mantenible de manera fácil, haciendo la aplicación modular.	Requiere de la instalación previa de Node.js, ya que funciona sobre este. Su instalación se realiza mediante npm, lo que podría conllevar problemas al pasar de una versión a otra. Presenta sintaxis complicada heredada de la primera versión de Angular.

React JS

React consiste en una biblioteca Javascript de código abierto cuyo objetivo es la creación de interfaces de usuario mediante el desarrollo de aplicaciones en una sola página (SPAs). No ofrece una implementación completa del patrón Modelo-Vista-Controlador, solo se ocupa de las vistas. No constituye un framework de trabajo, sino simplemente una biblioteca por lo que para compilar una SPA es necesario recurrir a bibliotecas adicionales [58].

Una ventaja importante de React es la existencia de un DOM virtual, que es capaz de optimizar las partes del DOM real que deben actualizarse, mejorando así el rendimiento, e incrementar la capacidad de prueba, ya que no requiere de un explorador para probar sus interacciones con el DOM virtual.

El modo de funcionamiento de React con HTML es bastante particular: no existe separación estricta entre el código y las etiquetas de marcado, ya que incrusta código HTML directamente en el de JavaScript. A esta sintaxis de tipo HTML que puede ser compilada desde JavaScript se la denomina JSX [57].

La siguiente tabla presenta sus ventajas e inconvenientes:

Ventajas	**Inconvenientes**
Los componentes se identifican con facilidad y son sencillos de diseñar e implementar. Permite la navegación entre componentes. Permite que los componentes sean procesados en el servidor en vez de en el cliente. Curva de aprendizaje poco pronunciada, presenta sintaxis relativamente sencilla. Es flexible y ligero. Sin coste, de código abierto y con frecuentes actualizaciones.	Se requiere del uso de extensiones, de lo contrario trabaja únicamente con la Vista. Carece de estándar de desarrollo, lo que obliga a los desarrolladores a tomar más decisiones. Su dominio en profundidad necesita de tiempo.

Vue JS

Vue.js constituye un marco de desarrollo web de código abierto basado en JavaScript. Su finalidad principal es la de facilitar la creación de interfaces de usuario y aplicaciones de una sola página (SPAs). Fue ideado por Evan You, tras utilizar Angular.js mientras trabajaba para Google [59].

Cuenta con una arquitectura progresiva, enfocada en la representación declarativa y basada en componentes. En caso de requerir funciones avanzadas para el desarrollo de aplicaciones complejas, tales como como administración de estado, enrutamiento y herra-

mientas de construcción, existen bibliotecas y paquetes compatibles con este framework.

Una característica destacable de VueJS es su excelente capacidad de integración y extender su uso de SPAs a interfaces web complejas. A esto se debe añadir su gran escalado, facilitando la construcción de grandes plantillas reutilizables en poco tiempo gracias a su estructura sencilla [57].

Sus ventajas y desventajas se muestran en la tabla siguiente:

Ventajas	**Inconvenientes**
Los componentes se distribuyen en forma de árbol jerárquico hasta completar la aplicación, de manera incremental. Presenta funcionalidades intuitivas, actuales y de uso fácil. Su sistema de componentes es reactivo, ya que establece comunicaciones padre-hijo a través de eventos asíncronos	No cuenta con documentación detallada. No se apoya en una comunidad tan extensa como la de otras alternativas. Su excesiva flexibilidad debida a la utilización de librerías externas, puede llegar a suponer un problema en proyectos de gran tamaño y si es desarrollado por diferentes miembros

Blazor WebAssembly

Blazor constituye un proyecto desarrollado por Microsoft ideado para facilitar la creación de aplicaciones de una sola página (SPAs) utilizando únicamente como lenguajes de programación C hashtag y Razor Pages, eliminando la necesidad de hacer uso de Javascript y frameworks derivados [60].

Concretamente, Blazor WebAssembly facilita la construcción del DOM en el lado del cliente, permitiendo también efectuar operaciones en el lado del servidor realizando llamadas a APIs con el objetivo de solicitar datos.

Este famework, conocido también como Wasm, consiste en un estándar que permite la ejecución de código binario en navegador web ofreciendo un rendimiento, en principio, mayor que Javascript.

El servidor web es el encargado de enviar al navegador las librerías directamente compiladas (dlls) cuando este se realice una petición desde la aplicación cliente. Gracias a esta tecnología, el navegador sabe interpretar aquello que debe ejecutarse. En consecuencia, el servidor web se desocupa de procesar lógica.

2.3.6. Justificación de las Tecnologías Elegidas

Una vez se han analizado y valorado todas las posibles tecnologías que se pueden emplear para implementar la nueva arquitectura del sistema, se procede a la elección de aquellas consideradas más adecuadas. Están han sido las siguientes:

- Para el almacenamiento de datos: Base de datos relacional, SQLite.
- Para la implementación de la API (BackEnd): Flask.
- Para el desarrollo de la aplicación web (FrontEnd): ReactJS.

Almacenamiento de datos

Teniendo en cuenta la naturaleza de los datos que se necesitan almacenar y el modo en el que estos van a ser gestionados, se ha llegado a la conclusión de que la opción que mejor se ajusta es una base de datos relacional, concretamente, se ha seleccionado SQLite.

Esta elección se debe principalmente a que las consultas y operaciones que se van a realizar sobre los datos son operaciones CRUD sencillas y mediante las bases de datos de tipo relacional se garantiza una integridad mayor, de manera que los datos que se visualicen en la interfaz de usuario de la aplicación y las respuestas que proporcione el chatbot coincidan y estén siempre actualizadas.

Asimismo, la simplicidad en el uso de SQLite frente a otras bases de datos y su orientación a aplicaciones sencillas sin tráfico excesivo de datos avalan esta elección.

Tecnología para la creación de la API REST

La alternativa seleccionada con el fin de crear y desplegar la API REST es Flask. La elección se debe a que este framework se amolda a la perfección con la arquitectura que se pretende alcanzar. Además, facilita el desarrollo de manera ágil y eficiente, contando también con todas las ventajas que ofrece el lenguaje de programación Python.

Tecnología para la creación de la Interfaz Web

La tecnología elegida para el desarrollo de la aplicación web es React JS. Los motivos principales que respaldan esta decisión son su flexibilidad, rapidez y organización del código, provocando que se se haya convertido en la opción favorita de aplicaciones a nivel mundial.

Asimismo, al ser una librería basada en Javascript, cuenta con el apoyo de una gran comunidad que pone a disposición del usuario una amplia gama de bibliotecas externas para añadir funcionalidades a las aplicaciones. Se debe destacar la existencia de una extensa documentación y numerosos tutoriales que facilitan el aprendizaje sobre esta herramienta.

Una característica clave que motiva al uso de React es el isomorfismo, gracias al que es posible construir páginas HTML con contenido renderizado previamente, hecho que facilita el desarrollo de la interfaz.

Capítulo 3

Propuesta de la solución

Una vez definidos los objetivos y la problemática que enfrenta el presente trabajo, tras haber realizado el estudio de los conceptos básicos sobre el tema y las distintas tecnologías alternativas en el Estado del Arte, el siguiente paso consiste en detallar la propuesta de la solución teniendo en cuenta todos los conocimientos alcanzados.

3.1. Descripción general de la solución

Tal como ya se ha mencionado, la idea que propone el presente proyecto es la implementación y desarrollo de una API Restful y una plataforma web para la gestión de los datos con los que trabaja Lyx, asistente virtual de la Universidad de Extremadura.

La plataforma web, a través de una sencilla interfaz, pretende facilitar el manejo de los datos por parte del personal del Servicio de Información de la UEx, de manera que sea posible añadir y modificar preguntas y respuestas de manera sencilla, siendo conscientes de que, en la mayor parte de los casos es necesaria una actualización periódica.

Por otro lado, con la implementación de la API se pretende alcanzar un mayor nivel de abstracción en el funcionamiento de la aplicación, pudiendo ser utilizada tanto por el

chatbot en sí como por la plataforma web.

Asimismo, para facilitar la actualización de los datos y su integridad, se migrarán desde los documentos de almacenamiento actual, archivos JSON, a una base de datos. Gracias a la creación de esta nueva base de datos, se podrá modificar la gestión de los diálogos, simplificándola para lograr una implementación más eficiente.

3.2. Arquitectura de la solución

Tal como se ha descrito en el estudio del Estado del Arte, el presente Trabajo de Fin de Grado pretende ampliar el framework para el desarrollo y despliegue de chatbots multiplataforma que emplea el asistente virtual de la Universidad de Extremadura. Este marco fue desarrollado por Javier Romero Álvarez en su Trabajo de Fin de Máster y las ampliaciones que se proponen en este TFG forman parte de los trabajos futuros citados en él.

3.2.1. Arquitectura inicial. Mejoras propuestas

Como punto de partida, se analizan los diferentes elementos que componen la arquitectura inicial del framework, esta es la creada por Javier Romero en su trabajo [15]. A través de este primer análisis, llevado a cabo en el estudio del Estado del Arte, se descubre el modo de funcionamiento de cada componente así como las relaciones e intercambio de información entre ellos.

Los componentes que conforman la arquitectura inicial junto con sus conexiones se representan en el siguiente diagrama:

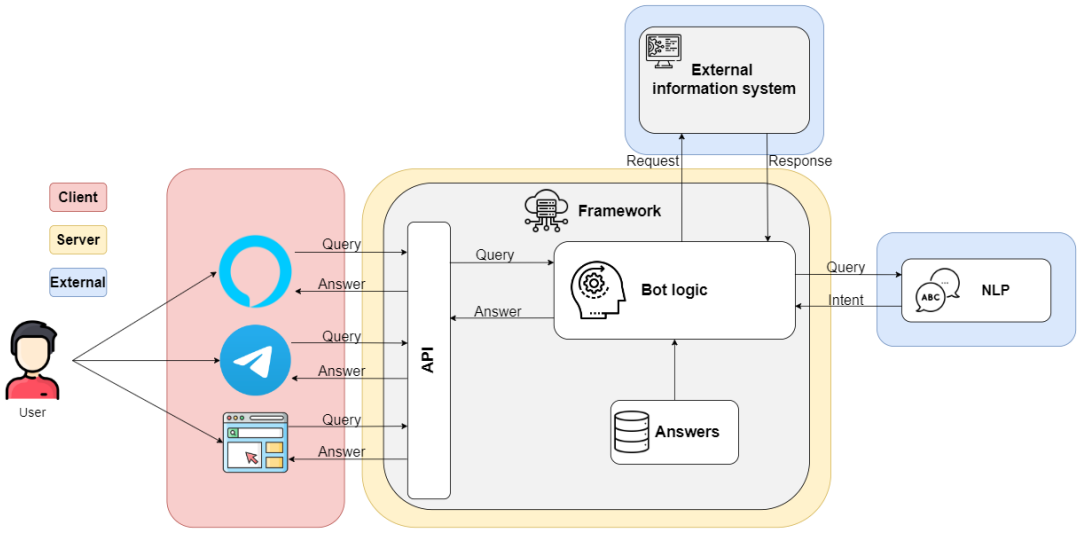

Figura 3.1: Arquitectura inicial del framework

A modo de recordatorio, el esquema que aparece a continuación incluye para cada componente la tecnología empleada:

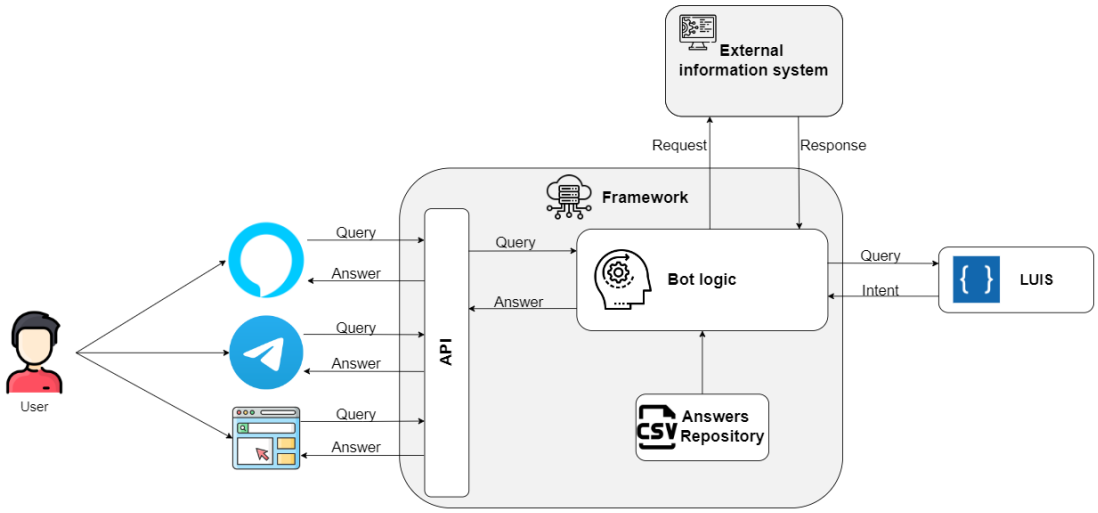

Figura 3.2: Arquitectura inicial del framework con tecnologías empleadas

Puntos débiles de esta arquitectura inicial, que las modificaciones propuestas en este TFG pretenden eliminar, son los siguientes:

- Los datos se almacenan en archivos CSV, que se deben modificar directamente si cambian respuestas de los intents. Esto puede provocar un riesgo en la integridad de los datos.
- El personal administrativo de la Universidad de Extremadura no tiene posibilidad de modificar de forma autónoma los datos, sino que deben comunicarse con los administradores del framework para que ellos efectúen los cambios necesarios, hecho que ralentiza las actualizaciones.
- El acceso a datos y a la plataforma NPL se realiza directamente desde la lógica del bot, no existe una capa de abstracción para la gestión de datos, lo que aumenta el acoplamiento entre componentes.
- El código perteneciente a la lógica del bot que gestiona determinadas preguntas dependientes de contextos es complejo y extenso, lo que provoca dificultad de comprensión y localización de errores en caso de fallos en las respuestas que da el asistente virtual.
- El único medio que permite entrenar la aplicación es la interfaz de LUIS de Azure, por lo que esta tarea solo la pueden realizar los administradores del chatbot.

El desarrollo de la API REST y la plataforma web buscan mejorar los puntos anteriores. Concretamente, las mejoras planteadas son las citadas a continuación:

- Llevar a cabo una migración parcial de los datos desde los archivos CSV a una base de datos SQL, para facilitar así el acceso y gestión de estos mejorando también su integridad.
- Desarrollo de una API REST que juegue el papel de capa de abstracción para el acceso a los datos y a la plataforma NPL, favoreciendo así una posible integración con diferentes bases de datos y plataformas de procesamiento de lenguaje natural. Asimismo, disminuye el acoplamiento entre componentes al reducir el manejo de datos desde la lógica del bot.

- Implementación de una aplicación web intuitiva y sencilla que permita al personal administrativo de la universidad editar los datos cuando sea necesario introducir cambios. La interfaz debe ofrecer además una función que permita entrenar la aplicación tras efectuar modificaciones.

- Gracias al desarrollo de la API y al uso de una base de datos, se simplifica el código que gestiona los diálogos dependientes de contexto. Esto se logra añadiendo nuevos campos en la base de datos

Cada uno de los aspectos anteriores se describe con su debido detalle en apartados posteriores.

3.2.2. Nueva arquitectura propuesta

La nueva arquitectura propuesta debe ampliar la ya existente añadiendo la aplicación web, la API REST y la base de datos. Un primer esquema de la arquitectura es el mostrado en la figura 3.3, en el que los nuevos componentes añadidos poseen fondo amarillo claro.

Tal como se puede observar en la figura, este primer esquema divide la arquitectura en dos capas, la denominada front-end o capa de presentación y el back-end o capa de acceso y manipulación de datos.

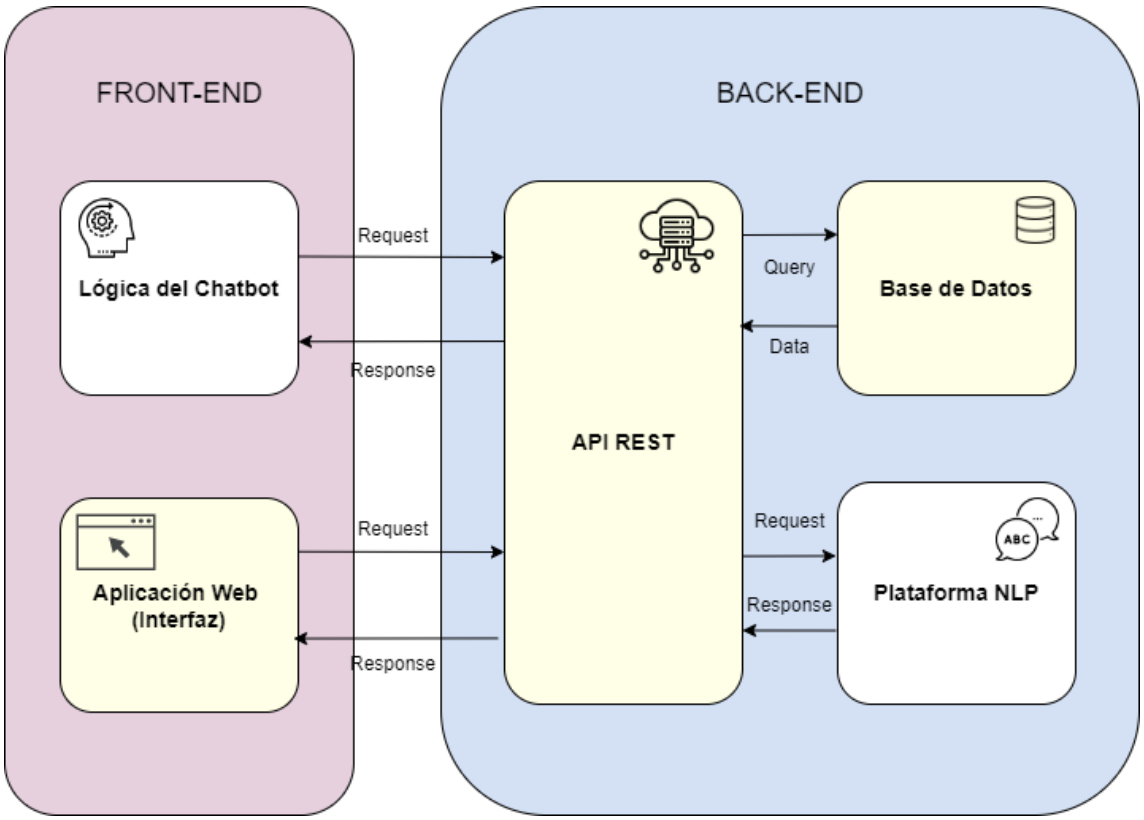

Figura 3.3: Arquitectura de los nuevos componentes

A continuación, se procede a describir estas capas:

Front-End

Se conoce como front-end a la parte de la arquitectura software que permite a los usuarios interactuar con el sistema [61]. En este caso, el front-end está conformado por las interfaces que permiten a los usuarios interaccionar con el Chatbot mediante los intents definidos, componente Chatbot en el diagrama de la figura; y por la aplicación web que facilita la gestión de los datos y el entrenamiento de la aplicación, componente Aplicación Web del diagrama.

El usuario puede plantear preguntas al asistente a través de Telegram, Alexa y vía web, por lo que estos constituyen los canales de comunicación o interfaces que facilitan la inter-

acción con los usuarios y forman el componente Chatbot del diagrama, ya implementado en la arquitectura inicial.

El otro componente que conforma el front-end consiste en la aplicación web que facilite la gestión de los datos necesarios para el funcionamiento del asistente. Esta gestión se realiza paralelamente sobre los datos almacenados en la base de datos y sobre la plataforma NLP, permitiendo también el entrenamiento de la aplicación.

Ambos elementos del front-end deben comunicarse con el back-end, concretamente se comunican con la API REST para facilitar la obtención de las respuestas, con el fin de ofrecerlas a los usuarios del asistente, y la gestión de datos por los usuarios administrativos.

En apartados posteriores se explica con más detalle la implementación y funcionamiento de la aplicación web.

Back-End

Recibe el nombre de back-end la capa que contiene la lógica de la aplicación, encargada del acceso, gestión y almacenamiento de los datos [61]. Esta es por tanto clave para el funcionamiento del sistema.

Tal como se comprueba observando el diagrama, en esta capa podemos diferenciar tres componentes: la API REST, la base de datos y la plataforma NLP. De estos, los que no existen en la arquitectura inicial son la API y la base de datos.

La función de la API REST es implementar una capa de abstracción para el acceso a datos. Esta capa debe ofrecer servicios y funciones que puedan ser utilizados por componentes externos, como la aplicación web, para gestionar los datos. Concretamente, implementa funciones que facilitan la realización de operaciones CRUD sobre los contextos, los intents o preguntas, sus respuestas y los sinónimos de cada pregunta. Asimismo, debe ofrecer un servicio que facilite el entrenamiento de la aplicación mediante la pla-

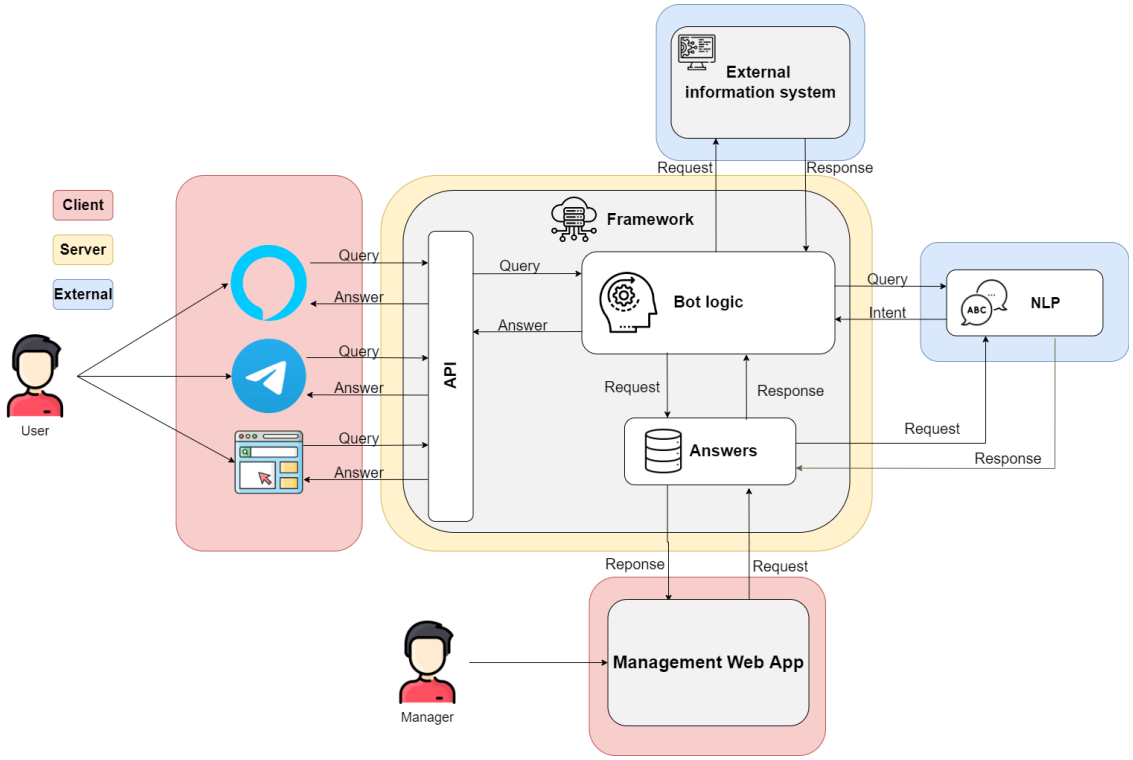

Figura 3.4: Nueva arquitectura del framework

taforma NLP. Es la API la que ofrece servicios al front-end y, por lo tanto, sirve como conector entre este y el back-end.

En cuanto a la base de datos, es la encargada de almacenar toda la información correspondiente a los contextos y los intents o preguntas. A ella accede la API para que los datos se puedan ser ofrecidos al usuario y, si es pertinente, gestionados por él en el front-end.

Los intents o preguntas y las distintas formas de plantear cada una se albergan en la plataforma de procesamiento de lenguaje natural para que el modelo sea entrenado y el asistente responda a las cuestiones planteadas por el usuario con precisión. Se debe recordar que este elemento de se encuentra ya definido en la arquitectura inicial.

El resultado de integrar los componentes de esta arquitectura con la inicial (la mostrada en la Figura 3.3) da como resultado la nueva arquitectura del framework que se pretende conseguir. Su esquema se muestra en la figura 3.4

En esta nueva arquitectura, los elementos que se añaden son los que aparecen destaca-

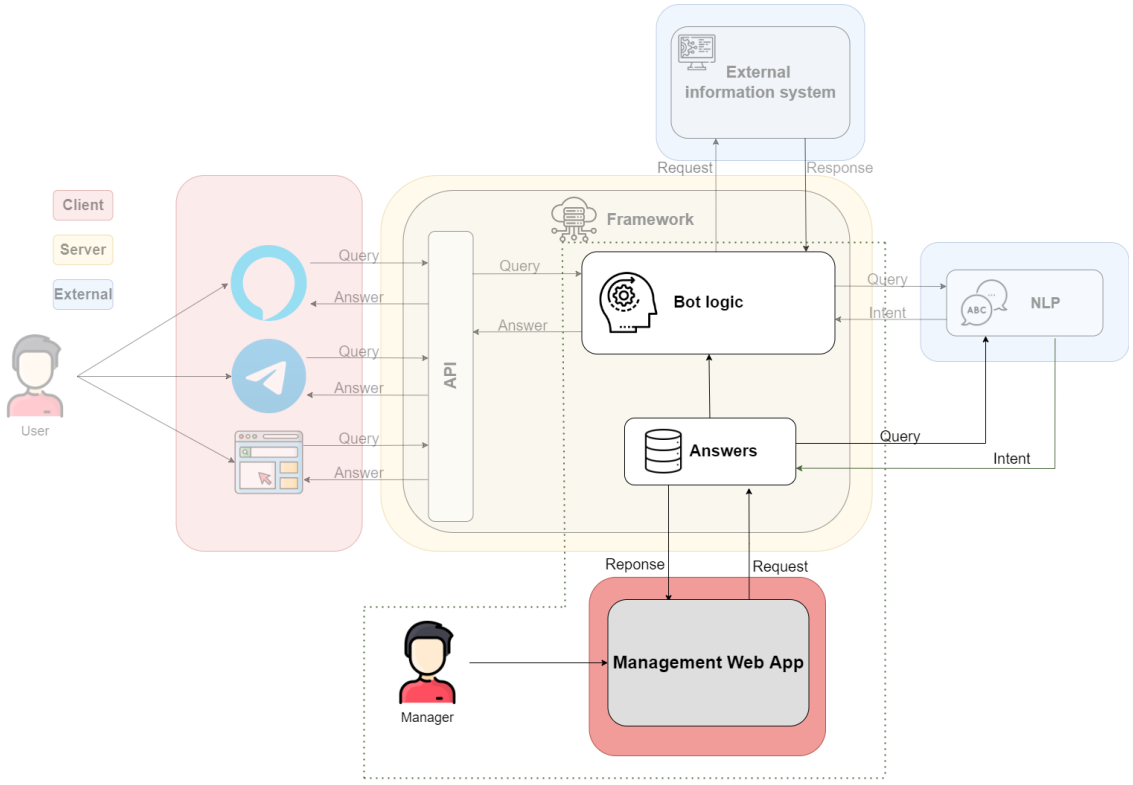

Figura 3.5: Componentes modificados y añadidos al framework

dos en amarillo en la figura 3.3. La API y la base de datos forman parte del componente Answers. Los componentes añadidos y modificados aparecen resaltados en el esquema de la imagen 3.5. Sobre estos componentes se centra principalmente el desarrollo del presente proyecto.

Si se especifica en cada uno de los componentes la tecnología que se ha decidido emplear para su implementación, resultan los diagramas de las figuras 3.6, 3.7 y 3.8

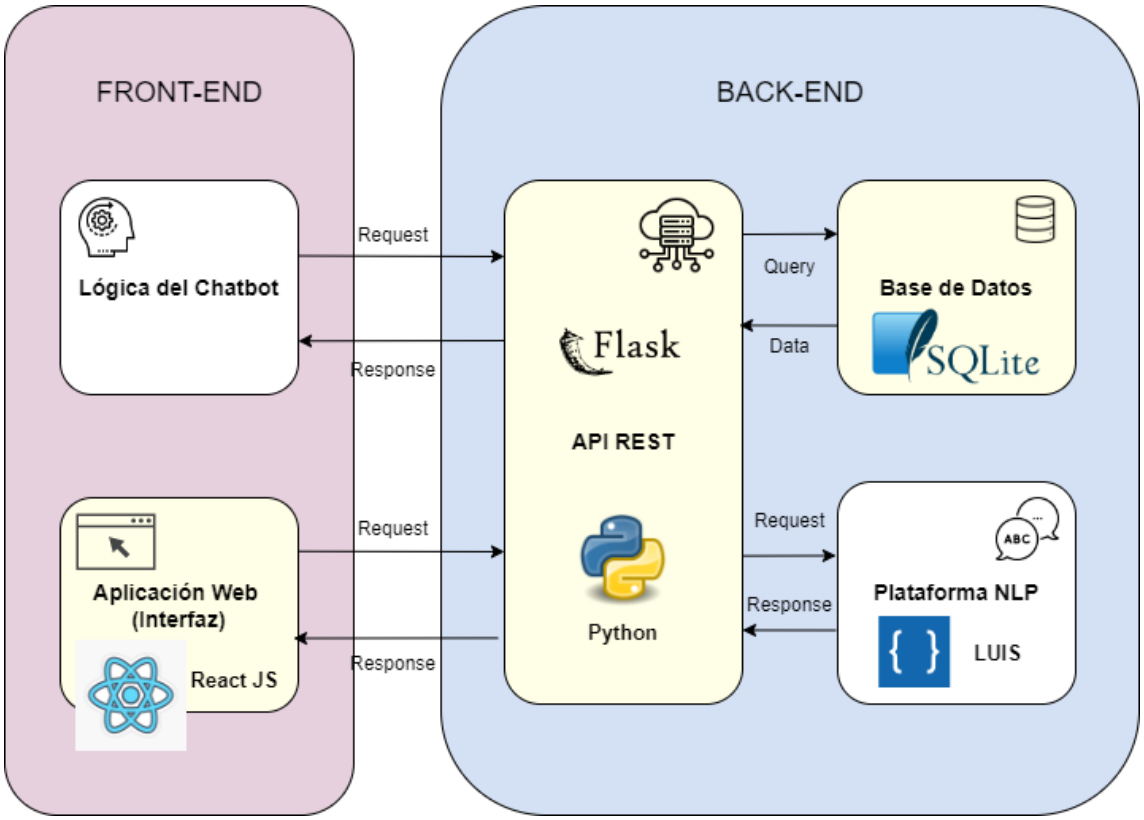

Figura 3.6: Diagrama de tecnologías y nuevos componentes de la arquitectura

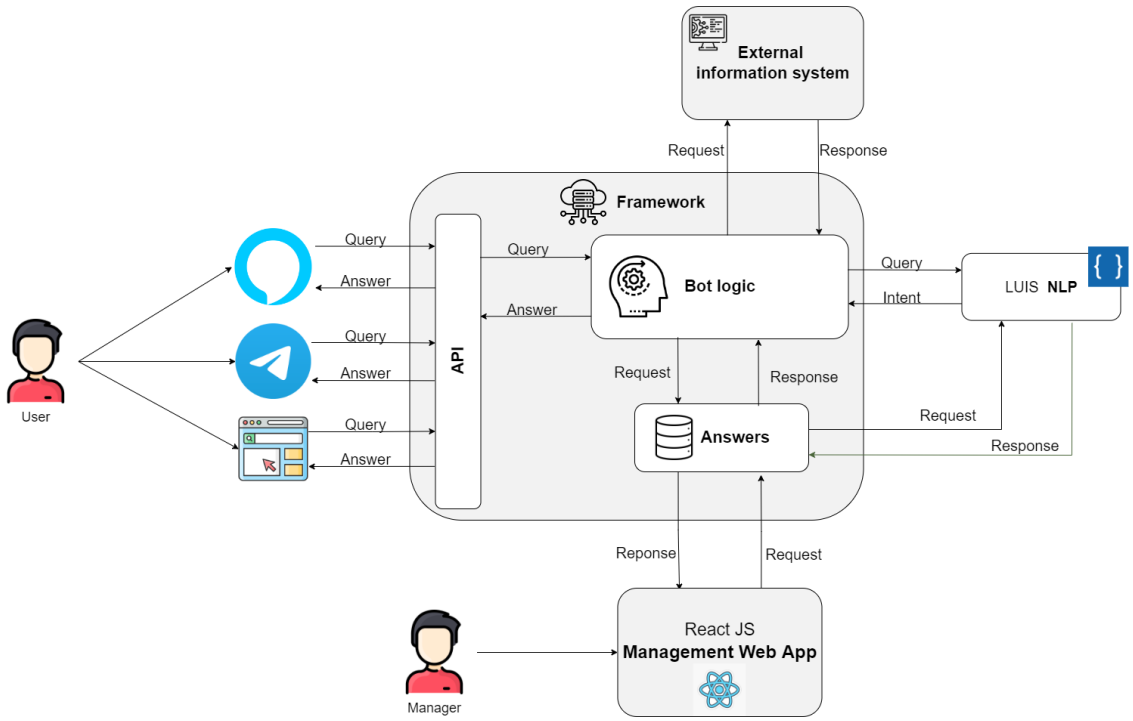

Figura 3.7: Diagrama de tecnologías y componentes de la arquitectura integrada

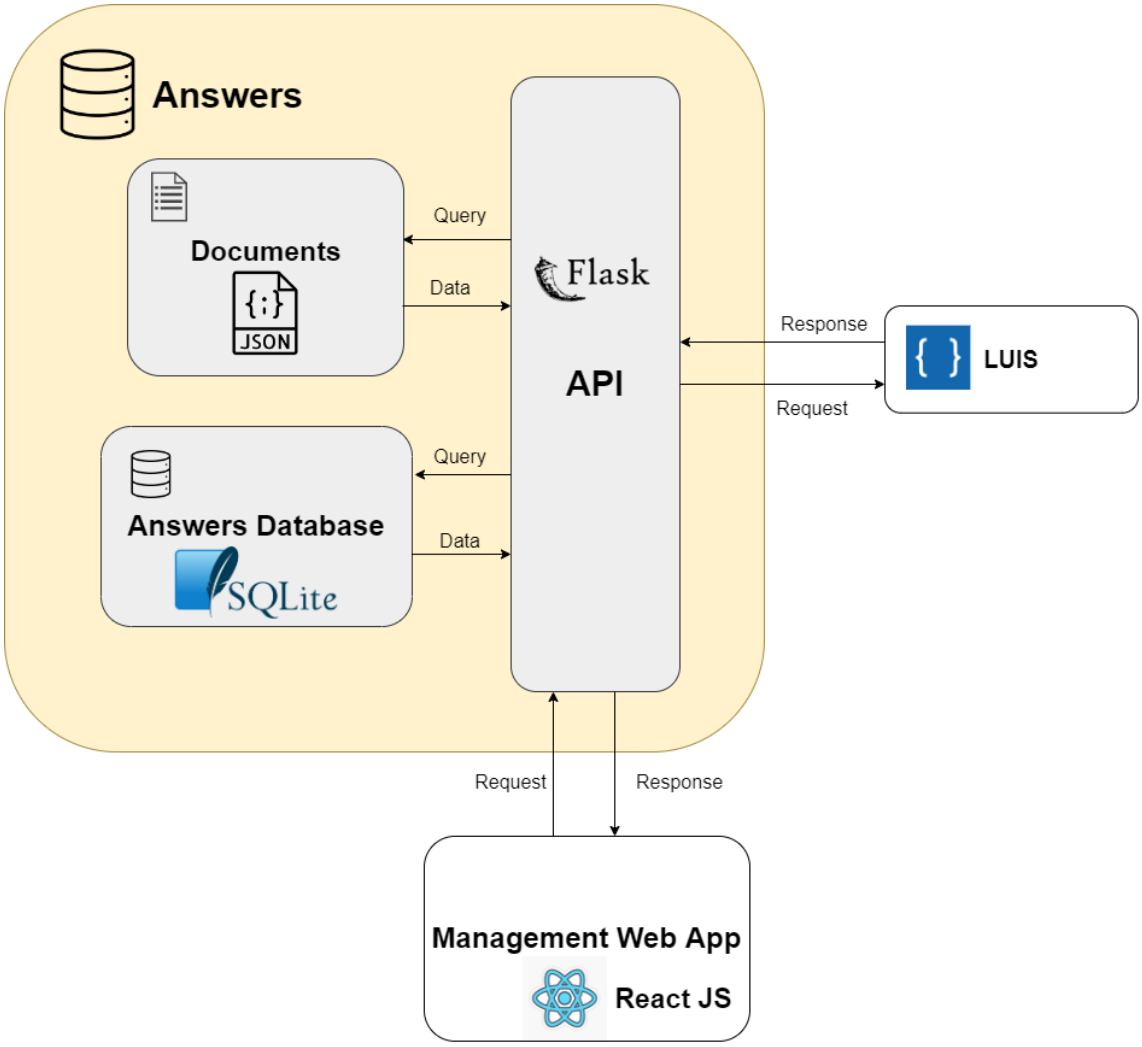

Figura 3.8: Diagrama de tecnologías del componente Answers

En las próximas secciones, se describe con detalle la implementación, desarrollo y funcionamiento de cada uno de los elementos nuevos añadidos a la arquitectura inicial.

3.2.3. Base de datos: contextos y preguntas

El primer componente nuevo añadido a la arquitectura es la base de datos, la cual consiste en una base de datos relacional. En ella se almacenan los contextos de las preguntas y las propias preguntas con sus respuestas.

Concretamente, tal como se describió en el estudio del Estado del Arte, la tecnología SQL elegida para implementar esta base de datos es SQLite, ya que sobre ella se van a realizar operaciones CRUD sencillas y al ser una base de datos del tipo relacional garantiza una mayor integridad, esto es, correctitud y completitud de la información.

Además, SQLite ofrece un uso más simple frente a otras bases de datos y su orientación a aplicaciones sencillas sin tráfico excesivo de datos justifican esta elección.

En esta base de datos se crean dos tablas: una tabla denominada **contextos** que guarda todos los contextos de las preguntas y otra tabla llamada **preguntas** que almacena todas las preguntas que es capaz de responder el chatbot. A continuación se explica el contenido de estas tablas:

- **Tabla contextos:** Esta tabla alberga todos los contextos de las preguntas. Los atributos de los que consta cada contexto son un identificador del contexto y una descripción.

- **Tabla preguntas:** Esta tabla guarda todos los intents o preguntas a los que puede responder el asistente. Los atributos de los que consta cada pregunta son un identificador del intent, un texto descriptivo de la pregunta, el texto que constituye la respuesta, un identificador del contexto al que pertenece la pregunta y un campo dependiente que indica si depende de algún otro intent.

  El identificador del contexto constituye una clave foránea hacia la tabla contextos. El atributo dependiente toma valor nulo en caso de no ser el intent dependiente de ningún otro. La función de este campo se explica más adelante en la especificación de los cambios en la lógica del bot.

Los datos de estas tablas son importados desde archivos CSV. El CSV de los intents es el que empleaba la arquitectura inicial del framework como repositorio de preguntas, mientras que el CSV de contextos era meramente orientativo, ya que la lógica del chatbot no emplea directamente la información sobre los contextos contenida en este archivo.

El atributo dependiente de la tabla preguntas no existía en el CSV de origen. Este se añade con el fin de simplificar la lógica que gestiona los diálogos, la cual se explica en secciones posteriores.

Las figuras siguientes muestran la estructura de las tablas descritas, incluyendo un fragmento de la información que contienen.

	Identificadores	Descripción
	Filtro	Filtro
1	AVIST	Alumnos visitantes
2	EBAU	Evaluación del Bachillerato para el Acceso a la Universidad
3	M25PAU	Acceso para mayores de 25
4	M40PAU	Acceso para mayores de 40
5	M45PAU	Acceso para mayores de 45
6	ALOJ	Alojamiento
7	BECA	Becas
8	TASA	Tasas
9	CE	Cambios de titulación/estudios
10	CV	Campus virtual
11	CONTINUACION	Continuidad de estudios

Figura 3.9: Fragmento de la tabla Contextos

	id_intencion	texto_pregunta	texto_respuesta	id_contexto	dependiente
	Filtro	Filtro	Filtro	Filtro	Filtro
1	ACF00	Tengo un título de técnico ...	Si estás en posesión de un títu...	ACF	*NULL*
2	ACF04	¿Cómo calculo mi nota de ...	Puedes calcular tu nota de ...	ACF	CALCULARNOTA
3	ACF01	¿Los que procedemos de ...	Los estudiantes procedentes d...	ACF	*NULL*
4	ACF02	¿Mi título de técnico superior d...	Independientemente de la ...	ACF	*NULL*
5	ACF03	¿Si deseo acceder a una ...	Podrás presentarte a la fase d...	ACF	*NULL*
6	AV00	Soy estudiante extranjero y ...	Puedes acogerte a la modalida...	ALVIST	*NULL*
7	AV01	¿Qué trámites debo realizar pa...	Tendrás que dirigir una solicitu...	ALVIST	*NULL*
8	AV02	¿Qué efectos académicos tiene...	Una vez finalizado el curso, los...	ALVIST	*NULL*
9	EBAU00	¿Qué es la Evaluación de ...	Es la prueba que deben supera...	EBAU	*NULL*
10	EBAU01	¿Es obligatoria?	Tienen que realizarla todos los...	EBAU	*NULL*
11	EBAU02	¿Dónde se solicita?	En la Secretaría del centro ...	EBAU	DONDESOLICITAR
12	EBAU03	En la solicitud, ¿qué debe ...	El estudiante indicará en la ...	EBAU	*NULL*
13	EBAU04	¿Cómo se estructura la EBAU?	Se estructura en dos fases, fas...	EBAU	*NULL*
14	EBAU05	¿En qué consiste el examen de...	En la fase de acceso, el exame...	EBAU	COMOES
15	EBAU06	¿Es necesario haber cursado la...	Excepto la materia de ...	EBAU	*NULL*
16	EBAU07	¿Puede examinarse de la ...	A los efectos de la evaluación ...	EBAU	*NULL*
17	EBAU08	¿Cuál es la calificación mínima ...	Para superar la EBAU es ...	EBAU	*NULL*
18	EBAU09	¿De qué materias hay que ...	Podrá presentarse a aquellas ...	EBAU	*NULL*
19	EBAU10	¿De cuántas materias puede ...	La normativa contempla que e...	EBAU	*NULL*
20	EBAU11	¿Cómo se calcula la nota de ...	La forma de calcular la nota se...	EBAU	CALCULARNOTA

Figura 3.10: Fragmento de la tabla Preguntas

3.2.4. API REST

La finalidad de la API REST, tal como ya se ha mencionado, es ofrecer una capa de abstracción en el acceso a los datos que necesita el asistente para su funcionamiento, ofreciendo una serie de funciones y servicios a componentes superiores.

Desde esta API se va a gestionar tanto la información guardada en la base de datos como aquella que se encuentra en la plataforma NLP, en este caso, LUIS de Azure.

Los beneficios que ofrece esta API son los siguientes:

1. En caso de ser necesario sustituir la base de datos, o la plataforma NLP encargada de realizar las predicciones, los componentes que utilizan los servicios y funciones que provee la API no tienen que ser modificados.

2. La API aísla la base de datos del exterior, reduciendo los accesos no deseados a

esta.

3. Los desarrolladores de componentes superiores que hagan uso de la API no necesitan saber cómo funciona la base de datos ni cómo se realiza el acceso a esta.

Tal como se mencionó en el Estado del Arte, la tecnología empleada para la implementación de esta API es Flask, framework que se amolda perfectamente a la arquitectura del sistema facilitando un desarrollo de manera ágil y eficiente. Asimismo, al estar escrito en Python, cuenta con con todas las ventajas que ofrece este lenguaje, permitiendo desarrollar la lógica de la API en un número reducido de líneas de código.

La API ofrece funciones para gestionar cada tipo de elemento almacenado en la base de datos, estos son contextos y preguntas con sus respectivos atributos. También accede a LUIS de Azure para gestionar los sinónimos de cada intent y entrenar el modelo para que pueda realizar las predicciones correctamente.

Con el fin de realizar esta gestión, se deben establecer dos tipos de conexiones, una con la base de datos y otra con la plataforma LUIS de Azure.

- Para establecer la conexión con la **base de datos** se hace uso de la librería **SQLite3** de Python [62], que ofrece funciones para conectarse, desconectarse, realizar operaciones y consultas sobre los datos a través de cursores y código SQL incrustado. Mediante el uso de esta conexión y las funciones que provee la librería SQLite3 se gestionan las tablas de contextos y preguntas existentes en la base de datos.

- Con el fin de crear una conexión con la **plataforma NLP**, se emplean las bibliotecas cliente del **SDK de LUIS de Azure** [63]. Existen bibliotecas para distintos lenguajes, en este caso se utiliza la de Python, que permite crear aplicaciones, agregar intenciones y entidades de aprendizaje automático con expresiones de ejemplo, entrenar, publicar la aplicación y consultar el runtime de predicción.

  Para conectarse LUIS, en primer lugar es necesario declarar una clase cliente en el

código de la API, asignándole las claves de autorización y puntos de acceso (*authoringkey* y *endpointkey*) correspondientes a la aplicación que se desee gestionar.

Ya que debe existir una única instancia de este cliente, se implementa el patrón de diseño singleton [64] en Python [65], de manera que no se permite la creación de más instancias de esta clase. Mediante las funciones que provee el cliente se realizan las gestiones pertinentes sobre los datos existentes en la aplicación de LUIS.

Se debe destacar que el formato que utiliza la API en la mayoría de casos para devolver datos y solicitarlos es JSON, ya que este es fácilmente manejable desde las tecnologías que conforman el front-end.

A continuación, se explican las funciones y servicios que implementa la API:

Gestión de contextos

La API permite la gestión de los contextos, almacenados en la base de datos. Cocretamente, implementa funciones que permiten realizar operaciones CRUD sobre estos, es decir, obtener todos los contextos, agregar nuevos, editar los existentes y eliminarlos. Estas funciones requieren únicamente de la conexión a la base de datos, ya que los contextos no existen en la plataforma LUIS.

Las funciones relativas a la gestión de contextos junto con sus rutas de acceso son las siguientes:

- **obetenerContextos():** Se accede mediante la ruta /contextos utilizando como método GET. Devuelve todos los contextos existentes en la base de datos en formato JSON.
- **aniadirContexto():** Se accede mediante la ruta /contextos utilizando como método POST. Permite insertar un nuevo contexto en la base de datos.

- **eliminarContexto():** Se accede mediante la ruta /contextos/idContexto utilizando como método DELETE. Permite eliminar un determinado contexto utilizando su identificador.

No se implementa una función de edición de contextos ya que, en la base de datos, cuentan solo con dos campos y, por lo tanto, es posible eliminar y volver a añadir el contexto modificado sin perder eficiencia y simulando, bajo la visión del usuario final, una edición real.

Gestión de intents

La API facilita la gestión de las preguntas o intents, guardados tanto en la base de datos como en la plataforma NLP, por lo que para acceder a ellos requiere de conexión a ambas fuentes.

Para evitar numerosos accesos continuos a LUIS y conseguir una mayor eficiencia, la API consta también de un archivo JSON que contiene todos los intents existentes en LUIS. Este archivo constituye el componente Documents Repository de la figura 2.6. La primera versión de este archivo se exporta desde la página web de LUIS y la API la actualizará siempre que sea necesario, de manera paralela a la base de datos y a la propia plataforma NLP. De este modo, se consiguen consultas más rápidas de los intents.

En concreto, la API implementa funciones que permiten realizar operaciones CRUD sobre los intents, esto es, obtener todos los intents, obtener las preguntas pertenecientes a un contexto, agregar nuevas, editar las existentes y eliminarlas.

Las funciones relativas a la gestión de intents junto con sus rutas de acceso son las siguientes:

- **obtenerObtenerPreguntasContexto():** Se accede a esta función mediante la ruta /contextos/idContexto utilizando como método GET. Devuelve todos los intents correspondientes a un determinado contexto pasado por parámetro en formato JSON.

- **obtenerIntents():** Se accede mediante la ruta /intents utilizando como método GET. Devuelve todos los intents reconocidos por el sistema.

- **obtenerRespuestaIntent():** Se accede a esta función mediante la ruta /respuesta/idIntent/ utilizando como método GET. Devuelve la respuesta a un determinado intent haciendo uso de su identificador, pasado por parámetro. Esta función solo devuelve la respuesta de aquellos intents que no son dependientes.

- **obtenerRespuestaIntentDependiente():** Se accede a esta función mediante la ruta /respuesta/idIntent/dependiente utilizando como método GET. Devuelve la respuesta de un determinado intent dependiente haciendo uso de los parámetros pasados a la función. Estos parámetros consisten en el identificador del intent y el identificador del intent del que depende. La finalidad y el modo en que funciona la nueva gestión de los intents dependientes se explica en apartados posteriores.

- **obtenerContextoIntent():** Se accede a esta función mediante la ruta /contexto/idIntent utilizando como método GET. Devuelve el contexto de un determinado intent utilizando el identificador del intent pasado por parámetro.

- **aniadirIntent():** Se accede a esta función mediante la ruta /intents utilizando como método POST. Permite agregar un nuevo intent en el sistema.

- **modificarIntent():** Se accede a esta función mediante la ruta /intents/idIntent utilizando como método PUT. Permite editar el intent que corresponde al identificador de intent pasado por parámetro.

- **modificarRespuestaIntent():** Se accede mediante la ruta /modificarRespuesta/idIntent utilizando como método PUT. Esta función permite modificar la respuesta a un determinado intent que se corresponde con el identificador pasado por parámetro.

- **eliminarIntent():** Se accede a esta función mediante la ruta /intents/idIntent utilizando como método DELETE. Permite eliminar un determinado intent utilizando su identificador.

De las funciones anteriores, solo aquellas que actualizan los datos relativos a los intents acceden a la base de datos, la plataforma LUIS y el archivo JSON, actualizándolos de forma casi simultánea. Aquellas funciones cuyo objetivo sea solo consultar, acceden únicamente a la base de datos, al archivo JSON o a ambas en función de los datos que requiera la consulta.

Gestión de sinónimos

Otro servicio más que ofrece la API consiste en la gestión de los sinónimos de las preguntas, estos son, las distintas formas de plantear un intent. Estos se albergan únicamente en la plataforma NLP para el posterior entrenamiento del modelo, que permite al sistema identificar las intenciones de los usuarios, por lo que para acceder a ellos solo se requiere de la conexión con LUIS de Azure.

Las funciones relativas a la gestión de sinónimos junto con sus rutas de acceso son las siguientes:

- **obtenerIntent():** Se accede a esta función mediante la ruta intents/idIntent utilizando como método GET. Devuelve todos los sinónimos del intent indicado en formato JSON.
- **aniadirSinonimo():** Se accede a esta función mediante la ruta sinonimos/idIntent utilizando como método POST. Añade un nuevo sinónimo al intent cuyo identificador se pasa por parámetro.
- **eliminarSinonimo():** Se accede a esta función mediante la ruta sinonimos/textoSinonimo utilizando como método DELETE. Elimina el sinónimo cuyo texto se pasa por parámetro.

Entrenamiento de la aplicación

El último servicio que ofrece la API es el entrenamiento de la aplicación. Este es de vital importancia para el óptimo funcionamiento del asistente, ya que si se añaden, modifican o eliminan intents y sinónimos sin entrenar seguidamente el modelo, los cambios no serán efectivos y el chatbot puede no reconocer preguntas, malinterpretarlas o dar respuestas erróneas entre otros probables fallos.

Para llevar a cabo este entrenamiento, la API implementa un método denóminado **entrenarModelo()** que emplea la función *train* incluida en la biblioteca cliente del SDK de LUIS. A este servicio se accede a través de la ruta /entrenarModelo.

Todas las funciones son probadas tras finalizar su implementación. El método de testeo se realiza mediante la herramienta Postman y se explica en secciones posteriores.

3.2.5. Aplicación Web

La finalidad principal de la aplicación web es ofrecer al usuario una interfaz simple e intuitiva que permita gestionar los datos que requiere el asistente virtual para su funcionamiento, además de facilitar el entrenamiento del modelo.

El usuario objetivo de esta aplicación es el personal del Sección de Información y Atención al Alumnado de la UEx (SIAA). Estos usuarios, aunque no son informáticos expertos, están habituados a trabajar con programas de gestión de datos, por lo que no les resultará difícil el manejo de la aplicación.

El tipo de aplicación por el que se ha optado es una aplicación web, pues se presenta como el más adecuado para para llevar a cabo la finalidad descrita y para ser integrado con la arquitectura desarrollada. Se plantea una página web responsive, es decir, adaptable a diferentes formatos de pantalla y se pueda emplear en distintos dispositivos, como tabletas o teléfonos inteligentes, además de ordenadores, aunque lo ideal para este tipo de gestión

es utilizar un ordenador.

Tal como se menciona en el estudio del Estado del Arte, la tecnología que se decide emplear para el desarrollo de esta aplicación es React JS, que proporciona flexibilidad, rapidez y buena organización del código. Además, se utiliza el framework de estilo Bulma [66], que simplifica el diseño de la interfaz y consigue que sea responsive.

La aplicación web realiza peticiones a la API REST creada para ofrecer las funciones que satisfacen los requisitos descritos. En los siguientes subapartados se describe brevemente el proceso de creación de la aplicación.

Requisitos generales de la aplicación

En primer lugar, se deben establecer los requisitos de los que consta la aplicación. Estos se definen a partir de aquellas tareas que el usuario debe ser capaz de realizar a través de la página web, es decir, tareas de gestión de los datos que utilza Lyx.

La especificación de requisitos es la siguiente:

- El sistema debe permitir al usuario consultar contextos, añadir nuevos, eliminarlos y modificar los existentes.
- La aplicación debe facilitar al usuario la gestión de los intents, permitiéndole consultarlos, clasificarlos por contexto, insertar nuevos, modificar los existentes y eliminarlos.
- El sistema debe permitir al usuario añadir nuevos sinónimos para las preguntas y eliminar existentes si es necesario.
- La aplicación debe ofrecer a usuario una función para entrenar el modelo de predicción de intents.
- El sistema debe permitir a los usuarios registrarse en el sistema, iniciar sesión y cerrarla.

- La aplicación debe asegurar seguridad en el acceso a datos, permitiendo utilizar los servicios que provee solo a los usuarios dados de alta en el sistema.
- El sistema debe proporcionar ayuda o indicaciones para el uso de la aplicación.

Casos de uso de la aplicación

Una vez especificados los requisitos generales de la aplicación, quedan identificados también los casos de uso, que se muestran en el diagrama siguiente (figura: 3.11).

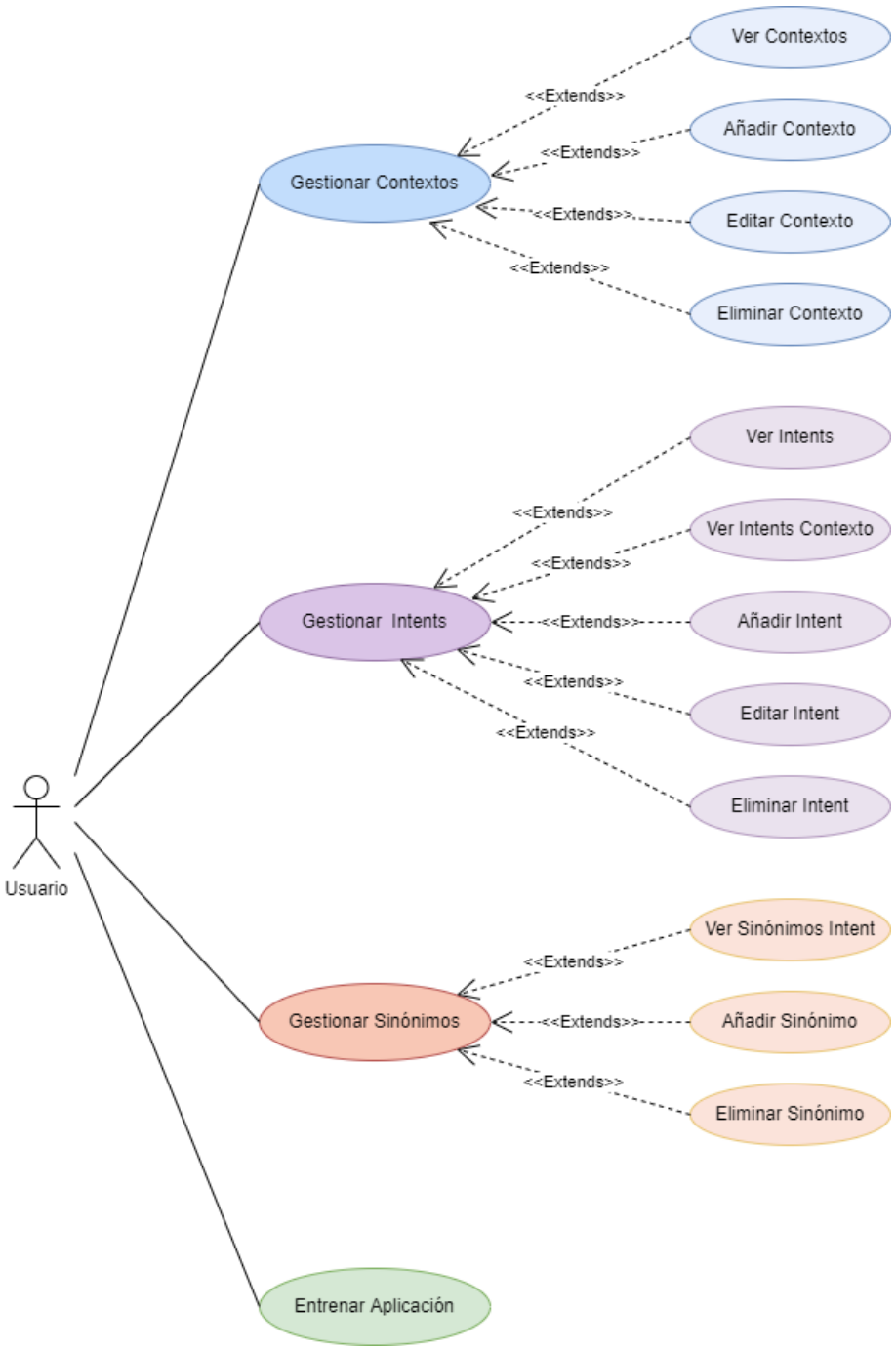

Figura 3.11: Casos de Uso de la Aplicación

Proceso de diseño

Una vez analizados los requisitos de la aplicación, se procede al desarrollo de la interfaz.

El primer paso para el diseño fue la creación de bocetos. Tras realizar varios, se seleccionaron los que mejor se adaptan a los requisitos descritos para continuar desarrollándolos.

Ya elegida la idea principal de diseño plasmada en los bocetos, se continúa realizando un prototipo o wireframe digital [67]. Este constituye el esquema de las páginas, una especie de guía sencilla y visual de la estructura de las diferentes pantallas de la aplicación. Este nos sirve para establecer una idea inicial de cómo transcurre el flujo a través de las diferentes pantallas y qué elementos contiene cada una.

Para la creación del prototipo digital de mayor fidelidad, se emplea la herramienta Balsamiq [68], que facilita la creación de wireframes de manera simple incluyendo iconos, botones, imágenes y permitiendo además añadir enlaces entre diferentes páginas del prototipo.

Para validar el prototipo creado, se realizan revisiones y se efectúan reuniones con miembros del SIAA a los que se les muestra el wireframe. Este proceso de testeo se describe con más detalle en apartados posteriores.

Las pantallas que componen el prototipo digital pueden visualizarse en el Anexo I.

Implementación de la aplicación

Ya diseñado y validado el prototipo, se procede a la creación de la aplicación real con React JS.

Para la gestión del Login y registro se emplea la plataforma Auth0 [69], que facilita la autenticación y autorización de usuarios, es adaptable a distintas tecnologías y sencilla

de instalar. Mediante esta plataforma se administran los usuarios, pudiendo dar de alta los nuevos que se registren a través de su correo electrónico, establecer roles y consultar todos los que tienen acceso a la aplicación.

El resto de páginas que forman la aplicación web, además de las limitaciones de uso y y el proceder de la aplicación en determinadas situaciones en las que el comportamiento podría resultar ambiguo, se describen con detalle en el Anexo II, que constituye el manual de usuario de la aplicación.

Se debe tener en cuenta que, aunque el usuario controle las funciones típicas de aplicaciones de gestión de información, existen otras no tan obvias, como el entrenamiento de la aplicación, por lo que se debe recordar la importancia de estas funciones en las páginas de ayuda de la aplicación.

3.2.6. Integración de los nuevos componentes en la arquitectura. Mejoras derivadas.

Una vez desarrollados los nuevos componentes, es necesario integrarlos en la arquitectura original sin que el funcionamiento del sistema se vea afectado. Las modificaciones que llevan a cabo para hacer efectiva esta integración, se realizan íntegramente en el componente lógica del bot. Las acciones que se realizan son las siguientes:

En primer lugar, se modifica el código para que en lugar de obtener las respuestas a las preguntas desde los ficheros CSV, lo haga ejecutando llamadas a la API desarrollada. De este modo, los datos proceden de la base de datos en lugar del CSV. Se debe tener en cuenta la existencia de ciertas preguntas que no se encuentran en la base de datos, estas son las relativas a las titulaciones, en qué ciudades y centros se pueden estudiar estas, qué titulaciones se imparten en cada centro y ciudad, notas de corte y ciudad en la qué se ubica cada centro. La lógica del bot para la obtención de las respuestas relativas a estas preguntas permanece sin modificaciones, es decir, continúa utilizando los ficheros CSV

correspondientes, ya que para estos casos la lógica funciona de manera diferente.

Una vez modificado el modo de obtención de los datos, es posible llevar a cabo una serie de cambios que simplifican la implementación de la lógica del bot haciéndola más eficiente. Concretamente, estas modificaciones se realizan sobre el componente encargado de gestionar los diálogos. Antes de explicar las transformaciones llevadas a cabo, es necesario recordar cómo funciona en la arquitectura original la gestión de los diálogos.

Gestión de los diálogos en la arquitectura original

El componente diálogos contiene la definición de las diferentes clases que implementan cada uno de los diálogos que puede lanzar la lógica del bot. Los diálogos son se utiliza para mantener el contexto de la conversación, teniendo en cuenta la anterior pregunta formulada por un usuario. Esta característica es la clave en los chatbots sensibles al contexto.

Para lograr esto, la lógica del bot comprueba el contexto del intent que ha sido reconocido por el sistema NLP y, si este es dependiente (”DEPENDIENTE”), la conversación se redirige al diálogo, que se almacena en el contexto del usuario. Cada uno de los diálogos verifica si el contexto con el que se ha lanzado es dependiente y, en tal caso, chequea las siguientes situaciones:

- Si el intent que ha lanzado el usuario está dentro de los que este el diálogo puede responder por el contexto, es el caso de los plazos, etc., y la intención dependiente se incluye en el diálogo, el valor de la intención se cambia en el contexto del usuario por la intención general que corresponde y se solicita la respuesta utilizando las diferentes funciones de respuesta que se definen en la clase de contexto.
- Si la intención dependiente no está incluida en ese contexto, se le dice al usuario que no se le puede ayudar con eso en ese momento.
- Si el contexto en el que se inició el diálogo no es “DEPENDIENTE”, la pregunta se responde sin ninguna modificación de la intención, ya que es entendido como una

intención general.

Toda esta lógica se simplifica en gran medida gracias a la API y la base de datos, tal como se explica en el apartado siguiente.

Gestión de los diálogos en la nueva arquitectura

Gracias a la integración de la API y la base de datos en la arquitectura, se pueden eliminar una gran parte de las clases del componente diálogos sin perder la sensibilidad al contexto. Estos son todos los diálogos menos los referidos a notas de corte, ciudades, centros y titulaciones, como ya se ha comentado.

El comportamiento de la lógica tras las modificaciones sería el siguiente:

Se comprueba el contexto del intent que ha sido reconocido por el sistema NLP. Si este es ”DEPENDIENTE”, se hace una llamada a la función de la API que obtiene las respuestas de los intents dependientes. Esta función accede a la base de datos para obtener la respuesta del intent utilizando su identificador y el del intent del cual depende. Este último está almacenado en el contexto del usuario y, en caso de ser nulo, se informa al usuario de que no se le puede ayudar con esa pregunta. Es en la obtención de las respuestas a los intents dependientes donde el nuevo atributo añadido a la tabla Preguntas de la base de datos juega un papel clave. Este campo se denomina “dependiente” e indica el identificador del intent del que depende otro. De esta manera toda la gestión realizada en las clases del componente diálogo queda sustituida por una única llamada a la API.

Si el contexto no es “DEPENDIENTE” y tampoco corresponde con alguno de los diálogos no modificados, la pregunta se responde llamado al método de la API que devuelve la respuesta a un intent genérico, es decir, no dependiente.

Para el resto de diálogos (ciudades, notas de corte, centros, etc.) la lógica se mantiene igual que en la arquitectura original.

Capítulo 4

Validación

En el presente capítulo, se describen las pruebas y tests de validación más relevantes llevados a cabo con el fin de comprobar el correcto funcionamiento del sistema tras las ampliaciones y modificaciones realizadas en su arquitectura. Estas pruebas se llevan a cabo sobre los nuevos componentes añadidos y sobre el sistema global. Así, se puede validar el óptimo comportamiento del sistema completo y verificar que no se dan errores durante su ejecución.

En primer lugar, se describen las pruebas realizadas sobre la API REST, posteriormente sobre la aplicación web y, por último, sobre el sistema completo.

4.1. Validación de la API REST

Previamente a su integración con el resto de elementos de la arquitectura, durante el desarrollo de la API, se han llevado a cabo tests sobre la API para verificar que responde a las peticiones de manera correcta.

La herramienta seleccionada para la realización de estas pruebas es Postman [70], que facilita la generación de llamadas a la API, además de gestionarlas y probarlas.

Se crea una prueba para cada función de la API, estableciendo los parámetros de la llamada si la función lo requiere, el código que debe retornar y comprobando los datos que retorna. Las llamadas que se realizan pretenden lograr las siguientes acciones:

- Mostrar todos los contextos.
- Crear un nuevo contexto.
- Editar el nombre del contexto.
- Añadir un nuevo intent al contexto previamente creado.
- Mostrar todos los intents existentes.
- Mostrar los intents de un contexto determinado.
- Editar la respuesta del intent creado.
- Editar otros parámetros del intent.
- Mostrar la respuesta del intent creado.
- Añadir un nuevo sinónimo para el intent.
- Eliminar el sinónimo creado.
- Eliminar el intent.
- Eliminar el contexto.

Se crea una colección de pruebas en la interfaz de Postman estableciendo una llamada para cada acción de la lista anterior. En la imagen de la figura 4.1 se puede visualizar la ejecución de la colección de pruebas completa. Como se puede comprobar, la API pasa todos los tests creados y todas las llamadas retornan el código 200 (ejecución correcta)

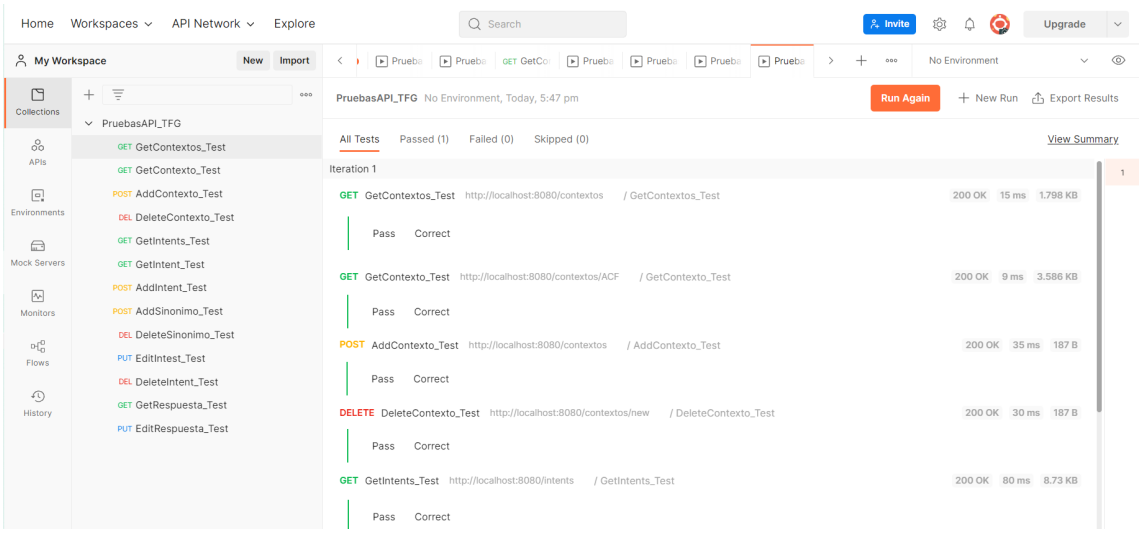

Figura 4.1: Pruebas de la API en Postman

Postman también permite consultar el tiempo que en milisegundos que tarda la API en responder y el tamaño de las respuestas devueltas, de modo que se puede comprobar la eficiencia.

4.2. Pruebas sobre la aplicación web

Sobre la aplicación web, el tipo principal de pruebas se han realizado son test de usuario, además de pruebas manuales de cada pantalla y revisiones.

Los tests de usuario consisten en una metodología o técnica de investigación que se utiliza en el diseño centrado en el usuario y que tiene como finalidad evaluar un producto digital, tanto en fases de diseño iniciales como en una fases de optimización [71].

Estos test se realizan sobre el wireframe, fase inicial de diseño, y distintas versiones de la aplicación, siendo los usuarios destinatarios de los test miembros del SIAA. Los test llevados a cabo son del tipo moderado, es decir, un tipo de test en el que el usuario es guiado en todo momento por una persona que va planteándole preguntas. La principal ventaja de este tipo de prueba es la posibilidad de obtener información adicional de los

usuarios mientras se interactúa con ellos.

Los miembros del SIAA, además de ser los usuarios finales de la aplicación, juegan el papel de clientes en el desarrollo de la arquitectura propuesta, por lo que mediante las reuniones, además de llevar a cabo tests de usuario, se pretende descubrir sus necesidades y satisfacerlas, tal como se procede en un proyecto de Ingeniería de Software.

Las pruebas se han realizado de manera remota mediante reuniones por vídeo conferencia con la representante del SIAA. Concretamente se realizó una reunión previa a la creación del primer prototipo para descubrir que espera el usuario de la aplicación. Una vez creado el boceto en papel y ser revisado por el equipo de desarrollo, se creó el wireframe digital. En este momento se lleva a cabo una segunda reunión con la representante del SIAA para validarlo. Tras la aprobación del prototipo digital por el cliente, que no puso ningún inconveniente al diseño y no expresó problemas de usabilidad, se procede a la creación de la aplicación.

Durante el proceso de desarrollo de la aplicación, se realizaron varias reuniones con el equipo de desarrollo para revisar el diseño y funcionamiento de la aplicación hasta solventar todos los problemas encontrados y disponer de una aplicación totalmente funcional que cumpliese con los requisitos esperados.

Para finalizar, se realiza una última reunión con la representante del SIAA con el objetivo de mostrarle el resultado del proceso de desarrollo y obtener una retroalimentación final, que resulta positiva.

4.3. Validación del sistema completo

Una vez probados los nuevos componentes de la arquitectura por separado, de manera independiente, y tras integrarlos con el resto de elementos de la arquitectura inicial, se procede a validar el funcionamiento del sistema completo. Se han llevado a cabo pruebas del sistema completo pasando por todas las fases del ciclo, desde el inicio al entrenamien-

to.

En primer lugar, se prueba el funcionamiento del chatbot en Telegram, testeando su comportamiento tanto en la versión móvil como en la versión web. Se presta especial atención en las respuestas proporcionadas para los intents dependientes, comprobando que no se produzcan repuestas erróneas o que no correspondan al contexto indicado. Asimismo, se han analizado los accesos a la base de datos y las peticiones a LUIS.

En cuanto a la integración de la nueva base de datos, se comprueba que tras realizar modificaciones desde la plataforma web, estas se hacen presentes en la base de datos, de manera que se mantiene la integridad de la información. Del mismo modo, se comprueba que los cambios que se llevan a cabo desde la aplicación web que afectan a los datos que alberga LUIS, se hacen efectivos en esta plataforma. Además, se prueba que el entrenamiento del modelo desde la aplicación se realiza de forma correcta. Esto es de importancia vital para el óptimo funcionamiento del chatbot, ya que si no se entrena el modelo tras realizar modificaciones en los datos, los cambios no son efectivos.

Capítulo 5

Gestión del Proyecto

En el presente capítulo se describe cómo se ha realizado la gestión del proyecto desarrollado, incluyendo las metodologías y herramientas empleadas para ello.

La organización integra del trabajo llevado a cabo, tanto la investigación previa acerca del mundo de los chatbots, análisis de la arquitectura inicial y tecnologías alternativas, como el desarrollo de la propuesta de la solución y su validación, han sido realizados siguiendo una **metodología ágil**. Concretamente, se ha empleado la metodología **SCRUM** junto con un desarrollo iterativo e incremental. En este caso, las reuniones de seguimiento se realizan con periodicidad semanal.

La herramienta de administración de tareas utilizada para la gestión del proyecto es Jira, que permite llevar un seguimiento del progreso y registro de incidencias según establece la metodología ágil utilizada. Además, se hace uso de un repositorio Git alojado en Bitbucket.

Para cada iteración se asignan un conjunto de tareas y se registran en Jira con el fin de resolverlas antes de la siguiente semana (*sprints* semanales). Una vez las tareas fijadas se completan, se procede a verificar su correcto funcionamiento y se asignan las tareas para la siguiente iteración.

Para gestionar las referencias consultadas en el proceso de investigación, se ha empleado la herramienta Mendeley, que facilita la realización de esta tarea de manera sencilla.

Se debe señalar que, aunque existe un proceso de investigación anterior al desarrollo, está ha continuado durante el transcurso del proyecto, ya que han surgido problemas y nuevas necesidades que requerían búsqueda de soluciones.

5.1. Resolución de incidencias en Jira

En el gráfico mostrado en la figura 5.1, es posible observar el diagrama de flujo acumulado generado en Jira. En este se tienen en cuenta todas las incidencias resueltas durante el tiempo que ha durado el desarrollo del proyecto, por lo que permite valorar el trabajo del mismo.

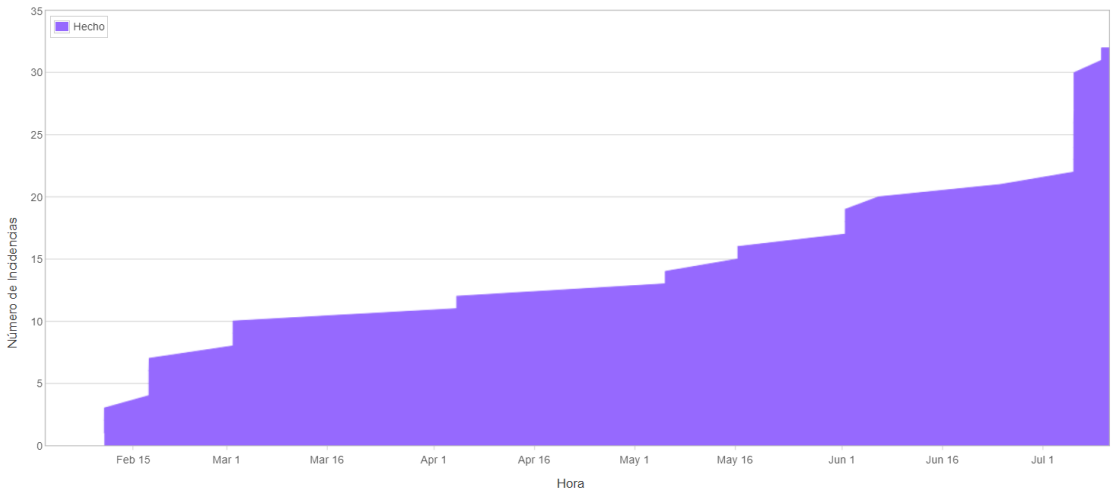

Figura 5.1: Diagrama de flujo acumulado del proyecto

Tal como se puede observar, el número de incidencias ha ido aumentando progresivamente a lo largo del desarrollo del proyecto. Al inicio, se realizan tareas simples de aprendizaje e investigación. Con el avance del proyecto, se van sumando a las anteriores tareas de implementación y desarrollo del sistema. Finalmente, se llevan a cabo las tareas

de documentación. En los últimos meses del curso, como puede comprobarse, se produce un incremento notable de las incidencias debido a un mayor tiempo de dedicación al proyecto.

Mediante Jira, es posible obtener también el número total de horas que se han dedicado a completar todas las incidencias. En total, se calculan unas 260 horas para el desarrollo de este proyecto, correspondiendo con las tareas de investigación y recopilación de información, además del desarrollo de la solución. A estas horas se debe sumar el tiempo empleado para la redacción de la presente memoria y para la preparación de la presentación oral.

Capítulo 6

Conclusiones y trabajos futuros

Se ha cumplido con el objetivo y alcance principal establecido al inicio del proyecto: se ha ampliado y mejorado la arquitectura del framework para el desarrollo de chatbots que emplea Lyx, asistente virtual de la Universidad de Extremadura, llevando a cabo el diseño, implementación y desarrollo de una API REST y una plataforma web para la gestión de los datos con los que trabaja el asistente.

La integración de estos nuevos componentes se ha realizado sin alterar el funcionamiento del chatbot y manteniendo las principales características del framework, esto es, se conserva su capacidad para permitir el desarrollo de chatbots multiplataforma y su diseño modular, siendo posible realizar cambios en los componentes del sistema sin tener que modificar los demás.

Con la implementación de la API se logra una abstracción en el acceso a datos, siendo posible así modificar también las fuentes de datos sin que el resto de componentes se vean afectados.

Con la aplicación web desarrollada, mediante su interfaz intuitiva y minimalista, se alcanza el objetivo de facilitar una gestión sencilla de los datos, permitiendo además el entrenamiento del modelo.

Mediante la creación de la base de datos se consigue una mejor gestión de los datos, siendo más accesibles y logrando una integridad y seguridad mayores.

6.1. Trabajos Futuros

Como ya se ha mencionado previamente, los resultados obtenidos con las ampliaciones desarrolladas cumplen con los objetivos establecidos para este proyecto. Sin embargo, existen una serie de trabajos futuros que podrían realizarse sobre el sistema actual:

- Desarrollar una biblioteca que facilite el desarrollo de chatbots multiplataforma basados en el framework que se amplía en este trabajo, lo que permite a los desarrolladores ahorrar tiempo en la escritura de código. Por ejemplo, se pueden llevar a cabo conexiones con un mayor número de canal. El desarrollador solo debería indicar los que hay que activar y establecer la configuración necesaria para los que dependen de ella.
- Probar el sistema en otros entornos, como atención al cliente o marketing, y analizar en qué plataformas los usuarios se sienten más cómodos interactuando con los asistentes.
- Probar la escalabilidad y el rendimiento del sistema con un número mayor de usuarios y plataformas y considerar cómo podría mejorarse replicando algunos componentes o todo el sistema.
- Considerar cómo podrían integrarse funcionalidades que solo están disponibles en ciertos canales y cómo afectarían al funcionamiento del sistema propuesto.
- Investigar la implementación de nuevas funcionalidades, cómo la obtención de certificados, expedientes y otros documentos mediante el uso de certificados digitales.
- Agregar en la página web una funcionalidad para exportar los datos a otros formatos, como CSV o Excel.

- Extender las modificaciones realizadas en el manejo de los diálogos al código que gestiona el asistente en Alexa, ya que solo se han hecho efectivas en Telegram.

Anexos

Anexo I: Prototipo inicial o wireframe de la aplicación

En este primer anexo, se muestran las páginas del prototipo digital o wireframe de la aplicación que se utilizó para realizar las pruebas con el usuario objetivo, este es, el personal del SIAA. A continuación se muestran las pantallas de este prototipo.

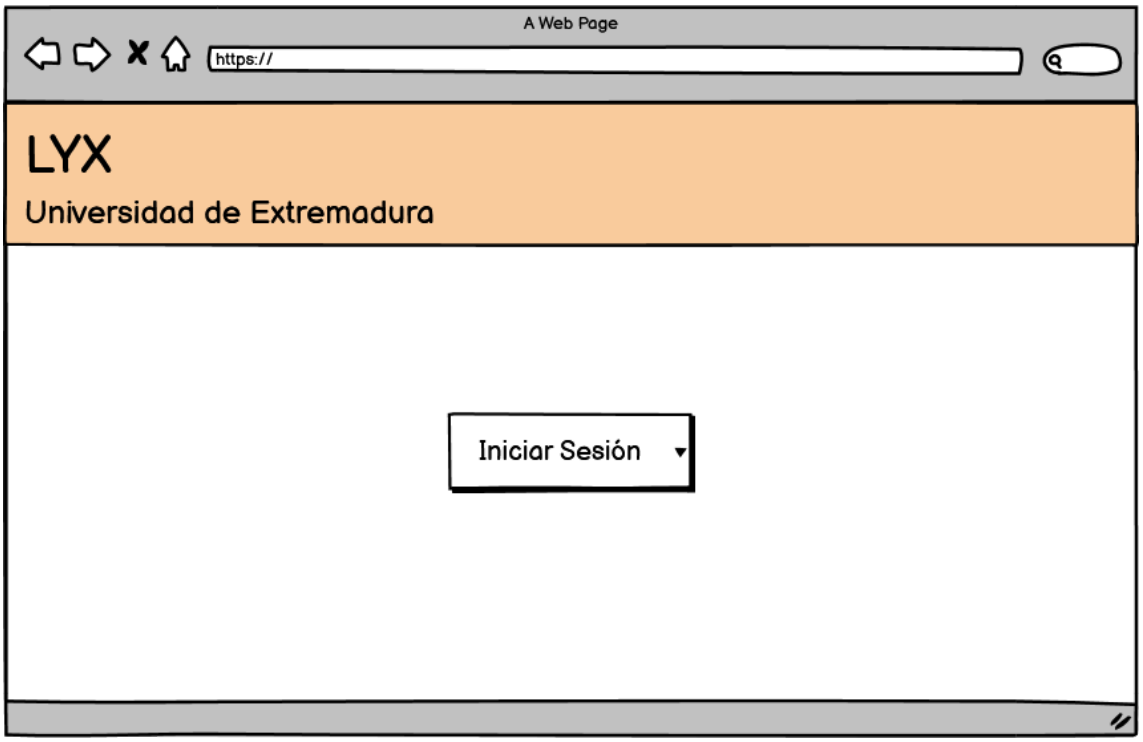

Figura 1: Página de bienvenida

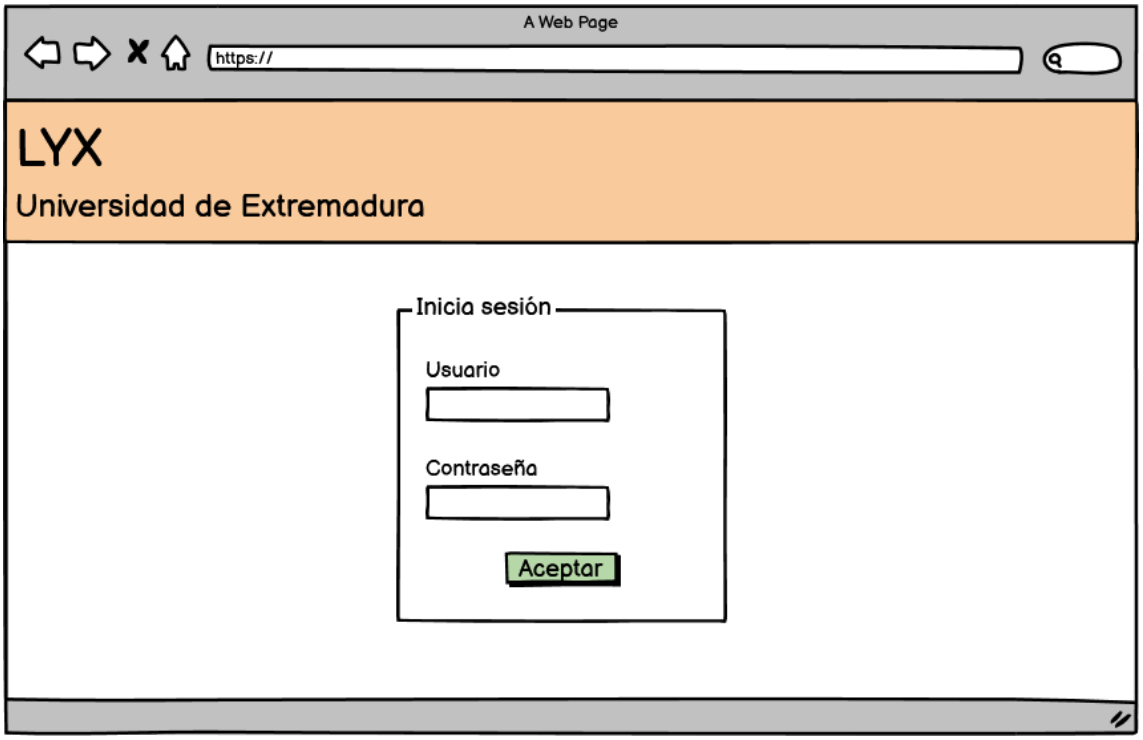

Figura 2: Página de inicio de sesión

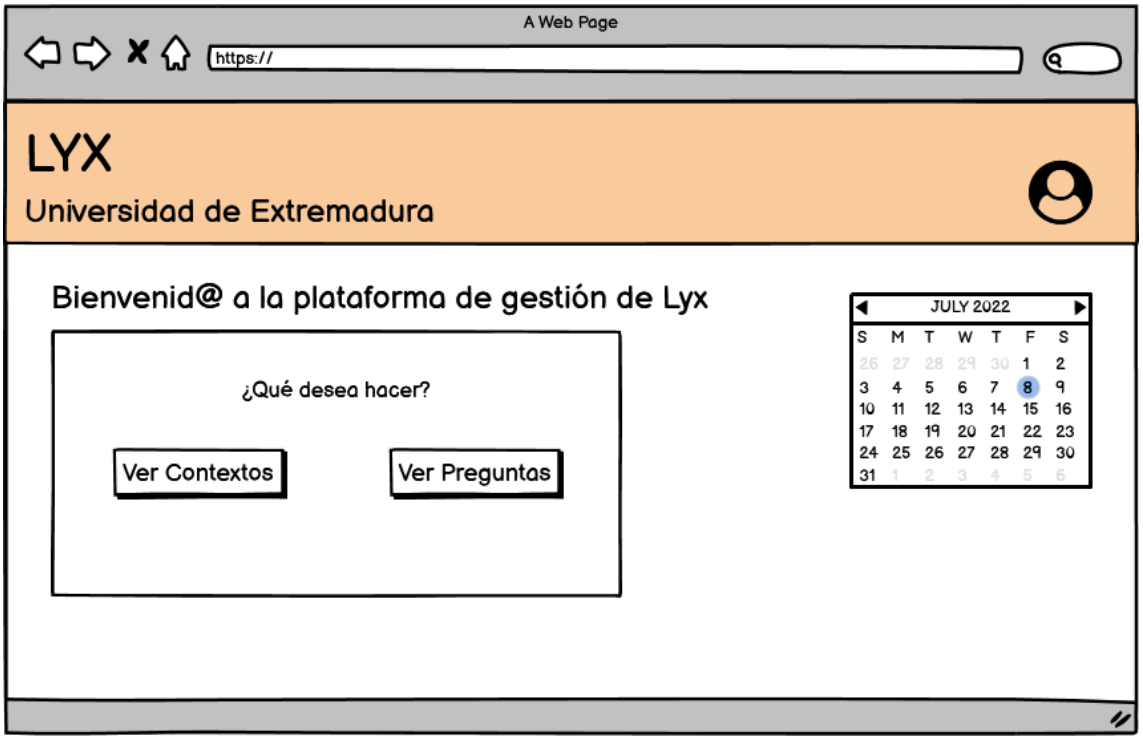

Figura 3: Página principal de la aplicación

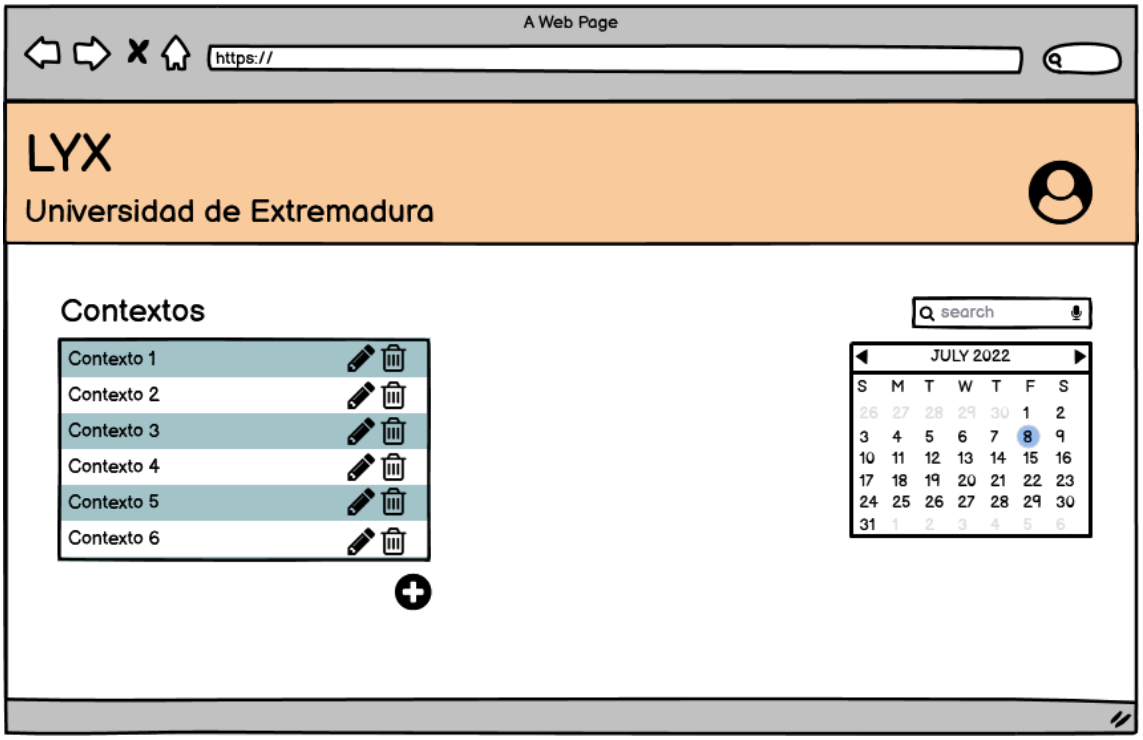

Figura 4: Página que muestra la lista de contextos

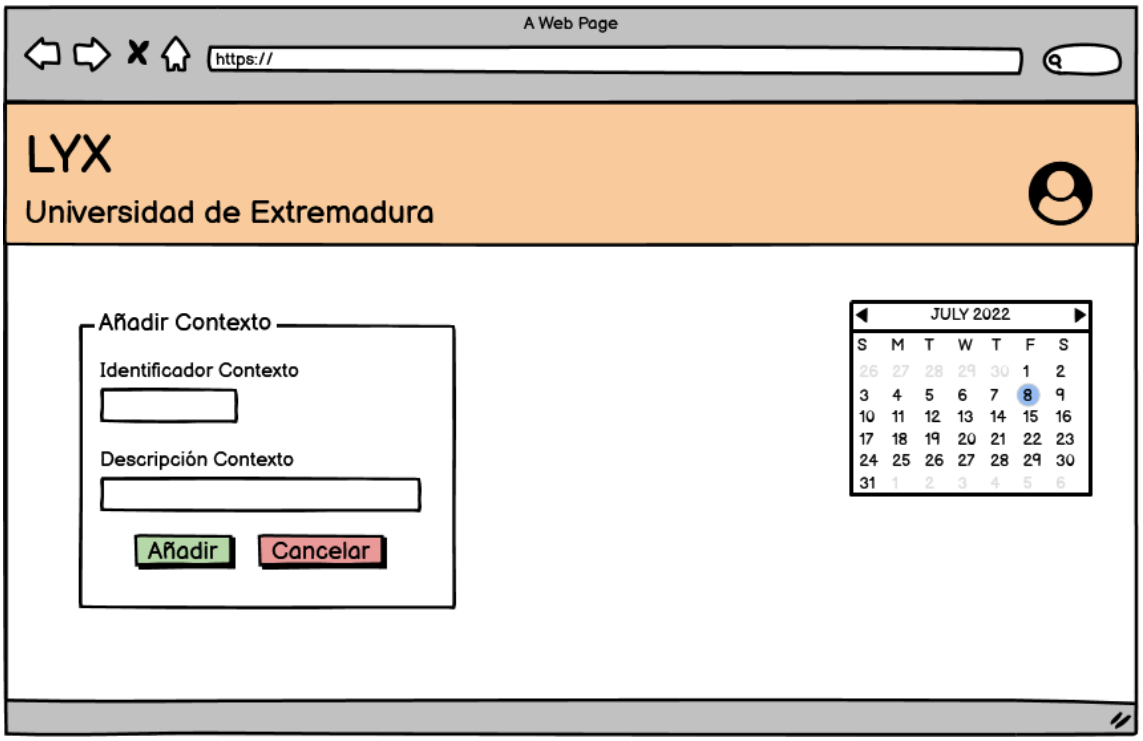

Figura 5: Página para editar/añadir un nuevo contexto

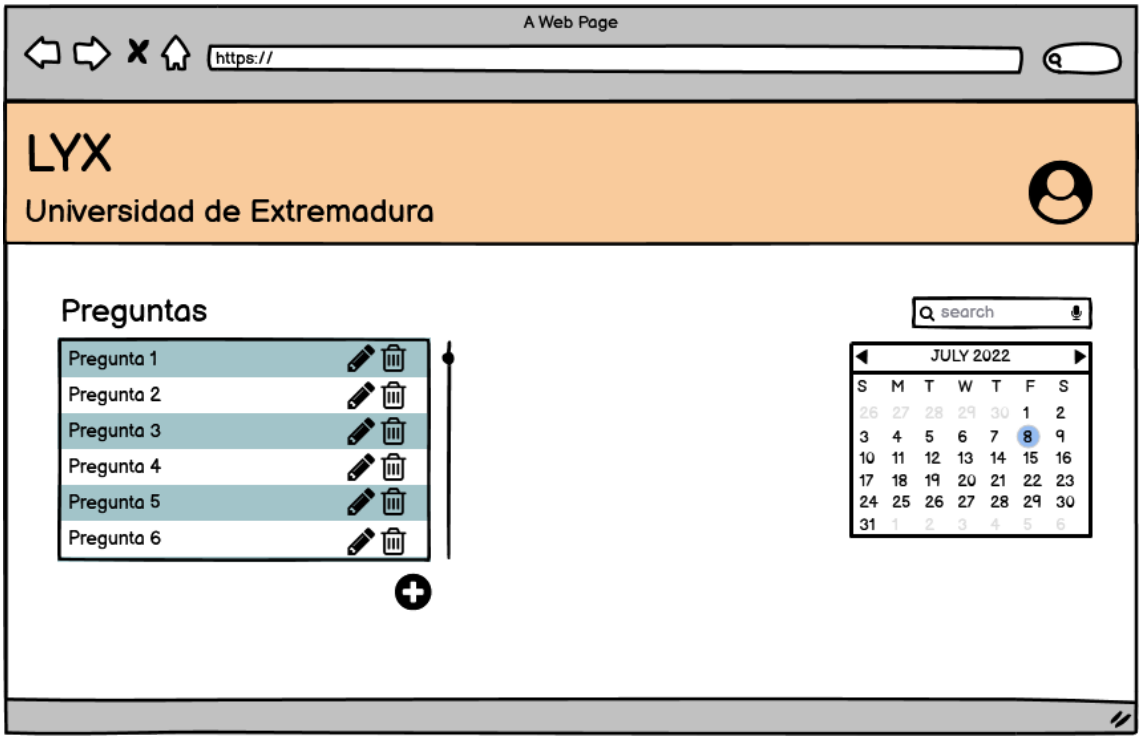

Figura 6: Página que muestra la lista de preguntas

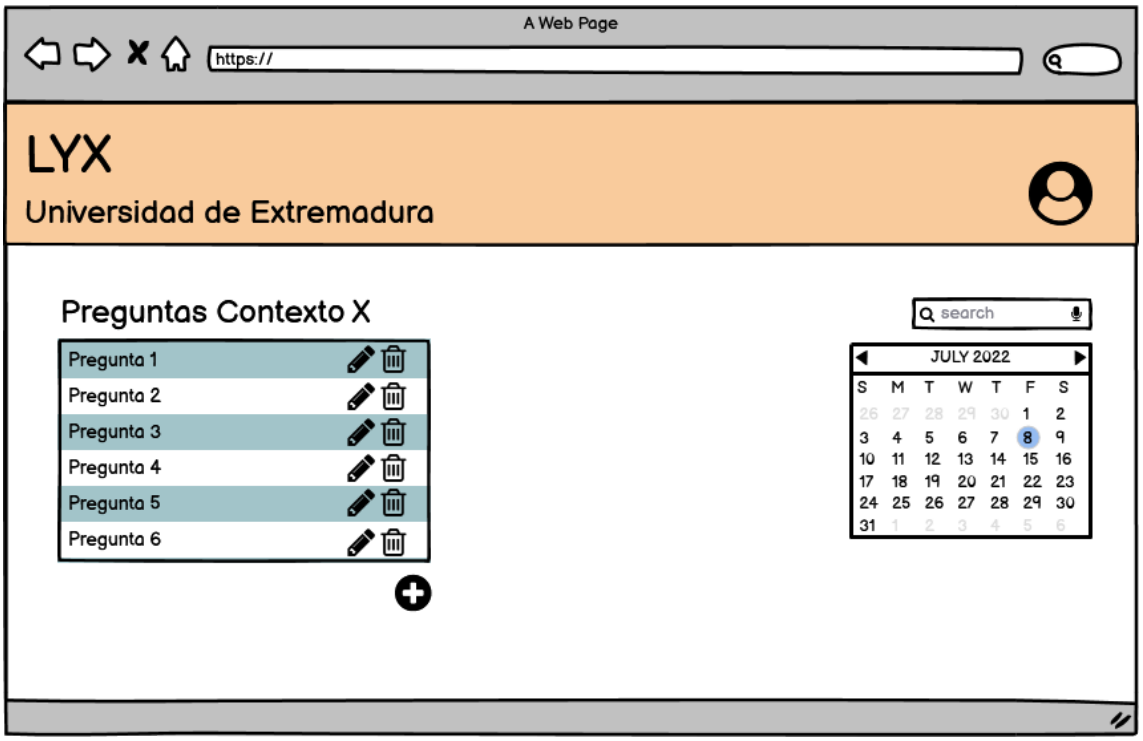

Figura 7: Página que muestra la lista de preguntas por contexto

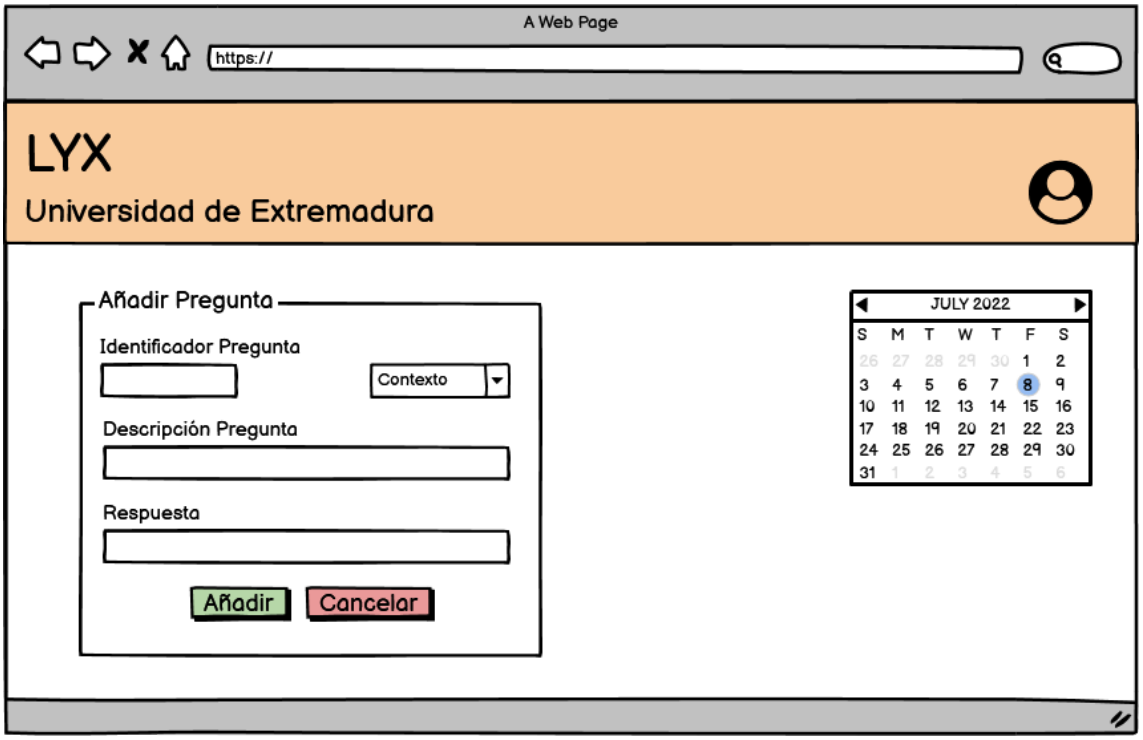

Figura 8: Página para editar/añadir una nueva pregunta

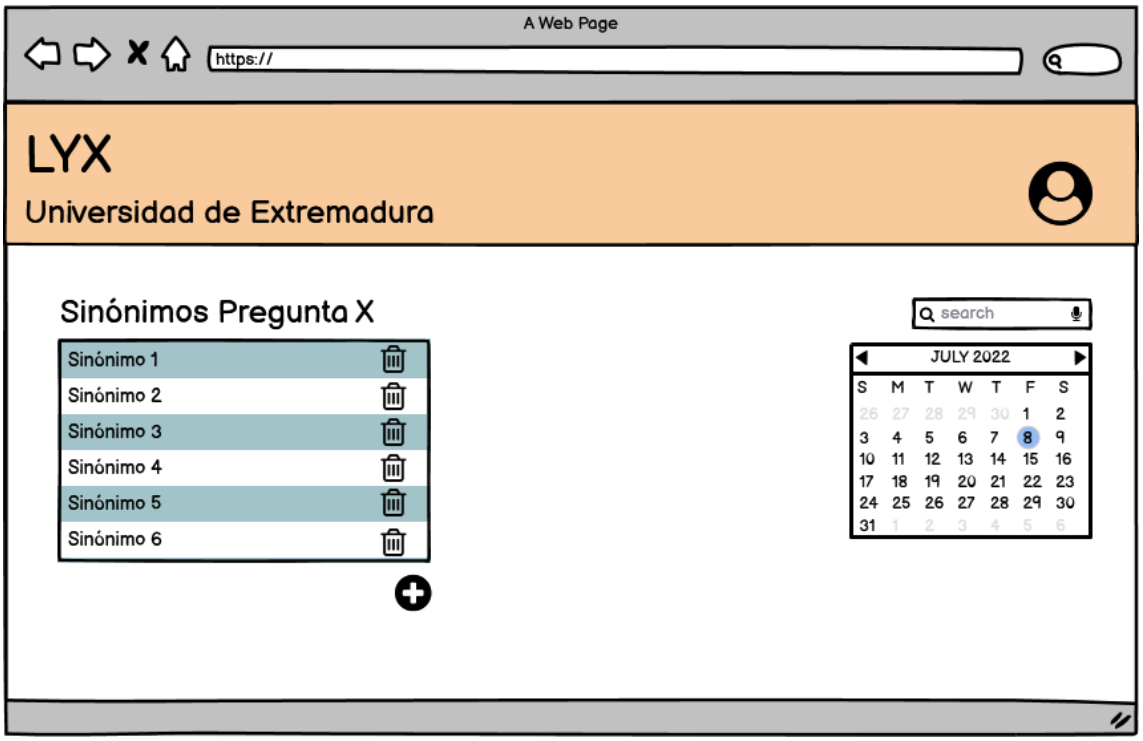

Figura 9: Página que muestra los sinónimos de un intent

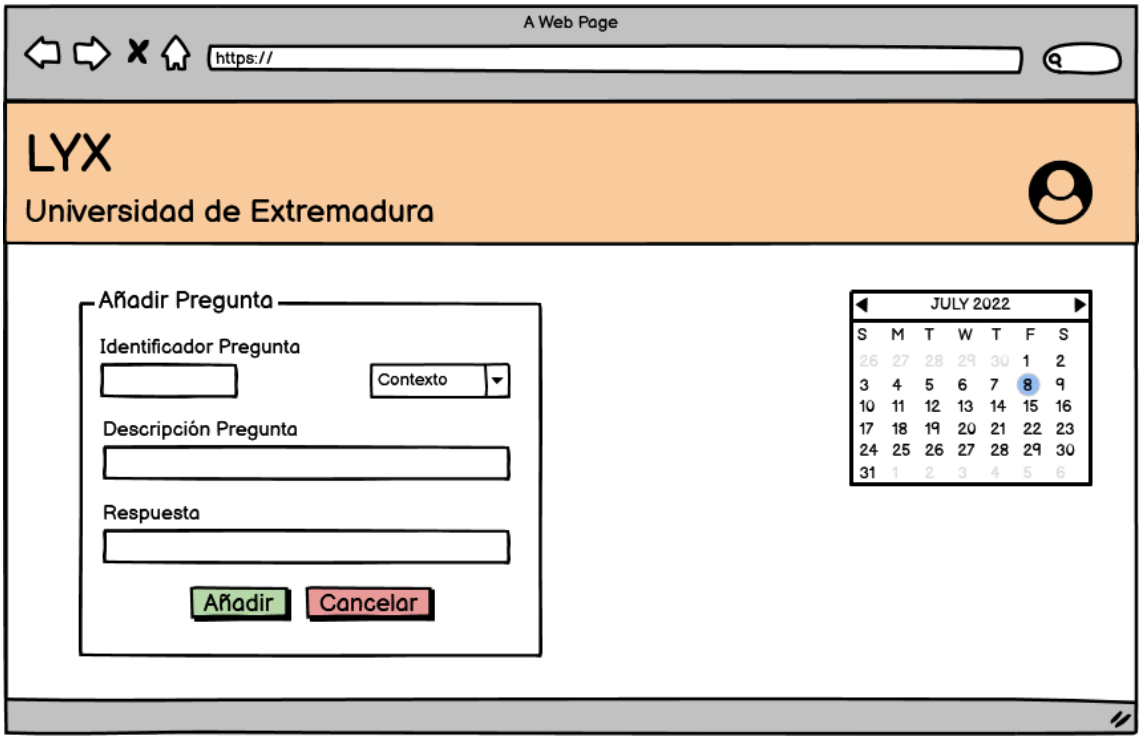

Figura 10: Página para editar/añadir una nueva pregunta

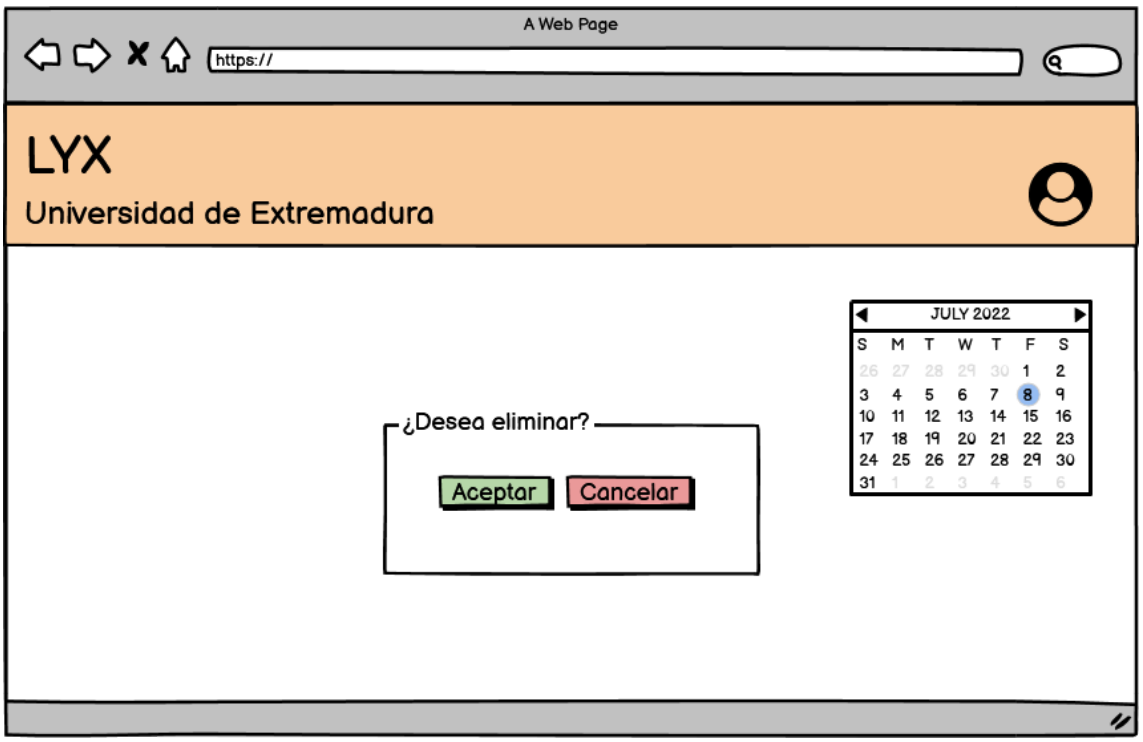

Figura 11: Página para confirmar el borrado

Anexo II: Manual de usuario de la aplicación

En este anexo, se procede a describir con detalle las instrucciones para el uso de la aplicación web, mostrando el flujo de pantallas de la aplicación y las funciones que permite realizar cada una.

Al iniciar la aplicación, la primera página que se muestra es la de bienvenida, que se observa en la figura

Figura 12: Página de bienvenida de la aplicación

Desde esta página, se accede al inicio e sesión y registro (botón de la parte superior derecha), que se realiza mediante una ventana común de *Login* proporcionada por la plataforma Auth0. Esta se muestra en la figura 13

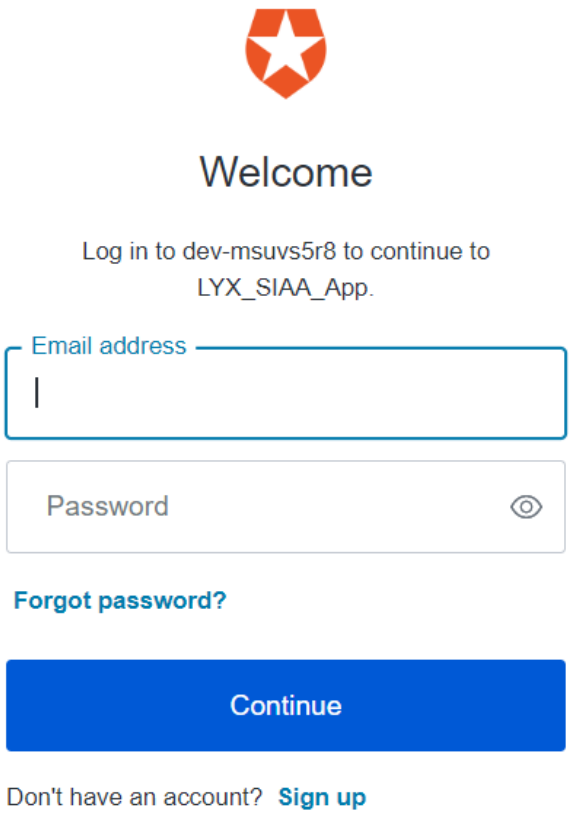

Figura 13: Ventana de inicio de sesión

En la figura 14 se observa el menú principal de la aplicación, desde el que es posible acceder a la gestión de los contextos y demás entidades de datos o simplemente acceder a la consulta de los intents y sus sinónimos. En esta segunda opción no se permite modificar los datos.

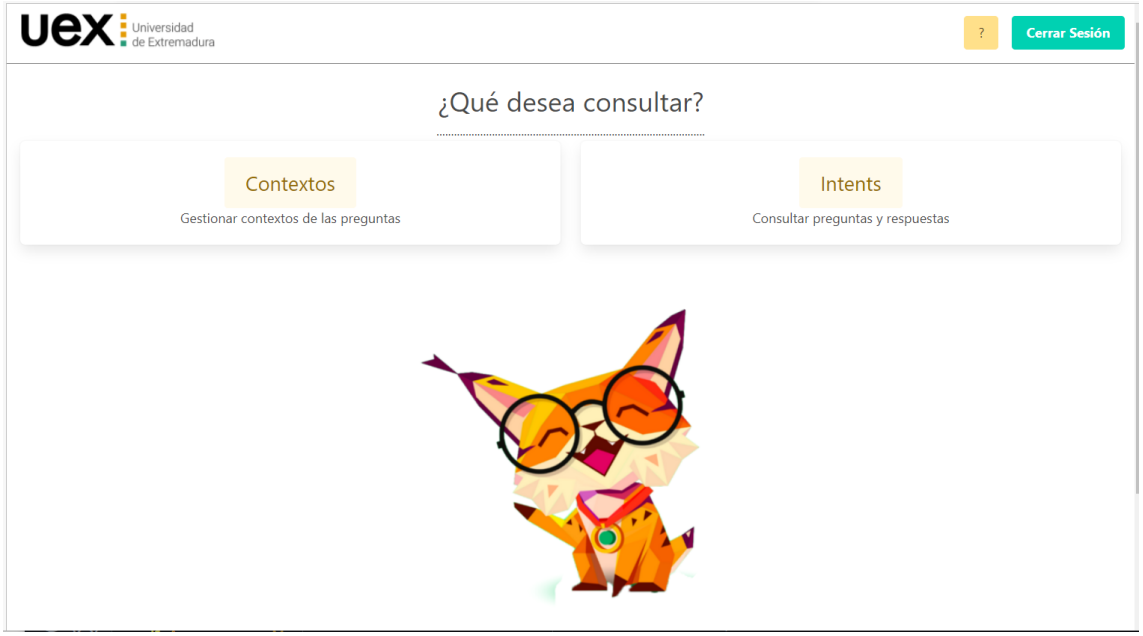

Figura 14: Página de menú inicial de la aplicación

Si se accede la opción de edición, esta es la opción contextos de la página mostrada en la figura 14, se llega a la página de visualización y edición de contextos mostrada en la imagen 15.

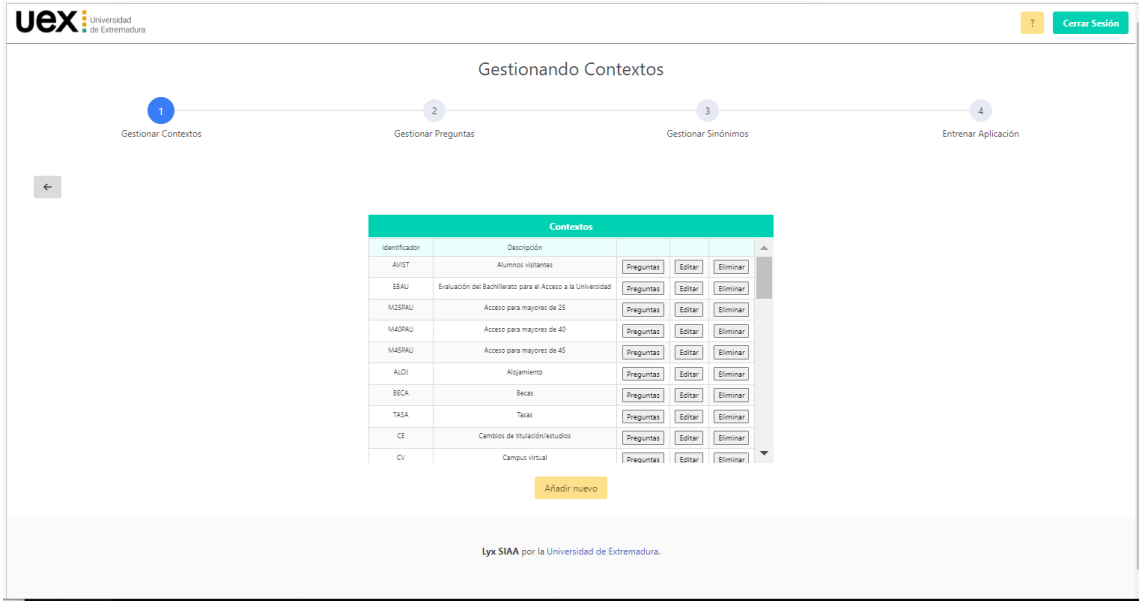

Figura 15: Página de visualización y edición de contextos

Esta constituye la primera fase del proceso de gestión de datos. Las fases de este proceso se muestran en la barra de progreso que aparece en la parte superior de la pantalla, que indica al usuario en que fase del proceso está en todo momento. En esta primera pantalla se pueden consultar los contextos existentes, editarlos, añadir nuevos y eliminarlos.

Si se clica sobre el botón .ªñadir Nuevo", la aplicación dirige al formulario que permite añadir nuevos contextos. Es necesario introducir un identificador de contexto y una descripción, tal como se muestra en la figura 16.

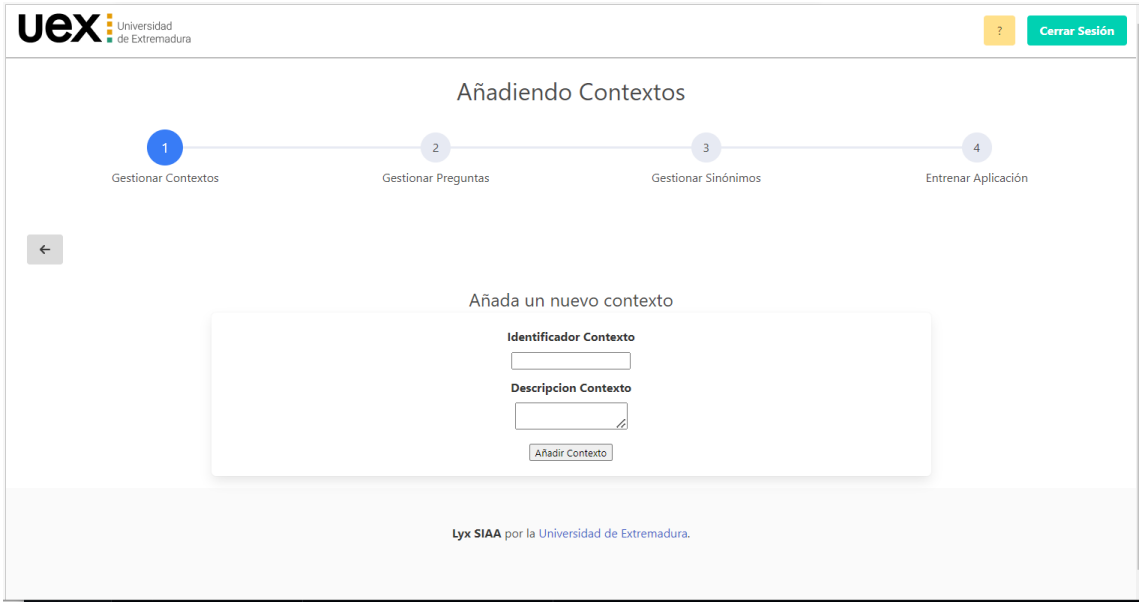

Figura 16: Página mediante la cual se añade un contexto

Si el usuario desea editar un contexto, la aplicación redirige a un formulario similar con los datos preestablecidos del contexto, de los cuales podrá editar solo la descripción, tal como se muestra en la figura 17. Si quiere eliminar un contexto, solo podrá hacerlo si este no contiene intents. Para informar al usuario de si puede eliminar un contexto o no y para confirmar esta decisión, se muestran ventanas de alerta tal como aparece en las capturas 18 y 19

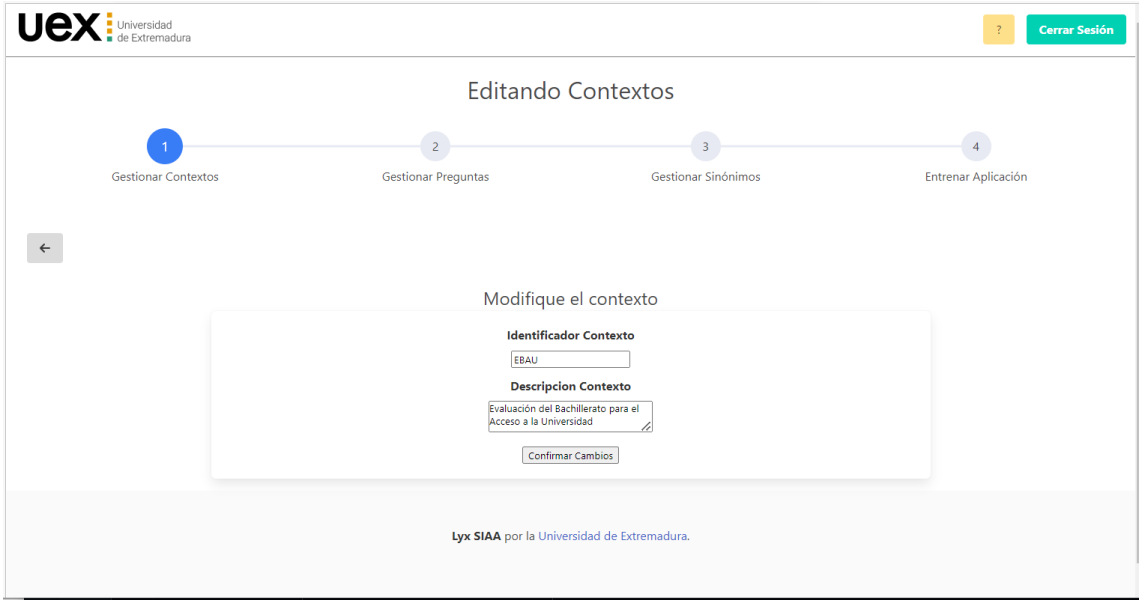

Figura 17: Página de edición de contexto

Tal como se muestra en la figura 18, siempre se pregunta al usuario si desea eliminar.

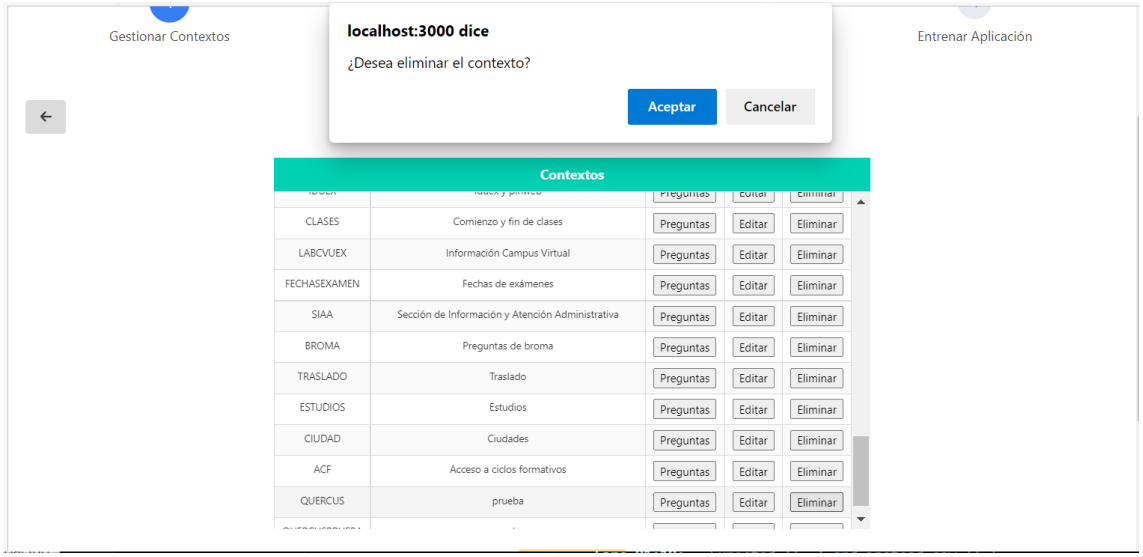

Figura 18: Ventana de confirmación para borrar contexto

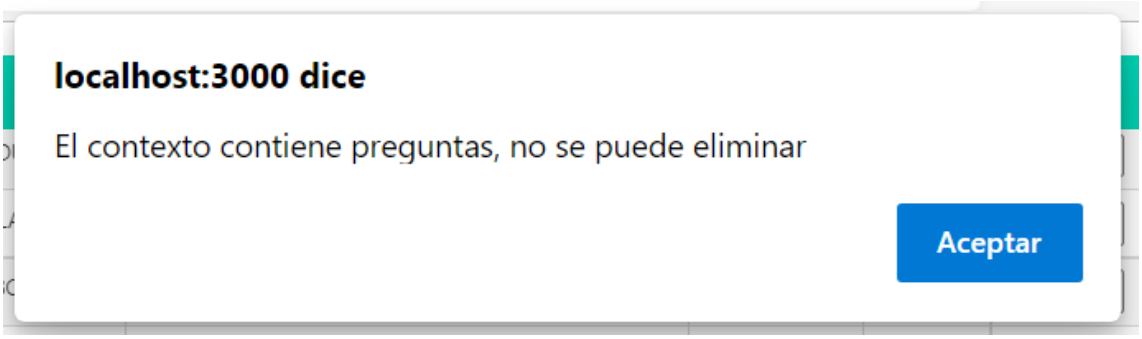

Figura 19: Ventana que informa de que no se puede eliminar un contexto

En el caso de querer eliminar un contexto que contenga preguntas, se avisa mediante la ventana de alerta mostrada en la figura 19.

Si en la página inicial de contextos se selecciona la opción ”Preguntas”de un determinado contexto, se llega a la pantalla que muestra los intents pertenecientes a ese contexto. Esta página constituye la segunda fase del proceso de gestión y se muestra en la figura 20

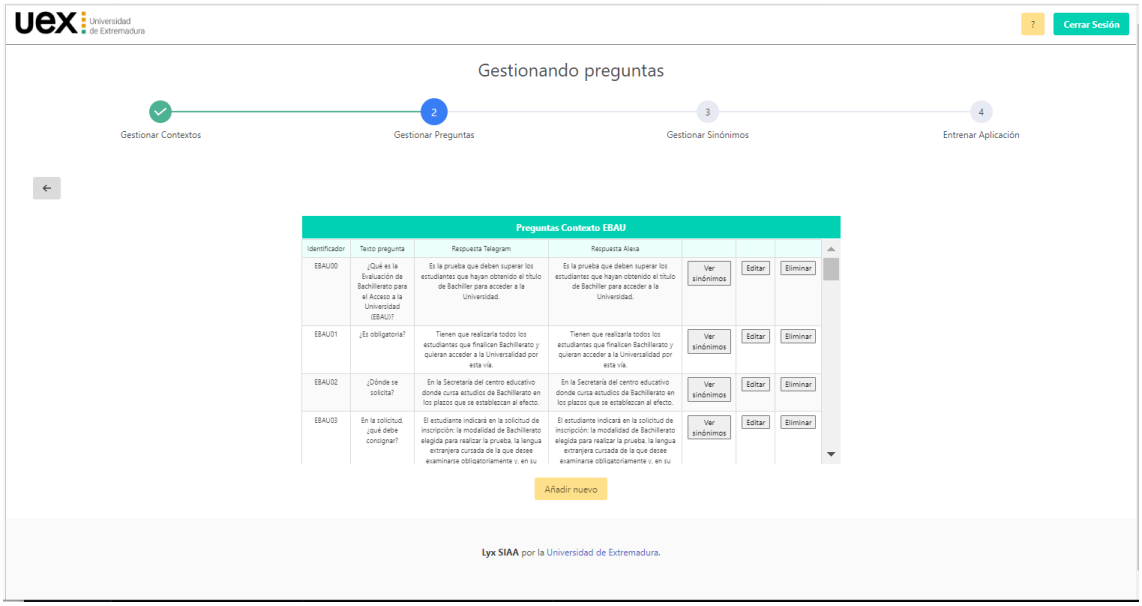

Figura 20: Página que facilita la gestión de los intents de un contexto

Desde la página de gestión de intents es posible consultarlos, editarlos, añadir nuevos y eliminarlos del mismo modo que se realiza para los contextos. Los formularios para añadir y editar intents son similares a los de los contextos y se muestran en las figuras 21

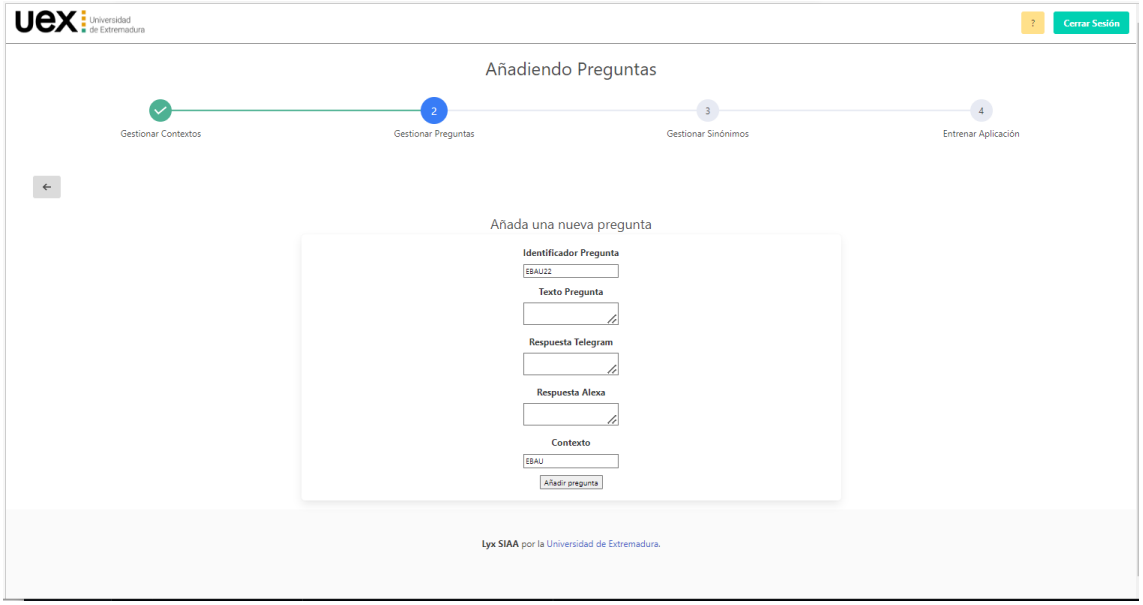

Figura 21: Página mediante la cual se añade un intent

Tal como se observa en la figura 21, los campos requeridos para insertar un nuevo intent son su identificador y el de su contexto, que se establecen automáticamente por defecto desde el contexto que se haya accedido, el texto de la pregunta, texto de respuesta para Telegram y texto de respuesta para Alexa.

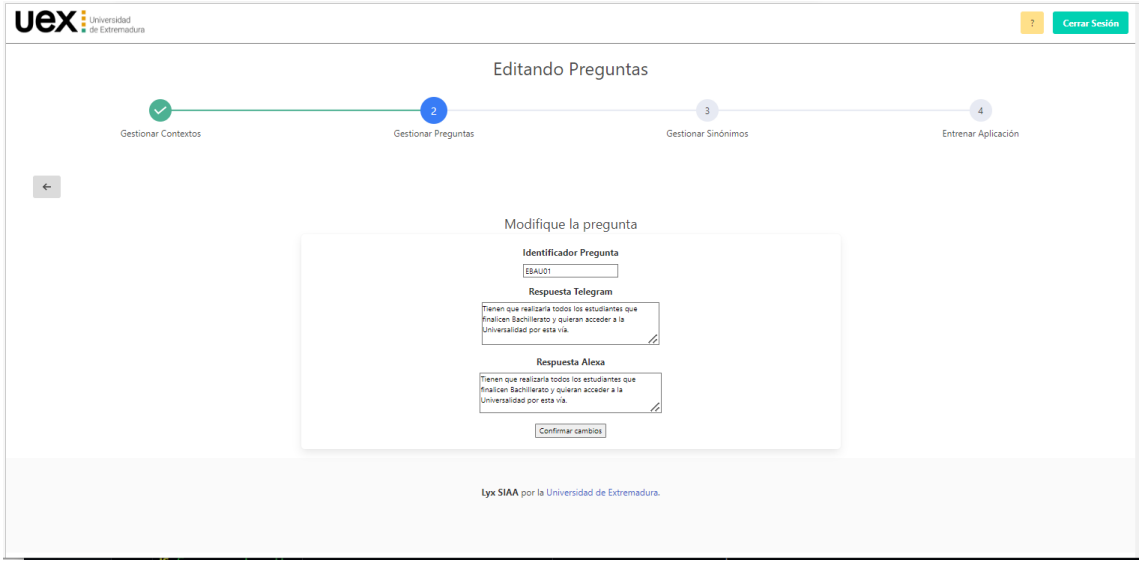

Figura 22: Página de edición de intents

En la figura 22 se observa la página que permite editar un intent. Los campos que se permiten editar, tal como puede comprobarse, son las respuestas de Telegram y Alexa. La edición de otros campos no se autoriza para evitar errores y ambigüedades en los datos.

Si en la pantalla que ofrece la lista de intents se clica sobre el botón ”Sinónimos”para un determinado intent, se accede a la página que muestra sus sinónimos, tercera fase del proceso de gestión. Un ejemplo de esta página es la que se muestra en la figura 23

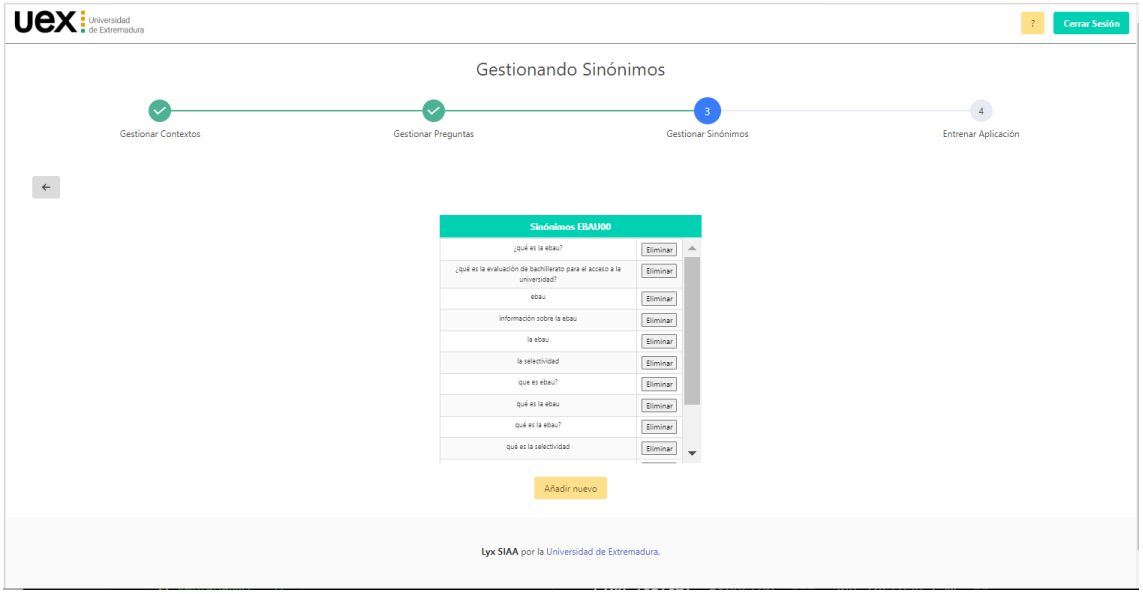

Figura 23: Página que muestra los sinónimos de un intent

Desde la página de gestión de sinónimos de un intent se puede acceder a las opciones de añadir y eliminar sinónimos para ese intent. Si se selecciona .Añadir nuevo”se abre un formulario de un único campo para insertar el texto del sinónimo. Este formulario se muestra en la figura 24

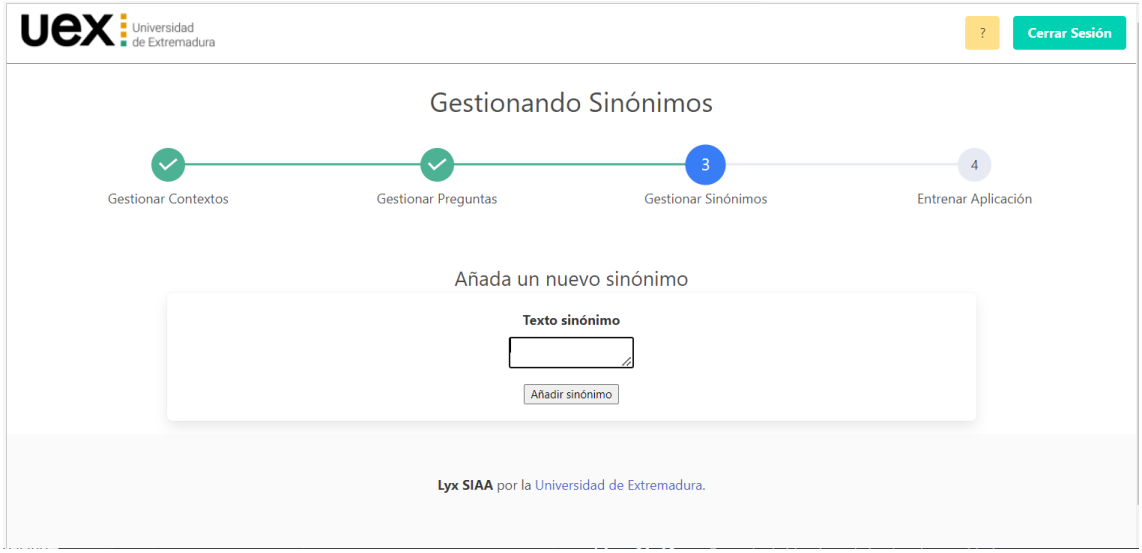

Figura 24: Página que contiene el formulario para añadir un sinónimo

Si se desea eliminar un sinónimo, se abrirá una ventana emergente para que el usuario confirme esta decisión, tal como se muestra en la figura 25

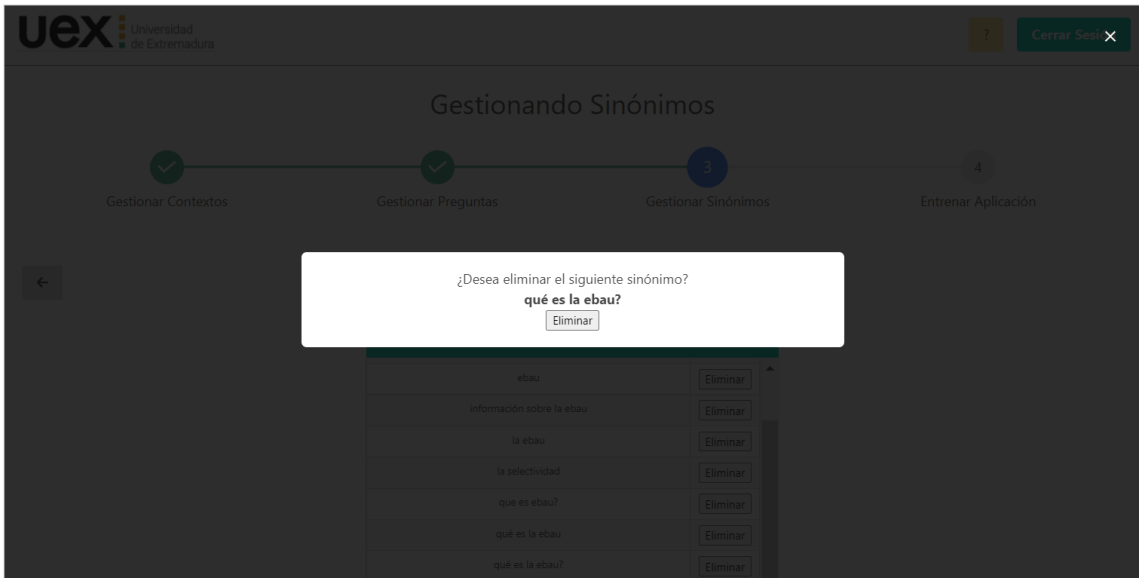

Figura 25: Ventana modal de confirmación para eliminar un sinónimo

Se pueden añadir y eliminar tantos sinónimos como el usuario quiera, pero tras realizar cualquier modificación, para finalizar el proceso es obligatorio y de vital importancia que el usuario pulse el botón ”Finalizar”para llegar a la opción de entrenar el modelo. Si no

se termina editando la aplicación, los cambios llevados a cabo no serán efectivos.

El botón ”Finalizar”, que puede observarse en la figura aparece automáticamente en la página de gestión de sinónimos siempre que se añadan nuevos o se eliminen.

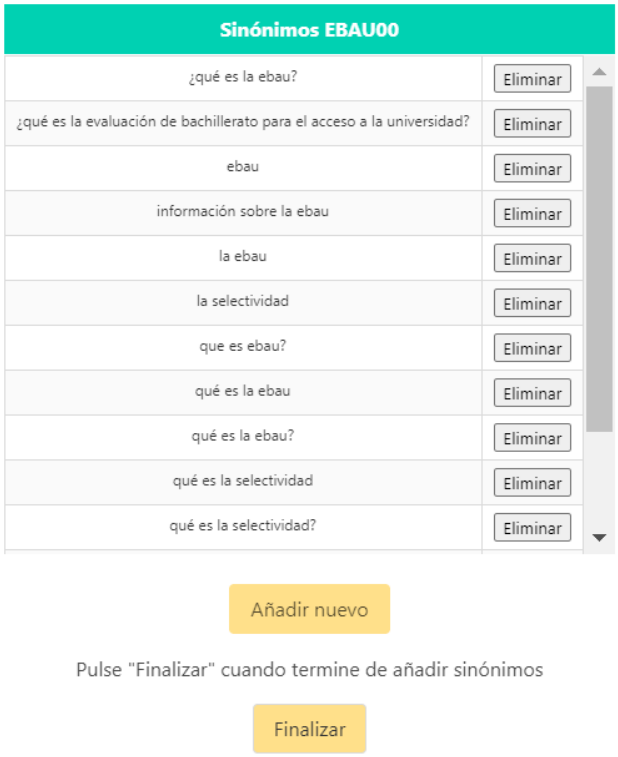

Figura 26: Opción ”Finalizar”tras realizar modificaciones en los sinónimos

Una vez el usuario haya terminado de añadir y eliminar sinónimos y pulse el botón para finalizar, la aplicación conduce a la página de entrenamiento del modelo, que constituye el último paso del proceso. Esta página se muestra en la figura 27

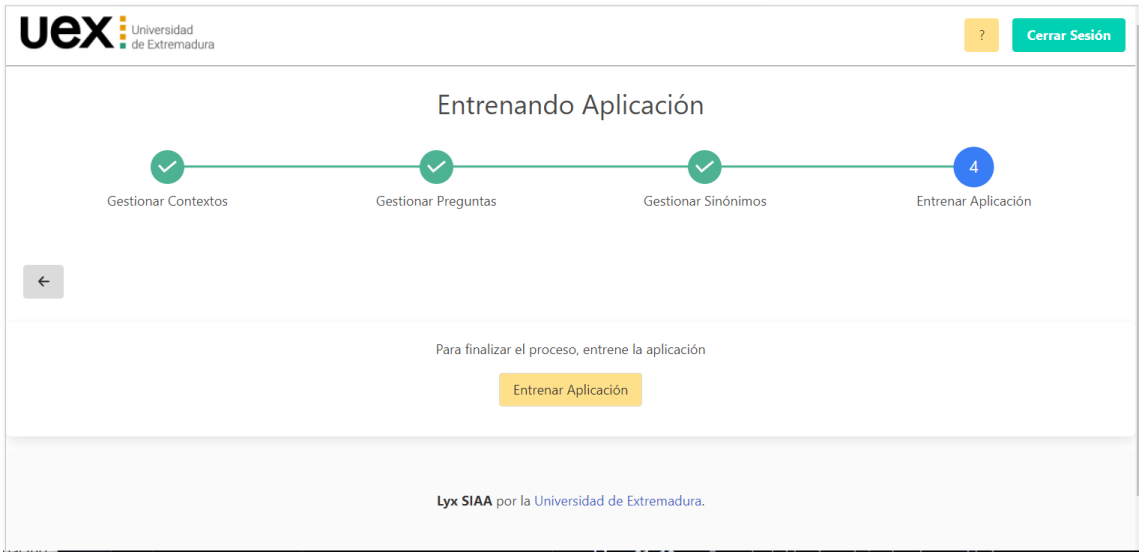

Figura 27: Página de entrenamiento de la aplicación

Una vez que el usuario pulse en la opción de entrenar y aparezca la ventana de alerta informando de que el entrenamiento ha finalizado, los cambios se harán efectivos en el sistema quedando entrenado el modelo y el chatbot actualizado. En este momento, la aplicación vuelve a la pantalla de inicio.

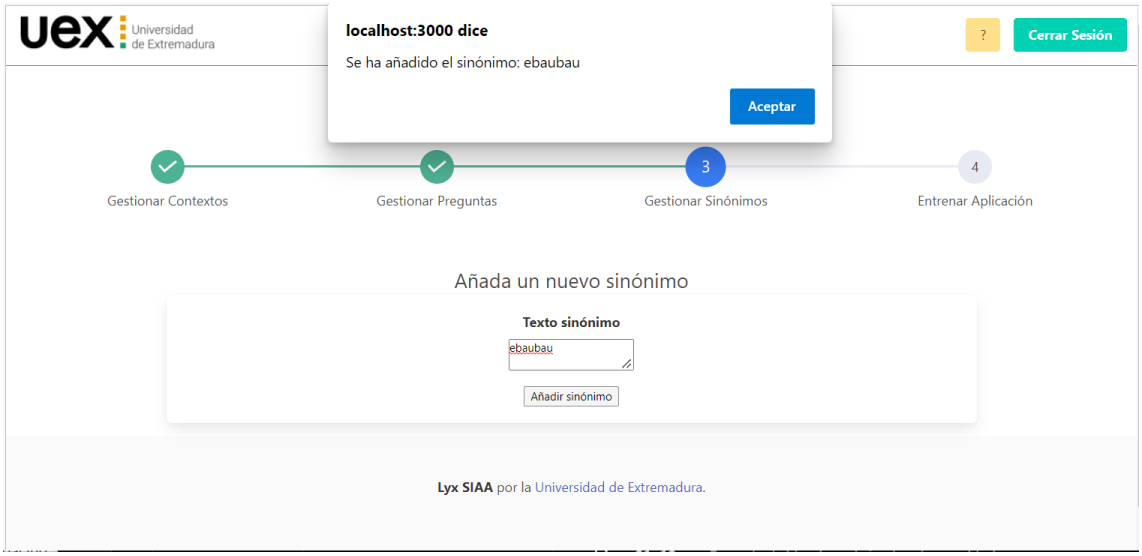

Figura 28: Ventana de alerta mostrada al añadir sinónimo

Se debe destacar que siempre que se hace una modificación, la aplicación informa al

usuario o pide su confirmación mediante ventanas de alerta como la mostrada en la figura 28

Si en la pantalla del menú inicial (figura 14 se selecciona la opción Ïntents", se accede al modo consulta de la aplicación, pudiendo visualizar todos los intents y sus sinónimos pero sin poder editarlos. La pantalla de consulta de intents se puede observar en la figura 29

Figura 29: Pantalla de consulta de intents

Desde esta página se puede acceder a los sinónimos de cada intent, pero como ya se ha comentado, solo en modo visualización sin posibilidad de añadir ni eliminar.

Bibliografía

[1] Why Businesses Are Using Chatbots — by Sam Schmir — Chatbots Magazine. https://chatbotsmagazine.com/why-businesses-are-using-chatbots-ae741312f99e. Última vez visitada 1 de junio de 2022. 1

[2] COVID 19: Un acelerador de la transformación digital — Deloitte Perú — Tecnología. https://www2.deloitte.com/pe/es/pages/technology/articles/COVID19-un-acelerador-de-la-transformacion-digital.html. Última vez visitada 1 de junio de 2022. 1

[3] Joaquín Fournier Guimbao. La transformación digital: un aliado estratégico en la era COVID. Última vez visitada 1 de junio de 2022. 1

[4] (PDF) Interaction and Outeraction: Instant Messaging in Action. https://www.researchgate.net/publication/2431347_Interaction_and_Outeraction_Instant_Messaging_in_Action. Última vez visitada 1 de junio de 2022. 2

[5] Bayan Abu Shawar and Eric Atwell. Chatbots: are they really useful? 2

[6] Ventajas y desventajas de los chatbots: todo lo que necesitas saber — Aivo. https://es.aivo.co/blog/advantages-and-disadvantages-of-chatbots. Última vez visitada 1 de junio de 2022. 2, 5

[7] Francisco José García-Peñalvo. Digital transformation in the universities: Implications of the covid-19 pandemic. *Education in the Knowledge Society*, 22, 2021. 2

[8] La UEx invita a los estudiantes a ”descubrir su potencial.ªnte el próximo curso 2021/22. https://www.regiondigital.com/noticias/especiales/346651-la-uex-invita-a-los-estudiantes-a-descubrir-su-potencial-ante-el-proxim html. Última vez visitada 1 de junio de 2022. 2

[9] Guillermo Pacho. Bots Dependientes del Contexto. 2017. 2

[10] Jaime Álvarado Valiente. Arquitectura orientada a servicios para la creación de bots conversacionales sensibles al contexto, utilizando técnicas de Aprendizaje Automático. 2020. 2

[11] Telegram. https://telegram.com.es/. Última vez visitada 1 de junio de 2022. 2

[12] Amazon Alexa Official Site: What is Alexa? https://developer.amazon.com/es-ES/alexa. Última vez visitada 1 de junio de 2022. 2, 5

[13] La OMS lanza un bot oficial para WhatsApp para mantenerte informado del coronavirus. https://www.xatakamovil.com/aplicaciones/oms-lanza-bot-oficial-para-whatsapp-para-mantenerte-informado-coronavirus. Última vez visitada 1 de junio de 2022. 5

[14] Javier Romero Álvarez. BotUEX: bot conversacional haciendo uso de Minería de Procesos y servicios cloud. 2020. 5

[15] Javier Romero Álvarez. Framework para el desarrollo y despliegue de chatbots multiplataforma. 2022. 5, 6, 7, 13, 15, 16, 48

[16] Siri - Apple (ES), url = https://www.apple.com/es/siri/. https://www.apple.com/es/siri/. Última vez visitada 1 de junio de 2022. 5

[17] Dialogflow. https://dialogflow.cloud.google.com. Última vez visitada 1 de junio de 2022. 5

[18] ¿Qué es Cortana? https://support.microsoft.com/es-ES. Última vez visitada 1 de junio de 2022. 5

[19] Microsoft acquires AI company to make Cortana and bots sound more human - The Verge. https://www.theverge.com/2018/5/21/17375482/microsoft-semantic-machines-acquisition-bots-cortana-human. Última vez visitada 1 de junio de 2022. 5

[20] Microsoft ha adquirido una empresa de IA cada dos meses desde el pasado mes de mayo. https://www.xataka.com/robotica-e-ia/microsoft-ha-adquirido-empresa-ia-cada-dos-meses-pasado-mes-mayo. Última vez visitada 1 de junio de 2022. 5

[21] Microsoft buys ai speech tech company nuance for 19.7 billion - the verge. https://www.theverge.com/2021/4/12/22379414/microsoft-buys-nuance-ai-speech-tech. Última vez visitada 1 de junio de 2022. 5

[22] Alex, el nuevo asistente virtual de la URJC - Universidad Rey Juan Carlos. https://www.urjc.es/todas-las-noticias-de-actualidad/6362-alex-el-nuevo-asistente-virtual-de-la-urjc. Última vez visitada 1 de junio de 2022. 5

[23] Lyx, el lince que guía a los universitarios — Hoy. https://www.hoy.es/extremadura/lince-guia-universitarios-20210630001032-ntvo.html. Última vez visitada 1 de junio de 2022. 6

[24] ¿qué son los bots? ¿son peligrosos? Última vez visitada 1 de junio. 10

[25] Milena Tsvetkova, Ruth García-Gavilanes, Luciano Floridi, and Taha Yasseri. Even good bots fight: The case of wikipedia. *PLOS ONE*, 12:e0171774, 2 2017. 10

[26] A Halfaker and J Riedl. Bots and cyborgs: Wikipedia's immune system. *Computer (Long Beach Calif)*, 45. 10

[27] In Depth Guide Of Chatbot Development 2019 - [Chatbot Guide]. https://www.esparkinfo.com/blog/in-depth-guide-of-chatbot.html. Última vez visitada 1 de junio. 10

[28] Qué son los bots de internet y qué tipos hay. https://www.testdevelocidad.es/2020/07/24/que-son-bots-internet/. Última vez visitada 1 de junio de 2022. 10

[29] Qué es un bot conversacional — Oracle España. https://www.oracle.com/es/chatbots/what-is-a-chatbot/. Última vez visitada 1 de junio de 2022. 10

[30] [U+25B7] ¿qué es un chatbot? cómo funciona y sus beneficios. https://www.inboundcycle.com/diccionario-marketing-online/chatbot. Última vez visitada 1 de junio de 2022. 10

[31] Chatbots, qué son y qué usos podemos darles - MADISON. https://madisonmk.com/chatbots-que-son-y-que-usos-podemos-darles/. Última vez visitada 1 de junio. 10

[32] Tipos de chatbot: ventajas y características - centribal, 2022. Última vez visitada 1 de junio. 11

[33] Eleni Adamopoulou and Lefteris Moussiades. An overview of chatbot technology. *IFIP Advances in Information and Communication Technology*, 584 IFIP:373–383, 2020. 15

[34] AnÁlisis comparativo de dos bases de datos sql y dos bases de datos no sql - pdf descargar libre. 22

[35] Conoce las bases de datos SQL a fondo — Ayuda Ley Protección Datos. https://ayudaleyprotecciondatos.es/bases-de-datos/sql/. Última vez visitada 1 de junio de 2022. 22

[36] Base de datos SQL: ¿cómo funciona y se gestiona esta base de datos? https://www.ticportal.es/glosario-tic/base-datos-sql. Última vez visitada 1 de junio de 2022. 22

[37] Plataforma de datos de Microsoft — Microsoft. https://www.microsoft.com/es-es/sql-server/. Última vez visitada 1 de junio de 2022. 23

[38] Servicios de Bases de Datos — Oracle España. https://www.oracle.com/es/database/. Última vez visitada 1 de junio de 2022. 24

[39] SQLite Home Page. https://www.sqlite.org/index.html. Última vez visitada 1 de junio de 2022. 26

[40] Features Of SQLite. https://www.sqlite.org/features.html. Última vez visitada 1 de junio de 2022. 26

[41] MySQL. https://www.mysql.com/. Última vez visitada 1 de junio de 2022. 28

[42] Introduction to NoSQL • Martin Fowler • GOTO 2012 - YouTube. https://www.youtube.com/watch?v=qI_g07C_Q5I. Última vez visitada 1 de junio de 2022. 30

[43] WHITEPAPER: BBDD NO SQL acenswhitepaper. 31

[44] MongoDB: La Plataforma De Datos Para Aplicaciones — MongoDB. https://www.youtube.com/watch?v=qI_g07C_Q5I. Última vez visitada 1 de junio de 2022. 31

[45] Archivo:Tabla comparativa de SGBD NoSQL.png - Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Archivo:Tabla_comparativa_de_SGBD_NoSQL.png. Última vez visitada 1 de junio de 2022. 32, 34, 35

[46] Apache Cassandra — Apache Cassandra Documentation. https://cassandra.apache.org/_/index.html. Última vez visitada 1 de junio de 2022. 33

[47] Redis. https://redis.io/. Última vez visitada 1 de junio de 2022. 35

[48] Redis para principiantes - BI Geek Blog. https://blog.bi-geek.com/redis-para-principiantes/. Última vez visitada 1 de junio de 2022. 35

[49] Las 7 mejores tecnologías para el desarrollo de API REST — Saasradar. https://saasradar.net/desarrollo-api-rest/. Última vez visitada 1 de junio de 2022. 36

[50] Express - Infraestructura de aplicaciones web Node.js. https://expressjs.com/es/. 36

[51] Welcome to Flask — Flask Documentation (2.1.x). https://flask.palletsprojects.com/en/2.1.x/. 37

[52] FastAPI. https://fastapi.tiangolo.com/. 38

[53] The web framework for perfectionists with deadlines — Django. https://www.djangoproject.com/. Última vez visitada 1 de junio de 2022. 39

[54] Tecnologías web comunes del lado cliente — Microsoft Docs. https://docs.microsoft.com/es-es/dotnet/architecture/modern-web-apps-azure/common-client-side-web-technologies. Última vez visitada 1 de junio de 2022. 40

[55] Angular. https://angular.io/. Última vez visitada 1 de junio de 2022. 41

[56] MVC - Glosario — MDN. https://developer.mozilla.org/es/docs/Glossary/MVC. Última vez visitada 1 de junio de 2022. 41

[57] Angular vs React vs Vue Español. Cual Elegir? - Openinnova. https://www.openinnova.es/angular-vs-react-vs-vue-espanol-cual-elegir/. Última vez visitada 1 de junio de 2022. 41, 42, 44

[58] React – Una biblioteca de JavaScript para construir interfaces de usuario. https://es.reactjs.org/. Última vez visitada 1 de junio de 2022. 42

[59] Vue.js - The Progressive JavaScript Framework — Vue.js. https://vuejs.org/. Última vez visitada 1 de junio de 2022. 43

[60] Hospedaje e implementación de asp.net core blazor webassembly — microsoft docs. https://docs.microsoft.com/es-es/aspnet/core/blazor/host-and-deploy/webassembly?view=aspnetcore-6.0. Última vez visitada 1 de junio de 2022. 44

[61] Qué es Frontend y Backend, diferencias y características - Platzi. https://platzi.com/blog/que-es-frontend-y-backend/. Última vez visitada 1 de junio de 2022. 52, 53

[62] sqlite3 — DB-API 2.0 interface for SQLite databases — Python 3.10.5 documentation. https://docs.python.org/3/library/sqlite3.html. 61

[63] Inicio rápido: API REST y bibliotecas cliente del SDK de Language Understanding (LUIS) - Azure — Microsoft Docs. https://docs.microsoft.com/es-es/azure/cognitive-services/luis/client-libraries-rest-api?tabs=windows&pivots=programming-language-python. Última vez visitada 1 de junio de 2022. 61

[64] Singleton Design Pattern — Implementation - GeeksforGeeks. https://www.geeksforgeeks.org/singleton-design-pattern/. Última vez visitada 1 de junio de 2022. 62

[65] Python Design Patterns. https://python-patterns.guide/. Última vez visitada 1 de junio de 2022. 62

[66] Bulma: Free, open source, and modern CSS framework based on Flexbox. https://bulma.io/. Última vez visitada 1 de junio de 2022. 67

[67] Qué es wireframing. https://www.unir.net/ingenieria/revista/que-es-wireframing/. Última vez visitada 1 de junio de 2022. 70

[68] Balsamiq. Rapid, Effective and Fun Wireframing Software — Balsamiq. https://balsamiq.com/. Última vez visitada 1 de junio de 2022. 70

[69] Auth0: Acceso seguro para todos. Pero no para cualquiera. https://auth0.com/es?utm_source=google&utm_medium=cpc&utm_term=-g-auth0&pm=true&utm_campaign=spain-es-brand-auth0&gclid=CjwKCAjw2f-VBhAsEiwAO4lNeGEaCopIM0qgniWgOzSFhSHbQ8O_vnb5L7lhn6YmHXPNlLt3akOYZRoCJDMQAvD_BwE. Última vez visitada 1 de junio de 2022. 70

[70] Postman API Platform — Sign Up for Free. https://www.postman.com/. Última vez visitada 1 de junio de 2022. 74

[71] La importancia de los test de usuarios en un diseño - Novicell. https://www.novicell.es/es/blog/importancia-de-los-test-de-usuarios-para-tu-diseno/. Última vez visitada 1 de junio de 2022. 76