Universidad de Valladolid

Escuela de Ingeniería Informática

Trabajo Fin de Grado

Grado en Ingeniería Informática
Mención Ingeniería del Software

Implementación de un agente conversacional mediante Microsoft Bot Framework y Azure Cognitive Services como herramienta de divulgación de la condición de la mujer en el contexto de la igualdad de género

Realizado por

Mun de Souza, Se Hee

Dirigido por

Gonzalo Tasis, Margarita

A las mujeres que siguen en pie y a las que ya no están,
a las que me acompañaron y a las que me apoyaron.

Agradecimientos

Quiero agradecer a las mujeres que me han apoyado y acompañado hasta aquí.
Mi hermana, mi madre, mi abuela, las profesoras que tuve, mi tutora... Vuestra fuerza y dedicación ha sido crucial para mí.

Resumen

La desigualdad y discriminación de género es un desafío a superar en las sociedades actuales, donde sus poblaciones, que aun contando con un acceso privilegiado a la información, a menudo sentencian este problema como inexistente o se niegan en reiteradas ocasiones a contrastar los datos a los que acceden para formar una opinión razonada.

Con este proyecto pretende abordar la desinformación al respecto de la desigualdad y discriminación de género utilizando fuentes veraces y públicas para la difusión de información actualizada sobre esta problemática.

Para lograr este objetivo, se hará uso de tecnologías de vanguardia como Microsoft Bot Framework y soluciones de inteligencia artificial como Microsoft LUIS. Estas herramientas permitirán generar experiencias de conversación naturales, facilitando discusiones relevantes sobre la temática. También se utilizarán herramientas que permitan el tratamiento y visualización de datos estadísticos, potenciando la presentación de los datos así como su asimilación.

El proyecto se desarrolla siguiendo el marco de trabajo ágil Scrum, iniciado en la etapa de planificación del proyecto, seguido por el análisis del sistema, su diseño, implementación y realización de pruebas.

Abstract

Inequality and discrimination based on gender are challenges that current societies must overcome. Despite having privileged access to information, populations often dismiss this problem as non-existent or repeatedly refuse to contrast the data they have access to in order to form a reasoned opinion.

This project aims to tackle the pervasive issue of misinformation surrounding gender inequality and discrimination by harnessing reliable and publicly available sources to disseminate accurate and up-to-date information to the public.

To achieve this objective, the project will leverage cutting-edge technologies such as Microsoft Bot Framework and artificial intelligence solutions as Microsoft LUIS. These tools will enable the generation of natural conversation experiences, facilitating meaningful discussions on the topic. Additionally, the project will employ data processing and statistical visualization tools, empowering the presentation of relevant data.

The project will adhere to the agile Scrum framework, starting with the project planning, followed by system analysis, design considerations, implementation and testing procedures.

Tabla de Contenidos

Lista de Figuras

Lista de Tablas

Lista de Algoritmos

Capítulo 1

Introducción

1.1 Introducción

En nuestra vida cotidiana hacemos un uso constante de tecnologías, muchas de ellas, basadas en la inteligencia artificial, que pasan inadvertidas en nuestra rutina. En los últimos años ha habido un gran crecimiento de estas tecnologías, cuyo consumo se realiza en gran parte a través de dispositivos móviles de manera continua. Empleamos estas tecnologías sobre todo para el asesoramiento, instrucción, resolución de dudas y entretenimiento.

Este proyecto tiene como objetivo la construcción de un bot conversacional con la finalidad de utilizarse como asesoramiento y contará con la habilidad de desarrollar diálogos con personas en plataformas móviles.

1.2 Motivación

Este trabajo está motivado por la actual situación de discriminación de género que sufren las mujeres alrededor de todo el planeta, y más concretamente España, país con un alto nivel de igualdad legislativa.

Cada vez son más los países que se suman a reformas legales, haciendo sus políticas más transparentes y claras en lo que respecta la defensa de los derechos humanos; sin embargo, estas reformas casi nunca son acompañadas por cambios en la mentalidad social. Las opiniones y valores ético-morales de cada individuo son cimentadas por toda la información colectiva recibida y aceptada por uno mismo, la cual no siempre es reflejo imparcial de la realidad; razón por la cual existen muchas personas que no reconocen la existencia de ningún tipo de discriminación de género.

Como seres humanos, lo que nos caracteriza es la mutabilidad, podemos pasar a estar mejor informados y llegar a cambiar de opinión con respecto a esta problemática. Este proyecto propone ayudar a la difusión de información correcta y actualizada de manera que esta sirva como referencia válida para el asesoramiento a cada individuo.

1.3 Objetivos

1.3.1 Objetivos del sistema

El propósito de este proyecto es la creación de un bot conversacional con herramientas de inteligencia artificial, capaz de entender el lenguaje natural utilizado por el usuario y desarrollar diálogos completos.

Los diálogos se centrarán en la temática de la discriminación de género y se empezará una conversación a partir de un tema que plantee el propio usuario. El sistema accederá a datos veraces e imparciales que proporciona el Instituto Nacional de Estadística.

El sistema estará disponible para todo tipo de públicos, siempre mayores de 18 años y se podrá utilizar a través de un ordenador, donde la herramienta será una aplicación de escritorio, o bien desde dispositivos Android o iOS, donde se ejecutará como aplicación móvil.

1.3.2 Objetivos personales

Los objetivos personales que se desean desarrollar son los que se describen a continuación:

- Conocer brevemente diversas herramientas utilizadas en el campo de la inteligencia artificial en general.
- Formación y posterior aplicación de las herramientas proporcionadas por Microsoft para la creación de chatbots y sus soluciones para la aplicación de inteligencia artificial.
- Formación y aplicación de herramientas que se utilizan para el manejo y visualización de datos.
- Comprender, interiorizar y poner en práctica de manera correcta las prácticas fundamentales de Scrum, uno de los marcos de trabajo más populares hoy en día.

1.4 Contexto tecnológico

En este apartado se realiza una breve introducción no exhaustiva a los componentes de arquitectura más relevantes de los bots conversacionales, así como aquellos elementos que conforman un diálogo, cuyo mecanismo es interesante conocer antes de llevar a cabo el proyecto.

1.4.1 Lenguaje natural

El lenguaje natural se refiere a la forma en que los seres humanos nos comunicamos [1] de manera escrita, oral o signada. El lenguaje como lo entendemos evoluciona a cada día y como personas podemos producir y entender el lenguaje de forma natural, sin embargo describir su funcionamiento formal nos sería de lo más complejo. Para que las máquinas puedan comprender el lenguaje natural, se debe de realizar un proceso de modelización matemática [2]. A grandes rasgos, para el procesamiento del lenguaje, se pueden distinguir los modelos lógicos basados en gramáticas y los modelos probabilísticos basados en *corpus* o datos.

La Lingüística es la ciencia que estudia el lenguaje, esto es su morfología, sintaxis, semántica, pragmática y fonética. Utilizando herramientas matemáticas, Noam Chomsky estudió diversos formalismos gramaticales desde un punto de vista lingüístico, sentando las bases de la lingüística moderna en los años 50 y permitiendo la aparición de modelos lógicos que pretendían reflejar la estructura lógica del lenguaje [3].

Los modelos lógicos de procesamiento del lenguaje natural son creados por lingüistas, quienes escriben reglas de reconocimiento de patrones estructurales basándose en formalismos gramaticales concretos.

El estudio del lenguaje ha evolucionado hacia la Lingüística Computacional, que aúna la Lingüística y herramientas computacionales; de esta manera en los años 90 surgieron los modelos probabilísticos del lenguaje natural, que empezaron a reemplazar los métodos formales basados en reglas estructurales.

En un modelo probabilístico del lenguaje [4], los lingüistas recogen una gran colección de textos escritos, denominado *corpus*, y a partir de su análisis definen una distribución de probabilidad para cada unidad lingüística (palabras, letras u oraciones); de esta manera se puede predecir la siguiente unidad lingüística en un contexto dado, sin recurrir a reglas gramaticales explícitas. Esta probabilidad asociada a cada término, puede ser calculada tras el análisis del *corpus* o ser calculada a partir del conocimiento aprendido por la máquina.

En este proyecto será necesario analizar el lenguaje natural que utiliza el usuario y comprender su significado para poder satisfacer su petición de información, para esta tarea se utilizarán diversas tecnologías que se desarrollarán en los siguientes capítulos.

1.4.2 Estructura de un diálogo

Entendemos por diálogo a la transmisión de información entre un emisor y receptor a través de un medio. En un diálogo ambas partes se turnan en sus respectivos roles, por lo que un diálogo es una secuencia de turnos, normalmente ordenada, donde cada parte empieza una nueva transmisión cuando la otra ha finalizado [5].

Un turno en un diálogo puede consistir en una palabra, una frase o un conjunto de frases. A la vez que los turnos crean un diálogo, los diálogos construyen una conversación.

1.4.3 Bots

La palabra bot es un acortamiento de la palabra robot, que viene a ser un programa informático que ejecuta sentencias o acciones, normalmente repetitivas de manera mucho más rápida en comparación con un ser humano.

La idea de un bot inteligente nació en los años 50 con Alan Turing, cuando empezó a considerarse la existencia de máquinas que tuviesen la capacidad de pensar. Con ello, desarrolló el Test de Turing, el cual pretende constatar si una máquina posee comportamientos inteligentes, lo cual la haría indistinguible de un ser humano [6].

Las funciones de los bots evolucionaron con los años y la tecnología, hoy en día no solo realizan tareas repetitivas, sino que son empleados para interactuar con personas en diversos escenarios y

desarrollar diálogos complejos.

Son utilizados por multitud empresas, dado que permiten generar valor a su negocio así como reducir drásticamente costes asociados al mantenimiento de recursos humanos. Igualmente son ampliamente utilizados por individuos con fines maliciosos [7].

Algunas funcionalidades de los bots son:

- Búsqueda masiva y análisis de información en la web.
- Aportar respuestas rápidas al usuario.
- Mantener conversaciones con usuarios ya sean simples o complejas.
- En videojuegos, pueden desempeñar el rol de otro jugador.
- Automatizar acciones: p.e. Realizar correcciones ortográficas automáticas en un documento de texto o gestionar llamadas entrantes a un call center.
- Explotación de vulnerabilidades en aplicaciones web.
- Recolección de cuentas de correo electrónico para la difusión de SPAM o estafas.

Los bots se pueden utilizar en básicamente cualquier ámbito y con un propósito positivo o negativo: Como en la educación, sanidad, e-commerce, marketing, política, etc.

1.4.4 Agentes conversacionales

Un agente conversacional o chatbot es un sistema software que opera como una interfaz entre seres humanos y una aplicación software, utilizando para ello el lenguaje natural escrito u oral. Los chatbots, agentes conversacionales o bots conversacionales son bots diseñados para simular el diálogo de un usuario con un ser humano.

Estos agentes pueden trabajar sobre un dominio abierto, donde las conversaciones no tienen propósito específico y los temas pueden cambiar rápidamente; o bien en un dominio cerrado, donde el conocimiento del agente es limitado y centrado en una temática o propósito específico, lo cual los capacita para poder dar una respuesta o solución más acertada y/o elaborada para un ámbito en concreto [8].

El usuario puede utilizar un bot mediante un canal, es decir, un medio por el cual el bot es accesible, estos pueden ser aplicaciones móviles como Facebook, Skype o Twitter, aplicaciones web o de escritorio.

El funcionamiento visto desde alto nivel, de un bot conversacional generalista se puede resumir de la siguiente manera [9]:

1. Mediante un canal, el usuario realiza una entrada de datos en formato de texto, imagen, audio o vídeo en función de las características del bot y su capacidad para procesar los datos de entrada.
2. El bot realiza un reconocimiento de la entrada del usuario y la interpreta.

3. Si necesario, el bot se puede conectar con otros servicios, acceder a APIs o bases de datos para poder generar una respuesta.

4. El bot evalúa las posibles respuestas, selecciona la respuesta más acertada y genera una salida al usuario.

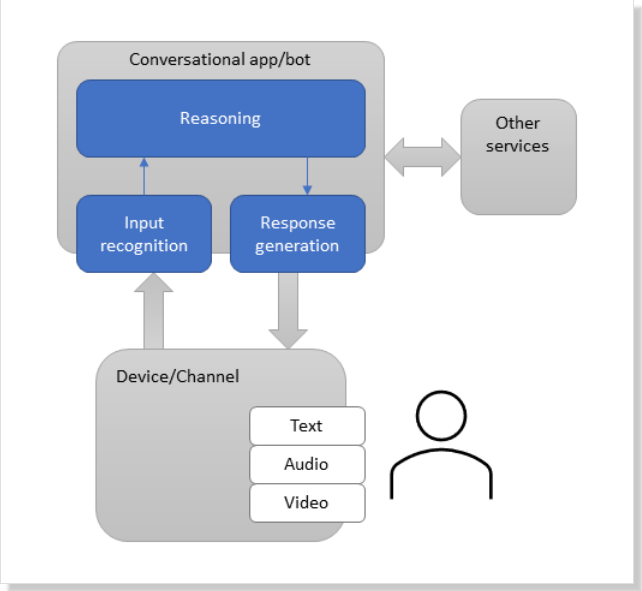

Figura 1.1: Funcionamiento de un bot

Se pueden diferenciar y agrupar los bots en función de varias características: su nivel de inteligencia, las acciones que permite realizar, la entrada de datos permitida, calidad de la experiencia de usuario, ámbito de uso, etc. En función de su inteligencia existen dos tipos principales de chatbots: Los orientados a tareas o basados en reglas y los basados en datos y predictivos o también llamados chatbots inteligentes [10].

Chatbots basado en reglas

Son bots que tienen una única función o propósito; generan respuestas automatizadas y predefinidas a las consultas del usuario, estas consultas deben estar muy bien estructuradas para que el bot pueda comprender la petición del usuario. Su capacidad es muy limitada y por lo general su respuesta depende de las palabras clave que obtenga del usuario.

Estos bots son ampliamente utilizados para tareas de soporte que no impliquen gran complejidad en la generación de la respuesta y en cierta medida pueden utilizar procesamiento del lenguaje natural para aportar al usuario una experiencia de conversación fluida y natural.

A continuación, se ilustra un ejemplo gráfico, donde se puede observar la estructura predefinida que debe de tener un diálogo entre el chatbot y el usuario, para que el bot pueda generar una respuesta o acción correcta.

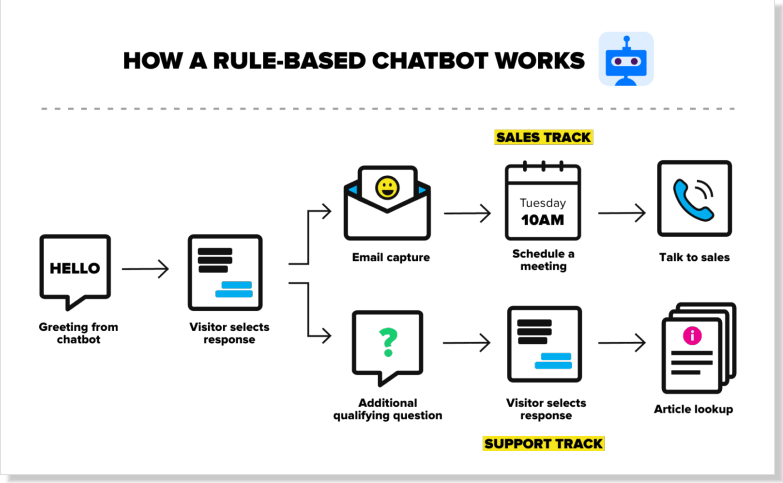

Figura 1.2: Ejemplo del funcionamiento de un bot basado en reglas

Chatbots inteligentes

Se clasifican como inteligentes porque utilizan en gran medida herramientas de aprendizaje automático y procesamiento del lenguaje natural. También son llamados asistentes virtuales y son mucho más avanzados que los chatbots orientados a tareas o basados en reglas. Pueden utilizar inteligencia predictiva y analítica para personalizar el perfil del usuario, adelantarse a sus necesidades, entender el contexto de su consulta así como la intención real de la misma y aprender sobre el usuario y sus necesidades mediante la interacción con él.

A continuación, se ilustra un ejemplo de la interacción de un usuario con un chatbot inteligente. Se puede observar que el usuario, quién se comunica de forma oral, no sigue ninguna estructura predefinida de interacción para expresar lo que necesita al bot, sino que este, en cada turno del diálogo, va realizando preguntas al usuario para así poder disponer de toda la información necesaria para llevar a cabo la petición del usuario [11].

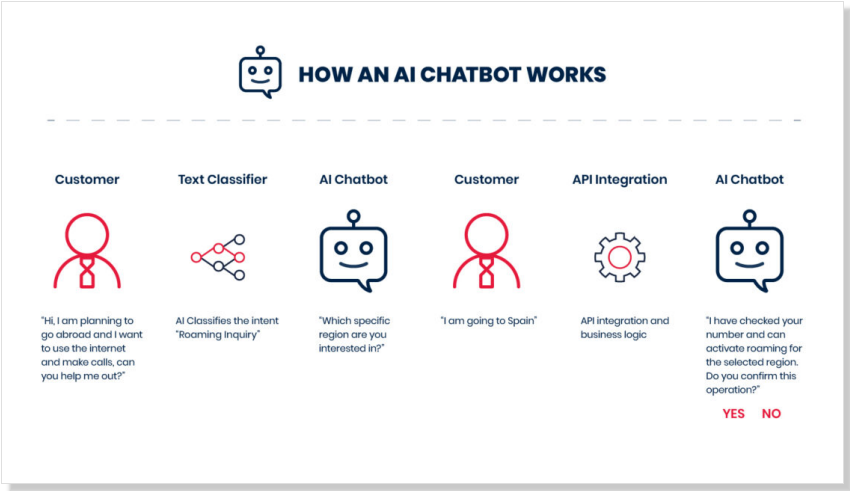

Figura 1.3: Ejemplo del funcionamiento de un bot inteligente

El chatbot a desarrollar en este proyecto, será un chatbot inteligente, dado que se emplearán herramientas de procesamiento del lenguaje natural y aprendizaje automático o *machine learning* que constituirán el *core* que permitirá entender la intención del usuario, procesarla y generar una respuesta acorde.

1.4.5 Machine learning

El *machine learning* o aprendizaje automático es una rama de la inteligencia artificial y un método de análisis de datos que automatiza la construcción de modelos analíticos, permite identificar patrones y tomar decisiones con la mínima intervención humana. A grandes rasgos, permite que las máquinas puedan aprender sin ser programadas para realizar tareas específicas, a medida que los modelos son expuestos a nuevos datos, éstos pueden adaptarse de forma independiente, es decir, aprenden de cálculos previos para producir decisiones y resultados confiables y repetibles [12].

El proceso de creación de un sistema de aprendizaje automático requiere cierto conocimiento de aprendizaje automático o ciencia de datos. Por ello se hará uso de los servicios "Cognitive Services" de Microsoft. Estos servicios proporcionan una parte o todos los componentes de una solución de aprendizaje automático: datos, algoritmos y modelos entrenados. Están diseñados para requerir conocimientos generales sobre los datos sin necesidad de tener experiencia con el aprendizaje automático.

En este proyecto, se hará uso de determinados servicios cognitivos, mientras que los modelos se deberán de crear y entrenar con datos que deberán ser recolectados de manera independiente.

1.4.6 Arquitectura de un chatbot

A grandes rasgos, los agentes conversacionales que utilizan procesamiento del lenguaje natural cuentan con mecanismos comunes en su arquitectura, es decir, elementos encargados de procesar la entrada del usuario y generar una salida al mismo, a continuación se citarán los más relevantes que intervienen en la gestión del diálogo [13,14].

Comprensión del lenguaje natural

La unidad de comprensión del lenguaje natural o en inglés NLU (natural language understanding), es el primer componente que examina el mensaje del usuario, y convierte la entrada de datos en una acción, también denominada **intención** o **intent**.

Los **intents** son la interpretación realizada sobre una frase de entrada o **utterance**. De manera que si un agente tiene como dominio de conocimiento la meteorología en España, en la siguiente frase del usuario "Quiero salir a pasear. ¿Debería llevar chubasquero?"; se podría estimar que la intención más probable del usuario es "obtener la meteorología de la ubicación actual", por lo tanto esta frase se podría tratar dentro del **intent** o intención citada.

La unidad de comprensión del lenguaje natural puede manejar campos de información llamados **entities** o entidades que el usuario debe de informar en el diálogo. Por ejemplo, si el agente conversacional tiene como objetivo facilitar la compra de viajes en tren; en la siguiente oración:

"Cómprame un billete a Madrid", la entidad `ciudad_destino` se rellenaría con la palabra Madrid. Una vez este campo es rellenado, el gestor del diálogo puede actualizar el estado de la conversación y pasar a la siguiente estado o acción.

En este proyecto la comprensión del lenguaje natural se logrará haciendo uso de los servicios cognitivos de Microsoft, en concreto LUIS.

Gestor del estado del diálogo

Es el componente encargado de mantener la información intercambiada con el usuario y permite a la política del diálogo, decidir la siguiente acción a realizar y el siguiente estado a presentar al usuario.

En este proyecto la gestión de los diálogos y sus estados se logrará encapsulando y definiendo la lógica de cada uno y los datos que manejan.

Política del diálogo

Cuando el estado del diálogo se actualiza, la política del diálogo se desencadena, encargada de decidir la siguiente acción a realizar teniendo en cuenta el nuevo estado. Esta entidad puede seleccionar una acción externa (conectarse con APIs o servicios de terceros) para satisfacer la petición del usuario, o seleccionar una acción interna (que permite modificar el desempeño del agente).

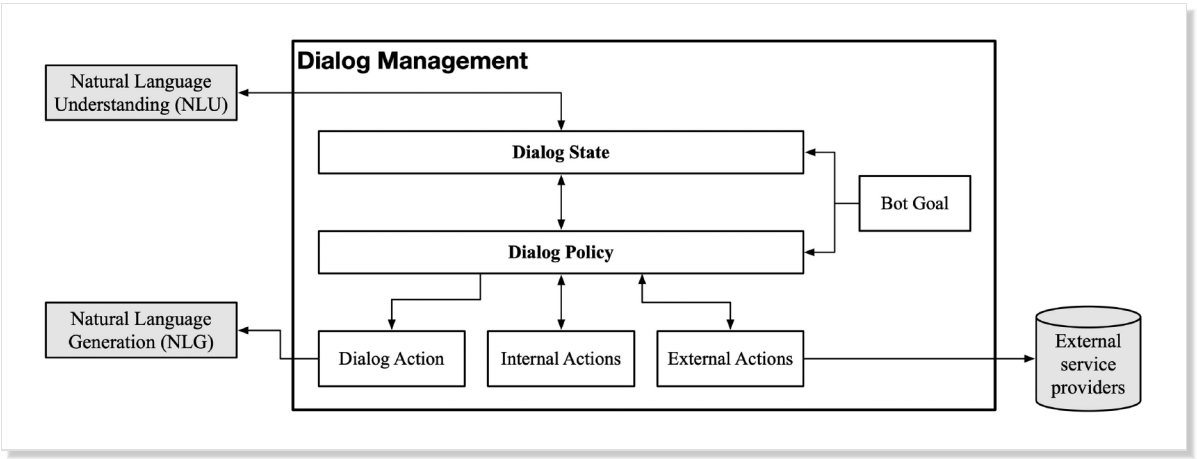

Figura 1.4: Arquitectura de un chatbot

1.4.7 Gestión del diálogo

Se pueden distinguir tres enfoques para abordar la gestión del diálogo y con ello el flujo de la conversación [13]:

- **Enfoque "manual" y determinista:** Los estados que pueden manejar estos gestores así como la política del diálogo son definidas por los desarrolladores y expertos del dominio del

problema mediante reglas codificadas manualmente. Este enfoque se puede modelar mediante una perspectiva **determinista.**

- **Enfoque probabilístico:** En lugar de definir reglas manualmente, como en el anterior enfoque, este tipo de gestor aprende las distintas reglas que debe aplicar, a partir de un gran conjunto de conversaciones reales entre humanos llamado *corpus*. Esta gestión del diálogo permite que el flujo de la conversación sea bastante más natural y al mismo tiempo menos predecible. Junto con el enfoque determinista, se hará uso de este planteamiento para que el chatbot pueda producir diálogos naturales con el usuario.
- **Enfoque híbrido:** Es una combinación de los enfoques anteriores, donde se hace uso de reglas codificadas manualmente, así como técnicas del enfoque probabilístico.

Enfoque determinista

Mediante este enfoque se pueden construir diálogos en formato de árbol, donde las ramas existen basadas en respuestas anteriores. Se pueden distinguir las siguientes gestiones de un diálogo [15]:

1. **Autómatas finitos:** Se predefinen una serie de estados que representan los estados en los que se puede encontrar la conversación en cada punto. Existe un número delimitado de estados, reglas y transiciones a otros estados que se lleva a cabo bajo ciertas condiciones. El flujo del diálogo es rígido y no admite desviaciones, por lo que son mayormente útiles para llevar a cabo tareas muy bien estructuradas (p.e. solicitar de manera secuencial datos al usuario), por lo general, en estos sistemas es el agente el que toma la iniciativa en el diálogo, iniciando el mismo o pidiendo en cada turno del diálogo, información al usuario.

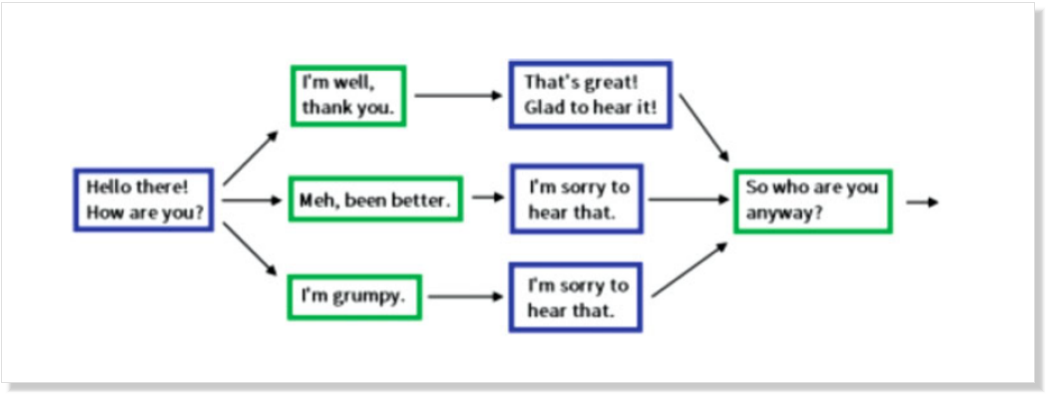

Figura 1.5: Ejemplo de un diálogo modelado como autómata finito

2. **Basado en reglas:** Léxicamente se utilizan como las reglas de producción en la Programación Lógica. Tienen un formato de *Cabeza :- Cuerpo* donde el Cuerpo constituye una serie de precondiciones y la Cabeza las acciones que se llevarán a cabo. Las precondiciones se instanciarán con el diálogo del usuario o se desencadenarán una vez reconocido un patrón. Estos chatbots son más flexibles que los anteriores (p.e. Reglas para ajustar el tema de la conversación una vez esta es desviada por el usuario).

3. **Basado en plantilla:** Es un autómata finito integrado con un modelo de datos, lo que permite al gestor del diálogo hacer un seguimiento de los "campos" informados, pudiendo contener datos o estar vacíos. Los campos se pueden rellenar en cualquier orden, por lo que el agente puede realizar preguntas al usuario en cualquier orden sin la necesidad de seguir una estructura fija en el diálogo, lo que le aporta más flexibilidad, al mismo tiempo, permite que el usuario pueda tomar cierta iniciativa en el diálogo (que éste empiece el diálogo o realice preguntas).

   Estos flujos de diálogo están controlados únicamente por la entrada de datos del usuario (oraciones) y los campos de datos. Gran parte de las soluciones de chatbots comerciales pueden utilizar este enfoque, como DialogFlow de Google, Amazon Lex o LUIS de Microsoft. A continuación, se ilustran ejemplos de plantillas, en el dominio de pedidos de pizza a domicilio.

`Pídeme una pizza {tamaño_pizza} de {sabor_pizza}`

`Mi nombre es {nombre_usuario}`

`Enviar a la dirección {direccion}`

   Cada palabra entre corchetes corresponde a un campo de información que el gestor debe identificar y manejar para actualizar el estado del diálogo.

   El usuario puede tomar la iniciativa de iniciar el diálogo e informar todos los campos que el agente necesita en una sola oración, o sino el propio agente puede hacer preguntas clave que suscite la respuesta por parte del usuario, para de esta manera rellenar los campos necesarios. Para nuestro ejemplo, mediante la pregunta del agente `¿De qué sabor quieres la pizza?` se intentaría rellenar el campo `sabor_pizza`.

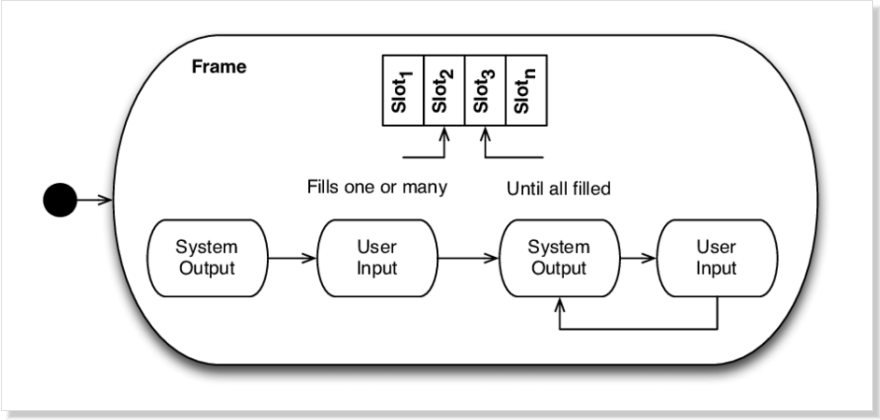

Figura 1.6: Gestión del diálogo mediante plantilla

Enfoque probabilístico

Estos enfoques suelen emplear un sistema de convicción, donde los estados se activan en base a la evidencia recolectada. Se pueden distinguir las siguientes gestiones de un diálogo mediante este

enfoque [15]:

1. **Basado en ejemplos:** El sistema es entrenado para proporcionar las respuestas con las que se entrenó, siempre que la petición del usuario concuerde con una oración del *corpus* utilizado como ejemplo en el entrenamiento. Por ejemplo, se puede haber entrenado el agente con el siguiente ejemplo:

   ```
   Usuario:  ¿Qué tal?
   ```

   ```
   Agente:  Estupendamente, ¿y tú?
   ```

   Por lo tanto si el usuario empieza una oración de la misma manera, el agente le responderá con el ejemplo proporcionado; por supuesto existen algoritmos que se ejecutan para determinar cuál es el ejemplo a utilizar. Este enfoque posee varias limitaciones, entre ellas, el tratamiento de errores (entrada de datos incorrecta por parte del usuario). Este enfoque se aplicará en el proyecto utilizando la plataforma que pone a disposición LUIS, donde se alimentará al modelo con un corpus de datos que en nuestro caso será redactado manualmente.

2. **Redes bayesianas:** Modelan la distribución estadística de eventos como variables o declaraciones del usuario. Constan de un grafo dirigido acíclico y tablas de probabilidad condicionales para cada nodo. Se utilizan en entornos intrínsecamente ruidosos, donde puede haber distorsión en el habla o en la escritura. Estas estructuras están diseñadas por expertos del dominio; tanto el grafo como las probabilidades iniciales se deben de definir de antemano por el experto.

3. **Cadena de Markov, procesos de decisión de Markov (MDP) y procesos de decisión de Markov parcialmente observables (POMDP):**

   Se utilizan para modelar niveles de incertidumbre; los modelos de Markov permiten optimizar el diálogo dado que el sistema tiene la capacidad de escoger la acción óptima a realizar en función de las métricas definidas, a las cuales se asocian refuerzos positivos o negativos. Estas métricas pueden variar en función del propósito del agente, se podría considerar para ello, la duración del diálogo, la satisfacción del usuario, tiempos de espera u otros.

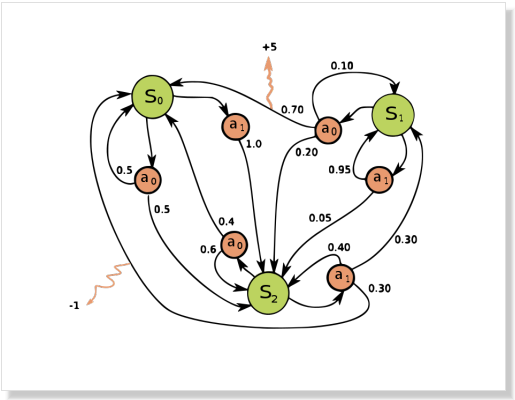

Figura 1.7: Estados y acciones en un proceso de decisión de Markov

1.5 Contexto social

La discriminación de género es un problema actual, en una sociedad y en una época con un acceso privilegiado a la información; en España hasta el 94.5% de la población ha tenido acceso a Internet en los últimos meses [16]. Aun así gran parte de la población se niega a contrastar los datos a los que acceden para formar una opinión razonada o bien sentencian este problema como inexistente.

Para contrarrestar la falta de información y desinformación al respecto de esta problemática, en este proyecto se utilizarán fuentes veraces y públicas para difundir información relevante, proporcionando datos actualizados de manera amigable a las personas interesadas en actualizar su opinión al respecto de esta temática.

1.6 Privacidad y tratamiento de datos sensibles

Inicialmente, se planteó almacenar determinados datos del usuario en función de las conclusiones que se podrían llegar a extraer del diálogo entre el agente y el usuario. Esto permitiría al sistema el perfilado del usuario y la posibilidad de ajustar las respuestas del agente a cada perfil, así como la presentación de cierta información u otra.

Tras la consulta con la delegación de protección de datos de la Universidad de Valladolid con respecto a la aplicación del RGPD y una vez que se realizó un profundo análisis de los riesgos que podría conllevar este tratamiento de datos personales, en este caso ideologías, así como la extracción de conclusiones sobre las mismas, se ha decidido no almacenar este tipo de información personal. Por ello se ha optado únicamente por el almacenaje del nombre de usuario de manera temporal y de manera permanente aunque totalmente anónima su correspondiente opinión o feedback sobre el diálogo mantenido.

1.7 Estructura del documento

Este documento está estructurado de la siguiente manera:

1. **Capítulo 1. Introducción:** Se presenta el proyecto al lector y la justificación de su elaboración en el contexto social y técnico en el que nos encontramos, también se describe la motivación en su desarrollo, objetivos del trabajo, así como la estructura de la memoria.

2. **Capítulo 2. Planificación:** Se presenta el marco de trabajo a utilizar y su adaptación al proyecto, la planificación y calendarización estimada de recursos así como la identificación de riesgos en el proyecto.

3. **Capítulo 3. Seguimiento del proyecto software:** Se documentan los avances y tareas realizadas en cada sprint del proyecto.

4. **Capítulo 4. Análisis del software:** Primer análisis realizado sobre el software que comprende requisitos, casos de uso y una modelización inicial en diagramas.

5. **Capítulo 5. Diseño del software:** Comprende el diseño final realizado, así como las decisiones de diseño tomadas.

6. **Capítulo 6. Implementación:** Se presentan las tecnologías utilizadas en todas las fases del proyecto, así como el proceso de implementación y puesta en funcionamiento del sistema.

7. **Capítulo 7. Pruebas:** En este capítulo se documentan las pruebas realizadas con el sistema.

8. **Capítulo 8. Conclusiones y trabajo futuro:** Se describen las conclusiones sobre el proyecto llevado a cabo así como algunas mejoras que se podrían realizar a futuro.

9. **Bibliografía**: Citas de varias referencias utilizadas a lo largo del proyecto, tanto para el desarrollo o escritura de la memoria.

10. **Anexos I, II y III:** Anexos de interés para el lector que comprenden la guía de uso, manual de instalación y enlace al repositorio del código fuente.

Capítulo 2

Planificación

2.1 Descripción del proyecto

Este proyecto tiene como finalidad la construcción de un chatbot de ámbito lúdico y/o instructivo capaz de dialogar con naturalidad con el usuario, siendo apta para su ejecución desde MS Teams.

2.1.1 Plan de proyecto

El plan del proyecto software tiene como finalidad la gestión de la concepción, estimación y puesta en marcha del proyecto. Esta gestión facilitará en gran medida la obtención de información relevante en cada momento, permitirá imputar gastos y realizar estimaciones de manera más fiable, además de permitir una toma de decisiones eficiente.

Para el desarrollo de este proyecto se utilizará un marco de trabajo ágil, que se detallará en la sección "Proceso de desarrollo". Se utilizará este marco de trabajo para poder generar entregables funcionales de forma temprana y detectar de manera anticipada cambios a realizar en la estructura y codificación del software.

2.1.2 Alcance del proyecto

El fin de este proyecto es la creación de un bot conversacional inteligente capaz de dialogar y solucionar dudas del usuario sobre la desigualdad de género y sus manifestaciones en la sociedad.

El chatbot resultante será capaz de responder preguntas concretas que el usuario realice y aportar información relevante sobre el ámbito de la conversación cuando esta información es requerida. La información que el chatbot facilitará al usuario serán datos reales y contrastados en todo momento.

El chatbot estará disponible para su uso desde aplicaciones de escritorio, web o móvil (Android e iOS) de Microsoft Teams.

2.1.3 Evolución del plan de proyecto

En la planificación del proyecto se registrará el seguimiento de los sprints que se realizarán, las historias de usuario abordadas, las tareas en las que se desglosan, el tiempo de dedicación en horas reales a cada una de ellas y sus fechas de inicio y fin.

2.2 Proceso de desarrollo

Se utilizará en este proyecto el marco de desarrollo de software ágil Scrum [17], el cual se centra en el desarrollo iterativo e incremental, donde los requisitos y soluciones evolucionan mediante la colaboración de equipos multifuncionales y auto organizados para generar productos de valor de forma temprana, adaptarse a los cambios y controlar los riesgos de forma más óptima.

Scrum es un marco de trabajo simple que promueve el trabajo colaborativo en equipo, posibilitando abordar problemas de gran complejidad en entornos cambiantes. Su conjunto de prácticas están basadas en el empirismo de la inteligencia colectiva (experimentación y observación) y el pensamiento Lean, que tiene como objetivo potenciar la mejora continua y para ello se realiza la medición y obtención de datos de manera constante para eliminar o corregir aquellos procesos que no aportan valor de negocio.

2.2.1 Características

- Scrum no se basa en el seguimiento de un plan establecido, sino en la adaptación continua a las circunstancias de la evolución del proyecto.
- Equipos pequeños, multifuncionales y autoorganizados.
- El producto se desarrolla en iteraciones cortas y fijas (sprints) de normalmente 2 a 4 semanas.
- El resultado de cada sprint es un entregable totalmente funcional que aporta valor real de negocio.
- Los requisitos pueden cambiar rápidamente.

2.2.2 Roles

A continuación se describen los roles principales que interactúan en un proyecto en el marco de trabajo ágil Scrum.

Product Owner: Profesional que media directamente con los *stackholders* o personas de interés. Su misión es asegurar que se cumplen las expectativas del negocio. Es el responsable de determinar el alcance y el plan de entregas, además de controlar el ROI y asegurar su maximización. Idealmente este rol debe de ser asumido por personal del equipo cliente, quien debe de conocer a la perfección el marco de trabajo a implementar así como el producto que se desea obtener. Algunas de sus tareas son:

- Aceptar o rechazar los resultados de cada sprint

- Prioriza funcionalidades según el valor de mercado.
- Ajusta funcionalidades y prioriza las iteraciones.
- Adapta el producto de cara a las necesidades de negocio.
- Modifica las prioridades del backlog para ajustarse a las expectativas a largo plazo.

Scrum Master: Profesional cuya función es hacer cumplir las prácticas, valores y principios Scrum. Debe de garantizar que el equipo es funcional y productivo, para ello resuelve los conflictos e interferencias que puedan tener lugar a lo largo del desarrollo del proyecto.

Development Team o Equipo de Desarrollo: Es el equipo de profesionales multifuncionales cuya misión es la construcción del producto. Cada individuo debe de tener gran autonomía, responsabilidad y capacidad de colaboración con los demás miembros. Cada persona del equipo de desarrollo participa en la definición de tareas en las que dividir cada requisito o historia de usuario así como la estimación del esfuerzo que supone realizarla.

Stackeholders o Interesados: Son las personas o entidades involucradas e interesadas en el éxito del proyecto: usuarios, clientes, accionistas, etc.

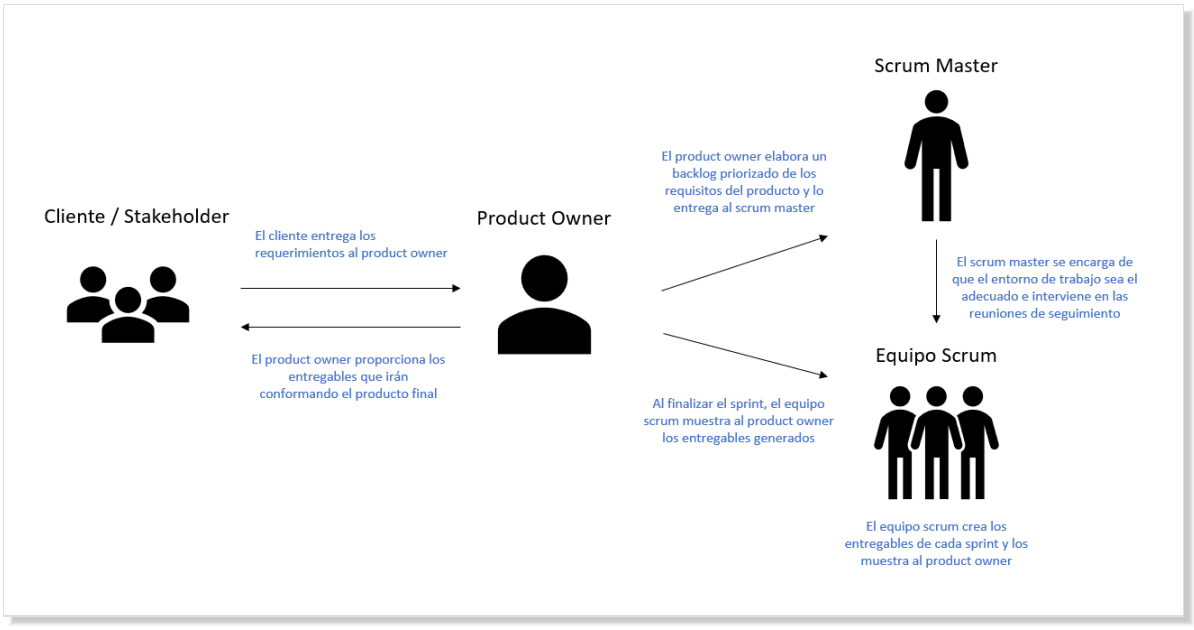

Figura 2.1: Roles en Scrum

2.2.3 Eventos

Los eventos en Scrum tienen como objetivo la inspección y adaptación de los artefactos a producir, así como la transparencia de la información. Estos eventos ocurren de forma regular para minimizar la necesidad de reuniones más largas y no definidas:

- **Sprint o iteración**: Son eventos de duración fija de normalmente 2 a 4 semanas. Todo el trabajo necesario para producir un incremento de producto se desarrolla dentro de un sprint. Un sprint empieza de forma inmediatamente posterior al término del anterior.

  Durante un sprint:

  - No se deben realizar cambios que pongan en peligro el objetivo del sprint.
  - El backlog de producto se puede refinar.
  - El alcance del sprint se puede renegociar con el product owner.
  - No se debe de cambiar la fecha de finalización del sprint.

  Se considera que un sprint ha tenido éxito si se ha cumplido el objetivo del sprint (sprint goal, que consiste en la implementación de una serie de elementos del backlog), en caso contrario, el sprint ha fracasado. Las tareas que se han incluido en un sprint y no se han finalizado a tiempo se incorporarán al product backlog, donde se volverán a repriorizar y en caso de seguir siendo prioritarias, se incorporarán al siguiente sprint.

- **Sprint planning o planificación del sprint**: Todo el equipo Scrum participa en este evento, donde el product owner expone las prioridades del negocio y se procede al análisis y planificación de los requisitos. Se determinan los elementos del product backlog que serán incluidos en el sprint y la descomposición de estos elementos en tareas con una duración normalmente no más larga que una jornada de trabajo, así como el grado de dificultad en la realización de las mismas, la cual será ilustrada en una escala de puntos de 1 a 5.

- **Daily Scrum o Scrum diario**: Es una reunión de normalmente 15 minutos, donde participan los desarrolladores del equipo y el Scrum master. El objetivo es inspeccionar el progreso realizado, enfocar el trabajo pendiente a realizar durante el día y localizar y dar solución a impedimentos en el desarrollo del sprint.

- **Sprint review o revisión del sprint**: Participa el equipo Scrum y tiene como propósito inspeccionar el producto resultante del sprint, así como determinar adaptaciones que se deben realizar en el mismo. Durante esta reunión el Scrum master presenta los resultados a los stackeholders así como la evolución realizada para lograr el producto objetivo. El product backlog puede modificarse para adaptarse a las nuevas circunstancias presentandas.

- **Sprint retrospective o retrospectiva del sprint**: Participan el equipo de desarrollo y Scrum master y tiene como objetivo el análisis de cómo se desarrolló el último sprint con objetivo de mejorar la calidad y efectividad del trabajo. Se identifican aquellos elementos que se deben de mejorar, estos pueden ser metodologías utilizadas, herramientas, tareas realizadas por personas, etc.

2.2.4 Artefactos

Los artefactos generados representan el trabajo o valor de negocio. Cada uno de ellos tiene como finalidad la transparencia de información de manera que cada individuo participante del proyecto

posea los mismos datos. Los artefactos se utilizan de igual manera para medir el progreso realizado hacia el objetivo de producto.

- **Product backlog (pila de producto)**: Es una lista de elementos que se necesitan para construir o mejorar el producto. El product backlog se puede refinar, es decir, dividir y precisar mejor sus elementos. Aquellos elementos del product backlog que los desarrolladores consideran que se podrán realizar en el periodo de un sprint, se incorporan al sprint backlog.

  De forma generalista, los elementos del backlog se pueden clasificar en tres grupos:

  - **Funcionalidad:** Son tareas que implican la construcción de nuevas características del producto, o mejoras sobre las ya existentes.
  - **Chore**: Son tareas que no aportan valor de negocio directo (aunque sí suponen un coste) y son de realización necesaria, antes o durante una tarea de funcionalidad; puede afectar a varias tareas de funcionalidad simultáneamente. Un ejemplo es la tarea "Preparación del entorno del desarrollo".
  - **Bugs**: Corrección de errores no previstos en el producto.

- **Sprint backlog (pila del sprint)**: El sprint backlog está compuesto por los elementos del product baclog seleccionados para el sprint, así como el objetivo del sprint y el plan de entrega del mismo. Los elementos del product backlog incorporados deben de intentar satisfacer el objetivo del sprint.

- **Increment o incremento**: Es una mejora de las funcionalidades del producto; cuando un elemento del product backlog es implementado y completado, se genera un incremento. Un sprint puede tener varios incrementos.

Otros meta artefactos de valor se pueden generar a lo largo del sprint:

- **Definición de realizado**: Es la definición que toma una tarea cuando se da por terminada o completada, lo cual puede incluir la realización de documentación pertinente, pruebas en el software etc. Esta definición debe de ser compartida por todo el equipo de desarrollo.

- **Objetivo del producto**: Es el producto objetivo a largo plazo del equipo Scrum, consta de los elementos del product backlog.

- **Gráficos burndown**: Gráfico que refleja el trabajo pendiente de realización en un sprint.

- **Gráficos burnup**: Gráfico que refleja el trabajo ya realizado en un sprint.

- **Gráficos de trabajo acumulado**.

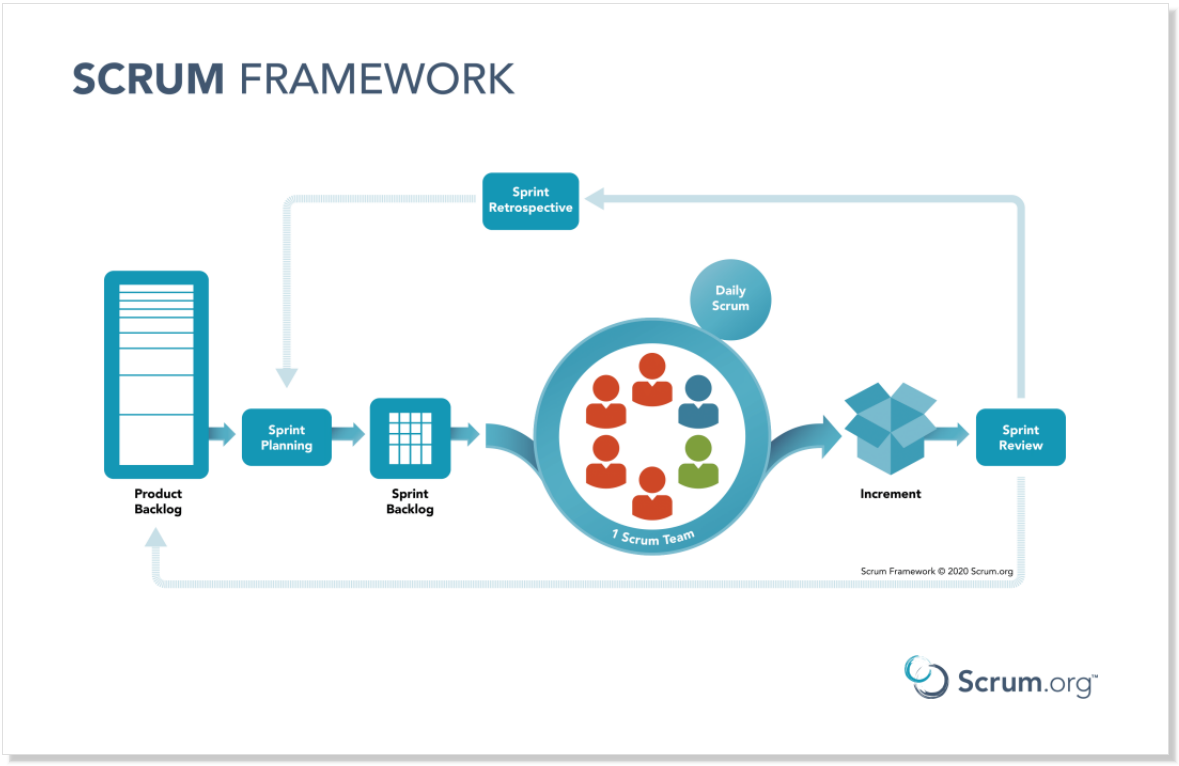

Figura 2.2: Scrum framework

2.3 Aplicación de Scrum al proyecto

Para el desarrollo de este proyecto se requieren de diversos perfiles profesionales, sin embargo el único perfil disponible es el de la estudiante, que deberá de asumir el rol del equipo de desarrollo, en este caso desarrollador senior; como el tema del trabajo de fin de grado ha sido propuesto por la alumna, esta también llevará a cabo los roles de product owner y stackeholder (como cliente). Mientras que la tutora responsable asumirá el perfil de Scrum master y stackeholder.

Rol	Descripción	Asumido por
Scrum Master	Scrum master	Gonzalo Tasis, Margarita
Product Owner	Responsable del negocio	Mun de Souza, Se Hee
Stackeholder	Representa sus intereses y prioridades	Gonzalo Tasis, Margarita Mun de Souza, Se Hee
Development Team	Equipo que desarrolla el producto	Mun de Souza, Se Hee

Tabla 2.1: Asignación de roles

Debido a la situación actual de la alumna, que se encuentra trabajando a jornada completa, se ha establecido una duración mínima para los sprints de 2 semanas y una duración máxima de 4 semanas.

La dedicación diaria mínima a cada sprint está establecida en una hora. Las reuniones se celebrarán los lunes cada 2 semanas, donde se llevarán a cabo los eventos de sprint review, sprint

retrospective y sprint planning del siguiente incremento.

Las historias de usuario que se incorporarán a cada sprint se determinarán en función del valor de negocio que aporten así como la dificultad que supongan.

2.3.1 Planificación inicial de sprints

La asignatura de Trabajo de Fin de Grado tiene actualmente una carga de trabajo de 12 ECTS que representan 300 horas de dedicación. Con lo cual, este proyecto está dimensionado para su realización en 300 horas. Las dedicación que la alumna puede realizar es de máximo 11 horas semanales de lunes a viernes. La distribución del Trabajo de Fin de Grado utilizando esta dedicación, corresponde a aproximadamente 30 semanas de trabajo. A continuación se ilustran los sprints a realizar, la dedicación en horas estimadas así como las fechas de inicio y finalización.

Existirán sprints opcionales, que se utilizarán en caso de la materialización de algún riesgo, con objetivo de contrarrestar sus efectos. La duración de estos sprint opcionales serán de 10 a 40 horas. Su utilización o no se estudiará en el capítulo de "Seguimiento del proyecto" y supondrá un incremento de la duración total del proyecto por cada sprint opcional utilizado.

La calendarización inicial estimada del proyecto es la siguiente:

Sprint	Descripción	Duración (semanas)	Comienzo	Fin
1	Sprint 1	4	28/06/2021	25/07/2021
2	Sprint 2	4	26/07/2021	22/08/2021
3	Sprint 3	4	23/08/2021	19/09/2021
4	Sprint 4	3	20/09/2021	10/10/2021
5	Sprint 5	3	11/10/2021	31/10/2021
6	Sprint 6	3	01/11/2021	21/11/2021
7	Sprint 7	3	22/11/2021	12/12/2021
8	Sprint 8	3	13/12/2021	26/12/2021
9	Sprint opcional			
10	Sprint opcional			
11	Sprint opcional			
12	Sprint opcional			

Tabla 2.2: Planificación inicial

2.3.2 Product backlog

El product backlog, backlog de producto o pila de producto, estará compuesto por historias de usuario (user stories). Tras analizar las historias de usuario, se han extraído las tareas en las que se puede desglosar cada historia, además de lo requisitos funcionales y no funcionales. A continuación, se ilustra una tabla con las historias de usuario identificadas ordenadas según su prioridad y con su estimación de complejidad sobre 5 puntos:

ID	Título	Descripción	Estim. sobre 5	Prioridad
HU01	Nombre usuario desde MS Teams	Como usuario de MS Teams quiero que el chatbot me llame por mi nombre, de manera que el diálogo pueda ser más personal.	2	−
HU02	Conocer condiciones de Uso desde MS Teams	Como usuario de MS Teams quiero aceptar la política de tratamiento de datos y privacidad para poder hacer uso de las funcionalidades del sistema.	1	^
HU03	Petición de información desde MS Teams	Como usuario de MS Teams quiero pedir información en texto para resolver una duda que tengo o indagar en un tema de manera rápida.	4	^
HU04	Tipo de información desde MS Teams	Como usuario de MS Teams quiero que obtener información verídica, actualizada, no subjetiva y no sesgada desde bases de datos gubernamentales, para que pueda informarme de manera correcta y con estadísticas reales.	5	−
HU05	Fluidez del diálogo en MS Teams	Como usuario de MS Teams quiero que el diálogo sea lo más fluido posible y parecido a un diálogo con un humano, de manera que no me frustre por entenderlo o seguir el hilo de la conversación.	4	v
HU06	Grabar opinión	Como usuario de MS Teams quiero poder dejar grabada mi opinión sobre mi satisfacción con el chat.	3	v

Tabla 2.3: Product backlog inicial

2.4 Plan de gestión de riesgos

Un riesgo es un evento incierto que en el supuesto de que tenga lugar, puede afectar a los objetivos de un proyecto, entre ellos su cronograma, costo o calidad. Por lo general, un riesgo tiene un impacto negativo en el proyecto, aunque también existen riesgos positivos o de oportunidad cuyo efecto en el proyecto favorece sus objetivos. Cada uno de estos eventos tienen consecuencias asociadas y se pueden agrupar en categorías en función de los efectos que puedan tener en el

proyecto.

Para poder gestionar de forma proactiva los riesgos negativos y evitar sus consecuencias, se debe de realizar un análisis de los mismos. En el análisis destacamos las prácticas para la mitigación y contención de riesgos recogidas en la Guía PMBOK [18] (Project Management Body of Knowledge):

- El objetivo de la mitigación es la reducción de la exposición al riesgo, decrementando de esta manera su probabilidad de ocurrencia y el impacto que podría provocar sobre el proyecto.
- El plan de contingencia consiste en el uso de recursos y técnicas de forma planificada para contener el riesgo una vez se materialice.

A continuación, se detallan los riesgos que han sido identificados y evaluados y pueden afectar negativamente a la consecución del proyecto software. Se ilustran en las siguientes tablas: El resumen del riesgo, su descripción, la consecuencia que tiene una vez materializado, la prioridad que toma su resolución una vez materializado, la probabilidad de ocurrencia, el impacto que tendría en el proyecto su ocurrencia, lo expuesto que se encuentra en proyecto a su ocurrencia, la categoría a la que pertenece el riesgo, el plan de mitigación y contingencia asociado al riesgo, la monitorización que se realiza sobre el mismo, así como los recursos estimados para la subsanación de las consecuencias asociadas al mismo.

R01	**Falta de conocimientos y/o experiencia del personal**
Descripción	El personal no ha realizado ningún proyecto con chatbots e inteligencia artificial y deberá pasar por el aprendizaje de todos los conceptos necesarios para llevar a cabo el proyecto.
Consecuencia	Retraso en la entrega del proyecto.
Prioridad	2/5
Probabilidad	65%
Impacto	Crítico
Exposición	1,95
Categoría	Personal
Plan de mitigación	Buscar información y manuales en internet, leer la documentación disponible de las herramientas a utilizar antes de pasar a las siguientes etapas del desarrollo.
Plan de contingencia	Documentarse adecuadamente y realizar cursos prácticos sobre las herramientas a utilizar. Redistribuir el sprint backlog no abordado en los siguientes sprints si siguen siendo relevantes y modificar la fecha fin del proyecto. Realización de horas extras de trabajo para la inclusión de dichos elementos en el sprint.
Monitorización	Antes de implementar cualquier funcionalidad del sistema comprobar que el personal dispone de todos los conocimientos necesarios.
Recursos estimados	3 horas para el aprendizaje de los conceptos necesarios.

Tabla 2.4: Definición de riesgos: R01

R02	**La formación del personal no se completó a tiempo**
Descripción	Completar la formación del personal implicado en el proyecto conlleva más tiempo del esperado.
Consecuencia	Retraso en la entrega del proyecto.
Prioridad	2/5
Probabilidad	40%
Impacto	Moderado
Exposición	2,00
Categoría	Personal
Plan de mitigación	Dedicar siempre que sea posible más tiempo de lo estimado a la formación, aun cuando no se ha concebido en la planificación.
Plan de contingencia	Dedicar más horas a la formación en horario de trabajo no habitual.
Monitorización	Seguimiento de las sesiones formativas realizadas y pendientes para cada semana.
Recursos estimados	5 horas para la formación extra.

Tabla 2.5: Definición de riesgos: R02

R03	**La falta de motivación reduce la productividad**
Descripción	El personal implicado carece de motivación en un determinado momento para avanzar en el trabajo.
Consecuencia	Retraso en la entrega del proyecto
Prioridad	2/5
Probabilidad	30%
Impacto	Moderado
Exposición	1,50
Categoría	Personal
Plan de mitigación	Dedicar siempre que sea posible más tiempo de lo estimado al avance del proyecto, aun cuando no se ha concebido en la planificación.
Plan de contingencia	Redistribuir el sprint backlog no abordado en los siguientes sprints si siguen siendo relevantes y modificar la fecha fin del proyecto. Realización de horas extras de trabajo para la inclusión de dichos elementos en el sprint.
Monitorización	Seguimiento de las tareas en curso y finalizadas.
Recursos estimados	5 horas extras de trabajo.

Tabla 2.6: Definición de riesgos: R03

R04	**La falta de especialización necesaria aumenta los posibles defectos del sistema.**
Descripción	El personal implicado carece de los conocimientos específicos de años de experiencia sobre la herramienta o lenguaje de programación a utilizar.
Consecuencia	Aumento potencial de errores en el sistema. Retraso en la entrega del proyecto para corregir desperfectos.
Prioridad	1/5
Probabilidad	35%
Impacto	Moderado
Exposición	3,85
Categoría	Personal
Plan de mitigación	No aplica.
Plan de contingencia	Consultar la documentación disponible y ejemplos prácticos. Modificar la fecha fin del proyecto. Estudiar la realización de un sprint para corregir desperfectos.
Monitorización	No aplica.
Recursos estimados	3 horas para la formación extra y 8 horas para un sprint de corrección de desperfectos.

Tabla 2.7: Definición de riesgos: R04

R05	**El personal necesita más tiempo de lo esperado para trabajar con el entorno y herramientas software.**
Descripción	El personal necesita más tiempo para comprender el correcto funcionamiento de la herramienta y entorno y poder aplicarlo correctamente.
Consecuencia	Retraso en la entrega del proyecto.
Prioridad	1/5
Probabilidad	20%
Impacto	Moderado
Exposición	0,60
Categoría	Personal
Plan de mitigación	Realización de cursos prácticos para conocer la herramienta y entorno.
Plan de contingencia	Consultar la documentación disponible y ejemplos prácticos. Modificar la calendarización estimada de los sprints. Modificar la fecha fin del proyecto.
Monitorización	Comprobar que se disponen de los conocimientos teóricos necesarios para poder utilizar el entorno y herramienta.
Recursos estimados	0,5 horas para realizar la replanificación de sprints y 2,5 horas para la consulta de documentación pertinente.

Tabla 2.8: Definición de riesgos: R05

R06	**El personal trabaja más lento de lo esperado.**
Descripción	El personal necesita más tiempo de lo estimado para finalizar sus tareas asignadas.
Consecuencia	Retraso en la entrega del proyecto.
Prioridad	3/5
Probabilidad	40%
Impacto	Crítico
Exposición	5,20
Categoría	Personal
Plan de mitigación	Siempre que sea posible avanzar en el desarrollo del proyecto los días no concebidos en la planificación.
Plan de contingencia	Redistribuir el sprint backlog no abordado en los siguientes sprints si siguen siendo relevantes y modificar la fecha fin del proyecto. Realización de horas extras de trabajo para la inclusión de dichos elementos en el sprint.
Monitorización	Seguimiento de las tareas finalizadas y pendientes en cada semana.
Recursos estimados	13 horas de trabajo extra para finalizar las tareas pendientes.

Tabla 2.9: Definición de riesgos: R06

R07	**Un diseño demasiado simple no cubre las características principales del sistema.**
Descripción	El diseño del sistema se ha hecho de manera muy sencilla sin abordar aspectos más profundos e importantes
Consecuencia	Volver a diseñar el sistema. Retraso en la entrega del proyecto.
Prioridad	2/5
Probabilidad	25%
Impacto	Crítico
Exposición	1,75
Categoría	Diseño e implementación
Plan de mitigación	Concretar los aspectos importantes del sistema antes de diseñar y comprobar que se han abordado durante el diseño.
Plan de contingencia	Rediseño del sistema. Inclusión de las tareas pendientes en el sprint backlog del siguiente sprint. Modificación de la fecha fin del proyecto si aplica.
Monitorización	Comprobación de que se disponen de todas las características importantes a plasmar del sistema antes de pasar al diseño.
Recursos estimados	7 horas para realizar el rediseño.

Tabla 2.10: Definición de riesgos: R07

R08	**Un diseño demasiado complejo implica una implementación demasiado compleja e improductiva.**
Descripción	El diseño del sistema se ha hecho de manera muy compleja, abordando aspectos innecesarios y añadiendo complicación a la implementación.
Consecuencia	Volver a diseñar el sistema. Retraso en la entrega del proyecto.
Prioridad	2/5
Probabilidad	25%
Impacto	Crítico
Exposición	1,75
Categoría	Diseño e implementación
Plan de mitigación	Concretar los aspectos esenciales del sistema antes de diseñar y comprobar que se han abordado durante el diseño.
Plan de contingencia	Rediseño del sistema. Inclusión de las tareas pendientes en el sprint backlog del siguiente sprint. Modificación de la fecha fin del proyecto si aplica.
Monitorización	Comprobación que se disponen de todas las características esenciales del sistema antes del diseño.
Recursos estimados	7 horas para realizar el rediseño.

Tabla 2.11: Definición de riesgos: R08

R09	**Se ha realizado un diseño incorrecto lo que implica una implementación incorrecta.**
Descripción	El diseño del sistema se ha hecho de manera pobre o incorrecta, haciendo que la implementación sea también incorrecta.
Consecuencia	Volver a diseñar el sistema. Retraso en la entrega del proyecto.
Prioridad	3/5
Probabilidad	40%
Impacto	Catastrófico
Exposición	2,80
Categoría	Diseño e implementación
Plan de mitigación	Concretar los aspectos esenciales del sistema antes de diseñar y comprobar que se han abordado durante el diseño.
Plan de contingencia	Rediseño del sistema. Inclusión de las tareas pendientes en el sprint backlog del siguiente sprint. Modificación de la fecha fin del proyecto si aplica.
Monitorización	Comprobación que se disponen de todas las características esenciales del sistema antes del diseño.
Recursos estimados	7 horas para realizar el rediseño.

Tabla 2.12: Definición de riesgos: R09

R10	**La utilización de una metodología o marco de trabajo desconocido implica errores en la implementación y más tiempo de lo esperado para la formación.**
Descripción	Al utilizar una metodología o marco de trabajo desconocido el proyecto es susceptible de sufrir errores de implementación. Se necesitará más tiempo para el aprendizaje de los conceptos necesarios y para la corrección de errores cometidos en etapas anteriores.
Consecuencia	Realizar otra formación. Corrección de errores en etapas anteriores. Retraso en la entrega del proyecto.
Prioridad	5/5
Probabilidad	30%
Impacto	Catastrófico
Exposición	3,90
Categoría	Diseño e implementación
Plan de mitigación	Adquirir todos los conceptos teóricos necesarios y comprobar que se dispone de ellos antes de pasar a la implementación.
Plan de contingencia	Consultar la documentación disponible. Realización de horas de trabajo extras. Modificar fecha fin de proyecto.
Monitorización	Comprobación que se disponen de todos los conocimientos necesarios antes de la implementación.
Recursos estimados	5 horas para la formación, 8 horas para corrección de errores en etapas anteriores.

Tabla 2.13: Definición de riesgos: R10

R11	**La falta de seguimiento del progreso hace que se desconozca que el proyecto está atrasado.**
Descripción	No se ha realizado un seguimiento del progreso del proyecto comparándolo con la planificación estimada, de manera que no se conoce que el proyecto está atrasado.
Consecuencia	Retraso en la entrega del proyecto. Modificación de la fecha fin del proyecto.
Prioridad	2/5
Probabilidad	10%
Impacto	Crítico
Exposición	0,10
Categoría	Proceso
Plan de mitigación	Realizar un seguimiento de la fase en la que nos encontramos y las tareas realizadas comparándolo con la planificación estimada.
Plan de contingencia	Modificar fecha fin de proyecto. Replanificar las fechas de los siguientes sprints.
Monitorización	Seguimiento de la planificación y comparación con el estado real del proyecto.
Recursos estimados	1 hora para la replanificación del calendario y sprints.

Tabla 2.14: Definición de riesgos: R11

R12	**Control de calidad precario.**
Descripción	No se ha realizado un correcto control de la calidad del proyecto haciendo que los errores en él se conozcan de manera tardía.
Consecuencia	Replanificación del calendario.
Prioridad	2/5
Probabilidad	35%
Impacto	Crítico
Exposición	2,45
Categoría	Proceso
Plan de mitigación	Realizar un control de todas las características del sistema y comprobar que constan en el proyecto.
Plan de contingencia	Replanificar calendario. Realización de horas extras para la corrección de la calidad.
Monitorización	Seguimiento de la lista de las características del sistema que se han de implementar, comprobar y controlar.
Recursos estimados	6 horas para la corrección de errores y 1 hora para la replanificación del calendario.

Tabla 2.15: Definición de riesgos: R12

R13	**La falta de rigor en el desarrollo conduce al consumo de más tiempo de lo esperado.**
Descripción	Se han ignorado los estándares y/o fundamentos del desarrollo software, lo que implica la recreación de los entregables.
Consecuencia	Replanificación del calendario. Retraso en la entrega del proyecto.
Prioridad	4/5
Probabilidad	30%
Impacto	Crítico
Exposición	5,70
Categoría	Proceso
Plan de mitigación	Conocer y aplicar los estándares en todas las fases del desarrollo software siguiendo con la especificación de la metodología del trabajo.
Plan de contingencia	Volver a fabricar los entregables. Replanificar calendario.
Monitorización	Seguimiento de las fases del desarrollo software y comprobación de que se aplican los fundamentos dictados.
Recursos estimados	18 horas para volver a producir los entregables y 1 hora para la replanificación del calendario.

Tabla 2.16: Definición de riesgos: R13

R14	**El exceso de rigor en el desarrollo conduce al consumo de más tiempo de lo esperado.**
Descripción	Se realiza la implementación siguiendo con exceso rigor los estándares y/o fundamentos del desarrollo software, lo que implica un consumo de tiempo innecesario.
Consecuencia	Replanificación del calendario. Retraso en la entrega del proyecto.
Prioridad	2/5
Probabilidad	20%
Impacto	Moderado
Exposición	0,20
Categoría	Proceso
Plan de mitigación	Conocer y aplicar los estándares esenciales en todas las fases del desarrollo software siguiendo con la especificación de la metodología del trabajo.
Plan de contingencia	Replanificar calendario.
Monitorización	Seguimiento de las fases del desarrollo software y comprobación que se aplican los fundamentos dictados sin excederse en complejidades.
Recursos estimados	1 hora para la replanificación del calendario.

Tabla 2.17: Definición de riesgos: R14

R15	**La gestión de riesgos del proyecto consume más tiempo de lo esperado.**
Descripción	Realizar una identificación, control y monitorización de los riesgos del proyecto consume más tiempo de lo estimado.
Consecuencia	Replanificación del calendario. Retraso en la entrega del proyecto.
Prioridad	1/5
Probabilidad	20%
Impacto	Moderado
Exposición	2,20
Categoría	Proceso
Plan de mitigación	Registrar los riesgos conocidos y que realmente podrían afectar a la consecución del proyecto.
Plan de contingencia	Actualizar la gestión de riesgos. Replanificar calendario.
Monitorización	Seguimiento de la planificación estimada y comparación con el estado real del proyecto.
Recursos estimados	1 hora para la replanificación del calendario y 10 horas para modificar la gestión de riesgos.

Tabla 2.18: Definición de riesgos: R15

R16	**Colapso mundial por el brote de una pandemia**
Descripción	Brote de una pandemia vírica o bacteriana que se extiende rápidamente por todo el planeta, causando el confinamiento domiciliar masivo de miles de millones de personas.
Consecuencia	Reducción de los medios físicos y/o psicológicos para realizar el proyecto, falta de concentración y pérdida de productividad. Retraso en la entrega del proyecto.
Prioridad	3/5
Probabilidad	1%
Impacto	Catastrófico
Exposición	0,64
Categoría	Fuerza mayor
Plan de mitigación	Planificar un horario de trabajo semanal, promover un ambiente de orden en el domicilio que incite a mantener la claridad mental y concentración.
Plan de contingencia	Replanificar el horario de trabajo semanal adaptándose a las necesidades actuales y condiciones mentales del personal implicado en el proyecto.
Monitorización	No aplica
Recursos estimados	Implica la utilización de 64 horas de trabajo extra para conseguir terminar el proyecto a tiempo.

Tabla 2.19: Definición de riesgos: R16

R17	**La entrega a tiempo del proyecto depende directamente del personal implicado, que por causas externas no puede seguir con la consecución del mismo.**
Descripción	El trabajo del personal implicado se ve suspendido por causas externas, lo que imposibilita la entrega del proyecto.
Consecuencia	Retraso en la entrega del proyecto.
Prioridad	5/5
Probabilidad	35%
Impacto	Catastrófico
Exposición	33,60
Categoría	Fuerza mayor
Plan de mitigación	No aplica
Plan de contingencia	Replanificar el calendario
Monitorización	No aplica
Recursos estimados	Utilización de 96 horas de más para conseguir terminar el proyecto.

Tabla 2.20: Definición de riesgos: R17

R18	**Trabajar en un entorno software desconocido causa problemas no previstos**
Descripción	No se conocen todas las características del entorno software a utilizar, lo cual puede causar contratiempos inesperados.
Consecuencia	Retraso en la entrega del proyecto.
Prioridad	2/5
Probabilidad	10%
Impacto	Moderado
Exposición	0,80
Categoría	Producto
Plan de mitigación	Hacer cursos prácticos para conocer mejor el funcionamiento de la herramienta.
Plan de contingencia	Consultar la documentación disponible del entorno a utilizar. Rediseño de la planificación.
Monitorización	Comprobar que se dispone de los conocimientos necesarios para utilizar la herramienta.
Recursos estimados	Utilización de 8 horas de más para el rediseño de la planificación y adquisición de los conocimientos necesarios para utilizar la herramienta.

Tabla 2.21: Definición de riesgos: R18

R19	**La planificación inicial no incluye todas las tareas necesarias.**
Descripción	Debido a un desarrollo deficiente de la planificación, no se incluyeron todas las tareas esenciales en la planificación.
Consecuencia	Reconstrucción de la planificación. Retraso en la entrega del proyecto.
Prioridad	3/5
Probabilidad	75%
Impacto	Moderado
Exposición	3,00
Categoría	Elaboración de la planificación
Plan de mitigación	Planificar un horario de trabajo semanal. Avanzar en el desarrollo del proyecto fuera de los días concebidos en la planificación.
Plan de contingencia	Replanificar el calendario
Monitorización	Seguimiento y comparación de las tareas desarrolladas y las recogidas en la planificación.
Recursos estimados	4 horas para la replanificación.

Tabla 2.22: Definición de riesgos: R19

R20	**La planificación es optimista en lugar de realista.**
Descripción	La planificación se ha concebido posicionándonos en el mejor caso.
Consecuencia	Reconstruir la planificación. Retraso en la entrega del proyecto.
Prioridad	3/5
Probabilidad	70%
Impacto	Moderado
Exposición	0,70
Categoría	Elaboración de la planificación
Plan de mitigación	Planificar un horario de trabajo semanal. Avanzar en el desarrollo del proyecto fuera de los días concebidos en la planificación.
Plan de contingencia	Replanificar el calendario.
Monitorización	Seguimiento de la fase del desarrollo, tareas e hitos desarrollados hasta el momento.
Recursos estimados	1 hora para la replanificación.

Tabla 2.23: Definición de riesgos: R20

R21	**Indisponibilidad del personal implicado en el desarrollo.**
Descripción	No se puede continuar con el trabajo por falta de disponibilidad del personal implicado.
Consecuencia	Retraso en la entrega del proyecto.
Prioridad	2/5
Probabilidad	65%
Impacto	Moderado
Exposición	1,30
Categoría	Elaboración de la planificación
Plan de mitigación	Planificar un horario de trabajo semanal a seguir. Siempre que sea posible avanzar en el desarrollo del proyecto fuera de los días concebidos en la planificación.
Plan de contingencia	Modificar la planificación inicial, priorizando las tareas y añadiendo márgenes de tiempo en la entrega de cada una.
Monitorización	Seguimiento del progreso del personal.
Recursos estimados	Implicación de 2 horas de más para poder seguir la replanificación del proyecto.

Tabla 2.24: Definición de riesgos: R21

R22	**No se puede construir un proyecto de la envergadura presentada en el tiempo asignado.**
Descripción	El tiempo fijado en la planificación para la consecución del proyecto es insuficiente.
Consecuencia	Reconstruir la planificación. Retraso en la entrega del proyecto.
Prioridad	3/5
Probabilidad	70%
Impacto	Catastrófico
Exposición	0,70
Categoría	Elaboración de la planificación
Plan de mitigación	Estimar desde el peor caso las horas necesarias para la consecución del proyecto. Planificar un horario de trabajo semanal a seguir. Avanzar en el desarrollo del proyecto fuera de los días concebidos en la planificación.
Plan de contingencia	Replanificar el calendario y la estimación de la fecha fin.
Monitorización	Seguimiento del tiempo empleado en cada fase del desarrollo, tareas e hitos.
Recursos estimados	1 hora para la replanificación.

Tabla 2.25: Definición de riesgos: R22

R23	**El proyecto conlleva la escritura de más líneas de código y/o documentación de lo esperado.**
Descripción	Se ha realizado una mala estimación del tamaño del proyecto.
Consecuencia	Modificar la planificación. Retraso en la entrega del proyecto.
Prioridad	2/5
Probabilidad	40%
Impacto	Moderado
Exposición	2,80
Categoría	Elaboración de la planificación
Plan de mitigación	Planificar un horario de trabajo semanal. Avanzar en el desarrollo del proyecto fuera de los días concebidos en la planificación.
Plan de contingencia	Replanificar el calendario
Monitorización	Seguimiento del esfuerzo empleado en cada tarea en función de horas.
Recursos estimados	1 hora para la replanificación y 6 horas para la escritura de código y/o documentación necesaria.

Tabla 2.26: Definición de riesgos: R23

R24	**La reestimación debido a un retraso en la planificación es demasiado optimista.**
Descripción	Se ha realizado una replanificación del proyecto deficiente.
Consecuencia	Modificar la planificación. Retraso en la entrega del proyecto.
Prioridad	2/5
Probabilidad	40%
Impacto	Moderado
Exposición	0,40
Categoría	Elaboración de la planificación
Plan de mitigación	Estimar desde el peor caso las horas necesarias para la consecución del proyecto. Realizar un seguimiento de las fases del desarrollo y siempre que sea posible avanzar el proyecto los días no concebidos en la planificación.
Plan de contingencia	Reconstrucción de la planificación. Replanificar el horario de trabajo semanal adaptándose a las necesidades actuales y condiciones del personal implicado.
Monitorización	Seguimiento del esfuerzo real empleado en cada tarea y contrastación con el estimado en la planificación.
Recursos estimados	1 hora para la replanificación.

Tabla 2.27: Definición de riesgos: R24

R25	**El retraso en una tarea produce retrasos en cascada en las tareas dependientes**
Descripción	Ha habido un retraso en una tarea maestra de la que dependen una o más, causando el retraso de las tareas dependientes.
Consecuencia	Modificar la planificación. Retraso en la entrega del proyecto.
Prioridad	1/5
Probabilidad	50%
Impacto	Insignificante
Exposición	3,50
Categoría	Elaboración de la planificación
Plan de mitigación	Realizar un seguimiento de las tareas realizadas y el esfuerzo supuesto y siempre que sea posible avanzar el proyecto los días no concebidos en la planificación.
Plan de contingencia	Modificar la planificación. Replanificar el horario de trabajo semanal adaptándose a las necesidades actuales.
Monitorización	Seguimiento del esfuerzo real empleado en cada tarea y contrastación con el estimado en la planificación.
Recursos estimados	1 hora para la replanificación y 6 horas para la terminación de la tarea pendiente.

Tabla 2.28: Definición de riesgos: R25

R26	**La revisión y/o tutorización del proyecto es más lenta de lo esperado.**
Descripción	La persona encargada de revisar y/o tutorizar el avance del proyecto se demora más de lo esperado en examinar el proyecto.
Consecuencia	Retraso en la entrega del proyecto.
Prioridad	2/5
Probabilidad	5%
Impacto	Moderado
Exposición	0,80
Categoría	Organización y gestión
Plan de mitigación	No aplica.
Plan de contingencia	No aplica.
Monitorización	Seguimiento del calendario propuesto para las revisiones.
Recursos estimados	Implica la espera de 16 horas más para la revisión del proyecto.

Tabla 2.29: Definición de riesgos: R26

R27	**El espacio de trabajo no está disponible cuando es necesario**
Descripción	El espacio de trabajo utilizado no se encuentra disponible cuando es requerido.
Consecuencia	Adaptar otro espacio de trabajo.
Prioridad	1/5
Probabilidad	10%
Impacto	Insignificante
Exposición	0,10
Categoría	Ambiente e infraestructura del desarrollo.
Plan de mitigación	No aplica.
Plan de contingencia	Utilizar otro espacio de trabajo distinto al habitual
Monitorización	No aplica
Recursos estimados	Utilización de 1 habitáculo distinto para llevar a cabo el trabajo.

Tabla 2.30: Definición de riesgos: R27

R28	**El espacio de trabajo es ruidoso o provoca la distracción**
Descripción	El espacio de trabajo no estimula la concentración.
Consecuencia	Retraso en la entrega de la tarea a realizar.
Prioridad	1/5
Probabilidad	20%
Impacto	Moderado
Exposición	0,20
Categoría	Ambiente e infraestructura del desarrollo.
Plan de mitigación	No aplica
Plan de contingencia	Utilizar otro habitáculo para la realización del trabajo
Monitorización	No aplica
Recursos estimados	Utilización de 1 habitáculo distinto para llevar a cabo el trabajo.

Tabla 2.31: Definición de riesgos: R28

R29	**Pérdida de datos**
Descripción	Se produce la pérdida parcial o total de los entregables, fuentes, y otros documentos del proyecto.
Consecuencia	Retraso en la entrega del proyecto.
Prioridad	5/5
Probabilidad	5%
Impacto	Catastrófico
Exposición	6,40
Categoría	Ambiente e infraestructura del desarrollo.
Plan de mitigación	Realizar copias de seguridad en la nube siempre que se modifique algún elemento del proyecto, uso de un sistema de control de versiones.
Plan de contingencia	Replanificar el calendario. Volver a producir los entregables y la documentación.
Monitorización	No aplica.
Recursos estimados	128 horas para volver a producir los entregables y documentos.

Tabla 2.32: Definición de riesgos: R29

R30	**Ausencia de herramientas hardware para realizar el proyecto**
Descripción	Las herramientas hardware necesarias no funcionan o dejan de estar disponibles cuando requeridas.
Consecuencia	Retraso en la entrega del proyecto.
Prioridad	5/5
Probabilidad	5%
Impacto	Crítico
Exposición	25
Categoría	Ambiente e infraestructura del desarrollo.
Plan de mitigación	Realizar copias de seguridad en nube para poder acceder al trabajo desde otro equipo.
Plan de contingencia	Adquirir un nuevo computador o seguir con el desarrollo del proyecto desde otro equipo disponible.
Monitorización	No aplica
Recursos estimados	500,00€ para la compra de otro equipo.

Tabla 2.33: Definición de riesgos: R30

R31	**Ausencia de medios software para realizar el proyecto**
Descripción	Las herramientas software necesarias no funcionan o dejan de estar disponibles cuando requeridas.
Consecuencia	Retraso en la entrega del proyecto, es posible que se haya de repetir el desarrollo de los entregables. Aplazamiento de la fecha de entrega del proyecto.
Prioridad	5/5
Probabilidad	2%
Impacto	Catastrófico
Exposición	1,92
Categoría	Ambiente e infraestructura del desarrollo
Plan de mitigación	No aplica
Plan de contingencia	Replanificar las herramientas software a utilizar en el proyecto, utilización de software de empresas fiables. Reconstruir la planificación.
Monitorización	No aplica
Recursos estimados	Utilización de 96 horas de más para la replanificación y reconstrucción del proyecto y sus entregables.

Tabla 2.34: Definición de riesgos: R31

R32	**La curva de aprendizaje para la nueva herramienta es más larga de lo esperado**
Descripción	El personal implicado en el desarrollo requiere de más tiempo para el aprendizaje y correcta utilización de la herramienta.
Consecuencia	Retraso en la entrega del proyecto.
Prioridad	2/5
Probabilidad	60%
Impacto	Moderado
Exposición	1,80
Categoría	Ambiente e infraestructura del desarrollo
Plan de mitigación	Buscar información y manuales en internet, leer la documentación disponible de las herramientas a utilizar.
Plan de contingencia	Modificar la planificación. Documentarse adecuadamente y realizar cursos prácticos sobre las herramientas a utilizar.
Monitorización	Antes de realizar cualquier implementación, comprobar que se dispone de todos los conocimientos necesarios.
Recursos estimados	3 horas de más para el aprendizaje del funcionamiento de la herramienta.

Tabla 2.35: Definición de riesgos: R32

R33	**No se ha solicitado información al usuario, por lo que el producto final no se ajusta a las necesidades del usuario.**
Descripción	No se ha solicitado suficiente información al usuario para conocer su problemática y lo que desea.
Consecuencia	Requisitos inestables. Retraso en la entrega del proyecto. Es necesario volver a fabricar los entregables. Modificación de la planificación.
Prioridad	5/5
Probabilidad	20%
Impacto	Catastrófico
Exposición	19,20
Categoría	Usuarios finales
Plan de mitigación	Mantener una comunicación constante y suficiente con el usuario final para saber si el producto se va adaptando a sus necesidades.
Plan de contingencia	Modificar la planificación. Modificación de entregables.
Monitorización	Antes de cada iteración comprobar que se seguirán los requisitos solicitados por el usuario.
Recursos estimados	Utilización de 96 horas de más para la modificación de entregables y finalización del proyecto.

Tabla 2.36: Definición de riesgos: R33

R34	**Los requisitos no se han definido correctamente**
Descripción	Definición incompleta o incorrecta de los requisitos de usuario.
Consecuencia	Adaptación de los requisitos.
Prioridad	4/5
Probabilidad	40%
Impacto	Catastrófico
Exposición	1,60
Categoría	Requisitos
Plan de mitigación	Mantener una comunicación constante y suficiente con el usuario final para poder definir correctamente sus requerimientos.
Plan de contingencia	Modificación de requisitos en el product backlog y sprint backlog si aplica. Hacer partícipe al cliente de la definición del backlog.
Monitorización	Tras la toma de requisitos, comprobar con el usuario final si coinciden con sus necesidades.
Recursos estimados	Utilización de 4 horas de más para la especificación de requisitos, construcción de historias de usuario y desglose en tareas.

Tabla 2.37: Definición de riesgos: R34

R35	**Los requisitos se han adaptado pero continúan cambiando.**
Descripción	Tras la modificación de los requisitos en el proyecto, nos encontramos con que estos han vuelto a cambiar.
Consecuencia	Adaptación de los requisitos.
Prioridad	4/5
Probabilidad	30%
Impacto	Crítico
Exposición	0,90
Categoría	Requisitos
Plan de mitigación	Mantener una comunicación constante y suficiente con el usuario final para poder definir correctamente sus requerimientos. Hacer el cliente partícipe de las reuniones donde se define el backlog.
Plan de contingencia	Modificación de requisitos en el product backlog y sprint backlog si aplica.
Monitorización	Tras la modificación de los requisitos, comprobar con el usuario final si coinciden con sus necesidades.
Recursos estimados	Utilización de 3 horas de más para la especificación de requisitos, construcción de historias de usuario y desglose en tareas.

Tabla 2.38: Definición de riesgos: R35

R36	**Se añaden requisitos extra al proyecto**
Descripción	Tras la especificación de requerimientos, se han añadido más requisitos fuera de la planificación.
Consecuencia	Adaptación de los requisitos.
Prioridad	3/5
Probabilidad	50%
Impacto	Crítico
Exposición	1,50
Categoría	Requisitos
Plan de mitigación	Mantener una comunicación constante y suficiente con el usuario final para poder definir correctamente sus requerimientos.
Plan de contingencia	Modificación de requisitos en el product backlog y sprint backlog si aplica.
Monitorización	Antes de empezar con un sprint, comprobar con el usuario final que todas sus necesidades se recogen en el product backlog.
Recursos estimados	Utilización de 3 horas de más para la especificación de requisitos, construcción de historias de usuario y desglose en tareas.

Tabla 2.39: Definición de riesgos: R36

R37	**Los módulos más propensos a tener errores implican más carga de trabajo**
Descripción	Los módulos del programa que tienden a tener más errores necesitan más tiempo para el diseño, implementación y comprobación.
Consecuencia	Retraso en la entrega del proyecto.
Prioridad	2/5
Probabilidad	50%
Impacto	Crítico
Exposición	3,50
Categoría	Producto
Plan de mitigación	Seguimiento de la fase y tareas a realizar en cada momento y corrección de errores en cada iteración.
Plan de contingencia	Modificar la planificación. Realización de horas extra de trabajo.
Monitorización	Seguimiento de cada fase, sus tareas y los riesgos que conllevan.
Recursos estimados	Utilización de 7 horas de más para la finalización de las tareas pendientes y replanificación del calendario.

Tabla 2.40: Definición de riesgos: R37

R38	**Utilizar la última tecnología en el mercado alarga la planificación de forma impredecible.**
Descripción	Lo último en tecnología del mercado puede implicar haber poca documentación o errores de la herramienta que aún no han sido detectados.
Consecuencia	Retraso en la entrega del proyecto.
Prioridad	2/5
Probabilidad	20%
Impacto	Moderado
Exposición	0,20
Categoría	Producto
Plan de mitigación	No aplica.
Plan de contingencia	Modificar la planificación.
Monitorización	No aplica.
Recursos estimados	1 hora para la replanificación.

Tabla 2.41: Definición de riesgos: R38

R39	**Desarrollo incorrecto de las funciones/métodos software**
Descripción	Se han desarrollado incorrectamente las funciones software a emplear.
Consecuencia	Volver a diseñar e implementar las funciones.
Prioridad	2/5
Probabilidad	30%
Impacto	Crítico
Exposición	0,90
Categoría	Producto
Plan de mitigación	Comprobar mediante pruebas que el funcionamiento de la función cumple con lo esperado.
Plan de contingencia	Modificación del código fuente de las funciones implicadas.
Monitorización	Realizar la traza de la función para saber que realiza lo esperado.
Recursos estimados	Utilización de 3 horas de más para corregir el comportamiento de los métodos.

Tabla 2.42: Definición de riesgos: R39

R40	**Los requisitos de compatibilidad con el sistema a integrar necesitan tiempo extra para el diseño, implementación y comprobación**
Descripción	Las tecnologías con las que se integrará el sistema software tienen requisitos muy estrictos para permitir la integración.
Consecuencia	Retraso en la entrega del proyecto.
Prioridad	4/5
Probabilidad	30%
Impacto	Crítico
Exposición	2,40
Categoría	Producto
Plan de mitigación	No aplica.
Plan de contingencia	Modificación del diseño, volver a implementar y comprobar el funcionamiento e integración con el sistema objetivo.
Monitorización	Comprobar con antelación las prescripciones para la integración con la herramienta.
Recursos estimados	Utilización de 8 horas de más para el rediseño, implementación, integración y comprobación del correcto funcionamiento.

Tabla 2.43: Definición de riesgos: R40

2.5 Análisis del presupuesto simulado

A continuación, se detallan los elementos que se contabilizarán en el presupuesto simulado inicial del proyecto.

2.5.1 Recursos humanos

Para que el progreso del proyecto pueda ser óptimo, se debe de contar para el desarrollo del mismo de 2 profesionales del ámbito TI. A los recursos humanos se les pagaría según convenio de aplicación en el año 2021.

Desarrollador senior especializado en inteligencia artificial, con experiencia en cada ciclo del desarrollo de software. Según convenio le corresponde una retribución anual de 24.250,00 euros y mensual de 1.732,14 Euros.

Scrum master con experiencia demostrable de más de 3 años: Le corresponde una retribución anual de 23.505,72 y mensual de 1.678,98 Euros.

Los salarios del actual convenio están cuantificados para un cómputo de 1.800 horas laborales anuales; por lo tanto, para la duración total estimada de este proyecto (300 horas) se calcula un coste de 7.959,29 Euros (4.041,67 + 3.917,62). A este valor se deben de añadir los costes asociados a pagos a Seguridad Social por cada trabajador contratado por duración determinada, según la legislación española actual:

- Contingencias comunes: El 23,60% sobre el bruto calculado (BCCC).
- Desempleo (duración determinada tiempo completo): 6,70% sobre el bruto calculado (BCCC).
- FOGASA (Fondo de Garantía Salarial): 0,20% sobre el bruto calculado (BCCC).
- Formación profesional: 0,60% sobre el bruto calculado (BCCC).
- Cuota por Incapacidad Temporal: 0,80% sobre el bruto calculado (BCCP).
- Cuota IMS: 0,70% sobre el bruto calculado (BCCP).

Se presupone una remuneración mensual de 30 días naturales al mes y 22 días laborales con jornada de 8 horas de trabajo al día. Con ello, los integrantes citados del equipo Scrum realizarán de manera individual un total de 176 horas por mes (no se incluyen horas extras), hasta la finalización del proyecto.

Desarrollador senior: Grupo cotización 2, retribución mensual: La empresa debe cotizar un total de 564,65 Euros al mes. Estimación para el proyecto de 962,47 Euros.

Scrum master: Grupo cotización 8, retribución mensual: La empresa debe cotizar un total de 547,32 Euros al mes. Estimación para el proyecto de 932,93 Euros.

Los costes simulados imputados a los recursos humanos para toda la duración del proyecto pasan a ser: 9.469,08 Euros.

Se tendrá en cuenta también los gastos administrativos que genere el proyecto software de manera indirecta, estos son los derivados de la gestión administrativa de los recursos humanos, revisión y mantenimiento de los acuerdos laborales en SEPE y Seguridad Social, generación de nómina mensual y gestión de una línea telefónica fija. Estos gastos administrativos ascenderán a 800 Euros para toda la duración del proyecto.

2.5.2 Recursos materiales

Herramientas software

Los softwares que serán utilizados en el desarrollo del proyecto serán en su mayoría, las versiones gratuitas de las mismas. A excepción de los siguientes softwares:

- **Microsoft Project 2019**: Una licencia valorada en 19,90 Euros.
- **Microsoft Visio Professional 2019**: Una licencia valorada en 19,90 Euros.
- **Azure bot service**
  - Cognitive Services: Language Understanding: Se utilizará la versión gratuita de 10.000 transacciones de predicción gratis al mes. Las transacciones adicionales tendrán un incremento en el coste de 1,265 Euros por 1000 transacciones de predicción.

Hardware

Las máquinas utilizadas para el desarrollo serán 2: Para el desarrollador y el Scrum master. A cada persona se destinará un portátil Lenovo Legion de 16GB de RAM, 512GB de SSD y procesador Intel(R) Core i7 con instalación de Windows 10 Home 64 bits valorado en 1.299,99€.

Según la tabla de coeficientes de amortización lineal se observa que la categoría a la que pertenecen estos equipos es "Equipos para procesos de información", con un coeficiente lineal máximo de 25%. La duración aproximada de este proyecto es de 7 meses, por lo que la amortización obtenida a la largo de la duración del proyecto será de 568,75 Euros, este valor viene dado por la siguiente fórmula.

$$\frac{3.899,97 \times 0,25 \times 7}{12}$$

Alquiler

Se imputarán gastos asociados al alquiler de una pequeña sala de coworking. No se deben tener en cuenta los gastos asociados al consumo de luz, agua, o Internet, dado que estos costes se incluyen en el precio de alquiler del espacio de coworking. Los viernes, el personal podrá trabajar de manera 100% remota, es decir los miembros del equipo Scrum trabajarán desde sus propias residencias y utilizarán herramientas colaborativas y reuniones telemáticas para desempeñar sus labores con

normalidad. El coste asociado al alquiler del espacio laboral asciende a 500 Euros para la duración del proyecto.

Al coste total imputado a los recursos necesarios se sumará un colchón económico que supone el 5% del coste calculado, de esta manera se podrá hacer frente a riesgos que se materialicen y afecten negativamente a la consecución del proyecto.

Recurso	Coste imputado en Euros (€)
Recursos humanos	9.469,08
Administración	800,00
Alquiler	500,00
Licencias y software	115,72
Equipos informáticos	568,75
Subtotal	11.453,55
Colchón del 5%	2.527,03
Total	**12.026,23**

Tabla 2.44: Presupuesto simulado inicial

2.6 Análisis de costes finales

Dado que el actual proyecto es un Trabajo de Fin de Grado no remunerado en el que participará únicamente la alumna y la tutora responsable de la supervisión del proyecto, no se imputará ningún coste a la contratación de estos miembros del equipo ni de otros profesionales, dado que será la alumna quien desempeñe el papel del desarrolladora, mientras la tutora desempeñará el rol de Scrum Master.

Coste asociado al alquiler: Parte proporcional del coste del alquiler de la vivienda actual de la alumna se imputará al proyecto, esto corresponde a un total de 162,50 Euros para la duración total del proyecto.

Parte proporcional del consumo de Internet se imputarán a gastos del proyecto, estos son: 14,17 Euros para la duración total del proyecto.

Parte proporcional de gastos derivados del consumo de luz, gas y agua de la vivienda se imputarán al proyecto: Correspondiendo a 16,67 Euros para la duración total del proyecto.

Con respecto a las licencias necesarias y software: Se utilizará la licencia de la Universidad de Valladolid para la obtención gratuita del software de MS Project 2019 y Visio Professional 2019. Se imputa un gasto de 150€ por la utilización de servicios de Azure no incluidos en la licencia de la UVa.

Con respecto al equipo informático: no se imputa ningún gasto debido a que los costes de los equipos informáticos de la alumna y tutora responsable, ya se encuentran totalmente amortizados.

Finalmente, no se aplica ningún tipo de colchón económico para subsanar la materialización de riesgos en el proyecto.

Recurso	Coste imputado en Euros (€)
Recursos humanos	0,00
Alquiler	162,50
Internet	14,17
Luz, gas y agua	16,67
Licencias y software	150,00
Equipos informáticos	0,00
Total	343,34

Tabla 2.45: Coste real final

Capítulo 3

Seguimiento del proyecto software

En este capítulo se realiza el seguimiento de cada sprint llevado a cabo en este proyecto, donde se indicarán varios datos de los mismos en formato tabla, entre ellos:

- Tareas llevadas a cabo, así como su clasificación.
- Identificador único asociado a la tarea.
- Historia de usuario, si corresponde, asociada a la tarea.
- Duración empleada en horas, para realizar la tarea.
- Estado de la tarea tras la finalización del sprint.

Como ya se ha mencionado en el capítulo de Planificación, aquellos ítems del sprint backlog que no se hayan podido completar en la duración de un sprint, se añadirán automáticamente al siguiente sprint backlog.

En la columna **Tipo** se indicará con:

- **HU (historia de usuario)**: Si la tarea implica el desarrollo de una nueva funcionalidad
- **Chore**: Tareas que no forman parte de ninguna historia de usuario pero su ejecución es necesaria en el proyecto, varias historias de usuario pueden verse afectadas por su ejecución.
- **Bug**: Si la tarea implica corrección de errores
- **Q (quality)**: Si la tarea implica la mejora de una funcionalidad ya desarrollada

3.1 Sprint 1

El primer sprint consiste enteramente en la formación de la alumna, investigación, búsqueda de información y documentación inicial para el mantenimiento de la memoria. De esta forma, en este sprint se ha abordado la documentación correspondiente a la introducción, planificación del proyecto, gestión de riesgos y costes, así como la documentación correspondiente al seguimiento del sprint 1.

ID	Tipo	Tarea	Duración	Estado
T001	Chore	Formación inicial	20h	Completado
T002	Chore	Documentación de la introducción	12 h	En progreso
T003	Chore	Documentación de la planificación y calendarización	9h	Completado
T004	Chore	Documentación del plan de riesgos y costes	9h	En progreso
T005	Chore	Documentación del seguimiento del sprint 1	0,5h	Completado
Total			**50,5h**	**Tareas pendientes**

Tabla 3.1: Desglose del sprint 1

3.2 Sprint 2

El segundo sprint ha iniciado con la documentación de diversas secciones de la introducción, planificación y gestión de riesgos, que aun no habían sido abordadas y quedaron como tareas pendientes en el sprint anterior. Igualmente se inició la documentación correspondiente a las tecnologías utilizadas y la definición de las historias de usuario iniciales a considerar en el proyecto (backlog inicial).

Por último se ha realizado una evaluación del impacto del software sobre la privacidad de los datos del usuario/a, con la intención de adecuar su funcionalidad de manera que su uso no consista en un riesgo o atentado hacia la privacidad de los datos, pudiendo ser estos sensibles.

ID	Tipo	Tarea	Duración	Estado
T002	Chore	Documentación de la introducción	7h	En progreso
T004	Chore	Documentación del plan de riesgos y costes	12h	Completado
T006	Chore	Documentación de las tecnologías a utilizar	5h	En progreso
T007	Chore	Definición inicial de historias de usuario	1h	Completado
T008	Chore	Estudio y evaluación del impacto sobre la privacidad de los datos	2h	Completado
T009	Chore	Documentación del seguimiento del sprint 2	0,5h	Completado
Total			**27,5h**	**Tareas pendientes**

Tabla 3.2: Desglose del sprint 2

3.3 Sprint 3

En este sprint, se han retomado las tareas no finalizadas en el sprint anterior que consistían en la redacción de la documentación sobre la introducción y tecnologías utilizadas.

Se ha proseguido con el desglose en tareas de las historias de usuario y el diseño inicial de la arquitectura del sistema. Posteriormente se ha preparado el entorno de desarrollo, descargando todas las herramientas necesarias, como el editor de código fuente, herramientas de depuración así como los paquetes de software precisos para el desarrollo del agente conversacional.

En este sprint se ha abordado la primera tarea proveniente de una historia de usuario. Esta historia de usuario no era la de mayor prioridad, sin embargo se ha decidido proseguir con su implementación debido a que presentaba una dificultad baja para llevarla a cabo y posibilitaría un aprendizaje rápido sobre el uso del SDK de Bot Framework.

ID	Tipo	Tarea	Duración	Estado
T002	Chore	Documentación de la introducción	5h	Completado
T006	Chore	Documentación de las tecnologías a utilizar	4h	Completado
T010	Chore	Desglose de historias de usuario en tareas	2h	Completado
T011	Chore	Planteamiento inicial del diseño y arquitectura	6h	Completado
T012	Chore	Preparación del entorno de desarrollo	2h	Completado
T013	HU05	Implementación de diálogo inicial de saludo	7h	Completado
T014	HU05	Pruebas unitarias	0,5h	Completado
T015	Chore	Documentación del seguimiento del sprint 3	0,5h	Completado
Total			**27h**	**Completado**

Tabla 3.3: Desglose del sprint 3

3.4 Sprint 4

En este sprint se han abordado tareas de la historia de usuario HU03, la cual está caracterizada por preguntas genéricas o específicas por parte del usuario/a y una respuesta acorde al contexto por parte del chatbot, para ello ha sido necesario alimentar la base de conocimiento que consta enteramente de pares de pregunta-respuesta (question and answers).

Por otro lado, esta historia también concibe la respuesta a preguntas sobre consejos para identificar actitudes machistas en el día a día, así como sugerencias para poder aplicar un enfoque de género a diversas cuestiones y recomendaciones éticas sencillas para ejercer el respeto hacia las mujeres y de esta manera ayudar en la lucha por la igualdad de género.

Para llevarlo a cabo ha sido necesario alimentar el servicio de procesamiento de lenguaje natural LUIS con intenciones y frases que representen el propósito del usuario/a, para posteriormente entrenar el modelo y realizar pruebas unitarias para verificar que la intención de las oraciones eran correctamente predichas.

ID	Tipo	Tarea	Duración	Estado
T016	HU03	Alimentación de base de conocimiento de Q&A	9h	En progreso
T017	HU03	Implementación de consejos y sugerencias feministas	10h	Completado
T018	HU03	Implementación de consejos para la identificación de comportamientos machistas	8h	Completado
T019	HU03	Incorporación de intenciones a LUIS y entrenamiento del modelo - Petición de consejos feministas y ayuda en la identificación de comportamientos machistas	3h	Completado
T020	HU03	Pruebas unitarias	0,5h	Completado
T021	Chore	Documentación del seguimiento del sprint 4	0,5h	Completado
Total			**31h**	**Tareas pendientes**

Tabla 3.4: Desglose del sprint 4

3.5 Sprint 5

En este sprint se ha retomado una tarea pendiente del sprint anterior y continuado con las tareas correspondientes a las historias de usuario HU01, HU02 y HU05, además se han creado dos nuevas tareas (T025 y T027) cuya finalidad es mejorar la experiencia del usuario/a.

La tarea T025 consiste en la creación de un diálogo de petición de retroalimentación al usuario/a, una vez se despide o desea finalizar la conversación, esta solicitud no tiene lugar siempre, sino que ocurre al azar, con una probabilidad del 33%. Su finalidad es conocer la opinión del usuario para así averiguar de qué manera mejorar su experiencia.

Por otro lado, la tarea T027 consiste en la implementación del envío de mensajes proactivos por parte del chatbot, con el fin de retomar el diálogo, una vez ha transcurrido un tiempo de margen

de inactividad por parte del usuario/a; de esta manera se intenta mantener el diálogo de forma constante en el tiempo.

ID	Tipo	Tarea	Duración	Estado
T016	HU03	Alimentación de base de conocimiento de Q&A	4h	En progreso
T022	HU02	Creación de tarjeta de términos y condiciones	1h	Completado
T023	HU02	Incorporación de tarjeta en diálogo de saludo inicial	3h	Completado
T024	HU02	Pruebas unitarias	0,5h	Completado
T025	HU03	Implementación básica del diálogo de solicitud de retroalimentación	2h	Completado
T026	HU03	Pruebas unitarias	0,5h	Completado
T027	HU05	Implementación de mensajes proactivos	4h	Completado
T028	HU05	Pruebas unitarias	0,5h	Completado
T029	HU01	Implementación de petición por nombre de usuario/a	1h	Completado
T030	HU01	Incorporación de intenciones a LUIS y entrenamiento del modelo - Detección del nombre, saludos y peticiones de ayuda	2h	Completado
T031	HU01	Pruebas unitarias	0,5h	Completado
T032	Chore	Documentación del seguimiento del sprint 5	0,5h	Completado
Total			**19,5h**	**Tareas pendientes**

Tabla 3.5: Desglose del sprint 5

3.6 Sprint 6

En este sprint se ha continuado con la tarea pendiente del sprint anterior T016; se han añadido mejoras visuales a las respuestas del bot e integrada la utilidad de estadísticas del Instituto Nacional de Estadística en España, con ello se ha empezado también la implementación del diálogo del INE, el cual tratará las preguntas del usuario/a si su intención es obtener datos estadísticos oficiales sobre la violencia de género en España.

Para que el chatbot pueda identificar la intención del usuario/a de obtener datos generales del INE, se han incorporado frases de ejemplo a LUIS, creado las entidades necesarias para el tratamiento de información clave y posteriormente entrenado el modelo.

ID	Tipo	Tarea	Duración	Estado
T016	HU03	Alimentación de base de conocimiento de Q&A	2h	En progreso
T033	Q	Mejoras visuales de preguntas y respuesta mediante tarjetas	3h	Completado
T034	HU04	Integración de la utilidad estadísticas del INE (España)	4h	Completado
T035	HU04	Recopilación de datos relevantes del INE (generales)	1.5h	Completado
T036	HU04	Incorporación de intenciones y entidades a LUIS, entrenamiento del modelo - Datos del INE (datos generales)	7h	Completado
T037	HU04	Implementación del diálogo INE	6h	En progreso
T038	HU04	Pruebas unitarias	1h	Completado
T039	Bug	Resolución de bugs	1h	Completado
Total			**25,5h**	**Tareas pendientes**

Tabla 3.6: Desglose del sprint 6

3.7 Sprint 7

En este sprint se ha continuado con la tarea pendiente T016. Esta tarea ha abarcado varios sprints y consumido más tiempo de lo esperado en su realización. Esto se debe a que ha sido necesario introducir muchos más pares pregunta-respuesta para poder ampliar el conocimiento del bot y además para que este pueda dar respuestas coherentes a una pregunta, se tuvo que informar varias preguntas formuladas de distinta manera para que así la predicción de la respuesta fuese correcta. Esta tarea se podía haber dividido en otras de menor duración que trataran de abarcar algún tema en específico; esto habría resultado en la finalización de cada tarea en sus respectivos sprints.

Igualmente se ha proseguido con la implementación del diálogo INE y se han añadido las entidades e intenciones a LUIS, correspondientes a datos estadísticos específicos del INE.

Se han añadido dos tareas de mejora de calidad de la experiencia: Mejoras en la respuestas dadas, haciéndolas más amigables y control del número de veces que el usuario/a se encuentra en un mismo estado del diálogo, con esto no se trata de reconducir el flujo del diálogo o cancelarlo, sino realizar la petición de la información necesaria de manera menos reiterante y más agradable. Tras la implementación, se ha continuado con las pruebas oportunas y resolución de diversos errores detectados.

ID	Tipo	Tarea	Duración	Estado
T016	HU03	Alimentación de base de conocimiento de Q&A	2h	Completado
T040	HU04	Recopilación de datos relevantes INE (específicos)	1h	Completado
T041	HU04	Incorporación de intenciones y entidades a LUIS, entrenamiento del modelo - Datos del INE (datos específicos)	7h	En progreso
T037	HU04	Implementación del diálogo INE	17h	En progreso
T042	Q	Mejoras en las respuestas del diálogo INE	2h	Completado
T043	Q	Control del número de veces que se ha pasado por un estado	1h	Completado
T044	HU04	Pruebas unitarias	1h	Completado
T045	Bug	Resolución de bugs	2h	Completado
Total			**33h**	**Tareas pendientes**

Tabla 3.7: Desglose del sprint 7

3.8 Sprint 8

En el sprint 8 se ha seguido trabajando en la base de conocimiento de preguntas y respuestas, en esta ocasión se han añadido imágenes y vídeos a diversas respuestas, con el objetivo de ayudar en la explicación de los términos tratados.

Se ha implementado la funcionalidad de visualización mediante gráficos, de las series de datos estadísticos extraídas del INE, que fue posible integrando las librerías de Seaborn y Pandas.

En este sprint se han finalizado también las dos tareas que estaban pendientes de sprints anteriores (T037 y T041).

ID	Tipo	Tarea	Duración	Estado
T046	HU03	Alimentación de base de conocimiento de Q&A (imágenes y vídeos)	2h	Completado
T041	HU04	Incorporación de intenciones y entidades a LUIS, entrenamiento del modelo - Datos del INE (datos específicos)	2h	Completado
T047	HU04	Visualización de datos con gráficos en diálogo INE	14h	Completado
T037	HU04	Implementación del diálogo INE	8h	Completado
T048	HU04	Pruebas unitarias	1h	Completado
T049	Bug	Resolución de bugs	5h	Completado
T050	Chore	Documentación del seguimiento de los sprints 6, 7 y 8	1h	Completado
Total			**33h**	**Completado**

Tabla 3.8: Desglose del sprint 8

3.9 Sprint 9

Durante este sprint se ha mejorado la implementación del diálogo Feedback, que permite al usuario poder valorar y puntuar su satisfacción con el chatbot. Además de mejorar algunos aspectos del diálogo INE en la identificación de entidades.

Se ha finalizado la sección de documentación sobre pruebas guiadas por casos de uso y resuelto bugs que se han encontrado durante las pruebas en funcionamiento local.

ID	Tipo	Tarea	Duración	Estado
T051	Q	Mejoras diálogo Feedback	3h	En progreso
T052	Chore	Revisión y correcciones de documentación	1h	Completado
T053	Chore	Documentación de pruebas guiadas por casos de uso	2h	Completado
T054	Q	Mejoras diálogo INE	4h	Completado
T055	Q	Mejoras mensaje de ayuda	1h	Completado
T056	Bug	Resolución de bugs	2h	Completado
Total			**13h**	**Tareas pendientes**

Tabla 3.9: Desglose del sprint 9

En este sprint se ha detenido el desarrollo del proyecto de manera temporal, esta interrupción ha tenido una duración de varios meses hasta que finalmente se retoma el desarrollo en el sprint 10. Este inciso ocurre debido a que se manifestaron diversos riesgos de los analizados inicialmente, entre ellos cabe destacar los siguientes: R01, R03, R05, R16, R17, R21 y R31.

El desarrollo del proyecto se vio afectado cuando tuvo lugar el inicio de la pandemia del virus Covid-19 y posterior confinamiento; y teniendo en cuenta que la utilización de las herramientas software eran más complejas de los esperado, ha sido necesario un retraso en la entrega del proyecto. Además, Por motivos personales y la indisposición de la desarrolladora, no se pudo seguir con el desarrollo del mismo según estaba planificado.

Cabe destacar también que el software provisto por Microsoft ha presentado diversos problemas de funcionamiento, lo que impedía su uso y por lo tanto avance del proyecto hasta que estos se solucionaran desde la parte del proveedor, o bien se analizara y llevara a cabo una solución alterna.

3.10 Sprint 10

En este sprint se ha podido finalizar las mejoras en la implementación del diálogo Feedback (T051) así como la integración de la base de datos en la nube Cosmos DB, de manera que las opiniones del usuario se queden reflejadas de manera anónima en un almacenamiento no volátil.

Se ha configurado y desplegado el sistema en Google Compute Engine para que funcione de manera ininterrumpida, además de haber realizado la conexión con el canal Microsoft Teams, la configuración necesaria y resolución de problemas derivados de este.

Se finalizó la documentación relativa a la implementación del sistema, los sprints 9 y 10 y además se llevaron a cabo mejoras en los gráficos del INE, de manera que la información al ser

enviada por el canal mantenga su claridad y resolución.

Finalmente se ha llevado a cabo una refactorización del código fuente, eliminando secciones de código irrelevantes, mejorando la legibilidad y documentación.

ID	Tipo	Tarea	Duración	Estado
T051	Q	Mejoras diálogo Feedback	2h	Completado
T057	HU06	Integración de base de datos	5h	Completado
T058	HU06	Pruebas unitarias	0.5h	Completado
T059	HU03	Desplegado servicio en Compute Engine	6h	Completado
T060	HU03	Conexión con canal Teams y pruebas	2h	Completado
T061	Chore	Revisión y correcciones de documentación	1h	Completado
T062	Chore	Documentación del capítulo Implementación	7h	Completado
T063	Chore	Refactorización de código fuente	2h	Completado
T064	Q	Mejoras en la visualización de gráficos	2h	Completado
T065	Bug	Resolución de bugs relacionados con Teams	3h	Completado
T066	Chore	Documentación del seguimiento de los sprints 9 y 10	1h	Completado
Total			**31.5h**	**Completado**

Tabla 3.10: Desglose del sprint 10

3.11 Sprint 11

En este sprint se ha llevado a cabo la finalización de diversas secciones de la documentación del proyecto, así como una revisión general de todos los capítulos donde se analizó la información reflejada en busca de incongruencias o bien actualizaciones.

ID	Tipo	Tarea	Duración	Estado
T067	Chore	Revisión y correcciones de documentación	4h	Completado
T068	Chore	Documentación del seguimiento del sprint 11 y resumen	1h	Completado
T069	Chore	Documentación del capítulo de pruebas	2h	Completado
T070	Chore	Documentación de anexos de instalación y uso	1h	Completado
T071	Chore	Documentación del capítulo de diseño	6h	Completado
T072	Chore	Documentación de conclusiones	0.5h	Completado
Total			**14.5h**	**Completado**

Tabla 3.11: Desglose del sprint 11

3.12 Resumen

Se han realizado 11 sprints para llevar a cabo el proyecto. El tiempo total empleado en estas iteraciones ha sido de 306 horas, por lo que se ha excedido de la planificación aproximadamente un total de 6 horas.

Durante el trascurso del proyecto han tenido lugar varios riesgos, algunos de los cuales han podido ser contenidos, mientras que otros han dado lugar a contratiempos que se han traducido en un retraso en la entrega del proyecto o mayor inversión del tiempo de la alumna en algunos casos.

Estos riesgos están sobre todo relacionados con la complejidad de la herramienta y la falta de experiencia en su uso, lo que obligaba a dedicar más tiempo en entender su funcionamiento; además de la poca documentación disponible, documentación incongruente o incorrecta de fuentes oficiales respecto al uso de las herramienta Microsoft con lenguaje Python, así como bugs y problemas de funcionamiento de las herramientas Microsoft con lenguaje Python, lo que llegó a impedir temporalmente pero en su totalidad el avance del proyecto o seguir con el flujo prefijado en la planificación.

Capítulo 4

Análisis del software

En este capítulo se abordará el estudio inicial del problema y se detallarán los requisitos y modelos realizados en el análisis del sistema, estos son, modelo de dominio, máquinas de estado, diagramas de actividad y de casos de uso.

Al contrario de lo que se suele pensar, las historias de usuario no son sinónimo de los requisitos del sistema, muchas veces se debe de analizar cada caso y realizar un estudio más profundo para poder extraer los requerimientos de construcción del software.

4.1 Requisitos del software

Los requisitos de un sistema software son las características y comportamientos que debe de tener el sistema a desarrollar, estos se identifican durante el análisis del problema y en este proyecto se han extraído a partir de las historias de usuario.

Siguiendo la metodología FURPS [19], los requisitos pueden pertenecer a alguna de las categorías siguientes:

- Requisitos funcionales: Definen el funcionamiento del sistema.
- Requisitos no funcionales: Definen propiedades emergentes del sistema, como los tiempos de respuesta, tipo de almacenamiento, etc.
- Requisitos de información: Es una forma especializada de un requisito funcional, describen la información que el sistema deberá almacenar.
- Reglas de negocio: Es una forma especializada de un requisito funcional, describen restricciones sobre el sistema.

4.1.1 Requisitos funcionales

1. El sistema deberá reconocer el diálogo, la intención y las entidades citadas en una entrada de datos del usuario.
2. El sistema debe responder con mensajes de texto a una pregunta que realiza el usuario de forma textual dentro del contexto de la aplicación.

3. El sistema debe de buscar información relevante y ofrecerla al usuario siempre que éste la solicite.

4. El sistema debe ofrecer al usuario distintos mensajes que ayuden a identificar la discriminación de género y formas de contrarrestarla.

5. El sistema comunicará un primer mensaje por pantalla ilustrando las políticas de privacidad que se aplicarán.

6. El sistema permitirá al usuario aceptar o rechazar los términos y condiciones que se aplicarán en la política de privacidad.

    - El sistema debe pasar a estado inactivo si el usuario rechaza los términos y condiciones.
    - El sistema debe funcionar con normalidad si el usuario acepta los términos y condiciones.

4.1.2 Requisitos no funcionales

1. El sistema debe ofrecer información útil para el usuario.

2. El sistema debe ser fácil de utilizar.

3. Las funcionalidades del sistema deben ser de fácil aprendizaje para el usuario.

4. Las respuestas que debe proporcionar el sistema tendrán un tiempo de procesamiento máximo de 7 segundos.

5. El sistema debe de estar operativo de manera continua durante todo el día.

6. El sistema deberá permitir la entrada de texto mediante el teclado.

4.1.3 Requisitos de información

1. El sistema debe almacenar en una base de datos la evaluación realizada por el usuario sobre el funcionamiento del bot.

4.1.4 Reglas de negocio

1. El sistema debe de obtener información actualizada y verídica sobre la violencia de género utilizando las bases de datos del INE.

4.2 Diagrama de casos de uso

En el análisis del sistema se pueden diferenciar varios tipos de interacciones que el usuario puede realizar con el agente conversacional, como se puede observar en la figura.

Los CUs **Iniciar conversación** y **Finalizar conversación** constan de frases y expresiones de aperturas de diálogos y cierres; junto con **Consultar manual de uso** y **Consultar Términos y Condiciones aceptadas**, estos casos de uso tienen en común el hecho de que no tienen relación

directa con el tema tratado en el diálogo. Tras iniciar el diálogo en sí mediante **Iniciar conversación**, el usuario puede realizar consultas al sistema mediante el caso de uso **Consultar chatbot**.

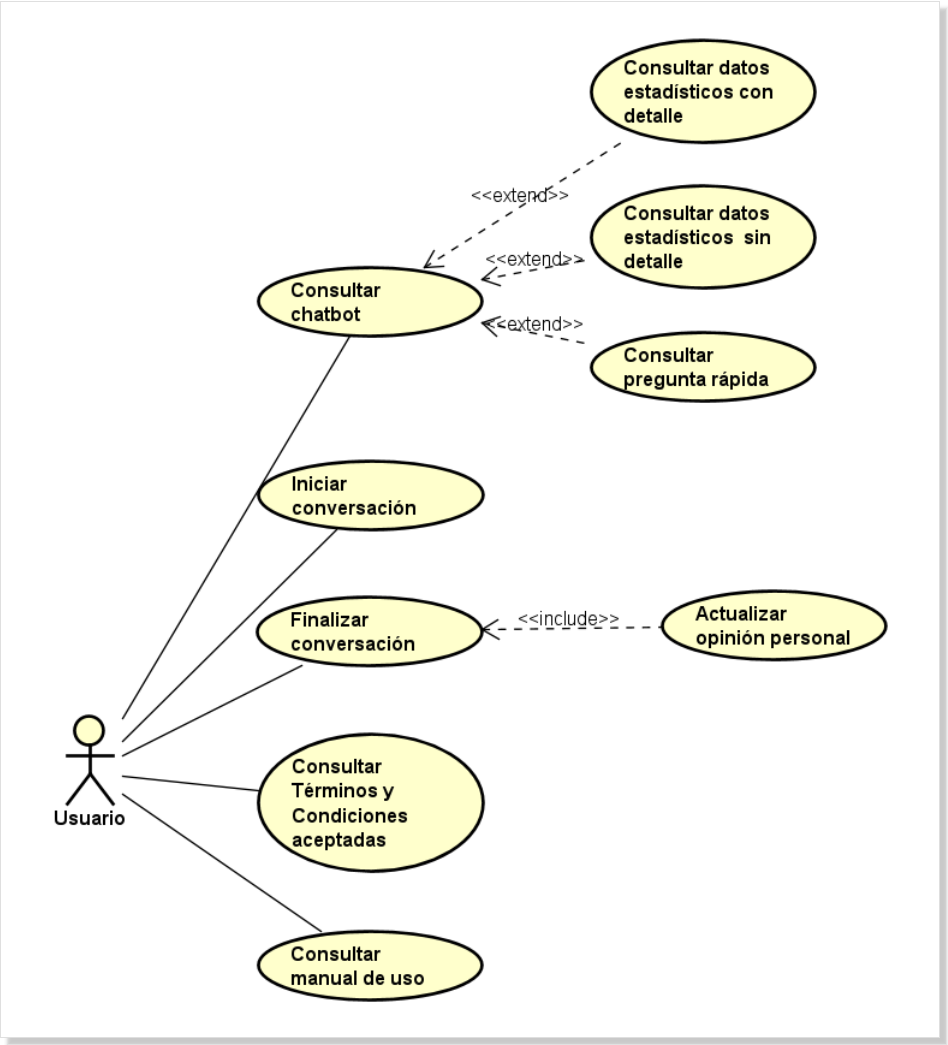

Figura 4.1: Diagrama de casos de uso del sistema

4.2.1 Especificación de casos de uso

A continuación, se describirá de forma más detenida los casos de uso del sistema.

CU-1	**Consultar chatbot**	
Descripción	El sistema deberá comportarse tal y como se describe en este caso de uso cuando el usuario requiera realizar preguntas al sistema	
Precondición	El actor ha aceptado los términos y condiciones de uso	
Secuencia normal	**Paso**	**Acción**
	1	El sistema solicita que se le realice una pregunta
	2	El usuario realiza una pregunta
	3	El sistema genera una respuesta y la informa al usuario
Postcondición	El sistema da una respuesta al usuario	
Secuencia alternativa	**Paso**	**Acción**
	2.A	Si el usuario quiere realizar una pregunta detallada sobre series de datos, se realiza el CU "Consultar datos estadísticos con detalle", a continuación este caso de uso continúa.
	2.B	Si el usuario quiere realizar una pregunta genérica sobre series de datos, se realiza el CU "Consultar datos estadísticos sin detalle", a continuación este caso de uso continúa.
	2.C	Si el usuario quiere realizar una pregunta rápida que no implique series de datos, se realiza el CU "Consultar pregunta rápida", a continuación este caso de uso continúa.

Tabla 4.1: Descripción de casos de uso: CU-1

CU-2	**Iniciar conversación**	
Descripción	El sistema deberá comportarse tal y como se describe en este caso de uso cuando el usuario vaya a iniciar una conversación con el sistema	
Secuencia normal	**Paso**	**Acción**
	1	El sistema da a conocer los términos y condiciones de uso de la aplicación y solicita al usuario que los acepte
	2	El usuario acepta los términos y condiciones de uso
	3	El sistema solicita al usuario su nombre para poder referirse al mismo
	4	El usuario introduce su nombre
	5	El sistema saluda al usuario por su nombre, muestra su guía de uso y el caso de uso finaliza
Postcondición	El usuario acepta los términos y condiciones de uso	
Secuencia alternativa	**Paso**	**Acción**
	2.A	El usuario no acepta los términos y condiciones de uso y el caso de uso queda sin efecto
	4.A	El usuario introduce una cadena no comprensible, el sistema no saluda al usuario por su nombre y muestra su guía de uso
	4.B	El usuario cancela el flujo del diálogo y el caso de uso queda sin efecto

Tabla 4.2: Descripción de casos de uso: CU-2

CU-3	**Finalizar conversación**	
Descripción	El sistema deberá comportarse tal y como se describe en este caso de uso cuando el usuario vaya a finalizar una conversación con el sistema	
Precondición	El actor ha aceptado los términos y condiciones de uso	
Secuencia normal	**Paso** 1 2	**Acción** El usuario utiliza un saludo de despedida para finalizar la conversación con el sistema El sistema se despide del usuario
Postcondición	La conversación entre el usuario y sistema finaliza	
Secuencia alternativa	**Paso** 2.A	**Acción** El sistema no comprende la intención del usuario y pide que reformule, el CU continúa en el paso 1

Tabla 4.3: Descripción de casos de uso: CU-3

CU-4	**Consultar Términos y Condiciones aceptadas**	
Descripción	El sistema deberá comportarse tal y como se describe en este caso de uso cuando el usuario requiera consultar los términos y condiciones de uso que ha aceptado	
Precondición	El actor ha aceptado los términos y condiciones de uso	
Secuencia normal	**Paso** 1 2	**Acción** El actor Usuario pide al sistema que se muestren las condiciones de uso de la aplicación El sistema aporta toda la información sobre las condiciones de uso solicitadas
Postcondición	El sistema ha informado sobre los términos y condiciones	
Secuencia alternativa	**Paso** 2.A	**Acción** El sistema no comprende la entrada del usuario y pide que reformule, el CU continúa en el paso 1

Tabla 4.4: Descripción de casos de uso: CU-4

CU-5	**Consultar manual de uso**	
Descripción	El sistema deberá comportarse tal y como se describe en este caso de uso cuando el usuario requiera consultar el manual de uso del sistema	
Precondición	El actor ha aceptado los términos y condiciones de uso	
Secuencia normal	**Paso** 1 2	**Acción** El actor Usuario pide al sistema el manual de uso El sistema aporta toda la información del manual al usuario en varios pasos
Postcondición	El sistema ha informado sobre su guía de uso	
Secuencia alternativa	**Paso** 2.A	**Acción** El sistema no comprende la entrada del usuario y pide que reformule, el CU continúa en el paso 1

Tabla 4.5: Descripción de casos de uso: CU-5

CU-6	**Consultar datos estadísticos con detalle**	
Descripción	El sistema deberá comportarse tal y como se describe en este caso de uso cuando el usuario requiera realizar consultas detalladas al sistema sobre series de datos estadísticos	
Precondición	El actor ha aceptado los términos y condiciones de uso	
Secuencia normal	**Paso**	**Acción**
	1	El usuario realiza una pregunta introduciendo el dato que desea conocer así como el periodo temporal en el que realizar la búsqueda
	2	El sistema comprueba que el dato de búsqueda y el periodo temporal son correctos, devuelve la respuesta al usuario y el caso de uso finaliza
Secuencia alternativa	**Paso**	**Acción**
	2.A	El sistema comprueba que el dato de búsqueda no es correcto y pide al usuario que introduzca un dato válido, el caso de uso continúa en el paso 1
	2.B	El sistema comprueba que el periodo de búsqueda no es correcto y pide al usuario que introduzca un periodo válido, el caso de uso continúa en el paso 1
	2.C	El sistema no comprende la entrada del usuario y pide que reformule, el CU continúa en el paso 1
	2.D	El usuario cancela el flujo del diálogo y el caso de uso queda sin efecto

Tabla 4.6: Descripción de casos de uso: CU-6

CU-7	**Consultar datos estadísticos sin detalle**	
Descripción	El sistema deberá comportarse tal y como se describe en este caso de uso cuando el usuario requiera realizar consultas sin detalle al sistema sobre series de datos estadísticos	
Precondición	El actor ha aceptado los términos y condiciones de uso	
Secuencia normal	**Paso**	**Acción**
	1	El usuario realiza una pregunta introduciendo el dato que desea conocer
	2	El sistema comprueba que el dato de búsqueda es correcto, devuelve la respuesta al usuario y el caso de uso finaliza
Secuencia alternativa	**Paso**	**Acción**
	2.A	El sistema comprueba que el dato de búsqueda no es correcto y pide al usuario que introduzca un dato válido, el caso de uso continúa en el paso 1
	2.B	El sistema no comprende la entrada del usuario y pide que reformule, el caso de uso continúa en el paso 1
	2.C	El usuario cancela el flujo del diálogo y el caso de uso queda sin efecto

Tabla 4.7: Descripción de casos de uso: CU-7

CU-8	**Consultar pregunta rápida**	
Descripción	El sistema deberá comportarse tal y como se describe en este caso de uso cuando el usuario requiera realizar preguntas rápidas y genéricas al sistema	
Precondición	El actor ha aceptado los términos y condiciones de uso	
Secuencia normal	**Paso**	**Acción**
	1	El usuario realiza una pregunta al sistema, sobre una temática tratada en el contexto de la aplicación
	2	El sistema comprueba que puede dar respuesta a la pregunta del usuario, envía la respuesta y el caso de uso finaliza
Secuencia alternativa	**Paso**	**Acción**
	2.A	El sistema no puede dar una respuesta coherente y pide al usuario que reformule la pregunta, el caso de uso continúa en el paso 1
	2.B	El sistema no comprende al usuario y pide que reformule, el caso de uso continúa en el paso 1

Tabla 4.8: Descripción de casos de uso: CU-8

CU-9	**Actualizar opinión personal**	
Descripción	El sistema deberá comportarse tal y como se describe en este caso de uso cuando el usuario requiera actualizar su opinión sobre el funcionamiento del bot	
Precondición	El actor ha aceptado los términos y condiciones de uso	
Secuencia normal	**Paso**	**Acción**
	1	El actor Usuario se despide del bot utilizando alguna expresión en español
	2	El sistema solicita al usuario que aguarde un momento antes de marcharse y pregunta sobre su experiencia durante el diálogo, mostrando las posibles opciones de respuesta
	3	El usuario selecciona la opción que aplique de entre las existentes
	4	El sistema solicita al usuario que desarrolle su opinión, invitando a que apunte formas de mejorar la experiencia que ha tenido
	5	El usuario introduce los datos requeridos
	6	El sistema actualiza la opinión del usuario e informa de ello
Postcondición	La opinión personal del usuario fue actualizada en el sistema	
Secuencia alternativa	**Paso**	**Acción**
	3.A	El usuario no introduce ningún dato válido y el caso de uso continúa en el paso 2
	3.B	El usuario cancela el flujo del diálogo y el caso de uso queda sin efecto
	5.A	El usuario no introduce ningún valor y el caso de uso continúa en el paso 6
	5.B	El usuario cancela el flujo del diálogo y el caso de uso queda sin efecto
	6.A	Si el dato introducido es ilegible, el sistema informa de ello al usuario y el caso de uso continúa en el paso 4

Tabla 4.9: Descripción de casos de uso: CU-9

4.3 Modelo de dominio

El diagrama UML a continuación representa el dominio de la aplicación en una primera etapa de análisis, cada clase representa una entidad del dominio, las cuales constan de atributos y relaciones que se representan como enlaces y sus respectivas multiplicidades, una descripción detallada de las clases, atributos, operaciones y relaciones se realizará en el capítulo de Diseño.

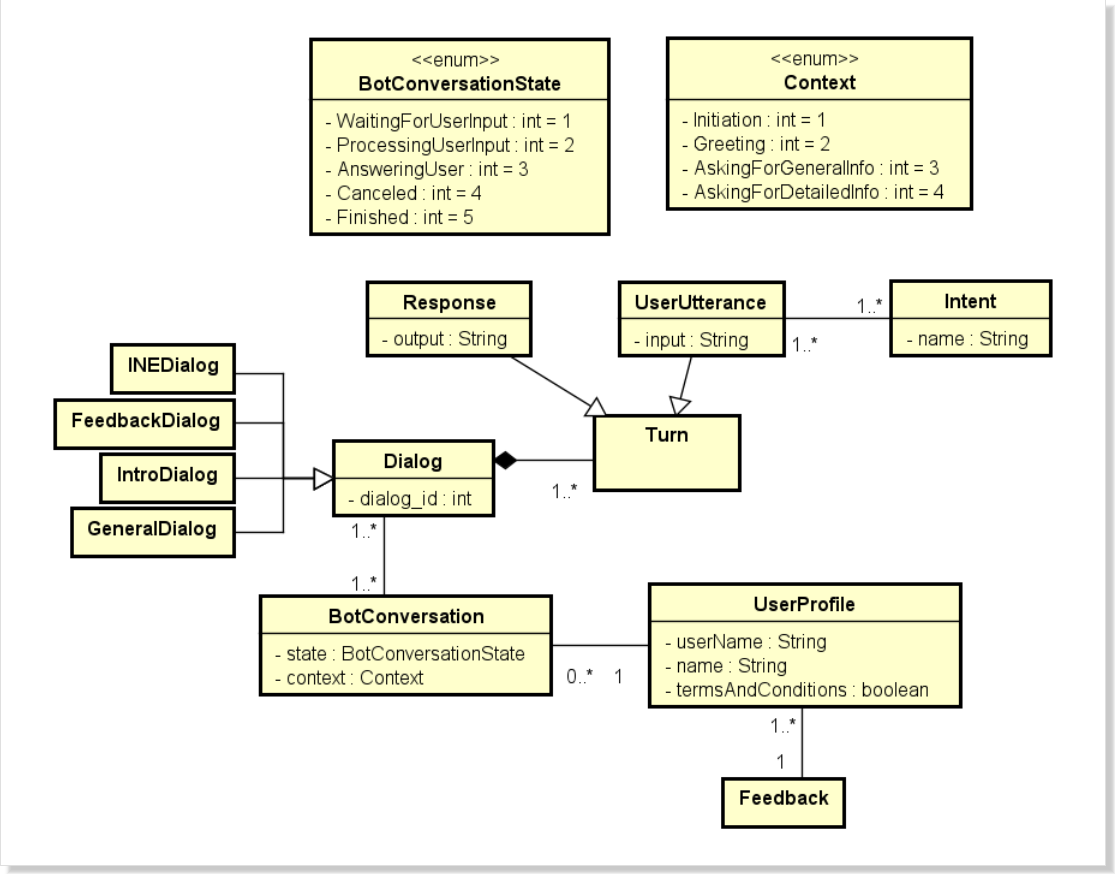

Figura 4.2: Modelo de dominio

La clase Dialog representa los diálogos que el usuario puede tener con el bot. Esta entidad tendrá como atributo su identificador único que permitirá identificar el diálogo en ejecución y distinguirlo de los demás que se pueden inicializar a continuación de este.
Un diálogo maneja información específica de cada contexto de conversación, por ello el usuario debe proveer la información necesaria en cada punto para que el bot pueda generar una respuesta correcta acorde a cada diálogo, en concreto se tratarán cuatro diálogos, INEDialog manejará los diálogos donde se requiera información del INE, FeedbackDialog permitirá al usuario introducir un feedback sobre el funcionamiento del bot, IntroDialog será el diálogo inicial que recibirá al usuario nada más empiece a funcionar, GeneralDialog manejará todas las demás peticiones del usuario.

Un diálogo está compuesto por turnos (Turn), donde se intercambian preguntas por parte del usuario (UserUtterance) y respuestas (Response) por parte del bot. La entrada del usuario tendrá normalmente una o varias intenciones (Intent) que el bot debe predecir, donde distintas frases de

entrada pueden tener la misma intención.

La clase BotConversation indica la conversación entre el usuario y el bot, una conversación consta de uno o más diálogos, y para conocer el estado del mismo se utiliza el atributo "state".

Un usuario puede tener cero o más conversaciones con el bot, aunque una vez inicializado el bot, al usuario se le asignará un UserProfile, es decir, un perfil de usuario. Para una misma conversación con el bot se mantendrá la información de este usuario y sólo se podrá crear otro perfil de usuario en caso de que el mismo elimine el historial y la conversación con el chatbot.

Cuando lo requiera el sistema, se puede solicitar que el usuario introduzca una valoración sobre su funcionamiento (Feedback), los feedbacks son únicos y es posible que un mismo usuario registre varias entradas con el paso del tiempo y uso del chatbot.

4.4 Máquinas de estado

En esta sección se detallan algunas máquinas de estado de las clases del modelo más relevantes, que pueden tener varios estados a lo largo de la duración de una conversación.

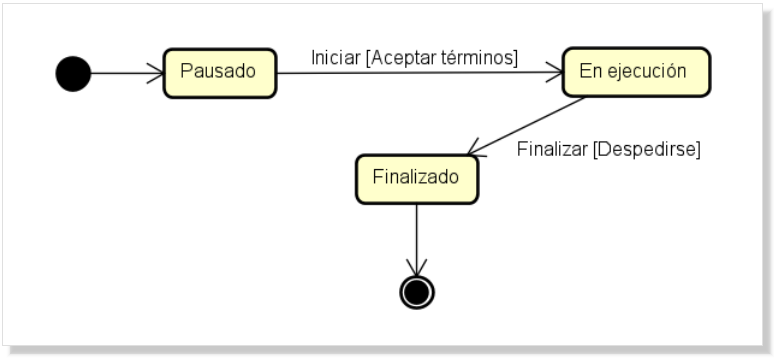

Figura 4.3: Máquina de estado de la clase BotConversation

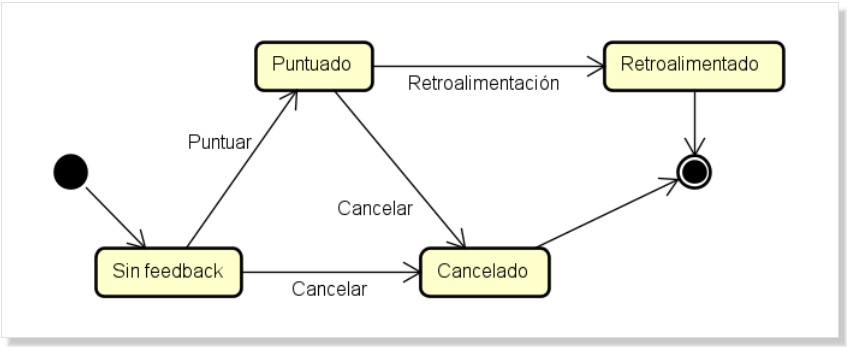

Figura 4.4: Máquina de estado de la clase Feedback

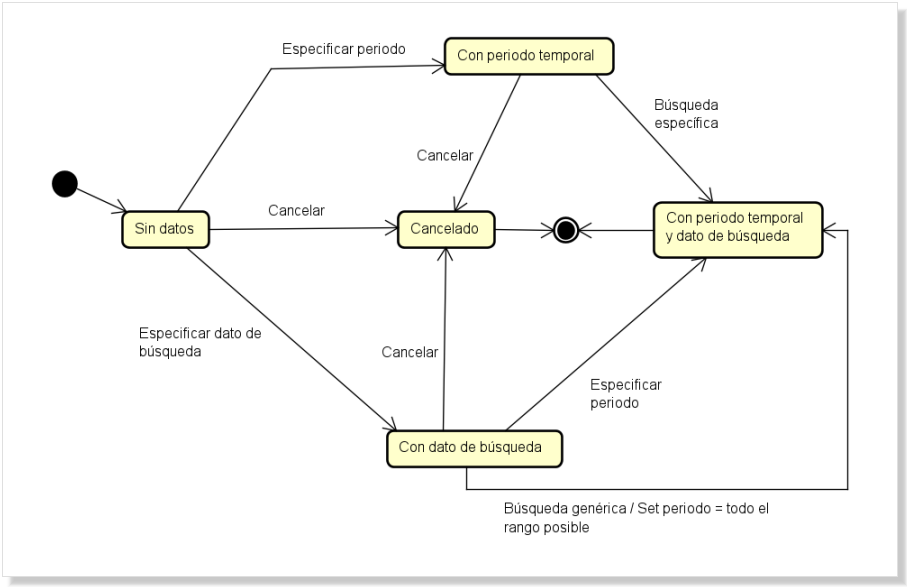

Figura 4.5: Máquina de estado de la clase INEDialog

4.5 Diagramas de actividad

Estos diagramas representan los flujos de trabajo o *workflows* que tienen lugar entre el usuario y sistema y tienen como objetivo plasmar qué acciones deben tomar lugar y determinar sus dependencias.

A través de estos diagramas podemos explicar actividades concurrentes en el sistema, decisiones e iteraciones. En esta sección se desarrollarán diagramas de actividad para los casos de uso más relevantes.

En el primer diagrama, se observa que el usuario debe iniciar la conversación, introduciendo cualquier entrada de texto válida a través del canal o aplicación destino; el chatbot detecta el mensaje y solicita al usuario que acepte las condiciones y términos de uso si aún no lo ha hecho.

En caso de que el usuario no acepte las condiciones, este podrá seguir realizando entradas de texto pero ninguna se procesará, hasta que no acepte las condiciones.

Al aceptar las condiciones, el bot solicita al usuario su nombre para poder referirse a él o ella, a partir de aquí el usuario puede escoger cancelar el flujo del diálogo y haciendo esto el bot dejará de realizarle peticiones, o bien proporcionar su nombre y continuar con el diálogo.

Cuando el usuario introduce la cadena de texto, el chatbot comprueba que realmente es un nombre y actualiza el nombre del usuario en su perfil (UserProfile); a continuación se saluda al usuario por su nombre y se muestra la guía de uso del chatbot.

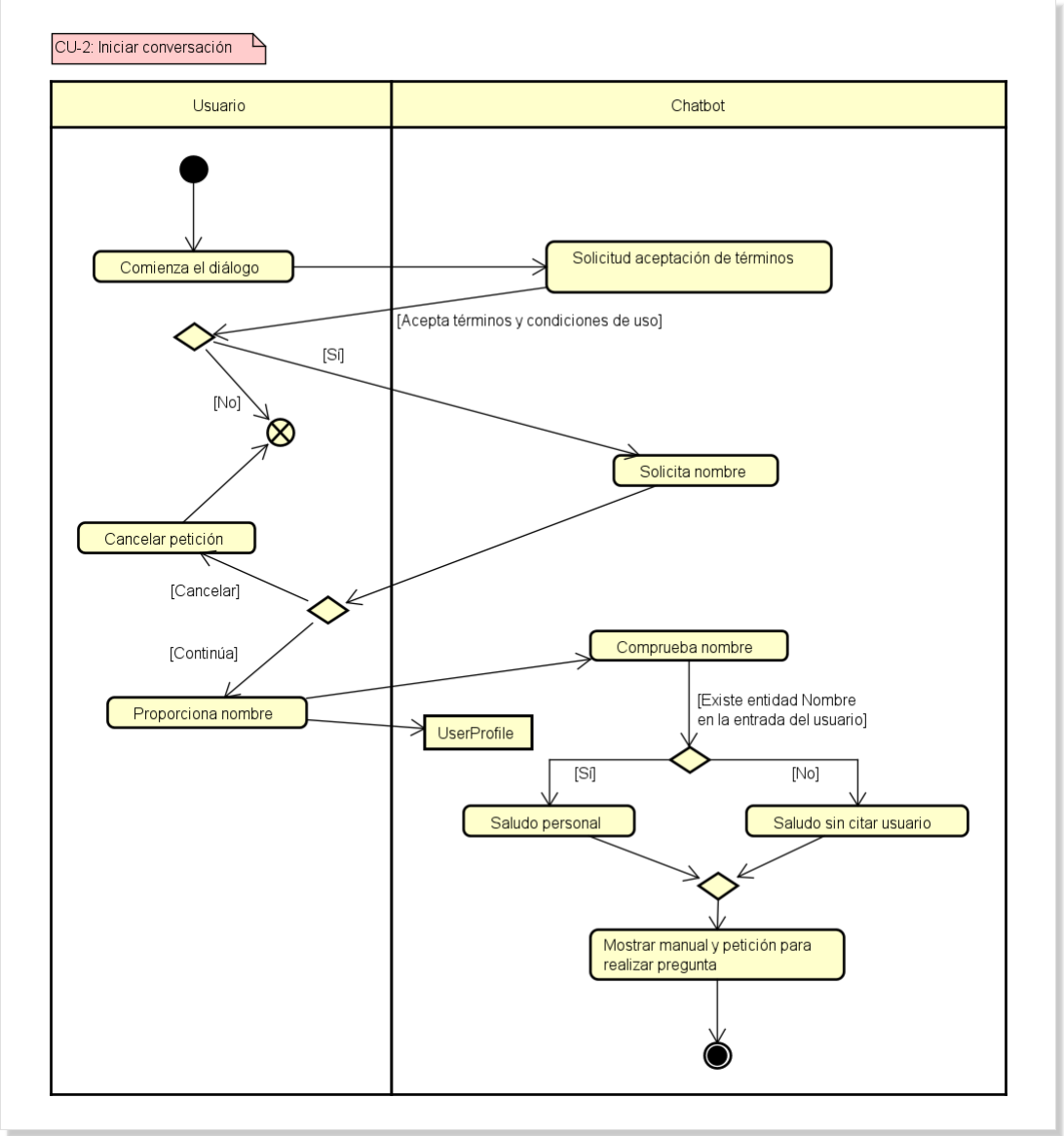

Figura 4.6: Diagrama de actividad del CU-2: Iniciar conversación

Para la realización del caso de uso 5, el usuario debe introducir una entrada de texto válida cuya intención sea la de consultar el manual o guía de uso, en caso de que el chatbot no entienda la entrada del usuario, pedirá al mismo que reformule su pregunta.

El usuario tendrá la opción entonces de reformular su pregunta o cancelar el flujo del diálogo.

En caso de que la intención del usuario sea detectada correctamente, el chatbot mostrará la guía de uso del sistema al usuario.

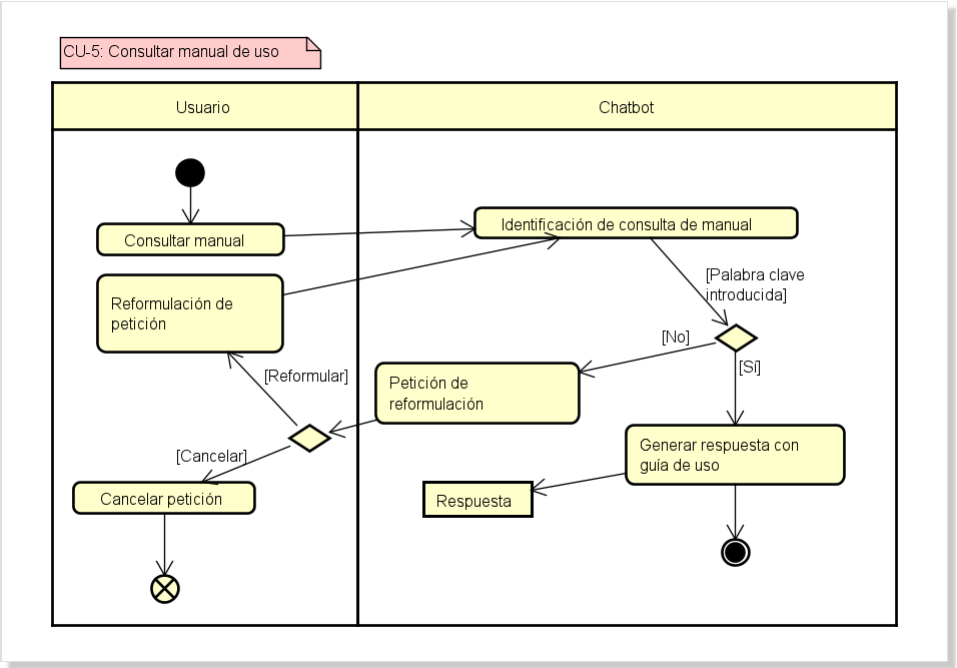

Figura 4.7: Diagrama de actividad del CU-5: Consultar manual de uso

Para la realización del caso de uso "Consultar datos estadísticos con detalle" el usuario debe empezar introduciendo una pregunta relacionada con datos estadísticos, el chatbot entonces intentará obtener todos los datos necesario para poder generar una respuesta, estos son: dato de búsqueda en concreto y periodo temporal de búsqueda. En caso de que algún dato sea incorrecto o faltante, se pedirá al usuario que vuelva a reformular su pregunta, al igual que en otras ocasiones, este tendrá la oportunidad de seguir con el diálogo o cancelar su petición.

Una vez el dato de búsqueda y el periodo temporal se hayan extraído de la entrada del usuario, el chatbot generará y mostrará una respuesta al usuario; si además el periodo temporal de búsqueda es mayor o igual a un rango de cuatro años, se ofrecerá la opción al usuario de poder visualizar los datos en una gráfica.

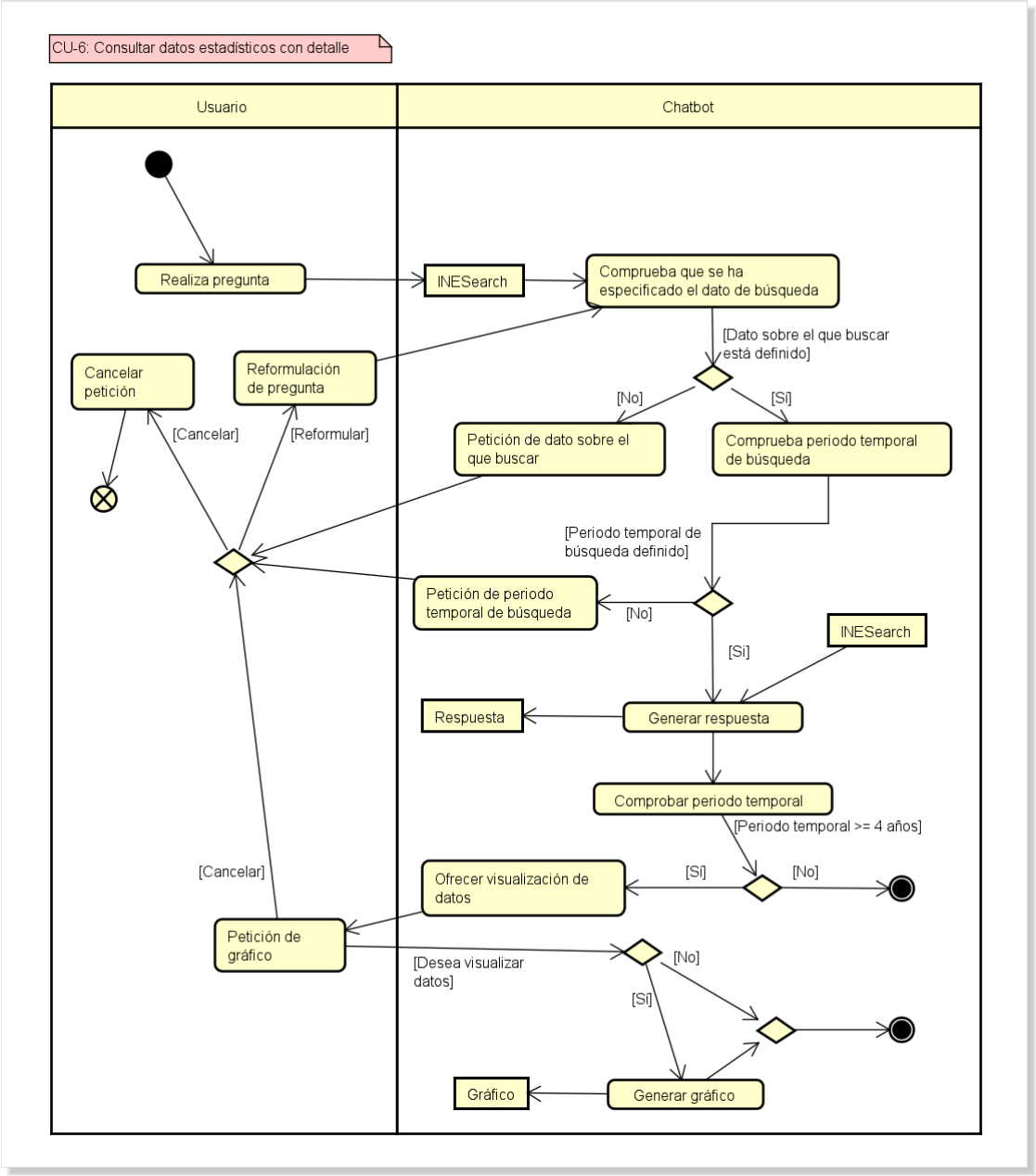

Figura 4.8: Diagrama de actividad del CU-6: Consultar datos estadísticos con detalle

El caso de uso 6 y 7 se parecen en gran medida, la diferencia entre ellos es que en este último caso, el tipo de dato que se solicita es de tipo genérico y además el usuario no introduce un periodo temporal específico dado que el rango temporal a considerar serán todos los años posibles donde se disponga de esa información.

Por ejemplo, la frase del usuario "dame el número de víctimas de violencia de género en España" se trataría como una consulta genérica, mientras que la frase "dame el número de víctimas de violencia de género en España que han sido acosadas" se trataría como una consulta específica y es obligatorio que el usuario especifique un periodo temporal.

En este flujo, después de proveer la respuesta, se genera automáticamente un gráfico que per-

mitirá al usuario observar los datos con mayor claridad.

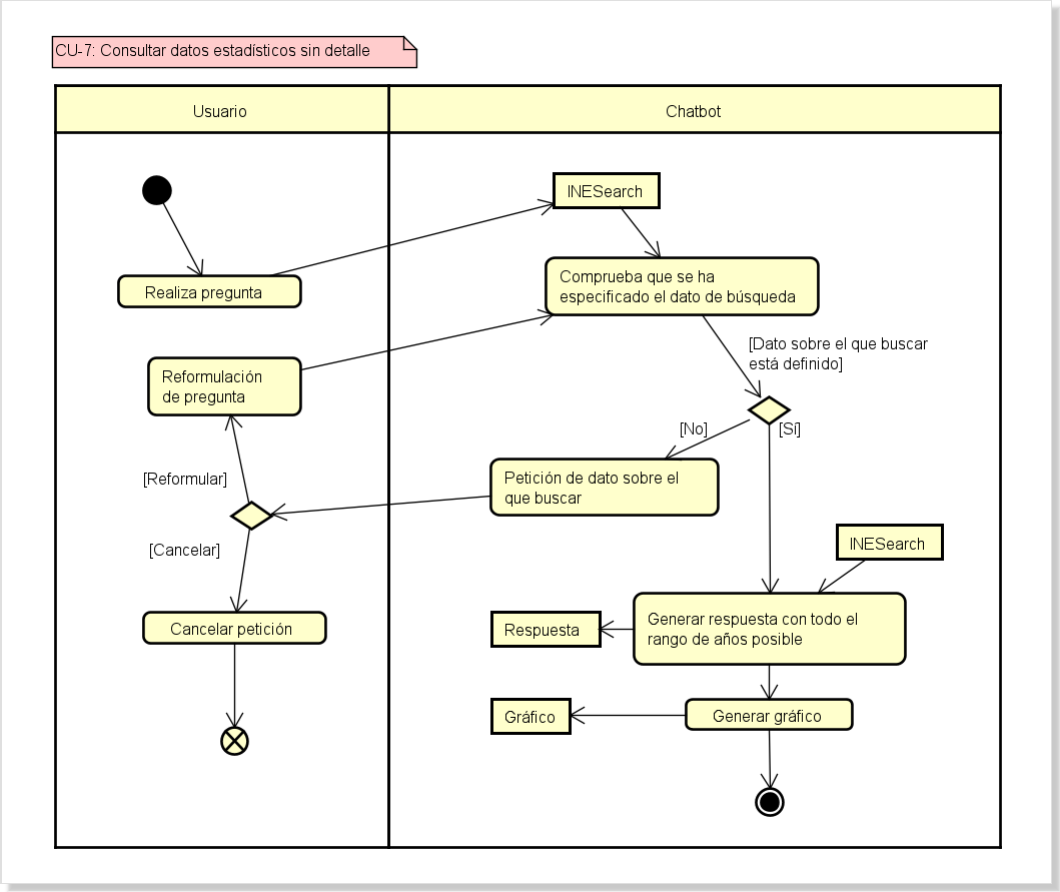

Figura 4.9: Diagrama de actividad del CU-7: Consultar datos estadísticos sin detalle

Para que el caso de uso “Actualizar opinión personal” se desarrolle, primero el usuario ha de despedirse del chatbot y que este identifique correctamente la intención del usuario de finalizar la conversación.

Al igual que en las otras veces, el usuario tiene posibilidad de reformular o cancelar el flujo del diálogo en todos sus turnos del diálogo.

Cuando el chatbot identifique correctamente la intención, este solicitará al usuario que aguarde un momento, y que puntúe su satisfacción y funcionamiento con un valor entero dentro de un rango de posibles valores, en caso de que el usuario no introduzca un valor permitido, el chatbot volverá a realizar la misma pregunta, insistiendo en que se introduzca un valor permitido.

Cuando el usuario ha introducido el valor, el flujo del diálogo pasará al siguiente estado, donde el chatbot solicitará al usuario una pequeña retroalimentación, donde el usuario tiene la posibilidad de expandir más su respuesta anterior si así lo desea. Cuando el usuario haya introducido una respuesta, el sistema guardará la opinión del usuario y le enviará un mensaje de agradecimiento por su colaboración.

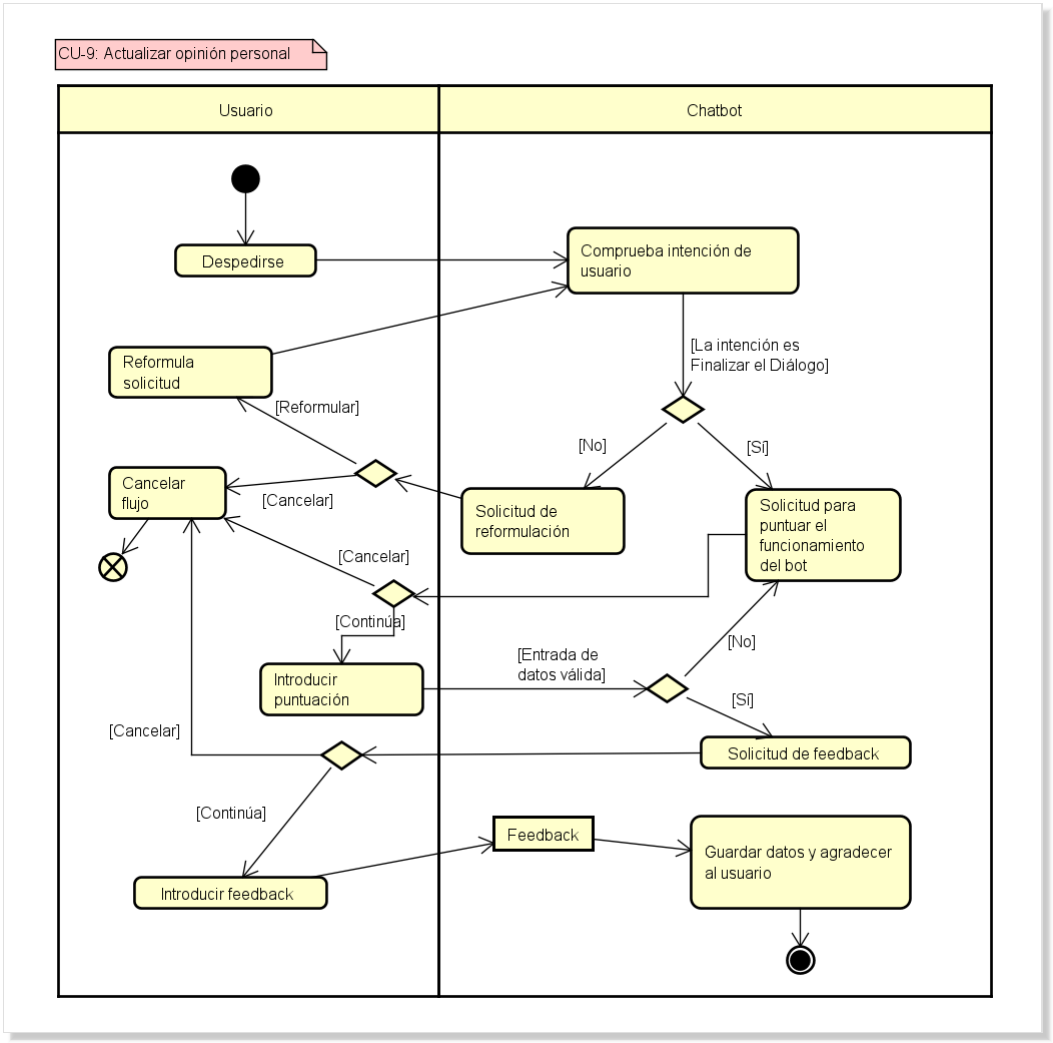

Figura 4.10: Diagrama de actividad del CU-9: Actualizar opinión personal

Capítulo 5

Diseño del software

En este capítulo se desarrollará el diseño realizado para el sistema, se detallan las decisiones de diseño tomadas y su arquitectura con más detalles que en el anterior apartado y con el foco en la implementación del sistema.

5.1 Decisiones de diseño

A continuación, se detallan las decisiones de diseño tomadas que marcarían las etapas de diseño e implementación.

- Se utilizará una base de datos NoSQL
- Se utilizará una arquitectura Cliente-Servidor
- Se usará el framework de Microsoft para la implementación del código fuente del chatbot
- Se utilizarán servicios de Microsoft para permitir al chatbot identificar las intenciones de las entradas de texto
- Se utilizará la plataforma de Google para desplegar la aplicación

5.2 Arquitectura

5.2.1 Arquitectura lógica

A continuación se define la arquitectura lógica para la aplicación, utilizando una arquitectura Cliente-Servidor.

Existirá una capa de interfaz que presenta los datos al cliente, pero esta es totalmente abstraída de nuestro sistema, esta capa es manejada por el proveedor de mensajería a utilizar, por lo tanto no existe implementación de la misma.

Por otro lado tenemos el servidor, que contiene toda la lógica de negocio de la aplicación. El servidor constará de un punto de acceso que permitirá aceptar peticiones HTTP del cliente, através de la API REST disponible del lado del servidor. Este punto de acceso es único y no customizable

dado que viene predefinido por Microsoft, las guías de implementación igualmente sugieren utilizar siempre este punto de acceso para manejar las peticiones de mensajes entrantes de clientes.

Existirán también paquetes de datos y recursos, en la primera se almacenará información temporal necesaria para la creación de gráficas, mientras que en "Resources" se almacenarán también temporalmente las imágenes generadas. El paquete "Config" consta de toda la configuración necesaria para que la aplicación pueda ponerse en funcionamiento, tanto la utilizada para la conexión con servicios de Microsoft, como la de seguridad.

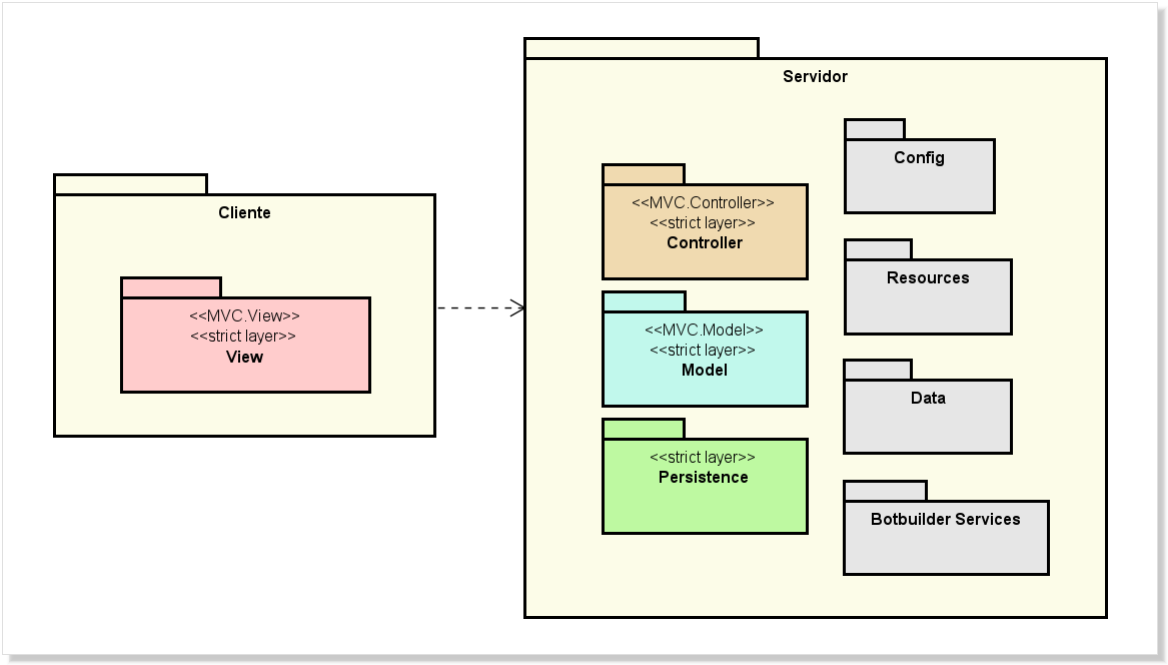

Figura 5.1: Arquitectura lógica

5.2.2 Diagrama de despliegue

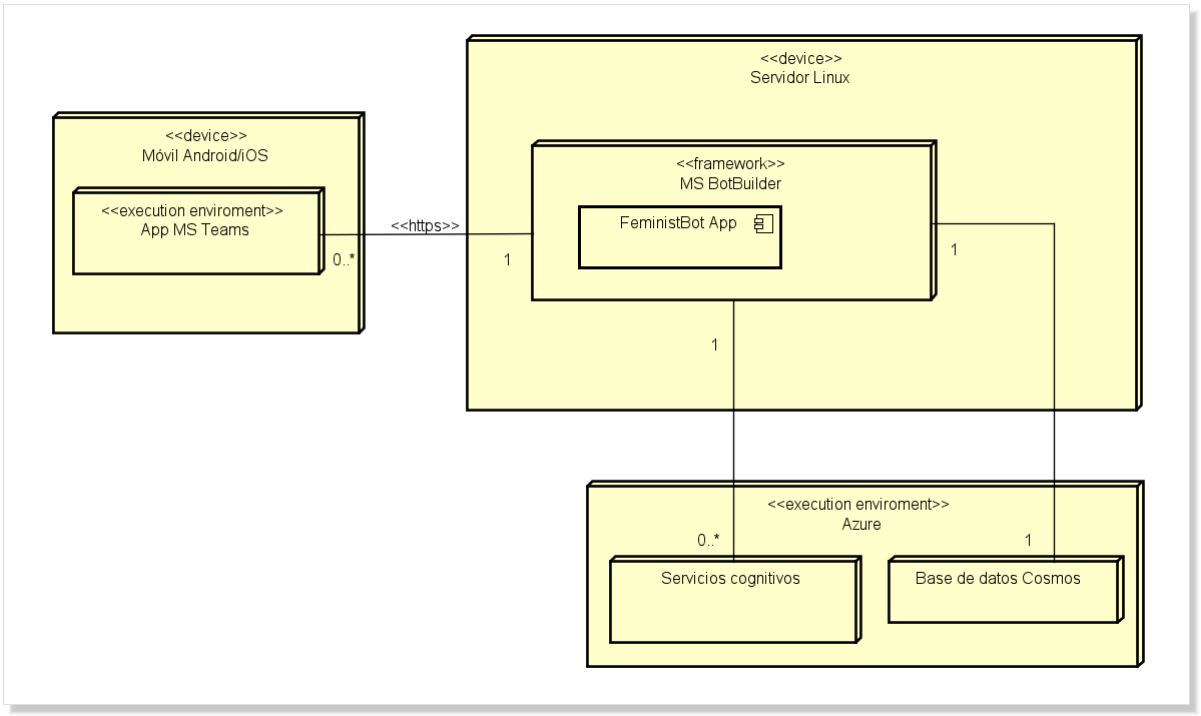

Figura 5.2: Diagrama de despliegue

5.2.3 Arquitectura del servidor

En la parte del servidor se utiliza una arquitectura por capas, diferenciando la lógica de negocio del acceso a datos, así como de las entidades del dominio.

En la capa de Controladores tendremos los gestores de la aplicación encargados de manejar la información que proviene de la vista, estos harán uso de los modelos, donde tendremos la definición de las entidades del dominio relevantes para el sistema, así como un paquete "Dialogs" que representan los diálogos que se desarrollarán entre usuario y chatbot. El Modelo se conectará con la capa de persistencia, que incluye la conexión con la base de datos y las sentencias de acceso a datos persistentes.

Se ha reflejado también a modo ilustrativo las dependencias con los servicios de Microsoft en el paquete "Botbuilder Services", Bot Builder es una colección de SDK y bibliotecas dentro de Microsoft Bot Framework que serán utilizadas en gran parte de la aplicación.

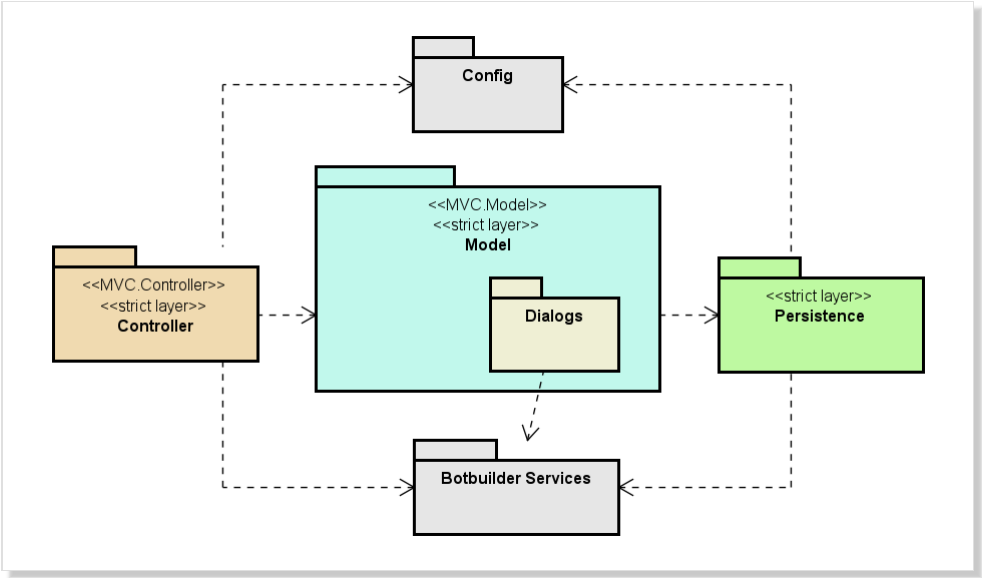

Figura 5.3: Arquitectura del lado del servidor

5.2.4 Arquitectura por capas

En la figura se ilustra el detalle de los paquetes de la aplicación. Por un lado tenemos la capa de Vista, cuya implementación es responsabilidad del proveedor de mensajería utilizado.

En el Controlador tendremos el controlador de vista "BotViewController" el cual es el primer punto de entrada a la aplicación y se encargará de recibir las peticiones provenientes de la Vista y despacharlas al controlador de la aplicación "DialogController", el cual controla el flujo de diálogos, realiza tareas genéricas y transmite y recupera información.

Dentro de la capa de Modelo, tendremos aquellas entidades relevantes para el sistema, como son "Response" que modelo la respuesta que el bot envía al cliente, o "UserProfile" que modela el perfil de usuario del cliente. Dentro de esta capa también se incluyen los diálogos que el usuario puede mantener con el chatbot, la estructura de estos diálogos siguen la propia filosofía de construcción de chatbots Microsoft e incluyen todo el manejo de información y lógica que ha de tener cada flujo de conversación específico.

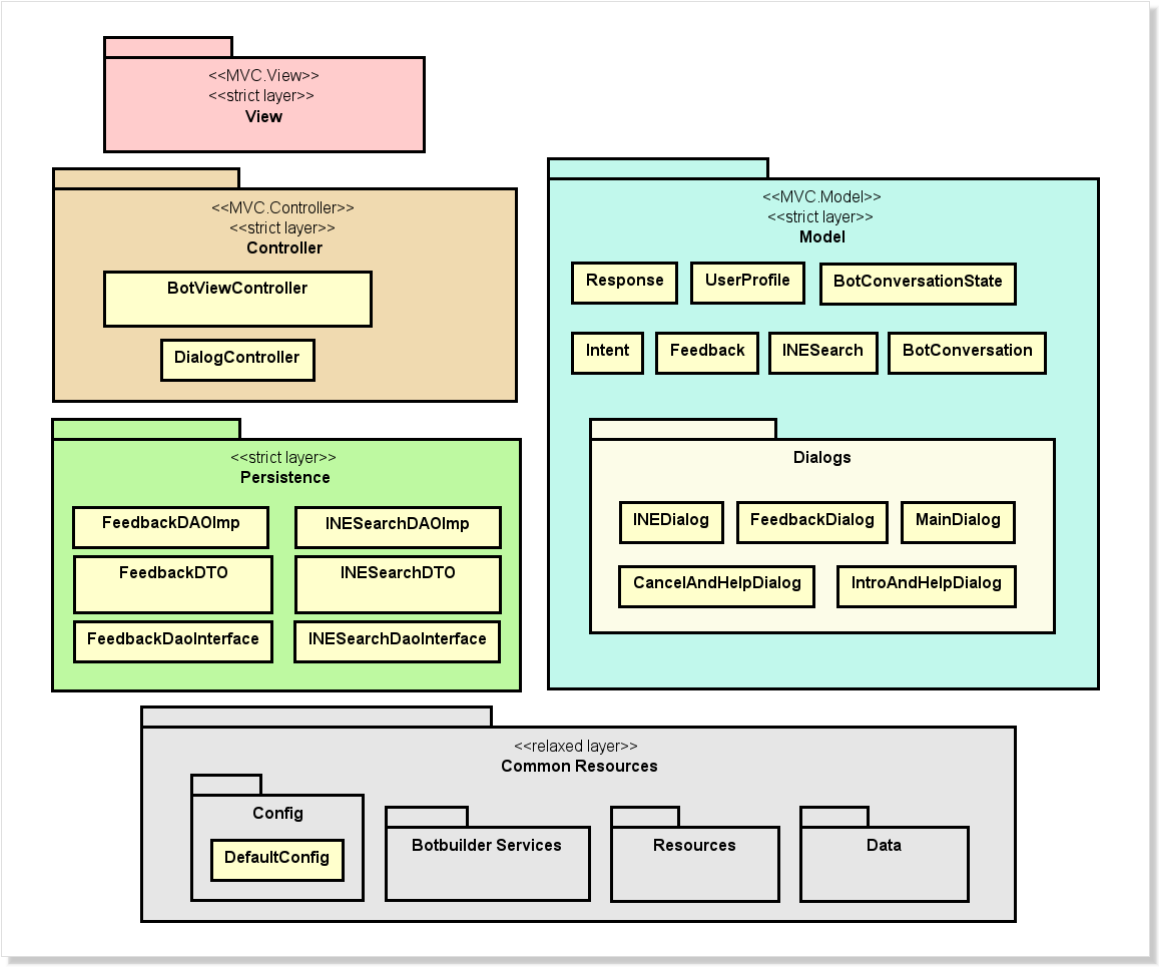

Figura 5.4: Diagrama de módulos por capas

En la capa de Persistencia tendremos clases que realizarán operaciones de persistencia, donde "FeedbackDaoImp" contendrá las operaciones para la conexión y escritura en la base de datos NoSQL y por otro lado "INESearchDaoImp" se encargará de hacer peticiones a la base de datos del INE, recuperar los datos y grabarlos de manera física y temporal en el servidor. También tendremos a las clases FeedbackDTO e INESearchDTO que modelan la transmisión u obtención de la información hacia/desde bases de datos.

Se ha ilustrado el paquete "Common Resources" que consta de paquetes de uso común en todo el sistema; en este caso, en el paquete "Data" será donde se graben los datos recuperados por "INESearchDaoImp" y en "Resources" se almacenarán los gráficos que posteriormente se enviarán al cliente.

Dependencias

Como se ha comentado antes, el controlador de vista será quién reciba las entradas del usuario, posteriormente en función del tipo de evento captado, se llamará al controlador "DialogController" para gestionarlo; este utilizará un diálogo u otro de los disponibles en el paquete "Dialogs". Los elementos del Modelo interactuarán entre sí para poder almacenar temporalmente la información

recibida.

La interacción con la Persistencia se realizará a través del Modelo, en concreto mediante clases de diálogo específicas.

Cada capa del Modelo, Vista y Controlador son estrictas: las clases de cada capa interactúan con clases de la capa inmediatamente inferior a la actual y no existen dependencias hacia el resto del sistema. Los paquetes "Resources", "Data", "Config" y "Botbuilder Services" se han representado en el paquete "Common Resources" haciendo símil a una capa relajada, a la que las demás capas pueden acceder en cualquier momento.

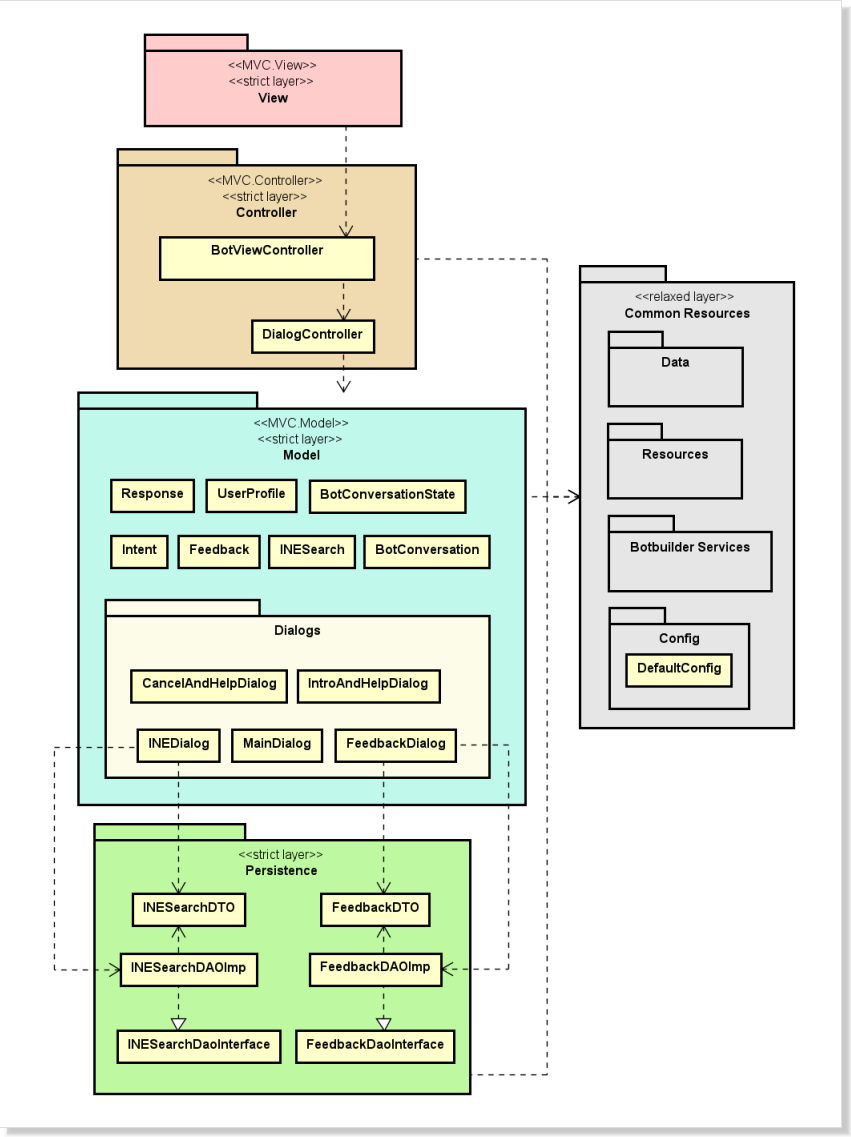

Figura 5.5: Diagrama de dependencias por capas

5.3 Patrones de diseño

En esta sección se definen los patrones de diseño software que se han utilizado para desarrollar el sistema y cómo se han aplicado.

5.3.1 Patrón MVC

El patrón arquitectónico utilizado es MVC (Modelo-Vista-Controlador) [20], el cual tiene como objetivo facilitar el desarrollo, mantenimiento y reutilización del software, para ello se separan los datos de la aplicación, la interfaz de usuario y la lógica de control en tres entidades diferenciadas, que son respectivamente el Modelo, la Vista y los Controladores.

- **Modelo**: Es donde se trabaja con la información que maneja el sistema, contiene las herramientas de acceso a los datos y lógica de negocio.
- **Vista**: Constituye la interfaz de usuario, es decir, aquellos elementos con los que el usuario puede interactuar.
- **Controlador**: Es el intermediario entre la vista y el modelo, gestionando el flujo de información entre ellos y aplicando la lógica de la aplicación.

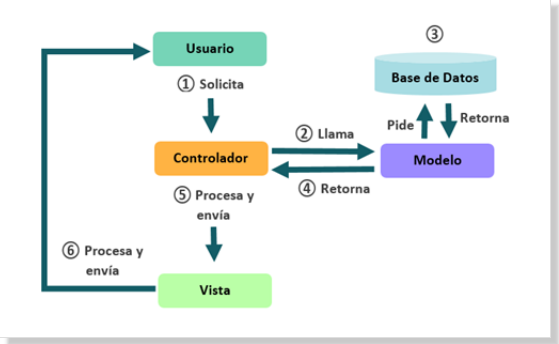

Figura 5.6: Funcionamiento de MVC

Los controladores reciben los eventos de entrada que provienen de la interacción con el usuario, y mediante las reglas de control asociadas a esos eventos, se pueden desencadenar peticiones al modelo o a la vista.

El modelo puede obtener los datos mediante mecanismos de persistencia, los cuales permiten acceder a la información y actualizar el estado de los datos, estos mecanismos constituirán la capa de persistencia.

A continuación se describirá cómo se ha aplicado el patrón en el proyecto.

- **Modelo**: Consta de la definición de aquellos elementos relevantes para el dominio de la aplicación. Dentro del esquema de implementación de chatbots Microsoft, los diálogos en sí, sus reglas y flujos se encapsulan separándolos del resto del entidades de la aplicación.

De esta manera también tendremos dentro de los modelos, la lógica de todos los diálogos utilizados en la aplicación.

- **Vista**: No se implementará una vista, dado que se hará uso del *frontend* o interfaz gráfica no customizable de la aplicación final utilizada.

  En el caso de utilizar un canal destino manejado mediante chat en texto, durante el desarrollo se tiene poca decisión sobre cómo mostrar los datos al usuario. Por ejemplo, se podría decidir si mostrar la información en formato de botón o texto, sin embargo no se puede controlar el tamaño del texto o botón.

  En lugar de la vista se implementará un controlador de vista, que se encargará de gestionar las interacciones que se pueden desarrollar entre el usuario y la vista en sí.

- **Controlador**: Se implementan controladores para realizar las tareas y eventos del sistema. Debido a la poca complejidad de los casos de uso, se implementará un controlador principal que se encargue de realizar tareas comunes y manejar la lógica de los diálogos en sí. Este utilizará elementos del Modelo si necesario. Por otro lado, el controlador de vista es quien manejará los eventos de entrada al sistema y llamará al controlador principal para satisfacer las solicitudes provenientes del cliente.

En resumen, la interacción entre capas se realizará de la siguiente manera: Desde la aplicación final, el usuario interactúa con la Vista, el Controlador recibirá los eventos provenientes de la Vista, en concreto el controlador de vista "BotViewController", mientras que el controlador de casos de uso "DialogController" realizará las operaciones necesarias y se comunicará con el Modelo, quien a su vez puede realizar peticiones a bases de datos mediante la capa de persistencia.

5.3.2 Patrón DAO y DTO

El patrón DAO (Data Access Object) nos permite separar la forma en se accede a los datos de las operaciones de negocio realizadas en el sistema, es decir, se encapsula la recuperación y el almacenamiento de los datos para que estas operaciones sean independientes de la fuente de datos que se utilice.

Esto se logra mediante la creación de una capa intermedia que actúa como un puente entre los objetos de la aplicación y la fuente de datos. Esta capa intermedia, nos permite realizar cambios en la fuente de datos sin tener que modificar directamente el código de negocio, lo cual cual mejora la modularidad del código fuente, y hace con que sea más fácil de mantener y modificar en el futuro.

El patrón DAO, utiliza generalmente interfaces para definir los métodos de acceso a datos que deben ser implementados por las clases concretas que interactúan con la fuente de datos.

En nuestro sistema, esto se logra mediante la capa "persistence" y dado que en lenguaje Python no existe el concepto en sí de interfaces como en el lenguaje Java, estas se pueden simular mediante clases abstractas sin implementación, que es lo que se logra con INESearchDaoInterface y FeedbackDaoInterface. Estas "pseudo-interfaces" se implementarán en las clases concretas INESearchDAOImp y FeedbackDAOImp respectivamente que manejarán la conexión y acceso a datos

persistentes, utilizando datos de los elementos del dominio que se requieran, en este caso el feedback del usuario y las búsquedas a realizar en el INE.

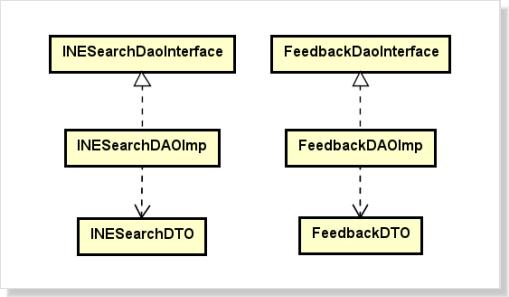

Figura 5.7: Aplicación de DAO y DTO

El formato o estructura que deben tener las búsquedas al INE o los datos del feedback se manejarán con objetos DTO (Data Transfer Object), los cuales son objetos de transferencia de datos utilizados normalmente entre capas o componentes para encapsular los datos y proveer una representación estructurada de los mismos.

Para almacenar un feedback se hará uso de FeedbackDTO que proveerá la estructura necesaria para grabar el objeto en base de datos; para obtener datos del INE se hará uso de INESearchDTO que proveerá la estructura necesaria para realizar las consultas a su base de datos.

5.3.3 Aplicación de GRASP

El término General Responsibility Assignment Software Patterns [21] se refiere a un conjunto de principios y pautas para asignar responsabilidades a clases y objetos en el diseño de software, lo que permitiría crear un diseño flexible, mantenible y de bajo acoplamiento.

Se citarán algunos principios y cómo se aplicaron:

Controlador

Este principio enuncia la definición de una clase o componente que actúe como intermediario entre las vistas y los modelos para coordinar las interacciones y el flujo de control. Esto se ha logrado aplicando el Patrón de arquitectura MVC.

Experto

Según este principio debemos asignar responsabilidad a la clase que tiene la información y el conocimiento necesario para realizar una tarea específica. En este caso podemos citar la clase "INEDialog" quien gestiona todo el flujo de información necesario para realizar una consulta al INE, y además cuenta con los métodos que permiten que estos mismos datos puedan ser reflejados visualmente en gráficos.

Creador

Se asigna la responsabilidad de crear instancias de objetos a una clase que tenga la información necesaria para tomar esa decisión. Nuevamente podemos poner de ejemplo a "INEDialog" quien instancia la clase "INESearch", esta clase modela una búsqueda sobre el INE, y el diálogo que la utiliza tiene todos los conocimientos para poder instanciarla y utilizarla.

Polimorfismo

Debemos utilizar el polimorfismo para permitir que diferentes clases implementen comportamientos similares mediante interfaces o herencia. Esto se logra en las clases del Modelo, donde los diálogos heredan de una superclase "CancelAndHelpDialog", compartiendo así funcionalidades comunes y manteniendo la lógica que debe de tener cada uno.

5.4 Diseño detallado

5.4.1 Clases del Controlador

La clase que actúa como controlador aplicando el patrón MVC es "DialogController", quien recibe peticiones del controlador de vista y se comunica con el Modelo siempre que necesario.

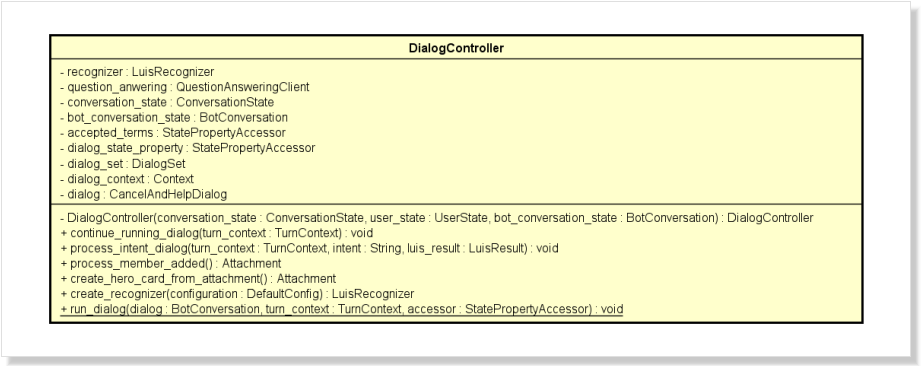

Figura 5.8: Diagrama detallado de clases del paquete Controlador

Quién gestionará los diversos eventos que tienen lugar en la vista será el controlador de vista "BotViewController"; las peticiones del usuario serán atendidas por esa clase, quien a su vez redireccionará las solicitudes al controlador "DialogController" en función del tipo de evento desencadenado.

5.4.2 Clases del Modelo

Para la implementación de la aplicación se ha partido del diagrama concebido en el análisis, sin embargo se han realizado algunos cambios sobre el mismo en cuanto se avanzaba en el proyecto.

Aparece la clase "INESearch" que modela una búsqueda a realizar sobre la base de datos del INE; se simplifican los posibles estados de "BotConversation" y desaparecen algunas clases que no aportaban mayor valor al sistema, como la clase que modelaba un turno del diálogo "Turn" o la que modelaba la entrada de datos del usuario "UsserUtterance".

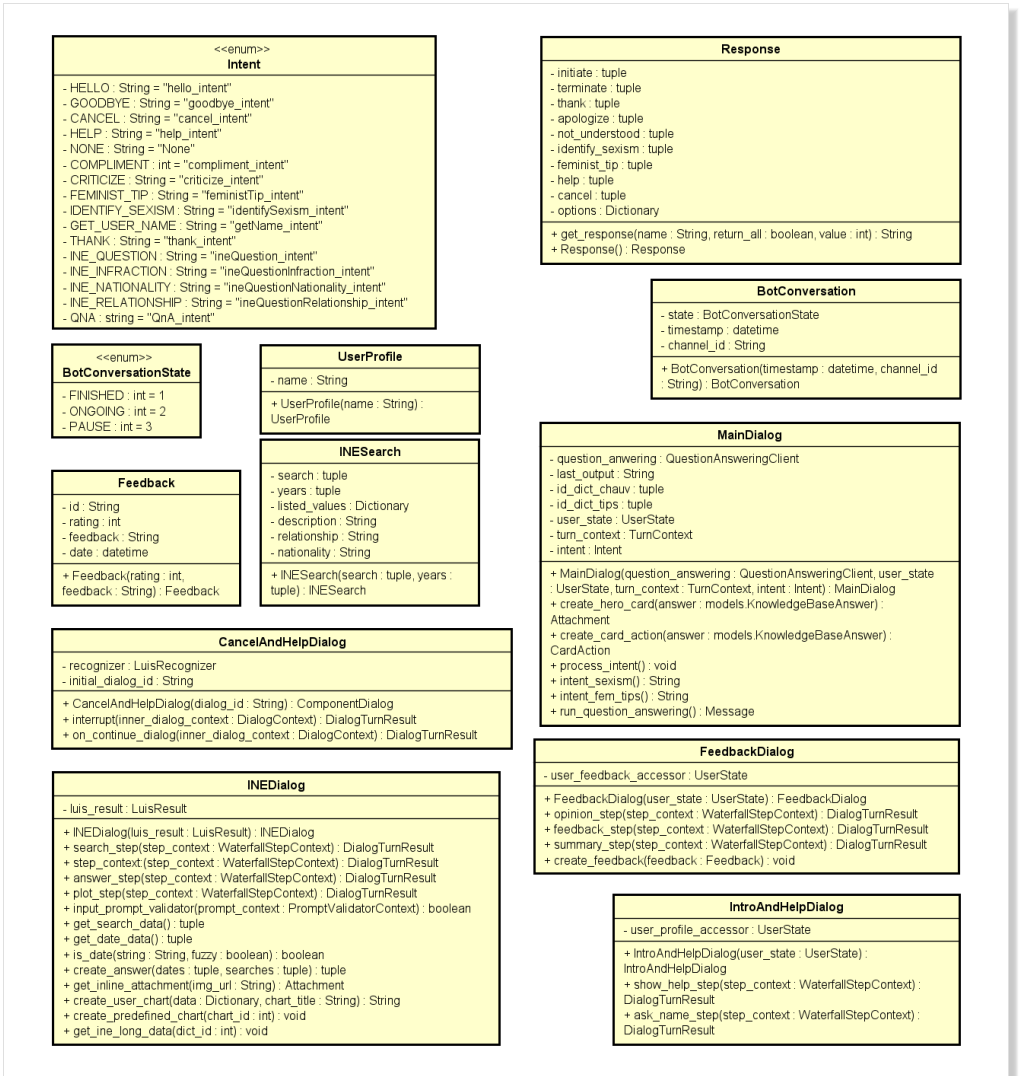

Figura 5.9: Diagrama detallado de clases del paquete Modelo

5.4.3 Clases de la Persistencia

En la capa de Persistencia tendremos a dos clases concretas que desarrollarán tareas específicas sobre la búsqueda de información en el INE o bien sobre el feedback del usuario. Estos se encargarán de escribir o leer de distintas fuentes de datos no volátiles.

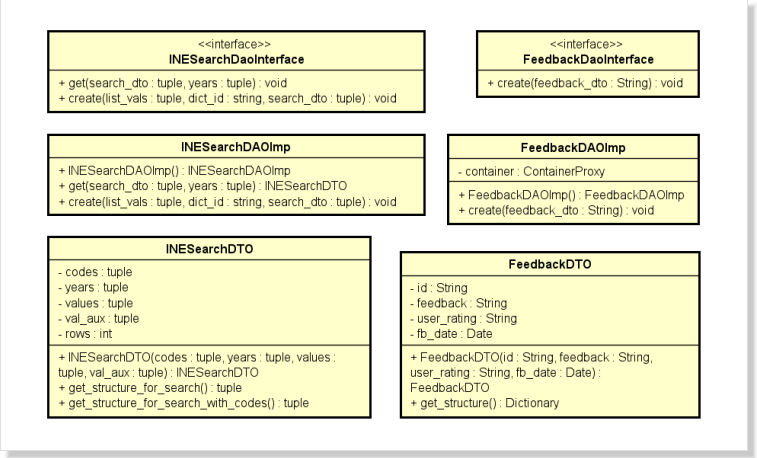

Figura 5.10: Diagrama detallado de clases del paquete Persistencia

5.4.4 Diagrama relacional

Debido a que no se almacenarán datos referente al usuario, sino únicamente su opinión con respecto a su experiencia de uso con el chatbot, solo contaremos con un contenedor de datos, cuyo esquema es el que sigue.

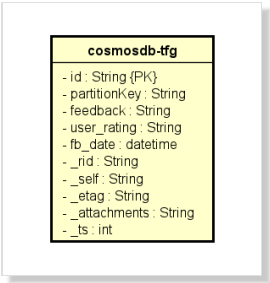

Figura 5.11: Diagrama relacional de la base de datos

Como la base de datos almacenará únicamente reseñas y opiniones del usuario, como `partitionKey` se ha definido la palabra clave **feedback**, con propósito de ampliar su uso en un futuro para almacenar otros tipos de datos.

En el campo `id` se almacenará un identificador único del ítem, en este caso, mantenido desde el propio bot. En el campo `feedback` se almacenará el feedback del usuario, en `user_rating` la puntuación sobre la satisfacción del usuario en una escala fija y en `fb_date` se almacenará la fecha en la que el usuario ha registrado su opinión.

Los demás campos, se crean automáticamente en cada nueva inserción de datos y sirven para poder identificar de forma unívoca a cada ítem.

- _rid: Identificador único del ítem
- _self: URI accesible del ítem

- _etag: Etiqueta de entidad utilizado para control optimista de la concurrencia
- _attachments: Adjuntos
- _ts: Marca de tiempo de la última actualización del ítem

5.5 Usuarios objetivo

Este sistema ha sido diseñado para su uso por parte de personas mayores de edad (+18), con cualquier tipo de nivel formativo o conocimientos sobre esta temática social. Igualmente personas que se encuentren a partir de la tercera edad y no dispongan de muchos conocimientos informáticos, pueden hacer uso de este sistema debido a que una vez instalado no requiere ningún tipo de facultad técnica avanzada para su funcionamiento, únicamente la capacidad de escritura mediante un teclado.

5.6 Diseño de la usabilidad

Los atributos de usabilidad que se utilizaron para guiar el desarrollo de este proyecto han sido los siguientes:

- **Facilidad de aprendizaje**: El mecanismo de uso de este sistema es mediante una aplicación de chat donde el usuario debe introducir entradas de texto mediante teclado que posteriormente serán procesadas, sean estas correctas o no siempre obtendrá una respuesta corta por pantalla a modo de mensajería instantánea, que guiará al usuario hacia los siguientes pasos del diálogo. No existen desviaciones de este flujo de funcionamiento, por lo que para el usuario será muy rápido su aprendizaje.
- **Facilidad de recuerdo**: El sistema funcionará mediante una aplicación de chat, lo que no supondrá un esfuerzo añadido si el usuario ya ha interactuado con otras aplicaciones de chat como WhatsApp. En efecto, si tiene conocimientos de cualquier otra aplicación de chat, utilizar el sistema será tan fácil como si de otra aplicación de mensajería instantánea se tratara.
- **Tratamiento de errores**: El manejo de errores se realiza en cada diálogo, comprobando el tipo de dato que el usuario ha proporcionado y en caso de que no sea correcto o falte información para procesar su solicitud, se notificará al usuario mediante mensajes claros sobre el dato que está requiriendo. Este proceso se hará tantas veces sea necesario para poder satisfacer la consulta de manera correcta o el usuario desista de realizarla.

5.7 Prototipado

Se adjuntarán en esta sección prototipos de diálogos realizados sobre el chatbot, donde se ilustra cómo deberían ser los diálogos entre usuario y agente conversacional.

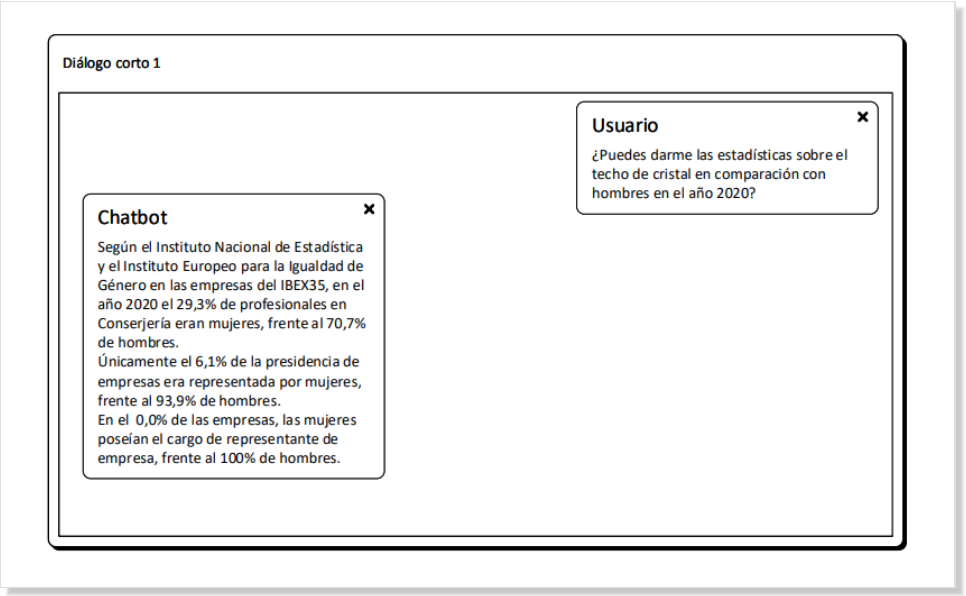

Figura 5.12: Prototipo de diálogo con pregunta específica de un año

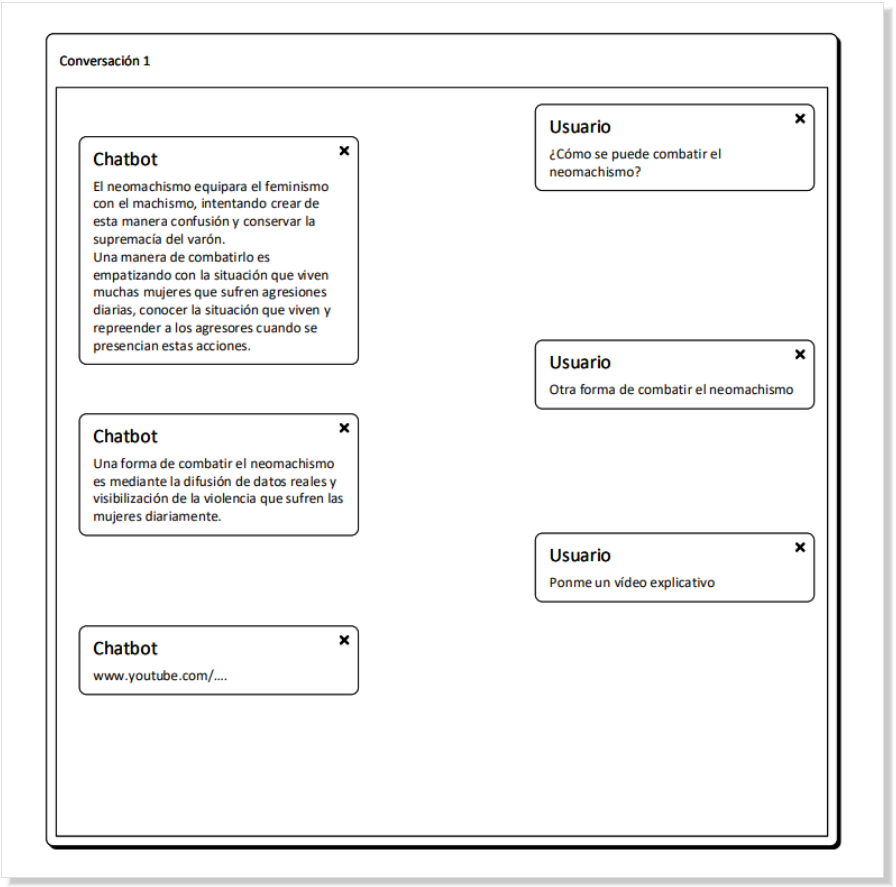

Figura 5.16: Prototipo de diálogo con preguntas y respuestas en cascada

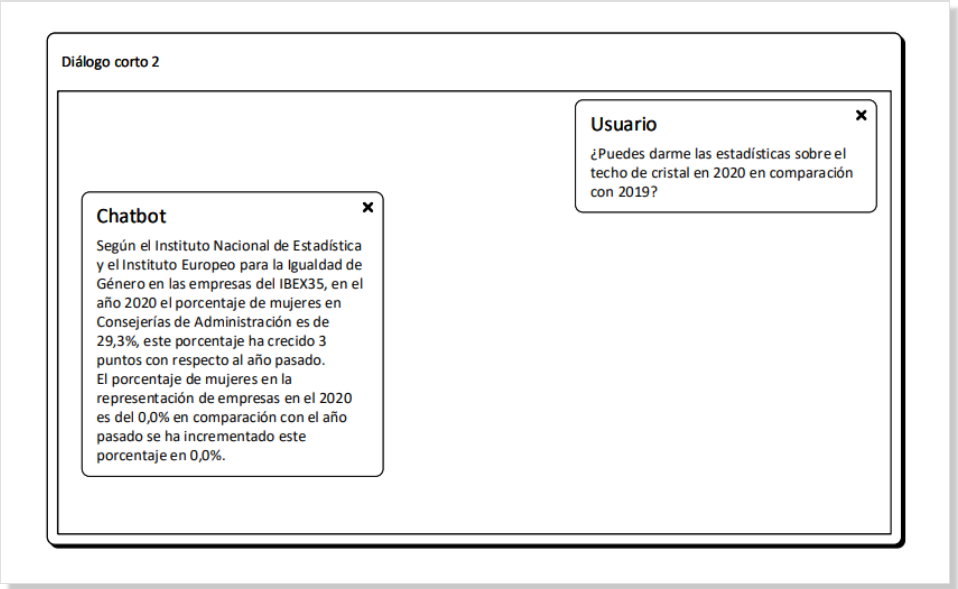

Figura 5.13: Prototipo de diálogo con pregunta específica de dos años

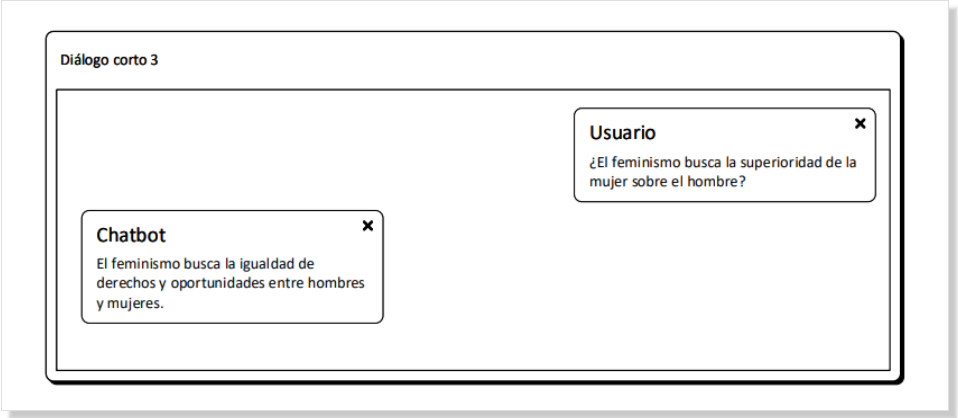

Figura 5.14: Prototipo de diálogo con pregunta de respuesta sencilla

Figura 5.15: Prototipo de diálogo con pregunta de respuesta sencilla

5.8 Privacidad desde el diseño

Como se comentó en la introducción del documento, en el inicio de este proyecto se planteó almacenar algunos datos personales del usuario en función de las conclusiones que se podrían llegar a extraer en los diálogos, de esta manera se podría generar contenido más personalizado a cada persona.

Para poder determinar los riesgos y normativas que se aplicaban al usar estos datos, se hizo una consulta el delegado de protección de datos de la Universidad de Valladolid en la aplicación del RGPD. Tras el análisis del uso de la información personal y riesgos que supondrían su uso (aunque anonimizada), el delegado concluyó que no se deberían almacenar estos datos o extraer conclusiones sobre los mismos.

Siguiendo su consejo profesional, únicamente se almacenará el feedback del usuario y su nombre, siendo este último dato usado de manera temporal, durante la sesión del chat que se mantiene con el bot. El almacenamiento temporal de esta información se realiza para poder dirigirse al usuario por su nombre y poder establecer una cercanía en el diálogo. Cuando el chat se elimine, dicha información se eliminará totalmente.

De forma anónima se almacenarán las opiniones y feedbacks de usuarios de manera persistente, lo que permitirá en un futuro estudiar las retroalimentaciones para poder mejorar la satisfacción del usuario.

Aparte de los datos citados, también se almacenarán datos no vinculantes al mismo que permitirán el correcto funcionamiento de la utilidad de preguntas y respuestas, como lo son la última pregunta que ha realizado el usuario, la respuesta enviada (dependiendo de la intención mapeada), así como el estado de la conversación. Todo esto permitirá poder gestionar correctamente el diálogo y ayudará en la generación de respuestas no monótonas. Todos estos datos se almacenan durante la sesión del chat y cuando el usuario decida eliminar la conversación, también se borrará toda la información asociada.

Con respecto a la seguridad de los datos transmitidos entre cliente y servidor, se utiliza un protocolo TLS (Transport Layer Security) para proteger la comunicación donde se incluye el uso de los algoritmos de encriptación AES (Advanced Encryption Standard) y RSA (Rivest-Shamir-Adleman).

Capítulo 6

Implementación

6.1 Tecnologías utilizadas

Entre las herramientas utilizadas para la escritura del documento, gestión documental, planificación y desarrollo técnico, se utilizó la suite de ofimática de Microsoft junto con las siguientes herramientas:

- Overleaf: Editor de textos en línea para la creación de documentos en LaTeX.
- Microsoft Project 2019: Permite abordar la planificación inicial de un proyecto así como su seguimiento. Se hará uso de esta herramienta para la correcta estimación del tiempo.
- Microsoft OneNote: Gestión de historias de usuario, bugs y seguimiento de sprints.
- Visio Professional 2019: Software de Microsoft que permite la realización de dibujos vectoriales así como ilustraciones visualmente atractivas.
- GitLab: Control de versiones del código fuente.
- Astah: Diseño de diagramas.

A continuación, se describirán las tecnologías más relevantes utilizadas para el desarrollo del bot.

6.1.1 Python

Python [22] es un lenguaje de programación de alto nivel interpretado, dinámico, multiparadigma y multiplataforma; admite parcialmente programación funcional con sintaxis LISP. Debido a su licencia GNU, es de libre distribución y gracias a su popularidad, cuenta con un enorme soporte por parte de su comunidad.

Surgió como un hobby de Guido Van Rossum y su nombre proviene del grupo de humoristas Monty Python, del cual Van Rossum era aficionado.

Características

- De propósito general: Esto significa que Python no está especialmente orientado a una tarea o ámbito específico; sino que se puede utilizar para varios propósitos, como es la creación de aplicaciones de escritorio, scripts, aplicaciones web, etc.
- Interpretado: Python no se compila a código máquina, sino que un intérprete ejecuta las sentencias del programa.
- Tipado dinámico: Esto significa que no es necesario indicar el tipo de la variable a utilizar, sino que este se adaptará al valor que se le asigne.
- Fuertemente tipado: No se permiten violaciones de tipos de datos, por ejemplo sumar cadenas de caracteres y enteros.
- Multiparadigma: Python soporta la programación orientada a objetos (OOP), programación imperativa y programación funcional.
- Multiplataforma: Con python se pueden crear programas que se pueden ejecutar en distintos sistemas operatos, como GNU/Linux, Windows, Mac OS, Solaris, Android, etc.

Figura 6.1: Logotipo de Python

Debido a la versatilidad de este lenguaje así como la facilidad de su implementación, en este proyecto se utilizará Python 3.7.9 para el backend del sistema, esto es, la lógica que seguirán los diálogos así como las transacciones que se deban de realizar a la base de datos, también permitirá implementar la conexión con los distintos canales de comunicación.

Entorno virtual de desarrollo

La instalación de todas las dependencias así como la ejecución, realización de pruebas y puesta en producción se llevarán a cabo un entorno virtual en Python, esto permitirá controlar mejor las dependencias instaladas debido a que estarán encapsuladas en un único espacio y además evitar posibles conflictos con los paquetes ya existentes en la máquina de desarrollo.

Para crear un entorno virtual se utilizará el módulo `venv`, con en el comando a continuación:

```
python -m venv /directorio/donde/crear/entorno/virtual/nombre_carpeta
```

Para activar el directorio ejecutamos el script de activación, desde el directorio donde se encuentra el directorio virtual que se acaba de crear, en Windows el comando es el que sigue:

```
.\.venv_bot\Scripts\Activate.ps1
```

6.1.2 Visual Studio Code

Desarrollado por Microsoft, Visual Studio Code es el editor de código fuente open source utilizado en este proyecto. Es una aplicación de escritorio muy ligera y permite la depuración y control integrado de Git, además cuenta con la posibilidad de incorporar varios complementos que facilitan el desarrollo del software.

Figura 6.2: Logotipo de VS Code

6.1.3 Git

Git [23] es una herramienta gratuita y de código abierto, desarrollada por Linus Torvalds que permite realizar el control de versiones del código fuente de un proyecto de manera distribuida, ya sea local o remota y de forma rápida y sencilla.

Está organizado en repositorios, los cuales almacenan los ficheros que constituyen el proyecto software. El proyecto se pueden dividir en ramas, donde cada rama proviene de otra y constituye una copia exacta de la estructura y ficheros de la rama origen.

Las ramas nos permiten desarrollar funcionalidades independientes, constituyen una línea de desarrollo única, donde las funcionalidades desarrolladas en cada rama no se entremezclan con aquello preexistente en otras ramas. La rama **master** o **main** es la rama principal, generada por defecto al crearse un repositorio Git.

Figura 6.3: Logotipo Git

La gestión de las versiones de manera local y remota permite realizar modificaciones sobre el código fuente de manera local, mientras se mantiene una versión estable del mismo en un repositorio remoto.

Un repositorio local está compuesto por tres árboles, manejados por Git. El primero es el directorio de trabajo donde desarrollamos. El segundo es el Index o Área de Stage, al que se pueden añadir ficheros mediante el comando `git add <fichero>`.

El Stage es una zona intermedia entre el directorio de trabajo y el Head, el tercer repositorio, donde se registran nuestros cambios mediante el comando `git commit -m "Comentario"`. El Head apuntará siempre al último commit realizado.

Finalmente para poder enviar los cambios realizados al repositorio remoto se realiza un `git push origin <rama>`, sobre la rama donde se quiere aplicar los cambios.

Igualmente, para poder fusionar una rama creada con sus debidas modificaciones, con otra rama, se utilizará el comando `git merge <rama>`. Con este comando se intentarán fusionar los códigos fuentes de ambas ramas, sin embargo esto no siempre ocurre a la perfección, se podrían generar conflictos que se deberán de resolver.

GitLab

GitLab es un entorno que permite la administración de proyectos Git. Proporciona al usuario una interfaz sencilla que a la vez provee potencia en su funcionalidad. En este proyecto, se utilizará un repositorio Git almacenado remotamente en GitLab.

Figura 6.4: Logotipo GitLab

6.1.4 Astah

Astah es una herramienta software que permite crear diagramas que potencian la comprehensión y abstracción de la información, pudiendo crear diseños en UML y diagramas de flujo, entre otros. Se utilizará en este caso para la creación de diagramas que representen la arquitectura del sistema.

Figura 6.5: Logotipo Astah

6.1.5 Pandas

Pandas [24] es una herramienta de código abierto disponible para Python, que permite la manipulación y análisis de datos de manera rápida y flexible. Entre sus características, se encuentra el manejo de series temporales y estructuras de más de una dimensión, el acceso a los datos mediante nombres de columnas o filas, realización de operaciones sobre los mismos o la lectura y escritura de ficheros estructurados.

Algunas fortalezas de esta librería se citan a continuación:

- Proporciona estructuras de datos flexibles sobre la que manipular datos y realizar operaciones
- Innumerables métodos y funciones de manipulación de datos
- Fácil integración con otras librerías
- Amplia documentación y soporte

Entre sus puntos negativos cabe destacar:

- Gran consumo de memoria
- Curva de aprendizaje elevada para personas sin conocimientos previos en el análisis de datos
- No es la opción adecuada para manipulación de datos en tiempo real

En este proyecto se utilizará esta librería para el tratamiento de las series de datos estadísticos a manejar.

Figura 6.6: Logotipo Pandas

6.1.6 Seaborn

Seaborn [25] es una librería construida sobre Matplotlib, que permite la visualización de datos estadísticos en Python. Provee una sintaxis más sencilla que Matplotlib, además de varios temas y formas de personalizar la estética de los gráficos de manera rápida. Seaborn está orientado al manejo de marcos de datos en Pandas, por lo que la combinación de ambas librerías se realiza de forma muy natural.

Entre sus fortalezas se puede destacar:

- Generación de gráficos estéticos y de alta calidad
- Integración directa con Pandas
- Soporte de diversos tipos de gráficos

Algunos aspectos negativos a tener en cuenta son los siguientes:

- La capacidad de personalización no es tan amplia como ocurre en Matplotlib
- No es la mejor opción para gráficos muy personalizados o complejos

En este proyecto se utilizará esta herramienta para proveer una visualización de los datos estadísticos, una vez han sido tratados con Pandas.

Figura 6.7: Logotipo Seaborn

6.1.7 Bot Framework Emulator

Es una aplicación de escritorio [26] que pone a disposición Microsoft para permitir depurar y probar el bot desarrollado, consta de una interfaz de chat y herramientas de depuración.

Este emulador simula una aplicación chat en web, por lo que aquí los mensajes, sean de tipo texto plano, enriquecido, con tarjetas de imágenes, vídeos u otros, siempre se visualizarán correctamente, con toda la funcionalidad que presentan; es en el canal destino donde podría haber cambios en la visualización de estos elementos.

Emulator permite inspeccionar en formato JSON, los metadatos de los mensajes intercambiados con el bot, como el tipo de actividad, URL del punto de conexión, canal de comunicación, entre otros; además de permitir comprobar las trazas del servicio LUIS, pudiendo observar toda la información relativa a la predicción de intenciones y entidades.

Para su ejecución simplemente se debe de poner el chatbot en ejecución local, mediante la siguiente instrucción que devolverá por terminal, la dirección URL en la que está ejecutándose el bot, en este caso ha sido:

```
python app.py
======== Running on http://localhost:8000 ========
```

Al abrir Emulator, debemos indicar la URL de conexión de la siguiente manera:

```
http://localhost:8000/api/messages
```

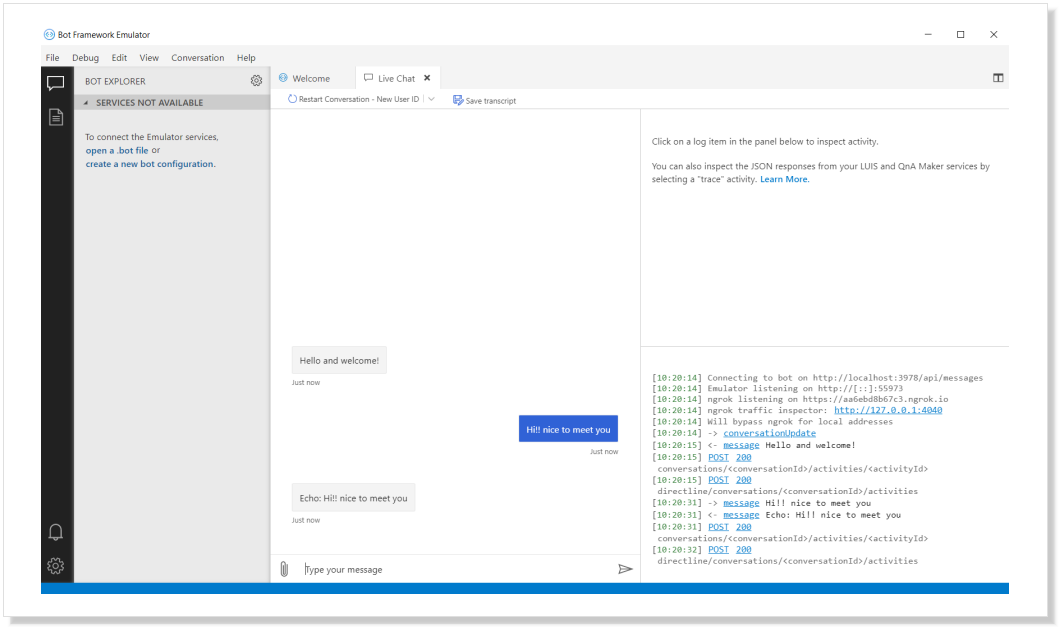

Figura 6.8: Interfaz gráfica de Bot Framework Emulator

6.1.8 Azure Cosmos DB

Cosmos DB [27] provee una base de datos NoSQL serverless distribuida alojada en Azure. Estas bases de datos son no relacionales, es decir, no se estructuran con filas y columnas en tablas, dada su estructura pueden administrar grandes cantidades de datos no estructurados o que cambian con rapidez.

Hay diversos tipos de bases de datos NoSQL pensados en optimizar el almacenamiento en función de la estructura del dato, de esta manera existen bases de datos que almacenan pares de clave/valor, datos estructurados como documentos JSON o como un grafos.

Algunos puntos fuertes de esta herramienta son:

- Posee un modelo de datos flexible, compatible con documentos, de tipo grafo, clave/valor y columnares
- Escalabilidad global y alta disponibilidad

Entre sus debilidades se pueden citar:

- Costos más altos en comparación con otras alternativas de almacenamiento en la nube
- Su configuración puede ser más compleja que otras alternativas

En el proyecto se utilizará una base de datos sencilla que almacenará documentos en JSON. Dado que no se puede almacenar ninguna información relativa al usuario, la única información a almacenar será la puntuación y el feedback que este le dará sobre el funcionamiento del propio bot y la experiencia que ha tenido.

En Azure Cosmos DB, al crear una base de datos nueva, se debe de crear también un contenedor, que es donde se alojarán todos los elementos que se van a insertar, en este caso, el feedback del

usuario. Con la creación de un nuevo contenedor, se debe definir un `partition key` o clave de partición que es una propiedad que se define en función de los datos que se estén manejando, la cual ayudará a Azure Cosmos DB a distribuir los datos eficientemente entre todas las distintas particiones existentes.

Los contenedores tienen esquemas arbitrarios (definición de la estructura del documento) pudiendo esta ser modificada en cada nueva inserción de datos, aun así, los datos se almacenarán en la misma partición siempre que compartan la misma clave de partición.

6.1.9 Azure Bot Service

Azure Bot Service [28] es un servicio de Microsoft que engloba a muchos otros, proporciona un entorno diseñado específicamente para la administración de bots, donde se puede depurar, desplegar y conectar el bot varios a canales, todo en un único entorno.

A continuación se citan algunos de sus puntos fuertes:

- Facilita el desarrollo y despliegue de chatbots y asistentes virtuales en múltiples canales de comunicación
- Escalabilidad y alta disponibilidad, permitiendo gestionar grandes volúmenes de interacciones y usuarios simultáneos

Algunas de sus debilidades son las siguientes:

- Se puede requerir configuración y personalización adicionales en función del canal destino utilizado
- Los costos pueden incrementarse en función de los servicios o volúmenes de tráfico utilizados
- La curva de aprendizaje puede ser elevada para personas sin conocimientos previos en el desarrollo de chatbots

Canales

Un canal permite la conexión entre el bot desarrollado y una plataforma o aplicación destino que el usuario/a final utilizará. Azure permite la conexión mediante canales estándar, como son Facebook Messenger, Skype, Slack o Teams entre otros.

Bot Framework SDK

Es el kit de desarrollo de software de Microsoft utilizado para desarrollar bots. Este SDK proveerá todas las herramientas necesarias para desarrollar, compilar y poner en funcionamiento el bot en local.

Con el uso de este SDK y otros servicios de Azure Bot Service, es posible el desarrollo y conexión del bot con diversas aplicaciones.

Algunas ventajas de su uso son:

- Amplia compatibilidad para el desarrollo en diversos lenguajes

- Flexibilidad en el diseño de conversaciones y lógica del bot
- Se integra con servicios de inteligencia artificial y aprendizaje automático

Algunas debilidades de esta herramienta son:

- Curva de aprendizaje elevada
- Varias características y funcionalidades están en constante evolución, lo que genera cambios en la forma de desarrollar el chatbot

Azure también pone a disposición otros servicios como Azure Cognitive Services, para la creación de bots inteligentes.

Azure Cognitive Services

Son servicios que utilizan API REST y permiten integrar inteligencia cognitiva a los bots desarrollados con tecnología Microsoft. Estos servicios permiten atribuir al bot la capacidad de ver, oír, hablar múltiples idiomas, comprender una solicitud, realizar búsquedas por Internet y tomar decisiones. Para este proyecto, serán relevantes los servicios de idioma.

Cada una de estas capacidades consisten en APIs individuales que incorporan varios servicios a los cuales se puede acceder mediante REST.

En lo que respecta a la capacidad de Idioma:

- Language Understanting Intelligence Service (LUIS): Servicio basado en la nube que aplica inteligencia de aprendizaje automático para extraer información detallada, de una conversación en lenguaje natural. LUIS maneja las predicciones posibles a una solicitud y las devuelve a la aplicación, quién decidirá cómo actuar sobre estos datos.
- Question Answering: Permite crear diálogos estructurados en formato de pregunta-respuesta. Para ello es necesario alimentar la base de conocimiento con diversas preguntas y las posibles respuestas a ellas.

Algunas ventajas derivadas del uso de Azure Cognitive Services son las siguientes:

- Fácil integración mediante el SDK de Bot Framework
- Alta escalabilidad y gran rendimiento
- Cuentan con algoritmos de aprendizaje automático y mejora continua

Algunas de sus desventajas serían:

- Dependencia de la conexión a Internet, si esta es inestable, afectará a la disponibilidad de los servicios
- Los costos pueden aumentar en gran manera en función de la carga de trabajo

6.1.10 Google Compute Engine

Es un servicio de procesamiento y almacenamiento que ofrece Google [29] como "infraestructura como servicio", permite crear máquinas virtuales en Microsoft o Linux gestionadas de manera automática y alojadas en el espacio de Google. Provee diversas opciones para personalizar el almacenamiento, optimización y escalabilidad.

Algunas fortalezas destacables son:

- Gran escalabilidad y rendimiento
- Personalización de la instancia
- Cuenta con un periodo de prueba muy amplio donde se puede hacer uso gratuitamente de servicios que podrían tener costos elevados

Entre las debilidades se pueden citar:

- Curva de aprendizaje alta
- La administración y configuración pueden ser complejas en algunas ocasiones

Se utilizará esta herramienta para poner en funcionamiento el chatbot de manera ininterrumpida, lo cual es necesario para que el bot pueda controlar adecuadamente cada estado en el que se encuentra la conversación y poder procesar cada entrada de datos en función del estado anterior en el que estaba.

6.2 Estructura del código fuente

El proyecto software está estructurado en diversas carpetas, se explicará en esta sección la organización y jerarquía entre las mismas.

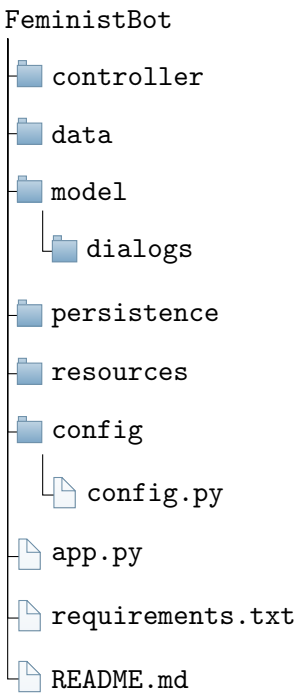

Figura 6.9: Jerarquía de carpetas del proyecto

El proyecto se ha estructurado en carpetas para recalcar la división en capas utilizando MVC. Además de ello, siguiendo el propio estilo de implementación de chatbots de Microsoft, se distingue una carpeta de bot o diálogos, donde se encontrará toda la lógica para la ejecución de diálogos.

En el directorio controller tendremos el fichero dialog_controller.py que consta de la clase `DialogController`, que actuará de intermediario entre el modelo y controlador de vista y será quien se encargue de la lógica de ejecución de los diálogos, así como gestionar toda la información sobre ellos, y el correcto inicio y finalización de los mismos.

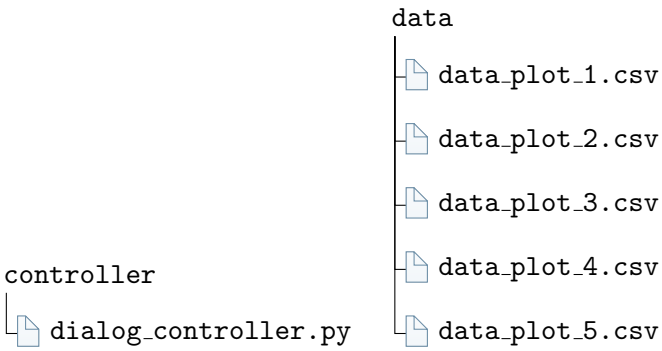

Figura 6.10: Jerarquía de los directorios controller y data

Dentro del directorio dialogs tendremos el código fuente de todos los diálogos que se manejarán en una conversación, donde en cada fichero constará una única clase que permitirá instanciar cada diálogo e iniciar así un flujo concreto de turnos en una conversación.

En el directorio data se almacenarán los datos extraídos del INE en ficheros csv siempre que sean necesarios para el diseño de nuevas gráficas. Estos ficheros se reescribirán con información

actualizada cada vez que se soliciten determinados datos en un amplio intervalo de tiempo.

En el directorio model tendremos aquellos ficheros que permitirán la creación de clases que representen un concepto relevante en la aplicación. En este caso, la entidad conversación, feedback, búsqueda en el INE, intención del usuario, respuesta del usuario y usuario en sí.

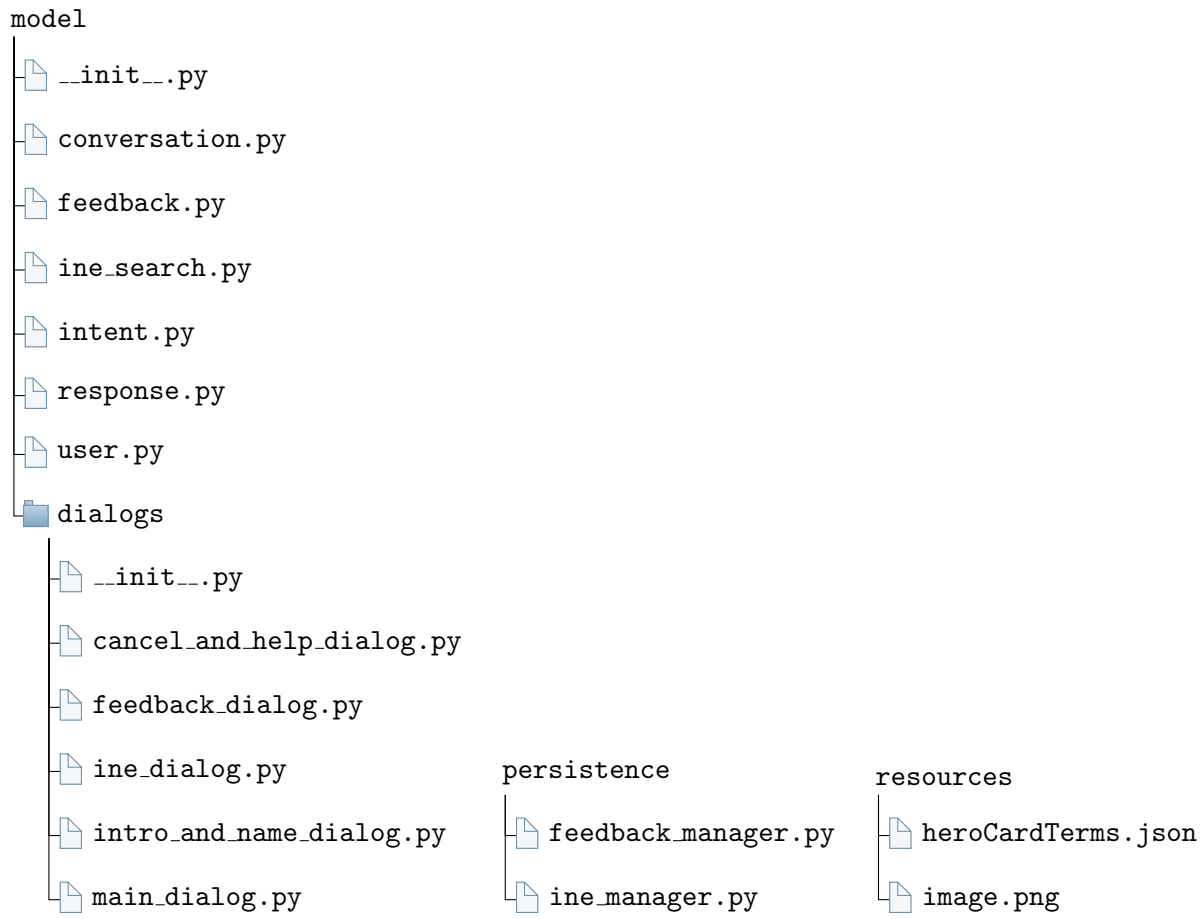

Figura 6.11: Jerarquía de los directorios model, persistence y resources

A continuación se muestra la definición de algunos de estos ficheros.

```
1  import uuid as uid
2  from datetime import date
3
4  class Feedback:
5      """
6      Constructor de la clase
7      @param self: el propio objeto Feedback
8      @param id: identificador unico del Feddback
9      @param rating: puntuacion del bot en una escala numerica fija
10     @param feedback: retroalimentacion del usuario sobre el bot
11     @param date: fecha del feedback recibido
12     """
13
14     def __init__(self, rating: int = 0, feedback: str = None):
```

```
        self.id = str(uid.uuid4())
        self.rating = rating
        self.feedback = feedback
        self.date = date.today().strftime('%Y-%m-%d %H:%M:%S')

    def get_id(self):
        return self.id

    def get_rating(self):
        return self.rating

    def get_feedback(self):
        return self.feedback

    def get_date(self):
        return self.date

    def set_rating(self, rating):
        self.rating = rating

    def set_feedback(self, feedback):
        self.feedback = feedback
```

Código Fuente 6.1: Definición de la clase Feedback

En el fichero ine_search.py tendremos la definición de las búsquedas que se pueden realizar con la API del INE además de la definición de la clase INESearch.

```

ine_all_searches = {
    "relationship" : ["conyugue","exconyugue","novio","exnovio",
        "pareja de hecho","ex pareja de hecho","en separacion"],
    "nationality" : ["europa_sin_espana","espana","africa",
        "america","asia","oceania"]
}
ine_code_description = {
    "condenado": {"code" :["VGD16981"], "desc":
        "hombres condenados con sentencia firme por violencia de género",
        "short_desc": "condenados"},
    "victima": {"code" :["VGD25"], "desc": "mujeres víctimas de violencia de gé
    nero",
        "short_desc": "víctimas"},

    # lugar de nacimiento del denunciado
    "europa": {"code" :["VGD19846"], "desc": "hombres europeos denunciados
        por violencia de género", "short_desc": "Europa"},
    "espana": {"code" :["VGD19845"], "desc": "hombres españoles denunciados
        por violencia de género", "short_desc": "España"},

    # infracciones penales imputadas al condenado con sentencia firme según tipo
    "homicidio": {"code" :["VGD4270"], "desc": "infracciones por homicidio
        y sus formas", "short_desc": "Homicidio"},
```

```
    "lesion": {"code" :["VGD4261"], "desc": "infracciones por lesiones",
        "short_desc": "Lesiones"},
    "secuestro": {"code" :["VGD4252"], "desc": "infracciones por secuestro
        y detenciones ilegales", "short_desc": "Secuestro"},

    # relacion con el denunciado
    "conyugue": {"code" :["VGD353"], "desc": "mujeres víctimas de su      cónyugue
    ",
        "short_desc": "Cónyugue"},
    "exconyugue": {"code" :["VGD352"], "desc": "mujeres víctimas de su ex cónyugue
    ",
        "short_desc": "Ex cónyugue"},
}

...
```

Código Fuente 6.2: Muestra de las definiciones de búsquedas sobre el INE

```
class INESearch:
    """
    Constructor de la clase
    @param self: el propio objeto INESearch
    @param search: búsqueda del usuari@
    @param years: Año a aplicar la búsqueda del usuari@
    """

    def __init__(self, search: list = [], years: list = []):
        self.search = search
        self.years = years
        self.listed_values = {}
        self.description = ""
        self.relationship = ""
        self.nationality = ""

    #-------- Getters ------------
    def get_search(self): return self.search

    def get_years(self): return self.years

    def get_relationship(self): return self.relationship

    def get_nationality(self): return self.nationality

    def get_listed_values(self): return self.listed_values

    def get_description(self): return self.description

    #-------- Setters ------------
    def set_search(self, search: list = []): self.search = search

    def set_years(self, years: list = []): self.years = years
```

```
34
35      def set_relationship(self, relationship: str = ""):
36          self.relationship = relationship
37
38      def set_nationality(self, nationality: str = ""):
39          self.nationality = nationality
40
41      def set_listed_values(self, listed_values: dict = {}):
42          self.listed_values = listed_values
43
44      def set_description(self, description: str = ""):
45          self.description = description
```

Código Fuente 6.3: Definición de la clase INESearch

En la carpeta persistence tendremos los ficheros que realizarán acciones sobre una base de datos, en este caso serán la inserción de feedback, la lectura de datos del INE y grabación de esos datos en servidor de aplicación.

En el directorio resources tendremos recursos que se utilizarán durante el chat, estas son imágenes generadas con Seaborn o bien la tarjeta de bienvenida del usuario `heroCardTerms.json`.

La tarjeta que se muestra al usuario en la bienvenida es del estilo Hero Card y al estar escrita en formato json se renderiza correctamente en la mayor parte de dispositivos. Su estructura es como la que sigue:

```
1
2  {
3    "contentType": "application/vnd.microsoft.card.hero",
4    "content": {
5      "type": "RichTextBlock",
6      "title": "Política de Privacidad, Términos y Condiciones de Uso",
7      "subtitle": "Antes de continuar, debes aceptar
8          los Términos y Condiciones de uso",
9      "text": "En virtud de lo establecido en el Reglamento (UE) 2016/679 del
10         Parlamento Europeo y del Consejo, de 27 de abril de 2016...",
11     "images": [
12       {
13         "url": "https://i.ibb.co/WWyYvP5/url_imagen.png"
14       }
15     ],
16     "buttons": [
17       {
18         "type": "openUrl",
19         "title": "Pulse para más información",
20         "value": "https://www.privacypolicies.com/live/url_fichero"
21       },
22       {
23         "type": "imBack",
24         "title": "Acepto",
25         "value": "Acepto"
26       }
27     ]
```

```
28    }
29 }
```

Código Fuente 6.4: Estructura de la tarjeta heroCardTerms.json

6.3 Configuración de Microsoft Azure

Microsoft Azure es la plataforma de computación en la nube que se utilizará para crear, alojar y probar todos los servicios de Microsoft necesarios para la construcción del bot, además de ofrecer diversas opciones de monitoreo, solución de incidencias, control de costes del recurso, escalabilidad, aplicación de seguridad, copias de seguridad, añadir integraciones con otras tecnologías, entre otras posibilidades.

Para poder utilizar Azure es necesario tener una cuenta en Microsoft Azure, además de un plan de facturación o suscripción; siempre que se limite el uso de los recursos, Microsoft no facturará nada por el uso de los mismos.

Contando con los requisitos anteriores, pasamos a crea dos grupos de recursos separados, estos no son más que espacios diferenciados, donde alojar los distintos recursos utilizados. La razón por la que se deben utilizar dos grupos, es debido a que Microsoft obliga a desplegar chatbots escritos en Python, en un grupo de recursos diferente si el mismo utiliza servicios de Microsoft.

- **FeministBot-GroupResource**: Donde se alojarán todos los servicios necesarios que utilizará el bot.
- **Python_GroupResource**: Donde se alojará únicamente el recurso de tipo "Bot de Azure" que permitirá conectar el bot a diversos canales de mensajería.

Las imágenes a continuación ilustran qué recursos son los alojados en cada grupo de recursos.

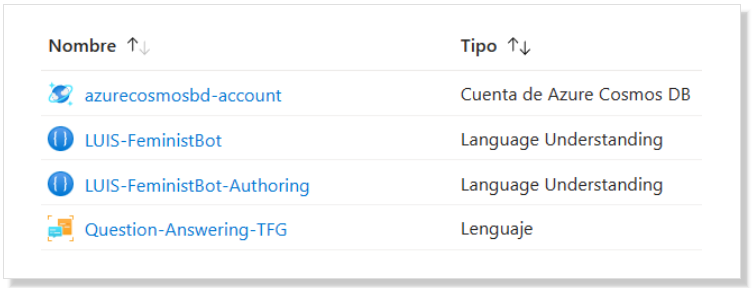

Figura 6.12: Elementos en el grupo de recursos FeministBot-GroupResource

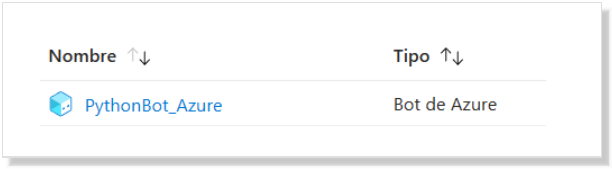

Figura 6.13: Elementos en el grupo de recursos Python_GroupResource

- **azurecosmosbd-account**: Cuenta de Azure Cosmos DB
- **LUIS-FeministBot**: Recurso de predicción de Language Understanding Service
- **LUIS-FeministBot-Authoring**: Recurso de creación de Language Understanding Service
- **Question-Answering-TFG**: Recurso Question Answering, perteneciente a Azure Cognitive Service que permite responder a preguntas que constan en la base de conocimiento
- **PythonBot_Azure**: Permitirá conectar el bot, una vez en funcionamiento, a diversos canales de mensajería de forma casi automática, evitando el tener que desarrollar manualmente la conexión con la aplicación que utilizará el usuario

6.4 LUIS

Este servicio de comprensión del lenguaje [30] permitirá al bot conocer o predecir la intención del usuario cuando este introduzca una entrada de texto.

Es obligatoria la creación de dos recursos LUIS en Azure, debido a que uno de ellos (LUIS-FeministBot-Authoring) se utilizará únicamente para la creación de intenciones, entidades y realizar pruebas solamente en el portal de LUIS, mientras que el otro recurso (LUIS-FeministBot) será el utilizado para realizar las predicciones fuera del portal, ya sea en un entorno de pruebas o real.

6.4.1 Intenciones y entidades

En el portal de LUIS y utilizando el recurso de creación LUIS-FeministBot-Authoring, se definen las intenciones que puede tener el usuario cuando está chateando con el bot, estas se han nombrado de la siguiente manera:

- **ineQuestion_intent**: El usuario quiere obtener datos del Instituto Nacional de Estadística relacionados con la violencia de género en España
- **QnA_intent**: El usuario ha hecho una pregunta genérica sobre la temática tratada
- **identifySexism_intent**: El usuario quiere consejos para identificar actitudes sexistas o machistas en su vida cotidiana
- **feministTip_intent**: El usuario quiere consejos para aplicar conductas anti discriminatorias en base al sexo o género
- **getName_intent**: El usuario ha dicho su nombre al bot
- **cancel_intent**: El usuario quiere cancelar lo que acaba de decir, salir o eliminar el actual flujo de diálogo en el que se encuentra
- **complimnet_intent**: El usuario ha realizado un cumplido al bot
- **critizice_intent**: El usuario ha criticado de manera negativa al bot

- **goodbye_intent**: El usuario se despide del bot
- **help_intent**: El usuario necesita ayuda para conocer el funcionamiento del bot
- **terms_of_use_intent**: El usuario desea conocer los términos de uso del sistema
- **thank_intent**: El usuario agradece al bot por su respuesta
- **None**: Con esta intención se clasificarán todas las entradas del usuario que se salgan fuera de la temática que se está tratando

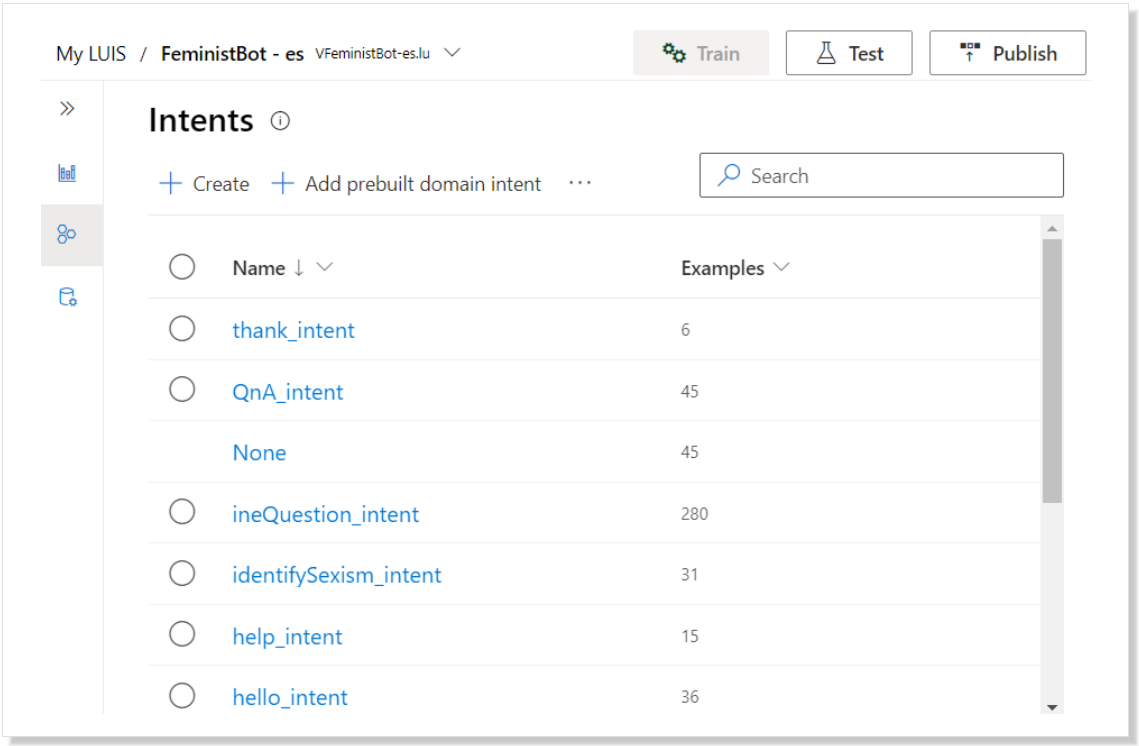

Figura 6.14: Intenciones definidas en LUIS

Una vez definidas las intenciones, es necesario alimentar a la base de cocimiento de LUIS con frases de ejemplo que servirán para que el motor de comprensión del lenguaje pueda aprender qué intención identificar con dichas frases y asociar a dicha intención frases similares, con las que no ha entrenado (entrada del usuario).

En la figura arriba se puede observar algunas intenciones definidas así como en la columna "Examples" el número de ejemplos que se ha provisto para cada intención. El número de ejemplos que se utiliza así como la estructura gramatical de las frases y las palabras utilizadas es de gran importancia para que el motor pueda aprender adecuadamente a identificar las intenciones en las frases del usuario.

Las frases que utiliza el usuario pueden contener información relevante que el bot puede utilizar para generar la respuesta, de esta manera, debemos extraer entidades de las oraciones.

Como se ha mencionado en la introducción del documento, las entidades no son más que palabras o conjunto de palabras que se refieren a algo en específico del mundo real.

En LUIS se pueden hacen uso de las entidades:

1. **Preconstruida**: Entidades ya predefinidas y disponibles para su uso, que permiten reconocer tipos comunes de información, como lo son las fechas, tiempos, medidas, etc.

2. **De tipo lista**: Lista cerrada de palabras con sus sinónimos.

3. **De tipo Regex**: Se extrae la entidad utilizando una expresión Regex.

4. **De tipo Pattern.Any**: Se extrae la entidad utilizando un patrón de una plantilla o frase, este patrón indica dónde la entidad empieza y termina. El patrón debe seguir reglas específicas y su uso es más indicado para frases con una estructura fija.

5. **De aprendizaje automático**: Extrae la entidad utilizando algoritmos de aprendizaje automático, en función del contexto y los ejemplos de los que ha aprendido. A su vez, este tipo de entidad puede contener subentidades cuyos tipos pueden ser de aprendizaje automático o cualquiera de los mencionados arriba.

En el proyecto se han definido las siguientes entidades:

Nombre	Tipo
infractionType	Lista
relationshipType	Lista
nationalityList	Lista
searchList	Lista
agradecer_expresion	Lista
datetimeV2	Preconstruita
name	Aprendiz. Automático
ineQuestion.attribute	Aprendiz. Automático
ineQuestion.nationality	Aprendiz. Automático
ineQuestion.search	Aprendiz. Automático
ineQuestion.withRelationship	Aprendiz. Automático

Tabla 6.1: Entidades definidas y sus tipos

- **ineQuestion.attribute**: Consta de tres subentidades: nationality, relationship e infraction. Cuando el usuario pregunte por la nacionalidad del infractor sin especificar el país, la relación que tenía la víctima con el agresor sin especificar alguna de las posibles o bien el tipo de infracción cometida sin especificar ninguna, se utilizarán estas entidades para extraer la información precisa del usuario.

- **ineQuestion.nationality**: Identifica la nacionalidad del infractor y está compuesta por algún elemento de nationalityList.

- **ineQuestion.search**: Identifica las posibles búsquedas o consultas que el usuario desea realizar sobre el INE y está compuesta por algún elemento de searchList. Dado que en una misma frase se puede realizar más de una consulta de datos, cada subentidad de searchList debe ir con una entidad ineQuestion.search distinta.

- **ineQuestion.withRelationship**: Consta de diversas subentidades y su objetivo es identificar entidades si el usuario solicita información donde la víctima tenía o no una relación específica con el agresor. Sus subentidades son:

  - exRelationship: Identifica las entidades que vengan a significar que la víctima ya no está en una relación con el agresor.
  - inRelationship: Identifica las entidades que vengan a significar que la víctima sigue en una relación con el agresor.
  - relationshipName: Identifica el nombre de la relación entre el agresor y la víctima.
    - conyuge
    - novio
    - pareja de hecho

    Para poder identificar correctamente las relaciones que ya habían finalizado, se ha optado por utilizar la lista relationshipType. Esto se debe a que no se extraían correctamente las entidades utilizando una única lista para describir las relaciones que habían terminado y las que no.

    Al emplear el enfoque citado, evitaríamos problemas como por ejemplo: En la supuesta frase de entrada del usuario, donde suponemos que habrá faltas ortográficas: "Cuántas chicas han sido agredidas por sus ex novios?" Si contáramos solo con entidades de tipo lista, donde se definen previamente las entradas "novios" y "ex novios", esta identificaría únicamente la palabra "novios", con lo cual no se estaría recogiendo adecuadamente la intención del usuario.

- **infractionType**: Entidad que identifica todas las infracciones que puede haber cometido el agresor (abuso sexual, agresión sexual, allanamiento de morada, amenazas, coacción, daños, homicidio, injurias, lesiones u otros delitos).

- **relationshipType**: Entidad que identifica las relaciones ya finalizadas entre la pareja y agresor (en separación, exnovio, excónyugue y ex pareja de hecho)

- **nationalityList**: Identifica las posibles nacionalidades de origen del agresor, clasificando según continente (África, América, Asia, Oceanía, España, resto de Europa).

- **searchList**: Identifica las posibles búsquedas que se pueden realizar al INE (personas absueltas, condenadas, infracciones o víctimas por violencia machista).

- **agradecer_expresion**: Identifica palabras y expresiones que dan a significar que el usuario está agradeciendo al bot.

- **datetimeV2**: Identifica horas, fechas, palabras y conjunto de palabras, que dan a significar que el usuario está hablando de un momento temporal.

- **name**: Identifica el nombre del usuario.

A continuación se ilustra cómo se refleja en el portal de LUIS la configuración de entidades de tipo lista y aprendizaje automático.

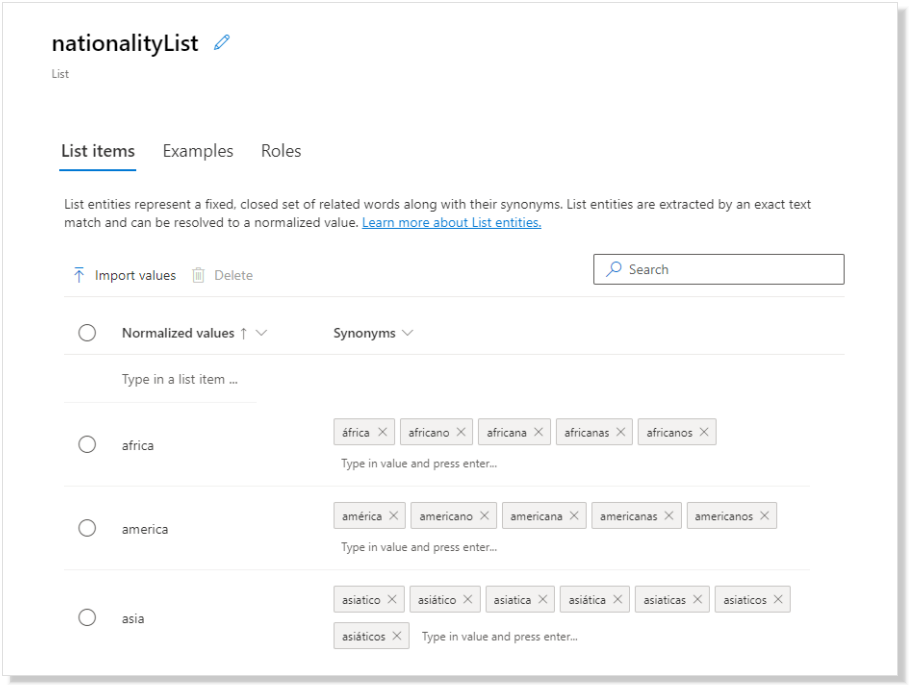

Figura 6.15: Entidad de tipo lista donde constan posibles opciones y sus sinónimos

Como se ha comentado antes, las entidades de tipo aprendizaje automático pueden contener subentidades cuyos tipos pueden variar.

En la imagen a seguir, se ve cómo ineQuestion.withRelationship está formado por tres entidades de aprendizaje automático, donde una de ellas (relationshipName) utiliza la entidad de tipo lista relationshipType y además está compuesta por otras subentidades.

Para cada intención se deben de introducir frases de ejemplo que simulen la entrada del usuario y marcar cuáles son las entidades que se deben identificar en cada caso. Finalizada esta tarea, se procede a entrenar el modelo.

El entrenamiento garantiza que todas las frases escritas manualmente como ejemplos, en cada intención, se identifiquen correctamente, así como todas las entidades marcadas de cada oración. Además de ello, posibilita que LUIS pueda aprender de los ejemplos provistos y predecir las intenciones de otras oraciones distintas así como identificar las entidades en ellas.

Con el entrenamiento del modelo, a cada frase se le asigna automáticamente un porcentaje de confianza sobre la predicción, dicho porcentaje representa el cuán confiable es la asignación de dicha frase a determinada intención.

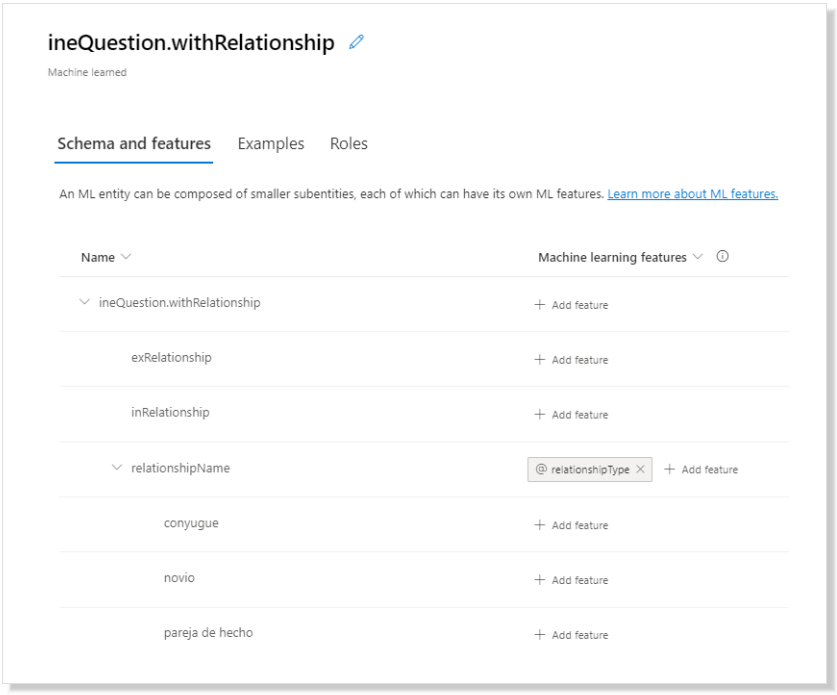

Figura 6.16: Entidad de tipo aprendizaje automático donde constan las posibles opciones y jerarquías de entidades

Figura 6.17: Ejemplos para la intención ineQuestion_intent

Tras entrenar el modelo y probarlo, se procede a publicarlo, esto permitirá que todas las actualizaciones realizadas sobre LUIS pasen a estar disponibles y accesibles como un servicio que se puede invocar desde un punto de acceso.

En la sección de código a continuación se puede observar cómo se configura el punto de acceso y se instancia el servicio para empezar a realizar predicciones sobre la entrada del usuario.

Al reconocedor se le debe de pasar un objeto de tipo `TurnContext`, que viene a contener toda la información sobre el diálogo que se ha procesado durante un turno de la conversación (canal utilizado, entrada de texto del usuario, valor de entrada, identificador del usuario, etc).

El resultado de llamar al método `recognizer.recognize(turn_context)` es un objeto de tipo `RecognizerResult`, en nuestro ejemplo, `luis_result`, del que se puede extraer o bien la intención más probable o bien una lista de las intenciones que se han predicho y sus respectivas probabilidades; además de poder obtener todas las entidades que se han podido extraer de la oración del usuario.

```
# Creación del reconocedor
# Accedemos al servicio usando el punto de acceso de LUIS-FeministBot

def create_recognizer(configuration: DefaultConfig):
    luis_is_configured = (configuration.LUIS_APP_ID and \
    configuration.LUIS_API_KEY and configuration.LUIS_API_HOST_NAME)

    if luis_is_configured:
        luis_application = LuisApplication(
            configuration.LUIS_APP_ID,
            configuration.LUIS_API_KEY,
            configuration.LUIS_API_HOST_NAME,
        )
        luis_option = LuisPredictionOptions(include_all_intents=True,
        include_instance_data=True, spell_check=True)
        return LuisRecognizer(luis_application, luis_option, True)

# Para reconocer la intención de una oración...
# Primero se debe de reconocer y luego extraer la intención predicha
# Pudiendo prefijar un porcentaje mínimo de certeza sobre la predicción realizada

recognizer = create_recognizer(_settings)
luis_result = await recognizer.recognize(turn_context)
intent = recognizer.top_intent(luis_result, min_score = 0.70)
```

Código Fuente 6.5: Creación de un reconocedor LUIS y predicción de intención

En caso de que la intención del usuario sea realizar una consulta de datos estadísticos, las entidades que se extraigan con el reconocedor LUIS permitirán determinar exactamente qué parámetros se utilizarán en la consulta a realizar sobre el INE.

Cada intención se identificará en el código fuente haciendo uso de una clase Enum, como se ve a continuación, se declara el valor de cada elemento con el nombre de la intención definida en el portal de LUIS.

```

from enum import Enum

class Intent(Enum):
    HELLO = "hello_intent"
    GOODBYE = "goodbye_intent"
    CANCEL = "cancel_intent"
    HELP = "help_intent"
    TERMS = "terms_of_use_intent"
```

```
10      NONE = "None"
11      COMPLIMENT = "compliment_intent"
12      CRITICIZE = "criticize_intent"
13      FEMINIST_TIP = "feministTip_intent"
14      IDENTIFY_SEXISM = "identifySexism_intent"
15      GET_USER_NAME = "getName_intent"
16      THANK = "thank_intent"
17      INE_QUESTION = "ineQuestion_intent"
18      INE_INFRACTION = "ineQuestionInfraction_intent"
19      INE_NATIONALITY = "ineQuestionNationality_intent"
20      INE_RELATIONSHIP = "ineQuestionRelationship_intent"
21      QNA = "QnA_intent"
```

Código Fuente 6.6: Intenciones definidas en una enumeración

Cada intención definida posibilitará manejar el flujo de la conversación iniciando y eliminando diálogos específicos.

6.5 Question Answering

Question Answering [31], antes llamado Question and Answer Maker (QnA Maker) también es un servicio cognitivo como LUIS y utiliza a su vez procesamiento del lenguaje natural para poder comprender y generar flujos de diálogos naturales.

Al igual que existía un portal para poder parametrizar y entrenar modelos de LUIS, también existe un portal para definir y probar pares pregunta-respuesta para Question Answering, al que se accede mediante la url https://language.cognitive.azure.com.

Los tipos de proyectos que se pueden crear y su finalidad varía entre las siguientes posibilidades:

- Entendimiento del lenguaje conversacional: Permite aplicar CLU a aplicaciones, dispositivos IoT, etc
- Respuesta a preguntas customizada: Se customizan una serie de preguntas y respuestas extraídas de nuestro corpus de datos
- Clasificación de texto customizada: Se customiza la clasificación de textos utilizando datos propios provistos
- Flujo de orquestación: Conexión entre CLU, Respuesta a preguntas customizada y LUIS
- Reconocimiento de entidades customizada: Se customiza la extracción de entidades

En este portal creamos el proyecto QAnswering-TFG de tipo “Respuesta a preguntas customizada”. A continuación, se seleccionan las fuentes de datos de las que se podrían extraer las respuestas, pudiendo ser URLs, documentos o bien una plantilla vacía sobre la que introducir manualmente los pares pregunta-respuesta.

Alimentamos a la base de conocimiento con diversas preguntas y sus sinónimos y se provee la respuesta a cada una de ellas. En cada respuesta se puede seleccionar añadir acciones sugeridas al

usuario, llamadas “Follow up prompts”, con las que se podría interactuar una vez se haya devuelto la respuesta al usuario.

De igual manera, se pueden añadir imágenes a las respuestas, modificar los metadatos del mensaje, así como marcar una respuesta para que sólo se muestre cuando el diálogo se encuentre en un contexto determinado del diálogo.

En la imagen se observa la pregunta, así como preguntas alternativas que pueden significar lo mismo (alternate questions), la respuesta provista y una acción sugerida “¡Quiero saber más!”.

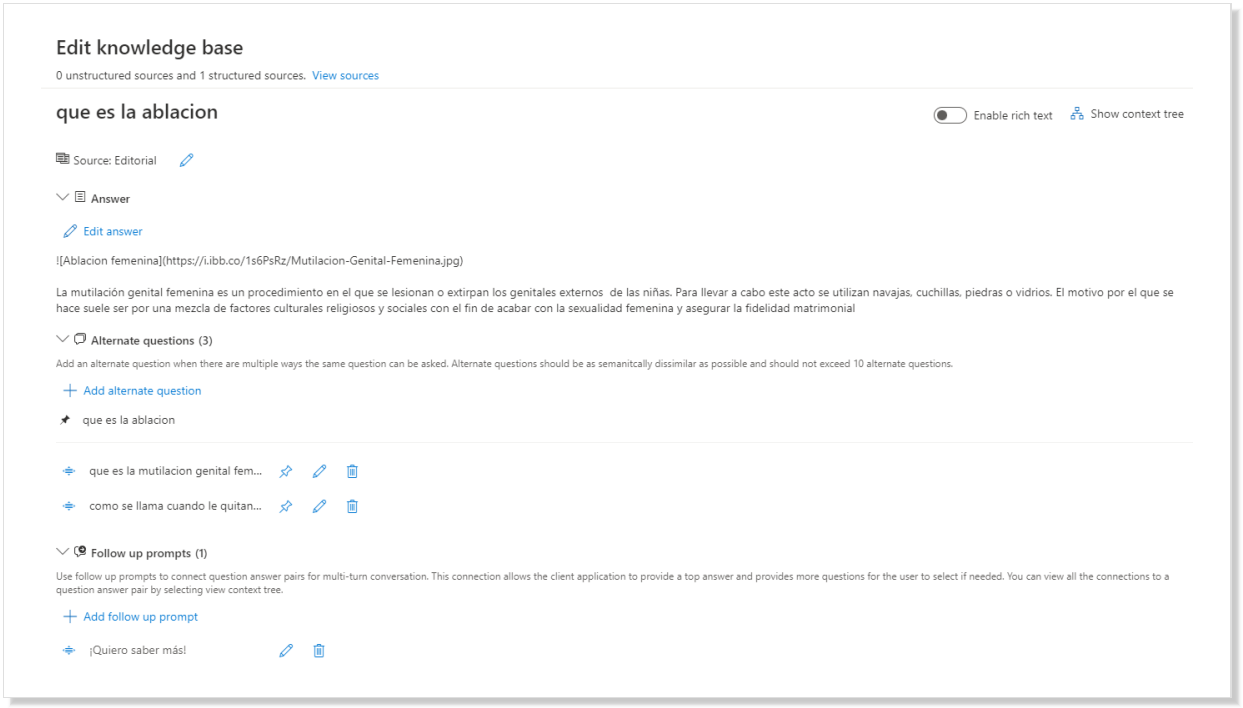

Figura 6.18: Ejemplo de par pregunta-respuesta

Una vez alimentada la base de conocimiento, se realizan pruebas con el chat incorporado para verificar que cada pregunta se responde correctamente en función de los ejemplos provistos. A cada pregunta que se realiza de prueba, se le asigna automáticamente un porcentaje de certeza para asociarla a un par pregunta-respuesta existente en la base de conocimiento. Finalmente, desplegamos los cambios para que estos estén accesibles desde un punto de conexión.

Para su funcionamiento, primeramente se debe crear una instancia de Question Answering proporcionando la configuración necesaria y luego extraer las posibles respuestas pudiendo fijar un umbral de confianza sobre la predicción.

Posteriormente se comprueba que exista un par pregunta-respuesta que supere el umbral fijado y si existen acciones sugeridas de usuario; finalmente se puede proceder a extraer las respuestas más probables y configurar la manera en que se mostrarán al usuario.

Durante la conversación se deberá mantener el contexto del diálogo siempre que en el turno anterior se haya utilizado un par pregunta-respuesta de Question Answering, de esta manera se podrá proveer también en caso necesario, respuestas que son pertenecientes a contextos específicos del diálogo con Question Answering.

6.6 Azure Cosmos DB

Para manejar la base de datos NoSQL que provee Microsoft, se utilizará el portal de Azure accesible desde https://portal.azure.com.

Accediendo al recurso azurecosmosbd-account definido previamente, tenemos la opción de crear nuevas bases de datos, nuevos contenedores, consultar ítems, realizar operaciones CRUD, definir procedimientos almacenados, *triggers*, funciones, o modificar la configuración del contenedor.

Para el proyecto se ha creado una base de datos con nombre FeministBotDB y un contenedor de nombre cosmosdb-tfg.

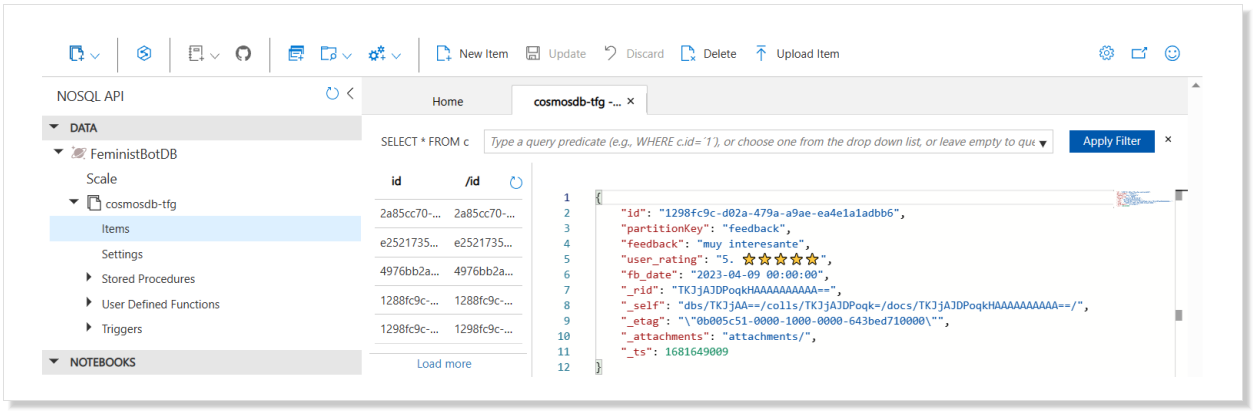

Figura 6.19: Explorador de datos en Azure Cosmos DB

Los documentos que se almacenarán tienen estructura JSON, como se puede observar a continuación.

```
{
    "id": "e2521735-afb7-4782-8b3e-79de105ed327",
    "partitionKey": "feedback",
    "feedback": "está curioso",
    "user_rating": "3. ",
    "fb_date": "2023-04-09 00:00:00",
    "_rid": "TKJjAJDPoqkEAAAAAAAAAA==",
    "_self": "dbs/TKJjAA==/colls/TKJjAJDPoqk=/docs/TKJjAJDPoqkEAAAAAAAAAA==/",
    "_etag": "\"01003cb5-0000-1000-0000-6432e4f50000\"",
    "_attachments": "attachments/",
    "_ts": 1681057013
}
```

Código Fuente 6.7: Estructura de un ítem de base de datos

La siguiente sección de código ilustra de manera simplificada la conexión con Azure Cosmos DB, donde se puede utilizar el método `create_item` para insertar un nuevo elemento en el contenedor.

```
def connect_to_cosmosdb(feedback):

    try:

```

```
        # Se accede mediante el punto de acceso, pasando como argumento
        # los datos propios de la configuración de la base de datos
        # y del contenedor creado

        client = cosmos_client.CosmosClient(settings.DB_HOST,
            {'masterKey': settings.DB_MASTER_KEY},
            user_agent="UsuarioCosmosDB",
            user_agent_overwrite=True)

        # Obtenemos la base de datos y contenedor

        db = client.get_database_client(settings.DB_ID)

        container = db.get_container_client(settings.DB_CONTAINER_ID)#

        # Creación del ítem en el contenedor
        container.create_item(body=feedback)

    except exceptions.CosmosHttpResponseError as e:

        print('\nCaught an error. {0}'.format(e.message))
```

Código Fuente 6.8: Conexión con Azure Cosmos DB e inserción de ítem

6.7 Instituto Nacional de Estadística

Para la obtención de datos fiables con respecto a la temática tratada, contamos con el INE.

El INE cuenta con un servicio API JSON que permite acceder mediante peticiones URL a toda la información disponible en INEbase.

INEbase es el sistema que utiliza el Instituto Nacional de Estadística para almacenar toda la información estadística en Internet y desde la cual realiza publicaciones.

La información estadística presente en INEbase que puede ser consultada a través de una URL, proviene de dos bases de datos distintas, la base de datos relacional Tempus3 y el repositorio de ficheros PC-Axis.

Para realizar la petición de datos la URL debe de seguir la siguiente estructura:

`https://servicios.ine.es/wstempus/js/{idioma}/{función}/{input}[?parámetros]`

Los campos que aparecen entre llaves, son obligatorios, mientras que los que aparecen entre corchetes, son opcionales y cambian en relación a la función considerada.

En el proyecto se utilizará el idioma ES, la función DATOS_SERIE y en el campo `input` se pasarán códigos de consulta sobre series de datos específicas. Estos códigos de consulta son prefijados por el INE y en el código fuente se encuentran ya mapeados para que en función de la intención del usuario, se utilice un código u otro en la consulta. Como `parámetros` se pasará únicamente el número de filas a devolver.

Se muestra a continuación el significado de algunos códigos utilizados:

- **VGD16981**: hombres condenados con sentencia firme por violencia de género

- **VGD4018**: hombres absueltos por delitos de violencia de género
- **VGD19846**: hombres europeos denunciados por violencia de género
- **VGD4252**: infracciones por secuestro y detenciones ilegales
- **VGD19917**: mujeres víctimas de su pareja cuando se encontraban en proceso de separación

La estructura de los datos devueltos es como la que sigue, a continuación se ilustra la respuesta recibida, tras realizar una petición sobre el código VGD16981. Para manejar los datos de la respuesta, se accede a las claves `Data`, `Valor` y `Anyo`.

```
{
"COD":"VGD16981",
"Nombre":"Total Nacional. Violencia de genero. Condenados.
Dato base. Sentencias firmes. Hombres. ",
"FK_Unidad":3,
"FK_Escala":1,
"Data":[
    {"Fecha":1420066800000, "FK_TipoDato":1, "FK_Periodo":28,
    "Anyo":2015, "Valor":24265.0, "Secreto":false}
    ,{"Fecha":1451602800000, "FK_TipoDato":1, "FK_Periodo":28,
    "Anyo":2016, "Valor":25959.0, "Secreto":false}
    ,{"Fecha":1483225200000, "FK_TipoDato":1, "FK_Periodo":28,
    "Anyo":2017, "Valor":27202.0, "Secreto":false}
    ,{"Fecha":1514761200000, "FK_TipoDato":1, "FK_Periodo":28,
    "Anyo":2018, "Valor":27972.0, "Secreto":false}
    ,{"Fecha":1546297200000, "FK_TipoDato":1, "FK_Periodo":28,
    "Anyo":2019, "Valor":30495.0, "Secreto":false}
    ,{"Fecha":1577833200000, "FK_TipoDato":1, "FK_Periodo":28,
    "Anyo":2020, "Valor":25436.0, "Secreto":false}
    ,{"Fecha":1609455600000, "FK_TipoDato":1, "FK_Periodo":28,
    "Anyo":2021, "Valor":33068.0, "Secreto":false}
    ]
}
```

Código Fuente 6.9: Estructura de una respuesta del INE

Los resultados de la consulta a la API del INE podrán o no ser almacenados en un fichero csv, en función de si estos datos se representarán o no de forma gráfica al usuario. De esta manera, serán almacenados siempre que el intervalo de datos supere un límite de años y el usuario requiera de la representación gráfica.

La cabecera de datos que se utiliza para escribir en el fichero csv será `numero` que representará el número de personas, delitos, etc. en cada caso; `anyo` que representará el año al que corresponde cada `numero`, es decir, el año sobre el que se realiza la consulta de datos; y finalmente `dato`, que representa la etiqueta o nombre de cada serie de datos.

```
numero,anyo,dato
81.0 ,2015 ,Homicidio
83.0 ,2016 ,Homicidio
```

```
12631.0 ,2015 ,Lesiones
13220.0 ,2016 ,Lesiones
20.0 ,2015 ,Secuestro
24.0 ,2016 ,Secuestro
6733.0 ,2015 ,Amenazas
7003.0 ,2016 ,Amenazas
1371.0 ,2015 ,Coacción
1453.0 ,2016 ,Coacción
3105.0 ,2015 ,Torturas
5260.0 ,2016 ,Torturas
50.0 ,2015 ,Agresión  sexual
63.0 ,2016 ,Agresión  sexual
19.0 ,2015 ,Abuso  sexual
22.0 ,2016 ,Abuso  sexual
101.0 ,2015 ,Allanamiento  de  morada
115.0 ,2016 ,Allanamiento  de  morada
193.0 ,2015 ,Injurias
575.0 ,2016 ,Injurias
240.0 ,2015 ,Daños
405.0 ,2016 ,Daños
9158.0 ,2011 ,Quebrantamiento  de  condena
7967.0 ,2012 ,Quebrantamiento  de  condena
409.0 ,2015 ,Otros  delitos
407.0 ,2016 ,Otros  delitos
```

Código Fuente 6.10: Muestra de datos de ficheros csv

La estructura tabular de estos ficheros es de tipo ancho, en Inglés "long data format" lo que viene a significar que cada elemento de la columna `dato` se repetirá varias veces en el conjunto de datos.

6.8 Pandas y Seaborn

Pandas y Seaborn funcionan juntos de manera óptima. Para empezar a utilizar ambas herramientas, es necesario instalar sus librerías e importarlas al proyecto.

En el proyecto se generaran gráficas predefinidas y gráficas personalizadas a petición del usuario. A continuación se ilustra un ejemplo de llamada a ambas APIs y la creación de un gráfico circular a partir de una serie de datos extraídas del INE.

Como se observa, se lee con Pandas y se accede a columnas específicas del fichero csv generado, donde se almacenaron con antelación los datos extraídos del INE. Tras la parametrización del gráfico, este se guarda en el servidor para que pueda ser enviado como un adjunto al usuario.

```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# Se lee con Pandas el fichero de datos
data2 = pd.read_csv('data/data_plot_2.csv')

```

```
# Se inicializa la temática de colores del gráfico
sns.set_theme(style="whitegrid")

# Se crea una nueva figura con Matplotlib sobre la que se creará el gráfico
fig2 = plt.figure()

# Se ajusta el tamaño
fig2.subplots_adjust(top=0.76, bottom=0.3)

# Se establece el porcentaje de separación de cada partición del gráfico
# accediendo a los datos mediante Pandas,
# generamos mayor separación en la partición cuyo valor es más alto
explode = [0.01 for _ in range(0, data2["dato"].nunique() -1)]
max_index = data2["numero"].idxmax()
explode.insert(max_index, 0.08)

# Se realiza sumatorio sobre
# la agrupación de los resultados en función del dato que representan
sums = data2.groupby(data2["dato"])["numero"].sum()

# Finalmente se crea el gráfico
chart = plt.pie(sums, labels=sums.index, labeldistance=1.15,
    colors=sns.color_palette("muted"),
    autopct = '%0.0f%%', startangle=80, explode=explode,
    textprops={'fontsize': 11}, pctdistance=0.75)

# Se asigna color de fondo
fig2.set_facecolor('lightgrey')

# Asignación de título y relación de ejes con el escalado de la imagen
plt.title("Relación de la víctima con el denunciado (España)")
plt.axis("equal")

# Se guarda la imagen para enviarla al usuario como un adjunto
plt.savefig("resources/img2.png", bbox_inches='tight')
plt.close(fig2)
```

Código Fuente 6.11: Utilización de Pandas y Seaborn para la creación de un gráfico

Para el ejemplo anterior, el gráfico generado es el que sigue:

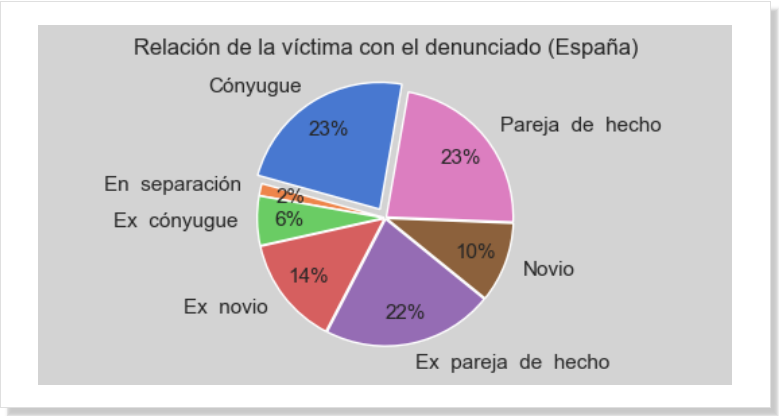

Figura 6.20: Gráfica creada utilizando Seaborn

6.9 Creación y gestión de diálogos

Con el SDK de Bot Framework se pueden hacer uso de diversos tipos de diálogos, es decir, manejar y construir flujos de conversaciones con la librería de diálogos que pone a disposición Microsoft, lo que facilita el control e implementación de un diálogo entre el bot y un ser humano. En función de la finalidad del diálogo, se pueden definir varios tipos, recordando que un diálogo puede consistir en el desarrollo de dos o más turnos en una conversación. A continuación se comentan algunos de ellos:

- **Diálogo componente**: Diálogo de propósito general que funciona como un contenedor que puede encapsular una pila de diálogos. Permite iniciar con el primer diálogo de la pila, este será el responsable de llamar al siguiente y así sucesivamente hasta finalizar con el último diálogo definido en la colección de datos. Cuando el último diálogo finaliza, el diálogo componente se destruye.

- **Diálogo de solicitud**: Se acoplan perfectamente con los diálogos en cascada. Consiste en un diálogo donde el bot solicita información al usuario y este le debe de proveer los datos. La solicitud de información se repetirá un número indeterminado de veces hasta que la información del usuario sea validada como correcta, el diálogo sea cancelado o se haya definido un número determinado de veces para insistir en la petición.

- **Diálogo en cascada**: Estos diálogos definen una serie linear de pasos o acciones sobre las que guiar al usuario. La evolución del diálogo se realiza en "cascada", es decir, en cada turno se solicita información al usuario, a continuación el bot espera por la respuesta y cuando el usuario provee la información solicitada, estos datos se pasarán al siguiente turno, lo que permitirá construir la solicitud de información del siguiente turno y así avanzar en el diálogo.

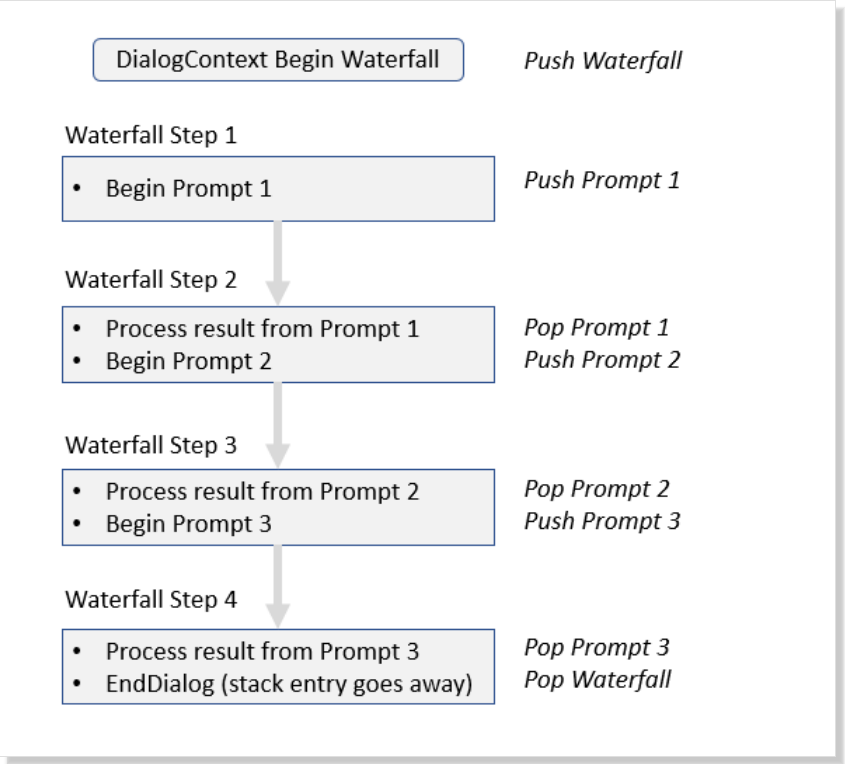

Figura 6.21: Ilustración gráfica de un diálogo en cascada

En este proyecto se hace uso de los diálogos de tipo componente, cascada y de solicitud entre otros, de Bot Framework.

Se define un diálogo inicial de tipo componente llamado `CancelAndHelpDialog`, del que heredarán el resto de diálogos. La finalidad de este diálogo es enviar un texto aclaratorio sobre el uso del bot al usuario cuando este haya pedido ayuda o bien cancelar la ejecución del diálogo actual cuando así lo haya solicitado. De esta manera, haciendo uso de los métodos de esta superclase, cuando un diálogo esté en ejecución, el usuario podrá pedir ayuda y ser contestado sin alterar su flujo.

A continuación se ilustra la definición de estos métodos en una clase simplificada.

```
...

class CancelAndHelpDialog(ComponentDialog):
    def __init__(self, dialog_id: str):
        super(CancelAndHelpDialog, self).__init__(dialog_id)

    # Este método se ejecutará en cada turno para realizar
    # verificaciones sobre la entrada del usuario
    async def on_continue_dialog(self, inner_dialog_context: DialogContext) ->
        DialogTurnResult:
        result = await self.interrupt(inner_dialog_context)
        if result is not None:
            return result

        return await super(CancelAndHelpDialog, self)
```

```
            .on_continue_dialog(inner_dialog_context)

    # Si el usuario no introduce alguna palabra clave de las definidas,
    # en este caso "ayuda" o "cancelar", el flujo será el predeterminado,
    # sino, se mostrará en el siguiente turno un mensaje
    # o bien de ayuda o bien de confirmación de cancelación del diálogo

    async def interrupt(self, inner_dialog_context: DialogContext) ->
        DialogTurnResult:

        if inner_dialog_context.context.activity.type == ActivityTypes.message:
            text = inner_dialog_context.context.activity.text.lower()

            if text in ("ayuda"):
                help_message = MessageFactory.text(
                    "Mensaje de ayuda al usuario", InputHints.expecting_input)
                await inner_dialog_context.context.send_activity(help_message)
                return DialogTurnResult(DialogTurnStatus.Waiting)

            if text in ("cancelar"):
                cancel_message = MessageFactory.text(
                    Diálogo cancelado, InputHints.ignoring_input)
                await inner_dialog_context.context.send_activity(cancel_message)
                return await inner_dialog_context.cancel_all_dialogs()
        return None
```

Código Fuente 6.12: Implementación de un diálogo componente

A continuación se ilustra de manera simplificada la implementación de un diálogo en cascada que hereda del diálogo `CancelAndHelpDialog`, aquí se hará uso de otros diálogos como el de tipo solicitud y el de tipo solicitud de texto.

Este diálogo se llamará `FeedbackDialog` y será el diálogo que se ejecute cuando queramos recibir una retroalimentación del usuario y conocer su opinión sobre su experiencia hasta el momento con el chatbot.

```
...

class FeedbackDialog(CancelAndHelpDialog):

    # En nuestro constructor llamamos al constructor de la super clase
    # Esto ppermitirá hacer uso de funcionalidades ya desarrolladas:
    # Cancelar el diálogo y responder a una solicitud de ayuda
    # sin interrumpir el flujo definido del  diálogo en cascada

    def __init__(self):
        super(FeedbackDialog, self).__init__(FeedbackDialog.__name__)

        # Durante la creación de la clase, se definen
        # los tipos de diálogos que se utilizarán:
```

```
15
16          # Un diálogo en cascada que consta de 2 pasos
17          # (opinion_step y feedback_step)
18
19          self.add_dialog(
20              WaterfallDialog(
21                  WaterfallDialog.__name__,
22                  [
23                      self.opinion_step,
24                      self.feedback_step,
25                  ],
26              )
27          )
28
29          # Un diálogo de tipo solicitud de información en texto
30          self.add_dialog(TextPrompt(TextPrompt.__name__))
31
32          # Un diálogo de tipo selección de opción
33          self.add_dialog(ChoicePrompt(ChoicePrompt.__name__))
34
35          # Se instancia el id del diálogo en ejecución
36          self.initial_dialog_id = WaterfallDialog.__name__
37
38  ...
```

Código Fuente 6.13: Implementación de un diálogo en cascada

Definimos los pasos del diálogo en cascada, estos son métodos que en cada caso devolverán un objeto de tipo `DialogTurnResult`, que encapsulan el resultado del turno en específico que está en ejecución.

En este ejemplo, el primer paso y método a ejecutarse es `opinion_step` y el segundo `feedback_step`. En cada caso sólo se pasará al siguiente paso si se ha obtenido una respuesta válida en el turno actual.

En el método `opinion_step` se utiliza un diálogo de tipo solicitud: Cuando se envíe la pregunta al usuario, el bot se mantendrá esperando en el mismo estado hasta que obtenga una respuesta del usuario. Esta respuesta tiene que ser una de entre todas las posibles, en este ejemplo, se debe de escoger un número dentro del intervalo de enteros [0, 3]; cualquier otra entrada de datos se considerará una entrada inválida, lo que provocará que el bot vuelva a mostrar el mismo mensaje "¿Estás satisfecho con mi funcionamiento?" y esperar por una respuesta.

Esto se repetirá hasta obtener una entrada válida del usuario; en el momento en que el mismo haya introducido un valor permitido, la conversación podrá pasar al siguiente paso, este es `feedback_step`, quién recibirá como argumento el resultado del paso anterior así como toda la información relativa al turno.

El resultado del turno anterior se puede almacenar en el turno siguiente, haciendo uso del contexto del diálogo e instanciando etiquetas; esto se logra mediante la siguiente sintaxis, en este ejemplo utilizando la etiqueta "rating":

```
step_context.values["rating"] = step_context.context.activity.text
```

Una vez este diálogo es finalizado, el diálogo en ejecución pasa a ser el último diálogo que estaba en ejecución, siempre que este no haya finalizado, similar a un sistema LIFO. Si no existen más diálogos que ejecutar en la pila de diálogos, el bot esperará a que el usuario solicite información para poder atenderlo.

En el proyecto, se utilizará la biblioteca de diálogos de Bot Framework en las siguientes ocasiones:

- Cuando se vaya a llevar a cabo el primer diálogo entre el usuario y bot (donde el chatbot dará la bienvenida y explicará su funcionamiento), este diálogo tiene de nombre `IntroAndHelpDialog`
- Para manejar las conversaciones donde se traten datos del INE: `INEDialog`
- Cuando el usuario vaya a dar feedback sobre el funcionamiento del bot: `FeedbackDialog`

Para el resto de diálogos, es decir, cuando el usuario pida información que requiera una consulta a la base de conocimiento de Question Answering, obtener consejos de cómo aplicar conductas no discriminatorias en base al sexo o género o bien consejos sobre cómo identificar los mismos, se hará uso de un diálogo propio llamado `MainDialog` que consistente en el formato pregunta-respuesta para cada turno; las respuestas a los dos últimos puntos se encuentran almacenadas en estructuras de datos en el propio código fuente.

La gestión sobre qué diálogo se está ejecutando, cuál es su estado y qué diálogo ejecutar en cada momento se realiza en el controlador de diálogo `DialogController`, en función de la intención del usuario se inicializarán unos diálogos u otros.

A continuación se ilustra de manera simplificada el control sobre los diálogos en ejecución.

Se deben definir variables de tipo `StatePropertyAccessor` que permitirán almacenar y acceder a propiedades definidas de un diálogo durante el transcurso de la conversación. En este caso se define la propiedad TermsAccepted que indica si el usuario ha aceptado los términos y condiciones de uso de la aplicación, además de la propiedad DialogState, que indica el estado en el que se encuentra el diálogo en ejecución. Esto se logra mediante la sintaxis:

```
dialog_state_property = conversation_state.create_property("DialogState")
```

Haciendo uso de la propiedad definida se instanciará una colección de diálogos donde se almacenará el diálogo en ejecución y aquellos diálogos que se llamen desde el primero. Esta colección de datos persistirá hasta la finalización de la conversación.

Para asignar un valor a una propiedad se utiliza la siguiente sintaxis, pasando como argumento el valor de la propiedad:

```
await accepted_terms.set(turn_context, "Acepto")
```

Mientras que para obtener el valor de la propiedad se utiliza la sintaxis:

```
await accepted_terms.get(turn_context)
```

Si aun no existe valor almacenado para la propiedad TermsAccepted, significa que esta es la primera vez que se ejecuta el chatbot, por lo tanto se debe dirigir la conversación con el diálogo de bienvenida al usuario. Al lanzar este diálogo, podremos saber si el usuario acepta o no los términos de uso, y actualizar la propiedad en cuestión. Para ello, primero es necesario instanciar el diálogo, comprobar que no exista un diálogo con su mismo identificador en ejecución y en caso afirmativo añadirlo a la pila de diálogos y ponerlo en ejecución.

Cuando el chatbot ya se ha iniciado, no se volverá a mostrar el diálogo de bienvenida, por lo tanto se debe dirigir al usuario mediante otros diálogos en función de la intención detectada en el turno actual utilizando LUIS.

Para guiar al usuario a otros diálogos, se debe comprobar siempre si existe un diálogo que se esté ejecutando actualmente, si el estado del mismo ha pasado a estado completado, se debe finalizar el mismo eliminándolo la pila de diálogos. Si no existe diálogo activo, se utiliza LUIS para conocer la intención del usuario en el turno actual. Cuando se haya extraído la intención del usuario, se utilizará un método específico para poder distinguir qué diálogos iniciar y con qué argumentos.

```
...

if results.status == DialogTurnStatus.Complete and dialog:

    # Limpiamos el stack
    await dialog_context.end_active_dialog(dialog.id)
```

Código Fuente 6.14: Gestión de diálogos: Eliminación de diálogos

El método `process_intent_dialog` se encargará de diferenciar exactamente qué diálogo iniciar en función de la intención identificada.

```
...

# En función de la intención se generarán unos diálogos u otros

async def process_intent_dialog(turn_context : TurnContext,
    intent : str, luis_result : LuisResult):

    # Si la intención es despedirse
    if intent == Intent.GOODBYE.value:
        dialog = FeedbackDialog(user_state)

    # Si la intención tiene como propósito solicitar datos del INE
    elif intent in (Intent.INE_QUESTION.value,...):
        dialog = INEDialog(luis_result)

    ...
```

Código Fuente 6.15: Gestión de diálogos: Despacho de diálogos en función de la intención del usuario

6.10 Google Compute Engine

En Google Compute Engine se utilizará una máquina virtual predefinida de propósito general para poder desplegar el código fuente, la cual tendrá como sistema operativo Debian versión 11 Bullseye, cuenta con 2 cores de procesamiento de datos de tipo Intel(R) Xeon(R) y 15 gibibytes de memoria RAM.

Lo primero será crear un proyecto desde la interfaz de Google Cloud, en este caso se ha creado un proyecto llamado FeministBot, tras revisar que los servicios de facturación están en funcionamiento para dicho proyecto, podemos proceder con los siguientes pasos de configuración:

1. Habilitar la API de Compute Engine en la plataforma de Google Cloud

2. Abrir una instancia de Cloud Shell y hacer `pull` de los últimos cambios del código fuente sobre el repositorio Git.

3. Dirigirnos al directorio de trabajo y activar el entorno virtual ejecutando el comando para linux

   `source venv_bot/bin/activate`

4. Dentro del directorio de trabajo instalar todas las dependencias con el comando `pip3 install -r requirements.txt`

5. Ejecutar la aplicación con el comando `python app.py`

Para agilizar el desarrollo, se ha optado por el uso de ngrok, una herramienta que permite exponer una url de un servicio cuya ejecución es local, a Internet.

1. Instalar ngrok utilizando el comando

```
curl -s https://ngrok-agent.s3.amazonaws.com/ngrok.asc |
sudo tee /etc/apt/trusted.gpg.d/ngrok.asc >/dev/null && echo
deb https://ngrok-agent.s3.amazonaws.com buster main |
sudo tee /etc/apt/sources.list.d/ngrok.list
&& sudo apt update && sudo apt install ngrok

```

2. Configurar el código de autenticación de ngrok con el comando

   `ngrok config add-authtoken <token>`

3. Iniciar un túnel en el puerto 8000 con el comando `ngrok http 8000`

Anotamos la url de redirección que se ha generado utilizando ngrok, esta será la dirección pública que utilicemos para hacer peticiones al bot, esta url será la que introduzcamos en el servicio Azure Bot a continuación.

6.11 Azure Bot

Este es un servicio que se ha creado en el portal de Azure, dentro del grupo de recursos Python_GroupResource, su finalidad es facilitar la conexión entre un bot desarrollado utilizando la tecnología de Microsoft cuyo despliegue no se realiza en el ecosistema de Azure, con canales de comunicación como son aplicaciones de chat, una interfaz web y servicios de correo como Outlook.

Desde su panel de configuración se puede establecer el nombre del chatbot así como la imagen que verá el usuario cuando esté chateando con el bot.

En la configuración se debe establecer el punto de conexión de mensajería, que será la dirección url de redirección generada por ngrok más la cadena "/api/messages".

Una vez realizado lo anterior, se selecciona el canal destino entre todas las opciones posibles, en este caso, Microsoft Teams, y pulsando en botón "Open in Teams" ya podemos empezar a chatear con el bot mediante la aplicación de escritorio de MS Teams.

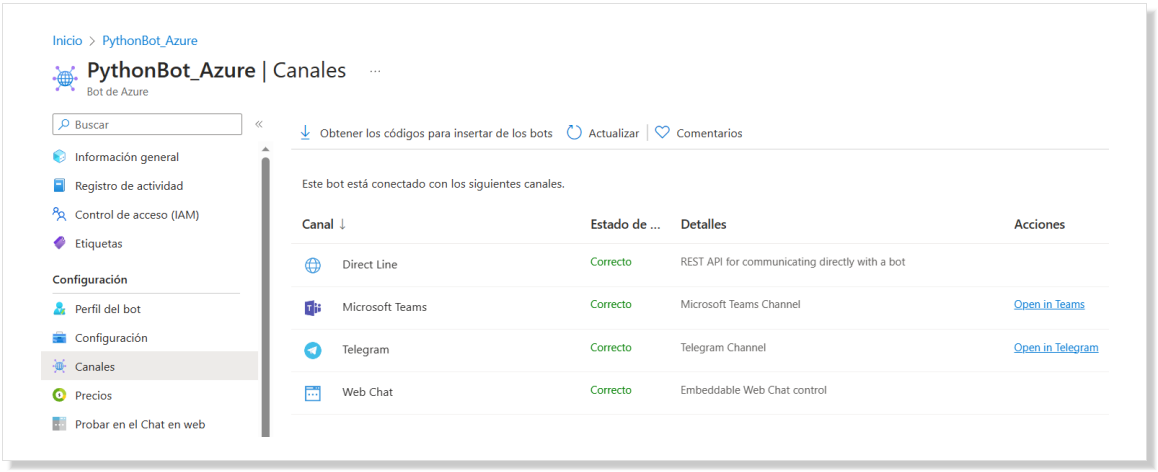

Figura 6.22: Canales utilizados en el despliegue

El canal "Web Chat" se ha utilizado para la realización de pruebas dentro del portal de Azure. En teoría, todos los elementos utilizados con el SDK de Bot Framework, ya sean diálogos específicos, todos los posibles tipos de tarjetas, y cualquier otro tipo de funcionalidad son totalmente funcionales en un chat web, y es en el resto de canales donde esta funcionalidad puede verse afectada, pudiendo no reproducirse una acción que se ejecuta perfectamente en web, no permitiendo la correcta visualización de contenidos o bien impidiendo totalmente la ejecución del chatbot, haciendo obligatoria una adaptación del mismo para que llegue a ser funcional en el canal deseado. En este caso, sí ha sido necesaria una adaptación del chatbot para su funcionamiento en MS Teams.

De forma posterior al desarrollo y depuración en MS Teams, se ha accedido al portal de desarrollo de aplicaciones de MS Teams `https://dev.teams.microsoft.com` para poder publicar el chatbot en la tienda de aplicaciones y así cualquier usuario de la organización UVa o usuario profesional de Teams pueda descargarla; el proceso que realiza Microsoft para la comprobación de aplicaciones nuevas es largo por lo que aun no ha sido posible tener el chatbot en la tienda a disponibilidad del público.

Capítulo 7

Pruebas

En este capítulo se describirán las pruebas realizadas en el sistema, las cuales tienen como objetivo asegurar que el sistema implementa los requisitos establecidos en la etapa de análisis y su funcionamiento es el esperado en cada escenario.

Para la verificación del sistema se realizaron diversas pruebas unitarias de forma correcta, sin embargo las pruebas que se describirán a continuación son en mayor parte pruebas de aceptación guiadas por los casos de uso.

7.1 Casos de prueba

7.1.1 Consultar chatbot

CP-1	**Realizar pregunta en un nuevo chat**
Descripción	Realizar una primera pregunta sobre datos estadísticos al bot
Entrada	En el chat se introduce el nombre de usuario “María”, se realiza la pregunta “cual es el numero de mujeres asesinadas por violencia de genero este año?”
Resultado esperado	El bot reconoce la pregunta, identifica las identidades en la oración y muestra un mensaje con el número de víctimas contabilizadas en dicho año
Correcto	

Tabla 7.1: Definición de casos de prueba: CP-1

CP-2	**Realizar pregunta con detalle - con datos válidos en un chat ya existente**
Descripción	Realizar una pregunta sobre datos estadísticos al bot con datos válidos en un chat existente
Entrada	En el chat se realiza la pregunta "cual es el numero de mujeres asesinadas por violencia de genero el año 2018?"
Resultado esperado	El bot reconoce la pregunta, identifica las identidades en la oración y muestra un mensaje con el número de víctimas contabilizadas en dicho año
Correcto	

Tabla 7.2: Definición de casos de prueba: CP-2

CP-3	**Indicar el nombre del usuario en un nuevo chat**
Descripción	Indicar el nombre del usuario al bot en un nuevo chat
Entrada	En el chat se escribe "me llamo María"
Resultado esperado	El bot reconoce la intención, identifica las identidades en la oración y saluda al usuario por su nombre
Correcto	

Tabla 7.3: Definición de casos de prueba: CP-3

CP-4	**No indicar el nombre del usuario en un nuevo chat**
Descripción	No se indica el nombre del usuario al bot en un nuevo chat
Entrada	En el chat se escribe "quiero saber cuál es mi propósito en la vida"
Resultado esperado	El bot no reconoce la intención ni identifica las identidades en la oración, por lo tanto saluda al usuario sin nombrarle
Correcto	

Tabla 7.4: Definición de casos de prueba: CP-4

CP-5	**Realizar una pregunta fuera del contexto de la aplicación en un chat existente**
Descripción	Realizar una pregunta que el bot no está diseñado para manejar
Entrada	En el chat se escribe "te gusta la pizza con piña?"
Resultado esperado	El bot no reconoce la intención ni identifica las identidades en la oración, por lo tanto envía un mensaje al usuario pidiendo que reformule la pregunta
Correcto	

Tabla 7.5: Definición de casos de prueba: CP-5

CP-6	**Realizar pregunta sin detalle - faltando algún dato en un chat ya existente**
Descripción	Realizar una pregunta sobre datos estadísticos al bot no indicando todos los datos necesarios en un chat existente
Entrada	En el chat se realiza la pregunta “cual es el numero de mujeres agredidas por sus novios?”
Resultado esperado	El bot reconoce la pregunta, identifica las identidades en la oración y determina que falta el dato del año y muestra un mensaje inquiriendo sobre una fecha en la que realizar la búsqueda
Correcto	

Tabla 7.6: Definición de casos de prueba: CP-6

CP-7	**Consultar pregunta rápida**
Descripción	Realizar una pregunta al bot que no sea sobre datos estadísticos dentro del contexto que se puede manejar, en un chat existente
Entrada	En el chat se realiza la pregunta “qué es la ablación?”
Resultado esperado	El bot reconoce la pregunta y envía una imagen con una respuesta corta al usuario
Correcto	

Tabla 7.7: Definición de casos de prueba: CP-7

CP-8	**Cancelar el diálogo cuando el bot está inquiriendo algún dato en un chat ya existente**
Descripción	Cancelar el diálogo cuando el bot pida al usuario más información sobre la pregunta a resolver
Entrada	En el chat se realiza la pregunta “cual es el numero de mujeres agredidas por sus ex maridos?”, posteriormente cuando el bot solicita un año sobre el que realizar la búsqueda, se escribe en el chat “cancelar”
Resultado esperado	El bot reconoce la intención final de cancelar el diálogo, y muestra un mensaje en el chat indicando que ha entendido la intención del usuario
Correcto	

Tabla 7.8: Definición de casos de prueba: CP-8

7.1.2 Iniciar conversación

CP-9	**Saludar al bot en un chat ya existente**
Descripción	Se saluda al bot en un chat ya existente
Entrada	En un chat ya existente se escribe "Hola!!"
Resultado esperado	El bot reconoce la intención y muestra un mensaje en el chat de saludo con una expresión de apertura de diálogo e inquiere sobre una pregunta a realizar por el usuario
Correcto	

Tabla 7.9: Definición de casos de prueba: CP-9

CP-10	**Saludar al bot en un chat nuevo sin aceptar los términos de uso**
Descripción	Se saluda al bot en un chat nuevo sin antes aceptar los términos de uso
Entrada	En un nuevo chat, se escribe "Buenos días"
Resultado esperado	El bot no responde, dado que no se han aceptado los términos y condiciones de uso
Correcto	

Tabla 7.10: Definición de casos de prueba: CP-10

7.1.3 Finalizar conversación

CP-11	**Despedirse del bot en un chat ya existente**
Descripción	Se despide al bot en un chat ya existente
Entrada	En un chat ya existente se escribe "Bye bye"
Resultado esperado	El bot reconoce la intención de despedirse del usuario y muestra un mensaje en el chat para despedirse del él/ella
Correcto	

Tabla 7.11: Definición de casos de prueba: CP-11

7.1.4 Consultar términos y condiciones

CP-12	**Se pide visualizar los términos y condiciones en un chat ya existente**
Descripción	Se solicita al bot los términos y condiciones de uso en un chat ya existente
Entrada	En un chat ya existente se escribe "terminos de uso"
Resultado esperado	El bot muestra un mensaje en el chat con un resumen del términos y condiciones que se aplican sobre los datos del usuario haciendo uso del bot, así como un enlace para consultar más a fondo los términos y condiciones que se aplican
Correcto	

Tabla 7.12: Definición de casos de prueba: CP-12

7.1.5 Consultar manual de uso

CP-13	**Se pide el manual de uso en un chat ya existente**
Descripción	Se solicita al bot mostrar el manual de uso de la aplicación en un chat ya existente
Entrada	En un chat ya existente se escribe “ayuda”
Resultado esperado	El bot reconoce la intención y muestra un mensaje en el chat con un resumen de las funcionalidades que proporciona así como ejemplos de uso
Correcto	

Tabla 7.13: Definición de casos de prueba: CP-13

CP-14	**Se pide el manual de uso cuando el bot está solicitando alguna información**
Descripción	Se solicita al bot mostrar el manual de uso de la aplicación cuando este ha solicitado al usuario cierta información para completar una búsqueda
Entrada	Cuando el bot ha solicitado información al usuario, se escribe en el chat “manual de uso”
Resultado esperado	El bot reconoce la intención y muestra un mensaje en el chat con un resumen de las funcionalidades que proporciona así como ejemplos de uso, posteriormente el turno retorna al usuario para que este introduzca la información solicitada por el bot
Correcto	

Tabla 7.14: Definición de casos de prueba: CP-14

7.1.6 Envío de opinión personal/feedback

CP-15	**Enviar feedback sobre la experiencia con el bot**
Descripción	Al finalizar la conversación, el usuario graba su opinión y experiencia con el chatbot
Entrada	En un chat ya existente se escribe “adiós”, a continuación el bot retiene al usuario para que puntúe su experiencia
Resultado esperado	El bot reconoce la intención de despedirse del usuario y le pide que aguarde un momento, a continuación muestra una serie de preguntas en el chat para pedir que el usuario valore su experiencia. Tras las respuestas recibidas por el usuario, el bot graba la valoración recibida y agradece al usuario.
Correcto	

Tabla 7.15: Definición de casos de prueba: CP-15

CP-16	Cancelar el envío de feedback sobre la experiencia con el bot
Descripción	Al finalizar la conversación, el usuario no graba su opinión y experiencia con el chatbot
Entrada	En un chat ya existente se escribe "adiós", a continuación el bot retiene al usuario para que puntúe su experiencia
Resultado esperado	El bot reconoce la intención de despedirse del usuario y le pide que aguarde un momento, a continuación muestra una serie de preguntas en el chat para pedir que el usuario valore su experiencia con el bot. El usuario escribe "callate" y el bot reconoce la intención de finalizar el diálogo actual de toma de feedback, por lo tanto muestra un mensaje en pantalla indicando que ha entendido la intención y no graba ninguna retroalimentación.
Correcto	

Tabla 7.16: Definición de casos de prueba: CP-16

Tras realizar varias pruebas se han detectado errores relacionados con el propio funcionamiento de Bot Framework. Los errores detectados fueron:

- De manera aleatoria se producía un error al deserializar la respuesta del usuario a JSON. Este error quedó controlado, generando un mensaje de respuesta genérico al usuario avisando del error, aunque no se pudo averiguar su origen y subsanarlo debido a que tiene lugar de manera repentina y aleatoria.
- Al volver a llamar a un mismo diálogo, los atributos del diálogo (propios del SDK de Bot Framework) pueden quedar obsoletos y reflejar información desactualizada. Para solucionarlo, en lugar de utilizar los atributos del diálogo se han creado y empleado atributos de clase.

7.2 Pruebas con usuario final

A continuación se describe brevemente las pruebas que realizaron usuarios finales y sus comentarios con respecto al funcionamiento del sistema, además de su valoración sobre los atributos de usabilidad utilizados en el diseño sobre una escala de 10 puntos.

Los acrónimos utilizados vienen a significar:

- F.A.: Facilidad de aprendizaje
- F.R.: Facilidad de recuerdo
- T.E.: Tratamiento de errores

Todas las personas que utilizaron el sistema son actualmente estudiantes en la Universidad de Valladolid, cursando carreras técnicas o económicas.

Usuario/a	Sexo	Edad	F.A.	F.R.	T.E.	Comentarios	Pruebas
María	F	19	9	10	10	Me ha gustado que use datos de España	Accede al chat, acepta los términos de uso, realiza pregunta sobre datos en España
Marta	F	22	10	10	6	Es interesante aunque me gustaría que pudiese darme otras respuestas cuando no sabe qué responder. Si se ampliara su conocimiento sería perfecto.	Accede al chat, acepta los términos de uso, realiza pregunta sobre datos en España y posteriormente, después de recibir una respuesta realiza varias preguntas fuera del contexto de la aplicación
Tamara	F	27	9	10	8	Me parece eficiente y didáctico	Accede al chat, acepta los términos de uso, realiza varias preguntas de respuesta rápida
José María	M	26	9	10	7	Sirve para consultar preguntas, aunque no sé si realmente la gente lo utilizaría	Accede al chat, acepta los términos de uso, realiza varias preguntas de respuesta rápida
Jin	F	26	10	10	8	Es muy interesante, creo que ayudaría a que las personas se enteraran más del tema junto con otras fuentes de información	Accede al chat, acepta los términos de uso, realiza varias preguntas de respuesta rápida y preguntas sobre datos en España
Víctor	M	23	9	9	7	Se ve que está trabajado, aunque yo añadiría más capacidades para que hable sobre más cosas	Accede al chat, acepta los términos de uso, realiza varias preguntas de respuesta rápida y preguntas sobre datos en España

Tabla 7.17: Pruebas con usuario final

Capítulo 8

Conclusiones y trabajo futuro

8.1 Conslusiones

Tras la conclusión del proyecto se puede decir que se ha podido implementar de manera satisfactoria los principales objetivos definidos para el sistema, estos eran desarrollar un chatbot con herramientas de inteligencia artificial que permitiera llevar cabo diálogos que trataran sobre la discriminación de género y tuvieran como propósito informar a los usuarios sobre esta problemática social.

Un poco más en detalle:

- Se han mantenido y actualizado bases de conocimiento
- Se han entrenado modelos de un servicio de inteligencia artificial
- Se ha podido predecir con una confianza aceptable las intenciones del usuario
- Se han desarrollado flujos de varios diálogos utilizando las herramientas de Microsoft propuestas al inicio
- Se ha desarrollado la funcionalidad de chat con el bot sobre la temática propuesta mediante Microsoft Teams
- Se ha permitido al usuario registrar su valoración y experiencia

8.2 Trabajo futuro

El sistema a día de hoy es usable y cumple con los requisitos fijados para el sistema, sin embargo existen varias vías de mejoras que se podrían abordar para ofrecer un servicio de mayor calidad al usuario, entre ellas:

- Mejoras en la predicción de intenciones del usuario
- Utilizar un lenguaje más amigable y respuestas menos técnicas
- Desplegar el chatbot en otras aplicaciones

- Permitir utilizar la voz como canal de comunicación
- Permitir mantener conversaciones sobre más aspectos de la temática tratada
- Permitir mantener conversaciones cuya finalidad no sea obtener una respuesta sino debatir sobre algún aspecto de la temática tratada

Bibliografía

[1] Machine Learning Mastery. (s.f.). Natural Language Processing (NLP) in Python: A Complete Guide. Recuperado el 20 de septiembre de 2021, de https://machinelearningmastery.com/natural-language-processing

[2] Grupo de Inteligencia Artificial y Reconocimiento de Formas, Universidad Autónoma de Madrid. (s.f.). ¿Qué es el Procesamiento del Lenguaje Natural? Recuperado el 15 de enero de 2022, de https://www.iic.uam.es/inteligencia/que-es-procesamiento-del-lenguaje-natural

[3] IMF Business School. (2020, junio 15). Procesamiento del Lenguaje Natural: modelos y usos prácticos. Recuperado el 5 de diciembre de 2021, de https://blogs.imf-formacion.com/blog/tecnologia/procesamiento-lenguaje-natural-modelos-usos-practicos-202006

[4] Martín Mateos, F. J., Ruiz Reina, J. L. (2016). Dpto. Ciencias de la Computación e Inteligencia Artificial Universidad de Sevilla. Procesamiento de lenguaje natural. Recuperado el 10 de noviembre de 2021, de https://www.cs.us.es/cursos/aia-2016/temas/tema-04.pdf

[5] Concepto. (s.f.) Diálogo. Recuperado el 25 de julio de 2021, de https://concepto.de/dialogo

[6] Wikipedia. (s.f.). Prueba de Turing. Recuperado el 2 de febrero de 2022, de https://es.wikipedia.org/wiki/Prueba_de_Turing

[7] Kaspersky Latam. (s.f.). ¿Qué son los bots? Recuperado el 10 de agosto de 2021, de https://latam.kaspersky.com/resource-center/definitions/what-are-bots

[8] Wikipedia. (s.f.). Bot conversacional. Recuperado el 20 de enero de 2022, de https://es.wikipedia.org/wiki/Bot_conversacional

[9] Microsoft. (s.f.). Bot Builder basics. Recuperado el 5 de noviembre de 2021, de https://learn.microsoft.com/en-us/azure/bot-service/bot-builder-basics?view=azure-bot-service-4.0

[10] Oracle. (s.f.). ¿Qué es un chatbot? Recuperado el 15 de septiembre de 2021, de https://www.oracle.com/es/chatbots/what-is-a-chatbot

[11] MindTitan. (s.f.). Types of Chatbots. Recuperado el 30 de octubre de 2021, de https://mindtitan.com/resources/guides/chatbot/types-of-chatbots

[12] SAS (s.f.). Machine Learning. Recuperado el 12 de diciembre de 2021, de https://www.sas.com/es_es/insights/analytics/machine-learning.html

[13] Harms, J. G., Kucherbaev, P., Bozzon, A., Houben, G. J. (2018). Approaches for Dialog management in conversational agents. IEEE Internet Computing, 22(6), 21-29. https://doi.org/10.1109/MIC.2018.2881519

[14] Greyling, C. (2020). Dialog Management Considerations for Chatbots. Recuperado el 15 de julio de 2021, de https://cobusgreyling.medium.com/dialog-management-considerations-for-

chatbots-6ed4dca65a80

[15] Galitsky, B. (2019). Developing Enterprise Chatbots: Learning Linguistic Structures. Springer. https://doi.org/10.1007/978-3-030-04299-8

[16] Instituto Nacional de Estadística. (s.f.). Población que usa Internet. Recuperado el 30 de enero de 2023, de https://www.ine.es/ss/Satellite?L=es_ES&c=INESeccion_C&cid=12599255-28782&p=1254735110672&pagename=ProductosYServicios%2FPYSLayout

[17] Scrum.org. (s.f.). What is Scrum? Recuperado el 10 de octubre de 2021, de https://www.scrum.org/learning-series/what-is-scrum

[18] PMI. (2017). A Guide to the Project Management Body of Knowledge (PMBOK Guide) (6th ed.). Recuperado el 25 de octubre de 2021, de http://faspa.ir/wp-content/uploads/2017/09/PMBOK6-2017.pdf

[19] Wikipedia. (s.f.). FURPS. Recuperado el 15 de febrero de 2022, de https://en.wikipedia.org/wiki/FURPS

[20] Rodrigo, G. R. (s.f.). Modelo-Vista-Controlador (MVC). Recuperado el 3 de mayo de 2022, de http://rodrigogr.com/blog/modelo-vista-controlador

[21] Wikipedia. (s.f.). GRASP. Recuperado el 18 de febrero de 2023, de https://es.wikipedia.org/wiki/GRASP

[22] Wikipedia. (s.f.). Python. Recuperado el 27 de abril de 2022, de https://es.wikipedia.org/wiki/Python

[23] Git. (s.f.). Git. Recuperado el 10 de abril de 2022, de https://git-scm.com

[24] Pandas. (s.f.). About Pandas. Recuperado el 5 de mayo de 2022, de https://pandas.pydata.org/about

[25] Seaborn. (s.f.). Seaborn: statistical data visualization. Recuperado el 21 de mayo de 2022, de https://seaborn.pydata.org

[26] Microsoft. (s.f.). Test and debug with the Emulator. Recuperado el 8 de mayo de 2022, de https://learn.microsoft.com/en-us/azure/bot-service/bot-service-debug-emulator?view=azure-bot-service-4.0&tabs=python

[27] Microsoft. (s.f.). Welcome to Azure Cosmos DB. Recuperado el 15 de febrero de 2023, de https://learn.microsoft.com/en-us/azure/cosmos-db/introduction

[28] Microsoft. (s.f.). Azure Bot Service. Recuperado el 28 de marzo de 2023, de https://learn.microsoft.com/es-es/azure/bot-service/?view=azure-bot-service-4.0

[29] Google Cloud. (s.f.). Compute Engine. Recuperado el 20 de mayo de 2023, de https://cloud.google.com/compute

[30] Microsoft. (s.f.). What is Language Understanding (LUIS)?. Recuperado el 12 de mayo de 2022, de https://learn.microsoft.com/es-es/azure/cognitive-services/luis/what-is-luis

[31] Microsoft. (s.f.). What is question answering?. Recuperado el 20 de mayo de 2022, de https://learn.microsoft.com/es-es/azure/cognitive-services/language-service/question-answering/overview

Anexo I: Manual de uso

A continuación se ilustrarán con imágenes las conversaciones mantenidas con el chatbot en el canal destino Microsoft Teams que permiten ilustrar cómo realizar peticiones al bot y cómo es el flujo de los diálogos. Este manual está guiado por los casos de uso definidos en el sistema.

Inicio de la conversación

El usuario debe aceptar los términos y condiciones de uso para empezar a chatear, a continuación el bot pregunta sobre el nombre del usuario para referirse a él o ella.

Figura 8.1: Inicio de la conversión

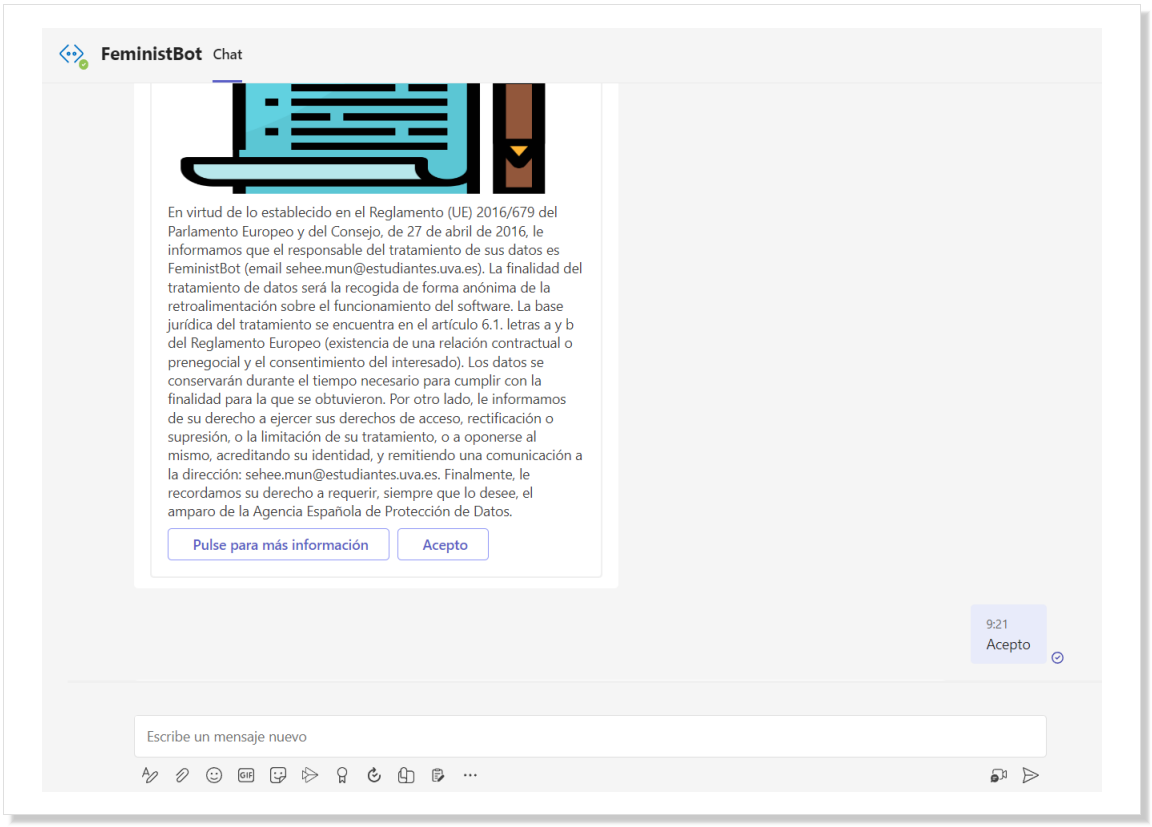

Figura 8.2: Consentimiento del usuario sobre los términos de uso

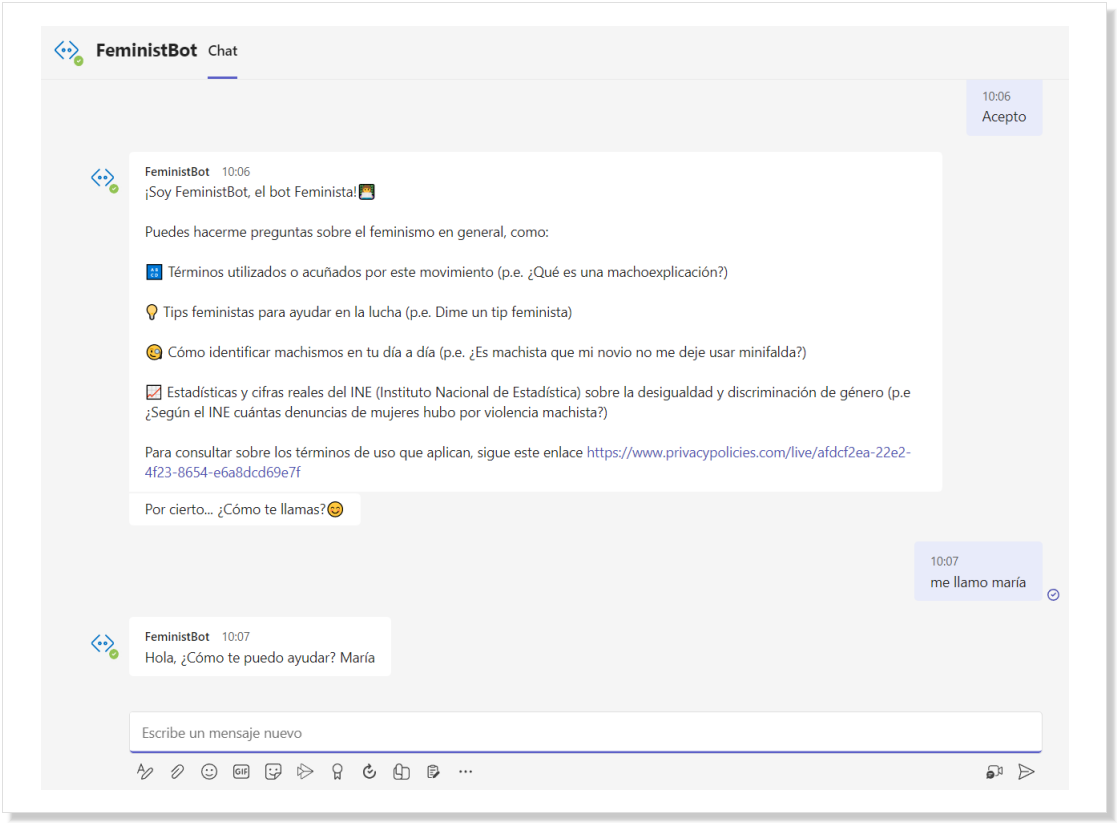

Figura 8.3: Descripción del funcionamiento del chatbot y se averigua el nombre del usuario

Consultas sobre el INE

El usuario puede hacer preguntas genéricas (sin especificar un intervalo temporal) o específicas (concretando un intervalo temporal). En el caso de que el chatbot envíe un gráfico al usuario, este tiene la opción de clicar sobre la imagen para hacerla más grande o descargarla a su equipo local.

Figura 8.4: Consulta genérica sobre nacionalidad

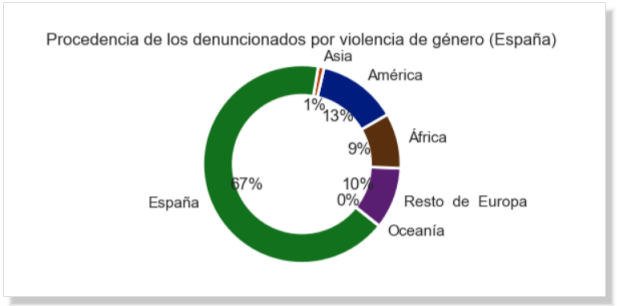

Figura 8.5: Gráfico generado sobre nacionalidad

A continuación, se observa una consulta específica, donde el chatbot inquiere sobre el año en que realizar la consulta al INE y acto seguido una pregunta genérica donde el usuario pregunta por el tipo de delito, en lugar del número de delitos.

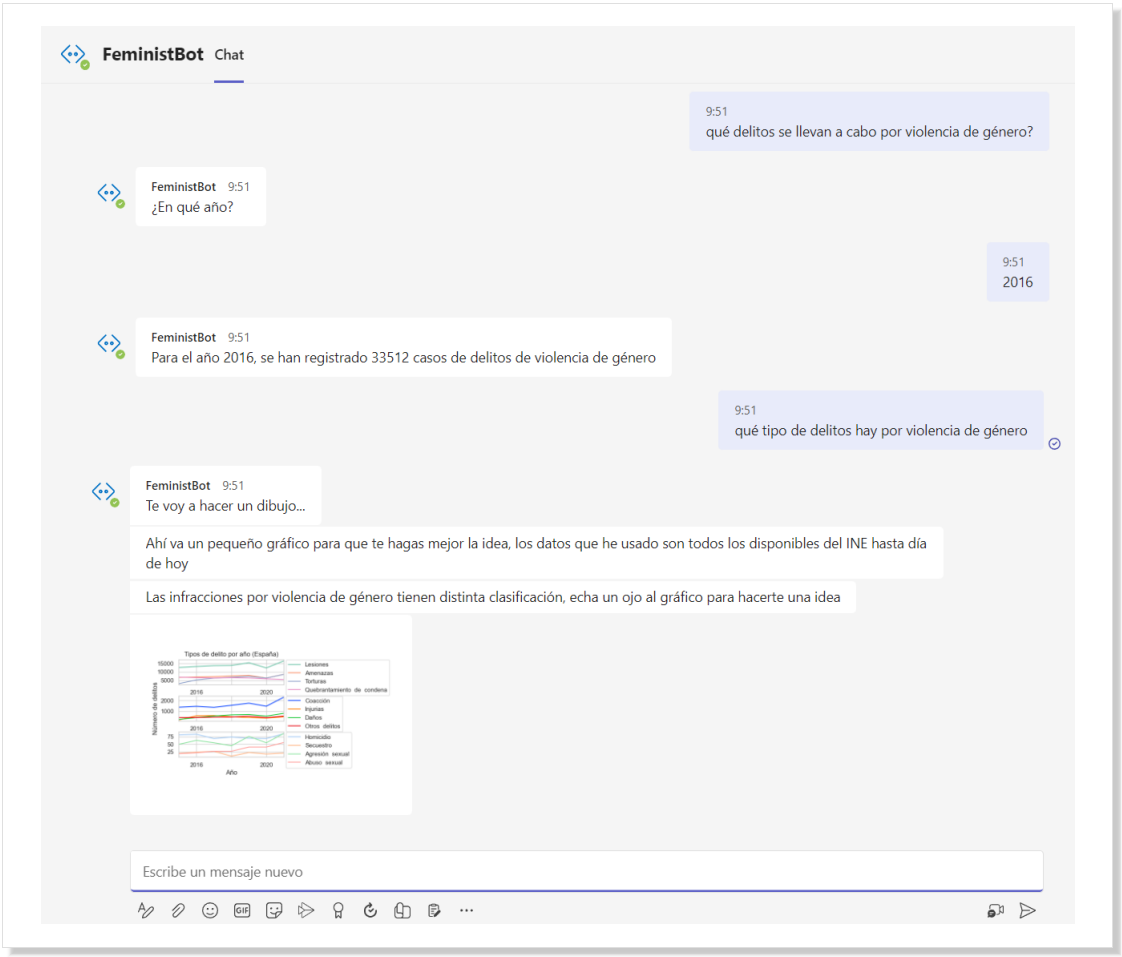

Figura 8.6: Consulta específica y genérica sobre delitos

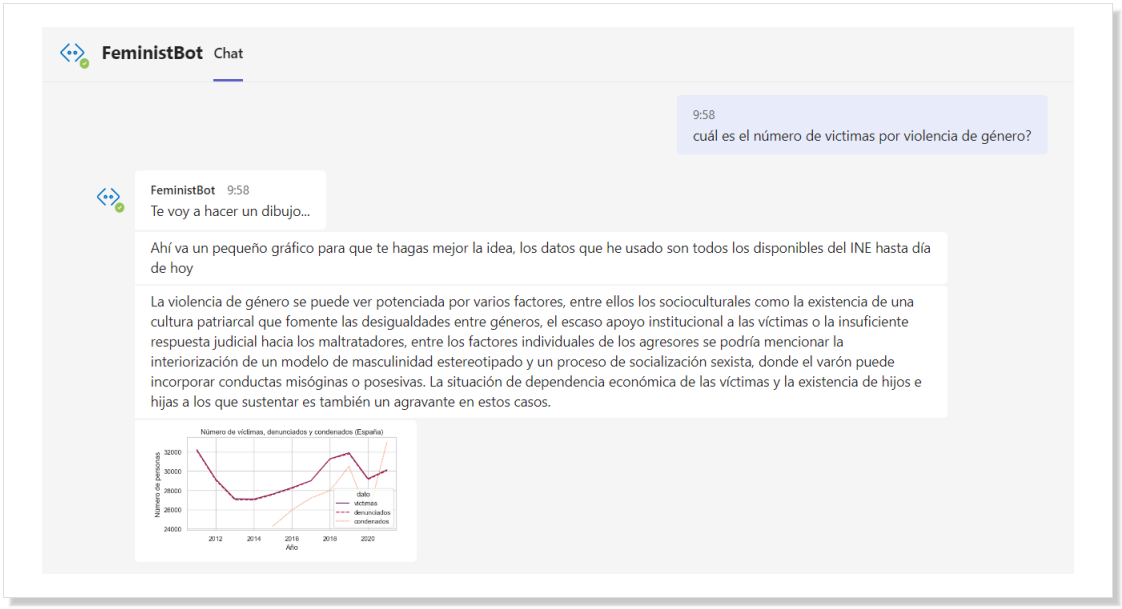

Figura 8.7: Consulta genérica sobre víctimas

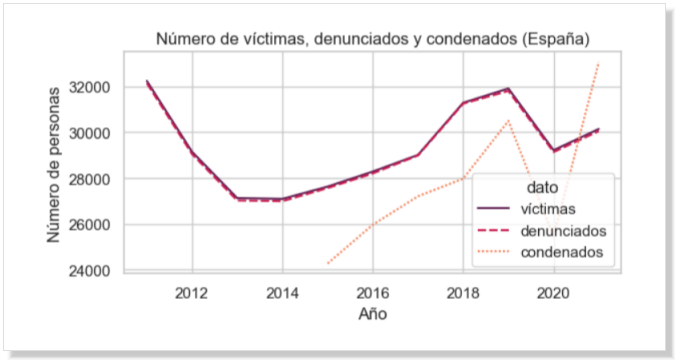

Figura 8.8: Gráfico generado sobre víctimas

Cuando el usuario hace una consulta específica pero el intervalo de tiempo es amplio, se le da la opción de ver una gráfica sobre los datos que ha pedido.

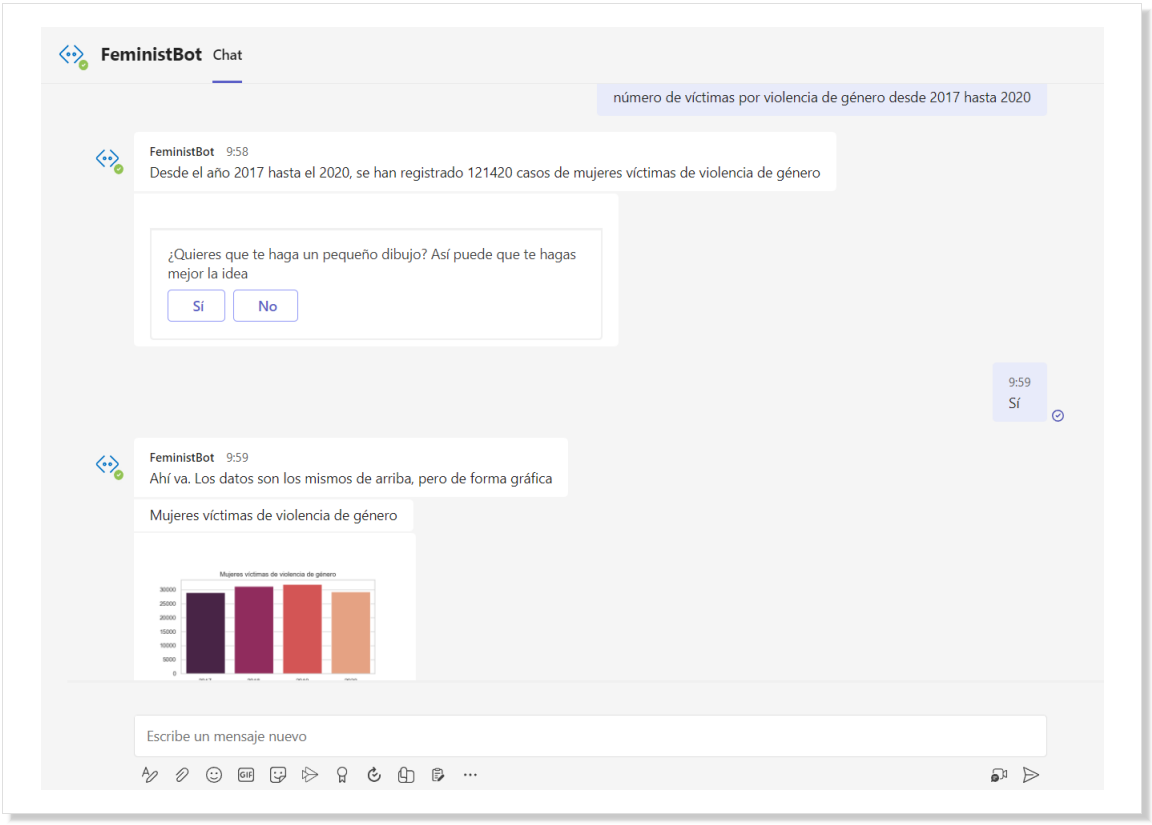

Figura 8.9: Consulta específica sobre víctimas que genera gráfico

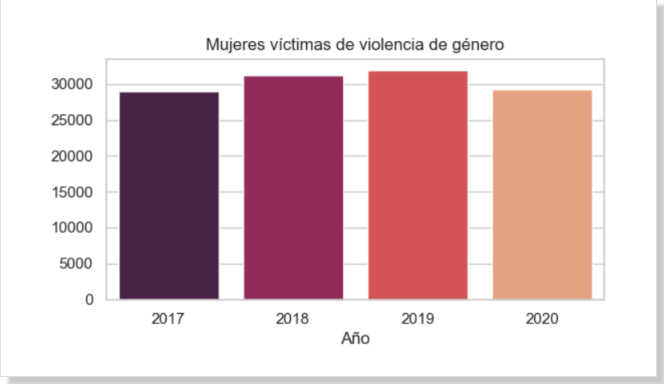

Figura 8.10: Gráfico sobre consulta específica de víctimas

Las consultas también se pueden realizar indicando un intervalo de tiempo abierto, por ejemplo "desde 2019" realizará la búsqueda desde el año indicado hasta el periodo actual; o utilizando un intervalo cerrado como por ejemplo "en el año 2020".

Se puede observar que cuando la intención del usuario es realizar una pregunta específica y el mismo no proporciona el dato del periodo de tiempo, el chatbot inquirirá sobre el año hasta que el usuario proporcione una respuesta válida.

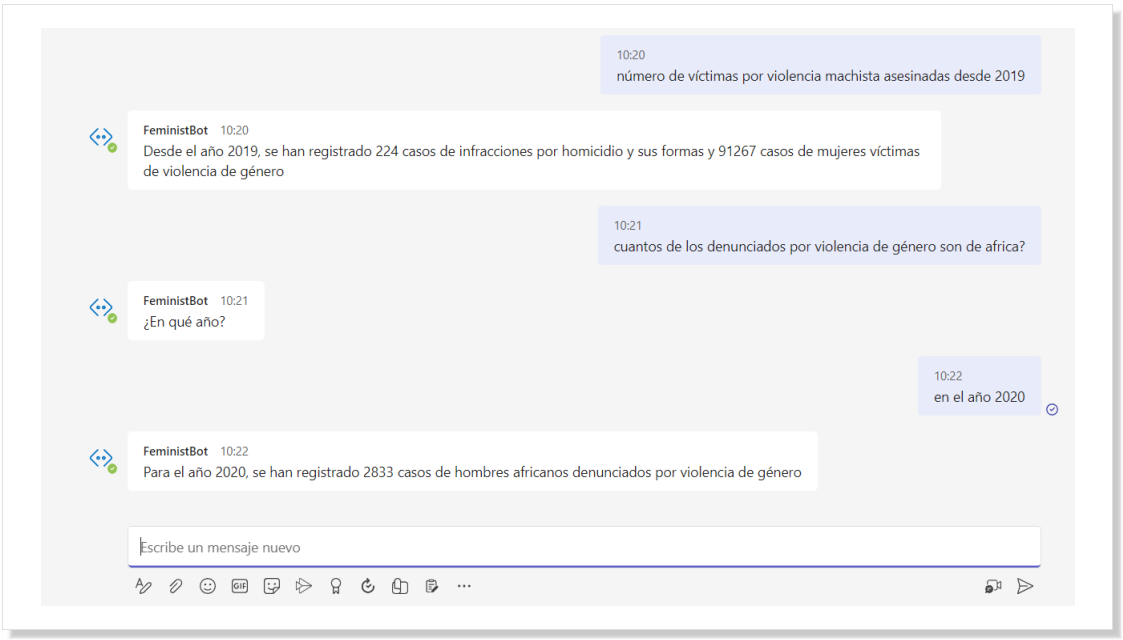

Figura 8.11: Consulta específica con intervalo de tiempo abierto y cerrado

En las siguientes imágenes se observa cómo el chatbot inquiere sobre el dato faltante un número determinado de veces para realizar una consulta al INE. Tras cuatro intentos el chatbot pide al usuario que sea más conciso, a lo que este finalmente responde con una fecha.

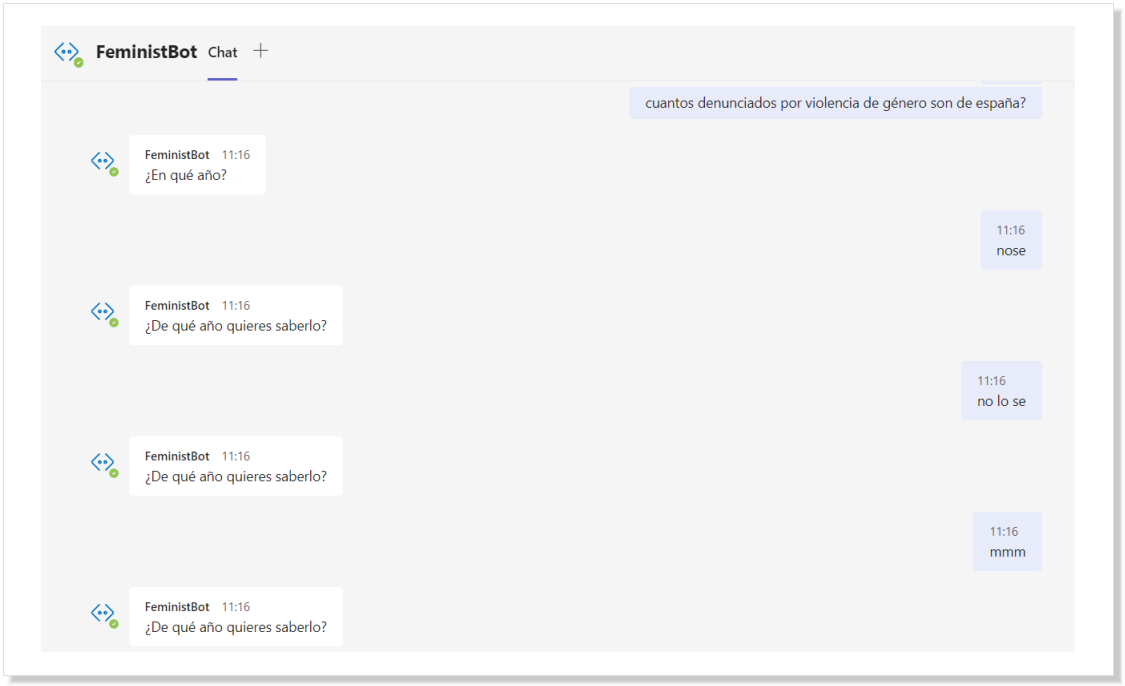

Figura 8.12: Chatbot inquiere sobre el dato fecha

Figura 8.13: Chatbot solicita al usuario que sea conciso tras un número de intentos solictando el dato fecha

En la siguiente imagen se observa que el usuario inicia una consulta sobre el INE, el chatbot inquiere sobre el dato faltante y a continuacón el usuario desea finalizar la interacción, a lo que el chatbot responde con un mensaje que asegura al usuario que ha entendido su intención.

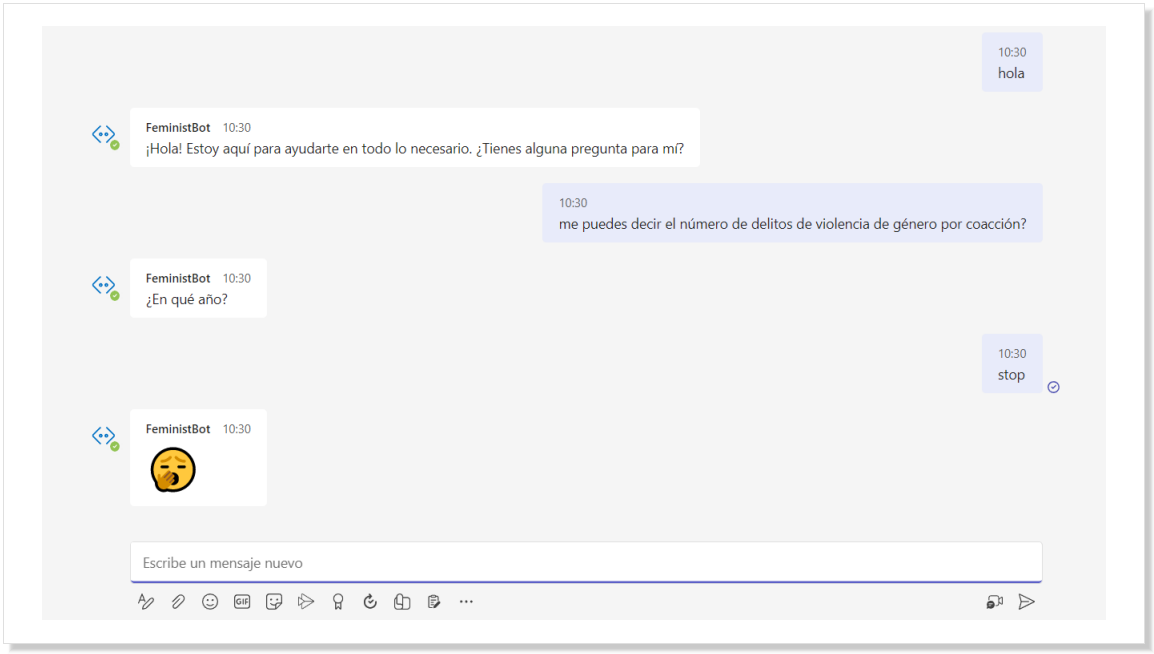

Figura 8.14: Cancelación de consulta

Consultas rápidas

Para las consultas rápidas de estructura Pregunta-Respuesta, es posible que se muestren imágenes descriptivas sobre lo que el usuario ha preguntado.

Figura 8.15: Ejemplos de consultas rápidas 1

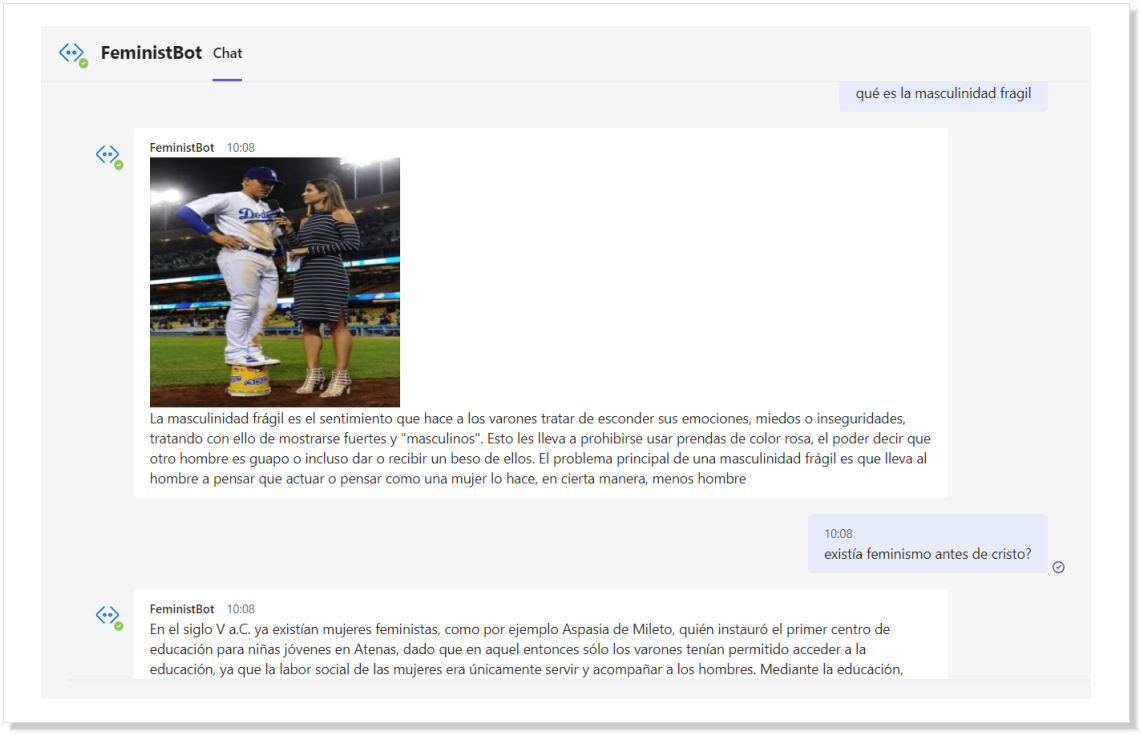

Figura 8.16: Ejemplos de consultas rápidas 2

A continuación se observan consultas del usuario sobre consejos para aplicar el feminismo.

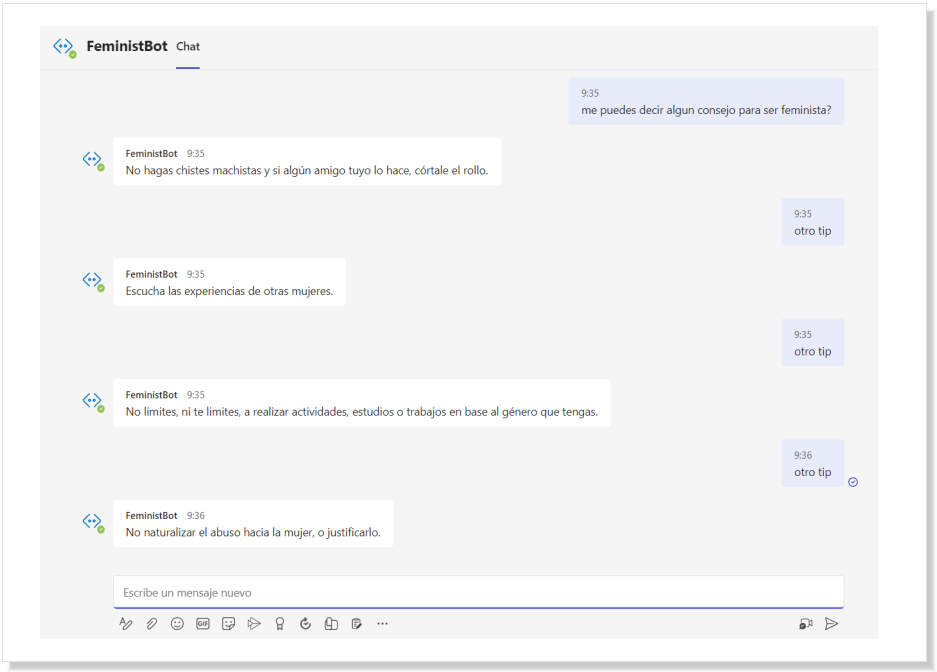

Figura 8.17: Consulta sobre consejos

A continuación se observa cómo el usuario realiza una pregunta y el bot no consigue extraer la intención con fidelidad por lo que pide al usuario que reformule. Tras la reformulación de la

pregunta, el chatbot consigue predecir la intención con un mayor porcentaje de confianza, para la cual genera una respuesta.

Como el usuario se detiene por dos minutos en los que no interactúa con el bot, el mismo llama su atención preguntándole si sigue estando al otro lado del chat, con la intención de continuar con la conversación.

Figura 8.18: Consulta sobre cómo identificar conductas machistas

Muchas consultas rápidas pueden generar tarjetas de diálogo, que permiten al usuario conocer más sobre el tema que ha preguntado haciendo clic sobre el botón de la tarjeta.

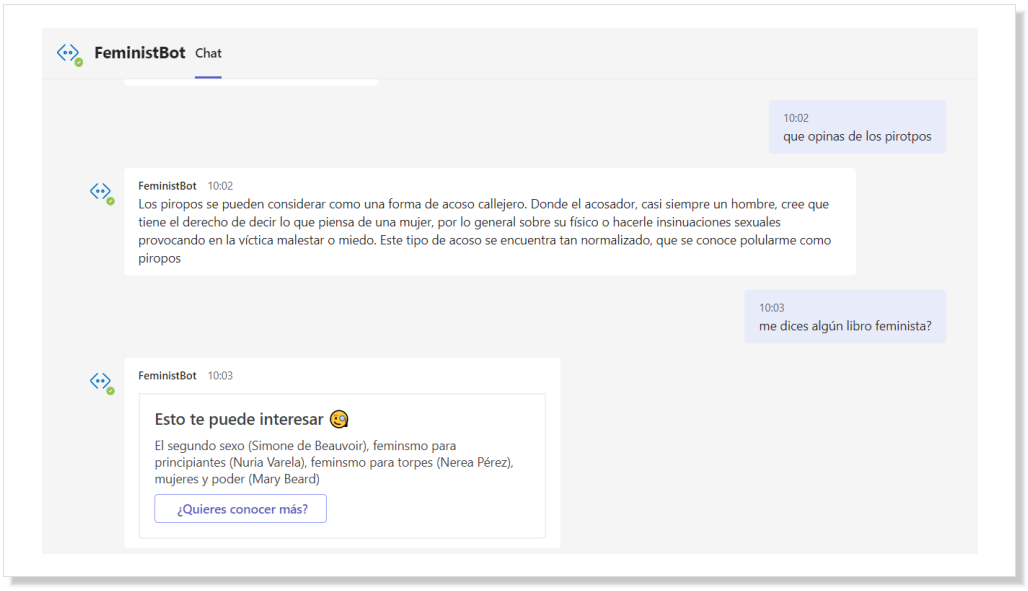

Figura 8.19: Consulta rápida que genera tarjeta

No es necesario preguntar por un término específico, sino que utilizando una frase descriptiva el chatbot puede asociarlo a un par Pregunta-Respuesta como se ve en el ejemplo “la ropa que lleva una chica justifica su violación?”.

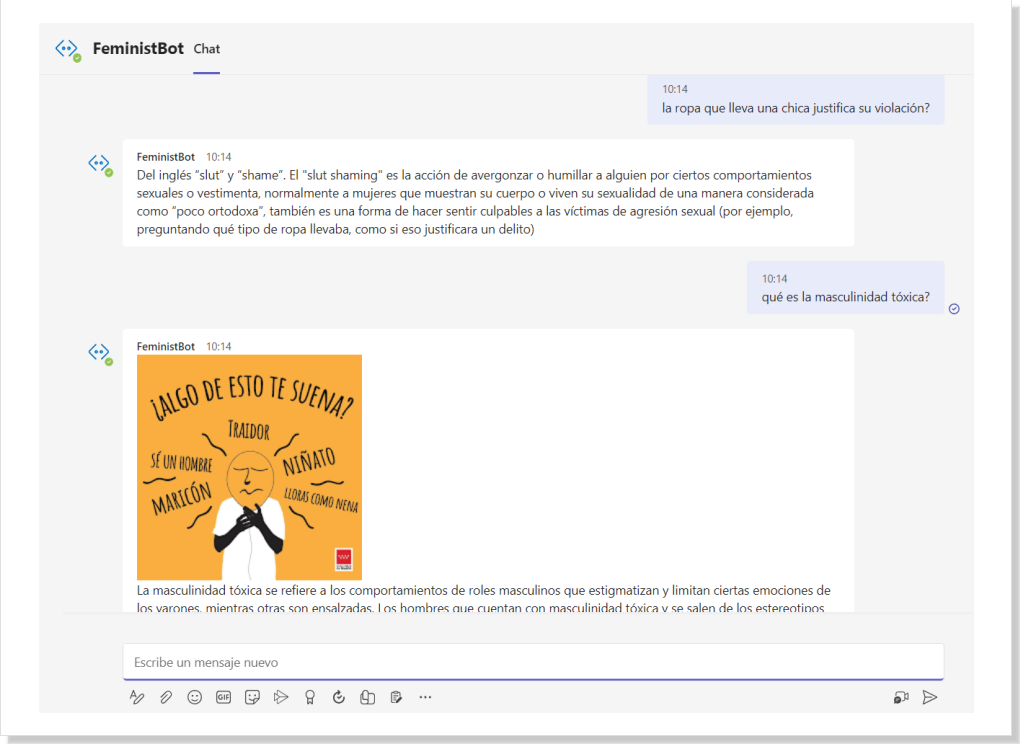

Figura 8.20: Ejemplo de frase que se asocia a un par Pregunta-Respuesta

Despedida

Tras utilizar el chatbot, el usuario podría despedirse del bot indicándolo con una frase o palabra, en este ejemplo "adios".

Se observa como el bot detiene al usuario y le pide una pequeña retroalimentación para conocer su satisfacción con la conversación que han tenido.

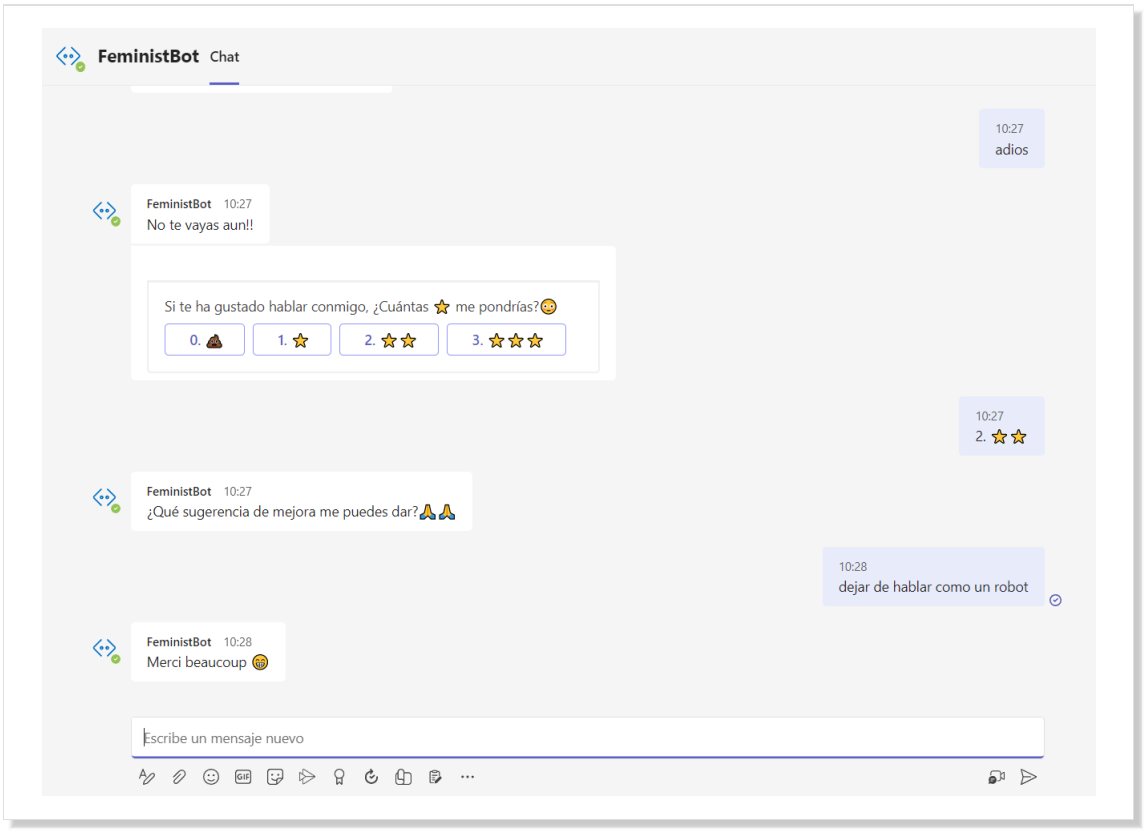

Figura 8.21: Despedida y petición de retroalimentación

Anexo II: Manual de instalación

Para proceder a la instalación del sistema, el usuario deberá contar con un dispositivo móvil u ordenador y tener instalada la aplicación Microsoft Teams.

El sistema actualmente no está disponible en la tienda online de Teams debido a que los administradores de la organización (UVa o Microsoft) deben verificar su uso y aceptar la aplicación en la tienda.

Cuando la misma esté disponible, se debe utilizar una cuenta profesional o educativa de MS Teams y agregar la aplicación siguiendo los pasos:

1. En el desplegable a la izquierda seleccionar el botón de "Aplicaciones" y en el buscador introducir FeministBot

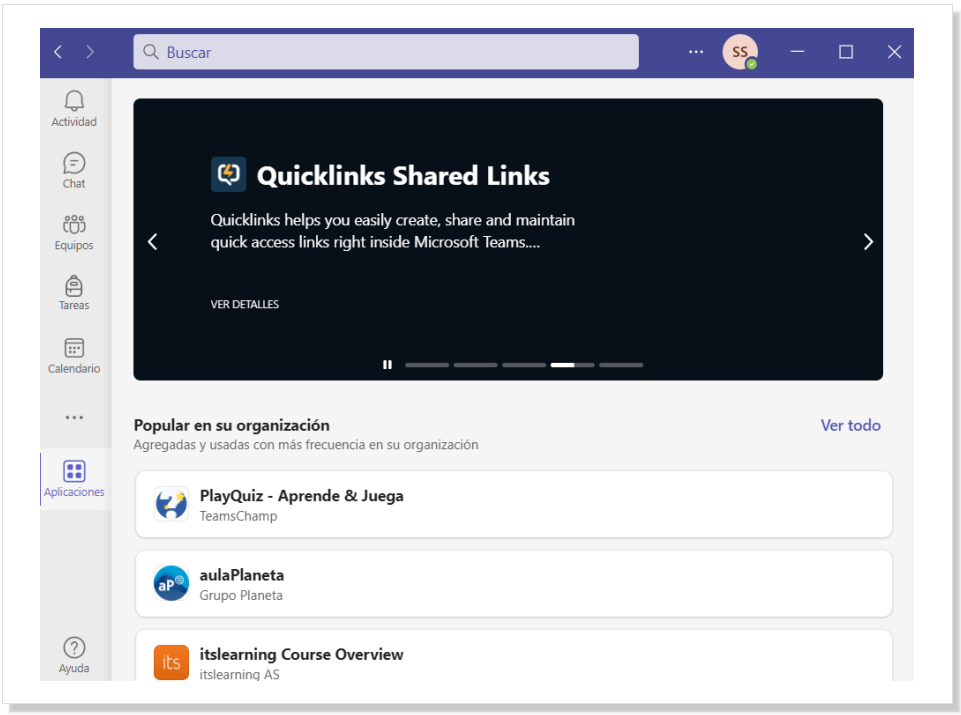

Figura 8.22: Aplicaciones en MS Teams

2. Agregar la aplicación a MS Teams (se deben tener suficientes permisos)

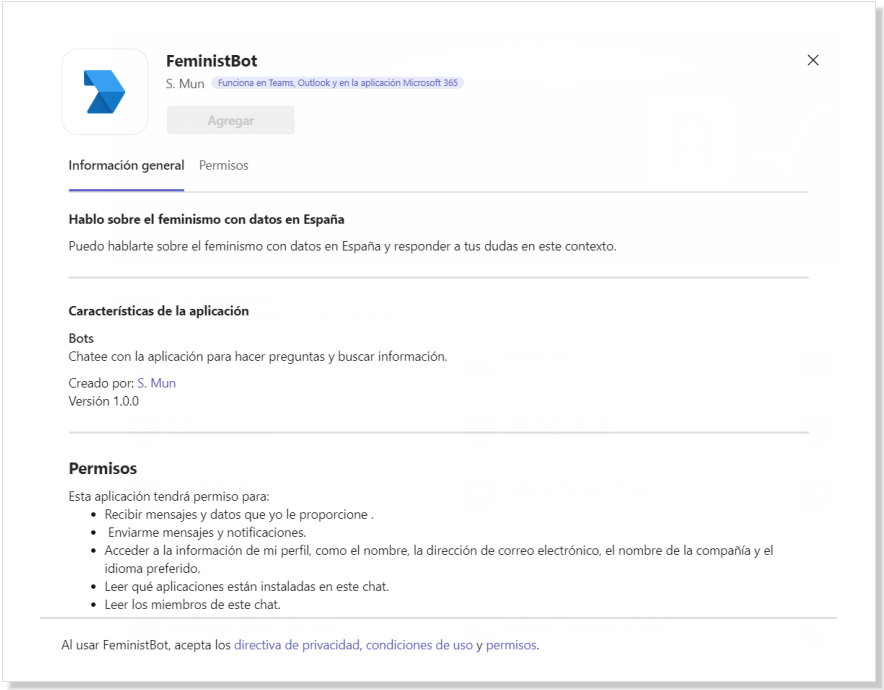

Figura 8.23: Agregar aplicación

Habiendo agregado la aplicación, ya se puede chatear con la aplicación dentro de Microsoft Teams.

Si no desea instalar la aplicación o cuando la misma aún no se encuentre disponible puede probarla mediante MS Teams, a través del siguiente enlace, que estará disponible temporalmente:
`https://teams.microsoft.com/l/chat/0/0?users=28:407c931f-2c03-45a0-89c6-594439f736a0`

Anexo III: Código fuente

El código fuente del proyecto se puede encontrar en el siguiente repositorio de GitLab: `https://gitlab.com/seheemun/tfg-project`