Universidad de Valladolid

Escuela de Ingeniería Informática

Trabajo Fin de Grado

Grado en Ingeniería Informática
Mención Tecnologías de la Información

Desarrollo de un gestor de agentes conversacionales configurables de propósito educativo

Autor:
D. Alejandro Merino Barrios

Universidad de Valladolid

Escuela de Ingeniería Informática

Trabajo Fin de Grado

Grado en Ingeniería Informática
Mención Tecnologías de la Información

Desarrollo de un gestor de agentes conversacionales configurables de propósito educativo

Autor:
D. Alejandro Merino Barrios
Tutores:
D.ª Alejandra Martínez Monés
D. Eduardo Gómez Sánchez

Agradecimientos

Me gustaría agradecer a todas las personas que me han ayudado durante mi formación académica y personal. A mis padres, Javier y Rosa por haber confiando en mí y por haberme ayudado y enseñado a lo largo de mi vida. A mis abuelos, Salvador y Elena por haberme querido y apoyado. A mis tíos, Elena y Carlos por haber sido como unos padres para mí. A mi pareja, Eva por haberme mostrado una apoyo incondicional durante el desarrollo de este Trabajo de Fin de Grado. Por lo último, gracias a todos los profesores que han formado parte durante mi formación académica como Ingeniero Informático.

Resumen

En este documento se describe el desarrollo de un gestor de agentes conversacionales configurables de propósito educacional. Se trata del *backend* de un sistema web en el que se puede gestionar y configurar profesores, cursos y agentes conversacionales. Además, cada agente corresponde con una actividad colaborativa en la que dos o más estudiantes interactúan sobre un tema. Los agentes conversacionales intervienen en la conversación en función de las entradas de los estudiantes o en función de diferentes eventos basados en el tiempo. Las conversaciones se desarrollan en una aplicación de chats de código libre, la cual se integra en el sistema para controlar todas sus operaciones. La aplicación escogida es Telegram. La realización de este proyecto sigue las fases de análisis, diseño, desarrollo y pruebas de un sistema software. Se ha desarrollado una arquitectura cliente-servidor con Flask estableciendo como medio de acceso a las operaciones una Rest API.

Abstract

This document describes the development of a configurable conversational agents manager for educational purposes. It is a web system backend where teachers, courses and conversational agents can be managed and configured. In addition, each agent corresponds to a collaborative activity in which two or more students interact on a topic. Conversational agents intervene in a conversation based on the students' inputs or based in different time-based events. Conversations take place in an open source chat application, which is integrated into the system to control all its operations. The chosen app is Telegram. The realization of this project follows the analysis, design, development and testing phases of a software system. A client-server architecture has been developed with Flask, establishing a Rest API as an access point to operations.

Índice general

1. Introducción **12**
- 1.1. Contexto . . . 12
- 1.2. Motivación . . . 14
- 1.3. Objetivos . . . 14
- 1.4. Organización de la memoria . . . 14

2. Plan del proyecto **17**
- 2.1. Metodología de desarrollo iterativo . . . 17
- 2.2. Planteamiento Inicial . . . 18
- 2.3. Análisis de riesgos . . . 19
- 2.4. Iteraciones . . . 24

3. Análisis **30**
- 3.1. Definición de conceptos, actores y acrónimos . . . 30
  - 3.1.1. Conceptos . . . 30
  - 3.1.2. Actores . . . 32
  - 3.1.3. Acrónimos . . . 32
- 3.2. Requisitos funcionales . . . 33
- 3.3. Requisitos no funcionales . . . 34
- 3.4. Requisitos de información . . . 35
- 3.5. Casos de uso . . . 37
  - 3.5.1. Casos de uso no contemplados . . . 57
- 3.6. Modelo de dominio . . . 57

4. Diseño **61**
- 4.1. Decisiones de diseño . . . 61
- 4.2. Arquitectura del sistema . . . 62
  - 4.2.1. Arquitectura del servidor . . . 62
  - 4.2.2. Diagrama de despliegue . . . 62
  - 4.2.3. Arquitectura Cliente . . . 63
- 4.3. Aplicación de chats . . . 63
- 4.4. Base de datos . . . 64
- 4.5. API Rest . . . 66
  - 4.5.1. Punto de acceso para iniciar sesión . . . 67
  - 4.5.2. Puntos de acceso para gestionar profesores . . . 67
  - 4.5.3. Puntos de acceso para gestionar cursos . . . 69
  - 4.5.4. Puntos de acceso para gestionar agentes conversacionales . . . 71
  - 4.5.5. Puntos de acceso para consultar salas de chat . . . 75
  - 4.5.6. Puntos de acceso para consultar estudiantes . . . 75

4.6. Patrón MC (Modelo Controlador) . . . 76
4.7. Diseño concurrente . . . 77
4.8. Sistema de colas . . . 77
4.9. Incorporación de múltiples idiomas . . . 79
4.10. Diagramas de flujo . . . 80
4.10.1. Inicio de sesión . . . 80
4.10.2. Gestión de un profesor . . . 81
4.10.3. Gestión de un curso . . . 85
4.10.4. Gestión de un agente conversacional . . . 90
4.10.5. Consulta de información de salas de chats . . . 96
4.10.6. Consulta de información de estudiantes . . . 97
4.10.7. Controlador de cursos . . . 98
4.10.8. Controlador de agentes conversacionales . . . 99

5. Desarrollo **102**
5.1. Entorno de Desarrollo . . . 102
5.2. Herramientas utilizadas . . . 102
5.3. Control de versiones . . . 102
5.4. Librerías utilizadas . . . 103
5.5. Estructura del proyecto . . . 104

6. Pruebas **107**
6.1. Casos de prueba . . . 107
6.1.1. Inicio de sesión . . . 107
6.1.2. Gestión de profesores . . . 108
6.1.3. Gestión de cursos . . . 110
6.1.4. Gestión de agentes conversacionales . . . 113
6.1.5. Búsqueda de un estudiante . . . 119
6.1.6. Búsqueda de una sala de chat . . . 120
6.1.7. Funcionalidades de cursos y agentes conversacionales . . . 120
6.2. Prueba de usabilidad con usuarios reales . . . 121

7. Conclusión **124**
7.1. Conclusiones . . . 124
7.2. Trabajo futuro . . . 125
7.3. Valoración personal . . . 125

Anexos **130**

A. Manual de instalación **130**
A.1. Requisitos . . . 130
A.2. Proceso de instalación . . . 130
A.3. Instalación de requisitos previos . . . 130
A.4. Configuración del proyecto . . . 130
A.5. Configuración de la base de datos . . . 131
A.6. Configuración de Tokens . . . 131
A.6.1. Información de Aplicación en Telegram . . . 132
A.6.2. Información MySQL . . . 132
A.6.3. Información Json Web Token . . . 132
A.6.4. Información de la Cola de Mensajes ZMQ: . . . 133

A.7. Ejecución del servidor 133
A.8. Configuración de ficheros de sesión 133

Capítulo 1

Introducción

1.1. Contexto

Un agente conversacional [1] es un sistema software que simula mantener una conversación con uno o más humanos a la vez. Estos agentes o *chatbots* proporcionan respuestas a los usuarios en función de las entradas introducidas por los mismos. En la actualidad, existe una gran cantidad de agentes conversacionales. Los más conocidos [2] son los asistentes virtuales *Siri*, *Alexa*, *Cortana* o *Google*. Además, pueden ser encontrados en cualquier ámbito, desde asistentes en una tienda de *eCommerce* hasta médicos virtuales.

Los agentes conversacionales se están aplicando también en el ámbito educativo, participando en conversaciones con grupos de estudiantes. En especial, estos agentes educacionales intervienen con el fin de fomentar la interacción entre los participantes. Dicha interacción se promueve mediante la realización de preguntas y comentarios, con el objetivo de que se profundice más en el tema del que se está hablando. A lo largo de estas conversaciones, el agente conversacional almacena trazas de lo sucedido en la conversación. Estas trazas se podrán analizar con múltiples propósitos.

Este proyecto de desarrollo de software se realiza bajo el contexto de trabajo fin de grado en Ingeniería Informática. Asimismo, está ligado al Grupo de Sistemas Inteligentes y Cooperativos / Educación, Medios, Informática y Cultura (GSIC/EMIC [1]). El grupo está involucrado en el proyecto europeo ColMOOC (Integrating Conversational Agents and Learning Analytics in MOOCs) [3]. El acrónimo MOOC [4] se refiere a *Massive Online Open Courses*, es decir, *cursos online masivos y abiertos*. La aparición de estos cursos ha sido una evolución de la educación abierta en internet. ColMOOC ofrece tecnologías inteligentes basadas en agentes conversacionales en MOOCs, con el objetivo de mejorar los beneficios de desencadenar las interacciones de los estudiantes.

Los agentes conversacionales del proyecto ColMOOC realizan intervenciones basadas en un *framework* de *charla académica productiva*, Academically Productive Talk (APT) [5]. El objetivo principal de estas intervenciones es que los estudiantes razonen los conceptos que se están debatiendo, teniendo en cuenta las contribuciones del resto de estudiantes en la conversación. Un mayor razonamiento de las tareas consigue una mayor capacidad de aprendizaje. Las intervenciones del agente son desencadenadas cuando se detecta

[1] GSIC/EMIC - Universidad de Valladolid: https://www.gsic.uva.es - último acceso: Julio 2020

uno o varios términos configurados en la estructura del agente conversacional. En las Figuras 1.1 y 1.2 se muestra un ejemplo de conversación entre un agente conversacional y dos estudiantes. Se puede observar como el agente conversacional, María, invita a los estudiantes a añadir un ejemplo sobre *los beneficios del aprendizaje colaborativo* y, más adelante, María invita al estudiante A a expresar de otro modo lo que ha dicho el estudiante B. De este modo, el agente intenta involucrar a ambos estudiantes a lo largo de la conversación.

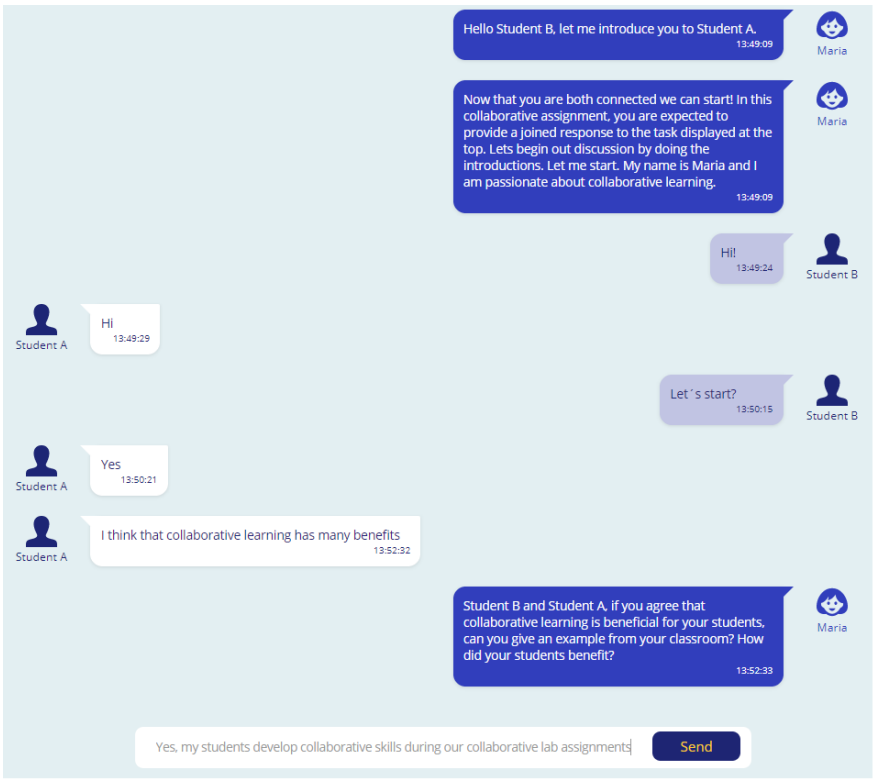


Figura 1.1: Agente Conversacional ColMOOC 1

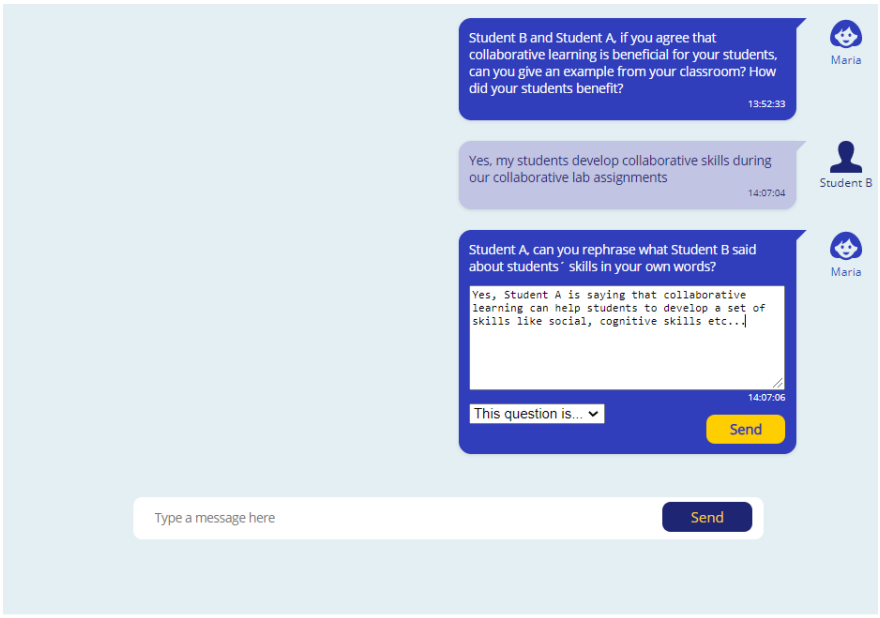


Figura 1.2: Agente Conversacional ColMOOC 2

1.2. Motivación

En el contexto del proyecto ColMOOC, arriba mencionado, desde el GSIC/EMIC se vio la necesidad de contar con una plataforma propia para configurar agentes conversacionales y, así, continuar con la investigación acerca de los mismos de forma autónoma. En la actualidad, el proyecto ColMOOC ofrece una plataforma web, en la que participan agentes conversacionales configurables, pero sus funcionalidades están muy restringidas y fuera del control del grupo de investigación.

El motivo de realización de este proyecto es la necesidad de una plataforma que permita la configuración de agentes conversacionales de propósito educacional a profesores, incorporando tecnologías y funcionalidades adicionales a las existentes en el ColMOOC. Además, surge la necesidad integrar una aplicación de chats, como WhastApp [2], Telegram [3], Discord [4], etc. De este modo, el acceso a la aplicación será más sencillo y el estudiante se encontrará en un medio que ya está acostumbrado a utilizar.

1.3. Objetivos

El objetivo de este proyecto consiste en el desarrollo del *backend* de un sistema web, encargado de gestionar agentes conversacionales de propósito educativo. También, tiene como objetivo integrar en el software una aplicación de chats de código abierto en la que se desarrollarán las conversaciones entre los alumnos y los diferentes agentes. En definitiva, los objetivos definidos han sido los siguientes:

- Desarrollar el *backend* de un sistema web para gestionar una plataforma de agentes conversacionales configurables.
- Desarrollar el software encargado del funcionamiento de los agentes conversacionales.
- Integrar las funcionalidades de los agentes conversacionales con una aplicación de chats de código abierto.

1.4. Organización de la memoria

La memoria ha sido organizada en nueve capítulos:

- **Capítulo 1. Introducción.** Presentación del proyecto.
- **Capítulo 2. Plan de proyecto.** Explicación de la metodología seguida para la realización del proyecto y de las diferentes etapas de las que ha constado.
- **Capítulo 3. Análisis.** Proceso de análisis de la aplicación. Consta de elicitación de requisitos, definición de conceptos, casos de uso y modelo de dominio.

[2]WhatsApp web: https://www.whatsapp.com - último acceso: Julio 2020
[3]Telegram web: https://web.telegram.org - último acceso: Julio 2020
[4]Discord web: https://discord.com - último acceso: Julio 2020

- **Capítulo 4. Diseño.** Fase de diseño del software. En él se incluye el diseño de las diferentes partes que forman el sistema.
- **Capítulo 5. Desarrollo.** Enumeración y explicación de las diferentes herramientas, tecnologías y técnicas utilizadas para el desarrollo del software.
- **Capítulo 6. Pruebas** Incluye una descripción de las pruebas realizadas.
- **Capítulo 7. Conclusión** Contiene los objetivos conseguidos, el trabajo futuro y una valoración personal.

Capítulo 2

Plan del proyecto

2.1. Metodología de desarrollo iterativo

Este proyecto ha sido realizado siguiendo la metodología de desarrollo iterativo [6]. Esta metodología surge de una evolución del modelo en cascada, como una mejora de sus carencias.

El desarrollo iterativo se compone de diferentes etapas, en las que se realiza un ciclo completo del proceso de ingeniería del software. En la Figura 2.1 se puede observar dicho proceso. Las fases de las que este se compone son:

- Análisis
- Diseño
- Desarrollo y pruebas
- Evaluación

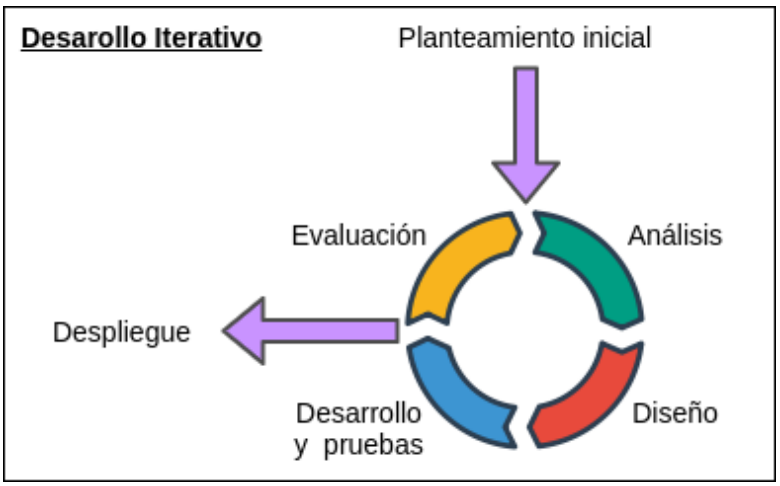


Figura 2.1: Metodología de Desarrollo Iterativo

Inicialmente, se realiza un primer planteamiento del proyecto con el que se pueda comenzar a iterar. Al final de cada ciclo se obtendrá una versión de la aplicación, con más funcionalidades que la anterior. Una vez completadas todas las iteraciones y, cuando los

requisitos estén validados, el sistema estará terminado .

Como se ha explicado en el primer capítulo, este proyecto está ligado al grupo de investigación GSIC/EMIC, en el que se tenía una idea inicial del proyecto. El principal motivo de que se haya seguido este tipo de metodología, ha sido que en el planteamiento inicial no estaban definidos todos los requisitos. Siguiendo este tipo de planificación, se pueden añadir y refinar requisitos a lo largo del proyecto.

El desarrollo iterativo se ajusta correctamente a este proyecto, puesto que permite la incorporación de nuevas necesidades o funcionalidades en cada iteración. Asimismo, se pueden realizar modificaciones sobre los requisitos al comienzo de cada etapa. De esta manera, esta metodología puede ser de gran utilidad para la corrección de fallos en el diseño o en el análisis.

2.2. Planteamiento Inicial

Al principio se establecieron iteraciones con una duración de quince días. Según avanzaba el estado del proyecto pasaron a tener una duración de siete días. Los hitos del proyecto venían marcados por el final de cada iteración, en el que tenía lugar una reunión.

La planificación de las tareas se realizaba en bloques de dos o tres iteraciones. Esto se debe a que se podrían generar nuevas tareas a lo largo de ese periodo de tiempo, que serían incorporadas en las siguientes iteraciones. Una vez se acercaba el final del ciclo, se planificaba el siguiente bloque. De la Figura 2.2 a 2.5 se pueden observar las diferentes tareas:

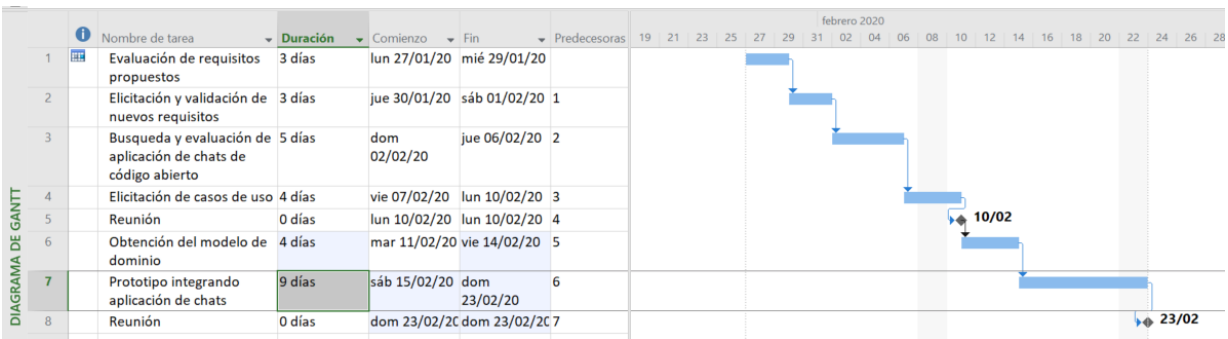
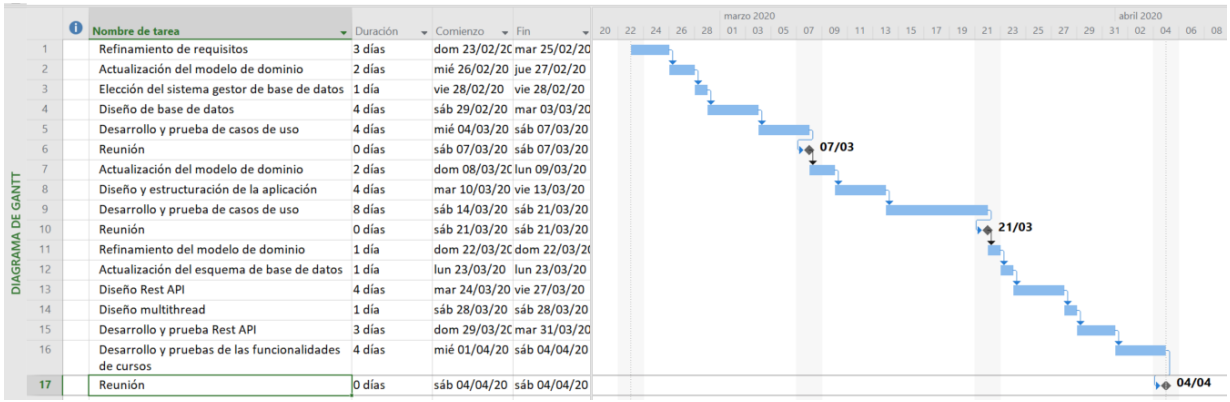
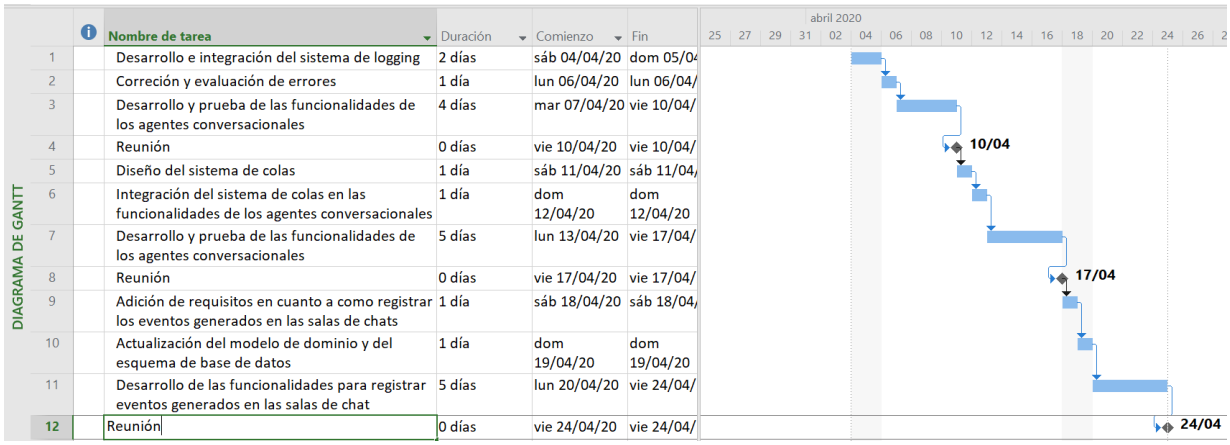


Figura 2.2: Iteraciones 1 y 2



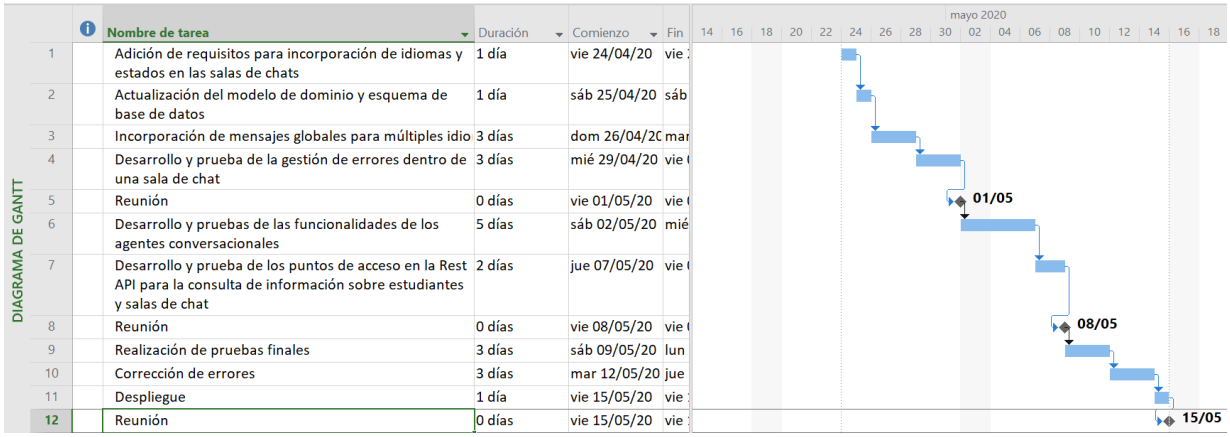
	Nombre de tarea	Duración	Comienzo	Fin
1	Refinamiento de requisitos	3 días	dom 23/02/20	mar 25/02/20
2	Actualización del modelo de dominio	2 días	mié 26/02/20	jue 27/02/20
3	Elección del sistema gestor de base de datos	1 día	vie 28/02/20	vie 28/02/20
4	Diseño de base de datos	4 días	sáb 29/02/20	mar 03/03/20
5	Desarrollo y prueba de casos de uso	4 días	mié 04/03/20	sáb 07/03/20
6	Reunión	0 días	sáb 07/03/20	sáb 07/03/20
7	Actualización del modelo de dominio	2 días	dom 08/03/20	lun 09/03/20
8	Diseño y estructuración de la aplicación	4 días	mar 10/03/20	vie 13/03/20
9	Desarrollo y prueba de casos de uso	8 días	sáb 14/03/20	sáb 21/03/20
10	Reunión	0 días	sáb 21/03/20	sáb 21/03/20
11	Refinamiento del modelo de dominio	1 día	dom 22/03/20	dom 22/03/20
12	Actualización del esquema de base de datos	1 día	lun 23/03/20	lun 23/03/20
13	Diseño Rest API	4 días	mar 24/03/20	vie 27/03/20
14	Diseño multithread	1 día	sáb 28/03/20	sáb 28/03/20
15	Desarrollo y prueba Rest API	3 días	dom 29/03/20	mar 31/03/20
16	Desarrollo y pruebas de las funcionalidades de cursos	4 días	mié 01/04/20	sáb 04/04/20
17	Reunión	0 días	sáb 04/04/20	sáb 04/04/20

Figura 2.3: Iteraciones 3, 4 y 5



	Nombre de tarea	Duración	Comienzo	Fin
1	Desarrollo e integración del sistema de logging	2 días	sáb 04/04/20	dom 05/04
2	Correción y evaluación de errores	1 día	lun 06/04/20	lun 06/04/
3	Desarrollo y prueba de las funcionalidades de los agentes conversacionales	4 días	mar 07/04/20	vie 10/04/
4	Reunión	0 días	vie 10/04/20	vie 10/04/
5	Diseño del sistema de colas	1 día	sáb 11/04/20	sáb 11/04/
6	Integración del sistema de colas en las funcionalidades de los agentes conversacionales	1 día	dom 12/04/20	dom 12/04/20
7	Desarrollo y prueba de las funcionalidades de los agentes conversacionales	5 días	lun 13/04/20	vie 17/04/
8	Reunión	0 días	vie 17/04/20	vie 17/04/
9	Adición de requisitos en cuanto a como registrar los eventos generados en las salas de chats	1 día	sáb 18/04/20	sáb 18/04/
10	Actualización del modelo de dominio y del esquema de base de datos	1 día	dom 19/04/20	dom 19/04/20
11	Desarrollo de las funcionalidades para registrar eventos generados en las salas de chat	5 días	lun 20/04/20	vie 24/04/
12	Reunión	0 días	vie 24/04/20	vie 24/04/

Figura 2.4: Iteraciones 6, 7 y 8



	Nombre de tarea	Duración	Comienzo	Fin
1	Adición de requisitos para incorporación de idiomas y estados en las salas de chats	1 día	vie 24/04/20	vie
2	Actualización del modelo de dominio y esquema de base de datos	1 día	sáb 25/04/20	sáb
3	Incorporación de mensajes globales para múltiples idio	3 días	dom 26/04/20	mar
4	Desarrollo y prueba de la gestión de errores dentro de una sala de chat	3 días	mié 29/04/20	vie
5	Reunión	0 días	vie 01/05/20	vie
6	Desarrollo y pruebas de las funcionalidades de los agentes conversacionales	5 días	sáb 02/05/20	mié
7	Desarrollo y prueba de los puntos de acceso en la Rest API para la consulta de información sobre estudiantes y salas de chat	2 días	jue 07/05/20	vie
8	Reunión	0 días	vie 08/05/20	vie
9	Realización de pruebas finales	3 días	sáb 09/05/20	lun
10	Corrección de errores	3 días	mar 12/05/20	jue
11	Despliegue	1 día	vie 15/05/20	vie
12	Reunión	0 días	vie 15/05/20	vie

Figura 2.5: Iteraciones 9, 10 y 11

2.3. Análisis de riesgos

Un riesgo [7] es la medida de la probabilidad de sufrir consecuencias adversas producidas por eventos futuros. El análisis de riesgos es un procedimiento que permite

la identificación y categorización de los mismos. La cuantificación de los riesgos se debe realizar siguiendo una matriz de valoración, la cual se puede ver en la Figura 2.6.

Matriz de valoración de riesgos		Consecuencias			
		Insignificante	Moderado	Dañino	Extremo
Probabilidad	Muy alta	Medio	Alto	Crítico	Crítico
	Alta	Medio	Alto	Alto	Crítico
	Media	Bajo	Medio	Alto	Alto
	Baja	Bajo	Bajo	Medio	Medio

Figura 2.6: Matriz de valoración de riesgos [8]

De la Tabla 2.1 a la Tabla 2.10 aparecen los diferentes riesgos identificados, junto con su probabilidad y su impacto o consecuencias:

Riesgo	R01 - Mala planificación
Descripción	Estimación errónea del tiempo que va a llevar una tarea. Esto produce que el resto de tareas no cumplan las fechas establecidas.
Probabilidad	Alta
Impacto	Dañino
Plan de contingencia	Priorizar la tareas con mayor importancia y reajustar la planificación del resto de tareas.

Tabla 2.1: R01: Mala planificación

Riesgo	R02 - Indisponibilidad del personal
Descripción	La persona encargada del desarrollo del proyecto no se encuentra disponible por motivos personales o enfermedad. Por esto, las fechas del proyecto no siguen el curso establecido.
Probabilidad	Baja
Impacto	Dañino
Plan de contingencia	Planificar de nuevo las tareas que se han visto afectadas teniendo en cuenta las fechas limite de cada iteración.

Tabla 2.2: R02: Indisponibilidad del personal

Riesgo	R03 - Falta de comprensión de los requisitos
Descripción	Los requisitos son malinterpretados por el desarrollador, produciendo así desajustes en la planificación del proyecto.
Probabilidad	Alta
Impacto	Moderado
Plan de contingencia	Contactar con el cliente para clarificar dichos requisitos y solucionar el error. Ajustar la planificación de las tareas afectadas.

Tabla 2.3: R03: Falta de comprensión de los requisitos

Riesgo	R04 - Cambio en los requisitos
Descripción	Una modificación en los requisitos ya existentes produce la creación de nuevas tareas, obligando a reajustar la planificación.
Probabilidad	Muy alta
Impacto	Moderado
Plan de contingencia	Incorporar nuevas tareas y reajustar la planificación para incorporar los cambios.

Tabla 2.4: R04: Cambio en los requisitos

Riesgo	R05 - Fallo en el diseño
Descripción	El diseño no se ajusta a lo esperado. Esto implica la creación de nuevas tareas para su modificación.
Probabilidad	Media
Impacto	Dañino
Plan de contingencia	Reajustar la planificación y corregir el error.

Tabla 2.5: R05: Fallo en el diseño

Riesgo	R06 - Fallo en el desarrollo
Descripción	Las funcionalidades desarrolladas contienen errores o *bugs*. La planificación será modificada para solucionar estos problemas.
Probabilidad	Muy alta
Impacto	Moderado
Plan de contingencia	Reajustar la planificación y corregir el error.

Tabla 2.6: R06: Fallo en el desarrollo

Riesgo	R07 - Falta de conocimientos sobre ciertas tecnologías
Descripción	Se desconocen ciertas tecnologías y por lo tanto se necesita aprender su utilización. Este proceso de aprendizaje requiere modificar la planificación.
Probabilidad	Media
Impacto	Moderado
Plan de contingencia	Reajustar planificación e incorporar tareas de aprendizaje para dichas tecnologías.

Tabla 2.7: R07: Falta de conocimientos sobre ciertas tecnologías

Riesgo	R08 - Perdida de información
Descripción	Perdida de documentación, código o incluso versiones de la aplicación.
Probabilidad	Baja
Impacto	Extremo
Plan de contingencia	Realizar una evaluación del impacto y reajustar la planificación.

Tabla 2.8: R08: Perdida de información

Riesgo	R09 - Fallo en el equipo
Descripción	Problemas en la máquina en la que se desarrolla el proyecto.
Probabilidad	Baja
Impacto	Extremo
Plan de contingencia	Reparar la maquina o sustituir la misma por una nueva. Ajustar la planificación en función del tiempo que se ha perdido.

Tabla 2.9: R09: Fallo en el equipo

Riesgo	R10 - Fallo en la aplicación de chats
Descripción	Los servidores de la aplicación de chats se encuentran inactivos.
Probabilidad	Baja
Impacto	Dañino
Plan de contingencia	Priorizar tareas que no impliquen el uso de la aplicación. Posponer las tareas que impliquen su uso.

Tabla 2.10: R10: Fallo en la aplicación de chats

A lo largo del proyecto han tenido lugar varios de estos riesgos. Para cada riesgo se ha aplicado el correspondiente plan de contingencia. Durante las primeras iteraciones se dieron los siguientes:

- R01 - Mala planificación: varias tareas tuvieron una estimación errónea de la carga de trabajo. Por lo tanto, se tuvieron que actualizar las tareas posteriores.
- R02 - Indisponibilidad del personal: debido a enfermedad y a situaciones personales, el desarrollador no pudo cumplir con las fechas limite de ciertas tareas.
- R03 - Falta de compresión de los requisitos: al principio del proyecto surgieron bastantes dudas en cuanto a los requisitos del sistema. Por eso, estos se fueron refinando según avanzaban las iteraciones. Esto supuso la incorporación de nuevas tareas en la planificación y la modificación de las existentes.
- R07 - Falta de conocimientos sobre ciertas tecnologías: dado que se trataba de las primeras etapas, tuvieron lugar nuevas tareas con el objetivo de aprender ciertas tecnologías.

Posteriormente, se produjeron los siguiente riesgos durante las iteraciones restantes:

- R01 - Mala planificación
- R02 - Indisponibilidad del personal
- R04 - Cambio en los requisitos: ciertos requisitos sufrieron ligeras variaciones. Se tuvo que adaptar la planificación para incorporar dichos cambios.
- R05 - Fallo en el diseño: se realizaron varias decisiones de diseño que tuvieron que ser modificadas. Esto conllevó el aplazar ciertas tareas para corregir el diseño.
- R06 - Fallo en el desarrollo: durante las pruebas del sistema se encontraron varios errores, los cuales tuvieron que ser arreglados.

Debido a la aparición de estos sucesos, las iteraciones sufrieron diferentes cambios durante su desarrollo:

- Tareas aplazadas para próximas iteraciones
- Incorporación de nuevas tareas a la iteración en curso
- Ampliación de la duración de las tareas

Además, las iteraciones no han tenido duraciones exactas dado que el final de las mismas venía marcado por una reunión. Ciertas reuniones tuvieron que ser pospuestas, alterando así la planificación global del proyecto. En la siguiente sección, se muestran las fechas exactas que han tenido las diferentes etapas, junto con las tareas que se completaron en cada una de ellas.

2.4. Iteraciones

Las iteraciones han tenido diferentes duraciones a lo largo del proyecto. Las primeras abarcaban alrededor de dos semanas, mientras que las últimas, una semana.

El final de cada iteración venía marcado por una reunión. En ella se producía la fase de evaluación del trabajo realizado por parte del cliente, en este caso, los tutores. De este modo, se planificaba las tareas a realizar en la siguiente iteración. En la Tabla 2.11 se puede observar los periodos de las once iteraciones que se han llevado a cabo.

Iteración	Inicio	Fin
1	27 de Enero	12 de Febrero
2	12 de Febrero	25 de Febrero
3	25 de Febrero	11 de Marzo
4	11 de Marzo	1 de Abril
5	1 de Abril	21 de Abril
6	21 de Abril	29 de Abril
7	29 de Abril	12 de Mayo
8	12 de Mayo	19 de Mayo
9	19 de Mayo	27 de Mayo
10	27 de Mayo	5 de Junio
11	5 de Junio	12 de Junio

Tabla 2.11: Calendario de iteraciones

El porcentaje de las fases del proceso de ingeniería del software no es el mismo en cada iteración. Al principio, las tareas tienen un mayor porcentaje de análisis y diseño. Por el contrario, según avanza el estado del proyecto, las actividades tienden a estar más enfocadas en el desarrollo y prueba del software. De la Tabla 2.12 a la tabla 2.22 aparecen las tareas de cada iteración.

Iteración	1
Tareas	▪ Evaluación de requisitos propuestos ▪ Elicitación y validación de nuevos requisitos ▪ Búsqueda y evaluación de aplicación de chats de código abierto

Tabla 2.12: Tareas Iteración 1

Iteración	2
Tareas	■ Elicitación de casos de uso ■ Obtención del modelo de dominio ■ Prototipo integrando aplicación de chats

Tabla 2.13: Tareas Iteración 2

Iteración	3
Tareas	■ Refinamiento de requisitos ■ Actualización del modelo de dominio ■ Elección del sistema gestor de base de datos ■ Diseño de la base de datos

Tabla 2.14: Tareas Iteración 3

Iteración	4
Tareas	■ Refinamiento de requisitos ■ Actualización del modelo de dominio ■ Diseño y estructuración de la aplicación ■ Desarrollo y prueba de los diferentes casos de uso

Tabla 2.15: Tareas Iteración 4

Iteración	5
Tareas	■ Refinamiento del modelo de dominio ■ Actualización del esquema de base de datos ■ Diseño Rest API ■ Diseño multithread para la incorporación de las funcionalidades de cursos y agentes conversacionales ■ Desarrollo y prueba Rest API ■ Desarrollo y prueba de las funcionalidades de cursos en la aplicación de chats

Tabla 2.16: Tareas Iteración 5

Iteración	6
Tareas	■ Desarrollo e integración del sistema de logging ■ Corrección y evaluación de errores ■ Desarrollo y prueba de las funcionalidades de agentes conversacionales en la aplicación de chats

Tabla 2.17: Tareas Iteración 6

Iteración	7
Tareas	■ Diseño del sistema colas ■ Integración del sistema colas en las funcionalidades de los agentes conversacionales

Tabla 2.18: Tareas Iteración 7

Iteración	8
Tareas	▪ Adición de requisitos en cuanto a como registrar los eventos generados en las salas de chat ▪ Actualización del modelo de dominio y esquema de base de datos ▪ Desarrollo de las funcionalidades para registrar los eventos generados en las salas de chat

Tabla 2.19: Tareas Iteración 8

Iteración	9
Tareas	▪ Adición de requisitos para incorporación de idiomas y estados en las salas de chats ▪ Actualización del modelo de dominio y esquema de base de datos ▪ Incorporación de mensajes globales para múltiples idiomas ▪ Desarrollo y prueba de la gestión de errores dentro de una sala de chat

Tabla 2.20: Tareas Iteración 9

Iteración	10
Tareas	▪ Desarrollo y prueba funcionalidades del agente conversacional ▪ Desarrollo y prueba de los puntos de acceso en la Rest API para la consulta de información sobre estudiantes y salas de chat

Tabla 2.21: Tareas Iteración 10

Iteración	11
Tareas	▪ Realización de pruebas finales ▪ Corrección de errores ▪ Despliegue

Tabla 2.22: Tareas Iteración 11

En la Tabla 2.13 se puede observar como la tarea *Elicitación de casos de uso* tuvo que ser aplazada de la iteración uno a la dos. El *Desarrollo y prueba de los casos de uso*, planificado en la iteración número tres, Tabla 2.14, fue pospuesto dado que se estimó erróneamente la carga de trabajo de las tareas previas. Además, en la iteración número cuatro, Tabla 2.15, se creó la tarea *Refinamiento de requisitos* para modificar y corregir los mismos. En la iteración número siete, Tabla 2.18, se eliminó la tarea *Desarrollo y prueba de las funcionalidades de los agentes conversacionales* ya que fue terminada en la etapa anterior, en un tiempo menor de lo planificado.

Capítulo 3

Análisis

En este capítulo, se explica detalladamente el sistema software a desarrollar. Se definen los conceptos y se especifican los requisitos, los actores, los casos de uso y el modelo de dominio. El diseño del sistema estará basado en este capítulo.

3.1. Definición de conceptos, actores y acrónimos

A continuación, se realiza una definición de los actores, conceptos y acrónimos que se van a utilizar a lo largo de esta memoria.

3.1.1. Conceptos

- **Curso (Course):** es el lugar en el que se publicarán los diferentes agentes conversacionales. En él, se podrá elegir con qué agente se desea mantener una conversación.
- **Sala de chat (Chatroom):** en ella tiene lugar la conversación entre el agente conversacional y los participantes. Estos, llegan a la sala de chat a través de la elección de un agente en un curso. Dentro de este proyecto, los participantes serán estudiantes del curso. Esta sala de chat, como se verá en los requisitos de información, tiene un estado. Este estado puede ser uno de los siguientes:
  - Esperando a alumnos (Waiting for Students)
  - Activa (Active)
  - Terminado con envío de resultado (Terminated with Submission)
  - Terminado sin envío de resultado (Terminated without Submission)
- **Agente Conversacional (Conversational Agent):** un chatbot o agente conversacional es un sistema software que participa en conversaciones con humanos. Interviene en dicha conversación en función de los mensajes de los diferentes participantes. Además, el agente tiene la posibilidad de dirigirse personalmente a uno o a varios de los participantes.
- **Patrón (Move):** un patrón forma parte de la estructura de un agente conversacional, el cual se identifica con una intervención. En función de las entradas de los participantes en la conversación, se lanzará un patrón u otro. Existen tres tipos de patrones: fijos,

opcionales y basados en conversación.

Los patrones fijos siempre van a formar parte de un agente conversacional:

- **Task:** el agente envía un mensaje cuando la sala de chat se completa, explicando la tarea a debatir.
- **Welcome:** el agente envía un mensaje de bienvenida cuando se une un participante a la sala de chat.
- **Initial Agent Silence:** segundos iniciales durante los cuales el agente conversacional no interviene.
- **Silence After Agent Talked:** segundos durante los que el agente conversacional no participa, después de haber intervenido.

Los patrones opcionales pueden estar, o no, configurados en un agente:

- **Break Student Silence:** el agente intenta reanimar la conversación con un mensaje cuando los estudiantes llevan un número de segundos sin haber participado.
- **Bring Back To Topic:** el agente intenta traer a los alumnos al tema tras llevar un número de mensajes sin encontrar ninguna condición que dispare un patrón.
- **Encourage One Student:** el agente intenta reanimar la participación de un estudiante cuando el otro lleva un número de intervenciones seguidas.
- **Encourage Submission:** el agente intenta animar a los estudiantes a enviar la tarea cuando han pasado al menos un número de minutos desde el comienzo de la conversación, y los estudiantes han hecho en total un número de intervenciones.

Al igual que los patrones opcionales, los basados en conversación pueden estar, o no, configurados en el agente. Además, pueden existir varios patrones del mismo tipo, pero configurados con diferente información:

- **Explain:** el agente invita al estudiante a añadir algo más sobre algo que ha mencionado.
- **Add On:** el agente invita a otro estudiante a añadir algo acerca de lo que el primer estudiante ha dicho.
- **Rephrase:** el agente invita a otro estudiante a expresar de otra manera distinta lo que el primer estudiante ha dicho.
- **Agree Disagree:** el agente invita a otro estudiante a estar de acuerdo o en desacuerdo sobre algo que el primer estudiante ha dicho.
- **Build On:** el agente sugiere a los estudiantes que hay alguna relación de lo que acaban de decir con otro concepto importante.
- **Verify:** el agente pide a los estudiantes comprobar una relación entre dos conceptos que acaban de mencionar.
- **Revoice:** el agente reformula lo que cree que los estudiantes han dicho, de una manera más formal.

- **Registros de una sala de chat (Chatroom Logs):** almacena los eventos que se producen en una sala de chat. Estos tipos pueden ser:

- Creación de la sala de chat (Chatroom Creation)
- Inicio de la sala de chat (Chatroom Init)
- Unión de un alumno (Student Joined)
- Intervención de un alumno (Student Intervened)
- Envío de un resultado por un alumno (Student Submitted Result)
- Confirmación de un resultado por un alumno (Student Confirmed Result)
- Salida de un alumno (Student Left)
- Intervención de un agente conversacional basada en un concepto y dirigida (Conversational Agent Intervened One Concept Directed)
- Intervención de un agente conversacional basada en un concepto y no dirigida (Conversational Agent Intervened One Concept Not Directed)
- Intervención de un agente conversacional basada en dos conceptos y dirigida (Conversational Agent Intervened Two Concepts Directed)
- Intervención de un agente conversacional basada en dos conceptos y no dirigida (Conversational Agent Intervened Two Concepts Not Directed)
- Intervención de un agente conversacional dirigida (Conversational Agent Intervened No Concepts Directed)
- Intervención de un agente conversacional no dirigida (Conversational Agent Intervened No Concepts Not Directed)

3.1.2. Actores

- **Profesor (Teacher):** es el actor encargado de las gestión de los cursos y agentes conversacionales en el sistema. También pueden gestionar a otros profesores, pero solo si tienen permisos de administración.

- **Alumno (Student):** es el actor que participa en las conversaciones con los diferentes agentes conversacionales. Él escoge el agente al que desea unirse desde un curso.

3.1.3. Acrónimos

En la Tabla 3.1 se detallan los acrónimos:

Acrónimo	Descripción
API	Application Programming Interface: conjunto de funciones y procedimientos que cumplen una o muchas funciones con el fin de ser utilizadas por otro software
Rest API	Representational State Transfer Application Programming Interface
CRUD	Create, Read, Update, Delete
JSON	JavaScript Object Notation
JWT	JSON Web Token
ZMQ	Zero Message Queue - Sistema de colas de mensajes
XML	Extensible Markup Language (Lenguaje de Marcado Extensible)
ISO	International Organization for Standardization (Organización internacional de Normalización)

Tabla 3.1: Acrónimos

3.2. Requisitos funcionales

Los requisitos funcionales describen las funcionalidades que incorporará el sistema. A lo largo de este proyecto se han establecido los siguientes requisitos:

- **RF01**: El sistema permitirá a profesores, con permisos de administración, realizar las operaciones CRUD sobre otros profesores.
- **RF02**: El sistema permitirá a los profesores el inicio de sesión.
- **RF03**: El sistema deberá permitir a un profesor crear un curso.
- **RF04**: El sistema deberá permitir a un profesor consultar la información de un curso.
- **RF05**: El sistema deberá permitir a un profesor actualizar la información de un curso.
- **RF06**: El sistema permitirá a un profesor, creador del curso, añadir a otros profesores al mismo.
- **RF07**: El sistema permitirá a un profesor, creador del curso, eliminar profesores del mismo.
- **RF08**: El sistema deberá permitir a un profesor realizar las operaciones CRUD sobre agentes conversacionales, dentro de un curso.
- **RF09**: El sistema permitirá a un profesor activar un agente conversacional, dentro de un curso.
- **RF10**: El sistema permitirá a un profesor desactivar un agente conversacional, dentro de un curso.

- **RF11**: El sistema deberá permitir visualizar la información de los estudiantes, dentro de un curso, a los profesores correspondientes.
- **RF12**: El sistema permitirá visualizar la información de salas de chats, asociadas a un curso, a los profesores correspondientes.
- **RF13**: El sistema permitirá que los estudiantes puedan unirse a una sala de chat con un agente conversacional, dentro de un curso.
- **RF14**: El sistema deberá permitir la interacción entre los estudiantes y los agentes conversacionales, dentro de una sala de chat.
- **RF15**: El sistema deberá permitir la gestión de patrones opcionales por parte del agente conversacional.
- **RF16**: El sistema deberá permitir la gestión de patrones basados en conversación por parte del agente conversacional.
- **RF17**: El sistema deberá ser capaz de gestionar la salida de estudiantes de salas de chats no terminadas.
- **RF18**: El sistema deberá actualizar correctamente el estado de las salas de chats.
- **RF19**: El sistema deberá registrar a los estudiantes cuando estos soliciten unirse a una sala de chat con un agente conversacional por primera vez.
- **RF20**: El sistema deberá soportar múltiples idiomas.
- **RF21**: El sistema deberá registrar todos los eventos generados en una sala de chat.

3.3. Requisitos no funcionales

Los requisitos no funcionales representan propiedades emergentes del sistema, como fiabilidad o tiempo de respuesta. Se muestran a continuación:

- **RNF01**: El sistema utilizará MySQL como sistema gestor de base de datos relacionales.
- **RNF02**: El sistema deberá estar disponible 24 horas, los 365 días del año.
- **RNF03**: El sistema soportará múltiples idiomas.
- **RNF04**: El sistema deberá ser accesible a través de peticiones HTTP, mediante una Rest API.
- **RNF05**: El sistema responderá en un tiempo máximo de 6 segundos.
- **RNF06**: El sistema usará Telegram como aplicación de chats de código libre, accesible desde dispositivos móviles, aplicaciones de escritorio y web.
- **RNF07**: El sistema utilizará el formato de ficheros JSON como medio de comunicación con el servidor.
- **RNF08**: El sistema deberá permitir la integración de un *frontend* en un futuro.

- **RNF09**: El sistema deberá permitir la integración de otra aplicación de chats en un futuro.
- **RNF10**: El estudiante deberá ser capaz de unirse a un agente conversacional en menos de 3 minutos la primera vez que usa el sistema.
- **RNF11**: El estudiante debe ser capaz de enviar y confirmar un resultado (una vez realizada la discusión) en menos de 3 minutos la primera vez que usa el sistema.
- **RNF12**: La tasa de errores cometidos por un estudiante deberá ser menor del 2 % de las intervenciones realizadas.
- **RNF13**: El estudiante deberá ser capaz de entender los errores cometidos y cómo recuperarse de ellos.

3.4. Requisitos de información

Los requisitos de información especifican la información a almacenar en el sistema de cada una de las entidades. A continuación, se pueden observar los requisitos de información:

- **RI01. Profesor (Teacher)**:
  - Identificador
  - Nombre de usuario
  - Contraseña
  - Teléfono
  - Identificador de usuario en la aplicación de chats
  - Nombre de usuario en la aplicación de chats
  - Permisos de administración
- **RI02. Alumno (Student)**:
  - Identificador
  - Teléfono
  - Identificador de usuario en la aplicación de chats
  - Nombre de usuario en la aplicación de chats
- **RI03. Curso (Course)**:
  - Identificador
  - Título
  - Fecha de creación
  - Descripción
  - Identificador de grupo en la aplicación de chats
  - Idioma del curso

- **RI04. Agente Conversacional (Conversational Agent)**:
  - Identificador
  - Hash de la tarea
  - Título de la tarea
  - Icono de la tarea
  - Descripción de la tarea
  - Idioma del agente
  - Tamaño de las salas de chat
  - Mensajes de bienvenida
  - Ventana de análisis gramático en el chat
  - Intervención mediante respuestas
  - Segundos de silencio inicial
  - Segundos de silencio después de intervenir
  - Estado
- **RI05. Sala de chat (Chatroom)**:
  - Identificador
  - Fecha de creación
  - Identificador de grupo en la aplicación de chats
  - Estado
- **RI06. Patrones Opcionales (Optional Moves)**:
  - Identificador
  - Identificador del patrón
  - Mensaje
  - Prioridad
  - Segundos
  - Minutos
  - Número de mensajes
- **RI07. Patrones basados en Conversación (Conversation Moves)**:
  - Identificador
  - Identificador del patrón
  - Mensaje
  - Prioridad
  - Dirigido
  - Primer concepto
  - Sinónimos del primer concepto

- Segundo concepto
- Sinónimos del segundo concepto

- **RI08. Registros de un Chatroom (Chatroom Logs)**:

  - Identificador
  - Sello de tiempo
  - Tipo de evento
  - Identificador del estudiante
  - Número de mensaje
  - Contenido del mensaje
  - Identificador del patrón
  - Primer termino desencadenado
  - Número de mensaje del primer termino desencadenado
  - Segundo termino desencadenado
  - Número de mensaje del segundo termino desencadenado
  - Identificador del estudiante al que se ha referido el agente conversacional
  - Contenido de la intervención

3.5. Casos de uso

Los casos de uso representan el conjunto de interacciones que se van a poder realizar con el sistema. Al tratarse del *backend* de un sistema web, estos se identifican con las acciones a realizar desencadenadas desde el futuro *frontend*. A pesar de esto, los casos de uso relacionados con el funcionamiento de los agentes conversacionales en la aplicación de chats, sí que tendrán como desencadenantes unos actores, los estudiantes.

En la Figura 3.1 aparece el diagrama de casos de uso. Se trata de un diagrama bastante lineal debido a que, al tratarse del *backend* de un sistema web, cada operación tiene lugar independientemente, sin incluir ni ser una especificación de ninguna otra operación. Si se tuviera el correspondiente *frontend* los casos de uso tendrían diferentes interdependencias, por ejemplo, el caso de uso *actualización de un profesor* podría incluir el caso de uso *consulta/búsqueda de información de un profesor*.

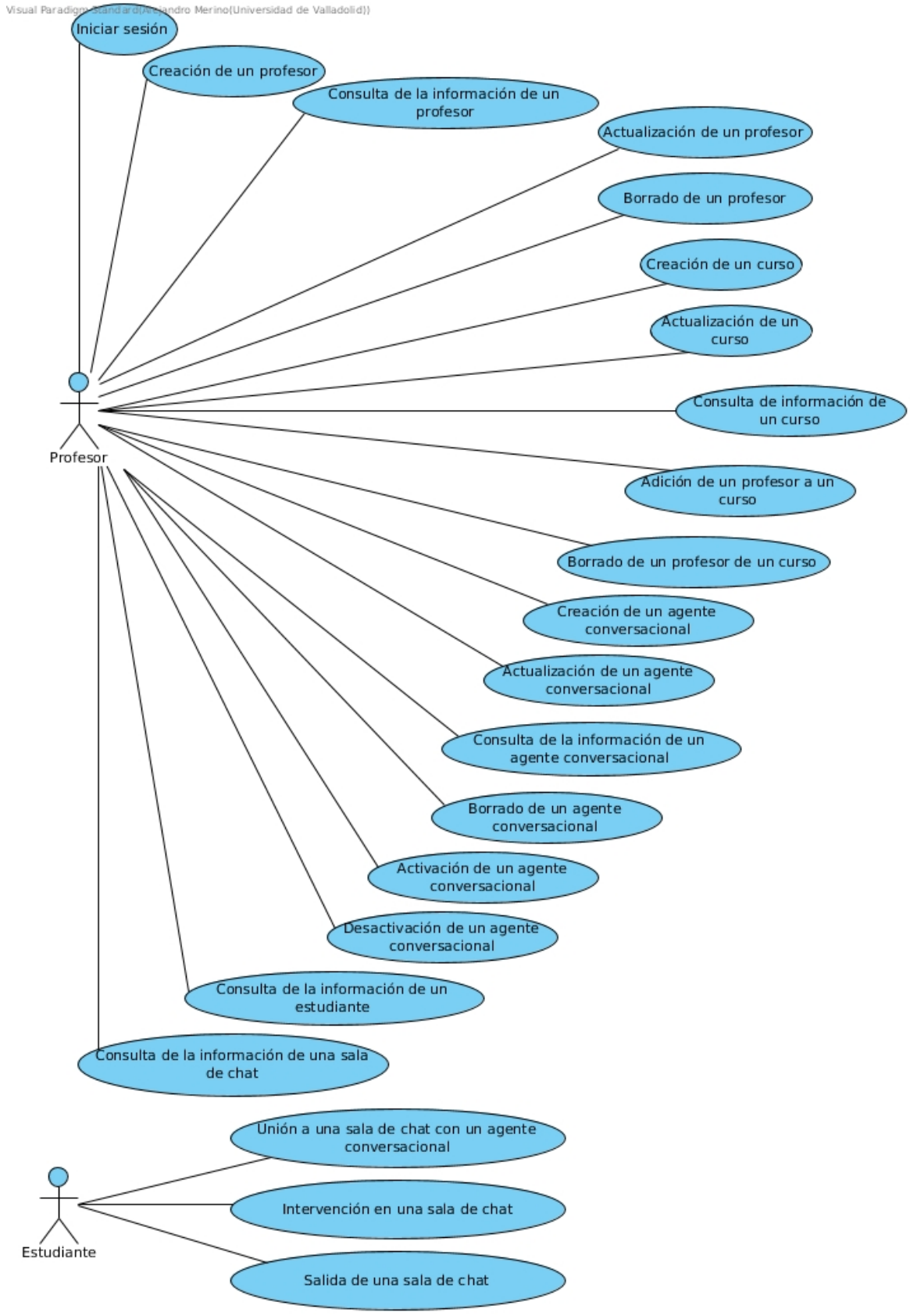


Figura 3.1: Diagrama de casos de uso

De la Tabla 3.2 a la Tabla 3.22 se muestra la especificación de todos los casos de uso:

Caso de uso	Un profesor inicia sesión
Identificador	CU01
Actor	Profesor
Precondición	El profesor está creado en el sistema
Descripción	El profesor inicia sesión con su nombre usuario y contraseña
Secuencia normal	1. El sistema recibe una petición de login 2. El sistema obtiene el nombre de usuario y la contraseña 3. El sistema comprueba que los datos son correctos 4. El sistema muestra el token de acceso y el caso de uso finaliza
Postcondición	El profesor ha iniciado sesión en el sistema
Excepciones	4a. El sistema comprueba que los datos no son correctos y el caso de uso finaliza

Tabla 3.2: Caso de uso: Inicio de sesión

Caso de uso	Creación de un profesor
Identificador	CU02
Actor	Profesor con permisos de administración
Precondición	El profesor ha iniciado sesión en el sistema
Descripción	El profesor con permisos de administración crea a un profesor
Secuencia normal	1. El sistema recibe una petición para crear a un profesor 2. El sistema obtiene los datos del profesor, de la petición 3. El sistema comprueba que los datos no son nulos 4. El sistema comprueba que el profesor está registrado en la aplicación de chats 5. El sistema registra al nuevo profesor en el sistema y el caso de uso finaliza
Postcondición	Se ha creado un profesor en el sistema
Excepciones	3a. El sistema comprueba que los datos son nulos y el caso de uso finaliza 4a. El sistema comprueba que el profesor no está registrado en la aplicación de chats y el caso de uso finaliza.

Tabla 3.3: Caso de uso: Creación de un profesor

Caso de uso	Consulta de información de un profesor
Identificador	CU03
Actor	Profesor con permisos de administración
Precondición	El profesor ha iniciado sesión en el sistema
Descripción	El profesor con permisos de administración consulta la información de un profesor
Secuencia normal	1. El sistema recibe una petición para consultar la información de un profesor 2. El sistema obtiene los datos del profesor a consultar, de la petición 3. El sistema obtiene la información del profesor 4. El sistema muestra la información del profesor y el caso de uso finaliza
Postcondición	Se ha consultado la información de un profesor
Excepciones	3a. El sistema no encuentra el profesor solicitado y el caso de uso finaliza

Tabla 3.4: Caso de uso: Consulta de la información de un profesor

Caso de uso	Actualización de un profesor
Identificador	CU04
Actor	Profesor con permisos de administración
Precondición	El profesor ha iniciado sesión en el sistema
Descripción	El profesor con permisos de administración actualiza los datos de un profesor
Secuencia normal	1. El sistema recibe una petición para actualizar un profesor 2. El sistema obtiene los datos del profesor a actualizar, de la petición 3. El sistema obtiene la información del profesor 4. El sistema comprueba que los datos no son nulos 5. El sistema comprueba que los datos han sido actualizados 6. El sistema comprueba que el profesor está registrado en la aplicación de chats 7. El sistema registra la nueva información del profesor y el caso de uso finaliza
Postcondición	Se han actualizado los datos de un profesor
Excepciones	3a. El sistema no encuentra el profesor solicitado y el caso de uso finaliza 4a. El sistema comprueba que los datos son nulos y el caso de uso finaliza 5a. El sistema comprueba que los datos no han sido modificados y el caso de uso finaliza 6a. El sistema comprueba que el profesor no está registrado en la aplicación de chats y el caso de uso finaliza.

Tabla 3.5: Caso de uso: Actualización de un profesor

Caso de uso	Borrado de un profesor
Identificador	CU05
Actor	Profesor con permisos de administración
Precondición	El profesor ha iniciado sesión en el sistema
Descripción	El profesor con permisos de administración borra a un profesor del sistema
Secuencia normal	1. El sistema recibe una petición para borrar a un profesor 2. El sistema obtiene los datos del profesor a consultar, de la petición 3. El sistema obtiene la información del profesor 4. El sistema registra el borrado del profesor y el caso de uso finaliza
Postcondición	Se ha borrado a un profesor del sistema
Excepciones	3a. El sistema no encuentra el profesor solicitado y el caso de uso finaliza

Tabla 3.6: Caso de uso: Borrado de un profesor

Caso de uso	Creación de un curso
Identificador	CU06
Actor	Profesor
Precondición	El profesor ha iniciado sesión en el sistema
Descripción	El profesor crea un curso
Secuencia normal	1. El sistema recibe una petición para crear un curso 2. El sistema obtiene los datos del curso, de la petición 3. El sistema comprueba que los datos no son nulos 4. El sistema crea el grupo en la aplicación de chats 5. El sistema registra el curso, muestra un enlace de invitación al grupo y el caso de uso finaliza
Postcondición	Se ha creado un curso en el sistema
Excepciones	3a. El sistema comprueba que los datos son nulos y el caso de uso finaliza

Tabla 3.7: Caso de uso: Creación de un curso

Caso de uso	Actualización de un curso
Identificador	CU07
Actor	Profesor
Precondición	El profesor ha iniciado sesión en el sistema
Descripción	El profesor actualiza la información un curso
Secuencia normal	1. El sistema recibe una petición para actualizar un curso 2. El sistema obtiene los datos del curso a actualizar, de la petición 3. El sistema obtiene la información del curso 4. El sistema comprueba que el profesor es el creador o tiene permisos de administración 5. El sistema comprueba que los datos no son nulos 6. El sistema comprueba que los datos han sido actualizados 7. El sistema actualiza la información del grupo en la aplicación de chats 8. El sistema registra la nueva información del curso y el caso de uso finaliza
Postcondición	Se han actualizado los datos de un curso en el sistema
Excepciones	3a. El sistema no encuentra el curso y el caso de uso finaliza 4a. El sistema comprueba que el profesor no es el creador del curso y no tiene permisos de administración. El caso de uso finaliza 5a. El sistema comprueba que los datos son nulos y el caso de uso finaliza 6a. El sistema comprueba que los datos no han sido actualizados y el caso de uso finaliza

Tabla 3.8: Caso de uso: Actualización de un curso

Caso de uso	Consulta de información de un curso
Identificador	CU08
Actor	Profesor
Precondición	El profesor ha iniciado sesión en el sistema
Descripción	El profesor consulta la información de un curso
Secuencia normal	1. El sistema recibe una petición para consultar la información de un curso 2. El sistema obtiene los datos del curso, de la petición 3. El sistema obtiene la información del curso 4. El sistema muestra la información del curso y el caso de uso finaliza
Postcondición	Se ha consultado la información un curso
Excepciones	3a. El sistema no encuentra el curso y el caso de uso finaliza

Tabla 3.9: Caso de uso: Consulta de información de un curso

Caso de uso	Adición de un profesor a un curso
Identificador	CU09
Actor	Profesor
Precondición	El profesor ha iniciado sesión en el sistema
Descripción	El profesor añade un profesor a un curso
Secuencia normal	1. El sistema recibe una petición para añadir a un profesor a un curso 2. El sistema obtiene los datos del profesor y el curso, de la petición 3. El sistema obtiene la información del curso 4. El sistema comprueba que el profesor es el creador o tiene permisos de administración 5. El sistema obtiene la información del profesor 6. El sistema comprueba que el profesor no está en el curso 7. El sistema añade al profesor al grupo en la aplicación de chats 8. El sistema registra la adición del profesor y el caso de uso finaliza
Postcondición	Se ha añadido un profesor a un curso
Excepciones	3a. El sistema no encuentra el curso y el caso de uso finaliza 4a. El sistema comprueba que el profesor no es el creador del curso y no tiene permisos de administración. El caso de uso finaliza 5a. El sistema no encuentra el profesor y el caso de uso finaliza 6a. El sistema comprueba que el profesor ya está en el curso y el caso de uso finaliza

Tabla 3.10: Caso de uso: Adición de un profesor a un curso

Caso de uso	Borrado de un profesor de un curso
Identificador	CU10
Actor	Profesor
Precondición	El profesor ha iniciado sesión en el sistema
Descripción	El profesor borra a un profesor de un curso
Secuencia normal	1. El sistema recibe una petición para borrar a un profesor de un curso 2. El sistema obtiene los datos del profesor y el curso, de la petición 3. El sistema obtiene la información del curso 4. El sistema comprueba que el profesor es el creador o tiene permisos de administración 5. El sistema obtiene la información del profesor 6. El sistema comprueba que el profesor está en el curso 7. El sistema elimina al profesor del grupo en la aplicación de chats 8. El sistema registra el borrado del profesor y el caso de uso finaliza
Postcondición	Se ha borrado un profesor de un curso
Excepciones	3a. El sistema no encuentra el curso y el caso de uso finaliza 4a. El sistema comprueba que el profesor no es el creador del curso y no tiene permisos de administración. El caso de uso finaliza 5a. El sistema no encuentra el profesor y el caso de uso finaliza 6a. El sistema comprueba que el profesor no está en el curso y el caso de uso finaliza

Tabla 3.11: Caso de uso: Borrado de un profesor de un curso

Caso de uso	Creación de un agente conversacional
Identificador	CU11
Actor	Profesor
Precondición	El profesor ha iniciado sesión en el sistema
Descripción	El profesor crea un agente conversacional dentro de un curso
Secuencia normal	1. El sistema recibe una petición para crear un agente conversacional 2. El sistema obtiene los datos del profesor, del curso y del agente conversacional, de la petición 3. El sistema obtiene la información del curso 4. El sistema comprueba que el profesor pertenece al curso o tiene permisos de administración 5. El sistema comprueba que no existe un agente conversacional con el mismo identificador dentro del curso 6. El sistema comprueba que los datos del agente conversacional son válidos 7. El sistema registra el agente conversacional y el caso de uso finaliza
Postcondición	Se ha creado un agente conversacional en el sistema
Excepciones	3a. El sistema no encuentra el curso y el caso de uso finaliza 4a. El sistema comprueba que el profesor no pertenece al curso y no tiene permisos de administración. El caso de uso finaliza 5a. El sistema comprueba que existe un agente conversacional con el mismo identificador dentro del curso y el caso de uso finaliza 6a. El sistema comprueba que los datos del agente conversacional no son validos y el caso de uso finaliza

Tabla 3.12: Caso de uso: Creación de un agente conversacional

Caso de uso	Actualización de un agente conversacional
Identificador	CU12
Actor	Profesor
Precondición	El profesor ha iniciado sesión en el sistema
Descripción	El profesor actualiza los datos de un agente conversacional dentro de un curso
Secuencia normal	1. El sistema recibe una petición para actualizar un agente conversacional 2. El sistema obtiene los datos del profesor, del curso y del agente conversacional, de la petición 3. El sistema obtiene la información del curso 4. El sistema obtiene la información del agente conversacional 5. El sistema comprueba que el profesor pertenece al curso o tiene permisos de administración 6. El sistema comprueba que no existe un agente conversacional con el mismo identificador dentro del curso 7. El sistema comprueba que los datos del agente conversacional son válidos 8. El sistema registra los nuevos datos del agente conversacional y el caso de uso finaliza
Postcondición	Se han actualizado los datos de un agente conversacional en el sistema
Excepciones	3a. El sistema no encuentra el curso y el caso de uso finaliza 4a. El sistema no encuentra el agente conversacional y el caso de uso finaliza 5a. El sistema comprueba que el profesor no pertenece al curso y no tiene permisos de administración. El caso de uso finaliza 6a. El sistema comprueba que existe un agente conversacional con el mismo identificador dentro del curso y el caso de uso finaliza 7a. El sistema comprueba que los datos del agente conversacional no son validos y el caso de uso finaliza

Tabla 3.13: Caso de uso: Actualización de un agente conversacional

Caso de uso	Consulta de la información de un agente conversacional
Identificador	CU13
Actor	Profesor
Precondición	El profesor ha iniciado sesión en el sistema
Descripción	El profesor consulta la información de un agente conversacional dentro de un curso
Secuencia normal	1. El sistema recibe una petición para consultar la información de un agente conversacional 2. El sistema obtiene los datos del profesor, del curso y del agente conversacional, de la petición 3. El sistema obtiene la información del curso 4. El sistema obtiene la información del agente conversacional 5. El sistema comprueba que el profesor pertenece al curso o tiene permisos de administración 6. El sistema muestra la información del agente conversacional y el caso de uso finaliza
Postcondición	Se ha consultado la información de un agente conversacional en el sistema
Excepciones	3a. El sistema no encuentra el curso y el caso de uso finaliza 4a. El sistema no encuentra el agente conversacional y el caso de uso finaliza 5a. El sistema comprueba que el profesor no pertenece al curso y no tiene permisos de administración. El caso de uso finaliza

Tabla 3.14: Caso de uso: Consulta de la información de un agente conversacional

Caso de uso	Borrado de un agente conversacional
Identificador	CU14
Actor	Profesor
Precondición	El profesor ha iniciado sesión en el sistema
Descripción	El profesor borra un agente conversacional de un curso
Secuencia normal	1. El sistema recibe una petición para borrar un agente conversacional 2. El sistema obtiene los datos del profesor, del curso y del agente conversacional, de la petición 3. El sistema obtiene la información del curso 4. El sistema obtiene la información del agente conversacional 5. El sistema comprueba que el profesor es el creador del agente conversacional, el creador del curso o tiene permisos de administración 6. El sistema registra el borrado del agente conversacional y el caso de uso finaliza
Postcondición	Se ha consultado la información de un agente conversacional en el sistema
Excepciones	3a. El sistema no encuentra el curso y el caso de uso finaliza 4a. El sistema no encuentra el agente conversacional y el caso de uso finaliza 5a. El sistema comprueba que el profesor no es ni el creador del agente conversacional, ni el creador del curso y no tiene permisos de administración. El caso de uso finaliza

Tabla 3.15: Caso de uso: Borrado de un agente conversacional

Caso de uso	Activación de un agente conversacional
Identificador	CU15
Actor	Profesor
Precondición	El profesor ha iniciado sesión en el sistema
Descripción	El profesor activa un agente conversacional en un curso
Secuencia normal	1. El sistema recibe una petición para activar un agente conversacional 2. El sistema obtiene los datos del profesor, del curso y del agente conversacional, de la petición 3. El sistema obtiene la información del curso 4. El sistema obtiene la información del agente conversacional 5. El sistema comprueba que el profesor es el creador del agente conversacional, el creador del curso o tiene permisos de administración 6. El sistema comprueba que el agente conversacional no está activado 7. El sistema registra la actualización de estado del agente conversacional y el caso de uso finaliza
Postcondición	Se ha activado un agente conversacional en el sistema
Excepciones	3a. El sistema no encuentra el curso y el caso de uso finaliza 4a. El sistema no encuentra el agente conversacional y el caso de uso finaliza 5a. El sistema comprueba que el profesor no es ni el creador del agente conversacional, ni el creador del curso y no tiene permisos de administración. El caso de uso finaliza 6a. El sistema comprueba que el agente conversacional está activado y el caso de uso finaliza

Tabla 3.16: Caso de uso: Activación de un agente conversacional

Caso de uso	Desactivación de un agente conversacional
Identificador	CU16
Actor	Profesor
Precondición	El profesor ha iniciado sesión en el sistema
Descripción	El profesor desactiva un agente conversacional en un curso
Secuencia normal	1. El sistema recibe una petición para desactivar un agente conversacional 2. El sistema obtiene los datos del profesor, del curso y del agente conversacional, de la petición 3. El sistema obtiene la información del curso 4. El sistema obtiene la información del agente conversacional 5. El sistema comprueba que el profesor es el creador del agente conversacional, el creador del curso o tiene permisos de administración 6. El sistema comprueba que el agente conversacional está activado 7. El sistema registra la actualización de estado del agente conversacional y el caso de uso finaliza
Postcondición	Se ha activado un agente conversacional en el sistema
Excepciones	3a. El sistema no encuentra el curso y el caso de uso finaliza 4a. El sistema no encuentra el agente conversacional y el caso de uso finaliza 5a. El sistema comprueba que el profesor no es ni el creador del agente conversacional, ni el creador del curso y no tiene permisos de administración. El caso de uso finaliza 6a. El sistema comprueba que el agente conversacional no está desactivado y el caso de uso finaliza

Tabla 3.17: Caso de uso: Desactivación de un agente conversacional

Caso de uso	Unión de un estudiante a una sala de chat con un agente conversacional
Identificador	CU17
Actor	Estudiante
Precondición	El estudiante se encuentra en el grupo de la aplicación de chats asociado a un curso
Descripción	El estudiante se une a una sala de chat con un agente conversacional
Secuencia normal	1. El sistema detecta un mensaje para unirse a una sala de chat con un agente conversacional 2. El sistema obtiene los datos del curso, del mensaje 3. El sistema obtiene la información del curso 4. El sistema comprueba que el mensaje está correctamente formado 5. El sistema obtiene la información del estudiante 6. El sistema comprueba que no hay ningún estudiante esperando en una sala de chat existente 7. El sistema crea la sala de chat 8. El sistema añade al estudiante a dicha sala de chat y el caso de uso finaliza
Postcondición	El estudiante se ha unido a una sala de chat con un agente conversacional
Excepciones	3a. El sistema no encuentra el curso y el caso de uso finaliza 4a. El sistema comprueba que el mensaje no está correctamente formado y el caso de uso finaliza 5a. El sistema no encuentra el estudiante y lo registra en el sistema 5b. El caso de uso continua en el paso 6 6a. El sistema comprueba que hay algún estudiante esperando en una sala de chat y el caso de uso continua en el paso 8

Tabla 3.18: Caso de uso: Unión de un estudiante a una sala de chat con un agente conversacional

Caso de uso	Intervención de un estudiante en una sala de chat
Identificador	CU18
Actor	Estudiante
Precondición	El estudiante se ha unido a una sala de chats con un agente conversacional
Descripción	El estudiante realiza una intervención en una sala de chat con un agente conversacional
Secuencia normal	1. El sistema detecta un mensaje de un estudiante en una sala de chat 2. El sistema obtiene los datos de la sala de chat, del mensaje 3. El sistema obtiene la información de la sala de chat 4. El sistema comprueba que el mensaje no es una petición para enviar el resultado 5. El sistema comprueba que el mensaje no es una petición para confirmar el envío del resultado 6. El sistema comprueba que han pasado los tiempos de espera 7. El sistema detecta que se ha disparado algún patrón y el agente conversacional interviene 8. El sistema registra el tipo de evento que ha tenido lugar y el caso de uso finaliza
Postcondición	El estudiante ha realizado una intervención en una sala de chat
Excepciones	3a. El sistema no encuentra la sala de chat y el caso de uso finaliza 4a. El sistema comprueba que el mensaje es una petición para enviar el resultado 4b. El sistema almacena el resultado para que este sea confirmado posteriormente y el caso de uso continua en el paso 8 5a. El sistema comprueba que el mensaje es una petición para enviar el resultado 5b. El sistema almacena el resultado final continua en el paso 8 6a. El sistema comprueba que no han pasado los tiempos de espera y el caso de uso continua en el paso 8

Tabla 3.19: Caso de uso: Intervención de un estudiante en una sala de chat

Caso de uso	Salida de un estudiante de una sala de chat
Identificador	CU19
Actor	Estudiante
Precondición	El estudiante se ha unido a una sala de chats con un agente conversacional
Descripción	El estudiante sale de una sala de chat
Secuencia normal	1. El sistema detecta la salida de un estudiante de una sala de chat 2. El sistema obtiene los datos sala de chat 3. El sistema obtiene la información de la sala de chat 4. El sistema comprueba que la sala de chat está vacía 5. El sistema cierra la sala de chat 6. El sistema actualiza el estado de la sala de chat y el caso de uso finaliza
Postcondición	El estudiante ha salido de una sala de chat
Excepciones	3a. El sistema no encuentra la sala de chat y el caso de uso finaliza 4a. El sistema comprueba que sala de chat no está vacía 4b. El sistema cierra y actualiza el estado de la sala de chat 4c. El sistema crea una nueva sala de chat, añade a los estudiantes que estaban esperando y el caso de uso finaliza

Tabla 3.20: Caso de uso: Salida de un estudiante de una sala de chat

Caso de uso	Consulta de información de un estudiante
Identificador	CU20
Actor	Profesor con permisos de administración
Precondición	El profesor ha iniciado sesión en el sistema
Descripción	El profesor con permisos de administración consulta la información de un estudiante
Secuencia normal	1. El sistema recibe una petición para consultar la información de un estudiante 2. El sistema obtiene los datos del estudiante a consultar, de la petición 3. El sistema obtiene la información del estudiante 4. El sistema muestra la información del estudiante y el caso de uso finaliza
Postcondición	Se ha consultado la información de un estudiante
Excepciones	3a. El sistema no encuentra el estudiante solicitado y el caso de uso finaliza

Tabla 3.21: Caso de uso: Consulta de la información de un estudiante

Caso de uso	Consulta de información de una sala de chat
Identificador	CU21
Actor	Profesor
Precondición	El profesor ha iniciado sesión en el sistema
Descripción	El profesor consulta la información de un estudiante
Secuencia normal	1. El sistema recibe una petición para consultar la información de una sala de chat 2. El sistema obtiene los datos de la sala de chat, de la petición 3. El sistema obtiene la información de la sala de chat 4. El sistema muestra la información de la sala de chat y el caso de uso finaliza
Postcondición	Se ha consultado la información un curso
Excepciones	3a. El sistema no encuentra la sala de chat y el caso de uso finaliza

Tabla 3.22: Caso de uso: Consulta de la información de una sala de chat

3.5.1. Casos de uso no contemplados

Existen varios casos de uso que no se han contemplado. Estos han sido:

- Borrado de un curso
- Borrado de una sala de chat
- Actualización de una sala de chat
- Borrado de estudiantes
- Actualización de estudiantes

Dado que uno de los principales objetivos de este proyecto es la investigación del comportamiento de los agentes conversacionales, se ha descartado la posibilidad de borrar los cursos, las salas de chats y los estudiantes. Con esto se consigue mantener toda la información para su estudio futuro. Por el mismo motivo, se ha descartado la posibilidad de actualizar estudiantes. La actualización de las salas de chats se ha descartado porque podría generar dudas en los estudiantes si se actualiza la información de la sala mientras están conversando.

3.6. Modelo de dominio

El modelo de dominio es una representación conceptual de las clases en el mundo real. En él se muestran las clases conceptuales ligadas entre sí, mostrando las relaciones existentes. En la Figura 3.2 se puede ver el modelo de dominio para este proyecto.

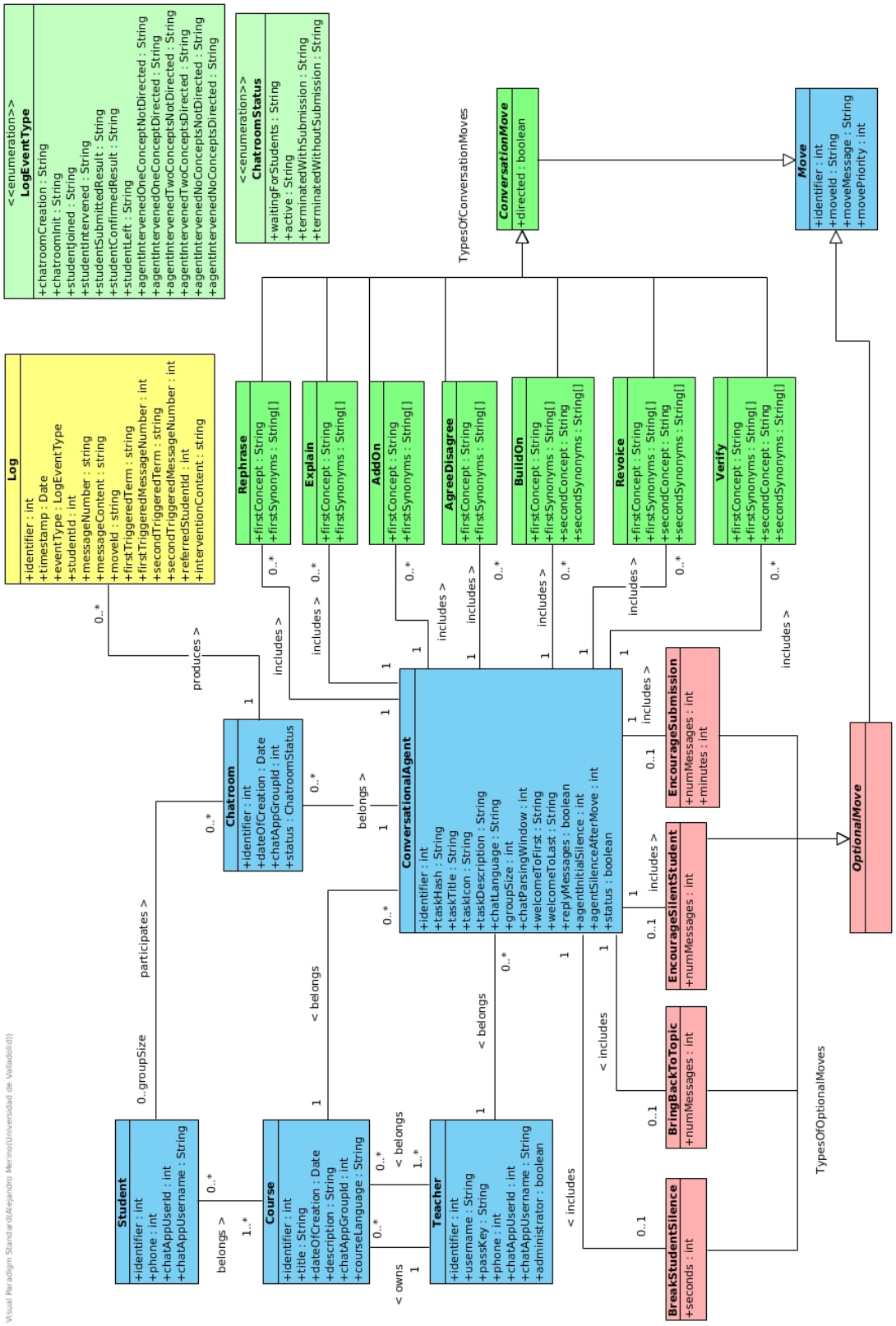


Figura 3.2: Modelo de dominio

En primer lugar, los cursos (Course) están relacionados con los profesores (Teacher) de dos posibles maneras. Un curso solo está asociado a un profesor, creador del mismo, pero un profesor puede ser creador de ninguno o varios. Un curso puede tener varios profesores participantes y un profesor puede participar en varios cursos.

En segundo lugar, un curso podrá tener varios estudiantes (Student) asociados y, a su vez, un estudiante podrá pertenecer a varios cursos.

En cuanto a los agentes conversacionales (ConversationalAgent), estos estarán relacionados con varias entidades:

1. Un agente tendrá un único profesor creador y un profesor podrá crear varios agentes.
2. Un curso podrá estar asociado con varios agentes pero un agente solo podrá pertenecer a un curso.
3. Existen cuatro especificaciones de patrones opcionales (OptionalMove): BreakStudentSilence, BringBackToTopic, EncourageSilentStudent y EncourageSubmission. Cada uno estará únicamente asociado a un agente y este podrá estar, o no, relacionado con los mismos.
4. Las siete especificaciones de patrones basados en conversación (ConversationMove) son: Rephrase, Explain, AddOn, AgreeDisagree, BuildOn, Revoice, Verify. Al igual que los opcionales, cada patrón estará relacionado solo con un agente. Este último, podrá estar asociado con varios o con ningún patrón.
5. Una sala de chat (Chatroom) estará relacionada con un agente conversacional y este podrá existir en varias salas o en ninguna.

En una sala de chat participarán un número de estudiantes definido en los atributos del agente conversacional. Además, los alumnos podrán participar en más de una sala de chat pero no pueden estar en dos salas activas o esperando a estudiantes, con el mismo agente conversacional.

Finalmente, una sala de chat tendrá asociado varios registros (Log), mientras que cada registro pertenecerá a única sala de chat.

Capítulo 4

Diseño

En este capítulo se explican las decisiones de diseño, la arquitectura software del sistema, junto con cada uno de sus componentes y las relaciones entre ellos. Este diseño se realiza siguiendo las restricciones y requisitos del análisis previo.

4.1. Decisiones de diseño

A continuación se enumeran, a modo de introducción, las decisiones de diseño sobre la arquitectura del sistema, los subsistemas utilizados y las tecnologías escogidas para dichos elementos :

- La arquitectura será Cliente-Servidor.
- El servidor se desarrollará utilizando Python.
- El servidor utilizado será Flask [1].
- El servidor utilizará una arquitectura en capas.
- Se utilizará el patrón Modelo-Controlador.
- El sistema gestor de base de datos será MySQL.
- El acceso a base de datos se realizará con la librería MySQL-Connector [2].
- La aplicación de chats será Telegram.
- La librería necesaria para integrar Telegram será Telethon[3].

En las siguientes secciones se detallan todas estas decisiones de diseño.

[1]Flask - Framework de desarrollo web: https://flask.palletsprojects.com - último acceso: Mayo 2020

[2]MySQL-Connector - Librería para trabajar con MySQL: https://dev.mysql.com/doc/connector-python - último acceso: Mayo 2020

[3]Telethon - Librería para trabajar con Telegram: https://docs.telethon.dev - último acceso: Junio 2020

4.2. Arquitectura del sistema

4.2.1. Arquitectura del servidor

En la Figura 4.1 se puede observar la arquitectura en capas del servidor. La capa *rest api server* se encarga de recibir y redireccionar las peticiones que llegan al servidor hacia otras capas, con el objetivo de que sean resueltas. En la capa *model* se encuentra la definición de las diferentes entidades que aparecen en el modelo de dominio.

La capa *controllers* es la encargada de resolver las peticiones, interactuando con la capa de acceso de base de datos, *database*. Además, en ella se encuentran los módulos encargados de realizar las operaciones en Telegram.

Common es la capa en la que se encuentran tanto los ficheros de idiomas, como las variables globales comunes para todo el servidor. En el paquete *logs* se registran todos los flujos de las capas *rest api server*, *controllers* y *database*. Finalmente, en el paquete *tokens* se encuentran todas las claves de seguridad necesarias para acceder a base de datos, al cliente de Telegram y las configuraciones de la Rest API.

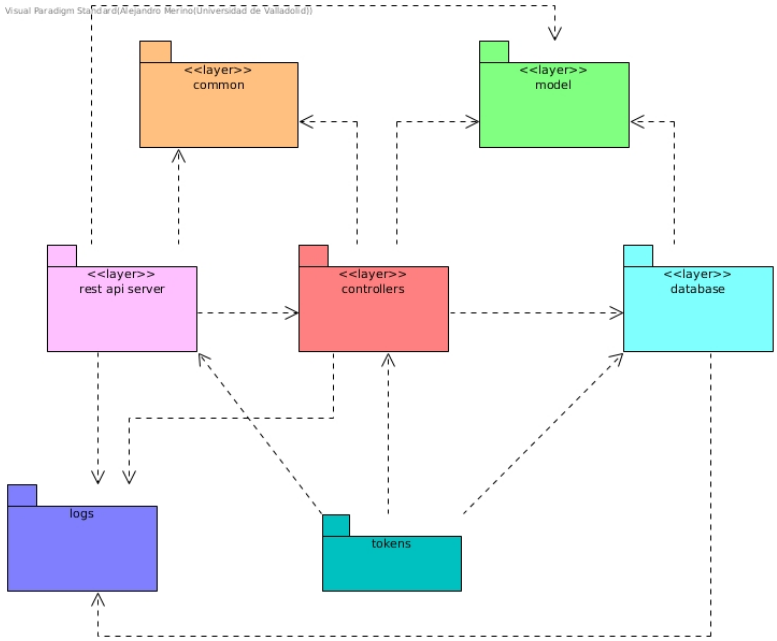


Figura 4.1: Arquitectura Servidor

4.2.2. Diagrama de despliegue

En la Figura 4.2 se muestra el diagrama de despliegue. En él se puede observar que la aplicación Rest API estará desplegada en un servidor Flask. La comunicación entre el sistema gestor de base de datos, MySQL, y la Rest API se realizará por medio de la librería *MySQL-Connector*.

El servidor también se conectará a los servidores de Telegram gracias a la aplicación que integra Telethon. Esta se comunicará mediante el protocolo de mensajería *MTProto* [9].

Los clientes pueden ser de dos tipos:

1. Clientes para profesores: en este proyecto se corresponden con aplicaciones capaces de realizar peticiones HTTP a la Rest API. En un futuro, cuando se integre un frontend en la aplicación, este cliente se corresponderá con un navegador web.

2. Clientes para estudiantes: los estudiantes interactúan con el servidor a través de Telegram. Debido a esto, cualquier cliente de Telegram es valido. Estos pueden ser de escritorio, web o móvil. Los clientes también utilizan el protocolo de mensajería *MTProto*.

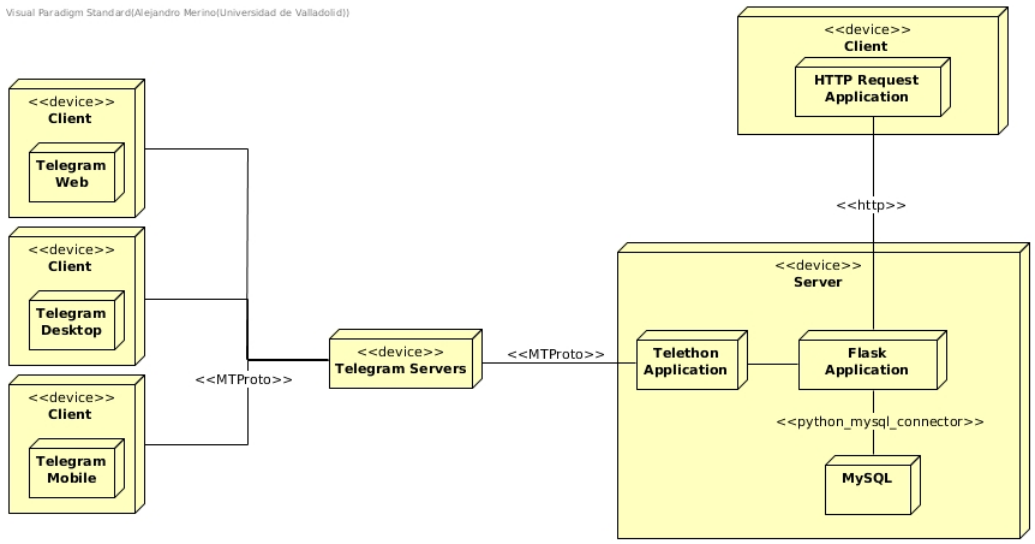


Figura 4.2: Diagrama de Despliegue

4.2.3. Arquitectura Cliente

Dado que este proyecto ha consistido en el backend de un sistema web, no se ha incluido la arquitectura del cliente. Si se quisiera integrar un frontend en el sistema, habría que incluir la arquitectura del mismo en esta sección. En definitiva, consistiría en un navegador web desde el que se accede al sistema.

4.3. Aplicación de chats

En primer lugar, se realizó un estudio sobre la posible aplicación a usar. Se necesitaba una plataforma de mensajería que permitiera tener un gran control sobre las conversaciones entre los usuarios y sobre sus funcionalidades. Además, esta plataforma debería ser *Open Source Software* y por lo tanto se planteó el uso de dos opciones:

- RocketChat[4]

- Telegram[5]

[4]Rocket Chat - Aplicación de mensajería: https://rocket.chat - último acceso: Julio 2020
[5]Telegram - Aplicación de mensajería: https://web.telegram.org - último acceso: Julio 2020

En segundo lugar, se procedió a revisar las API proporcionadas por cada una. Durante este proceso, se comprobó que ambas eran válidas y, por lo tanto, la elección se basó en su popularidad y experiencia. Dado que Telegram es bastante más popular se optó por incorporarla al proyecto.

Asimismo, con el objetivo de encontrar un medio para controlar esta aplicación de chats, se investigaron diferentes librerías de código abierto. En Telegram existe una librería que permite el desarrollo de bots, lo cual se asimiló con el concepto de agente conversacional. A pesar de ello, estos bots no tienen las mismas funcionalidades que un usuario real, por ejemplo, crear grupos, añadir usuarios a los mismos, etc. Esta necesidad se vio resuelta con una librería desarrollada en Python, llamada Telethon, que permitía el completo control sobre las acciones de un usuario en Telegram.

Telethon permite obtener un cliente de Telegram asociado a un usuario. Sin embargo, es necesario obtener un identificador y un hash de la API para garantizar la autenticidad del mismo. Además, la primera vez que es utilizado se solicitará una autenticación en dos pasos. Una vez se ha realizado, la información se almacenará en un fichero de sesión para posteriores ejecuciones.

4.4. Base de datos

En la Figura 4.3 se puede ver el diagrama entidad relación, obtenido a partir del modelo de dominio. Para su obtención se han seguido las reglas de transformación del diseño lógico de bases de datos [10].

En cuanto a la información disponible en este esquema, cada entidad del modelo de dominio tiene su tabla correspondiente. Las relaciones varios a varios han sido diseñadas de tal modo que se genera una tabla intermedia, en la que se encuentran los identificadores de ambas entidades. Estas tablas son: *studentBelongsToCourse*, *studentParticipatesInChatroom* y *teacherBelongsToCourse*.

El resto de relaciones son uno a uno o uno a varios. Por lo tanto, se absorbe hacia una tabla el identificador de la otra como clave foránea. Esto se da entre las siguientes relaciones:

- Course - Teacher (Creador)
- Conversational Agent - Course
- Conversational Agent - Teacher
- Chatroom - Conversational Agent
- Optional Move - Conversational Agent
- Conversation Move - Conversational Agent
- Chatroom Log - Chatroom

En el esquema se puede observar las diferentes características de cada columna:

- Llave amarilla & Llave roja: Clave primaria

- Rombo blanco: puede ser null
- Rombo azul: not null
- Rombo rojo: Clave foránea

Los tamaños máximos de cada columna han sido refinados a lo largo del diseño y desarrollo. En especial, la longitud de los mensajes que son enviados a través de Telegram no debe ser muy amplia, de lo contrario, los estudiantes tendrán dificultades para comprender el contenido de los mismos. En este proyecto, si se detectan tamaños superiores a los establecidos, se truncarán los datos introducidos por los profesores. Esta decisión se ha tomado con el objetivo de evitar errores a la hora de probar la aplicación. En un futuro, la entrada de datos será gestionada desde el cliente web evitando que llegue información errónea al *backend* del sistema web.

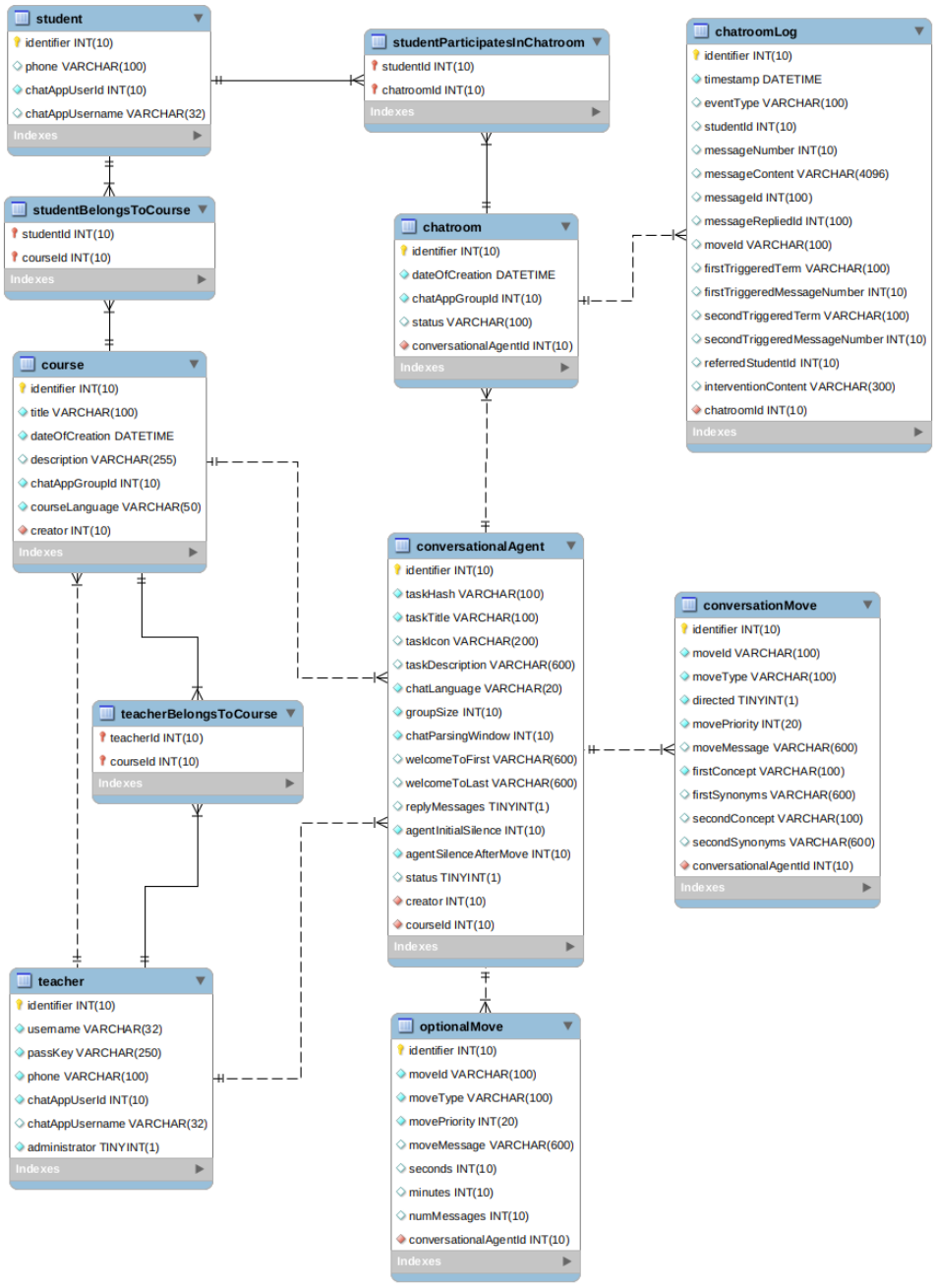


Figura 4.3: Diagrama Entidad Relación

Por último, el sistema gestor de base de datos escogido ha sido MySQL. Esto se debe a su popularidad, a su fiabilidad y a la experiencia previa que se tenía en él.

4.5. API Rest

Una API Rest [11] es cualquier interfaz lógica que utilice HTTP para obtener datos o desencadenar operaciones. Estos datos pueden ser de diferentes formatos, como JSON o XML. Las razones para elegir esta tecnología son su popularidad y la facilidad para integrar el futuro *frontend* en el sistema. Este solo necesitará realizar las correspondientes peticiones HTTP y, por lo tanto, ambas partes del sistema web estarán desacopladas. El formato elegido para el tratamiento de los datos es JSON, debido a la amplia experiencia que se tiene en él.

Conviene que la realización de las operaciones REST tuvieran algún tipo de seguridad. En nuestro caso, se ha decidido implementar un sistema de seguridad basado en JWT (JSON Web Token) [12]. Se trata de un estándar basado en JSON (RFC 7519) que permite la obtención de tokens de acceso para propagar identidades y permisos. De este modo, un profesor realizará login con su nombre de usuario y contraseña, obteniendo así un token secreto. Con esta clave podrá realizar las operaciones deseadas en la Rest API. En la Figura 4.4 se puede observar un esquema de este sistema.

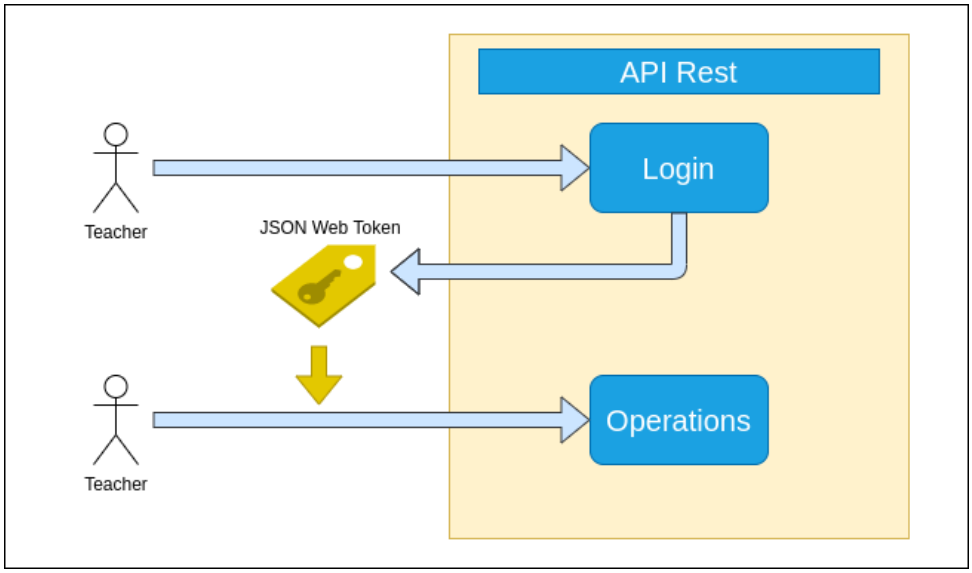


Figura 4.4: JSON Web Token

La Rest API ha sido diseñada de tal modo que posee diferentes puntos de acceso (endpoints) para cada entidad del sistema. De hecho, los diferentes casos de uso son realizados a través de los distintos puntos de acceso como */login*, */teachers*, */courses*, etc.

En los puntos de accesos donde tiene lugar la entrada de datos, se realiza una validación de los mismos. Estos deben cumplir una estructura definida a través de *JSON Schema* [6]. JSON Schema es un vocabulario que permite la anotación y validación de

[6]JSON Schema - Vocabulario para la validación JSON: https://json-schema.org - último acceso: Junio 2020

documentos JSON.

Además, a cada *endpoint* se podrá acceder a través de los métodos HTTP: POST, PUT, GET, DELETE. En las siguientes secciones se muestran cada uno de los puntos de acceso, junto con su método HTTP y su JSON Schema.

4.5.1. Punto de acceso para iniciar sesión

Punto de acceso	/login
Método	GET, POST
Descripción	Inicio de sesión de un profesor.
Autorización	Basic Auth: ▪ Username ▪ Password

Tabla 4.1: Login

4.5.2. Puntos de acceso para gestionar profesores

Los esquemas JSON necesarios para la gestión de profesores son los siguientes:

Código 4.1: JSON Schema creación y actualización de profesores

```
{
    "type": "object",
    "properties": {
        "username": { "type": "string" },
        "password": { "type": "string" },
        "phone": { "type": "string" },
        "chatAppUsername": { "type": "string" }
    },
    "required": ["username", "password",
                 "phone", "chatAppUsername"],
    "additionalProperties": False
}
```

Punto de acceso	/teachers
Método	POST
Descripción	Creación de un profesor. Operación realizada por otro profesor con permisos de administración.
Autorización	Bearer Token
Cuerpo (JSON Schema)	Código 4.1

Tabla 4.2: Creación de un profesor

Punto de acceso	/teachers
Método	GET
Descripción	Obtención de la información de todos los profesores. Operación realizada por un profesor con permisos de administración.
Autorización	Bearer Token

Tabla 4.3: Consulta de información de todos los profesores

Punto de acceso	/teachers/<string:data>
Método	GET
Descripción	Obtención de la información de un profesor específico, identificado por su nombre de usuario, teléfono o identificador. Operación realizada por un profesor con permisos de administración.
Autorización	Bearer Token

Tabla 4.4: Consulta de información un profesor

Punto de acceso	/teachers/<int:identificador>
Método	PUT
Descripción	Actualización de un profesor específico a través de su identificador. Operación realizada por un profesor con permisos de administración.
Autorización	Bearer Token
Cuerpo (JSON Schema)	Código 4.1

Tabla 4.5: Actualización de un profesor

Punto de acceso	/teachers/<int:identifier>
Método	DELETE
Descripción	Borrado de un profesor a través de su identificador. Operación realizada por un profesor con permisos de administración.
Autorización	Bearer Token

Tabla 4.6: Borrado un profesor

4.5.3. Puntos de acceso para gestionar cursos

A continuación aparecen los esquemas correspondientes a la creación de cursos, la actualización de cursos, la adición de profesores a los mismos y el borrado de profesores de ellos:

Código 4.2: JSON Schema creación de un curso

```
{
    "type": "object",
    "properties": {
        "title": { "type": "string" },
        "description": { "type": "string" },
        "courseLanguage": { "type": "string"},
        "teachers": {
            "type": "array",
            "items": {
                "type": "string"
            }
        }
    },
    "required": ["title","description","courseLanguage"],
    "additionalProperties": False
}
```

Código 4.3: JSON Schema actualización de un curso

```
{
    "type": "object",
    "properties": {
        "title": { "type": "string" },
        "description": { "type": "string" },
        "courseLanguage": { "type": "string"}
    },
    "required": ["title","description"
                ,"courseLanguage"],
    "additionalProperties": False
}
```

Código 4.4: JSON Schema adición y borrado de profesores en un curso

```
{
    "type": "object",
    "properties": {
        "teachers": {
            "type": "array",
            "minItems": 1,
            "items": {
                "type": "string"
            }
        }
    },
    "required": ["teachers"],
```

```
    "additionalProperties": False
}
```

Punto de acceso	/courses
Método	POST
Descripción	Creación de un curso.
Autorización	Bearer Token
Cuerpo (JSON Schema)	Código 4.2

Tabla 4.7: Creación de un curso

Punto de acceso	/courses
Método	GET
Descripción	Obtención de los cursos. Sí el profesor tiene permisos de administración, se muestran todos los cursos del sistema. En cambio, si no los tiene, se muestran los cursos creados y los cursos a los que pertenece un profesor.
Autorización	Bearer Token

Tabla 4.8: Consulta de información de cursos

Punto de acceso	/courses/<int:identifier>
Método	GET
Descripción	Obtención de la información de un curso.
Autorización	Bearer Token

Tabla 4.9: Consulta de información de un curso

Punto de acceso	/courses/<int:identifier>
Método	PUT
Descripción	Actualización de la información de un curso, título y descripción.
Autorización	Bearer Token
Cuerpo (JSON Schema)	Código 4.3

Tabla 4.10: Actualización de un curso

Punto de acceso	/courses/<int:identifier>/teachers
Método	POST
Descripción	Adición de profesores a un curso a través de sus nombres de usuario. Realizado por el profesor creador del curso.
Autorización	Bearer Token
Cuerpo (JSON Schema)	Código 4.4

Tabla 4.11: Adición de profesores a un curso

Punto de acceso	/courses/<int:identifier>/teachers
Método	DELETE
Descripción	Borrado de profesores de un curso a través de sus nombres de usuario. Realizado por el profesor creador del curso.
Autorización	Bearer Token
Cuerpo (JSON Schema)	Código 4.4

Tabla 4.12: Borrado de profesores de un curso

4.5.4. Puntos de acceso para gestionar agentes conversacionales

La definición del cuerpo de las peticiones para la creación y actualización de agentes conversacionales se muestra a continuación:

Código 4.5: JSON Schema Agentes Conversacionales

```
{
    "type": "object",
    "properties": {
        "taskTitle" : { "type": "string" },
        "taskHash" : { "type": "string" },
        "taskIcon" : { "type": "string" },
        "taskDescription" : { "type": "string" },
        "chatLanguage" : { "type": "string" },
        "groupSize" : { "type": "number" },
        "chatParsingWindow" : { "type": "number" },
        "agentInitialSilence" : { "type": "number" },
        "agentSilenceAfterMove" : { "type": "number" },
        "welcomeToFirst" : { "type": "string" },
        "welcomeToLast" : { "type": "string" },
        "replyMessages" : { "type": "boolean" },
        "optionalMoves" : {
            "type": "array",
```

```
            "items": {
                "type": "object",
                "properties": {
                    "moveType" : { "type": "string" },
                    "moveId" : { "type": "string" },
                    "message" : { "type": "string" },
                    "seconds" : { "type": "number" },
                    "minutes": { "type": "number" },
                    "numMessages": { "type": "number" },
                    "priority" : { "type": "number" }
                },
                "required": ["moveType","moveId","message",
                             "priority"],
                "additionalProperties": False
            }
        },
        "conversationMoves": {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "moveType" : { "type": "string" },
                    "moveId" : { "type": "string" },
                    "message" : { "type": "string" },
                    "firstConcept" : { "type": "string" },
                    "firstSynonyms" : {
                        "type": "array",
                        "items": {
                            "type": "string"
                        }
                    },
                    "secondConcept" : { "type": "string" },
                    "secondSynonyms" : {
                        "type": "array",
                        "items": {
                            "type": "string"
                        }
                    },
                    "directed" : {"type": "boolean"},
                    "priority" : { "type": "number" }
                },
                "required": ["moveType","moveId","message",
                        "firstConcept","priority","directed"],
                "additionalProperties": False
            }
        }
    },
    "required": ["taskTitle","taskHash","taskIcon","taskDescription",
                "chatLanguage","groupSize","chatParsingWindow",
```

```
                    "agentInitialSilence","agentSilenceAfterMove",
                    "welcomeToFirst","welcomeToLast","replyMessages",
                    "conversationMoves"],
    "additionalProperties": False
}
```

Punto de acceso	/courses/<int:identifier>/agents
Método	POST
Descripción	Creación de un agente conversacional asociado a un curso. Realizado por unos de los profesores del curso.
Autorización	Bearer Token
Cuerpo (JSON Schema)	Código 4.5

Tabla 4.13: Creación de un agente conversacional

Punto de acceso	/courses/<int:identifier>/agents
Método	GET
Descripción	Obtención de todos los agentes conversacionales de un curso. Realizado por unos de los profesores del curso.
Autorización	Bearer Token

Tabla 4.14: Consulta de información de los agentes conversacionales en un curso

Punto de acceso	/courses/<int:identifier>/agents/<string:taskHash>
Método	GET
Descripción	Obtención la información de un agente conversacional a través de su taskHash. Realizado por unos de los profesores del curso.
Autorización	Bearer Token

Tabla 4.15: Consulta de información de un agente conversacional

Punto de acceso	/courses/<int:identifier>/agents/<string:taskHash>
Método	PUT
Descripción	Actualización de un agente conversacional a través de su taskHash. Realizado por su creador o el creador del curso.
Autorización	Bearer Token
Cuerpo	Código 4.5

Tabla 4.16: Actualización de un agente conversacional

Punto de acceso	/courses/<int:identifier>/agents/<string:taskHash>
Método	DELETE
Descripción	Borrado de un agente conversacional a través de su taskHash. Realizado por su creador o el creador del curso.
Autorización	Bearer Token

Tabla 4.17: Borrado de un agente conversacional

Punto de acceso	/courses/<int:identifier>/agents/<string:taskHash>/enable
Método	PUT
Descripción	Activación de un agente conversacional a través de su taskHash. Realizado por su creador o el creador del curso.
Autorización	Bearer Token

Tabla 4.18: Activación de un agente conversacional

Punto de acceso	/courses/<int:identifier>/agents/<string:taskHash>/disable
Método	PUT
Descripción	Desactivación de un agente conversacional a través de su taskHash. Realizado por su creador o el creador del curso.
Autorización	Bearer Token

Tabla 4.19: Desactivación de un agente conversacional

4.5.5. Puntos de acceso para consultar salas de chat

Punto de acceso	/courses/<int:identifier>/chatrooms
Método	GET
Descripción	Obtención de la información de todas las salas de chat en un curso. Realizado por unos de los profesores del curso.
Autorización	Bearer Token

Tabla 4.20: Consulta de información de todas las salas de chat en un curso

Punto de acceso	/courses/<int:identifier>/agents/<string:taskHash>/chatrooms
Método	GET
Descripción	Obtención de la información de todas las salas de chat de un agente conversacional. Realizado por unos de los profesores del curso.
Autorización	Bearer Token

Tabla 4.21: Consulta de información de todas las salas de chat de un agente conversacional

Punto de acceso	/courses/<int:identifier>/agents/<string:taskHash>/chatrooms/<int:chatroomId>
Método	GET
Descripción	Obtención de la información de una sala de chat y de los eventos (logs) que han tenido lugar en ella. Realizado por unos de los profesores del curso.
Autorización	Bearer Token

Tabla 4.22: Consulta de información de una sala de chat

4.5.6. Puntos de acceso para consultar estudiantes

Punto de acceso	/students
Método	GET
Descripción	Obtención de la información de todos los estudiantes en el sistema. Realizado por un profesor con permisos de administración.
Autorización	Bearer Token

Tabla 4.23: Consulta de información de todos los estudiantes

Punto de acceso	/students/<string:data>
Método	GET
Descripción	Obtención de la información de un estudiante a través de su nombre de usuarios en la aplicación de chats o su identificador. Realizado por un profesor con permisos de administración.
Autorización	Bearer Token

Tabla 4.24: Consulta de información de un estudiante

4.6. Patrón MC (Modelo Controlador)

El sistema web se desarrolla mediante la utilización del patrón *Modelo Vista Controlador (MVC)*, pero como no se va a implementar el *frontend*, solo se trabajará con modelos y controladores. Los modelos se corresponden con cada entidad del modelo de dominio. Los controladores son los encargados de realizar las tareas de gestión de cada uno de los modelos.

Los controladores se encargan de almacenar la información en la base de datos y de realizar las operaciones correspondientes en Telegram. Se han diseñado cinco controladores, con sus respectivas operaciones, para la gestión de las diferentes entidades:

- Controlador de profesores (manage_teacher): creación, consulta, actualización y eliminación.
- Controlador de cursos (manage_course): creación, consulta, actualización, adición de profesores y borrado de profesores.
- Controlador de agentes conversacionales (manage_conversationalAgent): creación, consulta, actualización, eliminación, activación y desactivación.
- Controlador de salas de chat (manage_chatroom): creación, consulta, actualización de estado, adición de un estudiante, eliminación de un estudiante y registro de un evento.
- Controlador de estudiantes (manage_student): creación y consulta.

A parte de estos controladores, existen otros dos más, encargados de la monitorización los cursos y agentes conversacionales:

- Monitor de cursos (courses_listener): se encarga de resolver las peticiones de los estudiantes para unirse a una sala de chat con un agente conversacional. Se detectan diferentes mensajes en un curso y, los que cumplan la estructura de solicitud son tramitados, mientras que el resto son eliminados. La estructura de estos mensajes de solicitud es configurable a través de los ficheros de idioma.

  También, se encarga de gestionar las salidas de un estudiante de una sala de chat que se encuentre activa o esperando a estudiantes. Si un alumno abandona un *chatroom* mientras que está esperando, pueden tener lugar dos procesos diferentes:

  1. Si no hay nadie más esperando, se desecha la sala de chat, pasando así al estado *TerminatedWithoutSubmission*.

2. Si hay más estudiantes esperando, estos seguirán a la espera de que se complete la sala de chat.

- Monitor de agentes conversacionales (agents_listener): es el encargado de las funcionalidades de los agentes conversacionales. Cuando se inicia una conversación, este hilo gestiona el desencadenamiento de todos los patrones y registra todos los sucesos que tiene lugar en la sala de chats.

4.7. Diseño concurrente

En este proyecto se ha incorporado el uso de hilos para integrar varias partes del sistema junto con el servidor API Rest. Para ello se han usado técnicas de *threading* [7] y *multiprocessing* [8]. El uso de estas tecnologías se ha debido a la necesidad desarrollar varios módulos encargados de monitorizar Telegram. Con esto, se consigue detectar todos los eventos que tienen lugar en la cuenta asociada a esta aplicación de chats.

Dado que se necesita monitorizar varios elementos y mantener una independencia entre los mismos, se ha decido utilizar dos hilos concurrentes. Además, para la inicialización del servidor de colas, que será explicado en la siguiente sección, se debe utilizar otro hilo más. En definitiva, al inicio del servidor, se lanzan tres hilos paralelamente:

1. Hilo asociado a los cursos: encargado de lanzar el controlador de cursos.

2. Hilo asociado a los agentes conversacionales: encargado de lanzar el controlador de agentes conversacionales.

3. Hilo asociado al servidor de colas: encargado de inicializar el servidor de colas.

Cada uno de estos hilos debe utilizar un fichero de sesión diferente con el objetivo de no producir ningún *deadlock*. Esto se debe a que estos ficheros de sesión están continuamente activos y, no pueden ser utilizados para otra operación en un módulo distinto.

En cuanto al *multiprocessing*, se han utilizado subprocesos para integrar el sistema de colas, que será explicado en la siguiente sección en más detalle. Este sistema se ha utilizado para integrar el patrón opcional *BreakStudentSilence*. Cada vez que se recibe una intervención de un estudiante en una sala de chat se lanza un subproceso encargado de gestionar dicho patrón.

4.8. Sistema de colas

El principal motivo por el que se ha incorporado un sistema de colas, Figura 4.5, ha sido la falta de funcionalidades en Telethon. Esta librería permite trabajar con varios tipos de eventos en Telegram, pero no dispone de eventos basados en temporizadores.

[7]Python Threading - Librería estándar: https://docs.python.org/3/library/threading.html - último acceso: Junio 2020

[8]Python Multiprocessing - Librería estándar: https://docs.python.org/3/library/multiprocessing.html - último acceso: Junio 2020

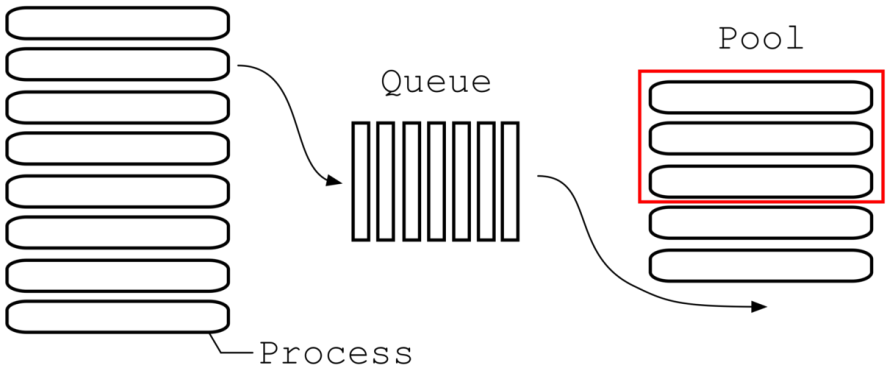


Figura 4.5: Sistema de Colas [13]

El patrón opcional *BreakStudentSilence* es desencadenado una vez que no se han detectado intervenciones de estudiantes durante un periodo de tiempo, configurable en el agente conversacional. Para incorporar esta funcionalidad en el sistema, se ha utilizado un sistema de colas ZMQ (Zero Message Queue) [9]. En la Figura 4.6 se puede observar un pequeño esquema.

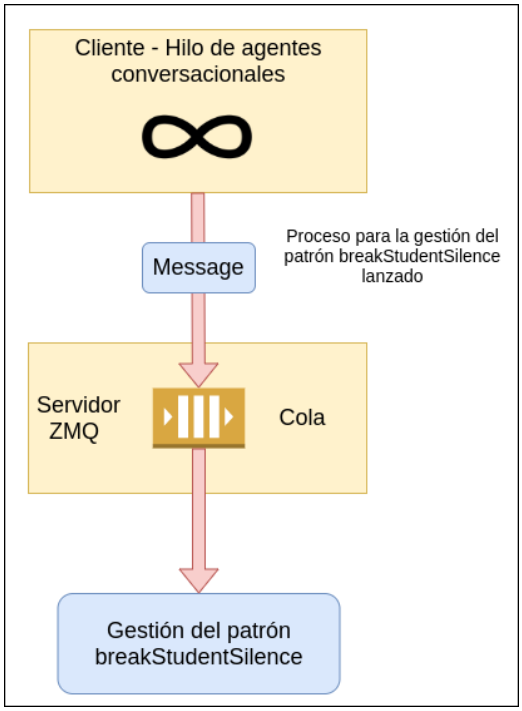


Figura 4.6: Esquema Diseño de Sistema de Colas

Para cada intervención de un estudiante, si el patrón todavía no ha sido desencadenado, se generará un subproceso desde el controlador de agentes conversacionales. Este esperará el tiempo configurado y una vez finalice, encolará un mensaje con la información necesaria. Si durante ese tiempo de espera se detecta una nueva intervención de un alumno, se reiniciará el subproceso. Si llega un mensaje a la cola, este será gestionado

[9]Zero MQ Python - Sistema de colas: https://zeromq.org/languages/python/ - último acceso: Junio 2020

enviando el mensaje configurado en el patrón a la sala de chat correspondiente. Además, se actualizará la información, reflejando que este patrón ya ha sido desencadenado en esa sala de chat.

4.9. Incorporación de múltiples idiomas

El agente conversacional usa ciertos mensajes por defecto en todas las conversaciones, Figura 4.7 y Figura 4.8. Dado que en los cursos y agentes se puede configurar el idioma, es necesario que esos mensajes globales se ajusten al idioma seleccionado en cada caso.

Figura 4.7: Mensaje *No existen agentes conversacional disponibles* en Inglés

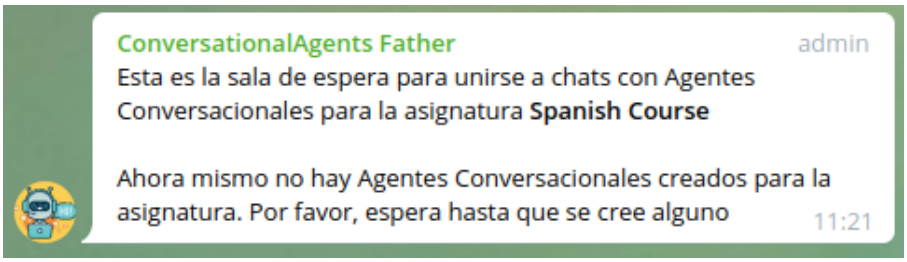


Figura 4.8: Mensaje *No existen agentes conversacional disponibles* en Español

Para conseguir esto, se ha decidido incorporar un fichero de mensajes globales para cada idioma. Estos archivos, en formato JSON, tendrán la misma estructura pero diferente contenido.

Los idiomas serán identificados por los códigos definidos en el ISO 639-1[10]. Los mensajes globales serán cargados dinámicamente cuando sea necesario su uso.

[10]ISO 639-1 - Nomenclatura de idiomas: https://es.wikipedia.org/wiki/ISO_639-1 - último acceso: Junio 2020

4.10. Diagramas de flujo

Un diagrama de flujo [14] representa gráficamente un proceso mediante la utilización de flechas y diferentes figuras geométricas, que indican condiciones, subprocesos, entrada de datos, etc. Dado que la aplicación consiste en el *backend* de un sistema web, se ha escogido los diagramas de flujo como medio para representar el tratamiento de los datos y la ocurrencia de acciones en cada uno de los casos de uso.

De la Figura 4.9 a la Figura 4.28 se muestran todos los diagramas de flujo para cada uno de los casos de uso.

4.10.1. Inicio de sesión

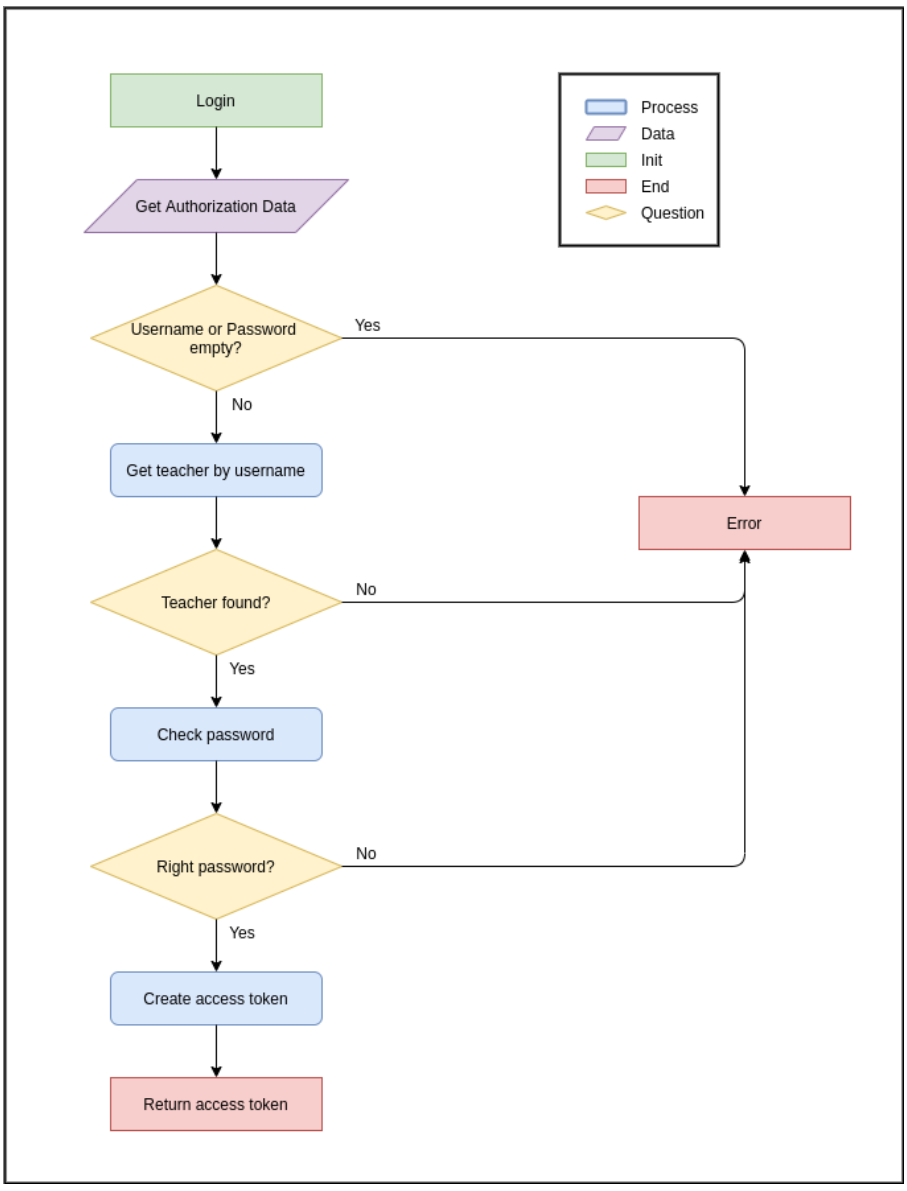


Figura 4.9: Login

4.10.2. Gestión de un profesor

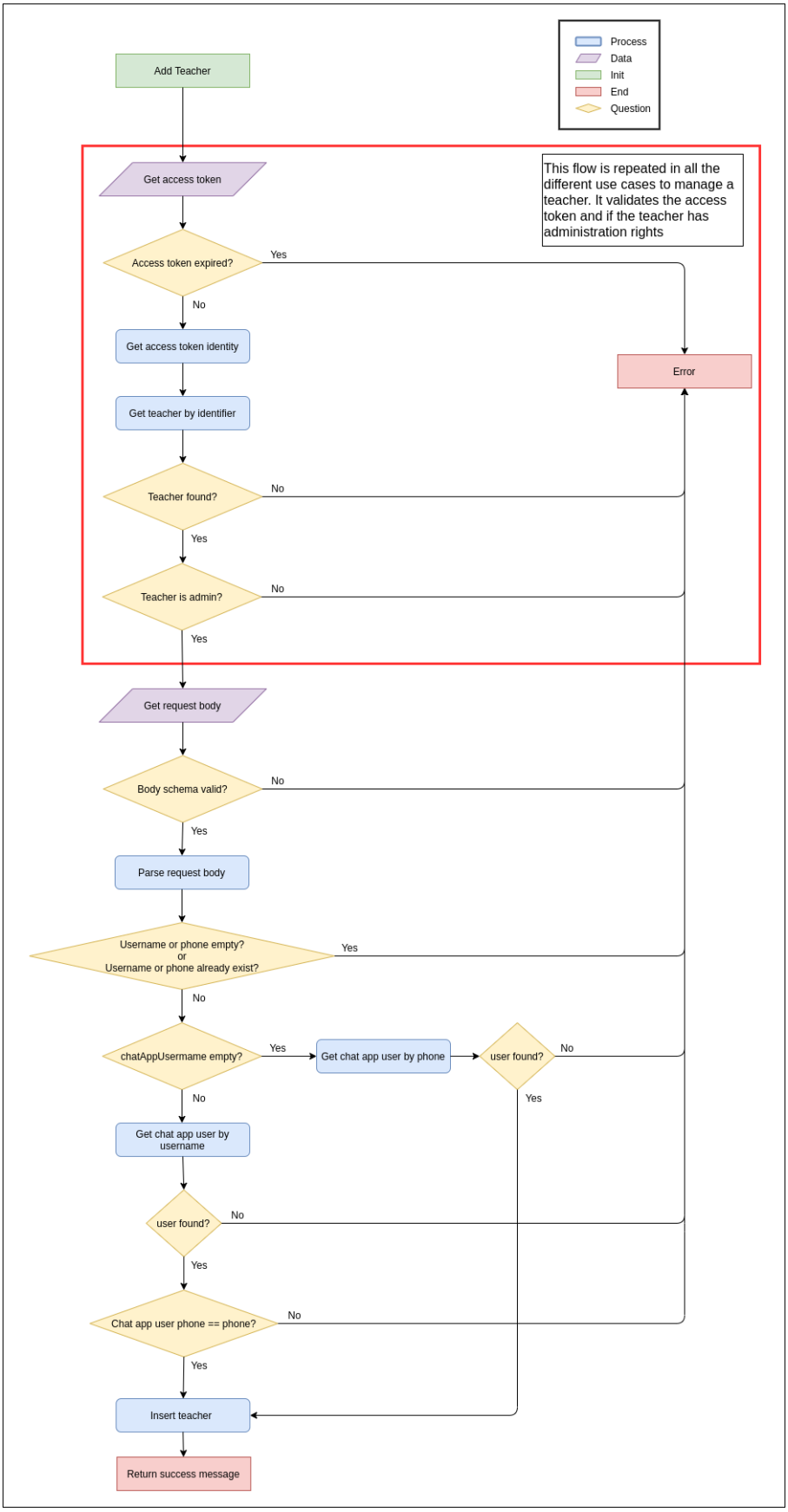


Figura 4.10: Creación de un profesor

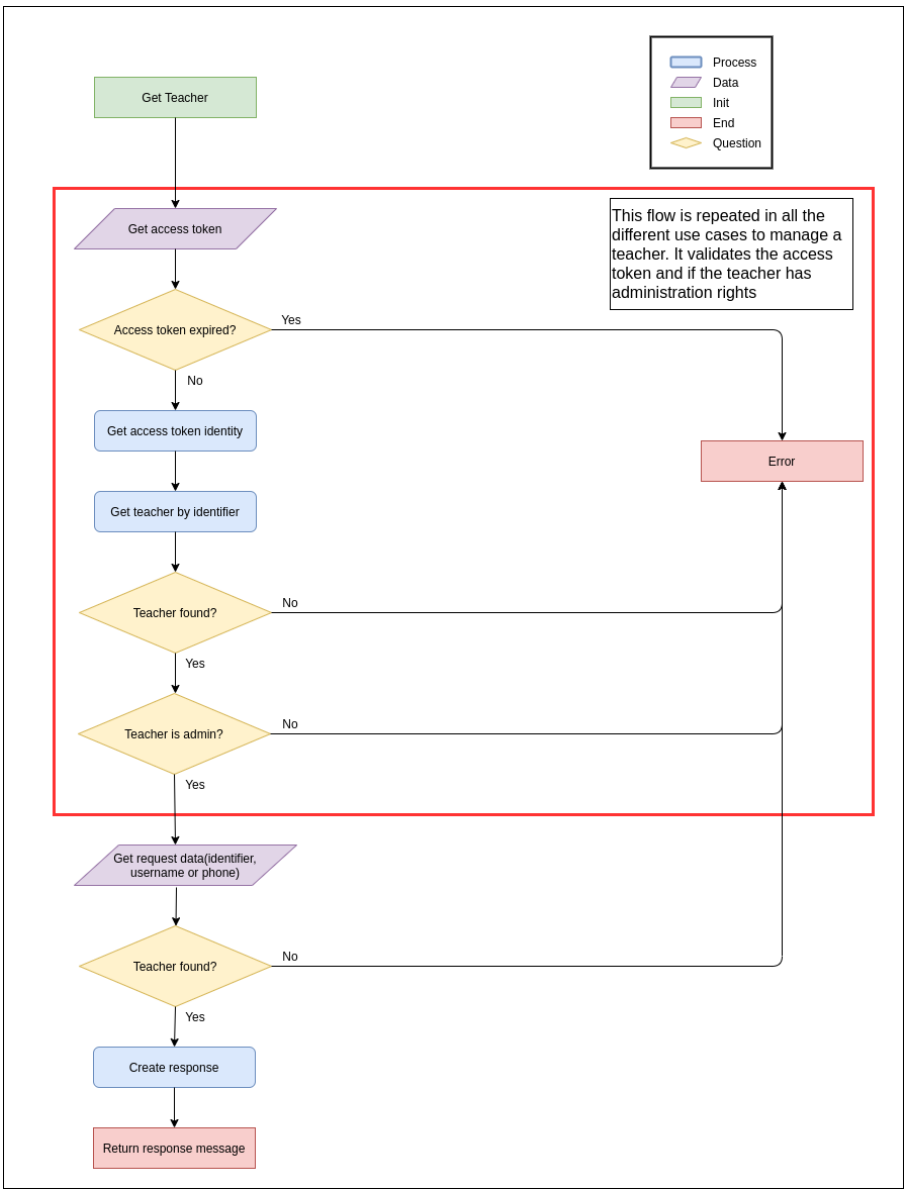


Figura 4.11: Consulta de la información de un profesor

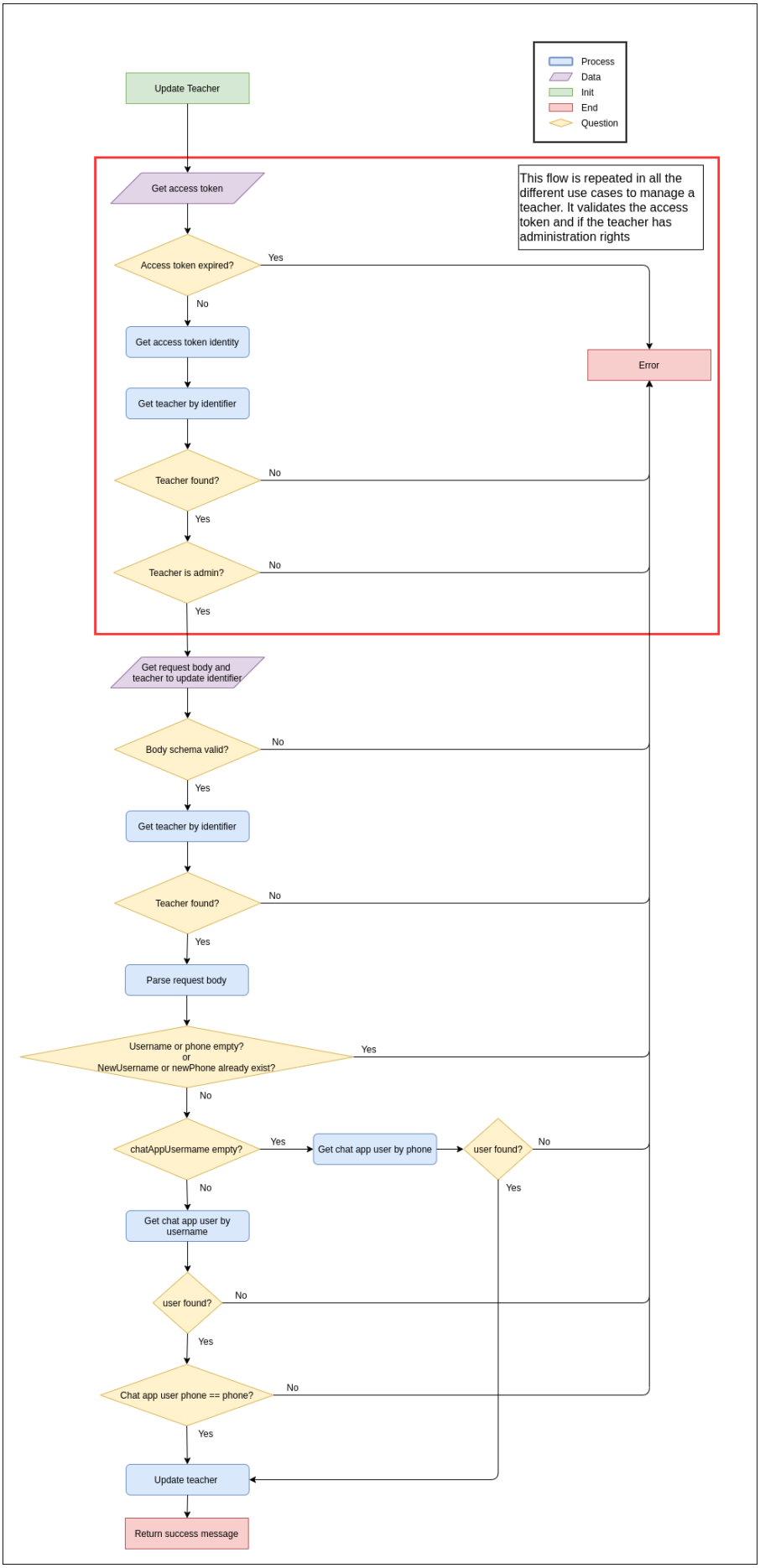


Figura 4.12: Actualización de un profesor

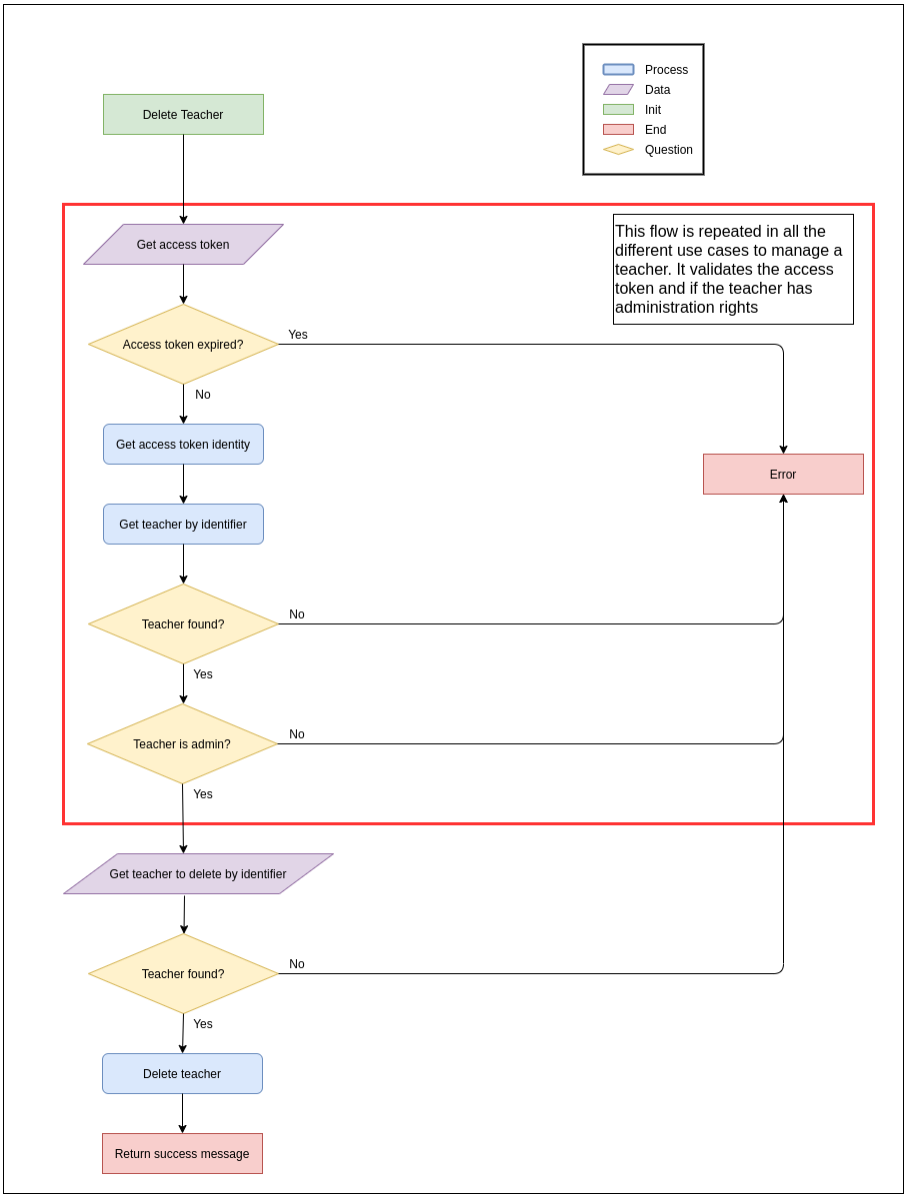


Figura 4.13: Borrado de un profesor

4.10.3. Gestión de un curso

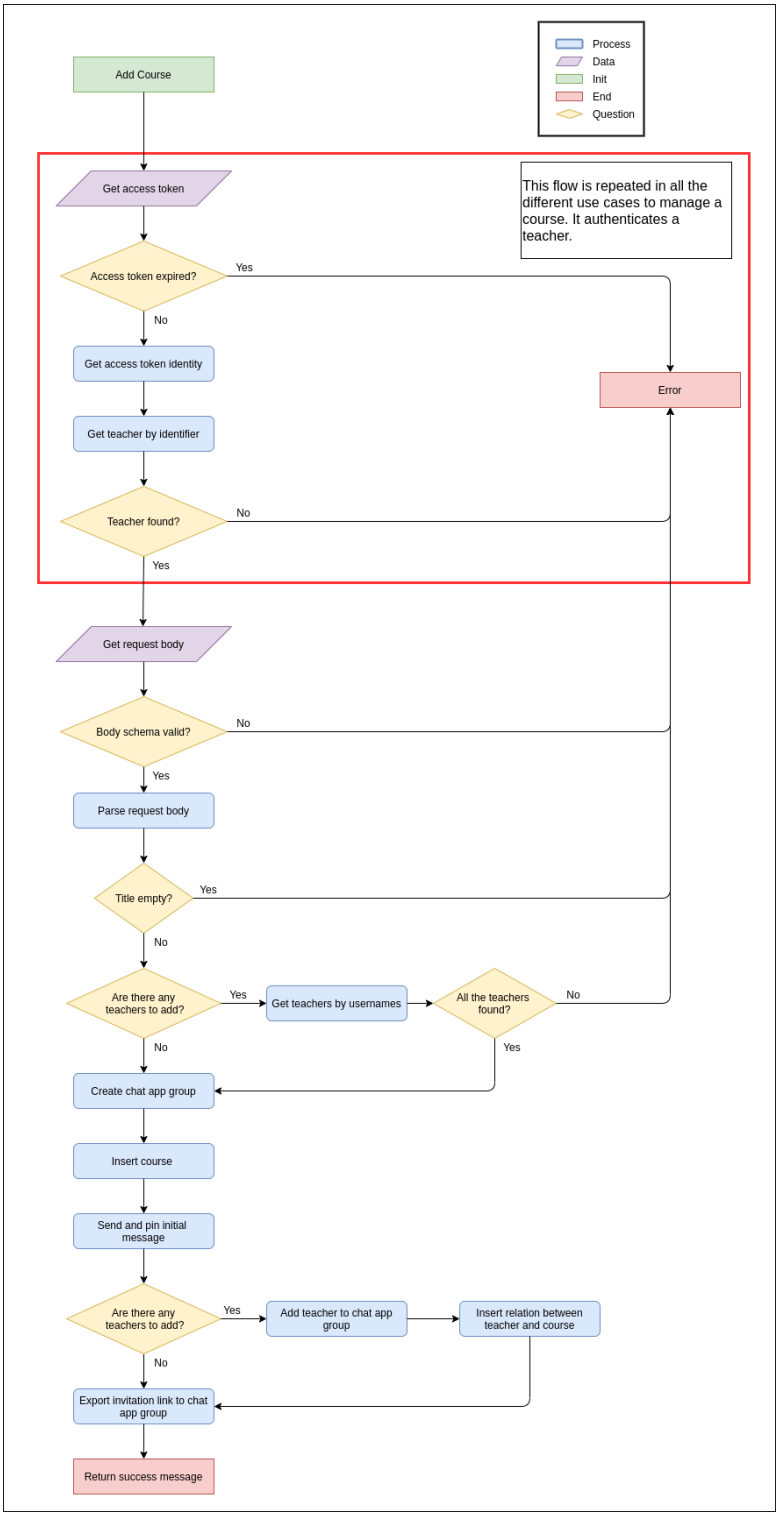


Figura 4.14: Creación de un curso

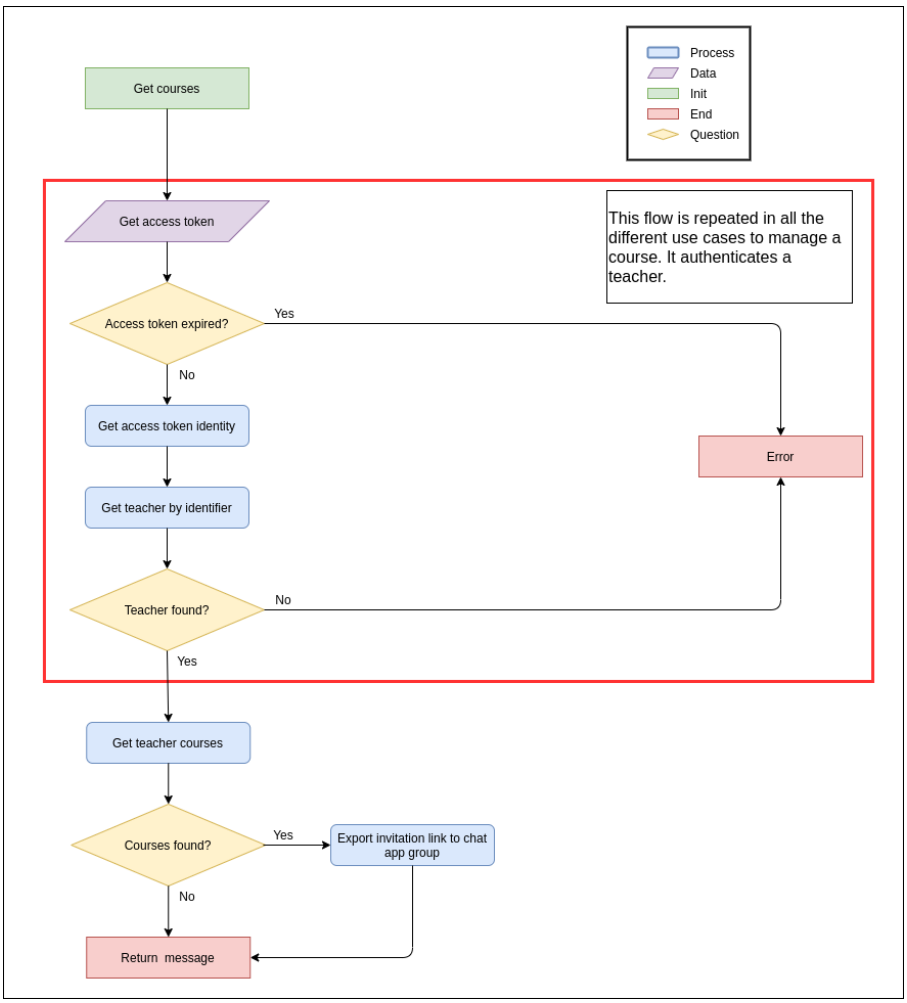


Figura 4.15: Consulta de información de cursos

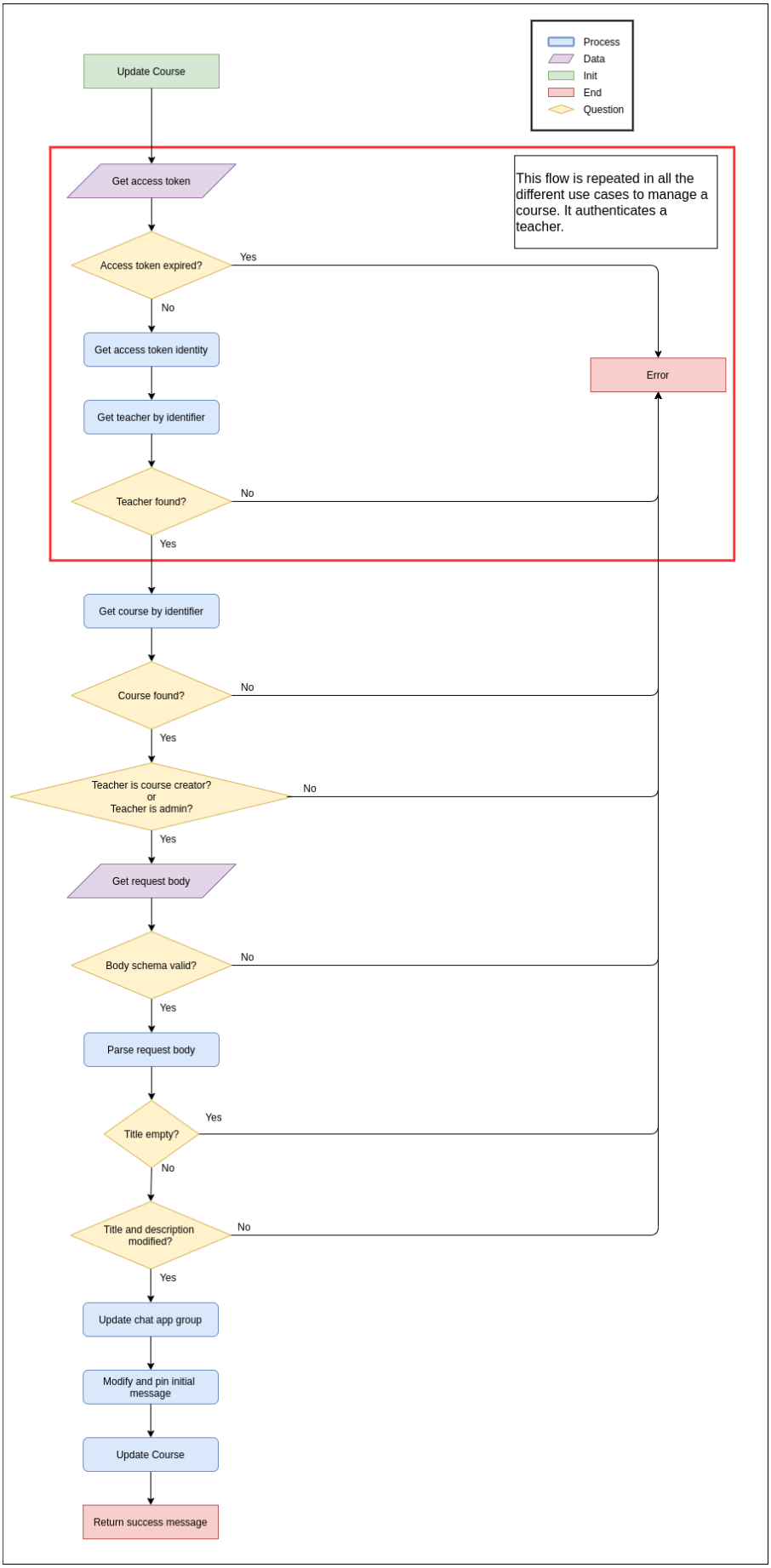


Figura 4.16: Actualización de un curso

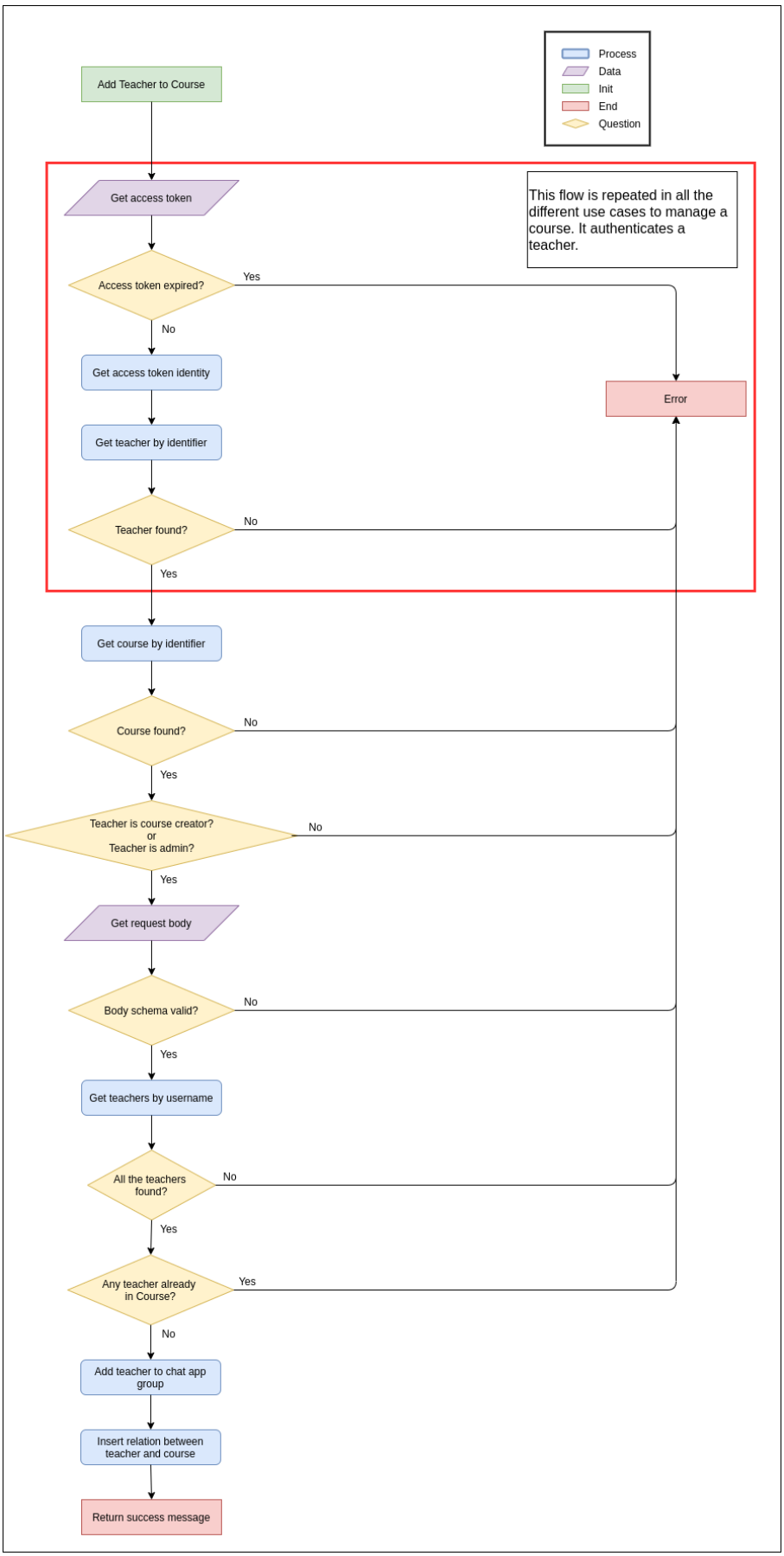


Figura 4.17: Adición de un profesor a un curso

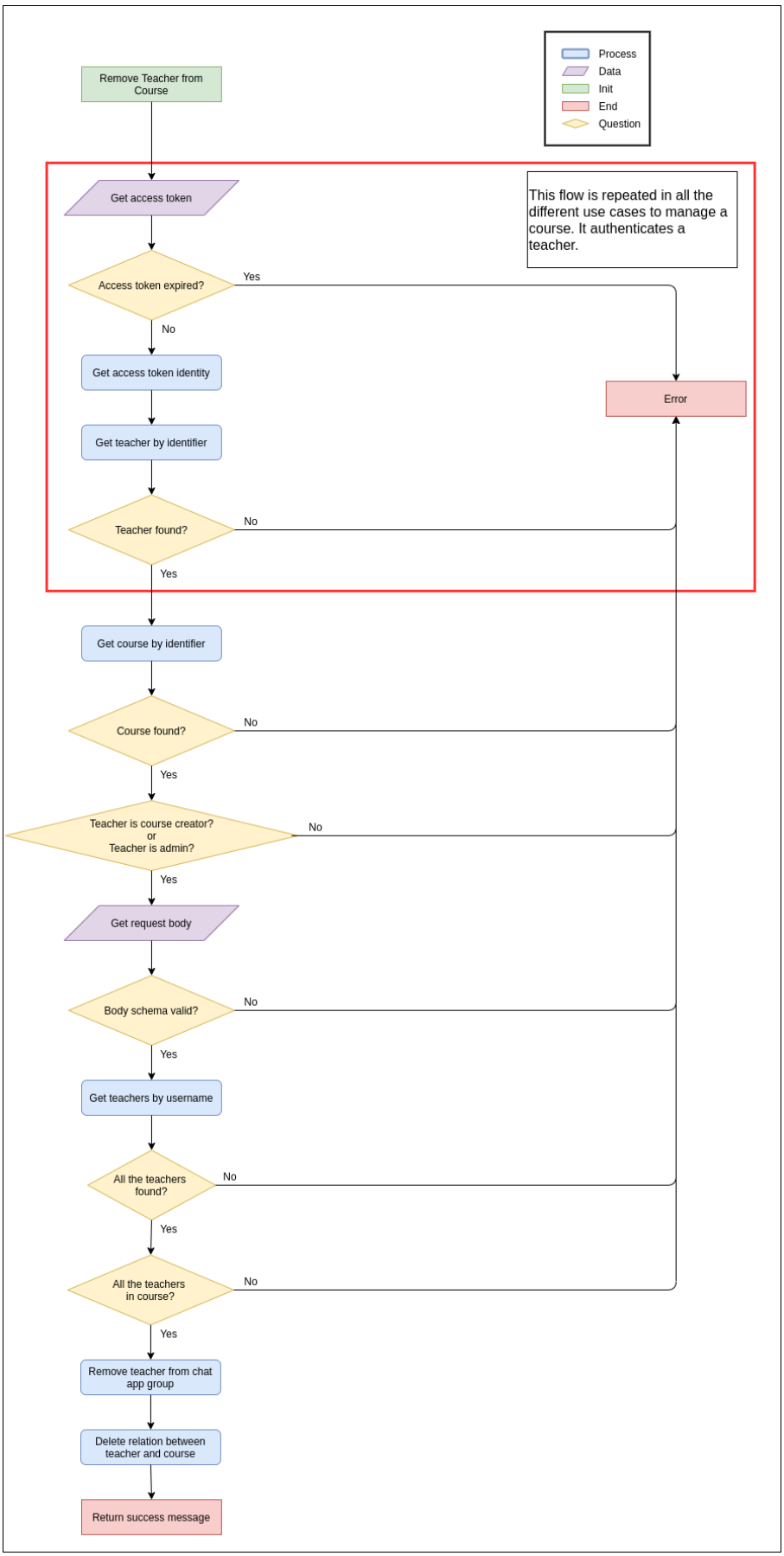


Figura 4.18: Borrado de un profesor de un curso

4.10.4. Gestión de un agente conversacional

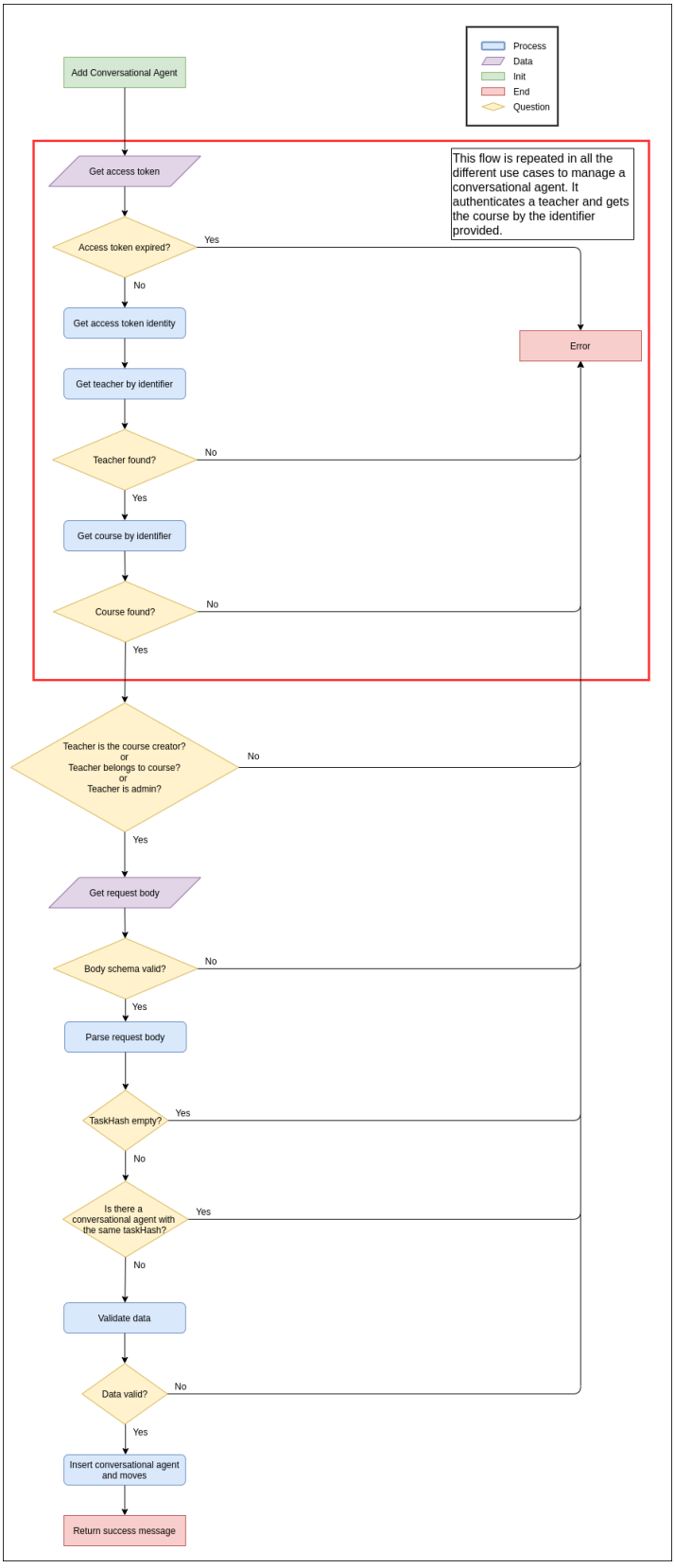


Figura 4.19: Creación de un agente conversacional

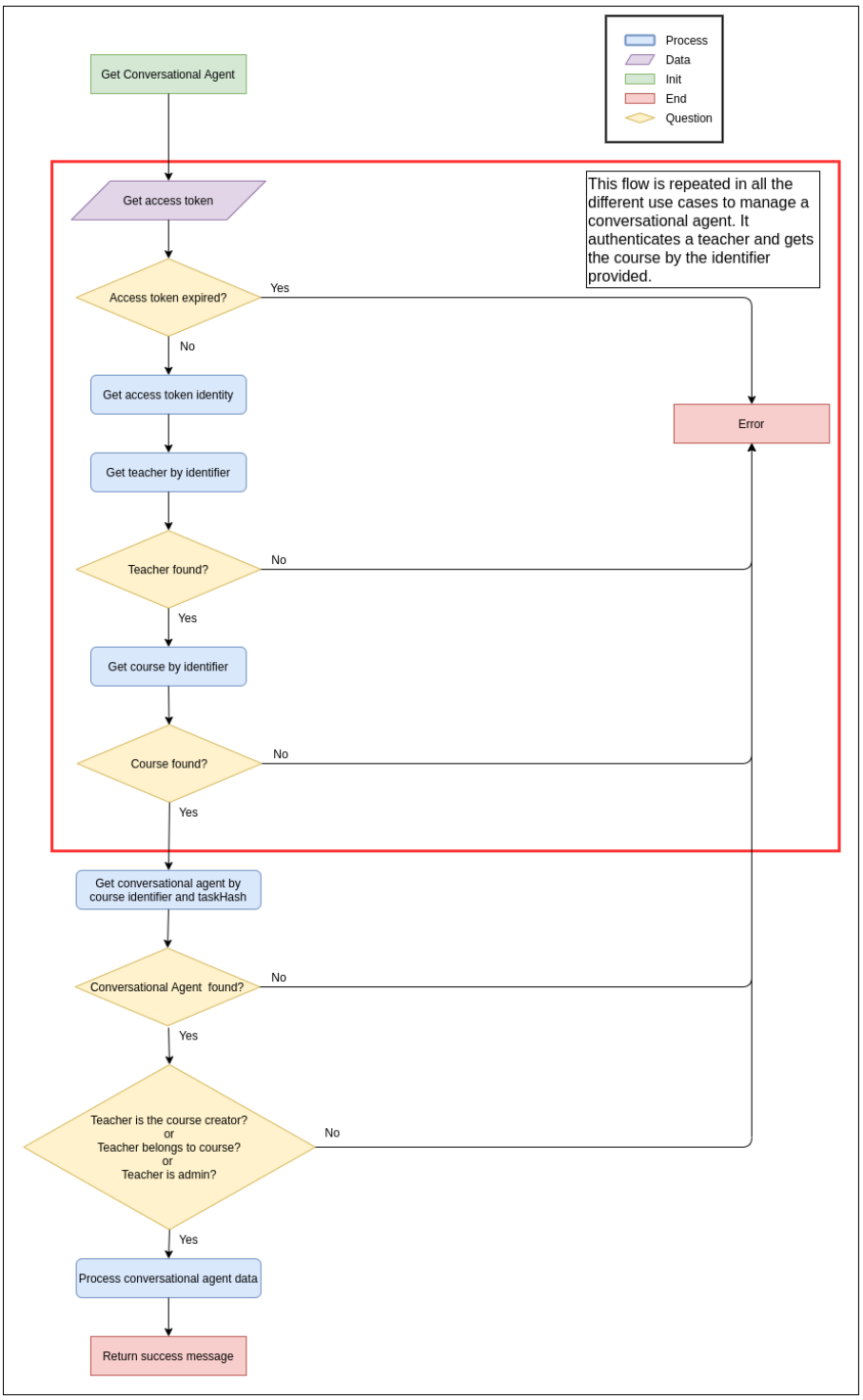


Figura 4.20: Consulta de información de un agente conversacional

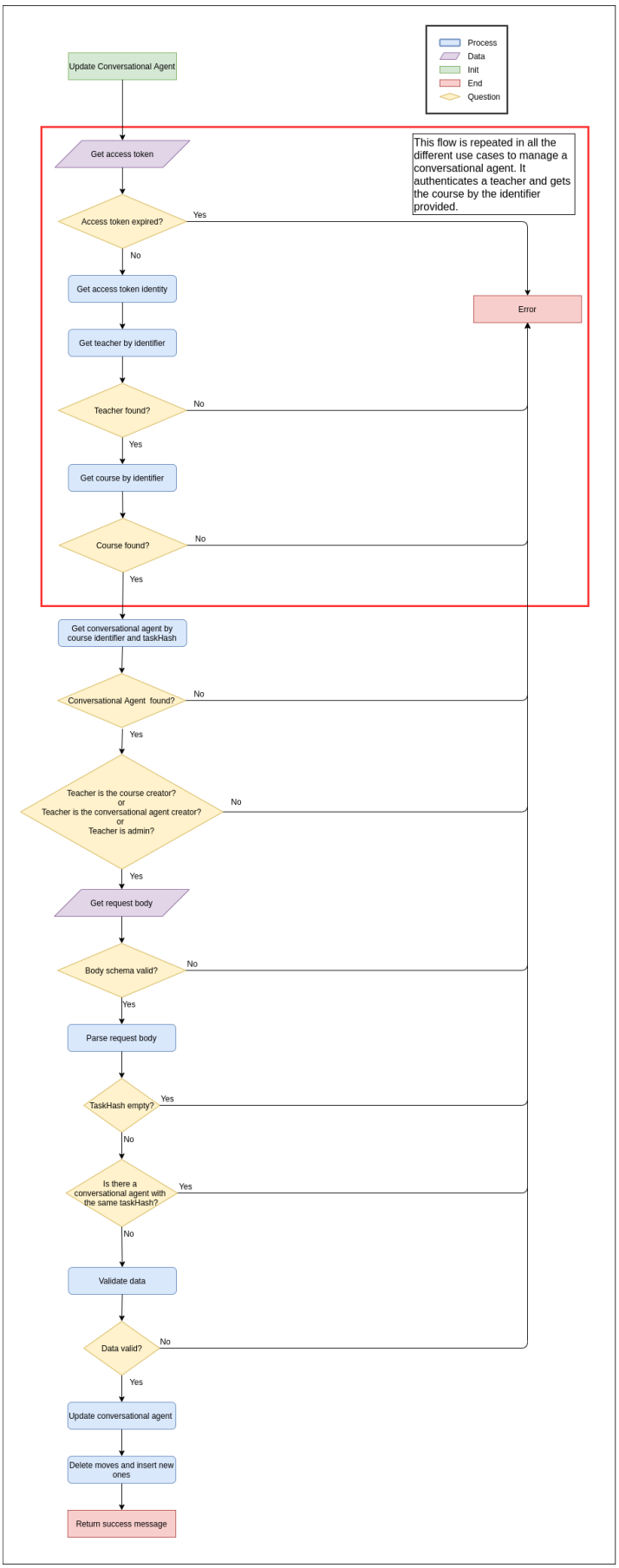


Figura 4.21: Actualización de un agente conversacional

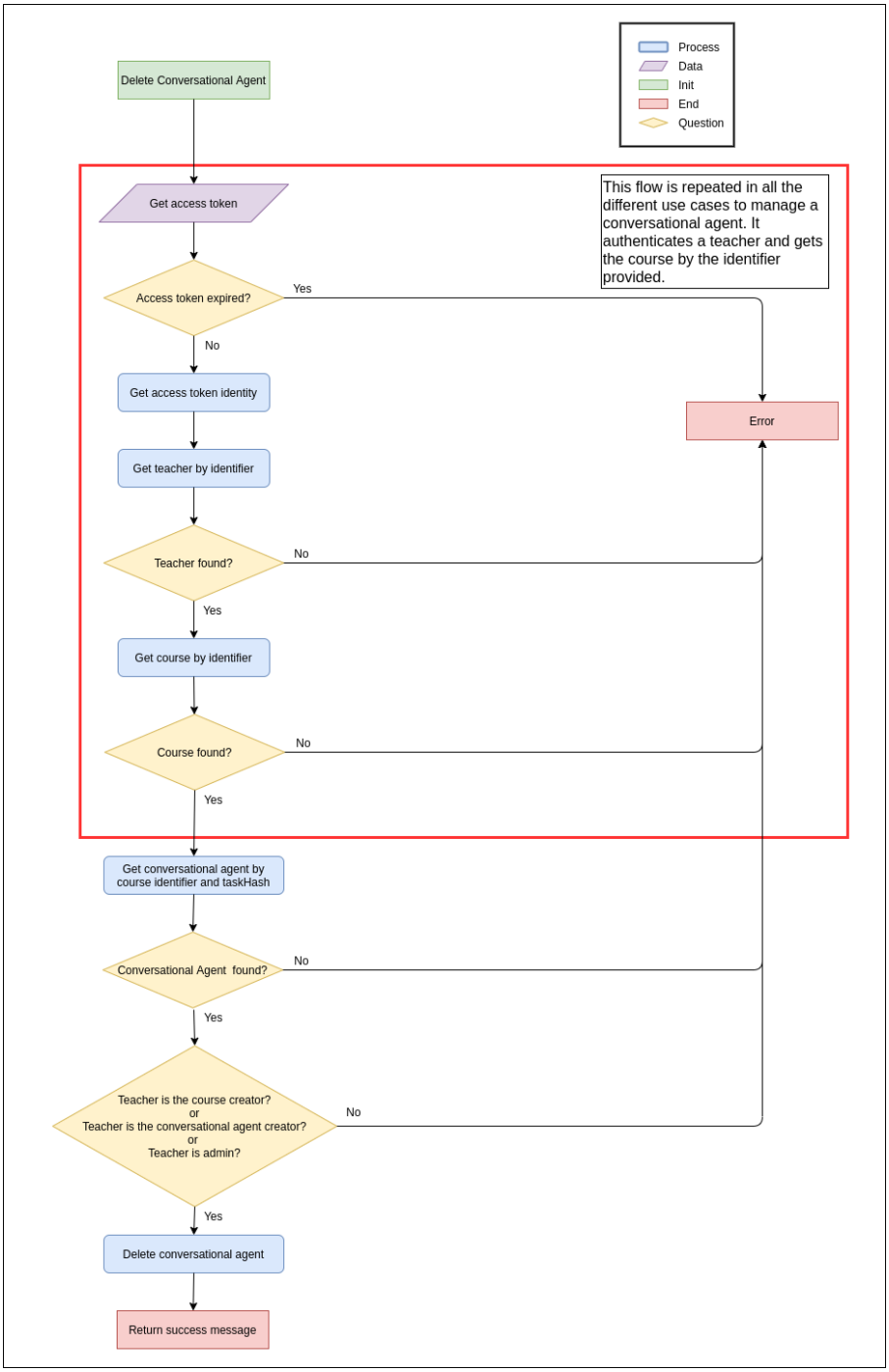


Figura 4.22: Borrado de un agente conversacional

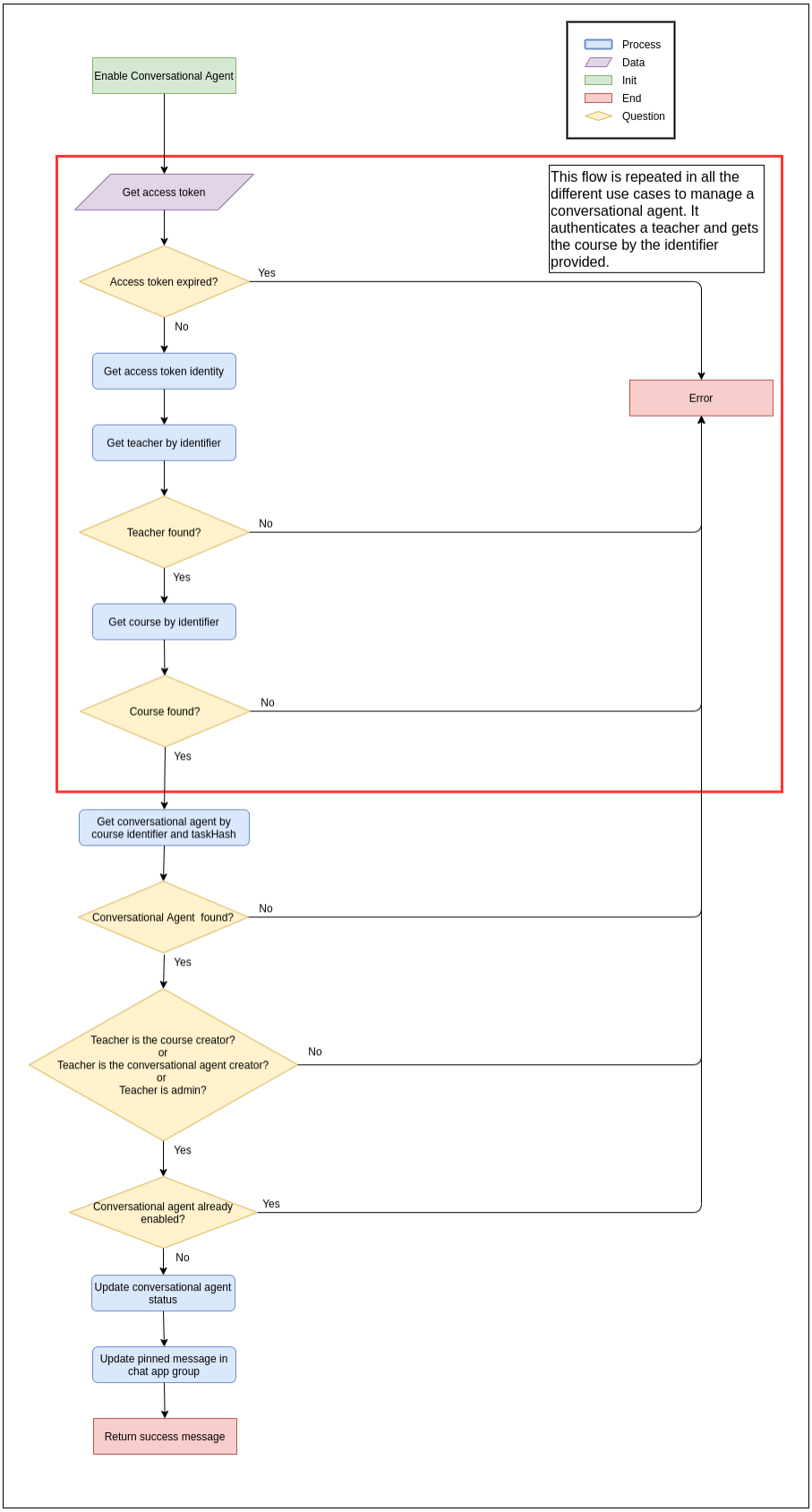


Figura 4.23: Activación de un agente conversacional

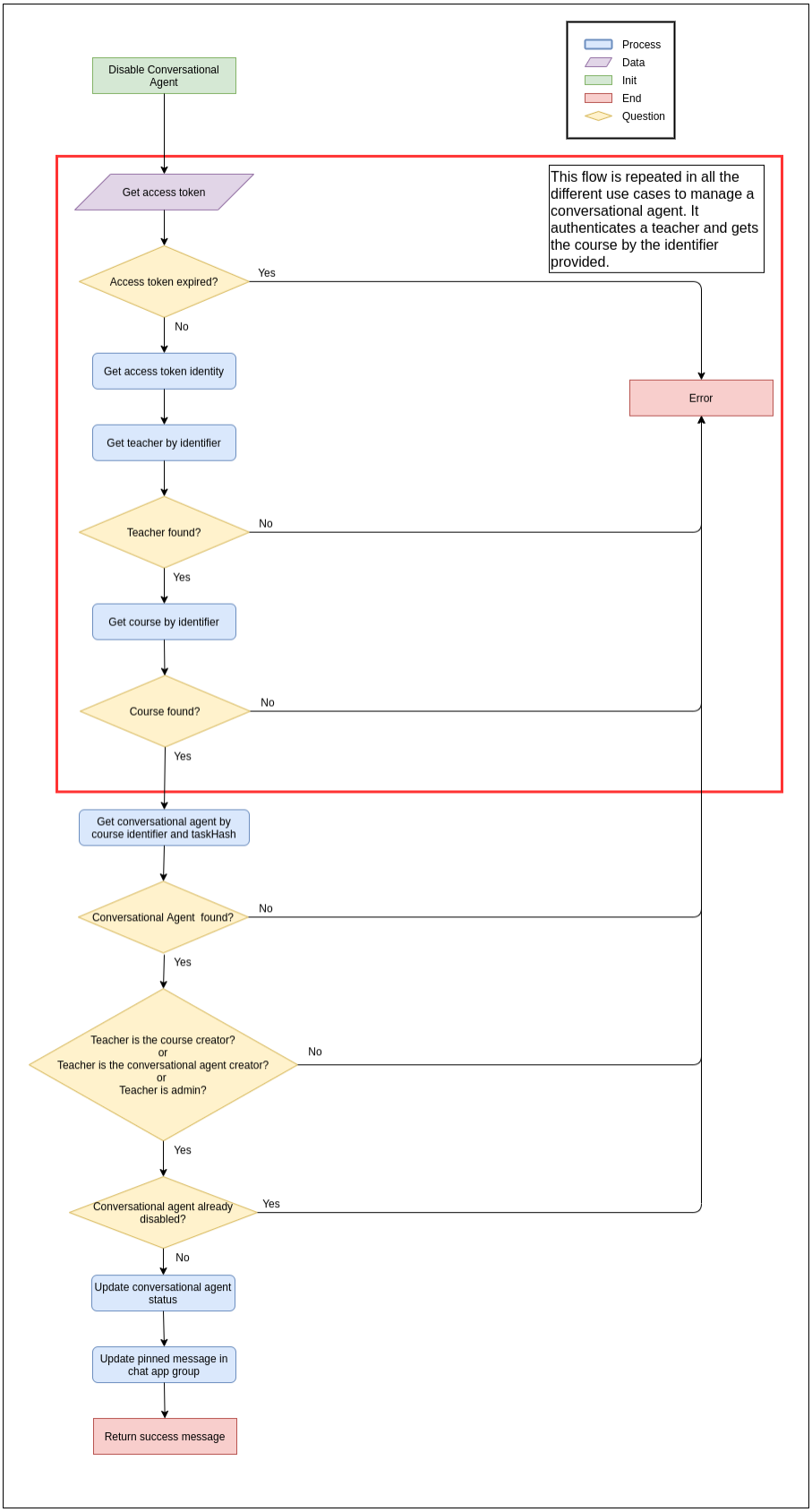


Figura 4.24: Desactivación de un agente conversacional

4.10.5. Consulta de información de salas de chats

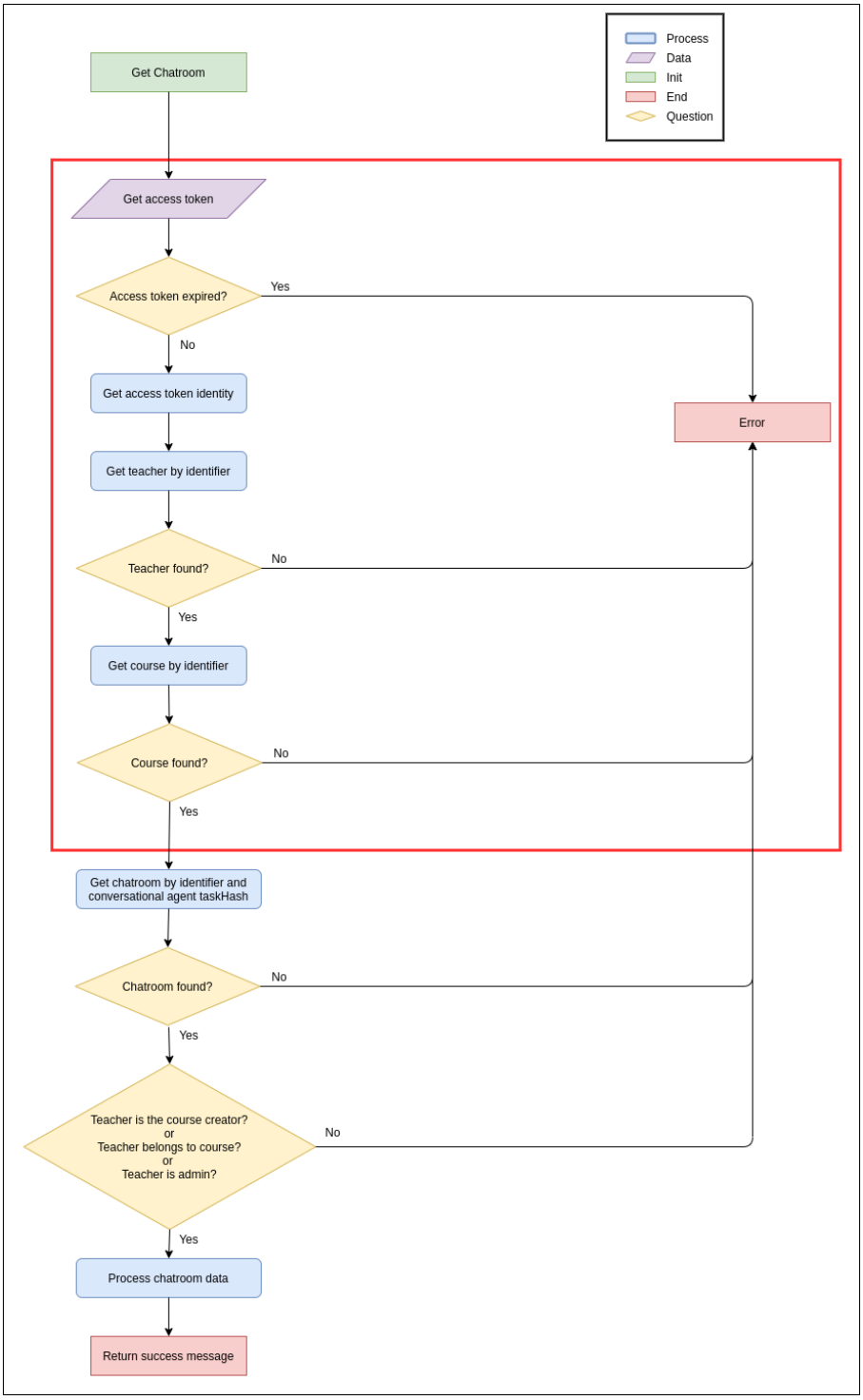


Figura 4.25: Consulta de información de una sala de chat

4.10.6. Consulta de información de estudiantes

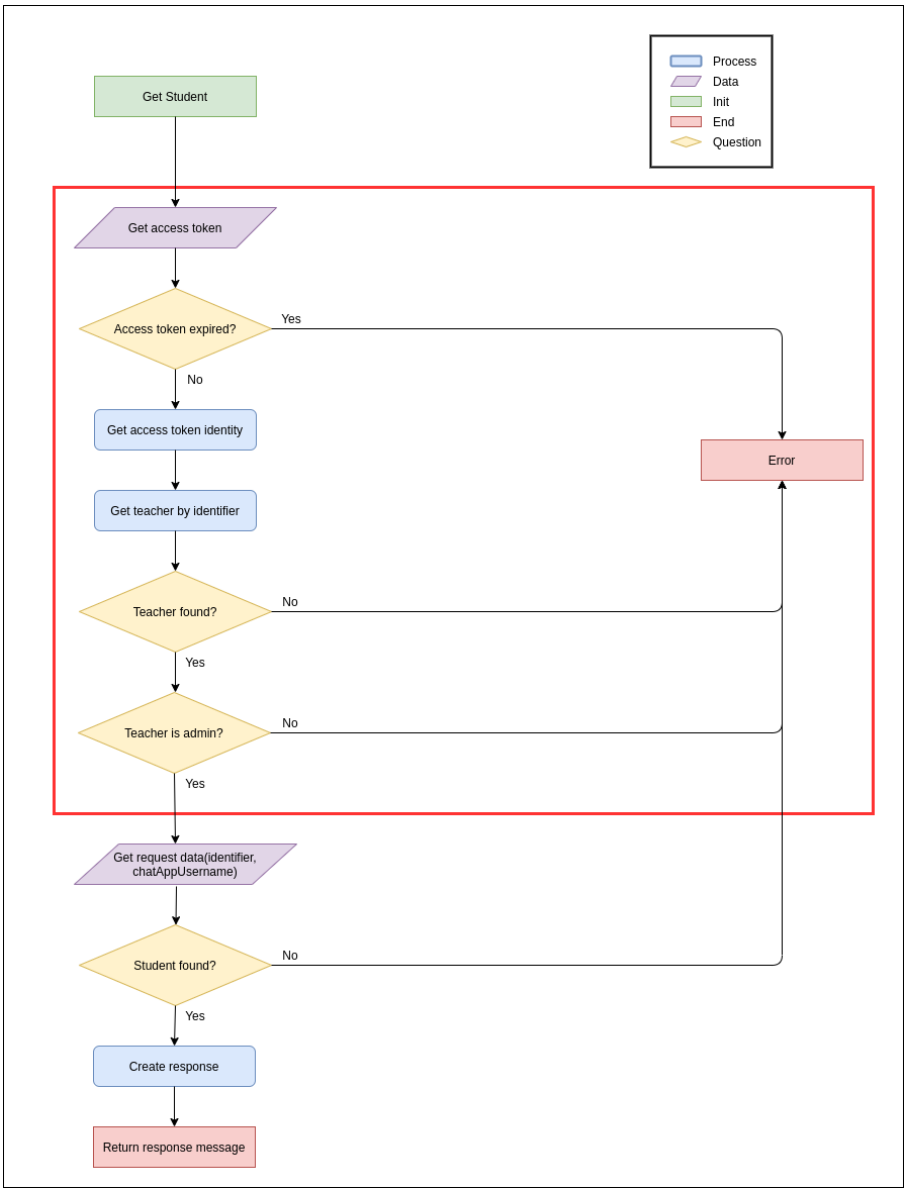


Figura 4.26: Consulta de información de un estudiante

4.10.7. Controlador de cursos

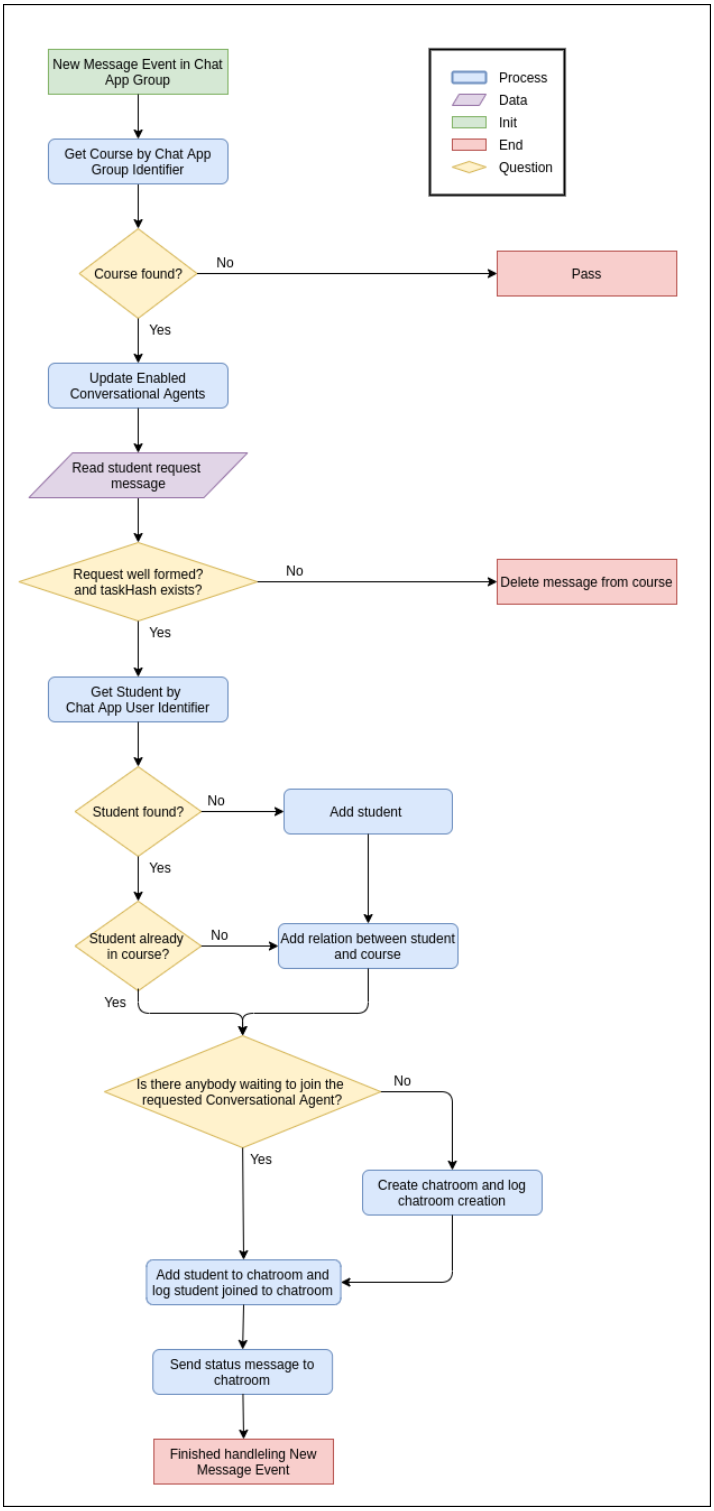


Figura 4.27: Eventos en controlador de cursos

4.10.8. Controlador de agentes conversacionales

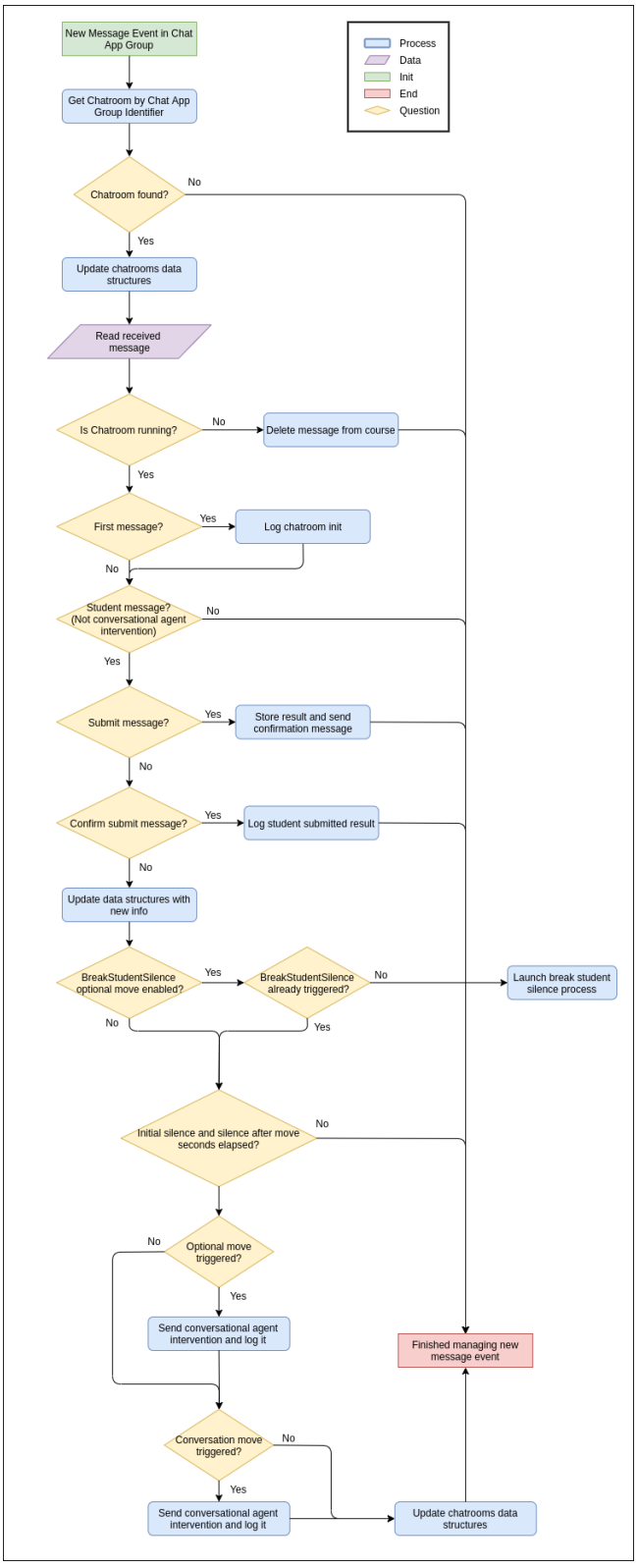


Figura 4.28: Eventos en controlador de agentes conversacionales

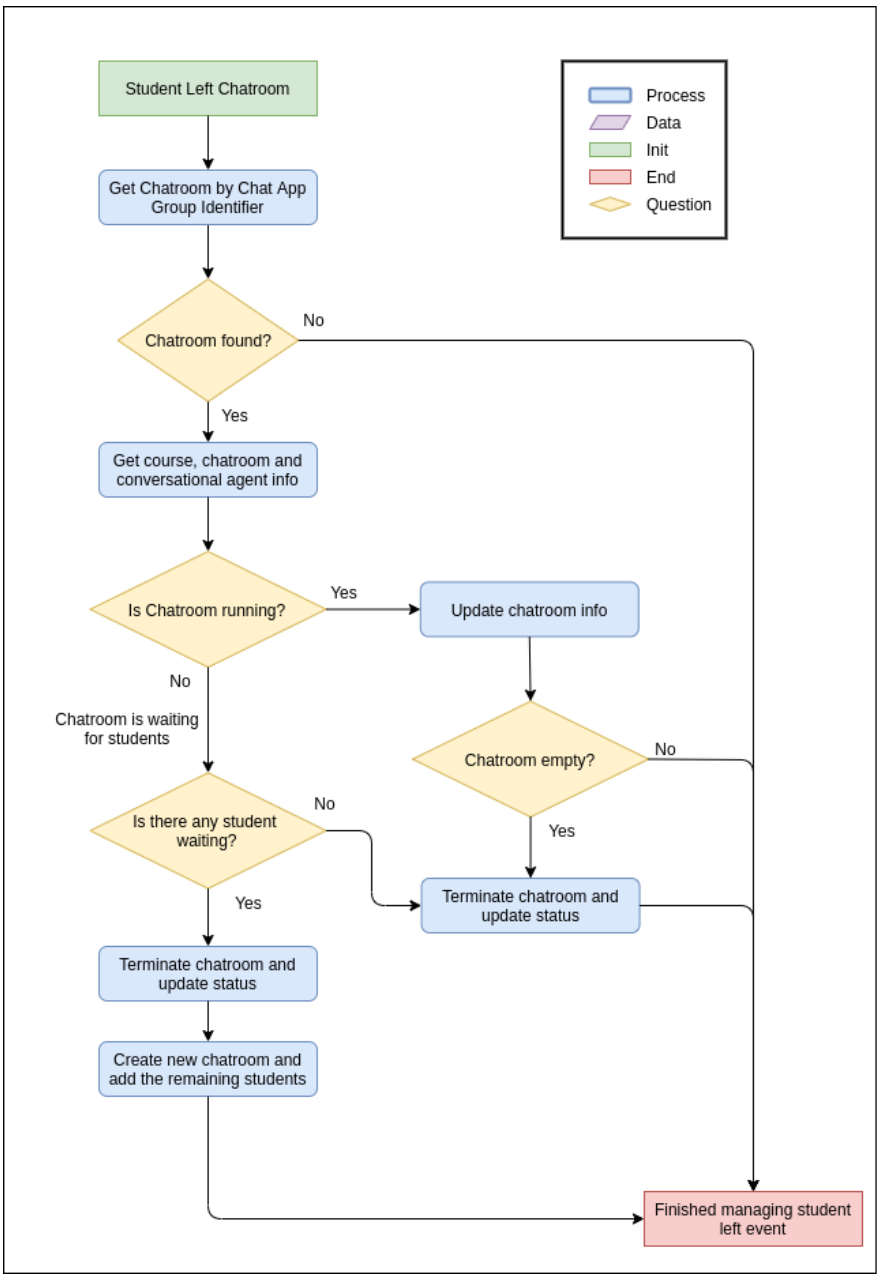


Figura 4.29: Salida de un estudiante de una sala de chat

Capítulo 5

Desarrollo

5.1. Entorno de Desarrollo

El entorno de desarrollo utilizado ha sido Visual Studio Code (VSCode) [1]. Este editor posee numerosas extensiones que permiten mejorar la fluidez del desarrollo. Además, permite la conexión directa con git.

5.2. Herramientas utilizadas

Las herramientas utilizadas durante el desarrollo han sido las siguientes:

- Postman [2]: aplicación muy potente que permite realizar peticiones a una API.
- MySQL Workbench [3]: permite la conexión con el sistema gestor de base de datos MySQL para realizar operaciones en la misma. Además, es capaz de generar diagramas y diferentes estadísticas.
- Visual Paradigm [4]: herramienta utilizada para la creación de diagramas.
- Draw.io [5]: herramienta utilizada para la creación de esquemas y diagramas.

5.3. Control de versiones

El sistema utilizado para controlar las versiones de la aplicación ha sido Github [6]. La metodología de trabajo ha sido la que se muestra en la Figura 5.1.

[1] VSCode - Entorno de desarrollo: https://code.visualstudio.com - último acceso: Julio 2020
[2] Postman - Herramienta para probar APIs: https://www.postman.com - último acceso: Julio 2020
[3] MySQL Workbench - Entorno gráfico para MySQL: mysql.com/products/workbench - último acceso: Julio 2020
[4] Visual Paradigm - Editor de diagramas: https://www.visual-paradigm.com - último acceso: Julio 2020
[5] Draw.io - Editor de diagramas y esquemas: https://app.diagrams.net - último acceso: Julio 2020
[6] Github - Sistema web para el control de versiones: https://github.com - último acceso: Julio 2020

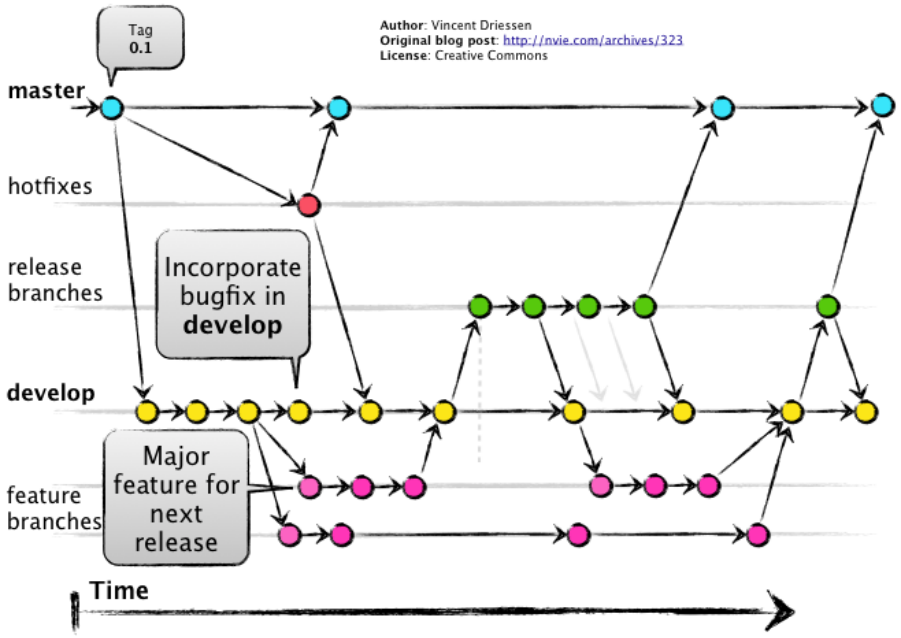


Figura 5.1: Utilización de ramas en Git [15]

Como se puede ver en la imagen anterior, para cada nueva funcionalidad de la aplicación que se iba a desarrollar, se creaba una rama empezando por el nombre *feature/*. Una vez finalizado dicho desarrollo, se incorporaba su contenido en la rama principal, denominada *Development*.

Finalmente, a la hora de publicar versiones, se creaba una nueva rama incorporando los contenidos de *Development*. A partir de esta rama, se publicaba una *release* con un *tag* indicando el número de versión. Los números de versión cumplen el formato [16] vX.Y.Z. En función del cambio realizado en el código se actualizará uno de estos números:

- X: se han realizado modificaciones en la funcionalidad.
- Y: se han añadido nuevas funcionalidades.
- Z: se han realizado revisiones y correcciones.

5.4. Librerías utilizadas

Se han utilizado numerosas librerías estándar de Python. Las librerías externas utilizadas han sido las siguientes:

- Telethon: utilizada para realizar las operaciones con Telegram.
- MySQL-Connector-Python: sirve para realizar la conexión de con MySQL.
- Flask: framework para aplicación web.
- Flask-Bcrypt: extensión de Flask utilizada para la encriptación de contraseñas.
- Flask-JWT-Extended: extensión de Flask utilizada para integrar la seguridad basada en JSON Web Token.

- Jsonschema: librería utilizada para la validación de los datos de entrada en la Rest API.
- ZMQ: librería utilizada para trabajar con el sistema de colas Zero Message Queue.

5.5. Estructura del proyecto

El proyecto tiene la estructura [17] que aparece en la Figura 5.2.

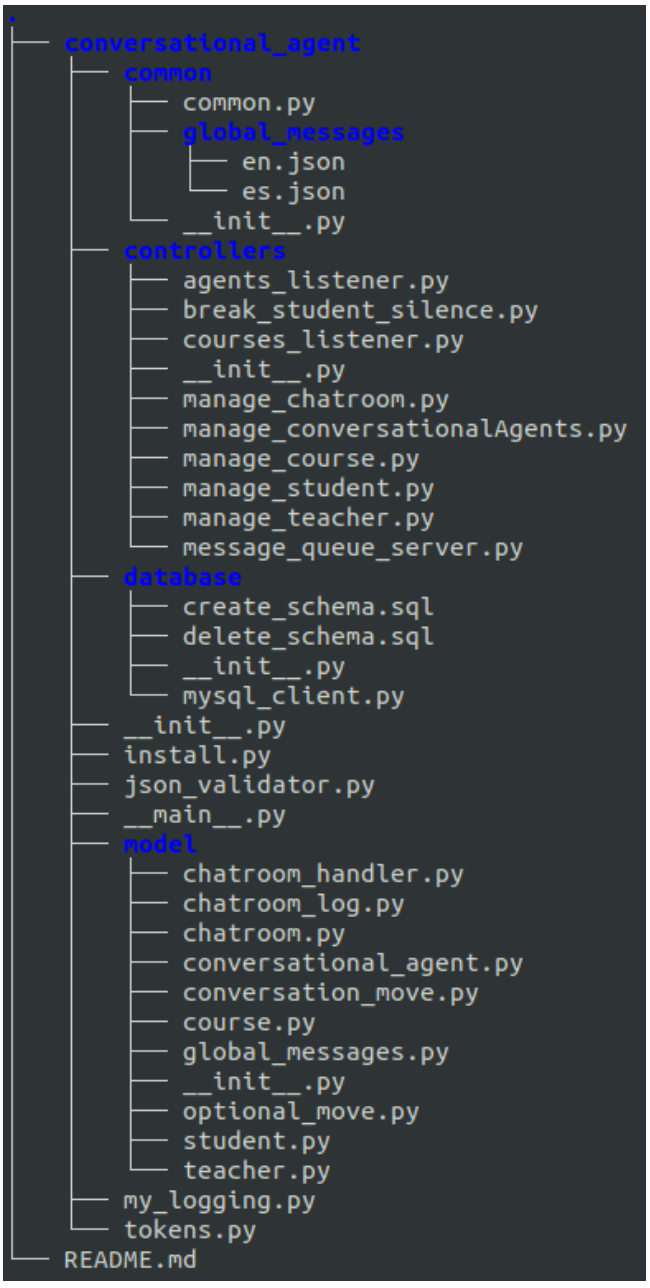


Figura 5.2: Estructura del proyecto

En la carpeta *common* se encuentran las variables globales y los ficheros de idiomas. Los controladores están almacenados en el directorio *controllers*, mientras que los modelos están en *model*. El módulo encargado de realizar las operaciones con base de datos se encuentra en *database*. En el directorio principal se encuentran los siguientes módulos:

- *json_validator*: se encarga de la validación de los datos de entrada con estructura JSON.
- *tokens*: datos de configuración y seguridad del sistema.
- *my_logging*: módulo para almacenar registros.
- *install*: se encarga de instalar las librerías externas.

Capítulo 6

Pruebas

En este capítulo aparecen las diferentes pruebas realizadas para comprobar el correcto funcionamiento de la aplicación. También, se explican las dificultades encontradas a la hora de probar ciertas funcionalidades. Finalmente, se comenta los resultados de una prueba de usabilidad realizada con usuarios reales.

6.1. Casos de prueba

6.1.1. Inicio de sesión

CP01	**Un profesor inicia sesión**
Proceso	Se realiza una petición GET al punto de acceso */login* con el nombre de usuario *alemeri* y contraseña *pruebasAgente2020*.
Resultado esperado	El sistema muestra el token de acceso necesario para realizar el resto de operaciones.

Tabla 6.1: CP01: Un profesor inicia sesión

CP02	**Un profesor no registrado inicia sesión**
Proceso	Se realiza una petición GET al punto de acceso */login* con el nombre de usuario *noExisto* y contraseña *passErronea*.
Resultado esperado	El sistema muestra un mensaje de error con la información de que no se ha encontrado dicho profesor.

Tabla 6.2: CP02: Un profesor no registrado inicia sesión

6.1.2. Gestión de profesores

CP03	**Creación de un profesor**
Proceso	Se realiza una petición POST al punto de acceso */teachers* con el nombre de usuario *rosba*, contraseña *rosba2020*, numero de telefono *+34XXXXXXXXX* y nombre de usuario en Telegram *rosba25* .
Resultado esperado	El sistema muestra un mensaje de que el profesor se ha añadido correctamente.

Tabla 6.3: CP03: Creación de un profesor

CP04	**Creación de un profesor ya existente**
Proceso	Se realiza una petición POST al punto de acceso */teachers* con el nombre de usuario *rosba*, contraseña *rosba2020*, número de teléfono *+34XXXXXXXXX* nombre de usuario en Telegram *rosba25* .
Resultado esperado	El sistema muestra un mensaje de error que el profesor ya existe en el sistema.

Tabla 6.4: CP04: Creación de un profesor ya existente

CP05	**Creación de un profesor no registrado en Telegram**
Proceso	Se realiza una petición POST al punto de acceso */teachers* con el nombre de usuario *lucas*, contraseña *lucas2020*, número de teléfono *+34XXXXXXXXX* nombre de usuario en Telegram *noExisto* .
Resultado esperado	El sistema muestra un mensaje de error que el profesor está registrado en Telegram.

Tabla 6.5: CP05: Creación de un profesor no registrado en Telegram

CP06	**Búsqueda de un profesor**
Proceso	Se realiza una petición GET al punto de acceso */teachers/alemeri*.
Resultado esperado	El sistema muestra un mensaje la información del profesor.

Tabla 6.6: CP06: Búsqueda de un profesor

CP07	**Búsqueda de un profesor que no existe**
Proceso	Se realiza una petición GET al punto de acceso */teachers/noExiste*.
Resultado esperado	El sistema muestra un mensaje de error informando que el profesor no ha sido encontrado.

Tabla 6.7: CP07: Búsqueda de un profesor que no existe

CP08	**Actualización de un profesor**
Proceso	Se realiza una petición PUT al punto de acceso */teachers/rosba* con el nombre de usuario *rosba2020*, contraseña *rosba2020*, número de teléfono *+34XXXXXXXXX*, nombre de usuario en Telegram *rosba25*.
Resultado esperado	El sistema muestra un mensaje de que el profesor se ha modificado correctamente.

Tabla 6.8: CP08: Actualización de un profesor

CP09	**Actualización de un profesor. Nombre de usuario no modificado**
Proceso	Se realiza una petición PUT al punto de acceso */teachers/rosba* con el nombre de usuario *rosba*, contraseña *rosba2020*, número de teléfono *+34XXXXXXXXX*, nombre de usuario en Telegram *rosba25*.
Resultado esperado	El sistema muestra un mensaje de error indicando que el nombre de usuario no se ha modificado.

Tabla 6.9: CP09: Actualización de un profesor. Nombre de usuario no modificado

CP10	**Actualización de un profesor. Nombre de usuario ya registrado**
Proceso	Se realiza una petición PUT al punto de acceso */teachers/rosba* con el nombre de usuario *lucas*, contraseña *rosba2020*, número de teléfono *+34XXXXXXXXX*, nombre de usuario en Telegram *rosba25*.
Resultado esperado	El sistema muestra un mensaje de error indicando que ya existe un usuario con ese nombre de usuario.

Tabla 6.10: CP10: Actualización de un profesor. Nombre de usuario ya registrado

CP11	**Eliminación de un profesor**
Proceso	Se realiza una petición DELETE al punto de acceso */teachers/rosba*.
Resultado esperado	El sistema muestra un mensaje de el profesor ha sido eliminado correctamente.

Tabla 6.11: CP11: Eliminación de un profesor

CP12	**Eliminación de un profesor no existente**
Proceso	Se realiza una petición DELETE al punto de acceso */teachers/noExisto*.
Resultado esperado	El sistema muestra un mensaje de error indicando que el profesor no existe.

Tabla 6.12: CP12: Eliminación de un profesor no existente

6.1.3. Gestión de cursos

CP13	**Creación de un curso**
Proceso	Se realiza una petición POST al punto de acceso */courses* con el título *Fundamentos de Agentes Conversacionales*, la descripción *Curso sobre Agentes Conversacionales*, el idioma *es* y el profesor *lucas*.
Resultado esperado	El sistema muestra un mensaje de que se ha creado el curso exitosamente, indicando su identificador, y en Telegram se puede observar que se ha creado el grupo correspondiente.

Tabla 6.13: CP13: Creación de un curso

CP14	**Creación de un curso con un idioma no disponible**
Proceso	Se realiza una petición POST al punto de acceso */courses* con el título *Fundamentos de Agentes Conversacionales*, la descripción *Curso sobre Agentes Conversacionales*, el idioma *nan* y el profesor *lucas*.
Resultado esperado	El sistema muestra un mensaje de error indicando que el idioma solicitado no está disponible.

Tabla 6.14: CP14: Creación de un curso con un idioma no disponible

CP15	**Actualización de un curso**
Proceso	Se realiza una petición PUT al punto de acceso */courses/1* con el título *Fundamentos de Agentes Conversacionales II*, la descripción *Curso sobre Agentes Conversacionales II* y el idioma *es*.
Resultado esperado	El sistema muestra un mensaje de que el curso ha sido actualizado correctamente y en Telegram se puede observar dicha modificación.

Tabla 6.15: CP15: Actualización de un curso

CP16	**Actualización de un curso no registrado en el sistema**
Proceso	Se realiza una petición PUT al punto de acceso */courses/11982423* con el título *Fundamentos de Agentes Conversacionales II*, la descripción *Curso sobre Agentes Conversacionales II* y el idioma *es*.
Resultado esperado	El sistema muestra un mensaje indicando que el curso no existe en el sistema.

Tabla 6.16: CP16: Actualización de un curso no registrado en el sistema

CP17	**Actualización de un curso no registrado en el sistema**
Proceso	Se realiza una petición PUT al punto de acceso */courses/1* con el título *Fundamentos de Agentes Conversacionales*, la descripción *Curso sobre Agentes Conversacionales* y el idioma *es*.
Resultado esperado	El sistema muestra un mensaje de error indicando que ni el título ni la descripción han sido modificados.

Tabla 6.17: CP17: Actualización de un curso sin modificar el título o la descripción

CP18	**Búsqueda de un curso**
Proceso	Se realiza una petición GET al punto de acceso */courses/1*.
Resultado esperado	El sistema muestra un mensaje indicando la información del curso.

Tabla 6.18: CP18: Búsqueda de un curso

CP19	**Búsqueda de un curso no registrado en el sistema**
Proceso	Se realiza una petición GET al punto de acceso */courses/11982423*.
Resultado esperado	El sistema muestra un mensaje indicando que el curso no existe en el sistema.

Tabla 6.19: CP19: Búsqueda de un curso no registrado en el sistema

CP20	**Adición de un profesor a un curso**
Proceso	Se realiza una petición POST al punto de acceso */courses/1/teachers* con el nombre de usuario del profesor *lucas*.
Resultado esperado	El sistema muestra un mensaje indicando que el profesor ha sido añadido correctamente al curso.

Tabla 6.20: CP20: Adición de un profesor a un curso

CP21	**Adición de un profesor, no registrado en el sistema, a un curso**
Proceso	Se realiza una petición POST al punto de acceso */courses/1/teachers* con el nombre de usuario del profesor *noExisto*.
Resultado esperado	El sistema muestra un mensaje de error indicando que el profesor no existe en el sistema.

Tabla 6.21: CP21: Adición de un profesor, no registrado en el sistema, a un curso

CP22	**Borrado de un profesor de un curso**
Proceso	Se realiza una petición DELETE al punto de acceso */courses/1/teachers* con el nombre de usuario del profesor *lucas*.
Resultado esperado	El sistema muestra un mensaje indicando que el profesor ha sido eliminado correctamente del curso.

Tabla 6.22: CP22: Borrado de un profesor de un curso

CP23	Borrado de un profesor no registrado en el curso
Proceso	Se realiza una petición DELETE al punto de acceso */courses/1/teachers* con el nombre de usuario del profesor *noExisto*.
Resultado esperado	El sistema muestra un mensaje de error indicando que el profesor no existe en el curso.

Tabla 6.23: CP23: Borrado de un profesor no registrado en el curso

6.1.4. Gestión de agentes conversacionales

CP24	Creación de un agente conversacional
Proceso	Se realiza una petición POST al punto de acceso */courses/1/agents* con la información del agente conversacional *Beneficios del aprendizaje colaborativo - BNFCOLAB*, Código 6.1.
Resultado esperado	El sistema muestra un mensaje indicando que el agente conversacional se ha creado correctamente.

Tabla 6.24: CP24: Creación de un agente conversacional

Código 6.1: Agente Conversacional Beneficios del aprendizaje colaborativo

```
{   "conversationalAgent" : {
    "taskTitle" : "Beneficios del aprendizaje colaborativo",
    "taskHash" : "BNFCOLAB",
    "taskIcon" : "http://blogbibliotecas.mecd.gob.es/wp-content/
          uploads/2017/04/Aprendizaje-cooperativo-destacada.jpg",
    "taskDescription" : "Discute con tus companeros y acuerda
              **tres** beneficios del aprendizaje colaborativo",
    "chatLanguage" : "es",
    "groupSize" : 2,
    "chatParsingWindow" : 2,
    "agentInitialSilence" : 30,
    "agentSilenceAfterMove" : 20,
    "welcomeToFirst" : "{issuingStudent}, te doy la bienvenida a
              esta sala de chat. Estamos esperando a que se unan
              los companeros que completen el grupo de discusion.
              Hasta entonces el chat permanecera inactivo y no
              podreis teclear mensajes",
    "welcomeToLast" : "{issuingStudent}, contigo se completa el
              grupo de discusion, y ya podeis comenzar a
              intercambiar mensajes",
    "replyMessages" : true,
    "optionalMoves" : [
        {
```

```
        "moveType" : "breakStudentSilence",
        "moveId" : "breakStudentSilence",
        "message" : "Parece que llevais mucho tiempo sin decir
            nada. Por que no intentais retomar la conversacion",
        "seconds" : 30,
        "priority" : 50
    },
    {
        "moveType" : "bringBackToTopic",
        "moveId" : "bringBackToTopic",
        "message" : "Quiza seria bueno que volvieseis a hablar
                    sobre la tarea",
        "numMessages" : 10,
        "priority" : 51
    },
    {
        "moveType" : "encourageSilentStudent",
        "moveId" : "encourageSilentStudent",
        "message" : "{otherStudent}, no quieres aportar tu
                algo a lo que esta diciendo tu companero",
        "numMessages" : 10,
        "priority" : 52
    },
    {
        "moveType" : "encourageSubmission",
        "moveId" : "encourageSubmission",
        "message" : "Quiza esta discusion se esta alargando
                    demasiado y es momento de que penseis en
                    dar una respuesta final, no creeis",
        "numMessages" : 20,
        "minutes" : 5,
        "priority" : 53
    }
],
"conversationMoves" : [
    {
        "moveId" : "explain01",
        "moveType" : "explain",
        "firstConcept" : "interdependencia positiva",
        "firstSynonyms" : ["dependencia mutua","dependencia"],
        "directed" : true,
        "message" : "{issuingStudent}, puedes aclarar un poco
                    mas que quieres decir con interdependencia
                    positiva",
        "priority" : 4
    },
    {
        "moveId" : "addOn01",
        "moveType" : "addOn",
```

```
    "firstConcept" : "interdependencia positiva",
    "firstSynonyms" : ["dependencia mutua","dependencia"],
    "directed" : true,
    "message" : "{otherStudent}, puedes anadir algo a lo
                 que tu companero ha dicho sobre la
                 interdependencia positiva",
    "priority" : 3
},
{
    "moveId" : "rephrase01",
    "moveType" : "rephrase",
    "firstConcept" : "interdependencia positiva",
    "firstSynonyms" : ["dependencia mutua","dependencia"],
    "directed" : true,
    "message" : "{otherStudent}, puedes reformular con otras
                 palabras lo que {issuingStudent} acaba de
                 decir sobre la interdependencia positiva",
    "priority" : 1
},
{
    "moveId" : "buildOn01",
    "moveType" : "buildOn",
    "firstConcept" : "interdependencia positiva",
    "firstSynonyms" : ["dependencia mutua","dependencia"],
    "secondConcept" : "jigsaw",
    "secondSynonyms" : ["puzle","puzzle"],
    "directed" : false,
    "message" : "Creeis que el jigsaw promueve la
                 interdependencia positiva",
    "priority" : 6
},
{
    "moveId" : "agreeDisagree01",
    "moveType" : "agreeDisagree",
    "firstConcept" : "responsabilidad individual",
    "firstSynonyms" : ["responsabilidad","dar cuentas"],
    "directed" : true,
    "message" : "{otherStudent} Estas de acuerdo en que la
                 responsibilidad individual puede aumentar
                 las preocupaciones de una persona",
    "priority" : 5
},
{
    "moveId" : "verify01",
    "moveType" : "verify",
    "firstConcept" : "interdependencia positiva",
    "firstSynonyms" : ["dependencia mutua","dependencia"],
    "secondConcept" : "jigsaw",
    "secondSynonyms" : ["puzle","puzzle"],
```

```
            "directed" : true,
            "message" : "{otherStudent} Estas de acuerdo con la
                         siguiente afirmacion __El patron jigsaw
                         promueve la interdependencia positiva__",
            "priority" : 50
        },
        {
            "moveId" : "addOn02",
            "moveType" : "addOn",
            "firstConcept" : "responsabilidad individual",
            "firstSynonyms" : ["responsabilidad","dar cuentas"],
            "directed" : true,
            "message" : "{otherStudent}, puedes anadir algo a lo
                            que tu companero ha dicho sobre la
                            responsabilidad individual",
            "priority" : 7
        },
        {
            "moveId" : "revoice01",
            "moveType" : "revoice",
            "firstConcept" : "interdependencia positiva",
            "firstSynonyms" : ["dependencia mutua","dependencia"],
            "secondConcept" : "jigsaw",
            "secondSynonyms" : ["puzle","puzzle"],
            "directed" : false,
            "message" : "Veo que estais hablando del jigsaw. Os
                         recuerdo que el jigsaw consiste en
                         repartir la tarea entre los miembros del
                         grupo de tal manera que sea necesaria
                         la contribucion de todos para resolver
                         un problema, por lo que promueve la
                         interdependencia positiva",
            "priority" : 2
        }
    ]
}
}
```

CP25	**Creación de un agente conversacional con un *taskHash* ya existente en el sistema**
Proceso	Se realiza una petición POST al punto de acceso */courses/1/agents* con la información del agente conversacional *Beneficios del aprendizaje colaborativo - BNFCOLAB*, Código 6.1.
Resultado esperado	El sistema muestra un mensaje de error indicando que ya existe un agente conversacional con el mismo *taskHash*.

Tabla 6.25: CP25: Creación de un agente conversacional con un *taskHash* ya existente en el sistema

CP26	**Actualización de un agente conversacional**
Proceso	Se realiza una petición PUT al punto de acceso */courses/1/agents/BNFCOLAB* con la información del agente conversacional *Beneficios del aprendizaje colaborativo - BNFCOLAB*, Código 6.1.
Resultado esperado	El sistema muestra un mensaje indicando que el agente conversacional se ha modificado correctamente.

Tabla 6.26: CP26: Actualización de un agente conversacional

CP27	**Actualización de un agente conversacional con un *taskHash* ya existente en el sistema**
Proceso	Se realiza una petición PUT al punto de acceso */courses/1/agents/BNFCOLAB* con la información del agente conversacional *Beneficios del aprendizaje colaborativo - YAEXISTE*, Código 6.1.
Resultado esperado	El sistema muestra un mensaje de error indicando que ya existe un agente conversacional con el mismo *taskHash*.

Tabla 6.27: CP27: Actualización de un agente conversacional con un *taskHash* ya existente en el sistema

CP28	**Búsqueda de un agente conversacional**
Proceso	Se realiza una petición GET al punto de acceso */courses/1/agents/BNFCOLAB*.
Resultado esperado	El sistema muestra un mensaje indicando la información del agente conversacional.

Tabla 6.28: CP28: Búsqueda de un agente conversacional

CP29	**Búsqueda de un agente conversacional no registrado en el sistema**
Proceso	Se realiza una petición GET al punto de acceso */courses/1/agents/NOEXISTO*.
Resultado esperado	El sistema muestra un mensaje indicando que el agente conversacional no existe en el sistema.

Tabla 6.29: CP29: Búsqueda de un agente conversacional no registrado en el sistema

CP30	**Borrado de un agente conversacional**
Proceso	Se realiza una petición DELETE al punto de acceso */courses/1/agents/BNFCOLAB*.
Resultado esperado	El sistema muestra un mensaje indicando que el agente conversacional ha sido eliminado correctamente.

Tabla 6.30: CP30: Borrado de un agente conversacional

CP31	**Borrado de un agente conversacional no registrado en el curso**
Proceso	Se realiza una petición DELETE al punto de acceso */courses/1/agents/NOEXISTO*.
Resultado esperado	El sistema muestra un mensaje indicando que el agente conversacional no existe en el sistema.

Tabla 6.31: CP31: Borrado de un agente conversacional no registrado en el curso

CP32	**Activación de un agente conversacional**
Proceso	Se realiza una petición PUT al punto de acceso */courses/1/agents/BNFCOLAB/enable*.
Resultado esperado	El sistema muestra un mensaje indicando que el agente conversacional ha sido activado correctamente.

Tabla 6.32: CP32: Activación de un agente conversacional

CP33	**Activación de un agente conversacional activo**
Proceso	Se realiza una petición PUT al punto de acceso */courses/1/agents/BNFCOLAB/enable*.
Resultado esperado	El sistema muestra un mensaje indicando que el agente conversacional ya se encuentra activo.

Tabla 6.33: CP33: Activación de un agente conversacional activo

CP34	**Desactivación de un agente conversacional**
Proceso	Se realiza una petición PUT al punto de acceso */courses/1/agents/BNFCOLAB/disable*.
Resultado esperado	El sistema muestra un mensaje indicando que el agente conversacional ha sido desactivado correctamente.

Tabla 6.34: CP34: Desactivación de un agente conversacional

CP35	**Desactivación de un agente conversacional desactivado**
Proceso	Se realiza una petición PUT al punto de acceso */courses/1/agents/BNFCOLAB/disable*.
Resultado esperado	El sistema muestra un mensaje indicando que el agente conversacional ya está desactivado.

Tabla 6.35: CP35: Desactivación de un agente conversacional desactivado

6.1.5. Búsqueda de un estudiante

CP36	**Búsqueda de un estudiante**
Proceso	Se realiza una petición GET al punto de acceso */students/nat*.
Resultado esperado	El sistema muestra un mensaje indicando la información del estudiante.

Tabla 6.36: CP36: Búsqueda de un estudiante

CP37	**Búsqueda de un estudiante no registrado en el sistema**
Proceso	Se realiza una petición GET al punto de acceso */students/noExisto*.
Resultado esperado	El sistema muestra un mensaje indicando que el estudiante no existe en el sistema.

Tabla 6.37: CP37: Búsqueda de un estudiante no registrado en el sistema

6.1.6. Búsqueda de una sala de chat

CP38	**Búsqueda de una sala de chat**
Proceso	Se realiza una petición GET al punto de acceso */courses/1/BNFCOLAB/chatrooms/1*.
Resultado esperado	El sistema muestra un mensaje indicando la información de la sala de chat.

Tabla 6.38: CP38: Búsqueda de una sala de chat

CP39	**Búsqueda de una sala de chat no registrada en el sistema**
Proceso	Se realiza una petición GET al punto de acceso */courses/1/BNFCOLAB/chatrooms/1128391*.
Resultado esperado	El sistema muestra un mensaje indicando que la sala de chat no existe en el sistema.

Tabla 6.39: CP39: Búsqueda de una sala de chat no registrada en el sistema

6.1.7. Funcionalidades de cursos y agentes conversacionales

Las pruebas de las funcionalidades de cursos y agente conversacionales se realizaron desde la propia aplicación de Telegram. Estas pruebas requerían de una mayor dificultad, dado que se necesitaba la participación de dos personas.

CP40	**Unión de un estudiante a una sala de chat con un agente conversacional**
Proceso	Se envía el mensaje */UNIRME BNFCOLAB* en el curso *Fundamentos de Agente Conversacionales*.
Resultado esperado	El sistema envía un mensaje de bienvenida en la correspondiente sala de chat.

Tabla 6.40: CP40: Unión de un estudiante a una sala de chat con un agente conversacional

CP41	**Unión de un estudiante a una sala de chat con un agente conversacional no registrado en el sistema**
Proceso	Se envía el mensaje */UNIRME NOEXISTE* en el curso *Fundamentos de Agente Conversacionales*.
Resultado esperado	El sistema elimina el mensaje enviado por el usuario.

Tabla 6.41: CP41: Unión de un estudiante a una sala de chat con un agente conversacional no registrado en el sistema

CP42	Salida de un estudiante de una sala de chat
Proceso	Se abandona la sala de chat con el agente conversacional *Beneficios del aprendizaje colaborativo - BNFCOLAB*.
Resultado esperado	El sistema registra la información correspondiente a la salida del estudiante.

Tabla 6.42: CP42: Salida de un estudiante de una sala de chat

CP43	Intervención de un estudiante en una sala de chat
Proceso	Se envía el mensaje *La interdependencia positiva es uno de los beneficios* en la sala de chat *Beneficios del aprendizaje colaborativo - BNFCOLAB*.
Resultado esperado	El sistema responde a la intervención con el mensaje *Puedes añadir algo a lo que tu compañero/a ha dicho sobre la interdependencia positiva*.

Tabla 6.43: CP43: Intervención de un estudiante en una sala de chat

Para comprobar el correcto funcionamiento de todos los patrones configurados en un agente conversacional, las pruebas realizadas se han realizado del mismo modo que figura en la Tabla 6.43.

6.2. Prueba de usabilidad con usuarios reales

Se realizó una prueba de usabilidad con usuarios reales, con el fin de realizar una evaluación formativa del sistema. En especial, se pretendía encontrar dificultades o problemas que puedan surgir a estudiantes a la hora de interactuar con el agente conversacional. En la prueba participaron doce voluntarios del grupo de investigación GSIC/EMIC, utilizando la última versión disponible del sistema.

Tras la realización de la prueba, se obtuvo *feedback* de los participantes sobre las dificultades que tuvieron. Cabe destacar los siguientes problemas:

1. Falta de explicaciones y aclaraciones sobre el funcionamiento del agente conversacional: ciertos usuarios no sabían como interactuar con el agente porque pensaban que tenían que introducir algún comando para que este interviniera.

2. Visualización escasa de la tarea a realizar: la tarea principal era difícil de localizar dada la cantidad de mensajes que el agente envía al principio. La tarea queda anclada globalmente en la sala de chat, pero varios participantes no se percataron de ello.

3. Falta de mensajes de error: cuando tiene lugar un error o el sistema cae los usuarios no son notificados y no tienen conocimiento de lo que ha ocurrido.

El primer error podría ser solucionado mejorando la estructura de los mensajes enviados por el agente conversacional y añadiendo más explicaciones. Para solucionar la

mala visualización de la tarea, se debería utilizar caligrafía en cursiva y en negrita para clarificar donde se encuentra el enunciada de la tarea. Este problema no depende del software, sino de como se configuran los agentes conversacionales, por lo que puede ser resuelto dando recomendaciones a los profesores. Habría que incorporar mensajes globales de error para que los usuarios sean conscientes de si el sistema está funcionando correctamente.

Además, durante esta prueba se detectaron ciertas limitaciones de Telegram. La API de Telegram prohíbe realizar la misma petición, con los mismos parámetros, en un tiempo reducido y si se detecta dicho evento, se bloquean las acciones con la API durante una determinada duración. Por lo tanto, habría que modificar ciertas funcionalidades del sistema para evitar que tenga lugar dicho error y, así, mejorar la escalabilidad del sistema.

Capítulo 7

Conclusión

7.1. Conclusiones

En el desarrollo de este TFG, se han conseguido desarrollar las funcionalidades básicas de un sistema gestor de agentes conversacionales configurables de propósito educativo. Los objetivos iniciales han sido cumplidos y para cada uno de ellos se han completado varios hitos:

1. Desarrollar el *backend* de un sistema web para gestionar una plataforma de agentes conversacionales configurables:

    - Implementación de una Rest API con Flask que permite la gestión y configurabilidad de profesores, cursos y agentes conversacionales.
    - Implementación de un sistema de seguridad para la Rest API basado en JSON Web Token.
    - Implementación de las funcionalidades para soportar múltiples idiomas a partir de ficheros JSON.

2. Desarrollar el software encargado del funcionamiento de los agentes conversacionales

    - Implementación de las funcionalidades que permiten a un estudiante unirse a una sala de chat con un agente conversacional desde un curso.
    - Implementación de las funcionalidades de los agentes conversacionales con el objetivo de desencadenar los diferentes patrones, en función de los eventos que tienen lugar durante la conversación.
    - Implementación de un sistema de registros encargado de almacenar todos los eventos ocurridos en una conversación.

3. Integrar las funcionalidades de los agentes conversacionales con una aplicación de chats de código abierto:

    - Integración de Telegram como medio en el que se desarrollan las conversaciones entre estudiantes y agentes conversacionales.

7.2. Trabajo futuro

El sistema gestor de agentes conversacionales configurables de propósito educativo puede ampliar sus funcionalidades y mejorar las mismas. Algunas de estas mejoras se enuncian a continuación:

- Implementación de un *frontend* que permita las gestión y configuración de profesores, cursos y agentes conversacionales desde una interfaz de usuario.
- Corrección y mejora de los fallos encontrados tras la realización de la prueba de usabilidad.
- Actualización de las funcionalidades que permiten a un estudiante unirse a una sala de chat para mejorar la escalabilidad.
- Adición de funcionalidades a los agentes conversacionales.
- Integración de diferentes aplicaciones de chats como Discord, WhatsApp, etc.

7.3. Valoración personal

El desarrollo de este proyecto me ha servido para obtener una mayor confianza en mi mismo y una mayor madurez profesional. La realización de un sistema software desde cero requiere de un gran un esfuerzo, el cual es gratificado al ver los buenos resultados del mismo.

Además, he conseguido ampliar mis conocimientos y mejorar los que ya tenía. En mi opinión, el sistema que he desarrollado podría mejorarse de muchas maneras pero, de esto trata el aprendizaje, un proceso continuo de mejora.

Bibliografía

[1] M. Winkler R. & Söllner. *Unleashing the Potential of Chatbots in Education: A State-Of-The-Art Analysis. s. In Academy of Management Annual Meeting (AOM)*. 2018. URL: https://www.alexandria.unisg.ch/254848/1/JML_699.pdf.

[2] Euroforum. *Chatbots Top Star*. 2018. URL: https://www.euroforum.es/blog/chatbots-top-star/#Cuales_son_los_chatbots_mas_top. (último acceso: Junio 2020).

[3] *The colMOOC: Integrating Conversational Agents and Learning Analytics in MOOCs, Comisión de la Unión Europea, ERASMUS+ KA2-Cooperation and innovation for good practices. Knowledge Alliances Call (588438-EPP-1-2017-1-EL-EPPKA2-K)*. 2018-2020. URL: https://colmooc.eu/. (último acceso: Julio 2020).

[4] *¿Qué es un Mooc?* 2018. URL: https://mooc.es/que-es-un-mooc/. (último acceso: Junio 2020).

[5] Stergios Tegos, Stavros Demetriadis y Anastasios Karakostas. "Promoting academically productive talk with conversational agent interventions in collaborative learning settings". En: *Computers & Education* 87 (2015), págs. 309-325. ISSN: 0360-1315. DOI: https://doi.org/10.1016/j.compedu.2015.07.014. URL: http://www.sciencedirect.com/science/article/pii/S0360131515300191.

[6] ASPgems. *Metodología de desarrollo de software (II) – Modelo de diseño iterativo*. 2019. URL: https://aspgems.com/metodologia-de-desarrollo-de-software-ii-modelo-de-diseno-iterativo/. (último acceso: Junio 2020).

[7] Mike Cotterell Bob Hughes. *Software Project Management Fifth Edition*. 2011.

[8] Nueva ISO 45001. *SG-SST: Análisis y evaluación de riesgos*. 2016. URL: https://www.nueva-iso-45001.com/2016/09/sg-sst-analisis-evaluacion-riesgos/. (último acceso: Julio 2020).

[9] Telegram. *Mobile Protocol: Detailed Description*. 2020. URL: https://core.telegram.org/mtproto/description. (último acceso: Junio 2020).

[10] Mercedes Martínez. *Tecnología y diseño de bases de datos - Diseño lógico y físico. Departamento de Informática. Universidad de Valladolid*. 2018.

[11] BBVA. *API REST: qué es y cuáles son sus ventajas en el desarrollo de proyectos*. 2016. URL: https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos. (último acceso: Junio 2020).

[12] Wikipedia. *JSON Web Token*. 2020. URL: https://es.wikipedia.org/wiki/JSON_Web_Token. (último acceso: Junio 2020).

[13] Konstantin Taletskiy. *Pool Limited Queue Processing in Python*. 2020. URL: https://towardsdatascience.com/pool-limited-queue-processing-in-python-2d0255. (último acceso: Junio 2020).

[14] Lucidchart. *¿Qué es un diagrama de flujo?* 2020. URL: https://www.lucidchart.com/pages/es/que-es-un-diagrama-de-flujo. (último acceso: Junio 2020).

[15] Daniel Maldonado. *Ramas en Git*. 2014. URL: https://elcodigok.blogspot.com/2014/03/ramas-en-git.html. (último acceso: Junio 2020).

[16] EDteam. *¿Cómo se deciden las versiones del software?* 2017. URL: https://ed.team/blog/como-se-deciden-las-versiones-del-software. (último acceso: Junio 2020).

[17] Jason C. McDonald. *Dead Simple Python: Project Structure and Imports*. 2020. URL: https://dev.to/codemouse92/dead-simple-python-project-structure-and-imports-38c6. (último acceso: Abril 2020).

Anexo A

Manual de instalación

A.1. Requisitos

- Sistema gestor de base de datos: MySQL
- Lenguaje de programación: Python versión $>$3.6
- Instalador de paquetes de python: pip3
- Entorno virtual para python3: virtualenv

A.2. Proceso de instalación

Este proceso de instalación ha sido desarrollado en un Ubuntu Server. En función del sistema operativo o la distribución del mismo, las primitivas para instalar paquetes o incluso el procedimiento podría variar. Versión de Ubuntu 20.4 LTS

A.3. Instalación de requisitos previos

1. Instalación de MySQL:
   a) sudo apt-get install mysql-server
   b) sudo mysql_secure_installation
2. Instalación de Python3: sudo apt-get install python3.6
3. Instalación de pip3 y virtualenv:
   a) sudo apt-get install python3-pip
   b) sudo apt-get install python3-virtualenv

A.4. Configuración del proyecto

1. Depositar la carpeta con el código fuente en la ubicación deseada.

2. Dentro de la carpeta con el código fuente, crear el entorno virtual: virtualenv nombre_del_entorno_virtual
3. Activar el entorno virtual: source nombre_del_entorno_virtual/bin/activate
4. Para desactivar el entorno virtual: deactivate
5. Ejecutar el archivo install.py para instalar todos los paquetes necesarios de python en el entorno virtual: python3 conversational_agent/install.py
6. Crear la carpeta logs en el directorio conversational_agent: mkdir logs

A.5. Configuración de la base de datos

1. Crear esquema de base de datos:
   a) Crear base de datos: create database database_name
   b) Ejecutar el script de creación de base de datos ubicado en conversational_agent/database/create_schema.sql. (!Importante) Dentro de la base de datos (use database_name) ejecutar: source path_to_project/conversational_agent/database/create_schema.sql
2. Crear un usuario en MySQL:
   a) create user user@localhost identified by 'password';
   b) grant all privileges on databaseName. to user@localhost;

A.6. Configuración de Tokens

El proyecto conversational agents manager necesita estar rellenado correctamente para que la aplicación funcione. En este apartado, se explica la información necesaria que debe contener ese archivo. El archivo tokens.py tiene la siguiente estructura:

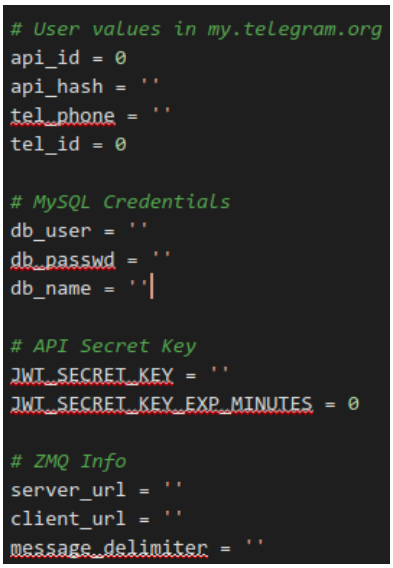


Figura A.1: Tokens

A.6.1. Información de Aplicación en Telegram

La información de aplicación necesaria en Telegram es la siguiente:

- API Id y API Hash: se obtienen al crear un proyecto en la plataforma de desarrolladores de Telegram (Creating your Telegram Application).
- Teléfono asociado a la cuenta de Telegram: comenzando por +CodigoPais. Ej +34666666666
- Identificador de usuario de Telegram: este identificador es muy importante para el correcto funcionamiento del software. Su obtención es algo complicada, bastaría con ejecutar el siguiente código (python3 nombre_del_script.py):

```
from telethon import TelegramClient
from telethon.tl.types import PeerUser
import asyncio
# Use your own values from my.telegram.org
api_id = 0
api_hash = ''

loop = asyncio.get_event_loop()
async def main():
    async with TelegramClient('anon', api_id, api_hash) as client:
        me = await client.get_entity('me')
        print(me.id)

loop.run_until_complete(main())
```

Figura A.2: Obtención del identificador de usuario en Telegram

A.6.2. Información MySQL

La información necesaria para que MySQL funcione correctamente es la siguiente:

- Nombre y contraseña de usuario en el sistema gestor MySQL
- Nombre de la base de datos

A.6.3. Información Json Web Token

La información necesaria para garantizar la autenticación en la Rest API es la siguiente:

- La clave secreta, la cual se compone de caracteres alfanuméricos, sirve como base para la encriptación usada por JWT.
- Tiempo, en minutos, en el que el token generado después del login, es válido.

A.6.4. Información de la Cola de Mensajes ZMQ:

La cola de mensajes necesita la siguiente información:

- Dirección donde se va a encontrar tanto cliente como servidor.
- Delimitador de mensajes.

Por defecto, zmq suele usar el puerto 5555. Utilizando la siguiente configuración por defecto, no se ha observado ningún problema.

```
# ZMQ Info
server_url = 'tcp://*:5555'
client_url = 'tcp://localhost:5555'
message_delimiter = '~~~~'
```

Figura A.3: Configuración por defecto ZMQ

A.7. Ejecución del servidor

Dentro del directorio con el código fuente ejecutar: python3 -m conversational_agent

A.8. Configuración de ficheros de sesión

En el proyecto se usan cuatro ficheros de sesión, de los cuales tres son creados automáticamente al inicio del servidor. Los tres primeros serán generados al ejecutar el script anterior, que pide tres veces el número de teléfono y manda a la cuenta de Telegram tres códigos de verificación (es uno distinto para cada sesión, pero no está claro cuál es cuál, por lo que hay que ir probando hasta acertar con las tres sesiones.

El cuarto fichero de sesión será será creado al realizar la primera petición sobre la API Rest (que no sea el login), y también pedirá que se introduzca el número de teléfono y posteriormente el código de verificación enviado a la cuenta de Telegram. ATENCIÓN: el número de teléfono y el código de verificación lo pide en la consola desde la que se ha ejecutado el servidor que, en esta ocasión, debe correr en primer plano.

Este proceso sólo es necesario realizarlo una vez. A partir de ese momento el servidor puede ejecutarse en segundo plano. Si se desea cambiar de ubicación el proyecto o actualizar la versión del mismo, se pueden copiar estos ficheros de sesión para así no realizar el proceso anterior.