



Escuela Técnica Superior de Ingeniería Informática

Ingeniería Informática: Ingeniería del Software

TRABAJO FIN DE GRADO

Gestión inteligente de la conformidad en procesos mediante flujos en Node-RED

Autor/es:
Álvaro Jesús Bernal Caunedo

Tutor/es:
Adela del Río Ortega
Manuel Resinas Arias de Reyna

Primera convocatoria

Curso 2023-2024

Calificación Obtenida: 9,8

Agradecimientos

Qué mejor forma de cerrar esta etapa que agradecer el apoyo y la ayuda recibidos por las personas mencionadas a continuación durante estos años de titulación.

A mis padres, por haberme inculcado valores, principios y libertades que hacen de mí la persona que soy hoy y darme cariño y apoyo incondicional.

A Adela del Río y Manuel Resinas, mis tutores y referentes en el mundo académico. Sin sus consejos, este trabajo no hubiera terminado de forma correcta y dentro de plazo. Gracias por ser pacientes, amables y responder a todas y cada unas de mis preguntas (que no fueron pocas).

A mis compañeros Álvaro González, Nicolás Herrera, Javier García y Antonio Rodríguez por hacerme reflexionar sobre el trabajo realizado y así sacar una mejor versión del mismo. Sin ellos, esta etapa sería completamente diferente y no me habría convertido en una persona curiosa y con ganas de aprender.

A mis compañeros José María García y Ronald Montoya, por ayudarme a desconectar en momentos de estrés dentro de la universidad y por brindarme ideas ingeniosas que nunca se me hubiesen ocurrido.

A todos ellos les debo, en mayor o menor medida, quién soy y cómo soy. Muchas gracias.

Resumen

En un mundo donde las empresas deben seguir una serie de políticas y estándares estrictos definidos por organismos regulatorios, es importante medir el cumplimiento normativo. Una práctica utilizada para medir dicho cumplimiento es la gestión de la conformidad de esa serie de procesos de negocio. Sin embargo, gestionar el cumplimiento de estos de procesos es algo complejo que requiere de conocimiento técnico sobre herramientas y tecnologías.

Este trabajo comienza con la contratación para participar en un proyecto de investigación dirigido por Cabanillas y cols., enfocado en desarrollar un sistema capaz de gestionar la conformidad de procesos de negocio. Esta propuesta definía un sistema de gestión de la conformidad basado en mashups (modelos que recopilan la información de cumplimiento de un proceso de negocio) capaz de facilitar a las empresas la verificación del cumplimiento normativo.

El objetivo del proyecto es ofrecer a los usuarios un sistema accesible que simplifique la gestión de la conformidad de los procesos de negocio, eliminando la necesidad de que los usuarios posean conocimientos técnicos sobre las herramientas utilizadas para esta gestión. Para ello, se desarrolló mediante el servicio de OpenAI un chatbot encargado de agilizar el diseño de mashups en la herramienta Node-RED, la cual se utilizó para recopilar la información de los procesos de negocio.

Para la realización de este proyecto, se emplearon ExpressJS como framework de backend junto con Node.js y React en el frontend, herramientas utilizadas en este proyecto de investigación. Finalmente, se desarrolló un asistente capaz de generar mashups a partir de una descripción de los mismos, facilitando a los usuarios la creación y descripción de mashups desde el sistema.

Palabras clave: Cumplimiento normativo, Chatbot, Procesos de negocio, Mashups, Inteligencia artificial.

Abstract

In a world where companies must follow a strict set of policies and standards defined by regulatory bodies, it is important to measure compliance. One practice used to measure compliance is compliance management of these business processes. However, managing the compliance of these processes is complex and requires technical knowledge of tools and technologies.

This work begins with the recruitment to participate in a research project led by Cabanillas y cols., focused on developing a system capable of managing the compliance of business processes. This proposal defined a compliance management system based on mashups (models that compile the compliance information of a business process) capable of making it easier for companies to verify regulatory compliance.

The aim of the project is to offer users an accessible system that simplifies the management of business process compliance, eliminating the need for users to have technical knowledge of the tools used for this management. To this end, a chatbot was developed using the OpenAI service to speed up the design of mashups in the Node-RED tool, which was used to collect information on business processes.

For the realisation of this project, ExpressJS was used as a backend framework together with Node.js and React on the frontend, tools used in this research project. Finally, a wizard was developed capable of generating mashups from a description of the mashups, making it easier for users to create and describe mashups from the system.

Index Terms: Business Compliance, Chatbot, Business processes, Mashups, Artificial intelligence.

Índice general

1. Introducción	1
1.1. Contexto	1
1.2. Motivación	2
1.3. Contribución	3
1.4. Base conceptual	3
1.5. Objetivos del Proyecto	4
1.6. Estructura del documento	5
2. Alcance del Proyecto y del Producto	6
2.1. Descripción del Alcance	6
2.2. Lista de Entregables	6
2.3. Restricciones	7
3. Especificación de Requisitos	8
3.1. Historias de Usuario	8
3.2. Criterios de Aceptación del Producto	18
3.3. Atributos de Calidad	19
3.4. Requisitos de Integración del Sistema	20
4. Metodología de Trabajo	22
4.1. Metodología de Gestión del Proyecto	22
4.2. Metodología de Desarrollo	22
5. Estructura de Desglose del Trabajo (EDT) y Diccionario	25
5.1. Fase de Ejecución	25
5.2. Fase de Cierre	35
6. Estimaciones de Tiempo, Costes y Cronograma	37
6.1. Estimaciones de Tiempo	38
6.2. Estimaciones de Costes	42
6.3. Cronograma	46
7. Diseño y Arquitectura del Producto	48
7.1. Diseño y Arquitectura	48
7.2. Ventajas e Inconvenientes de la Arquitectura Escogida	51
8. Tecnologías y herramientas	53
8.1. Análisis Tecnológico	53
8.2. Análisis de las Alternativas	56
8.3. Stack Tecnológico	61
9. Desarrollo del producto	64
9.1. Sprint 1	64
9.2. Sprint 2	66
9.3. Sprint 3	68
9.4. Sprint 4	70
10. Análisis del Sistema	74
10.1. Análisis de la Arquitectura	74

10.2. Análisis de los Nodos Realizados	78
10.3. Análisis del Chatbot	81
10.4. Integración con Otros Sistemas	88
10.5. Interfaz y Experiencia de Usuario	90
11. Evaluación y Resultados	93
11.1. Análisis de la Calidad del Código	93
11.2. Pruebas realizadas	94
11.3. Contribuciones a otros proyectos	98
11.4. Análisis de la Planificación	100
11.5. Coste Total de la Propiedad	104
11.6. Coste Final del Proyecto	109
12. Conclusiones	111
12.1. Valor Añadido	111
12.2. Errores Cometidos	111
12.3. Lecciones Aprendidas	112
12.4. Trabajo Futuro	112
13. Anexos	114
13.1. Términos, Definiciones y Acrónimos	114
13.2. Manual de Despliegue	115
13.3. Manual de Usuario	117
13.4. Manual de Pruebas	129
13.5. Informe de Uso de la IA	130
13.6. Documentación online del TFG	132
14. Referencias	133

Índice de cuadros

3.1. Objetivos / Historias de usuario	18
5.1. [PT 1.1.1] Formación	26
5.2. [PT 1.1.2] Configuración de entornos	27
5.3. [PT 1.1.3] Traducción de nodos	27
5.4. [PT 1.1.4] Creación inicial de nodos	28
5.5. [PT 1.1.5] Seguimiento y control	28
5.6. [PT 1.2.1] Empaquetado de nodos	29
5.7. [PT 1.2.2] Integración continua	29
5.8. [PT 1.2.3] Perfil de usuario	30
5.9. [PT 1.2.4] Estructura principal de la memoria	30
5.10. [PT 1.2.5] Seguimiento y control	31
5.11. [PT 1.3.1] Versión inicial del chatbot	31
5.12. [PT 1.3.2] Gestión de <i>mashups</i>	32
5.13. [PT 1.3.3] Mejora de la calidad del código	32
5.14. [PT 1.3.4] Pruebas	33
5.15. [PT 1.3.5] Seguimiento y control	33
5.16. [PT 1.4.1] Interfaz de usuario	34
5.17. [PT 1.4.2] Versión final del chatbot	34
5.18. [PT 1.4.3] Gestión del chatbot	34
5.19. [PT 1.4.4] Versión completa de la memoria	35
5.20. [PT 1.4.5] Seguimiento y control	35
5.21. [PT 2.1.1] Preparación de la entrega	36
6.1. Estimación de tiempo de actividades del Sprint 1	38
6.2. Estimación de tiempo de actividades del Sprint 2	39
6.3. Estimación de tiempo de actividades Sprint 3	40
6.4. Estimación de tiempo de actividades del Sprint 4	41
6.5. Estimación de tiempo de actividades del Cierre	42
6.6. Estimación de costes de las actividades	44
6.7. Desglose del coste de personal desde el punto de vista de la Empresa	44
6.8. Comparación de precios de diferentes servicios de NLP	45
6.9. Comparación para 200 transacciones mensuales	46
6.10. Coste total planificado	46
8.1. Comparación de Servicios de NLP	58
8.2. Comparativa de Generadores de Páginas Estáticas	61
10.1. Traducción de nodos de EA a Node-RED	78
11.1. Calidad de código de Front-end y Back-end	93
11.2. Actividades del Sprint 1	100
11.3. Actividades del Sprint 2	101
11.4. Actividades del Sprint 3	102
11.5. Estimación de tiempo de actividades del Sprint 4	103
11.6. Actividades del Cierre	104
11.7. Tiempo estimado y real por paquete de planificación	104
11.8. Costes estimados y reales por paquete de planificación	105
11.9. Desglose final del coste de personal	105

11.10	Costes estimados de procesamiento de tokens	108
11.11	Servicio de soporte al cliente	108
11.12	Costes mensuales del SLA	109
11.13	Desglose del TCO bienal	109
11.14	Comparación de costes reales y estimados	110

Índice de figuras

1.1. Comprobación de la existencia de una sección (Fuente: <i>Enterprise Architect</i> del proyecto “Procede”)	3
3.1. Vista de gestión del chatbot	9
3.2. Modal de activación/desactivación del chat	9
3.3. Editor de Node-RED	10
3.4. Listado de mashups	11
3.5. Aviso de mal funcionamiento de Node-Red	11
3.6. Generación de mashups	12
3.7. Conversación con datos insuficientes	13
3.8. Recomendación de descripción	14
3.9. Detalles del mashup	14
3.10. Mashups no disponibles	15
3.11. Conexión con otras herramientas	16
3.12. Inicio de sesión	18
5.1. Estructura de Desglose del Trabajo (EDT)	25
6.1. Cronograma de alto nivel	47
6.2. Cuentas de control	47
7.1. Diagrama de arquitectura lógica	48
7.2. Diagrama de clases UML	49
7.3. Flujo de funcionamiento principal del chatbot	51
8.1. Diagrama de contexto	62
9.1. Nodos de Procede	66
9.2. Vista de inicio de sesión	67
9.3. Respuesta de prueba del <i>Playground</i>	69
9.4. Administración del asistente	71
9.5. Documentación de la API	72
10.1. Infraestructura de STATUS	74
10.2. Arquitectura de la capa de lógica de negocio	75
10.3. Servicios del sistema STATUS	76
10.4. Diagrama de clases final	77
10.5. Subflujos de Node-RED (Fuente: Documentación de Node-RED ¹)	79
10.6. Nodos finales con iconos ilustrativos	79
10.7. Diagrama de clases	83
10.8. Descripción de <i>mashups</i>	84
10.9. Generación de <i>mashups</i> a partir de una descripción	85
10.10. Información del mashup	86
10.11. Modal de detalles del mashup	86
10.12. Activación/Desactivación del asistente	87
10.13. Conexión con Github y Trello	88
10.14. Modal para introducir el token de trello	89
10.15. Comunicación entre el sistema y node-RED	89

¹<https://nodered.org/docs/user-guide/editor/workspace/subflows>

10.16	Inicio de sesión en el editor	90
10.17	<i>Skeleton loader</i> de mensajes del chatbot	91
10.18	Estilo del mensaje en formato Markdown	92
10.19	Botones de conexión con Trello y Github	92
11.1.	Incidencias de código en Front-end	93
11.2.	Informe de pruebas de API de Postman	95
11.3.	Listado de <i>suites de tests</i>	95
11.4.	Pruebas unitarias de Node-RED	96
11.5.	Prueba de prompt mediante Postman	97
11.6.	Gráfica de rendimiento del sistema	97
11.7.	Tabla de rendimiento de la caché	98
11.8.	Contribución publicada en la página de Node-RED	99
11.9.	Tarifa de AppEngine	107
11.10	Tarifas de instancias de Amazon EC2	107
13.1.	Vista de inicio de sesión	117
13.2.	Credenciales de usuarios no válidas	118
13.3.	Editor no desplegado	118
13.4.	Inicio de sesión en Node-RED	119
13.5.	Editor de Node-RED	119
13.6.	Perfil del usuario	120
13.7.	Modal de conexión con Trello	120
13.8.	Listado de Mashups	121
13.9.	Modal de creación de mashups	121
13.10	Detalles del mashup	122
13.11	Enter Caption	122
13.12	Chat sin conversaciones existentes	123
13.13	Descripción y Detalle de un mashup	124
13.14	Modal de edición de nombre del hilo	124
13.15	Descripción demasiado corta	125
13.16	Mashup autogenerado	125
13.17	Eliminación del mashup autogenerado	126
13.18	Vista inicial del administrador de OpenAI	126
13.19	Instrucciones de un asistente	127
13.20	Modal de creación de asistentes	127
13.21	Alerta de campos vacíos	128
13.22	Confirmación de eliminación de asistentes	128
13.23	Edición del límite de asistentes	129
13.24	Límite inválido	129
13.25	Página de inicio de la documentación	132

Índice de extractos de código

10.1. Ejemplo básico de asistente	82
10.2. Ejemplo básico de flujo de Node-RED	85
11.1. Modificaciones para arquitectura ARM64	99

1. Introducción

1.1. Contexto

Debido al manejo de información confidencial y al tratamiento de datos personales, las organizaciones están obligadas a seguir estándares complejos como las normas ISO (“International Organization for Standardization”), la ley HIPAA (“Health Insurance Portability and Accountability Act”) o el Reglamento General de Protección de Datos (RGPD). La verificación del cumplimiento de estos estándares puede llegar a ser un reto para las organizaciones, especialmente en procesos que no siguen un modelo estructurado de proceso de negocio. Por ejemplo, la solicitud y obtención de licencias municipales para la apertura de nuevos negocios; donde la diversidad de requisitos, la coordinación entre entidades y la variabilidad según la ubicación representan desafíos significativos para su modelado. La coordinación efectiva entre equipos de una misma organización es esencial para garantizar que los procesos de negocio cumplan con los estándares establecidos y para evitar posibles brechas de conformidad que puedan resultar en sanciones legales o daños a la reputación de la organización.

Otro escenario en el que las organizaciones pueden necesitar verificar el cumplimiento de los procesos de negocio es para garantizar el cumplimiento de las políticas internas de la organización. Estas políticas pueden abarcar una amplia gama de áreas, como la seguridad de la información, la protección de la privacidad de los empleados y clientes, la gestión de riesgos, el control interno, entre otros aspectos clave para el funcionamiento adecuado y ético de la empresa.

Para abordar estos desafíos, las organizaciones implementan diversos mecanismos y herramientas, tales como el Control de Conformidad, que se refiere a las medidas, reglas o procedimientos establecidos para asegurar el cumplimiento con los requisitos legales, normativos o internos aplicables. Por ejemplo, una revisión mensual de los registros financieros por parte de un auditor externo para garantizar la transparencia y precisión de la información contable de una empresa.

Un recurso fundamental en este proceso es el Catálogo de Controles, una lista estructurada y detallada de los controles de conformidad que la organización debe cumplir. Este catálogo identifica y describe cada control específico, proporcionando información sobre su propósito, alcance, responsabilidades asociadas y criterios de cumplimiento. Por ejemplo, podría incluir controles relacionados con la protección de datos, la seguridad informática y la gestión de riesgos.

Además, existen diversos tipos de Comprobación de Conformidad que se refieren a las diferentes modalidades de evaluación y verificación del cumplimiento de los controles establecidos. Estas modalidades incluyen la comprobación de conformidad en tiempo de diseño (*Design-Time Compliance Checking*), que se realiza antes de la ejecución de los procesos para asegurar que cumplen con las reglas establecidas, y la comprobación de conformidad en tiempo de ejecución (*Run-Time Compliance Checking*), que se lleva a cabo durante la ejecución para monitorear y corregir posibles desviaciones en tiempo real.

Para facilitar y optimizar estos procesos, las organizaciones pueden utilizar un sistema de gestión de la conformidad, una plataforma o herramienta que automatiza la gestión y comprobación de los catálogos de controles. Este sistema permite la definición, implementación, monitoreo y reporte de los controles de conformidad, logrando una gestión más eficiente y efectiva de la conformidad normativa.

En consonancia con la importancia de garantizar la conformidad en los procesos de negocio, diversos estudios y artículos han abordado las diferentes fases de verificación dentro del ciclo de vida de estos procesos. En particular, mucha de esta información se detalla en el artículo “A

Mashup-Based Framework for Business Process Compliance Checking” escrito por Cabanillas y cols. (2020), que servirá como base principal para este Trabajo de Fin de Grado. En este artículo se define una estructura basada en *mashups* para comprobar la conformidad de procesos de negocio de forma anticipada (*forward compliance checking*). Estos *mashups* son flujos de trabajo basados en datos de diferentes fuentes que tienen como objetivo especificar reglas y hacer comprobaciones de conformidad de procesos de negocio en tiempo de diseño y de ejecución.

Los procesos de negocio se someten a una serie de controles, que consisten en conjuntos de reglas o condiciones que deben seguirse para verificar su cumplimiento, estos se evalúan mediante los *compliance mashups*. Para ilustrar cómo un *mashup* podría ser útil para verificar un control específico en el contexto de los procesos de negocio, consideremos el siguiente ejemplo: “Dentro de los Proyectos de Software, uno de los controles críticos es que el análisis funcional debe ser aprobado por el Business Sponsor”. A partir de estas comprobaciones de conformidad basadas en *mashups* (MCC), se construyó un sistema de gestión de conformidad de procesos de negocio (BPCMIS) para un departamento de TI de una empresa multinacional del ámbito del suministro energético como proyecto de I+D, este proyecto fue denominado “Procede”. En este proyecto se diseñaron una serie de componentes genéricos que se utilizaban para formar los *compliance mashups*.

La definición de *mashups* para la comprobación de conformidad puede ser un proceso complicado debido a la necesidad de integrar diversas fuentes de datos y coordinar múltiples reglas y controles. La especificación de estos *mashups* requiere un conocimiento profundo tanto de los procesos de negocio como de las normativas aplicables, además de habilidades técnicas para manejar herramientas y plataformas de desarrollo.

En base a los estudios mencionados, el ministerio concedió a la profesora e investigadora Cristina Cabanillas Macías (2012, 2020) un proyecto de prueba de concepto, encargado de desarrollar un sistema de gestión de conformidad basado en *mashups*. Para llevar a cabo este proyecto, se contrataron dos alumnos de la Escuela Técnica Superior de Ingeniería Informática (ETSII), Francisco Jesús Montero Martínez y yo, Álvaro Jesús Bernal Caunedo. Este trabajo de fin de grado se enmarca en ese proyecto, abordando la complejidad de definir estos *mashups* y proponer soluciones que ayuden a reducir dicha complejidad.

1.2. Motivación

A continuación se mencionan algunos de los motivos académicos, personales y prácticos que me han ayudado a escoger este tema:

- Reducir la complejidad de uso: Además del modelado de procesos de negocio, la creación de *mashups* para medir el cumplimiento de los procesos puede ser algo complejo si no se tienen conocimientos técnicos sobre el tema en cuestión. Si para la creación de estos *mashups* se va a utilizar una herramienta concreta y el usuario no está familiarizado con la misma, esto puede provocar una mala medición del cumplimiento del proceso de negocio.
- Elegir un tema anteriormente inexplorado: Personalmente, escoger un tema del que no tengo experiencia es una decisión que considero que me puede servir para aprender y para explorar distintos campos dentro de la Ingeniería del Software.
- Realizar un producto de utilidad para la sociedad: Este es uno de los motivos que me incentivó a elegir esta titulación, en este caso en concreto, llevar a cabo un producto que agilice el proceso de creación de *mashups* a los usuarios.

1.3. Contribución

El presente Trabajo de Fin de Grado (TFG) tiene como objetivo principal aportar soluciones prácticas para abordar la complejidad en la definición de mashups y facilitar la gestión de la conformidad de procesos de negocio en el marco de una prueba de concepto. Las contribuciones específicas incluyen el desarrollo de componentes específicos para la modelación y configuración de mashups, la gestión integral de mashups dentro del sistema de conformidad, y la simplificación del proceso de creación de mashups. Además, se incluirá un módulo de usuarios para facilitar la gestión de permisos y roles, así como la integración del sistema con herramientas existentes dentro del contexto de la prueba de concepto.

Estas contribuciones se enfocan en mejorar la eficiencia y la efectividad de la gestión de conformidad en procesos de negocio, proporcionando soluciones prácticas y orientadas a las necesidades específicas de las organizaciones en este ámbito. El otro técnico del proyecto se encargará de integrar el sistema con una herramienta capaz de monitorizar y mostrar estadísticas de cumplimiento.

1.4. Base conceptual

Como base del proyecto, se dispone de documentación referente al proyecto realizado por Cabanillas y cols. en colaboración con la empresa de suministro energético mencionada anteriormente. Para ese proyecto se utilizó *Enterprise Architect* como aplicación para modelar estos mashups. Dicha documentación servirá para utilizar los componentes diseñados en el proyecto “Procede” de base para la creación de mashups en el sistema.

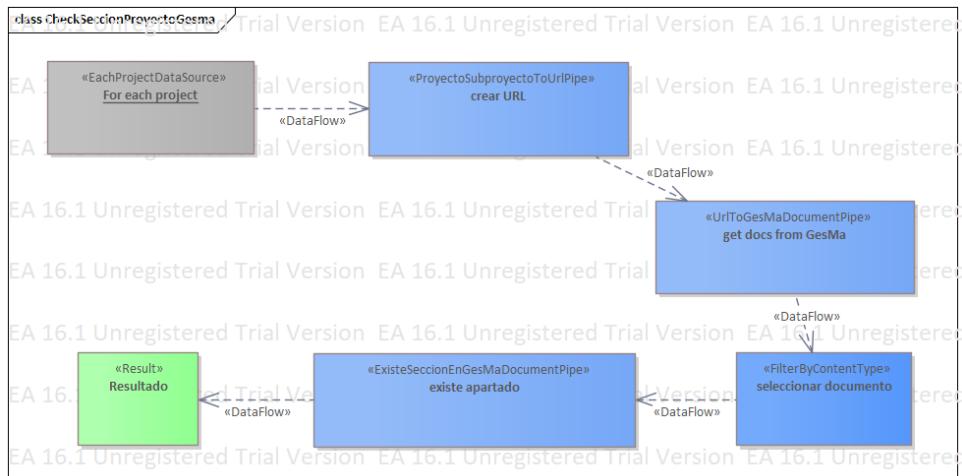


Figura 1.1: Comprobación de la existencia de una sección
(Fuente: *Enterprise Architect* del proyecto “Procede”)

La figura 1.1 muestra uno de los mashups desarrollados durante el proyecto “Procede”. Este especialmente se encarga de comprobar la existencia de una sección concreta en un documento de un proyecto y se devuelve la respuesta para ese proyecto.

1.5. Objetivos del Proyecto

Una vez mencionadas las motivaciones y habiendo distinguido el trabajo que va a realizar cada técnico para el proyecto, es esencial definir unos objetivos claros para la realización del proyecto. Dichos objetivos servirán de base para comenzar con la planificación inicial y para ser comparados con los resultados obtenidos.

1.5.1. Objetivos dentro de la investigación

A continuación, se detallan los objetivos específicos del proyecto:

- **[OBJ-01] Diseño de Componentes para la Modelación de Mashups:** Se desarrollarán componentes específicos que sean de utilidad para modelar y configurar mashups en el contexto de la gestión de conformidad. Estos componentes estarán diseñados para facilitar la integración y la interoperabilidad con diferentes fuentes de datos y herramientas de verificación de conformidad. Además de lo mencionado en el apartado anterior, también es importante integrar en esos componentes las herramientas impuestas por la prueba de concepto.
- **[OBJ-02] Gestión de Mashups:** Se incluirá la gestión de mashups como parte integral del sistema de gestión de conformidad, proporcionando herramientas y funcionalidades para la creación, edición, y gestión de mashups utilizados en la verificación de conformidad de procesos de negocio.
- **[OBJ-03] Simplificación de la Creación de Mashups:** Se buscará explorar y proponer métodos y herramientas que simplifiquen el proceso de creación de mashups para la verificación de conformidad en procesos de negocio. Esto implicará el análisis de técnicas existentes, así como el diseño de nuevos enfoques que permitan una creación más eficiente y efectiva de mashups.
- **[OBJ-04] Gestión de Autorización:** Se diseñará e implementará un módulo de autorización que permita el registro de usuarios por parte de los administradores, así como la asignación de permisos y privilegios.
- **[OBJ-05] Integración con Herramientas del Proyecto:** Se trabajará en la integración del sistema de gestión de conformidad con algunas herramientas impuestas por la prueba de concepto.

1.5.2. Objetivos académicos

Como Trabajo de Fin de Grado, se establecen los siguientes objetivos a nivel académico:

- Poner en práctica los conocimientos adquiridos a lo largo de la titulación.
- Aportar valor al proyecto de prueba de concepto
- Adquirir conocimientos sobre *User Experience (UX)* y *User Interface (UI)*.

En resumen, el proyecto tiene como objetivo desarrollar una parte de un sistema de gestión de conformidad basado en mashups para facilitar la verificación de conformidad en procesos de negocio. Esto implica diseñar componentes específicos, implementar un módulo de usuarios e integrar herramientas existentes para gestionar de manera eficiente la conformidad normativa en una organización.

1.6. Estructura del documento

De forma posterior a la introducción, en este documento se aborda en primer lugar el **Alcance del Proyecto y del Producto**, donde se describe detalladamente el alcance del proyecto, los entregables previstos y las restricciones que se han identificado.

Posteriormente, se encuentra la sección de la **Especificación de Requisitos**, que detalla las historias de usuario, los criterios de aceptación del producto y los requisitos de integración del sistema, proporcionando así una guía clara sobre qué se espera lograr.

La **Metodología de Trabajo** explica en detalle la metodología utilizada para gestionar y desarrollar el proyecto, lo que proporciona un marco organizativo para el proceso de desarrollo.

La **Estructura de Desglose del Trabajo (EDT) y Diccionario** presenta una vista estructurada del trabajo a realizar y los términos clave asociados, lo que ayuda a organizar y definir claramente las tareas.

A continuación, se encuentra la sección de **Estimaciones de Tiempo, Costes y Cronograma**, que ofrece una estimación del tiempo requerido, los costos asociados y un cronograma detallado de las actividades planificadas.

El **Diseño y Arquitectura del Producto** proporciona una visión detallada del diseño y la arquitectura del producto, lo que ayuda a comprender cómo se estructura y funciona el sistema.

En la sección de **Tecnologías y Herramientas**, se realiza un análisis de las tecnologías utilizadas y se presenta el stack tecnológico empleado en el proyecto.

El **Desarrollo del Producto** describe los diferentes sprints de desarrollo realizados, lo que ofrece una visión general del progreso del proyecto a lo largo del tiempo.

Seguidamente, en el apartado de **Análisis del Sistema**, se presenta un análisis detallado de la arquitectura, los nodos, el chatbot, la integración y la experiencia de usuario, proporcionando una comprensión completa del sistema desarrollado.

La sección de **Evaluación y Resultados** analiza la calidad del código, las pruebas realizadas, las contribuciones al proyecto y los costos asociados, lo que permite evaluar el éxito del proyecto.

En las **Conclusiones**, se ofrecen conclusiones finales, se identifican errores cometidos, se comparten lecciones aprendidas y se plantea el trabajo futuro que podría realizarse.

Los **Anexos** incluyen términos, manuales y otros documentos relevantes que complementan la información presentada en el documento principal.

Finalmente, en la sección de **Referencias**, se recogen todas las fuentes utilizadas para sustentar el trabajo realizado.

2. Alcance del Proyecto y del Producto

Debido a la complejidad de alguno de estos mashups para ser entendidos por perfiles no técnicos, se decidió extender el proyecto con un chatbot que dé soporte a los usuarios finales del sistema. Para ello, nos apoyaremos en los últimos avances tecnológicos en el campo de la Inteligencia Artificial. El campo de la inteligencia artificial ha experimentado un gran avance tecnológico en los últimos años, con el ejemplo más reciente y notable siendo el crecimiento exponencial de usuarios de ChatGPT, que ha multiplicado por tres su número de usuarios en solo un año (Dylan Patel (2023)), alcanzando más de 180 millones de usuarios activos mensuales en noviembre de 2023. Esta rapidez en la adopción y el crecimiento de la tecnología de inteligencia artificial es un indicador claro de su impacto y potencial para transformar la forma en que interactuamos y nos comunicamos con la tecnología.

2.1. Descripción del Alcance

El alcance del proyecto consiste en el desarrollo de una parte de un sistema de gestión de conformidad basado en mashups para facilitar la verificación de conformidad en procesos de negocio. Esto incluye el diseño de componentes específicos, la implementación de un módulo de usuarios y la integración de herramientas existentes para gestionar de manera eficiente la conformidad normativa en una organización. A continuación se detallan los límites específicos del proyecto:

- **Diseño de Componentes para la Modelación de Mashups:** Crear componentes específicos que faciliten la modelación y configuración de mashups en el contexto de la gestión de conformidad e integrar estos componentes con diferentes fuentes de datos y herramientas de verificación de conformidad para mejorar la interoperabilidad y eficiencia en el proceso de verificación de conformidad (como las herramientas impuestas por la prueba de concepto).
- **Gestión de Mashups:** Incluir la creación, edición y gestión de mashups utilizados en la verificación de conformidad de procesos de negocio. Además de permitir a los usuarios gestionar y personalizar los mashups según sus necesidades, lo que facilitará la verificación de conformidad y reducirá los errores.
- **Simplificación de la Creación de Mashups:** Desarrollar un chatbot que proporcione soporte a los usuarios finales del sistema de gestión de conformidad y realice acciones como describir mashups y generarlos a partir de una descripción.
- **Gestión de Autorización:** Diseñar e implementar un módulo de autorización que permita el registro de usuarios por parte de los administradores, así como la asignación de permisos y privilegios.
- **Integración con Herramientas del Proyecto:** Integrar del sistema de gestión de conformidad con herramientas existentes impuestas por la prueba de concepto.

2.2. Lista de Entregables

En base al alcance planteado, se reconocen los siguientes entregables:

- Manuales de usuarios (este entregable se incluye en el anexo 13.3).
- Manual de despliegue del sistema (este entregable se incluye en el anexo 13.2).

- Versión extendida del sistema (este entregable se incluye en el anexo 13.2).
- Documentación de la interfaz de usuario del chatbot, incluyendo ejemplos de cómo se diseñó la interfaz y cómo es la interacción con los usuarios (este entregable se incluye la sección 10.5).
- Documentación de la integración del chatbot (este entregable se incluye en el anexo 10.3).

2.3. Restricciones

Para realizar dicha prueba de concepto se escogieron las siguientes herramientas:

- Node-RED: Clerissi y cols. (2018) describen Node-RED como una plataforma de programación visual basada en Node.js que permite a los usuarios conectar dispositivos y servicios web de forma intuitiva mediante flujos de trabajo gráficos, facilitando la creación de aplicaciones IoT (*Internet of Things*) y de automatización. En el caso de la prueba de concepto, esta herramienta se utilizará para diseñar componentes (o conocidos en node-RED como nodos personalizados) capaces de crear *mashups* para medir el cumplimiento de la conformidad.
- Governify: Marco de gestión de acuerdos de servicio utilizado para realizar auditorías de manera automatizada. Esta plataforma fue el producto de un proyecto de investigación realizado por Gamez-Díaz y cols. Dentro del proyecto de investigación, se utilizará junto con la infraestructura de Bluejay (Subproyecto de Governify, García y cols. (2021)) para calcular y mostrar mediante gráficos el cumplimiento de la conformidad.

Según el contexto y el alcance mencionado, se establecen las siguientes restricciones para delimitar el desarrollo del proyecto:

- **Herramientas impuestas por el proyecto de investigación:** Las herramientas mencionadas anteriormente, Node-Red y Governify, son fundamentales dentro del sistema y serán empleadas tanto para la creación de *mashups* como para la evaluación de su cumplimiento. Además de estas, hay otras herramientas y tecnologías que han sido elegidas durante el desarrollo del proyecto de investigación por otros participantes.
- **Limitaciones de Uso del Chatbot:** El proyecto debe cumplir con las limitaciones de uso del chatbot, que incluyen la restricción de acceso a ciertos usuarios o la limitación del gasto que genera su uso.
- **Requisitos de Seguridad en Node-RED:** El proyecto debe cumplir con los requisitos de seguridad en Node-RED, que incluyen la implementación de mecanismos de autorización y autenticación en Node-RED para garantizar que solo los usuarios con roles específicos puedan editar los *mashups* desde el sistema.
- **Requisitos de Integración con Node-RED:** El proyecto debe cumplir con los requisitos de integración con Node-RED, que incluyen la comunicación entre el chatbot y Node-RED para la generación automática de *mashups* y para la descripción de éstos.

En resumen, estas restricciones proporcionan un marco claro y específico para el desarrollo del proyecto, garantizando la integridad, seguridad y eficacia de las soluciones implementadas.

3. Especificación de Requisitos

Este apartado se centra en definir las características del trabajo a realizar, de forma que se integre con éxito en el entorno del proyecto de investigación. Para recoger estos requisitos se utilizará de referencia la sección de ingeniería de requisitos de la junta de andalucía¹ y se seguirán algunas recomendaciones del artículo de Durán y Bernárdez (2001), en el que se define una metodología clave para especificar los requisitos de forma efectiva. Por otro lado, cabe destacar que se ha optado por no considerar requisitos de información, ya que no se implementará un sistema de información, sino la extensión de un sistema que tiene como objetivo monitorizar el cumplimiento normativo.

3.1. Historias de Usuario

A continuación, se presenta una recopilación de historias de usuarios que ilustran distintos escenarios de interacción con el sistema que se va a desarrollar. Las historias se presentan en dos modalidades: caso positivos y caso negativos, acompañados de sus respectivos mockups. Estas historias de usuario permiten comprender de manera detallada cómo los potenciales usuarios interactuarán con la aplicación en diferentes situaciones, desde experiencias exitosas hasta desafíos y obstáculos a superar. Este enfoque facilita el diseño centrado en el usuario y la identificación de áreas de mejora para optimizar la experiencia del usuario final.

3.1.1. Historia de usuario 1

Como administrador, quiero gestionar el chatbot para gestionar su uso y su disponibilidad

- **HU[1+1]:** Dado que soy un administrador “Alpha” autenticado en el sistema y que en el sistema hay cinco chats creados, cuando accedo a la sección de Administración, el sistema muestra un listado en el que aparecen los cinco chats. Además, aparece un panel para poder configurar el chatbot, permitiéndome así gestionar este servicio y consultar los costes asociados a este (ver Figura 3.1).

¹<https://www.juntadeandalucia.es/servicios/madeja/contenido/subsistemas/ingenieria/ingenieria-requisitos>

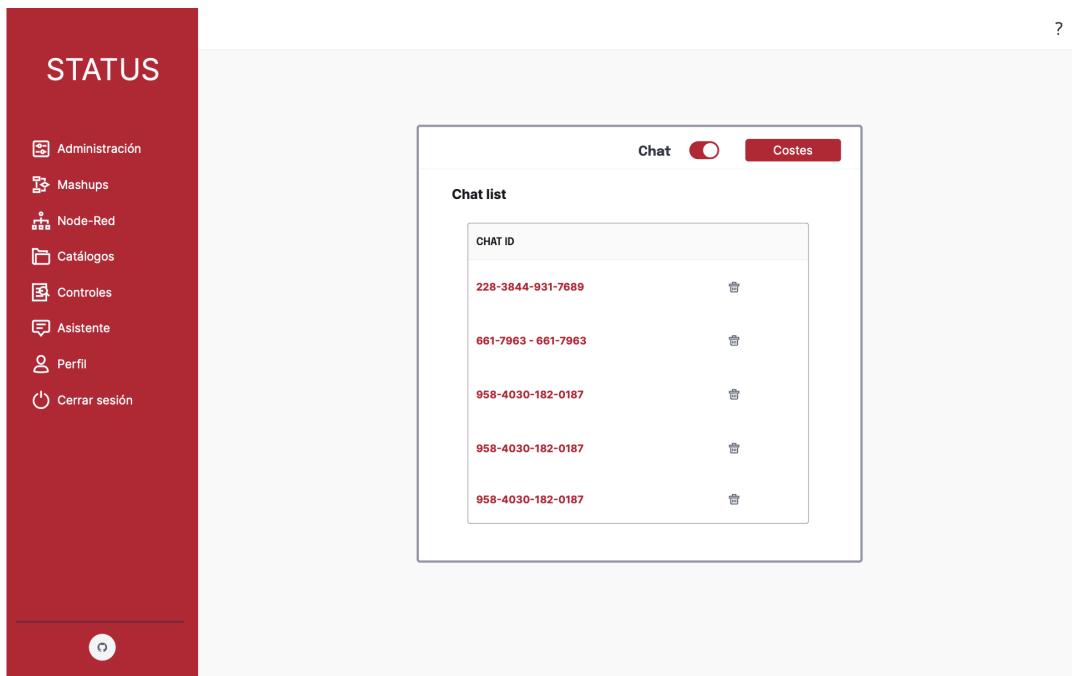


Figura 3.1: Vista de gestión del chatbot

- **HU[1+2]:** Dado que soy un administrador “Alpha” autenticado en el sistema y que el sistema tiene el servicio del chatbot activado, cuando accedo a la sección de Administración y pulso un botón para desactivar el chatbot, el sistema me muestra un modal para deshabilitar el chatbot (ver Figura 3.2).

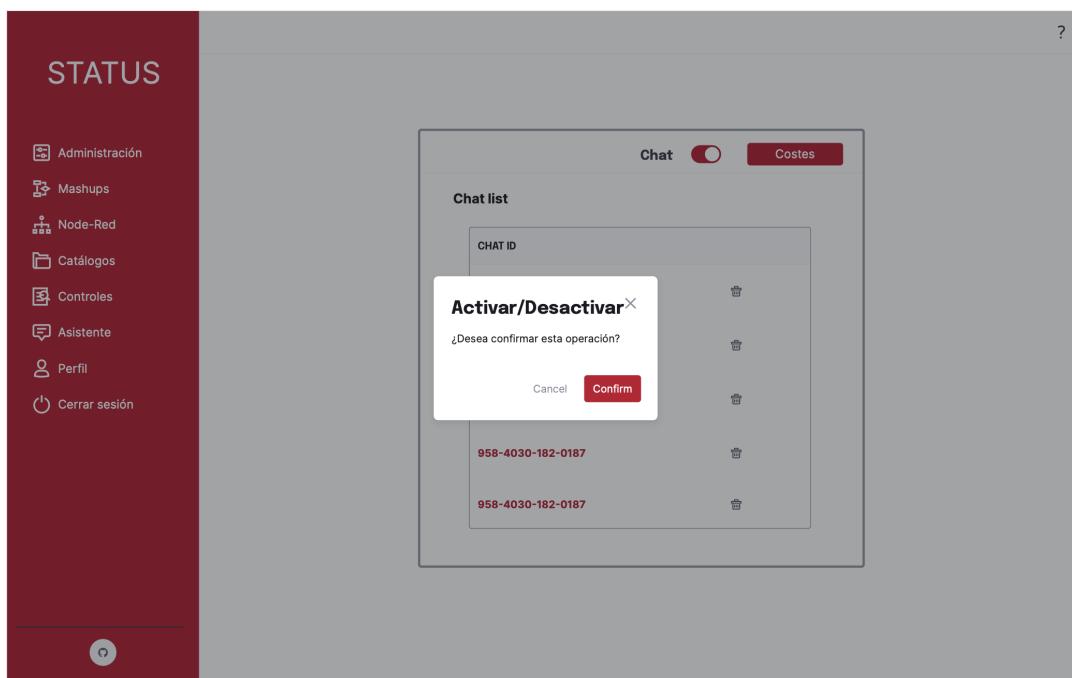


Figura 3.2: Modal de activación/desactivación del chat

3.1.2. Historia de usuario 2

Como técnico, quiero poder visualizar los mashups existentes para gestionar la medición del cumplimiento de los procesos de negocio correctamente.

- **HU[2+1]:** Dado que soy un técnico “Bravo” autenticado en el sistema y que tengo desplegado node-RED, cuando accedo a la sección de “Node-RED”, visualizo los mashups existentes del editor de Node-RED, entonces puedo hacer uso del editor desde el sistema y modificar los mashups en función del proceso de negocio a medir (ver Figura 3.3).

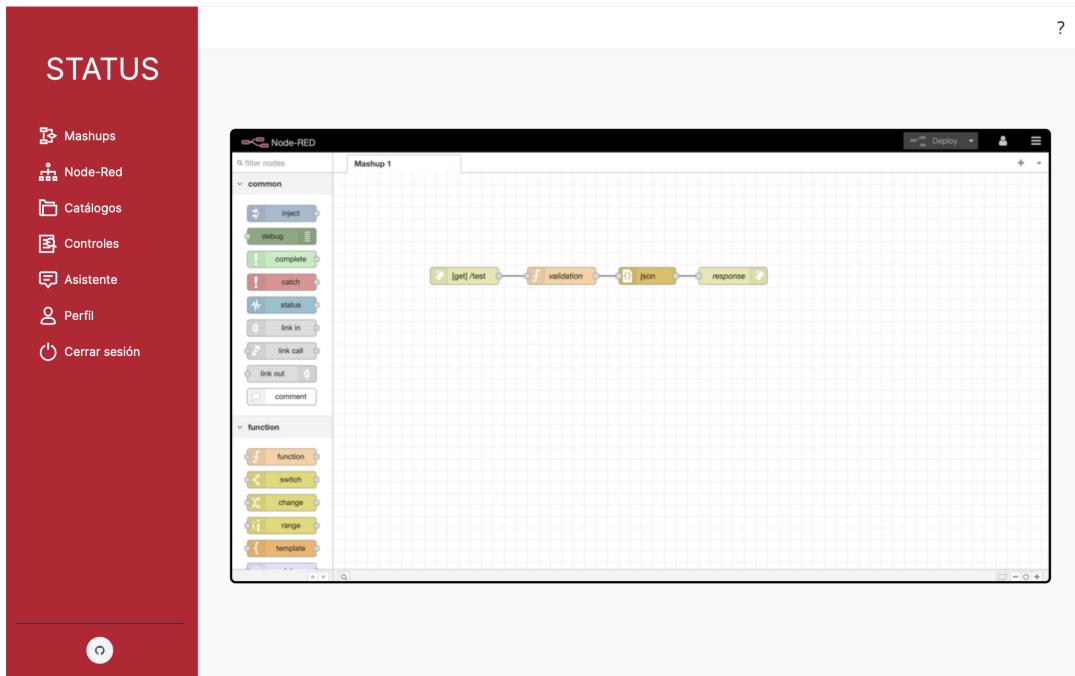


Figura 3.3: Editor de Node-RED

- **HU[2+2]:** Dado que soy técnico “Bravo” autenticado en el sistema y que hay 5 mashups creados en el sistema, cuando accedo a la sección “Mashups”, obtengo un listado de los mashups en el que aparecen los 5 existentes. Además, asociado a cada mashup aparece un botón “Eliminar”, de forma que quiero eliminar el mashup “Exist Url” y pincho en ese botón, se elimina ese mashup.(ver Figura 3.4).

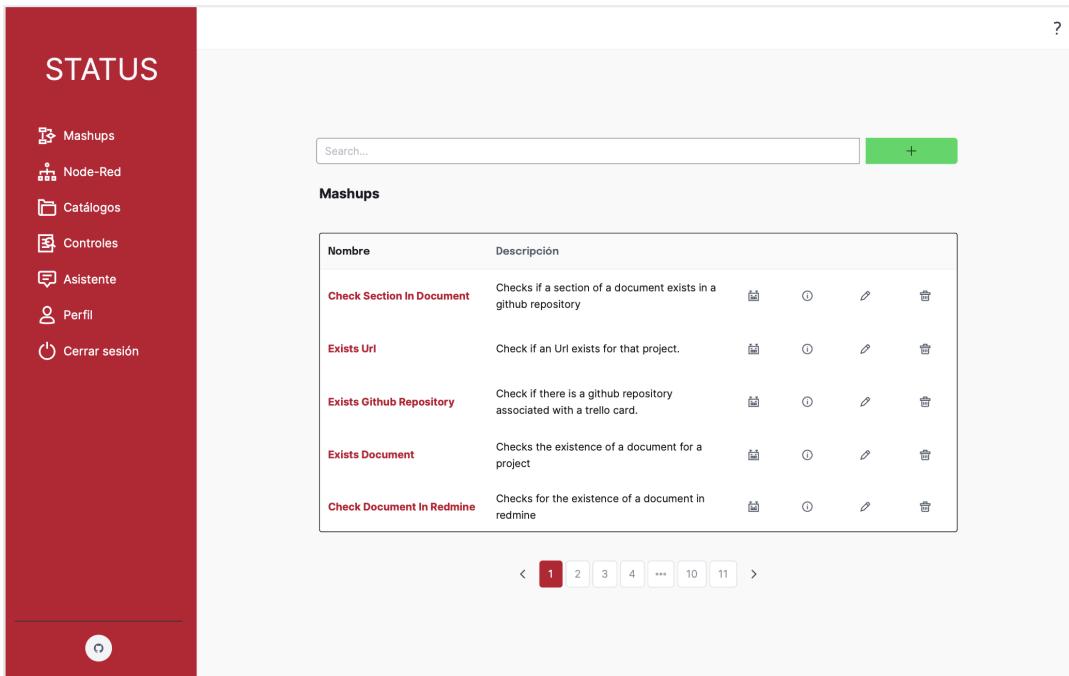


Figura 3.4: Listado de mashups

- **HU[2-1]:** Dado que soy un técnico “Bravo” autenticado en el sistema, cuando intento personalizar un mashup desde la sección de Node-RED pero dejo en blanco algún campo obligatorio para el mashup (como el *endpoint* del mashup), entonces el sistema muestra un mensaje de aviso mencionando que el nodo en cuestión está mal configurado, (ver Figura 3.5).

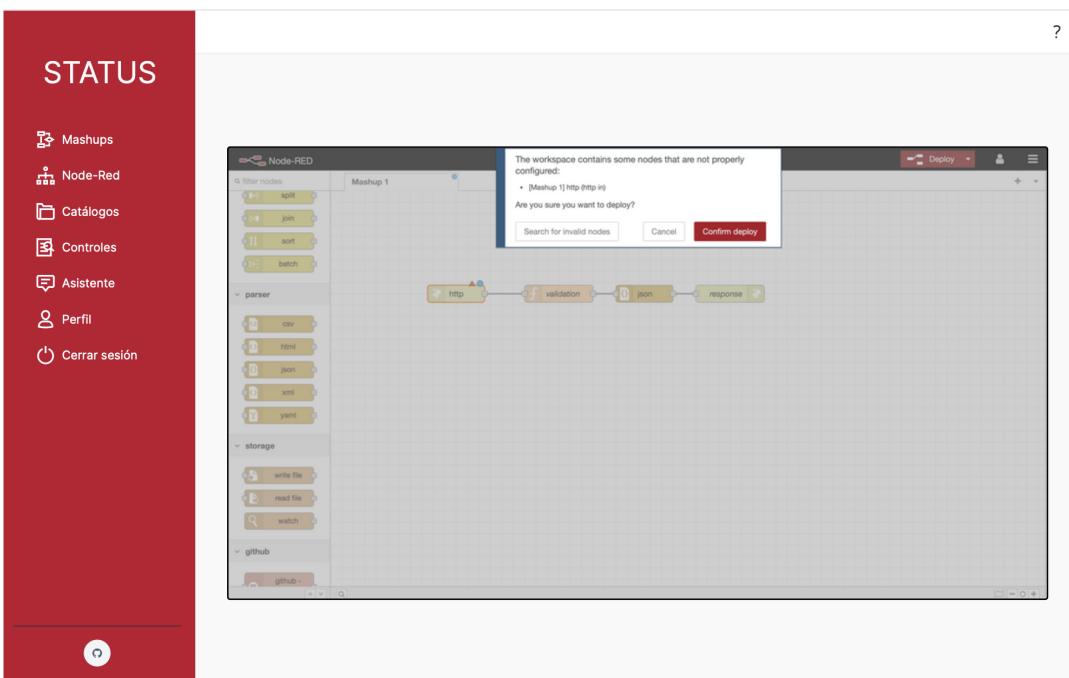


Figura 3.5: Aviso de mal funcionamiento de Node-RED

3.1.3. Historia de usuario 3

Como técnico, quiero poder generar un nuevo mashup a través del chatbot para crear mashups sin tener que comprender en profundidad el funcionamiento de Node-RED o de sus nodos.

- **HU[3+1]:** Dado que soy un técnico “Bravo” autenticado en el sistema, cuando accedo a la sección “Chat”, visualizo el chat y describo un modelo de mashup al chatbot, entonces el sistema crea un nuevo mashup utilizando siguiendo mi descripción y abre el editor de Node-RED mostrandome el mashup generado (ver Figura 3.6).

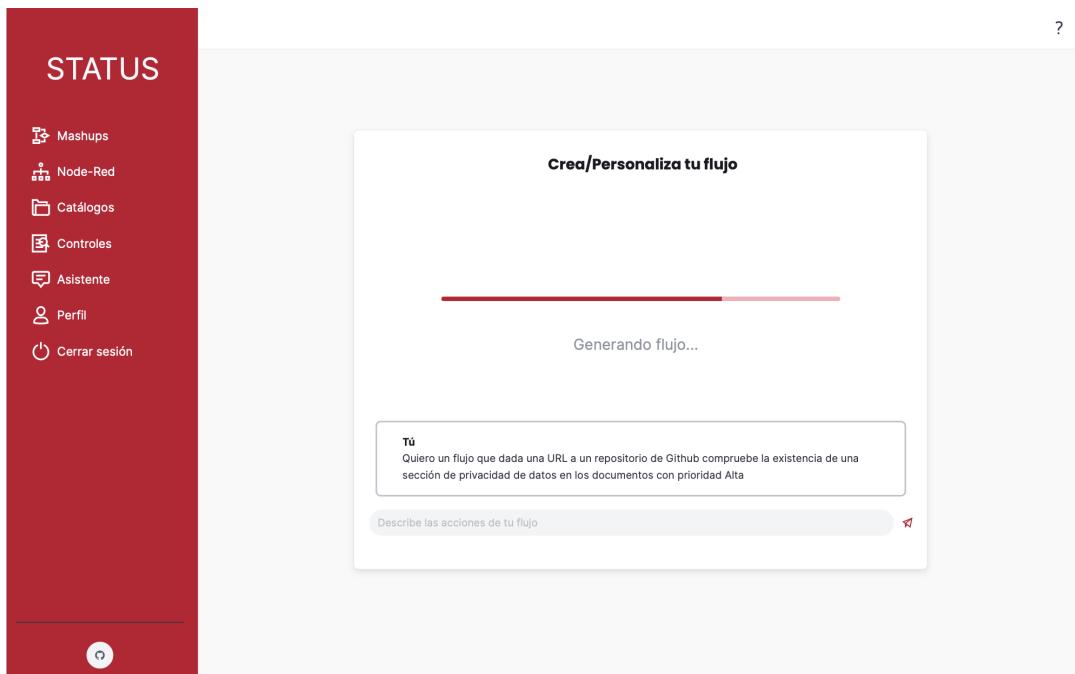


Figura 3.6: Generación de mashups

- **HU[3-1]:** Dado que soy un técnico “Bravo” autenticado en el sistema, cuando accedo a la sección “Chat” pero introduzco una descripción “Create something”, entonces el sistema muestra un mensaje indicando la invalidez de los datos introducidos y no crea el mashup hasta que la información sea válida (ver Figura 3.7).

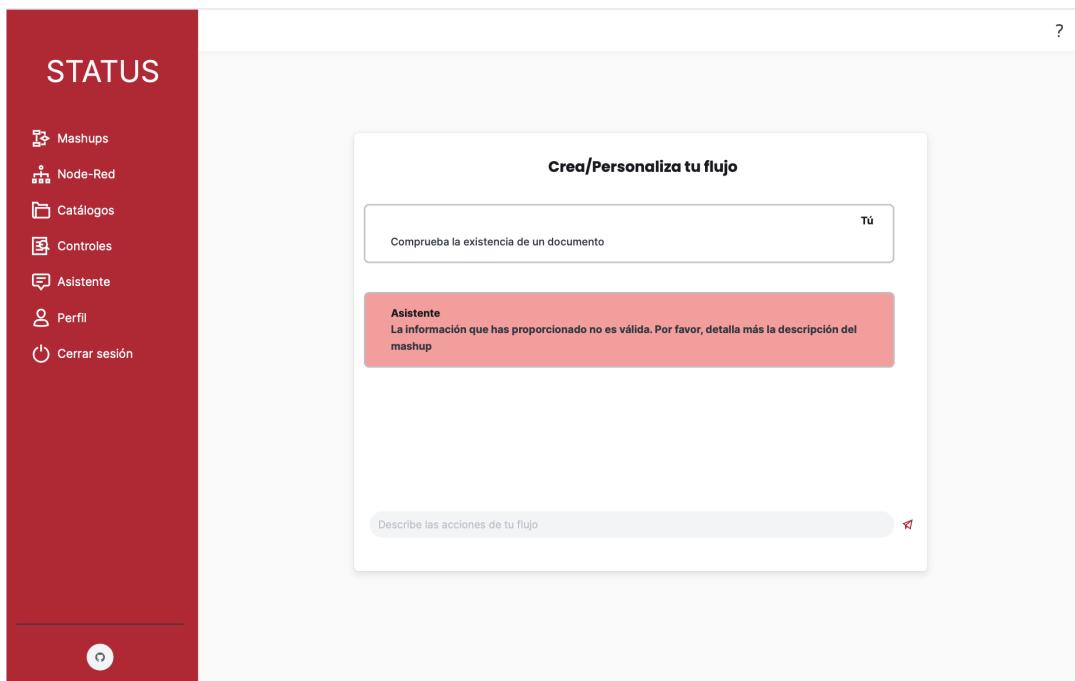


Figura 3.7: Conversación con datos insuficientes

3.1.4. Historia de usuario 4

Como técnico, quiero poder gestionar los mashups existentes para adaptarlos a nuevas necesidades de medición de la conformidad.

- **HU[4+1]:** Dado que soy técnico “Bravo” autenticado en el sistema y que hay un mashup que no tiene descripción, cuando quiero accedo al listado de mashups, entonces aparece un botón “Bot” asociado a cada mashup, de forma que si pulso ese botón el sistema genera una descripción del mashup.(ver Figura 3.8).

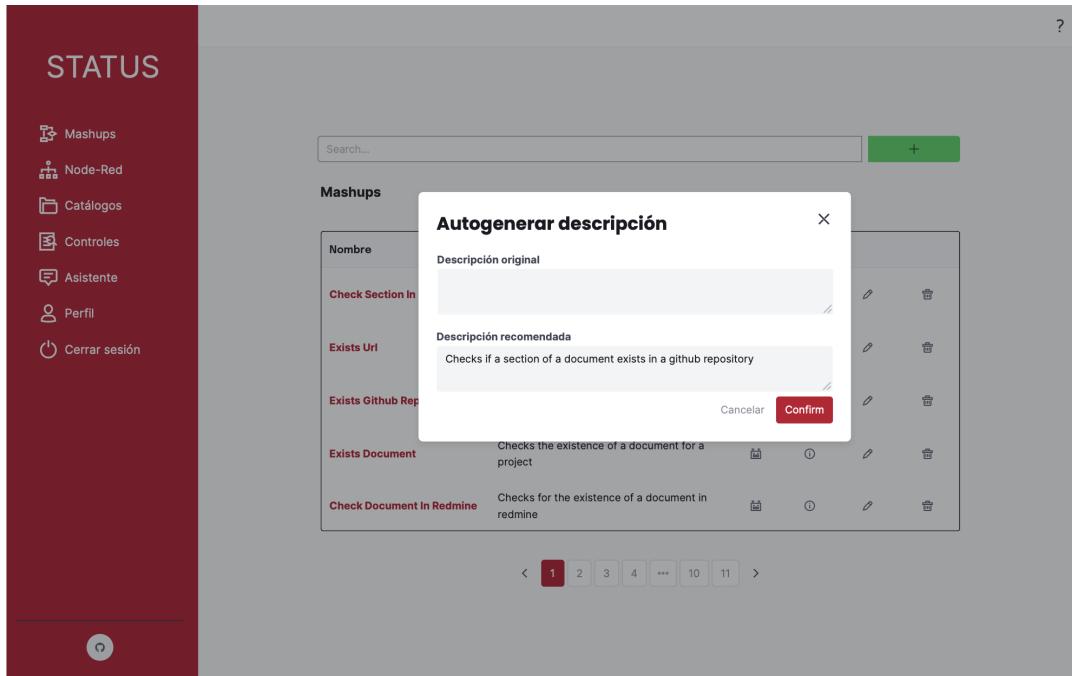


Figura 3.8: Recomendación de descripción

- **HU[4+2]:** Dado que soy técnico “Bravo” autenticado en el sistema, cuando quiero acceder a los detalles de un mashup “Check Section in Github”, entonces aparece un botón “Info” asociado a cada mashup, de forma que si pulso ese botón el sistema abre un modal con una descripción nodo a nodo del mashup “Check Section in Github” (ver Figura 3.9).

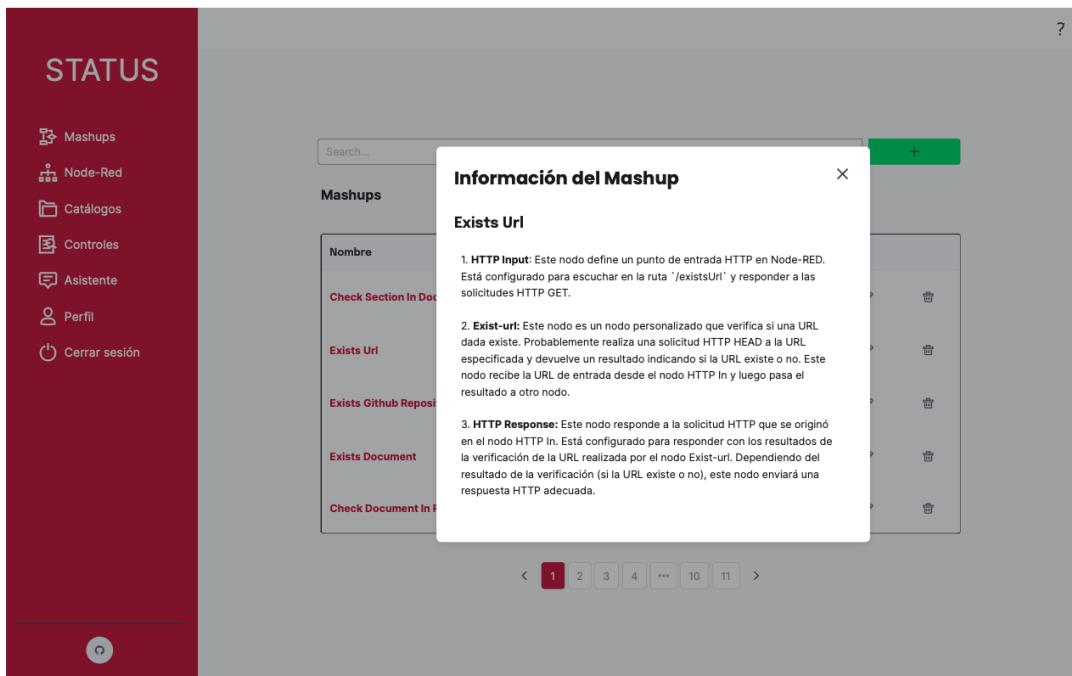


Figura 3.9: Detalles del mashup

- **HU[4+3]:** Dado que soy técnico “Bravo” autenticado en el sistema, cuando accedo al listado de mashup porque quiero editar un mashup “Exists Document”, entonces un botón asociado

a cada mashup aparece un botón “Editar”, de forma que si pincho en ese botón se me abre el editor de Node-RED con el contenido de “Exists Document”. (Ver en la Figura 3.3).

- **HU[4-1]:** Dado que soy un técnico “Bravo” autenticado en el sistema, cuando accedo a la sección de “Mashups” pero no hay mashups disponibles, entonces el sistema muestra un aviso de la ausencia de estos (ver Figura 3.10).

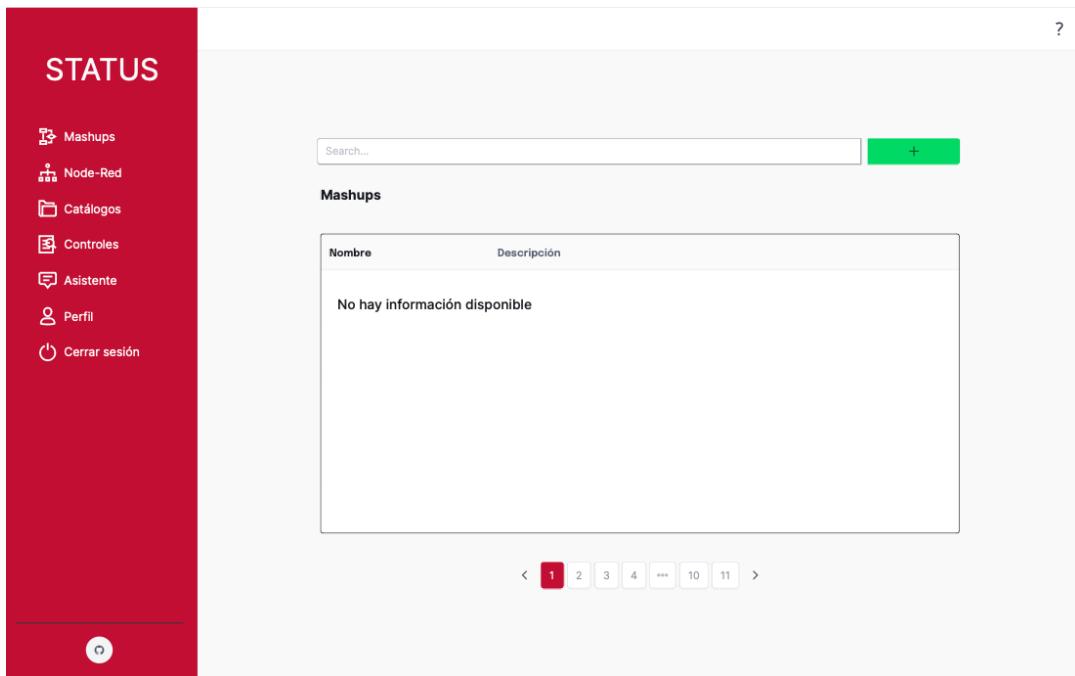


Figura 3.10: Mashups no disponibles

3.1.5. Historia de usuario 5

Como usuario registrado, quiero tener la opción de conectar cuentas externas, como GitHub o Trello, a mi perfil para integrar la información de esas cuentas con el sistema STATUS.

- **HU[5+1]:** Dado que soy un usuario “Charlie” registrado en el sistema, cuando accedo a la sección de Perfil, entonces aparecen botones para conectar una cuenta de GitHub o Trello. Además, si pulso en el botón de “Github” e ingreso mis credenciales de manera segura y autorizo la integración, entonces el sistema conecta con éxito “Github” a mi perfil (ver Figura 3.11).

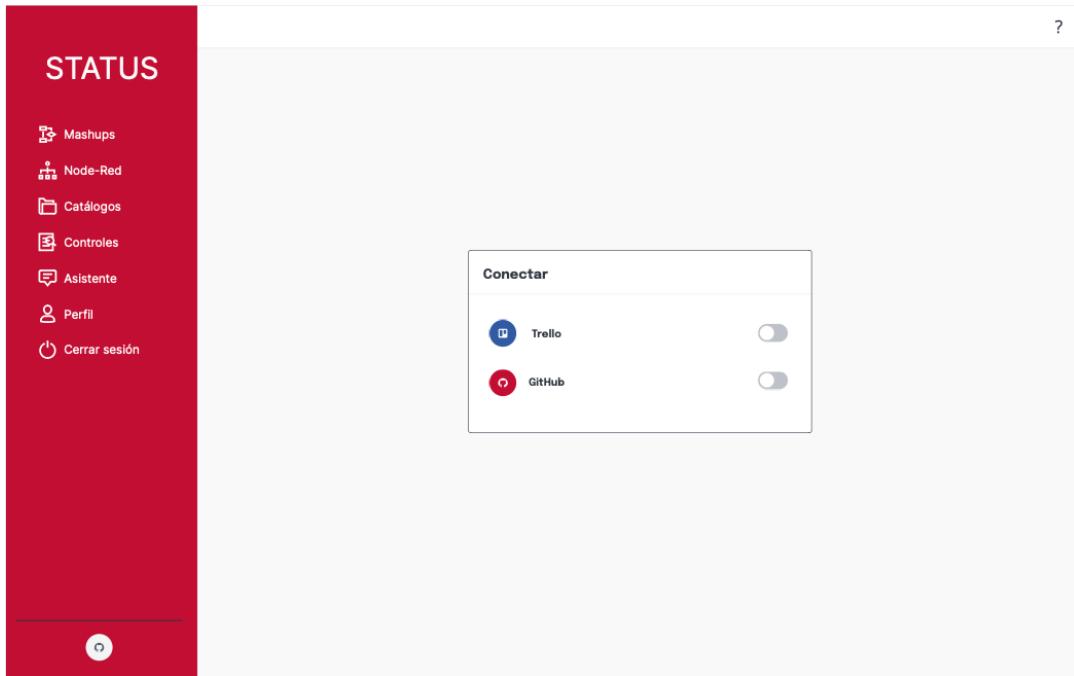


Figura 3.11: Conexión con otras herramientas

- **HU[5+2]:** Dado que soy un usuario “Charlie” registrado en el sistema y que tengo conectada la cuenta de Github, cuando accedo a mi perfil, entonces aparecen botones para desconectar la cuenta de GitHub. Además, si confirmo la acción pulsando en el botón “Github” y el sistema desconecta con éxito mi cuenta de Github, protegiendo así mi privacidad y desvinculando la información asociada (Ver en la Figura 3.11).

3.1.6. Historia de usuario 6

Como técnico, quiero usar componentes personalizados en Node-Red, para crear mashups que midan la conformidad de procesos de negocio.

- **HU[6+1]:** Dado que soy un técnico “Bravo” autenticado y que tengo un tablero de trello con una card asociada a un repositorio de GitHub “Acme”, cuando accedo a la sección “Node-Red” del sistema, tengo un componente dentro de la categoría “Procede” que extrae información de mi tablero “STATUS” de Trello, entonces puedo usarlos para comprobar la existencia de repositorios de Github asociados a las tarjetas de ese tablero de Trello, devolviéndome verdadero.
- **HU[6+2]:** Como técnico “Bravo” autenticado, que tengo un tablero de Trello con una card asociada a un repositorio de GitHub “Acme”, cuando accedo a la sección “Node-Red” del sistema, tengo un componente dentro de la categoría “Procede” que extrae información de mi tablero “STATUS” de Trello, entonces puedo usarlo para obtener la URL del repositorio de GitHub asociado a una tarjeta específica de ese tablero de Trello, devolviéndome el enlace a “Acme”.
- **HU[6+3]:** Como técnico “Bravo” autenticado, que tengo acceso a un conjunto de datos con una propiedad de fecha, cuando accedo a la sección “Node-Red” del sistema, tengo un componente dentro de la categoría “Procede” que me permite filtrar esos objetos por un intervalo de tiempo específico, entonces puedo usarlo para obtener los objetos cuyas fechas

estén dentro del intervalo especificado.

- **HU[6+4]:** Como técnico “Bravo” autenticado, que tengo acceso a un conjunto de datos con diversas propiedades, cuando accedo a la sección “Node-Red” del sistema, tengo un componente dentro de la categoría “Procede” que me permite filtrar esos objetos por un valor específico en una propiedad dada, entonces puedo usarlo para obtener los objetos que coincidan con el valor especificado.
- **HU[6+5]:** Como técnico “Bravo” autenticado, que tengo una lista de objetos, cuando accedo a la sección “Node-Red” del sistema, tengo un componente dentro de la categoría “Procede” que verifica si el tamaño de la lista coincide con un tamaño específico, entonces puedo usarlo para comprobar si la lista tiene el tamaño esperado.
- **HU[6+6]:** Como técnico “Bravo” autenticado, que tengo una URL a un repositorio de GitHub y la ubicación de un documento dentro de ese repositorio, cuando accedo a la sección “Node-Red” del sistema, tengo un componente dentro de la categoría “Procede” que comprueba si una cadena de texto específica está contenida en el documento, entonces puedo usarlo para verificar la existencia de esa cadena en el documento.
- **HU[6+7]:** Como técnico “Bravo” autenticado, que tengo una URL a un repositorio de GitHub y la ubicación de un documento dentro de ese repositorio, cuando accedo a la sección “Node-Red” del sistema, tengo un componente dentro de la categoría “Procede” que me permite obtener el texto del documento, entonces puedo usarlo para acceder y leer el contenido del documento especificado.

3.1.7. Historia de usuario 7

Como administrador, quiero tener la opción de registrar nuevos usuarios y asignarles permisos y privilegios, para asegurarme de que solo los usuarios autorizados puedan acceder y utilizar las funcionalidades del sistema de gestión de conformidad.

- **HU[7+1]:** Como usuario registrado “Alpha”, que tengo una cuenta con permisos de administrador, cuando accedo a la sección de “Inicio de Sesión” del sistema, tengo un formulario que me permite iniciar sesión con mis credenciales, entonces cuando introduzco mis credenciales de forma correcta y pulso en el botón de “Iniciar Sesión”, accedo al sistema y tengo acceso a todas las partes del sistema. (ver Figura 3.12).

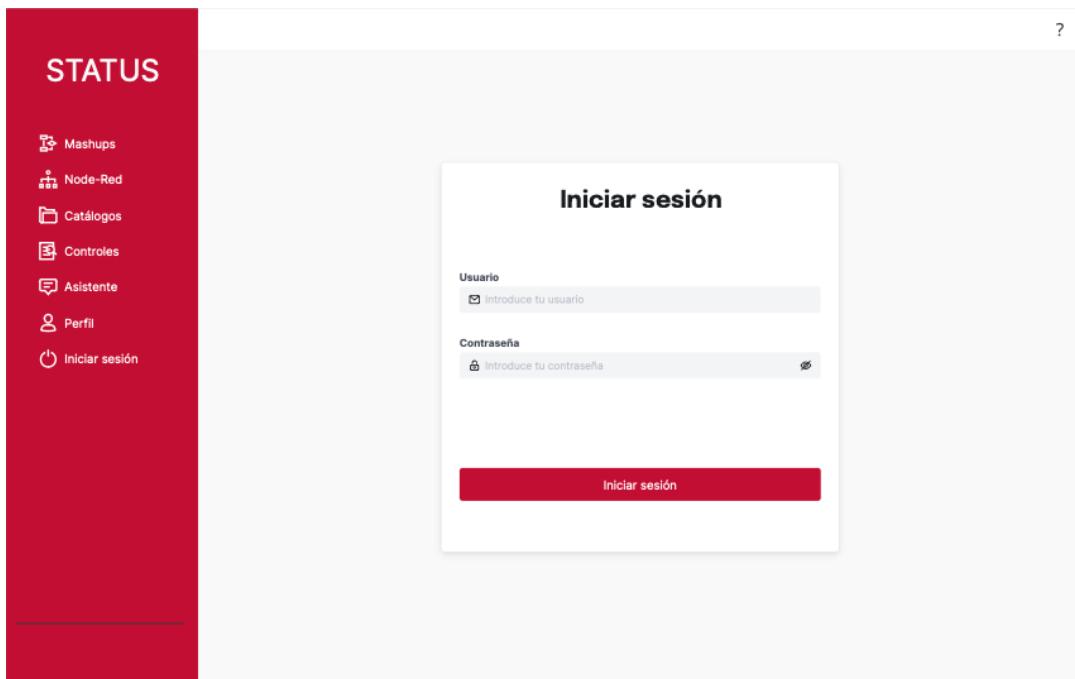


Figura 3.12: Inicio de sesión

Es importante mencionar que no se han contemplado el resto de funcionalidades de gestión de usuarios de forma consciente, siendo esto algo que se contemplará en etapas posteriores a la prueba de concepto.

3.1.8. Matriz de Trazabilidad

Para identificar el nivel de trazabilidad del sistema, se ha elaborado una matriz de trazabilidad que muestra la relación entre los diferentes objetivos y requisitos. En la matriz tenemos los objetivos dentro de la investigación (Apartado 1.5.1) en columnas y las historias de usuario (Apartado 3.1) en filas.

	OBJ-01	OBJ-02	OBJ-03	OBJ-04	OBJ-05
HU-01			x		
HU-02	x	x	x		x
HU-03			x		
HU-04		x	x		
HU-05	x				x
HU-06	x				x
HU-07	x			x	

Cuadro 3.1: Objetivos / Historias de usuario

3.2. Criterios de Aceptación del Producto

Los criterios de aceptación para el producto son los siguientes:

- **[PAC-01]:** Al acceder a la sección de Administración, los administradores autenticados deben visualizar un listado de todos los chats disponibles en el sistema y tener la capacidad de

configurar el chatbot, incluyendo la opción de consultar los costos asociados.

- **[PAC-02]:** Al desactivar el chatbot desde la sección de Administración, los administradores autenticados deben ser presentados con un modal de confirmación para confirmar la acción.
- **[PAC-03]:** Al ingresar a la sección de Node-RED, los técnicos autenticados deben ver una lista de los mashups existentes en el editor y poder modificar los mashups según los procesos de negocio a medir.
- **[PAC-04]:** Al acceder a la sección de Mashups, los técnicos autenticados deben visualizar un listado de todos los mashups existentes y tener la capacidad de eliminar un mashup específico de la lista. Además, si intentan personalizar un mashup desde Node-RED pero dejan en blanco algún campo obligatorio, el sistema debe mostrar un mensaje de aviso indicando el error de configuración.
- **[PAC-05]:** Al describir un modelo de mashup en el chat, se deben generar un nuevo mashup según la descripción proporcionada y abrir el editor de Node-RED mostrando a los técnicos el mashup generado. En caso de introducir una descripción inválida en el chat para crear un mashup, el sistema debe mostrar un mensaje de error y no crear el mashup hasta que la información sea válida.
- **[PAC-06]:** Al acceder al listado de mashups, los técnicos autenticados deben ver un botón asociado a cada mashup que les permita generar una descripción para aquellos que no lo la tengan. Asimismo, al visualizar los detalles de un mashup específico, deben poder ver una descripción nodo a nodo del mismo mediante un modal. Por otro lado, deben tener la capacidad de editar un mashup existente desde el listado y abrir el editor de Node-RED con el contenido del mashup seleccionado. En caso de no haber mashups disponibles en el sistema, se debe mostrar un aviso indicando la ausencia de estos.
- **[PAC-07]:** Al acceder a su perfil, los usuarios registrados deben poder conectar cuentas externas como GitHub o Trello, ingresar sus credenciales de manera segura y autorizar la integración con éxito. Además, deben tener la opción de desconectar las cuentas externas previamente conectadas, protegiendo así su privacidad.
- **[PAC-08]:** Al acceder a la sección de Node-RED, los técnicos autenticados deben tener acceso a una lista de componentes personalizados dentro de la categoría “Procede” para crear mashups que midan la conformidad de procesos de negocio.
- **[PAC-09]:** La documentación realizada debe ser comprensible para los usuarios finales del sistema.

3.3. Atributos de Calidad

3.3.1. Requisitos de usabilidad del sistema

- **[NFR-01] Sistema intuitivo:** El sistema deberá permitir a los usuarios completar las tareas principales en un máximo de tres clics, para mejorar la comodidad de uso de este y facilitar el aprendizaje de nuevos usuarios.
- **[NFR-02] Sesión renovable:** El sistema deberá mantener la sesión abierta un plazo mínimo de 1 hora con opción de prorrogación, para facilitar el uso por parte de los usuarios finales del sistema.

3.3.2. Requisitos de eficiencia del sistema

- **[NFR-03] Unificación del software:** El sistema deberá gestionar todos los procesos de gestión de cumplimiento de la conformidad posibles desde un solo software que lo unifique todo para reducir tiempos y agilizar procesos.

3.3.3. Requisitos de interoperabilidad del sistema

- **[NFR-04] Compatibilidad del sistema:** El sistema deberá ser compatible con distintos softwares, entre ellos Bluejay y Node-Red de forma que se puedan acceder a ellos sin provocar problemas de incompatibilidad o bloqueo del sistema.
- **[NFR-05] Soporte para API:** El sistema debe proporcionar interfaces de programación de aplicaciones (API) documentadas y seguras que permitan a otros sistemas acceder y actualizar datos del sistema de manera programática.

3.3.4. Requisitos de seguridad del sistema

- **[NFR-06] Control de acceso:** El sistema deberá evitar accesos indeseados del sistema usando un proceso de autenticación de usuario con clave al inicio del sistema, además de restringir el acceso a determinadas partes del sistema sin autenticación del usuario.
- **[NFR-07] Autenticación de usuarios:** El sistema debe requerir que los usuarios se autentiquen antes de acceder a funciones protegidas (Mediante la verificación de credenciales como nombre de usuario y contraseña).

3.3.5. Requisitos de mantenibilidad del sistema

- **[NFR-08] Documentación online:** El sistema deberá proporcionar una documentación online completa y actualizada que describa procedimientos de instalación, configuración y ejemplos de uso.

3.4. Requisitos de Integración del Sistema

Es importante resaltar que en este caso se ha excluido el requisito de integración con Bluejay, ya que no forma parte del alcance de este trabajo. Esta tarea será llevada a cabo por el otro técnico del proyecto, Francisco Montero.

- **[CRQ-01] Consumir servicios de Node-Red:** El sistema deberá integrarse con Node-RED para diseñar mashups que permitan medir el cumplimiento de la conformidad de manera eficiente y precisa.
- **[CRQ-02] Consumir servicios de Github:** El sistema deberá integrarse con Github para acceder a la información relevante para la medición del cumplimiento de procesos de negocio almacenada en esta plataforma. Además, este servicio también deberá integrarse en algunos componentes utilizados para la creación de mashups.
- **[CRQ-03] Consumir servicios de Trello:** El sistema deberá integrarse con Trello para acceder a la información relevante para la medición del cumplimiento de procesos de negocio almacenada en esta herramienta. Además, este servicio también deberá integrarse en algunos componentes utilizados para la creación de mashups (al igual que Github).

De forma adicional, el sistema también se integra con el servicio de Bluejay; sin embargo, esto es algo ajeno a este trabajo por lo que no será contemplado como un requisito de integración, sino cómo una herramienta que se debe conocer para el contexto del proyecto y el diseño del resto de partes del sistema.

4. Metodología de Trabajo

La metodología de trabajo estructura el enfoque tanto de la gestión del proyecto como del desarrollo del mismo. Se dividirá en dos partes principales: la metodología de gestión del proyecto, que abarcará las estrategias para dirigir el proyecto durante todas las fases del mismo; y la metodología de desarrollo, que detallará las técnicas y prácticas para la creación del producto o servicio. “Una buena planificación, combinada con un seguimiento y una evaluación eficaces, pueden desempeñar un rol importante para mejorar la eficacia de los programas y proyectos de desarrollo”(UNDP. Bureau for Development Policy y cols. (2009), pag 7)

4.1. Metodología de Gestión del Proyecto

Debido a las recomendaciones de gestión del desarrollo del TFG y al uso extendido de esta terminología a nivel nacional, se ha decidido utilizar una adaptación de la guía PMBOK (*Project Management Body of Knowledge*). En esta guía se definen cinco fases del proyecto: Inicio, Planificación, Ejecución, Seguimiento y Control y Cierre. Cada fase del proyecto ha sido adaptada para ajustarse al contexto específico del Trabajo de Fin de Grado (TFG), resultando en el siguiente esquema:

- **Inicio:** En este caso, corresponde a la fase de adjudicación y planteamiento inicial del TFG junto a los tutores.
- **Planificación:** En esta fase se define el alcance, se especifican los requisitos previamente mencionados, se realiza la Estructura de Desglose de Trabajo junto con su diccionario, el listado de actividades y tiempos y el cronograma del proyecto.
- **Ejecución:** Fase correspondiente a la implementación del proyecto que seguirá la metodología definida en el siguiente apartado.
- **Seguimiento y Control:** Durante esta fase, se llevarán a cabo sesiones de seguimiento periódicas con los tutores para evaluar el progreso del proyecto e identificar áreas de mejora. Además, se realizará un informe general de avance del proyecto que muestra el estado en ese momento y plantea los siguientes objetivos a nivel de desarrollo.
- **Cierre:** Correspondiente con la fase de entrega del proyecto.

4.2. Metodología de Desarrollo

Durante la fase de ejecución del proyecto, se ha tomado la decisión de emplear metodologías ágiles para gestionar y desarrollar este Trabajo de Fin de Grado (TFG). Específicamente, se ha optado por una adaptación de Scrum, un marco de trabajo ágil ampliamente reconocido en el desarrollo de software. Esta elección se fundamenta en la flexibilidad y capacidad de adaptación que ofrecen las metodologías ágiles, así como en su enfoque iterativo e incremental, que permite una mayor eficiencia en la entrega de resultados y una mejor gestión del cambio a lo largo del proyecto. Para clarificar la adaptación propuesta, se describirá la adaptación realizada de los eventos y artefactos de Scrum (Schwaber y Sutherland (2013)).

4.2.1. Eventos Scrum

- **Sprint Planning:** Se ha optado por no llevar a cabo este evento en particular, dado que la planificación ya ha sido detallada en el diccionario de la EDT. Por consiguiente, no será preciso efectuar una sesión de planificación en cada iteración del *Sprint*.
- **Daily Scrum:** Este evento ha sido modificado para ser una *Weekly Scrum*, reuniones semanales (distintas a las comentadas previamente de seguimiento), en las que se comenta el avance semanal y se plantea la próxima tarea a realizar.
- **Sprint Review:** Esta revisión se celebra de forma previa a las sesiones de seguimiento (al final de cada *Sprint*), en las que se muestra lo realizado hasta ese momento y se validan las tareas realizadas con los tutores.
- **Sprint Retrospective:** Esta retrospectiva se realiza de forma individual al final de cada *Sprint*, de ella se registran lecciones aprendidas (incluidas en esa sección) y se plantea un plan de acción de cara al siguiente *Sprint*.

4.2.2. Artefactos Scrum

- **Product y Sprint Goal:** En este caso estos objetivos están previamente definidos en la sección 1.5, a excepción de los objetivos a nivel de Sprint, ya que se considera que éstos generan incrementos necesarios para completar el desarrollo.
- **Product y Sprint Backlog:** Para los backlogs, se hará uso inicialmente del desglose de trabajo correspondiente a la fase de ejecución en el diccionario de la Estructura de Desglose de Trabajo (EDT).
- **Tablero Scrum:** Para gestionar el avance de las tareas, se implementará un tablero Scrum utilizando la funcionalidad de *GitHub Projects*. Este tablero proporcionará una visualización clara del estado de las tareas y permitirá un seguimiento eficiente del progreso del proyecto.
- **Incrementos:** Estos elementos serán generados en cada *Sprint*, además de contener los entregables definidos en la sección 2.2.
- **Definición de Hecho:** Una tarea o funcionalidad se considera terminada cuando cumpla los requisitos especificados, se haya realizado una revisión de código por parte de alguna herramienta de análisis y haya recibido la validación por parte del tutor.

4.2.3. Proceso de Análisis y Mejora de la Calidad

En este proyecto, es fundamental distinguir entre el análisis y la mejora de la calidad tanto en documentos como en código.

Calidad de los Documentos

Para garantizar la calidad de los documentos asociados al proyecto, se llevarán a cabo revisiones tanto por parte del estudiante como por parte de los tutores. Estas revisiones se realizarán periódicamente durante el desarrollo del TFG, con el objetivo de detectar y corregir posibles errores, inconsistencias o mejoras en la redacción, formato y contenido de los documentos.

Calidad del Código

En cuanto a la calidad del código, se empleará una herramienta de análisis de la calidad de código para evaluar diversos aspectos como la mantenibilidad, seguridad, eficiencia y cumplimiento de buenas prácticas de programación. Se establecerá un umbral mínimo de calidad, con la meta de alcanzar un nivel de calidad “A”. El umbral inicial se establecerá en “B”, y se realizarán esfuerzos constantes para mejorar la calidad del código hasta alcanzar el nivel deseado.

5. Estructura de Desglose del Trabajo (EDT) y Diccionario

Conforme al PMBOK versión 6 (Project Management Institute, Inc., 2017), la EDT se define como una herramienta clave para descomponer el alcance del proyecto en componentes manejables y jerárquicos. En este contexto, la figura 5.1 actúa como un mapa visual que ayuda a entender la totalidad del trabajo requerido en el proyecto. Por otro lado, el diccionario de la EDT proporciona detalles adicionales sobre cada elemento de la EDT, incluyendo descripciones, responsabilidades, criterios de aceptación y otras informaciones relevantes para la gestión del proyecto.

En este caso, se ha definido una estructura de la EDT *Hybrid Work Breakdown Structure* o EDT Orientada a Proyecto. Se ha decidido utilizar las iteraciones como paquetes de planificación. En este apartado no se contempla el número de entregables a realizar, puesto que esto ya ha sido especificado en el enunciado del alcance detalladamente.

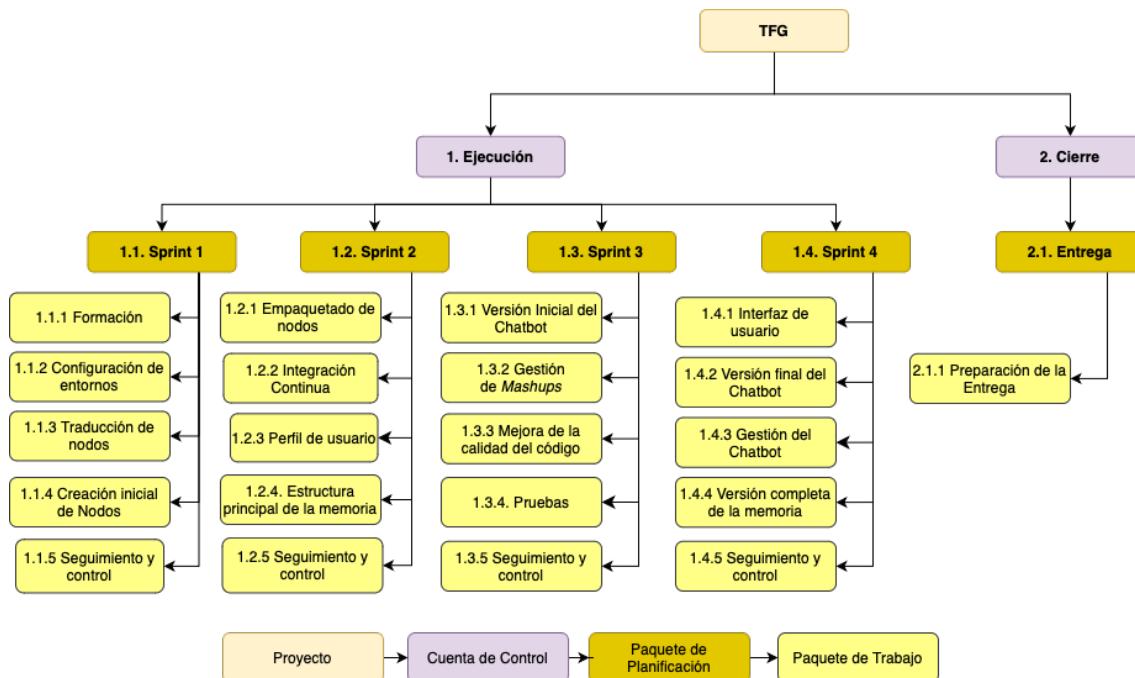


Figura 5.1: Estructura de Desglose del Trabajo (EDT)

Dentro de cada iteración o *Sprint*, se ha incorporado una fase de seguimiento y control de manera integral, reconociendo su vital importancia para garantizar el cumplimiento de objetivos y la detección temprana de desviaciones. Esta estrategia permite una gestión proactiva del proyecto al facilitar la identificación de riesgos y la implementación de ajustes necesarios en tiempo real, optimizando así la calidad y eficiencia de la ejecución.

5.1. Fase de Ejecución

A continuación se muestran los paquetes de trabajo de forma detallada de cada uno de los paquetes de planificación o *Sprints*. Los criterios de aceptación mencionados en cada paquete de trabajo representan criterios de aceptación a nivel de detalle, coexistiendo de manera simultánea

con los criterios de aceptación del proyecto en general. A partir de este punto, se empleará la abreviatura “PT” en lugar de “Paquete de Trabajo” para simplificar la denominación en las figuras.

Es necesario mencionar que las tareas realizadas junto con Francisco, técnico de STATUS, están exclusivamente relacionadas con la formación y configuración del sistema (tareas que hemos realizado ambos por separado). Cualquier otra tarea incluida en esta planificación no está relacionada con el trabajo realizado por dicho técnico.

5.1.1. Sprint 1

	Este paquete está compuesto por actividades relacionadas con la mejora de las habilidades y conocimientos.		
Entregables	Este paquete corresponde al entregable del prototipo del producto mínimo viable, en concreto al desarrollo y despliegue de nodos personalizados para Node-RED		
Actividad	Duración estimada	Puntos de historia	Recursos
1.1.1.1 Introducción a Node-RED	2 h	3	Álvaro Bernal
1.1.1.2 Introducción a Governify y Bluejay	6 h	8	Álvaro Bernal Francisco Montero
1.1.1.3 Introducción al <i>Collector-Events</i> de Bluejay	3 h	5	Álvaro Bernal Francisco Montero
Criterios de aceptación	<ul style="list-style-type: none"> ■ Demostrar un nivel básico de comprensión de las herramientas y dominio del material de formación. 		

Cuadro 5.1: [PT 1.1.1] Formación

Descripción	Debido a la restricción de uso de herramientas concretas, estas tienen una configuración determinada que tiene que ser tenida en cuenta		
Entregables	Al influir en actividades que generan a largo plazo varios entregables, este paquete no genera un entregable en concreto, aunque está directamente relacionado con el producto mínimo viable.		
Actividad	Duración estimada	Puntos de historia	Recursos
1.1.2.1 Configuración local de Node-RED	1 h	1	Álvaro Bernal
1.1.2.2 Configuración de contenedores Docker de Bluejay ¹	17 h	13	Álvaro Bernal Francisco Montero
1.1.2.3 Configuración del Collector-events en local	1 h	3	Álvaro Bernal Francisco Montero
Criterios de aceptación	<ul style="list-style-type: none"> ■ Se han realizado ejemplos prácticos una vez ya configurados los entornos. 		

¹ La estimación de 17 horas para la tarea se debe a las dificultades que Docker presenta en los Mac con arquitectura arm64.

Cuadro 5.2: [PT 1.1.2] Configuración de entornos

La consideración de las tareas de formación y configuración de Bluejay (cuadros 5.1 y 5.2) se debe a la necesidad de comprender el funcionamiento de los sistemas con los que se integrará, así como los formatos de respuestas necesarios para su correcto funcionamiento. Esto es especialmente relevante para el desarrollo de los nodos personalizados de Node-RED.

Descripción	Recopilar la información necesaria para crear nodos útiles para la generación de <i>mashups</i> a partir de ejemplos reales		
Entregables	Este paquete está directamente relacionado con el desarrollo y despliegue de nodos personalizados para Node-RED		
Actividad	Duración estimada	Puntos de historia	Recursos
1.1.3.1 Estudio de viabilidad de licencias de Enterprise Architect para acceder a los ejemplos reales	1 h	1	Álvaro Bernal
1.1.3.2 Recopilar información de los elementos del proyecto PROCEDE	3 h	3	Álvaro Bernal
1.1.3.3 Documento de <i>mashups</i> de PROCEDE	6 h	8	Álvaro Bernal
Criterios de aceptación	<ul style="list-style-type: none"> ■ La información recogida corresponde con la proporcionada por los elementos que aparecen en dichos ejemplos. ■ Recibe la aprobación de alguno de los tutores 		

Cuadro 5.3: [PT 1.1.3] Traducción de nodos

Descripción	Primera versión de los componentes útiles para la comprobación de la conformidad.		
Entregables	Este paquete corresponde al entregable del prototipo del producto mínimo viable, en concreto a despliegue de nodos personalizados para Node-RED.		
Actividad	Duración estimada	Puntos de historia	Recursos
1.1.4.1 <i>Filter-By</i>	1 h	3	Álvaro Bernal
1.1.4.2 <i>Filter-By-Date</i>	1 h	1	Álvaro Bernal
1.1.4.3 <i>For-Each-Project</i>	2 h	3	Álvaro Bernal
1.1.4.4 <i>Url-To-Document</i>	2 h	5	Álvaro Bernal
1.1.4.5 <i>Project-To-Url</i>	2 h	3	Álvaro Bernal
1.1.4.6 <i>Exists-Section-In-Doc</i>	2 h	3	Álvaro Bernal
Criterios de aceptación	<ul style="list-style-type: none"> ■ Se han realizado ejemplos con los nodos implementados. ■ Se ha recibido la aprobación por parte del Tutor. 		

Cuadro 5.4: [PT 1.1.4] Creación inicial de nodos

Descripción	Paquete orientado al seguimiento y control del proyecto de forma iterativa e incremental		
Entregables	Este paquete corresponde al entregable del prototipo del producto mínimo viable		
Actividad	Duración estimada	Puntos de historia	Recursos
1.1.5.1 Revisión del <i>Sprint</i>	1 h	1	Álvaro Bernal
1.1.5.2 Retrospectiva del <i>Sprint</i>	2 h	3	Álvaro Bernal
1.1.5.3 Lecciones aprendidas	1 h	1	Álvaro Bernal
1.1.5.4 Seguimiento de la calidad y recursos	1 h	1	Álvaro Bernal
1.1.5.5 Informe general del proyecto	2 h	3	Álvaro Bernal
Criterios de aceptación	<ul style="list-style-type: none"> ■ El trabajo realizado ha sido revisado por el tutor y en caso de no cumplir con los requisitos se han identificado los cambios necesarios 		

Cuadro 5.5: [PT 1.1.5] Seguimiento y control

5.1.2. Sprint 2

Descripción	Este paquete tiene como objetivo publicar en el gestor de dependencias npm los nodos creados.		
Entregables	Este paquete corresponde al entregable del prototipo del producto mínimo viable, en concreto a despliegue de nodos personalizados para Node-RED.		
Actividad	Duración estimada	Puntos de historia	Recursos
1.2.1.1 Investigar cómo se publican nodos en Node-RED	3 h	3	Álvaro Bernal
1.2.1.2 Publicación inicial de los nodos	3 h	3	Álvaro Bernal
1.2.1.3 Ejemplos de prueba con los nodos publicados	1 h	1	Álvaro Bernal
Criterios de aceptación	<ul style="list-style-type: none"> ■ Se han realizado ejemplos con los nodos publicados. ■ Se ha recibido la aprobación por parte de los tutores. 		

Cuadro 5.6: [PT 1.2.1] Empaque de nodos

Descripción	Automatización del despliegue local y publicación del paquete.		
Entregables	Este paquete corresponde al entregable del prototipo del producto mínimo viable, en concreto a despliegue de nodos personalizados para Node-RED.		
Actividad	Duración estimada	Puntos de historia	Recursos
1.2.2.1 Workflow de publicación en npm	1 h	3	Álvaro Bernal
1.2.2.2 Workflow de publicación en dockerhub	1 h	3	Álvaro Bernal
1.2.2.3 Instanciar un repositorio de despliegue local de Node-RED	1 h	1	Álvaro Bernal
1.2.2.4 Script de auto- configuración local de Node-RED	1 h	3	Álvaro Bernal
1.2.2.5 Despliegue del proyecto en Docker	2 h	3	Álvaro Bernal
Criterios de aceptación	<ul style="list-style-type: none"> ■ Al hacer un lanzamiento desde el repositorio de github, se publica de forma automática una nueva versión del paquete en npm. ■ Siguiendo los pasos del manual, se despliega node-RED en local. 		

Cuadro 5.7: [PT 1.2.2] Integración continua

Descripción	Sesión y perfil de usuarios		
Entregables	Este paquete corresponde al entregable del prototipo del producto mínimo viable		
Actividad	Duración estimada	Puntos de historia	Recursos
1.2.3.1 <i>Endpoints</i> de gestión de usuarios	1 h	1	Álvaro Bernal
1.2.3.2 Inicio de Sesión	2 h	3	Álvaro Bernal
1.2.3.3 Autenticación de usuarios	2 h	1	Álvaro Bernal
1.2.3.4 Renovación de sesión	1 h	1	Álvaro Bernal
1.2.3.5 Conexión con Github y Trello	2 h	3	Álvaro Bernal
Criterios de aceptación	<ul style="list-style-type: none"> ■ Se pueden realizar los casos de uso mencionados en las correspondientes historias de usuario (Sección 3.1.5) 		

Cuadro 5.8: [PT 1.2.3] Perfil de usuario

Descripción	Versión inicial de la memoria en LaTeX.		
Entregables	Este paquete corresponde al entregable de la memoria del proyecto		
Actividad	Duración estimada	Puntos de historia	Recursos
1.2.4.1 Inicializar el proyecto en LaTeX	2 h	1	Álvaro Bernal
1.2.4.2 Estructura inicial	1 h	1	Álvaro Bernal
1.2.4.3 Introducción	1 h	1	Álvaro Bernal
1.2.4.4 Alcance del proyecto	1 h	1	Álvaro Bernal
1.2.4.5 Especificación de Requisitos	5 h	5	Álvaro Bernal
1.2.4.6 Realización de los mockups	10 h	11	Álvaro Bernal
Criterios de aceptación	<ul style="list-style-type: none"> ■ La estructura de la memoria ha sido validada por los tutores. ■ Los apartados realizados han sido aprobados por los tutores. 		

Cuadro 5.9: [PT 1.2.4] Estructura principal de la memoria

Descripción	Paquete orientado al seguimiento y control del proyecto de forma iterativa e incremental		
Entregables	Este paquete corresponde al entregable del prototipo del producto mínimo viable		
Actividad	Duración estimada	Puntos de historia	Recursos
1.2.5.1 Revisión del <i>Sprint</i>	1 h	1	Álvaro Bernal
1.2.5.2 Retrospectiva del <i>Sprint</i>	2 h	3	Álvaro Bernal
1.2.5.3 Lecciones aprendidas	1 h	1	Álvaro Bernal
1.2.5.4 Seguimiento de la calidad y recursos	1 h	1	Álvaro Bernal
1.2.5.5 Informe general del proyecto	2 h	3	Álvaro Bernal
Criterios de aceptación	<ul style="list-style-type: none"> ■ El trabajo realizado ha sido revisado por el tutor y en caso de no cumplir con los requisitos se han identificado los cambios necesarios 		

Cuadro 5.10: [PT 1.2.5] Seguimiento y control

5.1.3. Sprint 3

Descripción	Implementación de las historias de usuario principales del chatbot (Sección 3.1.3)		
Entregables	Este paquete corresponde al entregable del prototipo del producto mínimo viable		
Actividad	Duración estimada	Puntos de historia	Recursos
1.3.1.1 Formación sobre la herramienta del chatbot	15 h	13	Álvaro Bernal
1.3.1.1 <i>Endpoints</i> de creación de chats	2 h	3	Álvaro Bernal
1.3.1.2 Vista del Chatbot	2 h	3	Álvaro Bernal
1.3.1.3 Historial de chats	1 h	1	Álvaro Bernal
1.3.1.4 Descripción de mashup	1 h	1	Álvaro Bernal
1.3.1.5 Autogenerar mashups	2 h	3	Álvaro Bernal
Criterios de aceptación	<ul style="list-style-type: none"> ■ Se pueden realizar los casos de uso mencionados en las correspondientes historias de usuario 		

Cuadro 5.11: [PT 1.3.1] Versión inicial del chatbot

Descripción	Operaciones sobre mashups en el sistema y en Node-RED		
Entregables	Este paquete corresponde al entregable del prototipo del producto mínimo viable		
Actividad	Duración estimada	Puntos de historia	Recursos
1.3.2.1 Listado de mashups de node-RED	2 h	3	Álvaro Bernal
1.3.2.2 Creación de mashups mediante API de node-RED	1 h	1	Álvaro Bernal
1.3.2.3 Eliminación de mashups mediante API de node-RED	1 h	1	Álvaro Bernal
1.3.2.4 Autorización de node-RED	0,5 h	1	Álvaro Bernal
Criterios de aceptación	<ul style="list-style-type: none"> ■ Se han corregido las incidencias de código encontradas. ■ No genera nuevas incidencias. 		

Cuadro 5.12: [PT 1.3.2] Gestión de *mashups*

Descripción	Actividades relacionadas con la mejora de la calidad		
Entregables	Este paquete corresponde al entregable del prototipo del producto mínimo viable		
Actividad	Duración estimada	Puntos de historia	Recursos
1.3.3.1 Refactorización ¹	3 h	5	Álvaro Bernal
Criterios de aceptación	<ul style="list-style-type: none"> ■ Se han corregido las incidencias de código encontradas. ■ No genera nuevas incidencias. 		

¹ Este paquete se centra en una única actividad, bajo la suposición de que la generación de incidencias de código será mínima.

Cuadro 5.13: [PT 1.3.3] Mejora de la calidad del código

Descripción	Paquete orientado a la realización de pruebas de los casos de uso principales		
Entregables	Este paquete corresponde al entregable del prototipo del producto mínimo viable		
Actividad	Duración estimada	Puntos de historia	Recursos
1.3.4.1 Pruebas de API	3 h	3	Álvaro Bernal
1.3.4.2 Investigar sobre las pruebas de Node-RED	6 h	8	Álvaro Bernal
1.3.4.3 Pruebas de Node-RED	4 h	5	Álvaro Bernal
1.3.4.4 Pruebas de calidad de <i>prompts</i>	4 h	1	Álvaro Bernal
Criterios de aceptación	<ul style="list-style-type: none"> ■ Las pruebas comprueban los casos de uso principales del proyecto. ■ Se han implementado pruebas de las partes del código más complejas. 		

Cuadro 5.14: [PT 1.3.4] Pruebas

Descripción	Paquete orientado al seguimiento y control del proyecto de forma iterativa e incremental		
Entregables	Este paquete corresponde al entregable del prototipo del producto mínimo viable		
Actividad	Duración estimada	Puntos de historia	Recursos
1.3.5.1 Revisión del <i>Sprint</i>	1 h	1	Álvaro Bernal
1.3.5.2 Retrospectiva del <i>Sprint</i>	2 h	3	Álvaro Bernal
1.3.5.3 Lecciones aprendidas	1 h	1	Álvaro Bernal
1.3.5.4 Seguimiento de la calidad y recursos	1 h	1	Álvaro Bernal
1.3.5.5 Informe general del proyecto	2 h	3	Álvaro Bernal
Criterios de aceptación	<ul style="list-style-type: none"> ■ El trabajo realizado ha sido revisado por el tutor y en caso de no cumplir con los requisitos se han identificado los cambios necesarios 		

Cuadro 5.15: [PT 1.3.5] Seguimiento y control

5.1.4. Sprint 4

Descripción	Mejora de la <i>UI</i> y de la <i>UX</i>		
Entregables	Este paquete corresponde a la entrega del producto final		
Actividad	Duración estimada	Puntos de historia	Recursos
1.4.1.1 <i>Skeleton loaders</i>	2 h	3	Álvaro Bernal
1.4.1.2 Mejorar el estilo del chat	4 h	3	Álvaro Bernal
1.4.1.3 Animación de elementos	2 h	1	Álvaro Bernal
Criterios de aceptación	<ul style="list-style-type: none"> ■ Se ha conseguido una mejora de la experiencia de usuario. 		

Cuadro 5.16: [PT 1.4.1] Interfaz de usuario

Descripción	Entrega de la documentación y del producto.		
Entregables	Este paquete corresponde a la entrega del producto final		
Actividad	Duración estimada	Puntos de historia	Recursos
1.4.2.1 Chat de asistencia a node-RED	4 h	5	Álvaro Bernal
1.4.2.2 Recomendación de descripciones	3 h	3	Álvaro Bernal
1.4.2.3 Creación de <i>mashups</i> auto-generados en Node-RED	2 h	3	Álvaro Bernal
1.4.2.4 Generar y almacenar "detalle" del <i>mashup</i>	3 h	5	Álvaro Bernal
Criterios de aceptación	<ul style="list-style-type: none"> ■ Se pueden realizar los casos de uso mencionados en las correspondientes historias de usuario (Sección 3.1.4) 		

Cuadro 5.17: [PT 1.4.2] Versión final del chatbot

Descripción	Listado de chats, activar/desactivar chat y limitaciones de uso		
Entregables	Este paquete corresponde al entregable del prototipo del producto mínimo viable		
Actividad	Duración estimada	Puntos de historia	Recursos
1.4.3.1 Activar/Desactivar chat	2 h	3	Álvaro Bernal
1.4.3.2 Limitación del chat	3 h	3	Álvaro Bernal
1.4.3.3 Listado de chats	2 h	3	Álvaro Bernal
Criterios de aceptación	<ul style="list-style-type: none"> ■ Se pueden realizar los casos de uso mencionados en las correspondientes historias de usuario (Sección 3.1.1) 		

Cuadro 5.18: [PT 1.4.3] Gestión del chatbot

Descripción	Versión completa de la memoria		
Entregables	Este paquete corresponde al entregable del prototipo del producto mínimo viable		
Actividad	Duración estimada	Puntos de historia	Recursos
1.4.4.1 Metodología de trabajo	1 h	1	Álvaro Bernal
1.4.4.2 EDT y diccionario	4 h	5	Álvaro Bernal
1.4.4.3 Estimaciones de Tiempo, Costes y Cronograma	3 h	5	Álvaro Bernal
1.4.4.4 Diseño y Arquitectura del Producto	4 h	5	Álvaro Bernal
1.4.4.5 Tecnologías y herramientas	3 h	1	Álvaro Bernal
1.4.4.6 Desarrollo del producto	4 h	3	Álvaro Bernal
1.4.4.7 Resultados	3 h	3	Álvaro Bernal
1.4.4.8 Conclusiones	1 h	1	Álvaro Bernal
1.4.4.9 Anexos	4 h	3	Álvaro Bernal
Criterios de aceptación	<ul style="list-style-type: none"> ■ Los apartados realizados han sido aprobados por los tutores. 		

Cuadro 5.19: [PT 1.4.4] Versión completa de la memoria

Descripción	Paquete orientado al seguimiento y control del proyecto de forma iterativa e incremental		
Entregables	Este paquete corresponde al entregable del prototipo del producto mínimo viable		
Actividad	Duración estimada	Puntos de historia	Recursos
1.4.5.1 Revisión del <i>Sprint</i>	1 h	1	Álvaro Bernal
1.4.5.2 Retrospectiva del <i>Sprint</i>	2 h	3	Álvaro Bernal
1.4.5.3 Lecciones aprendidas	1 h	1	Álvaro Bernal
1.4.5.4 Seguimiento de la calidad y recursos	1 h	1	Álvaro Bernal
1.4.5.5 Informe general del proyecto	2 h	3	Álvaro Bernal
Criterios de aceptación	<ul style="list-style-type: none"> ■ El trabajo realizado ha sido revisado por el cliente y en caso de no cumplir con los requisitos se han identificado los cambios necesarios 		

Cuadro 5.20: [PT 1.4.5] Seguimiento y control

5.2. Fase de Cierre

En la fase de cierre del proyecto, el desglose de los paquetes de trabajo se enfoca en las actividades finales necesarias para garantizar la entrega exitosa del proyecto. En proyectos de mayor tamaño se utilizan varios paquetes de trabajo para cada conjunto de actividades, sin embargo, debido al tamaño de este proyecto se ha decidido agrupar las actividades en un paquete de trabajo en

concreto.

Descripción	Preparación de la entrega y el cierre del proyecto		
Entregables	Este paquete corresponde a la entrega del producto final		
Actividad	Duración estimada	Puntos de historia	Recursos
2.1.1.1 Acciones de cierre	1 h	1	Álvaro Bernal
2.1.1.2 Acuerdo de Cierre con los tutores	1 h	1	Álvaro Bernal
2.1.1.3 Lecciones aprendidas	1 h	1	Álvaro Bernal
2.1.1.4 Presentación del TFG	12 h	11	Álvaro Bernal
2.1.1.5 Documentación online del TFG ¹	10 h	8	Álvaro Bernal
Criterios de aceptación	<ul style="list-style-type: none"> ■ Revisión y aprobación de los involucrados ■ Aprobación por parte de los tutores de los documentos de cierre generados 		

¹ Realizar una documentación online orientada al proyecto de investigación, en la que se incluya la información relevante del TFG para este.

Cuadro 5.21: [PT 2.1.1] Preparación de la entrega

6. Estimaciones de Tiempo, Costes y Cronograma

En PMBOK se plantean varias técnicas de estimación de las actividades, para este trabajo se ha decidido utilizar la estimación basada en tres valores para las estimaciones de tiempo y de costes (Project Management Institute, Inc. (2017), pag 201) y la distribución triangular para los cálculos de tiempos y costes esperados. Para mejorar la calidad de la lectura, las estimaciones se dividirán por Sprints (o Paquetes de Trabajo).

6.1. Estimaciones de Tiempo

Actividad	Tiempo Estimado			Tiempo Esperado
	Estimación Optimista	Estimación Más Probable	Estimación Pesimista	
Introducción a Node-RED	1 h	2 h	3 h	2 h
Introducción a Governify y Bluejay	3 h	6 h	9 h	6 h
Introducción al <i>Collector-Events</i> de Bluejay	1,5 h	3 h	4,5 h	3 h
Configuración local de Node-RED	0,5 h	1 h	1,5 h	1 h
Configuración de contenedores docker de Bluejay	10 h	17 h	24 h	17 h
Configuración del <i>Collector-events</i> en local	0,5 h	1 h	1,5 h	1 h
Estudio de viabilidad de licencias de Enterprise Architect	0,5 h	1 h	1,5 h	1 h
Recopilar información de los elementos del proyecto PROCEDE	1,5 h	3 h	4,5 h	3 h
Documento de <i>mashups</i> de PROCEDE	3 h	6 h	9 h	6 h
<i>Filter-By</i>	0,5 h	1 h	1,5 h	1 h
<i>Filter-By-Date</i>	0,5 h	1 h	1,5 h	1 h
<i>For-Each-Project</i>	1 h	2 h	3 h	2 h
<i>Url-To-Document</i>	1 h	2 h	3 h	2 h
<i>Project-To-Url</i>	1 h	2 h	3 h	2 h
<i>Exists-Section-In-Doc</i>	1 h	2 h	3 h	2 h
Revisión del <i>Sprint</i>	0,5 h	1 h	1,5 h	1 h
Retrospectiva del <i>Sprint</i>	1 h	2 h	3 h	2 h
Lecciones aprendidas	0,5 h	1 h	1,5 h	1 h
Seguimiento de la calidad y recursos	0,5 h	1 h	1,5 h	1 h
Informe de avance del proyecto	1 h	2 h	3 h	2 h
Total	30 h	57 h	84 h	57 h

Cuadro 6.1: Estimación de tiempo de actividades del Sprint 1

Actividad	Tiempo Estimado			Tiempo Esperado
	Estimación Optimista	Estimación Más Probable	Estimación Pesimista	
Investigar cómo se publican nodos en Node-RED	1 h	3 h	5 h	3 h
Publicación inicial de los nodos	2 h	5 h	8 h	5 h
Ejemplos de prueba con los nodos publicados	0,5 h	1 h	1,5 h	1 h
<i>Workflow</i> de publicación en npm	0,5 h	1 h	1,5 h	1 h
<i>Workflow</i> de publicación en dockerhub	0,5 h	1 h	1,5 h	1 h
Instanciar un repositorio de despliegue local de Node-RED	0,5 h	1 h	1,5 h	1 h
Script de auto- configuración local de Node-RED	0,5 h	1 h	1,5 h	1 h
Despliegue del proyecto en Docker	1 h	2 h	3 h	2 h
<i>Endpoints</i> de gestión de usuarios	0,5 h	1 h	1,5 h	1 h
Inicio de Sesión	1 h	2 h	3 h	2 h
Autenticación de usuarios	1 h	2 h	3 h	2 h
Renovación de sesión	0,5 h	1 h	1,5 h	1 h
Conexión con Github y Trello	1 h	2 h	3 h	2 h
Inicializar el proyecto en LaTeX	1 h	2 h	3 h	2 h
Estructura inicial	0,5 h	1 h	1,5 h	1 h
Introducción	0,5 h	1 h	1,5 h	1 h
Alcance del proyecto	0,5 h	1 h	1,5 h	1 h
Especificación de Requisitos	2,5 h	5 h	7,5 h	5 h
Realización de los mockups	6 h	10 h	14 h	10 h
Revisión del Sprint	0,5 h	1 h	1,5 h	1 h
Retrospectiva del Sprint	1 h	2 h	3 h	2 h
Lecciones aprendidas	0,5 h	1 h	1,5 h	1 h
Seguimiento de la calidad y recursos	0,5 h	1 h	1,5 h	1 h
Informe de avance del proyecto	1 h	2 h	3 h	2 h
Total	26 h	50 h	75 h	50 h

Cuadro 6.2: Estimación de tiempo de actividades del Sprint 2

Actividad	Tiempo Estimado			Tiempo Esperado
	Estimación Optimista	Estimación Más Probable	Estimación Pesimista	
Formación sobre la herramienta del chatbot	8 h	15 h	22 h	15 h
<i>Endpoints</i> de creación de chats	1 h	2 h	3 h	2 h
Vista del Chatbot	1 h	2 h	3 h	2 h
Historial de chats	0,5 h	1 h	1,5 h	1 h
Descripción de mashup	0,5 h	1 h	1,5 h	1 h
Autogenerar mashups	1 h	2 h	3 h	2 h
Listado de mashups de Node-RED	1 h	2 h	3 h	2 h
Creación de mashups mediante API de Node-RED	0,5 h	1 h	1,5 h	1 h
Eliminación de mashups mediante API de Node-RED	0,5 h	1 h	1,5 h	1 h
Autorización de Node-RED	0,25 h	0,5 h	0,75 h	0,5 h
Refactorización	1,5 h	3 h	4,5 h	3 h
Pruebas de API	1,5 h	3 h	4,5 h	3 h
Investigar sobre las pruebas de Node-RED	2 h	6 h	10 h	6 h
Pruebas de Node-RED	2 h	4 h	6 h	4 h
Pruebas de calidad de <i>prompts</i>	2 h	4 h	6 h	4 h
Revisión del <i>Sprint</i>	0,5 h	1 h	1,5 h	1 h
Retrospectiva del <i>Sprint</i>	1 h	2 h	3 h	2 h
Lecciones aprendidas	0,5 h	1 h	1,5 h	1 h
Seguimiento de la calidad y recursos	0,5 h	1 h	1,5 h	1 h
Informe de avance del proyecto	1 h	2 h	3 h	2 h
Total	24,75 h	48,5 h	72,25 h	48,5 h

Cuadro 6.3: Estimación de tiempo de actividades Sprint 3

Actividad	Tiempo Estimado			Tiempo Esperado
	Estimación Optimista	Estimación Más Probable	Estimación Pesimista	
<i>Skeleton loaders</i>	1 h	2 h	3 h	2 h
Mejorar el estilo del chat	2 h	4 h	6 h	4 h
Animación de elementos	1 h	2 h	3 h	2 h
Chat de asistencia a Node-RED	2 h	4 h	6 h	4 h
Recomendación de descripciones	1,5 h	3 h	4,5 h	3 h
Creación de <i>mashups</i> autogenerados en Node-RED	1 h	2 h	3 h	2 h
Generar y almacenar "detalle" del <i>mashup</i>	1,5 h	3 h	4,5 h	3 h
Activar/Desactivar chat	1 h	2 h	3 h	2 h
Limitación del chat	1,5 h	3 h	4,5 h	3 h
Listado de chats	1 h	2 h	3 h	2 h
Metodología de trabajo	0,5 h	1 h	1,5 h	1 h
EDT y diccionario	2 h	4 h	6 h	4 h
Estimaciones de Tiempo, Costes y Cronograma	1,5 h	3 h	4,5 h	3 h
Diseño y Arquitectura del Producto	2 h	4 h	6 h	4 h
Tecnologías y herramientas	1,5 h	3 h	4,5 h	3 h
Desarrollo del producto	3 h	4 h	5 h	4 h
Resultados	1,5 h	3 h	4,5 h	3 h
Conclusiones	0,5 h	1 h	1,5 h	1 h
Anexos	3 h	4 h	5 h	4 h
Revisión del <i>Sprint</i>	0,5 h	1 h	1,5 h	1 h
Retrospectiva del <i>Sprint</i>	1 h	2 h	3 h	2 h
Lecciones aprendidas	0,5 h	1 h	1,5 h	1 h
Seguimiento de la calidad y recursos	0,5 h	1 h	1,5 h	1 h
Informe de avance del proyecto	1 h	2 h	3 h	2 h
Total	32,5 h	61 h	89,5 h	61 h

Cuadro 6.4: Estimación de tiempo de actividades del Sprint 4

Actividad	Tiempo Estimado			Tiempo Esperado
	Estimación Optimista	Estimación Más Probable	Estimación Pesimista	
Acciones de cierre	0,5 h	1 h	1,5 h	1 h
Acuerdo de Cierre con los tutores	0,5 h	1 h	1,5 h	1 h
Lecciones aprendidas	0,5 h	1 h	1,5 h	1 h
Presentación del TFG	8 h	12 h	16 h	12 h
Documentación online del TFG	6 h	10 h	14 h	10 h
Total	15,5 h	25 h	34,5 h	25 h

Cuadro 6.5: Estimación de tiempo de actividades del Cierre

En base a las estimaciones realizadas se obtendrían las siguientes estimaciones totales:

- Estimación total optimista: 128,75 h
- Estimación total más probable: 241,5
- Estimación total pesimista: 355,25

El resultado total muestra una diferencia significativa entre los tres tipos de estimaciones. Esta incertidumbre se debe al uso de Node-RED, una tecnología totalmente desconocida. Por ello, en caso de que durante las dos primeras fases de desarrollo se consigan realizar las tareas en un tiempo similar al optimista se deberá realizar una re-planificación o aumento del alcance del proyecto para cumplir con el mínimo establecido.

Finalmente, basándonos en la estimación realizada, se proyecta una duración aproximada de 8 meses, desde el inicio como técnico para la prueba de concepto STATUS (13 de octubre de 2023) hasta la entrega de la memoria del TFG (27 de mayo de 2024).

6.2. Estimaciones de Costes

Con el fin de mejorar la inteligibilidad de la memoria, se va a realizar una estimación por tres valores a nivel de Sprint de los costes asociados a la contratación de personal, en lugar de realizar el desglose a nivel de actividad. Para el cálculo de los costes se utilizará el perfil informático de “Técnico Especialista Junior”, este tipo de perfiles (según aparece en el informe de perfiles proporcionado en la documentación del TFG), corresponde a un precio de 18,14€/h.

Actividad	Coste Estimado			Coste Esperado
	Estimación Optimista	Estimación Más Probable	Estimación Pesimista	
[PT 1.1.1] Formación	99,77€	199,54€	299,31€	199,54€
[PT 1.1.2] Configuración de entornos	199,54€	326,52€	489,78€	326,52€

Continua en la siguiente página

Continuación del cuadro 6.6

Actividad	Coste Estimado			Coste Esperado
	Estimación Optimista	Estimación Más Probable	Estimación Pesimista	
[PT 1.1.3] Traducción de nodos	90,7€	181,4€	272,1€	181,4€
[PT 1.1.4] Creación inicial de nodos	90,7€	181,4€	272,1€	181,4€
[PT 1.1.5] Seguimiento y control	63,49€	126,98€	190,47€	126,98€
[PT 1.2.1] Empaqueado de nodos	63,49€	126,98€	263,03€	126,98€
[PT 1.2.2] Integración continua	54,42€	108,84€	163,26€	108,84€
[PT 1.2.3] Perfil de usuario	90,7€	145,12€	217,68€	145,12€
[PT 1.2.4] Estructura principal de la memoria	199,54€	362,8€	471,64€	362,8€
[PT 1.2.5] Seguimiento y control	63,49€	126,98€	190,47€	126,98€
[PT 1.3.1] Versión inicial del chatbot	217,68€	417,22€	616,76€	417,22€
[PT 1.3.2] Gestión de mashups	40,81€	81,63€	122,45€	81,63€
[PT 1.3.3] Mejora de la calidad del código	27,21€	54,42€	81,63€	54,42€
[PT 1.3.4] Pruebas	136,05€	308,38€	480,71€	308,38€
[PT 1.3.5] Seguimiento y control	63,49€	126,98€	190,47€	126,98€
[PT 1.4.1] Interfaz de usuario	72,56€	145,12€	217,68€	145,12€

Continua en la siguiente página

Continuación del cuadro 6.6

Actividad	Coste Estimado			Coste Esperado
	Estimación Optimista	Estimación Más Probable	Estimación Pesimista	
[PT 1.4.2] Versión final del chatbot	108,84€	217,68€	326,52€	217,68€
[PT 1.4.3] Gestión del chatbot	63,49€	126,98€	190,47€	126,98€
[PT 1.4.4] Versión completa de la memoria	281,17€	490,59€	698,39€	490,59€
[PT 1.4.5] Seguimiento y control	63,49€	126,98€	190,47€	126,98€
[PT 2.1.1] Preparación de la entrega	272,1€	453€	625,83€	453€
Total	2.326,45€	4.380,31€	6.444,23€	4.380,31€

Cuadro 6.6: Estimación de costes de las actividades

Los costes de contratación de personal no se limitan simplemente al salario bruto del empleado. La empresa también incurre en costes adicionales, como contribuciones a la seguridad social y otros beneficios adicionales. En el cuadro 6.7 se hace un desglose de los gastos de contratación de personal.

Concepto	Base	Porcentaje	Coste
Contingencias comunes	4.380,31€	23,6 %	1.033,75€
Desempleo	4.380,31€	5,5 %	240,92€
I. profesional	4.380,31€	0,6 %	26,28€
MEI	4.380,31€	0,58 %	25,41€
AT y EP	4.380,31€	1,5 %	65,70€
FOGASA	4.380,31€	0,2 %	8,76€
Total Impuestos	-	-	1.400,82€
Base imponible	-	-	4.380,31€
Coste Total de Personal	-	-	5.781,13€

Cuadro 6.7: Desglose del coste de personal desde el punto de vista de la Empresa

En cuanto a los costes de amortización de recursos, estos pueden incluir la depreciación de activos tangibles, como equipos informáticos y software especializado utilizados en el proyecto. Para el cálculo del coste de amortización de un portátil de 1000€ durante 8 meses, es crucial determinar el método de amortización aplicable. Según la información proporcionada por Zamora (2022), parece que debemos evaluar el rendimiento de los equipos informáticos mediante el

método de estimación directa. En este contexto, los ordenadores se amortizan como máximo en 10 años, según la tabla de amortizaciones simplificada de la Agencia Tributaria¹.

El método de amortización más común es aplicar un porcentaje fijo sobre el precio de adquisición. Sin embargo, dado que el período de posesión es de solo 8 meses (aproximadamente), no se amortizará el coste completo del portátil en este tiempo. En lugar de eso, se amortizará una parte proporcional del coste total. La fórmula para calcular la amortización proporcional en este caso es:

$$\text{Amortización proporcional} = \left(\frac{\text{Coste total}}{\text{Total de meses de vida útil}} \right) \times \text{Meses de posesión}$$

Dado que el portátil tiene un coste de 1000€ y la vida útil máxima es de 10 años (120 meses), la fórmula se convierte en:

$$\text{Amortización proporcional} = \left(\frac{1000}{120} \right) \times 8 = \frac{1000}{15} \approx 66,67$$

Por lo tanto, el coste de amortización del portátil de 1000€ durante 8 meses sería de aproximadamente 66,67€.

Además del coste de la contratación de personal y los costes de amortización de recursos, se deben incluir en los costes del proyecto una reserva de gestión de 876€ para contingencias no identificadas o cambios no anticipados que puedan surgir durante la ejecución del proyecto y los costes de contratación de servicios externos. Para los costes de los servicios externos se debe realizar un análisis previo de las alternativas de servicios de Procesamiento del Lenguaje Natural (del inglés, *Natural Language Processing*). Para realizar en análisis se escogerán cuatro empresas con grandes avances en el NLP; en concreto, IBM², Microsoft³, Amazon⁴, Google⁵ y OpenAI⁶.

Servicio	Precio Mensual (USD)	Precio por Transacción (USD)
IBM Watsonx Assistant	140	0 (\$0.14 a partir de los 1000 usuarios)
Microsoft LUIS	1123 (1M transacciones)	1,4 (gratis hasta 10,000 transacciones/mes)
Amazon Lex	– (basado en uso)	0,00075
Google Dialogflow	– (basado en uso)	0,002
OpenAI Assistant	– (basado en uso)	Depende de la entrada y de la salida

Cuadro 6.8: Comparación de precios de diferentes servicios de NLP

Cómo la mayoría de los costes de los servicios analizados en el 6.8 dependen de su uso, vamos a suponer un uso de 200 transacciones al mes. Además, debido a la complejidad de estimación del

¹https://sede.agenciatributaria.gob.es/Sede/ayuda/manuales-videos-folletos/manuales-practicos/folleto-actividades-economicas/3-impuesto-sobre-renta-personas-fisicas/3_5-estimacion-directa-simplificada/3_5_4-tabla-amortizacion-simplificada.html

²<https://www.ibm.com/products/watsonx-assistant/pricing>

³<https://azure.microsoft.com/es-es/pricing/details/cognitive-services/language-understanding-intelligent-services/>

⁴<https://aws.amazon.com/es/lex/pricing>

⁵<https://cloud.google.com/dialogflow/pricing>

⁶<https://openai.com/api/pricing/>

servicio de OpenAI, que viene dado por las *prompts* de entrada y de salida, se ha decidido descartar del cálculo. Resultando así los costes mostrados en el cuadro 6.9.

Servicio	Coste Mensual para 200 transacciones (USD)
IBM Watsonx Assistant	140
Microsoft LUIS	1123
Amazon Lex	$0,00075 \times 200 = 0,15$
Google Dialogflow	$0,002 \times 200 = 0,4$

Cuadro 6.9: Comparación para 200 transacciones mensuales

A pesar de que la diferencia de precios sea considerable, actualmente no se puede decidir qué servicio es el indicado para el desarrollo de este proyecto, por lo que se realizará una media aritmética de los costes de los cuatro servicios para utilizar esta estimación como punto de partida. Suponiendo un uso de cuatro meses para cada una, el resultado total ascendería a 1171,31€ (sin la conversión serían \$1263.55). Finalmente, en el cuadro 6.10 se muestra el desglose total del coste estimado inicialmente.

Concepto	Coste
Costes de personal	5.781,13€
Servicio externo de NLP	1.171,31€
Costes de amortización de recursos	66.67€
Reserva de gestión	876€
Subtotal	7.895,11€
IVA(21 %)	1.657,97€
Total	9.553,08€

Cuadro 6.10: Coste total planificado

6.3. Cronograma

Una vez realizado el desglose de las actividades y sus tiempos, es importante situar el proyecto en un instante de tiempo determinado. El inicio del trabajo tiene lugar el día 13 de Octubre, día en el que comienza el desarrollo de la prueba de concepto STATUS (mencionado en el apartado 1). Cabe destacar que para realizar el cronograma que incluye la programación de realización de tareas, se ha tenido en cuenta que sólo una parte del proyecto de investigación tiene relación con este TFG. Por ello, se ha decidido usar un horario para el TFG de 4 horas semanales los días laborables, quedando así el cronograma de la Figura 6.1(de alto nivel para que sea más fácil de comprender).

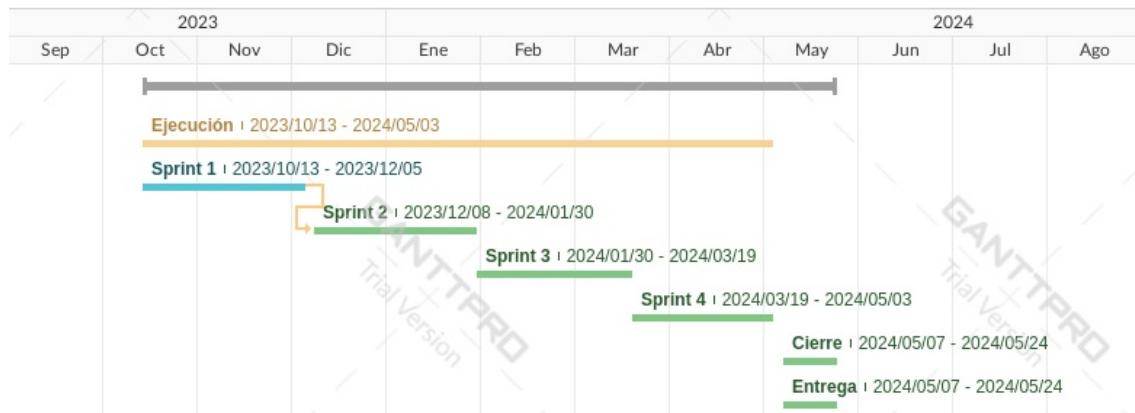


Figura 6.1: Cronograma de alto nivel

Como podemos ver en la Figura 6.1, la fecha de realización de la fase de ejecución da comienzo el día 13/10/2023 (una vez ya realizadas las fases de Inicio y Planificación) y finaliza en la fase de cierre el día 24/05/2024. Por otro lado, en el tiempo esperado en el apartado 6.1 no se han contemplado el horario laboral comentado anteriormente, por lo que en el cronograma (Figura 6.2) aparece una duración de 257 horas, como podemos ver en la siguiente figura.

Universidad de Sevilla | TFG

Nombre de tarea	Fecha de inicio	Duración	Fecha final
	2023/10/13	257h	2024/05/24
1 ▣ Ejecución	2023/10/13	233h	2024/05/03
1.1 ▣ Sprint 1	2023/10/13	58h	2023/12/05
1.2 ▣ Sprint 2	2023/12/08	63h	2024/01/30
1.3 ▣ Sprint 3	2024/01/30	55h	2024/03/19
1.4 ▣ Sprint 4	2024/03/19	54h	2024/05/03
2 ▣ Cierre	2024/05/07	24h	2024/05/24
2.1 ▣ Entrega	2024/05/07	24h	2024/05/24

Figura 6.2: Cuentas de control

Además, como el proyecto se está realizando de forma paralela a la realización del último año de la titulación, no se contempla un horario de trabajo ininterrumpido, por lo que se puede afirmar que el concepto de camino crítico definido en el PMBOK (Project Management Institute, Inc., 2017) no aplica. En caso de que hubiese retrasos en tareas que son predecesoras a otras, al haber holgura entre ellas se puede evitar el retraso en el cierre del proyecto (aún habiendo alguna varianza en el cronograma).

7. Diseño y Arquitectura del Producto

Dado el contexto de este proyecto y el avance paralelo del proyecto de investigación, es importante distinguir las partes del sistema que se van a desarrollar. Además, se planteará un diseño inicial del núcleo central del trabajo y se comentarán las ventajas e inconvenientes del diseño y la arquitectura escogidos.

7.1. Diseño y Arquitectura

En base a los requisitos definidos, se ha realizado un diagrama con la arquitectura lógica del sistema (ver la Figura 7.1). Esta arquitectura se corresponde al estilo de arquitectura basado en capas. Según el libro de patrones de arquitectura software de Richards (2015), “la arquitectura en capas se basa en la organización de componentes en capas horizontales con roles específicos dentro de la aplicación (como lógica de presentación, lógica de negocio, lógica de persistencia, etc.)”.

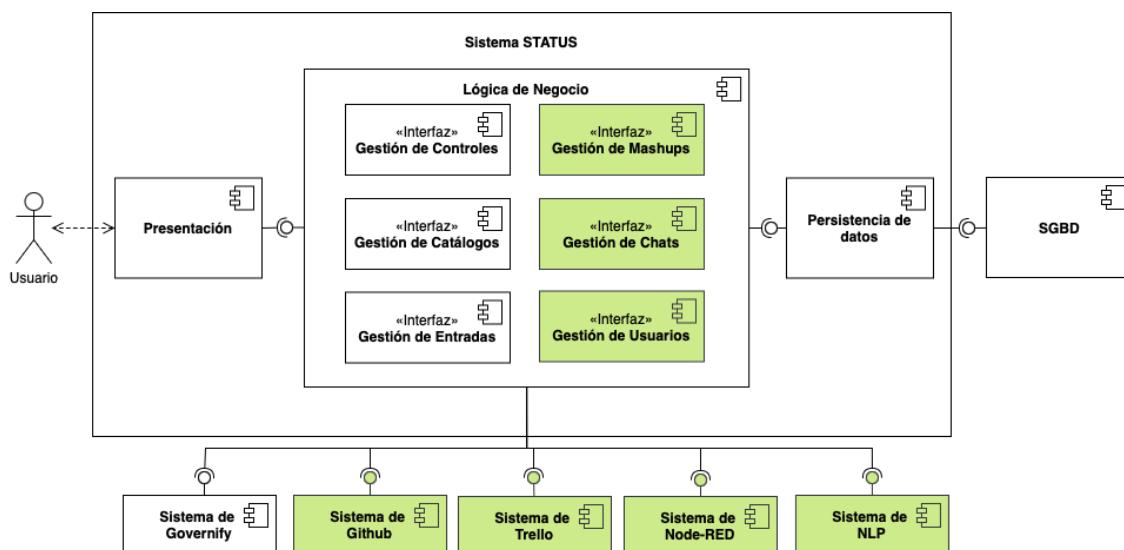


Figura 7.1: Diagrama de arquitectura lógica

Los componentes coloreados de verde en la capa de lógica de negocio del diagrama son los que se implementarán en este trabajo, mientras que en el resto de las capas se realizarán modificaciones relacionadas con la funcionalidad mencionada en los requisitos. Por otro lado, se excluyen la gestión de catálogos, controles y entradas, ya que estas serán realizadas por el otro técnico de la prueba de concepto. A continuación, se describen las capas que componen el sistema:

- **Capa de Presentación:** Responsable de manejar la lógica de interfaz de usuario y comunicación con el navegador.
- **Capa de Lógica de Negocio:** Encargada de ejecutar las reglas de negocio específicas asociadas con las solicitudes y de gestionar la comunicación con servicios externos para integrar funcionalidades adicionales al sistema.
- **Capa de Persistencia de Datos:** Responsable de interactuar con la base de datos y gestionar la persistencia de los datos.

El sistema de gestión de base de datos (SGBD), situado fuera del sistema, es responsable de administrar y organizar la información almacenada en la base de datos. Además, el SGBD facilita la interacción entre el usuario y la base de datos, permitiendo realizar consultas, actualizaciones y mantenimiento de los datos de manera eficiente.

Por último, los sistemas externos a STATUS también están coloreados ya que serán integrados o consumidos. De estos, Trello y GitHub se utilizarán para ofrecer a los usuarios la posibilidad de conectar estas plataformas al sistema, Node-RED se empleará para la gestión de mashups, y, por último, el sistema de NLP se utilizará para gestionar y crear chats que brinden soporte a los usuarios del sistema.

7.1.1. Modelo de datos

En la Figura 7.2 se muestra el modelo de datos inicial del sistema (realizado para la prueba de concepto) mediante el diagrama de clases UML, proporcionando una representación visual de las clases y sus relaciones dentro del sistema.

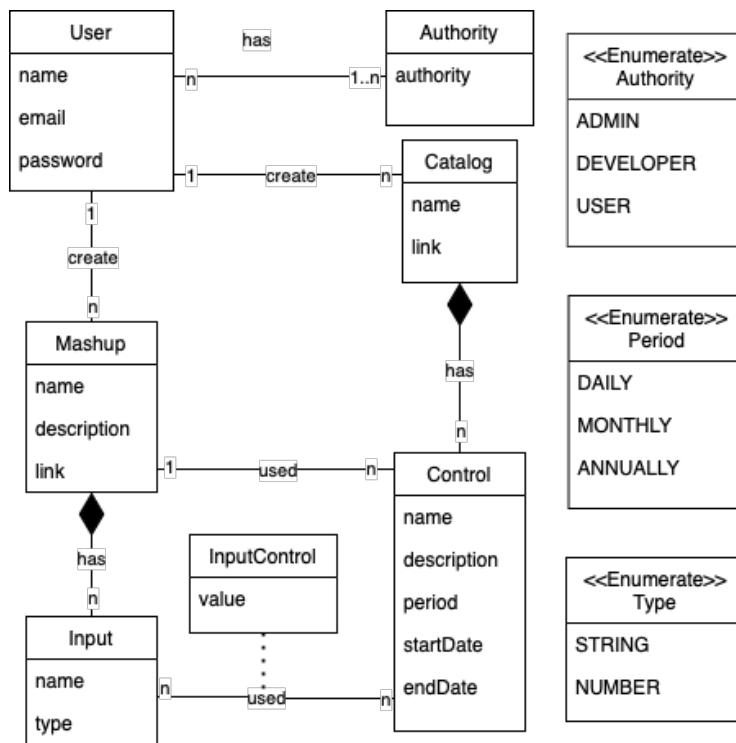


Figura 7.2: Diagrama de clases UML

El diagrama de clases UML del sistema STATUS se estructura en torno a las entidades principales y sus relaciones, detallando atributos clave que definen su funcionalidad y conexiones con otras entidades.

- **User**: Representa los usuarios del sistema con atributos como nombre, email y contraseña. Establece una relación de muchos a muchos (N:N) con la entidad Autoridad.
- **Authority**: Representa el nivel de autorización que tiene el usuario en el sistema; puede ser administrador, desarrollador o usuario.
- **Catalog**: Describe los catálogos de controles disponibles con atributos como nombre y enlace al tablero con los resultados de cumplimiento. Se relaciona de uno a muchos (1:N) con la

entidad Control y de muchos a uno (N:1) con Usuario, indicando la pertenencia a un usuario específico.

- **Control:** Define los controles realizados con detalles como nombre, descripción, periodo, fecha de inicio y fecha de fin. Tiene relaciones de muchos a uno (N:1) con Mashup y relación de muchos a muchos (N:N) con Entrada.
- **Input:** Representa las entradas utilizadas en los controles, con atributos como nombre y tipo. Establece una relación de muchos a muchos (N:N) con Control, mediada por una clase de asociación *InputControl* con atributo adicional de valor. Además, tiene una relación de muchos a uno (N:1) con Mashup, indicando su asociación a uno de estos elementos.
- **Mashup:** Describe los mashups disponibles con detalles como nombre, descripción y enlace al mashup en Node-RED. Se relaciona de muchos a uno (N:1) con Usuario, señalando el usuario propietario del mashup.

Cabe destacar que, aunque se presente un diagrama de clases del sistema, la implementación de estas clases no se ha realizado de forma completa al inicio; hasta el momento, únicamente se han desarrollado las clases de Control, Catálogo, Entrada, Mashup e InputControl (realizado por Francisco Montero).

Inicialmente, no se planean modificaciones en el modelo de datos, ya que su diseño dependerá del servicio específico utilizado para el chatbot. La adaptación de clases y relaciones estará determinada por las necesidades particulares de cada servicio. Sin embargo, se anticipa que la clase creada mantendrá una relación esencial con la entidad de usuario (Figura 7.3).

7.1.2. Diseño del chatbot

El flujo de trabajo del chatbot se divide en dos escenarios principales, dependiendo del tipo de solicitud recibida:

- **Solicitud para Generar un Mashup**
 1. El usuario envía una descripción al chatbot.
 2. El chatbot identifica que la solicitud es para generar un mashup.
 3. El chatbot envía la descripción al servicio de descripción de mashups.
 4. El servicio de descripción de mashups devuelve un mashup.
 5. El chatbot recibe el mashup y lo presenta al usuario.
- **Solicitud para Generar una Descripción**
 1. El usuario envía un mashup al chatbot.
 2. El chatbot identifica que la solicitud es para generar una descripción.
 3. El chatbot envía la descripción al servicio de descripción de mashups.
 4. El servicio de generación de mashups devuelve una descripción del mashup.
 5. El chatbot recibe la descripción del mashup y la presenta al usuario.

Este flujo de trabajo (Figura 7.3) permite al chatbot generar mashups a partir de descripciones de flujo, así como generar descripciones de flujo a partir de mashups, brindando una experiencia interactiva y útil para los usuarios.

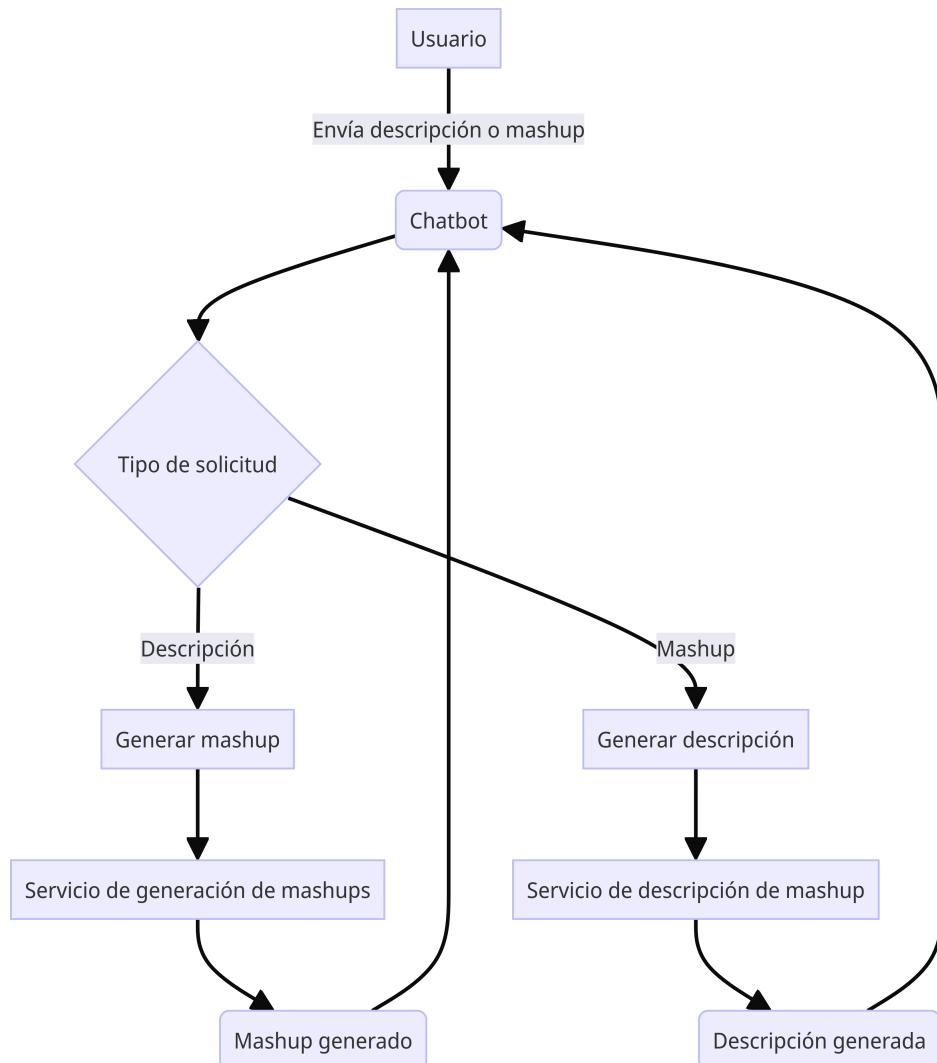


Figura 7.3: Flujo de funcionamiento principal del chatbot

7.2. Ventajas e Inconvenientes de la Arquitectura Escogida

La arquitectura seleccionada se analiza en base a sus aspectos clave, tomando como referencia el trabajo de Richards (2015).

7.2.1. Ventajas de la arquitectura en capas

- **Separación de preocupaciones:** Permite dividir la aplicación en capas con roles específicos, facilitando la gestión y mantenimiento.
- **Escalabilidad:** Puede escalar horizontalmente al separar las capas en diferentes servidores.
- **Reutilización de código:** Facilita la reutilización de componentes en diferentes partes de la aplicación.
- **Facilita el testing:** Al separar las capas, se facilita la realización de pruebas unitarias y de integración.

7.2.2. Inconvenientes de la arquitectura en capas

- **Sobrecarga de comunicación:** El paso de datos entre capas puede generar una sobrecarga en la comunicación.
- **Complejidad:** Puede resultar complejo gestionar la comunicación entre las diferentes capas.
- **Rendimiento:** El paso de datos entre capas puede afectar el rendimiento de la aplicación.
- **Cambios globales:** Un cambio en una capa puede afectar a otras capas, lo que puede resultar en cambios extensivos en la aplicación.

Este análisis destaca los aspectos positivos y negativos de la arquitectura en capas, proporcionando una visión integral para la toma de decisiones en el diseño y desarrollo del sistema.

8. Tecnologías y herramientas

Este capítulo proporciona una visión detallada de las decisiones tecnológicas clave que respaldan este trabajo. Comenzamos con un análisis tecnológico, donde examinamos las tecnologías que son necesarias para llevar a cabo el proyecto de investigación. Luego, exploramos diferentes alternativas, evaluando los aspectos positivos y negativos de cada una. Finalmente, presentamos nuestro stack tecnológico final, es decir, el conjunto de herramientas seleccionadas para implementar y desarrollar el TFG de manera efectiva.

8.1. Análisis Tecnológico

En este apartado se analizan las tecnologías y herramientas impuestas por el proyecto de investigación, ya sea por decisión del otro técnico (Francisco Montero) o por restricción de la propia prueba de concepto.

8.1.1. Node-Red

Node-RED¹ es una herramienta de código abierto para la creación de flujos de trabajo visuales basada en JavaScript (como se menciona en el apartado 1). Para realizar este análisis se utiliza de referencia el trabajo de Clerissi y cols. (2018).

■ Aspectos Positivos

- **Facilidad de uso:** Node-RED ofrece una interfaz gráfica intuitiva y fácil de usar, lo que permite a los usuarios crear flujos de trabajo complejos sin necesidad de escribir código.
- **Rápida prototipación:** Debido a su enfoque visual y la amplia gama de nodos predefinidos disponibles, Node-RED permite a los desarrolladores crear prototipos rápidamente y probar ideas sin tener que preocuparse por la complejidad del código.
- **Comunidad y soporte:** Node-RED cuenta con una comunidad activa de usuarios y desarrolladores que comparten recursos, nodos adicionales y soluciones a problemas comunes. Esto facilita la obtención de ayuda y el intercambio de conocimientos.
- **Flexibilidad y extensibilidad:** Node-RED es altamente flexible y puede integrarse con una variedad de servicios y plataformas, lo que lo hace ideal para proyectos que requieren interoperabilidad con sistemas externos.

■ Aspectos Negativos

- **Limitaciones en proyectos complejos:** A pesar de su facilidad de uso, Node-RED puede presentar limitaciones en proyectos extremadamente complejos o que requieran un alto rendimiento. En tales casos, puede ser necesario recurrir a soluciones más avanzadas.
- **Detección de Fallos y Desviaciones:** La libertad en el desarrollo de Node-RED puede dificultar la detección de fallos y desviaciones del comportamiento esperado del sistema, lo que puede resultar en problemas de mantenimiento y depuración.

¹<https://nodered.org>

En resumen, Node-RED es fácil de usar para la creación de flujos de trabajo visuales, especialmente adecuada para proyectos de prototipado rápido, integración de IoT y automatización. Como proyecto de código abierto, permite a los usuarios aprovechar una comunidad activa de desarrolladores y una amplia gama de recursos adicionales. Sin embargo, tras revisar la documentación de Node-RED, no se ha encontrado ningún ejemplo previo que muestre cómo recopilar información de otras fuentes de datos como las mencionadas (GitHub, Trello, ...). Por lo que encontrar una forma de implementar nodos útiles para la medición de cumplimiento puede requerir de investigación previa.

8.1.2. MySQL

Como sistema gestor de bases de datos (SGBD) para el proyecto se decidió usar MySQL. Esta herramienta es ampliamente utilizada debido a su simplicidad, notable rendimiento y facilidad de uso Santillán y cols. (2014). Además de su libre distribución bajo licencia GPL, MySQL está desarrollado en C/C++ optimizado para múltiples procesadores, lo que contribuye a su velocidad de respuesta destacable. Este SGBD se destaca por su disponibilidad en diversas plataformas y API en varios lenguajes, así como por su versatilidad como cliente-servidor o incrustado. Con un amplio conjunto de tipos de datos y métodos de almacenamiento, MySQL es una opción popular para aplicaciones comerciales y portales web de alto tráfico. Sin embargo, presenta limitaciones, como la carencia de algunas características avanzadas y menor compatibilidad con estándares, además de un enfoque en la eficiencia en lugar de incluir prestaciones competitivas.

8.1.3. Express JS

Como framework de *Back-End* para la prueba de concepto se decidió usar Express JS², un framework de aplicaciones web Node.js mínimo y flexible que proporciona un sólido conjunto de funciones para aplicaciones web y móviles. Para realizar este análisis se utiliza de referencia el trabajo de Hahn (2016).

■ Aspectos positivos

- **Sencillez y facilidad de uso:** Express.js tiene una curva de aprendizaje suave y una sintaxis minimalista que permite a los desarrolladores poner en marcha rápidamente sus aplicaciones web.
- **Flexibilidad:** A diferencia de otros frameworks más restrictivos, Express.js permite a los desarrolladores tomar decisiones arquitectónicas y elegir las herramientas que mejor se adapten a sus proyectos.
- **Gran comunidad y ecosistema:** Express.js cuenta con una comunidad activa y una amplia gama de complementos y middleware desarrollados por terceros, lo que facilita la integración de características adicionales en tu aplicación.
- **Rendimiento:** Express.js es conocido por su rendimiento rápido y eficiente, lo que lo convierte en una elección popular para aplicaciones web de alto rendimiento.

■ Aspectos negativos

- **Mayor responsabilidad del desarrollador:** Al ser un framework más ligero, Express.js no proporciona todas las características integradas que otros frameworks pueden ofrecer. Esto significa que los desarrolladores deben asumir más responsabilidades

²<https://expressjs.com>

al decidir y configurar las herramientas y módulos adicionales necesarios para su proyecto.

- **Menos estructura y convenciones:** Si prefieres un enfoque más estructurado y con convenciones predefinidas, Express.js puede resultar menos adecuado, ya que requiere una mayor toma de decisiones y una mayor planificación arquitectónica.

En resumen, Express.js es una excelente opción para proyectos que requieren flexibilidad y control, pero puede no ser la mejor opción para proyectos que necesitan una estructura más robusta y convenciones predefinidas.

8.1.4. React y Node.js

Como framework de *Front-End* se decidió usar React (impuesto por el proyecto), Aggarwal y cols. (2018) definen ReactJS como una biblioteca basada en componentes que se utiliza para el desarrollo de interfaces de usuario interactivas.

■ Aspectos positivos

- **Eficiencia en el rendimiento gracias al uso del virtual DOM:** React utiliza una representación virtual del DOM (*Document Object Model*, apartado 13.1) en lugar de manipular directamente el DOM del navegador, lo que resulta en un rendimiento más eficiente al minimizar las operaciones de actualización en la página.
- **Flujo unidireccional de datos:** React sigue el principio de unidireccionalidad de datos, lo que significa que los datos fluyen en una sola dirección a través de los componentes, facilitando el seguimiento de los cambios y la depuración del código.
- **Facilidad de uso de JSX para escribir componentes y eventos de enlace:** JSX es una extensión de sintaxis de JavaScript que permite escribir código HTML dentro de JavaScript, lo que facilita la creación de componentes y la gestión de eventos.
- **Capacidad para generar fuentes de datos canónicas sincronizadas entre componentes:** React facilita la sincronización de los datos entre diferentes componentes, lo que ayuda a mantener un estado coherente en toda la aplicación.
- **Respaldado por Facebook, Instagram y una comunidad de desarrolladores:** El respaldo de empresas como Facebook e Instagram, junto con una gran comunidad de desarrolladores, proporciona soporte continuo, actualizaciones y recursos para el desarrollo con React.

■ Aspectos negativos

- **Necesidad de herramientas adicionales para completar el desarrollo del proyecto:** Aunque React es potente por sí solo, a veces se requieren herramientas adicionales como Redux para manejar estados complejos o Webpack para la gestión de módulos, lo que puede agregar complejidad al proyecto.
- **Complejidad percibida por algunos desarrolladores al usar plantillas en línea y JSX:** Para algunos desarrolladores, la combinación de lógica JavaScript y marcado HTML dentro de JSX puede resultar confusa al principio, especialmente si están acostumbrados a trabajar con plantillas separadas.
- **Posibilidad de que los fallos ocurran en tiempo de compilación en lugar de en tiempo de ejecución, lo que puede resultar frustrante:** Los errores en React a menudo se detectan durante el tiempo de compilación debido a su naturaleza estática, lo que significa que los errores pueden ser menos intuitivos de detectar en comparación

con los sistemas más dinámicos donde los errores se producen durante la ejecución del programa.

8.1.5. Github y Git

Para la gestión del código se ha recomendado el uso de Github por parte de los vocales responsables del proyecto de investigación, esta herramienta es ampliamente utilizada en la titulación junto con Git. En el artículo “*A quick introduction to version control with Git and GitHub*” (Blischak y cols., 2016) se describe GitHub como una plataforma en línea que permite a los desarrolladores colaborar en proyectos de software utilizando el sistema de control de versiones Git. Git es un sistema de control de versiones distribuido que ayuda a rastrear cambios en archivos de código. GitHub proporciona un entorno centralizado para almacenar, gestionar y compartir repositorios Git, lo que facilita la colaboración, revisión de código, seguimiento de problemas y despliegue de aplicaciones. En resumen, Git se utiliza para controlar versiones localmente, mientras que GitHub se utiliza para colaborar de forma remota en proyectos de desarrollo de software.

8.2. Análisis de las Alternativas

En esta sección, se explorarán diversas alternativas tecnológicas con el objetivo de evaluar su viabilidad y efectividad en relación con requisitos del sistema como el servicio de NLP y la documentación online. Mediante un análisis detallado, se examinarán las distintas alternativas disponibles para encontrar la solución que más satisfaga los requisitos, incluyendo sus ventajas y desventajas, así como una descripción detallada de su uso principal.

8.2.1. Servicios de NLP

A continuación se analizan los servicios mencionados en el apartado 6.2 con el fin de implementar un chatbot que realice consultas sobre mashups en Node-RED.

IBM Watsonx Assistant

IBM Watsonx Assistant es un servicio de inteligencia artificial (IA) y procesamiento de lenguaje natural (NLP) que proporciona a los clientes respuestas rápidas, coherentes y precisas en cualquier plataforma de mensajería, aplicación, dispositivo o canal (Brutti (2024)).

■ Aspectos positivos

- Amplia variedad de herramientas y servicios para el análisis de lenguaje natural (NLP) y el aprendizaje automático.
- Ofrece una gran cantidad de recursos y documentación para ayudar a los desarrolladores a implementar sus proyectos.
- Permite la integración con una amplia gama de plataformas y tecnologías.
- Aprende de las conversaciones de los clientes para mejorar su capacidad de resolver problemas.

■ Aspectos negativos

- Puede ser complejo para usuarios sin experiencia técnica.

- Puede ser costoso, especialmente para proyectos pequeños o de bajo presupuesto.

Microsoft LUIS

Microsoft LUIS se utiliza principalmente para el reconocimiento de intenciones y la comprensión del lenguaje natural. Es ideal para proyectos que requieren una comprensión básica del lenguaje natural y una integración con Microsoft Azure (Nitin Mehrotra (2024)).

■ **Aspectos positivos**

- Es relativamente asequible, especialmente para proyectos pequeños o de bajo presupuesto.

■ **Aspectos negativos**

- No es tan amplio como el servicio de NLP de IBM Watson, lo que puede limitar su capacidad para manejar proyectos complejos.
- Poca documentación orientada al uso de API Rest.
- Está especialmente diseñado para resolver problemas de comprensión de lenguaje natural, por lo que puede que no sea capaz de realizar flujos de node-RED de forma correcta.
- Este servicio dejará de estar disponible a partir del 1 de Octubre de 2025. Además, para provocar la migración de los usuarios a *Azure OpenAI Service* (servicio de microsoft con modelos de OpenAI), a partir del 1 de Abril de 2023 cancelaron la creación de nuevos recursos.

Amazon Lex

Amazon Lex se utiliza principalmente para el reconocimiento de voz y el reconocimiento de intenciones. Es ideal para proyectos que requieren una comprensión básica del lenguaje natural y una integración con *Amazon Web Services* (AWS). A continuación se mencionan los aspectos positivos y negativos recogidos en el blog de Asper (2022) y en base a la revisión de la documentación de esta herramienta.

■ **Aspectos positivos**

- Permite la integración con AWS y otras tecnologías de Amazon.
- Es relativamente asequible, especialmente para proyectos pequeños o de bajo presupuesto.

■ **Aspectos negativos**

- No es tan amplio como el servicio de NLP de IBM Watson, lo que puede limitar su capacidad para manejar proyectos complejos.
- Documentación de API Rest amplia pero poco intuitiva.

Google Dialogflow

Google Dialogflow se utiliza principalmente para el reconocimiento de intenciones y la comprensión del lenguaje natural (Denis (2019)). Es ideal para proyectos que requieren una comprensión básica del lenguaje natural y una integración con *Google Cloud Platform* (GCP).

■ **Aspectos positivos**

- Permite la integración con GCP y otras tecnologías de Google.
- Es relativamente asequible, especialmente para proyectos pequeños o de bajo presupuesto.

■ **Aspectos negativos**

- No es tan amplio como el servicio de NLP de IBM Watson, lo que puede limitar su capacidad para manejar proyectos complejos.
- Documentación de API Rest amplia pero poco intuitiva.

OpenAI Assistant

OpenAI Assistant se utiliza principalmente para el reconocimiento de intenciones y la comprensión del lenguaje natural. Es ideal para proyectos que requieren una comprensión básica del lenguaje natural y una integración con una amplia gama de plataformas y tecnologías (Díaz (2024)).

■ **Aspectos positivos**

- Ofrece una documentación clara para ayudar a los desarrolladores a implementar sus proyectos.
- Permite la integración con una amplia gama de plataformas y tecnologías.
- Es relativamente asequible³, especialmente para proyectos pequeños o de bajo presupuesto.

■ **Aspectos negativos**

- No es tan amplio como el servicio de NLP de IBM Watson, lo que puede limitar su capacidad para manejar proyectos complejos.

Comparativa de las alternativas

Aspecto	IBM Watson Assistant	Microsoft LUIS	Amazon Lex	Google Dialogflow	OpenAI Assistant
Flexibilidad	Alta	Media	Media	Alta	Alta
Precio	Medio	Alto	Bajo	Bajo	Bajo
Complejidad	Baja	Alta	Media	Media	Media
Calidad de la Documentación	Alta	Media	Media	Media	Alta

Cuadro 8.1: Comparación de Servicios de NLP

En general, basándose en esta comparación (Cuadro 8.1) y considerando la flexibilidad, el precio más bajo, la complejidad y la calidad de la documentación, OpenAI Assistant parece ser una elección sólida.

³<https://devrain.com/blog/calculating-openai-and-azure-openai-service-model-usage-costs>

8.2.2. Generadores de páginas estáticas

A continuación se analizan herramientas de creación de páginas estáticas con el fin de realizar una documentación para el proyecto de investigación y para el TFG. Para este análisis se han escogido algunas de las herramientas más utilizadas⁴ por los usuarios de Github y se ha revisado la documentación de cada herramienta (además de tener conocimiento previo de ellas).

Hugo

Hugo⁵⁶ es un generador de sitios estáticos rápido y flexible escrito en Go, optimizado para la velocidad y diseñado para la flexibilidad. Es ampliamente utilizado para crear varios tipos de sitios web, incluyendo sitios de documentación.

■ Aspectos Positivos

- **Velocidad:** Hugo es conocido por su capacidad de renderizado, lo que lo hace adecuado para proyectos de documentación a gran escala.
- **Flexibilidad:** El sistema de plantillas y las canalizaciones de activos de Hugo permiten la personalización y la integración con varias herramientas y marcos de trabajo.

■ Aspectos Negativos

- **Difícil curva de aprendizaje:** Las funciones avanzadas y el sistema de plantillas de Hugo pueden resultar abrumadores para los principiantes.
- **Configuración compleja:** Hugo requiere una buena comprensión de sus archivos de configuración y sintaxis para configurarlo correctamente.

Jekyll

Jekyll⁷⁸ es un popular generador de sitios estáticos escrito en Ruby, conocido por su simplicidad y facilidad de uso. Jekyll es adecuado para proyectos de documentación más pequeños o con requisitos más sencillos, en los que la facilidad de uso y una gran comunidad son más importantes que el alto rendimiento.

■ Aspectos Positivos

- **Fácil de usar:** Jekyll tiene una configuración y sintaxis relativamente sencillas, lo que lo hace accesible para principiantes.
- **Gran comunidad:** Jekyll tiene una comunidad grande y activa, lo que garantiza que haya muchos recursos disponibles para la solución de problemas y la personalización.
- **Personalizable:** Jekyll soporta varios temas y plugins, permitiendo la personalización para satisfacer necesidades específicas.

■ Aspectos Negativos

- **Rendimiento limitado:** Jekyll no es tan rápido como Hugo, lo que puede ser una preocupación para proyectos de documentación a gran escala.

⁴<https://jamstack.org/generators/>

⁵<https://gohugo.io>

⁶<https://kinsta.com/blog/hugo-static-site/>

⁷<https://jekyllrb.com/docs/>

⁸<https://kinsta.com/blog/jekyll-static-site/>

Docusaurus

Docusaurus⁹¹⁰ es un generador de sitios estáticos diseñado específicamente para crear sitios de documentación, construido sobre React y Webpack. Docusaurus es ideal para proyectos que requieren una fácil configuración y personalización para sitios de documentación, donde la velocidad y el rendimiento son importantes.

■ Aspectos Positivos

- **Fácil de usar:** Docusaurus tiene un proceso de configuración sencillo y una interfaz fácil de usar, lo que lo hace accesible a desarrolladores sin amplios conocimientos de generadores de sitios estáticos.
- **Personalizable:** Docusaurus es compatible con varios temas y plugins, lo que permite personalizarlo para adaptarlo a necesidades específicas.
- **Rápido:** Docusaurus está diseñado para la velocidad y puede renderizar sitios de documentación rápidamente.
- **Experiencia previa:** Durante la asignatura “Ingeniería Informática y Práctica Profesional”, pude familiarizarme con esta herramienta. Esta experiencia previa me proporciona una base sólida para utilizar eficientemente la herramienta en el contexto de este TFG, reduciendo así el tiempo de aprendizaje y permitiéndome enfocarme directamente en la implementación y análisis de los resultados.

■ Aspectos Negativos

- **Opciones de personalización:** Son un tanto restringidas en comparación con otras herramientas más flexibles como Hugo o Jekyll. Esto puede ser un problema si necesitas realizar modificaciones muy específicas o avanzadas en el diseño o funcionalidad de tu sitio web.

Gitbook

Gitbook¹¹ es una plataforma para crear y gestionar documentación, centrada en la facilidad de uso y la colaboración. Gitbook es adecuado para proyectos que requieren una colaboración y un control de versiones sencillos, en los que la personalización no es una preocupación primordial.

■ Aspectos Positivos

- **Fácil de usar:** Gitbook tiene una interfaz sencilla e intuitiva, lo que lo hace accesible a desarrolladores sin amplios conocimientos de generadores de sitios estáticos.
- **Colaboración:** Gitbook permite una perfecta colaboración y control de versiones, lo que lo hace adecuado para proyectos con múltiples colaboradores.
- **Integración:** Gitbook se integra bien con varias herramientas y plataformas, como GitHub y GitLab.

■ Aspectos Negativos

- **Personalización limitada:** Las opciones de personalización de Gitbook son limitadas en comparación con otros generadores de sitios estáticos.

⁹<https://docusaurus.io>

¹⁰<https://tw-docs.com/docs/static-site-generators/docs-as-code/>

¹¹<https://docs.gitbook.com>

- **Dependencia de la infraestructura de Gitbook:** La funcionalidad de Gitbook depende de su propia infraestructura, que puede no ser adecuada para proyectos con requisitos específicos.

Mkdocs

Mkdocs¹² es un generador de sitios estáticos ligero escrito en Python, diseñado para crear sitios de documentación. Mkdocs es ideal para proyectos que requieren una fácil configuración y personalización para sitios de documentación, donde la velocidad y el rendimiento son importantes.

■ Aspectos Positivos

- **Fácil de usar:** Mkdocs tiene un proceso de configuración sencillo y una interfaz fácil de usar, lo que lo hace accesible a desarrolladores sin amplios conocimientos de generadores de sitios estáticos.
- **Rápido:** Mkdocs está diseñado para la velocidad y puede renderizar sitios de documentación rápidamente.
- **Personalizable:** Mkdocs soporta varios temas y plugins, permitiendo la personalización para satisfacer necesidades específicas.

■ Aspectos Negativos

- **Interfaz de usuario:** El estilo de la interfaz es muy básica en comparación al resto de alternativas.

Comparativa de las alternativas

Aspectos	Hugo	Jekyll	Docusaurus	GitBook	MkDocs
Escala	Proyectos grandes	Proyectos pequeños	Proyectos pequeños	Proyectos grandes	Proyectos pequeños
Personalización	Alta	Alta	Media	Alta	Media
Complejidad	Baja	Media	Baja	Media	Baja
Experiencia previa	No	No	Sí	No	No

Cuadro 8.2: Comparativa de Generadores de Páginas Estáticas

Basándonos en el análisis de la tabla de comparativa 8.2, se observa que no hay restricciones de escalabilidad dado que el proyecto es de tamaño pequeño. La complejidad es un factor clave, lo que destaca a Hugo, Docusaurus y MkDocs por su facilidad de uso. Sin embargo, el aspecto determinante es la experiencia previa, la cual solo está presente con Docusaurus. Por lo tanto, Docusaurus emerge como la herramienta más adecuada para satisfacer las necesidades del proyecto.

8.3. Stack Tecnológico

En este apartado, se detallarán las herramientas y tecnologías seleccionadas para la realización del proyecto, tomando como referencia el análisis efectuado en la sección de “Análisis de las

¹²<https://www.mkdocs.org/getting-started/>

Alternativas” (8.2), así como las herramientas establecidas por el proyecto de investigación, las cuales fueron evaluadas en el apartado correspondiente (8.1).

Para ofrecer una visión general del entorno tecnológico del proyecto, se incluye un diagrama de contexto que ilustra las interacciones principales del sistema (Figura 8.1). Este diagrama proporciona un marco visual que ayuda a comprender las herramientas que se van a utilizar en el proyecto.

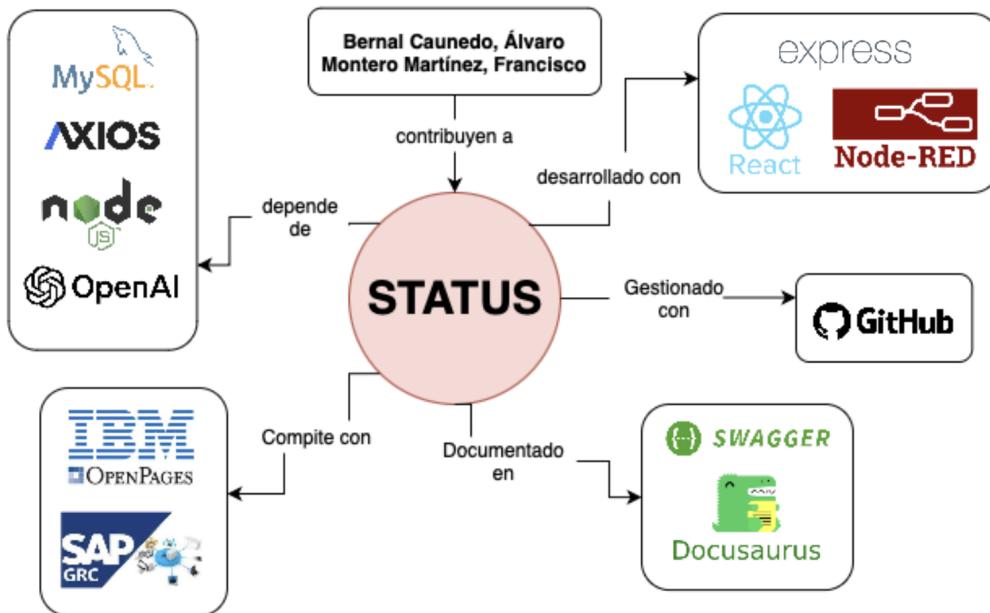


Figura 8.1: Diagrama de contexto

Para la implementación del chatbot, se utilizará OpenAI permitiendo la creación de conversaciones inteligentes y contextuales. MySQL será empleado como sistema de gestión de bases de datos, asegurando la persistencia y la eficiente manipulación de la información. Axios, una biblioteca de JavaScript, será utilizada para realizar peticiones HTTP desde el *frontend* al *backend*, facilitando la comunicación entre los distintos componentes del sistema. Node.js, como entorno de ejecución de JavaScript del lado del servidor, proporcionará la base para el desarrollo del *backend* del proyecto, garantizando la escalabilidad y el rendimiento requeridos.

Para la creación del *backend*, se empleará Express, un framework de Node.js que simplifica la creación de aplicaciones web y APIs, permitiendo una estructura organizada y modular. Node-Red será utilizado para la creación de mashups, facilitando la integración de diferentes servicios y dispositivos mediante un enfoque visual de tipo “arrastrar y soltar”. Por otro lado, React, una biblioteca de JavaScript, será utilizada en el *frontend* para construir interfaces de usuario interactivas y eficientes, mediante la gestión eficaz del estado de la aplicación y la renderización optimizada de los componentes.

Además del uso de la herramienta Swagger para documentar la API del proyecto, se empleará Docusaurus para generar una documentación completa y amigable del proyecto de investigación. Swagger permitirá describir y visualizar de forma precisa los endpoints y parámetros de la API, facilitando su comprensión y uso. Por otro lado, Docusaurus proporcionará un marco para crear y mantener una documentación clara y estructurada, permitiendo una fácil navegación y consulta por parte de los usuarios y desarrolladores.

En el ámbito de la gestión de la conformidad, se consideran dos competidores principales: IBM OpenPages y SAP GRC. IBM OpenPages ofrece una plataforma integral para gestionar ries-

gos y cumplimiento, proporcionando herramientas para la evaluación de riesgos, el cumplimiento normativo y la auditoría interna. Por su parte, SAP GRC ofrece soluciones para la gestión de riesgos, el control interno y el cumplimiento normativo, integradas con los sistemas empresariales de SAP, ofreciendo un enfoque holístico y centralizado para la gestión de la conformidad. El uso de estas tecnologías puede aplicarse en casos de gestión de la conformidad en un aspecto normativo. En cambio, esta prueba de concepto se centra en algo más genérico, orientado a un espectro más amplio de la gestión de la conformidad, como el cumplimiento del RGPD, algunas norma ISO o incluso prácticas recomendadas.

Por último, para la gestión de versiones y el trabajo colaborativo en el proyecto de investigación, se utilizará GitHub. GitHub permitirá a los miembros del equipo colaborar de forma eficiente en el desarrollo del proyecto, gestionando el código fuente, revisando y fusionando cambios, y manteniendo un registro histórico de las modificaciones realizadas.

9. Desarrollo del producto

En este capítulo se presenta el desarrollo del Trabajo de Fin de Grado, comentando las actividades llevadas a cabo a lo largo de la etapa de ejecución. Cada sprint representó una fase del proyecto, abordando tareas específicas y progresando hacia los objetivos establecidos al inicio de la memoria. Se mencionan de forma superficial las actividades realizadas en cada sprint, incluyendo las tareas de cierre que permitieron evaluar el avance realizado, identificar posibles mejoras y re-planificar las siguientes etapas del proyecto en caso de necesidad.

9.1. Sprint 1

El primer sprint, iniciado el 13 de Octubre, se centró en la formación y configuración de las herramientas que serán utilizadas e integradas en el sistema. Además de esto, se llevó a cabo un análisis de los componentes que podrían ser agregados a Node-RED para realizar mediciones de conformidad, seguido por su implementación.

9.1.1. Formación

Esta primera fase del proyecto fue una toma de contacto con las herramientas que se utilizan en el proyecto de investigación. Por ello, las primeras semanas están orientadas a la formación en estas tecnologías. Debido a que el técnico Franciso Montero se incorporó antes al proyecto de investigación, este no necesitó formación sobre Node-RED, ya que lo había hecho previamente (como se muestra en el paquete 5.1).

- **Formación de Node-RED:** Para esta formación, se realizó un curso básico de node-red¹ de 2,5 horas de duración.
- **Formación de Governify y Bluejay:** Esta formación fue algo más particular debido a que estas herramientas tienen menos popularidad (y por consiguiente menos documentación al respecto). Por ello, uno de los profesores e investigadores que propusieron esta herramienta, Pablo Fernández Montes, nos proporcionó a los técnicos unos vídeos de formación y configuración de Bluejay (Además de los enlaces a la documentación²³ de ambas herramientas).
- **Formación sobre el Collector-Events de Bluejay:** Este es uno de los servicios principales de bluejay, para la formación sobre este servicio se obtuvieron vídeos de formación al igual que en el punto anterior.

9.1.2. Configuración de entornos

Una vez realizada la formación, llegó la etapa de configuración y prueba de las tecnologías. El primer problema encontrado fue que el dispositivo utilizado era y es un Mac con chip M1, esto planteó desafíos significativos respecto a la configuración y el desarrollo, ya que algunos entornos de desarrollo y herramientas de configuración no son totalmente compatibles con esta arquitectura. Esto se debe a que muchos de los sistemas y herramientas de desarrollo están optimizados para

¹<https://youtube.com/playlist?list=PLcmn4IteQPYqcZuzR7VLkQsFeAN0fPsPN&si=6GTDMt-n71K38qkV>

²<https://docs.governify.io>

³<https://docs.bluejay.governify.io>

arquitecturas x86-64 tradicionales, lo que podría llevar a problemas de rendimiento, incompatibilidad de software o necesidad de adaptaciones para funcionar correctamente en los nuevos chips M1 de Apple (arquitectura ARM64).

La configuración y realización de pruebas con Node-RED fue algo sencillo, debido a la compatibilidad de esta herramienta con la arquitectura comentada anteriormente. Para la configuración de la infraestructura de Bluejay se siguieron los siguientes pasos:

1. Realizar la configuración siguiendo los vídeos, aunque estos tenían una duración de 3,5 horas con una gran cantidad de pasos a seguir.
2. Dado que después del primer intento no se realizó con éxito, se repitió el paso 1 para confirmar que se habían seguido los pasos correctamente.
3. Se solicitó ayuda a los técnicos encargados de Bluejay (aunque sin éxito porque no utilizaban esta arquitectura).
4. Se localizó el problema, no se podía desplegar los servicios en contenedores docker con la arquitectura arm64 porque las imágenes usaban la arquitectura x86_64.
5. Se configuraron los servicios para especificar la arquitectura ARM64 en las imágenes, siendo un éxito ya que se consiguió desplegar en local los servicios.
6. Uno de los servicios no funcionaba correctamente y tras solicitar ayuda al tutor, se consiguió resolver el problema (esta vez configurando un servicio de mongoDB aparte con la imagen con la arquitectura ARM64 por defecto).

Esta tarea fue el primer periodo de estancamiento del proyecto, en el que se utilizaron 25 horas a resolver el problema. Por otro lado, el *Collector-Events* al ser desplegado en local sin necesidad de utilizar un contenedor Docker, no supuso problemas de configuración.

9.1.3. Traducción de nodos

Después de instalar y configurar el entorno, hacía falta recopilar información sobre el proyecto “Procede”, para reutilizar componentes que iban a ser diseñados en Node-RED. Para ello, primero se tuvo que hacer un estudio de viabilidad de licencias de *Entreprise Architect* (EA) porque era la herramienta en la que tenían almacenados los *mashups*. Después de realizar el estudio, se concluyó en que era mejor alternativa migrar la información a otra herramienta gratuita, mediante la versión de prueba de EA. Posteriormente, se almacenó la información relevante junto con las imágenes de los mashups en un documento y se realizó un análisis de los componentes para plantear un diseño inicial de los nodos.

9.1.4. Creación inicial de nodos

Este paquete de trabajo consistió en la implementación inicial de “nodos”(llamados componentes en EA) a partir de la traducción del paquete de trabajo anterior. Estos nodos personalizados consistieron en código javascript con las funciones del nodo junto con código html para la apariencia del nodo en el Editor, resultando así los nodos que aparecen en la Figura 9.1.

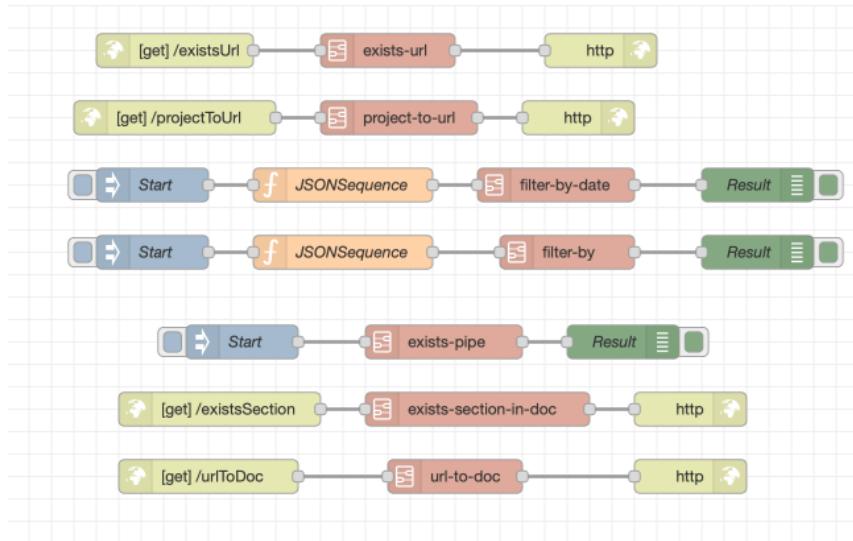


Figura 9.1: Nodos de Procede

9.1.5. Cierre del Sprint

Finalmente, se cerró el sprint el día 5 de Diciembre con la fase de seguimiento correspondiente, lo que permitió evaluar el progreso y generar los documentos de seguimiento necesarios para ajustar y mejorar el desarrollo del proyecto. Basándonos en la retrospectiva realizada, se han identificado puntos cruciales que merecen ser destacados:

- **Complejidad en la configuración del entorno:** La configuración de entornos se presentó como un desafío significativo. Este proceso puede ser especialmente abrumador para aquellos con poca experiencia y cuando la documentación disponible no es suficientemente detallada o clara.
- **Claridad en los objetivos de las tareas:** Es imprescindible tener una comprensión clara de los objetivos de cada tarea. Esto no solo facilita la ejecución eficiente de las actividades, sino que también garantiza que dichas tareas estén alineadas con las necesidades y metas del proyecto en su conjunto.

9.2. Sprint 2

El día 8 de Diciembre dió comienzo el segundo sprint, en el que se empaquetaron los nodos realizados en el sprint anterior para poder ser instalables en Node-RED, además de realizar actividades relacionadas con la integración continua, comenzar con el desarrollo del módulo de autenticación de usuarios y crear un esqueleto inicial de la memoria.

9.2.1. Empaquetado de nodos

Esta segunda fase comienza con la investigación sobre cómo publicar nodos en Node-RED, que resulta ser algo sencillo ya que para publicar este paquete, solo es necesario subirlo al sistema de gestión de paquetes npm y rellenar un formulario en node-RED⁴ indicando el nombre del

⁴<https://flows.nodered.org/add/node>

paquete. Además, en la documentación⁵ de Node-Red se especifica cómo configurar el directorio para poder ser publicado correctamente.

Finalmente, después de resolver algunos problemas encontrados durante la realización de este paquete de trabajo, se consiguió empaquetar los nodos para que fueran importables.

9.2.2. Integración continua

La realización de actividades de este paquete de trabajo fue algo más sencilla debido a las guías de despliegue en Docker de Node-RED⁶ y al catálogo de *workflows* que ofrece GitHub. Para mejorar la integración continua, se realizó un *workflow* llamado “*auto-publisher.yml*” encargado de actualizar las versiones de los paquetes en npm⁷ y en DockerHub⁸.

Además de esto, se creó un repositorio llamado “node-red-status-starter”⁹ con el objetivo de proporcionar un repositorio para el despliegue local de node-RED (con nuestros nodos importados y algún ejemplo de uso). Para llevar a cabo dicha configuración local se realizaron dos *scripts*, uno para los usuarios de Windows y otro para los de Mac, encargados de configurar el contenedor Docker de forma sencilla (junto con una guía de despliegue).

Por otra parte, también se creó un repositorio¹⁰ orientado a la instalación y despliegue local de la infraestructura de STATUS. En este repositorio se alojan algunos archivos de configuración de node-red, además de *scripts* que facilitan la instalación de los usuarios (para sistemas operativos MacOS, Linux y Windows), un docker-compose encargado de desplegar toda la infraestructura y una pequeña guía de instalación en el README del repositorio.

9.2.3. Perfil de usuario

Durante esta fase de implementación, no ha habido ninguna complicación destacable. Se ha empleado la librería JSON Web Token (JWT) para la autorización y autenticación de usuarios, con una expiración del token de 1 hora y del token de refresco de 1 día.

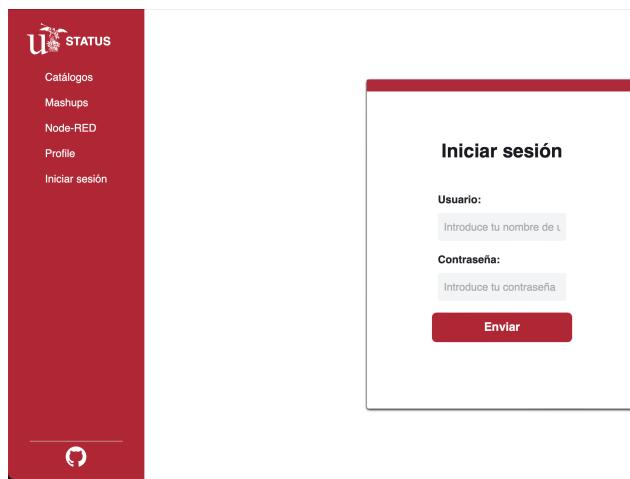


Figura 9.2: Vista de inicio de sesión

⁵<https://nodered.org/docs/creating-nodes/packaging>

⁶<https://nodered.org/docs/getting-started/docker>

⁷<https://www.npmjs.com/package/@alvarobc2412/status>

⁸<https://hub.docker.com/repository/docker/alvber/node-red-status/general>

⁹<https://github.com/statuscompliance/node-red-status-starter>

¹⁰<https://github.com/statuscompliance/infrastructure>

Se han establecido endpoints para la gestión de usuarios, incluyendo el inicio de sesión (Figura 9.2), autenticación de usuarios y renovación de sesión. Además, se ha agregado una verificación de autorización de JWT al middleware del *backend* para garantizar la seguridad en el acceso a estos *endpoints*. Por otro lado, también se realizaron las actividades correspondientes a la integración de servicios externos como Github Y Trello.

9.2.4. Estructura principal de la memoria

En esta fase del proyecto, se ha avanzado satisfactoriamente en la elaboración de la memoria. La estructura principal se ha desarrollado sin contratiempos, abarcando las secciones de Introducción, Alcance del proyecto, Especificación de requisitos y la Realización de los mockups.

Para la redacción de la memoria, se ha optado por utilizar la plataforma Overleaf, una herramienta que ya había sido utilizada para proyectos de otras asignaturas de la titulación. Esta elección ha facilitado la colaboración y revisión tanto por mi parte como por parte de mis tutores.

En cuanto a la creación de los mockups, se ha empleado la herramienta visily.ai¹¹. Esta plataforma se destaca por su capacidad para generar la estructura principal de un mockup a partir de una imagen de un boceto a mano alzada, gracias al uso de la inteligencia artificial. Esta metodología ha agilizado significativamente el proceso de diseño de la interfaz de usuario, permitiendo una visualización más precisa de cómo se verá la aplicación final.

9.2.5. Cierre del Sprint

Concluyó el sprint el día 30 de Enero con la fase de seguimiento correspondiente (al igual que en apartado 9.1), lo que posibilitó la evaluación del progreso y la generación de los documentos necesarios en esta segunda iteración. En esta fase, la retrospectiva concluyó con una lección aprendida muy importante:

- Antes de abordar una tarea sin experiencia previa o conocimiento, es crucial considerar si existen múltiples enfoques para llevar a cabo dicha tarea. En tales casos, es recomendable optar por el enfoque que esté respaldado por una mayor cantidad de documentación disponible. Esta elección puede facilitar significativamente la ejecución de la tarea, ya que proporciona un recurso invaluable para orientarse y resolver posibles desafíos que puedan surgir en el proceso.

9.3. Sprint 3

El tercer sprint se centra en el núcleo principal del trabajo, comenzando el día 30 de Enero con la implementación inicial del chatbot. Además de esto, a lo largo del sprint se integró Node-RED en el sistema, se resolvieron incidencias de código y se realizaron pruebas para mejorar la calidad del producto final.

9.3.1. Versión inicial del chatbot

Antes de comenzar con la implementación del núcleo de este trabajo, se realizó una formación sobre el asistente de OpenAI. Después de esto se hizo una implementación inicial del chatbot en la que se podían describir los mashups de Node-RED introduciendo en el chat un flujo en formato

¹¹<https://www.visily.ai>

JSON (formato que utiliza Node-RED para almacenar los flujos). Para poder obtener una versión del asistente que realizase descripciones acertadas de los mashups, se utilizó un modo de prueba de OpenAI llamado *Playground*. En la Figura 9.3 se muestra una de las pruebas realizadas para poder obtener una configuración útil del asistente para describir mashups (junto con la *prompt* dada).

```
[{"id": "93e6cf2049ba8ebd", "type": "tab", "label": "Flow 1", "disabled": false, "info": "", "env": []}, {"id": "2f0675c0444d821d", "type": "http in", "z": "93e6cf2049ba8ebd", "name": "", "url": "/mashup1", "method": "get", "upload": false, "swaggerDoc": "", "x": 420, "y": 440, "wires": [[{"id": "f5535015fa211c69"}]}, {"id": "40f3500e9e1a4ada", "type": "exists-section-in-doc", "z": "93e6cf2049ba8ebd", "name": "", "section": "", "x": 1060, "y": 440, "wires": [[{"id": "b2b1a192fcf78454"}]]}, {"id": "f5535015fa211c69", "type": "trello-collector", "z": "93e6cf2049ba8ebd", "boardId": "", "key": "", "token": "", "x": 640, "y": 440, "wires": [[{"id": "93e5466fd77063db"}]]}, {"id": "93e5466fd77063db", "type": "filter-by", "z": "93e6cf2049ba8ebd", "name": "", "valueType": "Clasification", "attribute": "", "value": "", "x": 840, "y": 440, "wires": [[{"id": "40f3500e9e1a4ada"}]]}, {"id": "b2b1a192fcf78454", "type": "http response", "z": "93e6cf2049ba8ebd", "name": "", "statusCode": "", "headers": {}, "x": 1290, "y": 440, "wires": []}]
```

asst_5XbEN8hVtM1DwehG1yZHTLYC

This flow involves an HTTP GET request at the "/mashup1" endpoint. The request triggers a series of actions:

- 1 A trello collector node collects the GitHub repositories linked to each existing card.
- 2 The output from the trello collector then goes through a filter to select objects based on a specified attribute (here, "Clasification").
- 3 Finally, the flow checks if a specific section exists in a document by using the "exists-section-in-doc" node.

Figura 9.3: Respuesta de prueba del *Playground*

9.3.2. Gestión de mashups

Las actividades de este paquete de trabajo se llevaron a cabo utilizando la documentación de Node-RED¹², lo que permitió agilizar significativamente la implementación de métodos de gestión de los mashups y la autorización de acceso a Node-RED.

- **Configuración de la API:** Se modificó el archivo de configuración principal de node-RED para configurar la API del administrador y para la autorización de usuarios.
- **Securización del Editor:** Se restringió el acceso al editor mediante usuario y contraseña. Estas credenciales se encriptan mediante la librería *bcrypt*, de forma que se almacena en la configuración de node-RED el *hash* de la contraseña y el nombre del usuario que tiene acceso. Además, los usuarios tendrán dos opciones para acceder al editor mediante credenciales: ya sea que node-RED esté previamente desplegado o que se despliegue después. Dependiendo de esta circunstancia, la sesión se iniciará desde un modal en el sistema o directamente en el editor.
- **Llamadas a la API:** Se crearon las llamadas correspondientes para consultar, crear, editar y eliminar mashups.

9.3.3. Mejora de la calidad del código

Este paquete de trabajo se centra en la refactorización del código de mala calidad analizado por la herramienta de análisis de código Codacy¹³. Sin embargo, durante el análisis de la arquitectura

¹²<https://nodered.org/docs/api/admin/>

¹³<https://www.codacy.com>

inicial del sistema se observó un punto crítico de vulnerabilidad del sistema, el uso de peticiones nativas en SQL a la base de datos (mencionado en el apartado 7.1). Por ello, se decidió añadir a este paquete de trabajo una actividad previa de refactorización del *backend* para incluir el ORM Sequelize¹⁴.

Después de esto, se realizó el análisis de calidad del código y se refactorizaron los problemas de código encontrados. Siendo la mayoría de errores encontrados:

- Variables definidas que no se utilizan a lo largo del código (Error grave).
- Parámetros sin validación (Error de seguridad).
- Espacios en blanco después de cada bloque de estilo en CSS (Error menor).

9.3.4. Pruebas

Para garantizar la fiabilidad y calidad del producto final, en este paquete de trabajo se realizaron las pruebas más apropiadas para el sistema (esto se decidió en la etapa de planificación inicial). Entre todas las posibilidades, se descartó la opción de realizar pruebas de interfaz de usuario debido a su corta vida útil. Las actividades planificadas de creación de pruebas de API, Node-RED y *prompts* no sufrieron atrasos ya que se pudieron realizar las pruebas planificadas correctamente mediante la herramienta Postman¹⁵.

9.3.5. Cierre del Sprint

Después de realizar el seguimiento correspondiente al sprint (concluyendo el día 19 de Marzo), se obtuvo la siguiente conclusión:

- Es correcto simplificar la implementación inicial del sistema para facilitar iteraciones y mejorar el producto; sin embargo, las decisiones sobre la infraestructura del sistema deben tomarse con plena conciencia para evitar futuros cambios en la tecnología. Cambiar de tecnología es arriesgado y puede introducir una cantidad significativa de deuda técnica, lo que puede resultar en mayores costos de mantenimiento y refactorización en el futuro, así como en posibles interrupciones del servicio y pérdida de tiempo productivo durante la implementación y adaptación a la nueva tecnología.

9.4. Sprint 4

Esta fue la última fase de desarrollo del trabajo, que comenzó el día 19 de Marzo con la mejora de la interfaz de usuario y continuó con la implementación final del chatbot, el desarrollo de una vista de administración del chatbot y el cierre de la versión completa de la memoria.

9.4.1. Interfaz de usuario

La actualización de la interfaz de usuario se llevó a cabo utilizando el lenguaje de estilos CSS durante este paquete de trabajo del tercer sprint. Se consideraron las actividades incluidas en el diccionario de la EDT, además de realizar algunas mejoras adicionales como la inclusión de una librería encargada de dar formato al texto en formato markdown de las respuestas del asistente.

¹⁴<https://sequelize.org/docs/v6/getting-started/>

¹⁵<https://www.postman.com>

9.4.2. Versión final del chatbot

Este paquete de trabajo tiene como objetivo brindar al usuario un asistente de creación y definición de mashups. Por ello, se realizaron cada una de las actividades propuestas para mejorar así la versión inicial del chatbot. Para la autogeneración de mashups mediante el chatbot se tuvo que diseñar un método que comunicase Node-RED con la respuesta proporcionada por el asistente de OpenAI. Además, para la descripción y el detalle de los mashups se diseñó un método capaz de almacenar toda la información del mashup dentro de Node-RED y poder mostrarla en el sistema.

9.4.3. Gestión del chatbot

En la etapa final del desarrollo e implementación, se diseñó la interfaz de administración del chatbot. En principio, se concibió como un listado de chats que permitía eliminarlos, así como acciones adicionales como desactivar el chatbot y acceder a los costos asociados. Como se mencionó en la sección 10.3.4, el asistente puede ser configurado con instrucciones adaptables para desempeñar roles específicos. Sin embargo, inicialmente no se consideró la posibilidad de crear un asistente con una configuración diferente a la predeterminada. Por tanto, se agregó un editor de instrucciones y un formulario para la creación de asistentes (ver la Figura 9.4).

Además, para la gestión del chatbot, se optó por utilizar el número de asistentes como criterio (dado que crear asistentes es más costoso que crear hilos). Por lo tanto, en lugar de mostrar el listado de chats, se presenta un listado de asistentes con su respectiva disponibilidad y la opción de eliminarlos (ver la Figura 9.4).

Instrucciones del asistente

You are an expert in interpreting node-RED flows with two main roles:

- If the user sends you a node-red flow, your role is:
 - To describe the flow you receive to non-technical people.
 - To describe the flow step by step to non-technical people.
- If the user sends you a description of a flow, your role is to return the node-red flow that corresponds to the description. Restrictions: Only the JSON, with a tab component and http in and response to test the flow and you must create randomly the id of the nodes with 16 characters.

Below is a list of customised nodes that can appear in flows along with their input and output:

- trello-collector: Given the id of a trello board, the API key and the token, return the github repositories linked

Gestión de asistentes

Nombre	Estado	Acciones
asst_4nWdNglySwbrn6...	Libre	OK
asst_SufNKCWwv7B0G...	Libre	OK
asst_XRxcJRlqhfoEISK...	Libre	OK
asst_2vrFnFrbuBEh4D...	Libre	OK

Figura 9.4: Administración del asistente

Por último, a pesar de que no se contemplase en la planificación inicial, siguiendo las recomendaciones de los tutores se realizó la documentación de la API mediante Swagger (ver en la Figura 9.5). Para facilitar dicha documentación se utilizó una librería¹⁶ llamada “swagger-jsdoc”,

¹⁶<https://www.npmjs.com/package/swagger-jsdoc>

que facilita la documentación mediante comentarios en el código (en lugar de utilizar un archivo YAML).

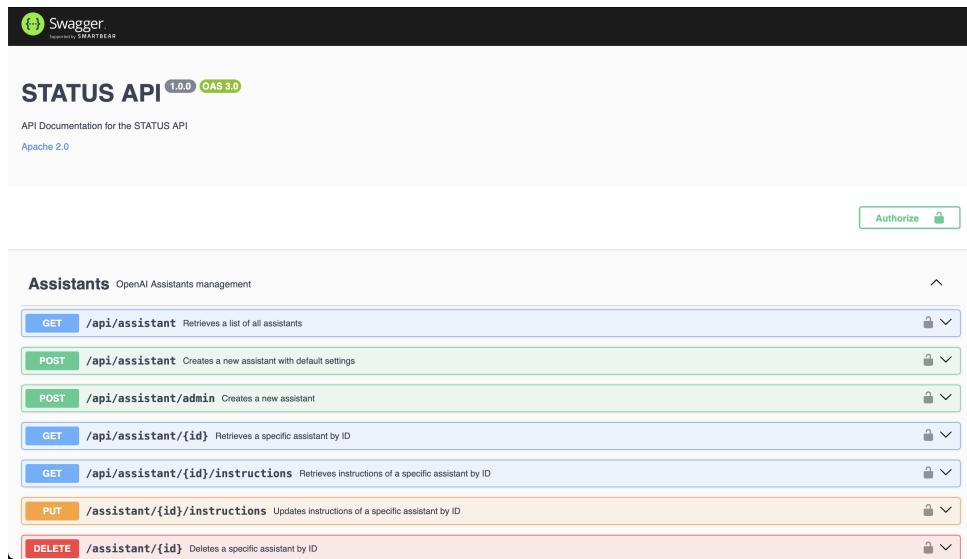


Figura 9.5: Documentación de la API

9.4.4. Versión completa de la memoria

Para la realización de esta memoria, he empleado una amplia gama de herramientas con el objetivo de garantizar un documento organizado, atractivo y detallado. La herramienta principal utilizada ha sido Overleaf, un editor online de LaTeX que ha sido fundamental en la elaboración de la memoria. Además de esta plataforma, he hecho uso de otras herramientas complementarias como Draw.io¹⁷ para la creación de diagramas de clase, EDT, diagrama de arquitectura lógica del sistema, diagrama de contexto, entre otros. También he empleado Gantt PRO¹⁸ para la elaboración del cronograma y Mermaidchart¹⁹ para la creación de diagramas de flujo y secuencia.

9.4.5. Cierre del Sprint

Durante este último sprint han surgido diferentes retos a superar para conseguir realizar con éxito el proyecto, entre ellos:

- **Uso de CSS para la mejora de estilos:** Mejorar los estilos del sistema mediante CSS con escaso conocimiento sobre este lenguaje ha ocasionado un gran gasto del tiempo, superando la estimación esperada.
- **Uso de estados globales en el lado del cliente:** Inicialmente se decidió emplear *hooks* de react del tipo *useState* para actualizar el estado de la configuración del sistema. Sin embargo, esta elección resultó en un incremento de la latencia del sistema debido a la alta frecuencia de solicitudes para actualizar el estado de manera global. Finalmente, se modificó la implementación para utilizar un hook que transmite la información a través del árbol de componentes de react llamado *useContext*. La adopción de *useContext* como hook para gestionar el estado global en el lado del cliente proporciona una solución más eficiente, organizada y escalable en comparación con el enfoque inicial utilizando *useState*.

¹⁷<https://app.diagrams.net>

¹⁸<https://ganttpro.com>

¹⁹<https://mermaid.js.org>

Este sprint terminó el día 3 de Mayo, dando comienzo a la última etapa del proyecto, el cierre. En esta última fase se realizó la documentación online requerida, las acciones de cierre correspondientes, se recogieron las lecciones aprendidas a lo largo del desarrollo y se realizó la presentación del trabajo.

10. Análisis del Sistema

En este capítulo se presenta un análisis detallado de la arquitectura, componentes, y funcionalidades del sistema desarrollado. Se examinan los aspectos técnicos del chatbot, así como la integración con otros sistemas y la interfaz de usuario. Este análisis proporciona una visión integral de la estructura y capacidades del sistema, destacando los elementos clave de su diseño y funcionamiento.

10.1. Análisis de la Arquitectura

El diagrama de despliegue de la Figura 10.1 muestra la infraestructura final del sistema, detallando los dispositivos y componentes que interactúan entre sí. El cliente del usuario utiliza NodeJS como entorno de ejecución para el componente Node-RED. Además, el cliente utiliza un navegador web para interactuar con la aplicación, comunicándose con otros componentes a través de HTTP. El servidor web, que también usa NodeJS, ejecuta una aplicación principal construida con el framework ExpressJS. La comunicación entre el servidor web y el cliente, así como con otros componentes, se realiza a través de HTTP y HTTPS.

El servidor de base de datos utiliza MySQL para almacenar la información del sistema y se interactúa con esta base de datos mediante Sequelize ORM, una herramienta de mapeo objeto-relacional. Además, el sistema se comunica con la infraestructura externa BlueJay a través de HTTP (para generar las gráficas de cumplimiento), y con el servidor de OpenAI a través de HTTPS.

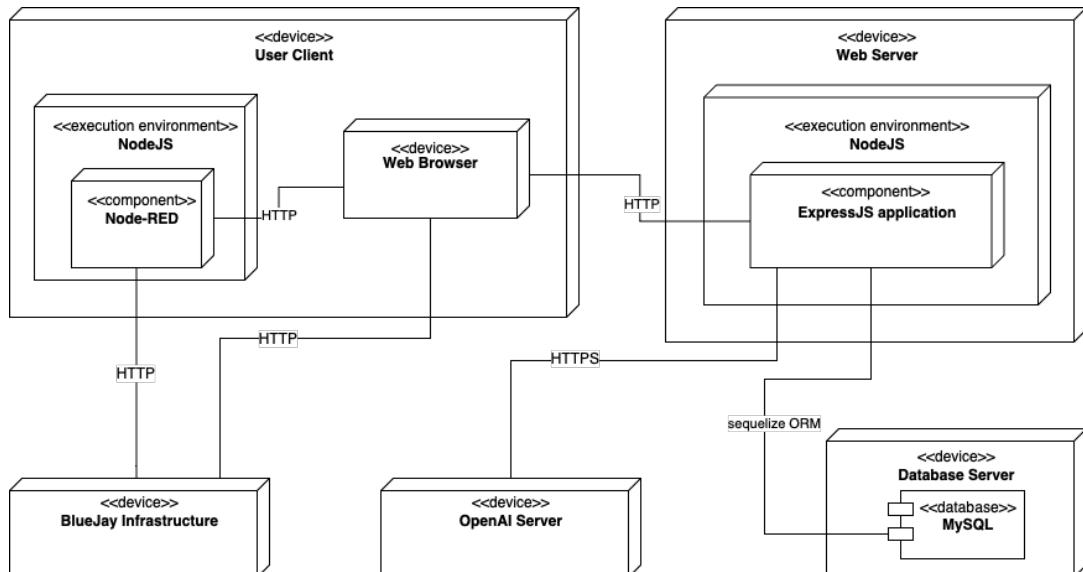


Figura 10.1: Infraestructura de STATUS

Por otro lado, el resultado final de la lógica del sistema coincide con la propuesta realizada en la Figura 7.1 del apartado 7.1, comunicando el sistema con servicios externos cómo son Node-RED, Bluejay y OpenAI. Además, se ha refactorizado la capa de lógica de negocio para contemplar modelos y middlewares, con el objetivo de mejorar la seguridad y eficiencia del sistema. Esta capa inicialmente solo disponía de controladores que se comunicaban directamente con la capa de persistencia de datos mediante peticiones SQL nativas y modelos (definidos en lenguaje

SQL). El uso de peticiones SQL nativas puede introducir vulnerabilidades de seguridad, como ataques de *SQL injection*, si no se manejan adecuadamente los datos proporcionados por el usuario. Además, este enfoque tiende a generar código altamente acoplado y difícil de mantener, ya que las consultas están dispersas a lo largo del código y cualquier cambio en la estructura de la base de datos puede requerir modificaciones extensas en múltiples partes del sistema.

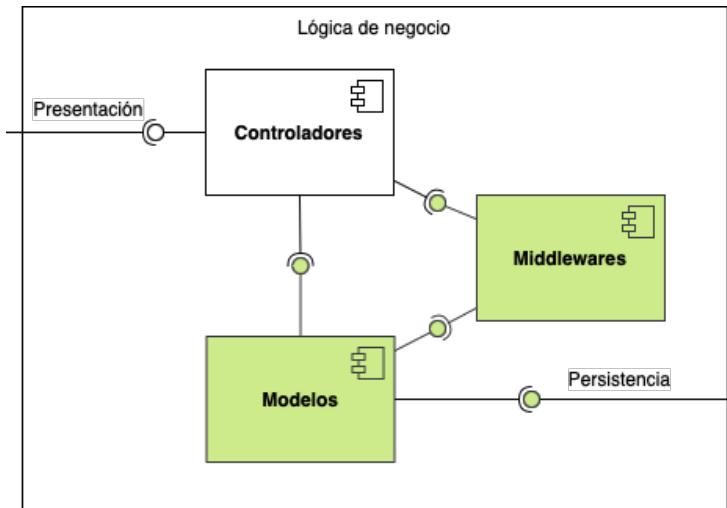


Figura 10.2: Arquitectura de la capa de lógica de negocio

Como se puede apreciar en la Figura 10.2, se han realizado cambios significativos en la arquitectura del sistema. Desde una estructura que solamente comprendía controladores y modelos en SQL, se ha evolucionado hacia una más robusta y flexible. Ahora, los modelos se adaptan para utilizar un mapeo objeto-relacional, también conocido como *Object-Relational Mapping* (ORM). Este ORM no solo simplifica las consultas a la base de datos, sino que también asegura la integridad y seguridad de los datos mediante técnicas como la prevención de inyección de SQL y la gestión de transacciones de manera robusta.

Cabe destacar, que el diagrama de servicios que aparece en la Figura 10.3 representa todos los servicios contenidos en el sistema; sin embargo, los únicos que se han implementado en este trabajo han sido los de Asistente, Hilo, Mashup, Configuración, Autorización y los servicios externos de OpenAi, Node-RED, Github y Trello.

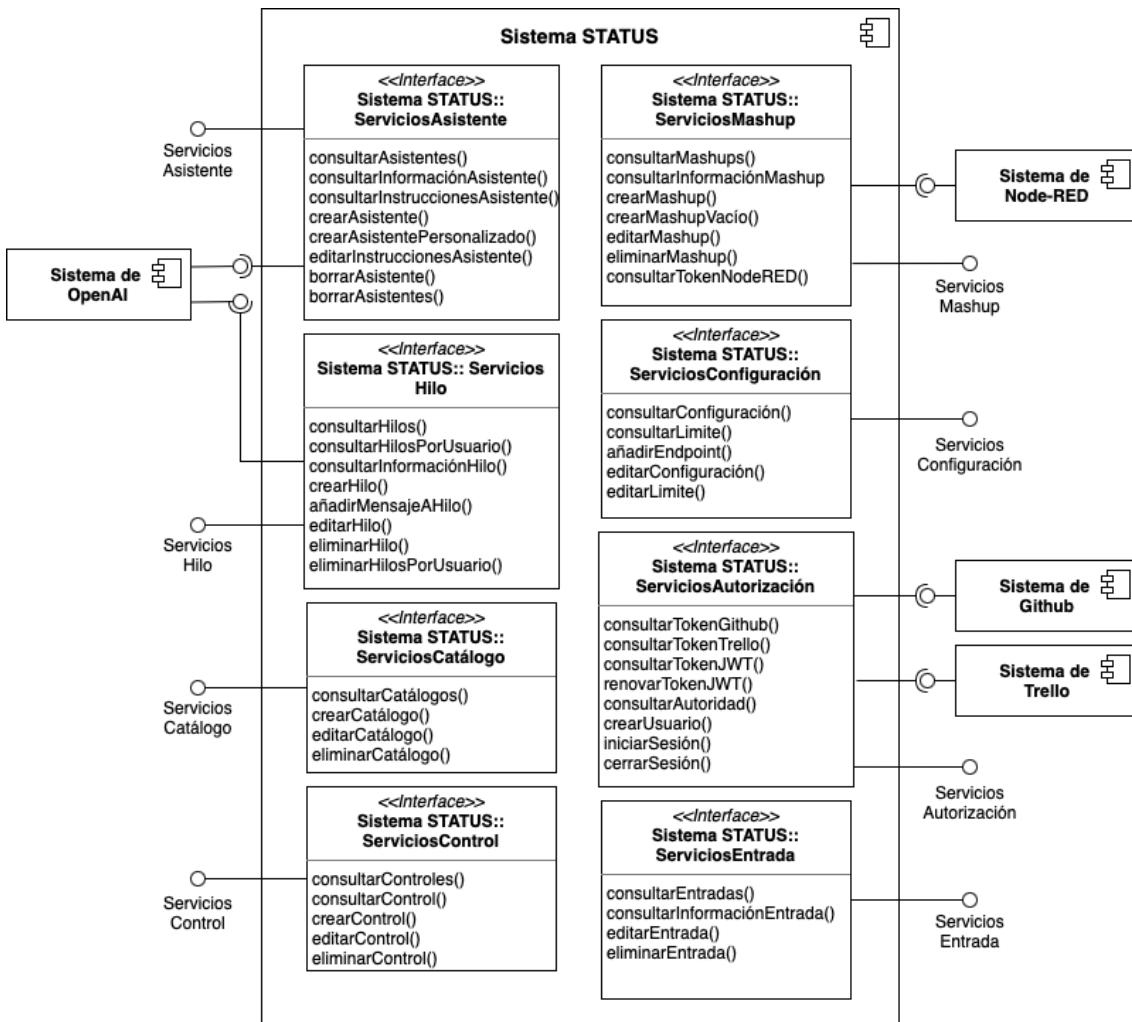


Figura 10.3: Servicios del sistema STATUS

A continuación, se describen los servicios asociados al Sistema STATUS mostrados en el diagrama de servicios la Figura 10.3:

- El módulo de **ServiciosAsistente** permite gestionar los asistentes en el sistema. Ofrece métodos para listar todos los asistentes disponibles, consultar información detallada e instrucciones asociadas a un asistente específico, y crear nuevos asistentes, incluyendo opciones de configuración personalizada. También permite editar las instrucciones de los asistentes, así como eliminar uno o todos los asistentes del sistema.
- El módulo de **ServiciosHilo** se encarga de la gestión de hilos de conversación. Incluye métodos para listar todos los hilos disponibles, consultar hilos asociados a un usuario específico, y obtener información detallada sobre un hilo en particular. Además, permite crear nuevos hilos, añadir mensajes a hilos existentes, y editar o eliminar hilos.
- El módulo de **ServiciosCatálogo** facilita la gestión de catálogos de controles en el sistema. Proporciona métodos para listar todos los catálogos disponibles, crear nuevos catálogos, y editar o eliminar catálogos existentes (Realizado por el técnico Francisco Montero).
- El módulo de **ServiciosControl** permite gestionar los controles en el sistema. Incluye métodos para listar todos los controles disponibles, consultar información detallada sobre un control específico, y crear, editar o eliminar controles (Realizado por el técnico Francisco Montero).

- El módulo de **ServiciosMashup** se encarga de la gestión de mashups en el sistema. Ofrece métodos para listar todos los mashups disponibles, consultar información detallada sobre un mashup específico, y crear nuevos mashups, incluyendo la opción de crear mashups vacíos. También permite editar y eliminar mashups existentes, y consultar el token de autenticación para el sistema Node-RED.
- El módulo de **ServiciosConfiguración** permite gestionar la configuración del sistema. Proporciona métodos para consultar la configuración actual, los límites establecidos, y añadir nuevos endpoints. Además, permite editar tanto la configuración como los límites del sistema.
- El módulo de **ServiciosAutorización** facilita la gestión de tokens y la autorización de usuarios. Incluye métodos para consultar tokens de autenticación para sistemas externos como Github y Trello, así como tokens JWT (JSON Web Token). También permite renovar tokens JWT, consultar la autoridad de un usuario, y gestionar usuarios mediante la creación, inicio de sesión y cierre de sesión.
- El módulo de **ServiciosEntrada** se encarga de la gestión de entradas de controles en el sistema. Ofrece métodos para listar todas las entradas disponibles, consultar información detallada sobre una entrada específica, y editar o eliminar entradas existentes (Realizado por el técnico Francisco Montero).

El Sistema STATUS integra varios servicios externos como OpenAI, Node-RED, Github y Trello, a través de sus respectivos tokens de autenticación, lo que permite una funcionalidad extendida. Por último, en la Figura 10.4 se muestra el modelo de datos resultante del desarrollo de este TFG, en él aparecen las últimas incorporaciones mencionadas en el apartado 9.4.3 (además de las modificaciones realizadas a lo largo de los sprints).

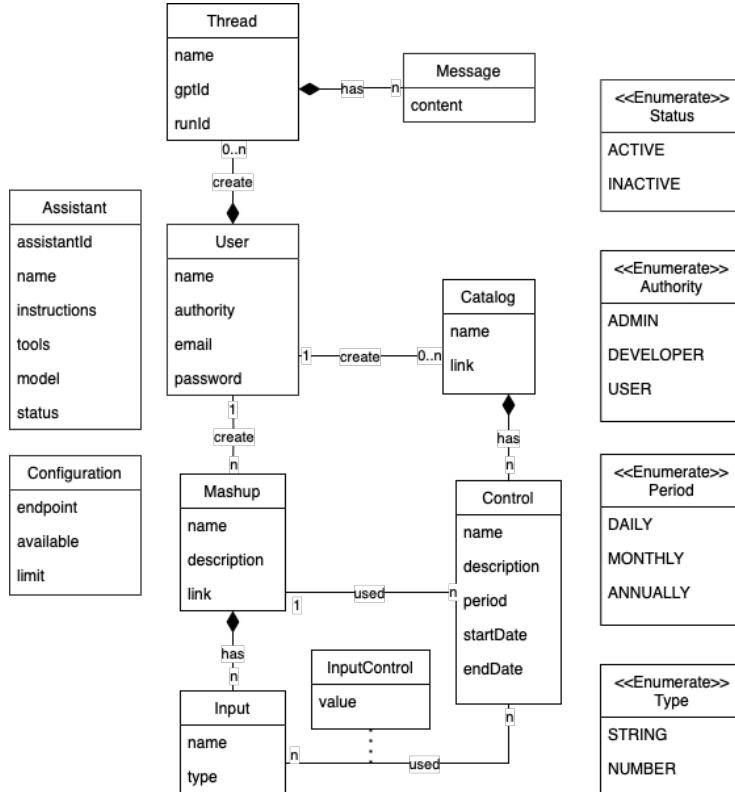


Figura 10.4: Diagrama de clases final

10.2. Análisis de los Nodos Realizados

Ese apartado se centra en comentar de forma detallada el análisis realizado para desarrollar los nodos personalizados de Node-RED, además de comentar algunos problemas que han aparecido a lo largo del desarrollo y un análisis de las decisiones tomadas.

10.2.1. Traducción de los nodos

Como se comentó en el apartado 9.1.3, en base a los mashups que aparecían en *Enterprise Architect* se realizó un documento en el que se detallaba la traducción realizada, resultando así el cuadro 10.1.

Componente EA	Node-RED
FilterByImportancia	
FilterByContentType	
FilterByActorPipe	
FilterByDateFromToday	filter by
FilterByState	
FilterByActividadPipe	
FilterByActivoPipe	
FilterByDate	filter by date
AndPipe	-
CopySplitPipe	split
ExisteSeccionEnGesMaDocumentPipe	exists section in doc
ExistsPipe	exists pipe
ExistsUrlInGesMaPipe	exists url
ForEachProject	split & join
ProyectoSubproyectoToUrlPipe	project to url
Result	http response
UrlToGesMaDocumentPipe	url to doc

Cuadro 10.1: Traducción de nodos de EA a Node-RED

Algunos de los componentes de *Enterprise Architect* no necesitaron traducción ya que, Node-RED incluye algunos nodos por defecto que ya realizan esas funciones (como *split*, *join* y *http response*).

10.2.2. Creación de los nodos

Con esa traducción, se realizó la implementación inicial de los nodos. Para realizar dicha implementación se escogió una forma sencilla e intuitiva de diseño de estos (Ver Figura 10.5).

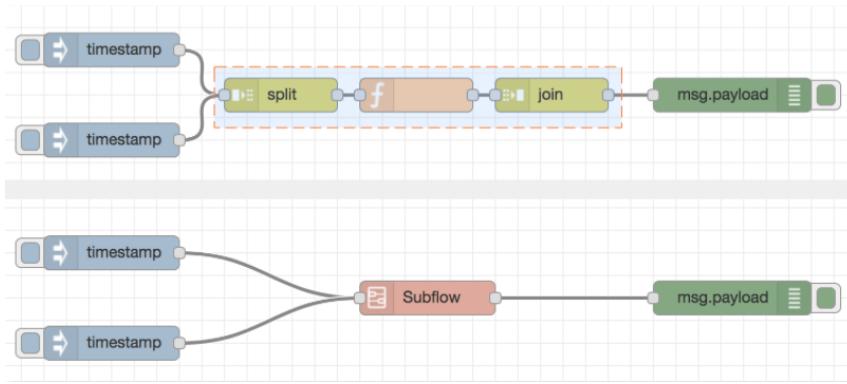


Figura 10.5: Subflujos de Node-RED
(Fuente: Documentación de Node-RED¹⁾)

Con este diseño se utilizan los nodos básicos de node-RED y se encapsulan en un nodo llamado “subflujo” (que se almacena en formato JSON), y para las acciones más complejas se utiliza el nodo “function” que permite ejecutar código en Javascript dentro del mashup.

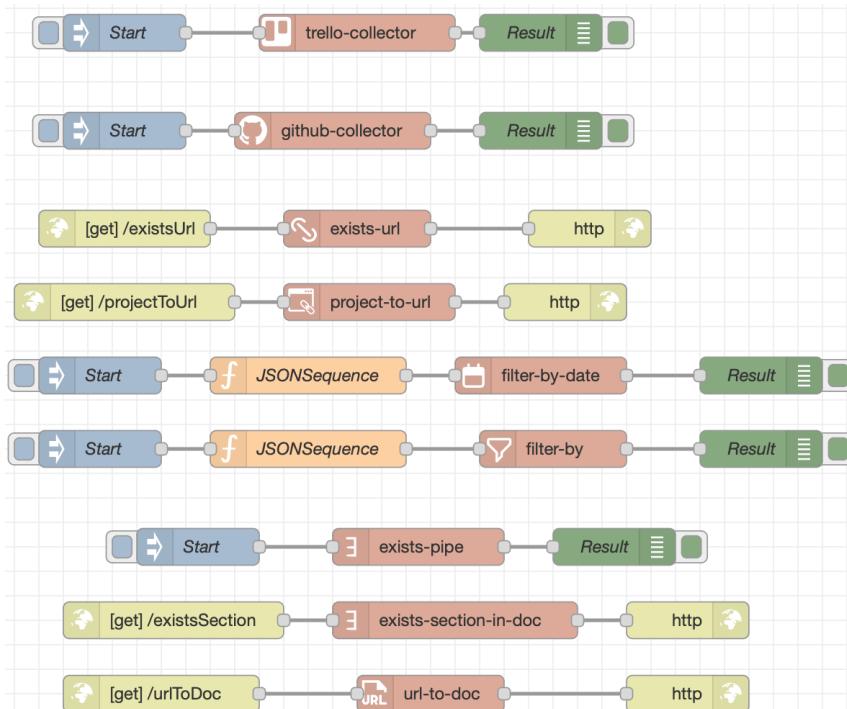


Figura 10.6: Nodos finales con iconos ilustrativos

En la Figura 10.6 podemos ver el resultado de esta la fase de implementación de los nodos del apartado 9.1.4. A continuación, se describen cada uno de los nodos realizados:

- **Exists-url:** Dada una tarjeta de un tablero de Trello, una APIKey de Trello y un token de Trello; devuelve *true* si esa tarjeta está asociada a un repositorio de GitHub.
- **Project-to-url:** Dado un tablero y una card de Trello, una APIKey de Trello y un token de Trello; devuelve la url del repositorio de GitHub asociado a esa tarjeta.
- **Filter-by-date:** Dado un intervalo de tiempo y el nombre de la propiedad que contiene la fecha de un objeto, filtra los objetos con la fecha dentro de ese intervalo.

- **Filter-by:** Dado un tipo de filtro, el valor a filtrar y el nombre de la propiedad a filtrar, filtra los objetos según el valor a filtrar.
- **Exists-pipe:** Dada una lista de objetos y un tamaño, devuelve *true* si el tamaño de la lista coincide con el tamaño introducido.
- **Exists-section-in-doc:** Dada una url a un repositorio de GitHub, la ubicación del documento a comprobar, una cadena de texto , una APIKey y un token de Github; devuelve *true* si la cadena está contenida en el documento.
- **Url-to-doc:** Dada una url a un repositorio de GitHub, la ubicación del documento y un token de Github; devuelve el texto del documento.

De forma adicional, siguiendo las recomendaciones de los tutores, se añadió a los nodos una icono más ilustrativo del utilizado por defecto en Node-RED. En la Figura 10.6 se puede ver la versión final de estos nodos personalizados con iconos introducidos como imágenes vectoriales (formato SVG). En dicha imagen aparecen los nodos personalizados diseñados por el técnico Francisco Montero *github-collector* y *trello-collector*.

10.2.3. Empaquetado y publicación de nodos

Como se comentó en el apartado 8.1 uno de los aspectos negativos de esta herramienta es la detección de fallos. En particular, durante la actividad de “Publicación inicial de nodos” se encontró que, si bien el paquete se empaquetaba correctamente al probarlo en el entorno local, no se importaban los subflujos creados. Tras varias pruebas se decidió indagar sobre el empaquetado de estos, que según la documentación de node-RED, tienen una forma diferente de empaquetarse² cómo módulos de la aplicación (dicha funcionalidad se considera experimental debido a que es reciente).

Después de eso, se intentó seguir la documentación de Node-RED para empaquetar los subflujos. No obstante, se encontró que el ejemplo proporcionado en la documentación era muy básico y no abordaba situaciones complejas. Además, se requerían cambios en la estructura del JSON de forma manual, lo cual representaba un desafío, especialmente con subflujos grandes (algún subflujo pequeño sí se logró importar correctamente). Esta tarea manual aumentaba el riesgo de cometer errores fácilmente, y dado que detectar estos errores en el empaquetado de subflujos es complicado, la tarea se volvía aún más desafiante.

Tras comentar la situación al tutor y debatir sobre una solución posible, se plantearon diferentes alternativas:

1. Continuar revisando la estructura de los subflujos y revisar el foro de la comunidad de desarrolladores de Node-RED, en busca de una solución.
2. Refactorizar los subflujos a “nodos personalizados” una tarea que puede llevar tiempo, aunque es más sencilla porque hay más documentación al respecto y ya se había realizado durante el curso de formación.

Finalmente, se escogió la segunda opción, aunque con la tarea adicional de revisar el foro en busca de algo que pudiera servir de ayuda para solucionar el problema inicial. No se encontraron entradas en el foro al respecto, aunque el aspecto positivo fue que no hubo problemas para refactorizar los subflujos a nodos personalizados. Estos nodos personalizados se realizan en código javascript, además de html para configurar las entradas y la apariencia del nodo en el editor de Node-RED. A pesar de ello, la actividad que inicialmente se estimó que iba a durar 5 horas, terminó costando 14 horas.

²<https://nodered.org/docs/creating-nodes/subflow-modules>

10.3. Análisis del Chatbot

El funcionamiento básico del asistente de OpenAI se basa en un modelo de inteligencia artificial llamado GPT (Generative Pre-trained Transformer), que se ha vuelto bastante popular en los últimos años. Este modelo ha sido entrenado con una amplia cantidad de texto de diversos temas y está diseñado para generar texto coherente y relevante en respuesta a las entradas proporcionadas por los usuarios (Kalla y cols., 2023).

10.3.1. Instrucciones del asistente

Un asistente³ tiene instrucciones y puede aprovechar modelos, herramientas y archivos para responder a las consultas de los usuarios. Las instrucciones del asistente son fundamentales para definir su comportamiento y personalidad. Las instrucciones pueden ser utilizadas para definir la personalidad del asistente, en este caso, para describir y generar mashups en Node-RED.

Con el fin de obtener una primera versión de las instrucciones del asistente, se hicieron pruebas iniciales con mashups para obtener su descripción y al revés en el *playground*⁴ de OpenAI. Los resultados iniciales fueron bastante buenos ya que, las descripciones realizadas por el asistente eran similares a la descripción real en la mayoría de ocasiones.

10.3.2. Interacción con el asistente

Se puede interactuar con el asistente a través de la API y a través de bibliotecas en Python o en NodeJS. La API permite a los desarrolladores crear y gestionar asistentes, mientras que las bibliotecas proporcionan una interfaz más fácil de usar para interactuar con el asistente (En el Extracto de código 10.1 se utiliza la biblioteca de Node.js “openai”).

10.3.3. Modelos y herramientas

El asistente de OpenAI cuenta con una variedad de modelos y herramientas que permiten a los asistentes realizar tareas específicas. Algunos de los modelos incluyen GPT-4, GPT-3.5, DALL-E, TTS, Whisper, Embeddings, Moderation, y otros⁵.

Actualmente, la API de asistentes admite tres tipos de herramientas: Intérprete de código, Búsqueda de archivos y Llamada a funciones. El intérprete de código permite a los asistentes ejecutar código Python o Javascript, lo que es útil para tareas que requieren análisis de datos o procesamiento de información. La recuperación de conocimientos permite a los asistentes acceder a bases de datos y recuperar información específica. La llamada a funciones permite a los asistentes interactuar con APIs externas y realizar tareas más complejas. En este caso, tanto el Intérprete de código como la Llamada a funciones pueden ser útiles para la implementación del chatbot, aunque con el objetivo de reducir la complejidad de la implementación se ha escogido utilizar el Intérprete de código.

³<https://platform.openai.com/docs/assistants/overview>

⁴<https://platform.openai.com/playground/assistants>

⁵<https://platform.openai.com/docs/models/overview>

10.3.4. Implementación del chatbot

El objeto del asistente es un conjunto de propiedades que incluyen el nombre, las instrucciones, las herramientas y el modelo donde se ejecuta (Ver el Extracto de código 10.1), entre otras propiedades más. Este asistente se encarga de ejecutar una serie de “hilos”(conversaciones entre el usuario y el asistente), que se componen de mensajes que pueden ser texto, imágenes o archivos. Los hilos se crean cuando un usuario inicia una conversación con el asistente y se ejecutan cuando el asistente procesa los mensajes y responde.

```
1 import OpenAI from "openai";
2 const openai = new OpenAI();
3
4 async function main() {
5   const assistant = await openai.beta.assistants.create({
6     name: "Math Tutor",
7     instructions: "You are a personal math tutor. Write and run code to
8       answer math questions.",
9     tools: [{ type: "code_interpreter" }],
10    model: "gpt-4-turbo"
11  });
12 main();
```

Extracto de código 10.1: Ejemplo básico de asistente

Para la integración del chatbot en el sistema, se decidió almacenar la siguiente información:

- **Assistant:** Entidad que almacena la información de cada asistente, sus atributos son el identificador del asistente en OpenAI, el nombre, las instrucciones, las herramientas, el modelo que utiliza y el estado.
- **Thread:** Entidad que almacena la información de los hilos de un usuario, sus atributos son el identificador del hilo en OpenAI, nombre del hilo e identificador de la última ejecución.
- **Message:** Entidad que almacena la información de los mensajes de un hilo, su único atributo es el contenido del mensaje.

Además, se hicieron algunas modificaciones en el modelo de datos para simplificar el diseño. En la Figura 10.7 aparece el diagrama de clases, incluyendo de color verde las entidades que han sido modificadas o incorporadas.

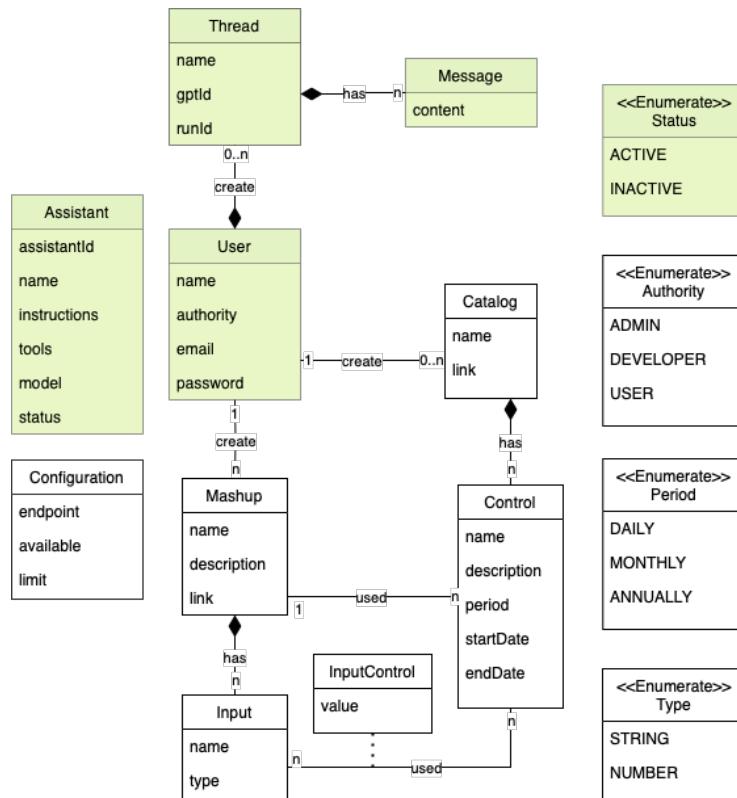


Figura 10.7: Diagrama de clases

Si comparamos la Figura 10.7 con el modelo de datos inicial (Figura 7.2), podemos observar que se ha eliminado la entidad *Authority* (que tenía una relación de muchos a muchos con usuario) y se ha añadido un atributo de tipo Enumerado a la entidad *User*. De esta forma, cuando un usuario tiene autoridad de administrador, este también tiene acceso a las funcionalidades de los técnicos y de los usuarios (evitando así relaciones de muchos a muchos).

Es importante resaltar que la entidad *Assistant* no tiene ninguna relación específica, ya que al procesar un hilo, OpenAI brinda la opción de utilizar cualquier asistente disponible. Con esto en mente, se busca desarrollar un algoritmo que, al crear un nuevo hilo o enviar un mensaje en un hilo ya existente, seleccione a un asistente con estado inactivo (atributo *status* de tipo enumerado). Este enfoque tiene como objetivo mejorar la eficiencia del chatbot y reducir las colas (hilos a la espera de ser ejecutados por un asistente). Por último, mencionar que en base a las pruebas informales realizadas, se ha establecido al asistente la siguiente configuración por defecto:

- **Nombre:** STATUS Assistant.
- **Modelo:** GPT-3.5-turbo-0125.
- **Herramientas:** Intérprete de código.
- **Estado:** Inactivo.

10.3.5. Recomendación de descripciones

El diagrama de secuencia de la Figura 10.8 describe la interacción entre un usuario, la capa de presentación, la capa de lógica de negocio y la API de OpenAI para procesar la solicitud de describir un mashup. El usuario envía una solicitud de mashup a la capa de presentación. Si el mensaje es válido, la capa de presentación pasa la solicitud a la capa de lógica de negocio, que a

su vez consulta a la API de OpenAI para obtener una descripción. La API de OpenAI responde con la descripción, que es devuelta al usuario a través de la capa de presentación. Si el mensaje es inválido, la capa de lógica de negocio responde directamente a la capa de presentación indicando la invalidez, y esta a su vez informa al usuario.

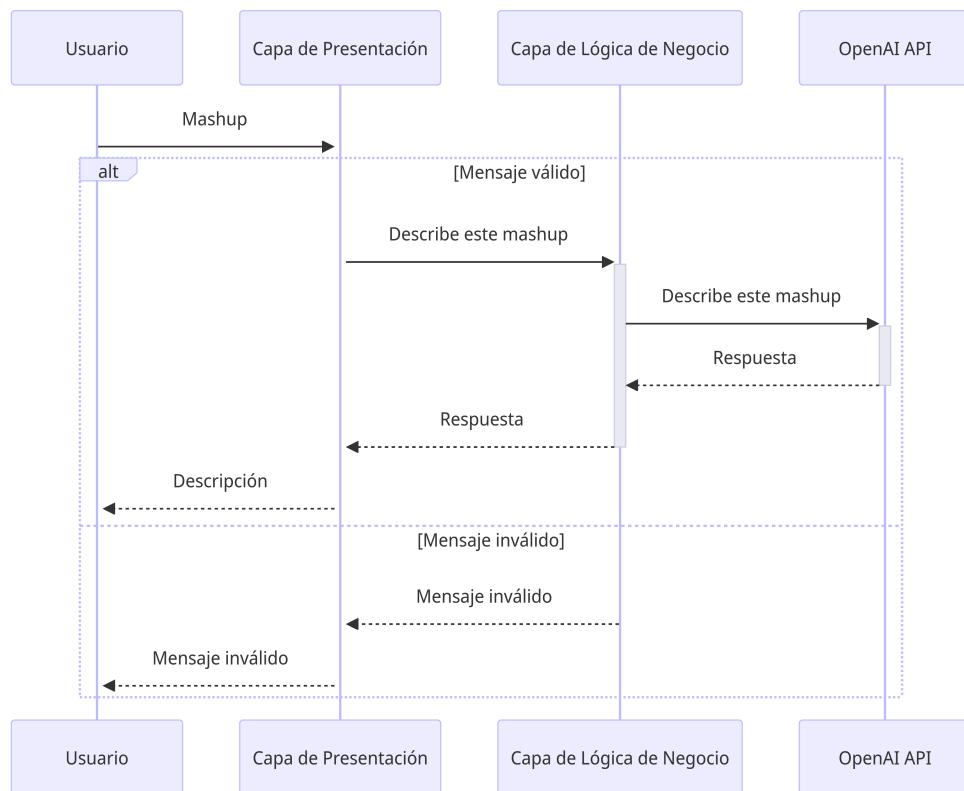


Figura 10.8: Descripción de *mashups*

10.3.6. Creación de mashups autogenerados

La Figura 10.9 ilustra el proceso de generación y adición de un mashup solicitado por un usuario, implicando a la capa de presentación, la capa de lógica de negocio, la API de OpenAI y Node-RED. El usuario solicita una descripción a la capa de presentación. Si el mensaje es válido, la capa de presentación solicita a la capa de lógica de negocio la generación del mashup, que a su vez consulta a la API de OpenAI para generararlo. La API de OpenAI devuelve la información generada, que luego es enviada a Node-RED para añadir el mashup. Node-RED confirma la adición, y la capa de Lógica de Negocio envía la respuesta a la capa de presentación, que finalmente muestra el editor de Node-RED al usuario. Si el mensaje es inválido, la capa de lógica de negocio informa a la capa de presentación, que comunica al usuario la invalidez del mensaje. Una de las instrucciones que se especifica al asistente es que al recibir la descripción de un mashup devuelva únicamente un JSON en markdown. El mensaje al recibirse es clasificado, si contiene un JSON se extrae el mashup y se crea en Node-Red mediante la API.

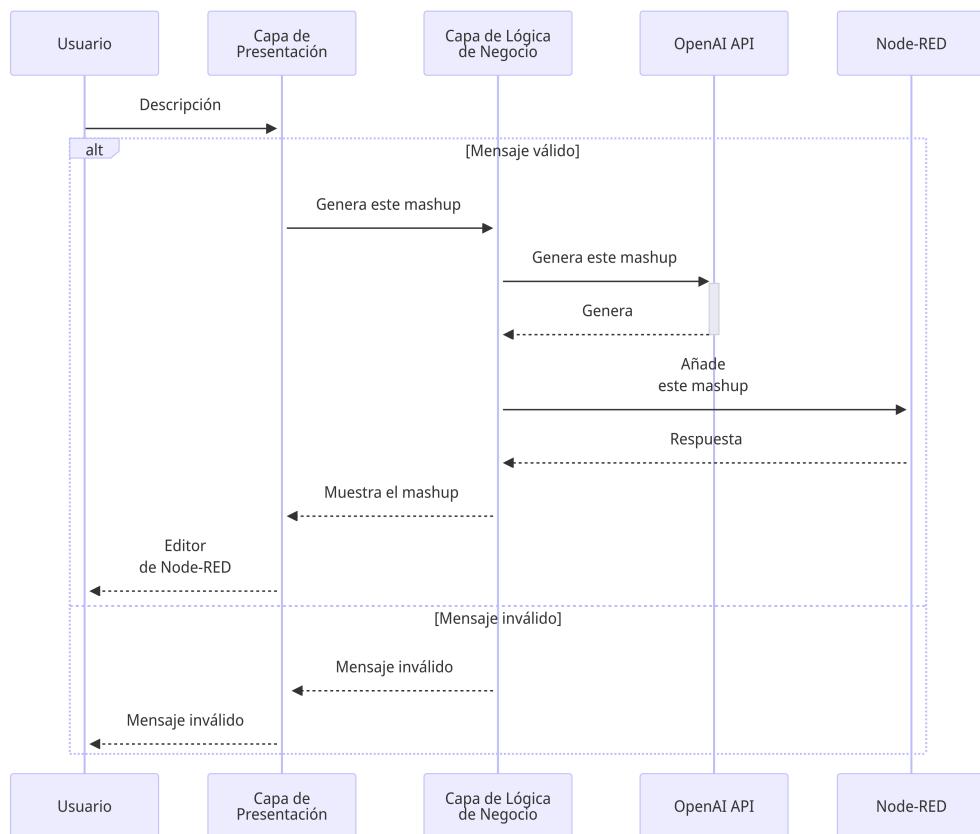


Figura 10.9: Generación de *mashups* a partir de una descripción

Un ejemplo de *prompt* sería:

Given a filter criteria, check if a section exists within the GitHub repository documents associated with cards on a trello board.

10.3.7. Generar y almacenar detalles del mashup

Cómo hemos comentado en lo largo de la memoria, los mashups de Node-RED son archivos de tipo JSON que almacenan toda la información correspondiente al flujo.

```

1  [
2      {
3          "id": "58fffae9dfa7005",
4          "type": "tab",
5          "label": "Check URL Existence",
6          "disabled": false,
7          "info": "Mashup description"
8      }
9  ]

```

Extracto de código 10.2: Ejemplo básico de flujo de Node-RED

El trozo de código 10.2 muestra un ejemplo básico de la estructura de un flujo de Node-RED. En caso de que el flujo contuviese algún componente se añadiría como un elemento más del array de objetos, distinguiéndose el flujo como un objeto de tipo “tab”. Uno de las propiedades de este objetos es la información del flujo, por ello se escogió almacenar en esta propiedad la

información correspondiente al mashup (tanto descripción cómo detalles). En la Figura 10.10 aparece un mashup de ejemplo junto con la estructura propuesta de descripción y detalle.

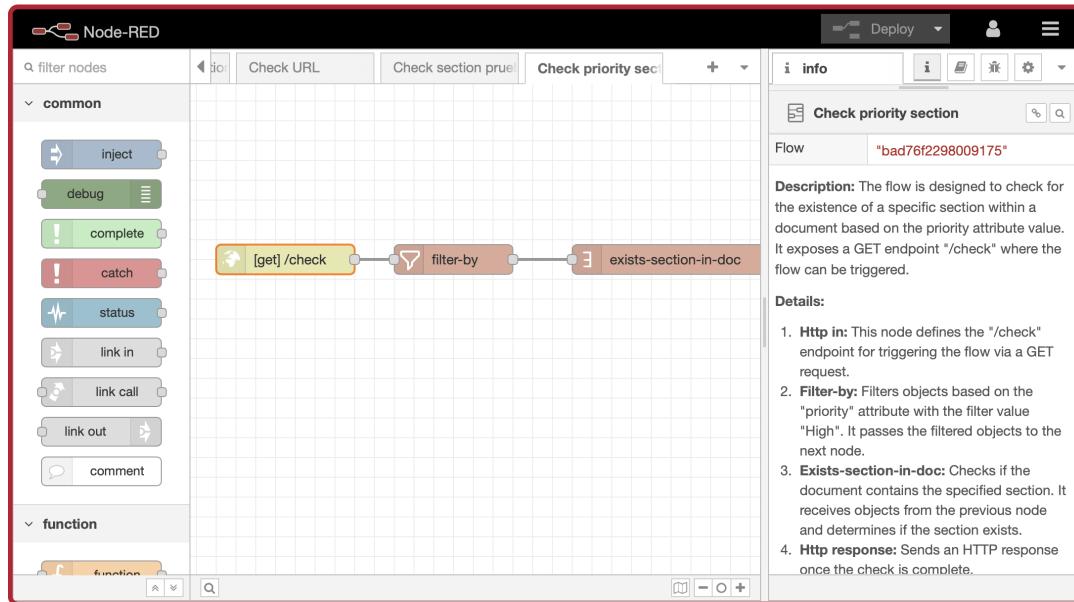


Figura 10.10: Información del mashup

Para la vista de detalle del mashup se formatea esta información del mashup de forma que se muestra cómo descripción únicamente el texto después de “*Description*” y cómo detalle lo que aparece después de “*Details*” (ver la figura 10.11).

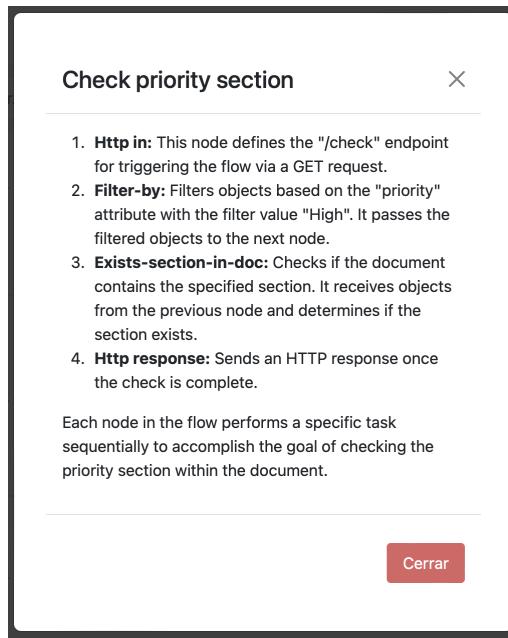


Figura 10.11: Modal de detalles del mashup

10.3.8. Configuración de OpenAI

Es crucial abordar la implementación de la activación y desactivación del chatbot, ya que esta funcionalidad juega un papel fundamental en su operatividad y disponibilidad. A través de

un diagrama de secuencia, se analiza la interacción entre la capa de presentación y la lógica de negocio, destacando los procesos clave involucrados en esta operación (ver la Figura 10.12).

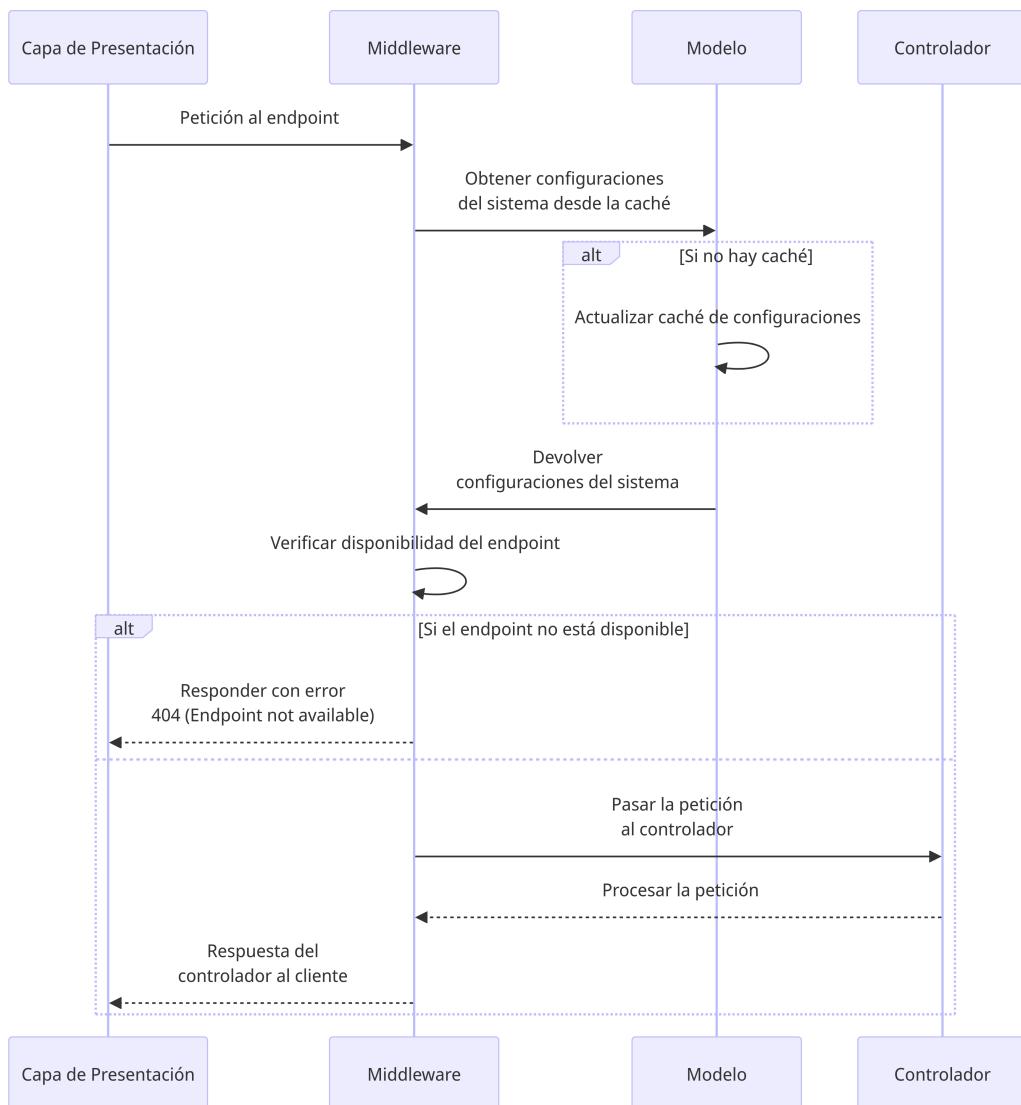


Figura 10.12: Activación/Desactivación del asistente

El diagrama de secuencia de la Figura 10.12 representa la comunicación entre la capa de presentación y la lógica de negocio para la funcionalidad de activación/desactivación del chatbot. En este diagrama, se observa cómo la capa de presentación envía una petición al middleware, el cual interactúa con el modelo para obtener la configuración del sistema. Si la caché no está presente, se actualiza. Luego, el middleware verifica si el endpoint está disponible. En caso afirmativo, la petición se transmite al controlador para su procesamiento. Finalmente, el controlador responde al middleware, que a su vez devuelve la respuesta al cliente.

Para implementar dicha configuración del sistema se ha añadido al diagrama de clases y a los modelos de la base de datos una entidad “Configuración” que contiene cómo atributos:

- **Endpoint:** Path relativo de la API del servicio que se está configurando.
- **Disponible:** Disponibilidad del endpoint (booleano) que representa si el endpoint está habilitado o no.

- **Límite:** Atributo que tiene como objetivo limitar el número de asistentes posibles, aunque se puede limitar la creación de otros endpoints.

Además de esto también se han implementado otras funcionalidades de gestión del chatbot como la eliminación de asistentes, el acceso a los costes que genera el servicio y la actualización de las instrucciones del asistente seleccionado.

10.4. Integración con Otros Sistemas

Para satisfacer los requisitos de integración que aparecen en el apartado 3.4, se detalla a continuación la integración de las plataformas Github, Trello y Node-RED. En el caso de las dos primeras, esta funcionalidad fue incluida en el apartado de “Perfil” del usuario (ver la Figura 10.13), almacenando los *tokens* de sesión de estas plataformas en el almacenamiento local del navegador.

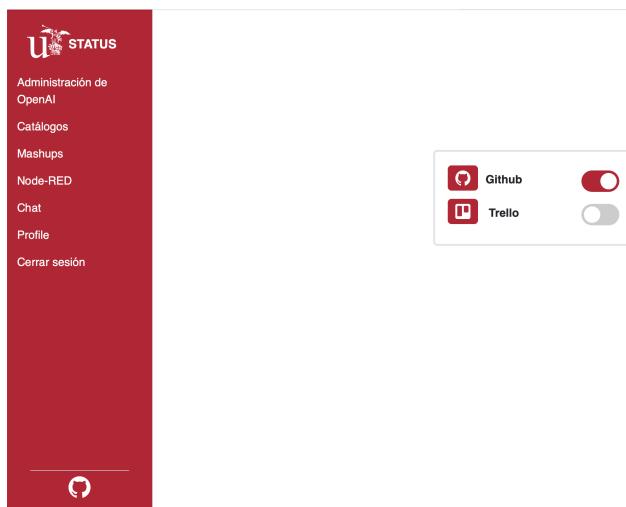


Figura 10.13: Conexión con Github y Trello

Para la conexión con Trello, se enfrentó un desafío particular: no es posible obtener automáticamente el token del usuario y almacenarlo. Por lo tanto, se optó por añadir un modal para que el usuario pueda introducir manualmente el token una vez el usuario pulsa el botón para conectar su cuenta de Trello (Figura 10.14).

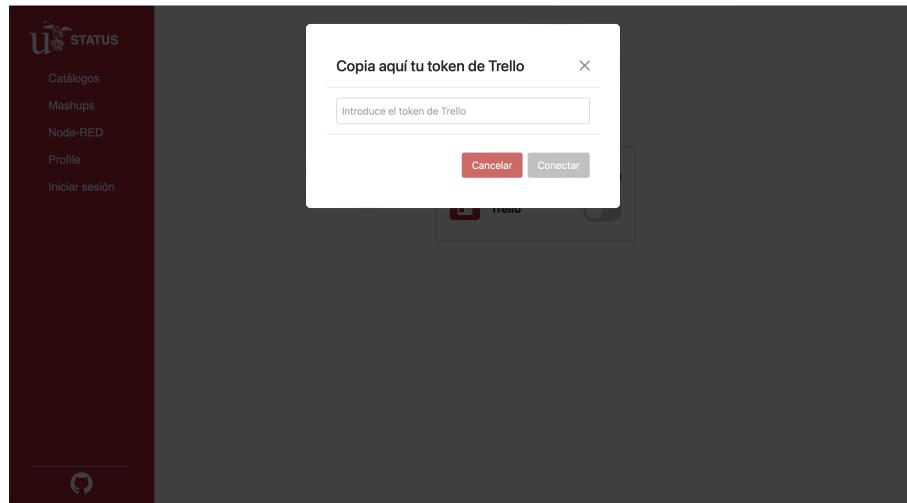


Figura 10.14: Modal para introducir el token de trello

Respecto a Node-RED, en la Figura 10.15 se muestra de forma sencilla la comunicación entre node-RED y el sistema, en concreto, el caso de acceso a la sección de *mashups*.

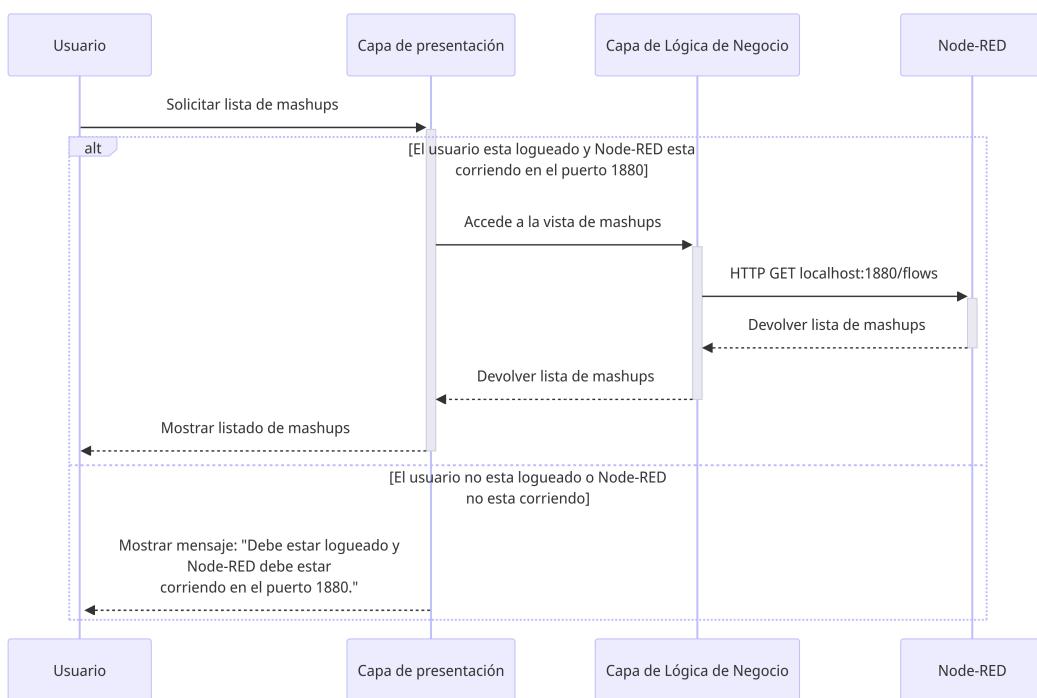


Figura 10.15: Comunicación entre el sistema y node-RED

Cuando un usuario desea acceder al listado de mashups, inicia la solicitud a través de la capa de presentación. Esta solicitud se procesa bajo ciertas condiciones: si el usuario está logueado y Node-RED está desplegado en el puerto 1880, la capa de presentación envía una solicitud a la capa de lógica de negocio para obtener el listado de mashups. La capa de lógica de negocio, a su vez, realiza una petición HTTP GET a `localhost:1880/flows` al despliegue local de Node-RED del usuario para obtener los mashups en tiempo real. Node-RED devuelve la lista de mashups a la capa de lógica de negocio, la cual transfiere esta información a la capa de presentación. Finalmente, la capa de presentación muestra la lista de mashups al usuario.

Sin embargo, si el usuario no está logueado o Node-RED no está corriendo en el puerto 1880, la capa de presentación muestra un mensaje al usuario indicando que debe estar logueado y que Node-RED debe estar ejecutándose en el puerto 1880 para poder acceder al listado de mashups.

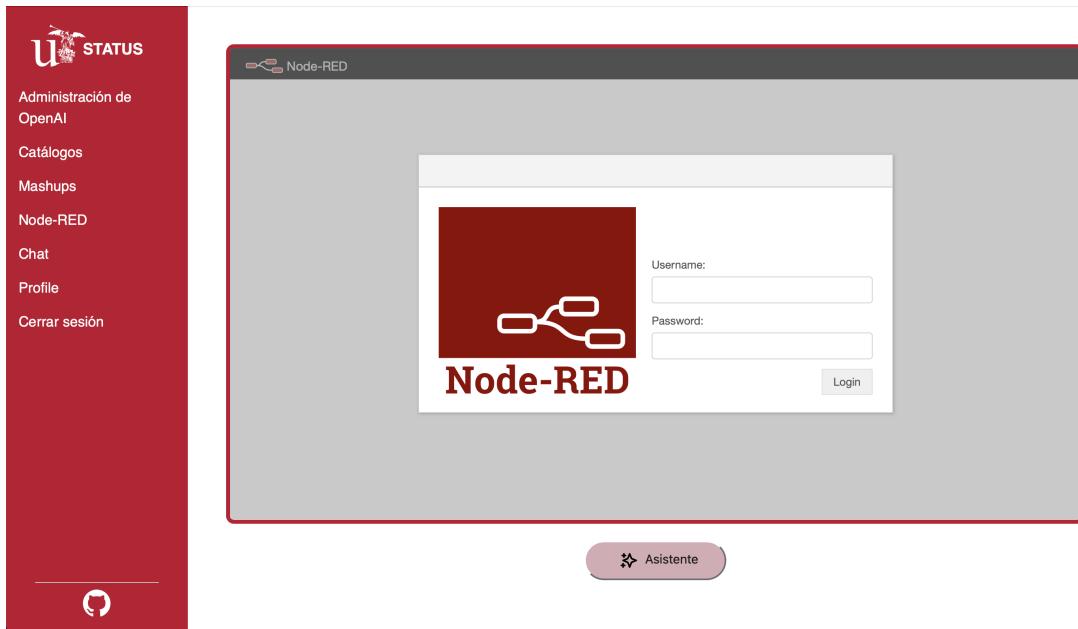


Figura 10.16: Inicio de sesión en el editor

Por último, en la Figura 10.16 se muestra la vista de “Node-RED” en el sistema, en la que se accede al editor de mashups. Además, como se mencionó en el apartado 9.3.2, para poder acceder al editor es necesario introducir las credenciales de autorización.

10.5. Interfaz y Experiencia de Usuario

En la era digital, la experiencia del usuario (UX) juega un papel fundamental en la percepción y la eficacia de cualquier aplicación de software. Una interfaz de usuario (UI) bien diseñada no solo facilita la interacción entre el usuario y el sistema, sino que también influye en la retención de usuarios y la satisfacción del cliente. Dentro de este contexto, este apartado se centra en tres elementos clave para mejorar la experiencia del usuario en un entorno de software: Skeleton Loaders, Mejoras en el Estilo del Chat y Animación de Elementos. Cada uno de estos aspectos contribuye significativamente a la percepción y la usabilidad del sistema, proporcionando una experiencia fluida y atractiva para el usuario final. A continuación, se mencionan algunos de las mejoras en la experiencia de usuario realizadas:

- **Feedback visual:** Proporcionar retroalimentación visual inmediata cuando el usuario interactúa con los elementos de la interfaz. Por ejemplo, cambiar la forma del cursor al pasar sobre un botón, resaltar un elemento seleccionado (ver Figura 10.17) o aumentar ligeramente su tamaño .
- **Transiciones suaves:** Utilizar transiciones suaves y animaciones para guiar al usuario a través de la aplicación de manera fluida y natural, evitando cambios bruscos o repentinos que puedan causar confusión. Por ejemplo, mensajes temporales que se muestran durante el tiempo de respuesta del asistente (ver Figura 10.17).
- **Consistencia en la Navegación:** Diseñar la aplicación teniendo en cuenta las necesidades

de todos los usuarios es crucial. Garantizar que la misma vista sea accesible desde distintas partes de la aplicación mejora la experiencia del usuario, facilitando la navegación y brindando coherencia en el flujo de interacción. Por ejemplo, poder acceder al editor de mashups para modificar uno desde el listado de mashups.

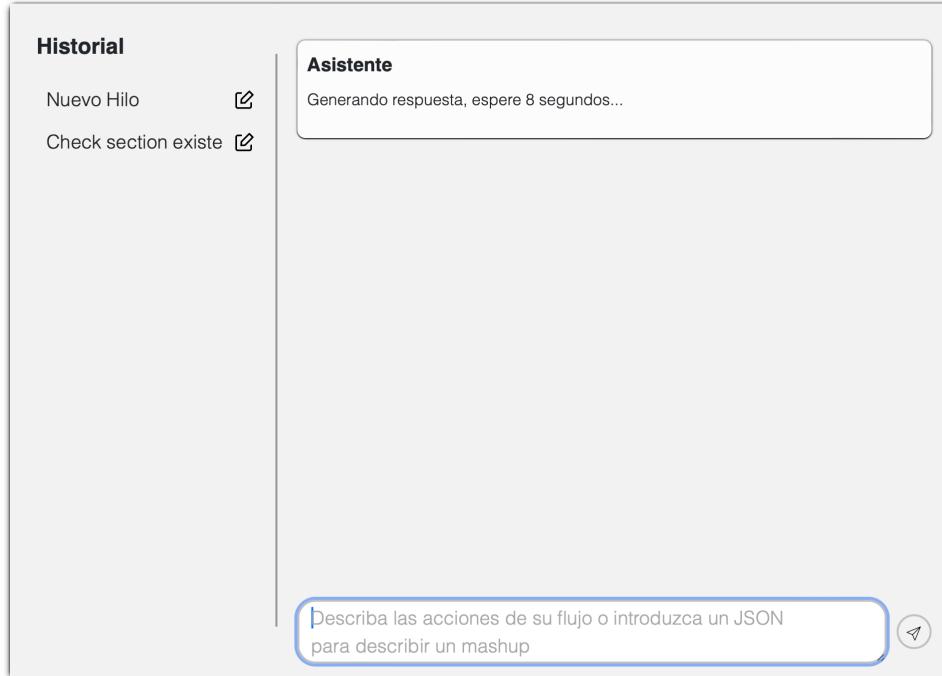


Figura 10.17: *Skeleton loader* de mensajes del chatbot

Los Skeleton Loaders, conocidos también como esqueletos de carga, representan una técnica de interfaz que permite mostrar una estructura visual básica de la página mientras se carga el contenido completo. Este enfoque ayuda a reducir la percepción de tiempo de espera, ofreciendo al usuario una indicación clara de que el sistema está procesando su solicitud. En la Figura 10.17 se muestra uno de los esqueletos de carga introducidos en el proyecto, este esqueleto se encarga de avisar al usuario que se está generando el mensaje del asistente.

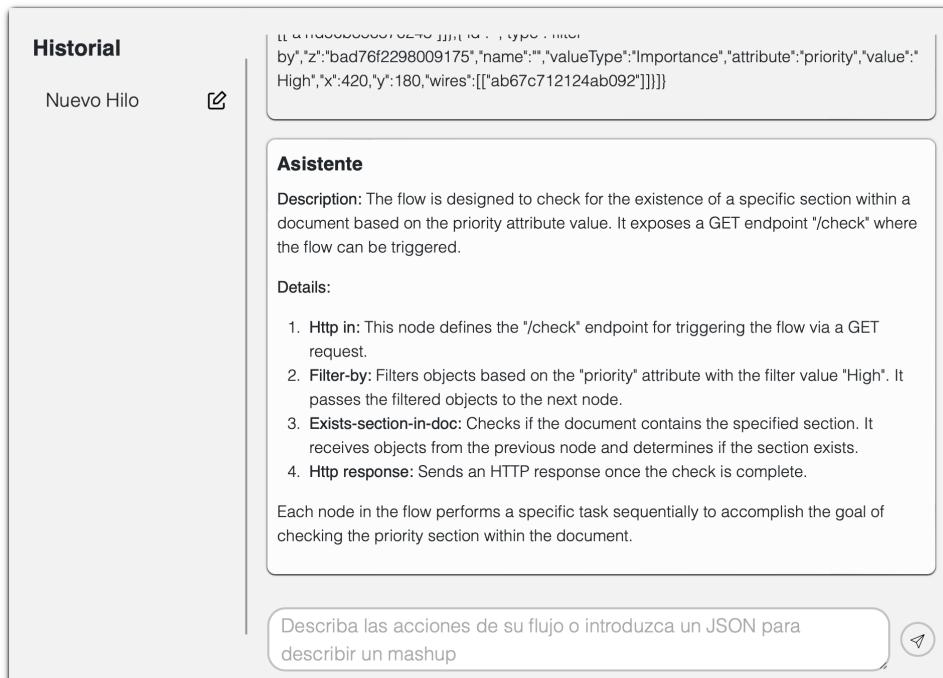


Figura 10.18: Estilo del mensaje en formato Markdown

Además, la mejora del estilo del chat implica la optimización de los elementos visuales y funcionales de la interfaz del chatbot, lo que permite una comunicación más eficiente y atractiva para los usuarios. Una de las formas en las que se ha hecho una mejora de la interfaz ha sido mediante el uso de markdown para mostrar cierta información (ver ejemplo en la Figura 10.18). Para renderizar el markdown se utilizó la librería de React llamada *React-Markdown*⁶. Por último, la incorporación de animaciones en diversos elementos de la interfaz agrega un nivel adicional de dinamismo y feedback visual, mejorando la interactividad y la usabilidad del sistema. Para implementar estas mejoras, se empleó el lenguaje de estilos CSS junto con los *hooks* de React, que se encargan de mantener un estado temporal a la espera de la información final. Un ejemplo de esto puede ser los botones (o también conocidos en inglés como *sliders*) utilizados para la conexión a Trello y/o a Github (ver en la Figura 10.19).



Figura 10.19: Botones de conexión con Trello y Github

Todo esto ha sido un reto a nivel académico (cómo se comenta en el apartado 1.5.2), ya que ha sido el primer contacto con el diseño gráfico mediante un lenguaje de estilos como es CSS. No obstante, por falta de experiencia, el tiempo necesario para completar esta serie de actividades se extendió considerablemente, pasando de la estimación inicial de 8 horas a un total de 15 horas.

⁶<https://github.com/remarkjs/react-markdown>

11. Evaluación y Resultados

Este capítulo aborda la evaluación del sistema y los resultados obtenidos durante su desarrollo. Se analiza la calidad del código, las pruebas realizadas y las contribuciones a otros proyectos, además de una revisión de la planificación y los costes asociados. Los resultados financieros y las conclusiones finales ofrecen una perspectiva completa sobre la efectividad y eficiencia del proyecto.

11.1. Análisis de la Calidad del Código

Uno de los métodos para asegurar la calidad de un software es realizar un análisis de la calidad del código. Para este análisis, se ha hecho uso de la herramienta Codacy¹, que permite analizar repositorios de código de Github. En el cuadro 11.1 podemos ver de forma resumida el análisis de la versión final de este proyecto.

	Front-end	Back-end
Calidad	A	A
Incidencias	2 %	0 %
Duplicaciones	13 %	9 %
Correcciones	57	11

Cuadro 11.1: Calidad de código de Front-end y Back-end

El cuadro presentado evalúa la calidad del código tanto en el Front-end como en el Back-end. En términos de calidad, ambos sectores reciben una calificación de “A”, lo que indica un alto nivel de mantenimiento del código en ambas áreas.

Al analizar los problemas detectados, el Front-end muestra un 2 % de incidencias, mientras que en el Back-end no se registran problemas. Esto sugiere que el código del Back-end es más robusto y libre de errores en comparación con el del Front-end, donde aún existe un pequeño margen de mejora. Cabe destacar que dichas mejoras están relacionadas con el cumplimiento de buenas prácticas de código o cambios en estilos del código como mantener el código espaciado e inteligible (ver la Figura 11.1).

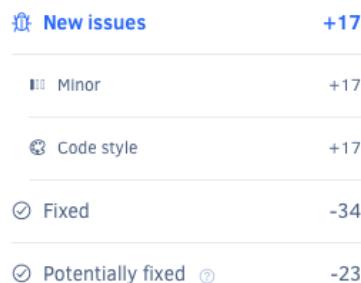


Figura 11.1: Incidencias de código en Front-end

En cuanto a las duplicaciones, el código del Front-end presenta un 13 % de duplicaciones, lo cual es significativamente mayor al 9 % registrado en el Back-end. Esto indica que el código

¹<https://www.codacy.com>

del Front-end podría beneficiarse de refactorizaciones para reducir la redundancia y mejorar la eficiencia. Dichas refactorizaciones se deberían realizar en posteriores iteraciones, aunque debido al alcance del TFG, esto se realizará como parte del proyecto de investigación (quedando fuera de este trabajo).

Por último, se realizaron 57 correcciones en el código del Front-end, mientras que en el Back-end se llevaron a cabo 11 correcciones. Esto indica que, aunque ambas áreas requirieron ajustes y mejoras, el Front-end necesitó un mayor número de intervenciones, debido a tener una mayor complejidad y la presencia de más errores iniciales en esta parte del código. Dicha complejidad viene relacionada con la integración de servicios externos como node-RED y cómo OpenAI.

En conclusión, el análisis de la calidad del código revela que, aunque tanto el Front-end como el Back-end alcanzan altos estándares, el Front-end muestra una mayor necesidad de refactorización y correcciones. En general, el uso de herramientas como Codacy ha proporcionado una visión detallada y precisa del estado actual del código, facilitando la identificación de áreas críticas para el desarrollo continuo que se han refactorizado a lo largo del desarrollo.

11.2. Pruebas realizadas

Antes de comenzar con la implementación de las pruebas, un paso primordial fue realizar un análisis de las herramientas y librerías que se podían utilizar para cada una de estas. Entre ellas, las opciones más llamativas eran:

- **Jest²:** Jest es una herramienta de testing para JavaScript que ofrece una integración perfecta con proyectos de React, facilitando la escritura y ejecución de pruebas unitarias y de snapshot. Sin embargo, su rendimiento puede verse afectado en proyectos muy complejos debido a su enfoque en la simplicidad y facilidad de uso.
- **Mocha³:** Mocha es un framework de testing flexible para JavaScript que permite personalizar y configurar pruebas según las necesidades del proyecto, soportando diferentes bibliotecas de aserción y estilos de pruebas. No obstante, su flexibilidad puede ser una desventaja para los principiantes, ya que requiere una configuración inicial más compleja en comparación con herramientas más integradas como Jest.
- **Postman⁴:** Postman es una herramienta potente para probar y documentar APIs, con una interfaz de usuario intuitiva que facilita la creación y ejecución de pruebas de integración y de endpoint. Sin embargo, su enfoque en las pruebas de API puede limitar su utilidad para proyectos que requieren pruebas de frontend complejas.

Debido a la experiencia previa con la herramienta y su facilidad de uso, se escogió Postman como herramienta para la implementación de las pruebas. Además, cabe destacar que se han implementado las pruebas utilizando la estrategia “*happy path*”, priorizando casos de uso ideales para validar el funcionamiento óptimo del sistema (con el objetivo de probar el núcleo principal del sistema).

11.2.1. Pruebas de API

Inicialmente, esta actividad estaba orientada a la implementación de pruebas unitarias sencillas; sin embargo, durante la implementación de estas pruebas se descubrió una forma de comuni-

²<https://jestjs.io/es-ES/>

³<https://jestjs.io/es-ES/>

⁴<https://www.postman.com>

car las respuestas de las pruebas de forma sencilla, realizando así algunas pruebas de integración. Por ejemplo, en lugar de realizar una prueba para la creación de un asistente y otra prueba separada para la eliminación de un asistente diferente, las pruebas se diseñaron para comunicarse entre sí. De esta manera, una prueba creaba un asistente y otra prueba eliminaba ese mismo asistente, validando así ambos procesos de manera integrada.

Cabe destacar, que esta actividad requirió ser actualizada una vez finalizada la implementación de otros módulos como el de administración de OpenAI. En base a las pruebas realizadas en el Sprint 3 y la actualización realizada durante la implementación de la gestión del chatbot, se obtuvo el informe resumido que aparece en la Figura 11.2.

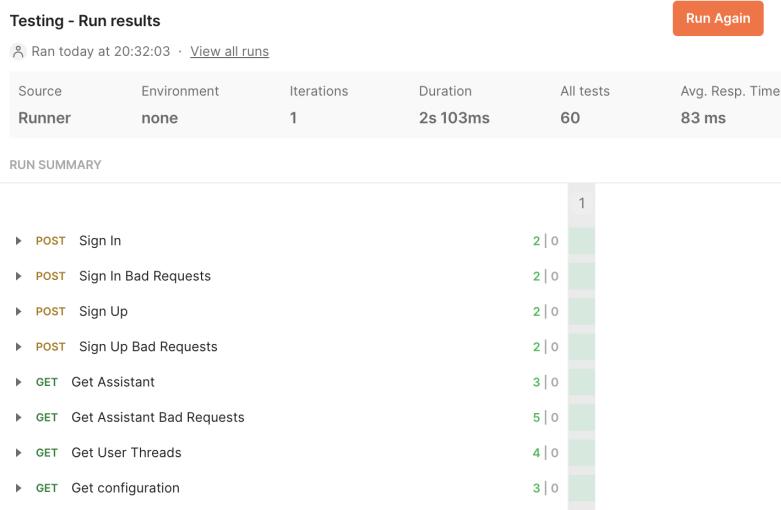


Figura 11.2: Informe de pruebas de API de Postman

Este informe resumido generado por Postman incluye el número total de pruebas unitarias (60 pruebas de la API del sistema), además de una lista con los diferentes conjuntos de pruebas unitarias que se han realizado junto con el número de pruebas de ese conjunto. Esta colección de pruebas podrá ser descargada en la documentación online y configurada siguiendo el manual de pruebas que aparece en los anexos (apartado 13.4).

▶ GET Get assistant by ID	4 0
▶ GET Get assistant limit	5 0
▶ GET Get assistant instructions	4 0
▶ POST Create Thread	5 0
▶ PUT Update Config	3 0
▼ GET Refresh token	4 0
Pass Response status code is 200	
Pass Response has the required fields	
Pass Access token is in a valid format	
Pass Response time is less than 500ms	
▶ POST Create New Assistant	3 0
▶ DELETE Delete assistant By Id	3 0

Figura 11.3: Listado de *suites de tests*

En la Figura 11.3 se muestran algunos de los tests unitarios que se han realizado. Además, aparece el detalle de *Refresh token* en el que se puede ver algunos de las pruebas unitarias que se realizan cómo comprobar el código de estado devuelto, comprobar que la respuesta tiene los

campos requeridos, comprobar que el token tiene el formato correcto y una prueba que comprueba el tiempo de respuesta de la petición.

11.2.2. Pruebas de Node-RED

En base al funcionamiento de Node-RED y a su integración en el sistema STATUS, esta herramienta puede ser probada de dos formas diferentes:

- Realizar pruebas de la API de Node-RED para comprobar las llamadas que se realizar a ésta y la correcta configuración de la herramienta en el sistema.
- Realizar pruebas de los nodos personalizados para comprobar el correcto funcionamiento de esta serie de componentes.

Debido al alcance del proyecto y al tiempo requerido para realizar esta serie de pruebas, se ha decidido descartar las pruebas de nodos personalizados. Esta decisión se fundamenta en varios motivos. Primero, el entorno de estas pruebas es muy específico, ya que la complejidad y el tiempo necesarios pueden variar considerablemente según la configuración del nodo personalizado, los parámetros que recibe y el ejemplo utilizado para la prueba. Además, estos nodos se encuentran en su primera versión de desarrollo, lo que implica que cualquier cambio en ellos requeriría una refactorización de las pruebas correspondientes. Por lo tanto, se considera que esta etapa es aún muy temprana para llevar a cabo esta serie de pruebas.

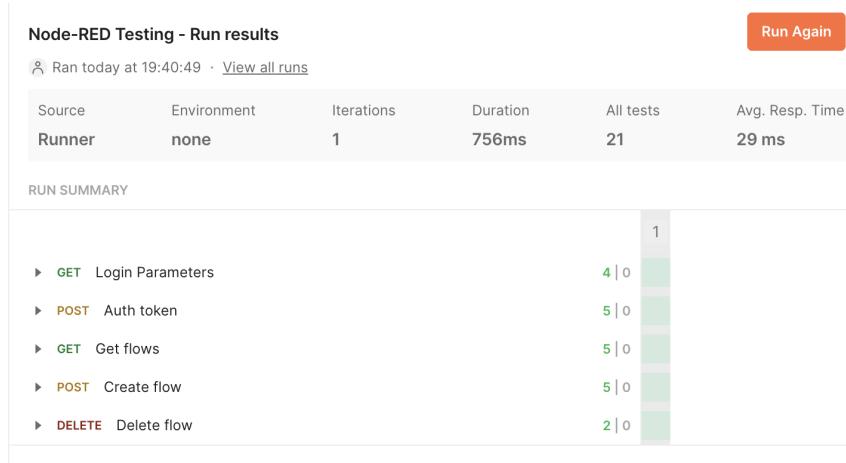


Figura 11.4: Pruebas unitarias de Node-RED

En la Figura 11.4 aparecen algunas de las pruebas unitarias realizadas en Node-RED. En este caso, se ha realizado una prueba de integración encargada de generar un token de la API de administración de Node-RED, crear un flujo aleatorio usando ese token y, posteriormente, eliminar dicho flujo.

11.2.3. *Prompt testing*

Para nuestro sistema, este tipo de pruebas debe adaptarse con el objetivo de medir la calidad de los prompts enviados al chatbot. Para ello, se realizó una prueba de extremo a extremo (*End-to-end*) que se encarga de crear una conversación, enviar un prompt para generar un mashup y comprobar la calidad de la respuesta obtenida.

PASS	Response status code is 200
PASS	Response time is less than 200ms
PASS	Options object must exist and be an object
PASS	Validate the response object in the body
PASS	Check prompt quality > 50%
PASS	Check prompt quality > 67%
FAIL	Check prompt quality > 83% AssertionError: expected 4 to be at least 5
FAIL	Check prompt quality = 100% AssertionError: expected 4 to be at least 6

Figura 11.5: Prueba de prompt mediante Postman

La adaptación presentada en la Figura 11.5 se prueba siguiendo este procedimiento: se crea una conversación utilizando la descripción de un mashup existente (para conocer cuál es la respuesta correcta), se obtiene la respuesta del asistente y se verifica que la respuesta contenga un mínimo de nodos esperados. En el ejemplo de la figura, dado que la respuesta solo contiene cuatro de los seis nodos esperados, las dos últimas verificaciones devuelven falso, resultando en un *Fail*.

11.2.4. Pruebas de carga

Response time trends during the test duration.

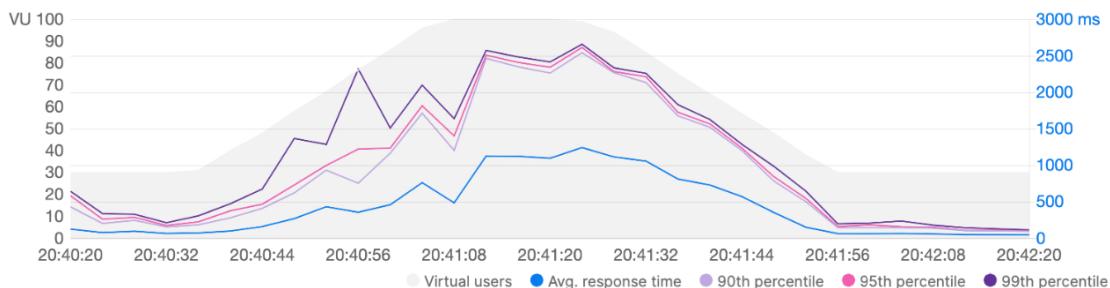


Figura 11.6: Gráfica de rendimiento del sistema

Para medir la eficiencia de la implementación del sistema de caché comentado en el apartado 9.4.3, se realizó posteriormente una prueba de rendimiento. Dicha prueba de rendimiento fue rea- lizada con 100 usuarios ya que, se estima que no accedan muchos usuarios al sistema. Además, se configuró la prueba para que fuese de tipo *peak*, teniendo una base inicial de 30 usuarios y au- mentando dicha base a 100 durante un instante de tiempo. En la Figura 11.6 se muestra un gráfico con el tiempo de respuesta del sistema durante la prueba, es importante mencionar que para poder comparar los resultados de la prueba se utilizó otra petición (en concreto la de inicio de sesión), por eso el tiempo de respuesta es cercano a 2500 en algunos tramos.

Request	Total requests	Requests/s	Min (ms)	Avg (ms)	90th (ms)	Max (ms)	Error %
POST Sign In http://localhost:3001/api/user/signIn	3,373	26.88	93	915	2,229	2,668	0
GET Get configuration http://localhost:3001/api/config	3,331	26.54	2	12	19	471	0

Figura 11.7: Tabla de rendimiento de la caché

Como podemos ver en la Figura 11.7, se han realizado aproximadamente el mismo número de peticiones a ambas rutas; sin embargo, la parte del sistema que utiliza la caché tiene un tiempo medio de respuesta de 12 milisegundos mientras que la ruta sin caché tarda aproximadamente 1 segundo. Este resultado refleja de forma clara el funcionamiento de esta implementación, reduciendo la carga de peticiones a la base de datos.

11.3. Contribuciones a otros proyectos

En este apartado se incluyen las aportaciones realizadas a lo largo del TFG; ya bien a proyectos *Open Source* o a otros proyectos de investigación dentro de la universidad.

11.3.1. Contribuciones a proyectos *Open Source*

Se ha realizado una contribución al proyecto *Open Source* Node-RED. En particular, se ha publicado una colección de nodos personalizados que están disponibles para su uso por parte de la comunidad. Estos nodos están diseñados para ampliar la funcionalidad de Node-RED y facilitar la integración con diferentes servicios y aplicaciones; en este caso, para recopilar información de distintas fuentes de datos y hacer verificaciones de cumplimiento.

En la Figura 11.8 se muestra una imagen de la contribución, la cual ha sido publicada en la página oficial de Node-RED. Esta contribución incluye documentación para ayudar a otros desarrolladores a implementar estos nodos en sus propios proyectos.

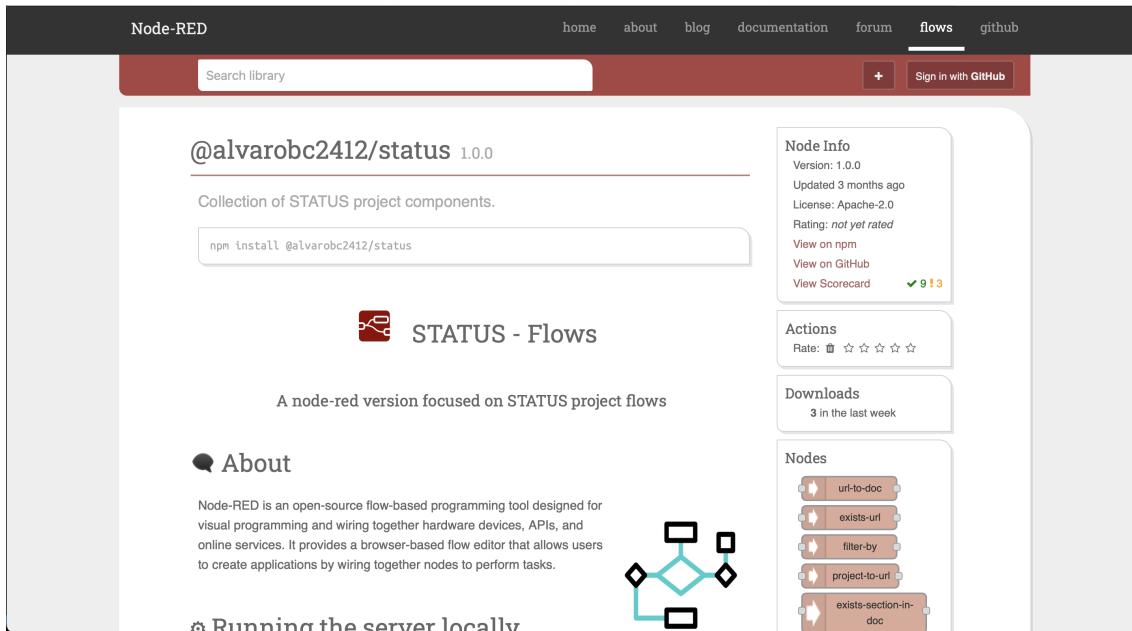


Figura 11.8: Contribución publicada en la página de Node-RED

11.3.2. Contribuciones a proyectos de investigación

Esta pequeña contribución se realizó en base a los problemas encontrados para la configuración del entorno de Bluejay mencionado en el apartado 9.1.2. Cómo se mencionó en dicho apartado, el docker-compose que levantaba la infraestructura no contemplaba arquitecturas de tipo arm64, por ello se realizaron modificaciones el YAML (ver extracto de código 11.1) y se les comunicó al equipo de desarrolladores para que fuese incluido en la documentación.

```

1  bluejay-collector-events:
2    platform: linux/amd64
3    container_name: bluejay-collector-events
4    image: "governify/collector-events:develop"
5    environment:
6      - NODE_ENV=production
7      - PORT=80
8      - KEY_SCOPE_MANAGER=${KEY_SCOPE_MANAGER}
9      - KEY_GITHUB=${KEY_GITHUB}
10     - KEY_PIVOTAL=${KEY_PIVOTAL}
11     - KEY_TRAVIS_PUBLIC=${KEY_TRAVIS_PUBLIC}
12     - KEY_TRAVIS_PRIVATE=${KEY_TRAVIS_PRIVATE}
13     - KEY_HEROKU=${KEY_HEROKU}
14     - KEY_CODECLIMATE=${KEY_CODECLIMATE}
15     - GOV_INFRASTRUCTURE=${GOV_INFRASTRUCTURE:?}
16   ports:
17     - "5500:80"
18   depends_on:
19     - bluejay-assets-manager
20     - bluejay-redis-ec
21   mem_limit: 700m
22   restart: "unless-stopped"
```

Extracto de código 11.1: Modificaciones para arquitectura ARM64

El extracto de código 11.1 muestra uno de los 12 micro-servicios que tiene la infraestructura de Bluejay llamado “Collector-events”. En la línea 2 del extracto, aparece una característica de

docker⁵ que permite emular la arquitectura x86_64, de esta forma se despliega con éxito la imagen sobre una arquitectura de tipo arm64. Esto se realizó para todos los servicios restantes, excepto para mongoDB⁶, la base de datos principal que utiliza Bluejay que en este caso se desplegó de forma separada al resto en una arquitectura arm64. Dichas modificaciones en los servicios de esta infraestructura forman la contribución realizada al proyecto de desarrollo de Bluejay.

11.4. Análisis de la Planificación

En este apartado se presentará un análisis de la planificación del proyecto, donde se mostrarán las tareas realizadas en cada sprint, acompañadas de sus tiempos estimados y reales.

11.4.1. Sprint 1

Actividad	Tiempo Estimado	Tiempo Real
Introducción a Node-RED	2 h	2 h
Introducción a Governify y Bluejay	6 h	6 h
Introducción al <i>Collector-Events</i> de Bluejay	3 h	3 h
Configuración local de Node-RED	1 h	0,5 h
Configuración de contenedores docker de Bluejay	17 h	25 h
Configuración del <i>Collector-events</i> en local	1 h	0,75 h
Estudio de viabilidad de licencias de Enterprise Architect	1 h	1 h
Recopilar información de los elementos del proyecto PROCEDE	3 h	3 h
Documento de <i>mashups</i> de PROCEDE	6 h	6,5 h
<i>Filter-By</i>	1 h	1 h
<i>Filter-By-Date</i>	1 h	1 h
<i>For-Each-Project</i>	2 h	2,5 h
<i>Url-To-Document</i>	2 h	1,75 h
<i>Project-To-Url</i>	2 h	2 h
<i>Exists-Section-In-Doc</i>	2 h	2 h
Revisión del <i>Sprint</i>	1 h	1 h
Retrospectiva del <i>Sprint</i>	2 h	2,5 h
Lecciones aprendidas	1 h	1 h
Seguimiento de la calidad y recursos	1 h	1 h
Informe de avance del proyecto	2 h	2 h
Total	57 h	65,5 h

Cuadro 11.2: Actividades del Sprint 1

En el cuadro 11.2 se presenta un desglose de las actividades del sprint y el tiempo real que ha requerido completarlas. Como podemos observar, la duración real está relativamente cerca de

⁵<https://docs.docker.com/build/building/multi-platform>

⁶<https://www.mongodb.com>

la estimación inicial, con una diferencia de 12,5 horas adicionales. Este retraso se debió a los problemas de configuración mencionados en el apartado 9.1.2.

11.4.2. Sprint 2

Actividad	Tiempo Estimado	Tiempo Real
Investigar cómo se publican nodos en Node-RED	3 h	2 h
Publicación inicial de los nodos	5 h	14 h
Ejemplos de prueba con los nodos publicados	1 h	1 h
<i>Workflow</i> de publicación en npm	1 h	0,5 h
<i>Workflow</i> de publicación en dockerhub	1 h	0,5 h
Instanciar un repositorio de despliegue local de Node-RED	1 h	0,5 h
Script de auto-configuración local de Node-RED	1 h	1 h
Despliegue del proyecto en Docker	2 h	1,5 h
<i>Endpoints</i> de gestión de usuarios	1 h	1 h
Inicio de Sesión	2 h	2 h
Autenticación de usuarios	2 h	2 h
Renovación de sesión	1 h	1 h
Conexión con Github y Trello	2 h	2,75 h
Inicializar el proyecto en LaTeX	2 h	3 h
Estructura inicial	1 h	2 h
Introducción	1 h	1,5 h
Alcance del proyecto	1 h	1 h
Especificación de Requisitos	5 h	4 h
Realización de los mockups	10 h	9,5 h
Revisión del <i>Sprint</i>	1 h	1 h
Retrospectiva del <i>Sprint</i>	2 h	2 h
Lecciones aprendidas	1 h	1 h
Seguimiento de la calidad y recursos	1 h	1 h
Informe de avance del proyecto	2 h	2 h
Total	50 h	57,75 h

Cuadro 11.3: Actividades del Sprint 2

Durante el segundo sprint hubieron también complicaciones, sobretodo en relación a la publicación inicial de los nodos que provocó la refactorización de los sublujos implementados en el primer sprint (mencionado en el apartado 9.2.1). Aun así, la estimación realizada se aproxima bastante a la real, quedando esta 7,75 horas por encima (ver el cuadro 11.3).

11.4.3. Sprint 3

Actividad	Tiempo Estimado	Tiempo Real
Formación sobre la herramienta del chatbot	15 h	13 h
<i>Endpoints</i> de creación de chats	2 h	2,75 h
Vista del Chatbot	2 h	3 h
Historial de chats	1 h	0,75 h
Descripción de mashup	1 h	1 h
Autogenerar mashups	2 h	1,5 h
Listado de mashups de Node-RED	2 h	3,5 h
Creación de mashups mediante API de Node-RED	1 h	2 h
Eliminación de mashups mediante API de Node-RED	1 h	1 h
Autorización de Node-RED	0,5 h	1 h
Refactorización	3 h	11 h
Pruebas de API	3 h	3 h
Investigar sobre las pruebas de Node-RED	6 h	6 h
Pruebas de Node-RED	4 h	4 h
Pruebas de calidad de <i>prompts</i>	4 h	4 h
Revisión del <i>Sprint</i>	1 h	1 h
Retrospectiva del <i>Sprint</i>	2 h	2 h
Lecciones aprendidas	1 h	1 h
Seguimiento de la calidad y recursos	1 h	1 h
Informe de avance del proyecto	2 h	2 h
Total	48,5 h	63,5 h

Cuadro 11.4: Actividades del Sprint 3

El inicio del sprint 3 fue bastante ajustado a la estimación inicial (ver el cuadro 11.4); sin embargo, el principal motivo del retraso en esta fase fue la refactorización del Back-end para utilizar el ORM Sequelize. Este cambio en la estructura del Back-end no se contempló en la planificación inicial, ya que se identificó la necesidad de la modificación posteriormente, durante el análisis de la arquitectura del sistema, además de seguir la recomendación de los tutores. Por ello, en este caso, el tiempo real se aproxima más a la estimación pesimista realizada en el apartado 6.1 (72,25 horas).

11.4.4. Sprint 4

Actividad	Tiempo Estimado	Tiempo Real
<i>Skeleton loaders</i>	2 h	3 h
Mejorar el estilo del chat	4 h	9 h
Animación de elementos	2 h	3 h
Chat de asistencia a Node-RED	4 h	2 h
Recomendación de descripciones	3 h	2 h
Creación de <i>mashups</i> autogenerados en Node-RED	2 h	3 h
Generar y almacenar "detalle" del <i>mashup</i>	3 h	5 h
Activar/Desactivar chat	2 h	6 h
Limitación del chat	3 h	4 h
Listado de chats	2 h	2 h
Documentación de la API ¹	-	4,5 h
Metodología de trabajo	1 h	1,5 h
EDT y diccionario	4 h	5 h
Estimaciones de Tiempo, Costes y Cronograma	3 h	7 h
Diseño y Arquitectura del Producto	4 h	4 h
Tecnologías y herramientas	3 h	6 h
Desarrollo del producto	4 h	7 h
Resultados	3 h	5 h
Conclusiones	1 h	1 h
Anexos	4 h	7 h
Revisión del <i>Sprint</i>	1 h	1 h
Retrospectiva del <i>Sprint</i>	2 h	2 h
Lecciones aprendidas	1 h	1 h
Seguimiento de la calidad y recursos	1 h	1 h
Informe de avance del proyecto	2 h	2 h
Total	61 h	94 h

¹ Esta tarea no se contempló inicialmente, por lo que no tiene estimación.

Cuadro 11.5: Estimación de tiempo de actividades del Sprint 4

Este sprint es el que presenta la mayor diferencia con respecto a la planificación inicial (ver el cuadro 11.5), con un excedente de 33 horas. Es importante destacar que parte de este retraso se debe a la elaboración de la memoria del proyecto. Personalmente, me considero una persona perfeccionista y me gusta dejar todo muy bien pulido. Por ello, gran parte del tiempo adicional se ha dedicado a mejorar la memoria, incluyendo la creación de diagramas, la búsqueda y recopilación de información relevante, y la consulta de apuntes de años anteriores, entre otros. Además, cabe mencionar que inicialmente no se contempló la tarea de documentar la API, lo cual ha sido un añadido importante que mejora la calidad del proyecto al proporcionar a los desarrolladores una documentación detallada de la API.

11.4.5. Cierre

Actividad	Tiempo Estimado	Tiempo Real
Acciones de cierre	1 h	1 h
Acuerdo de Cierre con el tutor	1 h	1 h
Lecciones aprendidas	1 h	1 h
Presentación del TFG	12 h	14 h
Documentación online del TFG	10 h	8 h
Total	25 h	25 h

Cuadro 11.6: Actividades del Cierre

Es importante mencionar que este apartado se ha completado una vez realizado el cierre del proyecto, con el objetivo de hacer un cálculo adecuado de las horas y, por ende, de los costes del proyecto, dejando este apartado como el último punto del TFG.

Paquete de planificación	Tiempo estimado	Tiempo real
Sprint 1	57 h	65,5 h
Sprint 2	50 h	57,75 h
Sprint 3	48,5 h	63,5 h
Sprint 4	61 h	94 h
Cierre	25 h	25 h
Total	241,5 h	305,75 h

Cuadro 11.7: Tiempo estimado y real por paquete de planificación

Cómo muestra el cuadro 11.7, inicialmente se había estimado un total de 241 horas para completar todas las actividades planificadas en el Trabajo de Fin de Grado (TFG). Sin embargo, al cierre del proyecto, el tiempo total dedicado alcanzó las 305,75 horas. Es importante destacar que, aunque se excedió la estimación inicial, el tiempo total invertido se encuentra dentro del rango exigido para el TFG, que requiere un mínimo de 300 horas. Este ajuste asegura que el proyecto cumple con los requisitos establecidos, garantizando al mismo tiempo una alta calidad en su desarrollo y documentación.

11.5. Coste Total de la Propiedad

En esta sección, se establece el coste total de la propiedad respectivo al proyecto. Para ello, se tiene en cuenta el precio de desarrollo del proyecto, y sus estimaciones de precio de operación.

11.5.1. Costes de capital (CapEx)

Costes de personal

Los costes de personal fueron inicialmente planificados; no obstante, para reflejar con precisión los gastos en función del tiempo real, es necesario llevar a cabo un cálculo actualizado. Por

ello, se presenta el cuadro 11.8, el cual compara los costes estimados y los costes reales, organizados por paquetes de planificación de la Estructura de Desglose del Trabajo (EDT). Es importante mencionar que para el cálculo se sigue utilizando el perfil que se planteó en la sección 6.2.

Paquete de planificación	Coste estimado	Coste real
Sprint 1	1.015,81€	1.188,17€
Sprint 2	870,12€	1.047,58€
Sprint 3	988,63€	1.151,89€
Sprint 4	1.107,35€	1.705,16€
Cierre	453€	453€
Total	4.380,31€	5.545,8€

Cuadro 11.8: Costes estimados y reales por paquete de planificación

Además, es importante tener en cuenta los costes sociales de contracción desde el punto de la empresa al igual que se tuvieron en cuenta en la sección de planificación de costes 6.2. Por ello, en el cuadro 11.9 se muestra el desglose total de personal.

Concepto	Base	Porcentaje	Coste
Contingencias comunes	5.545,8€	23,6 %	1.308,8€
Desempleo	5.545,8€	5,5 %	305,02€
I. profesional	5.545,8€	0,6 %	33,27€
MEI	5.545,8€	0,58 %	32,16€
AT y EP	5.545,8€	1,5 %	83,2€
FOGASA	5.545,8€	0,2 %	11,09€
Total Impuestos	-	-	1.773,54€
Base imponible	-	-	5.545,8€
Coste Total de Personal	-	-	7.319,34€

Cuadro 11.9: Desglose final del coste de personal

Es importante mencionar que en proyectos de desarrollo software se suelen incluir los costes de personal encargado del cumplimiento del Reglamento General de Datos (RGPD) o el coste del servicio de una empresa de auditorías de RGPD, aunque en este caso no aplica por el contexto del proyecto.

Costes de material

Este apartado viene contemplado en la estimación inicial de costes (sección 6.2), por lo que el coste se mantiene siendo el mismo, 66,67€.

Costes de licencias externas

En este proyecto, para la CI/CD, se utiliza *GitHub Actions*, pues además se utiliza GitHub para el desarrollo de este. Hay que tener en cuenta el límite de minutos para la CI/CD. Actualmente GitHub dispone de un plan gratuito⁷ para particulares y para organizaciones que dispone de 2.000

⁷<https://github.com/pricing>

minutos de *Github Actions* si el repositorio es gratuito (como es nuestro caso). En el caso de este proyecto, los *workflows* de github se ejecutan de forma puntual, para publicar actualizaciones de los nodos personalizados, por lo que en la mayoría de los casos no alcanzará el límite. Además, el resto de herramientas utilizadas no requieren de pago de licencia, ya que disponen de versión gratuita.

11.5.2. Costes de Operación (OpEx)

En esta sección, detallamos los costes destinados a mantener el servicio operativo una vez que esté finalizado. Dado que actualmente no el sistema STATUS no dispone de usuarios piloto, se espera un uso mínimo en las etapas iniciales (por parte de los desarrolladores). La infraestructura necesaria para sostener una aplicación con estas demandas es mínima, y es importante destacar que la carga en la aplicación no aumenta con el tiempo. En el caso de que el número de usuarios del sistema aumente en 10 unidades, los costes adicionales serían los siguientes:

Costes de despliegue

Inicialmente el sistema no tiene planeado ser desplegado; no obstante, es interesante realizar un estudio de las alternativas de despliegue y de los costes que esto podría suponer.

- **App Engine de Google Cloud:** El servicio ofrece dos tipos de entornos: estándar y flexible. La diferencia radica en que el entorno flexible es autoescalable ante el aumento de la carga, mientras que el estándar requiere escalado manual. Dado que nuestra aplicación experimentará una carga estable a lo largo del tiempo, se podría optar por el entorno estándar. Los costes de App Engine se determinan según el tipo de instancia desplegada, siendo más costosas aquellas con mayores recursos. Para la carga reducida de nuestra aplicación, las instancias más pequeñas serían suficientes. En este sentido, App Engine ofrece el tipo de instancia “F” para despliegues de aplicaciones frontend, entre las cuales se puede utilizar la instancia “F1”, la de menor potencia de la clase “F”.

Finalmente, el coste por hora para mantener activa y funcionando correctamente una instancia de tipo “F1” es de 0.06 dólares por hora. Sin embargo, App Engine también ofrece un nivel gratuito, donde estos precios no se aplicarán si el uso no supera un límite establecido por Google Cloud. Para las instancias de tipo “F”, este límite es de 28 horas por día, lo que implica que el despliegue y mantenimiento de la aplicación se considerarán gratuitos.

Además, App Engine ofrece otro tipo de instancia, la “B”, diseñada para el backend. Esta instancia presenta distintos subtipos según la velocidad y la capacidad. Por tanto, se puede considerar que el despliegue y mantenimiento de la aplicación serían gratuitos, dado que el uso esperado no excederá el límite establecido por Google Cloud.

En este caso, como se mencionó anteriormente, dada la baja carga de usuarios que recibirá la aplicación, el uso de las instancias más pequeñas será adecuado para el despliegue del backend. La instancia “B2” ofrece un precio de 0.1158€ por hora de funcionamiento. Sin embargo, App Engine proporciona 9 horas gratuitas en esta instancia. Por lo tanto, para mantener el sistema en funcionamiento durante todo el día (24 horas), el precio sería equivalente a las 15 horas restantes multiplicadas por el precio mencionado para cada día de operación.

Por lo tanto, el coste de mantenimiento mensual con la utilización de App Engine se muestra

en la Figura 11.9 (Se ha utilizado la calculadora de AppEngine⁸):

The screenshot shows two separate cost estimates for the same location (Frankfurt) using different instance types:

- Instance Type: F1**: 730 hours per month at EUR 0.00. A note indicates that a portion of the estimate fits within the [App Engine free tier](#).
- Instance Type: B2**: 1,460 hours per month at EUR 66.45.

Total Estimated Cost: EUR 66.45 per 1 month

Estimate Currency: EUR - Euro

Figura 11.9: Tarifa de AppEngine

- **Amazon EC2 de AWS:** Otra opción para el despliegue sería de utilizar Amazon Ec2⁹ de AWS, que además cuenta con planes de precio bajo demanda. En la Figura 11.10 se muestran algunas de las instancias disponibles en la zona para CPUs con 8 núcleos (suficientes para el uso que necesitamos).

Visualizando 33 de 301 instancias disponibles					
Nombre de la instancia	Tarifa por hora bajo demanda	vCPU	Memoria	Almacenamiento	Rendimiento de la red
t4g.2xlarge	0,2944 USD	8	32 GiB	Solo EBS	Hasta 5 gigabits
t3.2xlarge	0,3648 USD	8	32 GiB	Solo EBS	Hasta 5 gigabits
m7a.2xlarge	0,5168 USD	8	32 GiB	Solo EBS	Hasta 12 500 megabits
m7g.2xlarge	0,3636 USD	8	32 GiB	Solo EBS	Hasta 15 gigabits

Figura 11.10: Tarifas de instancias de Amazon EC2

En nuestro caso, la instancia más adecuada sería la t4g.2xlarge, que garantizaría un excelente rendimiento a un precio muy competitivo. Suponemos un uso de 2 horas diarias, resultaría en 60 horas de uso al mes. Finalmente, en base a estas suposiciones el resultado sería:

$$60 \times \$/h 0,2944 = \$17,664$$

⁸<https://cloud.google.com/products/calculator-legacy>

⁹<https://aws.amazon.com/es/ec2>

, lo que equivale a aproximadamente 16,21€ mensuales.

- **Otras opciones:** Además de las opciones planteadas, también cabe la posibilidad de desplegar el Front-end desde plataformas con planes gratuitos (como Vercel, Netlify, etc.) y desplegar el Back-end con alguna de las plataformas mencionadas.

Vercel, Netlify, y similares son plataformas que no proporcionan directamente bases de datos gestionadas como parte de sus servicios principales. Por ello, para el Back-end se puede desplegar en los servicios anteriormente mencionados. Esta combinación permite aprovechar las capacidades de despliegue optimizadas para frontend de Vercel o Netlify, mientras se utilizan soluciones especializadas para la gestión de bases de datos en el backend.

Costes de OpenAI

El coste mensual del servicio de la API de OpenAI¹⁰ se basa en el uso, de esta forma el coste del servicio se vuelve mucho más asequible. Para ilustrar esto, consideremos el modelo gpt-3.5-turbo-0125 (modelo más utilizado para realizar este proyecto). Si se realizan aproximadamente 100 peticiones de 1000 tokens al mes (unos 10 usuarios) utilizando este modelo, el coste se calcula como sigue en el cuadro 11.10.

Concepto	Coste por 1M tokens	Tokens procesados	Coste total
Entrada	\$0.50	100,000	\$0.05
Salida	\$1.50	100,000	\$0.15

Cuadro 11.10: Costes estimados de procesamiento de tokens

Según estos supuestos, el coste total serían aproximadamente 18 céntimos (\$ 0,2). En este cálculo no se ha contemplado el coste de creación de los asistentes ya que este coste ha sido retirado de las tarifas de OpenAI en la última actualización de la API, siendo actualmente gratuito.

Costes del Acuerdo a Nivel de Servicio

Dado que el sistema se encuentra en un estado de pre-producción, aún no se ha definido un Acuerdo de Nivel de Servicio (comúnmente conocido por sus siglas en inglés, SLA). Por ello, se propone uno básico en el cuadro 11.11 que resume el servicio de soporte al cliente, que comprende la resolución de incidencias, solicitudes de usuarios y las solicitudes de cambio (en inglés, Request for Changes).

Tipo de Servicio	Tiempo de Respuesta (TTR)	Cantidad Media (peticiones/mes)
Resolución de Incidencias	1 h	10
Solicitudes de Usuarios	1,5 h	5
RFC (<i>Request for Changes</i>)	4 h	1

Cuadro 11.11: Servicio de soporte al cliente

Sabiendo el tiempo que se va a dedicar a cada tipo de servicio, podemos calcular el coste que va a suponer este soporte. Para realizar el cálculo, mantenemos el precio de 18,14€ para el técnico encargado del soporte, quedando así el cálculo en el cuadro 11.12.

¹⁰<https://openai.com/api/pricing/>

Tipo de Servicio	Horas dedicadas/mes	Coste por hora	Coste total
Resolución de Incidencias	10 h	18,14 €/h	181,14 €
Solicitudes de Usuarios	7,5 h	18,14 €/h	136,05 €
RFC (Request for Changes)	4 h	18,14 €/h	72,56 €
Total			390,01 €

Cuadro 11.12: Costes mensuales del SLA

Costes de Personal

Para mantener el sistema operativo, también es necesaria la contratación de personal que se encargue de asegurar el correcto funcionamiento del sistema. Ya que el número de usuarios será bastante reducido, se considera adecuado contratar a un técnico encargado de esta tarea. Se estima que el técnico dedicará 4 horas semanales a mantener el sistema.

$$18,14\text{€}/\text{h} \times 4 \times 4 = 290,24\text{€}$$

11.5.3. Coste Bienal Total de la Propiedad

Es una práctica común realizar un cálculo bienal del Coste Total de la Propiedad (TCO); por ello, en el cuadro 11.13 se muestra el resultado total de este coste. Para simplificar el cuadro se utilizarán CapEx y OpEx, en lugar de los términos *Capital Expenditures* y *Operational Expenditures*.

Concepto	Tipo	Coste
Costes de personal (desarrollo)	CapEx	7.319,34€
Costes de material	CapEx	66,67€
Costes de licencias externas	CapEx	0€
Costes de despliegue	OpEx	16,21€/mes * 24 meses
Costes de OpenAI	OpEx	0,18€/mes * 24 meses
Costes de SLA	OpEx	390,01€/mes * 24 meses
Costes de Personal (operatividad)	OpEx	290,24€/mes* 24 meses
Total		24.105,38€

Cuadro 11.13: Desglose del TCO bienal

11.6. Coste Final del Proyecto

Utilizando como base los cálculos previamente realizados en la sección 11.5.3, presentamos en el cuadro 11.14 una comparación entre el coste total real de la realización del TFG y el coste total estimado. Los costes reales del servicio externo se han calculado en función a la estimación realizada en la sección 11.5.3, teniendo en cuenta que este se ha utilizado desde su implementación en el tercer sprint (5 meses).

Concepto	Coste Estimado	Coste Real	Diferencia
Costes de personal	5.781,13€	7.319,34€	-1.538,21€
Servicio externo de NLP	1.171,31€	0,9€	+1.170,41€
Costes de amortización de recursos	66,67€	66,67€	0€
Reserva de gestión	876€	508,2€	+367,8€
Subtotal	7.895,11€	7.895,11€	0€
IVA (21 %)	1.657,97€	1.657,97€	0€
Total	9.553,08€	9.553,08€	0€

Cuadro 11.14: Comparación de costes reales y estimados

Como se observa en el cuadro 11.14, se ha mantenido la estimación inicial de costes, gracias en parte al uso de 367,8€ de la reserva de gestión para evitar exceder dicha estimación. Este desfase en el coste se debe al hecho de que los costes reales superaron la estimación inicial. Principalmente, este aumento se atribuye a los costes de personal, que se vieron afectados por el considerable retraso en algunas tareas del proyecto. Por otro lado, el bajo coste del servicio de OpenAI también influyó positivamente en el balance general del proyecto, generando un ahorro de 1170,59€.

12. Conclusiones

Como parte final de este trabajo; se incluye el valor añadido, los errores cometidos, las lecciones aprendidas en base a esos errores y posibles trabajos futuros.

12.1. Valor Añadido

Este apartado se centra en la evaluación de los objetivos establecidos durante la planificación del proyecto y su correlación con los resultados obtenidos. Se examinarán tanto los objetivos planificados inicialmente como aquellos que se lograron alcanzar, destacando los hallazgos significativos.

Al comenzar este Trabajo de Fin de Grado, se establecieron una serie de objetivos académicos y de investigación con el fin de dirigir el desarrollo del proyecto y definir las metas a alcanzar. En relación con los objetivos académicos, se ha logrado cumplir con los propósitos establecidos. Sin embargo, es importante señalar que, a pesar de haber profundizado en el conocimiento sobre la experiencia de usuario e interfaz de usuario, reconozco que este ámbito es tan vasto que mi comprensión sigue siendo limitada. Además, este trabajo no solo me ha permitido aplicar los conocimientos adquiridos durante mi formación, sino que también ha contribuido significativamente a reforzar los conceptos aprendidos. Respecto a los objetivos dentro de la investigación, todos los objetivos propuestos se han cumplido (expuestos a lo largo de la sección 9). A continuación se comentan las principales funcionalidades que aportan valor al proyecto de investigación:

- **Chatbot:** Siendo el núcleo central del proyecto, considero que se ha conseguido realizar un asistente útil, personalizable y asequible económicamente.
- **Aumento de la seguridad:** La securización del sistema participa de forma indirecta en la mejora de la experiencia de usuario, mejorando la calidad del sistema.
- **Documentación del sistema:** Un aspecto a tener en cuenta es que al formar parte este trabajo del proyecto de investigación, toda la documentación de este trabajo que tenga relación con el proyecto, va a servir de utilidad para este.
- **Mejora de la infraestructura:** Se ha realizado una mejora de la configuración local del entorno de STATUS proporcionando a los desarrolladores la posibilidad de utilizar docker para facilitar el despliegue local.

12.2. Errores Cometidos

Uno de los errores más frecuentes que he cometido ha sido la planificación de la formación, siendo esto una tarea muy importante cuando no se tiene experiencia previa sobre el tema que se va a tratar. En múltiples ocasiones, he subestimado la importancia de comprender a fondo las herramientas y tecnologías que se utilizarían en un proyecto específico. Este error ha resultado en la necesidad de buscar documentación sobre herramientas que asumí conocer, solo para descubrir más tarde que mi comprensión era superficial en esos casos (React y CSS principalmente).

12.3. Lecciones Aprendidas

1. **Claridad en los objetivos de las tareas:** Es imprescindible tener una comprensión clara de los objetivos de cada tarea para facilitar su ejecución eficiente y garantizar que estén alineadas con las necesidades y metas del proyecto en su conjunto.
2. **Complejidad en la configuración del entorno:** La configuración de entornos puede ser un desafío significativo, especialmente para aquellos con poca experiencia y cuando la documentación disponible no es lo suficientemente detallada o clara.
3. **Consideración de múltiples enfoques:** Antes de abordar una tarea sin experiencia previa, es crucial evaluar si existen varios enfoques disponibles. Optar por el enfoque respaldado por una mayor cantidad de documentación disponible puede facilitar significativamente la ejecución de la tarea y resolver posibles desafíos.
4. **Mejora de estilos con CSS:** El intento de mejorar los estilos del sistema mediante CSS con conocimientos limitados en este lenguaje resultó en un gasto considerable de tiempo, superando las estimaciones esperadas. Escoger una tecnología menos personalizable pero más sencilla de utilizar es una mejor alternativa en estas situaciones.
5. **Formación previa:** Siempre es positivo realizar una formación sobre las herramientas y tecnologías que se van a utilizar en un proyecto, a pesar de que ya hayan sido usadas previamente (en ese caso, la formación será más breve o orientada a algo específico).

12.4. Trabajo Futuro

Aunque se hayan cumplido con los objetivos planificados, se han identificado algunos aspectos de mejora que podrían ser considerados por los técnicos del proyecto, entre los cuales me incluyo. Es fundamental reconocer que incluso cuando se alcanzan los objetivos establecidos, siempre hay margen para la mejora continua. Este enfoque proactivo nos permite identificar oportunidades para optimizar nuestros procesos y resultados en futuros proyectos.

- **Acciones sobre la descripción de un mashup:** Actualmente, es posible modificar la descripción de un mashup; no obstante, para poder hacerlo es necesario acceder a node-RED y modificar la descripción del flujo desde el editor. En base a esto se identifican las siguientes mejoras:
 - Modificar esta descripción desde el sistema, evitando así la necesidad de acceder a node-RED para realizar este cambio.
 - Añadir la opción de descartar una recomendación de descripción generada automáticamente por el asistente. Actualmente, cuando se genera una descripción automáticamente, esta se implementa de manera directa sin posibilidad de validación por parte del usuario. Al permitir descartar una recomendación, se brinda al usuario un mayor control sobre el contenido generado, permitiéndole decidir si la descripción autogenerada se ajusta a sus necesidades.
- **Implementación de pruebas de los nodos personalizados:** Con el objetivo de mejorar las pruebas realizadas, se podrían realizar mashups sencillos que prueben el menor número de componentes y que mediante un endpoint de Node-RED comprueben que la salida sea la esperada.
- **Asistentes con diferentes roles:** Todas las instrucciones de los asistentes están configuradas inicialmente para llevar a cabo las mismas acciones. Sin embargo, sería beneficioso introdu-

cir diferentes tipos de asistentes, cada uno especializado en realizar tareas específicas. Esta mejora proporcionaría una mayor comprensión y eficiencia en la ejecución de las tareas, al permitir que cada asistente se enfoque en una tarea particular en lugar de realizar múltiples acciones.

Basándonos en esta mejora, podríamos ofrecer al usuario la posibilidad de elegir el tipo de asistente que desean utilizar durante las conversaciones. Una posible implementación consistiría en un menú desplegable en la interfaz de usuario, donde los usuarios puedan seleccionar el tipo de asistente que mejor se adapte a sus necesidades específicas en ese momento. Por ejemplo, podrían elegir entre un asistente experto en la configuración de Node-RED, otro capaz de crear nodos personalizados, uno especializado en crear mashups a partir de descripciones, o uno que los detalle nodo a nodo, entre otros. Además, en la vista de administración, podríamos incluir filtros que permitan a los administradores comprobar el estado y las instrucciones de los asistentes agrupados por tipo. Esto facilitaría la gestión y supervisión de los asistentes, permitiendo una visión más clara y organizada de su funcionamiento. También podríamos considerar la posibilidad de limitar ciertas acciones o funcionalidades de los asistentes según su tipo.

Este proyecto ha sido un viaje lleno de altibajos, una auténtica montaña rusa de aprendizaje, desarrollo y documentación, marcando el cierre de una etapa importante en mi formación académica. Para realizarlo, he tenido que recapitular lo aprendido a lo largo de la titulación y seleccionar los aspectos que aportaban mayor valor, resultando en este trabajo. No se me ocurre una frase mejor para resumir todo esto que:

“La educación es lo que sobrevive cuando lo aprendido ha sido olvidado.”
Burrhus Frederic Skinner

13. Anexos

13.1. Términos, Definiciones y Acrónimos

13.1.1. Términos y Definiciones

- **Document Object Model:** Representación de páginas HTML mediante árboles de etiquetas (Chakrabarti, 2001).
- **Endpoint(API):** Un punto final de API es la ubicación de la API en la que un sistema interactúa con una API web. También es el punto de comunicación entre dos sistemas (Cloudflare, s.f.).
- **Hash:** Un hash es una función criptográfica que transforma una cadena de texto, como una contraseña, en una cadena de texto fija y no reversible. Esto significa que, dado un hash, no es posible recuperar la contraseña original. Hashes son ampliamente utilizados para almacenar contraseñas de manera segura en bases de datos y para verificar la autenticidad de usuarios en aplicaciones (AppMaster, 2023).
- **Hook:** Un hook es una característica de la biblioteca de React que permite administrar el estado y otras funciones sin tener que escribir una clase. Los hooks permiten a los desarrolladores acceder a las características de React, como el estado y el ciclo de vida, directamente en componentes funcionales, sin necesidad de crear clases que extiendan de “React.Component” (React, s.f.).

13.1.2. Acrónimos

API	<i>Application Programming Interface</i>
AWS	<i>Amazon Web Services</i>
BPCMIS	<i>Business Process Compliance Management System</i>
CR	<i>Constraint Requirement</i>
DTCC	<i>Design-Time Compliance Checking</i>
EA	<i>Enterprise Architect</i>
ETSII	Escuela Técnica Superior de Ingeniería Informática
GCP	<i>Google Cloud Platform</i>
GPT	<i>Generative Pre-trained Transformer</i>
HIPAA	<i>Health Insurance Portability and Accountability Act</i>
IoT	<i>Internet of Things</i>
ISO	<i>International Organization for Standardization</i>
JWT	<i>JSON Web Token</i>
MCC	<i>Mashup-based Compliance Checking</i>
NFR	<i>Non-Functional Requirement</i>
NLP	<i>Natural Language Processing</i>
ORM	<i>Object-Relational Mapping</i>
PAC	<i>Product Acceptance Criteria</i>
PMBOK	<i>Project Management Body of Knowledge</i>
PT	Paquete de Trabajo
RGPD	Reglamento General de Protección de Datos
RTCC	<i>Run-Time Compliance Checking</i>
SGBD	Sistema de Gestión de Base de Datos
SLA	<i>Service Level Agreement</i>
TCO	<i>Total Cost of Ownership</i>
UI	<i>User Interface</i>
UNDP	<i>United Nations Development Programme</i>
UX	<i>User Experience</i>

13.2. Manual de Despliegue

Este manual tratará de documentar el despliegue del sistema STATUS mediante contenedores docker.

13.2.1. Pre-requisitos de instalación

Antes de comenzar con la instalación es necesario instalar las siguientes herramientas:

- Docker: Puedes encontrar la guía de instalación para tu sistema operativo aquí.
- Docker-compose: Puedes encontrar la guía de instalación para tu sistema operativo aquí.

13.2.2. Recomendaciones

Es importante mencionar que esta instalación incluye un archivo `.env.deploy` que debe ser completado parcialmente por el usuario. Para utilizar la aplicación correctamente, debes ingresar lo siguiente en este archivo `.env`:

- Clave de API de OpenAI
- ID de Organización de OpenAI
- Clave secreta de cliente de Github

13.2.3. Guía de Instalación para Windows

1. Abre la terminal de Windows.
2. Clona el repositorio de GitHub de infraestructura¹ ejecutando el siguiente comando:

```
git clone https://github.com/statuscompliance/infrastructure
```

3. Encuentra la carpeta infrastructure o ejecuta:

```
cd .\infrastructure\
```

4. Ejecuta el script de configuración:

```
.\setup.bat
```

13.2.4. Guía de Instalación para MacOS/Linux

1. Abre una terminal.
2. Clona el repositorio de GitHub de infrastructure ejecutando el siguiente comando:

```
git clone https://github.com/statuscompliance/infrastructure
```

3. Cambia al directorio recién clonado:

```
cd infrastructure
```

4. Concede permisos de ejecución al script setup.sh:

```
chmod +x setup.sh
```

5. Ejecuta el script de configuración:

```
./setup.sh
```

Si has seguido los pasos correctamente, tendrás desplegado el sistema en la dirección <http://localhost:3000> y la documentación de la API en la dirección <http://localhost:3001/docs>.

¹<https://github.com/statuscompliance/infrastructure>

13.3. Manual de Usuario

Para poder probar el sistema de forma completa, es necesario registrarse, siendo esto algo que no se va a proporcionar a los usuarios (únicamente los administradores del sistema). Por ello, para poder realizar pruebas sobre el sistema se ha añadido un apartado de configuración en la documentación online para poder crear un usuario administrador (Para acceder directamente al apartado pulse aquí).

A continuación, se detallan los recursos desarrollados en el proyecto, junto con instrucciones claras sobre cómo utilizar cada una de las funcionalidades implementadas para facilitar su comprensión.

13.3.1. Autorización

Un usuario sin autenticarse accede al sistema y seleccionar la opción en la barra lateral “Iniciar sesión”. Entonces, carga la vista de inicio de sesión (ver la Figura 13.1) e introduce las credenciales de inicio de sesión.

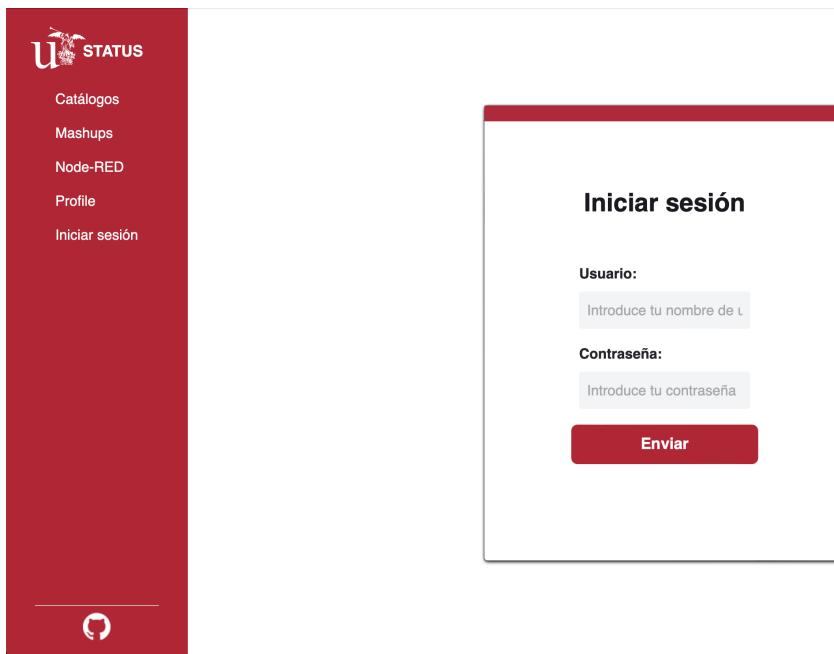


Figura 13.1: Vista de inicio de sesión

En caso de que las credenciales tengan algún error, el sistema muestra al ususario un aviso para que el usuario intente corregir el error (ver la Figura 13.2). En caso contrario, si el usuario accede satisfactoriamente al sistema, le aparecerán las opciones disponibles. El usuario iniciado en esta ocasión tiene el rol de administrador, por lo que tiene acceso a todas las partes del sistema.

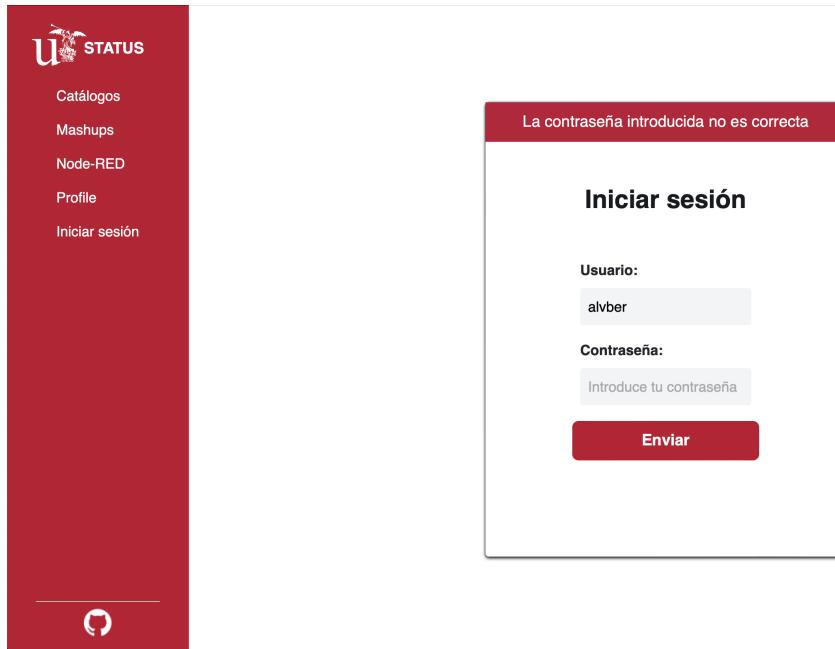


Figura 13.2: Credenciales de usuarios no válidas

13.3.2. Editor de Node-RED

Si el usuario autenticado selecciona en la barra lateral la opción de “Editor”, entonces accederá a la vista de la Figura 13.3 (en el caso de que haya iniciado Node-RED de forma posterior al sistema).

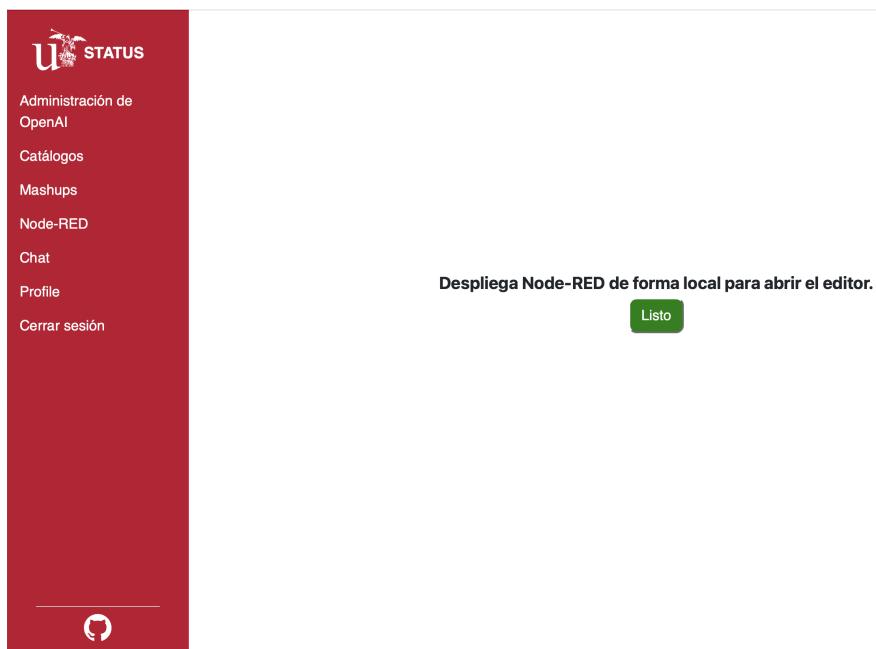


Figura 13.3: Editor no desplegado

Si el usuario tiene desplegado Node-RED y pulsa en listo, se abrirá el modal de la Figura 13.4 para que el usuario inicie sesión en Node-RED.

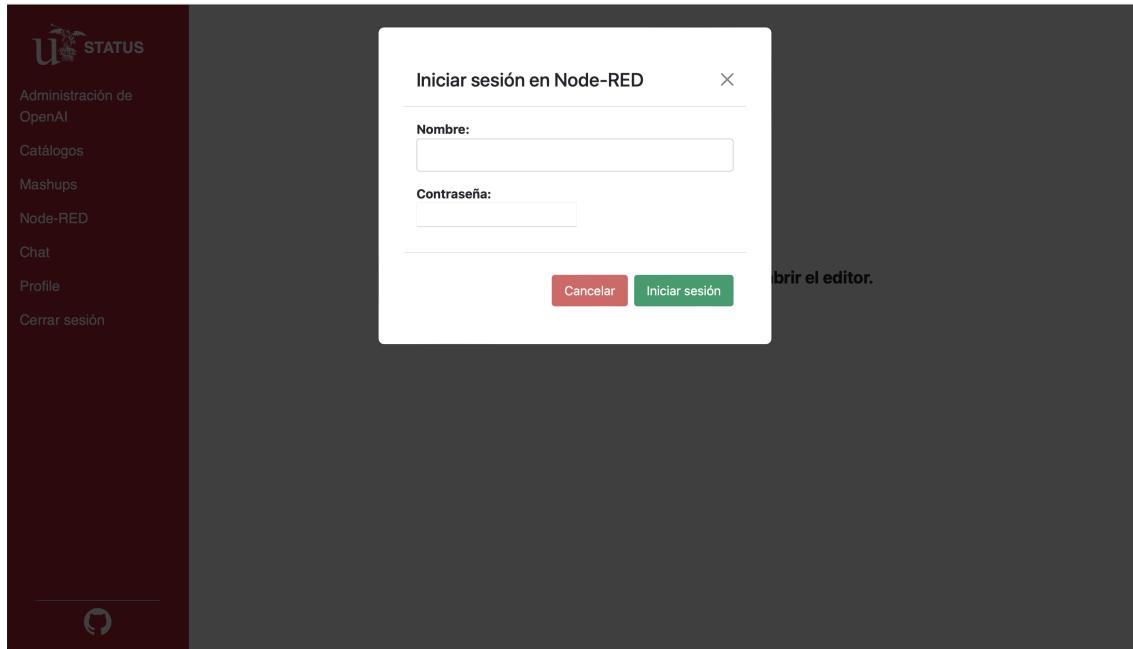


Figura 13.4: Inicio de sesión en Node-RED

Si las credenciales son correctas, se abrirá el editor en esa vista, mostrando el contenido que el usuario tenga almacenado (ver la Figura 13.5).

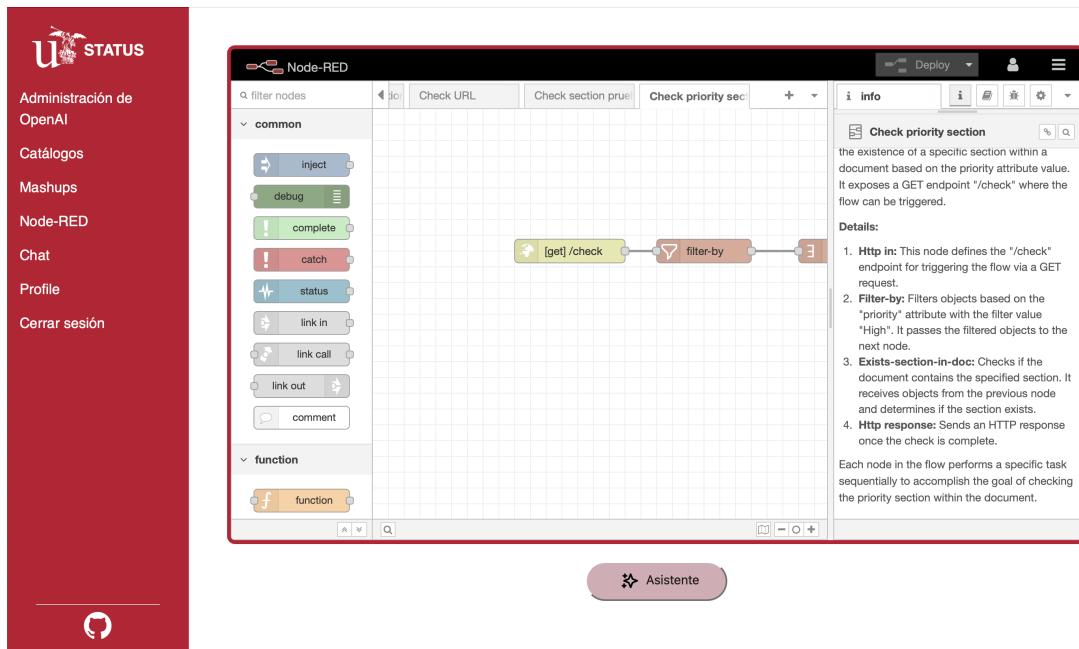


Figura 13.5: Editor de Node-RED

13.3.3. Perfil

Si el usuario autenticado accede a la sección de “Perfil”, el sistema muestra dos botones para conectar el sistema con Github y con Trello (ver la Figura 13.6). Si el usuario pulsa en alguno de los dos botones, se abrirá una vista para autorizar al sistema acceder a la información del usuario.

de esa aplicación. En el caso de Github, una vez concedida la autorización se regresará a la vista de perfil.

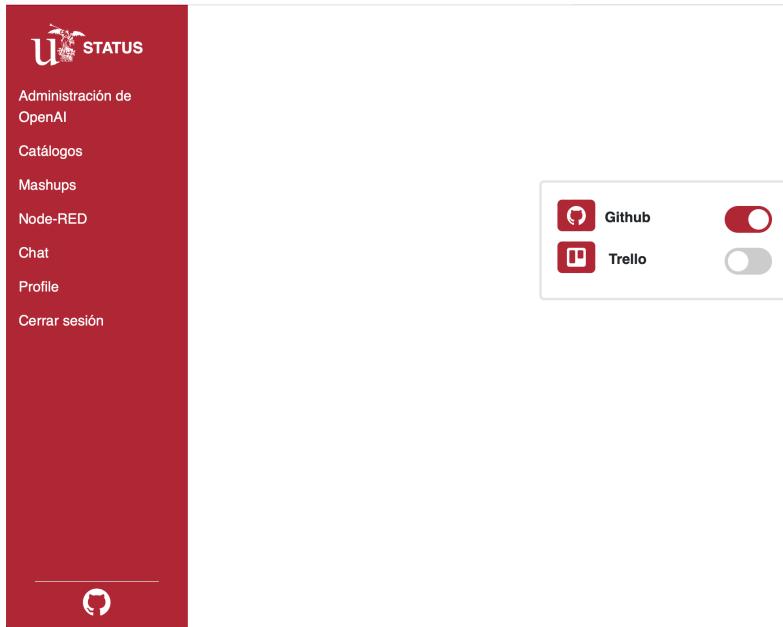


Figura 13.6: Perfil del usuario

En el caso de Trello, el sistema en lugar de hacerlo todo en la misma ventana abrirá una ventana aparte con Trello, donde aparecerá un token que el usuario deberá pegar e introducir en el modal que aparece en la Figura 13.7.

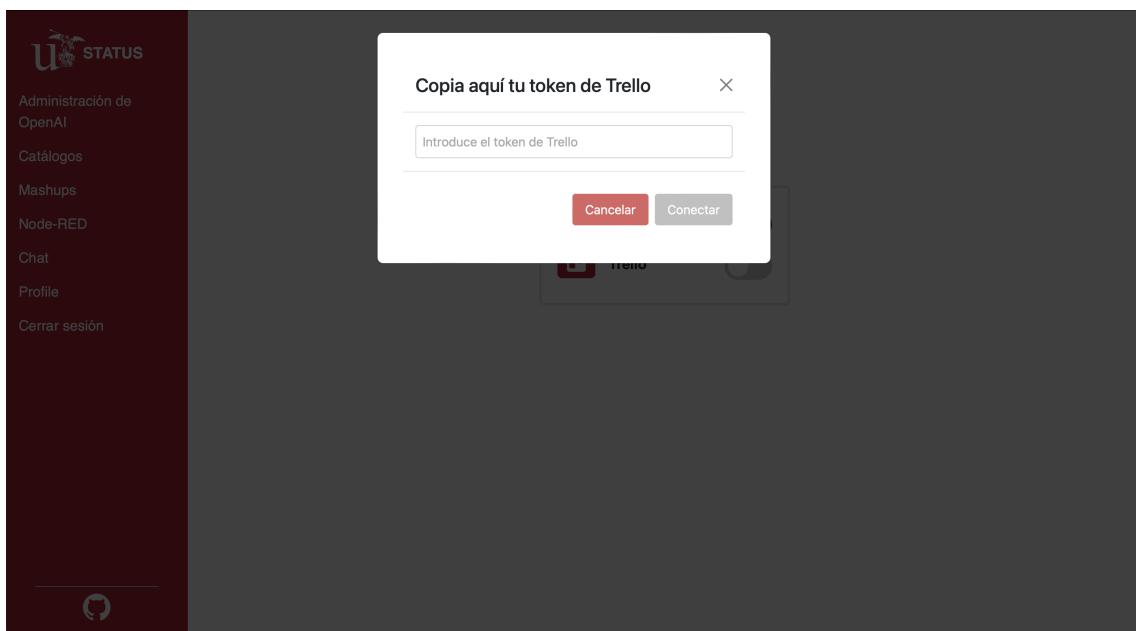


Figura 13.7: Modal de conexión con Trello

13.3.4. Mashups

Si el usuario autenticado accede a la sección de “Mashups”, el sistema muestra el listado de mashups disponibles (ver la Figura 13.8).

The screenshot shows a red sidebar on the left with a logo and navigation links: 'Administración de OpenAI', 'Catálogos', 'Mashups' (which is the active link), 'Node-RED', 'Chat', 'Profile', and 'Cerrar sesión'. The main content area has a search bar and a green '+' button. Below is a table with three rows of mashups:

Nombre	Descripción	Acciones
Check Repo Section	Check if a section exists within the GitHub repository documents associated with cards on a Trello board.	
Check URL	The flow checks whether a specific section exists in a document by sending a GET request to the /test1 endpoint. The flow requires the following parameters in the message body: the GitHub organisation name, the repository name, the file path, the file name, the section it has to contain, and the GitHub token.	
Check section prueba	This flow is designed to check if a specific section exists in a document. It listens for a GET request on the '/check' endpoint. The flow includes a custom node called	

At the bottom are navigation buttons: <<, <, 1, >, >>, 5, and a dropdown menu.

Figura 13.8: Listado de Mashups

Si el usuario pulsa en el botón de añadir mashups (representado con un símbolo +), el sistema muestra un modal para crear un mashup inicial (ver la Figura 13.9).

The screenshot shows the same red sidebar and list of mashups as in Figure 13.8. A modal window titled 'Nuevo Mashup' is open in the center. It contains two input fields: 'Nombre del mashup:' and 'Descripción:', both currently empty. At the bottom are 'Cancelar' and 'Continuar' buttons. The background list of mashups is partially visible.

Figura 13.9: Modal de creación de mashups

Si el usuario pulsa en el botón de detalle del mashup (representado con una letra i), el sistema

muestra un modal con la información del mashup, explicando nodo a nodo el funcionamiento del mashup (ver la Figura 13.10).

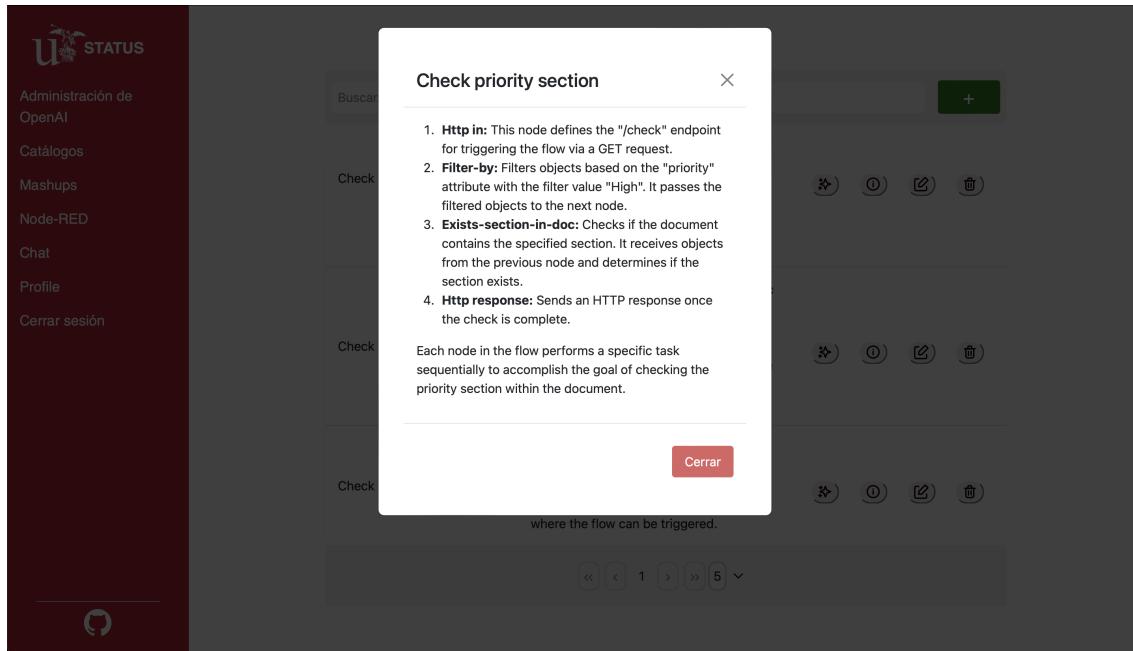


Figura 13.10: Detalles del mashup

Si el usuario pulsa en el botón de eliminar el mashup (representado con una papelera), el sistema muestra un modal de confirmación (ver en la Figura 13.11).

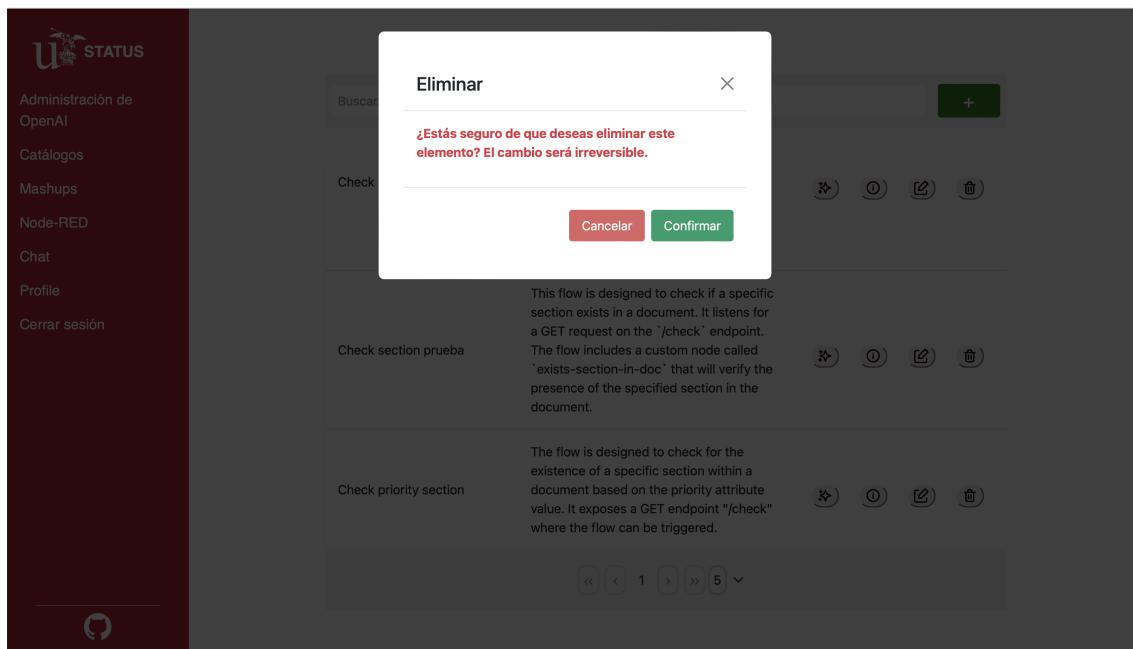


Figura 13.11: Enter Caption

Además de esto, en esta sección el usuario también puede:

- Buscar mediante la barra de búsqueda un nombre o descripción de un mashup.
- Acceder al editor mediante el botón de editar (representado con un lápiz).

- Seleccionar la página, cambiar de página y cambiar el número de elementos que aparecen en la vista.

13.3.5. Chat

Si el usuario autenticado accede a la sección de “Chat”, el sistema muestra un chat con los hilos existentes. En caso de que no existan hilos el sistema muestra una cadena de texto que lo notifica (ver la Figura 13.12).



Figura 13.12: Chat sin conversaciones existentes

Si el usuario envía un mashup de node-red, el chat devuelve una descripción y un detalle del flujo en cuestión de segundos (ver la Figura 13.13).

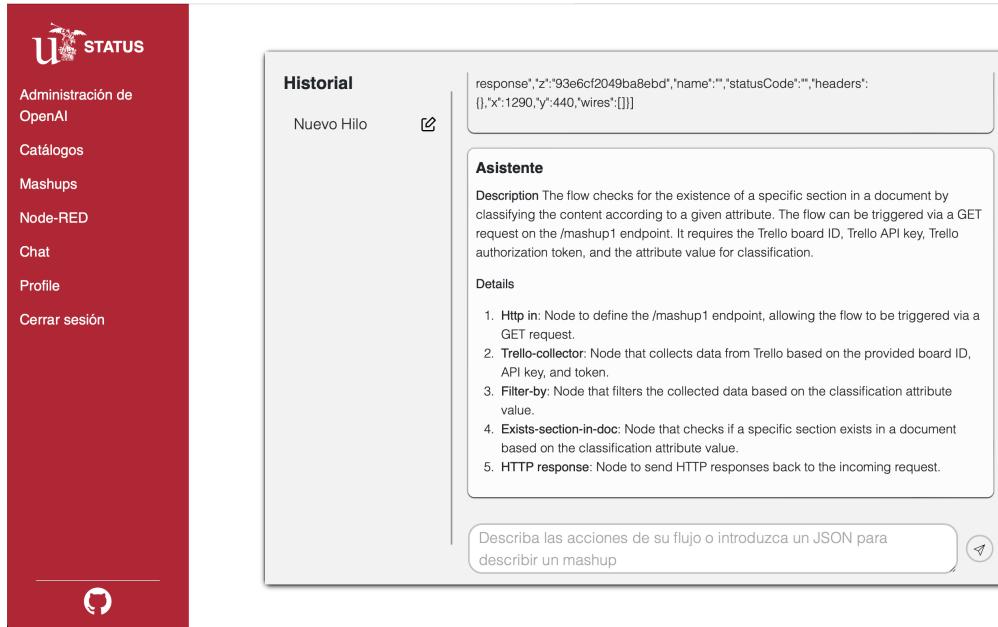


Figura 13.13: Descripción y Detalle de un mashup

El sistema al crear una conversación le asocia el valor por defecto “Nuevo Hilo”, si el usuario quiere, puede pulsar en el botón de editar nombre (representado como un lápiz a la derecha del nombre) y se abrirá un modal para editar el nombre del hilo (ver la Figura 13.14).

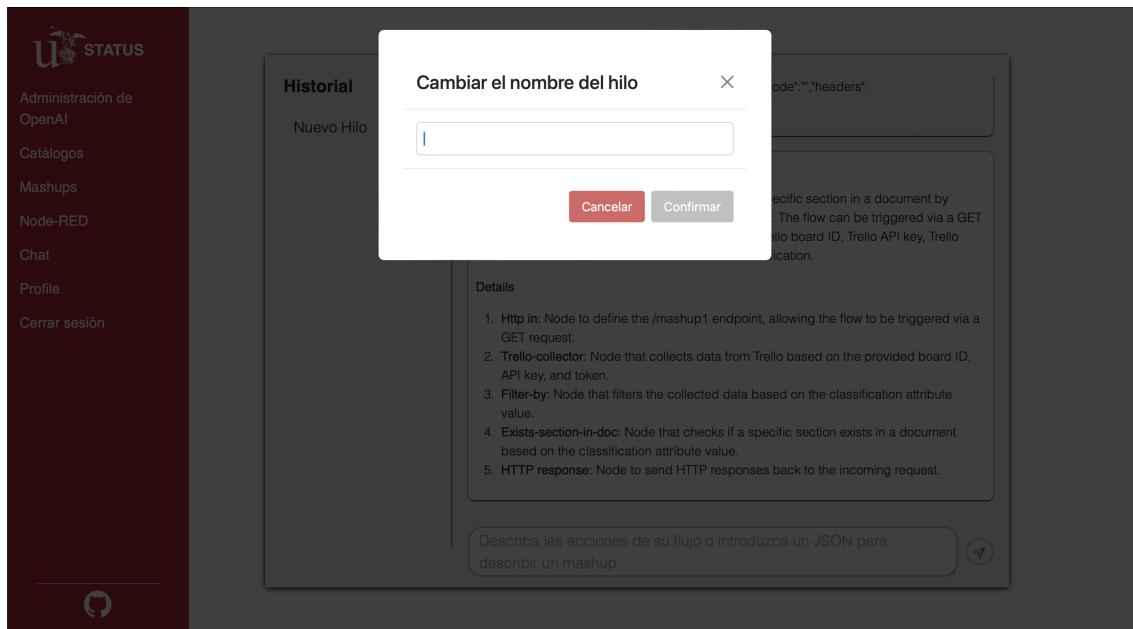


Figura 13.14: Modal de edición de nombre del hilo

Si el usuario envía una descripción demasiado corta, el asistente responde notificando al usuario que debe mejorar la descripción dada (ver la Figura 13.15).

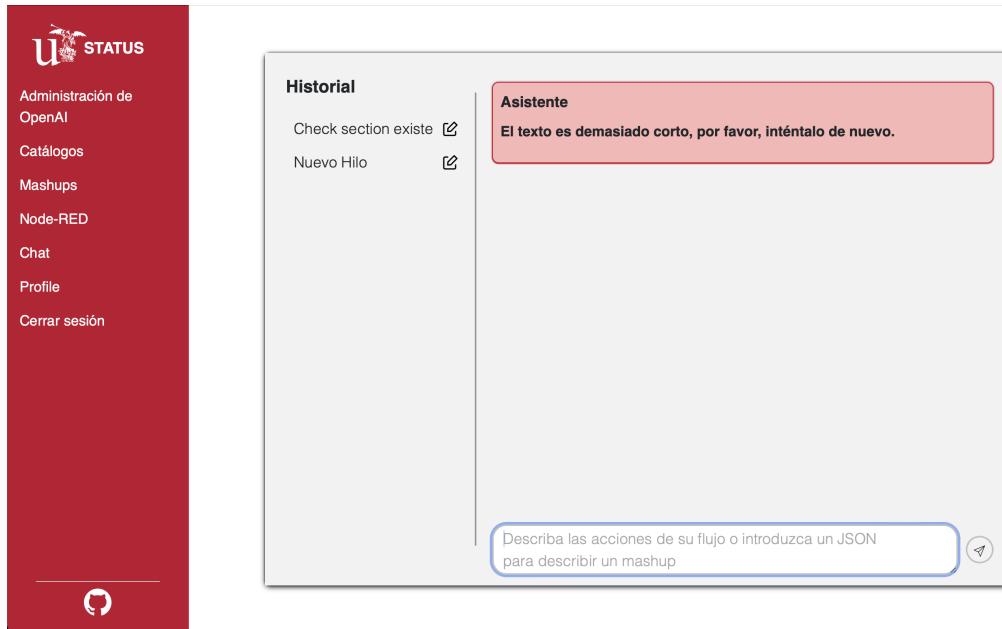


Figura 13.15: Descripción demasiado corta

Si el usuario envía una descripción de un mashup al chat, tras unos segundo se abre un editor distinto con el mashup creado (ver en la Figura 13.16). Además, el usuario puede volver al chat pulsando en el botón de regreso o eliminar el mashup autogenerado.

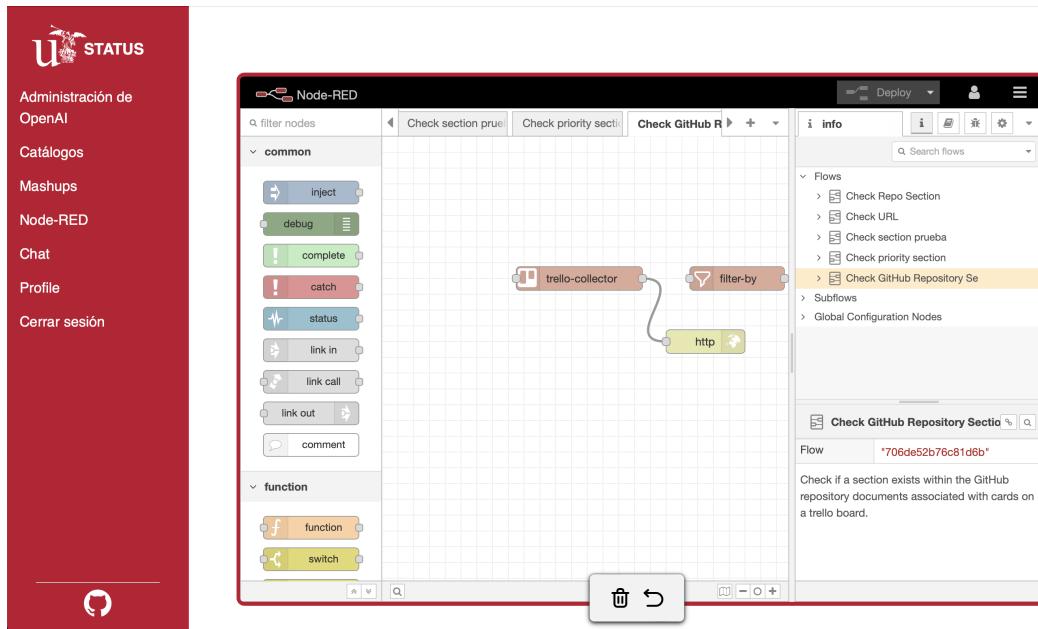


Figura 13.16: Mashup autogenerado

Si el usuario pulsa en eliminar el mashup y confirma esta acción, el editor muestra un mensaje notificando que se han realizado cambios (ver la Figura 13.17).

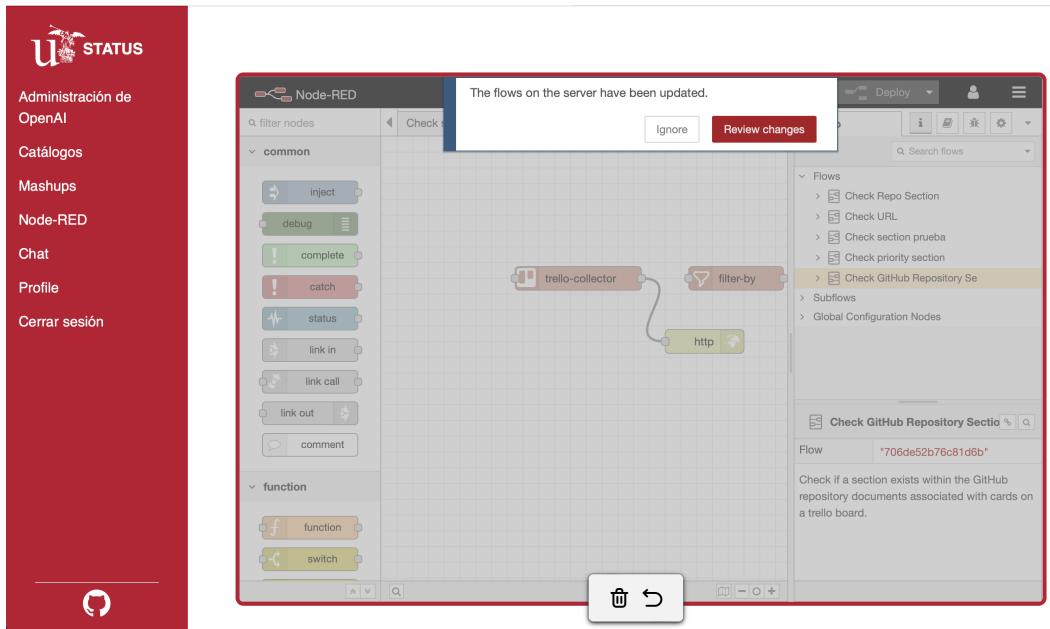


Figura 13.17: Eliminación del mashup autogenerado

13.3.6. Administración

Si el usuario autenticado tiene permisos de administrador, entonces en la barra lateral le aparece la opción “Administración de OpenAI”. Al acceder a esa sección del sistema, se muestran los asistentes existentes además de algunas acciones sobre el servicio de OpenAI como deshabilitar y habilitar el servicio (ver en la Figura 13.18).



Figura 13.18: Vista inicial del administrador de OpenAI

En la Figura 13.19 se muestra la vista de administración una vez el administrador ha seleccionado uno de los asistentes existentes.

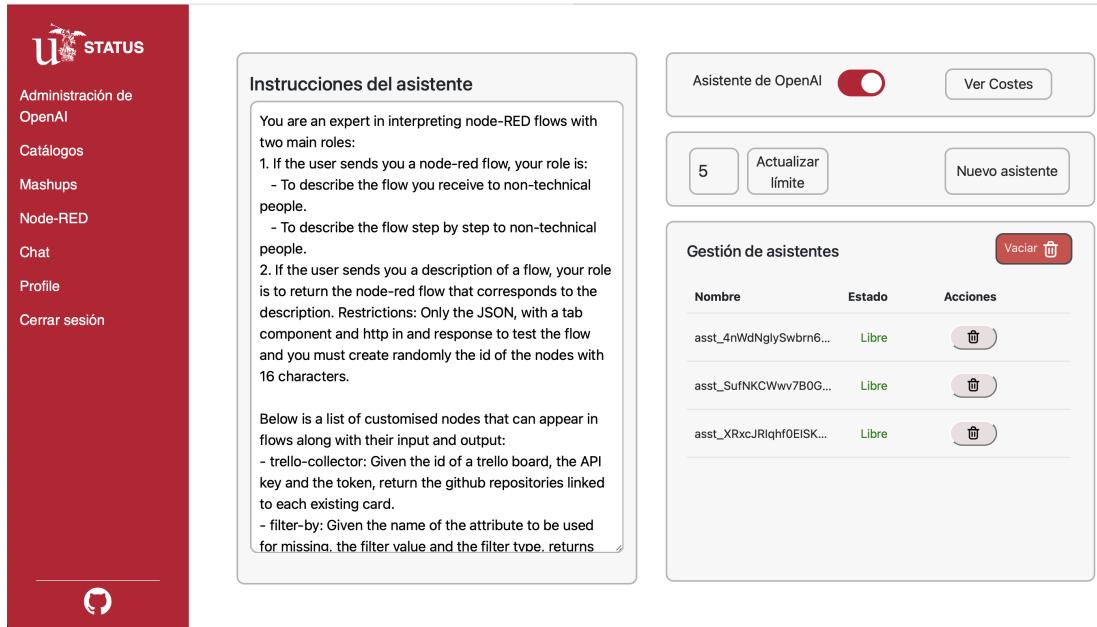


Figura 13.19: Instrucciones de un asistente

Si el usuario selecciona el botón de nuevo asistente, el sistema le muestra un modal para configurar un nuevo asistente con los campos mínimos requeridos (ver la Figura 13.20).

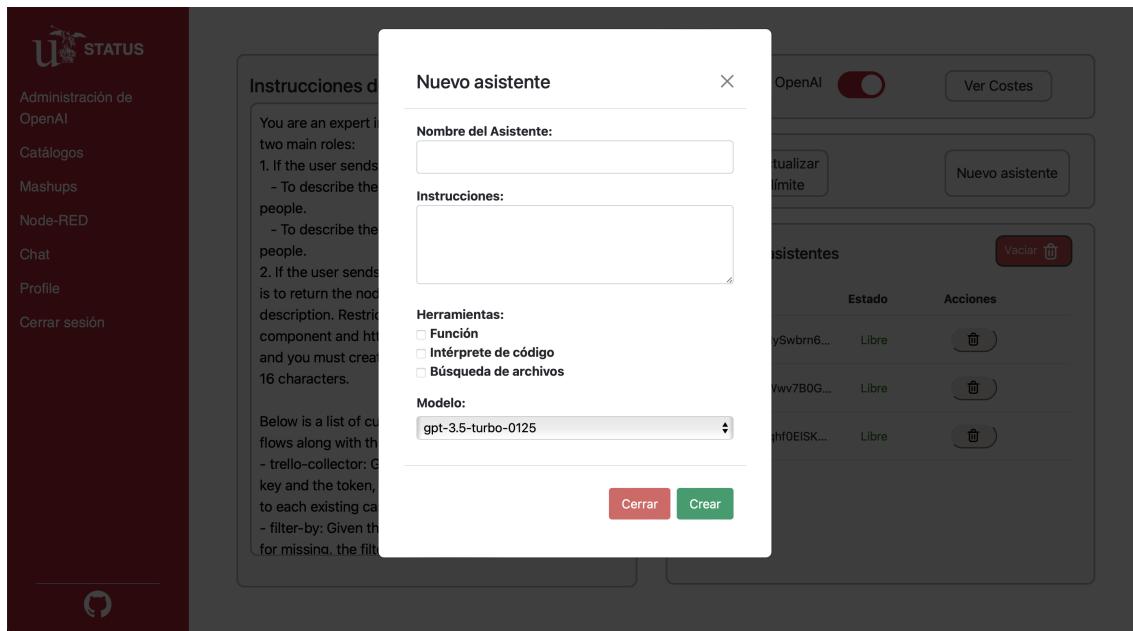


Figura 13.20: Modal de creación de asistentes

Por otro lado, si el administrador deja algún campo vacío, se muestra una alerta notificando que hay algún campo vacío (ver la Figura 13.21).

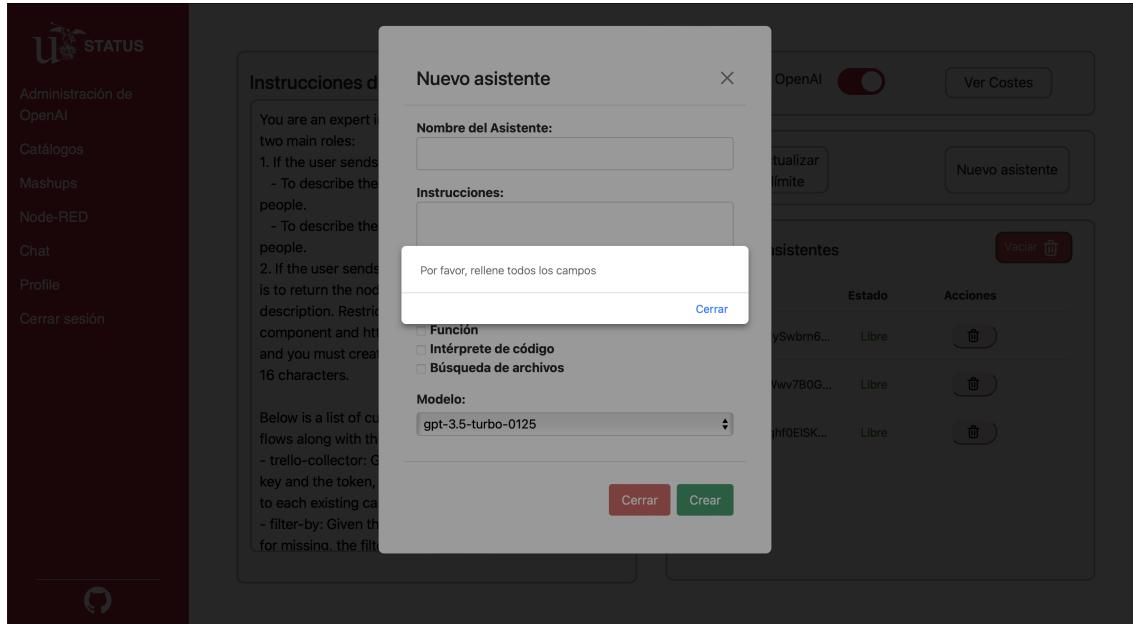


Figura 13.21: Alerta de campos vacíos

Además, el administrador puede eliminar un asistente o todos de forma simultánea mediante el botón de eliminar. Cómo vemos en la Figura 13.22 todas estas acciones requieren confirmación.

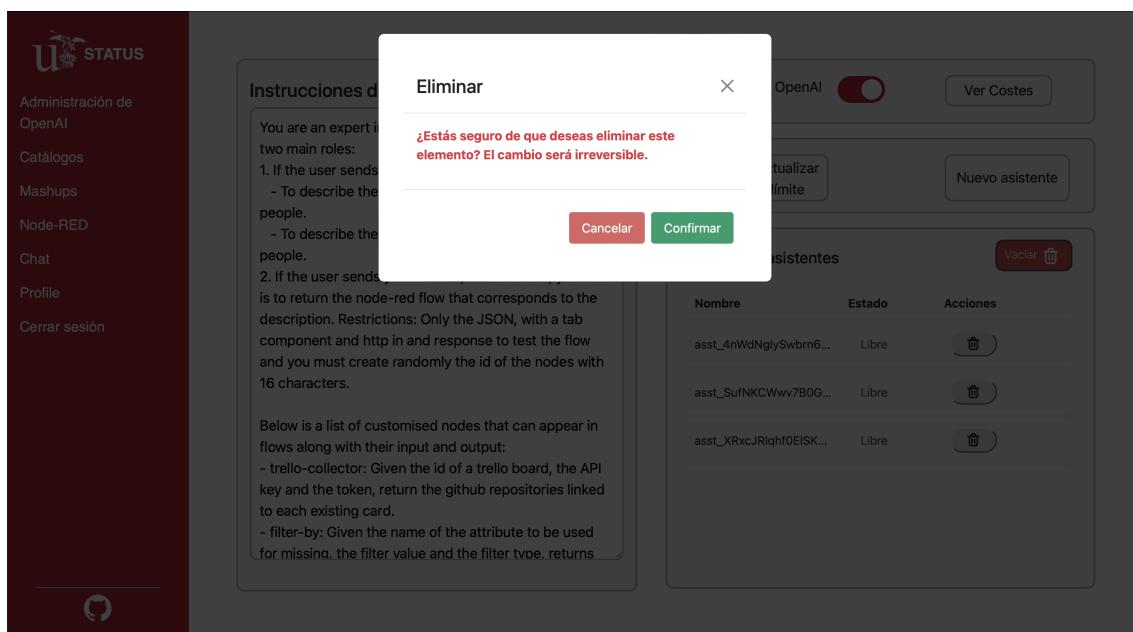


Figura 13.22: Confirmación de eliminación de asistentes

Si el administrador desea editar el límite de asistentes que se pueden crear, entonces este pulsa el botón de actualizar límite, en ese caso el botón se cambia de color y la celda del límite se vuelve editable (ver la Figura 13.23).

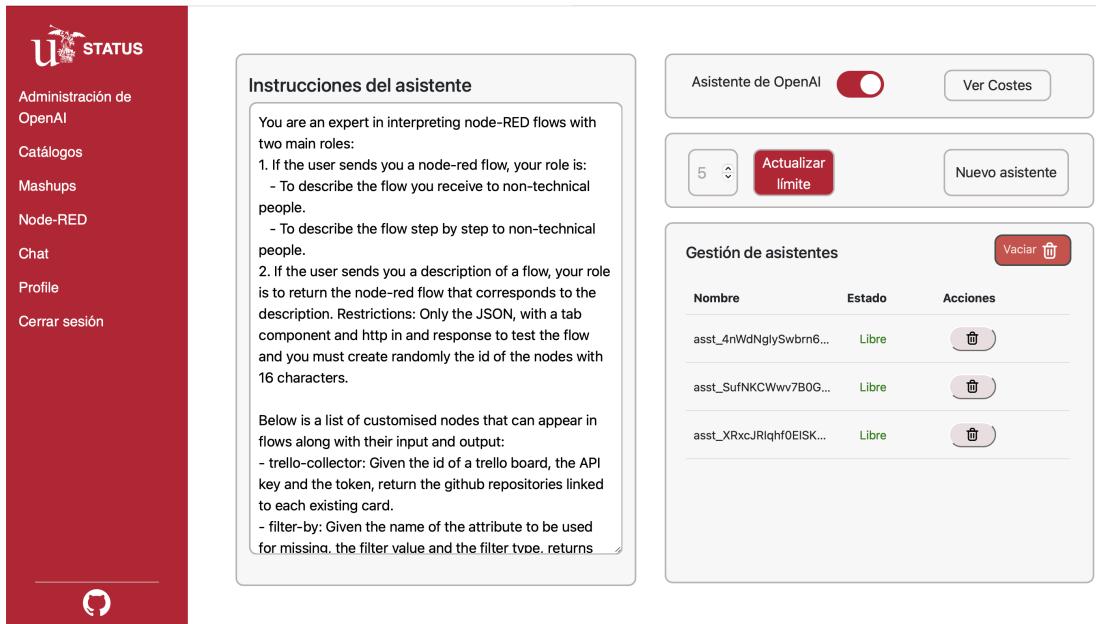


Figura 13.23: Edición del límite de asistentes

Una vez modificado el límite, si el límite introducido es válido, se actualizará el valor de la celda. En caso contrario, se mostrará una alerta con el error en cuestión cómo se muestra en la Figura 13.24.

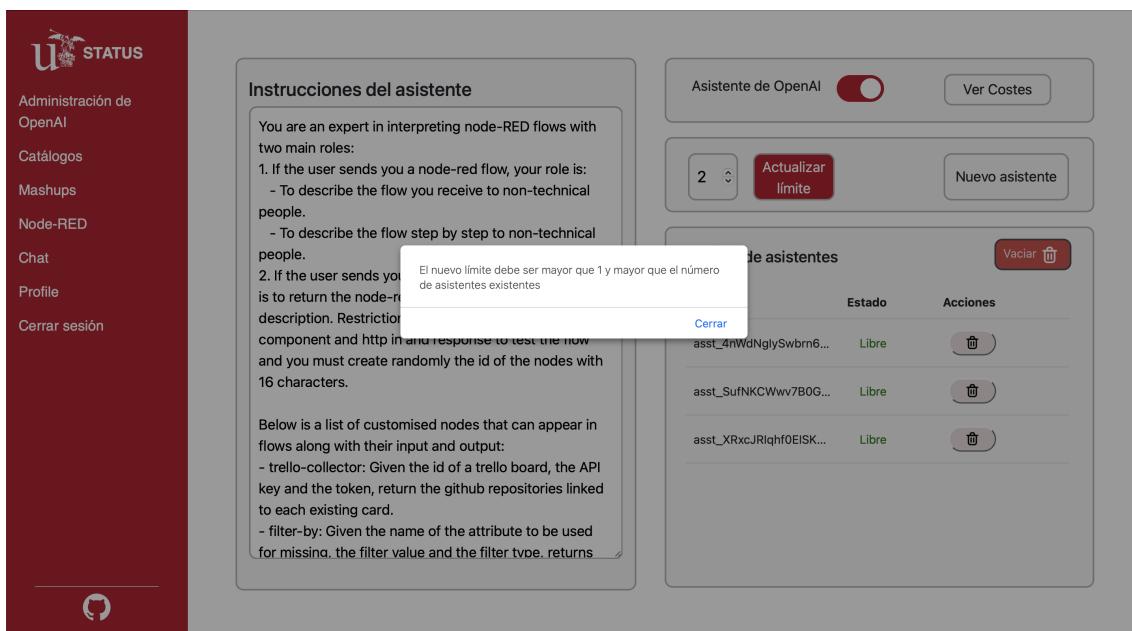


Figura 13.24: Límite inválido

13.4. Manual de Pruebas

13.4.1. Descargar e Instalar Postman

Para descargar Postman, sigue estos pasos:

1. Ve al sitio web oficial de Postman: Haz click aquí.
2. Selecciona la versión apropiada para tu sistema operativo (Windows, macOS, Linux) y haz clic en el botón de descarga.
3. Una vez descargado el archivo, sigue las instrucciones de instalación para tu sistema operativo.

13.4.2. Descargar la Colección de Pruebas

Accede a la documentación online del sistema haciendo click aquí y descarga la colección en cuestión

13.4.3. Importar una Colección de Pruebas en Postman

Una vez que hayas instalado Postman, puedes importar una colección de pruebas siguiendo estos pasos:

1. Abre Postman.
2. Haz clic en el botón **Import** en la esquina superior izquierda.
3. Selecciona **Upload Files** y busca el archivo de colección de pruebas en tu ordenador.
4. Haz clic en **Import** para importar la colección.

13.4.4. Lanzar una Colección de Pruebas en Postman

Una vez que hayas importado la colección de pruebas, puedes lanzarlas siguiendo estos pasos:

1. Selecciona la colección de pruebas en la barra lateral izquierda.
2. Haz clic en el botón **Run** en la esquina superior derecha.
3. Verifica que la colección de pruebas esté seleccionada y haz clic en **Run [nombre-de-la-colección]**.
4. Postman ejecutará las pruebas y te mostrará los resultados.

13.5. Informe de Uso de la IA

Esta sección está dedicada al uso que se ha dado a la inteligencia artificial durante la realización del proyecto, comentando aspectos positivos de su uso y algunas alucinaciones encontradas. Las herramientas que se han utilizado son ChatGPT², Perplexity³, Github Copilot⁴ y Visily.ai⁵.

²<https://chatgpt.com>

³<https://www.perplexity.ai>

⁴<https://github.com/features/copilot>

⁵<https://www.visily.ai>

13.5.1. ChatGPT

Esta herramienta ha sido una de las más utilizada, debido a la multitud de usos que se le ha dado, entre ellos:

- **Brainstorming:** Se ha utilizado como una de las fuentes para obtener ideas que puedan servir de utilidad para la memoria y el proyecto. Además de esto también se utilizaron ejemplos de memorias de años anteriores y las recomendaciones de la plataforma de Trabajos de Fin de Grado. Un ejemplo de uso sería: “Dame 10 posibles apartados de una sección de Tecnologías y herramientas de un TFG de Ingeniería del Software”.
- **Generación inicial de cuadros:** Otro uso ha sido la generación de cuadros útiles para la memoria. Aunque la mayoría de las veces estos cuadros han tenido que ser modificados posteriormente para ajustarlos a un formato adecuado, han sido muy útiles para tener una estructura inicial. En algunos casos fue necesario consultar la documentación de overleaf en busca de cambios más concretos. Un ejemplo de uso sería: “Haz un cuadro en LaTeX que tenga cómo columnas: Actividad, Tiempo estimado y Tiempo real, y filas esta lista de actividades [Lista-de-actividades]”.
- **Extractos de código personalizados:** Siendo daltónico, personalizar los extractos de código para que usen colores similares a los que suelen utilizar en entornos de desarrollo puede llegar a ser una tarea bastante complicada. Por eso, utilizar este chat para poder generar los estilos personalizados para cada lenguaje utilizado ha sido algo muy útil. Un ejemplo de uso sería: “Crea un nuevo estilo de extracto de código para que parezca un dockerfile (yaml) en LaTeX”.
- **Traducción de sh a bat:** Durante la realización de *scripts* de configuración, era importante realizar dichos *scripts* en un formato compatible para distintos sistemas operativos (al menos los más utilizados). Para facilitar la tarea se realizaba el script para el intérprete de comandos zsh y luego se traducía a bat. Un ejemplo de uso sería: “[script] Traduce ese script a .bat”.

13.5.2. Github Copilot

Esta herramienta, además de ser una de las más utilizadas, ha sido empleada de manera recurrente debido a su eficaz funcionamiento. El aspecto positivo más destacable de haberla usado es sus sistema de recomendación de código, que en lenguajes como javascript es bastante acertado. Sin embargo, el uso de esta herramienta requiere de conocimiento previo, en caso contrario no se podría identificar correctamente la validez de las recomendaciones realizadas.

Por otro lado, hubo una tarea específica que se vio perjudicada por un error de esta herramienta: la documentación de la API. A pesar de que se consideraba una tarea sencilla debido a la experiencia previa documentando APIs, una recomendación incorrecta de la herramienta resultó en la omisión de una línea crucial de configuración de la librería en el backend. Esto llevó a que se dedicara una hora a revisar la configuración de la librería, ya que algunas partes de la documentación funcionaban correctamente mientras que otras no.

13.5.3. Perplexity

Esta herramienta ha sido utilizada puntualmente para encontrar referencias que no se encontraban desde Google Académico, esta herramienta funciona de forma similar a chatGPT, lo que ocurre es que ésta permite a los usuarios usar 4 veces al día el modelo de pago, por lo que este incluye las referencias de la respuesta (permitiendo así contrastar la información).

13.5.4. Visily.ai

A diferencia de las anteriores, esta herramienta está diseñada para crear esquemas y prototipos estáticos de cualquier tipo. La parte de inteligencia artificial permite proporcionar bocetos a mano alzada, que la herramienta utiliza para autogenerar la estructura inicial del mockup, facilitando así el trabajo sobre dicha estructura.

13.5.5. Conclusiones

Las herramientas mencionadas han agilizado significativamente la realización de algunas tareas; sin embargo, es importante destacar que su uso suele ser inapropiado, ya que la información proporcionada no siempre se contrasta y se utiliza directamente como si fuera precisa. En mi caso, soy consciente de las posibles alucinaciones de la inteligencia artificial, por lo que he utilizado estas herramientas siendo plenamente consciente de los riesgos que esto conlleva.

13.6. Documentación online del TFG

Para llegar al mayor número de usuarios posible, se ha decidido redactar la documentación en inglés. Sin embargo, una tarea para la prueba de concepto será internacionalizar la documentación para incluir el español. En la Figura 13.25 aparece la página de inicio de la documentación, esta se puede consultar accediendo desde el siguiente enlace:

<https://status-docs.netlify.app>

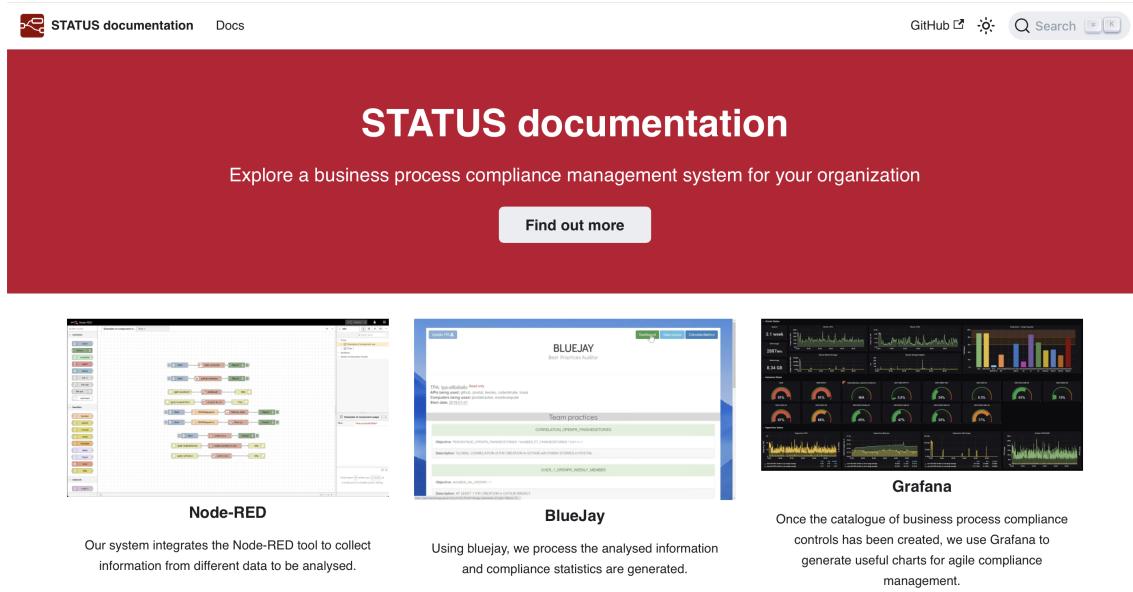


Figura 13.25: Página de inicio de la documentación

14. Referencias

- Aggarwal, S., y cols. (2018). Modern web-development using reactjs. *International Journal of Recent Research Aspects*, 5(1), 133–137.
- AppMaster. (2023). *Bcripta*. Descargado 21/02/2024, de <https://appmaster.io/es/glossary/bcripta-0>
- Asper. (2022). *Desarrollo de chatbots con amazon lex*. Descargado 23/03/2024, de <https://apser.es/chatbots-lenguaje-natural/>
- Blischak, J. D., Davenport, E. R., y Wilson, G. (2016). A quick introduction to version control with git and github. *PLoS computational biology*, 12(1), e1004668.
- Brutti, F. (2024). *Ibm watson assistant: Asistente virtual inteligente para empresas*. Descargado 23/03/2024, de <https://thepower.education/blog/ibm-watson-assistant-asistente-virtual-inteligente-para-empresas>
- Cabanillas, C., Resinas, M., y Ruiz-Cortés, A. (2012). Introducing a mashup-based approach for design-time compliance checking in business processes. En *International conference on advanced information systems engineering* (pp. 337–350).
- Cabanillas, C., Resinas, M., y Ruiz-Cortés, A. (2020). A mashup-based framework for business process compliance checking. *IEEE Transactions on Services Computing*, 15(3), 1564–1577.
- Chakrabarti, S. (2001). Integrating the document object model with hyperlinks for enhanced topic distillation and information extraction. En *Proceedings of the 10th international conference on world wide web* (pp. 211–220).
- Clerissi, D., Leotta, M., Reggio, G., y Ricca, F. (2018). Towards an approach for developing and testing node-red iot systems. En *Proceedings of the 1st acm sigsoft international workshop on ensemble-based software engineering* (pp. 1–8).
- Cloudflare. (s.f.). *¿qué es un punto final de api?* Descargado 26/03/2024, de <https://www.cloudflare.com/es-es/learning/security/api/what-is-api-endpoint/>
- Denis, C. (2019). *Dialogflow: la herramienta de google para la creación de chatbots*. Descargado 23/03/2024, de <https://www.makingscience.es/blog/dialogflow-la-herramienta-de-google-para-la-creacion-de-chatbots/>
- Durán, A., y Bernárdez, B. (2001). Metodología para el análisis de requisitos de sistemas software (versión 2.2). *Universidad de Sevilla, Universidad de Sevilla, España*.
- Dylan Patel, A. A. (2023). *The inference cost of search disruption – large language model cost analysis*. Descargado 21/02/2024, de <https://www.semanalysis.com/p/the-inference-cost-of-search-disruption>
- Díaz, S. (2024). *Crea un chatbot realmente inteligente con la api de openai: Guía práctica*. Descargado 23/03/2024, de <https://openwebinars.net/blog/chatbot-api-openai/>
- Gamez-Diaz, A., Fernandez, P., y Ruiz-Cortés, A. (2019). Governify for apis: Sla-driven ecosystem for api governance. En (p. 1120–1123). New York, NY, USA: Association for Computing Machinery. Descargado de <https://doi.org/10.1145/3338906.3341176> doi: 10.1145/3338906.3341176

- García, C., Guerrero, A., Zeitsoff, J., Korlakunta, S., Fernandez, P., Fox, A., y Ruiz-Cortés, A. (2021). Bluejay: A cross-tooling audit framework for agile software teams. En *2021 ieee/acm 43rd international conference on software engineering: Software engineering education and training (icse-seet)* (pp. 283–288).
- Hahn, E. (2016). *Express in action: Writing, building, and testing node.js applications*. Simon and Schuster.
- Kalla, D., Smith, N., Samaah, F., y Kuraku, S. (2023). Study and analysis of chat gpt and its impact on different fields of study. *International journal of innovative science and research technology*, 8(3).
- Nitin Mehrotra, E. U. (2024). *¿qué es language understanding (luis)?* Descargado 23/03/2024, de <https://learn.microsoft.com/es-es/azure/ai-services/luis/what-is-luis>
- Project Management Institute, Inc. (Ed.). (2017). *La guía de los fundamentos para la dirección de proyectos (guía del pmbok)* (Sexta ed.). Newtown Square, PA: Project Management Institute.
- React. (s.f.). *Presentando hooks*. Descargado 21/02/2024, de <https://es.legacy.reactjs.org/docs/hooks-intro.html>
- Richards, M. (2015). *Software architecture patterns* (Vol. 4). O'Reilly Media, Incorporated 1005 Gravenstein Highway North, Sebastopol, CA
- Santillán, L. A. C., Ginestà, M. G., y Mora, Ó. P. (2014). Bases de datos en mysql. *Universitat oberta de Catalunya*.
- Schwaber, K., y Sutherland, J. (2013). La guía de scrum. *Scrumguides. Org*, 1, 21.
- UNDP. Bureau for Development Policy, UNDP. Evaluation Office, y UNDP. Operations Support Group. (2009). *Handbook on planning, monitoring and evaluating for development results* (1st ed.). New York: UNDP.
- Zamora, E. (2022). *¿cómo se calcula la amortización de equipos informáticos?* Descargado 21/03/2024, de <https://www.holded.com/es/blog/amortizacion-de-equipos-informaticos>