Desarrollo de una herramienta para la gestión de componentes de procesos software

Autor
Carlos Plana García

Fecha
30/06/2022

Directores
Carme Quer Bosor (ESSI)
Xavier Franch Gutiérrez (ESSI)

Titulación
Grado en Ingeniería informática

Especialidad
Ingeniería del Software

Centro
Facultat d'Informàtica de Barcelona

Universidad
Universitat Politécnica de Barcelona

Resumen

Durante este proyecto se ha especificado, diseñado y construido una aplicación web completa para manejar un conjunto de componentes de procesos de software a la que se ha llamado SOPCOM. Esta aplicación web pretende facilitar a los investigadores la definición de estos componentes y sus relaciones según la situación en que se encuentren, por lo que para la gestión de los datos se ha decidido usar la Situational Method Engineering.

Resum

Durant el transcurs del projecte s'ha especificat, dissenyat i construït una aplicació completa per a gestionar un conjunt de components de processos software que s'ha anomenat SOPCOM. Aquesta aplicació web pretén facilitar els investigadors a l'hora de definir aquests components i les seves relacions segons la situació en què s'hagin d'utilitzar, així que per a la gestió d'aquests components s'ha decidit utilitzar la Situational Method Engineering.

Abstract

During the development of this project, a full web application named SOPCOM has been specified, designed and built in order to manage a set of software processes components. This project aims to help the researchers to define these components and its relations taking into account the situation in which it is being used, so to manage these components it was decided to apply Situation Method Engineering.

Agradecimientos

En primer lugar, me gustaría agradecer a mi familia por todo el apoyo que me han brindado durante todos estos años. A mis padres, por animarme siempre a hacer las cosas de la mejor manera posible. A mi hermana, que siempre ha sido un modelo a seguir y sin quien probablemente ahora mismo no estaría escribiendo esta memoria. Y a mis abuelos, que a pesar de no poder compartir este momento con todos ellos, sé que estarán contentos de verme acabar este proyecto.

En segundo lugar, agradecer a Carme Quer, Xavier Franch y Dolors Costal por el soporte que me han dado durante el desarrollo del proyecto, por las reuniones semanales para conocer el estado de la aplicación y por el tiempo dedicado a revisar la memoria.

Por último, también quería agradecer a todos mis amigos por los ánimos que me han dado y por esos ratos de desconexión y risas que de alguna forma también ayudan a sobrellevar mejor los momentos más tensos del trabajo.

Índice de contenidos

Índice de tablas

Índice de figuras

Índice de fragmentos de código

1. Contexto

1.1 Introducción

Este proyecto se ha desarrollado bajo el marco de la asignatura “Treball fi de grau” (TFG) de la Facultat d’Informàtica de Barcelona (FIB). El trabajo es de modalidad A, por lo que se efectúa para el centro mencionado anteriormente, específicamente se ha desarrollado para el grupo de investigación “Group of Software and Service Engineering” (GESSI) [1], el cual forma parte del departamento de Ingeniería de Servicios y Sistemas de Información (ESSI) de la UPC.

El trabajo se ha completado como una parte de las tareas a realizar en uno de los paquetes de trabajo (WP) del proyecto DOGO4ML [2], en el que también trabaja, entre otros, el grupo GESSI, específicamente está incluido en el paquete de trabajo WP1, cuya responsable es Dolors Costal.

Al principio no se había pensado en ningún nombre que darle a la aplicación, por lo que el autor de este trabajo propuso el nombre de SOPCOM que viene de las siglas del título de este proyecto en inglés (SOftware Processes COmponent Manager). Tras la aprobación de los directores y la responsable del WP1 se acordó usar ese nombre para identificar el proyecto.

Se puede acceder a la primera versión de SOPCOM desarrollada en este proyecto en el link http://gessi3.essi.upc.edu:1027 con una conexión VPN de la UPC. El código se puede encontrar separado en los repositorios siguientes: https://github.com/Carlos-PG/SOPCOM para el servicio web y https://github.com/Carlos-PG/SOPCOM-frontend para la aplicación cliente.

1.2 Conceptos

En esta sección se definen algunos conceptos relacionados con el proyecto que aparecerán a lo largo de esta memoria para facilitar la comprensión del contenido de ésta.

- **Situational Method Engineering.** Disciplina de investigación propuesta en el libro Situational Method Engineering [3] que, como bien indica el nombre, propone el desarrollo de métodos teniendo en cuenta la situación en que se usan.
- **Method Chunk o Method Fragment.** Componente central del Situational Method Engineering que representa un método formado por diversos method elements (ver a continuación) de distintos tipos. Un method chunk tiene un objetivo (goal) y se valora mediante uno o más criterios. En la figura 1 se puede ver un ejemplo de method chunk para la licitación de requisitos en formato de tabla.

id	Chu-ReqEli-01
Related chunks	
Abstract	true
Name	Requirements Elicitation
Description	This activity applies some selected technique to elicit a list of requirements from a set of stakeholders
Context criteria	-
Intention	Elicit requirements
Situation	Art-LiCt-01 - List of constraints , Art-LiGo-01 - List of goals , Art-LiSt-01 - List of stakeholders ,
Process part	Act-ElRq-01 - Elicit Requirements
Product part	Art-LiRq-01 - List of requirements
Roles	Rol-RqEn-01 - Requirements Engineer : Organizes and supervises the elicitation of the requirements , Rol-SeSt-01 - Set of Set of stakeholders : Provides information useful for requirements elicitation
Tools	Too-Jira-01 - Jira

Figura 1: Method chunk en formato tabla. Fuente: Elaboración propia

- **Method Element.** Partes que componen los method chunk. Estos pueden ser de cuatro tipos: tools, artefacts, activities o roles.
- **Work Package [4].** Las diferentes partes en las que se ha dividido un proyecto, con sus propios objetivos y subobjetivos. Esto permite que equipos de trabajo diferentes puedan trabajar simultáneamente sin interferir en el trabajo del otro.
- **API RESTful [5].** Una API RESTful es una interfaz de programación de aplicaciones (API) que sigue los principios de diseño de la arquitectura REST. Algunos de estos principios son: La uniformidad o la separación entre cliente y servidor.

1.3 Formulación del problema

El WP1 del proyecto DOGO4ML busca obtener un ciclo de vida holístico contínuo para un sistema software de aprendizaje automático (MLSS). Para ello surgió la necesidad de tener una herramienta para poder gestionar un catálogo de componentes de desarrollo software configurables.

Por las características de este proyecto y futuros proyectos que se puedan ver beneficiados de utilizar esta herramienta había que tener en cuenta diferentes escenarios en que podía aplicarse cada componente, por lo que desarrollar esta aplicación utilizando situational method engineering como enfoque era la opción más adecuada.

1.4 Objetivo

El objetivo final de este TFG es desarrollar una herramienta usable y extensible para la gestión de componentes de procesos software en forma de una aplicación web. Con esta

herramienta se podrán crear, visualizar, modificar y borrar los componentes y elementos relacionados con dichos componentes, como por ejemplo componentes que les dan soporte o los roles que estos desempeñan.

Dentro del marco del proyecto DOGO4ML y específicamente dentro de la fase WP1 [2], esta aplicación ayudará a conseguir los objetivos propios del proyecto. El principal objetivo del WP1 es el de especificar, diseñar e implementar un ciclo de vida holístico y configurable para sistemas software de aprendizaje automático alineando el desarrollo y los procesos operacionales de la Ingeniería del Software y la Ingeniería de Datos.

Para alcanzar este objetivo se han establecido diversos subobjetivos que también se pretenden alcanzar usando esta aplicación. Con la aplicación se pretende definir y obtener un catálogo de componentes mencionados anteriormente y posteriormente poder combinar dichos componentes, según su información contextual, para obtener ciclos de vida holísticos.

1.5 Stakeholders

En esta sección se describen las principales partes interesadas de este TFG, ya sea que se vayan a beneficiar de su resultado final o que simplemente estén relacionadas con su desarrollo.

1.5.1 DOGO4ML

Siendo este trabajo una parte del proyecto DOGO4ML, este actúa como el principal interesado para que este TFG se lleve a cabo. Esto incluye no solo al proyecto en sí, sino a todas las personas que participan en él.

1.5.2 Directores de proyecto

Tanto Carme Quer como Xavier Franch, no sólo como parte del proyecto mencionado antes, sino también como directores del TFG, son partes interesadas ya que se encargan de dirigir este trabajo.

1.5.3 Desarrollador

Carlos Plana García como estudiante que está realizando este proyecto como su Trabajo de final de grado es parte interesada ya que será el único encargado de desarrollar y documentar este trabajo y, por tanto, la finalización de este depende de él.

1.5.4 Grupos de investigación

Como TFG que sirve para implementar la idea del Situational Method Engineering, otros grupos de investigación interesados en profundizar en este campo pueden verse beneficiados del uso de la aplicación que se pretende implementar.

1.5.5 Empresas

Como en el caso anterior, cualquier empresa con interés en aplicar este tipo de metodología puede verse beneficiada por el uso del resultado de este TFG para facilitarles este hecho.

1.6 Estado del arte

A continuación, se analizan los proyectos o aplicaciones existentes que tienen una finalidad similar a la de este TFG y que aplican el enfoque del situational method engineering, ya que es el eje central del desarrollo.

La única aplicación que se conoce que cumple los requisitos indicados es un prototipo desarrollado en el marco del proyecto europeo “SUpporting evolution and adaptation of PERsonalized Software by Exploiting contextual Data and End-user feedback” (SUPERSEDE) [6]. Se trata de un proyecto coordinado por Fondazione Bruno Kessler [7] desarrollado entre 2015 y 2018. Este proyecto utilizaba los mismos términos para referirse a los componentes que formaban parte del sistema, como son Method Chunk o Method Element ya que también aplicaba el situational method engineering.

En el marco del proyecto SUPERSEDE solo se desarrolló un prototipo simple de la aplicación para estudiar cómo las empresas podrían adoptar el método propuesto. Este prototipo estaba muy centrado en el ámbito en el que se desarrolló y todos los componentes que lo formaban ya estaban predefinidos y no permitía la inserción de nuevos. A la vez, el prototipo no ha recibido mantenimiento desde que se finalizó el proyecto y ha quedado muy desactualizado. Es por este motivo que en este TFG se plantea el desarrollo de la nueva herramienta desde cero.

1.7 Riesgos

Durante el desarrollo de cualquier proyecto pueden surgir problemas o contratiempos que provoquen que se retrasen las tareas o se tengan que crear de nuevas. A continuación, se detallan los posibles riesgos que se han identificado a los que se puede enfrentar el TFG y la gestión que se llevará a cabo para minimizar su impacto. Una de las ventajas de utilizar la metodología Scrum es que permite gestionar los retrasos de una forma más flexible añadiendo tareas al backlog que se harían más tarde.

- **Desconocimiento del enfoque teórico:**
  Al ser nuevo para el desarrollador del proyecto el concepto de situational method engineering puede llevar a que el tiempo de aprendizaje se alargue más de lo esperado y eso provoque un efecto de arrastre hacia el resto de las tareas. Este riesgo está directamente relacionado con la tarea IT1.1, que consiste en el estudio del modelo conceptual. Se gestionará resolviendo las dudas junto con los responsables del proyecto y a medida que se hayan resuelto se empezarán las tareas que hayan podido verse afectadas por esas dudas.
- **Bugs:**
  Durante el desarrollo es posible que aparezcan errores de código o bugs que puedan hacer que una tarea se alargue más de lo previsto. Este riesgo se gestionará

con una tarea de testing para cada tarea de desarrollo de código para así asegurar el correcto funcionamiento de la aplicación a medida que va avanzando el proyecto.

- **Calendario**
  Es importante ser estrictos con las fechas de entrega fijadas para el proyecto, ya que no solo podrían afectar al propio desarrollador del TFG sino que también podría retrasar las tareas de otros grupos de trabajo del proyecto del cual depende este TFG. Durante la fase de desarrollo, si alguna de las tareas queda pendiente, siempre se puede pasar al siguiente sprint gracias al uso de Scrum como metodología, pero el límite está el día 6 de junio, cuando acaba el tercer sprint, ya que esas últimas dos semanas se han destinado únicamente a la redacción de la memoria del TFG.

1.8 Metodología

En este apartado se explica la metodología que se aplicará durante el desarrollo del TFG y las herramientas que se usarán para facilitar el seguimiento.

1.8.1 Metodología escogida

Por lo que respecta a la metodología a seguir durante el desarrollo del TFG, se aplicará una metodología ágil, en particular se usará Scrum [8], con la diferencia que los sprints ya han sido planificados y tienen las tareas asignadas.

Scrum es una metodología cíclica donde se dividen las tareas en diferentes iteraciones llamadas *Sprints* que se van ejecutando dentro de los tiempos establecidos que debe durar cada una de las iteraciones. Todas las tareas se agrupan en un conjunto llamado *Backlog* el cual puede ir variando a medida que avanza el trabajo. Cuando se empieza un sprint se sacan todas las tareas que se deciden realizar en este y una vez finalizado se incluyen aquellas que o bien no se han podido acabar o bien han aparecido nuevas durante el tiempo que ha durado la iteración, pero como se ha dicho antes, para este TFG los sprints ya tienen asignadas todas las tareas. También un aspecto importante de Scrum son las reuniones periódicas de cada grupo de trabajo para actualizar sobre el estado del sprint y una reunión al final de este para valorar cómo ha transcurrido.

En el caso de este proyecto se han establecido 3 iteraciones en lo que respecta al desarrollo más una primera iteración que incluye el trabajo de documentación propio de la asignatura de Gestió de Projectes (GEP). En relación a las reuniones, se han establecido semanalmente encuentros con los directores de este Trabajo Final de Grado y con la responsable del WP1.

1.8.2 Herramientas

Para facilitar el desarrollo del proyecto utilizando la metodología mencionada anteriormente se usarán las siguientes herramientas.

Git y Github. Git [9] es un sistema de control de versiones gratuito y de código abierto. Es muy útil para proyectos de desarrollo de software ya que permite tener control sobre los

cambios que se han ido realizando en el código a lo largo del tiempo a la vez que permite tener diferentes versiones a la vez en diferentes ramas. Junto con git, se usará Github [10] como repositorio en la nube para almacenar el código.

Taiga. Taiga [11] es una herramienta de código abierto que se usará para gestionar el Backlog y los sprints de la metodología Scrum. Ofrece una interfaz donde se puede, entre otras cosas, introducir tareas, asignar puntuaciones, crear sprints, finalizarlos y ver como evoluciona el proyecto y si se están cumpliendo los objetivos de cada iteración.

1.9 Organización de la memoria

A partir de este punto la memoria recoge los aspectos más técnicos sobre la planificación y el desarrollo del proyecto. A continuación, se resume el contenido de los apartados siguientes.

- **Planificación:** Incluye todo lo relacionado con la planificación inicial del proyecto como las tareas a realizar, la metodología a seguir y las herramientas que se decidieron utilizar.
- **Presupuesto:** Expone el hipotético coste económico que supondría el desarrollo del proyecto.
- **Especificación de requisitos:** Se muestran tanto los requisitos funcionales como los no funcionales así como el proceso de obtención de estos.
- **Arquitectura del sistema:** Se describen los elementos que forman la aplicación y los recursos necesarios para su funcionamiento.
- **Implementación:** Elementos importantes de la implementación a nivel de código de la aplicación, tanto de la parte cliente como la de servidor.
- **Pruebas:** Explica los métodos usados para testear la aplicación.
- **Aspectos legales:** Se analizan los aspectos legales a los que puede estar expuesto el proyecto.
- **Seguimiento del proyecto:** Explica cómo se ha desarrollado el proyecto y las desviaciones respecto a la planificación inicial.
- **Sostenibilidad:** Se analiza el proyecto desde tres puntos de vista: económico, social y medioambiental.
- **Conclusiones y trabajo futuro:** Se explican las conclusiones extraídas del trabajo realizado y se plantea que puede seguir a este proyecto.
- **Referencias:** Se listan todas las fuentes de información utilizadas durante el desarrollo del proyecto.
- **Anexo:** Incluye material adicional para la documentación de la aplicación.

2. Planificación

Como ya se ha explicado en el capítulo de introducción, durante el desarrollo se seguirá una metodología basada en Scrum [8]. En Scrum un proyecto se descompone en iteraciones que pueden durar de quince días a un mes natural. Las tareas que se realizan en cada iteración se deciden en el inicio de la iteración a partir de la priorización en ese momento de las tareas que aparecen en el Product Backlog (que incluye todas las tareas a realizar durante el desarrollo del proyecto). Cada iteración debe llegar a una versión potencialmente entregable del producto que se desarrolla. En este TFG la diferencia está en que las tareas que se realizarán en cada iteración ya han sido pactadas de antemano con los directores del proyecto. A continuación, se describirán las tareas que se han identificado para la realización de esta aplicación y se presentará la distribución temporal en un diagrama de Gantt.

Se han establecido con los directores de este TFG y la coordinadora del WP1 de DOGO4ML [2] repartir todas las tareas de desarrollo en tres iteraciones. A estas se le añaden una fase inicial previa en la que se realizará todas las tareas correspondientes a la gestión del proyecto y una fase final dedicada a la elaboración del informe final del TFG. Estas 5 iteraciones se efectuarán en el periodo de tiempo entre el 21 de febrero de 2022 y el 20 de junio de 2022. Por último, se ha añadido una fase de seguimiento que tiene la misma duración que el proyecto y que agrupa todas las tareas que representan las reuniones previstas a lo largo del desarrollo.

2.1 Tareas

A continuación, se listan las tareas que se han identificado para cada iteración de este TFG.

2.1.1 Fase de seguimiento

Esta fase empieza justo antes de la primera iteración y acaba justo después de la fase final, por lo tanto, del 20 de febrero de 2022 al 20 de junio de 2022. En realidad las tareas que se agrupan en esta fase de seguimiento del proyecto corresponderá a las reuniones de retrospectiva y planificación del final e inicio de cada iteración y a las reuniones semanales de seguimiento.

- Seguimiento semanal: Reuniones semanales con los *product owners* para actualizar sobre el estado del proyecto.
- FS1. Seguimiento previo a la primera iteración: Reunión de planificación de la primera iteración.
- FS2. Seguimiento previo a la segunda iteración: Reunión de retrospectiva de la primera iteración y de planificación de la segunda.
- FS3. Seguimiento previo a la tercera iteración: Reunión de retrospectiva de la segunda iteración y de planificación de la tercera.
- FS4. Seguimiento posterior a la tercera iteración: Reunión de retrospectiva de la tercera iteración.

2.1.2 Fase inicial

Esta fase empieza el 21 de febrero de 2022 y finaliza el 21 de marzo del mismo año. Incluye las tareas iniciales correspondientes a la definición y planificación temporal y económica del proyecto, así como el análisis de sostenibilidad del proyecto y la futura aplicación que se va a desarrollar. Esta primera iteración es importante ya que definirá lo que se va a hacer y los pasos que se seguirá para hacerlo.

- FI1 - Contextualización del proyecto: Establecer el marco en el que se desarrolla el proyecto y justificar porque es necesario.
- FI2 - Alcance del proyecto: Definir objetivos, subobjetivos y requisitos del proyecto.
- FI3 - Planificación temporal de las tareas: Planificar las tareas a realizar durante el proyecto asignarlas a los roles correspondientes.
- FI4 - Análisis económico: Estudiar el coste económico que supone el desarrollo.
- FI5 - Análisis de sostenibilidad: Estudiar el impacto económico, medioambiental y social del proyecto.
- FI6 - Documentación fita final: Juntar toda la documentación anterior arreglando errores. Para poder realizar esta tarea se deben acabar las anteriores.

2.1.3 Iteración 1

Esta iteración empieza el 21 de febrero y acaba el 28 de marzo. Esta agrupa todas las tareas iniciales de desarrollo, como el aprendizaje y comprensión del dominio del proyecto y la especificación, diseño e implementación de dos entradas (endpoints) bàsicas de los servicios web que darán acceso a la base de datos donde se guardará la información sobre los componentes de procesos software que se quiere gestionar. En concreto se trata de llegar a las entradas que permitan llenar la base de datos gestionada desde cero y que permita inicializar la base de datos en caso que sea necesario. Lo que se obtenga de esta iteración servirá como base para el resto de iteraciones. Esta primera iteración también será el primer contacto con las tecnologías que se van a usar durante todo el proyecto, situación que puede afectar a su duración.

- IT1.1 - Estudio del modelo conceptual: Estudiar el modelo conceptual proveído por los *product owners*.
- IT1.2 - Especificación de restricciones de integridad: A partir del modelo conceptual, extraer restricciones de integridad que debe cumplir el sistema.
- IT1.3 - Diseño de la base de datos: A partir del modelo conceptual diseñar una base de datos relacional.
- IT1.4 - Identificar historias de usuario: Concretar las historias de usuario para que se ajusten a los requisitos.
- IT1.5 - Proponer endpoints básicos: Decidir los endpoints básicos de la aplicación para empezar a llenar la base de datos.
- IT1.6 - Diseño de JSON: Decidir la estructura que ha de tener el JSON en las llamadas a los endpoints.
- IT1.7 - Diseño arquitectura lógica de los servicios web: Estructurar las aplicaciones web que acceden a la base de datos.
- IT1.8 - Implementación de endpoints básicos: Implementar los endpoints propuestos en la tarea IT1.5. Esta tarea depende de la IT1.1 a la IT1.6.
- IT1.9 - Testing: Probar el correcto funcionamiento de los endpoints implementados.

- IT1.10 - Documentación final: A medida que se desarrolla el proyecto, añadir a la documentación final la información necesaria relacionada con las tareas que se van realizando.

2.1.4 Iteración 2

Esta iteración contiene el desarrollo de las principales funcionalidades de los servicios web y, por tanto, es la iteración más importante ya que el correcto funcionamiento de la aplicación depende principalmente de lo que se desarrolle aquí. Se podrá afrontar con mayores garantías esta iteración gracias a que ya se ha practicado con las tecnologías y la integración entre ellas en la iteración previa. Se tratará de desarrollar el resto de entradas que serán necesarias, teniendo en cuenta los escenarios de uso que se prevén, para poder gestionar los datos de los componentes de procesos software. Está previsto que empiece el día 29 de marzo y acabe el 9 de mayo.

- IT2.1 - Entender escenarios de uso de la herramienta: Estudiar los escenarios de uso previstos de la aplicación.
- IT2.2 - Identificar historias de usuario: A partir de los escenarios de uso extraer nuevas historias de usuario.
- IT2.3 - Diseñar endpoints para operaciones CRUD necesarias: Especificar el resto de endpoints de la aplicación.
- IT2.4 - Implementación de endpoints para consultas: Implementar aquellos endpoints que sean consultas.
- IT2.5 - Testing consultas: Testear el correcto funcionamiento de los endpoints de la tarea IT2.4.
- IT2.6 - Implementación de endpoints para inserciones: Implementar aquellos endpoints que sean inserciones.
- IT2.7 - Testing inserciones: Testear el correcto funcionamiento de los endpoints de la tarea IT2.6.
- IT2.8 - Implementación de endpoints para modificaciones: Implementar aquellos endpoints que sean modificaciones.
- IT2.9 - Testing modificaciones: Testear el correcto funcionamiento de los endpoints de la tarea IT2.8.
- IT2.10 - Implementación de endpoints para borrados: Implementar aquellos endpoints que sean borrados.
- IT2.11 - Testing borrado: Testear el correcto funcionamiento de los endpoints de la tarea IT2.10.
- IT2.12 - Documentación final: A medida que se desarrolla el proyecto, añadir a la documentación final la información necesaria relacionada con las tareas que se van realizando.

2.1.5 Iteración 3

Esta será la tercera y última iteración de desarrollo. Se centrará en desarrollar la aplicación cliente que usará las entradas a los servicios web desarrolladas en las iteraciones previas. Por ello la duración se ha reducido para este sprint respecto a los tres anteriores, en este caso se empezará el 10 de mayo y acabará el 6 de junio.

- IT3.1 - Priorización de historia de usuario: Priorizar las historias de usuario obtenidas en las iteraciones anteriores.
- IT3.2 - Diseño de historias de usuario: Especificar las historias de usuario obtenidas en las iteraciones anteriores.
- IT3.3 - Implementación de historias de usuario: Implementar la aplicación cliente siguiendo las historias de usuario por orden de prioridad.
- IT3.4 - Documentación final: A medida que se desarrolla el proyecto, añadir a la documentación final la información necesaria relacionada con las tareas que se van realizando.

2.1.6 Fase final

Desde el 7 de junio hasta el 20 de junio estará el último sprint, que se centrará en la integración de la documentación realizada en cada una de las iteraciones y fases anteriores, en la memoria final de este TFG.

- FF1 - Documentación final: Acabar el documento para la entrega final del TFG con toda la información necesaria.

2.2 Recursos

Para poder realizar el proyecto será necesario disponer de un equipo y diversas herramientas que se describen a continuación.

2.2.1 Recursos humanos

Los diferentes roles necesarios para la realización de este proyecto son los siguientes.

- PO - Product Owner: Son los encargados de supervisar el proyecto. Incluye a los directores de este TFG Carme Quer y Xavier Franch además de la responsable del WP1 Dolors Costal.
- JP - Jefe de proyecto: Es el encargado de gestionar el proyecto y redactar la documentación correspondiente.
- AS - Arquitecto de Software: Es el encargado de definir la estructura de la base de datos y los servicios que acceden a esta.
- ED - Equipo de desarrollo: Equipo encargado del desarrollo del código siguiendo las historias de usuario y acogiéndose a los requisitos establecidos.
- ET - Equipo de testing: Es el equipo encargado de realizar todas las pruebas de testing para asegurar el funcionamiento esperado de la aplicación.

2.2.2 Recursos software

Estos son los recursos software que se utilizarán durante el transcurso del proyecto.

- GM - Google meet: Servicio de videollamadas de Google usado para las reuniones [12].
- GD - Google Drive: Servicio de almacenamiento en la nube de Google donde se comparten los archivos dedicados a la documentación [13].
- DC - Documentos de Google: Herramienta usada para la redacción de la documentación que también permite el trabajo colaborativo [14].
- HC - Hojas de cálculo de Google: Herramienta usada para hacer el diagrama de Gantt [15].

- T - Taiga: Herramienta de soporte de la metodología Scrum que se usa en este proyecto [11].
- G - Git: Herramienta de gestión de versiones [9].
- GH - Github: Repositorio de código en la nube [10].
- VS - Visual Studio Code: Editor de texto fuente usado durante el desarrollo de la aplicación [16].
- D - Docker: Herramienta que permite el despliegue de aplicaciones en contenedores [28]
- MS - MySQL: Sistema de gestión de bases de datos relacional [22].
- A - Angular: Framework de desarrollo de aplicaciones web [29].
- P - Postman: Herramienta usada para testear llamadas a APIs [30].
- D - diagrams.net: Herramienta para crear diagramas colaborativos [31].

2.2.3 Recursos hardware

Este apartado describe los distintos recursos hardware que se utilizaran durante el desarrollo.

- Portátil: Durante todas las tareas del proyecto se usará un ordenador portátil para usar todos los software mencionados anteriormente.

2.2.4 Recursos generales

Este TFG se realizará en su totalidad desde el domicilio del autor, por lo tanto, el único recurso a tener en cuenta en este apartado sería el consumo eléctrico que pueda derivar del desarrollo del proyecto.

2.3 Tabla resumen

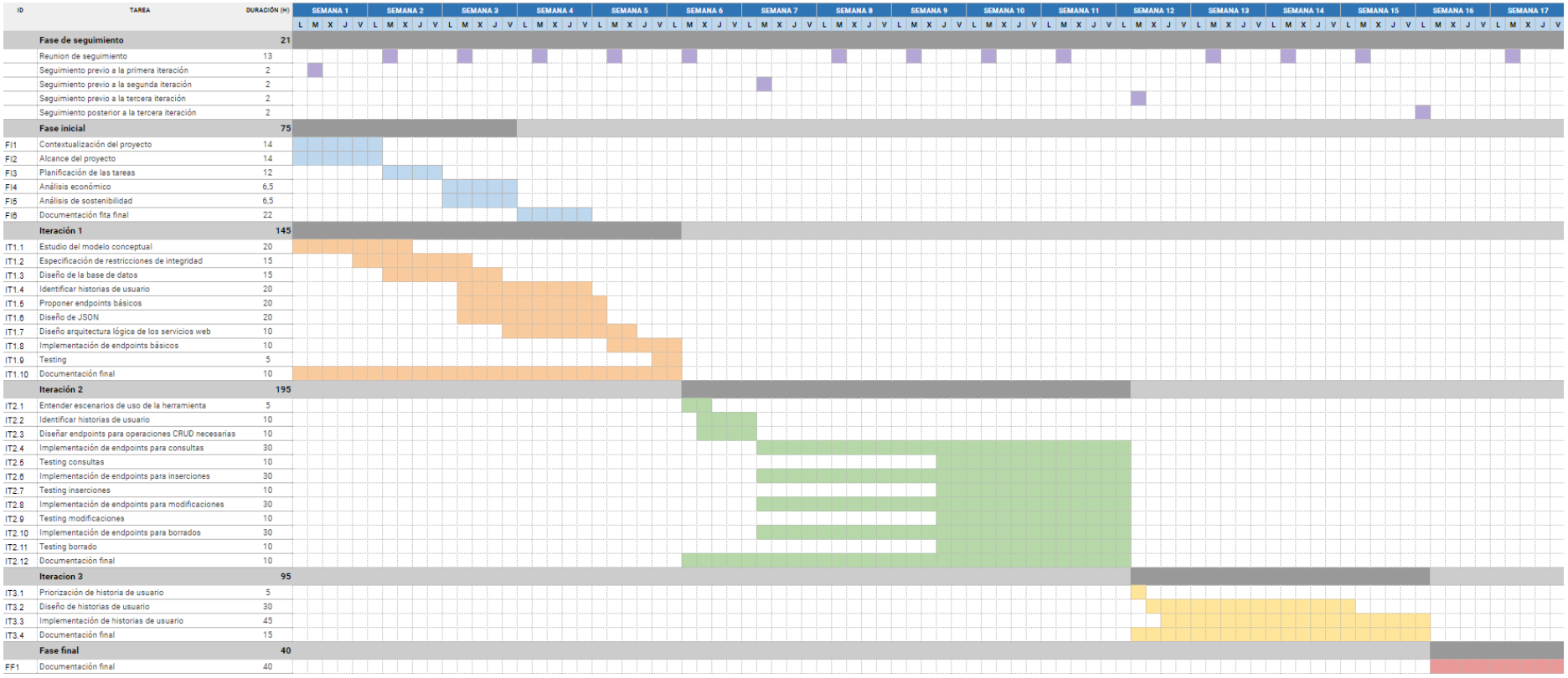
En la página siguiente, en la Tabla 1, se resumen las tareas con los recursos asignados a cada una y las horas previstas de duración.

ID	Nombre	Recursos humanos	Recursos software	Horas
	Seguimiento semanal	PO, JP	GM, DC, GD, T	13
FS1	Seguimiento previo a la primera iteración	PO, AS, ED	GM, DC, GD, T	2
FS2	Seguimiento previo a la primera iteración	PO, ED, ET	GM, DC, GD, T	2
FS3	Seguimiento previo a la primera iteración	PO, ED, ET	GM, DC, GD, T	2
FS4	Seguimiento posterior a la tercera iteración	PO, JP	GM, DC, GD, T	2
FI1	Contextualización del proyecto	JP	GD, DC	14
FI2	Alcance del proyecto	JP	GD, DC	14
FI3	Planificación temporal de las tareas	JP	GD, DC, HC	12
FI4	Análisis económico	JP	GD, DC	6.5
FI5	Análisis de sostenibilidad	JP	GD, DC	6.5
FI6	Documentación fita final	JP	GD, DC	22
IT1.1	Estudio del modelo conceptual	AS	GD, DC	20
IT1.2	Especificación de restricciones de integridad	AS	GD, DC	15
IT1.3	Diseño de la base de datos	AS	GD, DC, MS	15
IT1.4	Identificar y describir historias de usuario	AS	GD, DC	20
IT1.5	Proponer endpoints básicos	AS	GD, DC	20
IT1.6	Diseño de JSON	AS	GD, DC	20
IT1.7	Diseño arquitectura lógica de los servicios web	AS	GD, DC	10
IT1.8	Implementación de endpoints básicos	ED	VS, G, GH, D	10
IT1.9	Testing	ET	VS, G, GH, P	5
IT1.10	Documentación final	JP	GD, DC	10
IT2.1	Entender escenarios de uso de la herramienta	ED	GD, DC	5
IT2.2	Identificar historias de usuario	AS	GD, DC	10
IT2.3	Diseñar endpoints para operaciones CRUD necesarias	AS	GD, DC	10
IT2.4	Implementación de endpoints para consultas	ED	VS, G, GH, D	30
IT2.5	Testing consultas	ET	VS, G, GH, P	10
IT2.6	Implementación de endpoints para inserciones	ED	VS, G, GH, D	30
IT2.7	Testing inserciones	ET	VS, G, GH, P	10
IT2.8	Implementación de endpoints para modificaciones	ED	VS, G, GH, D	30
IT2.9	Testing modificaciones	ET	VS, G, GH, P	10
IT2.10	Implementación de endpoints para borrados	ED	VS, G, GH, D	20
IT2.11	Testing borrado	ET	VS, G, GH, P	5
IT2.12	Documentación final	JP	GD, DC	10
IT3.1	Priorización de historia de usuario	AS	GD, DC	5
IT3.2	Diseño de historias de usuario	AS	GD, DC,	20
IT3.3	Implementación de historias de usuario	ED	VS, G, GH, A	30
IT3.4	Documentación final	JP	GD, DC	15
FF1	Documentación final	JP	GD, DC	40

Tabla 1: Resumen de tareas. Fuente: Elaboración propia

2.4 Diagrama de Gantt

En la Figura 2 podemos ver el Diagrama de Gantt correspondiente a la planificación inicial de las tareas. Cada fase del proyecto está separada y tiene asignada un color distinto.



ID	TAREA	DURACIÓN (H)
	Fase de seguimiento	**21**
	Reunion de seguimiento	13
	Seguimiento previo a la primera iteración	2
	Seguimiento previo a la segunda iteración	2
	Seguimiento previo a la tercera iteración	2
	Seguimiento posterior a la tercera iteración	2
	Fase inicial	**75**
FI1	Contextualización del proyecto	14
FI2	Alcance del proyecto	14
FI3	Planificación de las tareas	12
FI4	Análisis económico	6,5
FI5	Análisis de sostenibilidad	6,5
FI6	Documentación fita final	22
	Iteración 1	**145**
IT1.1	Estudio del modelo conceptual	20
IT1.2	Especificación de restricciones de integridad	15
IT1.3	Diseño de la base de datos	15
IT1.4	Identificar historias de usuario	20
IT1.5	Proponer endpoints básicos	20
IT1.6	Diseño de JSON	20
IT1.7	Diseño arquitectura lógica de los servicios web	10
IT1.8	Implementación de endpoints básicos	10
IT1.9	Testing	5
IT1.10	Documentación final	10
	Iteración 2	**195**
IT2.1	Entender escenarios de uso de la herramienta	5
IT2.2	Identificar historias de usuario	10
IT2.3	Diseñar endpoints para operaciones CRUD necesarias	10
IT2.4	Implementación de endpoints para consultas	30
IT2.5	Testing consultas	10
IT2.6	Implementación de endpoints para inserciones	30
IT2.7	Testing inserciones	10
IT2.8	Implementación de endpoints para modificaciones	30
IT2.9	Testing modificaciones	10
IT2.10	Implementación de endpoints para borrados	30
IT2.11	Testing borrado	10
IT2.12	Documentación final	10
	Iteracion 3	**95**
IT3.1	Priorización de historia de usuario	5
IT3.2	Diseño de historias de usuario	30
IT3.3	Implementación de historias de usuario	45
IT3.4	Documentación final	15
	Fase final	**40**
FF1	Documentación final	40

Figura 2. Diagrama de Gantt. Fuente: Elaboración propia. Plantilla proporcionada por Hojas de Cálculo de Google

3. Presupuesto

Los recursos humanos y materiales mencionados con anterioridad suponen un coste económico. A este coste se le añaden contingencias y gastos indirectos que pueden darse durante el transcurso del proyecto. A continuación, se describen todos los costes derivados del proyecto y cómo se van a gestionar durante el desarrollo de éste.

3.1 Identificación y estimación de costes

En este apartado se presentan los diferentes costes que se han identificado y una aproximación de lo que suponen teniendo en cuenta los recursos humanos y materiales necesarios y los riesgos que se han detectado.

3.1.1 Recursos humanos

Dado que se trata de un trabajo final de grado, todos los roles identificados, excepto el de *Product Owner*, serán representados por el autor del trabajo. Pero para realizar una predicción real se tratará cada rol de forma diferente y con sueldos acordes a los actuales para cada puesto. En la Tabla 2 se puede ver el coste de cada rol de acuerdo a los sueldos brutos mínimos obtenidos de Tecnoempleo [17] y suponiendo que por contrato se trabajan 1750 horas al año.

Rol	Salario bruto por hora (€/h)	Horas totales (h)	Coste (€)
Jefe de proyecto	28	165	4620
Arquitecto de software	22	167	3674
Programador	17	161	2737
Tester	15	44	660
Total			11691

Tabla 2: Coste recursos humanos. Fuente: Elaboración propia

Exceptuando las reuniones, cada tarea tiene asignado un único recurso humano, por lo que el coste de cada tarea será resultado de multiplicar el precio hora del rol asignado por el número de horas previstas que va a durar. De esta manera el coste por tarea planificada es el que se puede ver en la Tabla 3 en la siguiente página.

ID	Nombre	Recursos humanos	Horas	Coste (€)
	Seguimiento semanal	PO, JP	13	364
FS1	Seguimiento previo a la primera iteración	PO, AS, ED	2	78
FS2	Seguimiento previo a la primera iteración	PO, ED, ET	2	64
FS3	Seguimiento previo a la primera iteración	PO, ED, ET	2	64
FS4	Seguimiento posterior a la tercera iteración	PO, JP	2	56
FI1	Contextualización del proyecto	JP	14	392
FI2	Alcance del proyecto	JP	14	392
FI3	Planificación temporal de las tareas	JP	12	336
FI4	Análisis económico	JP	6.5	182
FI5	Análisis de sostenibilidad	JP	6.5	182
FI6	Documentación fita final	JP	22	616
IT1.1	Estudio del modelo conceptual	AS	20	440
IT1.2	Especificación de restricciones de integridad	AS	15	330
IT1.3	Diseño de la base de datos	AS	15	330
IT1.4	Identificar y describir historias de usuario	AS	20	440
IT1.5	Proponer endpoints básicos	AS	20	440
IT1.6	Diseño de JSON	AS	20	440
IT1.7	Diseño arquitectura lógica de los servicios web	AS	10	220
IT1.8	Implementación de endpoints básicos	ED	10	170
IT1.9	Testing	ET	5	75
IT1.10	Documentación final	JP	10	280
IT2.1	Entender escenarios de uso de la herramienta	ED	5	85
IT2.2	Identificar historias de usuario	AS	10	220
IT2.3	Diseñar endpoints para operaciones CRUD necesarias	AS	10	220
IT2.4	Implementación de endpoints para consultas	ED	30	510
IT2.5	Testing consultas	ET	10	150
IT2.6	Implementación de endpoints para inserciones	ED	30	510
IT2.7	Testing inserciones	ET	10	150
IT2.8	Implementación de endpoints para modificaciones	ED	30	510
IT2.9	Testing modificaciones	ET	10	150
IT2.10	Implementación de endpoints para borrados	ED	20	340
IT2.11	Testing borrado	ET	5	75
IT2.12	Documentación final	JP	10	280
IT3.1	Priorización de historia de usuario	AS	5	110
IT3.2	Diseño de historias de usuario	AS	20	440
IT3.3	Implementación de historias de usuario	ED	30	510
IT3.4	Documentación final	JP	15	420
FF1	Documentación final	JP	40	1120
	Total			11691

Tabla 3: Coste tareas. Fuente: Elaboración propia

3.1.2 Hardware y software

Como recursos de hardware y software, se usará un portátil y diversos software durante el transcurso de este proyecto. Estos recursos también suponen un coste a tener en cuenta en el presupuesto.

En el caso de los recursos software que se usarán, todos son gratuitos o se pueden utilizar de forma gratuita, por lo tanto, no habrá costes relacionados con los softwares utilizados. En cambio, en cuanto al hardware, se utilizará un portátil durante las 540 horas que dura el proyecto. En la Tabla 4 se puede ver el resumen de los costes de hardware siguiendo el cálculo siguiente, teniendo en cuenta que 2022 tiene unos 249 días laborables [18] y la jornada laboral es de 5 horas.

$$Amortización = \frac{Precio}{Vida\ útil * días\ hábiles * jornada\ laboral} * horas\ proyecto$$

Recurso	Precio (€)	Vida útil (años)	Amortización
Portátil ASUS	1200	4	109

Tabla 4: Coste recursos Hardware. Fuente: Elaboración propia

3.1.3 Contingencias e imprevistos

En caso de problemas, se ha de tener en cuenta un margen de maniobra económica para poder solucionarlos. En el caso de las contingencias, se destinará un 10% extra del presupuesto del proyecto, el coste de estas se puede ver en la Tabla 5.

Nombre	Coste (€)	Contingencia (%)	Coste total (€)
Recursos humanos	11691	10	1169
Hardware	1200	10	120
Total			1343

Tabla 5: Coste contingencias. Fuente: Elaboración propia

En cuanto a los imprevistos, se pueden incluir en este apartado los identificados anteriormente. Pero para estos riesgos ya se proponen soluciones dentro de la planificación inicial. En cambio, otros riesgos como problemas con el ordenador sí que se han de tener en cuenta, como se refleja en la Tabla 6.

Imprevisto	Coste (€)	Riesgo (%)	Coste total (€)
Problema ordenador	200	5	10

Tabla 6: Coste de imprevistos. Fuente: Elaboración propia

3.1.4 Presupuesto final

Finalmente, sumando todos los costes identificados anteriormente, obtenemos el presupuesto final que se ha estimado para este proyecto en la Tabla 7.

Coste	Precio (€)
Recursos humanos	11691
Hardware	109
Contingencias	1343
Imprevistos	10
Total	13153

Tabla 7: Presupuesto final. Fuente: Elaboración propia

4. Especificación de requisitos

Los requisitos de una aplicación definen qué tipo de aplicación se ha de desarrollar para poder lograr los objetivos que se han propuesto alcanzar. En este apartado se explican los requisitos que debe cumplir este proyecto.

4.1 Proceso de obtención de requisitos

Al iniciar el proyecto se habían concretado una serie de ideas para los requisitos. Por lo que, las primeras reuniones con los clientes, en este caso los directores del proyecto y la responsable del WP1 de DOGO4ML, se dedicaron a establecer dichos requisitos de acuerdo a los objetivos del proyecto.

Para establecer los requisitos, los clientes expusieron los conceptos, atributos y relaciones entre ellos que debía gestionar la nueva herramienta, aportando un modelo conceptual UML, propuesto por miembros del grupo en [19,20]. Las primeras reuniones abordaron la comprensión del contexto del proyecto y de dichos conceptos, una vez quedaron claros se acordó los requisitos funcionales y no funcionales de la nueva herramienta.

4.2 Requisitos funcionales

Los requisitos funcionales definen qué debe permitir hacer el sistema. Junto con los stakeholders se han identificado los requisitos funcionales explicados en esta sección.

4.2.1 Listado de requisitos funcionales

- La aplicación debe permitir visualizar la información general de todos los method chunks existentes.
- La aplicación debe permitir visualizar toda la información general y los elementos relacionados con un method chunk.
- La aplicación debe permitir la inserción/modificación/borrado de tools.
- La aplicación debe permitir la inserción/modificación/borrado de artefacts.
- La aplicación debe permitir la inserción/modificación/borrado de activities.
- La aplicación debe permitir la inserción/modificación/borrado de roles.
- La aplicacion debe permitir la inserción/modificación/borrado de criteria.
- La aplicación debe permitir la inserción/borrado de goals.
- La aplicación debe permitir la inserción/modificación/borrado de method chunks.
- La aplicación debe permitir la inserción/borrado de relaciones con todo el resto de elementos del sistema.
- La aplicación debe permitir la visualización y descarga de toda la información de un method chunk y sus elementos relacionados en formato de tabla.

4.2.2 Descripción de historias de usuario

A continuación se incluyen las historias de usuario que describen los requisitos del sistema en formato de tabla. En cada tabla se indica el código identificador de la historia, el nombre, la descripción y los criterios de aceptación.

AMC	Add method chunk
Como usuario quiero poder añadir un method chunk con tal de poder gestionar un proceso de software.	
● Se ha creado un nuevo registro en la base de datos con la información del chunk. ● Se ha creado un registro en la base de datos por cada elemento válido relacionado. ● Se devuelve un mensaje en caso de error. ● Se actualiza la lista de chunks con el nuevo chunk. ● En la pantalla aparece el chunk actualizado y se puede acceder a la vista de tabla.	

GMC	Get method chunk
Como usuario quiero poder consultar un method chunk para ver toda la información del proceso de software y los elementos que lo forman.	
● Se ha devuelto en formato JSON el chunk indicado. ● En la pantalla aparece toda la información del chunk. ● Se ha informado en caso de que no haya podido encontrar el chunk.	

GAMC	Get all method chunk
Como usuario quiero poder obtener una lista con todos los method chunks existentes para poder ver todos los chunks que existen.	
● Se ha devuelto un array con la información básica de todos los chunks. ● Se ve una lista de todos los chunks existentes donde se puede acceder a ellos.	

UMC	Update method chunk
Como usuario quiero poder modificar un method chunk con tal de actualizar su información y los elementos que lo componen.	
● Se ha actualizado el registro del chunk indicado. ● Se han actualizado las relaciones válidas del chunk. ● Se ha indicado con un mensaje de error en caso de actualización inválida. ● Se actualiza la pantalla con los datos introducidos.	

DMC	Delete method chunk
Como usuario quiero poder eliminar un method chunk para quitar información que ya no necesito.	
● Se ha borrado el registro del chunk indicado. ● Se han borrado los registros de los elementos relacionados. ● Se ha informado en caso de ser un borrado inválido. ● Se actualiza la ventana quitando el chunk de la lista y navegando a la ventana para añadir uno nuevo.	

AT	Add tool
Como usuario quiero poder añadir una tool para poder asignarla a un method chunk o relacionarla con otra tool.	
● Se ha añadido un nuevo registro en la base de datos con la información de la tool. ● Se han añadido las relaciones válidas. ● Se ha informado con un mensaje de error de relaciones inválidas. ● Se actualiza la lista de tools con la nueva tool. ● Se muestra en pantalla la tool con la información actualizada.	

GT	Get tool
Como usuario quiero poder consultar una tool con tal de conocer toda su información y sus tools relacionadas.	
● Se ha devuelto la información completa de la tool en formato JSON. ● Se ha devuelto un mensaje de error en caso de no encontrar la tool indicada. ● Se muestra en la pantalla la información de la tool indicada.	

GAT	Get all tools
Como usuario quiero poder obtener todas las tools para ver las tools existentes.	
● Se ha devuelto un array en formato JSON con la información básica de todas las tools. ● Se muestra en pantalla una lista con todas las tools para acceder a ellas.	

UT	Update tool
Como usuario quiero poder modificar una tool para actualizar su información y las tools que están relacionadas.	
● Se actualiza el registro de la tool correspondiente. ● Se han actualizado las relaciones válidas de la tool. ● Se ha indicado en caso de error con un mensaje. ● Se actualiza la ventana con la nueva información introducida.	

DT	Delete tool
Como usuario quiero poder eliminar una tool con tal de poder deshacerme de información que no necesito y sus relaciones.	
● Se ha borrado el registro correspondiente de la base de datos. ● Se han borrado las relaciones existentes de la tool. ● Se informa en caso de error con un mensaje. ● Se actualiza la ventana quitando la tool de la lista y navegando a la ventana para añadir una nueva.	

AAR	Add artefact
Como usuario quiero poder añadir un artefact para poder asignarlo a un method chunk o relacionarlo con otro artefact.	
● Se ha añadido un nuevo registro en la base de datos con la información del artefact. ● Se han añadido las relaciones válidas. ● Se ha informado con un mensaje de error de relaciones inválidas. ● Se actualiza la lista de artefactos con el nuevo artefact. ● Se muestra en pantalla el artefact con la información actualizada.	

GAR	Get artefact
Como usuario quiero poder consultar un artefact con tal de conocer toda su información y sus elementos relacionados.	
● Se ha devuelto la información completa del artefacto en formato JSON. ● Se ha devuelto un mensaje de error en caso de no encontrar el artefacto indicado. ● Se muestra en la pantalla la información del artefacto añadido.	

GAAR	Get all artefact
Como usuario quiero poder obtener todos los artefacts para ver los artefacts existentes.	
● Se ha devuelto un array en formato JSON con la información básica de todos los artefacts. ● Se muestra en pantalla una lista con todas los artefacts para acceder a ellos.	

UPAR	Update artefact
Como usuario quiero poder modificar un artefact para actualizar su información y los elementos que están relacionados.	
● Se actualiza el registro del artefact correspondiente. ● Se han actualizado las relaciones válidas del artefact. ● Se ha indicado en caso de error con un mensaje. ● Se actualiza la ventana con la nueva información introducida.	

DAR	Delete artefact
Como usuario quiero poder eliminar un artefact con tal de poder deshacerme de información que no necesito y sus relaciones.	
● Se ha borrado el registro correspondiente de la base de datos. ● Se han borrado las relaciones existentes del artefact. ● Se informa en caso de error con un mensaje. ● Se actualiza la ventana quitando el artefact de la lista y navegando a la ventana para añadir uno nuevo.	

AAC	Add activity
Como usuario quiero poder añadir una activity para poder asignarla a un method chunk o relacionarla con otra activity.	
● Se ha añadido un nuevo registro en la base de datos con la información de la activity. ● Se han añadido las relaciones válidas. ● Se ha informado con un mensaje de error de relaciones inválidas. ● Se actualiza la lista de activities con la nueva activity. ● Se muestra en pantalla la activity con la información actualizada.	

GAC	Get activity
Como usuario quiero poder consultar una activity con tal de conocer toda su información y sus activities relacionadas.	
● Se ha devuelto la información completa de la activity en formato JSON. ● Se ha devuelto un mensaje de error en caso de no encontrar la activity indicada. ● Se muestra en la pantalla la información de la activity indicada.	

GAAC	Get all activities
Como usuario quiero poder obtener todas las activities para ver las activities existentes.	
● Se ha devuelto un array en formato JSON con la información básica de todas las activities. ● Se muestra en pantalla una lista con todas las activities para acceder a ellas.	

UAC	Update activity
Como usuario quiero poder modificar una activity para actualizar su información y las activities que están relacionadas.	
● Se actualiza el registro de la activity correspondiente. ● Se han actualizado las relaciones válidas de la activity. ● Se ha indicado en caso de error con un mensaje. ● Se actualiza la ventana con la nueva información introducida.	

DAC	Delete activity
Como usuario quiero poder eliminar una activity con tal de poder deshacerme de información que no necesito y sus relaciones.	
● Se ha borrado el registro correspondiente de la base de datos. ● Se han borrado las relaciones existentes de la activity. ● Se informa en caso de error con un mensaje. ● Se actualiza la ventana quitando la activity de la lista y navegando a la ventana	

para añadir una nueva.

AR	**Add role**
Como usuario quiero poder añadir un role para poder asignarlo a un method chunk o relacionarlo con otro role.	
● Se ha añadido un nuevo registro en la base de datos con la información del role. ● Se han añadido las relaciones válidas. ● Se ha informado con un mensaje de error de relaciones inválidas. ● Se actualiza la lista de roles con el nuevo role. ● Se muestra en pantalla el role con la información actualizada.	

GRL	**Get role**
Como usuario quiero poder consultar un role con tal de conocer toda su información y sus elementos relacionados.	
● Se ha devuelto la información completa del role en formato JSON. ● Se ha devuelto un mensaje de error en caso de no encontrar el role indicado. ● Se muestra en la pantalla la información del role añadido.	

GARL	**Get all role**
Como usuario quiero poder obtener todos los roles para ver los roles existentes.	
● Se ha devuelto un array en formato JSON con la información básica de todos los roles. ● Se muestra en pantalla una lista con todas los roles para acceder a ellos.	

URL	**Update role**
Como usuario quiero poder modificar un role para actualizar su información y los elementos que están relacionados.	
● Se actualiza el registro del role correspondiente. ● Se han actualizado las relaciones válidas del role. ● Se ha indicado en caso de error con un mensaje. ● Se actualiza la ventana con la nueva información introducida.	

DRL	**Delete role**
Como usuario quiero poder modificar un role para actualizar su información y los elementos que están relacionados.	
● Se ha borrado el registro correspondiente de la base de datos. ● Se han borrado las relaciones existentes del role.	

● Se informa en caso de error con un mensaje. ● Se actualiza la ventana quitando el role de la lista y navegando a la ventana para añadir uno nuevo.

ACR	**Add criterion**
Como usuario quiero poder añadir un criterion para poder relacionarlo al contexto de un method chunk.	
● Se ha añadido un nuevo registro en la base de datos con la información del criterion. ● Se han añadido los valores válidos. ● Se ha informado con un mensaje de error en caso de valores inválidos. ● Se actualiza la lista de criteria con el nuevo criterion. ● Se muestra en pantalla el criterion con la información actualizada.	

GCR	**Get criterion**
Como usuario quiero poder obtener un criterion para ver su información y todos los valores que puede adoptar.	
● Se ha devuelto la información completa del criterion y sus valores en formato JSON. ● Se ha devuelto un mensaje de error en caso de no encontrar el criterion indicado. ● Se muestra en la pantalla la información del criterion añadido.	

GACR	**Get all criterion**
Como usuario quiero poder obtener todos los criterion para poder elegir cual relacionar con un method chunk.	
● Se ha devuelto un array en formato JSON con la información básica de todos los criterion y sus valores. ● Se muestra en pantalla una lista con todas los criterion para acceder a ellos.	

UCR	**Update criterion**
Como usuario quiero poder actualizar un criterion para poder actualizar su información y los valores que puede tomar.	
● Se actualiza el registro del criterion correspondiente. ● Se han actualizado los valores válidos del criterion. ● Se ha indicado en caso de error con un mensaje. ● Se actualiza la ventana con la nueva información introducida.	

DCR	Delete criterion:
Como usuario quiero poder eliminar un criterion para poder borrar criterios que ya no necesito.	
● Se ha borrado el registro correspondiente de la base de datos. ● Se han borrado los valores existentes del criterion. ● Se informa en caso de error con un mensaje. ● Se actualiza la ventana quitando el criterion de la lista y navegando a la ventana para añadir uno nuevo.	

AG	Add goal
Como usuario quiero poder añadir un goal para poder asignarle una intención a un method chunk.	
● Se ha añadido un nuevo registro en la base de datos con la información del goal. ● Se ha informado con un mensaje en caso de error. ● Se actualiza la lista de goals con el nuevo goal.	

GAG	Get all goal
Como usuario quiero poder obtener todos los goals para elegir cual asignar a un method chunk.	
● Se ha devuelto un array en formato JSON con la información básica de todos los goals. ● Se muestra en pantalla una lista con todas los goals.	

UG	Update goal
Como usuario quiero poder modificar un goal para poder actualizar la información de un objetivo.	
● Se actualiza el registro del goal correspondiente. ● Se ha indicado en caso de error con un mensaje. ● Se actualiza la ventana con la nueva información introducida.	

DG	Delete goal
Como usuario quiero poder eliminar un goal para poder deshacerme de objetivos que ya no necesito.	
● Se ha borrado el registro correspondiente de la base de datos. ● Se informa en caso de error con un mensaje. ● Se actualiza la ventana quitando el goal de la lista.	

MCT	Method chunk table
Como usuario quiero poder visualizar y descargar un method chunk en formato de tabla para poder ver toda la información de forma clara.	
• Se ha actualizado la pantalla a una tabla con la información completa del chunk seleccionado. • Se da la opción de descargar la tabla en un archivo pdf.	

4.2.3 Modelo conceptual de datos

Como se ha mencionado anteriormente al describir el proceso seguido para la determinación de los requisitos de la herramienta, el modelo conceptual UML ha sido aportado inicialmente por los clientes, es decir, los directores de este proyecto. Este modelo es una aportación realizada por los investigadores del grupo GESSI en las publicaciones [19, 20]. Sin embargo, durante el estudio inicial realizado en las primeras reuniones del proyecto con los directores del proyecto y la responsable del WP1 del proyecto DOGO4ML, se determinaron algunos cambios necesarios a realizar en el modelo. Estos cambios se determinaron al pensar en cómo los usuarios de la nueva herramienta la usarían para proveerla con datos. Junto con los directores del proyecto y la responsable del WP1 del proyecto DOGO4ML, se fueron comentando estos detalles y se fueron acordando entre todos los cambios para que tanto el modelo como la base de datos implementada a partir de dicho modelo fueran adecuados para la herramienta a desarrollar.

Como se puede observar en la Figura 3, el componente central del modelo UML es el method chunk que representa un componente de un proceso software. A partir del method chunk salen diversas relaciones hacia el resto de componentes del modelo. En primer lugar, están los diferentes tipos de elementos que forman un chunk: tool, artefact, activity y role. Los method chunks, pueden estar relacionados entre ellos mediante tres tipos de relaciones. Las activities son el elemento principal de un chunk y de las relaciones entre actividades se derivan las relaciones entre los method chunks que tienen dichas activities. En segundo lugar, un method chunk también tiene un goal, es decir un objetivo que cumplir, y que puede ser compartido por diversos chunks. Por último, los criteria y sus values indican cómo valorar los criterios a tener en cuenta para cada method chunk.

Aparte del diagrama del modelo, también podemos ver las restricciones textuales que no se pueden implementar de forma gráfica. En primer lugar, estarían las claves externas, que indican que propiedad es la que identifica cada instancia. Las siguientes seis restricciones hacen referencia a la relación entre method elements MERel y sus especificaciones. En primer lugar, dos method elements sólo puede estar relacionado con otro del mismo tipo, además, las relaciones ActivityRel y ArtefactRel, como bien indican sus nombres, sólo pueden ser entre activities y artefacts respectivamente. A continuación, indica que las relaciones de especificación sólo pueden darse si el method element ‘padre’ es abstracto y, además, las relaciones MEStructRel no pueden ser cíclicas en ninguno de los dos casos. La siguiente restricción evita asignar valores de otro criterion a un method chunk que no tiene

ese criterion. El último de todos representa la clave débil de value, el cual puede estar repetido pero no para un mismo chunk.

Al final también se incluye una observación de la razón por la cual el atributo abstract? no se deriva y finalmente se describen las reglas de derivación para ChunkRel.

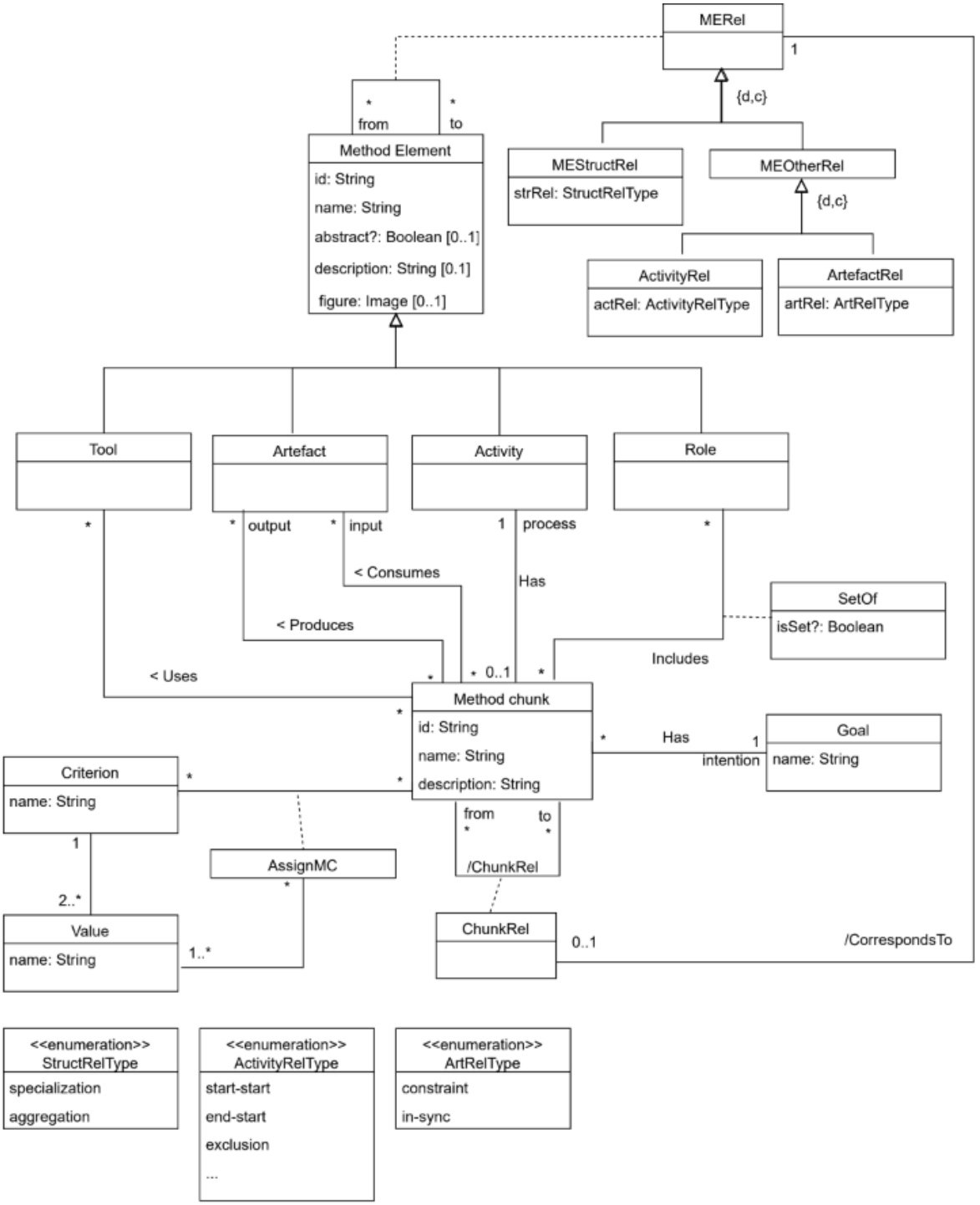


Figura 3: Modelo UML. Fuentes: [19][20]

Restricciones textuales:
1- External keys: (MethodChunk, id), (MethodElement, id), (Criterion, name), (Goal, name)
2- MERel can only associate method elements belonging to the same type
3- ActivityRel can only associate activity method elements
4- ArtefactRel can only associate artefact method elements
5- In a MEStructRel such that rel=specialization the method element at the “to” role must be abstract (abstract? must be true)
6- MEStructRel such that rel=specialization must be acyclic
7- MEStructRel such that rel=aggregation must be acyclic
8- An instance ‘amc’ of AssignMC can only be associated to values that are associated to its criterion (the criterion of ‘amc’)
9- A criterion cannot have two values with the same name

Observaciones:
The attribute abstract? of MethodElement is not derived because there may exist abstract method elements temporally without specializations.

Reglas de derivación:
1- The derived association /ChunkRel associates two method chunks x (with role from) and y (with role to) if and only if the association MERel associates the activity ax of method chunk x (with role from) and the activity ay of method chunk y (with role to).
2- The derived association /CorrespondsTo associates a /ChunkRel cr and a MERel mr if and only if cr associates two method chunks x (with role from) and y (with role to) and mr associates the activity ax of method chunk x (with role from) and the activity ay of method chunk y (with role to).

4.3 Requisitos no funcionales

Los requisitos no funcionales definen cualidades, restricciones y características de la herramienta a desarrollar. Pueden variar desde definir cómo debe ser el aspecto de la interfaz de la nueva herramienta, a establecer cuánto tiempo máximo debe tardar la nueva herramienta en realizar cierta acción. Para la herramienta que se desarrolla en este TFG se han establecido los siguientes requisitos no funcionales:

- **USB - Usabilidad:** La aplicación debe ser intuitiva y fácil de usar, los formularios de entrada de datos serán claros e indicarán en cada caso a qué campo pertenecen. También indicará al usuario de los errores que pueda estar cometiendo a la hora de crear los chunks.
- **EXT - Extensibilidad:** La aplicación debe ser fácilmente extensible, por lo tanto, la estructura del código deberá ser fácil de entender y de extender.
- **POR - Portabilidad:** La aplicación debe ser fácilmente desplegable y en caso de tener que hacer mantenimiento que se pueda desmontar y volver a montar de nuevo. Por ello se deberá usar la tecnología Docker para el despliegue de la aplicación.
- **DES - Desplegado:** La aplicación ha de quedar desplegada en un servidor del grupo GESSI para que la responsable del WP1 pueda hacer pruebas de aceptación y dar el visto bueno a la aplicación.

5. Arquitectura del sistema

La aplicación desarrollada en este proyecto está formada por diferentes partes, cada una con sus propias características y especificaciones que se describen en esta sección. La arquitectura de este proyecto consta, desde el nivel más bajo, de una base de datos donde se almacena la información, una API REST [5] y una aplicación cliente. Estos elementos se comunican mediante los protocolos o conexiones correspondientes según sea necesario y en conjunto proporcionan las funcionalidades necesarias para satisfacer los requisitos del proyecto.

5.1 Arquitectura física

Como se puede observar en la Figura 4, la arquitectura de la aplicación consta de tres componentes principales almacenados en un servidor conectado a internet. Por lo tanto, para que el usuario pueda utilizar la aplicación, lo único que necesita es un dispositivo con acceso a internet. De esta forma accediendo a la dirección correcta el servidor proveerá al dispositivo del usuario los archivos necesarios para la ejecución de la aplicación cliente.

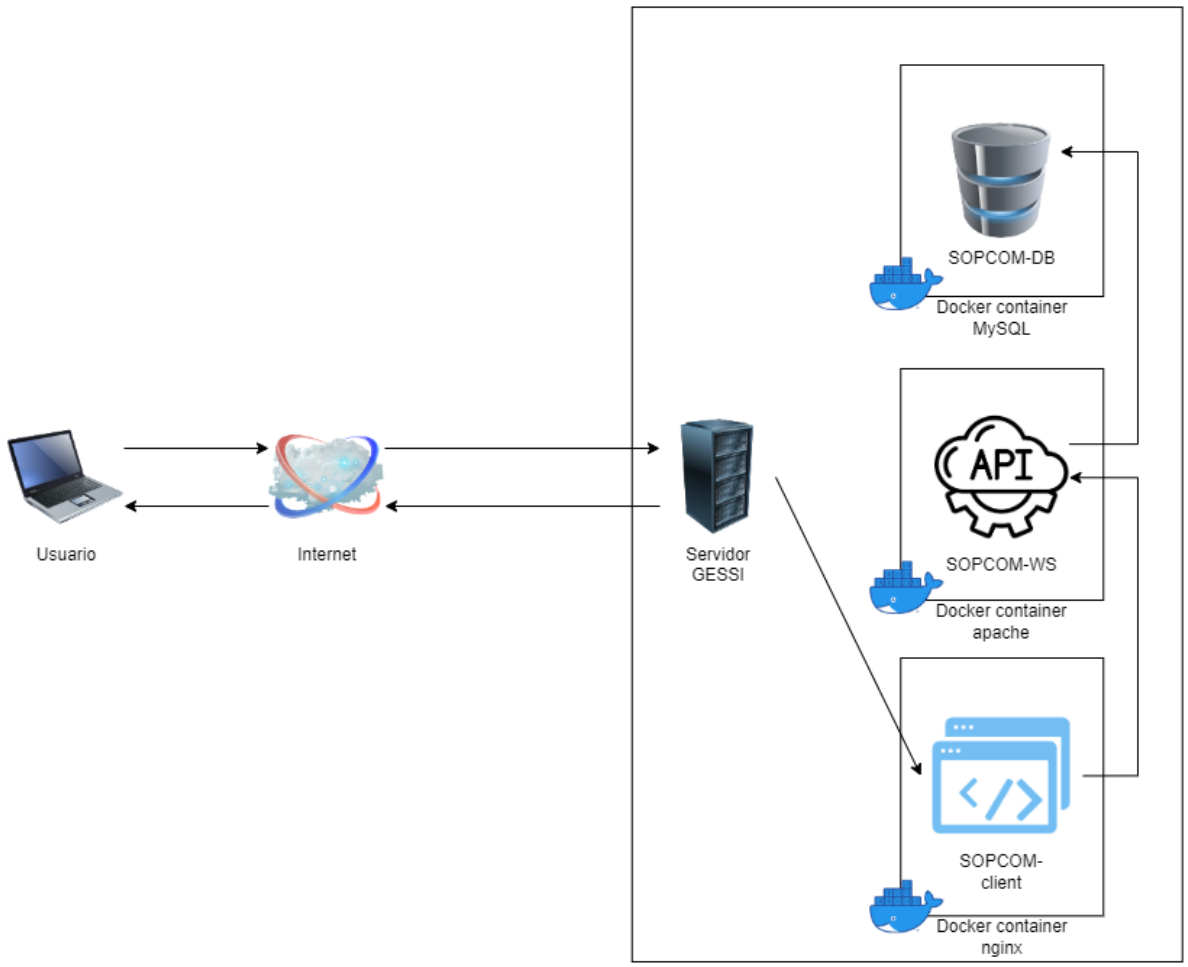


Figura 4: Arquitectura del sistema. Fuente: Elaboración propia.

En lo referente a la parte servidor, este contiene los tres componentes del proyecto: la aplicación cliente, el servicio web, y la base de datos. Cada uno de estos componentes está encapsulado en un contenedor utilizando Docker. Este software lo que permite es aislar el código, sus dependencias y la ejecución de este en unas cápsulas llamadas contenedores, los cuales se ejecutan de forma independiente al sistema operativo que maneja el dispositivo [28]. Se pueden entender estos contenedores como máquinas virtuales dentro del servidor que proveen un nuevo nivel de abstracción en la ejecución de los componentes de la aplicación. Estos contenedores utilizan imágenes que son paquetes que ya contienen todos los archivos necesarios para la configuración y ejecución del contenedor.

Las imágenes que usan cada uno de estos contenedores son diferentes ya que en cada caso el código que contienen y han de ejecutar es distinto. A continuación se explican las particularidades de cada contenedor.

- **SOPCOM-client:** Utiliza una imagen oficial del servidor llamado Nginx [32], el cual es un servidor de código abierto para llamadas HTTP, entre otras, y que permite aceptar múltiples tipos de sistemas operativos diferentes.

```
FROM nginx:1.17.1-alpine

COPY /nginx.conf /etc/nginx/nginx.conf
COPY /dist/sopcom-frontend /usr/share/nginx/html
```

Fragmento de código 1: Dockerfile de la aplicación cliente. Fuente: elaboración propia.

Como se puede ver en el Fragmento de código 1, el servidor es Nginx y el sistema operativo que se ejecuta en él, es Alpine [33], una distribución de Linux muy usada en estos casos debido al poco espacio que ocupa**.**

- **SOPCOM-WS:** En este caso se usa la imagen oficial de PHP con un servidor Apache para PHP [34], el cual usa como sistema operativo Debian, otra distribución de Linux.

```
FROM php:8.1.5-apache
RUN docker-php-ext-install mysqli && docker-php-ext-enable mysqli

COPY src/ /var/www/html/

RUN echo "ServerName localhost" >> /etc/apache2/apache2.conf
CMD ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
```

Fragmento de código 2: Dockerfile de SOPCOM-WS. Fuente: Elaboración propia.

Dadas las características del lenguaje PHP, que se explican en profundidad más adelante en el apartado de implementación, también necesita instalar el intérprete para procesar el código y en el caso de este proyecto instalar la extensión MySQLi de PHP [35].

- **SOPCOM-DB:** Por último, para la base de datos se usa la imagen oficial de Docker para MySQL, la cual añade los archivos necesarios para el sistema de gestión de base de datos.

```
mysql:
    container_name: sopcom-mysql
    image: mysql
    restart: on-failure
    ports:
      - "1025:3306"
    networks:
      - database
    environment:
      MYSQL_DATABASE: sopcom_mysql
      MYSQL_USER: admin
      MYSQL_PASSWORD: password
      MYSQL_ROOT_PASSWORD: example
    volumes:
      - ./mysql:/docker-entrypoint-initdb.d
```

Fragmento de código 3: Fragmento de docker-compose.yaml para mysql. Fuente: Elaboración propia

En este caso no era necesario utilizar un Dockerfile específico para este contenedor ya que todo lo necesario se puede indicar desde el archivo docker-compose, el cual se explica en el apartado de implementación.

También se propuso el uso de un cuarto contenedor que tuviese una imagen de phpMyAdmin [24] para poder acceder a SOPCOM-DB con una interfaz fácil de usar y desde el propio navegador. Al final se acabó descartando por el hecho de que no se vió la necesidad real de acceder directamente a la base de datos en este momento y de esta manera se evitaba la presencia de otro contenedor, lo que supondría un mayor uso del espacio del servidor. Además, para acceder directamente a la base de datos ya existen otras alternativas en forma de aplicación de escritorio como MySQL Workbench [36].

El uso de Docker para esta aplicación responde los requisitos EXT y POR, explicados en el apartado de requisitos no funcionales, ya que gracias a las funcionalidades que ofrece este software, se puede trabajar con los diferentes contenedores de forma independiente y rápida. Con unos pocos comandos se puede parar la ejecución de los contenedores, modificarlos y volverlos a poner en funcionamiento sin alterar el funcionamiento del resto. Por esa razón los tres elementos están separados en contenedores diferentes, para que en caso de que en el futuro se tenga que expandir la aplicación, se pueda hacer de forma fácil y rápida para cada componente sin afectar al comportamiento de los otros.

5.2 Arquitectura lógica

A continuación, se explicará la arquitectura lógica de los componentes del proyecto y los elementos que forman cada uno de estos componentes.

5.2.1 SOPCOM-client

Es la parte encargada de la visualización de los datos y se ejecuta en el dispositivo del usuario. Por tanto, pertenece a la capa de presentación de la aplicación. En la Figura 5 se pueden ver los elementos que forman parte de esta capa.

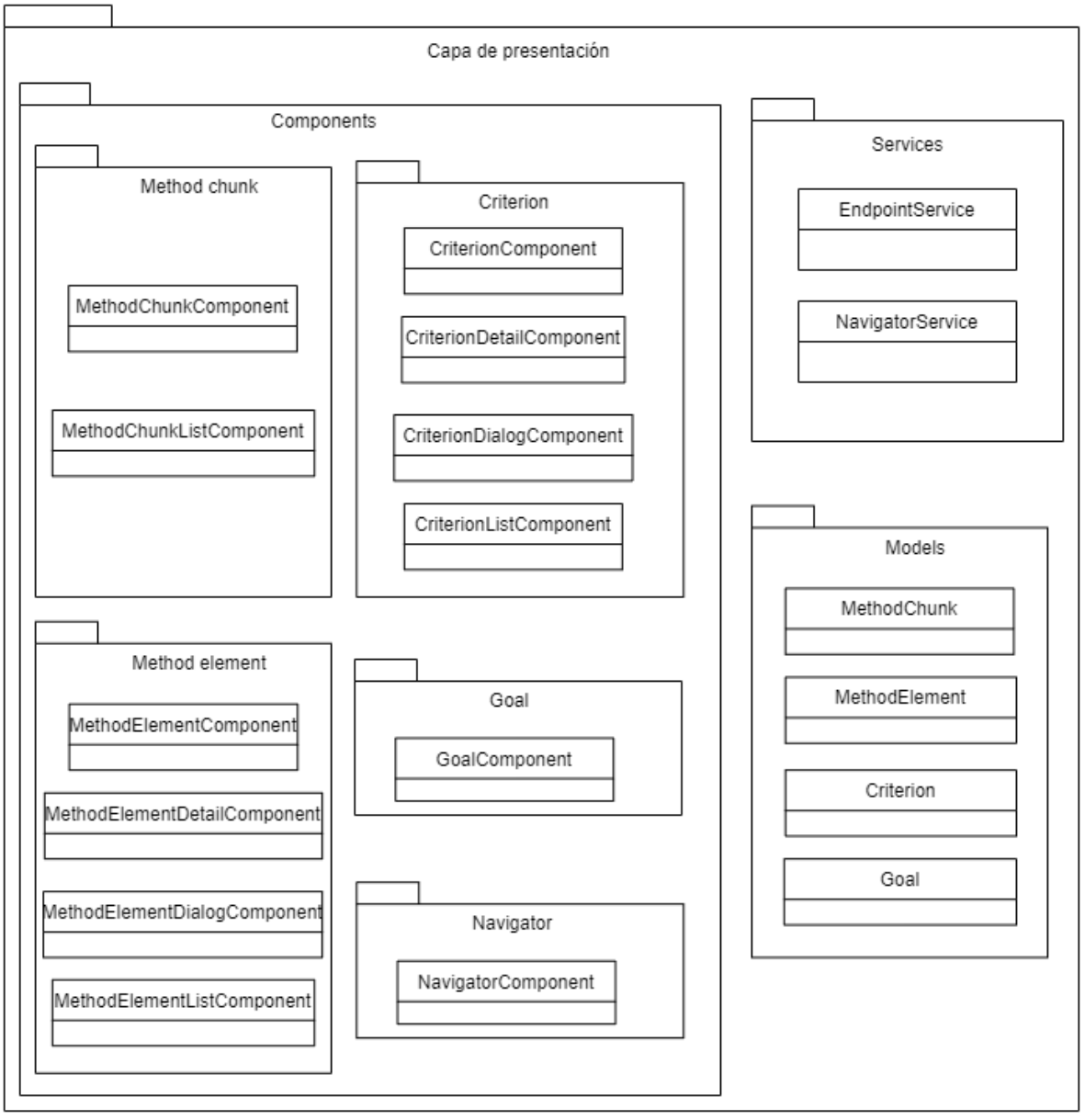


Figura 5: Arquitectura lógica de SOPCOM-client. Fuente: Elaboración propia.

El cliente SOPCOM, como se ve en la figura, se estructura en: Components, Models y Services.

Los componentes forman una parte visible de la pantalla. Están formados por archivos Typescript, HTML y CSS y se pueden tener diversos cargados al mismo tiempo. Algunos de estos tienen diversos subcomponentes, por ejemplo method element tiene:

- **MethodElementComponent:** Contiene la vista principal del method element y su formulario.
- **MethodElementDetailComponent**:Contiene la vista de un method element en una página.
- **MethodElementDialogComponent:** Contiene la vista de un method element en un cuadro de diálogo.
- **MethodElementListComponent:** Contiene la vista de diversos method elements, sin embargo este componente se acabó descartando.

Los modelos simplemente son representaciones de los elementos del sistema, y facilitan las funciones de traducción de objetos a formato JSON.

Por último, están los servicios, que son unas clases instanciadas una única vez y que son accesibles desde cualquier punto del código. El servicio a destacar de esta capa es el EndpointService ya que es el encargado de comunicarse con la capa de dominio.

5.2.2 SOPCOM-WS

Este componente contiene la capa de dominio de la aplicación la cual se comunica con la de presentación mediante el protocolo HTTP. En la Figura 6 se puede ver los elementos que forman parte de esta capa.

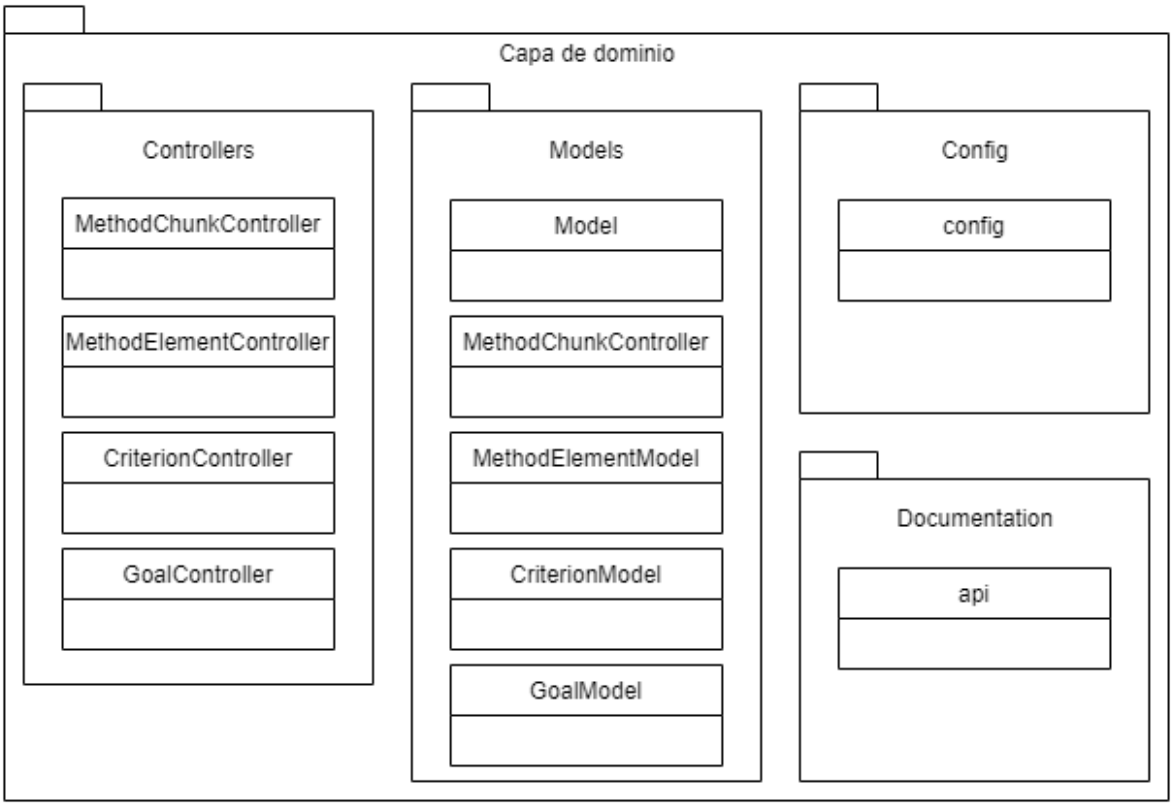


Figura 6: Arquitectura lógica de SOPCOM-WS. Fuente: Elaboración propia.

La implementación de los servicios web de SOPCOM, como se ve en la figura, se estructura en: Controllers, Models, configuration y documentation.

Los dos elementos más simples son los de Config y Documentation, que contienen información de la configuración de algunos elementos del sistema como la base de datos en el caso del primero y la documentación de los endpoints de la aplicación en el segundo.

A continuación, los elementos más importantes de esta capa son los controladores y los modelos. Los primeros son los que reciben de la capa de presentación las solicitudes HTTP y le comunican al modelo correspondiente que operaciones y con qué datos hay que realizar. El componente Model contiene los aspectos comunes de todos los modelos y uno de ellos es la conexión con la capa de persistencia.

La capa de dominio también se encarga de comunicar los cambios efectuados a la capa de presentación para que ésta actualice las vistas correspondientes.

5.2.3 SOPCOM–DB

Este último componente contiene la capa de persistencia y su único elemento es el sistema de gestión de bases de datos. A esta capa se accede mediante una conexión TCP/IP desde el Modelo en la capa de dominio.

5.3 Patrones utilizados

La arquitectura de este proyecto se ha desarrollado siguiendo el patrón de diseño MVC o Modelo-Vista-Controlador [21]. El objetivo de este patrón es el de separar la lógica de la aplicación de su visualización utilizando los tres conceptos que dan nombre al patrón.

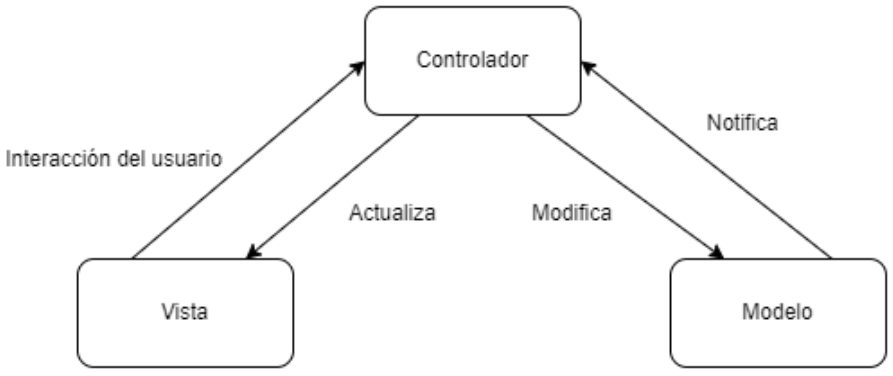


Figura 7: Diagrama MVC. Fuente: Elaboración propia.

En primer lugar, el modelo representa cada elemento del sistema y es el que contiene toda la lógica de dicho elemento. En el caso de esta aplicación gestiona los datos y se comunica con la base de datos.

La vista es la representación gráfica del elemento en cuestión y gestiona todas las interacciones del usuario y le envía la información al controlador. Por ejemplo, en esta

aplicación al hacer clic en guardar un elemento del sistema, la vista recoge los datos que están presentes y se los envía al controlador junto con la petición de guardarlo

Por último, el controlador es el encargado de la comunicación entre la vista de un elemento y el modelo del mismo. En esta aplicación cuando una llamada HTTP llega al servidor, dependiendo del recurso al que se quiera acceder, se llama al controlador de dicho recurso y este le indica al modelo que operación ha de realizar.

5.4 Ejemplos de diagramas de secuencia

En este apartado se presentan los diagramas de secuencia de dos endpoints de la aplicación para mostrar la forma en que se comunican los diferentes componentes del sistema. Las dos operaciones que se describirán a continuación, corresponden a la consulta de un method chunk y la creación de un criterio.

Consulta de un method chunk

La petición en la parte cliente (SOPCOM-client) se puede observar en la Figura 8 y comienza con la llamada a la función ngOnInit de un MethodChunkComponent. Esta llamada es característica del ciclo de vida de los componentes de angular ya que es la función que se llama nada más crear el componente y que hasta que no acabe no empieza a cargar el contenido del HTML. Pero lo que importa de esta llamada es la parte en la que se realiza la petición al servicio web para la consulta del method chunk.

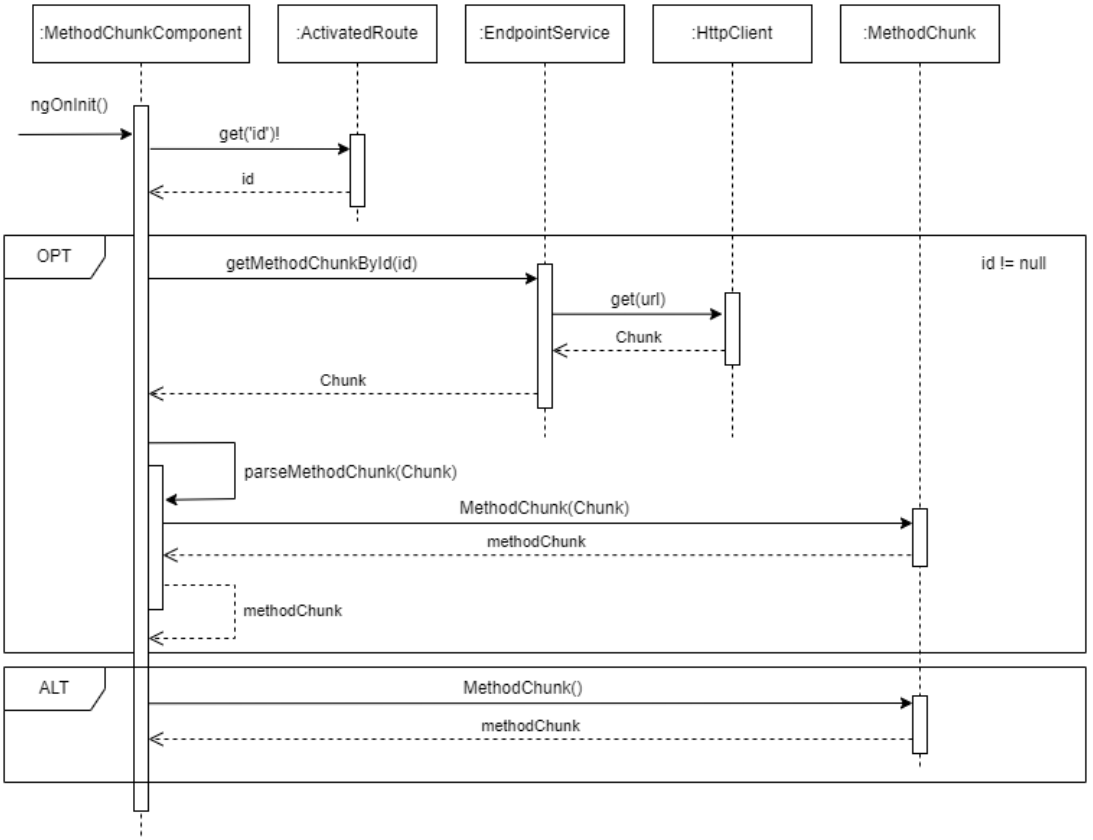


Figura 8: Diagrama de secuencia para la obtención de un method chunk en frontend.
Fuente: Elaboración propia.

Lo primero que se hace al invocar la función ngOnInit es leer el id que está en la url mediante la clase ActivatedRoute, la cual se encarga de gestionar la url asociada al componente actual [37] y las navegaciones dentro de la propia aplicación.

Después de obtener el id se comprobaría si este id es nulo o no, ya que, como se explicará más adelante, el mismo componente se usa para la creación y la modificación de los elementos del modelo conceptual. En este apartado el caso que importa es aquel donde el id no es nulo y, por tanto, se debe realizar una llamada a SOPCOM-WS para que devuelva toda la información del method chunk. En la figura 9 se puede ver la llamada al endpoint para obtener un method chunk que se ejecuta en SOPCOM-WS, donde primero se invoca la función del controlador de un method chunk para la obtención del chunk identificado por el id que se pasa como parámetro.

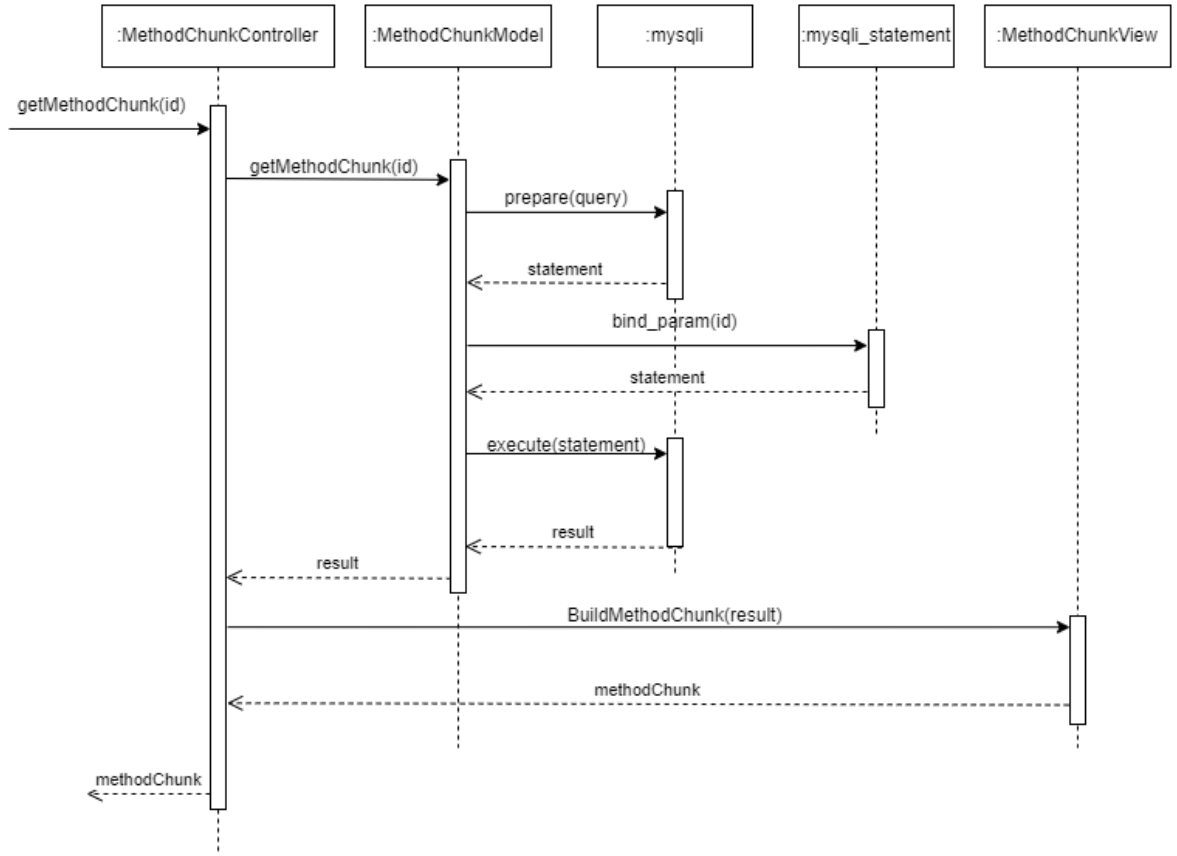


Figura 9: Diagrama de secuencia para la obtención de un method chunk en backend.
Fuente: Elaboración propia.

En este caso, lo que hace el controlador es llamar a la función de MethodChunkModel para obtener el la información del chunk con el id enviado. Dentro del modelo se utiliza la llamada prepare de la extensión de PHP MySQLi [35] que se explica en profundidad en el apartado de implementación. Lo que hace MySQLi es abrir una conexión con la base de datos e introducir la sentencia que se ha pasado como parámetro a la función prepare y esta devuelve una instancia de la clase mysqli_stmt la cual tiene asociada la query anterior. A continuación, se llama a la función bind_param, que cambia los carácteres '?' por las

variables que se le pasen como parámetro, en este caso el id. Por último, se llama a la función execute de MySQLi y ésta retorna los resultados obtenidos al ejecutar la sentencia en la base de datos.

El resultado de la función de obtener el chunk de MethodChunkModel contiene toda la información del chunk y se envía a la función buildMethodChunk de la clase MethodChunkView cuya función es preparar el array asociativo que se enviará en formato JSON como cuerpo de la respuesta de la solicitud HTTP y, a continuación, se envía la respuesta a la aplicación cliente.

Este ejemplo está simplificado ya que simplemente trata de explicar todas las conexiones entre los diferentes componentes de la aplicación más que la ejecución completa de la función. En el caso real se realizan más llamadas a la clase MySQLi para ejecutar las sentencias que recogen las relaciones del method chunk y la función que devuelve el JSON es el encargado también de juntar todos estos elementos.

Insert de un criterion

Se ha escogido explicar este ejemplo de inserción ya que todos siguen la misma estructura y este es el más simple de todos. Por lo que, para la explicación de cómo sucede un insert, éste es el que mejor se verá.

En primer lugar, se tiene que invocar la función en la aplicación cliente para guardar el formulario del criterion. En la Figura 10 se puede ver como se invoca la llamada en el componente del Criterion. Lo primero que se hace es comprobar si los datos que se han introducido son correctos, por ejemplo en este caso no se puede introducir el nombre de un criterion que ya existe ya que estaría rompiendo la restricción RT1 del modelo conceptual. En caso de error se llamaría a la función displayErrorMessage que se encargaría de informar al usuario de un error en los datos introducidos.

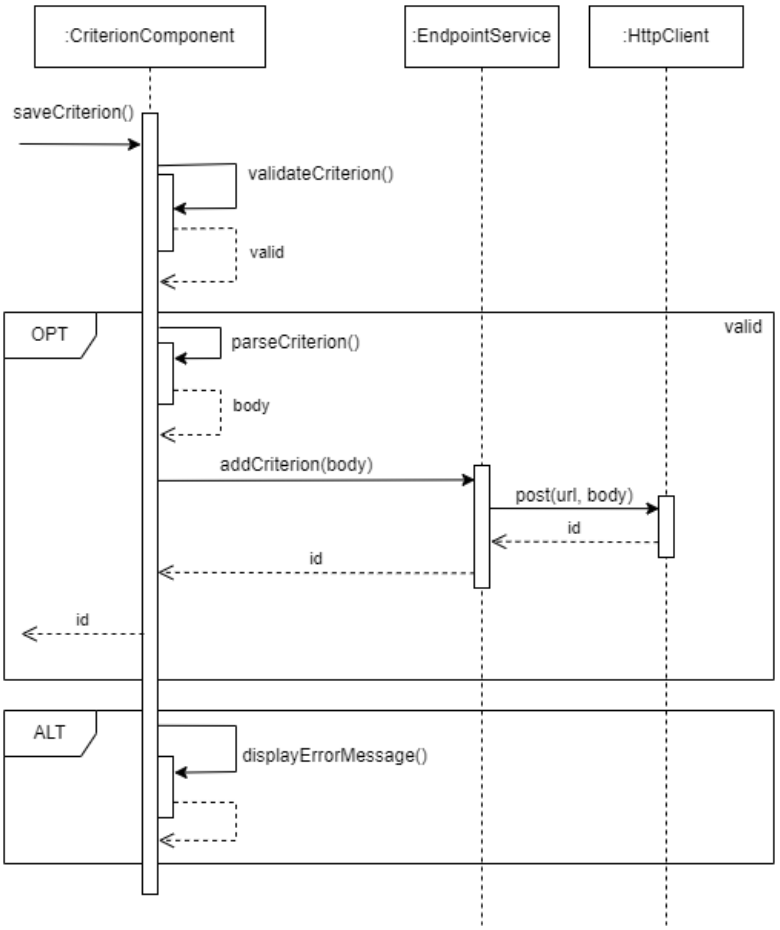


Figura 10: Diagrama de secuencia para la inserción de un criterion en frontend. Fuente: Elaboración propia.

En caso de que el formulario sea válido se prepara para enviar los datos a la API mediante el método parseCriterion, que crea el JSON que se enviará como cuerpo de la llamada HTTP. Como en el caso anterior, se llama a la función de añadir criterion del servicio de endpoints que a la vez llama el método post de la clase HttpClient con la url y el body. Esta clase se encarga de hacer la petición a SOPCOM-WS y después recibe el id del nuevo criterion insertado.

En la Figura 11 se puede encontrar el diagrama de secuencia de las funciones que se invocan al recibir SOPCOM-WS la solicitud de insertar un criterion. En primer lugar, se vuelve a comprobar la validez del criterion ya que la API no tiene porqué ser siempre llamada desde la aplicación cliente de este proyecto y la comprobación en el frontend sirve más para ahorrar la llamada a la SOPCOM-WS.

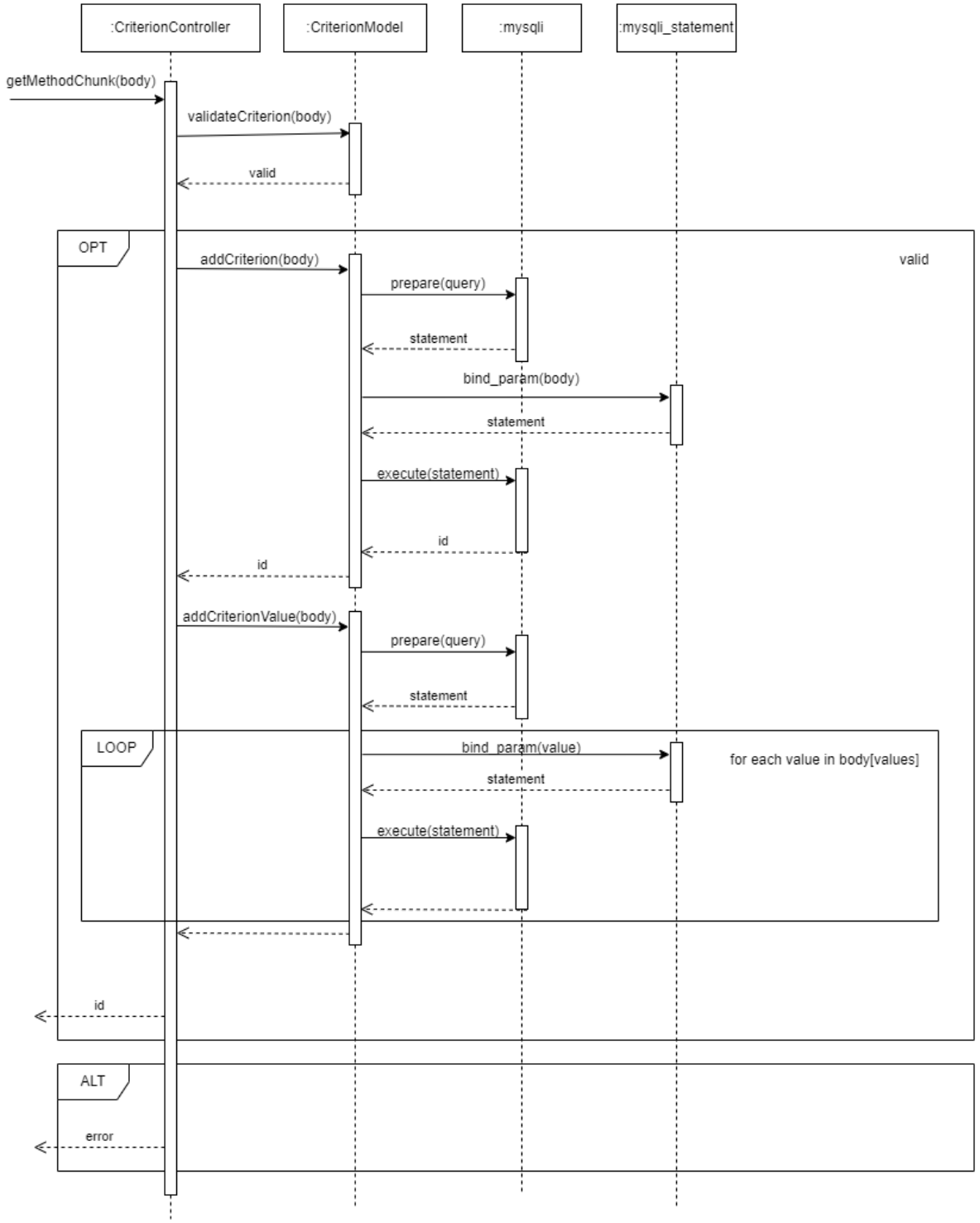


Figura 11: Diagrama de secuencia para la inserción de un criterion en backend. Fuente: Elaboración propia.

Una vez comprobada la validez de los datos, se llama a la función de añadir el criterion del modelo. Esta hace como en el caso anterior: primero abre la conexión con la base de datos con la función prepare de MySQLi, a continuación, asigna las variables con la función bind_param de mysqli_stmt y por último, llama a la función execute. En este caso, al ser una inserción, la función insert también retorna el id del nuevo registro de la base de datos.

El siguiente paso es llamar a la función del modelo para añadir los valores que se han enviado, por lo que se llama a la función addCriterionValue. En este momento se puede ver otra de las particularidades de la extensión de MySQLi que se explica más adelante, y es que cuando se ha de ejecutar sucesivamente la misma sentencia con valores distintos, no hace falta llamar a la función prepare para que devuelva el statement, sino que se puede llamar una vez e iterar sobre esa instancia de mysqli_stmt cambiando los valores con la función bind_param y después llamar a la función execute. Este proceso se repite por cada valor y al cuando acaba retorna como cuerpo de la respuesta HTTP el id del nuevo criterion añadido.

5.5 Bases de datos

La aplicación utiliza una única base de datos relacional donde se almacenan todos los componentes. El sistema de gestión utilizado es MySQL [22] debido a varias razones. En primer lugar es el sistema de gestión más utilizado junto con PHP [23], por lo que encaja perfectamente para desarrollar la implementación de la API que dará acceso a los datos. En segundo lugar porque el uso de una base de datos relacional permitirá definir muchas de las restricciones establecidas en el modelo conceptual de datos en el mismo esquema de la base de datos. Por último, por mi experiencia previa con esta tecnología.

La base de datos fue la primera parte de la aplicación que se completó, pero durante las primeras semanas de desarrollo del proyecto fue sufriendo cambios debido a errores que se fueron encontrando en el modelo conceptual y que se explicaran en los siguientes apartados.

5.5.1 Diseño lógico

El diseño lógico de la base de datos es acorde al modelo conceptual de datos UML de la Figura 3. En el fragmento de código 4 se puede ver el diseño lógico siguiendo el formato siguiente: nombre_de_la_tabla(propiedad 1, propiedad 2, …) y, a continuación, las claves foráneas y restricciones de tabla. La propiedad o propiedades subrayadas representan las claves primarias de dicha tabla.

```
MethodElement(id, name, abstract, description, figure, type)
{type} CHECK (type IN ‘tool’, ‘artefact’, ‘activity’, ‘role’)

Tool(id)
{id} references MethodElement

Artefact(id)
{id} references MethodElement

Activity(id)
{id} references MethodElement

Role(id)
{id} references MethodElement
```

```
MeRel(from, to)
{from} references MethodElement
{to} references MethodElement

MeStructRel(from, to, rel)
{from, to} references MeRel
{rel} CHECK (rel IN ‘specialization’, ‘aggregation’)

ActivityRel(from, to, rel)
{from} references Activity
{to} references Activity
{from, to} references MeRel
{rel} CHECK (rel IN ‘start-start’, ‘end-start’, ‘exclusion’)

ArtefactRel(from, to, rel)
{from} references Artefact
{to} references Artefact
{from, to} references MeRel
{rel} CHECK (rel IN ‘constraint’, ‘in-sync’)

Goal(id, name)
UNIQUE(name)

Criterion(id, name)
UNIQUE(name)

Value(id, name, criterion)
{criterion} references Criterion
UNIQUE(name, criterion)

AssignMC(idMC, criterion)
{idMC} references MethodChunk
{criterion} references Criterion

AssignMCValue(value, idMC, criterion)
{idMC, criterion} references AssignMC
{value} references Value

MethodChunk(id, name, description, activity, intention)
{activity} references Activity
{intention} references Goal
UNIQUE(activity)

MethodChunkUsesTool(idMC, idME)
{idMC} references MethodChunk
```

```
{idME} references Tool

MethodChunkProducesArtefact(idMC, idME)
{idMC} references MethodChunk
{idME} references Artefact

MethodChunkConsumesArtefact(idMC, idME)
{idMC} references MethodChunk
{idME} references Artefact

MethodChunkIncludesRole(idMC, idME, isSet)
{idMC} references MethodChunk
{idME} references Role

ChunkRel(fromMC, toMC, fromME, toME)
{fromMC} references MethodChunk
{toMC} references MethodChunk
{fromME, toME} references MeREl
```

Fragmento de código 4: Diseño lógico de la base de datos. Fuente: Elaboración propia.

Del diseño lógico es importante comentar una serie de tablas con aspectos importantes respecto al modelo conceptual y que han sido los que más evolucionaron durante las primeras semanas de desarrollo.

En primer lugar, como se puede observar el Fragmento de código 4, tanto las relaciones como la información derivada del UML también se guardarán en la base de datos para que sea más fácil y rápido acceder a esta información y no se tenga que calcular cada vez que se necesite obtener esta información.

También hay que destacar la presencia de dos especializaciones o generalizaciones que corresponden a los componentes: Method Element y MERel. Por las características de cada una, sus relaciones y las restricciones textuales del modelo, se ha tenido que pensar bien la forma de trasladar estas especializaciones a la base de datos a la vez que se cumplían las restricciones.

A la hora de diseñar una especialización en una base de datos se pueden adoptar cuatro estratégias diferentes:

- **Class Table Inheritance:** Crear una tabla para la superclase y para cada subclase. Las subclases usan una clave foránea que hace referencia a la superclase y de esta forma si cada subclase tiene relaciones diferentes puedes usar las tablas de cada una.
- **Concrete table Inheritance:** Crear una tabla para cada subclase y ninguna para la superclase. De este modo las subclases tendrían todas las columnas heredadas de la superclase.

- **Single table inheritance:** Crear una tabla únicamente para la superclase con una columna que indique la subclase a la que pertenece. De esta forma si la superclase es la que tiene todas las relaciones evitas crear las tablas de las subclases.
- **Mixta:** Utilizar una mezcla de los enfoques anteriores. Por lo que existirían una tabla para la superclase, con una columna que indique el tipo, y una para cada subclase.

En el caso de la tabla MethodElement hay una restricción textual que indica que una relación (MERel) entre dos method elements sólo puede darse entre dos del mismo tipo. Con esta restricción cualquiera de los tres métodos sería válido y quizás el tercero sería el más redundante al tener el campo type a la vez que las subtablas para cada tipo.

Pero todo cambia con la generalización de la tabla MERel, donde utilizar la tercera opción, es decir usando una única tabla, implicaría tener que declarar muchas columnas ya que cada subtipo de relación tiene un campo que hace referencia a valores de diferentes enumeraciones de valores y a la vez cada registro tendría dos valores nulos que no se utilizarían ya que no corresponden al tipo de relación de la subclase. La opción de concrete table inheritance funciona pero la tabla que corresponde a MEStructRel tendría claves foráneas a cuatro tablas distintas y, además, no se podría comprobar la restricción de clave primaria de la superclase. Por último, la opción mixta en este caso solo daría información redundante ya que la superclase no necesitaría esa información.

Por lo tanto, para la tabla MERel y sus especificaciones la opción usada ha sido class table inheritance. Esto provoca que el diseño de la tabla MethodElement cambie. En primer lugar, la opción de concrete class inheritance no nos interesa ya que se pierde la comprobación de clave primaria de method element y a la vez provocaría muchas claves foráneas a cada tabla. La opción single class inheritance podría ser correcta pero las restricciones que hacen referencia a que las relaciones entre artefacts sólo pueden darse entre dos method elements que sean del tipo artefact y lo mismo pero con el tipo activity. Por lo que me animaría a usar class table inheritance también, pero las relaciones estructurales pueden darse entre dos method elementos del tipo que sean siempre que sean del mismo y teniendo en cuenta la estratégia utilizada en las relaciones, la clave foránea va de la superclase MERel a la superclase MethodElement.

Finalmente llegamos a la opción mixta para los method elements. Utilizando esta estrategia podemos relacionar con una clave foránea MERel y MethodElement, a la vez que se unen la tabla de relaciones entre artefacts con clave primaria a la subclase de artefact y lo mismo con activity. De esta forma la única relación que no hemos podido implementar en la base de datos con las restricciones de tablas ha sido la de que dos method elements relacionados estructuralmente han de ser del mismo tipo.

Otra tabla interesante es la tabla Value, la cual tiene una restricción UNIQUE, es decir que no pueden haber dos registros con valores iguales en las columnas indicadas. En este caso son las columnas name y criterion, las cuales representan el nombre del valor y el criterion al cual está asignado este valor. Esto se debe a la restricción textual 9, un criterio no puede tener dos valores con el mismo nombre, es decir una clave débil de Value del modelo conceptual. Esto permitirá que puedan existir dos valores con el mismo nombre siempre que estén asignados a criterios diferentes.

El uso de la cláusula UNIQUE mencionado antes no es el único que se ha planteado en el diseño lógico. Otras tablas como Goal o Criterion también tienen esta restricción de tabla, solo que en este caso tiene el propósito de evitar la repetición de dos elementos de estos con el mismo nombre ya que en el modelo conceptual se indica que en ambos casos el nombre es la clave primaria. Pero se planteó que tener tablas con claves primarias de tipo string, y que no fuesen identificadores claros como podrían ser el número de DNI o un email, no era una buena práctica a nivel de diseño de la base de datos y se propuso añadir una columna llamada id para identificar a los registros de estas tablas.

El último aspecto a destacar es otro UNIQUE, esta vez en la tabla MethodChunk, que tiene como objetivo cumplir la restricción de la relación entre method chunk y activity del modelo conceptual en la que una activity solamente puede estar relacionada a un method chunk.

5.5.2 Diseño físico

A partir del diseño lógico a continuación, en el Fragmento de código 5 se puede encontrar el script sql para la creación de tablas de la base de datos que usa la aplicación.

```
CREATE TABLE method_element_type (
    id INT AUTO_INCREMENT,
    name VARCHAR(50) UNIQUE,
    PRIMARY KEY (id)
);

CREATE TABLE method_element (
    id VARCHAR(50),
    name VARCHAR(100) NOT NULL,
    abstract BIT(1) NOT NULL,
    description VARCHAR(200),
    figure VARCHAR(100),
    type INT,
    PRIMARY KEY (id),
    FOREIGN KEY (type)
        REFERENCES method_element_type(id)
        ON DELETE NO ACTION
);

CREATE TABLE tool (
    id VARCHAR(50),
    PRIMARY KEY (id),
    FOREIGN KEY (id)
        REFERENCES method_element(id)
        ON DELETE CASCADE
);

CREATE TABLE artefact (
    id VARCHAR(50),
    PRIMARY KEY (id),
    FOREIGN KEY (id)
        REFERENCES method_element(id)
```

```
        ON DELETE CASCADE
);

CREATE TABLE activity (
    id VARCHAR(50),
    PRIMARY KEY (id),
    FOREIGN KEY (id)
        REFERENCES method_element(id)
        ON DELETE CASCADE
);

CREATE TABLE role (
    id VARCHAR(50),
    PRIMARY KEY (id),
    FOREIGN KEY (id)
        REFERENCES method_element(id)
        ON DELETE CASCADE
);

CREATE TABLE goal (
    id INT AUTO_INCREMENT,
    name VARCHAR(50) UNIQUE,
    PRIMARY KEY (id)
);

CREATE TABLE method_chunk (
    id VARCHAR(50),
    name VARCHAR(100) NOT NULL,
    description VARCHAR(200) NOT NULL,
    activity VARCHAR(50) UNIQUE,
    intention INT NOT NULL,
    PRIMARY KEY (id),
    FOREIGN KEY (activity)
        REFERENCES activity(id)
        ON DELETE NO ACTION,
    FOREIGN KEY (intention)
        REFERENCES goal(id)
        ON DELETE NO ACTION
);

CREATE TABLE method_chunk_uses_tool(
    idMC VARCHAR(50),
    idME VARCHAR(50),
    PRIMARY KEY (idMC, idME),
    FOREIGN KEY (idMC)
        REFERENCES method_chunk(id)
        ON DELETE CASCADE,
    FOREIGN KEY (idME)
        REFERENCES tool(id)
        ON DELETE CASCADE
);

CREATE TABLE method_chunk_produces_artefact(
```

```
    idMC VARCHAR(50),
    idME VARCHAR(50),
    PRIMARY KEY (idMC, idME),
    FOREIGN KEY (idMC)
        REFERENCES method_chunk(id)
        ON DELETE CASCADE,
    FOREIGN KEY (idME)
        REFERENCES artefact(id)
        ON DELETE CASCADE
);

CREATE TABLE method_chunk_consumes_artefact(
    idMC VARCHAR(50),
    idME VARCHAR(50),
    PRIMARY KEY (idMC, idME),
    FOREIGN KEY (idMC)
        REFERENCES method_chunk(id)
        ON DELETE CASCADE,
    FOREIGN KEY (idME)
        REFERENCES artefact(id)
        ON DELETE CASCADE
);

CREATE TABLE method_chunk_includes_role(
    idMC VARCHAR(50),
    idME VARCHAR(50),
    isSet BIT(1),
    PRIMARY KEY (idMC, idME),
    FOREIGN KEY (idMC)
        REFERENCES method_chunk(id)
        ON DELETE CASCADE,
    FOREIGN KEY (idME)
        REFERENCES role(id)
        ON DELETE CASCADE
);

CREATE TABLE  criterion (
    id INT AUTO_INCREMENT,
    name VARCHAR(100) UNIQUE,
    PRIMARY KEY (id)
);

CREATE TABLE value (
    id INT AUTO_INCREMENT,
    name VARCHAR(50),
    criterion INT,
    UNIQUE (name, criterion),
    PRIMARY KEY (id),
    FOREIGN KEY (criterion)
        REFERENCES criterion(id)
        ON DELETE CASCADE
);
```

```
CREATE TABLE assign_method_chunk (
    idMC VARCHAR(50),
    criterion INT,
    PRIMARY KEY (idMC, criterion),
    FOREIGN KEY (idMC)
        REFERENCES method_chunk(id)
        ON DELETE CASCADE,
    FOREIGN KEY (criterion)
        REFERENCES criterion(id)
        ON DELETE CASCADE
);

CREATE TABLE assign_method_chunk_value (
    idMC VARCHAR(50),
    criterion INT REFERENCES Criterion,
    value INT REFERENCES Value,
    PRIMARY KEY(idMC, criterion, value),
    FOREIGN KEY (idMC, criterion)
        REFERENCES assign_method_chunk(idMC, criterion)
        ON DELETE CASCADE,
    FOREIGN KEY (value)
        REFERENCES value(id)
        ON DELETE CASCADE
);

CREATE TABLE struct_rel_type (
    id INT AUTO_INCREMENT,
    name VARCHAR(100) UNIQUE,
    PRIMARY KEY(id)
);

CREATE TABLE activity_rel_type (
    id INT AUTO_INCREMENT,
    name VARCHAR(100) UNIQUE,
    PRIMARY KEY(id)
);

CREATE TABLE artefact_rel_type (
    id INT AUTO_INCREMENT,
    name VARCHAR(100) UNIQUE,
     PRIMARY KEY(id)
);

CREATE TABLE me_rel (
    fromME VARCHAR(50),
    toME VARCHAR(50),
    PRIMARY KEY (fromME, toME),
    FOREIGN KEY (fromME)
        REFERENCES method_element(id)
        ON DELETE CASCADE,
    FOREIGN KEY (toME)
        REFERENCES method_element(id)
        ON DELETE CASCADE
```

```
);

CREATE TABLE me_struct_rel (
    fromME VARCHAR(50),
    toME VARCHAR(50),
    rel INT,
    PRIMARY KEY (fromME, toME),
    FOREIGN KEY (fromME, toME)
        REFERENCES me_rel(fromME, toME)
        ON DELETE CASCADE,
    FOREIGN KEY (rel)
        REFERENCES struct_rel_type(id)
);

CREATE TABLE activity_rel (
    fromME VARCHAR(50),
    toME VARCHAR(50),
    rel INT,
    PRIMARY KEY (fromME, toME),
    FOREIGN KEY (fromME, toME)
        REFERENCES me_rel(fromME, toME)
        ON DELETE CASCADE,
    FOREIGN KEY (rel)
        REFERENCES activity_rel_type(id),
    FOREIGN KEY (fromME)
        REFERENCES activity(id),
    FOREIGN KEY (toME)
        REFERENCES activity(id)
);

CREATE TABLE artefact_rel (
    fromME VARCHAR(50),
    toME VARCHAR(50),
    rel INT,
    PRIMARY KEY (fromME, toME),
    FOREIGN KEY (fromME, toME)
        REFERENCES me_rel(fromME, toME)
        ON DELETE CASCADE,
    FOREIGN KEY (rel)
        REFERENCES artefact_rel_type(id),
    FOREIGN KEY (fromME)
        REFERENCES artefact(id),
    FOREIGN KEY (toME)
        REFERENCES artefact(id)
);

CREATE TABLE chunk_rel (
    fromMC VARCHAR(50),
    toMC VARCHAR(50),
    fromME VARCHAR(50),
    toME VARCHAR(50),
    PRIMARY KEY (fromMC, toMC),
    FOREIGN KEY (fromMC)
```

```
        REFERENCES method_chunk(id)
        ON DELETE CASCADE,
    FOREIGN KEY (toMC)
        REFERENCES method_chunk(id)
        ON DELETE CASCADE,
    FOREIGN KEY (fromME, toME)
        REFERENCES me_rel(fromME, toME)
        ON DELETE CASCADE
);
```

Fragmento de código 5: Diseño físico de la base de datos. Fuente: Elaboración propia.

Como se ha mencionado en el apartado anterior, hay tablas que en el modelo conceptual tienen como clave el nombre del elemento. A nivel de base de datos se ha preferido utilizar identificadores de tipo int a identificadores de tipo varchar por preferencias del autor en estos casos donde el string puede ser muy variable, en cambio para los method chunks que también tienen un identificador de tipo varchar este no es tan variable al ser explícito que ese es el id. Para poder manejar estos ids se ha utilizado un mecanismo de MySQL llamado AUTO_INCREMENT [38]. Este mecanismo lo que permite es añadir, a menos que se indique uno diferente, un id nuevo y único para el registro a añadir. En este caso se usan para las tablas como goal, criterion o value.

En el Fragmento de código 5 también se pueden ver cuatro tablas nuevas respecto al del Fragmento de código 4 correspondiente al diseño lógico. Estas cuatro tablas nuevas son: method_element_type, struct_rel_type, activity_rel_type y artefact_rel_type. La primera se trata de una tabla con los tipos de method elements que hay y las otras tres son las numeraciones del modelo conceptual que indican los tipos de relación entre elementos. Estas tablas se han creado para cumplir en la base de datos el requisito no funcional de extensibilidad, de esta forma si se han de añadir nuevos tipos de relación o de method elements solo haría falta hacer un *insert* en la base de datos y crear una tabla nueva en el segundo caso.

En el diseño físico también podemos encontrar las claves foráneas con la opción de ON DELETE. Esto permite indicarle al sistema de gestión de bases de datos qué política adoptar en caso de que se elimine un registro de la tabla a la cual está referenciando. MySQL por defecto impone la política NO ACTION, la cual impide el borrado de la tabla referenciada. Pero en el caso de esta aplicación se han decidido junto con los directores diferentes claves foráneas con la política CASCADE que a parte de borrar de la tabla referenciada, también lo hace de la tabla en la que está la clave primaria. Las claves foráneas para las cuales no se ha añadido esta opción son las de las tablas de tipos comentadas antes y las de la tabla method_chunk, las cuales mantienen la opción por defecto. Las claves foráneas de la tabla method_chunk no pueden permitir el borrado ya que estaría rompiendo la restricción de las relaciones de method_chunk con activity y goal.

Para el diseño físico de la base de datos también se propusieron el uso de triggers [25], que permiten la ejecución de un código SQL asociado a una tabla al suceder un evento determinado. Durante las primeras reuniones donde se discutió el diseño de la base de datos se propuso el uso de estos elementos para hacer comprobaciones como la mencionada anteriormente. Al final, se decidieron no usar para intentar mantener todo el

código más accesible en la API y de esta manera en caso de que alguna de las restricciones se necesite cambiar en el futuro, no están dispersas en distintos triggers sino que están todas en el código de la API.

Finalmente, el uso de índices en la base de datos no se ha considerado necesario por el momento al no tener que hacer búsquedas que no sean por claves primarias, las cuales ya están indexadas por MySQL [26] con un índice BTREE por defecto.

5.6 Diseño de la interfaz del usuario

A continuación, se explica la organización de las ventanas de la aplicación cliente y los elementos que las forman. Cabe destacar el uso del framework de aplicaciones web Angular [29], el cual se basa en el desarrollo de webs de una sola página, donde en lugar de ir recargando la ventana cada vez que haya una navegación se cargan los componentes que pertenecen a cada pantalla. Por lo que se ha diseñado la interfaz teniendo en cuenta esta característica.

En la figura 12 se puede ver el diagrama completo de la interfaz y se pueden diferenciar cuatro grupos de componentes que se comentarán a continuación, por separado, al corresponder cada uno de ellos a un elemento diferente de la aplicación.

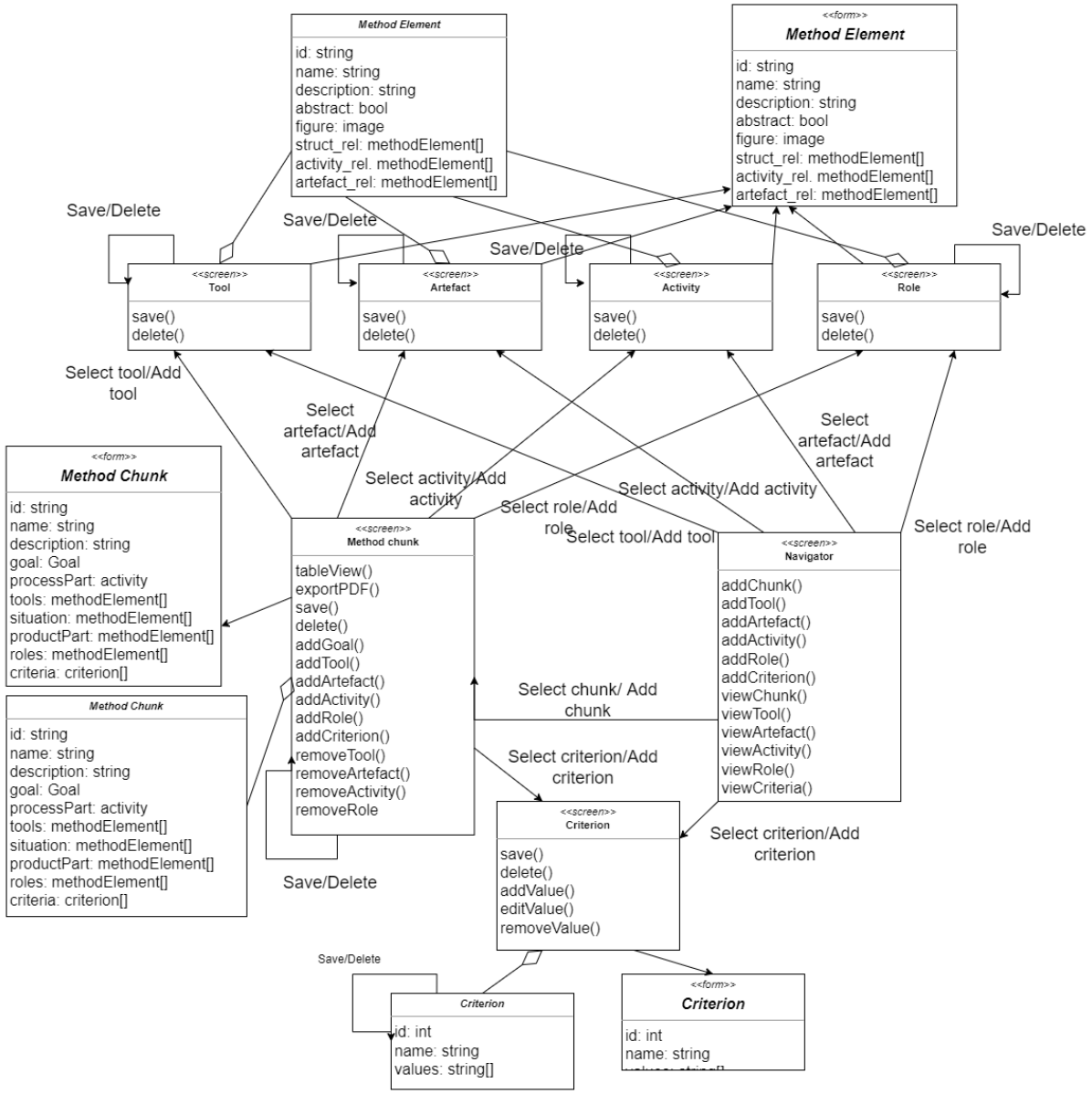


Figura 12: Diagrama de la interfaz. Fuente: Elaboración propia.

Pantalla de navegación

En primer lugar, está la pantalla de la derecha, ampliada para una mejor vista en la figura 13. Esta ventana vendría a ser la pantalla principal de la aplicación, donde se pueden encontrar todos los elementos que hay en el sistema y desde donde se realiza la mayor parte de la navegación de la página hacia el resto de ventanas.

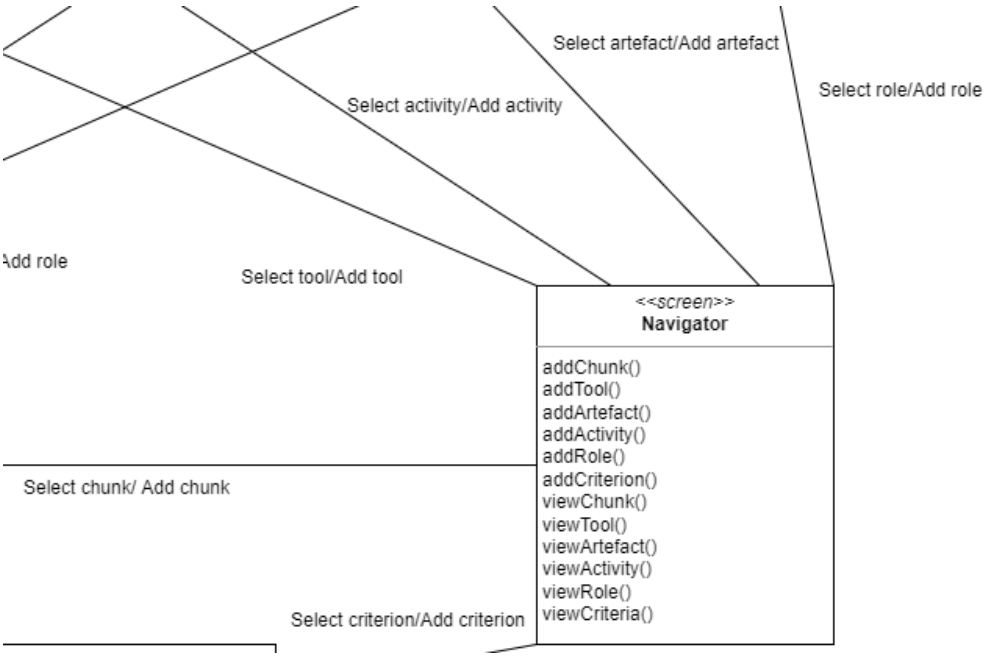


Figura 13: Ampliación a navigator del diagrama de la interfaz. Fuente: Elaboración propia.

Esta pantalla está siempre visible gracias a la característica que se ha mencionado anteriormente de Angular de esta forma la navegación por los diferentes tipos de elementos es siempre accesible y además tiene la particularidad de que la mayoría de sus elementos permiten el Drag and Drop, es decir, que se pueden arrastrar para efectuar diferentes operaciones con el resto de elementos.

Ventana de method chunk

La siguiente pantalla es la más importante de la aplicación ya que es donde se trabaja con el concepto principal de la aplicación, el method chunk. Esta pantalla es la que se carga por defecto junto con el componente mencionado anteriormente. También se puede acceder desde el propio componente de navegación haciendo click en los botones asociados a cada chunk que esté registrado en el sistema. La pantalla será siempre la misma, el único cambió está en que dependiendo si se carga con un chunk válido, se mostrará la información de este y en caso contrario se mostrará todo el formulario vacío.

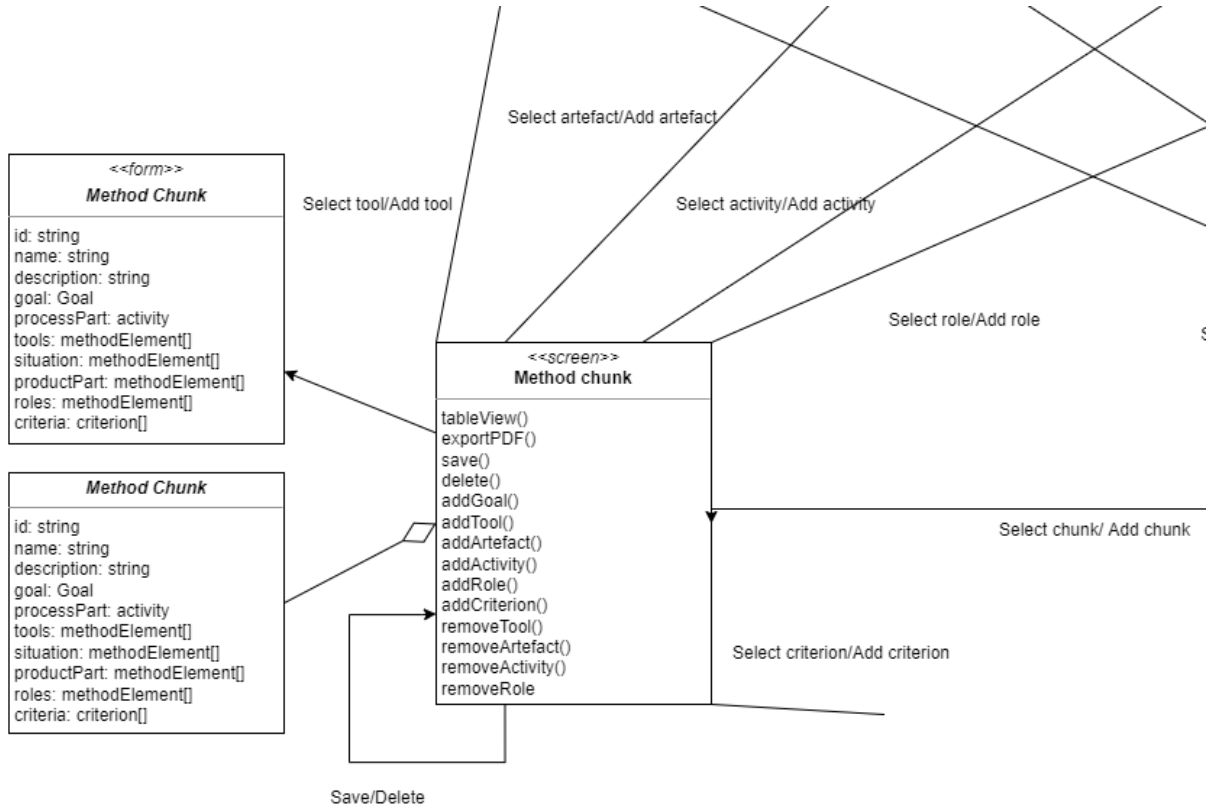


Figura 14: Ampliación a method chunk del diagrama de interfaz. Fuente: Elaboración propia.

Como se puede ver en la Figura 14, la pantalla de method chunk tiene una gran variedad de funciones. Las dos primeras hacen referencia a la historia de usuario MCT, permiten acceder a la tabla donde se presenta el chunk y descargarlo cada una de estas funciones se llaman con botones presentes en la pantalla. A continuación, están los dos métodos que realizan las operaciones de inserción/modificación y borrado con los datos del formulario o el method chunk asociados a esta pantalla, que también se acceden desde botones de la pantalla. Los últimos métodos que se pueden encontrar son los que se encargan de manejar las relaciones del chunk con los otros elementos del sistema.

Pantallas de method element

La tercera pantalla de la aplicación se trata de un conjunto de pantallas que usan el mismo componente pero que de cara al usuario se ven como pantallas diferentes y, por lo tanto, en el diseño de la interfaz también se representa así. En la figura 15 se puede ver la separación en cuatro pantallas diferentes de acuerdo a los cuatro tipos diferentes de method elements que hay en el modelo conceptual.

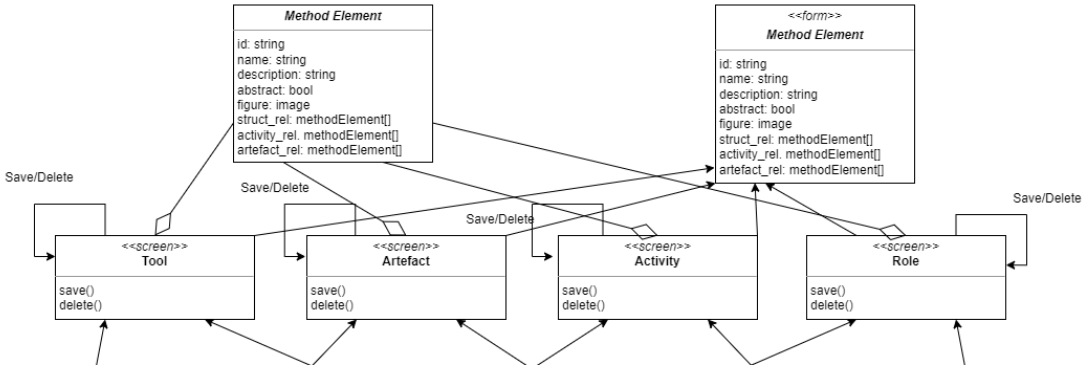


Figura 15: Ampliación a method element del diagrama de interfaz. Fuente: Elaboración propia.

Como se puede ver a pesar de la diferenciación de los cuatro tipos de method elements, las cuatro pantallas comparten el mismo formulario y la misma representación de Method Element. Aparte de esta característica, también tienen los métodos que realizan las operaciones de guardado y borrado de estos elementos. Cuando se selecciona uno de estos botones se actualiza la ventana con la nueva información correspondiente.

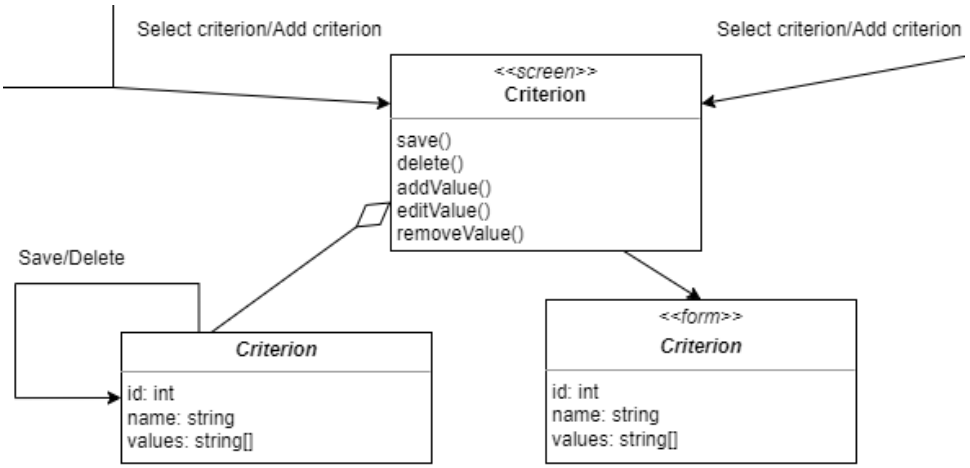


Figura 16: Ampliación a criterion del diagrama de interfaz. Fuente: Elaboración propia.

Pantalla de criteria

Por último, está la ventana de criterion, en la Figura 16, que es similar a las dos anteriores solo que en este caso usa se usa el formulario de Criterion. Para esta ventana también se añaden los métodos para gestionar los valores asociados a dicho criterion. Como las otras se actualiza al guardar o borrar el criterio actual.

6. Implementación

A continuación, se explicarán las características principales de la implementación de la aplicación diferenciando cada componente que forma parte del proyecto.

6.1 Tecnologías y lenguajes usados

Como para esta aplicación se han tenido que implementar todos los componentes de la aplicación, se han usado una gran variedad de lenguajes y tecnologías. A continuación, se describen cada una de estas separadas por componente de la aplicación.

6.1.1 SOPCOM–client

Ya que se trata de la aplicación con la que el usuario interactúa se han usado tecnologías orientadas a este tipo de usos.

- **Angular:** Es un framework para el desarrollo de aplicaciones web que se basa en el uso de webs de una sola página [29]. Esto permite cargar el hecho de navegar o recargar páginas sin necesidad de recargar toda la ventana, sino que simplemente se han de cargar los componentes que se tengan que ver en cada caso. Permite de forma fácil la modificación de tanto el aspecto con HTML y CSS como el scripting con Typescript.
- **Typescript:** Es el lenguaje de scripting que usa Angular. Es una extensión de Javascript que provee de detección de errores desde el momento de la escritura de código [39]. Principalmente avisa de errores para evitar accesos que no existen en los objetos que están presentes en el contexto actual. En el momento de la compilación Typescript se traduce a código Javascript para poder ser ejecutado en cualquier entorno de Javascript.
- **HTML:** HyperText Markup Language [40] es el lenguaje de programación utilizado para definir de qué tipo son y cómo se estructuran los componentes dentro del sistema utilizando etiquetas encapsuladas entre los carácteres ‘<’ y ‘>’.
- **CSS:** Cascading Styling Sheets [41] permite definir estilos para personalizar la renderización de los elementos HTML de una aplicación.
- **Angular material:** Es una librería de componentes para Angular [42] que provee un catálogo de componentes que se pueden usar en aplicaciones web que usan Angular con una simple etiqueta HTML.
- **Node JS y NPM:** Node JS es un entorno de ejecución para JavaScript [43] donde se ejecuta la aplicación. Como se ha mencionado antes, a pesar de que se programa con Typescript, en la compilación se traduce el código a Javascript. NPM es un gestor de paquetes de Node que permite a los desarrolladores compartir paquetes open source con funcionalidades diversas [44]. Por ejemplo Angular es un paquete de NPM: https://www.npmjs.com/package/@angular/core.

6.1.2 SOPCOM-WS

Para la API que conforma el servicio web de la aplicación se ha usado un lenguaje más orientado a la parte servidor y otras tecnologías que para complementar otras operaciones.

- **PHP:** Es un lenguaje de scripting de propósito general [45] que se usa frecuentemente en el desarrollo web. En el caso de esta aplicación se usa para

implementar la SOPCOM-WS sin ningún tipo de framework como podría ser Laravel. Una particularidad de PHP es que no necesita de ningún tipo de compilación o comando de ejecución, sino que al ser un lenguaje interpretado utiliza un intérprete que es lo que se ejecuta y procesa el código PHP.
- **MySQLi:** Una extensión de PHP para ejecutar sentencias directamente en una base de datos MySQL [35].

6.1.3 SOPCOM-DB

Para la parte de la base de datos se ha escogido un sistema de gestión de bases de datos que se complemente bien con las tecnologías de SOPCOM-WS.

- **MySQL:** Uno de los sistemas de gestión de bases de datos relacionales más usados [22], desarrollado por Oracle y que es el más usado con servicios web implementados con PHP.

6.2 Herramientas de desarrollo

Durante la etapa de desarrollo se han usado diversas herramientas para facilitar la implementación del código. A continuación, se detallan las herramientas y el uso que se ha hecho de estas.

- **Visual Studio Code:** La herramienta más utilizada de todas, un editor de código fuente que se puede adaptar a múltiples lenguajes de programación al mismo tiempo gracias a la gran cantidad de extensiones que existen para este software [16]. Además, de la capacidad de personalización que ofrece pudiendo cambiar el tema, los iconos o los comandos de teclas entre otros. Se ha usado para desarrollar tanto la aplicación cliente, como para el servicio web como para los scripts de creación de la base de datos en sql.
- **Postman:** Una herramienta para desarrollar y probar APIs [30] que ofrece una gran cantidad de funcionalidades orientadas las llamadas HTTP hacia las APIs que se estén desarrollando. Se explican más a fondo las funcionalidades en el apartado de Pruebas.
- **Docker:** Software utilizado para el despliegue de aplicaciones en contenedores que separan su contexto del sistema operativo de la máquina que aloja el código [28]. El uso de este software ha servido para poder desplegar fácilmente toda la aplicación en el servidor donde de momento va a ser usada la aplicación.
- **Git y Github:** Las herramientas usadas para la gestión de versiones [9] de la aplicación y el alojamiento en la nube de los repositorios con el código fuente [10].
- **Taiga:** Herramienta de soporte de la metodología Scrum [11] que se ha usado para la gestión de las tareas durante el desarrollo del proyecto.

6.3 Aspectos relevantes del código

En este apartado se explican las características principales del código implementado diferenciándose en dos partes: las características del código de la aplicación cliente y las de la API.

6.3.1 SOPCOM-client

Ya que la aplicación cliente se ha implementado usando el framework Angular, se han aprovechado de sus particularidades para implementar una interfaz de la aplicación de una sola página. Esto hace que haya distintas partes del código que sea importante comentar para entender su funcionamiento.

Componentes: Un componente es el elemento principal del desarrollo en Angular y consiste en el bloque básico de la interfaz [46]. Angular trabaja con un conjunto de componentes que pueden estar renderizados a la vez en la pantalla. Cada uno de estos componentes está formado por un archivo css, que define los estilos para ese componente, un archivo HTML, donde se introduce lo que se quiera ver en pantalla, y dos archivos Typescript, el primero lo añade automáticamente Angular y sirve para ejecutar sus tests, pero el importante de verdad es el último archivo. En este archivo se encuentra toda la definición del componente junto con sus métodos asociados.

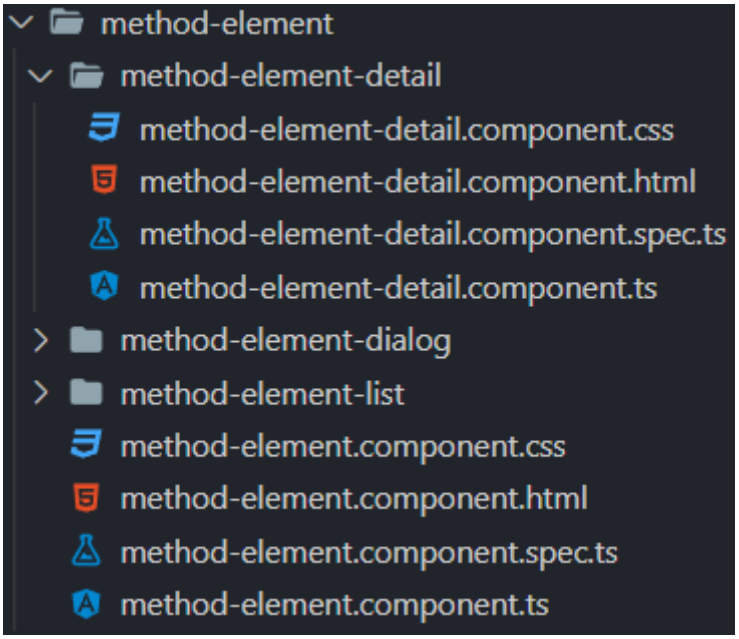


Figura 17: Estructura de un componente. Fuente: Elaboración propia.

Como se puede ver en la Figura 17, también se pueden crear jerarquías de componentes, a pesar de que esta organización de directorios dentro de directorios no crea ninguna dependencia real dentro del código.

Lógica en un mismo componente: Aprovechando la Figura 17, se puede ver cómo está el componente method-element y dentro de él tiene otros tres componentes: detail, dialog y list. Como parte del requisito no funcional EXT, se pensó como mantener toda la lógica de un mismo elemento del modelo conceptual dentro del mismo componente. Y de esta forma se llegó a esta organización de archivos. Por ejemplo, el componente method-element contiene todas las funciones y formularios para manejar los method-elements y los otros dos componentes lo que hacen es invocar este primer componente de formas diferentes. De esta forma en caso de tener que cambiar algo de lógica de un method elemento que afecte al frontend, no se tendría que cambiar en diversos componentes, sino solamente de uno. Y

el resultado de usar este método, es poder tener el mismo componente de dos formas distintas, como se puede ver en las Figuras 18 y 19.

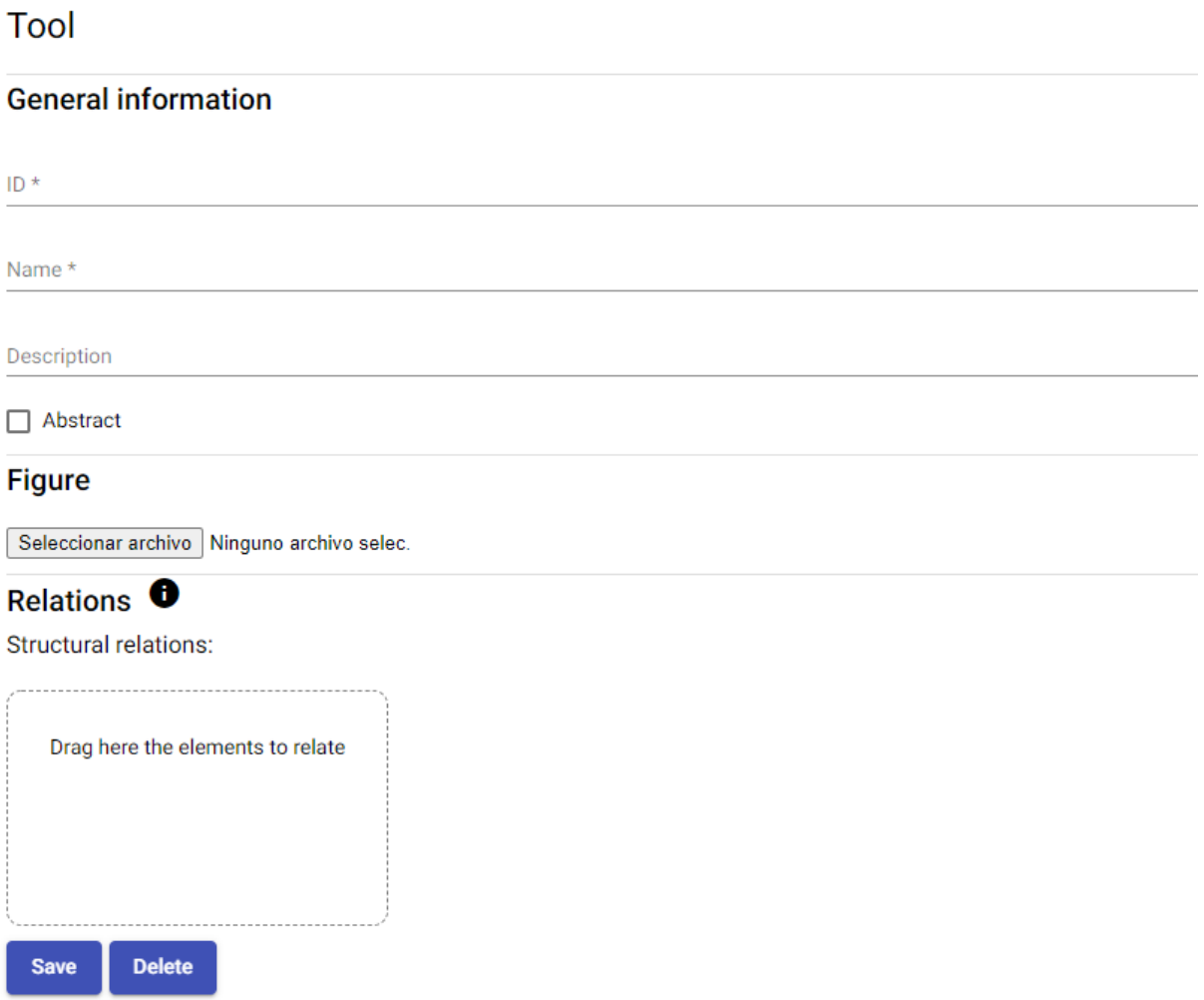


Figura 18: Vista del componente method-element invocado desde detail. Fuente: Elaboración propia

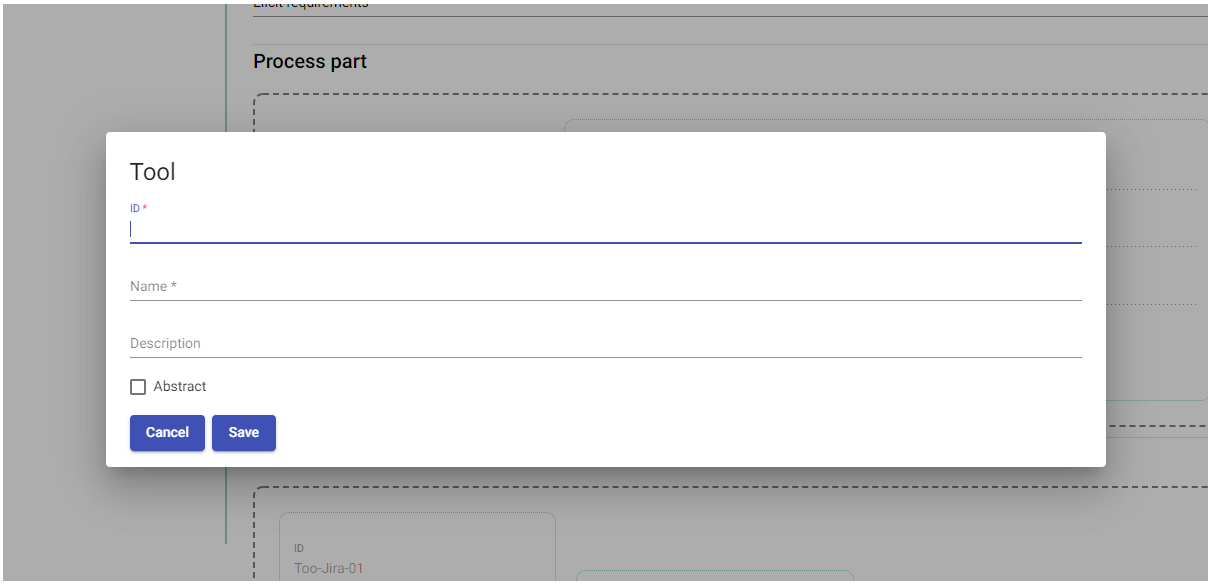


Figura 19: Vista del componente method-element invocado desde dialog. Fuente: Elaboración propia

Servicios: Un servicio de angular es una clase de la cual se crea únicamente una instancia, es decir que usa el patrón singleton [47]. La instancia de este servicio es accesible desde cualquier parte del código y esto permite almacenar datos y consultarlos desde cualquier punto de la aplicación.

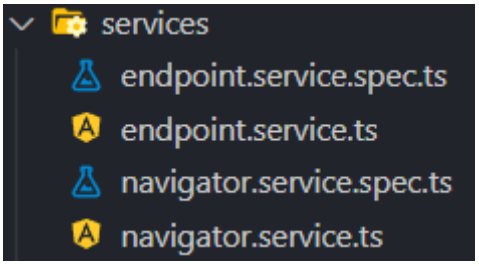


Figura 20: Servicios de la aplicación cliente. Fuente: Elaboración propia.

En la aplicación cliente se pueden encontrar dos servicios, como se puede ver en la Figura 20. El primero de ellos es el servicio de endpoints y su función es la de agrupar todas las llamadas a la API de forma que se puedan llamar desde cualquier parte del código simplemente invocando la función correspondiente al endpoint.

El otro servicio es el de navigator y sirve como soporte del componente con el mismo nombre. El Navigator, como se ha mencionado en el apartado de diseño de la interfaz, es un componente que siempre está visible en la interfaz y desde el cual se puede navegar a cualquier pantalla de la aplicación.

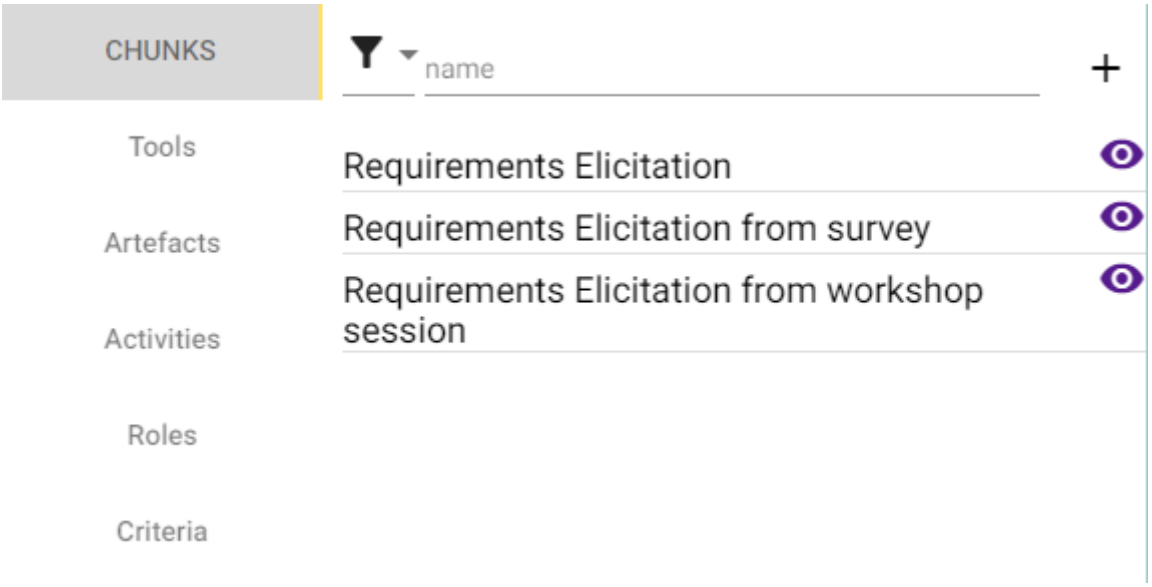


Figura 21: Navigator. Fuente: Elaboración propia.

En la Figura 21 se muestra el componente Navigator, mostrando los chunks que hay almacenados en la base de datos. Al hacer clic en cualquier elemento de la izquierda la lista cambia para mostrar los elementos de dicho tipo. En la parte superior hay un filtro que puede ser por nombre o por id y un botón para ir a la página de añadir un nuevo chunk. Al hacer clic en el ojo se hará routing para ver el formulario del chunk en cuestión.

Cada una de las listas que se usan en este componente se encuentra en el servicio ya que de esta forma el resto de componentes puedan acceder a ella para comprobar, por ejemplo, si se está intentando añadir nuevos elementos con un id ya existente o con nombres que puedan estar repetidos y rompan las restricciones del modelo conceptual.

El servicio también tiene las funciones para refrescar las listas de elementos que a su vez se comunican con el servicio de endpoints. De esta manera, al añadir o borrar o modificar un elemento del sistema no queda desactualizado ni hace falta recargar todo el componente, sino que se actualiza la lista del tipo de elemento.

Router-outlet: Como ya se ha explicado, la aplicación tiene una parte de la ventana que no cambia nunca y otra que va cambiando según la ruta. Para lograr esto se utiliza la directiva router-outlet en el código HTML que le indica a Angular que componente cargar cada vez que cambia la ruta [48]. El código HTML de la pantalla quedaría como en el fragmento de código 6.

```
<div>
  <div style="width: 30%;">
    <app-navigator></app-navigator>
  </div>
  <div style="width: 65%;">
    <router-outlet></router-outlet>
  </div>
```

```
</div>
```

Fragmento de código 6: Código HTML de la pantalla general de la aplicación. Fuente: Elaboración propia.

```
const routes: Routes = [
  {path: 'method-chunk/:id', component: MethodChunkComponent},
  {path: 'method-chunk', component: MethodChunkComponent},
  {path: 'tool/:id', component: MethodElementDetailComponent},
  {path: 'tool', component: MethodElementDetailComponent},
  {path: 'artefact/:id', component: MethodElementDetailComponent},
  {path: 'artefact', component: MethodElementDetailComponent},
  {path: 'activity/:id', component: MethodElementDetailComponent},
  {path: 'activity', component: MethodElementDetailComponent},
  {path: 'role/:id', component: MethodElementDetailComponent},
  {path: 'role', component: MethodElementDetailComponent},
  {path: 'criterion/:id', component: CriterionDetailComponent},
  {path: 'criterion', component: CriterionDetailComponent},
  {path: '**', redirectTo: 'method-chunk', pathMatch: 'full'},
];
```

Fragmento de código 7: Array de rutas y componentes simplificado. Fuente: Elaboración propia.

Para que la directiva de router-outlet asocie cada ruta con un componente, se declara en un módulo, un archivo de configuración de Angular, un array como el del Fragmento de código 7. En el caso del último elemento del array lo que hace es cuando la ruta no es ninguna de las declaradas anteriormente, redirige la página a la ruta indicada en la propiedad 'redirectTo'.

Guards: Los guards son interfaces de angular que dan permiso o deniegan el routing de la aplicación [49]. En el caso de esta aplicación se usan para comprobar que el usuario no haya hecho cambios en un formulario y después intente hacer routing a otra pantalla sin guardar, perdiendo los cambios realizados en el proceso. En el fragmento de código 8 se puede ver cómo se asigna un guard a una ruta.

```
{
    path: 'method-chunk/:id',
    component: MethodChunkComponent,
    canDeactivate:[ConfirmDeactivateGuardMC]
},
```

Fragmento de código 8: Ruta con guard asignado. Fuente: Elaboración propia.

```
export class ConfirmDeactivateGuardMC implements
CanDeactivate<MethodChunkComponent> {
    canDeactivate(target: MethodChunkComponent) {
```

```
        if (target.hasChanges) {
            return window.confirm('You have unsaved changes. Are you sure you
want to leave?');
        }
        return true;
    }
}
```

Fragmento de código 9: Función ejecutada al hacer routing de /method-chunk. Fuente: Elaboración propia.

En el fragmento de código 9 se puede ver la función canDeactivate, que es la que se ejecuta al detectar el guard el routing. En primer lugar, comprueba si el componente tiene algún cambio consultando la variable hasChanges y en caso afirmativo llama a la función confirm de Angular, la cual crea un cuadro de diálogo como el de la Figura 22 que simplemente retorna un booleano según el botón seleccionado..

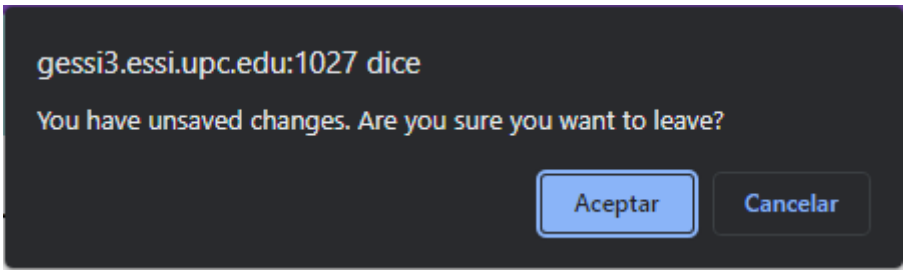


Figura 22: Cuadro de diálogo de confirmación de cambios. Fuente: Elaboración propia.

6.3.2 SOPCOM-WS

MySQLi: Es la extensión de PHP usada para ejecutar sentencias SQL en la base de datos. La forma en que esta extensión funciona es la siguiente: en primer lugar, se intenta establecer una conexión con la base de datos como se puede ver en el Fragmento de código 10.

```
$credentials = $config['database'];
try{
      conn = new mysqli($credentials['host'],
                        $credentials['user'],
                        $credentials['pw'],
                        $credentials['bd']);
} catch (Exception $e) {
      echo(json_encode(Array('error' => "Database connection failed")));
      header("Content-Type: application/json");
      http_response_code(400);
}
```

Fragmento de código 10: Conexión de MySQLi con la base de datos. Fuente: Elaboración propia.

A continuación, se ha de crear una sentencia SQL y llamar a la función prepare, la cual devuelve un manejador de sentencia [35] sobre el que se pueden llamar otras funciones. Una de estas funciones es la siguiente del Fragmento de código 11, bind_param, la cual recibe un string y un número variable de parámetros que corresponden a las variables que contiene la sentencia, identificadas con un '?', en orden de izquierda a derecha. El string indica el tipo de cada variable y también debe estar en orden.

En el ejemplo del Fragmento de código 11 se ve cómo se ejecutaría una sentencia para obtener toda la información de un method chunk. En la sentencia se ha puesto un '?' en el lugar en el que tendría que ir el id del method chunk y, a continuación, se ha sustituido el '?' por el id de la variable $id con la función bind_param. Como solo hay una variable y es de tipo string se indica únicamente con el string "s".

```
$statement = conn->prepare("SELECT * FROM method_chunk WHERE id = ?;");
$statement->bind_param("s", $id);
$statement->execute();
$rows = $statement->get_result();
while($row = $rows->fetch_assoc()) {
    $result[] = $row;
}
return $result;
```

Fragmento de código 11: Ejecución de una sentencia SQL con MySQLi. Fuente: Elaboración propia.

Lo siguiente para obtener los valores que resulten de esta sentencia es ejecutarla mediante la función execute y obtener los resultados con get_result. Por último, se ha de obtener cada registro como un array asociativo con el nombre de la columna como clave y el valor del registro de esa columna como valor. Esto se logra con la función fetch assoc que se ejecuta en bucle hasta que devuelve un null, que indica que ya no hay más registros.

```
$statement = conn->prepare("SELECT * FROM chunk_rel WHERE fromMC = ?
                                                    AND toMC = ?;");
$statement->bind_param("ss", $fromMC, $toMC);
$statement->execute();
$rows = $statement->get_result();
while($row = $rows->fetch_assoc()) {
    $result[] = $row;
}
return $result;
```

Fragmento de código 12: Ejecución de una sentencia SQL de dos variables con MySQLi. Fuente: Elaboración propia.

En el fragmento de código 12 se puede ver otro ejemplo, pero en este caso con dos variables. Por lo que la función bind_param tendrá como parámetros un string "ss", que

indica que tanto el primer elemento como el segundo son strings y, a continuación, las dos variables que contienen los strings de los ids.

POST imágenes: En el modelo conceptual de la Figura 3 se puede ver como un method element tiene la propiedad figure que es de tipo imagen. Esto implica que el servicio web del proyecto ha de permitir la subida y almacenamiento de imágenes. Una primera opción sería almacenar las imágenes directamente en la propia base de datos, pero esto no suele ser una buena práctica por lo que la otra alternativa es la de almacenar las imágenes en el propio sistema de archivos y en la base de datos almacenar la ruta hasta la imagen de cada method element.

El siguiente problema está en la forma de enviar toda la información del method chunk desde el cliente a SOPCOM-WS. Para enviar imágenes se ha de usar un objeto de tipo FormData, lo que no encaja con el endpoint de inserción y modificación de method elements ya que estos usan JSON. Además PHP solo permite adjuntar archivos en formato formData en peticiones HTTP POST y la aplicación también tenía que dejar cambiar la imagen.

Por lo que al fina,l se optó por usar un endpoint diferente para las imágenes. En el fragmento de código 13 se puede ver la creación de un formData al que se le asigna la imagen y se envía a la función encargada de hacer la solicitud POST a SOPCOM-WS.

```
let figureFormData = new FormData()
figureFormData.append('figure', this.figure)
this.endpointService.addMethodElementFigure(this.id,figureFormData).subscribe(
data => {
      this._snackBar.open("Figure updated!",
                          'X',
                          {duration: 2000, panelClass: ['green-snackbar']});
})
```

Fragmento de código 13: Función para subir imágenes. Fuente: Elaboración propia.

Una vez en el backend simplemente se han de usar la variable superglobal, aquellas accesibles desde cualquier punto del código, '$_FILES' para acceder a los archivos que se han subido junto a la solicitud POST [50]. Una vez tenemos la imagen en el servicio web solamente hay que copiarla en la carpeta del servidor deseada usando la función move_uploaded_file de PHP [51], y actualizar la base de datos con la ruta de la imagen.

7. Pruebas

En este apartado se describen las pruebas que se han realizado en cada componente para comprobar que el funcionamiento de la aplicación es el esperado.

7.1 SOPCOM-client

En la aplicación cliente se han hecho pruebas de aceptación a cargo de la responsable del WP1 Dolors Costal. Usando un juego de pruebas que consiste en la creación de diversos method chunks relacionados con la licitación de requisitos se encontraron diversos fallos y se propusieron una serie de cambios.

Entre los errores que se encontraron estaba la posibilidad de relacionar dos method elements de diferente tipo, lo cual rompía la restricción textual RT2 o la posibilidad de dar de alta un criterion con un nombre repetido, lo cual rompe la restricción textual RT1. También se detectaron un par de mensajes de error que estaban intercambiados y que a la hora de mostrarlos no coincidían con el error real.

Por otra parte se propuso la posibilidad de que los method chunks se pudiesen ver en formato tabla y que a la vez esta tabla se pudiese descargar. También se propuso la posibilidad de poder borrar goals o la figura de un method chunk.

Los errores mencionados anteriormente han podido ser solucionados y también se ha podido implementar la vista en formato tabla de un method chunk. En cambio algunas otras propuestas aún no se han podido completar, pero estas ya no se contemplan como cambios necesarios y, por tanto, pueden plantearse como trabajo a realizar en el futuro.

7.2 SOPCOM-WS

Como esta aplicación se centra principalmente en operaciones CRUD (inserción, lectura, modificación y borrado) sobre componentes de procesos software y no realiza ningún tipo de operación sobre estos datos más allá de los mencionados no se ha considerado un uso extensivo de testing en la parte de SOPCOM-WS. El testing realizado se ha centrado en tanto la API como la base de datos para comprobar cómo se tratan los datos según el endpoint usado.

Para realizar este testing se ha usado la aplicación Postman [30] cuya función, entre otras, es la de realizar llamadas a APIs fácilmente para poder ver su comportamiento. La interfaz de este software te permite organizar un área de trabajo o workspace virtual donde almacenar diferentes colecciones de llamadas a APIs en un menú lateral a la izquierda y poder acceder a ellas rápidamente como se puede ver en la figura 23.

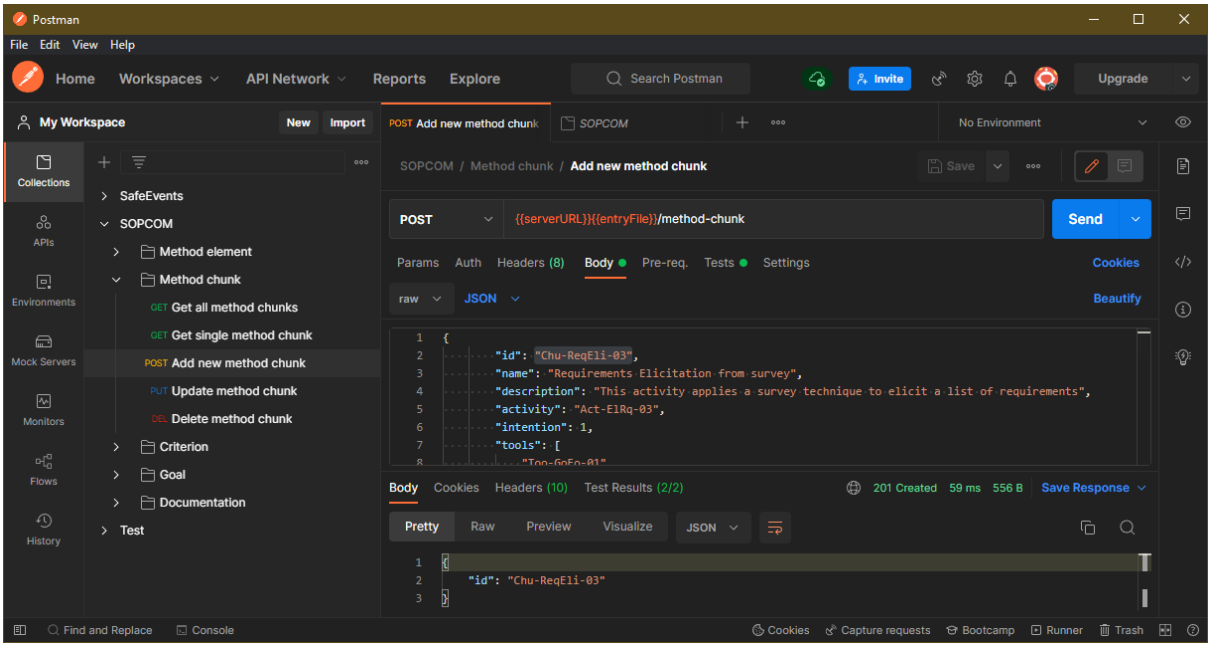


Figura 23: Interfaz de Postman con el cuerpo de la llamada. Fuente: Elaboración propia.

Esta aplicación también permite la definición de variables globales por cada colección de endpoints y de scripts que se ejecutan antes y después de la llamada a la aplicación que se usan para realizar el testing automático de las llamadas. En la figura 23 se puede ver la llamada que corresponde al endpoint de creación de un method chunk, en el centro se puede ver primero el tipo de operación http, en este caso POST, la url del endpoint formada por las variables globales de postman del url del server y la del archivo de entrada. Más abajo se encuentra el cuerpo de la llamada en formato JSON y, por último, en la parte inferior el cuerpo de la respuesta de la API. En la figura 24 se puede ver la misma interfaz pero con los tests definidos y los resultados de los tests.

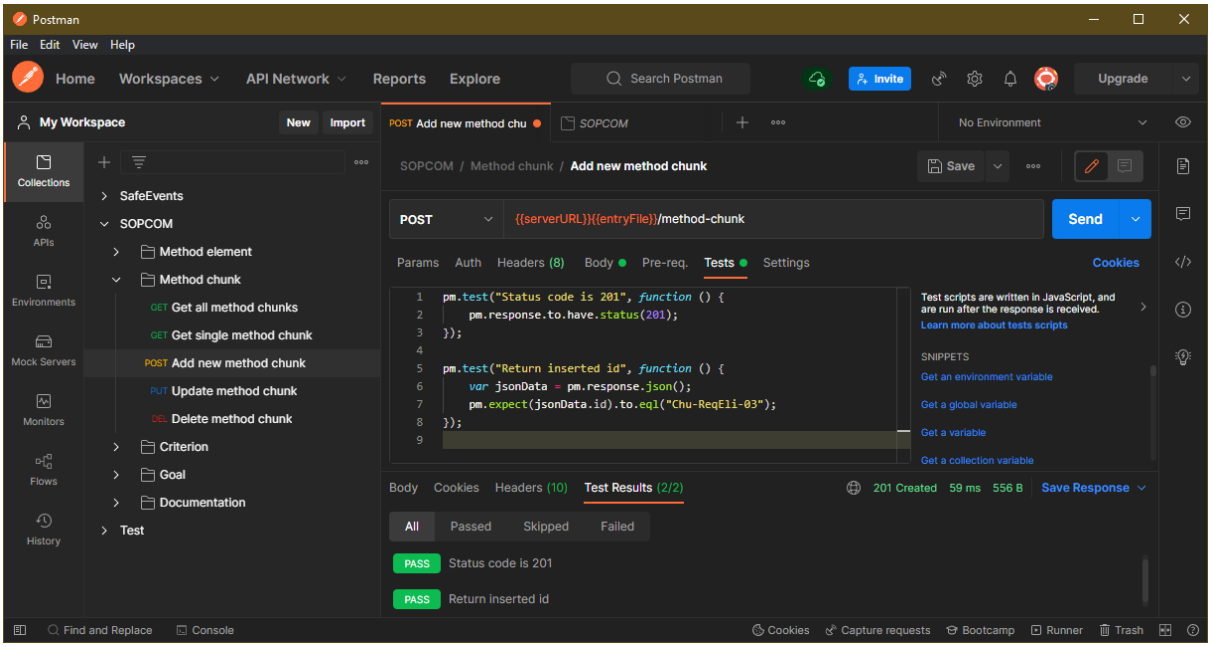


Figura 24: Interfaz de Postman con los tests. Fuente: Elaboración propia.

Con todas estas herramientas que Postman ofrece se han ido probando los endpoints del sistema. Por cada llamada en primer lugar se implementa el endpoint, a continuación, se crea el la llamada en la interfaz de Postman y se definen los tests que han de pasar. En el caso de la Figura 24, el primer test comprueba que el código de respuesta es el esperado, en este caso el 201 que para http se usa para indicar que la inserción ha sido correcta [52], y, a continuación, comprueba que el cuerpo de la respuesta contiene el identificador del chunk insertado. En caso de que los tests no pasen, se comprobaba el error devuelto por la API y se corregía en el código y se volvía a repetir hasta que los tests pasasen.

Ya que este método solamente puede comprobar las respuestas de los endpoints y no todos los cambios que sucedían en la base de datos, estas pruebas se complementaban consultando directamente los contenidos de la base de datos accediendo con la aplicación MySQL Workbench, que ofrece una interfaz gráfica para ejecutar sentencias y consultar de forma fácil los contenidos de la base de datos, o usando los endpoints de consulta para comprobar que los datos eran correctos.

8. Aspectos legales

A continuación, se explican los aspectos legales que se han tenido en cuenta a la hora de planificar y desarrollar el proyecto.

8.1 Leyes aplicables

La aplicación web no consulta ni almacena ningún tipo de información personal o del equipo de la persona que use la aplicación por lo que no aplica ningún tipo de ley de protección de datos. Asimismo, al ser un proyecto para un grupo de investigación y de momento no hay ningún plan de comercialización del software, tampoco aplican leyes de comercialización.

8.2 Licencias

El uso de algunas de las herramientas que se han usado en el desarrollo de la aplicación están suscritas a las licencias que les acompañan. La mayor parte de las herramientas usadas son de código abierto y, por tanto, tienen una licencia gratuita por el uso de sus servicios y en otros casos al no ser un proyecto distribuible no suponen ningún coste. En este apartado se mencionan algunas de las licencias que pueden afectar al proyecto en algún momento.

Ya que la aplicación cliente se ha desarrollado con Angular y Angular Material, se ha tenido que seguir las directrices de la licencia del mismo framework [27]. Se trata de una licencia MIT, por lo que se permite de forma gratuita el uso del código que proporciona angular, sin embargo es el usuario quien toma responsabilidad por aquellas modificaciones y la garantía del nuevo código.

En el caso del backend, PHP incluye una licencia que se tendría que añadir al proyecto en caso de que se estuviera distribuyendo dicho proyecto [53], cosa que en este momento para este no se contempla. El sistema de gestión de bases de datos MySQL está sujeto a una licencia GPL [54] o General Public License, lo que permite a cualquier usuario el uso gratuito de dicho software [55].

Por último, cabe destacar que para este proyecto no afecta, pero Docker dependiendo del tamaño de la empresa que use su software sí que obliga a obtener una licencia de pago [56] y la de Postman es opcional pero permite acceder a funcionalidades que no permite la licencia gratuita [57].

9. Seguimiento del proyecto

A continuación se explica cómo ha transcurrido el desarrollo del proyecto, los imprevistos que han ocurrido y los cambios que se han producido durante el proyecto respecto a lo planificado inicialmente.

9.1 Cambios respecto a la planificación inicial

Los principales cambios que ha sufrido la planificación inicial del proyecto son los siguientes:

- **Tareas de backend atrasadas:** Para poder probar correctamente la comunicación entre frontend y backend se retrasaron algunas tareas de implementación de la API a la espera de poder establecer dicha comunicación.
- **Tareas de frontend atrasadas:** El hecho de retrasar tareas de backend también implicó que tareas de desarrollo de frontend se tuviesen que realizar en el siguiente sprint, el que correspondía a la redacción de la memoria.
- **Nueva historia de usuario:** Con la aplicación cliente ya funcional, se propuso la posibilidad de poder visualizar los chunks en formato tabla, que es el formato con el que se habían presentado al principio de la planificación. Ya que esta tarea podría considerarse una nueva funcionalidad, se consideró oportuno crear una nueva historia de usuario que explicase esta funcionalidad, la historia en cuestión es la MCT.

El resto de tareas se acabaron dentro de los plazos establecidos, algunas incluso en un periodo más corto del esperado pero que después se vio compensado con el tiempo de implementación de la nueva historia de usuario.

9.2 Ejecución real

A pesar de los cambios mencionados en el apartado anterior, el proyecto se ha logrado desarrollar dentro de las 540 horas que corresponden a los 18 créditos de la asignatura del TFG. Pero dentro de estas horas, el tiempo real que ha tomado cada tarea ha sido en algunos casos diferente al planificado. En la Tabla 8 se puede ver una comparativa de las tareas que se han completado en un tiempo distinto al inicial.

ID	Horas iniciales (h)	Coste (€)	Horas finales (h)	Coste final(€)
IT2.8	30	510	25	425
IT2.10	20	340	15	255
IT3.3	30	510	40	680

Tabla 8: Variación de horas. Fuente elaboración propia

Esta variación viene dada ya que las tareas IT2.8 y IT2.10 son las últimas tareas realizadas en la Iteración 1, por lo que al llegar a estas ya se había cogido un buen ritmo de programación y ocurrían menos errores durante la implementación, esto hizo que al final de la iteración 1 se pudiesen acabar las tareas más rápidamente. En cambio, al entrar en la Iteración 2, donde se implementa la aplicación cliente y, por tanto, cambia el lenguaje de

programación, se vuelve a ir lento programando hasta que se vuelve a coger ritmo. Además, las también se ha de contar la inclusión de la historia de usuario MCT que añadía más trabajo a esta tarea de implementación de la aplicación cliente.

Por último, comentar otro imprevisto que sucedió durante la parte final del desarrollo relacionado con el despliegue de la aplicación. Desde el momento en que se tuvo acceso al servidor donde desplegar la aplicación se intentó efectuar sin éxito hasta que al final, después de contactar con el administrador del mismo servidor para tratar de buscar una solución, se logró hacer funcionar. Este imprevisto se logró completar en unas 6 horas distribuidas en diferentes días y se deben tener en cuenta a la hora de hacer el recuento final. Se supone que el encargado de realizar esta tarea ha sido el equipo de desarrollo y, por tanto, el coste de esta tarea queda como en la Tabla 9.

ID	Nombre	Recursos humanos	Horas	Coste (€)
	Desplegar aplicación	ED	6	102

Tabla 9: Coste tareas extra. Fuente: Elaboración propia.

Al fina,l el proyecto se desarrolló en unas 537 horas, teniendo en cuenta que las variaciones de la Tabla 8 no afectan al recuento de horas ya que en un caso se suman 10 horas y en el otro se restan otras 10 al final queda el mismo tiempo y a esto hay que sumarle las 6 horas dedicadas al despliegue de la aplicación en el servidor.

9.3 Control de gestión

Con el fin de mantener un control sobre el presupuesto que se maneja, al acabar una tarea se actualizarán los recursos dedicados a esta tarea de forma que sea más fácil detectar cualquier problema con el presupuesto y ajustarlo en caso de necesidad.

Para los recursos humanos y sus posibles desviaciones se valorarán los costes siguientes.

- Desvío total de horas de dedicación en horas: Horas de dedicación extra de la estimada en la planificación inicial.
  $horas\ aproximadas\ iniciales\ -\ horas\ reales\ finales$
  531 - 537 = -6 h
  Se han dedicado 6 horas más de las previstas en el desarrollo del proyecto.
- Desvío total de horas de dedicación en euros: Coste extra respecto al estimado inicialmente.
  $coste\ total\ aproximado\ inicial\ -\ coste\ total\ real\ final$
  11691 - 11793 = -102 €
  El coste final del proyecto ha sido 102 € más alto del previsto inicialmente.
- Desvío de recursos humanos en coste: Coste de los recursos humanos extra respecto al planificado inicialmente.
  $coste\ recursos\ humanos\ aproximado\ inicial\ -\ coste\ recursos\ humanos\ real\ final$
  La diferencia de coste de recursos humanos es la única diferencia del coste final, por lo que será el mismo coste que el punto anterior.
- Desvío de recursos software en coste: Coste de los recursos software extra respecto al planificado inicialmente.

$coste\ recursos\ software\ aproximado\ inicial\ -\ coste\ recursos\ software\ real\ final$
No se ha producido ningún coste software, ni al inicio ni al final del desarrollo.

- Desvío de recursos hardware en coste: Coste de los recursos hardware extra respecto al planificado inicialmente.
  $coste\ recursos\ hardware\ aproximado\ inicial\ -\ coste\ recursos\ hardware\ real\ final$
  No se ha producido ningún coste hardware, ni al inicio ni al final del desarrollo.

10. Sostenibilidad

En este apartado se presenta el análisis de sostenibilidad de este TFG. El estudio de sostenibilidad se presenta desde tres enfoques: social, ambiental y económico.

10.1 Dimensión económica

Para este proyecto se han estimado los diferentes recursos por tarea que se necesitarán para poder desarrollarlo y el coste que suponen cada uno de estos. Además, también se valoran diferentes problemas o imprevistos que pueden suceder y el coste que pueden suponer cada uno de estos,por lo que creo que el presupuesto final es completo y razonable.

Como ya se ha comentado anteriormente, este proyecto se ha planteado siguiendo un concepto teórico del cual no se han encontrado aplicaciones existentes que lo usen. Esto hace que sea una proyecto innovador, pero a la vez no tiene un propósito comercial, sino de investigación. Este proyecto se desarrolla para un grupo de investigación, por lo que no tiene una motivación económica detrás, aunque tampoco se ha descartado por completo esta posibilidad. Para que el proyecto pueda durar durante más tiempo, se implementará de tal forma que sea fácilmente extensible y así no tener que recurrir a un nuevo proyecto porque este se quede desactualizado.

Si en algún momento llega a darse el escenario en el que la aplicación se comercializa sí que se podría plantear la posibilidad de afectaciones en la viabilidad del proyecto, pero como este no se ha planteado de esta forma, ahora mismo no se plantea un escenario en el que peligre la viabilidad.

10.2 Dimensión social

Este proyecto me puede aportar un conocimiento global sobre el desarrollo de proyectos de software al tener que realizar todas las tareas de gestión, documentación, desarrollo, etc. También el hecho de que se aplique un concepto teórico diferente al habitual hace que sea novedoso y, por tanto, aprender diferentes formas de desarrollo.

La necesidad de este proyecto es real, ya que se ha detectado la falta de una aplicación que sea capaz de hacer lo que se propone realizar en este proyecto para cumplir los objetivos del proyecto del cual depende este. Esto también puede verse como un problema, ya que no hay ningúna aplicación de la cual se pueda partir y, por tanto, que se pueda actualizar o readaptar para cumplir los requisitos de este proyecto, lo cual ayudaría a reducir el tiempo y los recursos necesarios para completar este trabajo.

A corto plazo este TFG sólo beneficiará a todos aquellos implicados en el proyecto DOGO4ML, ya que la intención es que se use para este proyecto y se va a desarrollar teniendo en cuenta esto. Pero es posible que en el futuro esta aplicación se pueda extender de tal forma que las empresas puedan usarla para agilizar el desarrollo de sus proyectos.

Como este proyecto se trata simplemente de una herramienta de gestión de datos, la cual normalmente será usada por profesionales con conocimientos en el desarrollo de software, no se valora la posibilidad de generar dependencia hacia esta aplicación ni menos la posibilidad de dejar a alguien en posición de debilidad.

10.3 Dimensión ambiental

Durante el desarrollo del proyecto el coste ambiental será el de la electricidad que se usará ya que para todas las tareas se usará el ordenador y al finalizar la aplicación únicamente necesitará un servidor que aloje la api y la base de datos para poder escuchar las peticiones que le lleguen desde el cliente.

El proyecto no plantea una mejora en el aspecto medioambiental, en cuanto a los recursos que necesita una vez acabado son similares a otros proyectos que implementan una aplicación web.

Durante la vida útil del proyecto los recursos necesarios para mantener la aplicación será simplemente un servidor para alojar la api y la base de datos, pero también es importante destacar que solo tiene previsto usarse durante el desarrollo del proyecto DOGO4ML y al finalizarlo no hará falta seguir manteniendo el servidor activo a menos que se considere necesario para otro proyecto.

Este proyecto ha sido desarrollado con el requisito no funcional de la extensibilidad, por lo que es posible que en el futuro la aplicación se amplíe con nuevas funcionalidades. Claro que el hecho de una aplicación de mayor envergadura consume más recursos del servidor en el que se aloja, pero por el momento los cambios que se han podido llegar a plantear no supondría un enorme cambio en el consumo de estos recursos. Por otro lado estaría la posibilidad de que se desplegarse en diferentes servidores debido a que la aplicación se ha extendido por su comercialización, pero dado que la posibilidad de que pueda llegar a darse este escenario aún está muy lejos se puede decir que este proyecto no tendrá de momento un aumento notable de los recursos consumidos.

11.Conclusiones y trabajo futuro

A continuación, se exponen las conclusiones del proyecto y se proponen diferentes tareas para extender o mejorar la aplicación.

11.1 Competencias técnicas trabajadas y relación con la especialidad de ingeniería del software.

El proyecto ha consistido en el diseño e implementación de una aplicación web desde cero, formada por una base de datos, un servicio web y una aplicación cliente. Por lo que durante el desarrollo del proyecto se han ido trabajando diferentes competencias técnicas que han ido mejorando a lo largo del proyecto.

La competencia técnica principal del proyecto ha sido la CES1.1, que es en lo que ha consistido principalmente este proyecto, la construcción de un software siguiendo los requisitos impuestos inicialmente por los directores del proyecto y la responsable del WP1 de DOGO. Al final del proyecto se ha obtenido una aplicación que cumple dichos requisitos y que ha sido validada por Dolors Costal, responsable del WP1.

Otra competencia técnica que se ha trabajado bastante ha sido la CES1.5 y hace referencia a las tareas relacionadas con la base de datos. Como ya se ha comentado en el apartado de base de datos, durante las primeras semanas del proyecto la representación del diagrama UML en la base de datos fue cambiando debido a ajustes que se hacían en el modelo para poder adaptarlo a la aplicación. Al final, se ha obtenido una base de datos que cumple muchas de las restricciones del modelo sin necesidad de ser controladas por procedimientos almacenados o por código del servicio web, por lo que resulta una base de datos correcta y, además, con elementos que permiten extenderla fácilmente.

En cambio una competencia donde no se ha logrado alcanzar el nivel deseado ha sido la CES1.7. Pese a que el testing que se ha hecho no se considera malo, no se han encontrado formas de realizar este testing de forma más ágil y que de esta forma ya quedase hecho para futuros trabajos con esta aplicación.

La competencia CES2.1 también se ha trabajado especialmente en esta memoria. A pesar de que los requisitos del proyecto ya se habían establecido antes de tomarlo como TFG, se tuvieron que extender esos requisitos y asegurarse de que la aplicación los cumplía.

Las competencias CES1.2 y CES1.3 también se han trabajado, aunque no al nivel del resto, principalmente en las reuniones semanales. Donde se comentaban los problemas que se encontraban y se proponían soluciones que se acordaban entre los directores y el autor.

11.2 Conclusiones personales

La valoración personal que hago del proyecto en global es satisfactoria. Es verdad que siempre se puede mejorar el código o el testing, pero el lograr especificar, diseñar y construir una aplicación web en los cuatro meses que dura la asignatura del TFG hace que esté satisfecho con el resultado obtenido. Además de eso, cuenta con el visto bueno de los

directores del proyecto y de la responsable del WP1 de DOGO lo cual añade un plus a esa satisfacción.

Como estudiante de la especialidad de ingeniería del software he podido aplicar los conocimientos adquiridos en las asignaturas de esta especialidad en el proyecto. Por ejemplo ER para la especificación de requisitos o DBD a la hora de definir el diseño físico de la base de datos. Por lo tanto, el trabajo realizado durante el desarrollo de este TFG se ha realizado acorde a los conocimientos adquiridos en la especialidad de ingeniería del software.

A nivel de programación, he podido mejorar mis habilidades con el framework Angular utilizando, por ejemplo, los elementos de arrastre, y con PHP. Ya tenía experiencia previa con estos lenguajes, pero el hecho de poder trabajar con ellos desde cero ha hecho que se consoliden mis conocimientos de estos lenguajes. También he tenido la oportunidad de usar Docker, que era una tecnología que tenía pendiente estudiar desde hace tiempo y este proyecto me ha brindado la posibilidad de hacerlo.

Por último, como desarrollador de software he mejorado en todo el proceso de documentación del proyecto como en la especificación de requisitos o la planificación de proyectos. Estos conocimientos de gestión y documentación seguramente serán útiles en un futuro y, por tanto, haber podido trabajarlos en este proyecto ha sido de gran ayuda.

11.3 Trabajo futuro

Como uno de los requisitos de esta aplicación era el de extensibilidad, ya está presente la posibilidad de que un nuevo estudiante trabaje sobre este proyecto añadiendo nuevas funcionalidades. Algunas de las mejoras que se pueden hacer en un futuro sobre esta aplicación son las siguientes:

- **Mejora de la documentación OpenAPI:** A pesar de que al final se ha optado por aportar una documentación con Postman por su simplicidad a la hora de generarla, se comenzó la documentación del servicio web con la librería swagger-PHP [58], que permite generar la documentación a partir de una serie de comentarios con tags específicos que después se convierten en un archivo JSON. Por lo que se trataría de acabar esta documentación usando la guía que provee su creador.
- **Mejorar respuestas de SOPCOM-WS:** El servicio web desarrollado es permisivo desde el punto de vista en que si se intenta dar de alta datos que no cumplen las restricciones de integridad no responde como una petición incorrecta sino que se da de alta todo lo que sea correcto y no informa de todos los errores que se puedan dar. Este hecho sí que lo implementa la aplicación cliente, pero sería interesante mejorar este aspecto de SOPCOM-WS.
- **Gestión de usuarios:** Actualmente SOPCOM es una aplicación accesible por cualquier miembro de la UPC usando la conexión VPN de la propia universidad, pero en un futuro cuando se quiera compartir con otros grupos investigación, o con alguna empresa interesada, sería bueno implementar un sistema de gestión de usuarios para la aplicación.
- **Vista de listas:** A pesar de que se empezaron a implementar vistas de listas como una primera idea para la navegación entre elementos aprovechando también el hecho de tener una imagen asociada, al final se acabó usando el componente

navigator comentado en el apartado de diseño de la interfaz. Por lo que estos componentes quedaron descartados, pero aun así en el futuro puede ser interesante tener acceso a un listado de elementos más visual que el del navegador.

- **Ordenación:** En este momento no se considera necesario, pero puede ser que aprovechando los componentes de arrastre, se quiera implementar el orden en las relaciones para que el usuario las ordene a su gusto y en pantalla aparezcan en dicho orden.

12. Referencias

[1] Software and Service Engineering Group (GESSI) [en línea]. Universitat Politècnica de Catalunya. [Consulta: 26 febrero 2022] Disponible en: <https://gessi.upc.edu/en>

[2] Grupos de investigación GESSI y DTIM. Desarrollo, operativa y gobernanza de datos para sistemas software basados en aprendizaje automático (DOGO4ML). Memoria científico-técnica de solicitud de proyectos RETOS (Ministerio de ciencia, tecnología e innovación), convocatoria 2020. Investigadores principales: Oscar Romero, Xavier Franch.

[3] Henderson-Sellers B, Ralyté J, Ågerfalk P, Rossi M. Situational method engineering. Springer, 2014.

[4] Managing projects [en línea]. [Consulta: 01/03/2022] Disponible en: <https://ec.europa.eu/chafea/health/beneficiaries-corner/documents/factsheet-03.pdf>

[5] API REST [en línea]. [Consulta: 14/05/2022] Disponible en: <https://www.ibm.com/es-es/cloud/learn/rest-apis>

[6] SUpporting evolution and adaptation of PERsonalized Software by Exploiting contextual Data and End-user feedback (SUPERSEDE) [en línea]. [Consulta: 27 febrero 2022]. Disponible en: <https://cordis.europa.eu/project/id/644018/results>

[7] Fondazione Bruno Kessler (FBK) [en línea]. [Consulta: 26 febrero 2022] Disponible en: <https://www.fbk.eu/en/>

[8] Scrum [en línea]. [Consulta: 26 febrero 2022] Disponible en: <https://www.scrum.org/>

[9] Git [en línea]. Git. [Consulta: 27 febrero 2022] Disponible en: <https://git-scm.com/>

[10] Github [en línea]. Github. [Consulta: 27 febrero 2022] Disponible en: <https://github.com/>

[11] Taiga [en línea]. Taiga. [Consulta: 27 febrero 2022] Disponible en: <https://www.taiga.io/>

[12] Google Meet [en línea]. [Consulta: 6 marzo 2022] Disponible en: <https://meet.google.com/>

[13] Google Drive [en línea]. [Consulta: 6 marzo 2022] Disponible en: <https://www.google.com/intl/es_es/drive/>

[14] Documentos de Google [en línea]. [Consulta: 6 marzo 2022] Disponible en: <https://www.google.es/intl/es/docs/about/>

[15] Hojas de Cálculo de Google [en línea]. [Consulta: 6 marzo 2022] Disponible en: <https://www.google.es/intl/es/sheets/about/>

[16] Visual Studio Code [en línea]. [Consulta: 6 marzo 2022] Disponible en: <https://code.visualstudio.com/>

[17] Tecnoempleo [en línea]. [Consulta: 13 marzo 2022] Disponible en: <https://www.tecnoempleo.com/informe-empleo-informatica.php>

[18] Boletín oficial del estado [en línea]. Madrid: BOE, 1 de diciembre de 2021, núm. 287 p. 148179 a 148181 [Consulta: 13 marzo 2022]. Disponible en: <https://www.boe.es/diario_boe/txt.php?id=BOE-A-2021-19799 >

[19] Xavier Franch, Jolita Ralyté, Anna Perini, Alberto Abelló, David Ameller, Jesús Gorroñogoitia, Sergi Nadal, Marc Oriol, Norbert Seyff, Alberto Siena, Angelo Susi: A Situational Approach for the Definition and Tailoring of a Data-Driven Software Evolution Method. CAiSE 2018:603-618

[20] Xavier Franch, Aron Henriksson, Jolita Ralyté, Jelena Zdravkovic: Data-Driven Agile Requirements Elicitation through the Lenses of Situational Method Engineering. RE 2021: 402-407

[21] MDN Web Docs [en línea]. [Consulta 28 marzo 2022] Disponible en: <https://developer.mozilla.org/es/docs/Glossary/MVC>

[22] MySQL [en línea]. [Consulta: 7 mayo 2022] Disponible en: <https://www.mysql.com/>

[23] W3Schools [en línea]. [Consulta: 7 mayo 2022] Disponible en: <https://www.w3schools.com/php/php_mysql_intro.asp>

[24] phpMyAdmin [en línea]. [Consulta: 7 mayo 2022] Disponible en: <https://www.phpmyadmin.net/>

[25] MySQL 8.0 Reference Manual triggers [en línea]. [Consulta: 7 mayo 2022] Disponible en: <https://dev.mysql.com/doc/refman/8.0/en/triggers.html>

[26] MySQL 8.0 Reference Manual create index [en línea]. [Consulta: 7 mayo 2022] Disponible en: <https://dev.mysql.com/doc/refman/8.0/en/create-index.html>

[27] Angular license [en línea]. [Consulta: 7 mayo 2022] Disponible en: <https://github.com/angular/angular/blob/main/LICENSE>

[28] Home - Docker [en línea]. [Consulta: 26 mayo 2022] Disponible en: <https://www.docker.com/>

[29] Angular [en línea]. [Consulta: 26 mayo 2022] Disponible en: <https://angular.io/>

[30] Postman [en línea]. [Consulta: 26 mayo 2022] Disponible en <https://www.postman.com/>

[31] diagrams.net [en línea]. [Consulta: 15 junio 2022] Disponible en: <https://chrome.google.com/webstore/detail/diagramsnet/onlkggianjhjenigcpigpjehhpplldkc?hl=es-419>

[32] Nginx [en línea]. [Consulta: 17 junio 2022] Disponible en: <https://www.nginx.com/>

[33] Alpine [en línea]. [Consulta: 17 junio 2022] Disponible en: <https://www.alpinelinux.org/>

[34] Docker PHP [en línea]. [Consulta: 17 junio] Disponible en: <https://hub.docker.com/_/php>

[35] PHP MySQLi [en línea]. [Consulta: 18 junio] Disponible en: <https://www.php.net/manual/es/book.mysqli.php>

[36] MySQL Workbench [en línea]. [Consulta: 18 junio] Disponible en: <https://www.mysql.com/products/workbench/>

[37] Angular ActivatedRoute [en línea]. [Consulta: 19 junio] Disponible en: <https://angular.io/api/router/ActivatedRoute>

[38] MySQL 8.0 Reference Manual AUTO_INCREMENT [en línea]. [Consulta: 10 junio] Disponible en: <https://dev.mysql.com/doc/refman/8.0/en/example-auto-increment.html>

[39] TypeScript [en línea]. [Consulta: 16 junio] Disponible en: <https://www.typescriptlang.org/>

[40] HTML | MDN [en línea]. [Consulta: 16 junio] Disponible en: <https://developer.mozilla.org/es/docs/Web/HTML>

[41] CSS | MDN [en línea]. [Consulta: 16 junio] Disponible en: <https://developer.mozilla.org/es/docs/Web/CSS>

[42] Angular Material [en línea]. [Consulta: 16 junio] Disponible en: <https://material.angular.io/>

[43] Node JS [en línea]. [Consulta: 16 junio] Disponible en: <https://nodejs.org/es/>

[44] npm [en línea]. [Consulta: 16 junio] Disponible en: <https://www.npmjs.com/>

[45] PHP [en línea]. [Consulta: 16 junio] Disponible en: <https://www.php.net/>

[46] Angular Docs Component [en línea]. [Consulta: 18 junio] Disponible en: <https://angular.io/api/core/Component#description>

[47] Angular Docs Add services [en línea]. [Consulta: 18 junio] Disponible en: <https://angular.io/tutorial/toh-pt4#why-services>

[48] Angular Docs Router Reference [en línea]. [Consulta: 18 junio] Disponible en: <https://angular.io/guide/router-reference#router-outlet>

[49] Angular Docs Common Routing Tasks [en línea]. [Consulta: 18 junio] Disponible en: <https://angular.io/guide/router#preventing-unauthorized-access>

[50] PHP $_FILES [en línea]. [Consulta: 18 junio] Disponible en: <https://www.php.net/manual/es/reserved.variables.files.php>

[51] PHP move_uploaded_file [en línea]. [Consulta: 17 junio] Disponible en: <https://www.php.net/manual/es/function.move-uploaded-file.php>

[52] 201 Created [en línea]. [Consulta: 17 junio] Disponible en: <https://developer.mozilla.org/es/docs/Web/HTTP/Status/201>

[53] PHP Licensing [en línea]. [Consulta: 19 junio] Disponible en: <https://www.php.net/license/index.php>

[54] MySQL Community Edition [en línea]. [Consulta: 19 junio] Disponible en: <https://www.mysql.com/products/community/#:~:text=It%20is%20available%20under%20the,both%20relational%20and%20NoSQL%20applications>

[55] GNU General Public License [en línea]. [Consulta: 19 junio] Disponible en: <https://es.wikipedia.org/wiki/GNU_General_Public_License>

[56] Docker pricing [en línea]. [Consulta: 19 junio] Disponible en: <https://www.docker.com/pricing/faq/>

[57] Postman pricing [en línea]. [Consulta: 19 junio] Disponible en: <https://www.postman.com/pricing/>

[58] Zircote-swagger [en línea]. [Consulta: 20 junio] Disponible en: <https://zircote.github.io/swagger-php/>

Anexo

Documentación de SOPCOM-WS con POSTMAN


```
{
	"info": {
		"_postman_id": "fa9d138f-e997-4d80-8cd0-b1da887a84c1",
		"name": "SOPCOM-WS",
		"schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json",
		"_exporter_id": "13495339"
	},
	"item": [
		{
			"name": "Method chunk",
			"item": [
				{
					"name": "Get all method chunks",
					"event": [
						{
							"listen": "test",
							"script": {
								"exec": [
									"pm.test(\"Status code is 200\", function () {\r",
									"
pm.response.to.have.status(200);\r",
									"
"});"
								],
								"type": "text/javascript"
							}
						}
					],
					"request": {
						"method": "GET",
						"header": [],
						"url": {
							"raw": "{{serverURL}}{{entryFilePath}}/method-chunk",
							"host": [
								"{{serverURL}}{{entryFilePath}}"
							],
							"path": [
								"method-chunk"
							]
						}
					},
					"response": []
				},
				{
					"name": "Get single method chunk",
					"event": [
						{
							"listen": "test",
							"script": {
								"exec": [
									"pm.test(\"Status code is 200\", function () {\r",
									"
pm.response.to.have.status(200);\r",
									"
"});"
								],
```

```
                                                "type": "text/javascript"
                                        }
                                }
                        ],
                        "request": {
                                "method": "GET",
                                "header": [],
                                "url": {
                                        "raw":
"{{serverURL}}{{entryFilePath}}/method-chunk/:id",
                                        "host": [
                                                "{{serverURL}}{{entryFilePath}}"
                                        ],
                                        "path": [
                                                "method-chunk",
                                                ":id"
                                        ],
                                        "variable": [
                                                {
                                                        "key": "id",
                                                        "value": "Chu-ReqEli-01"
                                                }
                                        ]
                                }
                        },
                        "response": []
                },
                {
                        "name": "Add new method chunk",
                        "event": [
                                {
                                        "listen": "test",
                                        "script": {
                                                "exec": [
                                                        "pm.test(\"Status code is
201\", function () {\r",
                                                        "
pm.response.to.have.status(201);\r",
                                                        "});\r",
                                                        "\r",
                                                        "pm.test(\"Return inserted
id\", function () {\r",
                                                        "    var jsonData =
pm.response.json();\r",
                                                        "
pm.expect(jsonData.id).to.eql(\"Chu-ReqEli-03\");\r",
                                                        "});\r",
                                                        ""
                                                ],
                                                "type": "text/javascript"
                                        }
                                }
                        ],
                        "request": {
                                "method": "POST",
                                "header": [],
                                "body": {
                                        "mode": "raw",
                                        "raw": "{\r\n    \"id\":
\"Chu-ReqEli-01\",\r\n    \"name\": \"Requirements Elicitation\",\r\n    \"description\": \"This
activity applies some selected technique to elicit a list of requirements from a set of
stakeholders\",\r\n    \"activity\": \"Act-ElRq-01\",\r\n    \"intention\": 1,\r\n    \"tools\":
[\r\n        \"Too-Jira-01\"\r\n    ],\r\n    \"consumedArtefacts\": [\r\n
```

```
\"Art-LiCt-01\",\r\n        \"Art-LiGo-01\",\r\n        \"Art-LiSt-01\"\r\n    ],\r\n
\"producedArtefacts\": [\r\n        \"Art-LiRq-01\"\r\n    ],\r\n    \"roles\": [\r\n        {\r\n
\"id\": \"Rol-RqEn-01\",\r\n            \"isSet\": 0\r\n        },\r\n        {\r\n
\"id\": \"Rol-SeSt-01\",\r\n            \"isSet\": 1\r\n        }\r\n    ],\r\n
\"contextCriteria\": []\r\n}",
                                                "options": {
                                                        "raw": {
                                                                "language": "json"
                                                        }
                                                }
                                        },
                                        "url": {
                                                "raw":
"{{serverURL}}{{entryFilePath}}/method-chunk",
                                                "host": [
                                                        "{{serverURL}}{{entryFilePath}}"
                                                ],
                                                "path": [
                                                        "method-chunk"
                                                ]
                                        }
                                },
                                "response": []
                        },
                        {
                                "name": "Update method chunk",
                                "event": [
                                        {
                                                "listen": "test",
                                                "script": {
                                                        "exec": [
                                                                "pm.test(\"Status code is
201\", function () {\r",
                                                                "
pm.response.to.have.status(201);\r",
                                                                "});"
                                                        ],
                                                        "type": "text/javascript"
                                                }
                                        }
                                ],
                                "request": {
                                        "method": "PUT",
                                        "header": [],
                                        "body": {
                                                "mode": "raw",
                                                "raw": "{\r\n    \"name\": \"Requirements
Elicitation\",\r\n    \"description\": \"This activity applies some selected technique to elicit a
list of requirements from a set of stakeholders\",\r\n    \"activity\": \"Act-ElRq-01\",\r\n
\"intention\": 1,\r\n    \"tools\": [\r\n        \"Too-GoFo-01\",\r\n        \"Too-Jira-01\"\r\n
],\r\n    \"consumedArtefacts\": [\r\n        \"Art-LiGo-01\"\r\n    ],\r\n
\"producedArtefacts\": [\r\n        \"Art-LiRq-01\",\r\n        \"Art-LiSt-01\",\r\n
\"Test\"\r\n    ],\r\n    \"roles\": [\r\n        {\r\n            \"id\": \"Rol-RqEn-01\",\r\n
\"isSet\": 0\r\n        },\r\n        {\r\n            \"id\": \"Rol-SeSt-01\",\r\n
\"isSet\": 1\r\n        }\r\n    ],\r\n    \"contextCriteria\": []\r\n}",
                                                "options": {
                                                        "raw": {
                                                                "language": "json"
                                                        }
                                                }
                                        },
                                        "url": {
                                                "raw":
```

```
"{{serverURL}}{{entryFilePath}}/method-chunk/:id",
                                                "host": [
                                                        "{{serverURL}}{{entryFilePath}}"
                                                ],
                                                "path": [
                                                        "method-chunk",
                                                        ":id"
                                                ],
                                                "variable": [
                                                        {
                                                                "key": "id",
                                                                "value": "Chu-ReqEli-01"
                                                        }
                                                ]
                                        }
                                },
                                "response": []
                        },
                        {
                                "name": "Delete method chunk",
                                "event": [
                                        {
                                                "listen": "test",
                                                "script": {
                                                        "exec": [
                                                                "pm.test(\"Status code is
204\", function () {\r",
                                                                "
pm.response.to.have.status(204);\r",
                                                                "});"
                                                        ],
                                                        "type": "text/javascript"
                                                }
                                        }
                                ],
                                "request": {
                                        "method": "DELETE",
                                        "header": [],
                                        "url": {
                                                "raw":
"{{serverURL}}{{entryFilePath}}/method-chunk/:id",
                                                "host": [
                                                        "{{serverURL}}{{entryFilePath}}"
                                                ],
                                                "path": [
                                                        "method-chunk",
                                                        ":id"
                                                ],
                                                "variable": [
                                                        {
                                                                "key": "id",
                                                                "value": "Chu-ReqEli-01"
                                                        }
                                                ]
                                        }
                                },
                                "response": []
                        }
                ]
        },
        {
                "name": "Method element",
                "item": [
```

```
                {
                        "name": "Get all method elements",
                        "event": [
                                {
                                        "listen": "test",
                                        "script": {
                                                "exec": [
                                                        "pm.test(\"Status code is
200\", function () {\r",
                                                        "
pm.response.to.have.status(200);\r",
                                                        "});"
                                                ],
                                                "type": "text/javascript"
                                        }
                                }
                        ],
                        "request": {
                                "method": "GET",
                                "header": [],
                                "url": {
                                        "raw":
"{{serverURL}}{{entryFilePath}}/method-element",
                                        "host": [
                                                "{{serverURL}}{{entryFilePath}}"
                                        ],
                                        "path": [
                                                "method-element"
                                        ],
                                        "query": [
                                                {
                                                        "key": "type",
                                                        "value": "1",
                                                        "disabled": true
                                                }
                                        ]
                                }
                        },
                        "response": []
                },
                {
                        "name": "Get single method element",
                        "event": [
                                {
                                        "listen": "test",
                                        "script": {
                                                "exec": [
                                                        "pm.test(\"Status code is
200\", function () {\r",
                                                        "
pm.response.to.have.status(200);\r",
                                                        "});"
                                                ],
                                                "type": "text/javascript"
                                        }
                                }
                        ],
                        "protocolProfileBehavior": {
                                "disableBodyPruning": true
                        },
                        "request": {
                                "method": "GET",
                                "header": [],
```

```
                                        "body": {
                                                "mode": "formdata",
                                                "formdata": []
                                        },
                                        "url": {
                                                "raw":
"{{serverURL}}{{entryFilePath}}/method-element/:id",
                                                "host": [
                                                        "{{serverURL}}{{entryFilePath}}"
                                                ],
                                                "path": [
                                                        "method-element",
                                                        ":id"
                                                ],
                                                "variable": [
                                                        {
                                                                "key": "id",
                                                                "value": "Act-ElRq-01"
                                                        }
                                                ]
                                        }
                                },
                                "response": []
                        },
                        {
                                "name": "Get all method element types",
                                "event": [
                                        {
                                                "listen": "test",
                                                "script": {
                                                        "exec": [
                                                                "pm.test(\"Status code is
200\", function () {\r",
                                                                "
pm.response.to.have.status(200);\r",
                                                                "});\r",
                                                                "\r",
                                                                "pm.test(\"Body matches
tool string\", function () {\r",
                                                                "
pm.expect(pm.response.text()).to.include(\"tool\");\r",
                                                                "});\r",
                                                                "pm.test(\"Body matches
artefact string\", function () {\r",
                                                                "
pm.expect(pm.response.text()).to.include(\"artefact\");\r",
                                                                "});\r",
                                                                "pm.test(\"Body matches
activity string\", function () {\r",
                                                                "
pm.expect(pm.response.text()).to.include(\"activity\");\r",
                                                                "});\r",
                                                                "pm.test(\"Body matches
role string\", function () {\r",
                                                                "
pm.expect(pm.response.text()).to.include(\"role\");\r",
                                                                "});"
                                                        ],
                                                        "type": "text/javascript"
                                                }
                                        }
                                ],
                                "request": {
```

```
                                                "method": "GET",
                                                "header": [],
                                                "url": {
                                                        "raw":
"{{serverURL}}{{entryFilePath}}/method-element/types",
                                                        "host": [
                                                                "{{serverURL}}{{entryFilePath}}"
                                                        ],
                                                        "path": [
                                                                "method-element",
                                                                "types"
                                                        ]
                                                }
                                        },
                                        "response": []
                                },
                                {
                                        "name": "Get method element relation types",
                                        "event": [
                                                {
                                                        "listen": "test",
                                                        "script": {
                                                                "exec": [
                                                                        "pm.test(\"Status code is
200\", function () {\r",
                                                                        "
pm.response.to.have.status(200);\r",
                                                                        "});"
                                                                ],
                                                                "type": "text/javascript"
                                                        }
                                                }
                                        ],
                                        "request": {
                                                "method": "GET",
                                                "header": [],
                                                "url": {
                                                        "raw":
"{{serverURL}}{{entryFilePath}}/method-element/relations/types",
                                                        "host": [
                                                                "{{serverURL}}{{entryFilePath}}"
                                                        ],
                                                        "path": [
                                                                "method-element",
                                                                "relations",
                                                                "types"
                                                        ]
                                                }
                                        },
                                        "response": []
                                },
                                {
                                        "name": "Add new method element",
                                        "event": [
                                                {
                                                        "listen": "test",
                                                        "script": {
                                                                "exec": [
                                                                        "pm.test(\"Status code is
201\", function () {\r",
                                                                        "
pm.response.to.have.status(201);\r",
                                                                        "});\r",
```

```
                                                                "\r",
                                                                "pm.test(\"Return inserted
id\", function () {\r",
                                                                "    var jsonData =
pm.response.json();\r",
                                                                "
pm.expect(jsonData.id).to.eql();\r",
                                                                "});"
                                                        ],
                                                        "type": "text/javascript"
                                                }
                                        }
                                ],
                                "request": {
                                        "method": "POST",
                                        "header": [],
                                        "body": {
                                                "mode": "raw",
                                                "raw": "{\r\n    \"id\":
\"Act-ElRq-01\",\r\n    \"name\": \"Elicit Requirements\",\r\n    \"abstract\": true,\r\n
\"description\": null,\r\n    \"figure\": null,\r\n    \"type\": 3,\r\n    \"me_struct_rel\":
[],\r\n    \"activity_rel\": [],\r\n    \"artefact_rel\": []\r\n}",
                                                "options": {
                                                        "raw": {
                                                                "language": "json"
                                                        }
                                                }
                                        },
                                        "url": {
                                                "raw":
"{{serverURL}}{{entryFilePath}}/method-element",
                                                "host": [
                                                        "{{serverURL}}{{entryFilePath}}"
                                                ],
                                                "path": [
                                                        "method-element"
                                                ]
                                        }
                                },
                                "response": []
                        },
                        {
                                "name": "Add method element Image",
                                "event": [
                                        {
                                                "listen": "test",
                                                "script": {
                                                        "exec": [
                                                                "pm.test(\"Status code is
201\", function () {\r",
                                                                "
pm.response.to.have.status(201);\r",
                                                                "});"
                                                        ],
                                                        "type": "text/javascript"
                                                }
                                        }
                                ],
                                "request": {
                                        "method": "POST",
                                        "header": [],
                                        "body": {
                                                "mode": "formdata",
```

```
                                                "formdata": [
                                                        {
                                                                "key": "figure",
                                                                "type": "file",
                                                                "src": []
                                                        }
                                                ]
                                        },
                                        "url": {
                                                "raw":
"{{serverURL}}{{entryFilePath}}/method-element/:id/image",
                                                "host": [
                                                        "{{serverURL}}{{entryFilePath}}"
                                                ],
                                                "path": [
                                                        "method-element",
                                                        ":id",
                                                        "image"
                                                ],
                                                "variable": [
                                                        {
                                                                "key": "id",
                                                                "value": "Test"
                                                        }
                                                ]
                                        }
                                },
                                "response": []
                        },
                        {
                                "name": "Update method element",
                                "event": [
                                        {
                                                "listen": "test",
                                                "script": {
                                                        "exec": [
                                                                "pm.test(\"Status code is
201\", function () {\r",
                                                                "
pm.response.to.have.status(201);\r",
                                                                "});"
                                                        ],
                                                        "type": "text/javascript"
                                                }
                                        }
                                ],
                                "request": {
                                        "method": "PUT",
                                        "header": [],
                                        "body": {
                                                "mode": "raw",
                                                "raw": "{\r\n    \"name\": \"Elicit
Requirements\",\r\n    \"abstract\": true,\r\n    \"description\": null,\r\n    \"figure\":
null,\r\n    \"type\": 3,\r\n    \"me_struct_rel\": [],\r\n    \"activity_rel\": [],\r\n
\"artefact_rel\": []\r\n}",
                                                "options": {
                                                        "raw": {
                                                                "language": "json"
                                                        }
                                                }
                                        },
                                        "url": {
                                                "raw":
```

```
"{{serverURL}}{{entryFilePath}}/method-element/:id",
                                                        "host": [
                                                                "{{serverURL}}{{entryFilePath}}"
                                                        ],
                                                        "path": [
                                                                "method-element",
                                                                ":id"
                                                        ],
                                                        "variable": [
                                                                {
                                                                        "key": "id",
                                                                        "value": "Act-ElRq-01"
                                                                }
                                                        ]
                                                }
                                        },
                                        "response": []
                                },
                                {
                                        "name": "Delete method element",
                                        "event": [
                                                {
                                                        "listen": "test",
                                                        "script": {
                                                                "exec": [
                                                                        "pm.test(\"Status code is
204\", function () {\r",
                                                                        "
pm.response.to.have.status(204);\r",
                                                                        "});"
                                                                ],
                                                                "type": "text/javascript"
                                                        }
                                                }
                                        ],
                                        "request": {
                                                "method": "DELETE",
                                                "header": [],
                                                "url": {
                                                        "raw":
"{{serverURL}}{{entryFilePath}}/method-element/:id",
                                                        "host": [
                                                                "{{serverURL}}{{entryFilePath}}"
                                                        ],
                                                        "path": [
                                                                "method-element",
                                                                ":id"
                                                        ],
                                                        "variable": [
                                                                {
                                                                        "key": "id",
                                                                        "value": "Act-ElRq-01"
                                                                }
                                                        ]
                                                }
                                        },
                                        "response": []
                                }
                        ]
                },
                {
                        "name": "Criterion",
                        "item": [
```

```
                {
                        "name": "Get all criterions",
                        "event": [
                                {
                                        "listen": "test",
                                        "script": {
                                                "exec": [
                                                        "pm.test(\"Status code is
200\", function () {\r",
                                                        "
pm.response.to.have.status(200);\r",
                                                        "});"
                                                ],
                                                "type": "text/javascript"
                                        }
                                }
                        ],
                        "request": {
                                "method": "GET",
                                "header": [],
                                "url": {
                                        "raw":
"{{serverURL}}{{entryFilePath}}/criterion",
                                        "host": [
                                                "{{serverURL}}{{entryFilePath}}"
                                        ],
                                        "path": [
                                                "criterion"
                                        ]
                                }
                        },
                        "response": []
                },
                {
                        "name": "Get single criterion",
                        "event": [
                                {
                                        "listen": "test",
                                        "script": {
                                                "exec": [
                                                        "pm.test(\"Status code is
200\", function () {\r",
                                                        "
pm.response.to.have.status(200);\r",
                                                        "});"
                                                ],
                                                "type": "text/javascript"
                                        }
                                }
                        ],
                        "request": {
                                "method": "GET",
                                "header": [],
                                "url": {
                                        "raw":
"{{serverURL}}{{entryFilePath}}/criterion/:id",
                                        "host": [
                                                "{{serverURL}}{{entryFilePath}}"
                                        ],
                                        "path": [
                                                "criterion",
                                                ":id"
                                        ],
```

```
                                                "variable": [
                                                        {
                                                                "key": "id",
                                                                "value": "1"
                                                        }
                                                ]
                                        }
                                },
                                "response": []
                        },
                        {
                                "name": "Add new criterion",
                                "event": [
                                        {
                                                "listen": "test",
                                                "script": {
                                                        "exec": [
                                                                "pm.test(\"Status code is
201\", function () {\r",
                                                                "
pm.response.to.have.status(201);\r",
                                                                "});"
                                                        ],
                                                        "type": "text/javascript"
                                                }
                                        }
                                ],
                                "request": {
                                        "method": "POST",
                                        "header": [],
                                        "body": {
                                                "mode": "raw",
                                                "raw": "{\r\n    \"name\": \"Stakeholders
size\",\r\n    \"values\": [\r\n        \"High\",\r\n        \"Medium\",\r\n        \"Small\"\r\n
]\r\n}",
                                                "options": {
                                                        "raw": {
                                                                "language": "json"
                                                        }
                                                }
                                        },
                                        "url": {
                                                "raw":
"{{serverURL}}{{entryFilePath}}/criterion",
                                                "host": [
                                                        "{{serverURL}}{{entryFilePath}}"
                                                ],
                                                "path": [
                                                        "criterion"
                                                ]
                                        }
                                },
                                "response": []
                        },
                        {
                                "name": "Add new criterion Value",
                                "event": [
                                        {
                                                "listen": "test",
                                                "script": {
                                                        "exec": [
                                                                "pm.test(\"Status code is
201\", function () {\r",
```

```
                                                        "
pm.response.to.have.status(201);\r",
                                                        "});"
                                                ],
                                                "type": "text/javascript"
                                        }
                                }
                        ],
                        "request": {
                                "method": "POST",
                                "header": [],
                                "body": {
                                        "mode": "raw",
                                        "raw": "{\r\n    \"name\":\"Small\"\r\n}",
                                        "options": {
                                                "raw": {
                                                        "language": "json"
                                                }
                                        }
                                },
                                "url": {
                                        "raw":
"{{serverURL}}{{entryFilePath}}/criterion/:id/values",
                                        "host": [
                                                "{{serverURL}}{{entryFilePath}}"
                                        ],
                                        "path": [
                                                "criterion",
                                                ":id",
                                                "values"
                                        ],
                                        "variable": [
                                                {
                                                        "key": "id",
                                                        "value": "1"
                                                }
                                        ]
                                }
                        },
                        "response": []
                },
                {
                        "name": "Update value",
                        "event": [
                                {
                                        "listen": "test",
                                        "script": {
                                                "exec": [
                                                        "pm.test(\"Status code is
201\", function () {\r",
                                                        "
pm.response.to.have.status(201);\r",
                                                        "});"
                                                ],
                                                "type": "text/javascript"
                                        }
                                }
                        ],
                        "request": {
                                "method": "PUT",
                                "header": [],
                                "body": {
                                        "mode": "raw",
```

```
                                        "raw": "{\r\n    \"name\": \"High\"\r\n}",
                                        "options": {
                                                "raw": {
                                                        "language": "json"
                                                }
                                        }
                                },
                                "url": {
                                        "raw":
"{{serverURL}}{{entryFilePath}}/criterion/:id/values/:idV",
                                        "host": [
                                                "{{serverURL}}{{entryFilePath}}"
                                        ],
                                        "path": [
                                                "criterion",
                                                ":id",
                                                "values",
                                                ":idV"
                                        ],
                                        "variable": [
                                                {
                                                        "key": "id",
                                                        "value": "1"
                                                },
                                                {
                                                        "key": "idV",
                                                        "value": "1"
                                                }
                                        ]
                                }
                        },
                        "response": []
                },
                {
                        "name": "Update criterion",
                        "event": [
                                {
                                        "listen": "test",
                                        "script": {
                                                "exec": [
                                                        "pm.test(\"Status code is
201\", function () {\r",
                                                        "
pm.response.to.have.status(201);\r",
                                                        "});"
                                                ],
                                                "type": "text/javascript"
                                        }
                                }
                        ],
                        "request": {
                                "method": "PUT",
                                "header": [],
                                "body": {
                                        "mode": "raw",
                                        "raw": "{\r\n    \"name\": \"Stakeholders
expertise\"\r\n}",
                                        "options": {
                                                "raw": {
                                                        "language": "json"
                                                }
                                        }
                                },
```

```
                                        "url": {
                                                "raw":
"{{serverURL}}{{entryFilePath}}/criterion/:id",
                                                "host": [
                                                        "{{serverURL}}{{entryFilePath}}"
                                                ],
                                                "path": [
                                                        "criterion",
                                                        ":id"
                                                ],
                                                "variable": [
                                                        {
                                                                "key": "id",
                                                                "value": "1"
                                                        }
                                                ]
                                        }
                                },
                                "response": []
                        },
                        {
                                "name": "Delete criterion",
                                "event": [
                                        {
                                                "listen": "test",
                                                "script": {
                                                        "exec": [
                                                                "pm.test(\"Status code is
204\", function () {\r",
                                                                "
pm.response.to.have.status(204);\r",
                                                                "});"
                                                        ],
                                                        "type": "text/javascript"
                                                }
                                        }
                                ],
                                "request": {
                                        "method": "DELETE",
                                        "header": [],
                                        "url": {
                                                "raw":
"{{serverURL}}{{entryFilePath}}/criterion/:id",
                                                "host": [
                                                        "{{serverURL}}{{entryFilePath}}"
                                                ],
                                                "path": [
                                                        "criterion",
                                                        ":id"
                                                ],
                                                "variable": [
                                                        {
                                                                "key": "id",
                                                                "value": "1"
                                                        }
                                                ]
                                        }
                                },
                                "response": []
                        },
                        {
                                "name": "Delete criterion Value",
                                "event": [
```

```
                                        {
                                                "listen": "test",
                                                "script": {
                                                        "exec": [
                                                                "pm.test(\"Status code is
204\", function () {\r",
                                                                "
pm.response.to.have.status(204);\r",
                                                                "});"
                                                        ],
                                                        "type": "text/javascript"
                                                }
                                        }
                                ],
                                "request": {
                                        "method": "DELETE",
                                        "header": [],
                                        "url": {
                                                "raw":
"{{serverURL}}{{entryFilePath}}/criterion/:id/values/:idV",
                                                "host": [
                                                        "{{serverURL}}{{entryFilePath}}"
                                                ],
                                                "path": [
                                                        "criterion",
                                                        ":id",
                                                        "values",
                                                        ":idV"
                                                ],
                                                "variable": [
                                                        {
                                                                "key": "id",
                                                                "value": "1"
                                                        },
                                                        {
                                                                "key": "idV",
                                                                "value": "1"
                                                        }
                                                ]
                                        }
                                },
                                "response": []
                        }
                ]
        },
        {
                "name": "Goal",
                "item": [
                        {
                                "name": "Get all goals",
                                "event": [
                                        {
                                                "listen": "test",
                                                "script": {
                                                        "exec": [
                                                                "pm.test(\"Status code is
200\", function () {\r",
                                                                "
pm.response.to.have.status(200);\r",
                                                                "});"
                                                        ],
                                                        "type": "text/javascript"
                                                }
```

```
                                        }
                                ],
                                "request": {
                                        "method": "GET",
                                        "header": [],
                                        "url": {
                                                "raw":
"{{serverURL}}{{entryFilePath}}/goal",
                                                "host": [
                                                        "{{serverURL}}{{entryFilePath}}"
                                                ],
                                                "path": [
                                                        "goal"
                                                ]
                                        }
                                },
                                "response": []
                        },
                        {
                                "name": "Add new goal",
                                "event": [
                                        {
                                                "listen": "test",
                                                "script": {
                                                        "exec": [
                                                                "pm.test(\"Status code is
201\", function () {\r",
                                                                "
pm.response.to.have.status(201);\r",
                                                                "});"
                                                        ],
                                                        "type": "text/javascript"
                                                }
                                        }
                                ],
                                "request": {
                                        "method": "POST",
                                        "header": [],
                                        "body": {
                                                "mode": "raw",
                                                "raw": "{\r\n    \"name\": \"Elicit
requirements\"\r\n}",
                                                "options": {
                                                        "raw": {
                                                                "language": "json"
                                                        }
                                                }
                                        },
                                        "url": {
                                                "raw":
"{{serverURL}}{{entryFilePath}}/goal",
                                                "host": [
                                                        "{{serverURL}}{{entryFilePath}}"
                                                ],
                                                "path": [
                                                        "goal"
                                                ]
                                        }
                                },
                                "response": []
                        }
                ]
        },
```

```
                {
                        "name": "Documentation",
                        "item": [
                                {
                                        "name": "OpenAPI",
                                        "request": {
                                                "method": "GET",
                                                "header": [],
                                                "url": {
                                                        "raw":
"{{serverURL}}{{documentationFilePath}}",
                                                        "host": [

"{{serverURL}}{{documentationFilePath}}"
                                                        ]
                                                }
                                        },
                                        "response": []
                                }
                        ]
                }
        ],
        "event": [
                {
                        "listen": "prerequest",
                        "script": {
                                "type": "text/javascript",
                                "exec": [
                                        ""
                                ]
                        }
                },
                {
                        "listen": "test",
                        "script": {
                                "type": "text/javascript",
                                "exec": [
                                        ""
                                ]
                        }
                }
        ],
        "variable": [
                {
                        "key": "serverURL",
                        "value": "http://gessi3.essi.upc.edu:1026",
                        "type": "string"
                },
                {
                        "key": "entryFilePath",
                        "value": "/index.php",
                        "type": "string"
                },
                {
                        "key": "documentationFilePath",
                        "value": "/documentation/api.php",
                        "type": "string"
                }
        ]
}
```