Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA INFORMÁTICA DE GESTIÓN Y SISTEMAS DE INFORMACIÓN

TRABAJO FIN DE GRADO

DISEÑO Y DESARROLLO DE UN FRONT-END PARA LA HERRAMIENTA ADESMUS

Alumno/Alumna: Barquín Sainz, Pablo

Director/Directora: Villamañe Gironés, Mikel

Curso: 2017-2018

Fecha: Lunes, 16 de julio de 2018

Resumen

La informática ha evolucionado considerablemente en los últimos años. Los primeros sistemas estaban únicamente destinados a su uso por parte de personas con conocimientos técnicos, y eran algo lejano y desconocido para los ciudadanos de a pie. Sin embargo, viendo el increíble potencial que estos sistemas podían ofrecer, progresivamente se han ido orientando también a su uso por el público general.

Esto hacía necesario que se le otorgara una importancia mucho mayor a su usabilidad: los usuarios de a pie no pueden trabajar mediante comandos, necesitan interfaces con elementos familiares y orientadas expresamente hacia ellos, si el objetivo es precisamente conseguir un número elevado de ventas para orientar la informática a todos los públicos y hacer de ella un negocio. Es, precisamente, el objetivo de este Trabajo de Fin de Grado.

AdESMuS es una herramienta de gestión universitaria que carece de una interfaz de usuario. Es por tanto, inutilizable para una persona de a pie. En este proyecto se ha desarrollado una interfaz de usuario que cubra al sistema como una máscara y en la que se haya hecho especial hincapié en la usabilidad; a base de los elementos clásicos de las interfaces: ventanas, botones, campos, listas, etc.

El desarrollo también ha incluido un estudio realizado con diversos usuarios del entorno, con distintos niveles de conocimientos informáticos, para poner a prueba el sistema creado y cuantificar las mejoras que éste ha implementado.

Laburpena

Informatika nabarmen hazi da azken urteotan. Lehenengo sistemak ezagutza tekniko duten pertsonei bakarrik zeuden zuzenduta, eta herritar arruntentzako gauza arraro eta ezezagunak ziren. Hala eta guztiz ere, sistemek eskaintzen duten potentziala ikusita, pixkanaka-pixkanaka bere erabilera hedatu zen eta gaur egun ia jende guztiak erabiltzen ditu.

Horregatik sistemen erabilgarritasunari garrantzi handiagoa eman behar zaio: erabiltzaile arruntek ezin dute komandoen bidez lanean ibili, beraz, interfaze grafikoak behar dituzte. Horrela sistemak publiko orokorrago batera bideratzen dira eta sistemen garapena negozio bat egiteko aukera sortzen da. Hau da, zehazki, Gradu Amaierako Lan honen helburua da.

AdESMuS unibertsitate-kudeaketarako tresna da. Horregatik, ezin du edonor erabili. Proiektu honetan, maskara gisa sistema estaltzen duen erabiltzailearentzako interfazea garatu da erabilgarritasunari arreta berezia jarriz eta interfazeen elementu klasikoak erabiliz: leihoak, botoiak, eremuak, zerrendak, etab.

Garapenaren aparte, erabiltzaile batzuekin erabilgarritasun azterketa bat egin da. Erabiltzaileek ordenagailuei buruzko ezagutza maila desberdina zituen, eta garatutako sistema probatzeko eta hobekuntzak neurtzeko balio izan du.

Abstract

Computing has evolved greatly over the years. The first systems were only aimed at people with some technical knowkedge and were something distant and unknown for ordinary people. However, the amazing potential these systems offered has allowed common people to use them.

It was necessary to give importance to the usability: ordinary people cannot work by using commands. Instead, they need interfaces with familiar elements an focused just on them if the goal is a high number of sales. This would mean that computing is accessible to everyone and therefore, it can boost business. This is exactly the purpose of this End of Degree Project.

AdESMuS is a tool for university management that lacks a user interface. Consequently, it cannnot be used by the common man. This project has developed an interface that covers the system like a mask, putting special emphasis on usability and containing conventional elements of interfaces: windows, buttons, fields, lists, etc.

This project also includes some studies involving users with different levels of computing knowledge, in order to test how well the system works and assess the improvements it has meant.

Índice

7

8

9

A

B

Índice de figuras

Índice de tablas

1. Introducción

Los primeros sistemas informáticos modernos que se desarrollaron se basaban únicamente en comandos, claves y códigos que el usuario debía conocer e introducir exactamente como eran; ya que, de lo contrario, daban lugar a un error. Entonces, la informática no estaba orientada a todos los públicos; sino más bien a personas cuyo trabajo o estudios estaban relacionados con ella, como programadores.

Conforme los sistemas comenzaban a ser más potentes y la informática fue desarrollándose y ofreciendo más y más funciones, personas ajenas al mundo de la informática comenzaban a hacer uso de ella; y comenzaba a ser necesario un cambio en la usabilidad de los sistemas; de manera que fuesen más sencillos, más claros, más cómodos y que no obligasen al usuario a memorizar comandos ni parámetros de configuración. En otras palabras, lo que cada vez más usuarios necesitaban era una interfaz gráfica.

Las interfaces gráficas comenzaron a popularizarse en las décadas de los 80 y 90 del siglo XX, y desde entonces han facilitado enormemente las tareas a los usuarios; además de acercar las ventajas que aporta la tecnología a usuarios sin conocimientos de ella.

Este tipo de recursos funciona como una máscara que cubre a la interfaz de línea de comandos y media entre ella y el usuario. Es decir, transforman las acciones que el usuario realiza sobre ellas (como pulsar botones, marcar casillas, introducir texto, etc.), más intuitivas para éste, en los propios comandos que de manera natural se introducirían en el sistema.

Hoy en día, son muy pocas las herramientas informáticas que se desarrollan sin hacer uso de una interfaz. De hecho, se han popularizado de tal manera que el diseño de interfaces gráficas es hoy en día un área más del desarrollo de software. Las interfaces gráficas son el punto donde confluyen la ingeniería informática, el arte y la psicología; que se unen en busca de realizar el mejor prototipo posible: usable de cara al usuario y con un diseño atractivo y apropiado para la situación. Estas características influirán en el resultado final del producto tanto como el propio software; y por tanto, jugarán un papel importante en los beneficios económicos obtenidos a partir del mismo.

Precisamente el objetivo de este Trabajo de Fin de Grado es el desarrollo de una interfaz gráfica de usuario para un sistema ya desarrollado pero que carece de ella: AdESMuS.

1.1. Motivaciones para la elección del proyecto

La principal razón que me ha llevado a la elección de este trabajo es que, personalmente, el diseño de interfaces es una parte del desarrollo que siempre me ha motivado. Dado que desde hace tiempo me había planteado la posibilidad de orientarme hacia esta rama de la Ingeniería Informática, tanto en lo que a formación se refiere como profesionalmente; por lo que creo bastante conveniente la realización de un Trabajo de Fin de Grado cuya tarea sea, precisamente, el desarrollo de una interfaz.

Asimismo, el desarrollo de este proyecto de realiza íntegramente con tecnologías web (MEAN stack), lo que fue otro de los motivos principales; ya que durante el grado hemos tenido pocas oportunidades de aprender este tipo de recursos, excluyendo los últimos años; y el desarrollo web está en auge, en favor de las aplicaciones instalables o enfocadas a un único sistema operativo (por su inmensa escalabilidad y la escasa necesidad de realizar configuraciones o instalaciones). Creía muy útil aplicar estas tecnologías en el TFG, ya que de este modo me auto-formaría en ese campo durante la realización del proyecto.

Por último, y siendo de menor importancia, destacar otros motivos como: que el proyecto se realiza con software libre, que parte de una herramienta ya desarrollada y que abarca múltiples lenguajes.

2. Planteamiento inicial

En este apartado se indicará el planteamiento inicial del proyecto: objetivos, alcance, etc. Asimismo, se expondrán las aplicaciones y herramientas utilizadas en el desarrollo del mismo. Se analizarán los posibles riesgos que surgen durante la realización del proyecto, así como una estimación del coste económico y temporal.

2.1. Descripción

AdESMuS es una herramienta de evaluación de TFGs ya existente, cuyo código es el resultado del desarrollo de un Trabajo de Fin de Grado realizado en el año 2015.

AdESMuS carece de una interfaz gráfica; por lo que, el objetivo de este TFG es, precisamente, el desarrollo de una interfaz gráfica de usuario para la herramienta. La parte interna del sistema (la parte ya realizada) es lo que se conoce como back-end; mientras que la interfaz de usuario (la parte que se realizará) se denomina front-end. Con ambos términos se hará referencia a cada una de las dos partes de ahora en adelante. En resumen, el objetivo del proyecto es la realización del front-end para un programa que por momento cuenta únicamente con el back-end; de manera que el sistema ofrezca las mismas funciones, pero de una manera mucho más sencilla, rápida y usable.

Finalmente, y una vez terminada esta parte; se podrá realizar una ampliación del back-end. Se decidirá si se continúa el proyecto por este camino o se da por finalizado en función del tiempo disponible una vez terminada la primera parte.

2.2. Definiciones, acrónimos y abreviaturas

En este apartado se expondrán algunos conceptos básicos que tendrán gran importancia en el proyecto; y, por tanto, conocerlos es vital para comprender el proyecto.

2.2.1. Back-end y front-end

Cuando se desarrolla un programa, habitualmente el código se separa en dos grandes bloques: back-end, que consiste en el propio código para la ejecución interna del programa; y front-end, que es la parte que constituye la interfaz de usuario; y que media entre éste y el back-end. En teoría son dos sistemas distintos, aunque funcionen conjuntamente.

2.2.2. JSON

JavaScript Object Notation: es un tipo de archivo, utilizado a menudo en los servicios web para el transporte y almacenamiento de información. Funciona mediante clave-valor, es decir, para cada clave (cadena de caracteres) se corresponde un elemento, que puede ser un valor, un array o incluso otro JSON.

2.2.3. Angular

Angular es el framework utilizado durante para el desarrollo (ver 2.7. Herramientas).

2.2.4. Node.js

Node.js es una tecnología para el desarrollo de la parte back-end de los sistemas web. Node.js media entre el programador y el código JavaScript (en el que está basada) para crear back-ends más ricos de una manera mucho más sencilla y automatizada.

Figura 1 - Logotipo de Node.js

2.2.5. API REST

Las aplicaciones API REST son sistemas basados en peticiones: el usuario (o el front-end) invoca las peticiones que éstos ofrecen y devuelven los datos correspondientes tras ejecutar el código correspondiente. El back-end de AdESMuS es un API REST.

2.3. Objetivos

El proyecto buscará maximizar la usabilidad en el sistema creado, que posteriormente se evaluará mediante un pequeño estudio de usabilidad. A continuación se explicará en profundidad el objetivo.

Características buscadas

Se buscará que la interfaz desarrollada no se limite únicamente a transformar las acciones del usuario en código; sino que cumpla ciertos estándares de usabilidad. La usabilidad es la búsqueda de tres pautas en un sistema: eficiencia, efectividad y satisfacción por parte del usuario **[10]**. Asimismo, uno de los principales objetivos del Trabajo de Fin de Grado, ya con menor importancia, será que la interfaz tenga un diseño atractivo y adecuado para el contexto; ya que esto influirá en la apariencia del producto final.

Eficiencia

En una aplicación, la eficiencia consiste en el objetivo de ofrecer la mayor cantidad de información y funcionalidades posibles con el menor número de acciones por parte del usuario: clicks, botones, etc. Es decir, se buscará que el usuario encuentre las funcionalidades y el acceso a la información lo antes posible.

Por ejemplo, se buscará que se pueda llegar a la edición de elementos de una forma rápida. En lugar de buscar el elemento, pulsar en él y una vez abierto, permitir modificarlo, dar la opción de edición lo antes posible.

Otro objetivo es implementar un buscador de acceso rápido que se muestre al principio, permitiendo al usuario teclear y acceder a cualquier elemento de la base de datos.

Efectividad

La efectividad consiste en que cada funcionalidad realice de manera correcta la tarea para la que está programada. Es decir, que el funcionamiento sea exactamente el necesario.

Dado que AdESMuS no implementa el borrado en cascada, se creará un borrado en cadena que elimine todos los elementos asociados a un determinado elemento, para así poder eliminarlo.

Satisfacción por parte del usuario

En una aplicación, la satisfacción por parte del usuario se consigue haciendo lo posible por que éste reciba una impresión positiva del sistema desarrollado. En primer lugar, debido al funcionamiento de cada área según lo esperado; y además, mostrándole notas de ayuda o informativas y evitando los mensajes de error alarmantes y carentes de información para solucionarlos, para evitar que el funcionamiento ocasione frustraciones en el usuario.

En esta aplicación, se buscará mostrar sólo las acciones disponibles, para evitar que se produzcan confusiones o que el usuario llegue a creer que ha realizado

erróneamente alguna acción, y se buscará mostrar mensajes de error constructivos, entre otros aspectos.

Por ejemplo, la situación en la que la sesión caduca deberá ser contemplada. Se informará al usuario del suceso y se le llevará automáticamente a la página de inicio de sesión.

Preservar la seguridad

En todo momento se deberá preservar la seguridad que AdESMuS implementa. De nada sirve que AdESMuS implemente medidas de seguridad, si son ignoradas por el back-end. Cuando varios sistemas trabajan en común, un fallo de seguridad en uno de ellos puede comprometer a todo el sistema.

Para ello, se estudiarán las cinco capas de seguridad que el back-end implementa, y se tomarán las medidas pertinentes en cada una de ellas.

2.4. Alcance

En este apartado se expondrán los distintos paquetes y tareas en las que se ha dividido la realización del proyecto, así como las características de cada una de ellas (descripción, tareas precedentes y salidas) y la duración que tomará el desarrollo de las mismas.

El sistema AdESMuS (y, por tanto, el front-end) están divididos en siete partes, que de ahora en adelante se referirán con el nombre de áreas: usuarios, asignaturas, centros, departamentos, titulaciones, grupos y matrículas.

El trabajo estará dividido en seis grandes paquetes; que se estarán formados por distinto número de tareas, sumando un total de 39 en todo el proyecto. Estas tareas se realizarán siguiendo un determinado orden, independiente de los paquetes a los que pertenezca cada una de ellas y que será expuesto en el apartado 2.5. Planificación temporal. El EDT (Esquema de Descomposición Temporal), en el que se muestran todos los paquetes y las tareas que componen cada uno se ve en la Figura 2 - EDT.

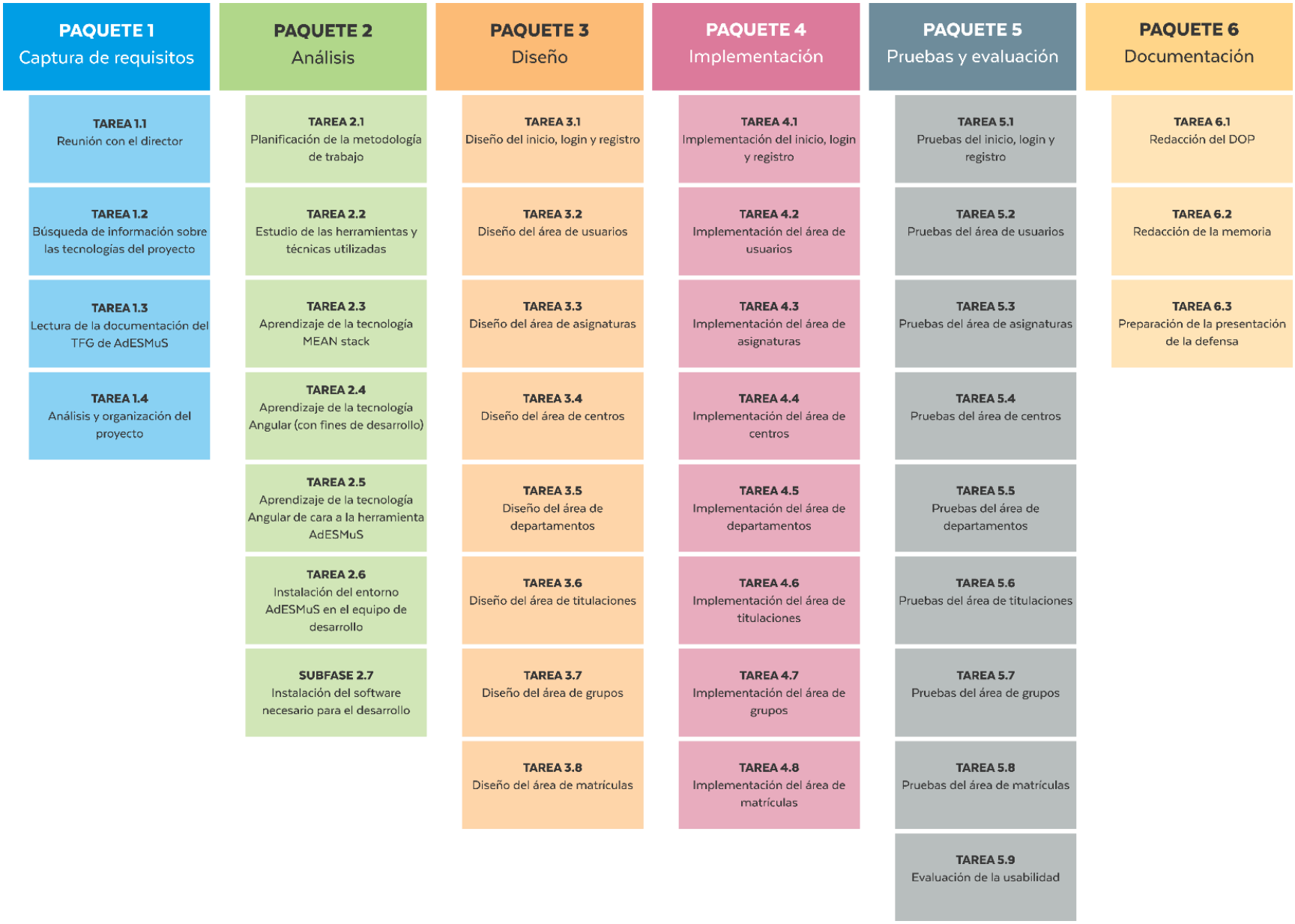

PAQUETE 1
Captura de requisitos
TAREA 1.1
Reunión con el director
TAREA 1.2
Búsqueda de información sobre las tecnologías del proyecto
TAREA 1.3
Lectura de la documentación del TFG de AdESMuS
TAREA 1.4
Análisis y organización del proyecto
PAQUETE 2
Análisis
TAREA 2.1
Planificación de la metodología de trabajo
TAREA 2.2
Estudio de las herramientas y técnicas utilizadas
TAREA 2.3
Aprendizaje de la tecnología MEAN stack
TAREA 2.4
Aprendizaje de la tecnología Angular (con fines de desarrollo)
TAREA 2.5
Aprendizaje de la tecnología Angular de cara a la herramienta AdESMuS
TAREA 2.6
Instalación del entorno AdESMuS en el equipo de desarrollo
SUBFASE 2.7
Instalación del software necesario para el desarrollo
PAQUETE 3
Diseño
TAREA 3.1
Diseño del inicio, login y registro
TAREA 3.2
Diseño del área de usuarios
TAREA 3.3
Diseño del área de asignaturas
TAREA 3.4
Diseño del área de centros
TAREA 3.5
Diseño del área de departamentos
TAREA 3.6
Diseño del área de titulaciones
TAREA 3.7
Diseño del área de grupos
TAREA 3.8
Diseño del área de matrículas
PAQUETE 4
Implementación
TAREA 4.1
Implementación del inicio, login y registro
TAREA 4.2
Implementación del área de usuarios
TAREA 4.3
Implementación del área de asignaturas
TAREA 4.4
Implementación del área de centros
TAREA 4.5
Implementación del área de departamentos
TAREA 4.6
Implementación del área de titulaciones
TAREA 4.7
Implementación del área de grupos
TAREA 4.8
Implementación del área de matrículas
PAQUETE 5
Pruebas y evaluación
TAREA 5.1
Pruebas del inicio, login y registro
TAREA 5.2
Pruebas del área de usuarios
TAREA 5.3
Pruebas del área de asignaturas
TAREA 5.4
Pruebas del área de centros
TAREA 5.5
Pruebas del área de departamentos
TAREA 5.6
Pruebas del área de titulaciones
TAREA 5.7
Pruebas del área de grupos
TAREA 5.8
Pruebas del área de matrículas
TAREA 5.9
Evaluación de la usabilidad
PAQUETE 6
Documentación
TAREA 6.1
Redacción del DOP
TAREA 6.2
Redacción de la memoria
TAREA 6.3
Preparación de la presentación de la defensa

Figura 2 - EDT

2.4.1. Ciclos de vida

El desarrollo de software tiene un ciclo de vida desde el momento en que se decide realizar hasta la finalización (o entrega) del mismo. Este ciclo tiene unas fases similares en la mayoría de desarrollos de software.

En el desarrollo de este proyecto, se ha optado por un ciclo de vida con cuatro fases (ver Figura 3 - Ciclo de vida del desarrollo). Se realizan dos pruebas, las pruebas del código (ver 8. Pruebas unitarias) y una revisión final, para así asegurarse de la no existencia de errores. Este ciclo de vida está directamente relacionado con la Estructura de Descomposición del Proyecto, simplemente es otra forma de representar el proceso, centrándose en el desarrollo y en el código. El ciclo de vida se aplica por separado a cada uno de los prototipos (ver Prototipos del ciclo de vida).

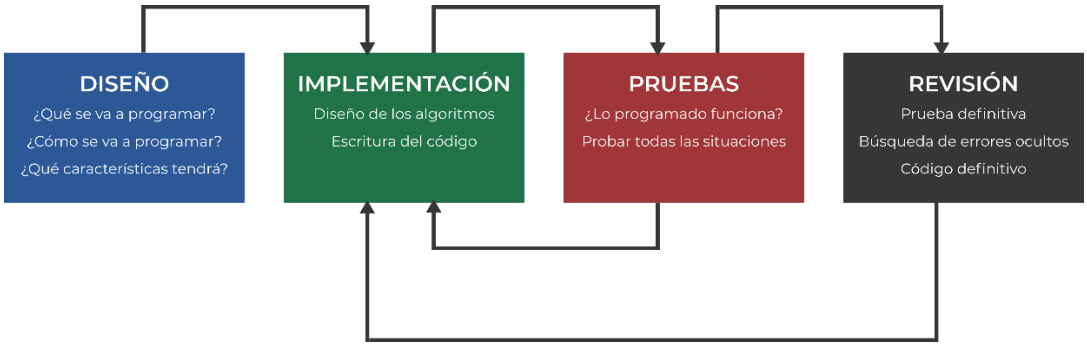


Figura 3 - Ciclo de vida del desarrollo

Prototipos del ciclo de vida

En el front-end se considerarán ocho prototipos, que tratan distintos tipos de datos y son independientes entre sí. Son los siguientes:

- Inicio, login y registro: Toda la parte correspondiente al inicio de la aplicación, el login y el registro.
- Usuarios: Toda la parte correspondiente a la operación sobre los datos en la base de datos de los usuarios: administradores, docentes y estudiantes.
- Asignaturas: Toda la parte correspondiente a la operación sobre los datos en la base de datos de las asignaturas.
- Centros: Toda la parte correspondiente a la operación sobre los datos en la base de datos de los centros.
- Departamentos: Toda la parte correspondiente a la operación sobre los datos en la base de datos de los departamentos.
- Titulaciones: Toda la parte correspondiente a la operación sobre los datos en la base de datos de las titulaciones.
- Grupos: Toda la parte correspondiente a la operación sobre los datos en la base de datos de los grupos.
- Matrículas: Toda la parte correspondiente a la operación sobre los datos en la base de datos de las matrículas.

2.4.2. PAQUETE 1: Captura de requisitos

En este paquete, el programador estudiará la naturaleza del proyecto, sus características, sus objetivos, etc. Se realizarán las reuniones pertinentes con el director del proyecto para recibir información acerca del mismo y aclarar las distintas dudas que puedan surgir.

TAREA 1.1: Reunión con el director

Tabla 1 - TAREA 1.1

Descripción	Consiste en realizar una reunión con el director en la que se explicará a fondo el objetivo del proyecto: situación inicial de la herramienta AdESMuS, características de la misma, objetivos del front-end que se desea desarrollar, etc. así como algunas características básicas en torno a las cuales girará el desarrollo: arquitectura REST, etc. Explicación de la posibilidad de ampliar del back-end.
Tareas precedentes	Ninguna.
Salidas	Notas tomadas durante la reunión acerca del proyecto, así como los archivos recibidos por parte del tutor.
Duración	2 horas.

TAREA 1.2: Búsqueda de información sobre las tecnologías del proyecto

Tabla 2 - TAREA 1.2

Descripción	Consiste en una búsqueda por parte del programador de información acerca de las tecnologías, herramientas y arquitectura del proyecto: tanto sobre en las que está realizada la herramienta inicial (el back-end) como sobre las que serán utilizadas durante el desarrollo: API REST, modelo de back-end y front-end, uso de tokens, MEAN stack, etc.
Tareas precedentes	1.1.
Entradas	Notas tomadas durante la reunión acerca del proyecto, así como los archivos recibidos por parte del tutor.
Salidas	Notas tomadas sobre las tecnologías sobre las que se haya hecho el aprendizaje.
Duración	5 horas.

TAREA 1.3: Lectura de la documentación del TFG de AdESMuS

Tabla 3 - TAREA 1.3

Descripción	Lectura y análisis de la documentación de la herramienta inicial de AdESMuS, que permita comprender su funcionamiento y características, haciendo especial énfasis en los conceptos de funcionalidades, implementación y arquitectura.
Tareas precedentes	1.2.
Salidas	Conceptos básicos anotados acerca de AdESMuS.
Duración	5 horas.

TAREA 1.4: Análisis y organización del proyecto

Tabla 4 - TAREA 1.4

Descripción	Análisis completo del proyecto que se va a realizar: objetivos, manera en la que se va a abordar, división de tareas, organización semanal del tiempo disposible, etc.
Tareas precedentes	1.3.
Entradas	Conceptos básicos anotados acerca de AdESMuS.
Salidas	Documento informal en el que se expliquen algunas características básicas de la metodología que se va a llevar.
Duración	5 horas.

2.4.3. PAQUETE 2: Análisis

Consiste en un estudio del proyecto más profundo que la captura de requisitos y más centrado en el propio desarrollo del mismo; donde el programador analizará las herramientas, tecnologías o métodos que utilizará y adquirirá los conocimientos necesarios sobre ellos; así como sobre el propio proyecto.

TAREA 2.1: Planificación de la metodología de trabajo

Tabla 5 - TAREA 2.1

Descripción	Se planificará la metodología con la que se va a llevar a cabo el trabajo: cuándo se comenzará, equipo en el cual tendrá lugar el desarrollo, política de copias de seguridad (frecuencia de realización, datos que se volcarán, lugar de almacenamiento), etc.
Tareas precedentes	1.4.
Entradas	Notas tomadas durante la reunión acerca del proyecto.
Salidas	Documento informal en el que se especifique la organización que se va a llevar en la realización del proyecto.
Duración	5 horas

TAREA 2.2: Estudio de las herramientas y técnicas utilizadas

Tabla 6 - TAREA 2.2

Descripción	Analizando exhaustivamente el proyecto, sus características y sus objetivos, así como investigando sobre las tecnologías, herramientas o métodos disponibles, se decidirá cuáles serán los escogidos para el desarrollo.
Tareas precedentes	2.1.
Salidas	Notas tomadas sobre las tecnologías sobre las que se haya hecho el aprendizaje.
Duración	3 horas

TAREA 2.3: Aprendizaje de la tecnología MEAN stack

Tabla 7 - TAREA 2.3

Descripción	Se realizará un auto-aprendizaje sobre el uso de MEAN stack (MongoDB, Express, Angular y Node.js), dado que es la base del proyecto. Se le dará un enfoque general, debido a que únicamente es la interfaz la que se desarrollará, en un principio.
Tareas precedentes	6.1.
Duración	13 horas

TAREA 2.4: Aprendizaje de la tecnología Angular (con fines de desarrollo)

Tabla 8 - TAREA 2.4

Descripción	Se realizará un auto-aprendizaje sobre el framework Angular. En este caso, el enfoque será de cara al desarrollo; razón por la cual se realiza esta profundización.
Tareas precedentes	2.3.
Duración	13 horas

TAREA 2.5: Aprendizaje de la tecnología Angular de cara a la herramienta AdESMuS

Tabla 9 - TAREA 2.5

Descripción	Se buscará el vínculo entre los conocimientos sobre MEAN stack (y, especialmente, sobre Angular) adquiridos y la herramienta AdESMuS para que sea posible aplicarlos y trabajar sobre ella.
Tareas precedentes	2.4.
Entradas	Conceptos básicos anotados acerca de AdESMuS.
Duración	9 horas

TAREA 2.6: Instalación del entorno de AdESMuS en el equipo de desarrollo

Tabla 10 - TAREA 2.6

Descripción	Instalación del entorno completo de AdESMuS en el equipo propio, en el que se va desarrollar el proyecto. Se probarán las funciones básicas de la herramienta.
Tareas precedentes	2.5.
Salidas	Herramienta y entorno AdESMuS instalado en el equipo de desarrollo.
Duración	5 horas

TAREA 2.7: Instalación del software necesario para el desarrollo

Tabla 11 - TAREA 2.7

Descripción	Instalación inicial del software y las herramientas que se utilizarán durante el desarrollo.
Tareas precedentes	2.6.
Salidas	Software necesario para el desarrollo instalado.
Duración	5 horas

2.4.4. PAQUETE 3: Diseño

Se realizará el diseño por separado del front-end cada una de las áreas de la herramienta, acompañado de los diagramas que sean necesarios: casos de uso, modelos de dominio, esquemas de la base de datos, etc. En este caso, dado que el tanto el back-end como la base de datos ya están diseñadas y en un principio no van a sufrir ningún tipo de modificación, no requerirán diseño.

TAREA 3.1: Diseño del inicio, login y registro

Tabla 12 - TAREA 3.1

Descripción	Se diseñará el inicio, login y registro del sistema. Es decir, la pantalla inicial y las opciones que permiten iniciar sesión o registrarse para los usuarios que no lo estén y tengan los permisos necesarios.
Tareas precedentes	2.7.
Salidas	Prototipo a ordenador del inicio, login y registro.
Duración	20 horas

TAREA 3.2: Diseño del área de usuarios

Tabla 13 - TAREA 3.2

Descripción	Se diseñará el área de usuarios; es decir, todas las operaciones que se pueden realizar sobre los tres tipos de usuario en el sistema, según los permisos con los que se cuente: administrador, docente y estudiante.
Tareas precedentes	4.1.
Salidas	Prototipo a ordenador del área de usuarios.
Duración	20 horas

TAREA 3.3: Diseño del área de asignaturas

Tabla 14 - TAREA 3.3

Descripción	Se diseñará el área de asignaturas; es decir, todas las operaciones que se pueden realizar sobre las asignaturas.
Tareas precedentes	4.2.
Salidas	Prototipo a ordenador del área de asignaturas.
Duración	20 horas

TAREA 3.4: Diseño del área de centros

Tabla 15 - TAREA 3.4

Descripción	Se diseñará el área de centros; es decir, todas las operaciones que se pueden realizar sobre los centros.
Tareas precedentes	4.3.
Salidas	Prototipo a ordenador del área de centros.
Duración	20 horas

TAREA 3.5: Diseño del área de departamentos

Tabla 16 - TAREA 3.5

Descripción	Se diseñará el área de departamentos; es decir, todas las operaciones que se pueden realizar sobre los departamentos.
Tareas precedentes	4.4.
Salidas	Prototipo a ordenador del área de departamentos.
Duración	20 horas

TAREA 3.6: Diseño del área de titulaciones

Tabla 17 - TAREA 3.6

Descripción	Se diseñará el área de titulaciones; es decir, todas las operaciones que se pueden realizar sobre las titulaciones.
Tareas precedentes	4.5.
Salidas	Prototipo a ordenador del área de titulaciones.
Duración	20 horas

TAREA 3.7: Diseño del área de grupos

Tabla 18 - TAREA 3.7

Descripción	Se diseñará el área de grupos; es decir, todas las operaciones que se pueden realizar sobre los grupos.
Tareas precedentes	4.6.
Salidas	Prototipo a ordenador del área de grupos.
Duración	20 horas

TAREA 3.8: Diseño del área de matrículas

Tabla 19 - TAREA 3.8

Descripción	Se diseñará el área de matrículas; es decir, todas las operaciones que se pueden realizar sobre las matrículas.
Tareas precedentes	4.7.
Salidas	Prototipo a ordenador del área de matrículas.
Duración	20 horas

2.4.5. PAQUETE 4: Implementación

Para cada área de AdESMuS, una vez realizado el diseño definitivo y especificado de manera clara lo que cómo deberá ser el resultado final, se procederá escribir el código para implementar el front-end.

TAREA 4.1: Implementación del inicio, login y registro

Tabla 20 - TAREA 4.1

Descripción	Se implementará el inicio, login y registro del sistema. Es decir, la pantalla inicial y las opciones que permiten iniciar sesión o registrarse para los usuarios que no lo estén y tengan los permisos necesarios.
Tareas precedentes	3.1.
Entradas	Prototipo a ordenador del inicio, login y registro.
Salidas	Inicio, login y registro implementados y funcionando según lo diseñado.
Duración	25 horas

TAREA 4.2: Implementación del área de usuarios

Tabla 21 - TAREA 4.2

Descripción	Se implementará el área de usuarios; es decir, todas las operaciones que se pueden realizar sobre los tres tipos de usuario en el sistema, según los permisos con los que se cuente: administrador, docente y estudiante.
Tareas precedentes	3.2.
Entradas	Prototipo a ordenador del área de usuarios.
Salidas	Área de usuarios implementada y funcionando según lo diseñado.
Duración	25 horas

TAREA 4.3: Implementación del área de asignaturas

Tabla 22 - TAREA 4.3

Descripción	Se implementará el área de asignaturas; es decir, todas las operaciones que se pueden realizar sobre las asignaturas.
Tareas precedentes	3.3.
Entradas	Prototipo a ordenador del área de asignaturas.
Salidas	Área de usuarios asignaturas y funcionando según lo diseñado.
Duración	25 horas

TAREA 4.4: Implementación del área de centros

Tabla 23 - TAREA 4.4

Descripción	Se implementará el área de centros; es decir, todas las operaciones que se pueden realizar sobre los centros.
Tareas precedentes	3.4.
Entradas	Prototipo a ordenador del área de centros.
Salidas	Área de centros implementada y funcionando según lo diseñado.
Duración	25 horas

TAREA 4.5: Implementación del área de departamentos

Tabla 24 - TAREA 4.5

Descripción	Se implementará el área de departamentos; es decir, todas las operaciones que se pueden realizar sobre los departamentos.
Tareas precedentes	3.5.
Entradas	Prototipo a ordenador del área de departamentos.
Salidas	Área de departamentos implementada y funcionando según lo diseñado.
Duración	25 horas

TAREA 4.6: Implementación del área de titulaciones

Tabla 25 - TAREA 4.6

Descripción	Se implementará el área de titulaciones; es decir, todas las operaciones que se pueden realizar sobre las titulaciones.
Tareas precedentes	3.6.
Entradas	Prototipo a ordenador del área de titulaciones.
Salidas	Área de titulaciones implementada y funcionando según lo diseñado.
Duración	25 horas

TAREA 4.7: Implementación del área de grupos

Tabla 26 - TAREA 4.7

Descripción	Se implementará el área de grupos; es decir, todas las operaciones que se pueden realizar sobre los grupos.
Tareas precedentes	3.7.
Entradas	Prototipo a ordenador del área de grupos.
Salidas	Área de grupos implementada y funcionando según lo diseñado.
Duración	25 horas

TAREA 4.8: Implementación del área de matrículas

Tabla 27 - TAREA 4.8

Descripción	Se implementará el área de matrículas; es decir, todas las operaciones que se pueden realizar sobre las matrículas.
Tareas precedentes	3.8.
Entradas	Prototipo a ordenador del área de matrículas.
Salidas	Área de matrículas implementada y funcionando según lo diseñado.
Duración	25 horas

2.4.6. PAQUETE 5: Pruebas y evaluación

Las pruebas se corresponden con la revisión de la implementación de cada una de las áreas de AdESMuS, para asegurarse de que el sistema funciona según lo esperado. Se realiza tras la implementación, y se verifican todos los casos posibles, poniendo a prueba la aplicación para asegurarse de que el funcionamiento es correcto. En caso de encontrar errores, se debe analizar el código para buscar en qué punto se encuentra el problema y reimplementar las partes que así lo necesiten. Formalmente, esta situación está contemplada; y las horas extra que se necesiten en el caso de encontrar errores en esta etapa se consideran parte del paquete de Pruebas. Por este motivo, se ha estimado una duración media de cuatro horas en cada área.

La última fase, la evaluación, es el estudio de usabilidad realizado para testear la aplicación (ver 7. Evaluación de la usabilidad)

TAREA 5.1: Pruebas del inicio, login y registro

Tabla 28 - TAREA 5.1

Descripción	Se probará el inicio, login y registro del sistema. Es decir, la pantalla inicial y las opciones que permiten iniciar sesión o registrarse para los usuarios que no lo estén y tengan los permisos necesarios.
Tareas precedentes	4.1.
Entradas	Inicio, login y registro implementados.
Salidas	Inicio, login y registro funcionando correctamente.
Duración	4 horas

TAREA 5.2: Pruebas del área de usuarios

Tabla 29 - TAREA 5.2

Descripción	Se probará el área de usuarios; es decir, todas las operaciones que se pueden realizar sobre los tres tipos de usuario en el sistema, según los permisos con los que se cuente: administrador, docente y estudiante.
Tareas precedentes	4.2.
Entradas	Área de usuarios implementada
Salidas	Área de usuarios funcionando correctamente.
Duración	4 horas

TAREA 5.3: Pruebas del área de asignaturas

Tabla 30 - TAREA 5.3

Descripción	Se probará el área de asignaturas; es decir, todas las operaciones que se pueden realizar sobre las asignaturas.
Tareas precedentes	4.3.
Entradas	Área de asignaturas implementada
Salidas	Área de asignaturas funcionando correctamente
Duración	4 horas

TAREA 5.4: Pruebas del área de centros

Tabla 31 - TAREA 5.4

Descripción	Se probará el área de centros; es decir, todas las operaciones que se pueden realizar sobre los centros.
Tareas precedentes	4.4.
Entradas	Área de centros implementada
Salidas	Área de centros funcionando correctamente
Duración	4 horas

TAREA 5.5: Pruebas del área de departamentos

Tabla 32 - TAREA 5.5

Descripción	Se probará el área de departamentos; es decir, todas las operaciones que se pueden realizar sobre los departamentos.
Tareas precedentes	4.5.
Entradas	Área de departamentos implementada
Salidas	Área de departamentos funcionando correctamente
Duración	4 horas

TAREA 5.6: Pruebas del área de titulaciones

Tabla 33 - TAREA 5.6

Descripción	Se probará el área de titulaciones; es decir, todas las operaciones que se pueden realizar sobre las titulaciones.
Tareas precedentes	4.6.
Entradas	Área de titulaciones implementada
Salidas	Área de titulaciones funcionando correctamente
Duración	4 horas

TAREA 5.7: Pruebas del área de grupos

Tabla 34 - TAREA 5.7

Descripción	Se probará el área de grupos; es decir, todas las operaciones que se pueden realizar sobre los grupos.
Tareas precedentes	4.7.
Entradas	Área de grupos implementada
Salidas	Área de grupos funcionando correctamente
Duración	4 horas

TAREA 5.8: Pruebas del área de matrículas

Tabla 35 - TAREA 5.8

Descripción	Se probará el área de matrículas; es decir, todas las operaciones que se pueden realizar sobre las matrículas.
Tareas precedentes	4.8.
Entradas	Área de matrículas implementada
Salidas	Área de matrículas funcionando correctamente
Duración	4 horas

TAREA 5.9: Evaluación de la usabilidad

Tabla 36 - TAREA 5.9

Descripción	Se realizará el estudio de usabilidad para testear la usabilidad de la aplicación desarrollada (ver 7. Evaluación de la usabilidad)
Tareas precedentes	5.8.
Entradas	Sistema implementado
Salidas	Sistema implementado y testeado
Duración	8 horas

2.4.7. PAQUETE 6: Documentación

En el apartado de documentación se elaborarán todos los documentos asociados al Trabajo de Fin de Grado: la propia memoria del proyecto; que en todo caso lo acompaña y sirve de referencia para cualquier persona que así lo desee, y la presentación en formato Microsoft PowerPoint que será expuesta durante la defensa.

TAREA 6.1: Redacción del DOP

Tabla 37 - TAREA 6.1

Descripción	Redacción del Documento de Objetivos del Proyecto. Este documento se corresponde con los tres primeros apartados de la documentación del proyecto (Intoducción, Planteamiento inicial y Antecedentes) y es un pequeño estudio acerca del proyecto, elaborado en cualquier caso previamente al desarrollo del mismo. Finalmente, este documento será parte de la memoria.
Tareas precedentes	2.2.
Salidas	Documento de Objetivos del Proyecto redactado.
Duración	60 horas

TAREA 6.2: Redacción de la memoria

Tabla 38 - TAREA 6.2

Descripción	Redacción de la memoria del Trabajo de Fin de Grado. La memoria es un documento formal que sirve de referencia al proyecto, y en el que se expone toda la información relativa al mismo, así como a su desarrollo.
Tareas precedentes	2.7.
Entradas	Documento de Objetivos del Proyecto redactado.
Salidas	Memoria redactada.
Duración	200 horas

TAREA 6.3: Preparación de la presentación de la defensa

Tabla 39 - TAREA 6.3

Descripción	Consiste en la elaboración de la presentación y el vídeo que se expondrán durante la defensa.
Tareas precedentes	4.9 y 5.2.
Entradas	Todo el proyecto (excepto la preparación de la presentación de la defensa) finalizado.
Salidas	Presentación de la defensa elaborada.
Duración	30 horas

2.5. Planificación temporal

En la Figura 4 - Planificación temporal en diagrama Gantt se muestra la planificación temporal; es decir, un diagrama Gantt con la representación gráfica en el tiempo de la realización del proyecto. Se muestran todos los paquetes y cada una de las tareas que los componen. Como se puede observar, el orden en el que se llevan a cabo los paquetes y sus tareas no coincide necesariamente con los números que los identifiquen.

De esta manera, en lugar de elaborar al final la documentación al término del desarrollo, se escribirá durante el transcurso del mismo; de manera que también haga las veces de diario y ésta y el desarrollo se enriquezcan mutuamente.

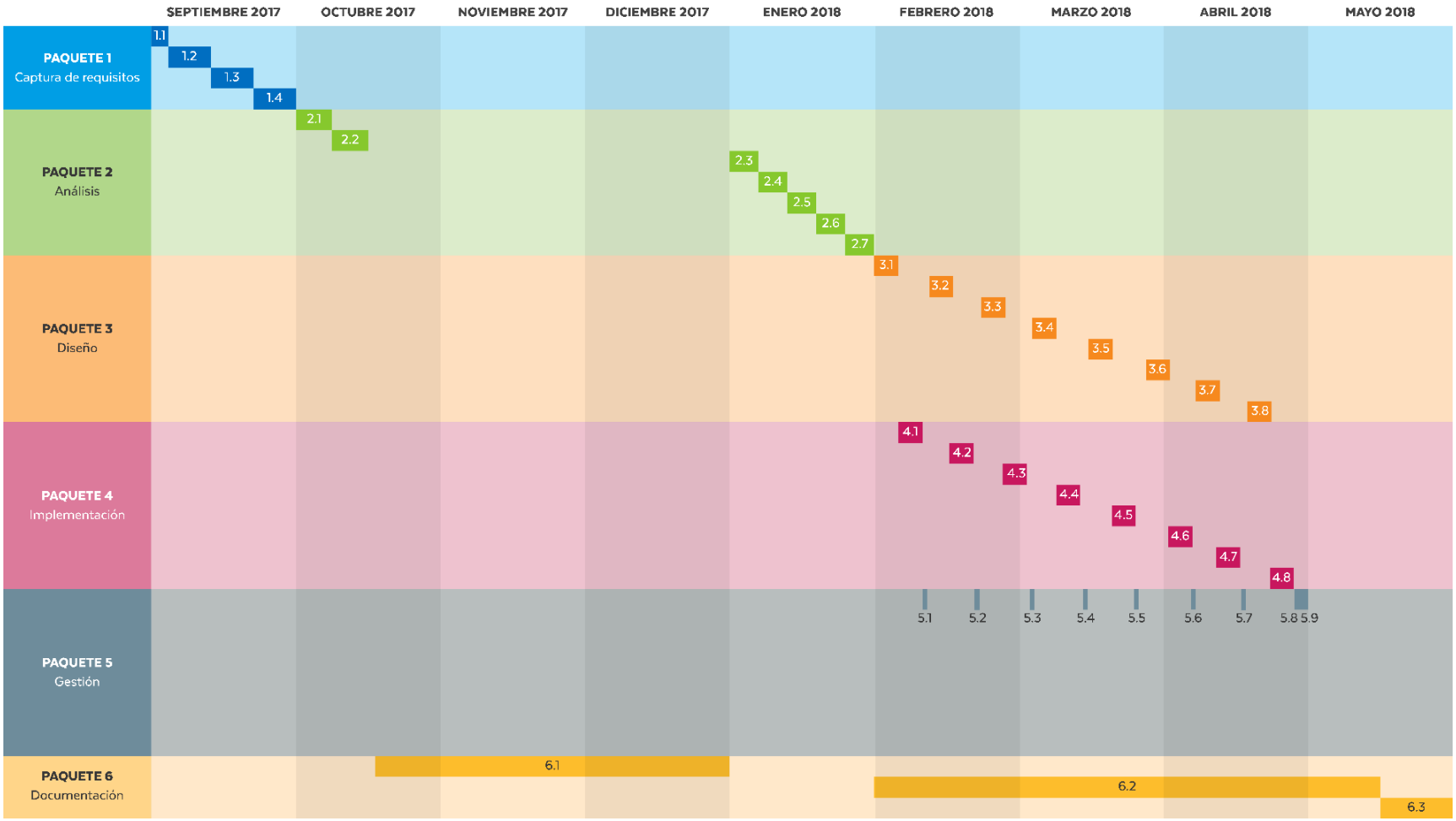


Figura 4 - Planificación temporal en diagrama Gantt

En la Tabla 40 - Horas dedicadas a cada una de las tareas se puede ver la suma total de horas, que asciende a 450.

Tabla 40 - Horas dedicadas a cada una de las tareas

TAREA	Número de horas
1.1: Reunión con el director	2
1.2: Búsqueda de información sobre las tecnologías del proyecto	5
1.3: Lectura de la documentación del TFG de AdESMuS	5
1.4: Análisis y organización del proyecto	5
2.1: Planificación de la metodología de trabajo	5
2.2: Estudio de las herramientas y técnicas utilizadas	3
2.3: Aprendizaje de la tecnología MEAN stack	13
2.4: Aprendizaje de la tecnología Angular (con fines de desarrollo)	13
2.5: Aprendizaje de la tecnología Angular de cara a la herramienta AdESMuS	9
2.6: Instalación del entorno de AdESMuS en el equipo de desarrollo	5
2.7: Instalación del software necesario para el desarrollo	5
3.1: Diseño del inicio, login y registro	20
3.2: Diseño del área de usuarios	20
3.3: Diseño del área de asignaturas	20
3.4: Diseño del área de centros	20
3.5: Diseño del área de departamentos	20
3.6: Diseño del área de titulaciones	20
3.7: Diseño del área de grupos	20
3.8: Diseño del área de matrículas	20
4.1: Implementación del inicio, login y registro	25
4.2: Implementación del área de usuarios	25
4.3: Implementación del área de asignaturas	25
4.4: Implementación del área de centros	25
4.5: Implementación del área de departamentos	25
4.6: Implementación del área de titulaciones	25
4.7: Implementación del área de grupos	25
4.8: Implementación del área de matrículas	25
5.1: Pruebas del inicio, login y registro	4
5.2: Pruebas del área de usuarios	4
5.3: Pruebas del área de asignaturas	4
5.4: Pruebas del área de centros	4
5.5: Pruebas del área de departamentos	4
5.6: Pruebas del área de titulaciones	4
5.7: Pruebas del área de grupos	4
5.8: Pruebas del área de matrículas	4
5.9: Evaluación de la usabilidad	8
6.1: Redacción del DOP	60
6.2: Redacción de la memoria	200
6.3: Preparación de la presentación de la defensa	30
TOTAL	**450**

Se trabajará todos los días de lunes a viernes, aproximadamente 2 horas y media; durante nueve meses: entre septiembre de 2017 y mayo de 2018 (ambos inclusive). Este ritmo de trabajo resulta en 50 horas mensuales, que en nueve meses hacen 450, la duración del proyecto.

2.6. Arquitectura

En el apartado de arquitectura se explicará la estructura y el funcionamiento del sistema desarrollado. El front-end desarrollado no se realiza de manera independiente, sino que se construye orientado hacia AdESMuS y para funcionar con éste. Por tanto, no tiene sentido hablar de la arquitectura de este proyecto como tal, sino de la arquitectura del conjunto formado por el front-end desarrollado y el back-end existente.

El sistema tiene una arquitectura de back-end y front-end. El back-end, de manera independiente, es una API REST (ver 3.2. Arquitectura), que es el que opera con la base de datos. El back-end recibe peticiones, realiza en la base de datos las acciones correspondientes y devuelve una respuesta, si procede.

Por su parte, el front-end desarrollado constituye una interfaz situada entre el usuario y el back-end. El usuario hace uso de la interfaz, que se encarga de "traducir" las acciones del usuario en las peticiones pertinentes sobre el back-end. Posteriormente, recoge las respuestas de éste y se las muestra al usuario, de una forma usable y agradable.

En la Figura 5 - Back-end, front-end, servidor y cliente se puede ver un esquema del funcionamiento del conjunto del back-end y el front-end desarrollado.

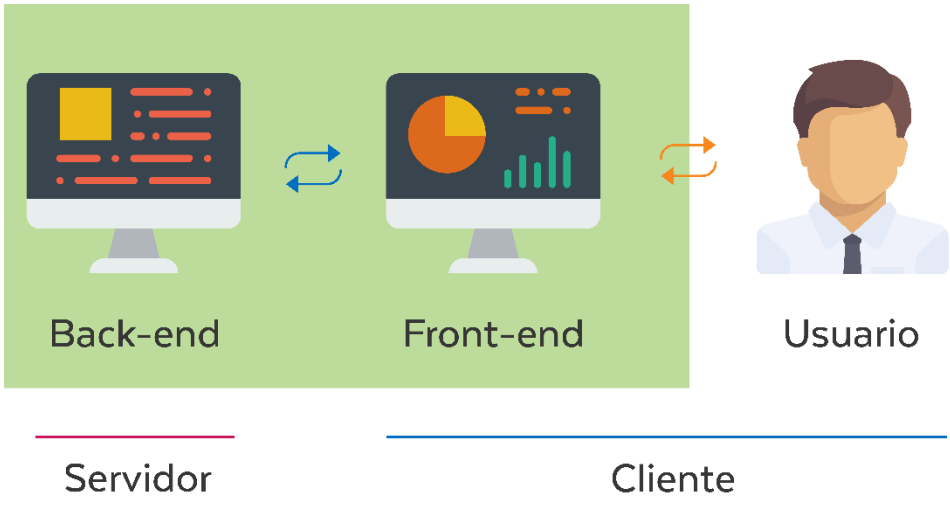


Figura 5 - Back-end, front-end, servidor y cliente

El back-end y el front-end desarrollado, aunque funcionan en conjunto, son dos aplicaciones completamente independientes. De hecho, usan diferentes tecnologías (el back-end está realizado en Node.js y el front-end se ha desarrollado en Angular) y diferente estructuración.

El back-end de AdESMuS funciona en un servidor propio que utiliza Ubuntu Server; mientras que el front-end es un proyecto en Angular independiente.

2.7. Herramientas

En este apartado se realizará una explicación de todas las herramientas utilizadas para el proyecto, y cuál ha sido la función de cada una de ellas.

Angular

Angular es un framework para el desarrollo de la parte front-end o de los sistemas web. Angular se programa en código TypeScript (en el que está basado) y es muy útil en el desarrollo de interfaces front-end, ya que automatiza gran cantidad de procesos y ordena y estructura el código. Es la principal tecnología utilizada en el desarrollo.

Figura 6 - Logotipo de Angular

Oracle VM VirtualBox

Es un software de código abierto que permite administrar máquinas virtuales en el equipo. Será el utilizado para la máquina que actúa como servidor del sistema AdESMuS.

Visual Studio Code

Es un editor de texto de Microsoft compatible con prácticamente cualquier formato de programación como de texto (JAR, PY, JS, XML, HTML, CSS, PHP, TXT, etc.). Se adapta automáticamente al tipo de archivo, mostrándolo de una manera clara y ordenada, mediante colores y estilos; facilitando considerablemente la tarea de programar.

También permite operar en la consola y abrir ficheros con varios archivos, lo que es muy útil para trabajar en Angular. Su uso es gratuito en Windows.

Figura 7 - Logotipo de Visual Studio Code

Microsoft Word

Es un software de procesamiento de texto, que será utilizado a la hora de escribir la documentación.

Google Chrome

Google Chrome es un navegador web con el que se realizarán todas las pruebas del proyecto.

Mozilla Firefox

Es un navegador web software libre que será utilizado para realizar las pruebas del proyecto de manera secundaria; en caso de producirse algún problema con el uso de Google Chrome.

Ubuntu Server

Ubuntu Server es una edición de Ubuntu especial para funcionar como servidor, que carece de interfaz gráfica y su uso habitual es de manera externa. Será el sistema operativo instalado en el servidor; ya que es el que ya utiliza AdESMuS. Al igual que todos los productos Linux, es completamente gratuito y de código abierto.

Google Drive

Plataforma de almacenamiento de archivos online de Google. Se usará para almacenar las copias de seguridad, que serán subidas en el momento de su creación.

Postman

Postman es un software que agiliza la tarea de las pruebas con el código; ya que las automatiza, evitando que el programador deba llevarlas a cabo de forma manual.

2.8. Gestión de riesgos

En este apartado se expondrán los posibles riesgos que implica el desarrollo del proyecto; ya sean que afecten a la salud o situaciones que puedan darse durante el desarrollo, que lo ralenticen, compliquen o den lugar a puntos muertos.

2.8.1. Pérdida accidental de los archivos

Un posible riesgo es el borrado o pérdida accidental de los archivos del equipo, debido a la aparición de un virus o simplemente a la desaparición de los archivos por accidente.

Prevención

Utilizar un antivirus en el ordenador.

Realizar copias de seguridad periódicas (semanales) de todos los archivos: tanto del desarrollo como de los recursos utilizados y la documentación. Estas copias, en el momento en que se creen, se guardarán en Google Drive y se irán grabando en CDs, de manera que la información estará almacenada en tres lugares: en el propio equipo, en la nube y en discos externos.

Plan de contingencia

Recuperar el trabajo perdido restaurando la última copia de seguridad subida a Google Drive. En el hipotético caso de que se hubiera borrado, se restauraría la última copia existente en los CDs.

Probabilidad

Probabilidad de que la información se elimine por accidente y no esté en la papelera de reciclaje, o de que desaparezca del equipo sin una razón aparente.

La probabilidad de este riesgo se considera baja.

Impacto

Bajo, únicamente es necesario restaurar la última copia y sólo se perdería el trabajo realizado desde ésta.

2.8.2. Caída de la luz eléctrica

Consistiría en una pérdida de alimentación del equipo durante el trabajo.

Prevención

Utilizar el ordenador con la batería puesta, que se encarga de suministrar energía en caso de ausencia de corriente eléctrica.

Guardar con frecuencia el progreso avanzado.

Plan de contingencia

Repetir todo el trabajo realizado desde la última vez en que se guardó la información, ya que se ha perdido.

Probabilidad

La probabilidad de este riesgo se considera baja.

Impacto

Pérdida del trabajo no guardado; es decir, del trabajo realizado desde la última vez en que se guardó el progreso.

2.8.3. Enfermedad o problema de salud

Consistiría en un problema de salud que interfiriese en el transcurso normal del proyecto.

Prevención

Llevar unos horarios coherentes, que permitan realizar descansos y eviten largas jornadas de trabajo.

Evitar exponerse a riesgos y llevar un estilo de vida sano.

Plan de contingencia

Acudir a una consulta médica para que se realice un diagnóstico. Dependiendo de la gravedad del estado de salud, se continuará el trabajo a un ritmo menor o se realizará una parada; hasta que haya una recuperación y permita continuar con el ritmo habitual.

Probabilidad

La probabilidad de este riesgo se considera media.

Impacto

Alteración del ritmo de trabajo, ralentizándolo o incluso pausándolo durante un intervalo de tiempo variable.

2.8.4. Problema con las versiones

Lanzamiento de una nueva versión de Angular que provocara que la utilizada hasta el momento quedase anticuada («deprecated») o actualización del software automática/accidental que provocase un cambio en la configuración del proyecto.

Prevención

Comprobar periódicamente el estado de las versiones: si se lanzan actualizaciones, qué cambios incorporan, etc.

Asegurarse de que las actualizaciones automáticas están desactivadas, de manera que el software se actualice únicamente cuando se desee, y habiendo realizado previamente una copia de seguridad.

Plan de contingencia

En caso de un fallo en la configuración provocado por una actualización de software: restaurar la última copia de seguridad guardada.

En caso de que la versión utilizada quede anticuada, realizar una actualización asegurándose minuciosamente de qué cambios provocará en el código; y siempre habiendo realizado copias de seguridad periódicas que permitan ser restauradas en caso de cometer un error.

Probabilidad

La probabilidad de este riesgo se considera media.

Impacto

Alteración del ritmo de trabajo, haciendo necesario invertir tiempo en solucionar el problema.

2.8.5. Desconocimiento de una tecnología o lenguaje utilizado

Consiste en que se dé el caso de que sea necesario hacer uso de una tecnología o lenguaje desconocidos.

Prevención

Antes de empezar la parte de desarrollo y durante ella, informarse acerca de las tecnologías utilizadas, mediante la visualización de vídeo-tutoriales sobre ellas, la lectura de documentación o bibliografía relacionada o la propia documentación del propio Trabajo de Fin de Grado de AdESMuS.

Informarse previamente de manera adecuada sobre qué lenguajes y tecnologías son necesarios para el desarrollo del TFG, de manera que esta situación no se de de manera repentina.

Plan de contingencia

Se realizaría una pausa en el desarrollo, durante la cual se adquirirían los conocimientos necesarios para poder continuar con el trabajo. Se continuaría en el momento en el que se decidiera que ya se domina la tecnología lo suficiente como para poder ser utilizada.

Probabilidad

La probabilidad de este riesgo se considera media-alta.

Impacto

Alteración del ritmo de trabajo, haciendo necesario invertir tiempo en conocer el funcionamiento de dicha tecnología.

2.8.6. Fallo en el funcionamiento de la herramienta AdESMuS

El posible fallo que pueda sufrir la herramienta AdESMuS. En este caso, se trataría de un error en un sistema que no ha sido desarrollado en este proyecto; ya que AdESMuS viene de un TFG previo.

Prevención

Asegurarse de que el sistema está perfectamente configurado antes de comenzar la tarea del desarrollo; y verificarlo periódicamente durante el transcurso del mismo.

Evitar realizar cambios profundos en el equipo, y especialmente en el software utilizado para el desarrollo; ya que posteriormente puede ocasionar problemas o errores.

Plan de contingencia

Si el problema es algo sencillo, se intentará solucionar reconfigurando lo que fue modificado. En caso de no ser eficaz esta solución, se procedería a restaurar una copia de seguridad previa en la que el sistema funcionase correctamente.

Por último, y si lo especificado anteriormente no surtiera ningún efecto, sería necesario ponerse en contacto con el tutor del proyecto; y, si fuera posible, con el creador de la herramienta.

Probabilidad

La probabilidad de este riesgo se considera alta.

Impacto

Alteración del ritmo de trabajo, ocasionando frustración en el desarrollador y, en el peor de los casos, impidiendo que sea posible avanzar hasta la solución del problema.

2.9. Evaluación económica

El coste total del desarrollo de la herramienta se calculará en función de las horas totales invertidas, proponiendo un coste por hora basado en datos reales; ya que el desarrollo de un Trabajo de Fin de Grado en este campo se asemeja a la forma de trabajar en los puestos relacionados con el desarrollo.

Asimismo, también se tendrán en cuenta otros gastos que implicará el desarrollo aparte de las horas de trabajo. En este caso; y dado que es materialmente imposible calcular con exactitud este tipo de costes, se realizará una estimación matemática.

2.9.1. Mano de obra

El cálculo del coste de una hora de trabajo de desarrollo, se ha realizado teniendo en cuenta que el proyecto ha sido realizado en España en el año 2017. Para ello, se ha empleado el salario mínimo interprofesional más reciente, establecido por convenio el 18 de enero del mismo año.

En empresas de ingeniería y oficinas de estudios técnicos, el salario mínimo mensual establecido para licenciados, titulados de 2º y 3er ciclo universitario o analistas es de 1.687,02 € **[1]**. Dado que en un empleo a jornada completa se trabajan 200 horas mensuales, el precio de la hora es de 8,43 €. Si lo multiplicamos por las 450 horas que dura el proyecto, obtenemos 3793,50 € de mano de obra.

2.9.2. Hardware

El proyecto ha sido desarrollado en una máquina Lenovo G500, cuyo precio fue de 595 € **[2]** (ver Tabla 41 - Precio y vida útil del Lenovo G500).

Tabla 41 - Precio y vida útil del Lenovo G500

	Vida útil estimada	Precio
Lenovo G500	60 meses	595 €

$$\text{Amortización mensual} = \frac{\text{Precio}}{\text{Vida útil estimada}}$$

$$\text{Amortización durante el proyecto} = \text{Amortización mensual} \times \text{Duración del proyecto}$$

Como la duración del proyecto es de 9 meses:

Tabla 42 - Amortización del Lenovo G500

	Amortización mensual	Amortización durante el proyecto
Lenovo G500	10 €/mes	90 €

Por tanto, como se observa en la Tabla 42 - Amortización del Lenovo G500, el valor del equipo durante el proyecto es de 90 €.

2.9.3. Software

El equipo utiliza el sistema operativo Windows 10. La licencia del mismo fue concedida gratuitamente por la Escuela de Ingeniería de Bilbao a través de Microsoft Imagine, aunque en el caso de un puesto de trabajo, la licencia se habría comprado; y su precio habría sido de 145 € **[3]**.

Figura 8 - Logotipo de Microsoft Imagine

El procesador de textos utilizado, Microsoft Word, ya estaba incluido en el sistema operativo de serie. Por tanto, no tiene sentido incluirlo como un coste en el apartado de software.

El resto de las herramientas son de código abierto o de uso gratuito, por lo que no han ocasionado ningún coste.

2.9.4. Gastos indirectos

Los gastos indirectos son los gastos que no se pueden medir porque son compartidos con otras actividades ajenas al proyecto; y por tanto no se puede hablar de gastos ocasionados, al menos de manera directa.

Luz eléctrica

Los gastos de luz eléctrica serían los producidos por el equipo durante la realización del trabajo. Dado que el único equipo es un ordenador Lenovo G500, se realizará una estimación de los gastos de luz ocasionados a partir de la potencia del mismo.

Dado que la duración del trabajo es de 9 meses, con un ritmo de 4 horas diarias, en total cada mes se habrán trabajado unas 90 horas, que suponen 810 horas totales. En la Tabla 43 - Potencia máxima y energía total consumida del Lenovo G500 se puede ver la potencia a la que equivale.

Tabla 43 - Potencia máxima y energía total consumida del Lenovo G500

	Potencia del adaptador	Energía total consumida
Lenovo G500	65 W **[4]**	52,65 kWh

Se supone que el proyecto está realizado en España; donde el precio medio de la energía es de 0,182 €/kWh **[5]**.

Por tanto, el producto de ambos valores resulta en que el coste energético total asciende a 9,58 €.

Conexión a Internet

Para el desarrollo del proyecto ha sido necesaria una conexión a Internet. La conexión utilizada es de fibra óptica y tiene un coste de 40 €/mes.

Para aplicarlo a una duración de 9 meses, realizando el producto, se tiene que el coste total es de 360 €.

El total de gastos indirectos tiene un valor de 367,37 €.

2.9.5. Gastos totales

Para calcular los gastos totales; (gastos aplicados en función de las horas estimadas en cada caso) basta con sumar los gastos en cada una de las áreas.

Tabla 44 - Gastos totales del proyecto

Área	Coste
Mano de obra	3.793,50 €
Hardware	90 €
Software	145 €
Gastos indirectos	376,95 €
TOTAL	4.405,45 €

Como se visualiza en la Tabla 44 - Gastos totales del proyecto, se tiene que los gastos de realización del proyecto ascienden a un total de 4.405,45 €.

Dado que el proyecto está contemplado como un trabajo de la Universidad, el objetivo del mismo no es obtener lucro, sino crear un sistema para la comunidad educativa, así como aprender durante el desarrollo del mismo.

3. Antecedentes: AdESMuS

La propuesta de este Trabajo de Fin de Grado proviene de una herramienta, AdESMuS, que fue desarrollada en un TFG anterior **[6]**.

AdESMuS (Adaptable Evaluation System Using Multiple Sources) es una herramienta, que hasta el momento no cuenta con interfaz gráfica (únicamente está desarrollado el back-end) y su función es realizar la evaluación de cualquier asignatura.

AdESMuS gestiona todo tipo de datos universitarios (centros, titulaciones, asignaturas, estudiantes) que almacena en una base de datos. Sin embargo, no cuenta con una interfaz gráfica, problema del cual ha nacido este TFG: la cración de una interfaz o front-end para AdESMuS.

3.1. Funcionamiento

En el sistema AdESMuS, los usuarios deben iniciar sesión (o registrarse, si no disponen de una cuenta) para operar. Los usuarios pueden ser de cuatro tipos: Supremo, Administrador, Docente y Estudiante; y estos roles condicionarán los distintos permisos que el usuario tendrá dentro de la plataforma.

El sistema gira en torno a siete conceptos: usuario, asignatura, centro, departamento, titulación, grupo y matrícula; que están relacionados entre sí y constituyen el esqueleto de la base de datos de la herramienta. Existen gran diversas relaciones entre los elementos: las asignaturas pertenecen a titulaciones, los grupos pertenecen a centros, se pueden asociar titulaciones y centros, etc. Los usuarios de AdESMuS se consideran un elemento más y son de tres tipos, según los permisos de los que dispongan: administrador, docente y estudiante.

Los usuarios, en función de los permisos que les otorgue su rol, pueden operar sobre la información que almacena la base de datos, realizando las funciones habituales: consultar, añadir, eliminar, modificar, etc. La base de datos de AdESMuS está implementada en MySQL, y su modelo de dominio se puede ver en la Figura 20 - Modelo de dominio de AdESMuS.

Una vez el usuario ha iniciado sesión, la herramienta está dividida en siete áreas: relativas a cada uno de los conceptos anteriormente explicados: área de usuarios, área de asignaturas, área de centros, etc. donde el usuario podrá llevar a cabo las operaciones sobre la base de datos relativas al concepto en cuestión.

3.2. Arquitectura

El back-end cuenta con una serie de peticiones disponibles (añadir, obtener, editar, etc.) que suelen ir acompañadas de los datos necesarios. Al invocar estas peticiones, el back-end ejecuta en la base de datos la orden correspondiente, y devuelve información como los datos solicitados o si la operación se ha realizado correctamente. Esta información se devuelve en formato JSON (ver 2.2.2. JSON).

3.2.1. API REST

La herramienta AdESMuS ya desarrollada se caracteriza por tener una arquitectura basada en API REST (REpresentational State Transfer, ver 2.2.5. API REST). Esta arquitectura es conocida porque los sistemas que la utilizan carecen de estado (stateless) (y, por tanto, no se retiene información como qué usuario es el que ha iniciado sesión, si lo ha hecho de manera correcta ni la contraseña introducida). Al contrario, en cada petición al servidor, se adjuntarán las credenciales introducidas cifradas (token). El servidor, una vez lo recibe, en primer lugar verificará si es correcto; y en tal caso, se actuará como si el usuario hubiera iniciado la sesión y se atenderá la petición.

Estos datos viajan en las URLs que el servidor recibe en cada momento; ya que en ellas se especifica tanto la información solicitada por el usuario como todo lo que el servidor requiere para poder devolverla (usuario, contraseña, etc.). Es decir, en lugar de alojar volatilmente la sesión que se ha iniciado, en cada una de las peticiones es necesario incluir las credenciales en la URL, mediante el protocolo HTTP.

Para ello se hace uso de los tokens, que son una forma de inicio de sesión ligera, segura y escalable (permite ser usada en distintos dispositivos). Al iniciar sesión, el sistema genera un token para la sesión, que funciona como una llave, y la devuelve en la respuesta. Cada vez que se desee realizar una petición, se debe introducir el token en la URL. El servidor comprueba la validez del token, y si es correcto, atiende la petición. El funcionamiento se puede observar en la Figura 9 - API REST.

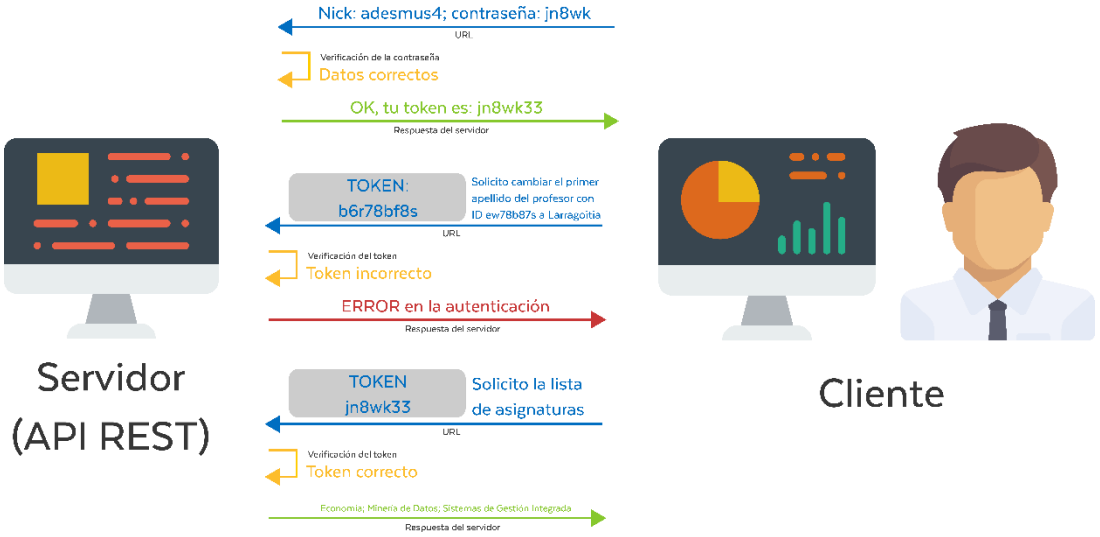


Figura 9 - API REST

3.3. Ejemplo de una petición

Las peticiones son de cuatro tipos: GET (obtener), POST (registrar), PUT (modificar) y DELETE (borrar). En la Tabla 45 - Ejemplo de una petición al back-end de AdESMuS se puede ver el ejemplo de la sintaxis de una petición para modificar el nombre de una titulación.

Tabla 45 - Ejemplo de una petición al back-end de AdESMuS

	Descripción	Ejemplo real

URL	**PUT**
	`https://ip/api/degree/id`
	En *ip* se introduce la dirección IP en la que funciona el servidor
	En *id* se introduce el ID de la titulación en cuestión
		PUT
		`https://`
		`192.168.56.101:3200/api/degree`
		`/4hb7n7748ndfndne8778448m`

Headers	`x-access-token: token` Se añade el token del usuario para poder realizar la operación (ver 3.2.1. API REST)	`x-access-token: n7dsdn7834n7dms87`

Body enviado	{
	"nombreTitulacion" : *nuevo nombre*
	}
		{
		"nombreTitulacion" :
		“Grado en Filología Griega”
		}

Respuesta	**200 OK**	**200 OK**

JSON recibido	{
	"codTitulacion" : *id de la titulación*,
	"nombreTitulacion" : *nuevo nombre*
	}
		{
		"codTitulacion" :
		4hb7n7748ndfndne8778448m,
		"nombreTitulacion" : “Grado en Filología Griega”
		}

3.4. Diseño del front-end

Previamente al comienzo de la programación de este proyecto, se analizó qué herramientas se utilizarían para el desarrollo del front-end, qué permitía cada una de ellas y cuáles de estos recursos serían incluidos en la interfaz.

La tecnología elegida para el desarrollo fue Angular, porque es un framework que suele utilizarse conjuntamente con Node.js, y dado que el back-end está desarrollado en ese lenguaje, es muy apropiado desarrollar el front-end en Angular. Asimismo, es una tecnología en auge, que hoy en día utilizan la mayor parte de las aplicaciones web; en favor del JavaScript, HTML y CSS puros; ya que permite todas las funciones que éstos ofrecen, estructura y organiza el código de una forma muy práctica en componentes y además aporta algunas ventajas.

Figura 10 - Logotipo de Angular

Uno de los rasgos más característicos de la programación en Angular es el binding: la comunicación entre la vista y el controlador es en directo. Si un determinado valor cambia en el código, automáticamente se actualizará también su representación en la vista; y de la misma manera, si el valor de un elemento de entrada cambia en la vista (como un campo) automáticamente cambiará el valor del elemento de la variable del código que tenga asignada.

La característica del binding es muy apropiada para este proyecto, debido a que las respuestas a las peticiones se reciben de forma asíncrona. Cuando se realiza una petición al back-end, el tiempo que se tarde en recibirla depende de diversos factores: la velocidad con la que pueda trabajar el servidor en ese momento, si en ese mismo momento el servidor está atendiendo más peticiones, e incluso la velocidad de la conexión si el servidor se encuentra remotamente y conectado a través de la red.

Por ejemplo, si se realiza una petición como “obtener todas las asignaturas” y el número total de asignaturas es de 83; las asignaturas no se recibirán todas a la vez, sino que irán llegando progresivamente una por una, dependiendo la velocidad de los factores anteriormente explicados.

Al realizar una petición como esta, el front-end tiene una variable “asignaturas”, que se trata de un array en el que almacenará las asignaturas recibidas. La vista contendrá una tabla que las muestre; y la variable asignaturas y esta tabla están conectadas mediante binding. Se realiza la petición y, cada vez que una asignatura se recibe se añade al array. Gracias al binding, inmediatamente al añadir la asignatura al array ya se muestra en la tabla en la vista. Esto aporta dos ventajas:

- Evita tener que recargar la página cada vez que un elemento se recibe, ya que de forma automática ya aparece en la tabla al recibirse.
- Permite que no sea necesario esperar a recibir todos los elementos para visualizar algunos en la interfaz. Sin el binding, no sería factible recargar toda la página por cada elemento recibido, por lo que esto se haría al recibir el último elemento, lo que ocasionaría que el usuario sólo visualizase los elementos al haberse recibido todos. De este modo, los elementos aparecen progresivamente en la interfaz según se reciben, lo que evita que el usuario se frustre pensando que ha cometido algún error **[7]**.

4. Captura de requisitos

En este capítulo se expondrá la captura de requisitos. Se trata de uno de los pasos más importantes en el desarrollo de una aplicación, ya que es un análisis de todas las funciones que se desean implementar en la misma.

Todos los requisitos que se van a implementar parten del back-end existente. Es decir, el desarrollo del front-end no ha añadido nuevas funcionalidades a la aplicación. Sin embargo, el front-end automáticamente hace uso de las funciones del back-end a una velocidad muy superior a la que podría realizarlas un ser humano directamente sobre un back-end, lo que permite que algunas tareas laboriosas utilizando el back-end se hayan logrado automatizar, pero haciendo uso en el fondo de las mismas funciones.

Los permisos de cada rol también se han mantenido. El usuario puede iniciar sesión en el momento de abrir la aplicación, a partir de ese momento, se le tratará como administrador, docente o estudiante (según corresponda) y se le ofrecerán las funcionalidades disponibles conforme al rol. Dado que el usuario potencial del front-end desarrollado es en todo caso el usuario cotidiano (con distintos niveles de permiso); algunas funciones del back-end no pueden ser realizadas a través del front-end, como las relativas al rol supremo (que no puede operar a través del front-end) (ver 6.6. Limitaciones del front-end).

4.1. Casos de uso

En la Figura 11 - Jerarquía de actores se puede ver la jerarquía de usuarios en el front-end. El usuario Empleado no existe en el sistema, simplemente es un usuario auxiliar que aparece en los casos de uso para aglutinar las funciones del administrador y el docente.

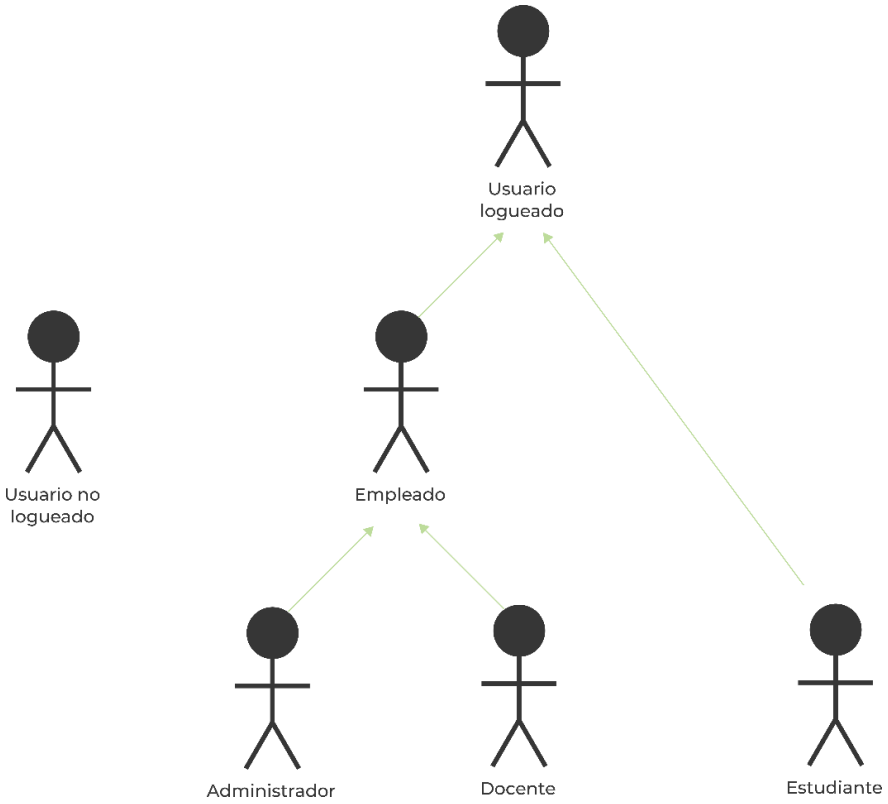


Figura 11 - Jerarquía de actores

4.1.1. Casos de uso del usuario no logueado

El usuario no logueado únicamente puede iniciar sesión, para convertirse en un usuario logueado. En el front-end no existen otras funciones que puedan ser llevadas a cabo sin iniciar sesión. El caso de uso se puede ver en la Figura 12 - Casos de uso del usuario no logueado).

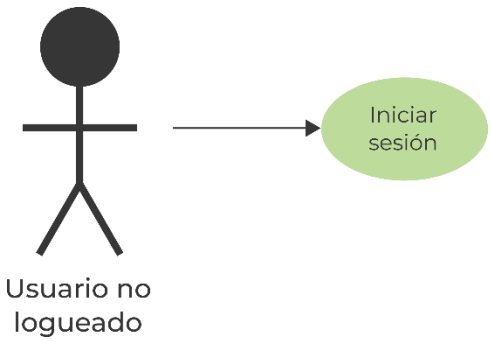


Figura 12 - Casos de uso del usuario no logueado

4.1.2. Casos de uso del usuario logueado

El usuario logueado puede consultar toda la información relacionada con centros, titulaciones, departamentos, asignaturas, grupos y estudiantes; así como cerrar sesión. (Figura 13 - Casos de uso del usuario logueado).

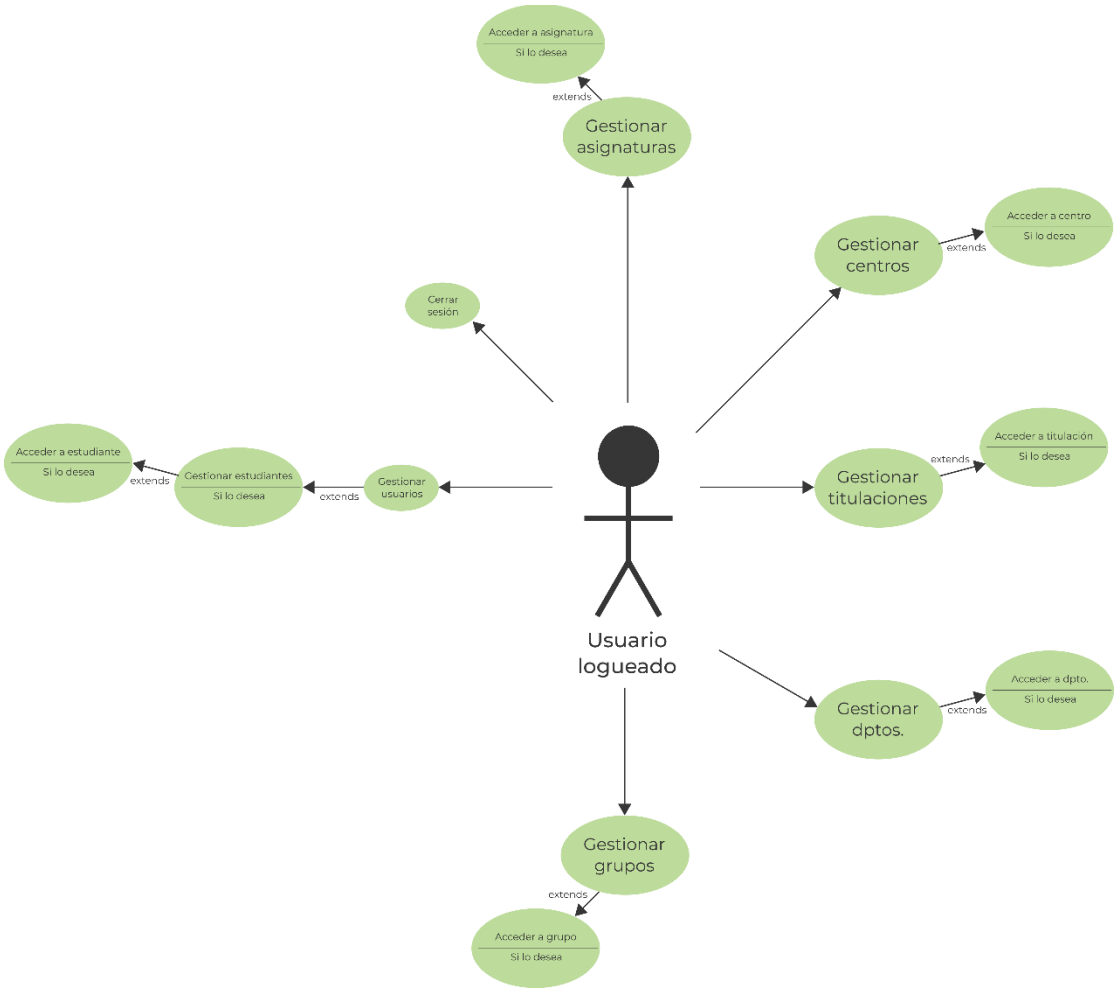


Figura 13 - Casos de uso del usuario logueado

4.1.3. Casos de uso del estudiante

El estudiante es el usuario logueado con menos permisos. Además de las funciones del usuario logueado, puede editar su perfil. Se pueden ver en la Figura 14 - Casos de uso del estudiante.

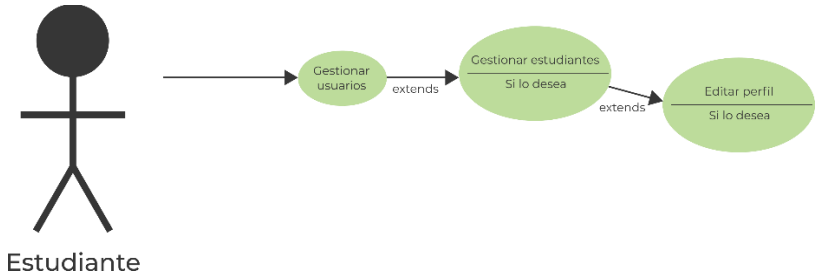


Figura 14 - Casos de uso del estudiante

4.1.4. Casos de uso del empleado

El empleado es un usuario surgido por las características de los casos de uso, que incluye a los administradores y a los docentes (el personal de una universidad). Como tal, puede acceder a la información de cualquier docente. Se puede ver en la Figura 15 - Casos de uso del empleado.

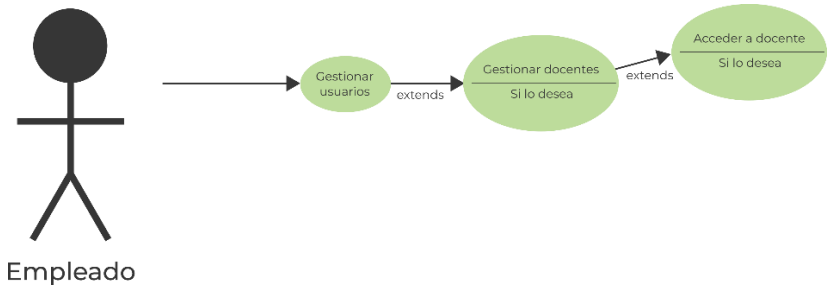


Figura 15 - Casos de uso del empleado

4.1.5. Casos de uso del docente

El docente, además de las funciones del empleado, puede editar su perfil. Se puede ver en la Figura 16 - Casos de uso del docente.

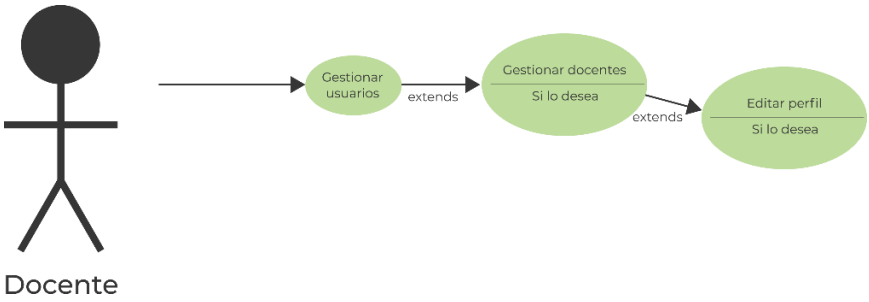


Figura 16 - Casos de uso del docente

4.1.6. Casos de uso del administrador

El administrador es el usuario con más permisos. Puede editar toda la información de la base de datos, excepto la relativa a otros administradores. Se puede ver en la Figura 17 - Casos de uso del administrador.

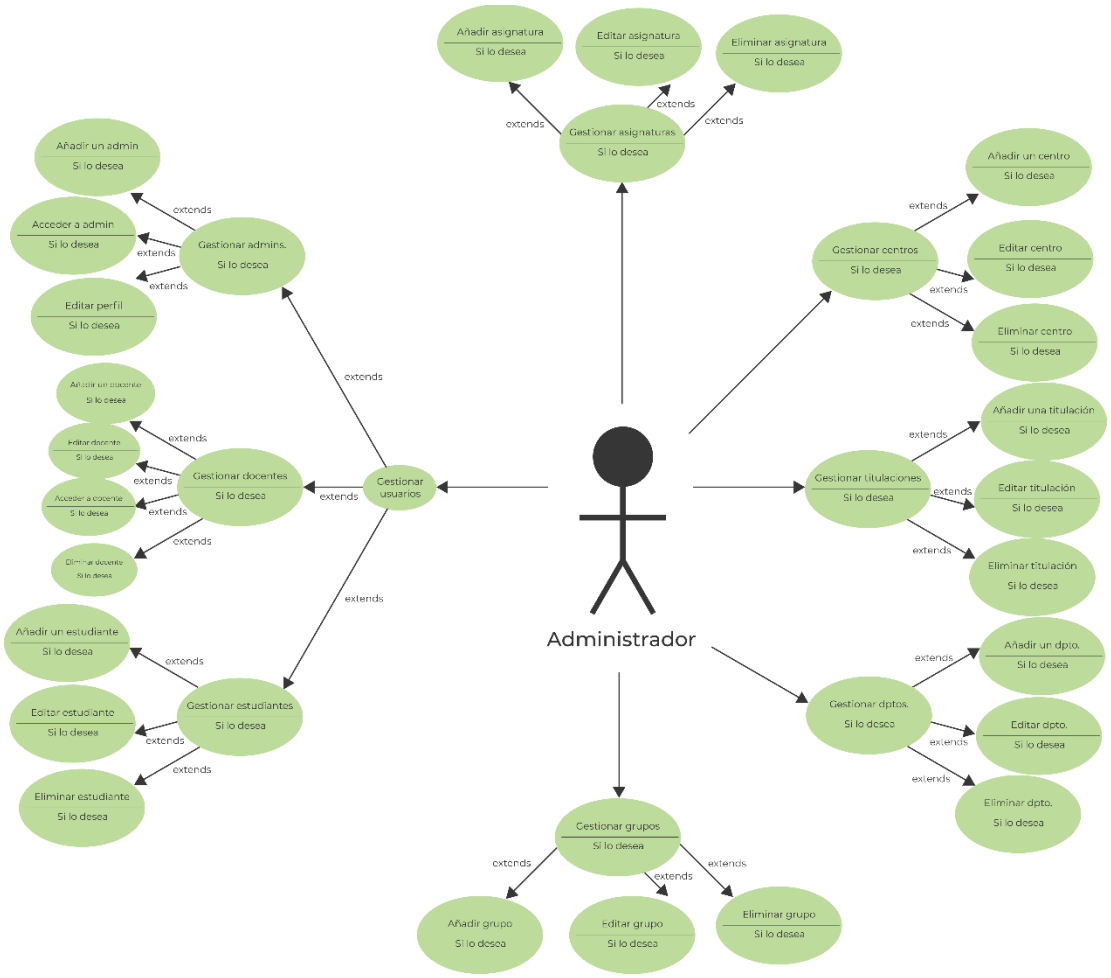


Figura 17 - Casos de uso del administrador

4.1.7. Explicación de los casos de uso

Relativos a la sesión

- Iniciar sesión: realiza un login en el sistema y lleva al menú
- Cerrar sesión: lleva al inicio de sesión y termina la sesión

Relativos al centro

- Acceder a centro: obtener la información de un centro determinado
- Añadir un centro (individualmente o mediante archivo CSV): añade a la base de datos un centro
- Editar centro: modifica el nombre o los elementos asociados de un centro
- Eliminar centro: borra el centro de la base de datos

Relativos a la titulación

- Acceder a titulación: obtener la información de una titulación determinada
- Añadir una titulación (individualmente o mediante archivo CSV): añade a la base de datos una titulación
- Editar titulación: modifica el nombre o los elementos asociados de una titulación
- Eliminar titulación: borra la titulación de la base de datos

Relativos al departamento

- Acceder a departamento: obtener la información de un departamento determinado
- Añadir un departamento (individualmente o mediante archivo CSV): añade a la base de datos un departamento
- Editar departamento: modifica el nombre o los elementos asociados de un dpto.
- Eliminar centro: borra el departamento de la base de datos

Relativos a la asignatura

- Acceder a asignatura: obtener la información de una asignatura determinada
- Añadir una asignatura (individualmente o mediante archivo CSV): añade a la base de datos una asignatura
- Editar asignatura: modifica el nombre o los elementos asociados de una asignatura
- Eliminar asignatura: borra la asignatura de la base de datos

Relativos al grupo

- Acceder a grupo: obtener la información de un grupo determinado
- Añadir un grupo (individualmente o mediante archivo CSV): añade a la base de datos un grupo
- Editar grupo: modifica el nombre o los elementos asociados de un grupo
- Eliminar grupo: borra el grupo de la base de datos

Relativos al usuario

- Acceder a usuario: obtener la información de un usuario determinado
- Añadir un usuario (individualmente o mediante archivo CSV): añade a la base de datos un usuario
- Editar usuario: modifica el nombre o los elementos asociados de un usuario
- Eliminar usuario: borra el usuario de la base de datos

5. Análisis y Diseño

En este apartado se explicará la fase que sucede a la captura de requisitos: el análisis y el diseño: la organización del código y los diagramas correspondientes.

5.1. Introducción: Angular

El front-end de AdESMuS es una aplicación independiente desarrollada en Angular, que implementa una interfaz hacia el usuario, y traduce las acciones de éste en peticiones pertinentes al back-end, para posteriormente traducir los datos devueltos por el back-end a datos mostrados hacia el usuario de una manera cómoda, rápida, agradable e intuitiva. En la programación en Angular se utilizan distintos elementos, siendo los principales los módulos, los componentes y los servicios.

El sistema está formado por un único módulo (App module) que alberga una serie de componentes. Estos componentes consisten en fragmentos de distinto tamaño de código HTML (acompañados por código TypeScript y hojas de estilos CSS) que se lanzan u ocultan. Los componentes no son independientes entre sí, sino que a menudo unos incluyen a otros.

Mediante el routing, se indica qué componente debe ser el que se muestre, lo que implicará la misma acción a todos los componentes que incluya. Por ejemplo, si existe un componente B, que en el interior de sus etiquetas HTML incluye una referencia a C; si se ejecuta un routing a C se mostrará e iniciará sólo el componente C, mientras que si se ejecuta un routing a B se mostrará e iniciará B, y por lo tanto también C en su interior. En la Figura 18 - Routing en Angular se puede ver un esquema del funcionamiento del routing y la inclusión de unos componentes por otros.

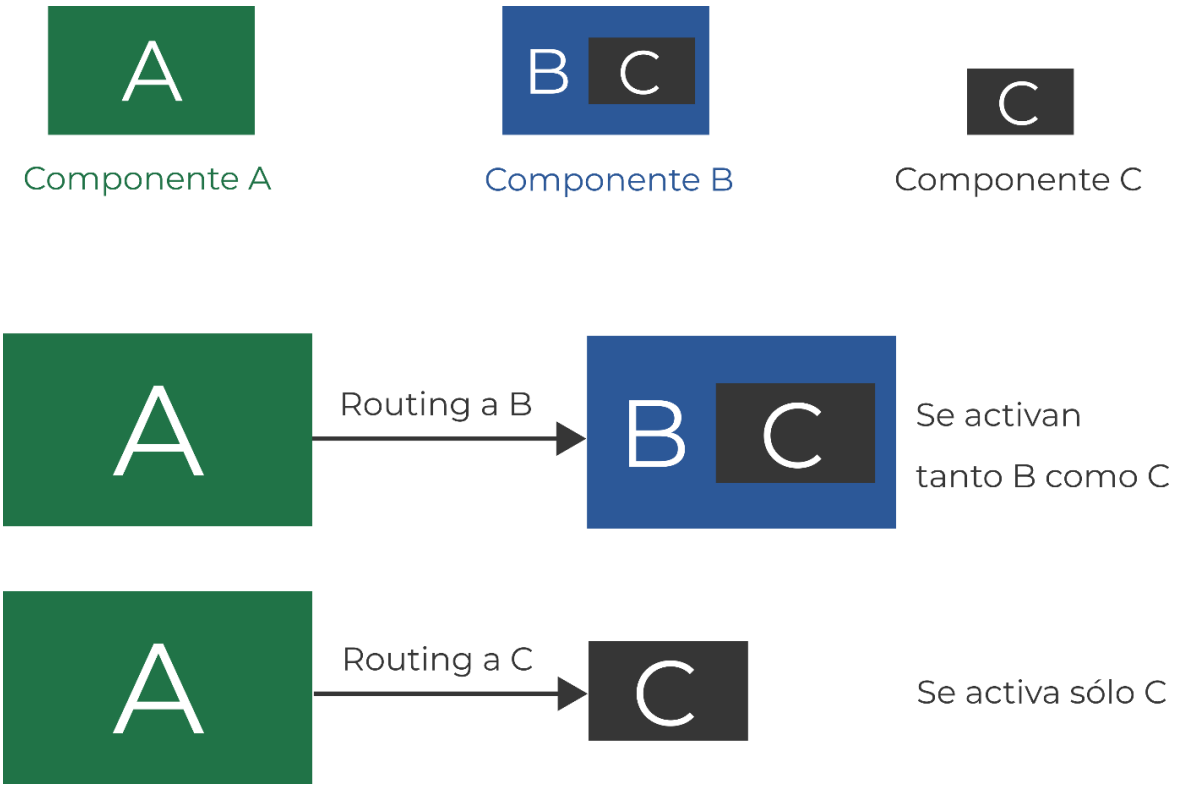


Figura 18 - Routing en Angular

Por último, los componentes se ven complementados por los servicios, que son llamados por éstos y albergan el código encargado de realizar las peticiones (en este caso, al back-end) y que cierran los elementos utilizados en Angular.

5.2. Estructura de la aplicación

La estructura del front-end se puede ver en la Figura 19 - Estructura del front-end. A continuación, se explicará el contenido de la misma.

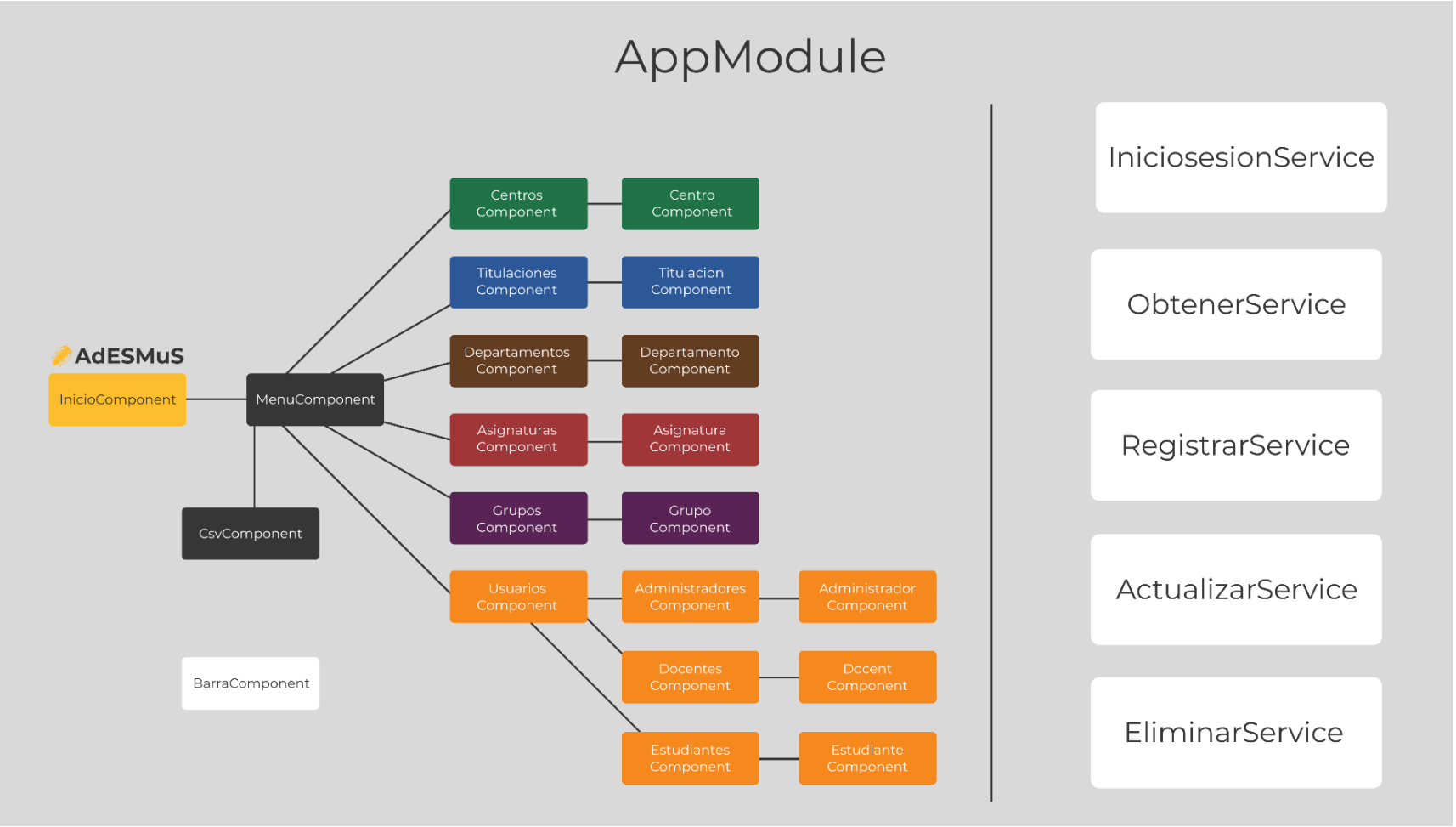


Figura 19 - Estructura del front-end

5.2.1. Componentes

InicioComponent es el componente utilizado para realizar el inicio de sesión. Este componente no incluye a ningún otro y es la ventana de inicio de sesión. En ella, se introducen el nick y la contraseña y el sistema realiza la petición de inicio de sesión. Si el inicio ha sido correcto, se hace un routing a MenuComponent, desde donde se puede acceder a todas las funciones del sistema.

Para cada tipo de entidad en AdESMuS han sido desarrollados dos componentes: uno para operar con todos los elementos de ese tipo y uno para operar con un elemento en concreto, y se distinguen porque el nombre es singular o plural. También existe el componente puente UsuariosComponent que hace de intermediario ante los componentes de tipos de usuario. Ver Tabla 46 - Componentes de tipos de entidad.

Tabla 46 - Componentes de tipos de entidad

Para todos los elementos	Para un elemento
CentrosComponent	CentroComponent
TitulacionesComponent	TitulacionComponent
DepartamentosComponent	DepartamentoComponent
AsignaturasComponent	AsignaturaComponent
GruposComponent	GrupoComponent
AdministradoresComponent	AdministradorComponent
DocentesComponent	DocenteComponent
EstudiantesComponent	EstudianteComponent

A los componentes de tipos de entidad de elementos en concreto se hace routing con dos parámetros, que son necesarios para ejecutar estos componentes: el ID del elemento en cuestión (ID del centro, ID del docente, etc.) y si se van a editar los datos o sólo a consultar.

BarraComponent es un componente incluido en la zona superior de los componentes de tipos de entidad y es una barra con botones para moverse fácilmente entre los distintos tipos.

Por último, el componente CsvComponent es el que implementa toda la importación de datos a partir de archivos CSV y la ventana correspondiente.

5.2.2. Servicios

AdESMuS consta de cinco servicios: IniciosesionService, ObtenerService, RegistrarService, ActualizarService y EliminarService. Cada uno incluye las llamadas al back-end relativas a las funciones que su nombre indica: IniciosesionService realiza un inicio de sesión a partir de un nick y una contraseña, ObtenerService incluye llamadas como obtener un centro, obtener todos los centros u obtener todos los grupos asociados a una asignatura; RegistrarService incluye llamadas como registrar un grupo o asociar un centro y un departamento; ActualizarService incluye llamadas como modificar una asignatura (su nombre) y EliminarService incluye llamadas como eliminar una titulación o desasociar un estudiante con un grupo y una asignatura.

5.3. Diseño de los prototipos

El diseño de cada prototipo se realizará mediante un procedimiento expuesto a continuación:

- En primer lugar, se ejecutará la herramienta AdESMuS (el back-end ya existente) para estudiar su funcionamiento en profundidad y tomar notas acerca de posibles soluciones (recursos que puedan ser aplicados al desarrollar el front-end para crear una interfaz); siempre desde el punto de vista del programador.
- Una vez analizado el back-end y propuestos distintos recursos o maneras de desarrollar la interfaz, se realizará un prototipo completo del front-end a papel, teniendo en cuenta las distintas posibilidades que ofrece Angular y cómo se van a aplicar.
- Ya creado el prototipo, será necesaria una reunión con el tutor para mostrárselo. Debido a que no es factible una reunión para cada una de las áreas del diseño, se enviará la propuesta de diseño por correo electrónico. Posteriormente, serán modificadas las partes que así lo precisen, hasta conseguir el visto bueno por parte del director.
- Una vez aprobado el diseño de la interfaz por parte del director, se elaborará el diseño gráfico definitivo del front-end: el prototipo a ordenador, en el que se especificarán de manera clara todas sus partes y qué recursos de Angular se utilizarán para implementarlas.

5.4. Modelo de dominio

En cuanto al modelo de dominio de la base de datos, no se puede hablar de un diseño de modelo de dominio como tal, ya que en el desarrollo del front-end no se diseña ningún modelo de dominio, sino que se utiliza el ya existente. Se puede observar en la Figura 20 - Modelo de dominio de AdESMuS.

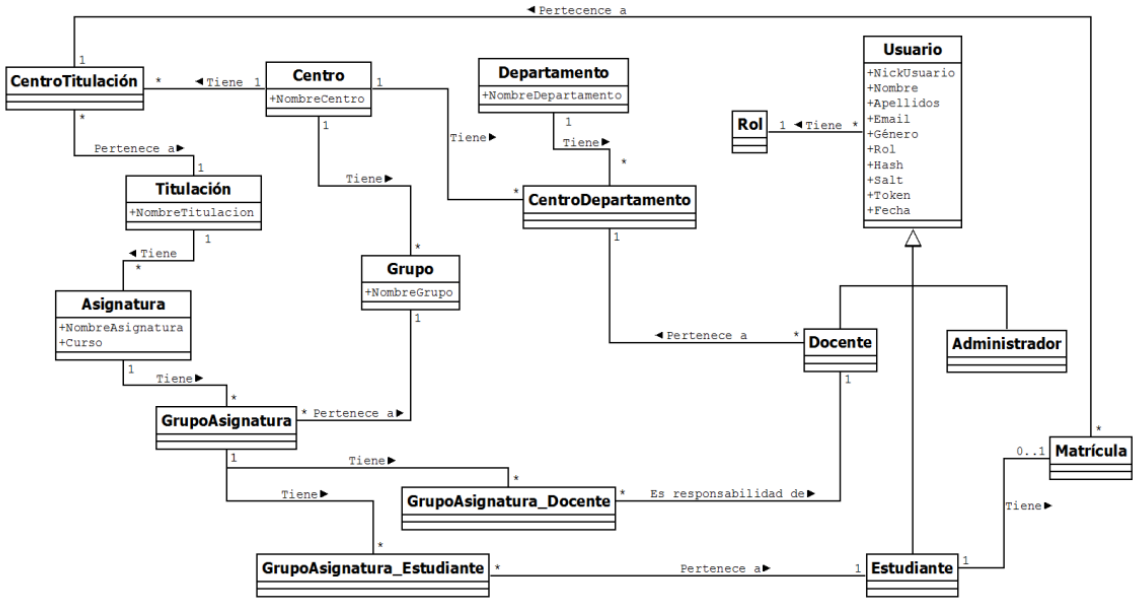


Figura 20 - Modelo de dominio de AdESMuS

El sistema gira en torno a siete conceptos: usuario, asignatura, centro, departamento, titulación, grupo y matrícula; que están relacionados entre sí y constituyen el esqueleto de la base de datos de la herramienta. Existen gran diversas relaciones entre los elementos: las asignaturas pertenecen a titulaciones, los grupos pertenecen a centros, se pueden asociar titulaciones y centros, etc. Los usuarios de AdESMuS se consideran un elemento más y son de tres tipos, según los permisos de los que dispongan: administrador, docente y estudiante (dado que las funciones relativas al supremo no se han desarrollado, ver 6.6. Limitaciones del front-end).

6. Desarrollo

En este apartado se expondrá el curso del desarrollo del proyecto y las técnicas utilizadas. Serán explicadas las partes más relevantes del desarrollo, ya sea por tener una importancia remarcable, por haber ocasionado una dificultad considerable o porque las características del back-end o del entorno hayan dado lugar a imprevistos en la elaboración del front-end.

6.1. Introducción

El proyecto busca construir un front-end, es decir, una interfaz gráfica para AdESMuS que lo haga usable. El front-end realiza al back-end las peticiones correspondientes a partir de las acciones del usuario, quedado este únicamente para entregar al front-end los datos necesarios; y recibiendo toda la información de éste, que se la muestra de una manera agradable y rápida, diseñado para realizar de una forma cómoda las tareas que el usuario desee realizar, e incluso informándole de las posibles.

El back-end funciona a base de peticiones, que son enviadas con parámetros y que devuelven una respuesta determinada. Sin embargo, aunque las funciones sencillas se pueden realizar con una sola petición, para acciones más avanzadas es necesario realizar un elevado número de peticiones; lo que implica que si el usuario desea realizar una tarea deba observar la lista de peticiones posibles, decidir cuáles van a ser las necesarias, realizarlas en el orden correcto, y; en muchos casos, anotar códigos recibidos que serán necesarios para peticiones posteriores.

6.2. Diseño de la interfaz

Previamente al comienzo de la programación, se analizó qué herramientas se utilizarían para el desarrollo, qué permitía cada una de ellas y cuáles de estos recursos serían incluidos en la interfaz.

La tecnología elegida para el desarrollo fue Angular, porque es un framework que suele utilizarse conjuntamente con Node.js, y dado que el back-end está desarrollado en ese lenguaje, es muy apropiado desarrollar el front-end en Angular. Asimismo, es una tecnología en auge, que hoy en día utilizan la mayor parte de las aplicaciones web; en favor del JavaScript, HTML y CSS puros **[11]**; ya que permite todas las funciones que éstos ofrecen, estructura y organiza el código de una forma muy práctica en componentes y además aporta algunas ventajas: automatiza algunos procesos, soporta gran cantidad de librerías, posee un funcionamiento jerárquico por componentes muy práctico, etc. **[12]**.

Figura 21 - Logotipo de Angular

6.3. Realización de la aplicación en Angular

Llegado el momento de comenzar la programación, se ha instalado Angular en el equipo, y posteriormente el software elegido para el desarrollo: Visual Studio Code (ver: Visual Studio Code). Ambos se han instalado descargándose de sus webs oficiales y abriendo sus instaladores ejecutables **[13]**. También se ha implantado la máquina que alberga al servidor de AdESMuS y se han realizado en ella las configuraciones pertinentes: el establecimiento de la dirección IP pública que utilizará AdESMuS (a la que se realizarán las peticiones), así como configurar el directorio con el código del back-end como una carpeta compartida para permitir acceder a su contenido desde fuera de la máquina (ver Figura 22 - Configuración de la carpeta del código compartida).

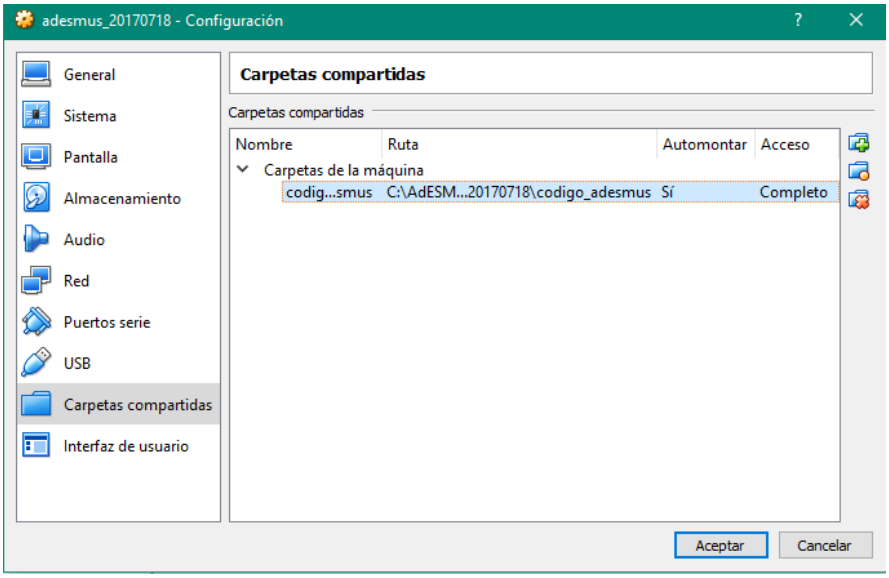


Figura 22 - Configuración de la carpeta del código compartida

La programación se ha realizado en Visual Studio Code, y en todo momento la máquina de AdESMuS ha estado funcionando. También ha sido de gran ayuda Postman, un software que se utiliza para realizar peticiones, y que se usó para las pruebas en el desarrollo del back-end **[6]**. En este caso, Postman se ha usado para probar y analizar las peticiones antes de invocarlas en el código del front-end.

En la Figura 23 - Esquema del desarrollo se puede observar la base del desarrollo de este proyecto. Destacar que el back-end y el front-end son dos sistemas distintos que se ejecutan a la vez, pero que como tales no dependen uno del otro.

Figura 23 - Esquema del desarrollo

En Angular, la administración del proyecto se realiza desde la terminal. Es por esa razón por la cual Visual Studio Code es un software tan apropiado para el desarrollo en Angular: porque permite operar con la terminal.

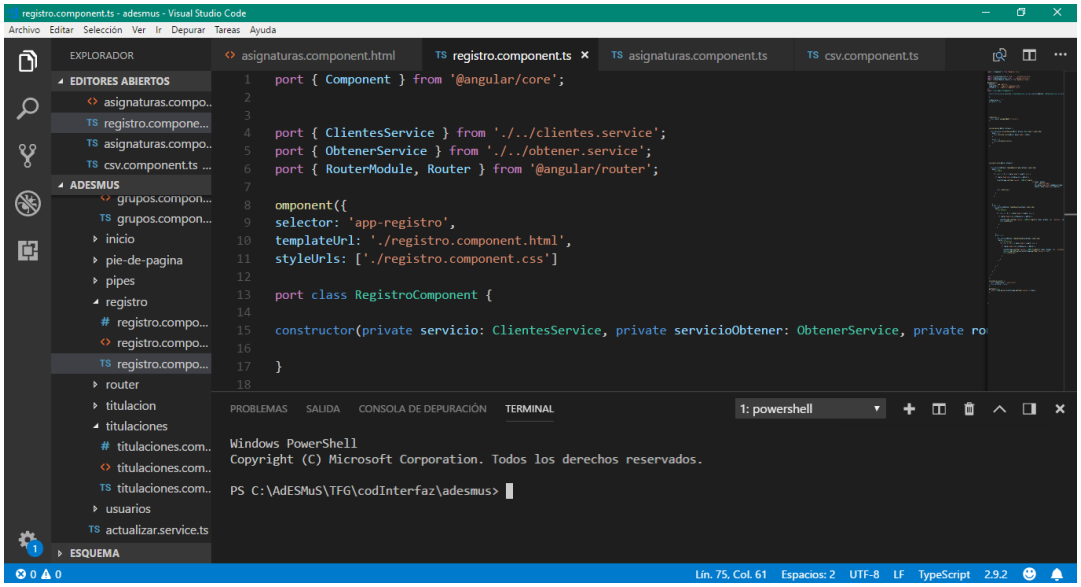


Figura 24 - Aspecto del entorno de trabajo en Visual Studio Code

Para comenzar la programación, se ha ejecutado el comando para crear un nuevo proyecto. Posteriormente, se ha generado el módulo AppModule, mediante su correspondiente comando y se han creado los componentes y servicios necesarios **[13]**. Una vez seguidos estos pasos, se ha comenzado a escribir el código.

6.4. Binding: la solución para la asincronía

Uno de los rasgos más característicos de la programación en Angular es el binding: la comunicación entre la vista y el controlador es en directo. Si un determinado valor cambia en el código, automáticamente se actualizará también su representación en la vista; y de la misma manera, si el valor de un elemento de entrada cambia en la vista (como un campo) automáticamente cambiará el valor del elemento de la variable del código que tenga asignada.

La característica del binding es muy apropiada para este proyecto, debido a que las respuestas a las peticiones se reciben de forma asíncrona. Cuando se realiza una petición al back-end, el tiempo que se tarde en recibirla depende de diversos factores: la velocidad con la que pueda trabajar el servidor en ese momento, si en ese mismo momento el servidor está atendiendo más peticiones, e incluso la velocidad de la conexión si el servidor se encuentra remotamente y conectado a través de la red.

Por ejemplo, si se realiza una petición como "obtener todas las asignaturas", se recibirán las asignaturas en un JSON, pero se deberán realizar más peticiones para conseguir las titulaciones a las que pertenecen. Estos datos se irán recibiendo progresivamente, según se vayan procesando las llamadas.

Al realizar una petición como esta, el front-end tiene una variable "asignaturas", que se trata de un array en el que almacenará las asignaturas recibidas y sus titulaciones. La vista contendrá una tabla que las muestre; y la variable asignaturas y esta tabla están conectadas mediante binding. Se realiza la petición y, cada vez que la titulación de una asignatura se recibe se añade al array. Gracias al binding, inmediatamente al añadir la titulación al array ya se muestra en la tabla en la vista. Esto aporta dos ventajas:

- Evita tener que recargar la página cada vez que un elemento se obtiene, ya que de forma automática ya aparece en la tabla al recibirse.
- Permite que no sea necesario esperar a recibir todos los elementos para visualizar algunos en la interfaz. Sin el binding, no sería factible recargar toda la página por cada elemento recibido, por lo que esto se haría al recibir el último elemento, lo que ocasionaría que el usuario sólo visualizase los elementos al haberse recibido todos. De este modo, los elementos aparecen progresivamente en la interfaz según se reciben, lo que evita que el usuario se frustre pensando que ha cometido algún error **[7]**.

Por sus múltiples ventajas, el binding se ha utilizado en toda la aplicación.

6.5. Limitaciones del back-end

El desarrollo del front-end, en todo momento, está programado orientado hacia el back-end, por lo que las funciones que éste incluya y sus características condicionarán de manera directa el desarrollo del front-end. El front-end funciona invocando las llamadas que el back-end implementa, e interpretando sus respuestas. Si las posibilidades que ofrece el back-end son limitadas, el desarrollo se ve comprometido, ya que se obliga al front-end a invocar más llamadas para obtener todos los datos que necesite o para llevar a cabo una determinada tarea.

Es el caso de AdESMuS, que en algunos casos no ofrece todas las posibilidades que debería, lo que ha sido uno de los mayores problemas durante el desarrollo: ha sido necesario realizar un front-end más denso para sortear las carencias del back-end o en dos determinadas situaciones intervenir en peticiones del back-end.

6.5.1. Escaso abanico de peticiones

El back-end de AdESMuS cuenta con una serie de peticiones posibles para administrar la información almacenada en la base de datos. Al comenzar el desarrollo de este proyecto se ha analizado el modelo de dominio, las peticiones posibles y la información que devuelven, y se ha llegado a la conclusión de que las peticiones del back-end son escasas y pobres. En algunos casos ofrecen información muy limitada que hace necesario realizar otras peticiones para disponer de todos los datos; por ejemplo, al solicitar todas las asignaturas únicamente se reciben sus IDs, pero se necesitan realizar más peticiones para a partir de éstos IDs obtener sus nombres y poder mostrarlos. Incluso se da el caso de dos datos determinado que no es posible obtener por medio de ninguna petición: el centro y el departamento de un docente y el centro y la titulación de la matrícula de un estudiante.

Desde el principio se ha buscado que el front-end muestre al usuario la mayor cantidad de información posible en una misma ventana. Por ejemplo; dado que una asignatura siempre pertenece a una y sólo una titulación, si se muestra una lista de asignaturas en formato de tabla es evidente que es muy útil añadir una columna que indique la titulación a la que pertenece cada una. Y a la hora de mostrar una titulación, lo más apropiado para ello es mostrar el nombre que la identifique de entre sus atributos, ya que es más informativo hacia el usuario que mostrar el ID.

Otra limitación del back-end se encuentra, por ejemplo, al intentar obtener todos los grupos que tiene un centro: mientras existen peticiones como “obtener todas las titulaciones asociadas a un centro” u “obtener todos los departamentos asociados a un centro”, no existe la petición “obtener todos los grupos de un centro”. Esto implica recurrir a la petición “obtener todos los grupos”, recorrer la respuesta y filtrar los grupos en base a su centro para obtener únicamente los grupos cuyo centro sea el centro en cuestión.

6.5.2. Falta de datos en las respuestas

Como se ha explicado anteriormente, en algunos casos, las respuestas de las peticiones son incompletas. La petición “Obtener lista de asignaturas”, por ejemplo, no indica el nombre de la titulación a la que pertenece, limitándose a indicar el ID de la misma. Esto obliga al front-end a realizar en ese mismo momento la petición “Obtener una titulación” (a partir de su ID) para acceder a su nombre y mostrarlo.

El problema del rol

Continuando en la misma línea; al iniciarse sesión en el back-end, en la respuesta se devuelven datos como el token, que incluye el rol; sin embargo, se encuentra cifrado por lo que no es posible acceder a él. Cierto es que un usuario, cuando inicia sesión, ya sabe si es administrador, docente o estudiante y no necesita que el back-end se lo indique. Siendo estrictos, este dato tampoco es necesario para el desarrollo de un front-end; ya que si el usuario no tiene permiso para realizar una función, el back-end no lo permitirá.

Sin embargo, se ha considerado más elegante que en todo momento el front-end conozca el rol del usuario, y en lugar de intentar realizar las funciones que ordene el usuario y toparse con un mensaje de error, simplemente no mostrar las opciones que el rol no permita llevar a cabo. Es decir; si el rol del usuario no permite eliminar, en lugar de que al pulsar en "Eliminar" aparezca un mensaje de error, directamente no mostrar el botón de eliminar. Así, el usuario sólo ve los datos y las funciones sobre las que tiene potestad para operar, y la pantalla queda más despejada.

Por tanto, era necesario conocer el rol del usuario. Para ello, se ha optado por un algoritmo prueba-error. Gracias a que un usuario no puede tener acceso a la información de los usuarios que se encuentran por encima de él (un docente no puede tener acceso a los administradores y un estudiante no puede tener acceso ni a los administradores ni a los docentes), se ha conseguido desarrollar un algoritmo que obtenga el rol del usuario. El procedimiento se puede ver en el diagrama de flujo de la Figura 25 - Diagrama de flujo del algoritmo de obtención del rol. Este algoritmo se ejecuta al momento de iniciar sesión y se almacena el dato obtenido, de manera que en toda la aplicación al usuario sólo se le mostrarán las opciones que tenga permiso para ejecutar.

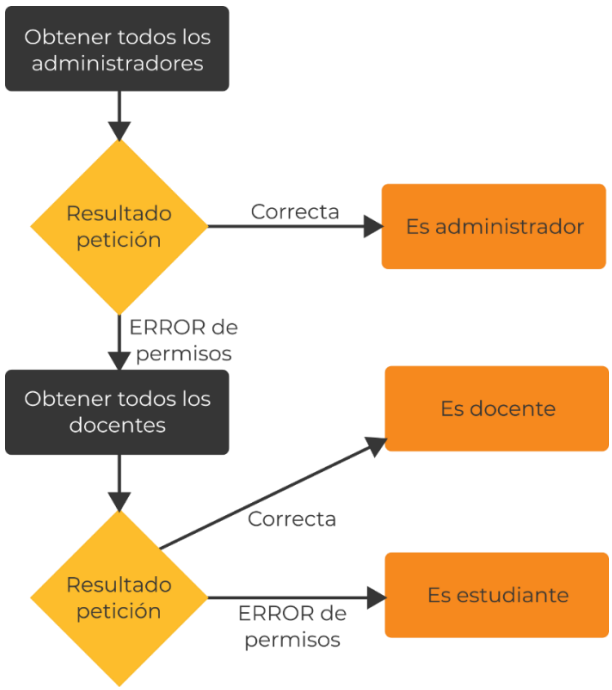


Figura 25 - Diagrama de flujo del algoritmo de obtención del rol

6.5.3. Intervención en el back-end

Una de las soluciones posibles al problema de la escasez de funcionalidades, que hubiese evitado esta necesidad de realizar un elevado número de peticiones, era intervenir en el back-end y modificar algunas de las peticiones para que devolviesen directamente la información necesaria, así como añadir nuevas peticiones más oportunas que las existentes.

Dada la complejidad de esta tarea, y debido a que el principal objetivo del proyecto era crear un sistema entre el usuario y el back-end que mediase entre ellos y no intervenir en el sistema ya existente; se ha optado por solventar esta dificultad en la medida de lo posible en el front-end, realizando todas las peticiones necesarias hasta tener la información deseada, en lugar de realizar modificaciones en el sistema.

Existen dos casos en los que ha sido necesaria la intervención en el back-end: el acceso al centro-titulación de una matrícula y el acceso al centro-departamento de un docente. La información de cuál es el centro-titulación al que se corresponde una matrícula y cuál es el centro-departamento al que pertenece un docente se establece correctamente al registrar una matrícula y un docente, incluso se almacena en la base de datos. Sin embargo, no hay ninguna posibilidad de acceder a esta información mediante las peticiones existentes. Para el caso de la matrícula, sí se indica el nombre del centro y la titulación; pero esta información no es útil ya que un centro-titulación no se identifica mediante nombres, sino mediante IDs: el del centro-titulación, o bien la combinación del ID del centro y el ID de la titulación. Pero en ningún caso un nombre, ya que aunque en la mayor parte de los casos sí es suficiente para identificar a un elemento, puede darse el caso de que existan varios elementos con el mismo nombre, ya que es un caso permitido por el back-end.

Por dicho motivo, ha sido necesario intervenir en el código de ambas peticiones. En cada caso, se han intervenido tanto la petición “obtener uno” como “obtener todos”, lo que resulta en cuatro peticiones.

Por motivos de seguridad, los IDs de los elementos de AdESMuS y con los que el usuario trabaja están cifrados mediante un algoritmo de cifrado bidireccional que se encuentra implementado en el back-end. Por tanto, han sido necesarios dos pasos: acceder al dato en la petición y cifrarlo, ya que el front-end trabaja con los datos del mismo modo que el usuario.

Modificaciones realizadas

Como se ha explicado previamente, al realizar la petición “obtener una matrícula” u “obtener todas las matrículas”, se obtenían por respuesta, en cada matrícula, una serie de datos, entre ellos el nombre del centro y el de la titulación a los que estaban asociadas; pero no los IDs de éstos, que son los que verdaderamente aportan la información.

En el caso de los docentes, ni siquiera se obtenían los nombres del centro y el departamento, por lo que se ha realizado una modificación similar para obtener los IDs.

Los datos requeridos ya se almacenan en las tablas Matrícula y Docente de la base de datos, por lo que la solución para acceder a los datos ha sido modificar las sentencias SQL de la peticiones, que se encuentran en los archivos *matricula.js* y *docente.js*, respectivamente, dentro del módulo de gestión universitaria. Los cambios realizados se puede ver en las siguientes tablas: Tabla 47 - Sentencia SQL de matrículas y

Tabla 48 - Sentencia SQL de docentes.

Tabla 47 - Sentencia SQL de matrículas

Sentencia SQL original
`SELECT codMatricula,nickUsuario,nombre,apellidos,email, centro.nombreCentro,titulacion.nombreTitulacion FROM matricula`
Sentencia SQL modificada
`SELECT codMatricula,nickUsuario,nombre,apellidos,email, **CentrotitulacioncodCentroTitulacion**,centro.nombreCentro, titulacion.nombreTitulacion FROM matricula`

Tabla 48 - Sentencia SQL de docentes

Sentencia SQL original
`SELECT docente.codDocente, docente.Usuariocodusuario, usuario.nickUsuario, usuario.nombre, usuario.apellidos, usuario.email, rol.rango FROM docente`
Sentencia SQL modificada
`SELECT docente.codDocente, **docente.CentrodepartamentocodCentroDepartamento**, docente.Usuariocodusuario, usuario.nickUsuario, usuario.nombre, usuario.apellidos, usuario.email, rol.rango FROM docente`

Posteriormente, ha sido necesario realizar otra modificación en ambos casos, para que los valores se devuelvan cifrados, que es como AdESMuS los necesita. En este caso, ha tenido lugar en los archivos *router_matriculas.js* y *router_docentes.js*, dentro del módulo de comunicación. En ellos se ha realizado una pequeña modificación, para indicar que cifre también los valores *CentrotitulacioncodCentroTitulacion* y *CentrodepartamentocodCentroDepartamento*.

6.6. Limitaciones del front-end

A continuación, se explicarán algunas funcionalidades de AdESMuS que no han sido implementadas en el front-end por distintas razones.

AdESMuS cuenta con cuatro usuarios: supremo, administrador, docente y estudiante. El supremo, sin embargo, no se considera un usuario real ni una persona de la comunidad universitaria **[6]**, dado que es un usuario creado en la aplicación para poder administrar todos los datos desde el punto de vista técnico, sin preocuparse por el rol ni los permisos (de hecho, ni siquiera existen peticiones relativas al supremo como "obtener todos los supremos", ni aparecen en el modelo de dominio: 5.4. Modelo de dominio). Los usuarios reales en la aplicación son el administrador, el docente y el estudiante.

Por este motivo, en el front-end no se ha desarrollado la parte correspondiente a este rol, ya que no tiene sentido que un usuario ficticio utilice el front-end. Por este motivo, si la base de datos de AdESMuS se encuentra vacía (sin usuarios), debe crearse un usuario inicial, necesariamente desde el back-end, para poder utilizar la aplicación.

Igualmente, aunque son objetivos del desarrollo del sistema la eficacia y la eficiencia (mostrar la mayor información disponible con el mínimo número de acciones por parte del usuario, ver 6.5. Limitaciones del back-end); existen ciertos datos que no le son mostrados al usuario: hashes, salts y, por motivos de seguridad, los tokens.

6.7. Seguridad en la aplicación

Hoy en día, la seguridad es una de las áreas más importantes en el desarrollo de aplicaciones, especialmente las que tratan datos delicados o privados. El back-end de AdESMuS ya implementa una serie de medidas de seguridad **[6]**, y por supuesto ha sido algo que ha primado constantemente en la creación del front-end: tener en cuenta y mantener las medidas implementadas en el back-end para preservar la seguridad que este implementa.

El back-end de AdESMuS cuenta con cinco capas de seguridad. Para el desarrollo del front-end, ha sido necesario tomar en consideración las características de cada una de ellas, con el objetivo de preservar la seguridad que AdESMuS tiene.

6.7.1. Primera capa: Token

El token es un código secreto que identifica una sesión iniciada y autentica al usuario, como si de una llave se tratase. El front-end almacena este token al recibirlo de parte del back-end, pero en ningún momento lo muestra; ni siquiera al usuario en cuestión.

El token debe ser almacenado en un lugar donde se pueda obtener durante toda la sesión y desde cualquier parte de la aplicación. Dado que durante una sesión se realizan routings y se cambia de componentes, no tiene sentido almacenarlo como un atributo del componente.

En su lugar, se ha optado por utilizar el *local storage* o almacenamiento local de Angular. Este recurso permite almacenar información no en un componente, sino en la sesión en general, y a la que se puede acceder desde cualquier componente. El *local storage* funciona como una lista clave-valor: para cada clave en forma de texto almacena un objeto. En este caso, se utiliza

la clave 'sesion', y se guarda en ella un JSON con cuatro datos relativos a la sesión: el token, el rol, el ID y el nombre del usuario. La línea que almacena esta información en el momento de iniciar sesión se puede ver en la Figura 26 - Guardado de los datos de la sesión.

```
localStorage.setItem('sesion', JSON.stringify
                                        ({ token: pToken,
                                           rol: 'administrador',
                                           id: data['body'][i].codAdministrador,
                                           nombre: data['body'][i].nombre}));
```

Figura 26 - Guardado de los datos de la sesión

Mientras que utilizando el back-end el usuario debía anotar su token, haciendo uso del front-end, éste se encarga de ese trabajo y el usuario no necesita ver su token, lo que aporta seguridad extra al sistema, evitando que sea visto por terceros.

En la Figura 27 - Ejemplo de adición del token en una petición se puede observar cómo se accede al valor del token en el *local storage* y se adjunta en la petición para obtener todos los grupos.

```
obtenerTodosGrupo() {
  this.servicioObtener.todosGrupo(JSON.parse(localStorage.getItem('sesion')).token).subscribe(
    resp => {

      this.grupos = resp['body'];

    }
  );
```

Figura 27 - Ejemplo de adición del token en una petición

Tras un tiempo sin que el back-end reciba peticiones (en este caso, porque el usuario no utiliza el front-end), el token caduca y deja de ser válido en las siguientes peticiones, lo que hace necesario iniciar sesión de nuevo para obtener otro. En el caso del front-end, automáticamente se traslada al usuario al inicio de sesión.

El token es creado a partir de una serie de datos, entre los que se encuentra la denominada huella. Este valor es uno de los que se deben incluir en la petición en el momento de iniciar sesión. Al iniciar la sesión, el sistema crea el valor del token a partir de distintos datos y cifra el conjunto, pero es vital que este token sea único, ya que tiene el mismo funcionamiento que una llave.

La huella sirve para garantizar la no repetición del token, aunque el valor que se le da sólo influye en la seguridad, y no en el funcionamiento, siendo posible establecer cualquier valor para la misma.

Como la huella es uno de los valores que se utilizan para la creación del token, con huellas distintas siempre se obtendrán tokens distintos. Por tanto, se debe buscar una forma de utilizar huellas que asegure que estas serán únicas.

Se ha optado por que el front-end de AdESMuS automáticamente asigna a la huella un valor que se obtiene al invocar la función *Date.now()*. Este valor es un número entero, que se corresponde con los milisegundos que han transcurrido desde el 1 de enero de 1970 a las 00:00:00. Es, por tanto, un valor único que jamás se repetirá: lo ideal para evitar que dos sesiones tengan la misma huella.

6.7.2. Segunda capa: Roles

En AdESMuS existen los roles de usuario; disponiendo de distintos permisos cada uno. En el front-end se ha respetado este sistema para preservar la seguridad. Por tanto, los datos que un usuario no tiene permiso para ver, simplemente no se muestran. Asimismo, los botones, campos y otros elementos utilizados para funcionalidades de modificación únicamente aparecen en el front-end si el usuario tiene potestad para realizar dichas funciones, mostrándose únicamente los elementos y datos con los que el usuario tenga permiso para interactuar.

6.7.3. Tercera y cuarta capas: Encriptación interna del back-end

En estas capas, no ha sido necesario tomar ninguna medida, ya que únicamente afecta al funcionamiento interno del back-end y por tanto, no es relevante que sean tenidas en cuenta a la hora de desarrollar en front-end.

6.7.4. Quinta capa: Encriptación de identificadores

Todos los elementos que AdESMuS almacena en su base de datos están provistos de un ID, un código que los identifica. Este código es un número entero asignado de forma ordinal; es decir, el ID del primer elemento será 1, el del segundo será 2, etc. Sin embargo, el back-end siempre devuelve los IDs cifrados, ya que es como fuera del mismo se debe trabajar con ellos, para así evitar que se conozca el orden del elemento, ya que podría aportar información innecesaria. Este cifrado es bidireccional y se encuentra implementado en el back-end.

Si no se hubiese intervenido en el back-end (ver 6.5.3. Intervención en el back-end), esta capa no habría sido relevante a la hora de desarrollar el front-end. Sin embargo, dado que ha sido necesario realizar una modificación para obtener el valor de dos IDs, también ha sido necesario realizar una pequeña modificación del código interno para que ambos valores se devuelvan cifrados (explicado en el apartado anteriormente citado).

6.8. La eliminación en cadena

En una base de datos, las tuplas de una determinada tabla están relacionadas con tuplas de otras tablas. Por ejemplo; si la una BD almacena personas y vehículos, y cada vehículo pertenece siempre a una persona, cada tupla de la tabla "Vehículo" irá relacionada con una tupla de la tabla "Persona" (su dueño). Si se desea eliminar una persona de la BD, los vehículos que tuviese carecerían de sentido, y por ello es necesario que también se eliminen al eliminar la persona.

Gran cantidad de BDs ya están provistas de una solución a este problema: el borrado en cascada. Al eliminar una tupla, automáticamente eliminan todas las tuplas de todas las tablas que pierden la coherencia por esta eliminación: si se elimina una ciudad, se eliminan todas las personas que habitan en esa ciudad, y por tanto todos los vehículos que poseen cada una de esas personas, lo que da lugar a un borrado en cadena que se lleva a cabo de manera automática.

Por desgracia para este desarrollo, la base de datos de AdESMuS no lleva implementado el borrado en cascada. En su lugar, sí incluye un sistema de prevención a esta situación: simplemente no permite eliminar tuplas si éstas van a dejar a otras tuplas "huérfanas". Es decir, no permite borrar un elemento si existen elementos de otro tipo asociados a él: primero se deben borrar éstos. Esto recibe el nombre de restricciones de integridad.

El hecho de poder eliminar un elemento con un click era algo imprescindible en el front-end, una tarea que habría sido muy sencilla de haber implementado la BD el borrado en cascada. Sin embargo, este hecho, al igual que otras carencias del back-end; ha engordado el código del front-end, obligando a buscar y eliminar todas las tuplas que quedarían huérfanas desde el front-end antes de proceder a la eliminación.

A este problema se ha sumado el hecho de que la base de datos de AdESMuS tiene un gran número de tablas y relaciones entre ellas: una funcionalidad tan sencilla como es eliminar un centro, obliga a que deban eliminarse:

- Todos los centro-titulación del centro
    - Todas las matrículas de cada uno de ellos
- Todos los centro-departamento del centro
    - Todos los docentes de cada uno de ellos
        - Todos los grupo-asignatura-docente de cada uno de ellos
- Todos los grupos del centro
    - Todos los grupo-asignatura de cada uno de ellos
        - Todos los grupo-asignatura-estudiante de cada uno de ellos
        - Todos los grupo-asignatura-docente de cada uno de ellos

Por tanto, durante el desarrollo, el borrado ha sido una de las partes que mayores dificultades han ocasionado, ya que ha sido necesario llevar a cabo este borrado de cada tipo de elemento en el front-end.

6.9. Importación de datos mediante CSV

Una de las partes que se ha desarrollado por requerimiento del cliente final ha sido la importación de datos mediante archivos CSV. A pesar de no ser una parte requerida por la estructura del back-end, se ha considerado una parte relevante del desarrollo.

Muchas aplicaciones de hoy en día trabajan con grandes cantidades de datos; y por tanto, son imprescindibles las funciones que permitan trabajar con más de un dato a la vez. La base de datos de AdESMuS está preparada y desarrollada para albergar las cantidades de datos correspondientes a la información que puede necesitar almacenar una universidad; y, por tanto, no es factible que sólo sea posible trabajar con datos de forma individual.

Esto hacía necesario implementar una forma de añadir datos a la BD a partir de la información existente en un archivo, de manera que se automatice el proceso de lectura de los datos (en un formato específico, requerido por la aplicación) y las llamadas a las peticiones de adición a la base de datos.

Aunque existen multitud de formatos en los que se podría realizar esta importación (TXT, Microsoft Excel, JSON, XML…) se ha optado por el CSV, ya que se estructura en filas y columnas, al estilo de una hoja de cálculo, pero es un formato abierto **[8]**. En la Figura 28 - Formato de un CSV para importar asignaturas se puede ver la estructura requerida para la importación de datos (en este caso, de asignaturas). En la primera columna se indica el nombre, y en la segunda el ID de la titulación que ya debe existir en la BD.

	A	B
1	Francés Técnico	322f9adcad1904a38ab718ea2429fb58
2	Economía y Empresa	4552f288cc55118a8394fdcef4b1dbe3
3	Matematika eta Estatistika	322f9adcad1904a38ab718ea2429fb59
4	Econometría	4552f288cc55118a8394fdcef4b1dbe4
5	Giza Anatomia	322f9adcad1904a38ab718ea2429fb60
6	Física	4552f288cc55118a8394fdcef4b1dbe5
7	Hezkuntzaren Soziologia	322f9adcad1904a38ab718ea2429fb61
8		

Figura 28 - Formato de un CSV para importar asignaturas

Para implementarlo, se ha incluido un botón “Importar CSV” en todos los componentes de cada tipo de entidad, que lleva a la ventana de importación de datos mediante archivos CSV. Esta ventana incluye un botón para subir archivos; y al seleccionar uno mediante el explorador del equipo, que se abre automáticamente, se recoge por el front-end.

Posteriormente, se lee su contenido y se despieza: se separan las filas (que se corresponden con los elementos), y posteriormente se parten en celdas (que se corresponden con cada uno de los atributos requeridos según el tipo de entidad correspondiente). Estos elementos, ya preparados, se insertan en la base de datos mediante una llamada recursiva al método “registrar” del tipo correspondiente. El código resumido correspondiente al método de importación se puede ver a continuación.

En la Figura 29 - Código resumido de la importación de datos mediante un archivo CSV (I) se observa el código que comienza la lectura del archivo, y convierte el contenido a una lista de elementos (separando por los saltos de línea, es decir, las filas).

```
tipo = '';
Name: string;
myFile: File;
importarCSV(files: any) {
  var reader = new FileReader();
  reader.readAsText(files[0], "ISO-8859-1");
  var that = this;
  reader.onload = function (evt) {
    var texto = reader.result;
    var elementos = texto.split("\n");
```

Figura 29 - Código resumido de la importación de datos mediante un archivo CSV (I)

En la Figura 30 - Código resumido de la importación de datos mediante un archivo CSV (II) se ve cómo dependiendo del tipo de elemento se hace el despiece de celdas de distinta manera (ya que no todos los elementos requieren los mismos atributos) y se llaman a distintas funciones.

```
    if (that.tipo == 'centros') {
      for (var i = 0; i < elementos.length; i++) {
        var nombreCentro = elementos[i].split(";")[0];
        that.registrarUnoCentro(nombreCentro);
      }
    }
    else if (that.tipo == 'titulaciones') {
      for (var i = 0; i < elementos.length; i++) {
        var nombreTitulacion = elementos[i].split(";")[0];
        that.registrarUnoTitulacion(nombreTitulacion);
      }
    }
    /*
    [...]
    */
```

Figura 30 - Código resumido de la importación de datos mediante un archivo CSV (II)

Por último, en la Figura 31 - Código resumido de la importación de datos mediante un archivo CSV (III) se ve cómo, si la importación ha sido correcta se muestra el mensaje de confirmación, mientras que si no lo ha sido se muestra el mensaje de error correspondiente.

```
    alert(elementos.length + ' elementos añadidos correctamente');
  }
  reader.onerror = function (evt) {
    alert('Error: se ha detenido la importación por la existencia de un dato corrupto');
  }
}
```

Figura 31 - Código resumido de la importación de datos mediante un archivo CSV (III)

Es probable que el contenido del archivo tuviese algún error: por ejemplo, puede darse el caso de que la estructura del archivo no sea la esperada, o que se esté haciendo referencia mediante un ID a

un elemento que no se encuentre en la base de datos. En tal caso, se detendrá la importación, avisando al usuario mediante un mensaje de error, aunque los datos ya cargados hasta el momento quedarán importados.

6.9.1. Codificación y estándar

La codificación utilizada para el texto del archivo CSV ha sido ISO-8859-1, que permite el uso de tildes y eñes, ya que la codificación por defecto, ASCII, no las acepta **[9]**; y AdESMuS debe poder alojar datos en castellano.

7. Evaluación de la usabilidad

En este apartado se evaluará y testeará la usabilidad del sistema, así como la del back-end por sí sólo para analizar la calidad del proyecto llevado a cabo y en qué medida ha facilitado el uso de la herramienta AdESMuS. El análisis del back-end, aunque a efectos está verificando algo no creado en este proyecto, ayudará a comparar los resultados con los del front-end; para así cuantificar la mejora ganada, que es uno de los principales objetivos de este proyecto.

Para ello, se acudió al tutor del TFG para asegurarse de que la trayectoria seguida era la correcta, a la vez que se realizó un estudio práctico, que sirvió tanto para evaluar la usabilidad como para retroalimentar el desarrollo con mejoras propuestas por los usuarios.

7.1. Estudio mediante una tarea

Se ha realizado un estudio con seis usuarios cercanos, con distintos niveles de conocimientos informáticos: desde estudiantes del Grado en Ingeniería Informática de Gestión y Sistemas de Información (tres de ellos) hasta personas con conocimientos en tecnología muy limitados que hacen un uso muy reducido de la informática y siempre para las funcionalidades más básicas.

Para ello, se ha propuesto a todos los usuarios realizar una misma tarea con el front-end desarrollado; para analizar las dificultades o problemas encontrados y el tiempo consumido. El enunciado ofrecido a los usuarios sometidos al estudio se puede ver en la Figura 32 - Ejercicio propuesto.

Modifica el nombre del departamento llamado “Idiomas y lenguas modernas” a “Idiomas y lenguas extranjeras”

DATOS
usuario: adesmus1
contraseña: 14069w

Figura 32 - Ejercicio propuesto

7.2. Resultados del estudio

Todos los usuarios fueron capaces de modificar el nombre del departamento haciendo uso del front-end. Cinco de ellos (entre los cuales se encontraban los estudiantes del Grado) lo consiguieron al primer intento, incluso dos de ellos haciendo uso de los campos de búsqueda, mientras que el restante tuvo que navegar por la aplicación hasta encontrar el dato que buscaba.

7.3. Análisis de los resultados y mejoras propuestas

Los pasos necesarios para llevar a cabo la tarea se pueden ver en la Tabla 49 - Pasos para llevar a cabo la tarea.

Tabla 49 - Pasos para llevar a cabo la tarea

Hipotético uso sin front-end	Front-end
1.- Realizar petición "Login", incluyendo en el body el usuario y la contraseña e introduciendo cualquier valor para la huella. Anotar el token recibido, ya que será necesario para realizar las peticiones posteriores. 3.- Realizar petición "Obtener todos los departamentos" (incluyendo el token) y buscar en la respuesta el departamento indicado, uno por uno. Anotar el código del departamento. 4.- Realizar petición "Actualizar un departamento" (incluyendo el token y el código del departamento), e indicando el nuevo nombre. **Comprobación** Realizar petición "Obtener un departamento" (incluyendo el token y el código del departamento) y comprobar que el departamento ya aparece con el nuevo nombre.	1.- Introducir usuario y contraseña y pulsar en "INICIAR SESIÓN". 2.-Teclear el nombre del departamento en la funcionalidad de búsqueda hasta que aparezca y pulsar en el icono del lápiz. 3.- Modificar el nombre en el campo y pulsar en "GUARDAR". **Comprobación** En la barra, pulsar en "CENTROS" y comprobar que en la lista aparece ya con el nuevo nombre.

Sin que el usuario lo note, el front-end realiza un gran número de tareas, lo que da lugar a un sistema muy cómodo hacia el usuario. Tras este estudio, se puede comprobar que el front-end desarrollado, por su parte; es eficiente, efectivo y produce satisfacción al usuario, ya que en muy poco tiempo todos los usuarios han sido capaces de completar la tarea; debido a que la estructura y la maquetación de la interfaz son las que implementan la mayoría de interfaces hoy en día.

7.3.1. Comentarios de los usuarios testeados

Algunas propuestas de los usuarios han sido incluidas en el desarrollo definitivo del front-end. La mayoría se trataban de cambios menores relacionados con la terminología utilizada, la ubicación de los elementos en la página o los colores utilizados. Algunos de los más relevantes han sido los siguientes:

- Las imágenes de fondo que contiene la aplicación deberían adquirir un tono más oscuro o difuminado que permita distinguirlas de los elementos.
- Convendría poder ordenar los elementos alfabéticamente en orden ascendente y descendente.
- Es muy cómodo disponer del botón del lápiz, que evita tener que entrar al elemento, y permite acceder directamente a la pantalla de editar.
- La búsqueda rápida del menú es muy directa, ya que se muestra inmediatamente al iniciar sesión.

Los comentarios de mejora se han utilizado como retroalimentación al desarrollo y han sido puestos en práctica, por lo que todos ellos han sido incluidos en la versión definitiva. Por su parte, el tutor del proyecto sugirió la funcionalidad de importación de elementos mediante archivos CSV, que también ha sido llevada a la práctica.

8. Pruebas unitarias

En este apartado serán expuestas todas las pruebas realizadas durante el desarrollo. Mientras que en otros desarrollos, como la creación de back-ends o de aplicaciones en Java existen herramientas específicas para la realización de pruebas; en Angular la manera más práctica de testear el funcionamiento del código programado es probándolo desde el navegador como un usuario, haciendo uso de la consola en los casos necesarios.

8.1. Pruebas relativas al inicio y cierre de sesión

Tabla 50 - Pruebas relativas al inicio y cierre de sesión

Caso nº	Características	Situación esperada	Situación ocurrida	Motivo del error
1a	Inicio con datos correctos	Redirección al menú personal	No hay redirección	Error en la petición de inicio de sesión
1b	Inicio con datos correctos	Redirección al menú personal	Redirección al menú personal	
2a	Inicio con datos incorrectos	Mensaje de error y marcado en rojo de los campos	Mensaje de error y marcado en rojo de los campos	
3a	Cierre de sesión	Redirección al inicio y borrado del token	Redirección al inicio sin borrado del token	La llamada al método clear no es correcta
3b	Cierre de sesión	Redirección al inicio y borrado del token	Redirección al inicio y borrado del token	

8.2. Pruebas relativas al buscador inicial

Tabla 51 - Pruebas relativas al buscador inicial

Caso nº	Características	Situación esperada	Situación ocurrida	Motivo del error
1a	Búsqueda de texto	Aparecen los elementos correspondientes ordenados alfabéticamente	No aparecen los estudiantes	Una condición del "if" incorrecta, los estudiantes están entrando como docentes
1b	Búsqueda de texto	Aparecen los elementos correspondientes ordenador alfabéticamente	Aparecen todos los elementos, pero ordenados por su tipo	Por una razón desconocida, en el *ngFor se debe indicar siempre el orden al final, ya que de lo contrario no funciona.
1c	Búsqueda de texto	Aparecen los elementos correspondientes ordenador alfabéticamente	El icono de departamentos no aparece en los departamentos	El nombre del archivo no es *departamentos.png* sino *dptos.png*
1d	Búsqueda de texto	Aparecen los elementos correspondientes ordenados alfabéticamente	Aparecen los elementos correspondientes ordenados alfabéticamente	

8.3. Pruebas relativas a la gestión de centros

Tabla 52 - Pruebas relativas a la gestión de centros

Caso nº	Características	Situación esperada	Situación ocurrida	Motivo del error
1a	Obtener todos los centros	Aparecen todos los nombres de los centros	No aparece nada	El JSON no se está recibiendo: necesario acepar la advertencia de seguridad
1b	Obtener todos los centros	Aparecen todos los nombres de los centros	No aparece nada	Al tomar el atributo del JSON se debe realizar con un punto en lugar de corchetes
1c	Obtener todos los centros	Aparecen todos los nombres de los centros	Aparecen todos los nombres de los centros	
2a	Búsqueda de centro por texto	Aparecen los centros correspondientes	Aparecen los centros correspondientes (pero distinguiendo entre mayúsculas y minúsculas)	Se debe añadir *.toUpperCase()* en el texto buscado y en los nombres para que sólo tenga en cuenta las letras
2b	Búsqueda de centro por texto	Aparecen los centros correspondientes	Aparecen los centros correspondientes	
3a	Añadir un centro	El centro se añade	El centro se añade	
4a	Modificar un centro (su nombre)	El nombre se modifica	No ocurre nada	Error en los parámetros en la petición PUT
4b	Modificar un centro (su nombre)	El nombre se modifica	El nombre se modifica	
5a	Eliminar un centro	El centro se elimina	El centro no se elimina	DELETE no se está ejecutando correctamente: se debe recoger el resultado
5b	Eliminar un centro	El centro se elimina	Aparece un error	El centro tiene elementos asociados
5c	Eliminar un centro	El centro se elimina	El centro se elimina	

8.4. Pruebas relativas a la gestión de titulaciones

Tabla 53 - Pruebas relativas a la gestión de titulaciones

Caso nº	Características	Situación esperada	Situación ocurrida	Motivo del error
1a	Obtener todas las titulaciones	Aparecen todos los nombres de las titulaciones	Aparecen todos los nombres de las titulaciones	
2a	Búsqueda de titulación por texto	Aparecen las titulaciones correspondientes	Aparecen las titulaciones correspondientes	
3a	Añadir una titulación	La titulación se añade	La titulación no se añade	Se debe modificar: *degree* en lugar de *center*
3b	Añadir una titulación	La titulación se añade	La titulación se añade	
4a	Modificar una titulación (su nombre)	El nombre se modifica	El nombre se modifica	
5a	Eliminar una titulación	La titulación se elimina	La titulación no se elimina	DELETE no se está ejecutando correctamente: se debe recoger el resultado

8.5. Pruebas relativas a la gestión de departamentos

Tabla 54 - Pruebas relativas a la gestión de departamentos

Caso nº	Características	Situación esperada	Situación ocurrida	Motivo del error
1a	Obtener todos los departamentos	Aparecen todos los nombres de los departamentos	Aparecen todos los nombres de los departamentos	
2a	Búsqueda de departamento por texto	Aparecen los departamentos correspondientes	Aparecen otros departamentos	El campo por el que se está filtrando era el ID en lugar del nombre
2b	Búsqueda de departamento por texto	Aparecen los departamentos correspondientes	Aparecen los departamentos correspondientes	
3a	Añadir un departamento	El departamento se añade	El departamento se añade	
4a	Modificar un departamento (su nombre)	El nombre se modifica	El nombre se modifica	
5a	Eliminar un departamento	El departamento se elimina	El departamento no se elimina	No se está adjuntando el token para el borrado, por lo que no se realiza
5b	Eliminar un departamento	El departamento se elimina	El departamento se elimina	

8.6. Pruebas relativas a la gestión de asignaturas

Tabla 55 - Pruebas relativas a la gestión de asignaturas

Caso nº	Características	Situación esperada	Situación ocurrida	Motivo del error
1a	Obtener todas las asignaturas	Aparecen todos los nombres y las titulaciones de las asignaturas	No aparece nada	Error de sintaxis en el GET
1b	Obtener todas las asignaturas	Aparecen todos los nombres y las titulaciones de las asignaturas	Aparecen los nombres pero no las titulaciones	El método que obtiene el nombre de la titulación se debe reimplementar, ya que la respuesta es asíncrona
1c	Obtener todas las asignaturas	Aparecen todos los nombres y las titulaciones de las asignaturas	Aparecen todos los nombres y las titulaciones de las asignaturas	
2a	Búsqueda de asignatura por nombre	Aparecen las asignaturas correspondientes	Aparecen todas las asignaturas	Error en el filtro: no se ha aplicado
2b	Búsqueda de asignatura por nombre	Aparecen las asignaturas correspondientes	Aparecen las asignaturas correspondientes	
3a	Búsqueda de asignatura por titulación	Aparecen las asignaturas correspondientes	Aparecen las asignaturas correspondientes	
4a	Añadir una asignatura	La asignatura se añade	La asignatura no se añade	No se está guardando correctamente la titulación seleccionada
4b	Añadir una asignatura	La asignatura se añade	La asignatura se añade	
5a	Modificar una asignatura (su nombre)	El nombre se modifica	El nombre se modifica	
6a	Eliminar una asignatura	La asignatura se elimina	La asignatura se elimina	

8.7. Pruebas relativas a la gestión de grupos

Tabla 56 - Pruebas relativas a la gestión de grupos

Caso nº	Características	Situación esperada	Situación ocurrida	Motivo del error
1a	Obtener todos los grupos	Aparecen todos los nombres de los grupos	Aparecen todos los nombres de los grupos	
2a	Búsqueda de grupo por texto	Aparecen los grupos correspondientes	Aparecen otros grupos	El campo por el que se está filtrando era el centro en lugar del nombre
2b	Búsqueda de grupos por nombre	Aparecen los grupos correspondientes	Aparecen los grupos correspondientes	
3b	Búsqueda de grupos por texto	Aparecen los grupos correspondientes	Aparecen los grupos correspondientes	
4a	Añadir un grupo	El grupo se añade	El grupo se añade	
5a	Modificar un grupo (su nombre)	El nombre se modifica	El nombre se modifica	
6a	Eliminar un grupo	El grupo se elimina	El grupo no se elimina	No se está adjuntando el token para el borrado, por lo que no se realiza
6b	Eliminar un grupo	El grupo se elimina	El grupo se elimina	

8.8. Pruebas relativas a la gestión de estudiantes

Tabla 57 - Pruebas relativas a la gestión de estudiantes

Caso nº	Características	Situación esperada	Situación ocurrida	Motivo del error
1a	Obtener todos los estudiantes	Aparecen todos los datos de los estudiantes	No aparece nada	Error de sintaxis en el GET
1b	Obtener todos los estudiantes	Aparecen todos los datos de los estudiantes	Aparece todo excepto los apellidos	La función que inserta una coma después de los apellidos no está mostrando los apellidos
1c	Obtener todos los estudiantes	Aparecen todos los datos de los estudiantes	Aparecen todos los datos de los estudiantes	
2a	Búsqueda de estudiante por nombre	Aparecen los estudiantes correspondientes	Aparecen todos los estudiantes	Error en el filtro: no se ha aplicado
2b	Búsqueda de estudiante por nombre	Aparecen los estudiantes correspondientes	Aparecen los estudiantes correspondientes	
3a	Búsqueda de estudiante por titulación de matrícula	Aparecen los estudiantes correspondientes	Aparecen los estudiantes correspondientes	
4a	Añadir un estudiante	El estudiante se añade	El estudiante no se añade	No se está guardando correctamente el centro-titulación de matrícula seleccionado
4b	Añadir un estudiante	El estudiante se añade	El estudiante se añade	
5a	Modificar un estudiante (su nombre)	El nombre se modifica	El nombre se modifica	
6a	Eliminar un estudiante	El estudiante se elimina	El estudiante se elimina	

8.9. Pruebas relativas a la gestión de docentes

Tabla 58 - Pruebas relativas a la gestión de docentes

Caso nº	Características	Situación esperada	Situación ocurrida	Motivo del error
1a	Obtener todos los docentes	Aparecen todos los nombres de los docentes	Aparecen todos los nombres de los docentes	
2a	Búsqueda de docentes por texto	Aparecen los docentes correspondientes	Aparecen otros docentes	Error en la condición de la sentencia *if*
2b	Búsqueda de docente por nombre	Aparecen los docentes correspondientes	Aparecen los docentes correspondientes	
3a	Añadir un docente	El docente se añade	El docente se añade	
4a	Modificar un docente (su nombre)	El nombre se modifica	El nombre se modifica	
5a	Eliminar un docente	El docente se elimina	El docente no se elimina	Se ha realizado la prueba con un docente que tiene grupo-asignatura asociados
5b	Eliminar un docente	El docente	El docente	

8.10. Pruebas relativas a la gestión de administradores

Tabla 59 - Pruebas relativas a la gestión de administradores

Caso nº	Características	Situación esperada	Situación ocurrida	Motivo del error
1a	Obtener todos los administradores	Aparecen todos los datos de los administradores	No aparece nada	El JSON recibido se está interpretando de forma incorrecta
1b	Obtener todos los administradores	Aparecen todos los datos de los administradores	No aparece nada	Advertencia de seguridad no aceptada
1c	Obtener todos los administradores	Aparecen todos los datos de los administradores	Aparecen todos los datos de los administradores	
2a	Búsqueda de administrador por nick	Aparecen los administradores correspondientes	Aparecen los administradores correspondientes (pero distinguiendo entre mayúsculas y minúsculas)	Se debe añadir de nuevo *.toUpperCase()* en el texto buscado y en los nombres para que sólo tenga en cuenta las letras
2b	Búsqueda de administrador por nick	Aparecen los administradores correspondientes	Aparecen los administradores correspondientes	
3a	Añadir un administrador	El administrador se añade	El administrador se añade	
4a	Modificar un administrador (su nombre)	El nombre se modifica	No ocurre nada	Error en los parámetros en la petición PUT
4b	Modificar un administrador (su nombre)	El nombre se modifica	El nombre se modifica	
5a	Eliminar un administrador	El administrador se elimina	El administrador no se elimina	DELETE no se está ejecutando correctamente: se debe recoger el resultado
5b	Eliminar un administrador	El administrador se elimina	El administrador se elimina	

9. Conclusiones y trabajo futuro

Como último apartado, se expondrán las conclusiones obtenidas tras el desarrollo del proyecto.

9.1. ¿Se han cumplido los objetivos?

En primer lugar, se analizará hasta qué punto se han cumplido los objetivos fijados.

Eficiencia

Se ha creado una aplicación eficiente, en la que se pueden realizar las acciones con el mínimo número de acciones posible.

Junto a todos los elementos en los que el usuario tiene permiso para editar/eliminar, se ha colocado un botón de editar y otro de eliminar, para que no sea necesario acceder al elemento y esta función pueda ser llevada a cabo desde fuera.

También se ha implementado el buscador en el menú principal para acceder desde él a cualquier elemento del sistema.

Efectividad

Se ha perseguido en todo momento la máxima efectividad, aunque en algunos casos no ha sido posible tenerla. Por ejemplo, en algunos casos la aplicación se comporta de una manera lenta, ya que los datos tienen que viajar en una petición y mostrarse, lo que podría tomar más tiempo si el equipo está llevando a cabo otras tareas.

Satisfacción por parte del usuario

Se ha conseguido la satisfacción por parte del usuario, como se ha podido comprobar con usuarios reales (ver 7. Evaluación de la usabilidad). No se muestran los elementos para acciones que el usuario no tiene permiso para llevar a cabo y se ha controlado la caducidad de la sesión, llevando al usuario a la página de inicio.

Preservar la seguridad

En lo que ha seguridad se refiere, se ha preservado en todo momento. Se ha controlado la privacidad de los tokens, se ha trabajado con los IDs cifrados y se han preservado los roles del usuario.

9.2. Novedades del proyecto con respecto a trabajos anteriores

Personalmente, este proyecto ha sido muy distinto a los realizados previamente en la vida académica en todos los sentidos. En primer lugar, nunca se había realizado un trabajo de estas dimensiones, siendo la mayoría de ellos trabajos más cotidianos, de duraciones no superiores a los tres meses; que además, se habían realizado de manera grupal. Tampoco se había trabajado en un proyecto con una tecnología desconocida en el momento del inicio que requiriese auto-formación, sino que se habían utilizado siempre formas de trabajo ya conocidas.

De la misma manera, los trabajos realizados previamente tenían un carácter más informal y apenas contaban con una documentación; donde el propio sistema creado era lo que se evaluaba (el producto), y prácticamente no se les daba importancia a otros aspectos como el tiempo dedicado, las pruebas realizadas, la evaluación económica o las referencias bibliográficas. En este caso, sin embargo, se ha realizado un trabajo más formal, y con una documentación elaborada y donde el más mínimo detalle es juzgado; y en el cuál la elaboración de una documentación de calidad tiene una importancia superior al resultado del sistema creado.

9.3. El proyecto previsto y el proyecto real

Mientras que la forma ideal de trabajo y la que se especificó en el EDT (ver 2.5. Planificación temporal) consistía en dedicar un tiempo establecido todos los días y en realizar la documentación mientras se programaba y realizar los diseños antes que los desarrollos, el trabajo real ha sido bien distinto.

En primer lugar, indicar que es una tarea muy complicada planificar un desarrollo de software, y más aún cuando se trabaja de forma individual, en un proyecto tan novedoso y con una tecnología nunca antes utilizada. Los paquetes de trabajo y los tiempos especificados en la planificación temporal se elaboraron desde el desconocimiento y, por tanto, el resultado final fue bien distinto.

Lo cierto es que el proyecto se empieza a conocer en el momento en que se comienza la programación. Hasta entonces, el trabajo realizado en el DOP (Documento de Objetivos del Proyecto) se basa en, como coloquialmente se dice, “dar palos de ciego”. Se planifica un desarrollo que se desconoce, a pesar de intentar documentarse todo lo posible. Se escribe sobre herramientas a utilizar o procedimientos a seguir que en muchos casos, cambian en el momento en que se comienza con el desarrollo.

9.3.1. El tiempo dedicado a la formación

Aparte de la formación realizada previa al comienzo, durante el desarrollo han surgido problemas o dudas que han requerido una mayor indagación en la programación en Angular, como la elaboración de tablas o el tratamiento de los datos asíncronos. Esto ha dado lugar a que el tiempo necesario para la formación haya sido superior al inicialmente previsto.

9.3.2. Las partes del sistema

Una parte que no transcurrió como se había planeado desde un principio eran las partes con las que contaba el sistema. En la planificación se consideraron como equiparables en dificultad un prototipo de centros, otro de inicio de sesión, etc. que posteriormente han sido bien distintos entre sí. Mientras que el inicio de sesión se ha realizado relativamente rápido, algunas partes como los centros o las titulaciones han llevado mucho más tiempo.

De la misma forma, al observar el modelo de dominio, se puede pensar que centros, titulaciones, grupos, docentes… tienen unas características parecidas; y sin embargo difieren bastante en su desarrollo, debido a principalmente la cadena de relaciones que tienen con otros elementos. Los centros, por ejemplo, llevan asociados a multitud de elementos, por lo que han sido una de las áreas más laboriosas.

9.3.3. El código repetido: los algoritmos ya programados

En otro sentido, tampoco todos los prototipos han llevado el mismo tiempo, puesto que el primero en realizarse ha sido mucho más largo que el último, para el cuál ya se disponía de código similar realizado. Por eso, tras el desarrollo se puede ver cómo no era factible asignar el mismo tiempo a todos los prototipos, ni al diseño de los mismo

9.3.4. Funcionalidades surgidas durante el desarrollo

Durante el desarrollo han surgido algunas funcionalidades que en un principio no estaban previstas, como la importación de datos a partir de ficheros CSV, que han implicado más tiempo para el desarrollo.

9.3.5. Riesgos contemplados y producidos

A continuación se analizarán los riesgos que se contemplaron, si se han producido, si los planes de contingencia han funcionado y qué daños se han producido.

Se han dado tres situaciones desfavorables: un borrado accidental de archivos, un desconocimiento de la tecnología utilizada y un fallo en la herramienta AdESMuS.

Borrado de archivos

En el caso del borrado de los archivos, se produjo un borrado manual por accidente. Sin embargo, se restauró la última copia de seguridad, realizada el día anterior, por lo que solamente se perdió el trabajo de un día.

Desconocimiento de la tecnología

En diversas ocasiones se ha producido un desconocimiento de la tecnología Angular. Para ello, se ha pausado el desarrollo, se han analizado diversos manuales para solventar el problema y posteriormente se ha continuado con la implementación. Únicamente se ha parado una semana para poder continuar con la programación.

Fallo de AdESMuS

Este fallo ha sido el más común. En diversas ocasiones, la configuración de la máquina virtual de AdESMuS ha dado problemas, lo que ha ocasionado que no puedan realizarse peticiones al back-end. Para ello, se ha acudido al tutor, que en todas las ocasiones ha conseguido reconfigurar la máquina con éxito. Estos fallos han implicado no poder implementar durante cortos períodos de tiempo.

En la Tabla 60 - Tiempo final dedicado a cada tarea se puede ver la comparativa del tiempo previsto para cada tarea y el finalmente dedicado. Como se puede apreciar, ha sido muy superior al inicialmente previsto, y se ha concentrado mayormente en la implementación y en la elaboración de la documentación.

Tabla 60 - Tiempo final dedicado a cada tarea

TAREA	Número de horas	
	Previstas	**Reales**
1.1: Reunión con el director	2	2
1.2: Búsqueda de información sobre las tecnologías del proyecto	5	0
1.3: Lectura de la documentación del TFG de AdESMuS	5	1
1.4: Análisis y organización del proyecto	5	1
2.1: Planificación de la metodología de trabajo	5	1
2.2: Estudio de las herramientas y técnicas utilizadas	3	1
2.3: Aprendizaje de la tecnología MEAN stack	13	0
2.4: Aprendizaje de la tecnología Angular (con fines de desarrollo)	13	20
2.5: Aprendizaje de la tecnología Angular de cara a la herramienta AdESMuS	9	10
2.6: Instalación del entorno de AdESMuS en el equipo de desarrollo	5	3
2.7: Instalación del software necesario para el desarrollo	5	2
3.1: Diseño del inicio, login y registro	20	5
3.2: Diseño del área de usuarios	20	5
3.3: Diseño del área de asignaturas	20	4
3.4: Diseño del área de centros	20	1
3.5: Diseño del área de departamentos	20	1
3.6: Diseño del área de titulaciones	20	1
3.7: Diseño del área de grupos	20	1
3.8: Diseño del área de matrículas	20	1
4.1: Implementación del inicio, login y registro	25	40
4.2: Implementación del área de usuarios	25	40
4.3: Implementación del área de asignaturas	25	35
4.4: Implementación del área de centros	25	60
4.5: Implementación del área de departamentos	25	30
4.6: Implementación del área de titulaciones	25	40
4.7: Implementación del área de grupos	25	40
4.8: Implementación del área de matrículas	25	45
5.1: Pruebas del inicio, login y registro	4	3
5.2: Pruebas del área de usuarios	4	3
5.3: Pruebas del área de asignaturas	4	2
5.4: Pruebas del área de centros	4	3
5.5: Pruebas del área de departamentos	4	2
5.6: Pruebas del área de titulaciones	4	2
5.7: Pruebas del área de grupos	4	1
5.8: Pruebas del área de matrículas	4	2
5.9: Evaluación de la usabilidad	8	9
6.1: Redacción del DOP	60	70
6.2: Redacción de la memoria	200	250
6.3: Preparación de la presentación de la defensa	30	35
TOTAL	**450**	**772**

9.4. De los errores se aprende: ¿Qué cambiaría si tuviese que volver a empezar?

El trabajo ha sido una gran experiencia personal. No solo por los conocimientos técnicos adquiridos ni el auto-aprendizaje realizado, sino por que el hecho de realizar un proyecto con estas características aporta madurez y aprendizaje. Mientras que las partes resueltas con éxito han aportado motivación, de los errores se ha aprendido y gracias a ellos, si volviese a ser septiembre de 2017, muchas partes se realizarían de forma completamente distinta.

La actitud al comienzo del proyecto era muy distinta que la existente al final. Al comenzar el trabajo, realmente era algo complicado empezar a abordar un trabajo tan grande y tan formal, que estaba previsto desde el comienzo del grado. Al principio, se intentó comenzar el proyecto de una forma similar a los trabajos anteriormente realizados, mientras que según fue transcurriendo el mismo, la forma de trabajo se fue progresivamente adaptando a la que un Trabajo de Fin de Grado requiere.

Si se comenzara de nuevo el proyecto, se habría dedicado más tiempo a la formación inicial en la programación en Angular antes de comenzar con el desarrollo, ya que esto posteriormente habría repercutido en una mayor eficiencia tanto en el diseño como en la implementación de los prototipos.

De la misma manera, se habrían realizado unos diseños más profundos de cada prototipo, dado que se realizaron simples bocetos y posteriormente se comenzó a escribir el código. De haber diseñado adecuadamente cada parte del sistema desde un principio, se habría ahorrado más tiempo en el desarrollo.

Por otro lado, en lo que a la documentación se refiere, en el caso de realizar de nuevo el proyecto se habría documentado de una forma más regular, en lugar de concentrar al final del proyecto la tarea de la documentación. Mientras que en un principio se le dio una importancia mayor al desarrollo, en las últimas semanas se ha priorizado la documentación, por lo que en un nuevo proyecto se haría de una forma más regular.

9.5. Líneas de trabajo futuro

Tras este Trabajo de Fin de Grado, se plantean algunas tareas que surgen al completar este desarrollo.

En cuanto al proyecto desarrollado, tras la finalización quedan algunas partes abiertas a la mejora:

- Debería buscarse una forma de agilizar el funconamiento del sistema y la carga de los datos, realizando modificaciones tanto en el back-end como en el front-end. A menudo la llegada de los datos se ralentiza: en algunos casos, debido a la propia naturaleza de las peticiones y a condiciones ajenas a la programación; pero en otros, debido a que la falta de funcionalidades en el back-end hace necesario que en el front-end se realice un mayor trabajo, que sobrecarga el sistema.
- Aunque tal y como está desarrollado el conjunto funciona, el back-end de AdESMuS debería implementar gran cantidad de funciones que ha sido necesario desarrollar en el front-end. Es decir, trasladar código al back-end.

- Por otro lado, el back-end, además de mejorarse, puede ser ampliado con un modelo de dominio mayor, más relaciones y más funcionalidades que permitan desarrollar un sistema que administre más datos: prácticas en empresa, créditos, prematrículas, tutorías, notas, etc. Una herramienta como AdESMuS puede ser muy interesante para la comunidad universitaria si se amplía, ya que el auge de las tecnologías de la información hace necesario que la tecnología utilizada por la Universidad esté en constante adaptación a nuevos métodos de enseñanza.

- Asimismo, dado que el futuro de la tecnología pasa por todos los dispositivos, una tarea muy interesante sería el desarrollo de AdESMuS en otros dispositivos como los móviles, en forma de aplicación. La UPV/EHU ya cuenta con una aplicación oficial para iOS y Android: Gaur; muy útil para los alumnos, ya que pueden consultar sus calificaciones en cualquier momento y lugar. Por tanto, sería un proyecto muy apropiado el de llevar AdESMuS a los dispositivos móviles, como una evolución del actual Gaur.

Figura 33 - Desarrollo de AdESMuS para dispositivos móviles

Esta última opción es, probablemente, la más interesante, ya que, en caso de darle uso real, sería de gran utilidad para alumnos y profesores; y actualmente existen herramientas para desarrollar aplicaciones para iOS y Android en conjunto, como Xamarin. El proyecto podría centrarse en la aplicación o también en modificar el back-end, que en este caso se alojaría en un servidor remoto.

Anexo I: Casos de uso extendidos

En este anexo se representarán los casos de uso extendidos de cada una de las funciones del front-end.

Iniciar sesión

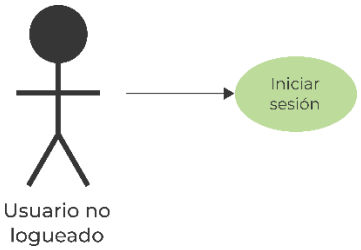
Descripción	Caso de uso que permite realizar un inicio de sesión en el sistema AdESMuS.
Representación	
Actores	Usuario no logueado.
Precondiciones	No estar identificado.
Requisitos no funcionales	Ninguno.
Flujo de eventos	**1**. Se muestran dos campos: el nick o nombre de usuario y la contraseña. El usuario teclea sus datos y pulsa en "ACCEDER" **Si los datos son correctos** **2a**. Se redirige al menú personal del usuario **Si los datos no son correctos** **2b**. Aparece un mensaje indicando la situación para que el usuario rellene de nuevo los campos
Postcondiciones	La sesión está iniciada.

Figura 34 - Interfaz gráfica: Iniciar sesión

Cerrar sesión

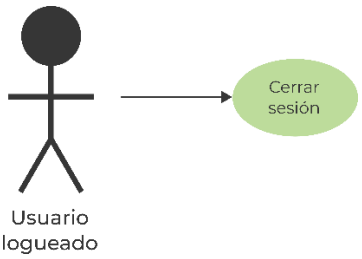
Descripción	Caso de uso que permite cerrar la sesión abierta cuando se ha terminado para garantizar la seguridad.
Representación	
Actores	Usuario logueado.
Precondiciones	Disponer de un token válido.
Requisitos no funcionales	Ninguno.
Flujo de eventos	**1**. El usuario pulsa en el botón de "Cerrar sesión", que aparece en todas las pantallas estando logueado. **2**. Se redirige a la página de inicio de sesión y se borra el token almacenado.
Postcondiciones	No hay ninguna sesión iniciada.

Figura 35 - Interfaz gráfica: Cerrar sesión

Gestionar centros

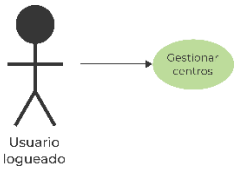
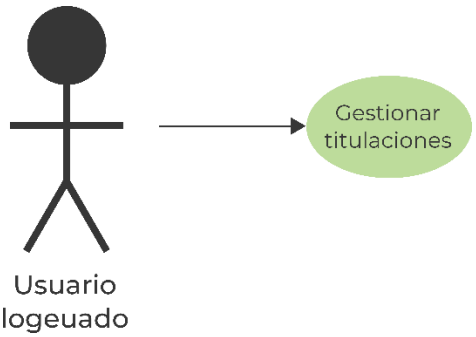
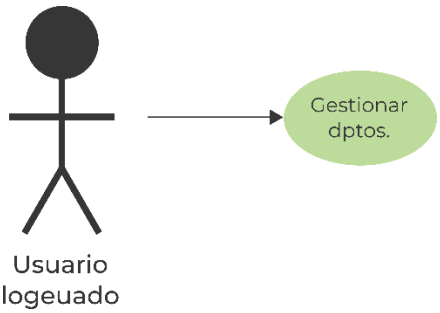
Descripción	Caso de uso que permite realizar todas las operaciones relativas a los centros.
Representación	
Actores	Usuario logueado.
Precondiciones	Disponer de un token válido.
Requisitos no funcionales	Ninguno.
Flujo de eventos	**1**. El usuario pulsa en el botón de "CENTROS" para acceder al área de centros. **Si pulsa sobre el nombre de un centro (subcaso: acceder a centro)** **2a**. Se muestra toda la información de ese centro **Si pulsa en el botón "AÑADIR" (subcaso: añadir un centro)** **2b**. Se abre la página de edición **2c**. El usuario teclea el nombre del centro **2c**. El usuario pulsa en "GUARDAR" **2d**. Se redirige de nuevo al área de centros **Si pulsa en el botón del lápiz de un centro (subcaso: editar centro)** **2e**. Se abre la página de edición **Si modifica el nombre** **2fa**. El usuario teclea el nuevo nombre del centro **2g**. El usuario pulsa en "GUARDAR" **2h**. Se redirige a la información de ese centro **Si pulsa en el botón de la papelera de un centro (subcaso: eliminar centro)** **Si el centro tiene elementos asociados** **2ia**. El sistema informa de la situación al usuario y le pregunta si desea borrar el centro y todos los elementos asociados **Si pulsa "Sí"** **2iba**. Se elimina el centro y todos sus elementos **Si el centro no tiene elementos asociados** **2ic**. El sistema pide la confirmación de la eliminación al usuario **Si pulsa "Sí"** **2ida**. Se elimina el centro
Postcondiciones	Operaciones sobre los centros realizadas.

Figura 36 - Interfaz gráfica: Gestionar centros

Gestionar titulaciones

Descripción	Caso de uso que permite realizar todas las operaciones relativas a las titulaciones.
Representación	
Actores	Usuario logueado.
Precondiciones	Disponer de un token válido.
Requisitos no funcionales	Ninguno.
Flujo de eventos	**1**. El usuario pulsa en el botón de "TITULACIONES" para acceder al área de titulaciones. **Si pulsa sobre el nombre de una titulación (subcaso: acceder a titulación)** **2a**. Se muestra toda la información de esa titulación **Si pulsa en el botón "AÑADIR" (subcaso: añadir una titulación)** **2b**. Se abre la página de edición **2c**. El usuario teclea el nombre de la titulación **2c**. El usuario pulsa en "GUARDAR" **2d**. Se redirige de nuevo al área de titulaciones **Si pulsa en el botón del lápiz de una titulación (subcaso: editar titulación)** **2e**. Se abre la página de edición **Si modifica el nombre** **2fa**. El usuario teclea el nuevo nombre de la titulación **2g**. El usuario pulsa en "GUARDAR" **2h**. Se redirige a la información de esa titulación **Si pulsa en el botón de la papelera de una titulación (subcaso: eliminar titulación)** **Si la titulación tiene elementos asociados** **2ia**. El sistema informa de la situación al usuario y le pregunta si desea borrar la titulación y todos los elementos asociados **Si pulsa "Sí"** **2iba**. Se elimina la titulación y todos sus elementos **Si la titulación no tiene elementos asociados** **2ic**. El sistema pide la confirmación de la eliminación al usuario **Si pulsa "Sí"** **2ida**. Se elimina la titulación
Postcondiciones	Operaciones sobre las titulaciones realizadas.

Gestionar departamentos

Descripción	Caso de uso que permite realizar todas las operaciones relativas a los departamentos.
Representación	
Actores	Usuario logueado.
Precondiciones	Disponer de un token válido.
Requisitos no funcionales	Ninguno.
Flujo de eventos	**1**. El usuario pulsa en el botón de "DEPARTAMENTOS" para acceder al área de departamentos. **Si pulsa sobre el nombre de un departamento (subcaso: acceder a departamento)** **2a**. Se muestra toda la información de ese departamento **Si pulsa en el botón "AÑADIR" (subcaso: añadir un departamento)** **2b**. Se abre la página de edición **2c**. El usuario teclea el nombre del departamento **2c**. El usuario pulsa en "GUARDAR" **2d**. Se redirige de nuevo al área de departamentos **Si pulsa en el botón del lápiz de un departamento (subcaso: editar departamento)** **2e**. Se abre la página de edición **Si modifica el nombre** **2fa**. El usuario teclea el nuevo nombre del departamento **2g**. El usuario pulsa en "GUARDAR" **2h**. Se redirige a la información de ese departamento **Si pulsa en el botón de la papelera de un departamento (subcaso: eliminar departamento)** **Si el departamento tiene elementos asociados** **2ia**. El sistema informa de la situación al usuario y le pregunta si desea borrar el departamento y todos los elementos asociados **Si pulsa "Sí"** **2iba**. Se elimina el departamento y todos sus elementos **Si el departamento no tiene elementos asociados** **2ic**. El sistema pide la confirmación de la eliminación al usuario **Si pulsa "Sí"** **2ida**. Se elimina el departamento
Postcondiciones	Operaciones sobre los departamentos realizadas.

Gestionar asignaturas

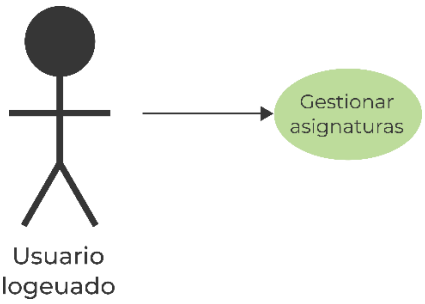
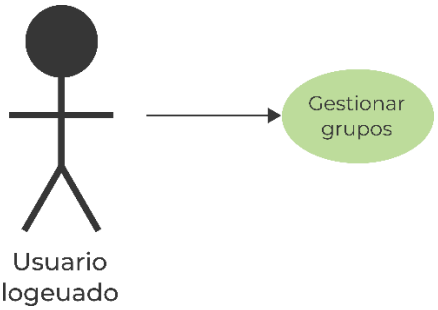
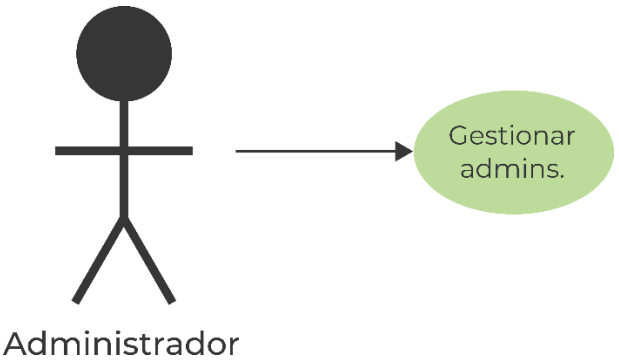
Descripción	Caso de uso que permite realizar todas las operaciones relativas a las asignaturas.
Representación	
Actores	Usuario logueado.
Precondiciones	Disponer de un token válido.
Requisitos no funcionales	Ninguno.
Flujo de eventos	**1**. El usuario pulsa en el botón de "ASIGNATURAS" para acceder al área de asignaturas. **Si pulsa sobre el nombre de una asignatura (subcaso: acceder a asignatura)** **2a**. Se muestra toda la información de esa asignatura **Si pulsa en el botón "AÑADIR" (subcaso: añadir una asignatura)** **2b**. Se abre la página de edición **2c**. El usuario teclea el nombre de la asignatura y selecciona la titulación **2c**. El usuario pulsa en "GUARDAR" **2d**. Se redirige de nuevo al área de asignaturas **Si pulsa en el botón del lápiz de una asignaturas (subcaso: editar asignatura)** **2e**. Se abre la página de edición **Si modifica el nombre** **2fa**. El usuario teclea el nuevo nombre de la asignatura **2g**. El usuario pulsa en "GUARDAR" **2h**. Se redirige a la información de esa asignatura **Si pulsa en el botón de la papelera de una asignatura (subcaso: eliminar asignatura)** **Si la asignatura tiene elementos asociados** **2ia**. El sistema informa de la situación al usuario y le pregunta si desea borrar la asignatura y todos los elementos asociados **Si pulsa "Sí"** **2iba**. Se elimina la asignatura y todos sus elementos **Si la asignatura no tiene elementos asociados** **2ic**. El sistema pide la confirmación de la eliminación al usuario **Si pulsa "Sí"** **2ida**. Se elimina la asignatura
Postcondiciones	Operaciones sobre las asignaturas realizadas.

Figura 37 - Interfaz gráfica: Gestionar asignaturas

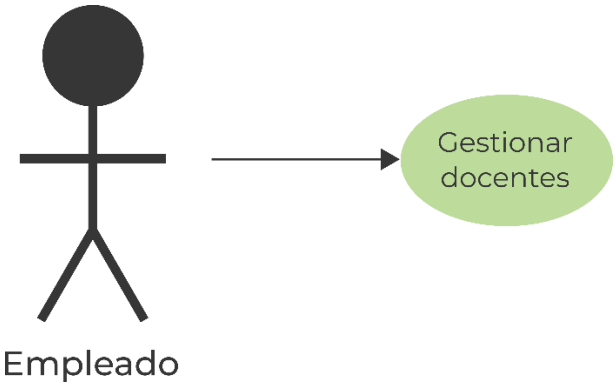
Gestionar grupos

Descripción	Caso de uso que permite realizar todas las operaciones relativas a los grupos.
Representación	
Actores	Usuario logueado.
Precondiciones	Disponer de un token válido.
Requisitos no funcionales	Ninguno.
Flujo de eventos	**1**. El usuario pulsa en el botón de "GRUPOS" para acceder al área de departamentos. **Si pulsa sobre el nombre de un grupo (subcaso: acceder a grupo)** **2a**. Se muestra toda la información de ese grupo **Si pulsa en el botón "AÑADIR" (subcaso: añadir un grupo)** **2b**. Se abre la página de edición **2c**. El usuario teclea el nombre del grupo y selecciona el centro **2c**. El usuario pulsa en "GUARDAR" **2d**. Se redirige de nuevo al área de grupos **Si pulsa en el botón del lápiz de un grupo (subcaso: editar grupo)** **2e**. Se abre la página de edición **Si modifica el nombre** **2fa**. El usuario teclea el nuevo nombre del grupo **2g**. El usuario pulsa en "GUARDAR" **2h**. Se redirige a la información de ese grupo **Si pulsa en el botón de la papelera de un grupo (subcaso: eliminar grupo)** **Si el grupo tiene elementos asociados** **2ia**. El sistema informa de la situación al usuario y le pregunta si desea borrar el grupo y todos los elementos asociados **Si pulsa "Sí"** **2iba**. Se elimina el grupo y todos sus elementos **Si el grupo no tiene elementos asociados** **2ic**. El sistema pide la confirmación de la eliminación al usuario **Si pulsa "Sí"** **2ida**. Se elimina el grupo
Postcondiciones	Operaciones sobre los grupos realizadas.

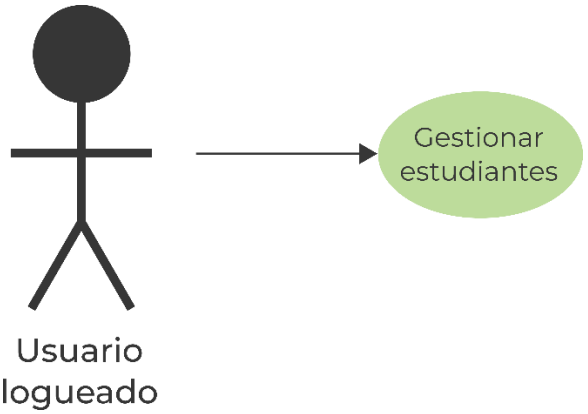
Gestionar administradores

Descripción	Caso de uso que permite realizar todas las operaciones relativas a los administradores.
Representación	
Actores	Administrador.
Precondiciones	Disponer de un token válido.
Requisitos no funcionales	Ninguno.
Flujo de eventos	**1**. El usuario pulsa en el botón de "USUARIOS" para acceder al área de usuarios. **2**. El usuario pulsa en el botón de "Administradores" para acceder al área de administradores. **Si pulsa sobre el nombre de un administrador (subcaso: acceder a administrador)** **3a**. Se muestra toda la información de ese administrador **Si pulsa en el botón "AÑADIR" (subcaso: añadir un administrador)** **3b**. Se abre la página de edición **3c**. El usuario teclea el nombre del administrador y selecciona el administrador **3c**. El usuario pulsa en "GUARDAR" **3d**. Se redirige de nuevo al área de administradores **Si pulsa en el botón del lápiz de un administrador (subcaso: editar administrador)** **3e**. Se abre la página de edición **Si modifica algún dato** **3fa**. El usuario teclea los nuevos datos **3g**. El usuario pulsa en "GUARDAR" **3h**. Se redirige a la información de ese administrador
Postcondiciones	Operaciones sobre los administradores realizadas.

Gestionar docentes

Descripción	Caso de uso que permite realizar todas las operaciones relativas a los docentes.
Representación	
Actores	Empleado.
Precondiciones	Disponer de un token válido.
Requisitos no funcionales	Ninguno.
Flujo de eventos	**1**. El usuario pulsa en el botón de "USUARIOS" para acceder al área de usuarios. **2**. El usuario pulsa en el botón de "Docentes" para acceder al área de docentes. **Si pulsa sobre el nombre de un docente (subcaso: acceder a docente)** **3a**. Se muestra toda la información de ese docente **Si pulsa en el botón "AÑADIR" (subcaso: añadir un docente)** **3b**. Se abre la página de edición **3c**. El usuario teclea el nombre y los apellidos del docente y selecciona el centro-departamento **3c**. El usuario pulsa en "GUARDAR" **3d**. Se redirige de nuevo al área de docentes **Si pulsa en el botón del lápiz de un docente (subcaso: editar docente)** **3e**. Se abre la página de edición **Si modifica algún dato** **3fa**. El usuario teclea los nuevos datos **3g**. El usuario pulsa en "GUARDAR" **3h**. Se redirige a la información de ese docente **Si pulsa en el botón de la papelera de un docente (subcaso: eliminar docente)** **Si el docente tiene elementos asociados** **3ia**. El sistema informa de la situación al usuario y le pregunta si desea borrar el docente y todos los elementos asociados **Si pulsa "Sí"** **3iba**. Se elimina el docente y todos sus elementos **Si el docente no tiene elementos asociados** **3ic**. El sistema pide la confirmación de la eliminación al usuario **Si pulsa "Sí"** **3ida**. Se elimina el docente
Postcondiciones	Operaciones sobre los docentes realizadas.

Gestionar estudiantes

Descripción	Caso de uso que permite realizar todas las operaciones relativas a los docentes.
Representación	
Actores	Usuario logueado.
Precondiciones	Disponer de un token válido.
Requisitos no funcionales	Ninguno.
Flujo de eventos	**1**. El usuario pulsa en el botón de "USUARIOS" para acceder al área de usuarios. **2**. El usuario pulsa en el botón de "Estudiantes" para acceder al área de estudiantes. **Si pulsa sobre el nombre de un estudiante (subcaso: acceder a estudiante)** **3a**. Se muestra toda la información de ese estudiante **Si pulsa en el botón "AÑADIR" (subcaso: añadir un estudiante)** **3b**. Se abre la página de edición **3c**. El usuario teclea el nombre y los apellidos del estudiante y selecciona el centro-titulación de matrícula (si lo desea). **3c**. El usuario pulsa en "GUARDAR" **3d**. Se redirige de nuevo al área de estudiantes **Si pulsa en el botón del lápiz de un estudiante (subcaso: editar estudiante)** **3e**. Se abre la página de edición **Si modifica algún dato** **3fa**. El usuario teclea los nuevos datos **3g**. El usuario pulsa en "GUARDAR" **3h**. Se redirige a la información de ese estudiante **Si pulsa en el botón de la papelera de un estudiante (subcaso: eliminar estudiante)** **Si el estudiante tiene elementos asociados** **3ia**. El sistema informa de la situación al usuario y le pregunta si desea borrar el estudiante y todos los elementos asociados **Si pulsa "Sí"** **3iba**. Se elimina el estudiante y todos sus elementos **Si el estudiante no tiene elementos asociados** **3ic**. El sistema pide la confirmación de la eliminación al usuario **Si pulsa "Sí"** **3ida**. Se elimina el estudiante
Postcondiciones	Operaciones sobre los estudiantes realizadas.

Anexo II: Diagramas de secuencia

En este anexo se representarán los diagramas de secuencia de cada una de las funciones del front-end.

Iniciar sesión

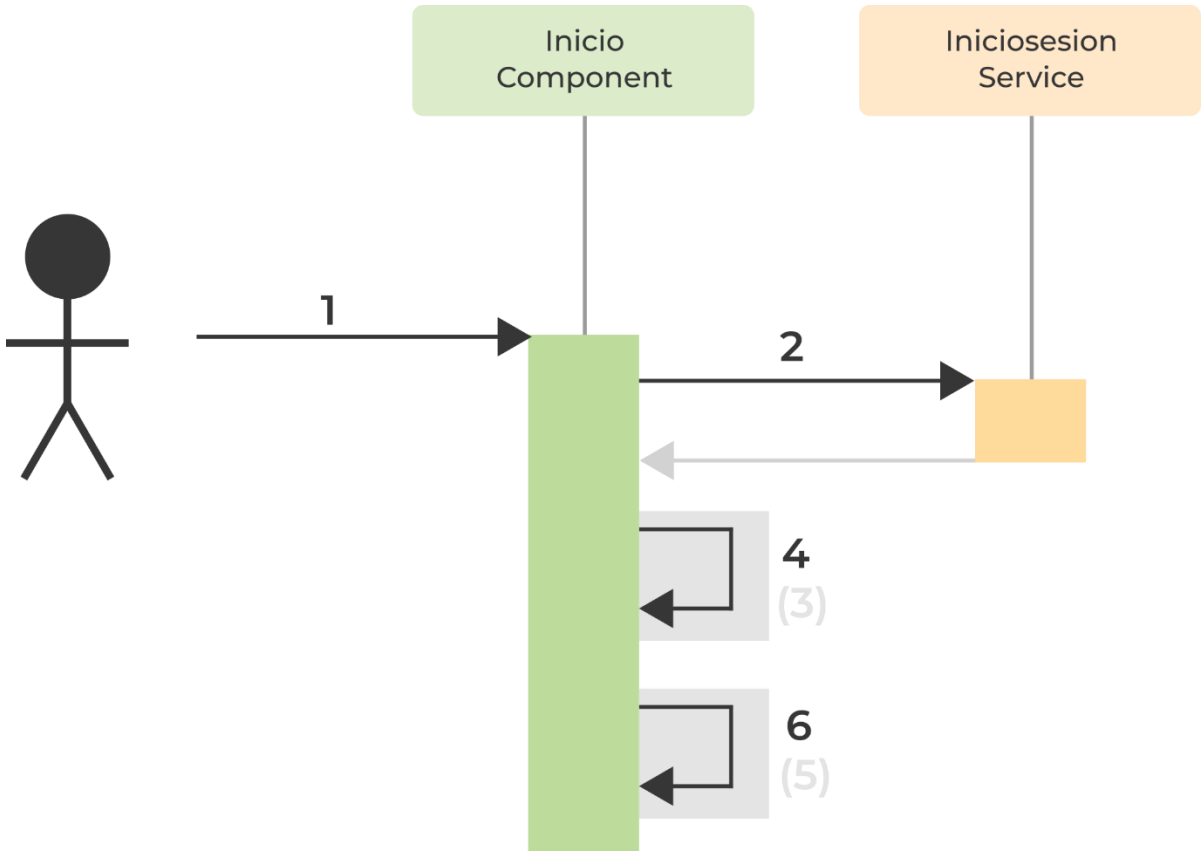


Figura 38 - Diagrama de secuencia: Iniciar sesión

Leyenda

1: iniciarSesion(nick, clave)

2: iniciarSesion(Nick, clave)

3: Si la contraseña es correcta

4: Routing a MenuComponent

5: Si la contraseña no es correcta

6: Mostrar mensaje de error

Cerrar sesión

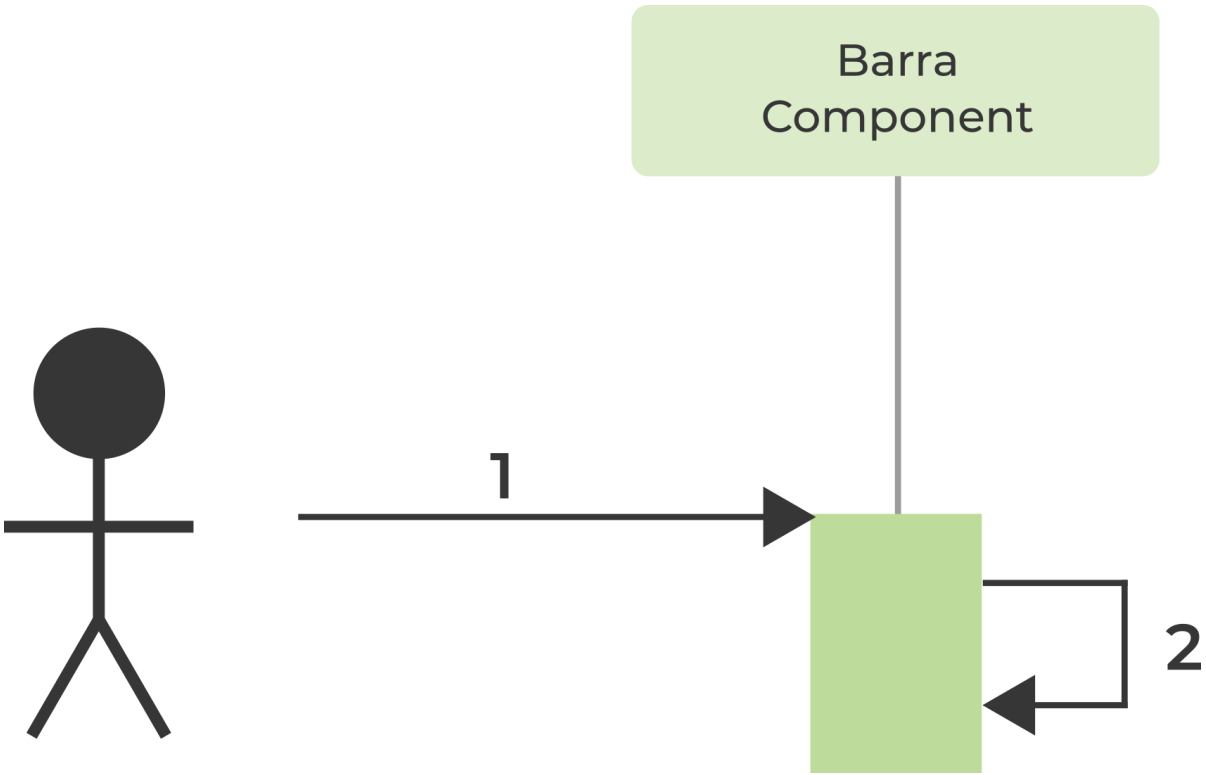


Figura 39 - Diagrama de secuencia: Cerrar sesión

Leyenda

1: cerrarSesion()

2: Borrar token y routing a InicioComponent

Añadir un centro

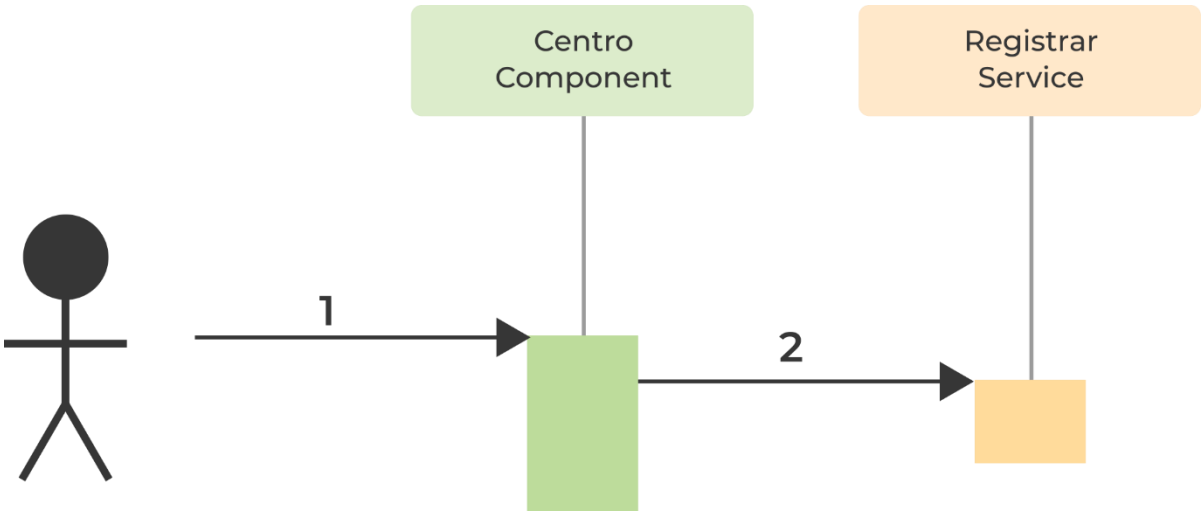


Figura 40 - Diagrama de secuencia: Añadir un centro

Leyenda

1: registrarUnCentro(nombre)

2: registrarUnCentro(nombre)

Acceder a centro

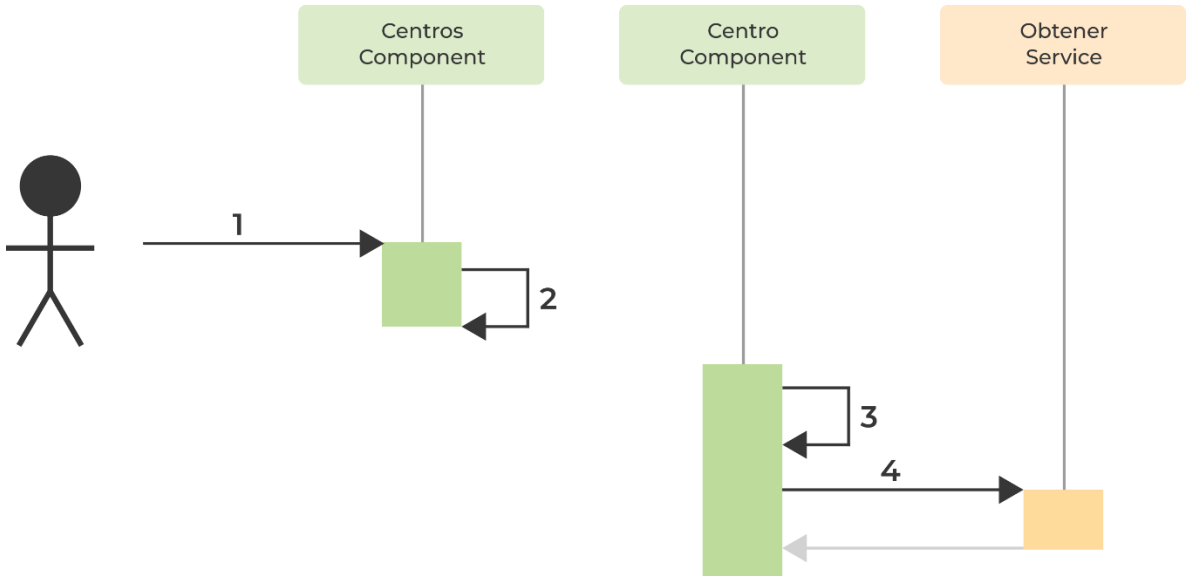


Figura 41 - Diagrama de secuencia: Acceder a centro

Leyenda

1: irACentro(idCentro)

2: Routing a CentroComponent

3: obtenerUnCentro(idCentro)

4: obtenerUnCentro

Editar centro

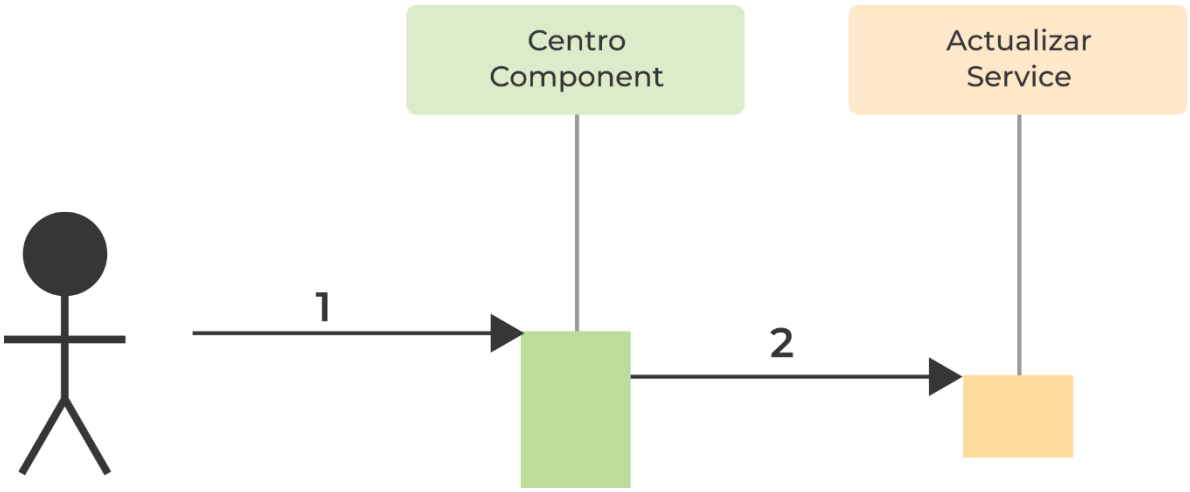


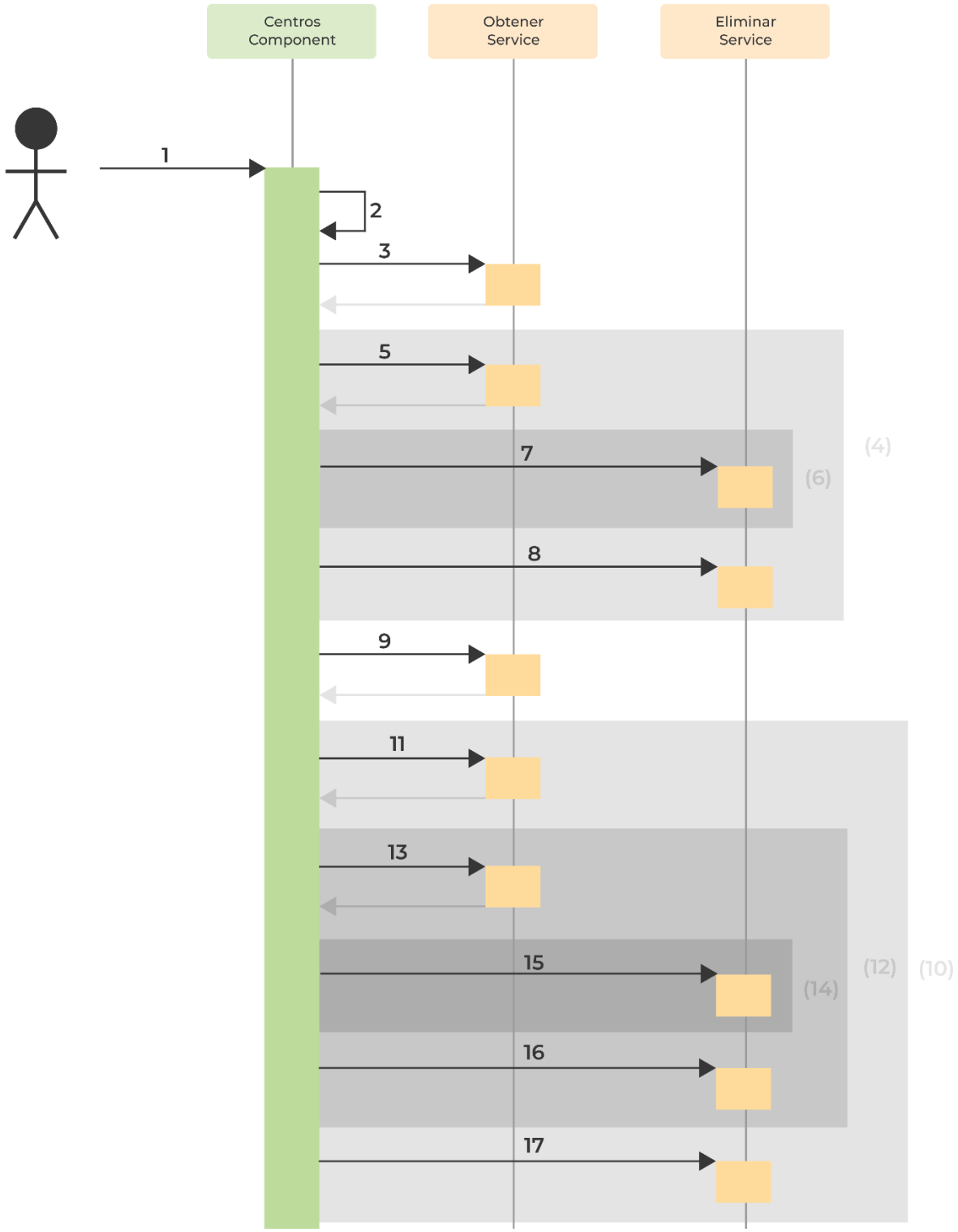
Figura 42 - Diagrama de secuencia: Editar centro

Leyenda

1: modificarDatos(nuevoNombre)

2: actualizarUnCentro(idCentro, nuevoNombre)

Eliminar centro



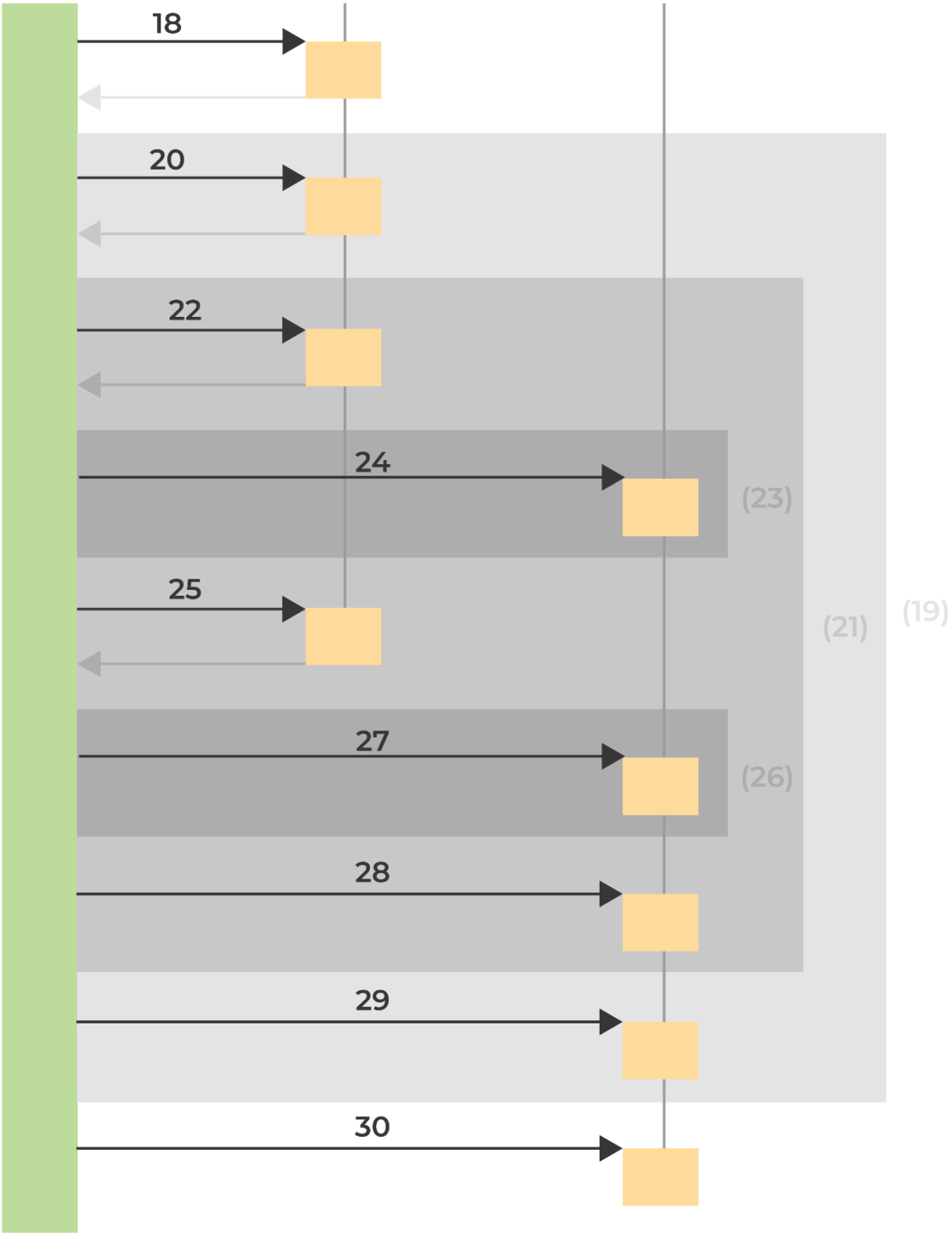


Figura 43 - Diagrama de secuencia: Eliminar centro

Leyenda

1: eliminarUnCentro

2: eliminarDatosAsociadosAUnCentro(idCentro)

3: obtenerTodosLosCentroTitulacionDeUnCentro(idCentro)

4: Para cada centro-titulación

5: obtenerTodasLasMatriculas()

6: Para cada matrícula y si pertenece al centro-titulación

7: eliminarUnaMatricula(idMatricula)

8: eliminarUnCentroTitulacion(idCentro, idTitulacion)

9: obtenerTodosLosCentroDepartamentoDeUnCentro(idCentro)

10: Para cada centro-departamento

11: obtenerTodosLosDocentes()

12: Para cada docente y si pertenece al centro-departamento

13: obtenerTodosGrupoAsignaturaDocenteDeUnDocente(idDocente)

14: Para cada grupo-asignatura-docente

15: eliminarUnGrupoAsignaturaDocente(idGrupo, idAsignatura, idDocente)

16: eliminarUnDocente(idDocente)

17: eliminarUnCentroDepartamento(idCentro, idDepartamento)

18: obtenerTodosLosGrupos()

19: Para cada grupo y si pertenece al centro

20: obtenerTodosLosGrupoAsignaturaDeUnGrupo(idGrupo)

21: Para cada grupo-asignatura

22: obtenerTodosLosGrupoAsignaturaDocente()

23: Para cada grupo-asignatura-docente y si pertenece al grupo-asignatura

24: eliminarUnGrupoAsignaturaDocente(idGrupo, idAsignatura, idDocente)

25: obtenerTodosLosGrupoAsignaturaEstudiante(idGrupo, idAsignatura, idEstudiante)

26: Para cada grupo-asignatura-estudiante y si pertenece al grupo-asignatura

27: eliminarUnGrupoAsignaturaEstudiante(idGrupo, idAsignatura, idEstudiante)

28: eliminarUnGrupoAsignatura(idGrupo, idAsignatura)

29: eliminarUnGrupo(idGrupo)

29: eliminarUnCentro(idCentro)

Añadir una titulación

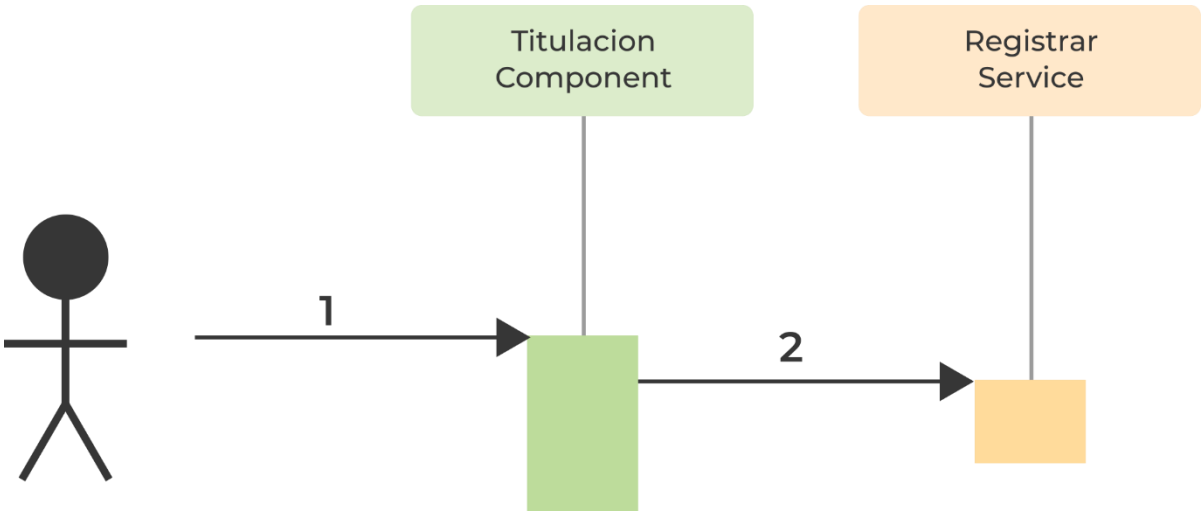


Figura 44 - Diagrama de secuencia: Añadir una titulación

Leyenda

1: registrarUnaTitulacion(nombre)

2: registrarUnaTitulacion(nombre)

Acceder a titulación

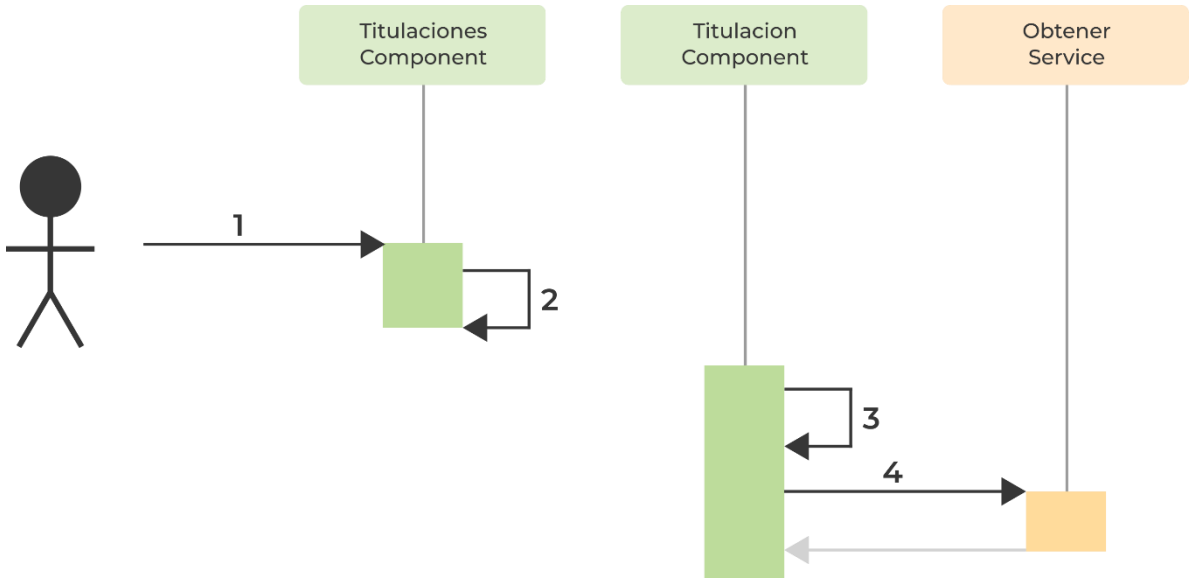


Figura 45 - Diagrama de secuencia: Acceder a titulación

Leyenda

1: irATitulacion(idTitulacion)

2: Routing a TitulacionComponent

3: obtenerUnaTitulacion(idTitulacion)

4: obtenerUnaTitulacion(idTitulacion)

Editar titulación

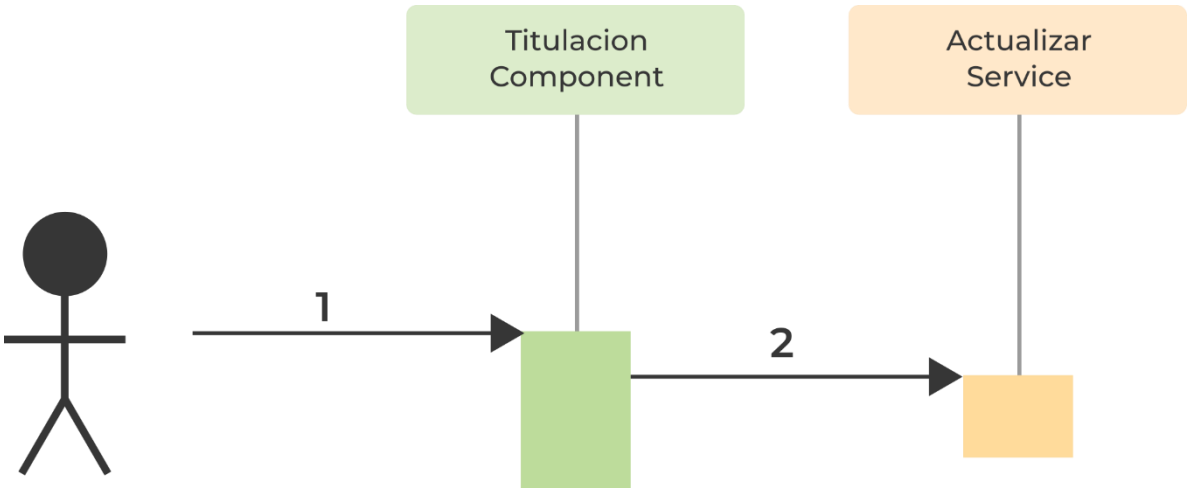


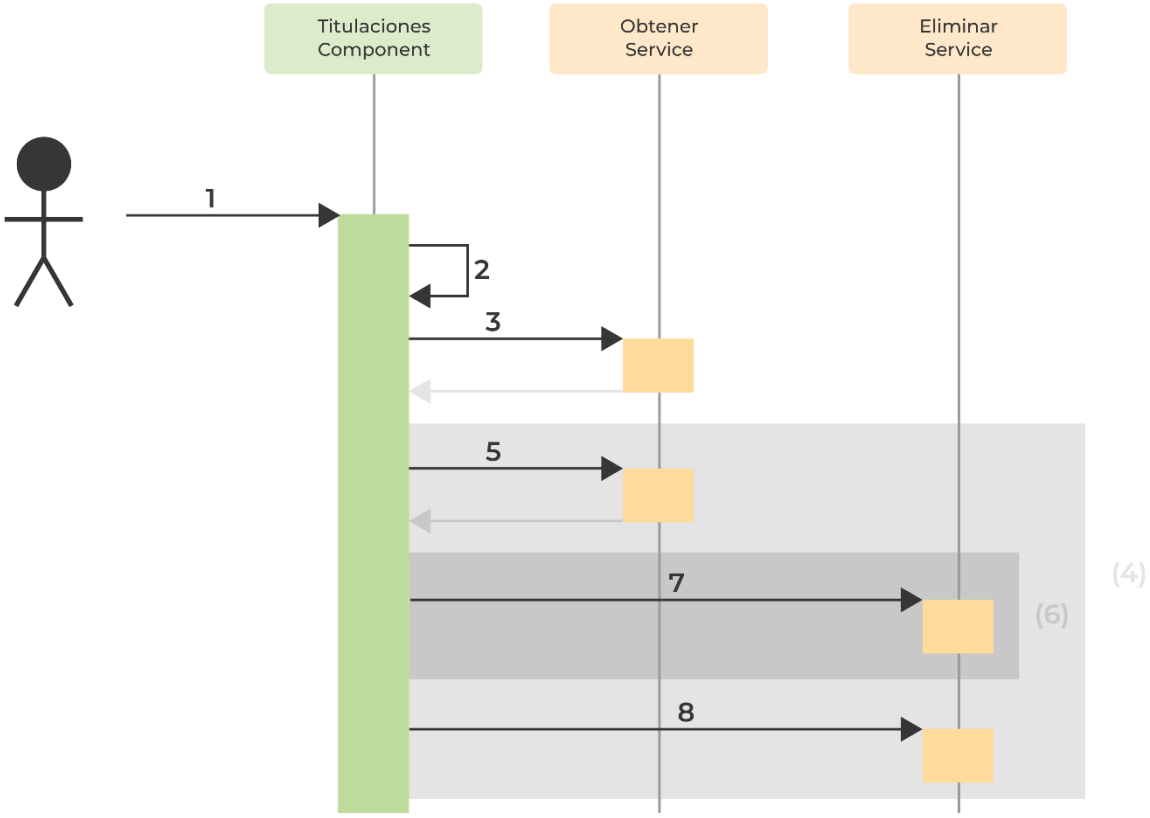
Figura 46 - Diagrama de secuencia: Editar titulación

Leyenda

1: modificarDatos(nuevoNombre)

2: actualizarUnaTitulacion(idTitulacion, nuevoNombre)

Eliminar titulación



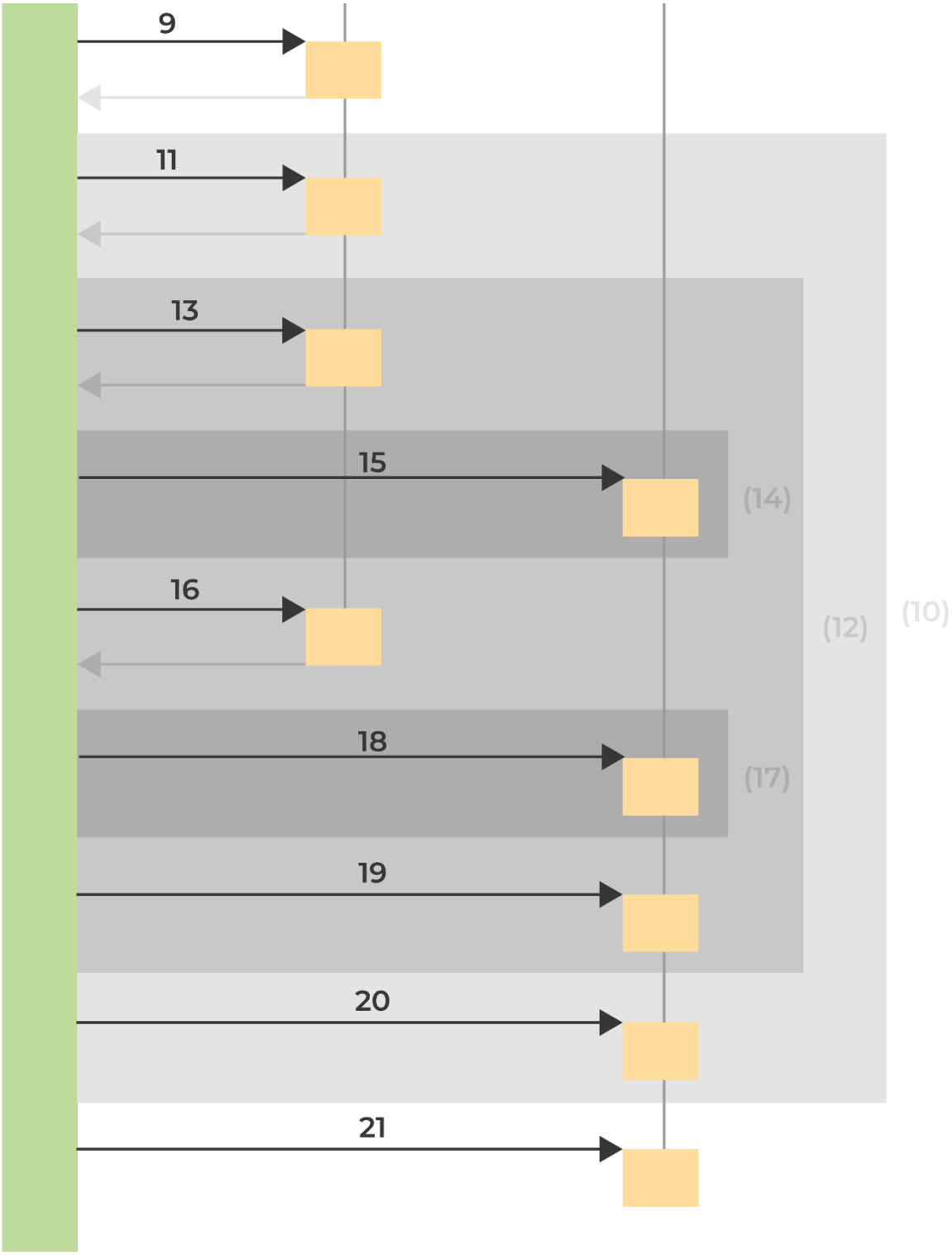


Figura 47 - Diagrama de secuencia: Eliminar titulación

Leyenda

1: eliminarUnaTitulacion(idTitulacion)

2: eliminarDatosAsociadosAUnaTitulacion(idTitulacion)

3: obtenerTodosLosCentroTitulacionDeUnaTitulacion(idTitulacion)

4: Para cada centro-titulación

5: obtenerTodasLasMatriculas()

6: Para cada matrícula y si pertenece al centro-titulación

7: eliminarUnaMatricula(idMatricula)

8: eliminarUnCentroTitulacion(idCentro, idTitulacion)

9: obtenerTodasLasAsignaturas()

10: Para cada asignatura y si pertenece a la titulación

11: obtenerTodosLosGrupoAsignatura()

12: Para cada grupo-asignatura y si pertenece a la asignatura

13: obtenerTodosLosGrupoAsignaturaDocenteDeGrupoAsignatura(idGrupo, idAsignatura)

14: Para cada grupo-asignatura-docente

15: eliminarUnGrupoAsignaturaDocente(idGrupo, idAsignatura, idDocente)

16: obtenerTodosLosGrupoAsignaturaEstudianteDeGrupoAsignatura(idGrupo, idAsignatura)

17: Para cada grupo-asignatura-estudiante

18: eliminarUnGrupoAsignaturaEstudiante(idGrupo, idAsignatura, idEstudiante)

19: eliminarUnGrupoAsignatura(idGrupo, idAsignatura)

20: eliminarUnaAsignatura(idAsignatura)

21: eliminarUnaTitulacion(idTitulacion)

Añadir un departamento

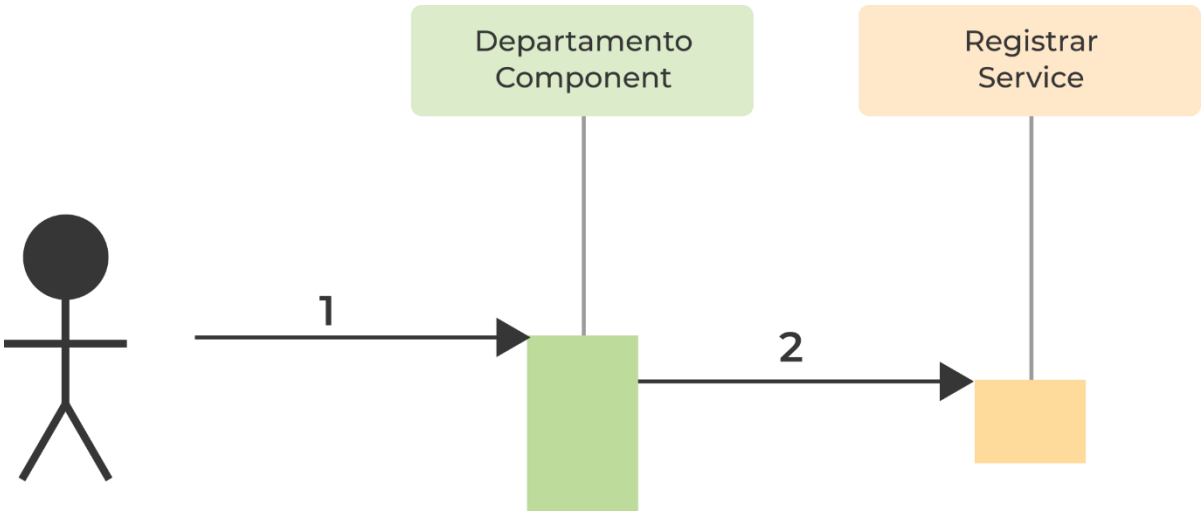


Figura 48 - Diagrama de secuencia: Añadir un departamento

Leyenda

1: registrarUnDepartamento(nombre)

2: registrarUnDepartamento(nombre)

Acceder a departamento

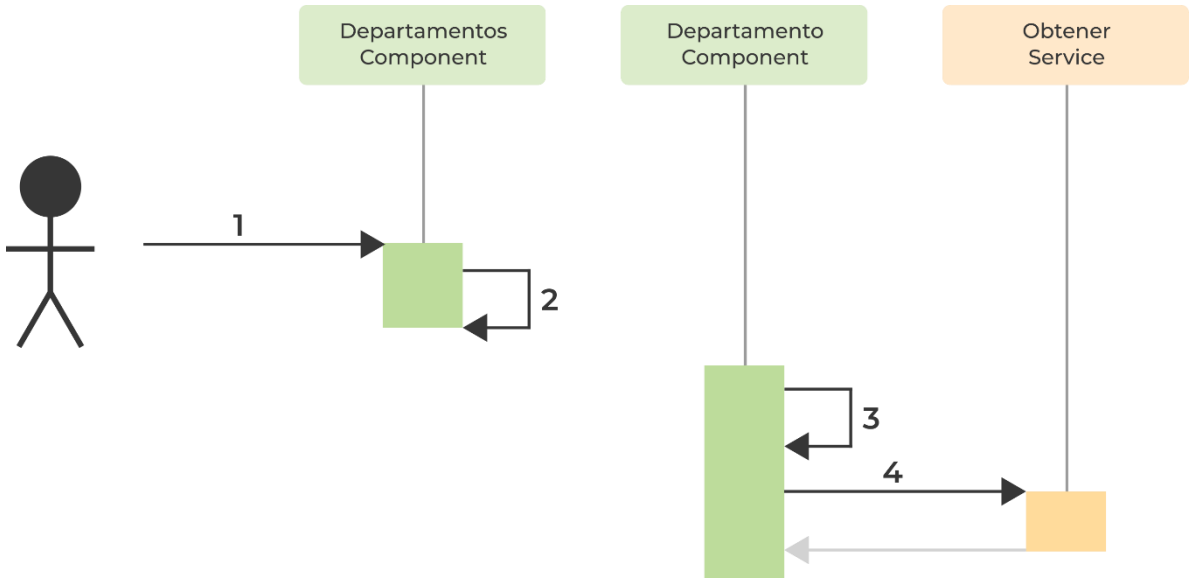


Figura 49 - Diagrama de secuencia: Acceder a departamento

Leyenda

1: irADepartamento(idDepartamento)

2: Routing a DepartamentoComponent

3: obtenerUnDepartamento(idDepartamento)

4: obtenerUnDepartamento(idDepartamento)

Editar departamento

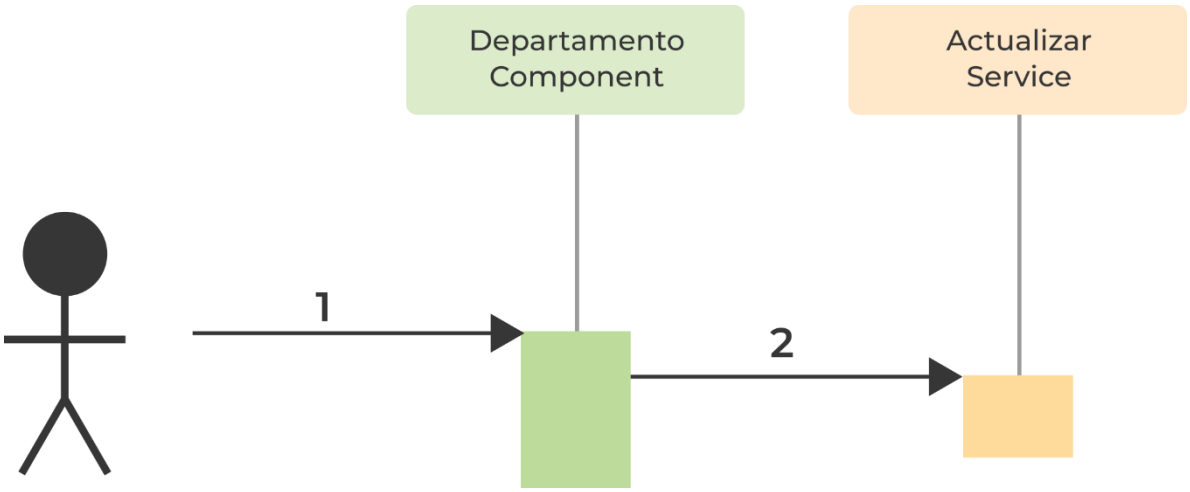


Figura 50 - Diagrama de secuencia: Editar departamento

Leyenda

1: modificarDatos(nuevoNombre)

2: actualizarUnDepartamento(idDepartamento, nuevoNombre)

Eliminar departamento

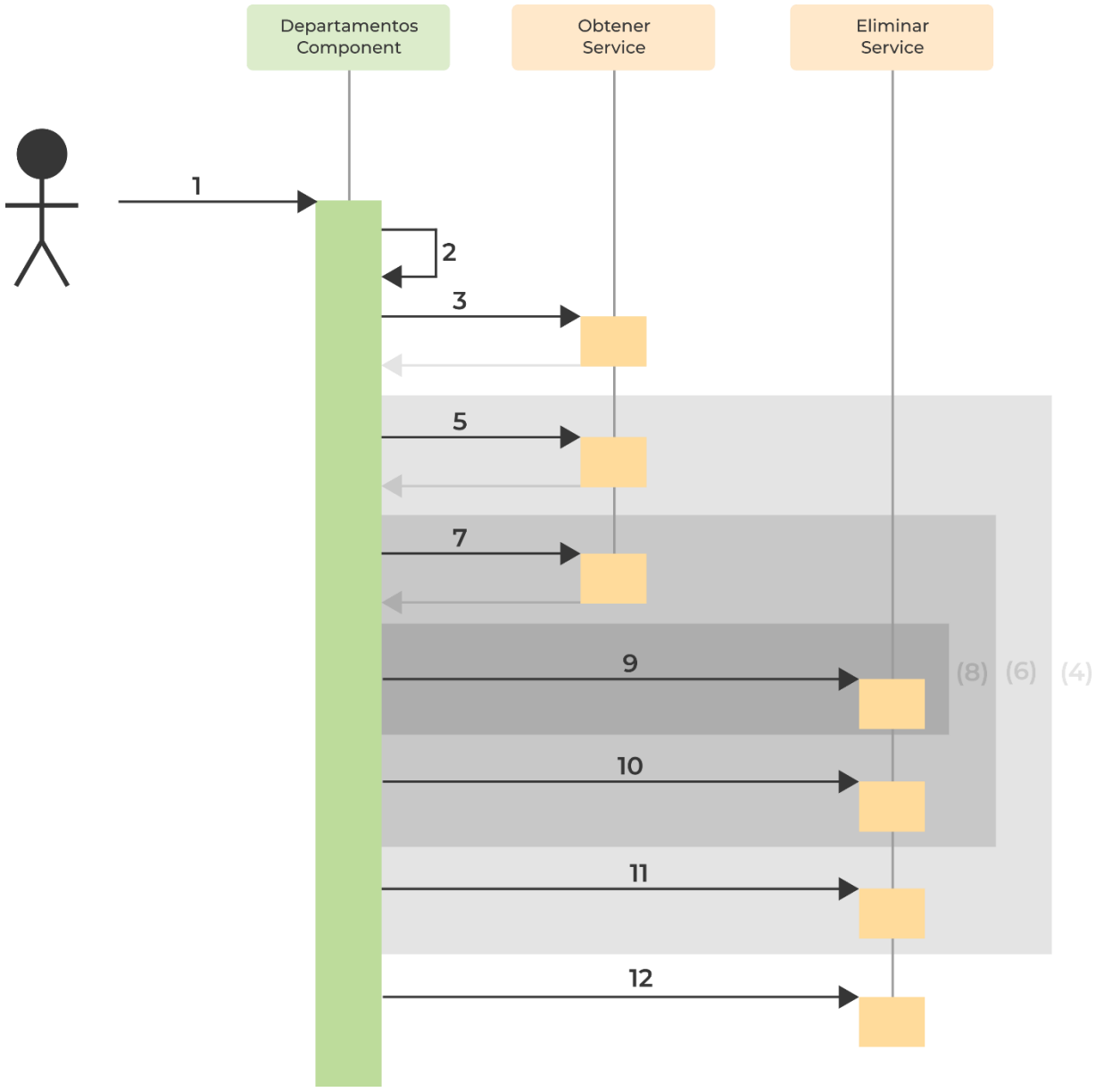


Figura 51 - Diagrama de secuencia: Eliminar departamento

Leyenda

1: eliminarUnDepartamento(idDepartamento)

2: eliminarDatosAsociadosAUnDepartamento(idDepartamento)

3: obtenerTodosLosCentroDepartamentoDeUnDepartamento(idDepartamento)

4: Para cada centro-departamento

5: obtenerTodasLosDocentes()

6: Para cada docente y si pertenece al centro-departamento

7: obtenerTodosLosGrupoAsignaturaDocente()

8: Para cada grupo-asignatura-docente y si pertenece al docente

9: eliminarUnGrupoAsignaturaDocente(idGrupo, idAsignatura, idDocente)

10: eliminarUnDocente(idDocente)

11: eliminarUnCentroDepartamento(idCentro, idDepartamento)

12: eliminarUnDepartamento(idDepartamento)

Añadir una asignatura

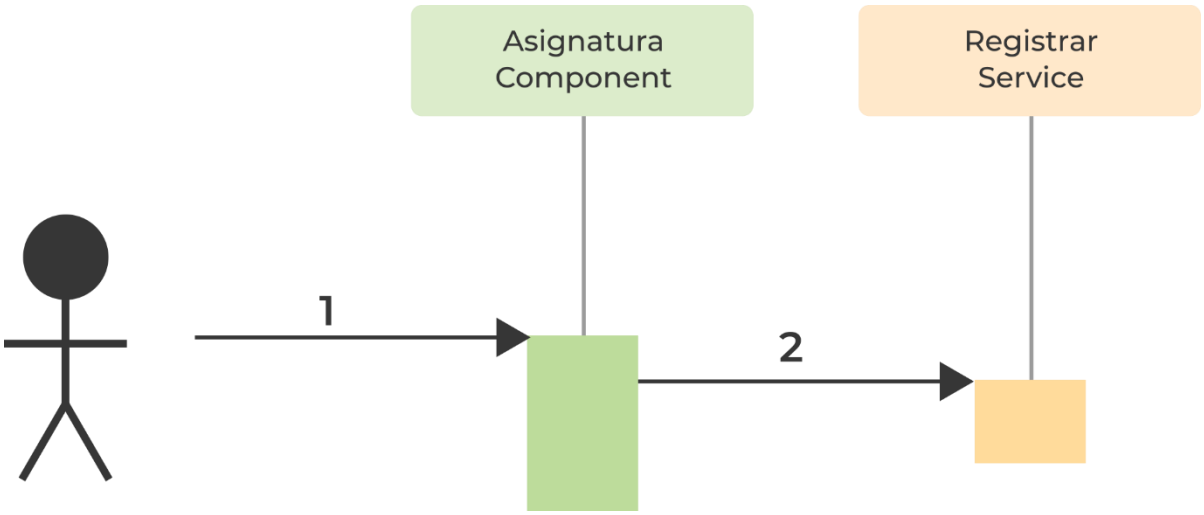


Figura 52 - Diagrama de secuencia: Añadir una asignatura

Leyenda

1: registrarUnaAsignatura(nombre, idTitulacion)

2: registrarUnaAsignatura(nombre, idTitulacion)

Acceder a asignatura

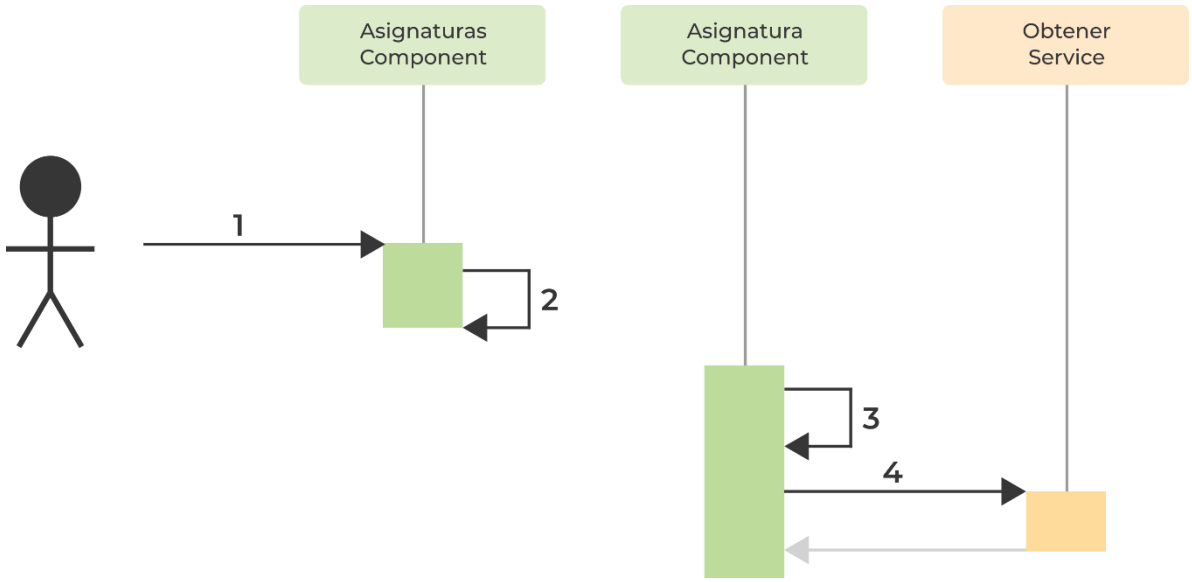


Figura 53 - Diagrama de secuencia: Acceder a asignatura

Leyenda

1: irAAsignatura(idAsignatura)

2: Routing a AsignaturaComponent

3: obtenerUnaAsignatura(idAsignatura)

4: obtenerUnaAsignatura(idAsignatura)

Editar asignatura

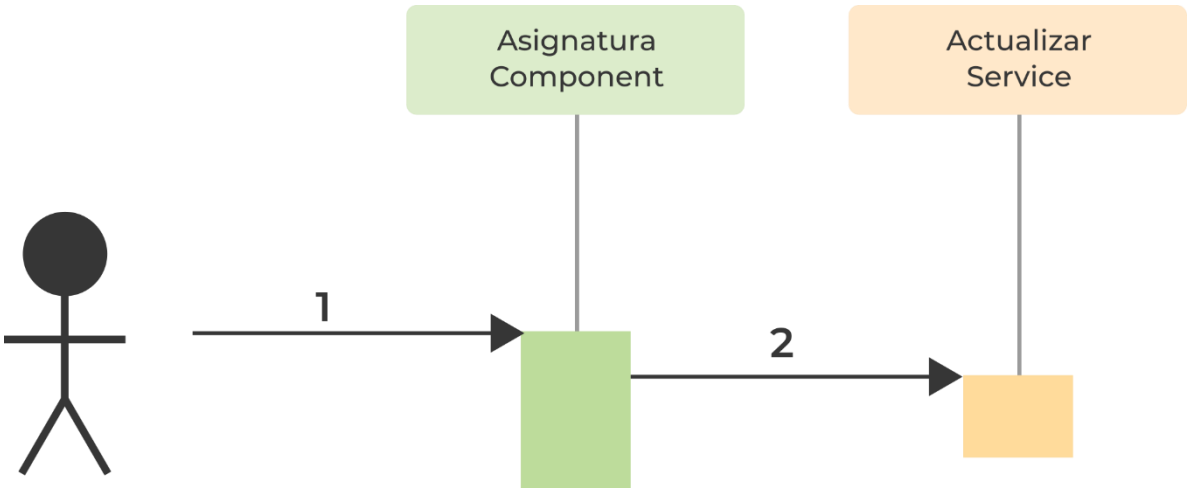


Figura 54 - Diagrama de secuencia: Editar asignatura

Leyenda

1: modificarDatos(nuevoNombre)

2: actualizarUnaAsignatura(idAsignatura, nuevoNombre)

Eliminar asignatura

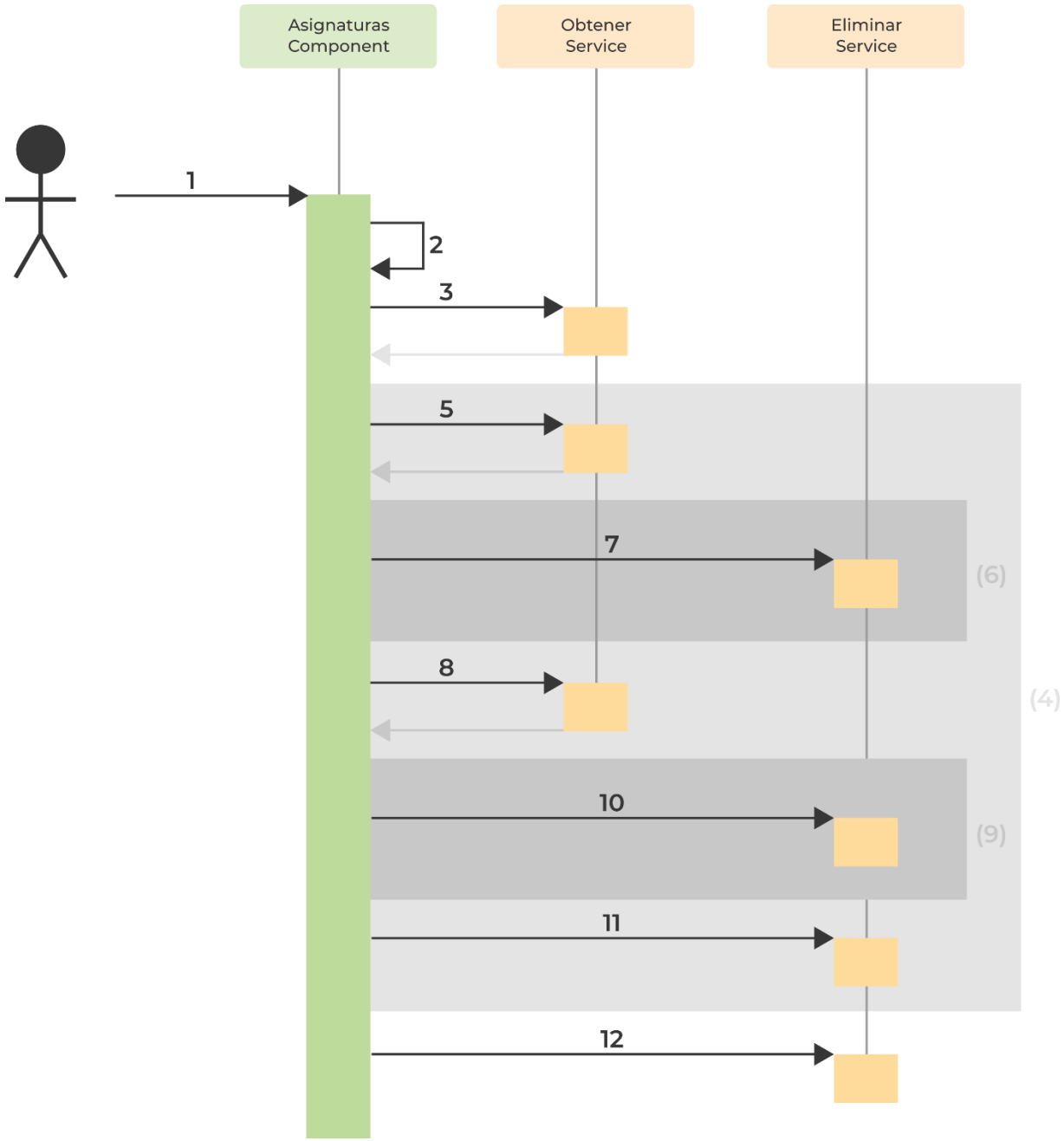


Figura 55 - Diagrama de secuencia: Eliminar asignatura

Leyenda

1: eliminarUnaAsignatura(idAsignatura)

2: eliminarDatosAsociadosAUnaAsignatura(idAsignatura)

3: obtenerTodosLosGrupoAsignatura()

4: Para cada grupo-asignatura y si pertenece a la asignatura

5: obtenerTodosLosGrupoAsignaturaDocenteDeGrupoAsignatura(idGrupo, idAsignatura)

6: Para cada grupo-asignatura-docente

7: eliminarUnGrupoAsignaturaDocente(idGrupo, idAsignatura, idDocente)

8: obtenerTodosLosGrupoAsignaturaEstudianteDeGrupoAsignatura(idGrupo, idAsignatura)

9: Para cada grupo-asignatura-estudiante

10: eliminarUnGrupoAsignaturaEstudiante(idGrupo, idAsignatura, idEstudiante)

11: eliminarUnGrupoAsignatura(idGrupo, idAsignatura)

12: eliminarUnaAsignatura(idAsignatura)

Añadir un grupo

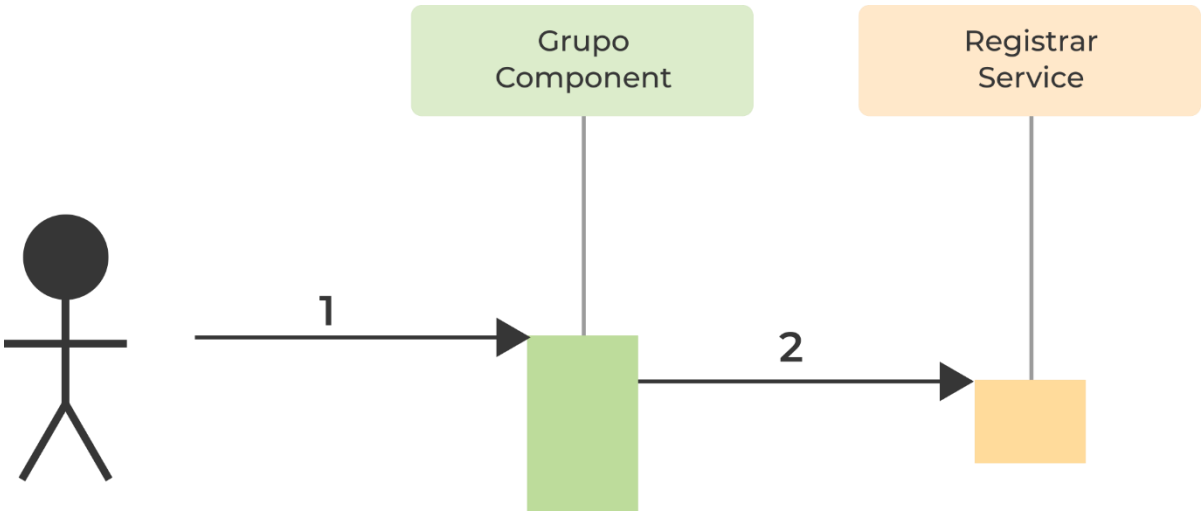


Figura 56 - Diagrama de secuencia: Añadir un grupo

Leyenda

1: registrarUnGrupo(nombre, idCentro)

2: registrarUnGrupo(nombre, idCentro)

Acceder a grupo

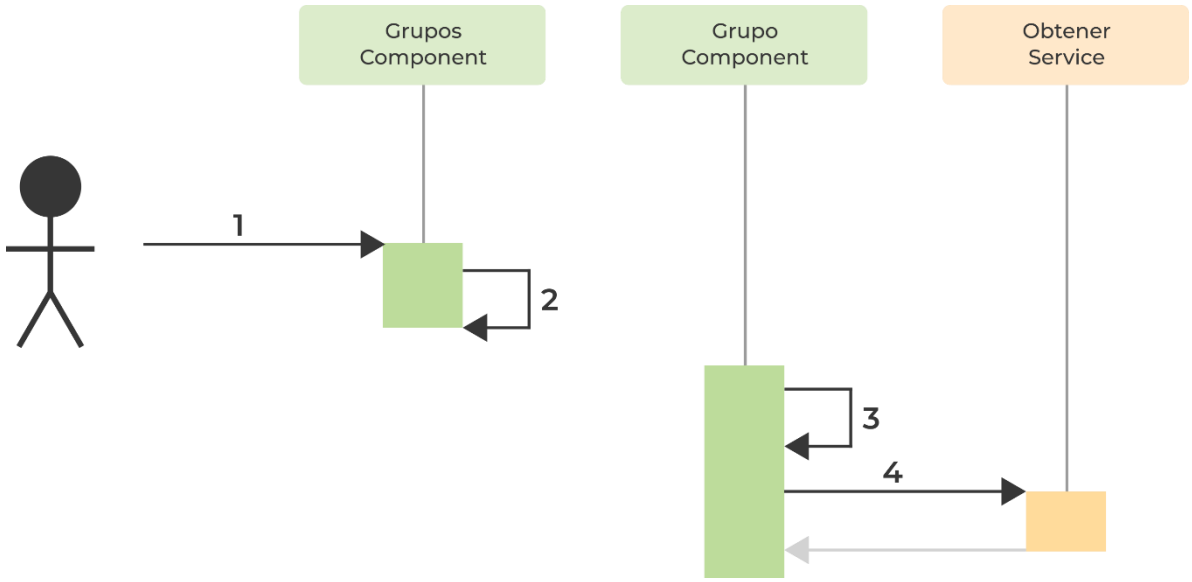


Figura 57 - Diagrama de secuencia: Acceder a grupo

Leyenda

1: irAGrupo(idGrupo)

2: Routing a GrupoComponent

3: obtenerUnGrupo(idGrupo)

4: obtenerUnGrupo(idGrupo)

Editar grupo

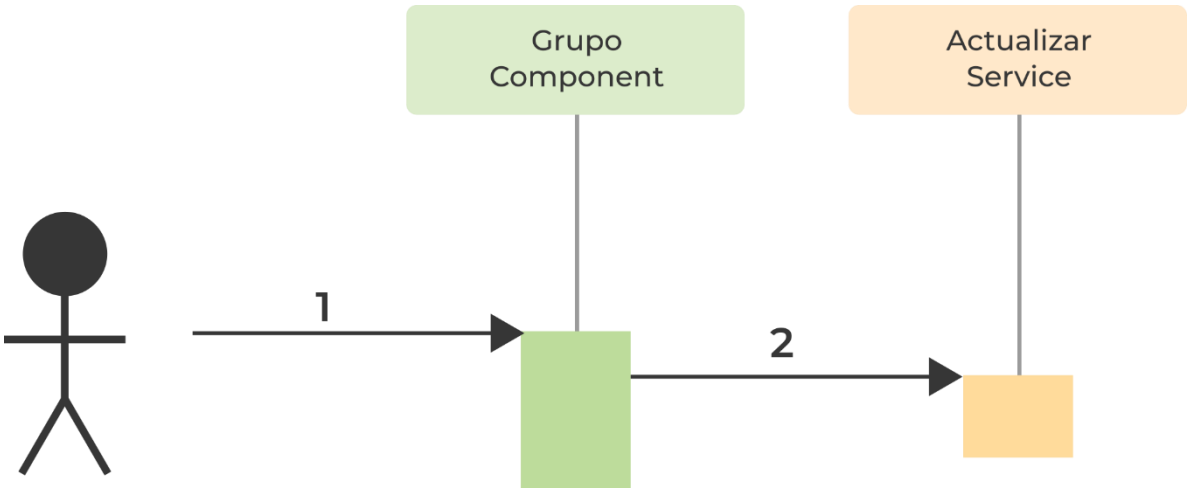


Figura 58 - Diagrama de secuencia: Editar grupo

Leyenda

1: modificarDatos(nuevoNombre)

2: actualizarUnGrupo(idGrupo, nuevoNombre)

Eliminar grupo

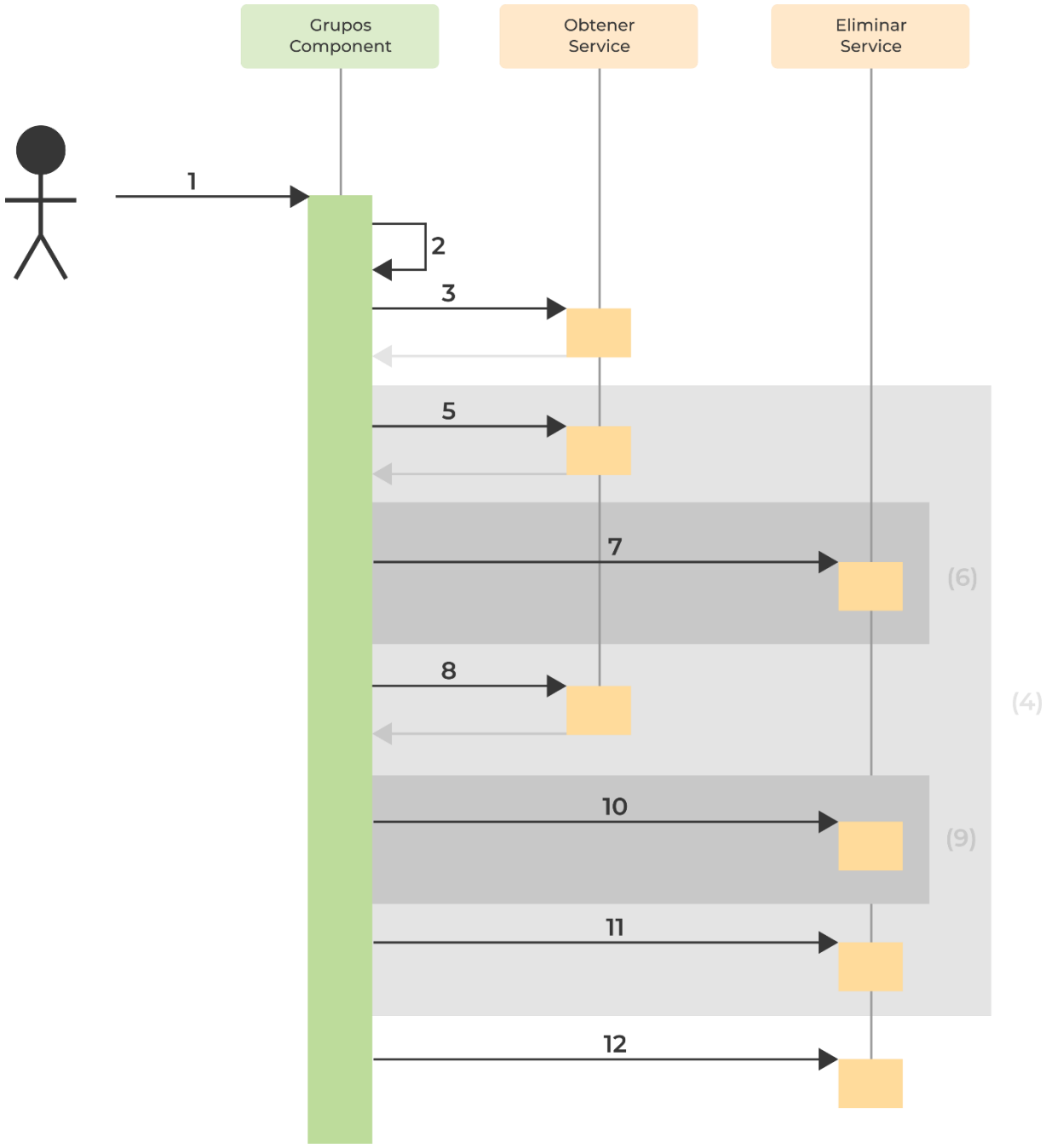


Figura 59 - Diagrama de secuencia: Eliminar grupo

Leyenda

1: eliminarUnGrupo(idGrupo)

2: eliminarDatosAsociadosAUnGrupo(idGrupo)

3: obtenerTodosLosGrupoAsignatura()

4: Para cada grupo-asignatura y si pertenece al grupo

5: obtenerTodosLosGrupoAsignaturaDocenteDeGrupoAsignatura(idGrupo, idAsignatura)

6: Para cada grupo-asignatura-docente

7: eliminarUnGrupoAsignaturaDocente(idGrupo, idAsignatura, idDocente)

8: obtenerTodosLosGrupoAsignaturaEstudianteDeGrupoAsignatura(idGrupo, idAsignatura)

9: Para cada grupo-asignatura-estudiante

10: eliminarUnGrupoAsignaturaEstudiante(idGrupo, idAsignatura, idEstudiante)

11: eliminarUnGrupoAsignatura(idGrupo, idAsignatura)

12: eliminarUnGrupo(idGrupo)

Añadir un administrador

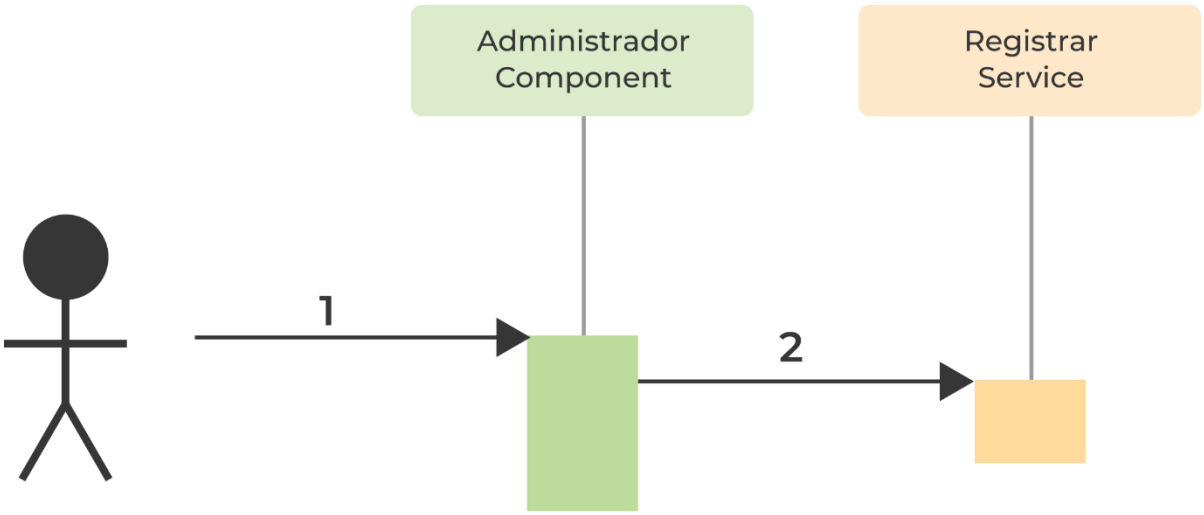


Figura 60 - Diagrama de secuencia: Añadir un administrador

Leyenda

1: registrarUnAdministrador(nick, nombre, apellidos, email, género, contraseña)

2: registrarUnAdministrador(nick, nombre, apellidos, email, género, contraseña)

Acceder a administrador

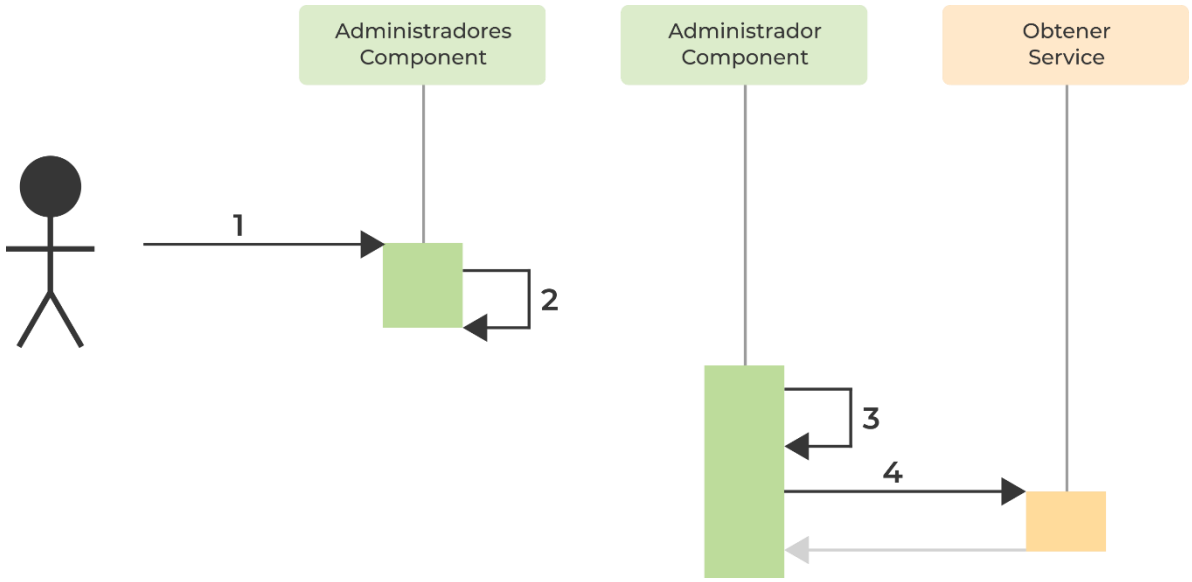


Figura 61 - Diagrama de secuencia: Acceder a administrador

Leyenda

1: irAAdministrador(idAdministrador)

2: Routing a AdministradorComponent

3: obtenerUnAdministrador(idAdministrador)

4: obtenerUnAdministrador

Editar administrador (a sí mismo)

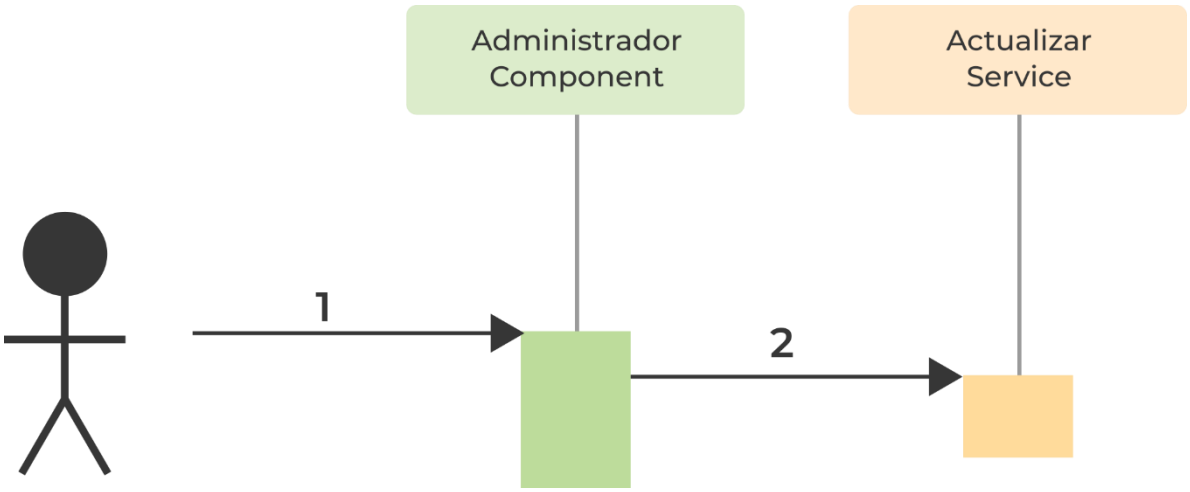


Figura 62 - Diagrama de secuencia: Editar administrador

Leyenda

1: modificarDatos(nuevoNick, nuevoNombre, nuevosApellidos, nuevoEmail, nuevoGénero, nuevaContraseña)

2: actualizarUnAdministrador(idAdministrador, nuevoNick, nuevoNombre, nuevosApellidos, nuevoEmail, nuevoGénero, nuevaContraseña)

Añadir un docente

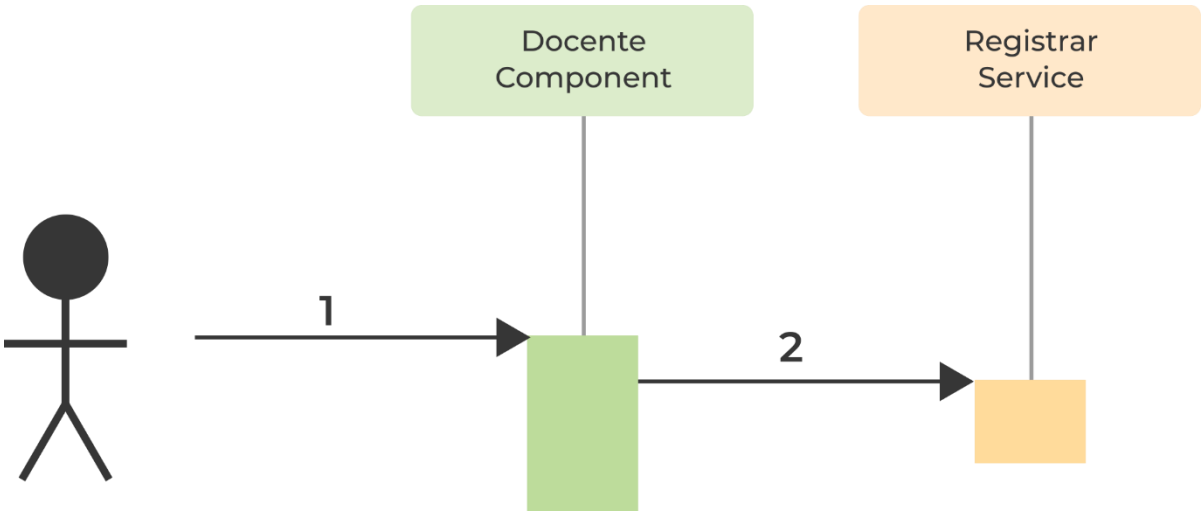


Figura 63 - Diagrama de secuencia: Añadir un docente

Leyenda

1: registrarUnDocente(nick, nombre, apellidos, email, género, contraseña)

2: registrarUnDocente(nick, nombre, apellidos, email, género, contraseña)

Acceder a docente

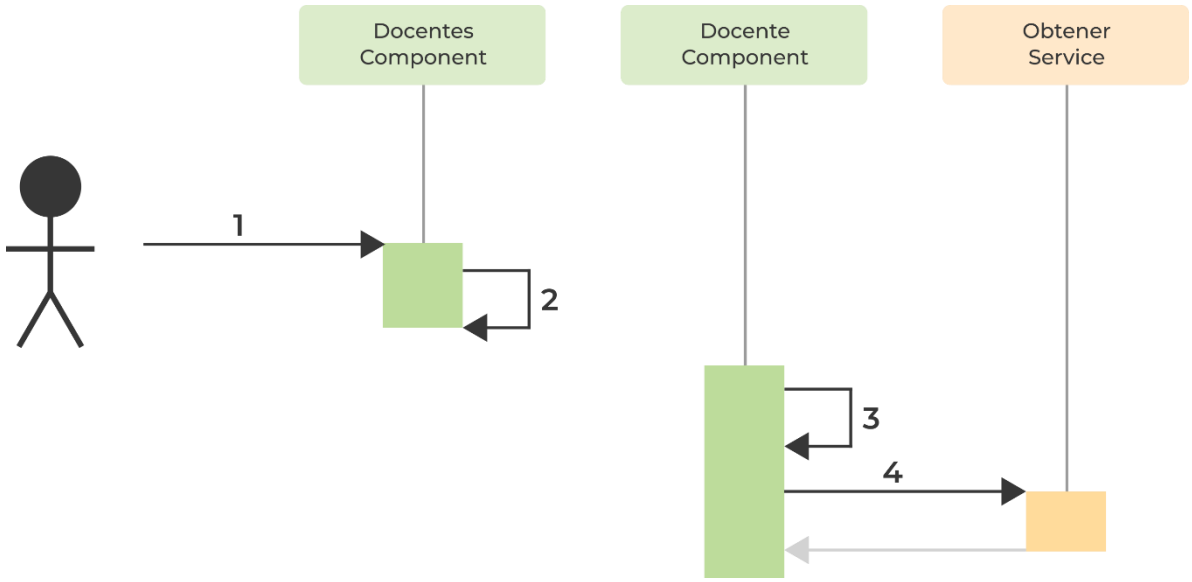


Figura 64 - Diagrama de secuencia: Acceder a docente

Leyenda

1: irADocente(idDocente)

2: Routing a DocenteComponent

3: obtenerUnDocente(idDocente)

4: obtenerUnDocente(idDocente)

Editar docente

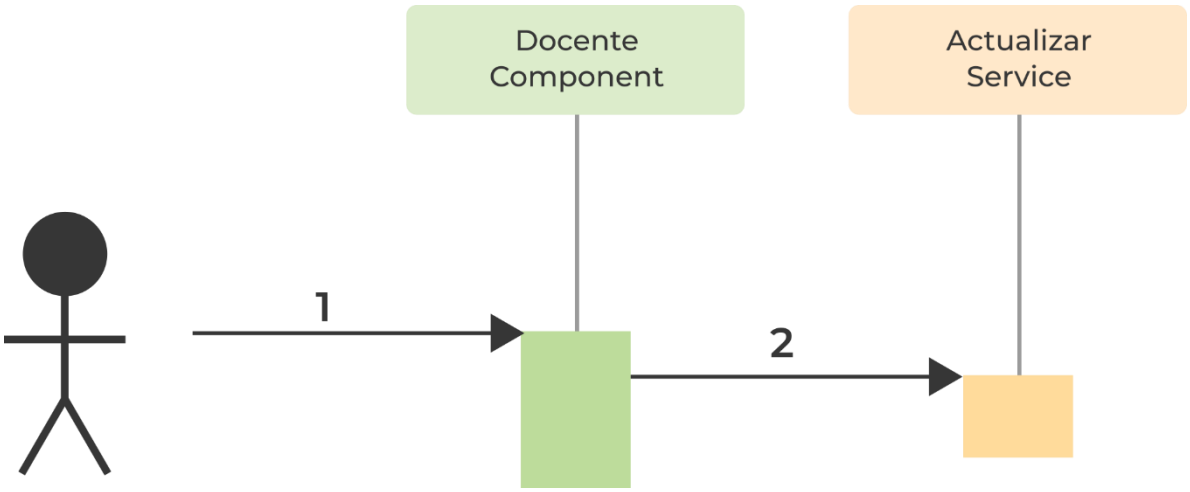


Figura 65 - Diagrama de secuencia: Editar docente

Leyenda

1: modificarDatos(nuevoNick, nuevoNombre, nuevosApellidos, nuevoEmail, nuevoGénero, nuevaContraseña)

2: actualizarUnDocente(idDocente, nuevoNick, nuevoNombre, nuevosApellidos, nuevoEmail, nuevoGénero, nuevaContraseña)

Eliminar docente

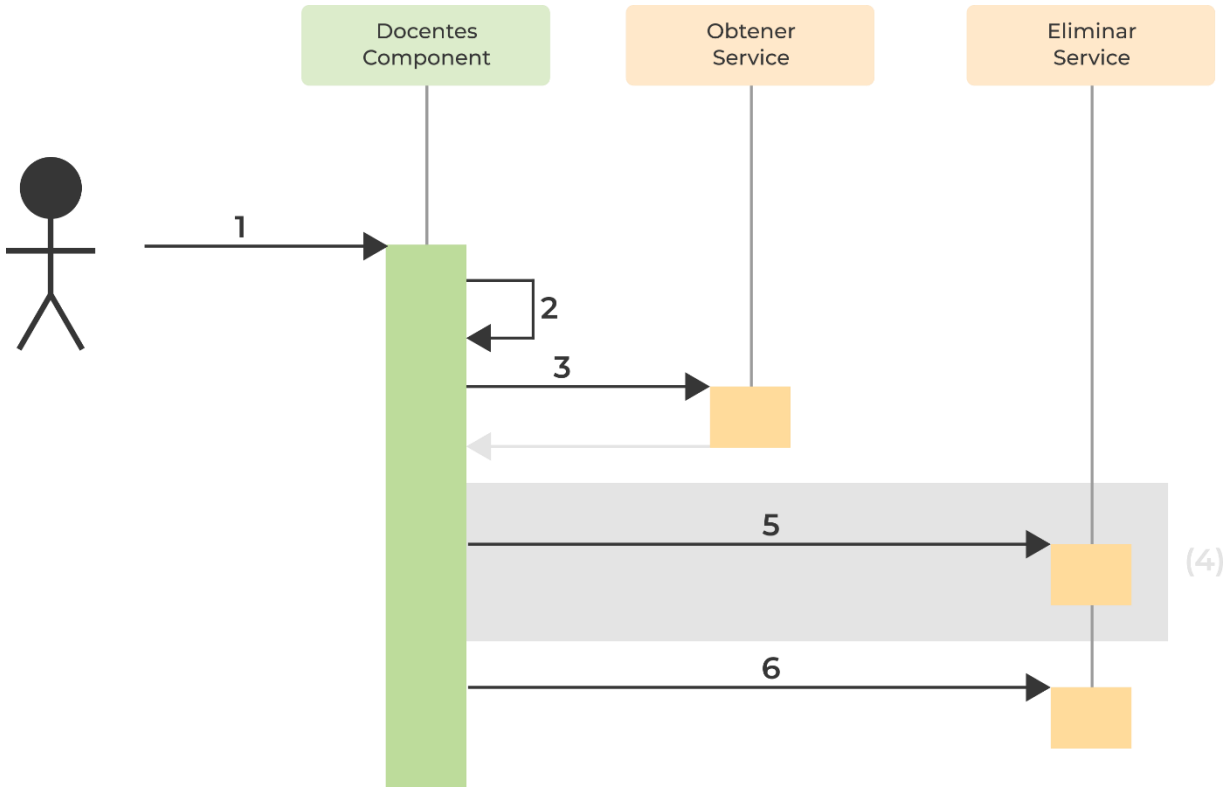


Figura 66 - Diagrama de secuencia: Eliminar docente

Leyenda

1: eliminarUnDocente(idDocente)

2: eliminarDatosAsociadosAUnDocente(idDocente)

3: obtenerTodosLosGrupoAsignaturaDocenteDeUnDocente(idDocente)

4: Para cada grupo-asignatura-docente

5: eliminarUnGrupoAsignaturaDocente(idGrupo, idAsignatura, idDocente)

6: eliminarUnDocente(idDocente)

Añadir un estudiante

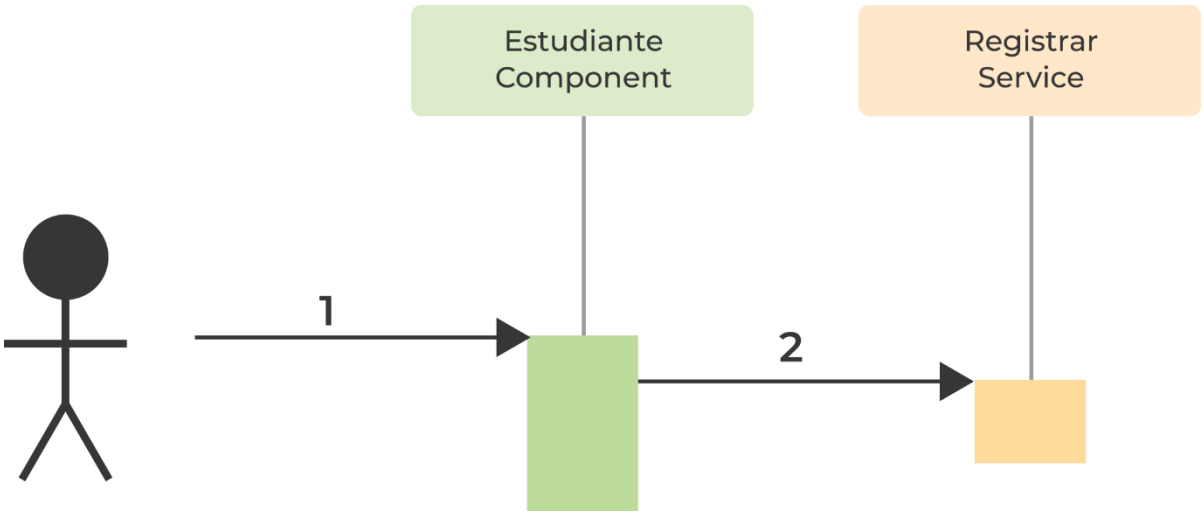


Figura 67 - Diagrama de secuencia: Añadir un estudiante

Leyenda

1: registrarUnEstudiante(nick, nombre, apellidos, email, género, contraseña)

2: registrarUnEstudiante(nick, nombre, apellidos, email, género, contraseña)

Acceder a estudiante

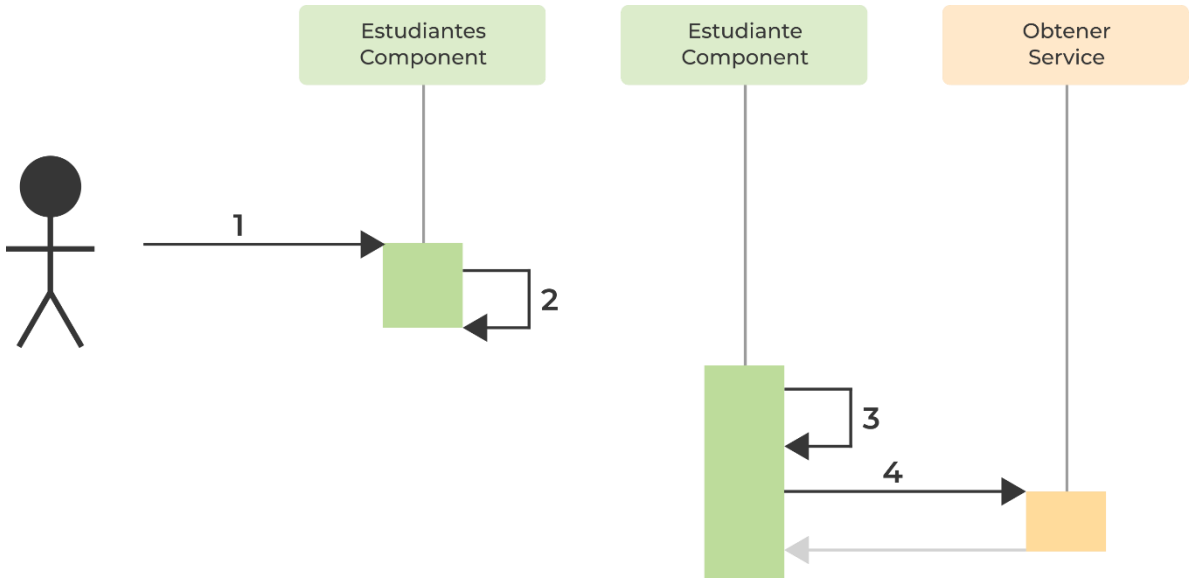


Figura 68 - Diagrama de secuencia: Acceder a estudiante

Leyenda

1: irAEstudiante(idEstudiante)

2: Routing a EstudianteComponent

3: obtenerUnEstudiante(idEstudiante)

4: obtenerUnEstudiante(idEstudiante)

Editar estudiante

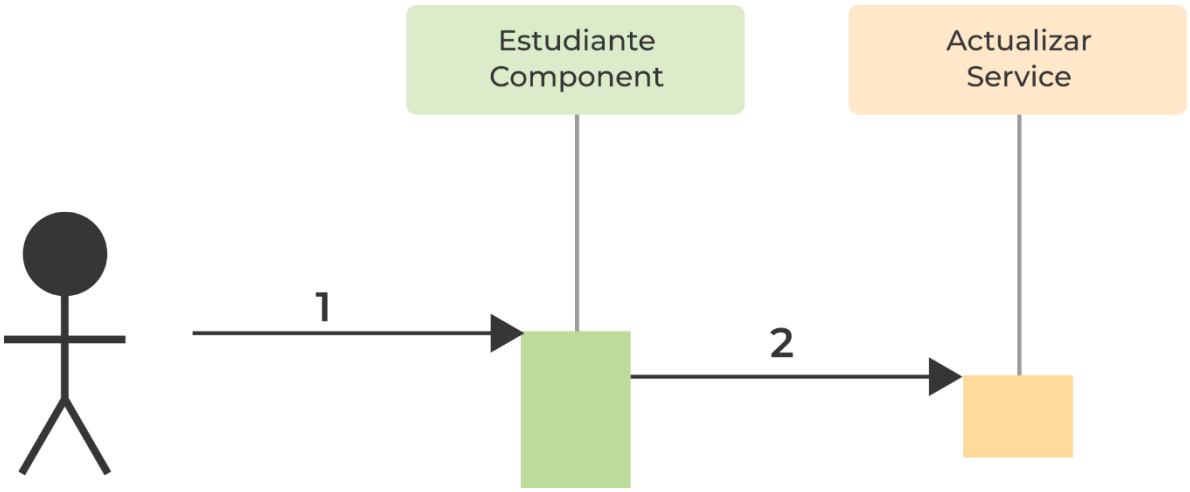


Figura 69 - Diagrama de secuencia: Editar estudiante

Leyenda

1: modificarDatos(nuevoNick, nuevoNombre, nuevosApellidos, nuevoEmail, nuevoGénero, nuevaContraseña)

2: actualizarUnEstudiante(idDocente, nuevoNick, nuevoNombre, nuevosApellidos, nuevoEmail, nuevoGénero, nuevaContraseña)

Eliminar estudiante

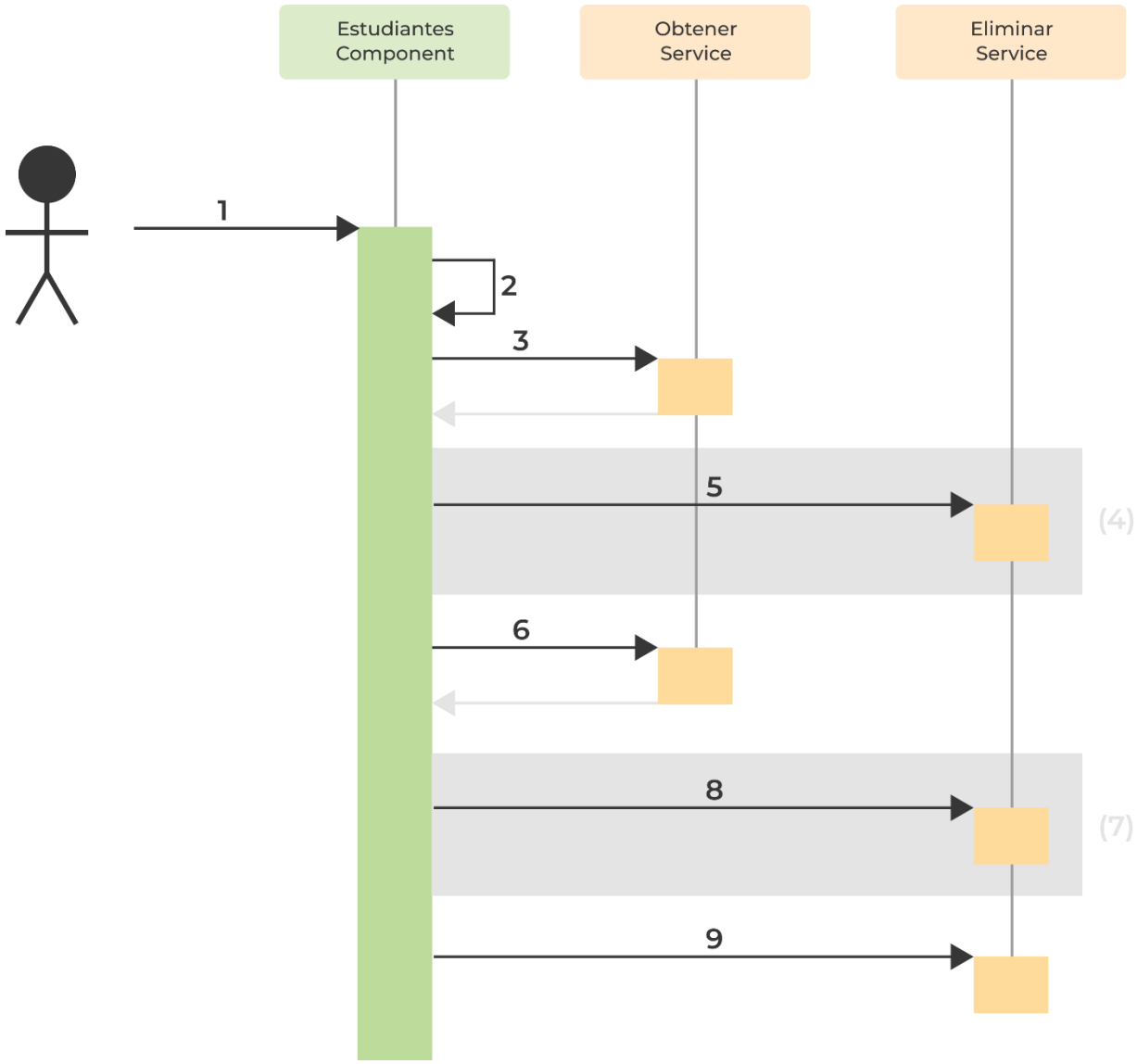


Figura 70 - Diagrama de secuencia: Eliminar estudiante

Leyenda

1: eliminarUnEstudiante(idEstudiante)

2: eliminarDatosAsociadosAUnEstudiante(idEstudiante)

3: obtenerTodosLosGrupoAsignaturaEstudianteDeUnEstudiante(idEstudiante)

4: Para cada grupo-asignatura-estudiante

5: eliminarUnGrupoAsignaturaEstudiante(idGrupo, idAsignatura, idEstudiante)

6: obtenerTodasLasMatriculas()

7: Para cada matrícula y si pertenece al estudiante

8: eliminarUnaMatricula(idMatricula)

9: eliminarUnEstudiante(idEstudiante)

Bibliografía

[1] Boletín Oficial del Estado, número 15, sección 3, página 4536. Gobierno de España. *Ministerio de Empleo y Seguridad Social*. boe.es/boe/dias/2017/01/18/pdfs/BOE-A-2017-542.pdf

[2] Lenovo Essential G500. *PcComponentes*. pccomponentes.com/lenovo-essential-g500-i7-3632qm-4gb-500gb-hd8570m-15-6-

[3] Windows 10 Home. *Microsoft*. microsoft.com/es-es/store/d/Windows-10-Home/D76QX4BZNWK4/1NT3

[4] Lenovo IdeaPad G500. *Amazon*. amazon.es/Lenovo-IdeaPad-G500-Portatil-Graphics/dp/B00ISQ8BNQ

[5] Precio del kWh en España. *tarifasgasluz*. tarifasgasluz.com/faq/precio-kwh/espana

[6] AdESMuS. Memoria. *Fco. Javier García Carvajal*. addi.ehu.es/bitstream/handle/10810/19673/Memoria.pdf

[7] Mandel Golden Rules (Golden Rules of User Interface Design). *Nick Babich*. uxplanet.org/golden-rules-of-user-interface-design-19282aeb06b

[8] RFC 4180 - Common Format and MIME Type for Comma-Separated Values (CSV) Files. tools.ietf.org/html/rfc4180

[9] Codificación de caracteres: Conceptos básicos. w3.org/International/articles/definitions-characters/index.es

[10] ISO/IEC 9241. iso.org/standard/52075.html

[11] AngularJS y el futuro del desarrollo web. *Jorge Álvarez*. alvareznavarro.es/desarrollo-web/2012/10/angularjs-y-el-futuro-del-desarrollo-web/

[12] Las 5 principales ventajas de usar Angular para crear aplicaciones web. campusmvp.es/recursos/post/las-5-principales-ventajas-de-usar-angular-para-crear-aplicaciones-web.aspx

[13] Manual de Angular. desarrolloweb.com/manuales/manual-angular-2.html