



**Facultad de Ingeniería**  
Universidad de Deusto

**Ingeniaritza Fakultatea**  
Deustuko Unibertsitatea

## **Grado en Ingeniería Informática Informatikako Ingeniaritzako Gradua**

### **Proyecto fin de grado Gradu amaierako proiekta**

Desarrollo de una aplicación web basada en una arquitectura de microservicios que permita la posterior implantación de una plataforma de experimentación para algoritmos matemáticos y de inteligencia artificial

Alexander Moreno Lobato

Director: Iker Pastor López

Bilbao, julio de 2020





**Facultad de Ingeniería**  
Universidad de Deusto

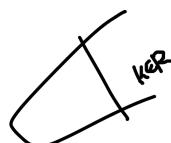
**Ingeniaritza Fakultatea**  
Deustuko Unibertsitatea

## **Grado en Ingeniería Informática Informatikako Ingeniaritzako Gradua**

### **Proyecto fin de grado Gradu amaierako proiekta**

Desarrollo de una aplicación web basada en una arquitectura de microservicios que permita la posterior implantación de una plataforma de experimentación para algoritmos matemáticos y de inteligencia artificial

Alexander Moreno Lobato



Director: Iker Pastor López

Bilbao, julio de 2020



# RESUMEN

El proyecto consiste en el diseño y desarrollo de una aplicación web modular basada en una arquitectura de microservicios, de manera que si uno de los módulos o servicios falla los demás podrán seguir funcionando sin problemas. El objetivo principal de la aplicación es el de ser la base para implantar una plataforma de experimentación en la que investigadores y expertos puedan subir sus investigaciones y algoritmos matemáticos o de inteligencia artificial. La aplicación se desarrollará siguiendo ciertas pautas de diseño y seguridad para que los datos y la información que los usuarios introduzcan en el sistema se mantengan seguros en todo momento.

El desarrollo de la aplicación se puede dividir en dos partes importantes. Por un lado, se encuentra el backend (conjunto de microservicios), que se encargará de gestionar las peticiones entrantes y las respuestas a dichas peticiones. El backend se implementará utilizando un framework de desarrollo basado en Python, como pueden ser Django o Flask, ya que son considerados como buenos ejemplos en el ámbito de los microservicios. Además, para el almacenamiento de los datos se utilizará un sistema de base de datos NoSQL, como es MongoDB por su flexibilidad a la hora de guardar y recuperar información. Por otro lado, tendríamos el frontend, es decir, la parte del sistema que será con la que interactúe el usuario. El frontend se desarrollará empleando tecnologías clásicas como HTML, CSS mediante Bootstrap y JavaScript utilizando alguna librería o framework, como podrían ser Vue o React.

## Descriptores:

- Aplicación web.
- Microservicios.
- React, Javascript, HTML, CSS.
- Python, Flask.
- NoSQL, MongoDB.
- Seguridad.



# ÍNDICE GENERAL

<b>ÍNDICE DE FIGURAS</b>	<b>ix</b>
<b>ÍNDICE DE TABLAS</b>	<b>xiii</b>
<b>1. INTRODUCCIÓN</b>	<b>1</b>
1.1. MOTIVACIÓN . . . . .	2
1.2. ORGANIZACIÓN DEL DOCUMENTO . . . . .	2
<b>2. OBJETIVOS Y ALCANCE</b>	<b>5</b>
2.1. OBJETIVOS . . . . .	5
2.1.1. Objetivo principal y específicos . . . . .	5
2.1.2. Objetivos secundarios . . . . .	6
2.2. ALCANCE . . . . .	7
<b>3. ANTECEDENTES Y JUSTIFICACIÓN</b>	<b>9</b>
3.1. ESTADO DEL ARTE . . . . .	9
3.1.1. Microservicios . . . . .	9
3.1.2. Comparativa entre una arquitectura basada en microservicios y una arquitectura monolítica . . . . .	11
3.2. ÚLTIMAS TENDENCIAS . . . . .	12
<b>4. METODOLOGÍA</b>	<b>15</b>
4.1. METODOLOGÍA UTILIZADA . . . . .	15
4.2. FASES DEL PROYECTO . . . . .	16
4.3. TAREAS PRINCIPALES . . . . .	17
4.4. HITOS DEL PROYECTO . . . . .	18
4.5. EDT . . . . .	19
<b>5. DESARROLLO</b>	<b>21</b>
5.1. ESPECIFICACIÓN DE REQUISITOS DEL SISTEMA . . . . .	21
5.1.1. Stakeholders . . . . .	21
5.1.2. Reglas de negocio y restricciones del proyecto . . . . .	22
5.1.3. Modelo de casos de uso . . . . .	23
5.1.4. Requisitos . . . . .	35
5.2. ESPECIFICACIÓN DEL DISEÑO . . . . .	36
5.2.1. Arquitectura del sistema . . . . .	37
5.2.2. Descripción de la arquitectura . . . . .	37

5.2.3. Modelos del diseño . . . . .	38
5.3. TECNOLOGÍAS UTILIZADAS . . . . .	40
5.4. CONSIDERACIONES SOBRE LA IMPLEMENTACIÓN . . . . .	48
5.4.1. Entorno de trabajo . . . . .	48
5.4.2. Implementación de la interfaz de usuario . . . . .	49
5.4.3. Implementación de los microservicios . . . . .	51
5.4.4. Características de seguridad implementadas . . . . .	52
5.4.5. Resumen de las características implementadas . . . . .	53
5.5. PLAN DE PRUEBAS . . . . .	54
5.5.1. Objetivos de las pruebas y criterios de aceptación . . . . .	54
5.5.2. Estrategia . . . . .	55
5.5.3. Pruebas generales de funcionamiento . . . . .	56
5.5.4. Pruebas de rendimiento . . . . .	57
5.5.5. Pruebas de seguridad . . . . .	64
5.5.6. Pruebas de usabilidad e interfaz de usuario . . . . .	74
5.5.7. Análisis de los resultados obtenidos . . . . .	75
5.6. INCIDENCIAS DEL PROYECTO . . . . .	76
<b>6. PLANIFICACIÓN</b>	<b>79</b>
6.1. PLAN DE RECURSOS HUMANOS . . . . .	79
6.1.1. Organización . . . . .	79
6.1.2. Definición de roles . . . . .	79
6.2. PLAN DE TRABAJO . . . . .	80
6.2.1. Febrero . . . . .	81
6.2.2. Marzo . . . . .	82
6.2.3. Abril . . . . .	83
6.2.4. Mayo . . . . .	84
6.2.5. Junio . . . . .	85
6.3. PERT . . . . .	86
6.4. CRONOGRAMA DEL PROYECTO . . . . .	87
<b>7. PRESUPUESTO</b>	<b>89</b>
7.1. RECURSOS HUMANOS . . . . .	89
7.2. RECURSOS MATERIALES . . . . .	90
7.3. PRESUPUESTO TOTAL . . . . .	90
<b>8. CONCLUSIONES Y PROPUESTAS FUTURAS</b>	<b>91</b>
8.1. CONCLUSIONES . . . . .	91
8.2. TRABAJO FUTURO . . . . .	92
<b>9. BIBLIOGRAFÍA</b>	<b>95</b>
<b>10. ANEXO I: MANUAL DE USUARIO</b>	<b>101</b>
10.1. USO DE LA APLICACIÓN . . . . .	101
10.1.1. Usuarios . . . . .	101
10.1.2. Investigadores . . . . .	102

10.1.3. Administradores . . . . .	105
10.2. PUESTA EN MARCHA DE LA APLICACIÓN EN LOCALHOST . . . . .	109
10.2.1. Frontend . . . . .	109
10.2.2. Backend . . . . .	111
10.3. DESPLIEGUE DE LA APLICACIÓN EN LA NUBE . . . . .	114
<b>11. ANEXO II: DIMENSIÓN ÉTICA DEL PROYECTO</b>	<b>119</b>
11.1. APLICACIÓN DE MI CUESTIONARIO DE EVALUACIÓN SOCIAL DE TECNOLOGÍAS . . . . .	119
11.2. ANÁLISIS Y VALORACIÓN ÉTICA DEL PROYECTO . . . . .	122
11.3. APLICACIÓN DE MI PROPIA ÉTICA PROFESIONAL PERSONAL . . . . .	123
11.3.1. Principios éticos . . . . .	124
11.3.2. Normas de actuación . . . . .	125
11.3.3. Conclusión valorativa . . . . .	129



# ÍNDICE DE FIGURAS

3.1. Arquitectura basada en microservicios <sup>1</sup> . . . . .	10
3.2. Arquitectura monolítica vs. Arquitectura basada en microservicios <sup>2</sup> . . . . .	13
4.1. Metodología en cascada. . . . .	15
4.2. EDT del proyecto . . . . .	19
5.1. Diagrama de casos de uso . . . . .	23
5.2. Diagrama de actividad del PUC1 . . . . .	24
5.3. Diagrama de secuencia del PUC1 . . . . .	24
5.4. Diagrama de actividad del PUC2 . . . . .	25
5.5. Diagrama de secuencia del PUC2 . . . . .	25
5.6. Diagrama de actividad del PUC3 . . . . .	26
5.7. Diagrama de secuencia del PUC3 . . . . .	26
5.8. Diagrama de actividad del PUC4 . . . . .	27
5.9. Diagrama de secuencia del PUC4 . . . . .	27
5.10. Diagrama de actividad del PUC5 . . . . .	28
5.11. Diagrama de secuencia del PUC5 . . . . .	28
5.12. Diagrama de actividad del PUC6 . . . . .	29
5.13. Diagrama de secuencia del PUC6 . . . . .	29
5.14. Diagrama de actividad del PUC7 . . . . .	30
5.15. Diagrama de secuencia del PUC7 . . . . .	30
5.16. Diagrama de actividad del PUC8 . . . . .	31
5.17. Diagrama de secuencia del PUC8 . . . . .	31
5.18. Diagrama de actividad del PUC9 . . . . .	32
5.19. Diagrama de secuencia del PUC9 . . . . .	32
5.20. Diagrama de actividad del PUC10 . . . . .	33
5.21. Diagrama de secuencia del PUC10 . . . . .	33
5.22. Diagrama de actividad del PUC11 . . . . .	34
5.23. Diagrama de secuencia del PUC11 . . . . .	34
5.24. Arquitectura del sistema . . . . .	37
5.25. Diagrama de clases . . . . .	38
5.26. Logo de MongoDB <sup>4</sup> . . . . .	40
5.27. Gráfico comparativo entre la velocidad de MongoDB y MySQL <sup>5</sup> . . . . .	41
5.28. Proceso de replicación maestro-esclavo y maestro-maestro respectivamente <sup>6</sup> . . . . .	41
5.29. Logo de React <sup>7</sup> . . . . .	42

5.30. Logo de Vue <sup>8</sup> . . . . .	42
5.31. Número de descargas realizadas por año en <i>npm</i> desde diferentes frameworks <sup>9</sup> . . . . .	43
5.32. Tendencias en las búsquedas de frameworks en Google <sup>10</sup> . . . . .	43
5.33. Logo de Flask <sup>11</sup> . . . . .	44
5.34. Logo de Django <sup>12</sup> . . . . .	44
5.35. Logo de Docker <sup>13</sup> . . . . .	45
5.36. Logo de Virtualbox <sup>14</sup> . . . . .	45
5.37. Diferencia entre un contenedor de software y una máquina virtual <sup>15</sup> . [Fuente] . . . . .	46
5.38. Logo de Heroku <sup>16</sup> . . . . .	47
5.39. Diagrama de tecnologías de la arquitectura. . . . .	47
5.40. Logo de JSON <sup>17</sup> . . . . .	48
5.41. HTTP y HTTPS <sup>18</sup> . . . . .	48
5.42. Rendimiento del frontend - 200 peticiones . . . . .	59
5.43. Rendimiento del frontend - 400 peticiones . . . . .	60
5.44. Rendimiento del frontend - 600 peticiones . . . . .	60
5.45. Rendimiento del frontend - 800 peticiones . . . . .	61
5.46. Rendimiento del frontend - 1000 peticiones . . . . .	61
5.47. Rendimiento de un microservicio - 200 peticiones . . . . .	62
5.48. Rendimiento de un microservicio - 400 peticiones . . . . .	62
5.49. Rendimiento de un microservicio - 600 peticiones . . . . .	63
5.51. Rendimiento de un microservicio - 1000 peticiones . . . . .	63
5.50. Rendimiento de un microservicio - 800 peticiones . . . . .	64
5.52. Vega Scanner - Primer escaneo . . . . .	65
5.53. Vega Scanner - Problema leve (Falso positivo) . . . . .	65
5.54. Vega Scanner - Problema grave (Falso positivo) . . . . .	66
5.55. Vega Scanner - Problema grave 2 (Falso positivo) . . . . .	66
5.56. Vega Scanner - Solución problemas del tipo <i>Info</i> . . . . .	67
5.57. Vega Scanner - Problema de severidad media . . . . .	68
5.58. Vega Scanner - Problema de severidad media 2 (Falso positivo) . . . . .	68
5.59. Vega Scanner - Solución problema de severidad media . . . . .	69
5.60. Skipfish - Resultados primer escaneo . . . . .	70
5.61. Skipfish - Problema de severidad media . . . . .	70
5.62. Wapiti - Resultados frontend . . . . .	71
5.63. Wapiti - Resultados microservicio de usuarios . . . . .	72
5.64. Wapiti - Resultados microservicio de investigadores . . . . .	73
5.65. Wapiti - Resultados microservicio de administradores . . . . .	74
 6.1. Organigrama de la organización . . . . .	79
6.2. Plan de trabajo de febrero . . . . .	81
6.3. Plan de trabajo de marzo . . . . .	82
6.4. Plan de trabajo de abril . . . . .	83
6.5. Plan de trabajo de mayo . . . . .	84
6.6. Plan de trabajo de junio . . . . .	85
6.7. Diagrama PERT. . . . .	86
6.8. Cronograma completo del proyecto. . . . .	87

10.1. Página principal de la aplicación web. . . . .	102
10.2. Formulario de registro para investigadores. . . . .	102
10.3. Mensaje informativo tras enviar el formulario de registro. . . . .	103
10.4. Formulario de login para investigadores. . . . .	103
10.5. Panel que permite subir contenido a la aplicación web. . . . .	103
10.6. Panel que permite actualizar el nombre de documentos. . . . .	104
10.7. Página principal de investigador. . . . .	104
10.8. Menú desplegable de investigador. . . . .	105
10.9. Perfil de investigadores. . . . .	105
10.10Página de inicio de sesión para administradores. . . . .	106
10.11Panel donde hay que introducir el código de seguridad. . . . .	106
10.12Página principal de administradores. . . . .	107
10.13Página de gestión para administradores. . . . .	107
10.14Botón de cambio de idioma para usuarios. . . . .	108
10.15Botón de cambio de idioma para investigadores y administradores. . . . .	108
10.16Ventana para cambiar de idioma. . . . .	108
10.17Página de visualización de un documento. . . . .	109
10.18MongoDB Atlas - Enlace para conectar el cluster con la aplicación. . . . .	112
10.19MongoDB Atlas - Creación de usuario con permisos sobre el cluster. . . . .	112
10.20MongoDB Atlas - Modificar acceso de red a MongoDB Atlas. . . . .	113
10.21Repositorios de Github para desplegarlos en la nube. . . . .	114
10.22Heroku - Crear una aplicación en la plataforma. . . . .	114
10.23Heroku - Conectar aplicación con un repositorio de Github. . . . .	117
10.24Heroku - Desplegar rama de un repositorio de Github vinculado con Heroku. . . . .	117
10.25Heroku - Acceder a una aplicación desplegada. . . . .	117



# ÍNDICE DE TABLAS

3.1. Resumen comparativo sobre la arquitectura monolítica y la arquitectura basada en microservicios <sup>3</sup> . . . . .	11
7.1. Detalle del precio por hora de cada recurso (I.V.A. no incluido). . . . .	89
7.2. Desglose de precios por recurso (I.V.A. no incluido). . . . .	89
7.3. Desglose de precios por tarea (I.V.A. no incluido). . . . .	90
7.4. Desglose de precios por material (I.V.A. no incluido). . . . .	90
7.5. Presupuesto total del proyecto (I.V.A. no incluido). . . . .	90
11.2. Valoración final de la herramienta a partir del cuestionario social de evaluación de tecnologías. . . . .	120
11.1. Síntesis de impactos detectados con el cuestionario social de evaluación de tecnologías.	121



## Capítulo 1

# INTRODUCCIÓN

El presente documento representa la memoria del proyecto de fin de grado de un alumno de la Universidad de Deusto llamado Alexander Moreno Lobato para el grado en Ingeniería Informática. El proyecto en cuestión está enfocado, por lo tanto, a una temática concreta dentro de una rama de la ingeniería como es la informática. El objetivo de este escrito es el de dejar plasmado en papel u otro medio digital la documentación y la memoria del proyecto realizado. El título del proyecto es: “*Desarrollo de una aplicación web basada en una arquitectura de microservicios que permita la posterior implantación de una plataforma de experimentación para algoritmos matemáticos y de inteligencia artificial*”. Cabe destacar que este proyecto ha sido llevado a cabo de forma individual con la guía del profesor Iker Pastor López de la Universidad de Deusto.

El principal objetivo del proyecto es el de desarrollar una aplicación basada en una arquitectura de microservicios, modular y extensible que permita que investigadores suban contenido como informes, algoritmos, investigaciones, documentos, etc. Además, todo este contenido estará a la disposición de los usuarios de la aplicación de tal forma que puedan consumirlo fácilmente. Con esto, se busca desarrollar lo que podría ser considerado como una base para la posterior implantación de una plataforma de experimentación en la que investigadores y técnicos puedan compartir conocimientos, algoritmos, artículos, etc. Dicha plataforma se podría conseguir con el desarrollo e implementación futura de nuevos módulos o extensiones dirigidas a la aplicación que se busca crear en este proyecto.

Además, el proyecto se ha enfocado hacia la investigación e implementación de características de seguridad que mantengan un nivel de seguridad lo más alto posible a la hora de acceder a la aplicación, en el tratamiento de datos sensibles de los usuarios o en la puesta en marcha de los microservicios y el frontend. En el resto de la introducción se describe la motivación que me ha llevado a la elección y realización del proyecto, incluyendo también una sección en la que se define la organización que siguen el resto de apartados de la memoria.

## 1. INTRODUCCIÓN

### 1.1. MOTIVACIÓN

La principal motivación que me ha llevado a elegir esta temática de proyecto es mi interés personal por el desarrollo de aplicaciones y la programación. Además, como durante la carrera los proyectos realizados para las distintas asignaturas se fundamentaban en una arquitectura monolítica, en esta ocasión me ha motivado aún más el hecho de variar en ese aspecto tomando como base una arquitectura de microservicios.

Por otra parte, el llevar a cabo este proyecto me ha permitido investigar y adquirir conocimientos sobre la materia, lo que me puede ser de gran utilidad en mi futuro profesional ya que cada vez hay más empresas que tienden a utilizar arquitecturas basadas en microservicios en sus desarrollos [1]. Además, tiene ciertas ventajas frente a la arquitectura monolítica que se comentarán posteriormente en el apartado de *Antecedentes y justificación*.

El principal problema que pueden tener las aplicaciones que se basan en una arquitectura monolítica es que suelen ser bastante grandes, por lo que la tarea de mantener, actualizar o extender la aplicación se complica cada vez más. Por lo tanto, con la arquitectura que se presenta en este proyecto, se busca solventar este gran problema que puede hacer que la labor de los profesionales que se encarguen de realizar este tipo de tareas sea una auténtica faena.

### 1.2. ORGANIZACIÓN DEL DOCUMENTO

En esta sección se define el orden y contenido de los apartados que conforman la memoria del proyecto. A continuación se enumeran y describen de forma breve cada uno de ellos:

1. **Introducción:** se explica brevemente el objetivo primordial del proyecto, la motivación que me ha llevado a elegir esta temática y la organización de la memoria dividida en apartados.
2. **Objetivos y alcance:** se especifican de una forma más detallada tanto los objetivos principales como los objetivos secundarios que forman el proyecto. Además, también se incluye el alcance del proyecto, compuesto por un listado de “entregables” junto con una breve descripción de cada uno de ellos.
3. **Antecedentes y justificación:** se describe el contexto que rodea el proyecto y que ha dado pie a su realización. Más concretamente, se habla sobre el estado del arte y las últimas tendencias de los microservicios, aportando datos e información que justifiquen la oportunidad de llevar a cabo el proyecto.
4. **Metodología:** se detalla el procedimiento de trabajo seguido a la hora de ejecutar el proyecto, así como las fases, hitos y tareas principales que lo componen. De igual forma, se incluye un apartado con el *EDT* del proyecto.
5. **Desarrollo:** este es el capítulo que puede considerarse como el núcleo del proyecto, en el que se explican en detalle el diseño del sistema, los requisitos funcionales y no funcionales, las tecnologías utilizadas en el desarrollo de la aplicación web, algunas consideraciones sobre la implementación, los tipos de pruebas realizadas y las incidencias que se han dado durante su transcurso.

6. **Planificación:** se centra en especificar como se han organizado las tareas que conforman el proyecto a lo largo del tiempo: el tiempo empleado en cada una de ellas, el rol o roles que han participado en su realización...
7. **Presupuesto:** consiste en detallar los diferentes costes del proyecto, tanto materiales como de RRHH. Para ello, se incluyen varias tablas en las que se tienen en cuenta los costes humanos y materiales del proyecto, así como su coste total.
8. **Conclusiones:** se recogen las aprendizajes más significativos que han supuesto para mi persona la realización del proyecto. Además, se mencionan algunas propuestas que tienen como objetivo mejorar el sistema desarrollado en un futuro.
9. **Bibliografía:** es el listado de referencias que se han consultado durante el transcurso del proyecto para la redacción de algunos apartados de la memoria.
10. **Definiciones, acrónimos y abreviaturas:** se recogen y aclaran las definiciones, acrónimos y abreviaturas que hayan aparecido a lo largo de la memoria del proyecto y que puedan generar dudas o sean relevantes por alguna razón.
11. **Anexos:** está formado por documentos, que podríamos considerar independientes a la memoria del proyecto, que se ha considerado oportuno que se incluyan en capítulos aparte.
  - a) **Anexo I: Manual de Usuario.**  
Se centra en explicar la forma de utilizar las distintas funcionalidades que ofrece la aplicación web. Asimismo, se detalla como poner en marcha y desplegar la aplicación completa tanto en localhost como en la nube.
  - b) **Anexo II: Dimensión ética del proyecto.**  
Este anexo procura analizar el proyecto desde el punto de vista de la ética y dejar claro si la aplicación o el proyecto en sí pueden ser considerados éticamente correctos o aceptables.



## Capítulo 2

# OBJETIVOS Y ALCANCE

En este capítulo se describen más detalladamente los objetivos y los distintos puntos que conforman el alcance del proyecto. Para este proyecto los objetivos se han clasificado en dos tipos, por un lado se encuentran los objetivos principales, y por otro lado, los secundarios.

### 2.1. OBJETIVOS

#### 2.1.1. Objetivo principal y específicos

El objetivo principal del proyecto consiste en el desarrollo de una solución software, más concretamente una aplicación web, basada en una arquitectura de microservicios que permita servir de base para la futura implantación de una plataforma de experimentación dirigida a investigadores y técnicos. Con esta aplicación base los investigadores o técnicos podrán registrarse, iniciar sesión y subir sus documentos e investigaciones que quieran compartir con la comunidad.

La aplicación podríamos dividirla en dos partes bien diferenciadas. En primer lugar, tendríamos el frontend, la parte del sistema que está en constante interacción con el usuario. Y en segundo lugar, se encuentra el backend, la parte del sistema que se encarga de gestionar todas las peticiones entrantes provenientes del frontend.

Dentro del objeto del proyecto, en lo que respecta al backend de la aplicación, podemos observar 3 servicios fundamentales que se busca implementar:

- **Servicio de usuarios:** permite que usuarios que visiten la página web puedan consultar y consumir el contenido publicado por los investigadores. Dicho contenido, como informes, algoritmos, artículos..., podrá ser visualizado o en su defecto, descargado sin mayor complejidad.
- **Servicio de investigadores:** los investigadores serán capaces de crear una cuenta en el sistema para, posteriormente, iniciar sesión. Una vez hecho esto, podrán realizar diferentes acciones dentro de la aplicación como por ejemplo: subir nuevo contenido, modificar los datos de su perfil, eliminar contenido subido, modificar el título de los documentos subidos, etc.

## 2.OBJETIVOS Y ALCANCE

- **Servicio de administradores:** en el sistema de base de datos podrán ser registrados usuarios “especiales” de forma manual que actuarán como administradores. Estos podrán identificarse en el panel administrativo de la aplicación, donde dispondrán de todo el contenido almacenado en el sistema por los investigadores y tendrán la capacidad de aceptar o rechazar la creación de nuevas cuentas de investigador. Cuando lo consideren necesario podrán eliminar contenido almacenado en el sistema que consideren inadecuado para la plataforma.

En relación al desarrollo del frontend, decir que su misión será la de conectar la “figura” del usuario con la funcionalidad del sistema, es decir, los clientes que se conecten a la aplicación interactuarán con el frontend, y este realizará las peticiones y gestiones necesarias intercambiando datos con los microservicios con el fin de proporcionarles a los usuarios la funcionalidad oportuna en cada momento.

Para llevar a cabo el desarrollo de la aplicación y cumplir con los objetivos principales del proyecto, es necesario también realizar un estudio y análisis previo de tecnologías de tal forma que se elijan las que mejores resultados puedan tener de cara a la obtención de la solución final.

### 2.1.2. Objetivos secundarios

Dentro de este apartado se incluyen otros objetivos que a pesar de ser secundarios, son realmente necesarios e importantes para el buen desarrollo del proyecto, así como para mantener la integridad y seguridad de los datos con los que trabaja el sistema.

Por un lado, estaría la seguridad del sistema en general y los datos, sobre todo de los datos sensibles como contraseñas. Es realmente importante bastionar el sistema a través de técnicas, políticas y herramientas de seguridad como por ejemplo el hashing de contraseñas, de tal manera que pueda hacerle frente a todo tipo de ataques informáticos mitigando sus impactos negativos lo máximo posible.

Por otro lado, se encuentra el objetivo de identificar las tecnologías óptimas y más adecuadas que permitan implementar la arquitectura basada en microservicios de la mejor manera posible. Para ello, tal y como se ha dicho anteriormente, se realizará un estudio previo comparando distintas alternativas junto con sus ventajas y desventajas. Con las conclusiones que se extraigan del estudio, se elegirán las tecnologías que mejores resultados puedan obtener y menos inconvenientes puedan generar.

Además de lo anterior, cabe destacar que se procurará implementar una interfaz gráfica lo más intuitiva y sencilla posible, de forma que los usuarios de la aplicación no tengan ninguna dificultad a la hora de utilizarla. También es importante que la aplicación tenga un rendimiento razonable de tal forma que los procesos que lleve a cabo se hagan rápidamente, además de que se les proporcione una respuesta a las peticiones de los clientes lo antes posible.

Por último, con el objetivo de asegurar que la solución final cumpla con los objetivos, características y alcance definidos en este capítulo de la documentación del proyecto, se llevaran a cabo una serie

de pruebas software de diferentes tipos, como por ejemplo: pruebas generales de funcionamiento, pruebas de seguridad, pruebas de rendimiento, etc.

## 2.2. ALCANCE

La lista que aparece a continuación es completa y conforma el alcance del proyecto. Solo estarán incluidos en el proyecto los entregables que aparecen en ella:

- A1: Recopilación de requisitos funcionales y no funcionales con los que poder llevar a cabo la posterior implementación de la aplicación web.
- A2: Estudio que compare distintas tecnologías con el fin de identificar las que mejor resultados puedan obtener en la creación, desarrollo y ejecución de microservicios. Además de esto, se tendrá en cuenta también si son tecnologías libres o propietarias.
- A3: Diseño de la arquitectura basada en microservicios ideada para la creación de la aplicación web.
- A4: Definición de características de seguridad de la aplicación web con las que poder mantenerla lo más segura posible frente a ataques que pueda sufrir por parte de agentes maliciosos.
- A5: El frontend de la aplicación web formado por una interfaz gráfica intuitiva y sencilla de utilizar, que interactúe con los usuarios y les permita darle uso a todas las funcionalidades del sistema disponibles.
  - Implementación de las distintas páginas que forman la aplicación web.
  - Aplicación de estilos CSS a los componentes de las páginas web desarrolladas.
  - Implementación de *i18n* para mostrar las páginas de la aplicación web en diferentes idiomas.
  - Implementación de características de seguridad: sistema de registro de investigadores y sistema de inicio de sesión de administradores en 2 pasos.
- A6: El backend de la aplicación web formado por un conjunto de microservicios que reciban, y a su vez, gestionen y den respuesta a todas las peticiones enviadas desde la interfaz gráfica (frontend).
  - Implementación de los métodos o funciones necesarias para responder a todas las peticiones provenientes del frontend y proporcionar los datos que se requieran en cada momento.
  - Implementación de características de seguridad: JWT Token y hashing de contraseñas.
- A7: Creación de una base de datos NoSQL que permita recibir, almacenar y modificar los datos referentes al contenido subido por los investigadores como nombres de usuario, contraseñas, documentos, etc. Este punto se podría considerar como parte del backend de la aplicación.
- A8: Elaboración de una plan de pruebas en el que se detalle el conjunto de comprobaciones realizadas sobre la aplicación web final, de igual forma que los resultados obtenidos.

## 2.OBJETIVOS Y ALCANCE

- A9: Redacción de la memoria del proyecto (este documento), en la que se deja por escrito el proceso seguido para la realización y desarrollo del mismo. De igual manera, se incluirá en ella información relevante que se considere oportuna para evitar cualquier posible duda que pueda surgir durante su lectura.
- A10: Creación de un manual de usuario donde se expliquen las funcionalidades y características principales de la solución final, además de como utilizar los distintos componentes que conforman la interfaz gráfica. Dentro del manual también se incluirá un apartado en el que se explique el proceso seguido para el despliegue de la arquitectura completa y la aplicación web tanto en local como en la nube.
- A11: Redacción de un apartado o anexo en el que se tenga en cuenta la perspectiva ética en relación al proyecto. Su objetivo será principalmente el de asegurar que el proyecto y la aplicación desarrolladas es éticamente aceptable.

Dentro de este apartado, es preciso detallar también el listado de las cuestiones que no están incluidas en el alcance del proyecto:

- N1: El proyecto no se centra en el diseño de UI. Por lo tanto, no se implementará una interfaz excesivamente compleja o atractiva visualmente.
- N2: No se implementaran una gran cantidad de funcionalidades en la aplicación, ya que el proyecto se ha enfocado más a la preparación de la arquitectura, los microservicios, la implementación de características de seguridad, la redacción de la documentación, etc.

## Capítulo 3

# ANTECEDENTES Y JUSTIFICACIÓN

Este capítulo se centra en describir las condiciones del entorno en las que se lleva a cabo el proyecto. Con esto, se busca recoger datos e información sobre la situación actual en lo que respecta a los microservicios y a la arquitectura basada en microservicios en la que se ha basado el desarrollo del proyecto. A continuación, se habla del estado del arte y las últimas tendencias referentes a este ámbito. Así, se busca conocer mejor el contexto y la situación en la que se encuentran los microservicios con el fin de justificar la oportunidad de realización del proyecto.

### 3.1. ESTADO DEL ARTE

#### 3.1.1. Microservicios

Los microservicios son un paradigma de desarrollo software que consiste en la creación de una aplicación como un conjunto de servicios pequeños, que se ejecutan independientemente del resto y en procesos diferentes [2]. La conexión entre ellos suele establecerse mediante el protocolo HTTP y cada uno se centra en una o unas pocas funcionalidades, de tal manera que se consigue un sistema capaz de funcionar sin tener operativos todos los microservicios a la vez [3].

Cada vez se están viendo más a menudo en el mundo empresarial. Es por eso que algunas empresas han empezado a migrar sus sistemas y aplicaciones hacia el ámbito de los microservicios ya que muchos desarrolladores se han dado cuenta de que este tipo de paradigmas o arquitecturas influyen de una forma muy positiva en factores como el rendimiento o la estabilidad de una aplicación [4], características que son clave en el desarrollo de aplicaciones.

En lo que respecta a las aplicaciones web, los microservicios pueden aportarles grandes ventajas, porque normalmente suelen ser aplicaciones formadas por diferentes “bloques” de funcionalidad que podríamos organizar o estructurar como un microservicio independiente cada uno de ellos. De esta forma se consigue una mayor eficiencia, escalabilidad y disponibilidad [5]. Además, se pueden adaptar a distintas necesidades de negocio lo que permite desarrollar una amplia variedad de aplicaciones utilizando los microservicios.

### 3. ANTECEDENTES Y JUSTIFICACIÓN

Cabe destacar que debido a las oportunidades competitivas que han surgido en el mercado, las empresas se ven en la obligación de realizar entregas en el menor tiempo posible. Esto se puede considerar otra razón de peso por las que las empresas están poco a poco evolucionando y empezando a emplear metodologías o paradigmas más novedosos como podrían ser los microservicios. Como cada vez la cantidad de dispositivos, usuarios e información va aumentando considerablemente surge la necesidad de la implementación de aplicaciones capaces de darles respuesta a una gran cantidad de peticiones y transacciones simultáneamente siendo necesaria una buena gestión de los recursos disponibles de forma que la aplicación cumpla con sus objetivos.

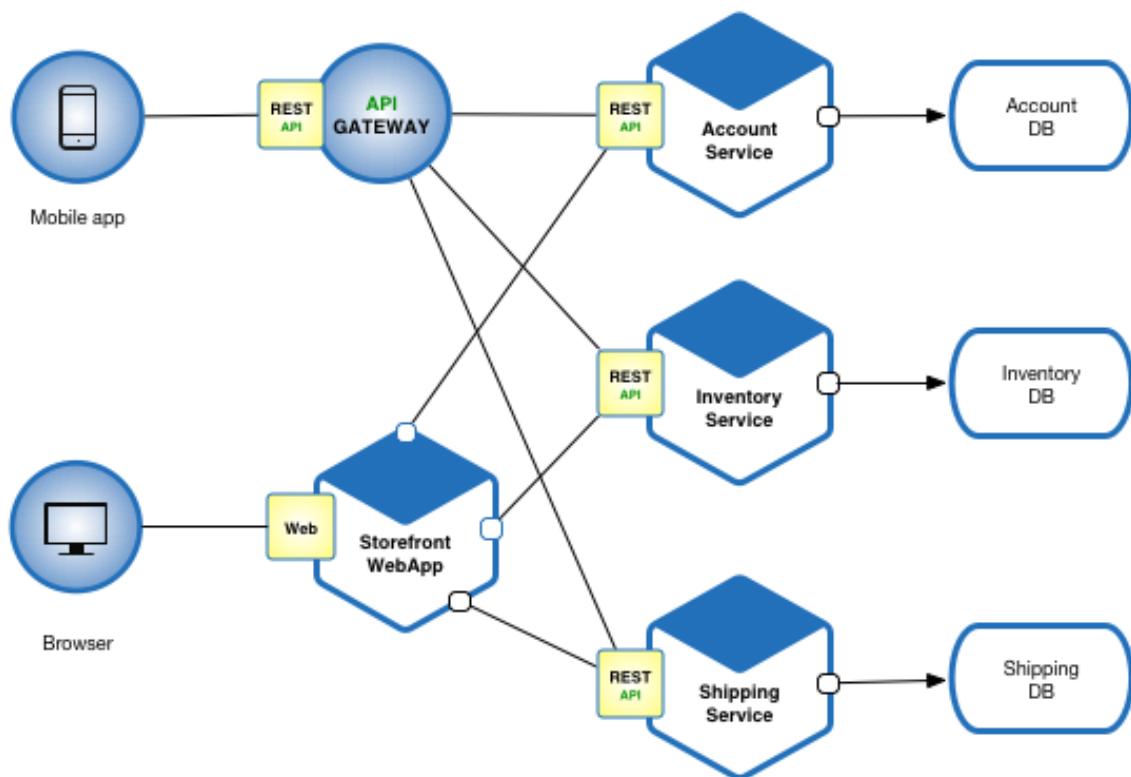


Fig. 3.1: Arquitectura basada en microservicios<sup>1</sup>.

En la actualidad, se busca el despliegue de estos recursos computacionales en la nube, para así mejorar su gestión habilitando la posibilidad de acceder a ellos desde cualquier sitio de forma remota. Las aplicaciones tradicionales suelen estar basadas en una arquitectura monolítica, lo que conlleva que sean bastante grandes. Esto hace que se incremente la complejidad a la hora de migrar las aplicaciones a la nube, cosa que con los microservicios se soluciona ya que como se ha comentado previamente, las arquitecturas basadas en microservicios permiten un mejor escalado, despliegue, integración... debido a la simpleza y ligereza con la que se busca implementarlos [6].

<sup>1</sup>Fuente: <https://microservices.io/patterns/microservices.html>

### 3.1.2. Comparativa entre una arquitectura basada en microservicios y una arquitectura monolítica

En el pasado, las aplicaciones se han basado en una arquitectura monolítica, que consiste en tener toda la funcionalidad de la aplicación en un mismo bloque o conjunto de servicios. Sin embargo, se puede observar como las empresas se van decantando por soluciones más dirigidas hacia los microservicios. En este apartado se intentan recoger las conclusiones más relevantes extraídas tras realizar un estudio comparativo entre ambas, de manera que quedé lo más claro posible las ventajas que tienen las arquitecturas basadas en microservicios frente a las arquitecturas monolíticas. Para ello, se sintetizan en la siguiente tabla sus características más representativas:

Arquitectura monolítica	Arquitectura basada en microservicios
Utiliza una única base de datos que se conecta con el backend de la aplicación.	Suele utilizar múltiples bases de datos que suelen conectarse con uno o varios microservicios.
La funcionalidad implementada en las aplicaciones que poseen una arquitectura monolítica se concentra en un mismo servicio, bloque o proceso. Toda su lógica se encuentra centralizada en un mismo punto.	En las aplicaciones que tienen una arquitectura basada en microservicios la funcionalidad se reparte en varios bloques. Normalmente, cada microservicio implementa la funcionalidad referente a un ámbito de la aplicación.
Es fácil de desarrollar y no hay que pensar la forma de dividir en servicios las distintas funcionalidades de la aplicación. Además, como es la que se ha utilizado tradicionalmente, existe mucha información sobre desarrollos o proyectos empleando este tipo de arquitectura.	Permite escalar cada microservicio de manera distinta según las necesidades específicas de cada uno. Además, al ser un conjunto de "proyectos" pequeños, ligeros e independientes se pueden desarrollar de una manera más flexible y sencilla.
Funciona de forma autónoma, y no depende de otros servicios.	Permite una mejor asignación de recursos y ofrece un mayor rendimiento gracias a la ligereza que poseen los microservicios.
Debido a su gran tamaño, surgen más complicaciones a la hora de detectar y corregir errores o bugs. Además, con el tiempo, se va haciendo más y más grande.	Cuando una parte del sistema se hace demasiado grande, es posible dividirla creando nuevos bloques o microservicios más pequeños y fáciles de gestionar.
No funciona bien con aplicaciones que van a ser ampliadas con regularidad.	Funciona estupendamente con aplicaciones que se actualizan continuamente.
El backend de la aplicación se suele implementar utilizando un único lenguaje de programación.	Permite la implementación de cada microservicio utilizando un lenguaje de programación distinto.
Cuando se quiere desplegar una nueva funcionalidad o es necesario reiniciar la aplicación, se debe reiniciar el sistema completo lo que puede generar un descontento entre los usuarios debido a los tiempos de espera. Además, cuanto más grande y compleja es la arquitectura, más aumenta el número de posibles vulnerabilidades que podrían suponer un riesgo para ella.	Proporciona una mayor seguridad, aislamiento de código y tolerancia a fallos ya que en caso de que surja alguna complicación en un microservicio, las demás partes de la aplicación podrán continuar funcionando sin problemas. Incluso se podrá reiniciar dicho microservicio rápidamente en otra máquina para así evitar posibles pérdidas de datos e información sensible.

Tabla 3.1: Resumen comparativo sobre la arquitectura monolítica y la arquitectura basada en microservicios<sup>2</sup>.

## 3.2. ÚLTIMAS TENDENCIAS

Como se ha comentado previamente, las empresas están empezando a migrar sus sistemas hacia arquitecturas basadas en microservicios. La mayoría de ellas lo que buscan es conseguir una mayor escalabilidad y velocidad en sus aplicaciones por lo que los microservicios son una de las mejores opciones. Actualmente, hay múltiples tendencias entorno al ámbito de los microservicios, sin embargo, a continuación se comentan algunas de las más representativas.

Primero, estaría la tendencia de llevar todo a la nube. Los desarrolladores están empezando a desplegar sus aplicaciones en plataformas en la nube donde es fundamental que las aplicaciones sean fácilmente escalables y dispongan de una gran velocidad [9, 10]. Por un lado, la escalabilidad ofrece múltiples beneficios como la posibilidad de ofrecer un servicio ininterrumpido, soportar más demanda de usuarios y ahorrar en costes hardware [11]. Por otro lado, la velocidad permite principalmente evitar esperas excesivas, y consigue que los usuarios en general estén más contentos con el sistema [12]. Teniendo esto en cuenta, gracias al pequeño tamaño y la simpleza de cada microservicio se pueden escalar y actualizar de forma sencilla, además de que al tener una complejidad tan reducida poseen una alta velocidad.

Otra tendencia que merece la pena comentar es la que está relacionada con la demanda incesante de las empresas que consiste en la necesidad continua de actualizar sus aplicaciones y productos tecnológicos. Una arquitectura basada en microservicios les facilita el trabajo a las personas que vayan a encargarse de dichas actualizaciones ya que por su diseño se reduce la complejidad que puede suponer el añadir nuevos módulos o extensiones a la aplicación.

Por último, cabe destacar otra tendencia vinculada de manera más directa con este proyecto que consistiría en acercar el desarrollo web cada vez más al paradigma de los microservicios. Gracias a esto, se pueden crear aplicaciones web de forma más rápida y eficiente, siendo estas mucho más ligeras que si hubieran sido diseñadas e implementadas basándose en otro tipo de arquitecturas, como por ejemplo la arquitectura monolítica [13]. Al fin y al cabo, cuanta menos complejidad y más pequeñas sean las aplicaciones, más eficientes serán a la hora de satisfacer las necesidades de los usuarios, y además, menos vulnerabilidades tendrán. Por lo tanto, será más difícil que agentes maliciosos encuentren puntos débiles de los que aprovecharse para atacar el sistema.

Como se puede observar se está produciendo un cambio realmente notorio en el mundo del desarrollo de aplicaciones, en el que cada vez se les da más relevancia a los microservicios. Algunos ejemplos de este cambio podemos encontrarlos en empresas muy conocidas, siendo estas, algunas de las más grandes y con mayores beneficios del mundo como por ejemplo: Amazon, Coca Cola, eBay, Netflix, Spotify o Uber [14].

---

<sup>2</sup>Referencias: [7, 8]

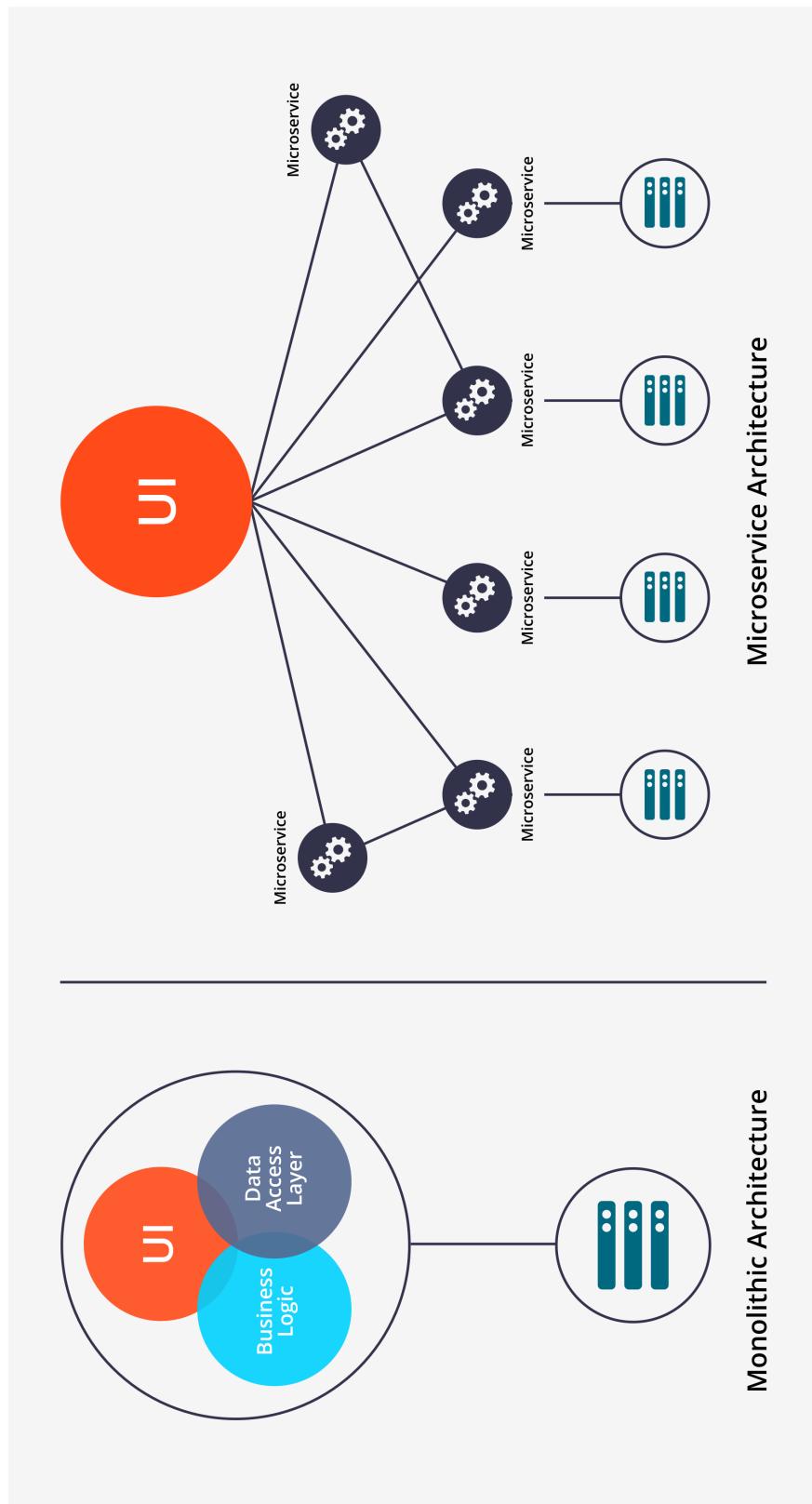


Fig. 3.2: Arquitectura monolítica vs. Arquitectura basada en microservicios<sup>3</sup>.

<sup>3</sup>Fuente: <https://medium.com/startlovingyourself/microservices-vs-monolithic-architecture-c8df91f16bb4>



## Capítulo 4

# METODOLOGÍA

Este capítulo se centra en describir el método seguido para la realización del proyecto de forma que quede clara la estructura de la metodología seguida en todo el transcurso del mismo. Este capítulo se organiza como se define a continuación. Primero, se especifica el método de trabajo empleado en el que se explican los pasos seguidos durante toda la ejecución del proyecto. Posteriormente, se enumeran y definen las diferentes fases que conforman el proyecto, junto con sus respectivas tareas principales. Y por último, se incluye un diagrama que representa el *EDT* del proyecto.

### 4.1. METODOLOGÍA UTILIZADA

Para llevar a cabo el proyecto se ha utilizado una metodología en cascada ya que al tener un único recurso las diferentes fases que forman el proyecto, excepto la documentación, se han realizado de forma secuencial. Cabe destacar que la fase de documentación se ha ido completando a lo largo de todo el proyecto, desde el comienzo hasta la finalización del mismo.

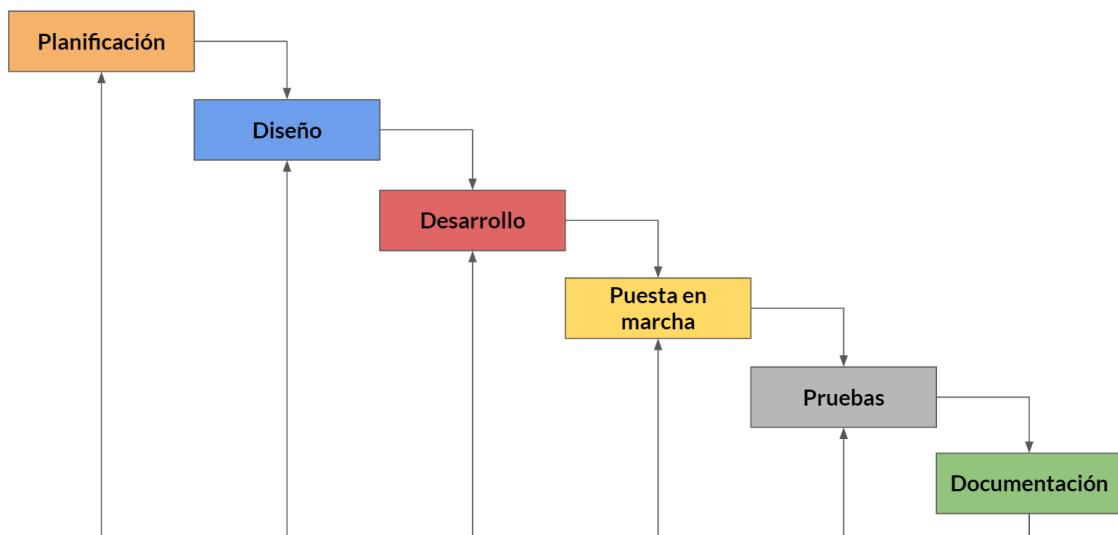


Fig. 4.1: Metodología en cascada.

## 4.METODOLOGÍA

La fortaleza más relevante de la metodología en cascada, y por la que se ha decidido utilizarla, es que se le da bastante importancia a la planificación del proyecto [15]. Esto, para un trabajo como este (proyecto de fin de grado), es de gran utilidad ya que teniendo una planificación lo más detallada posible, se puede saber en todo momento cual es el siguiente paso a seguir y se evitan desaprovechamientos de tiempo innecesarios. Además, gracias esta metodología se ha conseguido cumplir con los objetivos del proyecto de forma segura y teniendo todo bajo control, un factor realmente importante cuando hay que enfrentarse a un proyecto diferente a lo que se está acostumbrado.

Aunque esta metodología fluya de manera secuencial, no significa que no se pueda volver a una fase anterior. Por ejemplo, en caso de encontrarnos en la fase de la puesta en marcha, y se detecte que hay algún fallo o aspecto a mejorar en alguna fase anterior, se puede volver atrás y corregir el problema cuando se considere oportuno.

### 4.2. FASES DEL PROYECTO

El definir una metodología desde un principio es imprescindible para realizar el proyecto de una manera organizada, eficiente y ajustada al tiempo disponible. Una mala organización puede suponer la aparición de grandes complicaciones a lo largo de todo el proyecto. Es por esto que al comienzo del mismo, se lleva a cabo una fase de planificación, en la que se identifican todas las tareas que se deben llevar a cabo para obtener la solución final, así como su duración y carga de trabajo.

Tras la fase de planificación, se encuentran las fases de diseño y desarrollo de la aplicación web en las que se incluyen la implementación del frontend y el backend, el diseño o la forma de almacenar los datos y documentos en la base de datos, la elección de tecnologías y lenguajes de programación web, el diseño de la arquitectura basada en microservicios... Como se puede observar, estas dos fases, sobre todo la de desarrollo, son el núcleo principal del proyecto, por lo tanto deben realizarse de la manera más eficiente, eficaz y óptima posible.

Después de finalizar el desarrollo del sistema encontramos la fase que consiste en poner en marcha la arquitectura diseñada previamente. Esto conlleva el arranque de todos los microservicios de manera que interactúen entre sí, con las bases de datos correspondientes y con el frontend, posibilitando el uso de todas las funcionalidades del sistema implementadas en la fase anterior. Una vez que el sistema completo esté operativo se hacen una serie de pruebas y comprobaciones con el objetivo de asegurar la fiabilidad, el buen funcionamiento y la compatibilidad entre todos los componentes que conforman la arquitectura del sistema, como son los microservicios, las bases de datos y el frontend.

Por último, tendríamos la fase de documentación. Una vez comprobado y validado el correcto funcionamiento del sistema, se procede a redactar un manual de usuario o los apartados faltantes de la memoria con toda la información que se considere oportuna. Cabe destacar que la memoria de este proyecto se ha ido redactando en paralelo al desarrollo de la solución final.

Durante todas las fases comentadas anteriormente se han ido realizando reuniones intermedias entre el director del proyecto y el alumno. Con esto se busca realizar un seguimiento continuo,

solucionar posibles dudas o problemas que hayan podido surgir y definir objetivos a corto plazo de tal forma que el transcurso del proyecto no se vea detenido en ningún momento.

### 4.3. TAREAS PRINCIPALES

En este apartado se recogen las fases y las tareas principales que es necesario completar para cumplir con el alcance del proyecto:

1. **Planificación:** definición y organización de las fases y tareas a completar en el transcurso del proyecto. Además, se llevará a cabo un plan de trabajo en el que se definirá el esfuerzo (en horas) necesario emplear en cada tarea, así como varios diagramas en los que se pueda representar de una forma más gráfica el tiempo de esfuerzo que se haya reflejado en el plan de trabajo.
  - a) **T1:** Primeras investigaciones sobre la temática.
  - b) **T2:** Estudio y comparativa entre distintas tecnologías, frameworks y lenguajes de programación web.
  - c) **T3:** Definición de fases y tareas principales.
  - d) **T4:** Definición de los objetivos y alcance.
  - e) **T5:** Creación del EDT.
2. **Diseño:** diseño de la arquitectura que conforma el sistema. En esta fase se incluye el formato en el que se van a almacenar los datos y documentos en la base de datos, las especificación de requisitos, el diseño de la arquitectura del sistema, etc.
  - a) **T6:** Recopilación de requisitos funcionales y no funcionales.
  - b) **T7:** Selección de las tecnologías, frameworks y lenguajes de programación para el posterior desarrollo.
  - c) **T8:** Diseño de la arquitectura basada en microservicios.
  - d) **T9:** Estructura o formato de los datos y contenido que se almacenará en la base de datos.
  - e) **T10:** Definición de características de seguridad.
3. **Desarrollo:** implementación del backend y frontend de la aplicación utilizando las tecnologías y los lenguajes de programación web elegidos, el plan de pruebas, explicación de las tecnologías utilizadas finalmente, y las incidencias ocurridas durante el transcurso del proyecto.
  - a) **T11:** Implementación del frontend, o sea, la UI con la que interactúen los usuarios.
  - b) **T12:** Implementación del backend, es decir, los microservicios que formen la arquitectura.
  - c) **T13:** Integración del frontend y el backend, junto con las bases de datos que se consideren necesarias.

#### 4.METODOLOGÍA

- d) **T14:** Almacenamiento de datos en la base de datos.
4. **Puesta en marcha:** arranque de la arquitectura, y por consiguiente, de los microservicios, bases de datos y frontend. Todo el proceso seguido se explica más detalladamente en el *Anexo I: Manual de usuario*.
- a) **T15:** Arranque de la aplicación web, los microservicios y la base de datos utilizando Docker.
5. **Pruebas:** comprobaciones y testeos sobre la aplicación web operativa. La realización de las pruebas sobre la aplicación se explican dentro del apartado 5.5 de la memoria.
- a) **T16:** Pruebas de la aplicación web para comprobar el buen funcionamiento y la interacción entre todos los componentes de la arquitectura.
- b) **T17:** Pruebas de rendimiento.
- c) **T18:** Análisis de los resultados obtenidos de las pruebas.
6. **Documentación:** redacción de los apartados restantes de la memoria, manuales y cualquier otro documento que se considere necesario incluir para clarificar cualquier duda que pueda surgir durante la utilización del sistema.
- a) **T19:** Redacción de la memoria del proyecto.
- b) **T20:** Redacción de un manual de usuario.
- c) **T21:** Redacción de las dificultades y conclusiones surgidas en el transcurso del proyecto.

#### 4.4. HITOS DEL PROYECTO

Para asegurar el buen transcurso del proyecto se han definido una serie de hitos intermedios que deberán cumplirse para poder completar el proyecto al 100 %. Las distintas etapas que conforman el proyecto se ejecutarán de manera secuencial de tal manera que no se iniciará una etapa sin haber concluido las anteriores previamente, ya que las etapas intermedias o la última dependen directamente de sus predecesoras.

Para este proyecto se puede identificar 1 hito por cada etapa (planificación, diseño, desarrollo...), que se cumplirán tras finalizar cada una de ellas. Por lo tanto, al final del proyecto se habrán completado 6 hitos en total. Estos hitos también podrían considerarse como productos intermedios que se corresponden con diferentes apartados de la memoria como la planificación, las especificaciones del diseño, el desarrollo o la documentación.

## 4.5. EDT

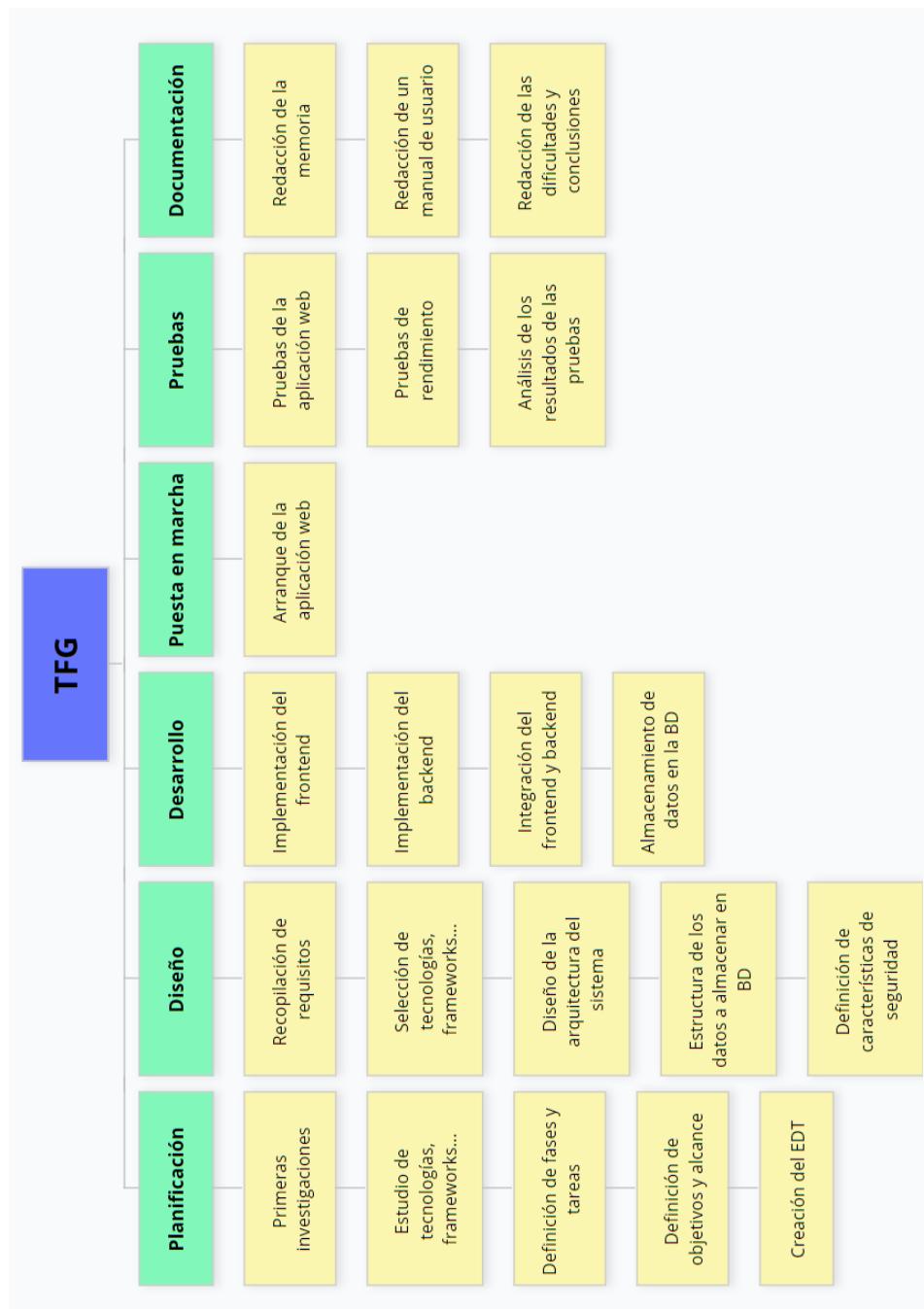


Fig. 4.2: EDT del proyecto



## Capítulo 5

# DESARROLLO

Este capítulo es el núcleo principal del proyecto. Por lo tanto, es uno de los más largos y extensos. En él, se explican detalladamente las cuestiones y conceptos más específicos vinculados con el proceso de desarrollo de la solución final. El capítulo está organizado como se define a continuación.

Primero, se realiza la especificación de requisitos del sistema, donde se definen los stakeholders, reglas de negocio, restricciones del proyecto, casos de uso y requisitos funcionales y no funcionales del sistema que se va a desarrollar. Luego, se especifican cuestiones relacionadas con el diseño de la solución final, al igual que se justifica la elección de las tecnologías utilizadas en el proyecto. Posteriormente, se incluyen ciertas consideraciones sobre la implementación, además de un plan de pruebas que sirva para verificar el buen y óptimo funcionamiento de la aplicación. Por último, se encuentra un manual de usuario fácil de leer y entender, junto con un apartado en el que se relaten las incidencias o conclusiones más significativas surgidas durante el transcurso del proyecto.

### 5.1. ESPECIFICACIÓN DE REQUISITOS DEL SISTEMA

En este apartado se detalla toda la información necesaria para obtener y redactar los requisitos que conforman el sistema. Para ello, se determinan cuáles son los stakeholders, restricciones, casos de uso, etc. Estos son factores imprescindibles a tener cuenta para la obtención del listado final de requisitos (funcionales y no funcionales).

#### 5.1.1. Stakeholders

Tener en cuenta a los stakeholders es una tarea necesaria para identificar y especificar los requisitos del sistema a desarrollar. Este proyecto no tiene un cliente o afectado definido, pero teniendo en cuenta sus objetivos, alcance y el área al que está enfocado podemos identificar 2 tipos de stakeholders: científicos y técnicos (investigadores).

La solución está dirigida sobre todo a los stakeholders previamente mencionados ya que son los que van a utilizar la aplicación para compartir sus investigaciones, algoritmos, artículos... Además, a la aplicación se le podrán ir añadiendo nuevas funcionalidades en el futuro que permitan convertir

## 5.DESARROLLO

la solución final de este proyecto en una plataforma de experimentación, de tal forma que se pueda experimentar y trabajar con los algoritmos que hayan subido al sistema.

No hay que olvidarse también de los usuarios que visiten la página, porque estos también podrán ver y consumir el contenido disponible en la aplicación. No obstante, los visitantes no se han considerado stakeholders debido a que no es un grupo de personas claramente definido y no se ven afectados ni beneficiados de igual manera que los científicos o técnicos.

### 5.1.2. Reglas de negocio y restricciones del proyecto

A continuación aparecen 2 listados que representan el conjunto de reglas de negocio y restricciones vinculadas al proyecto respectivamente.

Reglas de negocio:

1. Para subir contenido al sistema o realizar cualquier otra acción como investigador hay que haber solicitado la creación de una cuenta y esta debe haberla confirmado un administrador web.
2. Para realizar cualquier acción como administrador web hay que disponer de una cuenta administrativa previamente introducida en la BD de forma manual por el administrador de la BD.
3. Para poder mostrar contenido en la página web tiene que haber contenido almacenado en la BD.
4. Para poder eliminar una cuenta de investigador tiene que estar registrada en el sistema.
5. Para eliminar un contenido concreto tiene que estar almacenado en la BD del sistema, al igual que el investigador que lo haya subido.

Restricciones:

1. En el desarrollo de la solución final, se utilizarán tecnologías o herramientas de código abierto.
2. Durante el transcurso del proyecto se utilizarán los medios ofrecidos por parte de la Universidad de Deusto y los propios del alumno, como equipos informáticos, tutorías, etc.
3. Una vez finalizado el desarrollo de la solución final, la información que se almacene en el sistema será accesible para los científicos, técnicos y administradores.
4. Durante la ejecución del proyecto solo habrá comunicación entre el director del proyecto y el alumno, a no ser que se considere de vital importancia consultar a otra persona debido a sus conocimientos sobre un tema concreto o por cuestiones de organización de la asignatura del proyecto de fin de grado.
5. En el proyecto no podrán participar otros alumnos ya que es de carácter individual.
6. La solución final no se podrá vender o publicar sin la autorización de la universidad.
7. La aplicación se implementará teniendo en cuenta que será necesario que se conecte con el servicio de correo electrónico de Google.

### 5.1.3. Modelo de casos de uso

#### 5.1.3.1. Casos de uso

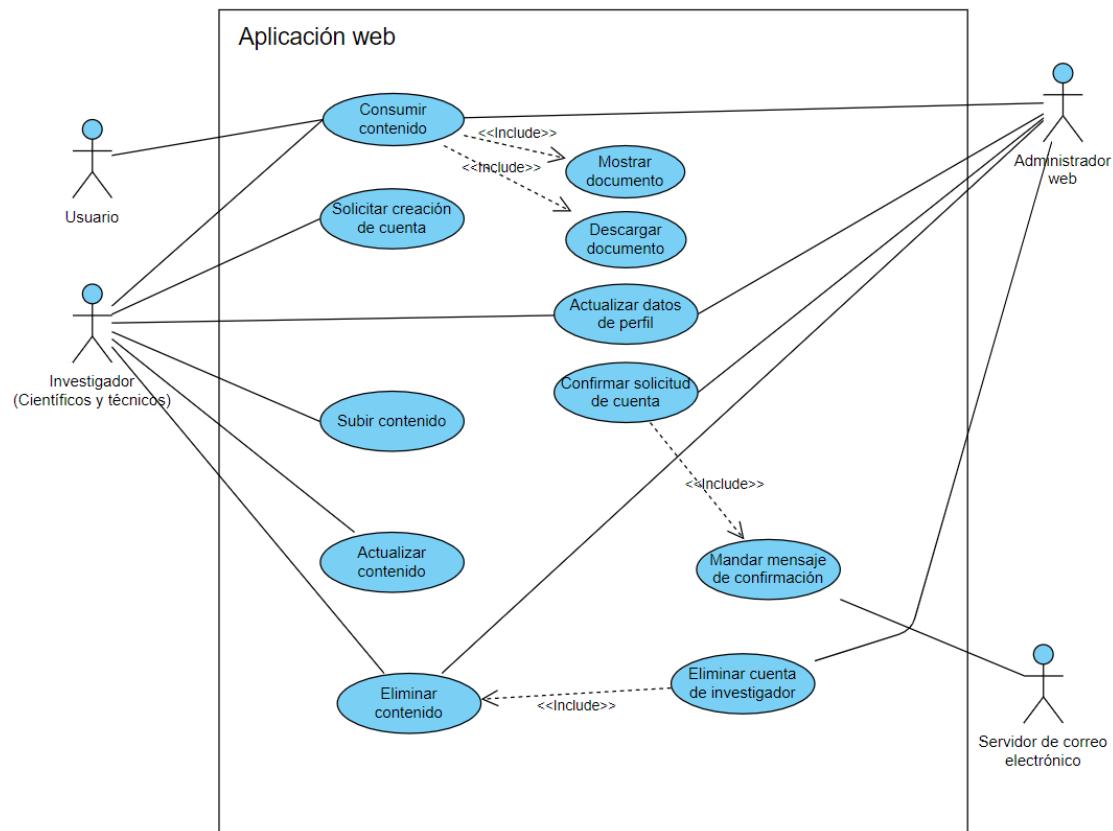


Fig. 5.1: Diagrama de casos de uso

## 5.DESARROLLO

### 5.1.3.2. Especificación de los casos de uso

- PUC1: Consumir contenido.

Propósito: Mostrar documentos, algoritmos, artículos... en la página web, o en su defecto, permitir la descarga de contenido que no se pueda visualizar directamente en el navegador.

Actores: Usuario, investigador y administrador.

Precondición: Acceder a la aplicación desde el navegador web.

Escenario principal:

1. Se comprueba que se cumple la regla de negocio 3.
2. Se muestra el contenido almacenado en el sistema en formato de lista.
3. Se selecciona un documento de la lista.
4. Se comprueba si el documento se puede visualizar.
5. Se muestra el documento seleccionado. Se termina caso de uso.

Variantes:

- 1a) No se cumple la regla de negocio 3. No se puede mostrar contenido alguno. Finaliza caso de uso.
- 4a) El documento no se puede visualizar. Se da la opción de descargar el documento para que el usuario pueda verlo directamente desde su equipo. Se termina caso de uso.

Diagramas:

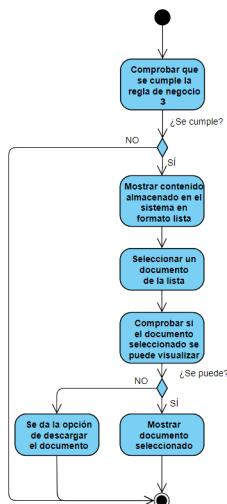


Fig. 5.2: Diagrama de actividad del PUC1

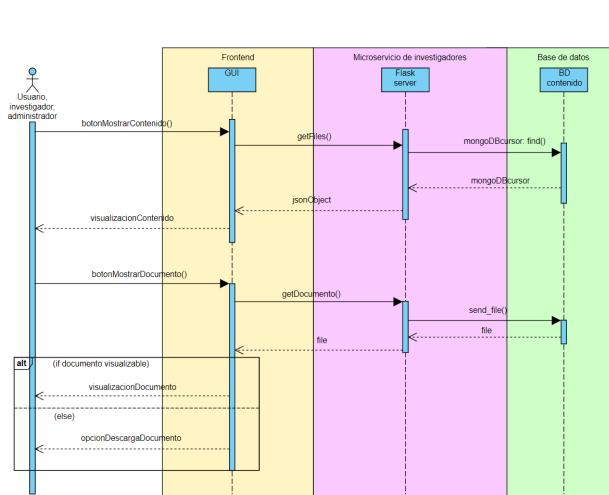


Fig. 5.3: Diagrama de secuencia del PUC1

■ **PUC2:** Solicitar creación de cuenta.

Propósito: Enviar una petición al sistema para crear una cuenta como investigador. Posteriormente, un administrador tiene que confirmar la solicitud.

Actores: Investigador.

Precondición: Haber accedido a la página web y haberle dado al botón *Registro*.

Escenario principal:

1. Se muestra el formulario de creación de cuenta.
  2. Se rellenan los campos del formulario.
  3. Se comprueba si los datos introducidos son válidos.
  4. Se envía la solicitud de creación de cuenta con los datos introducidos en el formulario.
- Se termina caso de uso.

Variantes:

- 3a) Los datos no son válidos. Se cancela el proceso de creación de solicitud. Finaliza caso de uso.

Diagramas:

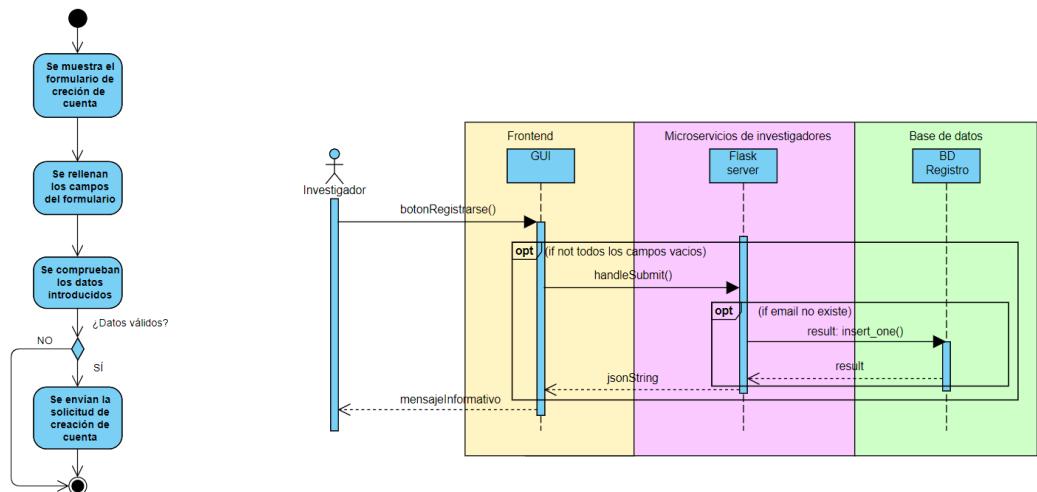


Fig. 5.5: Diagrama de secuencia del PUC2

Fig. 5.4: Diagrama de actividad del PUC2

## 5.DESARROLLO

- **PUC3:** Subir contenido.

Propósito: Compartir artículos, algoritmos, archivos... con la comunidad.

Actores: Investigador.

Precondición: Haber iniciado sesión en la aplicación como investigador y darle al botón *Subir*.

Escenario principal:

1. Se comprueba si se cumple la regla de negocio 1.
2. Se muestra el panel donde subir el archivo que se quiera compartir.
3. Se sube el archivo desde el equipo.
4. Se comprueba si el nombre del archivo es válido.
5. Se confirma la subida del archivo. Se termina caso de uso.

Variantes:

- 1a) No se cumple la regla de negocio 1. No se puede subir contenido. Acaba caso de uso.
- 4a) El nombre no es válido. Se cancela el proceso de subida de contenido. Finaliza caso de uso.

Diagramas:

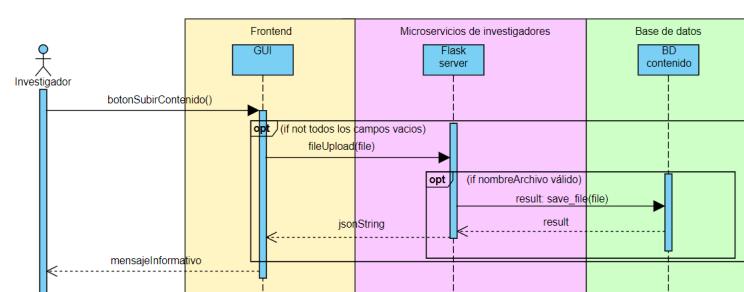
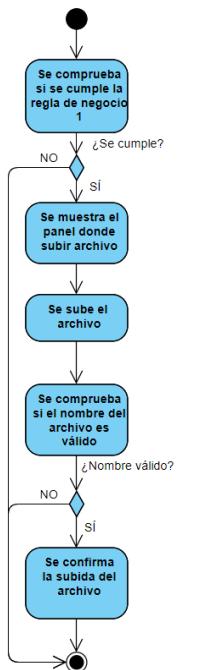


Fig. 5.7: Diagrama de secuencia del PUC3

Fig. 5.6: Diagrama de actividad del PUC3

■ **PUC4:** Actualizar contenido.

Propósito: Modificar el título o nombre de los documentos subidos por el investigador.

Actores: Investigador.

Precondición: Haber iniciado sesión en el sistema como investigador y darle al botón de actualización situado en un documento concreto.

Escenario principal:

1. Se comprueba si se cumple la regla de negocio 1.
2. Se muestra un panel con un formulario y la información actual.
3. Se introduce la nueva información.
4. Se comprueba si la nueva información es válida.
5. Se actualiza el documento en cuestión con la nueva información. Se termina caso de uso.

Variantes:

- 1a) No se cumple la regla de negocio 1. No se puede subir contenido. Acaba caso de uso.
- 4a) La nueva información no es válida. Se cancela el proceso de actualización. Acaba caso de uso.

Diagramas:

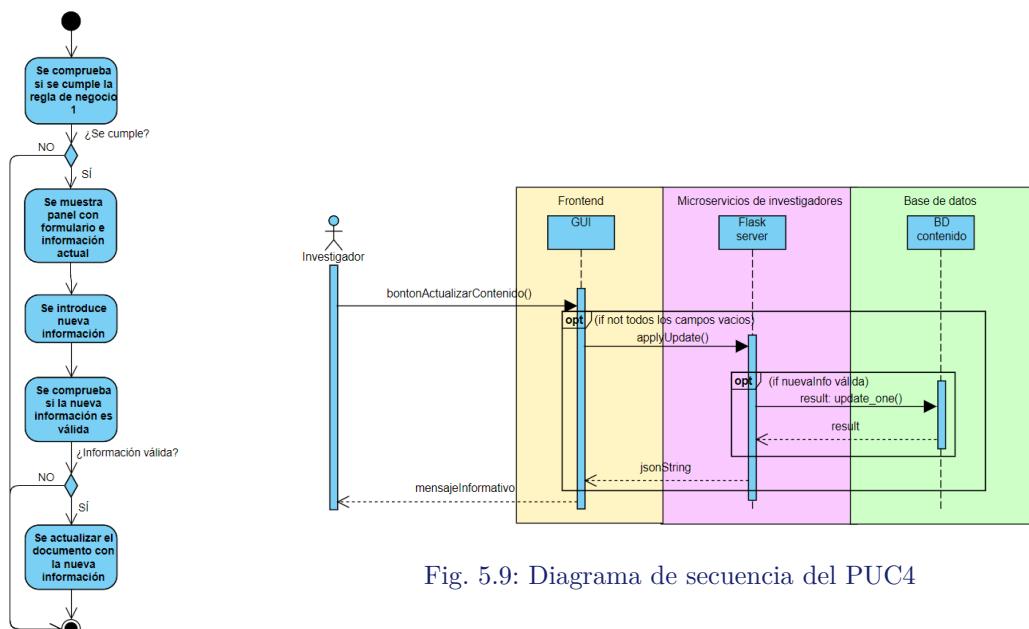


Fig. 5.9: Diagrama de secuencia del PUC4

Fig. 5.8: Diagrama de actividad del PUC4

## 5.DESARROLLO

- **PUC5:** Eliminar cuenta de investigador.

Propósito: Eliminar una cuenta de investigador, al igual que todo el contenido subido por el investigador dueño de dicha cuenta.

Actores: Administrador.

Precondición: Haber iniciado sesión en el sistema como administrador, y darle al botón *Todo el contenido*.

Escenario principal:

1. Se comprueba si se cumple la regla de negocio 2.
2. Se comprueba si se cumple la regla de negocio 4.
3. Se muestran los investigadores registrados en el sistema.
4. Se elige y elimina un investigador junto con todos sus documentos. Se termina caso de uso.

Variantes:

- 1a) No se cumple la regla de negocio 2. Se cancela la eliminación. Finaliza caso de uso.
- 2a) No se cumple la regla de negocio 4. Se cancela la eliminación. Acaba caso de uso.

Diagramas:

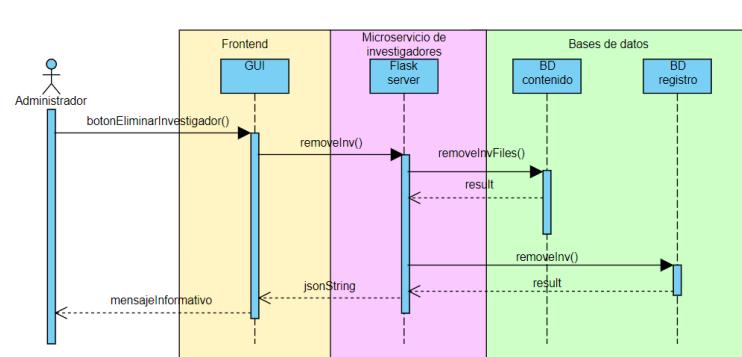
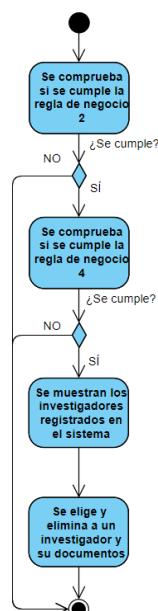


Fig. 5.11: Diagrama de secuencia del PUC5

Fig. 5.10: Diagrama de actividad del PUC5

- **PUC6:** Eliminar contenido.

Propósito: Eliminar documentos que el investigador no quiera que aparezcan en el sistema o que el administrador considere inadecuados.

Actores: Investigador y administrador.

Precondición: Haber iniciado sesión en el sistema como investigador o administrador. Los administradores le tiene que haber dado al botón *Todo el contenido*.

Escenario principal:

1. Se comprueba si se cumple la regla de negocio 5.
2. Se muestra el contenido almacenado en el sistema.
3. Se elige y elimina un documento. Se termina caso de uso.

Variantes:

- 1a) No se cumple la regla de negocio 5. Se cancela la eliminación. Finaliza caso de uso.

Diagramas:

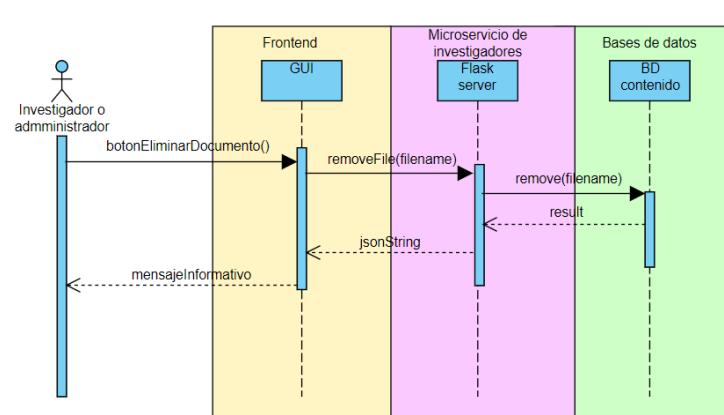


Fig. 5.12: Diagrama de actividad del PUC6

Fig. 5.13: Diagrama de secuencia del PUC6

## 5.DESARROLLO

- **PUC7:** Actualizar datos de perfil.

Propósito: Modificar los datos de una cuenta de investigador o administrador.

Actores: Investigador y administrador.

Precondición: Haber iniciado sesión en la aplicación y darle al botón *Mi perfil*.

Escenario principal:

1. Se comprueba si se cumple la regla de negocio 1 para investigadores o la 2 para administradores.
2. Se muestra un formulario junto con los datos de investigador o administrador actuales.
3. Se llenan los campos del formulario con los nuevos datos.
4. Se comprueba que los datos introducidos son válidos.
5. Se actualizan los datos del investigador o administrador. Se termina caso de uso.

Variantes:

- 1a) No se cumple la regla de negocio 1 o 2. No se puede actualizar. Acaba caso de uso.
- 4a) Los datos no son válidos. Se cancela la actualización. Finaliza caso de uso.

Diagramas:

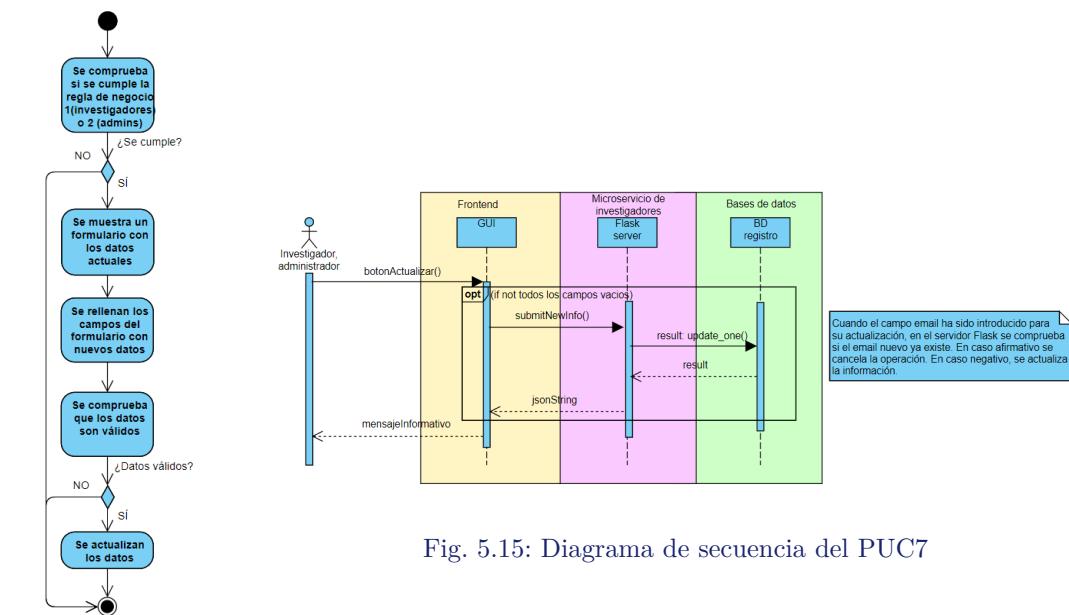


Fig. 5.15: Diagrama de secuencia del PUC7

Fig. 5.14: Diagrama de actividad del PUC7

■ **PUC8:** Confirmar solicitud de cuenta.

Propósito: Permitir que el investigador pueda iniciar sesión con la cuenta que ha solicitado crear.

Actores: Administrador.

Precondición: Haber iniciado sesión en la aplicación como administrador y que el investigador haya realizado una solicitud de creación de cuenta.

Escenario principal:

1. Se comprueba si se cumple la regla de negocio 2.
2. Se muestra la información referente a la solicitud (datos del investigador).
3. Se comprueban los datos para verificar si es un investigador fiable.
4. Se confirma la creación de la cuenta.
5. Se envía un mensaje de correo electrónico informativo. Se termina caso de uso.

Variantes:

- 1a) No se cumple la regla de negocio 2. No se puede aceptar o rechazar ninguna solicitud. Finaliza caso de uso.
- 3a) No es un investigador fiable o es un impostor. Se rechaza la creación de la cuenta. Se envía un mensaje de correo electrónico informativo. Acaba caso de uso.

Diagramas:

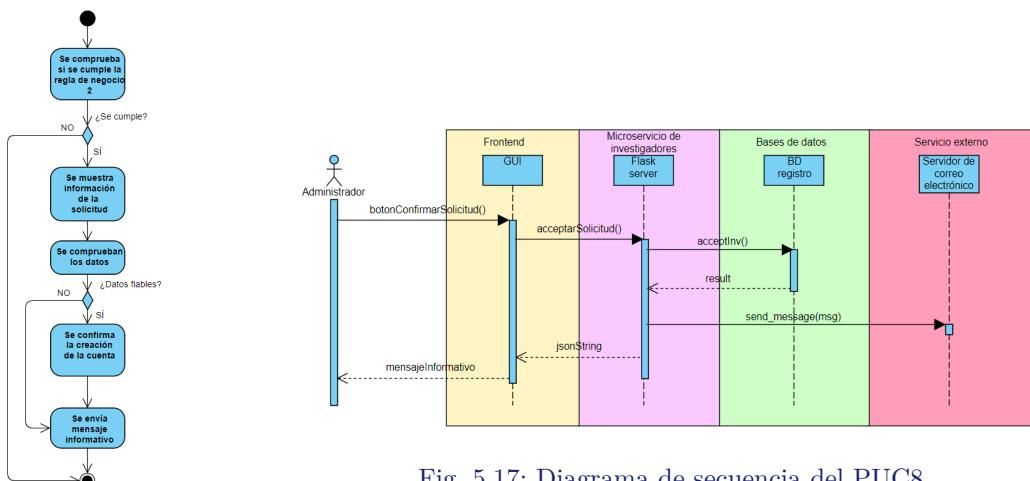


Fig. 5.17: Diagrama de secuencia del PUC8

Fig. 5.16: Diagrama de actividad del PUC8

## 5.DESARROLLO

- **PUC9:** Mostrar documento.

Propósito: Visualizar un documento concreto.

Precondición: Haber seleccionado previamente un documento.

Escenario principal:

1. Se muestra el documento seleccionado.

Diagramas:



Fig. 5.18: Diagrama de actividad del PUC9

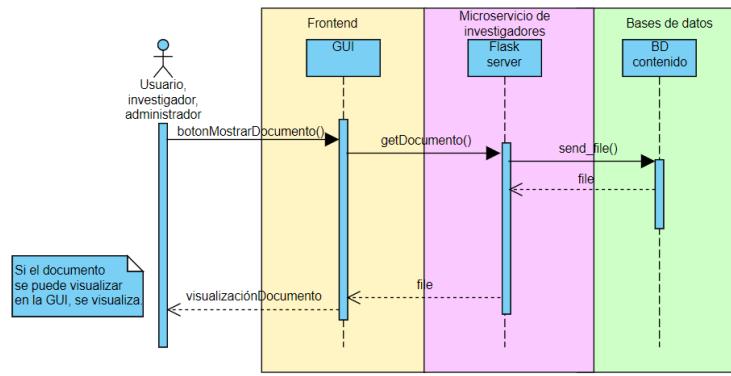


Fig. 5.19: Diagrama de secuencia del PUC9

■ **PUC10:** Descargar documento.

Propósito: Darle la opción al usuario de descargar el documento seleccionado.

Precondición: Haber seleccionado previamente un documento.

Escenario principal:

1. Se da la opción de descarga del documento seleccionado.
2. Se descarga el documento. Se termina caso de uso.

Variantes:

- 1a) El usuario cancela la descarga. No se descarga el documento. Finaliza caso de uso.

Diagramas:

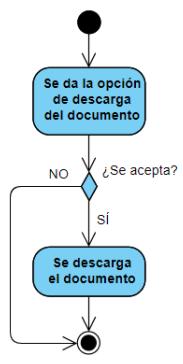


Fig. 5.20: Diagrama de actividad del PUC10

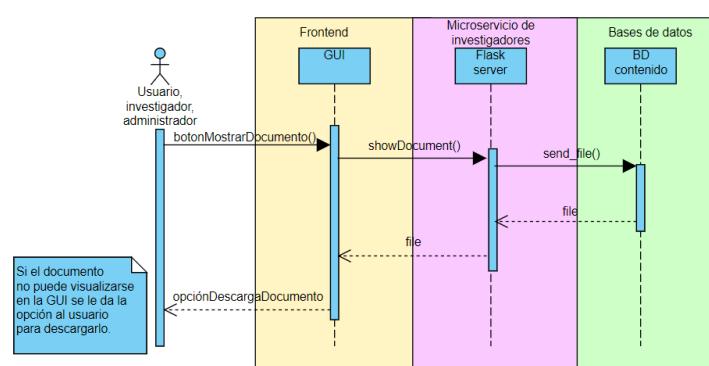


Fig. 5.21: Diagrama de secuencia del PUC10

## 5.DESARROLLO

- **PUC11:** Mandar mensaje de confirmación.

Propósito: Informar al receptor del mensaje si se ha aceptado o no su solicitud de creación de cuenta.

Actores: Servidor de correo electrónico.

Precondición: Se tiene que haber aceptado o rechazado una solicitud.

Escenario principal:

1. Se manda mensaje informativo.

Diagramas:



Fig. 5.22: Diagrama de actividad del PUC11

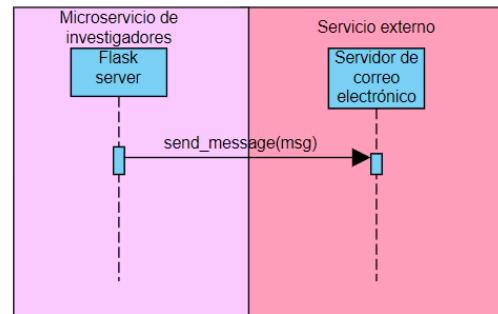


Fig. 5.23: Diagrama de secuencia del PUC11

### 5.1.4. Requisitos

Los requisitos que conforman el sistema podemos diferenciarlos en 2 grandes grupos primordiales: requisitos funcionales y no funcionales. A continuación aparecen listados todos los requisitos del proyecto.

#### 5.1.4.1. Requisitos funcionales

- R1: La aplicación deberá permitir que los usuarios que la visiten puedan consumir el contenido subido y compartido por los investigadores.
- R2: Deberá permitir que los investigadores puedan crearse una cuenta en el sistema.
- R3: Deberá permitir que los investigadores puedan iniciar sesión en el sistema utilizando sus credenciales de acceso definidas en el registro y una vez aceptada la creación de la cuenta por un administrador.
- R4: Deberá permitir que los investigadores puedan subir contenido como informes, algoritmos, imágenes... al sistema.
- R5: Deberá permitir que los investigadores puedan gestionar su propio contenido, ya sea modificando o eliminando documentos.
- R6: Deberá permitir que investigadores puedan cambiar los datos referentes a su cuenta como el nombre de usuario, contraseña o email.
- R7: Se conectará con el servicio de correo electrónico de Google para enviar un mensaje de confirmación cuando se acepte una solicitud de creación de cuenta.
- R8: El sistema deberá guardar las credenciales e información de cada investigador en la BD de registro.
- R9: El sistema deberá almacenar el contenido subido por los investigadores en la BD de contenido.
- R10: Contendrá un panel de administración en el que los administradores podrán iniciar sesión con sus credenciales personales.
- R11: Deberá mostrarles a los administradores un listado de las solicitudes de usuarios que quieren registrarse en el sistema.
- R12: Deberá permitir a los administradores aceptar o rechazar solicitudes de usuarios que quieren crearse una cuenta.
- R13: Deberá permitir que los administradores puedan gestionar el contenido almacenado en el sistema proporcionándoles la capacidad de eliminar cualquier tipo de contenido que consideren inadecuado.
- R14: Los microservicios deberán conectarse con su respectiva BD, con el frontend de la aplicación y entre ellos, si fuese necesario, de manera que puedan gestionar y ejecutar todas las actividades, procesos e intercambios de información que sean necesarios.

## 5.DESARROLLO

### 5.1.4.2. Requisitos no funcionales

R1: Las páginas web de la aplicación deberán ser responsivas, es decir, que puedan visualizarse correctamente desde dispositivos con distintas prestaciones, tamaños de pantallas, etc.

R2: Deberá tener una curva de aprendizaje que no supere la hora.

R3: Las consultas a la BD no deberán tardar más de 2 segundos. Deberá devolver y mostrar el resultado en el tiempo más breve posible.

R4: Deberá tener un buen rendimiento que permita a los usuarios utilizarla sin problemas y sin tiempos de espera innecesarios.

R5: Deberá ser user friendly, es decir, que sea intuitiva y fácil de usar sin ayuda de un manual.

R6: El sistema almacenará las contraseñas y cualquier dato sensible después de haberles aplicado un proceso de hashing.

R7: El sistema mostrará mensajes de error que informen al usuario cuando realice una acción incorrecta o cometa un error.

R8: Las credenciales de acceso de los administradores de la aplicación serán introducidas manualmente en la base de datos por el administrador de la BD.

R9: Deberá desarrollarse aplicando recomendaciones de seguridad que mejoren la seguridad de los datos y del sistema en general.

R10: Deberá realizarse un control y seguimiento del desarrollo de la aplicación.

R11: El frontend de la aplicación solicitará solo los datos requeridos al microservicio correspondiente para evitar el envío de información innecesaria o redundante.

R12: Deberá disponer de medidas de seguridad implementadas mediante diferentes técnicas para evitar posibles vulnerabilidades que puedan ser explotadas por usuarios malintencionados.

R13: Deberá idearse un plan de pruebas que asegure el correcto funcionamiento de la aplicación, así como su facilidad de uso y aprendizaje, su buena usabilidad...

## 5.2. ESPECIFICACIÓN DEL DISEÑO

En este apartado se definen y representan mediante diagramas las distintas cuestiones relacionadas con el diseño utilizado a la hora de crear la aplicación web. Se trata de un punto realmente importante que hay que tener claro antes de comenzar con la fase de implementación, ya que puede evitar o prevenir futuras complicaciones o problemas que puedan surgir.

### 5.2.1. Arquitectura del sistema

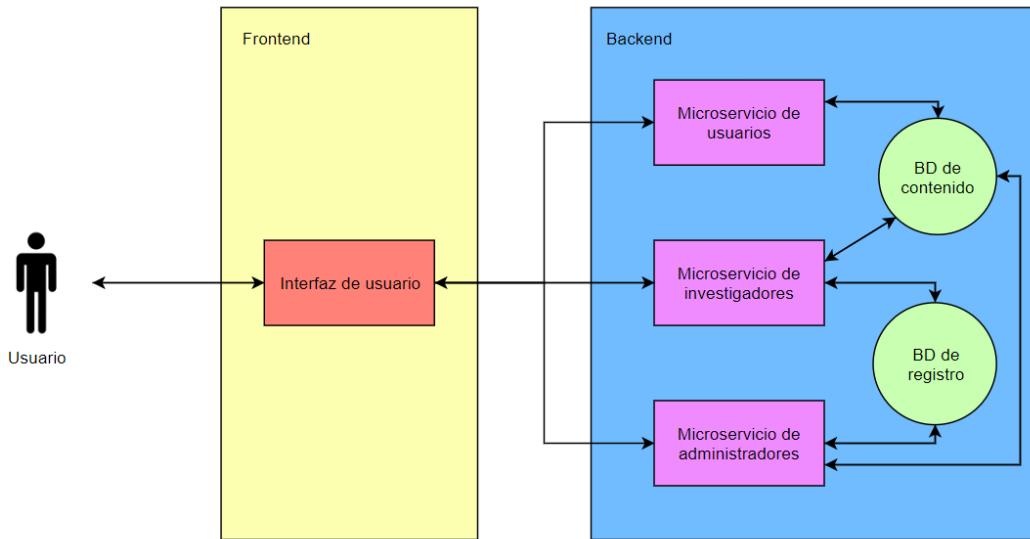


Fig. 5.24: Arquitectura del sistema

### 5.2.2. Descripción de la arquitectura

Como se puede apreciar en el diagrama del apartado anterior, la arquitectura de la aplicación puede dividirse en 2 partes fundamentales. Por un lado se encuentra el frontend, compuesto por una interfaz de usuario con la que interactúa el usuario. Y por otro lado está el backend, formado por una serie de microservicios que se conectan con el frontend y suponen la lógica de la aplicación.

El frontend actúa como un nodo intermedio que conecta la figura del usuario con el backend. El usuario, que en este caso podría ser un investigador, un administrador o un visitante de la página web, puede interactuar con los distintos componentes que forman la interfaz gráfica para así utilizar las funcionalidades disponibles. La interacción entre el usuario y la UI se hace principalmente mediante botones o paneles de información. Cuando se pulsa un botón normalmente es necesario realizar una petición a alguno de los microservicios, depende de cual sea la funcionalidad requerida y del tipo de usuario que esté utilizando la aplicación en cada momento.

El backend recibe las peticiones enviadas desde el frontend y les ofrece respuesta enviando la información o los datos requeridos obtenidos a través de consultas a la base de datos. Podemos diferenciar 3 microservicios distintos:

- **Microservicio de usuarios:** encargado de dar respuesta a las peticiones provenientes del frontend relacionadas con la funcionalidad que permite que los usuarios consuman el contenido subido por los investigadores.

## 5.DESARROLLO

- **Microservicio de investigadores:** se centra en dar respuesta a las peticiones provenientes del frontend vinculadas con la funcionalidad que permite que investigadores puedan registrarse, iniciar sesión, subir y gestionar sus documentos, investigaciones, archivos, etc.
- **Microservicio de administradores:** encargado de dar respuesta a las peticiones provenientes del frontend relacionadas con la funcionalidad que permite que los administradores puedan confirmar el registro de nuevos investigadores y eliminar contenido que consideren inadecuado.

Por último, cabe destacar que el sistema de BD de la aplicación consta de 2 bases de datos:

- **BD de registro:** en esta, se guardan todos los datos referentes a los investigadores y administradores registrados en el sistema, como el email, el nombre de usuario y la contraseña. A esta base de datos se accede desde el microservicio de investigadores y el de administradores.
- **BD de contenido:** en esta, se almacenan los documentos y archivos subidos por los investigadores. A este base de datos se accede desde todos los microservicios.

### 5.2.3. Modelos del diseño

En este subapartado se recogen los diagramas que definen los diferentes aspectos relacionados con el diseño del sistema desarrollado, así como la estructura de las colecciones que componen la BD.

#### 5.2.3.1. Diagrama de clases

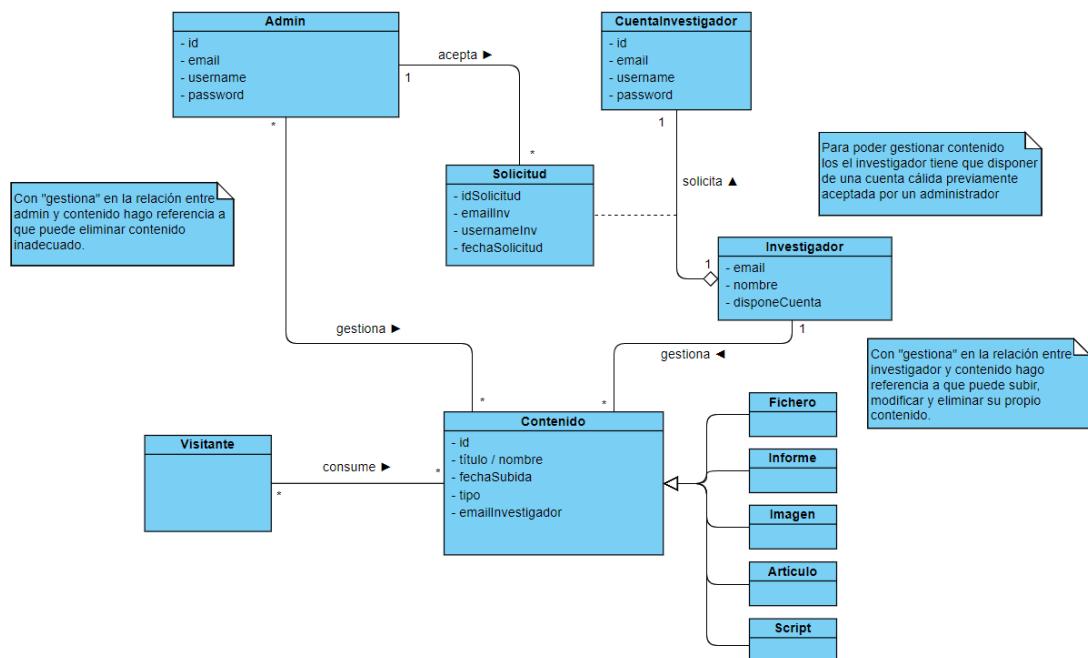


Fig. 5.25: Diagrama de clases

### 5.2.3.2. Formato de las colecciones de las BD

- BD de registro:

- no\_accepted\_invs:

```
{
  "_id" : <ObjectId>,
  "email" : <string>,
  "username" : <string>,
  "password" : <hash>,
  "datetime" : <timestamp>
}
```

- accepted\_invs:

```
{
  "_id" : <ObjectId>,
  "email" : <string>,
  "username" : <string>,
  "password" : <hash>
}
```

- admins:

```
{
  "_id" : <ObjectId>,
  "email" : <string>,
  "username" : <string>,
  "password" : <hash>
}
```

- BD de contenido:

- fs.files:

```
{
  "_id" : <ObjectId>,
  "length" : <num>,
  "chunkSize" : <num>,
  "uploadDate" : <timestamp>,
  "md5" : <hash>,
  "filename" : <string>,
  "contentType" : <string>,
  "aliases" : <string array>,
  "metadata" : <any>,
  "investigator" : <string>
}
```

- fs.chunks:

```
{
  "_id" : <ObjectId>,
  "files_id" : <ObjectId>,
  "n" : <num>,
  "data" : <binary>
}
```

### 5.3. TECNOLOGÍAS UTILIZADAS

Antes de llevar a cabo la implementación de la aplicación y los microservicios, es realmente importante realizar un estudio o comparativa entre las distintas tecnologías disponibles con el objetivo de elegir las que mejores resultados puedan conseguir y más se adecuen a las necesidades del proyecto. Este apartado se centra en definir las tecnologías utilizadas en el desarrollo de la aplicación, además de los motivos que justifican su elección. En total, podemos identificar 5 cuestiones o componentes en los que se ha tenido que elegir entre una tecnología u otra para su implementación:

#### 1. BD

En este caso la elección de la tecnología no ha sido una tarea muy complicada. La duda inicialmente estaba entre si utilizar una base de datos basada en documentos como es *MongoDB*, o una base de datos relacional como *MySQL*. Al final, se eligió *MongoDB* [16], por su gran flexibilidad y escalabilidad a la hora de almacenar y recuperar información [17]. Esto es gracias a que no tiene restricciones en su modelo de datos como otras alternativas *SQL* que sí que lo tienen. Además, puede trabajar tanto con datos estructurados como no estructurados lo que permite almacenar información formada por distintos atributos o campos sin seguir un formato concreto.



Fig. 5.26: Logo de MongoDB<sup>4</sup>.

Hay desarrolladores que afirman que cuando se trata de grandes bases de datos, MongoDB posee una mayor velocidad que otras basadas en *SQL*. En el siguiente gráfico se puede observar como Mongo obtiene mejores resultados que una base de datos MySQL a la hora de realizar consultas *Update* o *Insert*:

---

<sup>4</sup>Fuente: <http://www.formadoresit.es/formacion-en-empresas/formacion/cursos-big-data/mongo-db/>

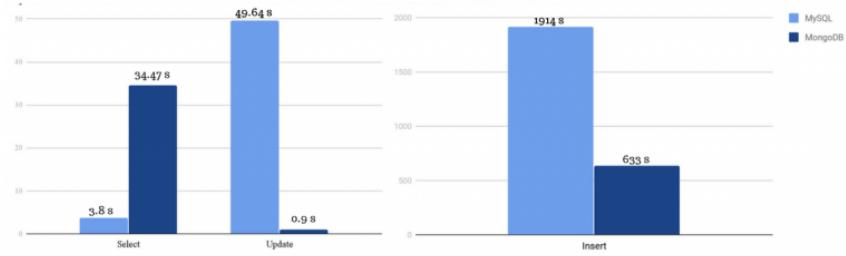


Fig. 5.27: Gráfico comparativo entre la velocidad de MongoDB y MySQL<sup>5</sup>.

Las pruebas realizadas para conseguir los datos mostrados en la gráfica anterior se llevaron a cabo para 1.000.000 de registros, utilizando *MySQL 5.7.9* y *MongoDB 3.2.0*, en una instancia *Amazon m4.xlarge* configurada de forma predeterminada que tenía *Ubuntu 14.4 x64* instalado como sistema operativo [18].

Como la aplicación desarrollada en este proyecto trabaja con documentos (ficheros, imágenes, presentaciones...), MongoDB es realmente útil a la hora de tratar con ellos, ya que dispone de un sistema de gestión de documentos bastante cómodo y fácil de utilizar con el que guardar y recuperar documentos rápidamente. Cabe destacar también que las BD de Mongo es posible replicarlas de tal forma que se creen varias copias de los datos que se almacenen. Así, en caso de error o fallo en la BD principal, es posible recuperar la información perdida. El proceso de replicación por defecto que utiliza *MongoDB* consiste en disponer de 1 base de datos que actúe como principal y el resto como secundarias. La idea es que las lecturas y escrituras se realicen sobre la BD principal para posteriormente, replicarlas en las secundarias [19]. A esto se le llama *replicación maestro-esclavo*.

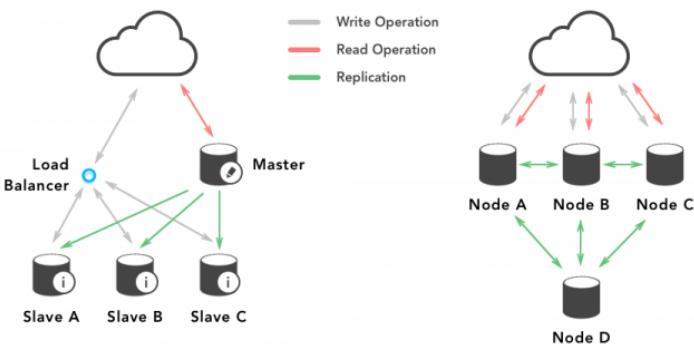


Fig. 5.28: Proceso de replicación maestro-esclavo y maestro-maestro respectivamente<sup>6</sup>.

<sup>5</sup>Fuente: <https://dzone.com/articles/comparing-mongodb-and-mysql>

<sup>6</sup>Fuente: <https://dzone.com/articles/comparing-mongodb-and-mysql>

## 5.DESARROLLO

En definitiva, para la implementación de servicios o aplicaciones que vayan a extenderse muy a menudo, se vayan a desplegar en la nube o vayan a tener una base de datos de gran tamaño a largo plazo, lo recomendable sería utilizar MongoDB porque se podrán obtener mejores resultados que con otras alternativas.

### 2. Frontend

Para la implementación del frontend se decidió utilizar un framework de desarrollo ya que agilizan, sobre todo, las primeras fases del desarrollo. Además, suelen disponer de una gran comunidad en la que apoyarse cuando surje alguna complicación.

A la hora de elegir un framework, el dilema estaba entre Vue.js [20] y React [21], 2 frameworks basados en JavaScript bastante utilizados para la creación de interfaces gráficas de páginas web. Al principio, comencé utilizando Vue ya que es un framework que ha ido ganando popularidad los últimos años, tiene una curva de aprendizaje más sencilla que otras alternativas, un buen rendimiento y requiere de menos líneas de código, lo que hace que se pueda tener un código más limpio. Sin embargo, a la hora de implementar algunas funcionalidades surgieron ciertas complicaciones que suponía una gasto de tiempo excesivo el solucionarlas.



Fig. 5.29: Logo de React<sup>7</sup>.



Fig. 5.30: Logo de Vue<sup>8</sup>.

Es por eso que decidí cambiar de framework y utilizar React. Gracias a este cambio, pude continuar con el desarrollo del frontend sin problemas y sin complicaciones como las que surgieron con Vue. Además, React es un framework que posee una gran flexibilidad debido a que se pueden crear componentes reutilizables, lo que permite extender y optimizar fácilmente el proyecto. Esta última característica, puede ser particularmente útil a largo plazo, ya que si en un futuro se quiere expandir o añadir más módulos al backend se podrá ampliar la UI a la par sin mayor dificultad.

A pesar que Vue.js haya ganado importancia en los últimos tiempos, React ha sido el framework de Javascript líder en el mercado a lo largo de 2018 y 2019, y seguramente lo siga siendo en 2020. Esto se puede ver analizando diversos datos y cuestiones acerca de los frameworks de JavaScript más usados. Entre toda la información que puede encontrarse sobre este tema,

<sup>7</sup>Fuente: <https://betabeers.com/blog/de-backend-a-frontend-reactjs-309/>

<sup>8</sup>Fuente: <https://victorroblesweb.es/2017/04/22/instalar-vuejs-2-hola-mundo/>

se han recogido 2 gráficas en las que se puede ver claramente la gran diferencia que existe entre el uso que le dan los desarrolladores a React respecto a otros frameworks como Vue.js, Angular, etc.

En esta primera gráfica se recogen las descargas por año realizadas utilizando *npm* (el gestor de paquetes de Node.js) para múltiples frameworks de JavaScript:

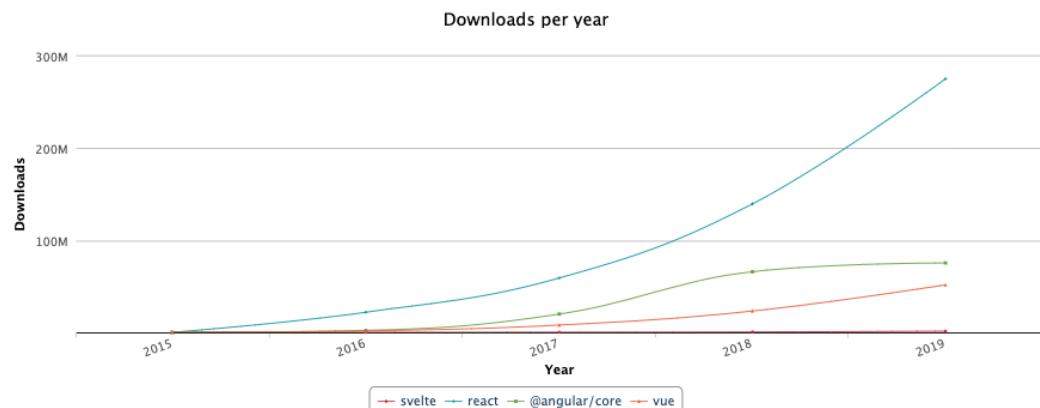


Fig. 5.31: Número de descargas realizadas por año en *npm* desde diferentes frameworks<sup>9</sup>.

En esta otra gráfica se pueden ver el número de veces que se han realizado búsquedas en Google relacionadas con cada uno de los frameworks incluidos en ella:

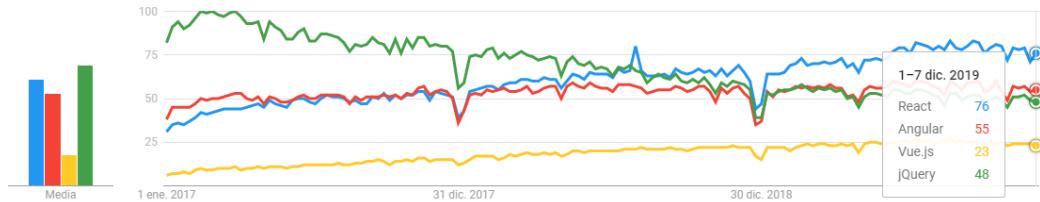


Fig. 5.32: Tendencias en las búsquedas de frameworks en Google<sup>10</sup>.

Gracias a estas 2 gráficas se puede ver la fuerza y el apoyo que tiene React por parte de la comunidad de desarrolladores en general, por lo que se puede considerar como una muy buena opción tanto si se posee experiencia en el desarrollo de aplicaciones como si no, ya que dispone de una amplia comunidad, foros y documentación, así como una dificultad de aprendizaje bastante baja (desde mi experiencia).

<sup>9</sup>Fuente: <https://medium.com/javascript-scene/top-javascript-frameworks-and-topics-to-learn-in-2020-and-the-new-decade-ced6e9d812f9>

<sup>10</sup>Fuente: <https://trends.google.com/trends/explore?date=2017-01-01%202019-12-01&q=%2Fm%2F012l1vxv,%2Fg%2F11c6w0ddw9,%2Fg%2F11c0vmgx5d,%2Fm%2F0268gyp>

### 3. Backend

El backend se ha desarrollado utilizando Python un lenguaje bastante versátil que se utiliza en múltiples áreas como la inteligencia artificial, Big Data, desarrollo web, etc. En el área de los microservicios es uno de los lenguajes más utilizados, junto con un framework minimalista basado en Python llamado *Flask* [22]. Para el desarrollo de los microservicios que conforman el backend del proyecto se ha decidido utilizar Flask porque tiene una arquitectura mucho más simple y ligera que otros frameworks como por ejemplo Django [23], se puede aprender mucho más fácilmente y se pueden crear aplicaciones con pocas líneas de código.



Fig. 5.33: Logo de Flask<sup>11</sup>.



Fig. 5.34: Logo de Django<sup>12</sup>.

Django es una buena opción también para el ámbito de los microservicios. Sin embargo, proporciona y trae consigo una variedad muy amplia de funcionalidades [24], de las que solo unas pocas hubieran sido aprovechadas en caso de haberlo utilizado para el desarrollo del backend de la aplicación. A la hora de implementar microservicios es indispensable que se escriba la menor cantidad de líneas de código posibles, porque se busca conseguir una arquitectura capaz de proporcionar u ofrecer ciertas funcionalidades en el menor tiempo posible. Por lo tanto, es imprescindible escribir un código corto, óptimo y eficiente, además de que el framework o tecnología que se utilice sea lo más ligera posible, como es el caso de Flask.

Flask también es bueno a la hora de construir prototipos rápidos [25] por lo que teniendo en cuenta que para finalizar el proyecto se dispone de un tiempo limitado, puede ser de gran ayuda a la hora de crear un prototipo o una demo sencilla de lo que sería la aplicación web objetivo del proyecto. De esta forma, es posible darle más énfasis y dedicarle más tiempo al desarrollo de la arquitectura basada en microservicios, a su puesta en marcha, a la implementación de características de seguridad para la aplicación y a la documentación del proyecto.

Cabe destacar que aunque Flask parezca un proyecto pequeño, la verdad es que existe la posibilidad de añadirle muchos paquetes y extensiones que le permiten implementar cualquier funcionalidad que se nos ocurra [26], desde *i18n*, gestión de sesiones, hashing de contraseñas, gestión de peticiones entrantes, renderizado de páginas HTML, enrutamiento de URL, etc.

Por diseño, Flask es mucho más flexible que Django a la hora de extender una aplicación, y permite a los desarrolladores implementarla libremente y sin restricciones [27]. Lo más probable es que en un futuro la aplicación haya que extenderla y ampliarla, por lo tanto, teniendo en cuenta lo dicho anteriormente, y que Flask nos permite añadir al proyecto solamente lo necesario para que la aplicación que se esté desarrollando funcione perfectamente, se

<sup>11</sup>Fuente: [https://www.pngitem.com/middle/iRbwTbx\\_flask-python-logo-hd-png-download/](https://www.pngitem.com/middle/iRbwTbx_flask-python-logo-hd-png-download/)

<sup>12</sup>Fuente: <https://www.djangoproject.com/community/logos/>

puede considerar como la mejor opción para el desarrollo de los microservicios del proyecto. Además, resulta ser un framework idóneo para construir microservicios lo menos complejos posibles, consiguiendo así una mayor velocidad y un menor número de vulnerabilidades de seguridad que otras aplicaciones o sistemas más complejos podrían tener.

#### 4. Puesta en marcha de los microservicios

Para ejecutar los microservicios implementados simultáneamente se han tenido en cuenta dos alternativas distintas: despliegue en localhost y despliegue en la nube. Por un lado, para su puesta en marcha en localhost (aunque podría utilizarse para un despliegue público también) se ha elegido una tecnología o herramienta conocida como *Docker* [28], por ser de código libre y permitir la puesta en marcha de múltiples contenedores de software con las prestaciones mínimas necesarias para lanzar un único proceso por contenedor que ejecute el microservicio correspondiente. De esta forma, se logra un menor consumo de recursos, lo que posibilita tener una enorme cantidad de contenedores activos a la vez, así como una mayor velocidad de ejecución y rendimiento a la hora de dar respuesta a las peticiones provenientes del frontend de la aplicación.

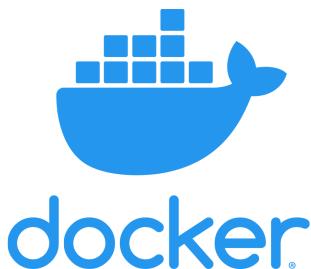


Fig. 5.35: Logo de Docker<sup>13</sup>.



Fig. 5.36: Logo de Virtualbox<sup>14</sup>.

Cabe destacar que la puesta en marcha de los microservicios se podría hacer utilizando también sistemas virtualizados, como máquinas virtuales. No obstante, se considera que los contenedores de software son una mejor opción a la hora de escalar las diferentes partes de la aplicación ya que poseen diversas ventajas sobre los sistemas virtualizados [29]:

- Los contenedores de software se pueden iniciar mucho más rápido que las máquinas virtuales (en unos pocos segundos), mientras que las máquinas virtuales pueden llegar a tardar varios minutos.
- Es posible desplegar las diferentes partes de la aplicación mucho más rápido que las máquinas virtuales ya que simplemente se necesita un Dockerfile con el que crear la imagen correspondiente para lanzar el contenedor. Además, el Dockerfile se puede subir a alguna plataforma desde la que se pueda descargar o acceder a él fácilmente.

<sup>13</sup>Fuente: <https://1000marcas.net/docker-logo/>

<sup>14</sup>Fuente: <https://favpng.com/png/view/virtual-machine-virtualbox-virtual-machine-macos-operating-systems-live-cd-png/0dCHzvu4>

## 5.DESARROLLO

- Se pueden construir y lanzar contenedores desde diferentes máquinas intercambiando entre ellas solamente un fichero de texto (Dockerfile). Para hacerlo con máquinas virtuales, hay que compartir o enviar la máquina virtual completa o su imagen, que suele ocupar varios gigabytes de espacio de almacenamiento.
- Los contenedores se pueden crear o eliminar ejecutando un simple comando, lo que hace que sea mucho más sencillo escalar la aplicación creando nuevas instancias de los microservicios.
- Los contenedores consumen menos recursos del sistema que las máquinas virtuales. Como los contenedores de software son procesos que funcionan con los requisitos mínimos, consumen muy pocos recursos computacionales. Sin embargo, como las máquinas virtuales disponen de su propio sistema operativo virtualizado, son más complejas y grandes por lo que requieren una mayor potencia de computación que los contenedores.

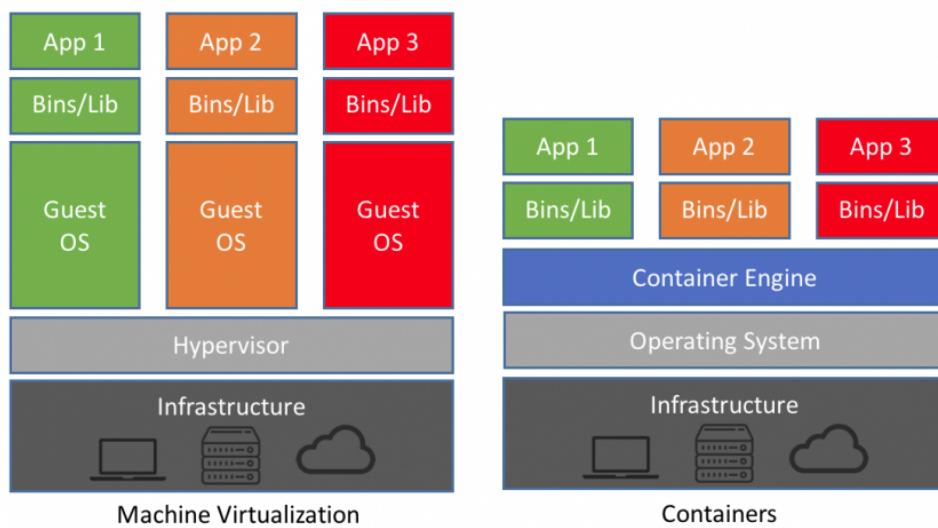


Fig. 5.37: Diferencia entre un contenedor de software y una máquina virtual<sup>15</sup>.  
[Fuente]

Por otro lado, para su puesta en marcha en la nube se ha utilizado una plataforma de computación en la nube basada en contenedores Linux llamada *Heroku* [30] en la que se han podido desplegar tanto los microservicios como la aplicación implementada con React. De esta forma, se puede utilizar la aplicación desde cualquier sitio y de una forma más segura ya que Heroku proporciona a las aplicaciones desplegadas en su plataforma certificados TLS con renovación automática.

<sup>15</sup>Fuente: <https://blog.netapp.com/blogs/containers-vs-vms/>

Fig. 5.38: Logo de Heroku<sup>16</sup>.

Se ha decidido utilizar Heroku para el despliegue en la nube principalmente porque ya se contaba con experiencia previa en la plataforma, y además, las aplicaciones se pueden desplegar simplemente conectando los repositorios de Github correspondientes directamente con Heroku. Este proceso se explica con más detalle en los anexos, concretamente, en el tercer apartado del *Anexo I: Manual de Usuario*.

## 5. Conexión entre los componentes de la aplicación

En este proyecto las 2 conexiones principales que se deben tener en cuenta, son la que se da entre el frontend y los microservicios, y la que se da entre cada microservicio con la BD correspondiente.

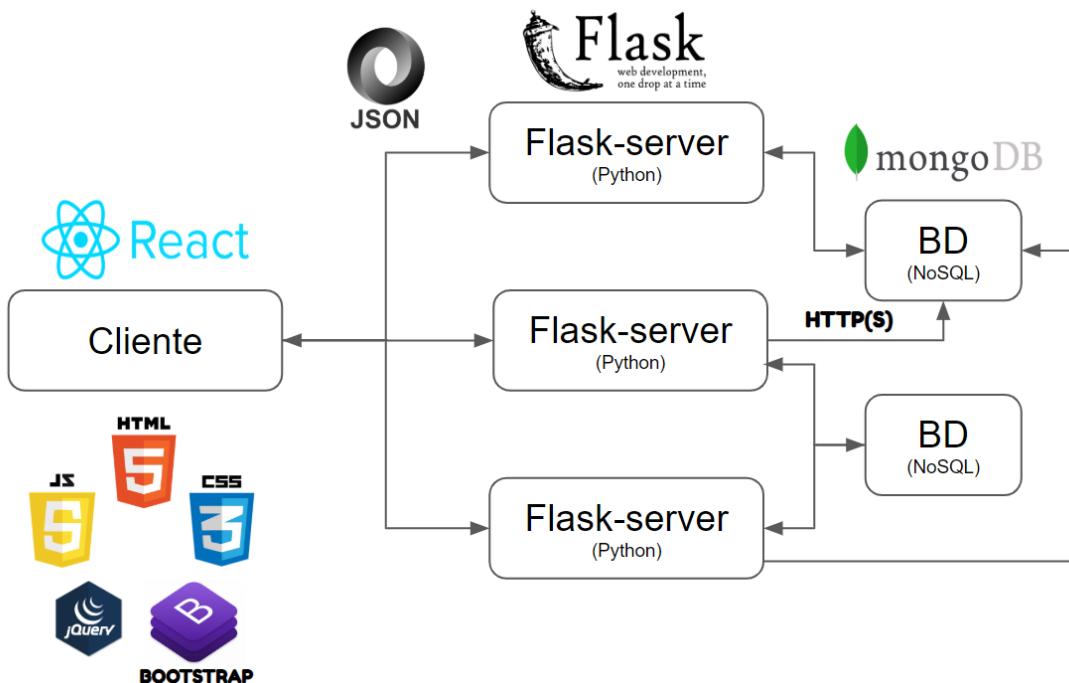


Fig. 5.39: Diagrama de tecnologías de la arquitectura.

Para la conexión del frontend con los microservicios se ha utilizado una librería llamada *Axios* [31], con la que se realizan peticiones HTTP(S) desde la interfaz gráfica hasta el microservicio que proporcione la funcionalidad requerida en cada momento. Cabe destacar que la información requerida por el frontend va incluida en las respuestas enviadas por los mi-

<sup>16</sup>Fuente: <https://estebanromero.com/herramientas-emprender-desarrollar-proyectos/heroku-una-plataforma-para-la-creacion-de-aplicaciones/>

## 5.DESARROLLO

croservicios en formato *JSON* (un formato ligero de intercambio de datos) [32].

Por otro lado, para la conexión entre cada microservicio y la base de datos correspondiente, se ha utilizado una librería de Python, también incluida en Flask, llamada *PyMongo* [33]. Esta librería permite iniciar la conexión con la BD y ejecutar tanto consultas básicas como complejas desde el propio código, mediante una URI o una dirección IP y puerto.



Fig. 5.40: Logo de JSON<sup>17</sup>.



Fig. 5.41: HTTP y HTTPS<sup>18</sup>.

Ejemplo de un documento en formato *JSON* almacenado en la base de datos:

```
{ "_id" : { "$oid" : "5ec0895fce5952e0de26d79e" },
  "email" : "test1@test1.com",
  "username" : "Test1",
  "password" : "$2a$10$pHq8/AxkaqUPtMi0omhSoe0/j6QEyf0Wx10G0a.FgYPMfmu.9p65G"
}
```

## 5.4. CONSIDERACIONES SOBRE LA IMPLEMENTACIÓN

En este apartado se incluyen las consideraciones pertinentes relacionadas con la implementación de la solución final que puedan servir de ayuda a otras personas o alumnos que estén llevando a cabo proyecto similares.

### 5.4.1. Entorno de trabajo

La mayor parte del proceso de desarrollo e implementación de la aplicación web se ha realizado utilizando un editor de textos llamado *Visual Studio Code* [34]. Este editor lo considero mucho más completo y cómodo de utilizar que otros con los que he trabajado anteriormente ya que, aparte de ser visualmente atractivo, ofrece múltiples funciones realmente útiles dirigidas a desarrolladores de software. Entre sus características podemos destacar:

- **El explorador de archivos:** permite tener al alcance todos los ficheros y archivos que necesitemos en cada momento durante el proceso de implementación. Ayuda a ahorrar tiempo y mantener organizado los diferentes tipos de archivos que componen el proyecto.
- **Las extensiones:** permiten añadir funcionalidad extra al editor. Gracias a esto, es capaz de adaptarse a cualquier tipo de proyecto, sin importar el lenguaje de programación que se esté

<sup>17</sup>Fuente: <https://www.olifid.com/?lang=en>

<sup>18</sup>Fuente: <https://posabilities.co.uk/ssl-the-website-padlock/>

usando. Existen extensiones para diferentes lenguajes de programación y aplicaciones como Python, Javascript, Docker, Git, Debuggers...

- **La herramienta de búsqueda:** permite la búsqueda rápida y eficaz de palabras clave en el código. Esta es una funcionalidad común en este tipo de editores pero indispensable para lograr un desarrollo óptimo y veloz. Es por eso que se incluye en este listado.
- **El debugger:** se incorpora la funcionalidad de un depurador en el propio editor con el que poder analizar el código línea a línea con el objetivo de encontrar errores de programación.
- **La consola:** viene con una consola incluida dentro del editor, lo que nos permite ejecutar comandos directamente como si de la consola del sistema se tratara.
- **La compatibilidad con Git:** permite utilizar Git sin la necesidad de salirse del editor.
- **La compatibilidad con Docker:** permite utilizar Docker sin la necesidad de salirse del editor.

Durante toda la implementación se ha utilizado Git [35], un gestor de versiones, con el que se han ido subiendo a GitHub todos los cambios realizados al código fuente de la aplicación. Github [36], a su vez, es un servicio web que permite alojar proyectos utilizando Git de forma gratuita. Además, identificándose como estudiante ofrece funcionalidades extra como estadísticas referentes a la actividad en los repositorios o creación de repositorios privados. Esta herramienta la considero fundamental a la hora de desarrollar cualquier tipo de aplicación o software ya que permite tener un buen control y organización a la hora de gestionar el código fuente. También, actúa como si de una copia de seguridad se tratase, permitiendo recuperar una versión anterior en caso de error o complicación mayor.

Por último, es preciso señalar que como se ha utilizado Docker para el lanzamiento de los microservicios, el proceso de implementación se ha realizado, desde un principio, en un equipo con un sistema operativo Linux, porque para su instalación en una máquina Windows era necesario disponer de *Windows 10 64-bit: Pro, Enterprise, or Education (Build 15063 or later)*. Asimismo, la preparación del sistema para la ejecución de Docker era mucho más simple y rápida en Linux que en Windows.

#### 5.4.2. Implementación de la interfaz de usuario

La interfaz de usuario es una parte fundamental del sistema ya que es la que está en constante interacción con los usuarios. Por lo tanto, hay que implementarla de tal forma que sea intuitiva y fácilmente utilizable. Como se ha dicho anteriormente, la tecnología empleada para el desarrollo del frontend ha sido React.

Sin embargo, antes de decantarme por esta opción comencé la implementación de la UI usando Vue CLI, un sistema de Vue para desarrollos rápidos, con el que creé una primera estructura de proyecto y empecé a implementar diferentes componentes de la página web como el formulario de registro o de login. Debido a que no había trabajado nunca con este framework, tuve algunos problemas al implementar algunas funcionalidades o al escribir el código correspondiente a las

## 5.DESARROLLO

conexiones entre el frontend y los microservicios. Esto hizo que le dedicase demasiado tiempo a la resolución de estos problemas y llegué a la conclusión de que era mejor probar con otro framework en el que me sintiera más cómodo y con el que comprobar si podía obtener mejores resultados.

Es por eso que probé a utilizar React, un framework con el que tuve mejores resultados que con Vue. Es preciso resaltar que yo no estaba acostumbrado a utilizar React, pero al menos lo conocía algo más que Vue. A pesar de no conocerlo bien, conseguí habituarme a él rápidamente y pude implementar las funcionalidades y conexiones en las que tuve problemas previamente mucho más rápido y de forma más amena. Para ello, utilicé una herramienta de React desarrollada por Facebook llamada *create-react-app*, mediante la que pude generar la estructura base del proyecto (sustituta de la primera estructura creada con *Vue.js*). Para ello, fue necesario tener instalado Node.js (un entorno de ejecución de Javascript) [37], además de su gestor de paquetes *npm*.

Esta herramienta la verdad que es de gran ayuda cuando se quiere desarrollar un proyecto nuevo sin perder demasiado tiempo en preparar y organizar su estructura, ya que permite crearlo automáticamente mediante un solo comando. Además, proporciona todo lo necesario para comenzar a programar lo antes posible, sin necesidad de instalar paquetes extras. Su instalación es realmente sencilla. Simplemente hay que ejecutar el siguiente comando en la consola:

```
npm install -g create-react-app (Windows)  
sudo npm install -g create-react-app (Linux)
```

Tras su instalación solo quedaría generar la estructura del proyecto siguiendo las indicaciones que aparecen en la documentación oficial de React [38]. A continuación se muestra la estructura del proyecto obtenida tras finalizar el proceso:

```
my-app :  
    README.md  
    node_modules  
    package.json  
    .gitignore  
    public :  
        favicon.ico  
        index.html  
        logo192.png  
        logo512.png  
        manifest.json  
        robots.txt  
    src :  
        App.css  
        App.js  
        App.test.js  
        index.css  
        index.js  
        logo.svg  
        serviceWorker.js
```

El uso de este tipo de herramientas y frameworks para la implementación de UI son muy beneficiosos para los desarrolladores, sobre todo cuando se quieren agilizar las primeras etapas de la fase de implementación que suelen ser las más tediosas. Además, no se puede olvidar que proporciona una estructura estupenda para tener un orden y control sobre todos los archivos y carpetas con las que trabajemos a lo largo de todo el proyecto.

A la hora de implementar el frontend de una aplicación, React ofrece una gran variedad de posibilidades, módulos y paquetes con los que crear interfaces de usuario de forma sencilla. En el caso de este proyecto, se han necesitado varios módulos con los que implementar diferentes funcionalidades de la aplicación, como por ejemplo: *react-router* para el enrutamiento de la página web, *react-i18next* para aplicar traducciones al texto de las distintas páginas, etc.

Para crear las páginas web de la aplicación se han utilizado tecnologías clásicas como son HTML, CSS y Javascript. Sin embargo, se ha empleado una sintaxis que se recomienda utilizar con React llamada *JSX* [39]. Esta sintaxis es una extensión de JavaScript, por lo que ambas son muy similares. Gracias a ella podemos escribir código HTML para crear páginas web a la vez que trabajamos con variables, variables de estado (*state*), código en JavaScript, componentes, constructores... Además, se pueden importar archivos CSS a las plantillas JSX de tal manera que los estilos que se hayan definido en ellos se apliquen a los componentes HTML correspondientes. Como se puede observar es un lenguaje que combina estas 3 tecnologías, y que permite implementar funcionalidades de una página web a la vez que se trabaja en su diseño y aspecto.

#### **5.4.3. Implementación de los microservicios**

Como bien se ha comentado antes, los microservicios tienen que ser lo más ligeros posibles de tal manera que dispongan de una gran velocidad para responder a todas las peticiones entrantes por parte de los usuarios de la aplicación web. Por eso se utilizó Flask, un framework ligero, minimalista y basado en Python que con pocas líneas de código permite la implementación de un servidor web operativo. Por sus características, Flask es una de los frameworks más utilizados en la programación de microservicios cuando se utiliza Python como lenguaje de programación.

Una arquitectura basada en microservicios se puede implementar de forma que cada microservicio esté escrito en un lenguaje diferente. A pesar de ello, en este caso, todos los microservicios que conforman el backend del proyecto se han implementado con Flask (Python) ya que tenía cierta experiencia con él y era idóneo para construirlos lo más livianos y óptimos posibles.

En lo que respecta a este proyecto, la implementación de los microservicios que forman el backend de la aplicación web se ha llevado a cabo en 3 scripts de Python, uno por cada microservicio implementado. De esta forma, la tarea de realizar búsquedas en el código se puede agilizar, ya que es posible saber donde buscar en todo momento gracias a que el código se encuentra centralizado en 3 puntos. Esto es gracias a que los microservicios no requieren de una cantidad de código demasiado grande, por lo que los scripts no son excesivamente largos o complejos. No obstante, si con el tiempo los scripts se vuelven más y más grandes lo mejor sería dividir el código en diferentes bloques

## 5.DESARROLLO

o archivos para así mantener una mejor organización asegurando que sea fácil realizar tareas de mantenimiento o actualización de código.

Cada uno de los microservicios que forman el backend de la aplicación web se puede considerar como una REST API que ha sido implementada para responder a las peticiones provenientes del frontend enviando en la respuesta una serie de datos o información en formato JSON (habiéndose autenticado con anterioridad), para posteriormente mostrársela a los usuarios, o en su defecto, llevar a cabo algún otro proceso.

### 5.4.4. Características de seguridad implementadas

Durante todo el desarrollo de la aplicación se han ido ideando e implementando medidas de seguridad y prevención de cara a proteger a los usuarios del sistema de ataques informáticos, robos de credenciales, etc. En este proyecto se han implementado múltiples características que aumentan la seguridad de la aplicación y que vale la pena mencionar en este apartado.

Primero, estaría el hashing de contraseñas. Esta característica consiste en aplicarle a las contraseñas una función de hashing antes de almacenarlas en la base de datos. De esta manera, cada contraseña se almacenará como una cadena de caracteres fija sin sentido que no podrá ser revertida con el fin de obtener la cadena original. Gracias a este proceso, aunque alguien consiguiese de alguna forma el hash de una contraseña, no podrá descifrar su significado por lo que le será inútil. A la hora de implementar el sistema de login de la aplicación y comprobar si la contraseña introducida por el usuario se corresponde con el hash almacenado en la BD, pueden usarse funciones que permiten comparar un hash con una contraseña en texto plano obteniendo un valor booleano *true* si coinciden o *false* si no coinciden.

En segundo lugar, se encuentra el JWT Token que, en el caso de este proyecto, se ha utilizado para llevar a cabo un proceso de autorización que permita solo a los clientes, que se hayan autenticado correctamente, acceder a funcionalidades restringidas. Gracias al JWT Token, no es posible acceder a ciertas funcionalidades restringidas del microservicio de investigadores y administradores sin haberse autenticado y haber recibido un token válido que asegure que el cliente que se está conectando es de confianza. En este proyecto, se ha implementando de tal manera que, cuando un investigador o administrador envía sus datos de inicio de sesión, se comprueba en el microservicio correspondiente si son correctos, y en caso de serlo, se le devuelve un JWT Token que, posteriormente, se incluirá en todas las peticiones que envíe el cliente al microservicio mientras mantenga la sesión iniciada. Cualquier petición que solicite alguna de las funcionalidades restringidas, sin incluir en ella el token o incluyendo un token inválido, será rechazada y se le devolverá al cliente un error *401 Unauthorized*.

En tercer lugar, cabe destacar que el sistema de registro de investigadores y el sistema de login de los administradores se ha implementado de manera que tengan que hacerlo en dos pasos. Por un lado, cuando un nuevo investigador quiere crearse una cuenta, envía una solicitud, con todos los datos referentes a la cuenta, al microservicio de investigadores donde se procederá a almacenar dicha solicitud en la BD. Luego, para que el investigador pueda acceder al sistema con su

nueva cuenta, un administrador deberá haber aceptado su solicitud previamente. Por otro lado, cuando un administrador va a iniciar sesión en la aplicación debe introducir su dirección de correo electrónico y contraseña. Además, se le enviará un mensaje a la dirección de correo en cuestión en el que se incluye un código de un solo uso que tendrá que introducir en la aplicación con el fin de confirmar que es la persona que dice ser.

Por último, mencionar que, en la máquina donde se ha servido la aplicación web, se ha instalado y configurado un firewall de tal manera que solo permita el acceso a los puertos del sistema necesarios para el buen funcionamiento del frontend y todos los microservicios. Asimismo, también se intentó obtener un certificado TLS/SSL para funcionar bajo HTTPS. Sin embargo, es necesario que el servidor web disponga de un dominio público para solicitar un certificado oficial, por lo que, en una primera instancia, la aplicación se dejó funcionando bajo HTTP en localhost. Posteriormente, se ideó una alternativa que consiste en desplegar tanto el frontend como los microservicios en un servicio en la nube gratuito que funcione bajo HTTPS, como es *Heroku*. Esta alternativa se explica más adelante, más concretamente en el *Anexo I* de la memoria, en el que se habla sobre el manual de usuario y el proceso a seguir para poner en marcha tanto la aplicación del frontend como los microservicios.

#### **5.4.5. Resumen de las características implementadas**

Este apartado se centra en resumir en un listado todas las características o funcionalidades que se han implementado en la aplicación web final. Se puede dividir en 4 apartados bien diferenciados. Primero, estarían las características enfocadas a los usuarios o visitantes de la página web de la aplicación. Después, se encuentran las funcionalidades que pueden utilizar los investigadores o técnicos que dispongan de una cuenta en el sistema. Luego, las características dirigidas a los administradores que se encargan de gestionar el contenido y los investigadores que se hayan registrado. Y por último, otras funcionalidades más genéricas como internacionalización, características de seguridad, etc.

- **Usuarios:**

- Visualizar el listado completo de documentos o archivos almacenados en el sistema.
- Visualizar cada documento directamente en el navegador, o en su defecto, dar la opción de descarga.

- **Investigadores o técnicos:**

- Sistema de registro en dos pasos.
- Sistema de inicio de sesión (una vez que su cuenta haya sido aceptada por un administrador).
- Subir nuevos documentos o archivos al sistema.
- Visualizar su listado personal de documentos o archivos subidos.
- Eliminar documentos o archivos subidos.
- Cambiar el título o nombre de cualquiera de sus documentos o archivos.
- Modificar los datos de su cuenta, como el nombre de usuario, contraseña o email.

## 5.DESARROLLO

- Visualizar cada uno de sus documentos subidos directamente en el navegador, o en su defecto, dar la opción de descarga.
- Sistema de cierre de sesión.

### ■ Administradores:

- Sistema de inicio de sesión en dos pasos (envío de un código de verificación al correo electrónico del administrador que deberá introducir en la página para poder iniciar sesión correctamente).
- Visualización de un listado de solicitudes de creación de cuenta enviadas por investigadores o técnicos que se quieran registrar en el sistema.
- Aceptar o rechazar solicitudes de creación de cuenta.
- Visualización de un listado completo de investigadores o técnicos, y sus respectivos documentos o archivos.
- Eliminar documentos de investigadores o técnicos.
- Eliminar cuentas de investigadores o técnicos junto con todos sus documentos.
- Sistema de cierre de sesión.

### ■ Otras funcionalidades:

- Base de datos *MongoDB* en la nube con *MongoDB Atlas*.
- Despliegue y puesta en marcha de la aplicación web completa en local mediante *Docker*.
- Despliegue y puesta en marcha de la aplicación web completa en *Heroku* (plataforma en la nube).
- *i18n*, incluyendo 3 idiomas: español, inglés y euskera.
- Hashing de contraseñas a la hora de almacenarlas en la base de datos, JWT Token y aplicación de otras características de seguridad descritas en el apartado anterior.
- Sistema de mensajes que informan a los usuarios si ha ocurrido algún error o no a la hora de realizar cualquier acción en la aplicación.

## 5.5. PLAN DE PRUEBAS

En este apartado se busca especificar las pruebas realizadas al software desarrollado, así como detallar las herramientas o técnicas utilizadas a lo largo del proceso de pruebas que puedan servir de ayuda a otros desarrolladores o estudiantes que estén llevando a cabo proyectos de desarrollo web. Esto es porque las pruebas descritas en este apartado pueden ser aplicadas sin ningún inconveniente a cualquier proyecto de este estilo.

### 5.5.1. Objetivos de las pruebas y criterios de aceptación

El plan de pruebas es una cuestión imprescindible a tener en cuenta en cualquier tipo de desarrollo software ya que gracias a él es posible conocer el estado del producto desarrollado, así como tomar

medidas que solucionen problemas que se hayan detectado durante el proceso. En este proyecto los objetivos de las pruebas son muy variados y se explican próximamente. Sin embargo, podemos identificar algunos puntos clave que merece la pena sintetizar en este subapartado.

El plan de pruebas busca fundamentalmente asegurar que la aplicación en su totalidad (frontend y backend) funcione de la forma esperada y proporcione todas las funcionalidades que se han especificado en los requisitos funcionales y no funcionales del diseño. De la misma forma, se busca garantizar que la aplicación tenga un rendimiento aceptable, así como unas características de seguridad mínimas ante posibles ataques que pueda sufrir. Posteriormente, en la estrategia seguida para el proceso de testeo se detallan los distintos tipos de pruebas que se han realizado, incluyendo en cada una de ellas, una descripción breve y clara de sus objetivos individuales.

Cabe destacar que algunas de las pruebas, como por ejemplo las de rendimiento, se han llevado a cabo en una red privada local o en localhost, por lo que es posible que los resultados no sean 100 % realistas. Sin embargo, el procedimiento seguido para la realización de las pruebas, que se explica en los próximos apartados, aunque se haya llevado a cabo localmente, es aplicable a cualquier servidor web que sea accesible desde internet y que disponga de su propio nombre de dominio y certificado TLS/SSL.

Los criterios de aceptación de las pruebas se basarán en una análisis de los resultados obtenidos en cada tipo de prueba. En dicho análisis, se comprobará si los datos que se han extraído gracias a las pruebas son lo suficientemente válidos y fiables de los que sacar conclusiones útiles que puedan servir para mejorar o conocer la situación actual de la aplicación web. Todo esto se hará teniendo en cuenta los recursos hardware y software que se han tenido disponibles para la realización de las pruebas.

### 5.5.2. Estrategia

La estrategia seguida para el testeo del sistema está formada por un conjunto de pruebas de distinta índole que buscan asegurar que la aplicación funciona todo lo bien que debería. Este es el listado completo de pruebas realizadas:

- **Pruebas generales de funcionamiento:** consisten básicamente en acceder a la aplicación web y utilizarla con total normalidad como si de un usuario se tratase, de manera que se puedan identificar posibles errores o bugs que no hayan sido identificados durante la fase de desarrollo. En este caso se han realizado pruebas con los 3 tipos de usuarios que tiene la aplicación: usuarios (visitantes), investigadores y administradores.
- **Pruebas de rendimiento:** se centran en conseguir información sobre el rendimiento que tienen la aplicación y el servidor web donde se aloja mediante diferentes pruebas con distintos niveles de carga de trabajo.
- **Pruebas de seguridad:** consisten en asegurar que las medidas de seguridad implementadas en el desarrollo de la aplicación funcionan de la forma esperada, de manera que se mantenga segura su propia integridad, al igual que la de los datos con los que trabaja.

## 5.DESARROLLO

- **Pruebas de usabilidad e interfaz de usuario:** permiten hacerse una idea de la facilidad con la que los usuarios pueden utilizar la aplicación, además del grado de dificultad que tiene a la hora de aprender a utilizarla.

### 5.5.3. Pruebas generales de funcionamiento

La pruebas generales de funcionamiento se han enfocado en probar la distintas funcionalidades implementadas en la aplicación. Para ello, con los 3 microservicios operativos, se ha accedido a la página web y se ha ido interactuando con ella de diferentes formas actuando como si de un visitante normal, investigador o administrador se tratara. De esta forma, se ha podido ver si todo funciona bien o hay algún componente que falla.

Primero, como visitante, simplemente se pueden visualizar los documentos o archivos subidos por los investigadores, por lo que se ha comprobado que se solicitan y se muestran los datos en la página principal nada más entrar. Además de esto, se ha verificado si es posible visualizar los documentos dándole al botón *Visualizar* ligado a cada uno de los documentos mostrados. En caso de que, debido al formato del archivo no se pueda mostrar directamente en el navegador, se da la opción de descarga para así poder visualizarlo mediante un programa externo.

En segundo lugar, como investigador, por un lado se tiene que testear diferentes aspectos relacionados con el sistema de registro e inicio de sesión, y por otro lado, funcionalidades a las que pueden acceder una vez iniciada su sesión. Respecto al registro hay varias cuestiones que hay que comprobar para asegurar que se llevan a cabo satisfactoriamente:

- La gestión de los datos introducidos en el formulario de registro para solicitar la creación de una nueva cuenta.
- La creación de un nuevo documento en la base de datos con los datos de la solicitud.
- La generación de un mensaje informativo tras haber registrado la solicitud o en caso de haber ocurrido algún error.

Una vez que un administrador haya aceptado o rechazado la solicitud del investigador, hay que comprobar que se recibe un mensaje de correo electrónico informativo que comunica la decisión tomada. Posteriormente, si la cuenta ha sido aceptada se podrá probar el sistema de login iniciando sesión con el email y contraseña introducida en el formulario de registro.

Una vez iniciada la sesión, hay que probar varias funcionalidades:

- **Listado de archivos:** confirmar que se visualizan todos los archivos subidos por el investigador que ha accedido al sistema. Es importante que solo se muestren los suyos propios y no los de otros investigadores.
- **Subida de archivos:** verificar que los archivos subidos por los investigadores sean gestionados y almacenados correctamente tras su subida, aceptando archivos con extensiones compatibles con el sistema.

- **Actualización del perfil:** comprobar que el formulario para actualizar los datos de la cuenta de investigador gestiona los datos de la manera esperada, y que posteriormente, se modifican todos los documentos necesarios en la base de datos.
- **Modificación del nombre de archivos:** al igual que la actualización del perfil de investigador, sería aplicar unas comprobaciones similares cuando se modifica el nombre de uno de los documentos ya registrados en el sistema.
- **Eliminación de archivos subidos:** verificar si al darle al botón con el símbolo de una papelera vinculado a un documento, este se elimina correcta y completamente de la base de datos. Es fundamental que se elimine exclusivamente el documento “seleccionado” por el investigador y no cualquier otro.

Por último, tendríamos las funcionalidades que pueden disfrutar los administradores. Entre ellas, podemos encontrar el sistema de login, similar al de los investigadores. Cabe destacar que la cuenta de administrador habría que crearla manual y directamente en la base de datos. Además, otras funcionalidades que habría que testear serían las siguientes:

- **Listado de solicitudes:** comprobar si se solicitan y muestran correctamente todas las solicitudes registradas en la base de datos.
- **Aceptar o rechazar solicitud:** verificar que se elimina la solicitud de la base de datos al rechazarla, y en caso contrario, crear la cuenta de usuario en la base de datos, además de eliminar la solicitud.
- **Listado de contenido:** asegurar que se solicita y muestra en un listado todas la información relativa a los investigadores registrados junto a todo el contenido subido por estos. En este caso se muestra todo los contenidos de la base de datos.
- **Eliminar documento o cuenta de investigador:** testear el proceso de eliminación de un documento sería parecido a la funcionalidad que tienen los investigadores. Sin embargo, eliminar una cuenta de investigador tiene algo más de complejidad por lo que al darle al botón hay que asegurarse de que se borran todos los documentos subidos por él, al igual que la propia cuenta.

#### 5.5.4. Pruebas de rendimiento

Para este tipo de pruebas se ha utilizado un programa de Apache llamado *ApacheBench* que permite el envío de múltiples peticiones a un servidor web con el fin de probar su capacidad para darles respuesta. El programa da la opción de fijar el número y la concurrencia con la que se envían las peticiones por lo que es posible decidir a cuanta carga someter al servidor dependiendo de sus características.

Para automatizar el proceso de testeo, he creado un script de bash que ejecuta varios comandos de *ab* (*ApacheBench*) probando con distintos valores en las opciones que definen el número de peticiones y la concurrencia. El contenido del script aparece a continuación:

## 5.DESARROLLO

```
#!/bin/bash

for n in 200 400 600 800 1000
do
    for c in 1 10 50 100
    do
        ab -n $n -c $c -s 10 -g ./csv/rendimiento-$n-$c.csv <url>
    done
done
```

Opciones utilizadas en el comando ab:

- **n:** Número de peticiones totales.
- **c:** Concurrencia. Número de peticiones que se enviarán a la vez. Por ejemplo, si n tiene un valor de 200 y c un valor de 10, se irán mandando peticiones de 10 en 10 hasta haber enviado 200.
- **s:** Timeout. Número de segundos que esperará a recibir respuesta en caso de no estar recibiendo la.
- **g:** Gnu-file. Genera un archivo compatible con gnuplot que recogerá los resultados obtenidos por ab en la ruta especificada.

Es preciso decir que “<url>” hay que sustituirlo por la dirección o la IP donde esté alojada la aplicación web.

El script anterior ha sido utilizado para generar archivos csv donde se registran los resultados que se van obteniendo. Posteriormente, gracias a una herramienta llamada *gnuplot* se han creado gráficos con los que se pueden analizar de manera más sencilla los datos guardados en los archivos csv. Para ello, es necesario crear unos archivos .p donde se especifican ciertas configuraciones necesarias para que gnuplot pueda hacer el trabajo de crear los gráficos. El contenido de estos archivos (.p) es como se muestra a continuación:

```
set terminal png size 600
set output "resultados-200.png"
set title "200 peticiones"
set size ratio 0.6
set grid y
set xlabel "peticiones"
set ylabel "tiempo de respuesta (ms)"
plot "../csv/rendimiento-200-1.csv" using 9 smooth sbezier with
    lines title "1 c", \
    "../csv/rendimiento-200-10.csv" using 9 smooth sbezier
    with lines title "10 c", \
```

```
"../csv/rendimiento-200-50.csv" using 9 smooth sbezier
    with lines title "50 c",
"../csv/rendimiento-200-100.csv" using 9 smooth sbezier
    with lines title "100 c"
```

Algunas de las configuraciones que se definen dentro de estos archivos son, el tamaño o el nombre de la imagen que se va a generar, el título o el tamaño del gráfico incluido en la imagen, etc. Asimismo, en las últimas líneas se definen las rutas correspondientes a los archivos csv creados previamente con la herramienta ab.

En este proyecto, he realizado pruebas de rendimiento por un lado, del frontend, y por otro lado, de uno de los microservicios que forma parte del backend. Como los microservicios han sido desarrollados y desplegados utilizando las mismas tecnologías tendrán un rendimiento similar, es por ello que solo he realizado las pruebas sobre el microservicio de usuarios.

A continuación se incluyen todas las gráficas obtenidas a partir de las pruebas de rendimiento ejecutadas.

### Gráficas de las pruebas hechas al frontend

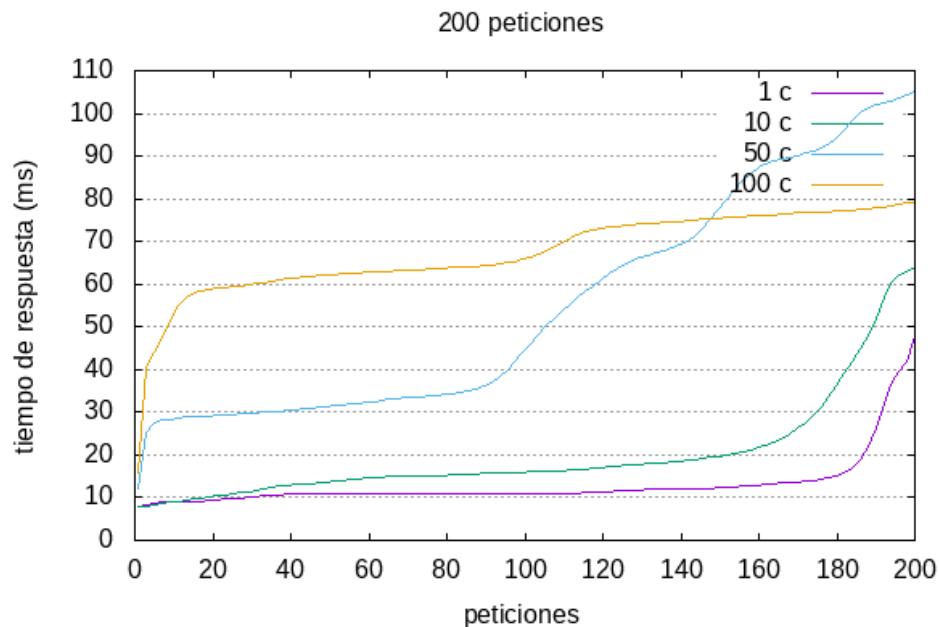


Fig. 5.42: Rendimiento del frontend - 200 peticiones

## 5.DESARROLLO

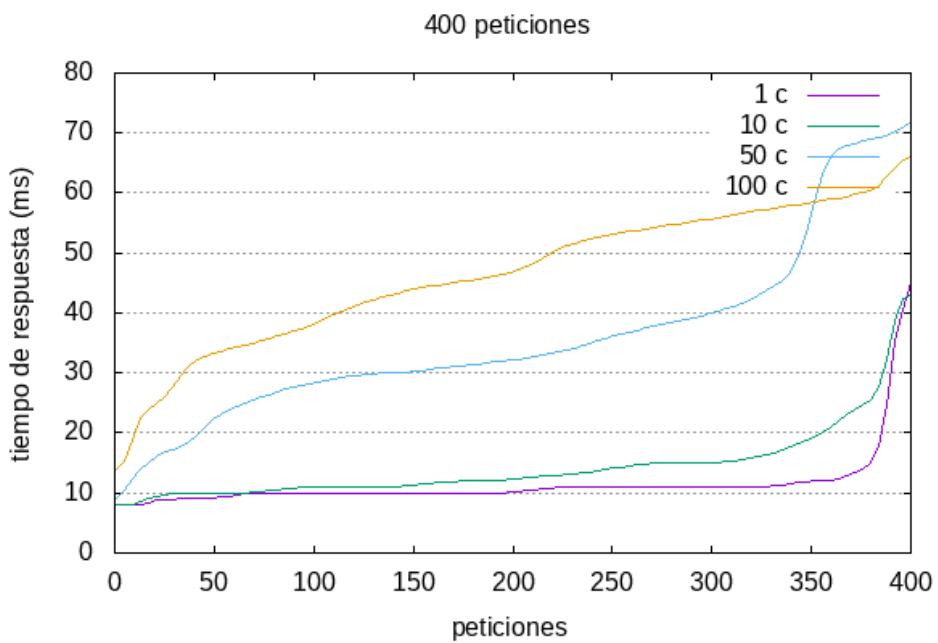


Fig. 5.43: Rendimiento del frontend - 400 peticiones

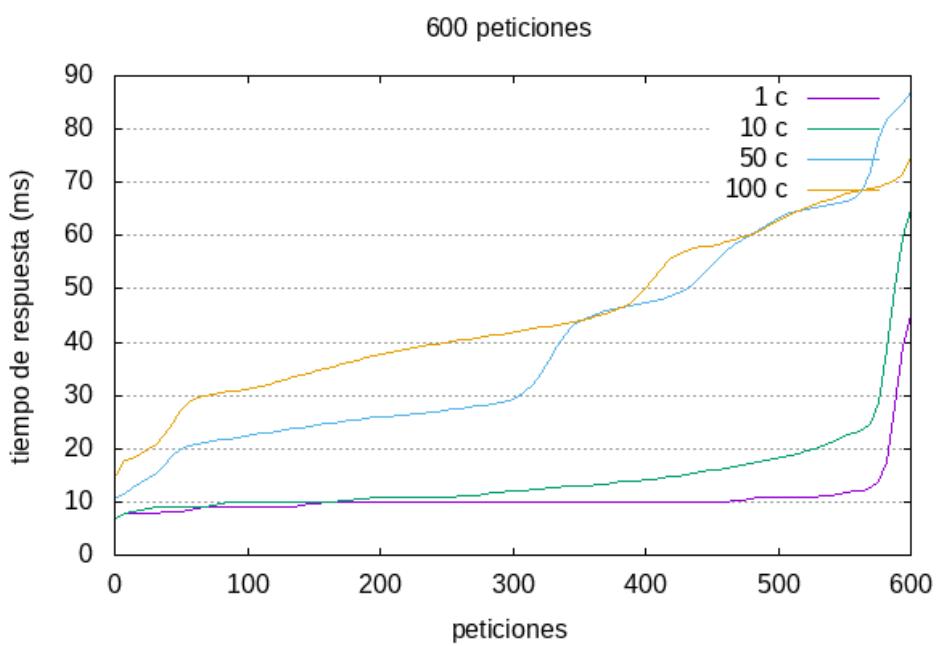


Fig. 5.44: Rendimiento del frontend - 600 peticiones

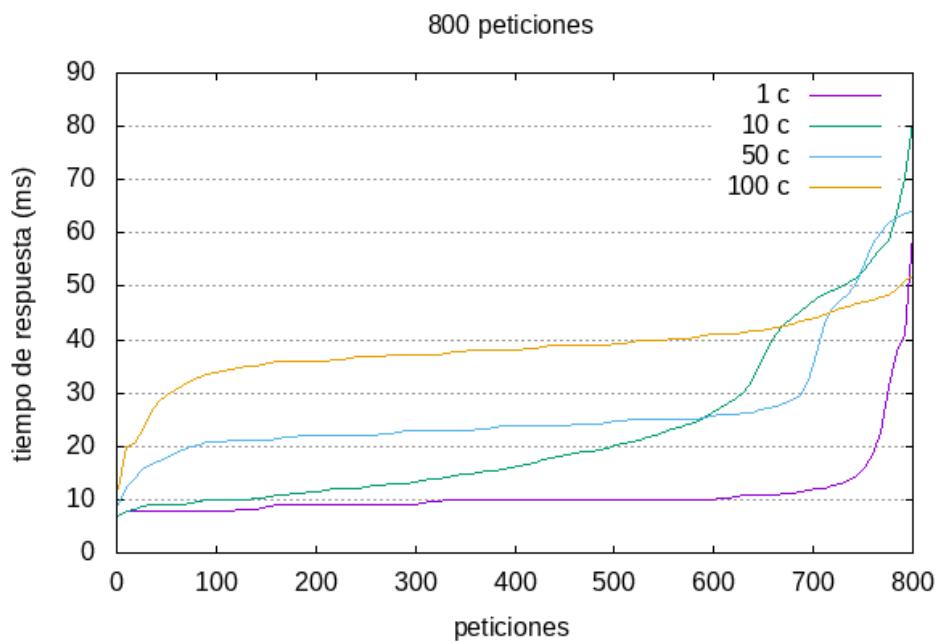


Fig. 5.45: Rendimiento del frontend - 800 peticiones

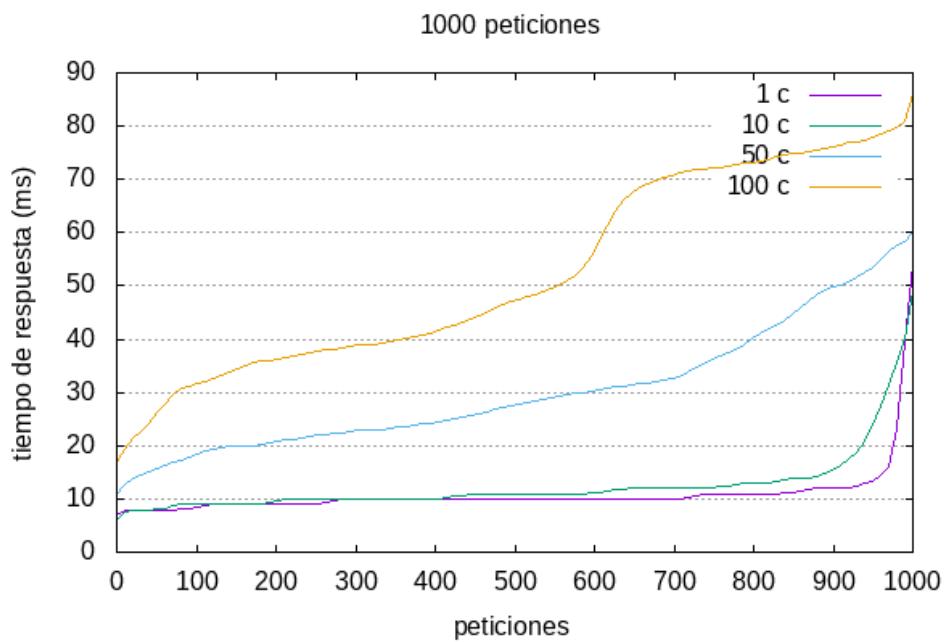


Fig. 5.46: Rendimiento del frontend - 1000 peticiones

## 5.DESARROLLO

### Gráficas de las pruebas hechas al microservicio de usuarios

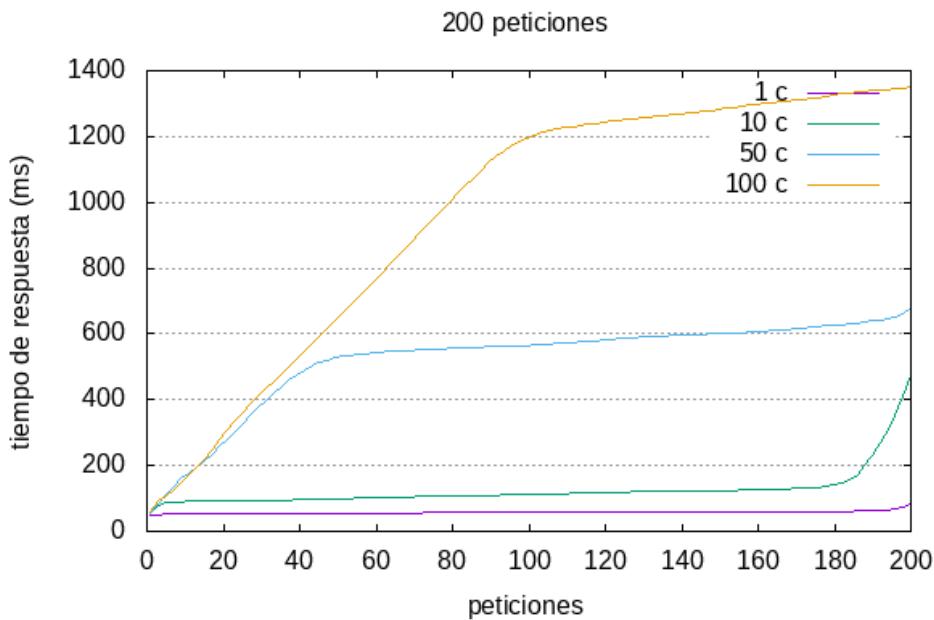


Fig. 5.47: Rendimiento de un microservicio - 200 peticiones

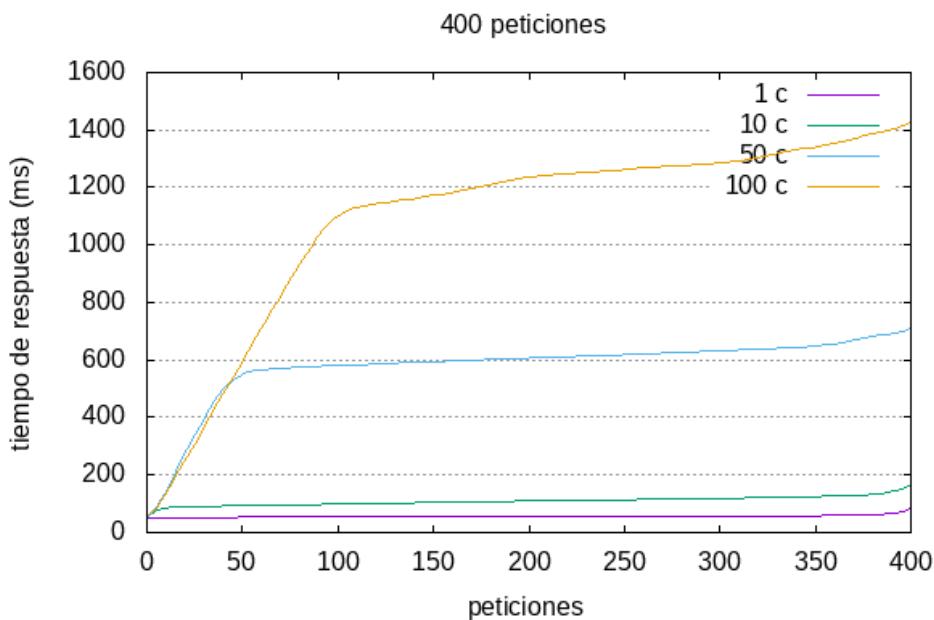


Fig. 5.48: Rendimiento de un microservicio - 400 peticiones

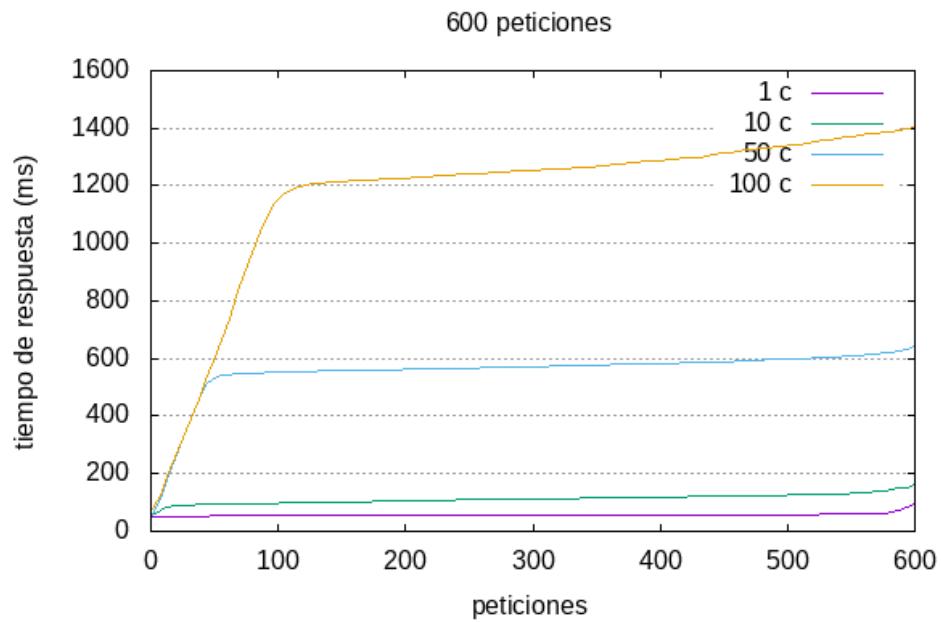


Fig. 5.49: Rendimiento de un microservicio - 600 peticiones

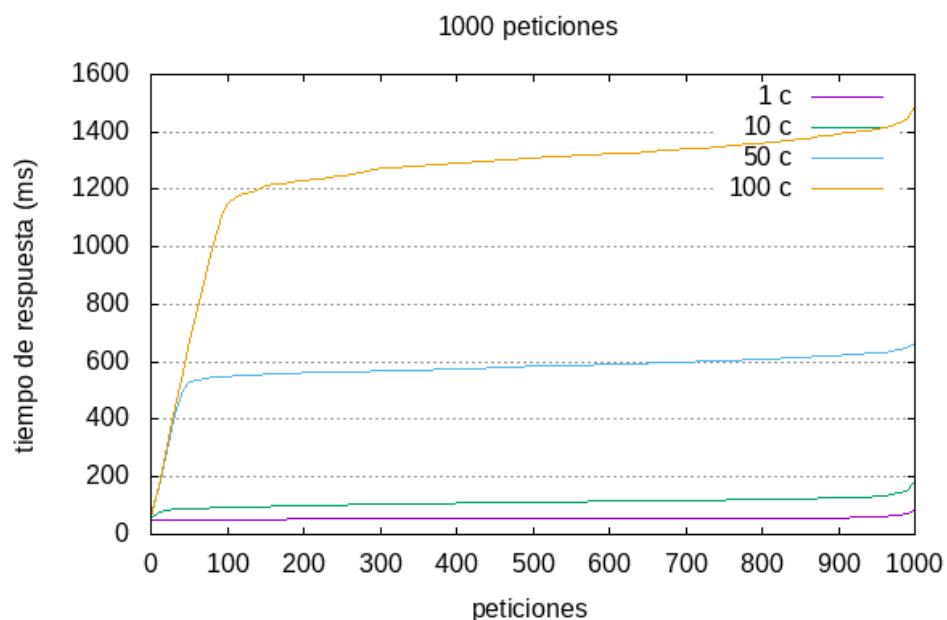


Fig. 5.51: Rendimiento de un microservicio - 1000 peticiones

## 5.DESARROLLO

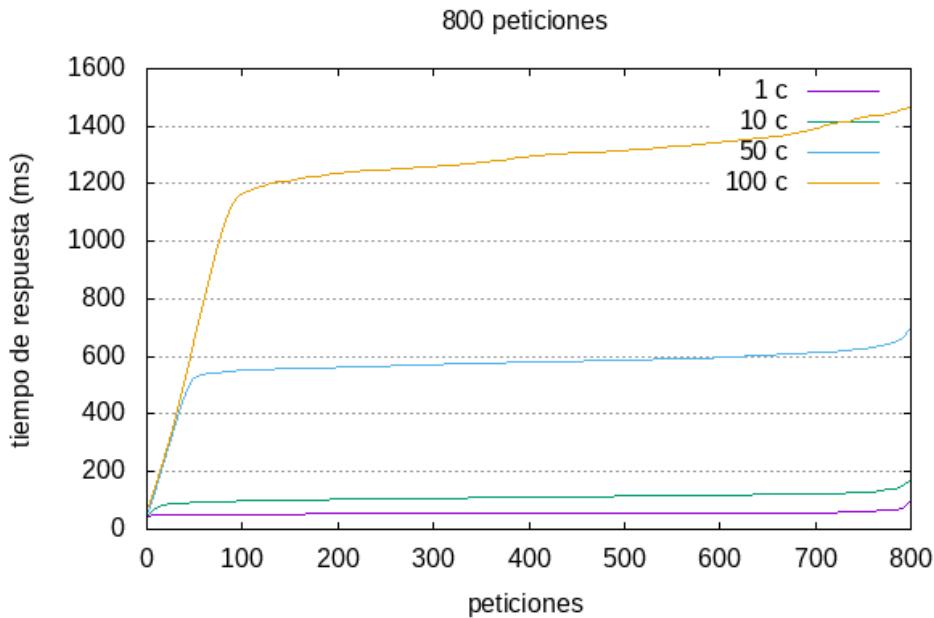


Fig. 5.50: Rendimiento de un microservicio - 800 peticiones

### 5.5.5. Pruebas de seguridad

Para asegurar la seguridad del servidor y la aplicación web, se han realizado múltiples pruebas de seguridad utilizando una serie de herramientas desarrolladas para ese fin. A continuación se describen los resultados obtenidos tras las pruebas, además de soluciones y explicaciones acerca de algunos problemas detectados por dichas herramientas.

En proyectos de desarrollo software es fundamental realizar comprobaciones de seguridad durante, y una vez concluido la fase de desarrollo e implementación. De igual manera, es muy útil y beneficioso para este tipo de pruebas la utilización de varias herramientas o técnicas con las que analizar la aplicación ya que algunas detectan ciertos problemas o posibles vulnerabilidades que otras no son capaces de identificar. En este proyecto se han utilizado 3 herramientas con las que se han analizado la aplicación en busca de vulnerabilidades como SQL Injection, Cross-Site Scripting (XSS), revelación intencionada de información sensible, ejecución de comandos, CRLF Injection, entre otras [40–42].

#### 5.5.5.1. Vega Scanner

Vega Scanner es una herramienta de código abierto escrita en Java que permite analizar y testear la seguridad de un sitio web de una forma sencilla. Para utilizarla, simplemente hay que descargar la versión compatible con el sistema operativo que se esté utilizando y lanzar un análisis definiendo

la IP o dominio del servidor web objetivo. Además, como dispone de una interfaz gráfica para su uso, todo el proceso de análisis y visualización de resultados se lleva a cabo de forma automática y sin complicación alguna.

En este caso, a la hora de lanzar el análisis se han seleccionado todas las vulnerabilidades del listado que ofrece la herramienta, para así que sea lo más completo posible. Tras un primer análisis con Vega, se han obtenido los siguientes resultados:

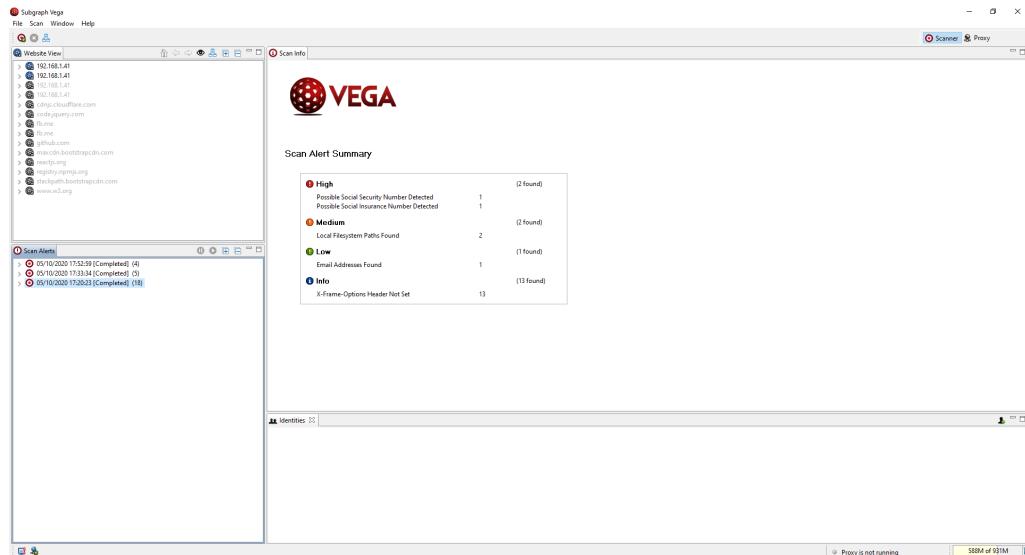


Fig. 5.52: Vega Scanner - Primer escaneo

A primera vista puede parecer que hay bastantes problemas, y puede asustar el hecho de que haya detectado 2 vulnerabilidades severas o de gravedad alta. Sin embargo, si se investiga más a fondo en que consisten y la localización en las que han sido detectadas, se puede observar que ambas son un falso positivo ya que los números que han hecho saltar la alarma no son realmente números de la seguridad social. Pasa algo parecido con la vulnerabilidad leve relacionada con las direcciones de correo electrónico. Estos falsos positivos pueden darse muy a menudo cuando Vega detecta números o cadenas de caracteres con el mismo patrón que los números de la seguridad social o emails, como es en el caso de las próximas 3 capturas:

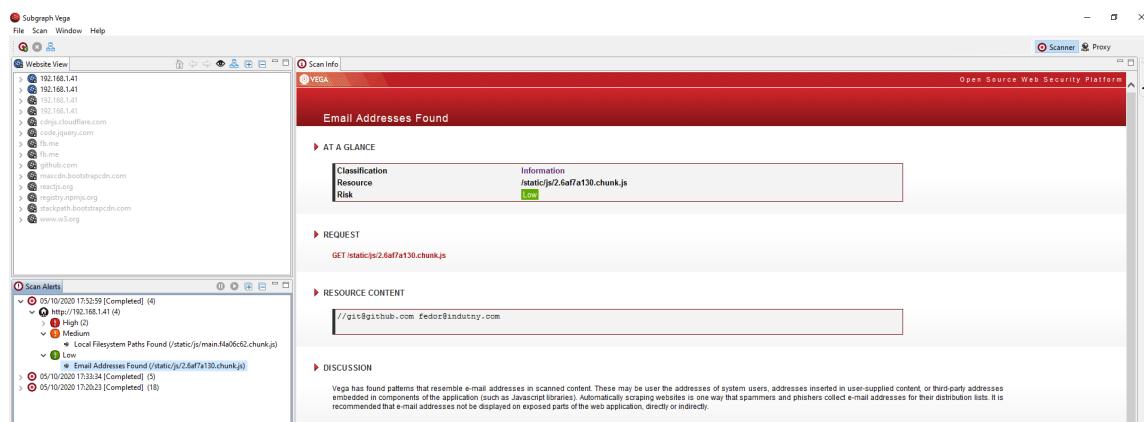


Fig. 5.53: Vega Scanner - Problema leve (Falso positivo)

## 5.DESARROLLO

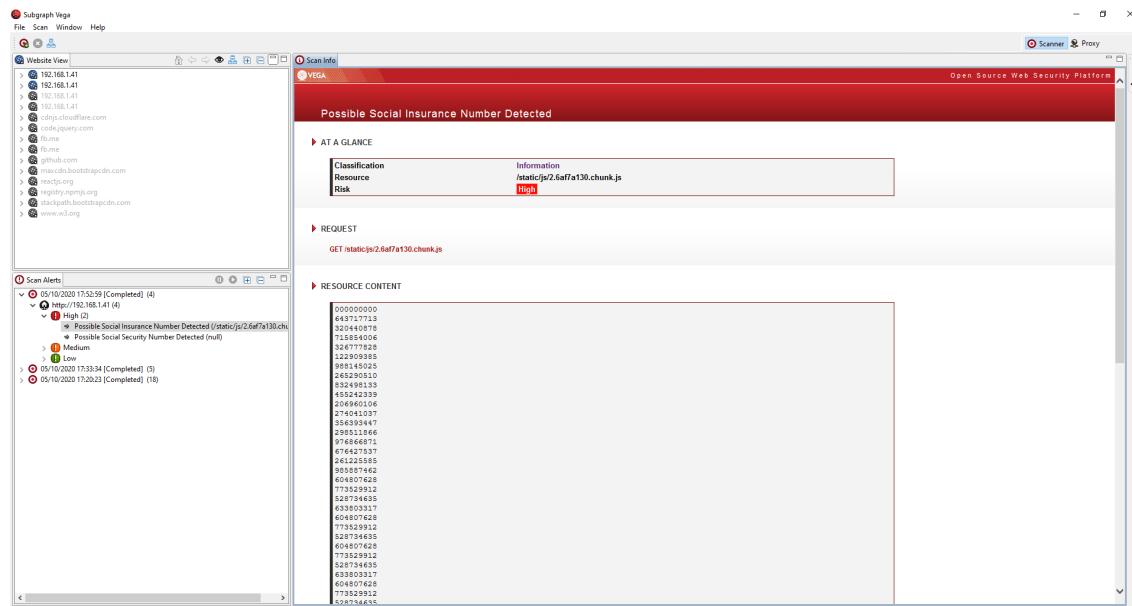


Fig. 5.54: Vega Scanner - Problema grave (Falso positivo)

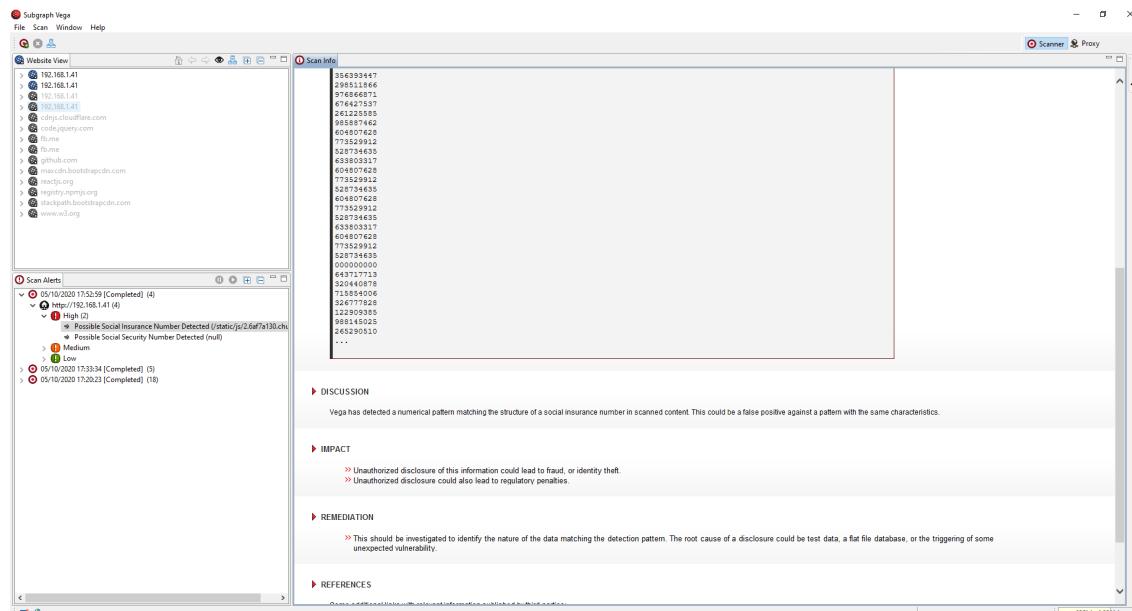


Fig. 5.55: Vega Scanner - Problema grave 2 (Falso positivo)

Por otro lado los problemas detectados del tipo *Info* y *Medium* sí que pueden considerarse vulnerabilidades reales que pueden solucionarse. El problema relacionado con las cabeceras *X-Frame-Options* puede aprovecharse para realizar un ataque llamado *Clickjacking attack* que consiste en hacer creer al usuario de la página web que está haciendo click en algo que realmente es otra cosa. Para solucionar este problema, se ha añadido una nueva línea en la configuración de Nginx (servidor web que está sirviendo la aplicación web) dentro del archivo *nginx.conf*, que establece que se envíen dichas cabeceras con el valor *SAMEORIGIN*. Definiendo dichas cabeceras de esta forma lo se está haciendo es permitir que, en caso de querer mostrar una determinada página web de la aplicación en un iframe por ejemplo, solo lo pueda hacer la propia aplicación.

La línea añadida a la configuración de Nginx y que hace esto posible es la siguiente:

```
add_header X-Frame-Options "SAMEORIGIN";
```

Con esto se habrían solucionado las 13 vulnerabilidades de tipo *Info*. Tras realizar un nuevo escaneo, se puede ver como han desaparecido del listado:

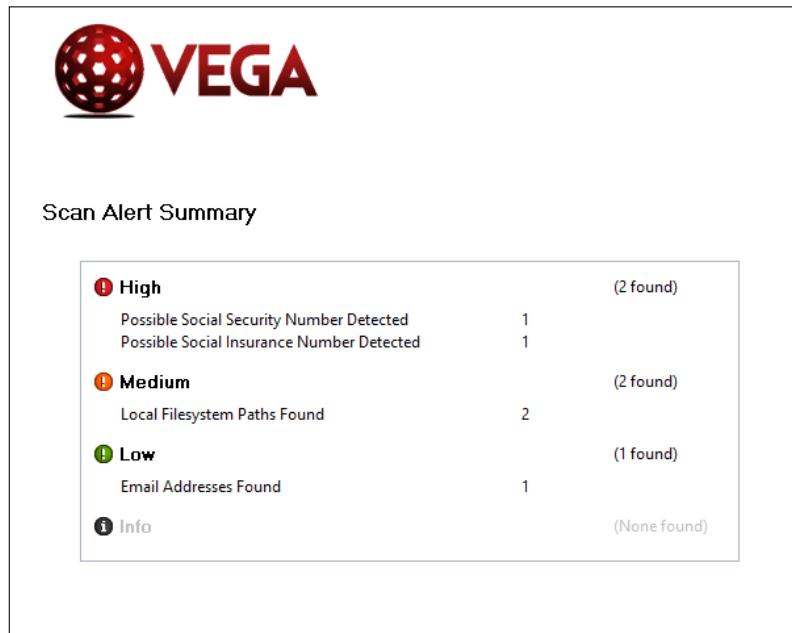


Fig. 5.56: Vega Scanner - Solución problemas del tipo *Info*.

Por último, tendríamos las vulnerabilidades de tipo *Medium* que se deben a que Vega a detectado un par de rutas absolutas en algún archivo. Esto podría ser de utilidad para algún atacante de cara a hacerse con información relevante sobre el sistema que soporta la aplicación, con la que poder preparar otros tipos de ataque. Para resolver este problema se puede ir a las localizaciones donde Vega a detectado las rutas absolutas y convertirlas en rutas relativas. Es preciso señalar que uno de los problemas de gravedad media ha sido también un falso positivo ya que una de las rutas la ha captado parcialmente, y debido a esto, la ha considerado absoluta cuando en realidad, comprobando el archivo donde se encuentra la ruta y teniéndola en cuenta en su totalidad, se puede afirmar que es relativa.

La ruta completa sería la siguiente: *static/media/generic.515625e4.png*

## 5.DESARROLLO

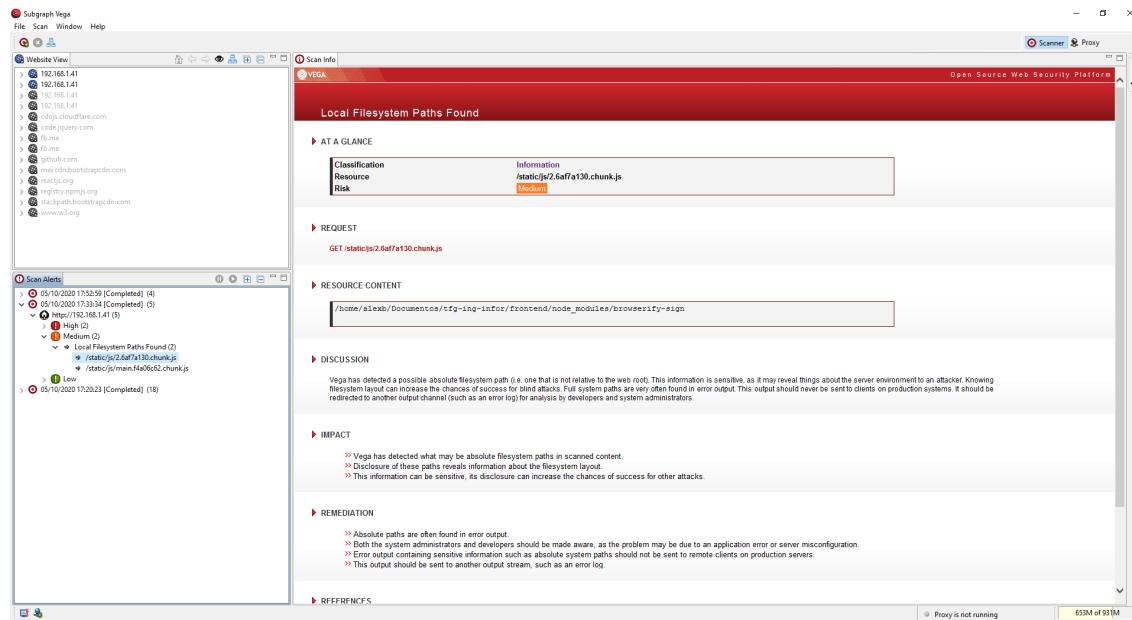


Fig. 5.57: Vega Scanner - Problema de severidad media

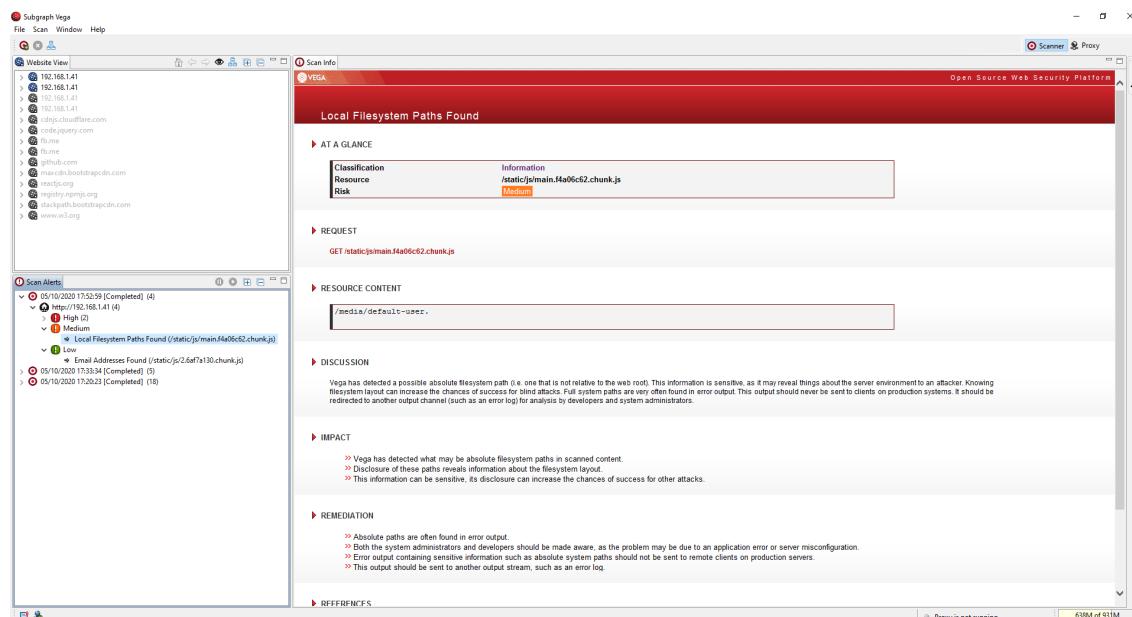


Fig. 5.58: Vega Scanner - Problema de severidad media 2 (Falso positivo)

Después de convertir la ruta absoluta en relativa y haber realizado un último análisis, se puede ver en los siguientes resultados como se ha resuelto satisfactoriamente el problema de severidad media referente a la ruta corregida:

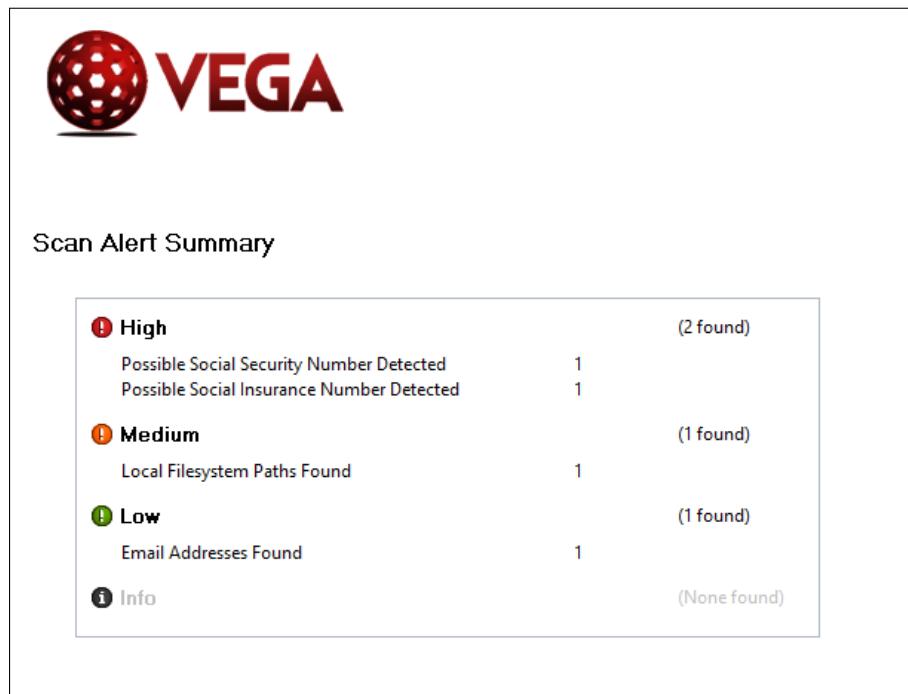


Fig. 5.59: Vega Scanner - Solución problema de severidad media

Las demás cuestiones que aparecen en los resultados ya se han analizado anteriormente y no han supuesto un peligro real para la seguridad de la aplicación por lo que no es necesario llevar a cabo ninguna acción en relación a ellas.

#### 5.5.5.2. Skipfish

Otra herramienta utilizada para la realización de pruebas de seguridad ha sido Skipfish. Skipfish es una herramienta de reconocimiento de seguridad de aplicaciones web que, primero, analiza el sitio web exhaustivamente para generar un mapa completo de la aplicación. Una vez obtenido dicho mapa, realiza sobre él una serie de comprobaciones de seguridad y genera un informe bastante detallado con los resultados obtenidos.

Para lanzar Skipfish se puede utilizar el siguiente comando:

```
skipfish -o <carpeta> http://<target>/
```

Donde pone <carpeta> podemos fijar el directorio donde queremos que nos genere el informe final, mientras que <target> sería el nombre de dominio o la IP del servidor web donde esté alojada la aplicación web.

La ejecución del análisis será algo más tardía que las de las otras 2 herramientas. Sin embargo, proporciona informes bastante detallados. Tras la realización de un primer escaneo estos son los resultados obtenidos:

## 5.DESARROLLO

The screenshot shows the Skipfish web application scanner interface. At the top, it displays the scanner version (2.0b), random seed (c0cf0bd9f6), scan date (Sun May 10 12:45:00 2020), total time (0 hr 1 min 48 sec 647 ms), and a link to check problems if any were found.

**Crawl results - click to expand:**

- http://192.168.1.41/ (Orange icon, 5 vulnerabilities, 3 notes, 31 errors, 25 warnings)
- http://192.168.1.41:5000/ (Green icon, 12 vulnerabilities, 8 notes, 4 errors)
- http://192.168.1.41:5001/ (Green icon, 12 vulnerabilities, 8 notes, 4 errors)
- http://192.168.1.41:5002/ (Green icon, 8 vulnerabilities, 8 notes, 2 errors)

**Document type overview - click to expand:**

- application/javascript (6)
- application/xhtml+xml (0)
- image/png (2)
- image/x-ms-bmp (1)
- text/css (1)
- text/html (2)
- text/plain (1)

**Issue type overview - click to expand:**

- External content embedded on a page (higher risk) (1)
- Node should be a directory; detection error? (1)
- Parent behavior checks failed (no brute force) (1)
- Numerical filename - consider enumerating (1)
- Incorrect or missing charset (low risk) (2)
- Incorrect or missing MIME type (low risk) (2)
- HTTP authentication required (1)
- Resource not directly accessible (1)

Fig. 5.60: Skipfish - Resultados primer escaneo

Como se puede apreciar en la captura anterior, Skipfish muestra muchísima información sobre distintas cuestiones. La mayoría son aspectos o notas (color verde) sobre la aplicación que no suponen un peligro. No obstante, hay que tener cuidado y analizar detenidamente los problemas que ha detectado de severidad media (color naranja). En la siguiente captura se puede ver con más detalle con que están relacionados dichos problemas.



### Crawl results - click to expand:

A detailed view of a crawl result for the URL http://192.168.1.41/. It shows the following information:

- Code: 200, length: 3525, declared: text/html, detected: text/html, charset: [none] [ show trace + ]
- External content embedded on a page (higher risk)**
- 5 sub-items, each detailing a specific external content inclusion with its code, length, declaration, detection, and memo (link to the source).

Fig. 5.61: Skipfish - Problema de severidad media

En este caso, no suponen un peligro, ya que el contenido externo incluido en la aplicación web proviene de organizaciones oficiales y seguras: Bootstrap, JQuery y Cloudflare. Además, esos enlaces a sus páginas web son necesarios para cargar estilos CSS y scripts de Javascript, creados por ellos, que han sido utilizados en este proyecto para desarrollar algunas funcionalidades de la aplicación.

### 5.5.5.3. Wapiti

La última herramienta que he empleado ha sido Wapiti, un escáner de vulnerabilidades web de código abierto y gratuito. Para utilizarlo simplemente hay que instalarlo y ejecutar el siguiente comando en la consola:

```
wapiti -u http://<target>/
```

El elemento <target> habría que sustituirlo por el dominio o la IP del servidor web que queremos analizar. Si todo va bien, se iniciará el escaneo y al finalizar se creará un directorio en el que se irán guardando todos los resultados de los análisis que se vayan haciendo. Los resultados se pueden visualizar en el navegador ya que los archivos guardados tienen una extensión HTML.

A continuación se incluyen los resultados obtenidos después de analizar tanto el frontend como el backend de la aplicación web:

<b>Wapiti vulnerability report</b>	
Target: <a href="http://192.168.1.41/">http://192.168.1.41/</a>	
Date of the scan: Sun, 10 May 2020 16:10:01 +0000. Scope of the scan: folder	
<b>Summary</b>	
<b>Category</b> <span style="float: right;">Number of vulnerabilities found</span>	
Inyección SQL	0
Inyección SQL a ciegas	0
Manejo de fichero	0
Cross Site Scripting	0
Inyección CRLF	0
Ejecución de comandos	0
Bypass de Htaccess	0
Fichero de backup	0
Fichero potencialmente peligroso	0
Server Side Request Forgery	0
Open Redirect	0
XXE	0
Error interno del servidor	0
Consumo de recursos	0

Fig. 5.62: Wapiti - Resultados frontend

## 5.DESARROLLO

**Wapiti vulnerability report**

Target: <http://192.168.1.41:5000/>

Date of the scan: Sun, 10 May 2020 16:11:08 +0000. Scope of the scan: folder

---

**Summary**

Category	Number of vulnerabilities found
Inyección SQL	0
Inyección SQL a ciegas	0
Manejo de fichero	0
Cross Site Scripting	0
Inyección CRLF	0
Ejecución de comandos	0
Bypass de Htaccess	0
Fichero de backup	0
Fichero potencialmente peligroso	0
Server Side Request Forgery	0
Open Redirect	0
XXE	0
Error interno del servidor	0
Consumo de recursos	0



Fig. 5.63: Wapiti - Resultados microservicio de usuarios

## Wapiti vulnerability report

Target: <http://192.168.1.41:5001/>

Date of the scan: Sun, 10 May 2020 16:13:56 +0000. Scope of the scan: folder

---

### Summary

Category	Number of vulnerabilities found
Inyección SQL	0
Inyección SQL a ciegas	0
Manejo de fichero	0
Cross Site Scripting	0
Inyección CRLF	0
Ejecución de comandos	0
Bypass de Htaccess	0
Fichero de backup	0
Fichero potencialmente peligroso	0
Server Side Request Forgery	0
Open Redirect	0
XXE	0
Error interno del servidor	0
Consumo de recursos	0

Fig. 5.64: Wapiti - Resultados microservicio de investigadores

## 5.DESARROLLO

Wapiti vulnerability report	
Target: http://192.168.1.41:5002/	
Date of the scan: Sun, 10 May 2020 16:14:03 +0000. Scope of the scan: folder	
Summary	
Category	Number of vulnerabilities found
Inyección SQL	0
Inyección SQL a ciegas	0
Manejo de fichero	0
Cross Site Scripting	0
Inyección CRLF	0
Ejecución de comandos	0
Bypass de Htaccess	0
Fichero de backup	0
Fichero potencialmente peligroso	0
Server Side Request Forgery	0
Open Redirect	0
XXE	0
Error interno del servidor	0
Consumo de recursos	0

Fig. 5.65: Wapiti - Resultados microservicio de administradores

Como se puede apreciar en las capturas anteriores Wapiti no ha detectado ninguna vulnerabilidad en el frontend ni en los microservicios. Como se puede apreciar cada herramienta ofrece resultados completamente distintos, razón por la que es fundamental emplear múltiples herramientas de este estilo al realizar pruebas de seguridad en aplicaciones software.

### 5.5.6. Pruebas de usabilidad e interfaz de usuario

En este tipo de pruebas se prioriza el análisis de la interfaz o los componentes visuales de la aplicación con los que interactúa el usuario. Es fundamental que cuando un usuario utiliza una aplicación se sienta cómodo y no se frustre debido a una complejidad excesiva de los pasos a dar para ejecutar una determinada acción. Es por esto que hay que dedicarle algo de tiempo a la revisión de la usabilidad y la interfaz de usuario de cualquier aplicación de manera que se asegure su facilidad de uso y aprendizaje.

Las pruebas de usabilidad consisten en realizar comprobaciones de navegación y contenido. Por un

lado, las comprobaciones de navegación sirven para confirmar el buen funcionamiento de todos los enlaces o hipervínculos presentes en las distintas páginas de la aplicación, de igual forma que se asegura que el usuario pueda acceder a la página o menú principal desde cualquier sitio. Por otro lado, la comprobaciones de contenido consisten en analizar todas las páginas de la aplicación en busca de errores ortográficos o gramaticales. Además, hay que asegurarse de que toda la información que aparezca en la aplicación sea comprensible sin mayor dificultad, que las imágenes tengan un tamaño razonable y que la fuente utilizada pueda leerse adecuadamente.

Las pruebas de interfaz de usuario se centran principalmente en verificar que la aplicación es responsive, es decir, que puede ser utilizada en múltiples plataformas con distintas prestaciones sin problemas. Para ello, habría que probar la aplicación en dispositivos como tabletas o dispositivos móviles que tengan diferentes resoluciones, de forma que se pueda analizar la calidad de visualización que tiene la interfaz de usuario en dichos dispositivos. En caso de algún elemento de la interfaz no se vea correctamente, habría que corregir su estilo hasta que se consiga una buena visualización en cualquier tipo de plataforma o dispositivo.

### 5.5.7. Análisis de los resultados obtenidos

En total, como se ha detallado previamente, se han llevado a cabo 4 tipos de pruebas:

1. Pruebas generales de funcionamiento.
2. Pruebas de rendimiento.
3. Pruebas de seguridad.
4. Pruebas de usabilidad e interfaz de usuario.

En general, los resultados de todas ellas han sido positivos y beneficiosos para el proyecto ya que han permitido solucionar errores o problemas presentes en la aplicación web que no habían sido detectados con anterioridad. Por un lado, las pruebas generales de funcionamiento, usabilidad e interfaz de usuario han confirmado que la aplicación es fácilmente utilizable, intuitiva y que requiere un tiempo de aprendizaje muy corto. Además, se han podido arreglar varios bugs relacionados con algunas funcionalidades del microservicio de investigadores.

Por otro lado, las pruebas de rendimiento muestran que la parte del frontend responde a las peticiones increíblemente rápido. Los microservicios, en cambio, no disponen de una velocidad tan rápida. Esto es debido seguramente al servidor de producción utilizado para servir la aplicación principal de los microservicios, ya que puede que no tenga capacidad suficiente para soportar tanta carga de trabajo como la que le envía el software utilizado para la ejecución las pruebas de rendimiento. Además, a esto, habría que añadirle el hecho de que el dispositivo desde el que se han realizado las pruebas no dispone de una gran fuerza computacional.

Por último, en las pruebas de seguridad se han obtenido resultados de todo tipo. En la mayoría de los casos, las “vulnerabilidades” que habían hecho saltar las alarmas de las herramientas empleadas, no tenían gran relevancia o eran falsos positivos. Aún así, los problemas que verdaderamente

## 5.DESARROLLO

podrían suponer un peligro para la aplicación han podido ser detectados, revisados, y solucionados, en caso de ser necesario, gracias a este tipo de pruebas.

### 5.6. INCIDENCIAS DEL PROYECTO

Para finalizar con el capítulo de desarrollo, se recogen en este apartado las incidencias que han surgido durante el transcurso del proyecto. En general, se puede decir que las incidencias o problemas surgidos no han sido graves. Sin embargo, vale la pena reflexionar acerca de ellos con el fin de mejorar mis prácticas como desarrollador, así como aprender diferentes formas de evitarlos en proyectos futuros.

Cabe destacar que en este proyecto no han surgido muchas incidencias. En definitiva, podemos identificar 3 de ellas como las más representativas del proyecto:

- **Complicaciones con Vue.js**

Al principio del proyecto se comenzó el desarrollo del frontend de la aplicación con Vue.js, un framework que ha ido ganando fama y fuerza en estos últimos tiempos. No obstante, para mí era una tecnología totalmente nueva por lo que tuve que empezar aprendiendo cómo se utilizaba. Durante el proceso de aprendizaje, poco a poco iba implementando diferentes partes de la UI, pero llegó un punto en el que necesitaba emplear un tiempo excesivo para implementar una funcionalidad que consistía en conectarse a uno de los microservicios para realizar una acción. Esto hizo que no me sintiera cómodo con el framework.

Es por eso que decidí cambiar de tecnología y usar React. Este framework sí que lo conocía más que Vue.js, no por haberlo utilizado antes, sino porque en alguna asignatura de la carrera lo había visto de manera teórica. Por lo tanto, ya conocía y tenía en mente las bases que me permitieron agilizar el proceso de implementación. Gracias a este cambio pude implementar mucho más rápidamente el frontend y lo que con Vue.js tardé múltiples días en implementar, con React lo hice en 1 único día.

- **Imagen base de los Dockerfiles**

Esta incidencia o complicación me causó bastantes quebraderos de cabeza ya que me costó varias horas descubrir cuál era el problema. El problema en cuestión consistía en que los documentos que no podían ser visualizados en la aplicación debido a su formato (.docx, .ppt, etc.), se mostraban con caracteres extraños en vez de dar la opción de descarga del documento. La clave para solucionar este error, se encontraba en los Dockerfiles de los microservicios ya que solo ocurría cuando utilizaba la aplicación desplegada con Docker. La imagen base que utilicé al principio se llamaba *python:3.8-slim-buster*, ya que era más ligera que otras, sin embargo, la codificación que esta usaba no era *UTF-8* por lo que los documentos no se codificaban adecuadamente para redirizarse en el navegador como deberían.

Hasta encontrar cuál era el motivo del problema, barajé múltiples posibilidades (el navegador, el código fuente del frontend, el código fuente de los microservicios...). Por eso me llevo

bastante tiempo solucionarlo. Al final, cuando me di cuenta, cambié la imagen base y utilicé una imagen genérica de *Ubuntu 18.04* con la que desapareció el problema de codificación.

#### ■ Errores generales de programación

Por último, esta podría ser una incidencia que engloba varios errores típicos que suelen darse en cualquier proyecto de desarrollo de software. En este caso, cabe mencionar algunos de los más notorios como:

- La imposibilidad de enviar directamente un *ObjectId* de un documento de Mongo en la respuesta de una petición enviada desde el backend al frontend: esto ha hecho que sea necesario crear un codificador JSON de tal manera que antes de enviar cualquier *ObjectId* en la respuesta a una petición se codifique su valor. De esta manera, se evita el error de serialización que se da al intentar enviar objetos o campos como el *ObjectId* de Mongo.
- Los problemas de red que se daban al principio al intentar realizar peticiones del frontend a alguno de los microservicios: este problema se daba porque las aplicaciones de Flask denegaban todos los accesos a recursos de origen cruzado (Cross Origin Resource Sharing), por lo que era necesario configurar las aplicaciones para que permitan que clientes externos a la propia aplicación puedan acceder a los recursos (rutas, imágenes, páginas...) de la aplicación. Para ello, se utilizó el paquete *Flask-CORS* por defecto, que habilita el soporte CORS en todas las rutas de la aplicación y, para todos los orígenes y métodos.
- Los típicos errores de sintaxis, variables no declaradas, null o vacías, al escribir mal el nombre de una variable, el scope de las variables en React, el no poder acceder al objeto o variable *this* en algunos métodos del frontend, la sangría en Python, errores de conexión con la base de datos, etc.



## Capítulo 6

# PLANIFICACIÓN

En este capítulo se detalla cómo se han organizado y planificado las distintas fases y tareas que conforman el proyecto. Para ello, se incluyen una serie de diagramas e imágenes en las que se detallan las actividades, tareas, fechas y recursos necesarios para la realización del proyecto, además de la planificación temporal definida mediante un cronograma.

### 6.1. PLAN DE RECURSOS HUMANOS

#### 6.1.1. Organización

En este apartado se incluyen las personas que participan en el proyecto junto con el rol que desempeña cada una:



Fig. 6.1: Organigrama de la organización

#### 6.1.2. Definición de roles

- **Director del proyecto:** Es el responsable de que los demás participantes finalicen los trabajos y actividades a realizar en el tiempo que se ha establecido para ello, así como dar indicaciones para las siguientes tareas del proyecto. Además, coordina la validación de los entregables generados y se asegura de que cumplen con las necesidades de los stakeholders.

## 6.PLANIFICACIÓN

- **Diseñador:** Es el encargado de realizar el diseño de la aplicación. Se encarga de identificar a los stakeholders, sus necesidades y las restricciones de negocio, elaborar el listado de requisitos funcionales y no funcionales, elaborar diagramas de secuencia, casos de uso, clases, etc.
- **Desarrollador:** Es el responsable de llevar a cabo el desarrollo software de la aplicación, que incluye la interfaz de usuario y los microservicios. La UI que implemente tiene ser clara y sencilla de modo que se pueda utilizar sin tener conocimientos informáticos avanzados, y no tener una curva de aprendizaje demasiado grande. Además, se encarga de la puesta en marcha del sistema una vez finalizada la fase de implementación.
- **Experto:** Ofrece información indispensable a los diferentes personas que forman el equipo de trabajo del proyecto con el fin de evitar problemas relacionados con el dominio para el que está dirigida la solución software. Dispone de amplios conocimientos en su campo, por lo que es la persona idónea para que realice la fase de planificación del proyecto.
- **Tester:** Es el encargado de realizar diferentes tipos de pruebas a la aplicación tras su desarrollo, de manera que se pueda asegurar su buen funcionamiento y rendimiento. De igual forma, es el responsable de documentar los resultados obtenido en dichas pruebas.
- **Científicos y técnicos:** Podríamos llamarlos “investigadores”, considerados como los stakeholders del proyecto. El director conoce sus necesidades y si es necesario puede ponerse en contacto con ellos para recabar información necesaria para el diseño y desarrollo de la aplicación.

### 6.2. PLAN DE TRABAJO

En este apartado se detallan las etapas que conforman el proyecto junto con sus respectivas tareas. En total se han realizado 21 tareas divididas en 6 etapas: planificación, diseño, desarrollo, puesta en marcha, pruebas y documentación.

A continuación se encuentran los planes de trabajo del proyecto, 1 por cada mes. En cada plan se especifican las fechas, responsables, horas de esfuerzo, prerequisitos y el estado de cada tarea a lo largo del transcurso del proyecto, así como pequeños cronogramas que muestran el trabajo realizado en cada momento. Es importante aclarar que la tarea 19 (“Redacción de la memoria”) se encuentra marcada en verde en todos los planes de trabajo. Esa marca significa que no se dispone de un momento concreto para su realización sino que se ha ido completando de manera continua e incremental, desde el comienzo hasta el final del proyecto.

## 6.2.1. Febrero

		Febrero																											
Tareas	Responsable	Fecha de inicio	Fecha final	Horas	Prerrequisitos	Estado	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29		
<b>Planificación (F1)</b>																													
T1 - Investigaciones iniciales	Experto	9-feb.		28		Completado																							
T2 - Estudio tecnologías	Experto	14-feb.	12-feb.	4		Completado																							
T3 - Definición fases y tareas	Experto	20-feb.	23-feb.	4		Completado																							
T4 - Objetivos y alcance	Experto	25-feb.	28-feb.	8		Completado																							
T5 - Creación EDT	Experto	29-feb.	29-feb.	4		Completado																							
<b>Diseño (F2)</b>																													
T6 - Recopilación requisitos	Diseñador	04-mar	11-mar	16		Sin completar																							
T7 - Selección de tecnologías	Diseñador	02-mar	03-mar	4		Sin completar																							
T8 - Diseño de la arquitectura	Diseñador	12-mar	15-mar	12		Sin completar																							
T9 - Formato datos BD	Diseñador	16-mar	18-mar	4		Sin completar																							
T10 - Definir seguridad	Diseñador	19-mar	22-mar	8		Sin completar																							
<b>Desarrollo (F3)</b>																													
T11 - Implementación frontend	Desarrollador	24-mar	26-abr	64		Sin completar																							
T12 - Implementación backend	Desarrollador	26-mar	23-abr	52		Sin completar																							
T13 - Integración frontend y backend	Desarrollador	24-abr	28-abr	16		Sin completar																							
T14 - Almacenar datos en BD	Desarrollador	29-abr	01-may	8		Sin completar																							
<b>Puesta en marcha (F4)</b>																													
T15 - Arranque de la aplicación	Desarrollador	02-may	04-may	8		Sin completar																							
<b>Pruebas (F5)</b>																													
T16 - Pruebas de interacción	Testeador	11-may	19-may	44		Sin completar																							
T17 - Pruebas de rendimiento	Testeador	20-may	25-may	20		Sin completar																							
T18 - Análisis de los resultados	Testeador	27-may	29-may	8		Sin completar																							
<b>Documentación (F6)</b>																													
T19 - Redacción de la memoria	Desa., test. y dise.	09-feb	24-jun	80		En curso																							
T20 - Redacción manual usuario	Desarrollador	05-may	10-may	16		Sin completar																							
T21 - Redacción conclusiones	Desa., test. y dise.	10-jun.	13-jun.	8		Sin completar																							

Fig. 6.2: Plan de trabajo de febrero

## 6. PLANIFICACIÓN

### 6.2.2. Marzo

Fig. 6.3: Plan de trabajo de marzo

### 6.2.3. Abril

Fecha de inicio		09-feb	Fecha final		24-jun	Avance general		70%	Abril																													
Tareas	Responsable	Fecha de inicio	Fecha final	Horas	Prerequisitos	Estado	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30		
<b>Planeación (F1)</b>																																						
T1 - Investigaciones iniciales	Experto	9-feb.	12-feb.	28		Completado																																
T2 - Estudio tecnologías	Experto	14-feb.	17-feb.	8		Completado																																
T3 - Definición fases/tareas	Experto	20-feb.	23-feb.	4		Completado																																
T4 - Objetivos y alcance	Experto	25-feb.	28-feb.	8		Completado																																
T5 - Creación EDT	Experto	29-feb.	29-feb.	4		Completado																																
<b>Diseño (F2)</b>																																						
T6 - Recopilación requisitos	Designer	04-mar	11-mar	16		Completado																																
T7 - Selección de tecnologías	Designer	02-mar	03-mar	4		Completado																																
T8 - Diseño de la arquitectura	Designer	12-mar	15-mar	12		Completado																																
T9 - Formato datos BD	Designer	16-mar	18-mar	4		Completado																																
T10 - Definir seguridad	Designer	19-mar	22-mar	8		Completado																																
<b>Desarrollo (F3)</b>																																						
T11 - Implementación frontend	Desarrollador	24-mar	26-abr	64		Completado																																
T12 - Implementación backend	Desarrollador	26-mar	23-abr	52		Completado																																
T13 - Integración frontend y backend	Desarrollador	24-abr	28-abr	16		Completado																																
T14 - Almacenar datos en BD	Desarrollador	29-abr	01-may	8		En curso																																
<b>Puesta en marcha (F4)</b>																																						
T15 - Arranque de la aplicación	Desarrollador	02-may	04-may	8		Sin completar																																
<b>Pruebas (F5)</b>																																						
T16 - Pruebas de interacción	Testeador	11-may	19-may	16		Sin completar																																
T17 - Pruebas de rendimiento	Testeador	20-may	25-may	20		Sin completar																																
T18 - Análisis de los resultados	Testeador	27-may	29-may	8		Sin completar																																
<b>Documentación (F6)</b>																																						
T19 - Redacción de la memoria	Desa., test. y dise.	09-feb	24-jun	80		En curso																																
T20 - Redacción manual usuario	Desarrollador	05-may	10-may	16		Sin completar																																
T21 - Redacción conclusiones	Desa., test. y dise.	10-jun..	13-jun..	8		Sin completar																																

Fig. 6.4: Plan de trabajo de abril

## 6. PLANIFICACIÓN

#### 6.2.4. Mayo

Fig. 6.5: Plan de trabajo de mayo

## 6.2.5. Junio

Junio																												
					1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
<b>Plaintificación (F1)</b>																												
T1 - Investigaciones iniciales	Experto	9-feb.	12-feb.	28																								
T2 - Estudio tecnologías	Experto	14-feb.	17-feb.	4																								
T3 - Definición fase y tareas	Experto	20-feb.	23-feb.	8																								
T4 - Objetivos y licance	Experto	25-feb.	28-feb.	4																								
T5 - Creación EDT	Experto	29-feb.	29-feb.	8																								
<b>Diseño (F2)</b>																												
T6 - Recopilación requisitos	Desañador	04-mar	11-mar	44																								
T7 - Selección de tecnologías	Desañador	03-mar	03-mar	16																								
T8 - Diseño de la arquitectura	Desañador	12-mar	15-mar	4																								
T9 - Formato datos BD	Desañador	16-mar	18-mar	12																								
T10 - Defini seguridad	Desañador	19-mar	22-mar	4																								
<b>Desarrollo (F3)</b>																												
T11 - Implementación frontend	Desarrollador	24-mar	26-abr	64																								
T12 - Implementación backend	Desarrollador	26-mar	23-abr	52																								
T13 - Integración frontend y back	Desarrollador	24-abr	28-abr	16																								
T14 - Almacenar datos en BD	Desarrollador	29-abr	01-may	8																								
<b>Puesta en marcha (F4)</b>																												
T15 - Arraque de la aplicación	Desarrollador	02-may	04-may	8																								
<b>Pruebas (F5)</b>																												
T16 - Pruebas de interacción	Testeador	11-may	19-may	44																								
T17 - Pruebas de rendimiento	Testeador	20-may	25-may	16																								
T18 - Análisis de los resultados	Testeador	27-may..	29-may..	8																								
<b>Documentación (F6)</b>																												
T19 - Redacción de la memoria	Desa., test. y dise.	09-feb	24-jun	104																								
T20 - Redacción manual usuario	Desarrollador	05-may	10-may	80																								
T21 - Redacción conclusiones	Desa., test. y dise.	10-jun..	13-jun..	16																								

Fig. 6.6: Plan de trabajo de junio

## 6.PLANIFICACIÓN

### 6.3. PERT

Es preciso aclarar que las fechas finales de las tareas que aparecen en el diagrama PERT tienen un día más que en los demás apartados de la planificación. Sin embargo, esto no significa que sean rangos de fechas distintos. De cierta manera representan lo mismo, solo que en este apartado el día explicitado como fecha final no se encuentra incluido. Por lo tanto, en la tarea 1 por ejemplo, la fecha final real sería el 12 de febrero aún apareciendo 13 en el diagrama. Como la herramienta utilizada para la elaboración del diagrama contaba los días automáticamente y se quería mantener el número de días en cada tarea igual que en los demás apartados, se ha decidido hacerlo así.

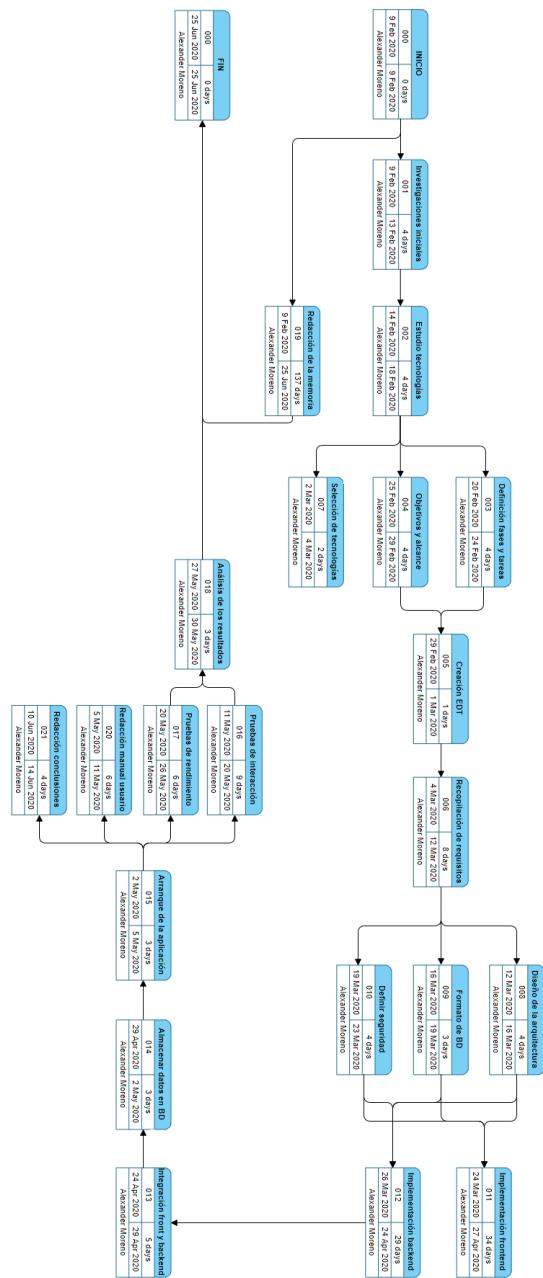


Fig. 6.7: Diagrama PERT.

## 6.4. CRONOGRAMA DEL PROYECTO

Los colores que se pueden apreciar en el cronograma simbolizan las etapas de las que se compone el proyecto. De esta forma, el color naranja representa la etapa de planificación, el azul la etapa de diseño, el rojo el desarrollo, el amarillo la puesta en marcha, el gris las pruebas, y por último, el verde la documentación.

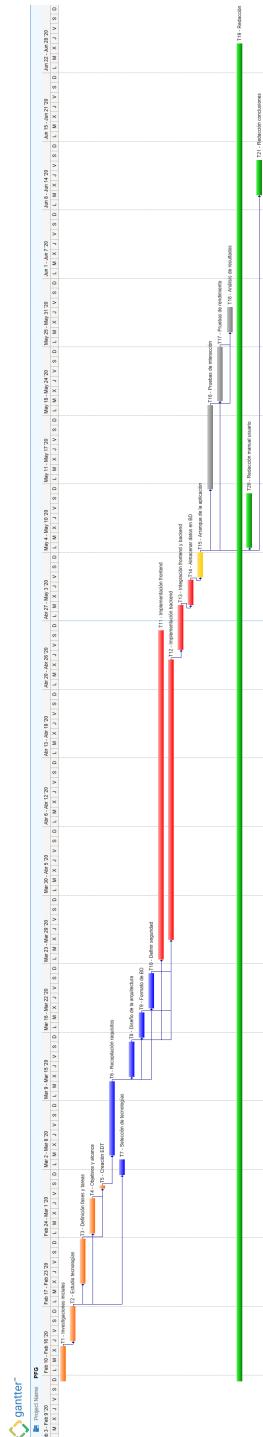


Fig. 6.8: Cronograma completo del proyecto.



## Capítulo 7

# PRESUPUESTO

En este capítulo se define el presupuesto aproximado que recoge todos los costes referentes al proyecto. Por un lado, se encuentran los costes de los recursos humanos, y por otro lado, están los costes relacionados con los materiales o herramientas utilizadas. Por último, también se incluye un cuadro resumen en el que se especifica el coste total que supone el proyecto.

### 7.1. RECURSOS HUMANOS

Recurso	Perfil	Precio/h
R1	Director de proyecto	25 €
R2	Diseñador	12 €
R3	Desarrollador	15 €
R4	Experto	20 €
R5	Tester	22 €

Tabla 7.1: Detalle del precio por hora de cada recurso (I.V.A. no incluido).

Recurso	Esfuerzo	Precio total
R1	20 h.	500 €
R2	72 h.	864 €
R3	202 h.	3.030 €
R4	28 h.	560 €
R5	66 h.	1.452 €
<b>Total</b>	<b>388 h.</b>	<b>6.406 €</b>

Tabla 7.2: Desglose de precios por recurso (I.V.A. no incluido).

En la siguiente tabla no se recogen las horas empleadas por el director ya que no participa como tal en ninguna tarea. Esto es porque tiene un rol más relacionado con el control y seguimiento del trabajo realizado que se hace a lo largo de todo el proyecto en momentos puntuales. Sin embargo, en la tabla anterior sí que se ha tenido en cuenta.

## 7.PRESUPUESTO

Tarea	Recurso	Esfuerzo	Precio total
T1	R4	4 h.	80 €
T2	R4	8 h.	160 €
T3	R4	4 h.	80 €
T4	R4	8 h.	160 €
T5	R4	4 h.	80 €
T6	R2	16 h.	192 €
T7	R2	4 h.	48 €
T8	R2	12 h.	144 €
T9	R2	4 h.	48 €
T10	R2	8 h.	96 €
T11	R3	64 h.	960 €
T12	R3	52 h.	780 €
T13	R3	16 h.	240 €
T14	R3	8 h.	120 €
T15	R3	8 h.	120 €
T16	R5	16 h.	352 €
T17	R5	20 h.	440 €
T18	R5	8 h.	176 €
T19	R2,R3,R5	80 h.	1.265 €
T20	R3	16 h.	240 €
T21	R2,R3,R5	8 h.	125 €
<b>Total</b>		<b>368 h.</b>	<b>5.906 €</b>

Tabla 7.3: Desglose de precios por tarea (I.V.A. no incluido).

## 7.2. RECURSOS MATERIALES

Material	Nombre	Uds.	P. Unitario	Precio total
M1	Máquina Linux	1	650€	650 €
<b>Total</b>				<b>650 €</b>

Tabla 7.4: Desglose de precios por material (I.V.A. no incluido).

## 7.3. PRESUPUESTO TOTAL

Recursos	Coste total
Humanos	6.406 €
Materiales	650 €
<b>Total</b>	<b>7.056 €</b>

Tabla 7.5: Presupuesto total del proyecto (I.V.A. no incluido).

## Capítulo 8

# CONCLUSIONES Y PROPUESTAS FUTURAS

En este capítulo se recogen las conclusiones y aprendizajes más significativos que se han extraído tras la realización del proyecto, la forma en la que se han resuelto los distintos objetivos que forman el proyecto y las líneas de mejora que podrían considerarse como trabajo futuro que busca pulir y completar aún más la arquitectura y la aplicación web propuesta.

### 8.1. CONCLUSIONES

Al comienzo del proyecto se han definido una serie de objetivos que se han ido cumpliendo a largo de todas sus fases. Dichos objetivos se pueden sintetizar en 3 puntos fundamentales: la elección de las tecnologías para el desarrollo del proyecto, la identificación e implementación de las características o medidas de seguridad y el desarrollo de la aplicación web (frontend y backend) utilizando las tecnologías elegidas y aplicando las medidas de seguridad ideadas.

Todos ellos, se han cumplido procurando adecuarse a las necesidades de la aplicación web utilizando las tecnologías que se han considerado más oportunas para su desarrollo, así como las medidas y características de seguridad que aseguren la integridad tanto de los datos con los que trabaja el sistema como los de la propia aplicación.

A continuación aparece un listado en el que se recogen los objetivos más significativos del sistema junto con las tecnologías, herramientas o elementos que han permitido su cumplimiento:

- **Desarrollo del frontend:** React, Node.js, npm, JavaScript, HTML y CSS.
- **Desarrollo del backend:** Flask, Python y MongoDB.
- **Implementación de medidas de seguridad:** hashing de contraseñas, JWT Token, registro en dos pasos, inicio de sesión de administradores en dos pasos, pruebas de seguridad, HTTPS (en la nube)...
- **Elección de tecnologías para el desarrollo de la aplicación:** conclusiones sacadas

## 8.CONCLUSIONES Y PROPUESTAS FUTURAS

de la investigación y análisis realizado a las diferentes alternativas, su facilidad de uso, la experiencia previa que se poseía en cada una de ellas, de código libre, etc.

En general, el proyecto se ha ejecutado de forma bastante fluida y sin graves complicaciones que hayan causado el retraso o la detención de los avances del mismo. Considero que ha sido una gran oportunidad para conocer y aprender nuevos conocimientos sobre una arquitectura distinta a la que estamos acostumbrados en el grado. Además, me ha hecho darme cuenta de la importancia que están ganando las arquitecturas basadas en microservicios en el mundo de las grandes empresas ya que poseen ciertas ventajas frente a la arquitectura monolítica.

Cabe destacar que implementar los microservicios que forman el backend de la aplicación me ha permitido gestionar todo el código de una forma más organizada que en ocasiones anteriores. Además, he podido ver y presenciar de primera mano esas ventajas (de las que se ha hablado anteriormente) que tienen los microservicios. Todo lo que he trabajado para sacar adelante el proyecto, estoy seguro que me servirá más adelante en mi carrera profesional si me dedico al desarrollo de software, más concretamente, al desarrollo de aplicaciones.

Para finalizar con la conclusiones, se recogen los aprendizajes más relevantes que he adquirido gracias a la realización de este proyecto:

- Desarrollar los diferentes apartados que forman el proyecto, me ha otorgado experiencia adicional que me será de gran utilidad en un futuro a la hora de llevar a cabo otros proyectos similares durante mi carrera profesional.
- He adquirido nuevos conocimientos relacionados con las arquitecturas basadas en microservicios enfocadas al desarrollo web, al igual que he aprendido a crear aplicaciones con un framework, que no había utilizado nunca en la práctica, como es React.
- He profundizado aún más en el ámbito de la seguridad de aplicaciones web, mejorando mis prácticas en cuanto a la instalación, puesta en marcha y desarrollo seguros de este tipo de aplicaciones manteniendo un nivel de seguridad lo más alto posible en todo momento.
- He descubierto nuevas formas y herramientas con las que realizar análisis de seguridad a aplicaciones web.
- He mejorado mi método de investigación y recogida de información para la posterior redacción de la documentación del proyecto.
- Me ha servido para mejorar mis habilidades en la redacción de textos y documentos académicos como es la memoria de un trabajo de fin de grado.

### 8.2. TRABAJO FUTURO

Aunque se hayan completado los objetivos del proyecto satisfactoriamente, se pueden identificar una serie de cuestiones o características que podrían ser implementadas en un futuro con el fin de aumentar la calidad de la arquitectura y la aplicación web desarrolladas. A continuación se incluye un listado en el que se recogen las líneas de mejora que podrían aplicarse a este proyecto:

- Añadir más módulos o microservicios al backend que proporcionen nuevas funcionalidades a la aplicación web, como la posibilidad de ejecutar algoritmos escritos en distintos lenguajes de programación, visualizar todo tipo de documentos en la página web, etc.
- Reestructurar la arquitectura con los nuevos microservicios que se implementen, iniciando nuevas bases de datos junto con los microservicios que se consideren oportunos.
- Poner en marcha el backend de la aplicación utilizando bases de datos de Mongo en local, es decir, en la misma máquina donde se encuentren los microservicios, en vez de hacerlo en la nube mediante *MongoDB Atlas*.
- Implementar nuevas medidas de seguridad que aseguren partes de la arquitectura o aplicación que no se hayan tenido en cuenta en este proyecto.
- Añadir más traducciones al frontend de la aplicación de tal forma que la interfaz de usuario pueda funcionar bajo nuevos idiomas aparte del español, inglés y euskera. Para ello, se puede emplear directamente el módulo llamado *react-i18next*, el mismo que se ha utilizado para implementar el i18n en el proyecto.
- Mejorar el aspecto de la interfaz de usuario utilizando estilos que la vuelvan más atractiva visualmente. Además, a la hora de visualizar el contenido de un archivo se le podría dar un formato u otro dependiendo del lenguaje de programación en el que esté escrito.
- Investigar nuevas tecnologías que puedan ser una mejor opción respecto a las utilizadas en el desarrollo del proyecto. En caso de encontrar alguna que dé mejores resultados, se podría sustituir la tecnología utilizada por la nueva, adaptando la aplicación y la arquitectura de forma que el sistema siga funcionando correctamente.
- Disponer de un mejor hardware que soporte el despliegue de la aplicación adecuadamente aportando un mejor rendimiento a la aplicación web. Además, un mejor hardware permitiría lanzar cada microservicio o el frontend múltiples veces de forma que, utilizando mecanismos o herramientas como *HAProxy* [43], se pueda balancear la carga entre ellos, logrando una mayor velocidad en la resolución de peticiones y manteniendo una alta disponibilidad de los servicios ofrecidos por la aplicación.
- Disponer de un dominio propio con el que solicitar un certificado SSL/TLS de tal forma que la aplicación pueda trabajar bajo HTTPS sin la necesidad de desplegarla en la nube.



## Capítulo 9

# BIBLIOGRAFÍA

- [1] Los “microservicios”, la nueva tendencia entre grandes empresas y startups. <https://www.baquia.com/emprendedores/los-microservicios-la-nueva-tendencia-entre-grandes-empresas-y-startups>, (consultado el 08/04/2020).
- [2] Daniel López, Edgar Maya, et al. Arquitectura de software basada en microservicios para desarrollo de aplicaciones web. 2017. <https://pdfs.semanticscholar.org/3840/fba79e2cadde6704234f3906bf1b07afc7a9.pdf>, (consultado el 22/04/2020).
- [3] Ana Cidat Vila. Microservicios para big data genómico en la nube. 2018. <https://riunet.upv.es/bitstream/handle/10251/109390/Cidat%20-%20Microservicios%20para%20Big%20Data%20Gen%C3%B3mico%20en%20la%20Nube.pdf?sequence=1&isAllowed=y>, (consultado el 22/04/2020).
- [4] Fausto López. Microservicios en las empresas. <https://blog.conasa.es/blog/microservicios-en-las-empresas-que-son-y-para-que-sirven>, (consultado el 22/04/2020).
- [5] Manuel Pérez-Herrera Cuadrillero. Trabajo Fin de Grado: Arquitecturas basadas en microservicios. [http://oa.upm.es/37346/1/PFC\\_MANUEL\\_PEREZ\\_HERRERA CUADRILLERO\\_2015.pdf](http://oa.upm.es/37346/1/PFC_MANUEL_PEREZ_HERRERA CUADRILLERO_2015.pdf), (consultado el 22/04/2020).
- [6] Fredy H Vera-Rivera, Carlos Mauricio Gaona Cuevas, and Hernán Astudillo. Desarrollo de aplicaciones basadas en microservicios: tendencias y desafíos de investigación. *Revista Ibérica de Sistemas e Tecnologias de Informação*, (E23):107–120, 2019. [https://www.researchgate.net/publication/338582836\\_Desarrollo\\_de\\_aplicaciones\\_basadas\\_en\\_microservicios\\_tendencias\\_y\\_desafios\\_de\\_investigacion](https://www.researchgate.net/publication/338582836_Desarrollo_de_aplicaciones_basadas_en_microservicios_tendencias_y_desafios_de_investigacion), (consultado el 22/04/2020).
- [7] DEVOPS: ARQUITECTURA MONOLÍTICA VS MICROSERVICIOS. <https://www.chakray.com/es/devops-arquitectura-monolitica-vs-microservicios/>, (consultado el 24/04/2020).
- [8] De sistemas monolíticos a microservicios. <https://www.aplyca.com/es/blog/aplicaciones-monoliticas-o-microservicios>, (consultado el 24/04/2020).
- [9] What is Scalability in the Cloud? <https://stonefly.com/blog/what-is-scalability-in-the-cloud>, (consultado el 05/05/2020).

## 9.Bibliografía

- [10] The Need for Speed in the Cloud. <https://www.kodiakdata.com/the-need-for-speed-in-the-cloud/>, (consultado el 05/05/2020).
- [11] Radha Krishna Prasad. Application Scalability — How To Do Efficient Scaling. <https://dzone.com/articles/application-scalability-how-to-do-efficient-scalin>, (consultado el 05/05/2020).
- [12] Why Does Site Speed Matter? <https://www.cloudflare.com/learning/performance/why-site-speed-matters/>, (consultado el 05/05/2020).
- [13] Five Microservices Trends in 2020. <https://www.charterglobal.com/five-microservices-trends-in-2020/>, (consultado el 05/05/2020).
- [14] 10 companies that implemented the microservice architecture and paved the way for others. <https://divante.com/blog/10-companies-that-implemented-the-microservice-architecture-and-paved-the-way-for-others/>, (consultado el 05/05/2020).
- [15] Pros y contras de la metodología en cascada. <https://obsbusiness.school/es/blog-project-management/metodologia-agile/pros-y-contras-de-la-metodologia-en-cascada#:~:text=La%20metodolog%C3%ADA%20en%20cascada%20es,un%20proceso%20de%20dise%C3%B1o%20secuencial.&text=El%20%C3%A9nfasis%20de%20la%20metodolog%C3%ADA,como%20el%20plan%20est%C3%A9n%20claros.>, (consultado el 30/05/2020).
- [16] MongoDB. <https://www.mongodb.com/>, (consultado el 14/04/2020).
- [17] MongoDB: la base de datos NoSQL flexible y escalable. <https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/mongodb-presentacion-y-comparacion-con-mysql/>, (consultado el 24/05/2020).
- [18] Mihir Shah. MongoDB vs. MySQL. <https://dzone.com/articles/comparing-mongodb-and-mysql>, (consultado el 24/05/2020).
- [19] Mihir Shah. MongoDB vs MySQL: A Comparative Study on Databases. <https://www.simform.com/mongodb-vs-mysql-databases/#section3>, (consultado el 24/05/2020).
- [20] Vue. <https://vuejs.org/>, (consultado el 14/04/2020).
- [21] React. <https://es.reactjs.org/>, (consultado el 14/04/2020).
- [22] Flask. <https://palletsprojects.com/p/flask/>, (consultado el 14/04/2020).
- [23] Django. <https://www.djangoproject.com/>, (consultado el 14/04/2020).
- [24] Is Django as a microservice a good idea? <https://www.quora.com/Is-Django-as-a-microservice-a-good-idea>, (consultado el 27/05/2020).
- [25] Ximena Rodríguez. Django vs Flask. <https://openwebinars.net/blog/django-vs-flask/>, (consultado el 27/05/2020).
- [26] Flask's documentation. <https://flask.palletsprojects.com/en/1.1.x/>, (consultado el 27/05/2020).

- [27] Michael Herman. Django vs. Flask in 2019: Which Framework to Choose. <https://testdriven.io/blog/django-vs-flask/>, (consultado el 27/05/2020).
- [28] Docker. <https://www.docker.com/>, (consultado el 14/04/2020).
- [29] Sergey C., Sviatoslav A. Advantages of Using Docker for Microservices. <https://rubygarage.org/blog/advantages-of-using-docker-for-microservices>, (consultado el 28/05/2020).
- [30] Heroku. <https://www.heroku.com/>, (consultado el 15/05/2020).
- [31] Axios. <https://github.com/axios/axios>, (consultado el 14/04/2020).
- [32] JSON. <https://www.json.org/json-es.html>, (consultado el 16/04/2020).
- [33] PyMongo. <https://pymongo.readthedocs.io/en/stable/>, (consultado el 14/04/2020).
- [34] Visual Studio Code. <https://code.visualstudio.com/>, (consultado el 16/04/2020).
- [35] Git. <https://git-scm.com/>, (consultado el 16/04/2020).
- [36] Github. <https://github.com/>, (consultado el 16/04/2020).
- [37] Node.js. <https://nodejs.org/es/>, (consultado el 17/04/2020).
- [38] Create React App. <https://create-react-app.dev/docs/getting-started/>, (consultado el 17/04/2020).
- [39] Presentando JSX. <https://es.reactjs.org/docs/introducing-jsx.html>, (consultado el 13/06/2020).
- [40] Vega Vulnerability Scanner. <https://subgraph.com/vega/>, (consultado el 28/04/2020).
- [41] Skipfish: Penetration Testing Tools. <https://tools.kali.org/web-applications/skipfish>, (consultado el 28/04/2020).
- [42] Wapiti : a Free and Open-Source web-application vulnerability scanner. <https://wapiti.sourceforge.io/>, (consultado el 28/04/2020).
- [43] HAProxy: The Reliable, High Performance TCP/HTTP Load Balancer. <http://www.haproxy.org/>, (consultado el 07/06/2020).
- [44] Getting Started on Heroku with Python. <https://devcenter.heroku.com/articles/getting-started-with-python#define-a-procfile>, (consultado el 21/05/2020).
- [45] Garantía de los derechos digitales. <https://www.dcd.es/garantia-de-los-derechos-digitales/>, (consultado el 31/05/2020).



# DEFINICIONES, ACRÓNIMOS Y ABREVIATURAS

## DEFINICIONES

- **Frontend:** en el ámbito del desarrollo web, son todas las cuestiones relacionadas con lo que el usuario ve, es decir, con la interfaz gráfica. Normalmente los lenguajes vinculados a este concepto son HTML y CSS.
- **Backend:** en el ámbito del desarrollo web, hace referencia a la parte servidora con la que interactúa el frontend. Está formado por todas las características que no son visibles para el usuario como la lógica de la aplicación, las conexiones y consultas de BD, la conexión con otros servidores o servicios, etc.
- **Framework de desarrollo:** conjunto de herramientas enfocadas a un ámbito concreto que tienen como propósito hacer más productivo el trabajo de desarrolladores y programadores, así como agilizar el proceso de implementación y testeo de aplicaciones.
- **Stakeholder:** persona u organización que se ve afectada directa o indirectamente por el producto final.
- **URI:** cadena de caracteres que identifica un recurso como una página, un documento, etc. No hay que confundirla con una URL. Podríamos decir que una URL es un tipo específico de URI.
- **Dirección IP:** serie de números que se le asignan a un único dispositivo a la vez, que sirven para identificarlo dentro de una red que utilice el protocolo de internet.
- **Puerto:** interfaz, hardware o software, que permite el envío y recepción de datos entre distintos dispositivos, pudiendo estar estos conectados por cable o de forma inalámbrica.
- **Bug:** error o problema de software que provoca un comportamiento inesperado en una aplicación, sistema, etc.
- **Serializar:** proceso de codificación de un objeto a un conjunto de bytes o un formato más entendible como podría ser XML o JSON, de tal forma que sea posible transmitirlo a través de una conexión de red sin problemas.
- **Internacionalización:** consiste en diseñar e implementar software de forma que pueda funcionar bajo diferentes idiomas. De esta forma, 2 personas de distintas localizaciones podrán

## 9. DEFINICIONES, ACRÓNIMOS Y ABREVIATURAS

utilizar el mismo software que soporte ambos idiomas.

### ACRÓNIMOS

- **UI:** User Interface (Interfaz de Usuario).
- **EDT:** Work Breakdown Structure (Desglose de Trabajo del proyecto).
- **BD:** Base de Datos.
- **URI:** Uniform Resource Identifier (Identificador Uniforme de Recursos).
- **IP:** Internet Protocol (Protocolo de Internet).
- **IA:** Inteligencia Artificial.
- **PUC:** Product Use Case (Caso de uso del producto).
- **PERT:** Program Evaluation and Review Techniques (Técnicas de revisión y evaluación de programas).
- **TLS:** Transport Layer Security (Seguridad de la Capa de Transporte).
- **SSL:** Secure Sockets Layer (Capa de Sockets Segura).

### ABREVIATURAS

- **RRHH:** Recursos humanos.
- **vs.:** Versus.
- **p. ej.:** Por ejemplo.
- **i18n:** Internationalization (Internacionalización).

## Capítulo 10

# ANEXO I: MANUAL DE USUARIO

En este anexo se incluye el manual de usuario en el que se explica la forma de utilizar las diferentes funcionalidades que se han implementando en la aplicación web desde la perspectiva de los 3 tipos de usuarios que la pueden usar: usuarios (visitantes de la página web), investigadores y administradores. Además, se detallan los pasos seguidos tanto para la puesta en marcha del frontend y los microservicios en localhost con *Docker* como para su despliegue en la nube, más concretamente, en *Heroku*.

### 10.1. USO DE LA APLICACIÓN

Como se ha comentando en otras ocasiones la aplicación puede ser utilizada por 3 tipos de usuarios distintos: usuarios normales que visiten la página web, investigadores que dispongan de una cuenta con la que iniciar sesión y subir contenido, y por último, administradores que puedan acceder al sistema con su cuenta especial para gestionar cuentas de investigadores o el contenido que suban estos últimos.

#### 10.1.1. Usuarios

Cualquier persona que se conecte a la aplicación web, lo primero que verá será una página principal en la que se mostrarán en un listado todos los documentos almacenados en el sistema que hayan subido los investigadores. Además, en la parte superior se encuentra una barra de navegación con diferentes enlaces a otras páginas de la aplicación, como el login o registro, incluyendo una barra de búsqueda para filtrar el listado de documentos por nombre.

En esta página, se mostrará información relativa a cada documento. De igual forma, los usuarios podrán visualizar o descargar el documento que deseen dándole al botón azul situado debajo de cada uno de ellos. En caso de que se puedan visualizar en la propia página no se iniciará la descarga, pero en caso contrario, al darle al botón se iniciará el proceso de descarga directamente.

## 10.ANEXO I: MANUAL DE USUARIO

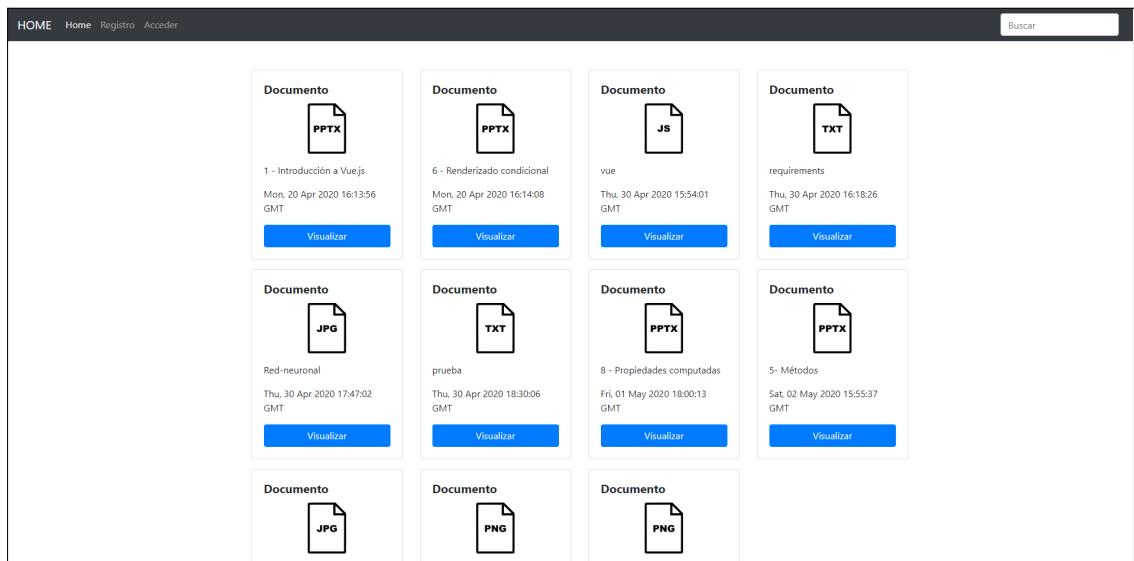


Fig. 10.1: Página principal de la aplicación web.

### 10.1.2. Investigadores

Los investigadores podrán registrarse en el sistema dándole al botón de registro situado en el panel superior de la página principal que les llevará al formulario de solicitud de registro.

A screenshot of a registration form titled 'Email'. It includes fields for 'Email' (with placeholder 'Email' and a note 'We'll never share your email with anyone else.'), 'Username' (placeholder 'Username'), 'Contraseña' (placeholder 'Contraseña'), 'Repetir contraseña' (placeholder 'Repetir contraseña'), and a large blue 'Registrarse' button at the bottom.

Fig. 10.2: Formulario de registro para investigadores.

Tras introducir los datos correspondientes y darle al botón de registro se enviará una solicitud que se guardará en el sistema. Dicha solicitud deberá ser aceptada por un administrador. Sea cual sea su respuesta (aceptar o rechazar la solicitud), se enviará un mensaje de correo electrónico a la

dirección introducida en el formulario de registro.

Cabe destacar que cada vez que se realiza una acción importante (registro, subir un archivo, modificar datos, eliminar documentos...) en la aplicación se le muestra al usuario un mensaje informativo de forma que sepa si todo el proceso ha ido bien o ha ocurrido algún error.



Fig. 10.3: Mensaje informativo tras enviar el formulario de registro.

Una vez que un administrador haya autorizado la creación de la cuenta el investigador en cuestión, podrá iniciar sesión en la aplicación desde la página de login.

The screenshot shows a login form with the following fields and buttons:

- A header message: "Bienvenido a nuestro sistema de login"
- A label "Email" above an input field containing the placeholder "Email".
- A label "Contraseña" above an input field containing the placeholder "Contraseña".
- A blue button labeled "Iniciar sesión" at the bottom.

Fig. 10.4: Formulario de login para investigadores.

Una vez iniciada la sesión, el investigador podrá subir contenido dándole al botón *Subir...* Una vez pulsado el botón, se mostrará un panel que permite subir archivos desde nuestro equipo.



Fig. 10.5: Panel que permite subir contenido a la aplicación web.

## 10. ANEXO I: MANUAL DE USUARIO

Todo el contenido que vayan subiendo a la aplicación se irá mostrando en la página principal general, de igual forma que en la página principal del investigador (en una tabla) donde podrán eliminar o modificar el nombre de archivos que hayan subido pulsando el botón correspondiente situado a la derecha de cada documento. Asimismo, los títulos de los documentos serán enlaces que permitirán a los investigadores comprobar que los documentos se pueden visualizar o descargar correctamente.

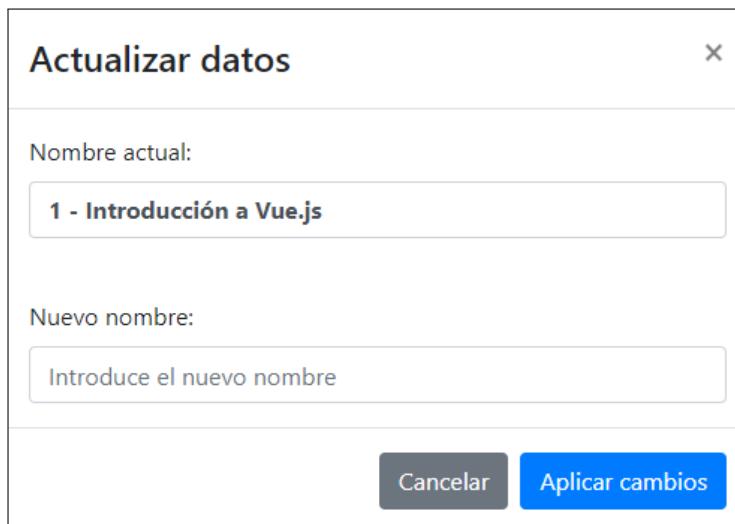


Fig. 10.6: Panel que permite actualizar el nombre de documentos.

No.	Tipo	Nombre	Última modificación		
1	.pptx	<a href="#">1 - Introducción a Vue.js</a>	Mon, 20 Apr 2020 16:13:56 GMT		
2	.pptx	<a href="#">6 - Renderizado condicional</a>	Mon, 20 Apr 2020 16:14:08 GMT		
3	.js	<a href="#">vue</a>	Thu, 30 Apr 2020 15:54:01 GMT		
4	.txt	<a href="#">requirements</a>	Thu, 30 Apr 2020 16:18:26 GMT		
5	.jpg	<a href="#">Red-neuronal</a>	Thu, 30 Apr 2020 17:47:02 GMT		
6	.txt	<a href="#">prueba</a>	Thu, 30 Apr 2020 18:30:06 GMT		
7	.pptx	<a href="#">8 - Propiedades computadas</a>	Fri, 01 May 2020 18:00:13 GMT		
8	.pptx	<a href="#">5- Métodos</a>	Sat, 02 May 2020 15:55:37 GMT		

Fig. 10.7: Página principal de investigador.

Los investigadores dispondrán de un pequeño menú desplegable en el panel superior, en la parte derecha, que contendrá dos botones, uno de ellos para acceder a su perfil y otro para cerrar sesión en la aplicación.

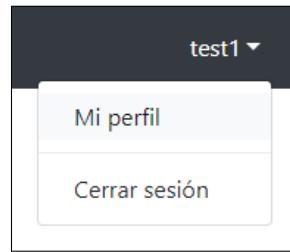


Fig. 10.8: Menú desplegable de investigador.

Dentro de la página del perfil podrán modificar su datos personales como la dirección de correo electrónico, el nombre de usuario o la contraseña. Para realizar cualquier tipo de cambio en estos datos es necesario introducir también la contraseña actual registrada en el sistema con el fin de confirmar los cambios.

Esta captura de pantalla muestra la interfaz de usuario para editar el perfil de un investigador. En la parte superior izquierda, se muestra el enlace 'HOME Home'. En la parte superior derecha, se ve el nombre 'test1'. El título de la página es 'Mi perfil'. A la izquierda de la sección de formulario, hay una silueta negra de una persona. Los campos de formulario incluyen:

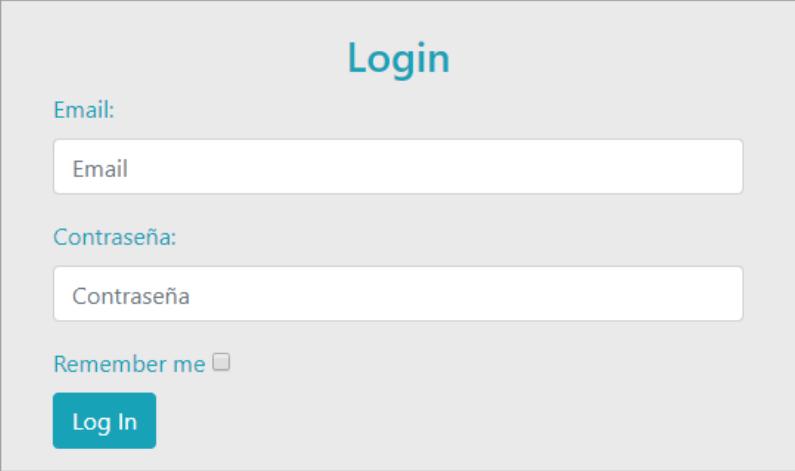
- Nombre actual:** test1  
Introduce un nuevo nombre
- Email actual:** test1@test.com  
Introduce un nuevo email
- Modificación de contraseña:**
  - Introduce una nueva contraseña
  - Introduce tu contraseña actual
- Actualizar**

Fig. 10.9: Perfil de investigadores.

### 10.1.3. Administradores

Los administradores podrán iniciar sesión accediendo a la página web y añadiendo a la url base `/admin`. De esta manera, se les mostrará la siguiente página.

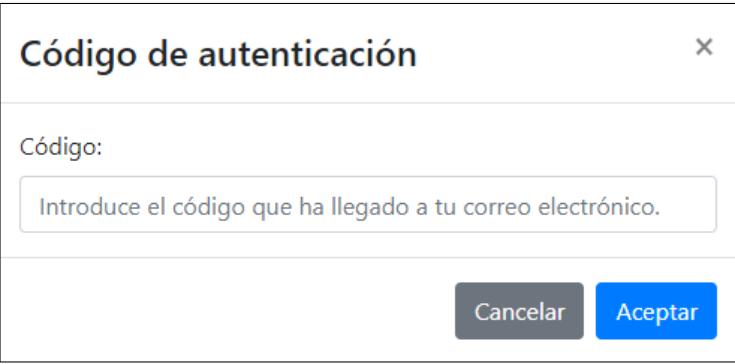
## 10. ANEXO I: MANUAL DE USUARIO



The image shows a login interface titled "Login". It features two input fields: "Email" and "Contraseña" (Password). Below these is a "Remember me" checkbox. At the bottom is a blue "Log In" button.

Fig. 10.10: Página de inicio de sesión para administradores.

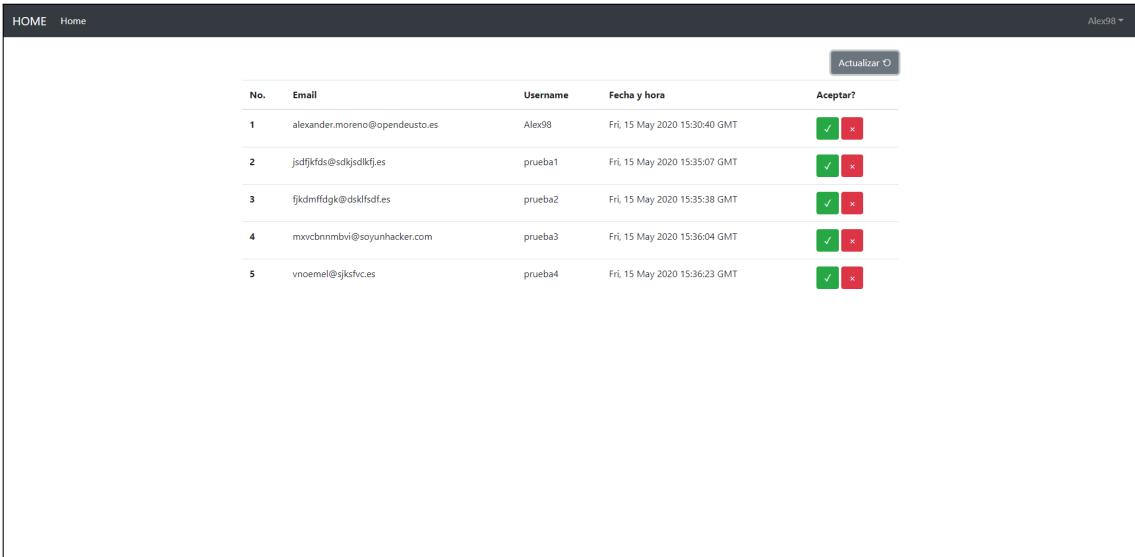
La cuenta de administrador es necesario que se haya creado manualmente en la BD ya que no hay ningún sistema de registro de administradores implementado en la aplicación. Una vez que se hayan introducido unas credenciales válidas, el sistema enviará un mensaje de correo electrónico a la dirección vinculada con la cuenta con la que se está intentando iniciar sesión. Dentro del mensaje se podrá encontrar un código de 6 caracteres que habrá que introducir en el panel que nos aparecerá tras haberle dado al botón *Log In*. Es preciso mencionar que el código es de un solo uso por lo que si se cancela el proceso de login por cualquier motivo habría que introducir el último código que se haya recibido, quedando inutilizado cualquier código anterior.



The image shows a modal window titled "Código de autenticación". It contains a single input field labeled "Código:" with the placeholder text "Introduce el código que ha llegado a tu correo electrónico.". At the bottom are two buttons: "Cancelar" (Cancel) and "Aceptar" (Accept).

Fig. 10.11: Panel donde hay que introducir el código de seguridad.

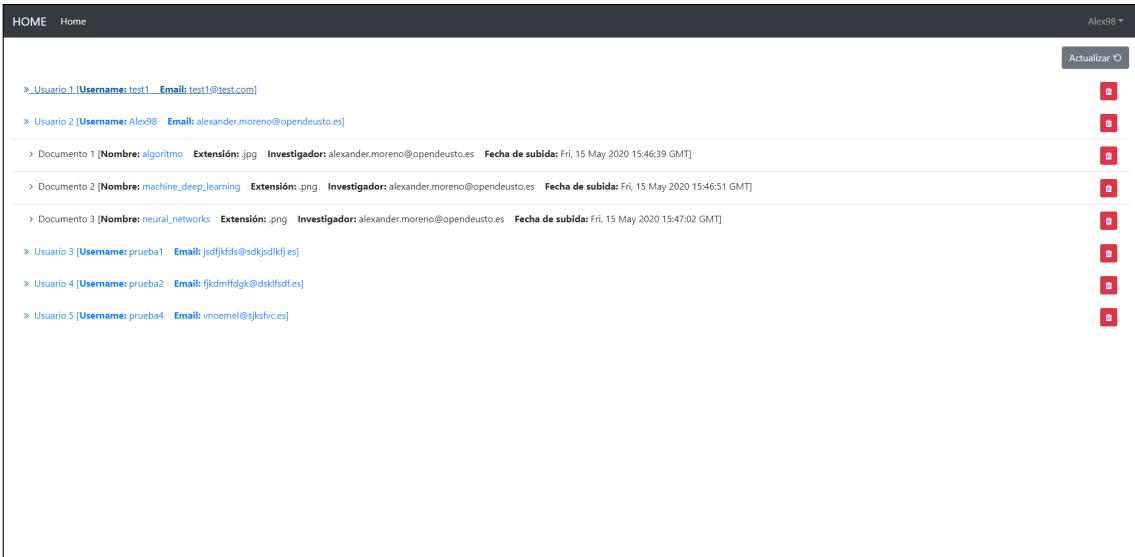
Tras iniciar sesión correctamente el administrador accederá a la página principal de administradores donde le aparecerá un listado con todas las solicitudes de investigadores que quieren crearse una cuenta. A la derecha de cada solicitud se encontrarán 2 botones con los que se podrá aceptar o rechazar la solicitud en cuestión.



No.	Email	Username	Fecha y hora	Aceptar?
1	alexander.moreno@opendeusto.es	Alex98	Fri, 15 May 2020 15:30:40 GMT	 
2	jsdfjkfds@sdkjsdlkj.es	prueba1	Fri, 15 May 2020 15:35:07 GMT	 
3	fjkmffdgk@dsklf sdf.es	prueba2	Fri, 15 May 2020 15:35:38 GMT	 
4	mxvcbnnmbvi@soyunhacker.com	prueba3	Fri, 15 May 2020 15:36:04 GMT	 
5	vnoemel@sjksfc.es	prueba4	Fri, 15 May 2020 15:36:23 GMT	 

Fig. 10.12: Página principal de administradores.

Al igual que los investigadores, los administradores tiene un menú desplegable en la parte derecha de la barra superior. En el caso de los administradores, tienen 2 opciones: cerrar sesión o ir a otra página en la que podrán gestionar el contenido y las cuentas de investigadores almacenadas en el sistema. En esta última página, se mostrará un listado formado por todos los investigadores registrados en la aplicación, y haciendo click encima de un investigador es posible visualizar todos los documentos que haya subido a la plataforma. Además, los títulos de los documentos será enlaces a los documentos de tal forma que los administradores podrán visualizarlos para verificar si es un contenido adecuado para la plataforma.



» Usuario 1 [Username: test1 Email: test1@test.com]	
» Usuario 2 [Username: Alex98 Email: alexander.moreno@opendeusto.es]	
> Documento 1 [Nombre: algoritmo Extensión: jpg Investigador: alexander.moreno@opendeusto.es Fecha de subida: Fri, 15 May 2020 15:46:39 GMT]	
> Documento 2 [Nombre: machine_deep_learning Extensión: png Investigador: alexander.moreno@opendeusto.es Fecha de subida: Fri, 15 May 2020 15:46:51 GMT]	
> Documento 3 [Nombre: neural_networks Extensión: png Investigador: alexander.moreno@opendeusto.es Fecha de subida: Fri, 15 May 2020 15:47:02 GMT]	
» Usuario 3 [Username: prueba1 Email: jsdfjkfds@sdkjsdlkj.es]	
» Usuario 4 [Username: prueba2 Email: fjkmffdgk@dsklf sdf.es]	
» Usuario 5 [Username: prueba4 Email: vnoemel@sjksfc.es]	

Fig. 10.13: Página de gestión para administradores.

## 10.ANEXO I: MANUAL DE USUARIO

En todas las páginas excepto en las de visualización de documentos, habrá un botón mediante el que se podrá actualizar el idioma de la aplicación. El símbolo del botón en cuestión es una especie de globo terráqueo azul como se puede ver en las siguientes capturas:



Fig. 10.14: Botón de cambio de idioma para usuarios.

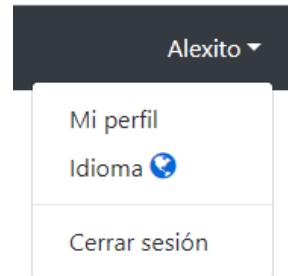


Fig. 10.15: Botón de cambio de idioma para investigadores y administradores.

Una vez que le demos al botón de cambiar de idioma, se abrirá una ventana en la que podremos seleccionar el idioma deseado entre los que estén disponibles. Para aplicar los cambios bastaría con darle al botón *Guardar*.



Fig. 10.16: Ventana para cambiar de idioma.

Por último, la siguiente captura muestra el aspecto que tiene la página de visualización de un documento cuando el archivo se puede mostrar directamente en la propia página sin necesidad de descargarlo:

```
VOLVER

/*
 * Vue.js v2.6.10
 * (c) 2014-2019 Evan You
 * Released under the MIT License.
 */
(function (global, factory) {
  'use strict';
  if (typeof exports === 'object' && typeof module !== 'undefined') {
    module.exports = factory();
  } else if (typeof define === 'function' && define.amd) {
    define(factory);
  } else {
    global.Vue = factory();
  }
})('use strict');

var emptyObject = Object.freeze({});

// These helpers produce better VM code in ES engines due to their
// explicitness and function inlining.
function isObject (v) {
  return v === undefined || v === null
}

function isDef (v) {
  return v !== undefined && v !== null
}

function isTrue (v) {
  return v === true
}

function isFalse (v) {
  return v === false
}

/***
 * Check if value is primitive.
 */
function isPrimitive (value) {
  return (
    typeof value === 'string' ||
    typeof value === 'number' ||
    // $flow-disable-line
    typeof value === 'symbol' ||
    typeof value === 'boolean'
  )
}

/* Quick object check - this is primarily used to tell
 * objects from primitive values when we know the value
 * is a JSOM-compliant type.
 */
function isObject_(obj) {
  return obj !== null && typeof obj === 'object'
}

/***
 * Get the raw type string of a value, e.g., [object Object].
 */
function rawType (value) {
  return Object.prototype.toString.call(value)
}
```

Fig. 10.17: Página de visualización de un documento.

## 10.2. PUESTA EN MARCHA DE LA APLICACIÓN EN LOCALHOST

Antes que nada, para poner en marcha la aplicación adecuadamente, es necesario asegurarse de que se tienen instaladas varias herramientas. Por un lado, para el frontend, hay que instalar *Node.js* y su gestor de paquetes *npm*. Cabe destacar, que normalmente *npm* ya viene incluido en la propia instalación de *Node*. Por otro lado, para el backend, básicamente se necesitaría *Docker*.

A continuación, es necesario descargar el proyecto, en mi caso de Github, de forma que tengamos el código del proyecto en nuestro equipo. Una vez hecho esto podemos proceder a poner en marcha la aplicación en localhost.

### 10.2.1. Frontend

Primero de todo, para poner en marcha el frontend tenemos que acceder a la carpeta llamada *frontend* situada dentro del directorio del proyecto desde una consola. Una vez dentro, ejecutaremos el siguiente comando de npm con el fin de instalar todos los paquetes necesarios para continuar con la puesta en marcha y asegurar su buen funcionamiento:

```
npm install
```

Cuando termine la instalación, ejecutamos el siguiente comando que generará un directorio llamado *build* en el que se incluirán todos los archivos necesarios para servir la aplicación:

```
npm run build
```

## 10.ANEXO I: MANUAL DE USUARIO

A partir de este punto podemos servir el frontend de diferentes formas. Una de ellas podría ser utilizando un paquete de npm llamado *serve*. Simplemente habría que instalarlo con *npm install serve* y ejecutarlo con *serve -s build*. También se le podría indicar el puerto en el que queremos servir la aplicación añadiendo una opción *-p <port>* al comando anterior. La otra opción, que es la que se ha elegido en este proyecto, consiste en utilizar un servidor web. El servidor web utilizado para servir la aplicación ha sido Nginx debido a sus características que lo hacen mejor que otras opciones disponibles, como por ejemplo su alto rendimiento, su capacidad de optimizar el uso de los recursos del sistema, su gran comunidad, etc.

Para servir la aplicación con Nginx, primero tenemos que instalarlo mediante el comando que aparece a continuación:

```
sudo apt-get install nginx
```

Cuando lo tengamos instalado, abriremos el archivo de configuración */etc/nginx/sites-enabled/default* donde modificaremos varios campos de tal forma que Nginx pueda servir el frontend sin problemas. El contenido de dicho archivo en cuestión sería así:

```
server {  
    listen 80 default_server;  
    listen [::]:80 default_server;  
    root <Ruta directorio build>;  
  
    index index.html index.htm index.nginx-debian.html;  
  
    server_name _;  
  
    location / {  
        try_files $uri /index.html;  
    }  
}
```

Es preciso señalar que hay que sustituir *<Directorio build >* por la ruta del directorio *build* que hemos creado anteriormente con el comando *npm run build*. Hecho esto, habría que reiniciar el servicio de nginx de manera que se actualicen los cambios realizados en el archivo de configuración.

```
sudo service nginx restart
```

Con esto ya podríamos visitar la página web abriendo cualquier navegador y escribiendo en la barra de direcciones *127.0.0.1* o *localhost*. Es posible también conectarse al frontend desde otros equipos conectados en la misma LAN introduciendo en la barra de direcciones la IP privada del equipo donde se aloja el servidor. Por último, cabe mencionar que, en caso de haber actualizado el código del frontend, es necesario ejecutar de nuevo el comando *npm run build* para que se vean

reflejados los cambios en Nginx, y por consiguiente, en la página web.

### 10.2.2. Backend

La puesta en marcha del backend del proyecto consiste en lanzar 3 contenedores de software con *Docker*, 1 por cada microservicio implementado. Dentro del directorio de cada microservicio se puede encontrar un *Dockerfile* con el que se pueden crear imágenes de Docker con las que lanzar los contenedores en cualquier momento. En los Dockerfiles, se incluye toda la configuración necesaria para instalar en los contenedores todos los paquetes requeridos de forma que puedan inicializarse correctamente y sin errores. Además, se define también el puerto que se va a exponer en el exterior del contenedor. En el caso del microservicio de usuarios el puerto expuesto es el 5000, en el microservicio de investigadores el 5001 y en el microservicio de administradores el 5002. A continuación se muestra el contenido de uno de los Dockerfile ya que los 3 son muy similares:

```
FROM ubuntu:18.04

WORKDIR /app

ADD . /app

RUN apt-get update && apt-get install -y python3 python3-pip

RUN pip3 install --trusted-host pypi.python.org -r requirements.txt

EXPOSE 5000

CMD ["python3", "waitress_server.py"]
```

Como los Dockerfiles ya están incluidos en el proyecto, no es necesario realizar ninguna acción extra sobre ellos. Cabe destacar, que como Flask no dispone de un servidor de producción propio se ha optado por utilizar *Waitress* ya que se puede lanzar con un script sencillo de Python. Con los Dockerfiles, primero tendríamos que crear las imágenes de los 3 microservicios. Para ello, ejecutamos el siguiente comando:

```
sudo docker build -t <nombre_imagen> <ruta_dockerfile>
```

En el comando anterior tenemos que sustituir los campos que aparecen entre signos de mayor y menor que, con el nombre de la imagen que queramos poner (p. ej. *users-microservice*) y la ruta del Dockerfile correspondiente. En mi caso, a la hora de crear la imagen suelo ejecutar directamente el comando desde el directorio donde se encuentra el Dockerfile, de esta manera solo tengo que poner “.” en la ruta.

Una vez que tengamos las 3 imágenes, solo quedaría lanzar los microservicios. Sin embargo, antes de hacer eso, es necesario disponer de una cuenta de *MongoDB Atlas*, un servicio en la nube que

## 10.ANEXO I: MANUAL DE USUARIO

nos permite poner en marcha un cluster alojado en AWS, Azure o Google Cloud Platform. En dicho cluster es posible crear varias bases de datos de Mongo. Para este proyecto, se ha utilizado su versión gratuita que, aunque tenga ciertas limitaciones, suple perfectamente las necesidades de almacenamiento de la aplicación web desarrollada.

Para conectar el cluster y las 2 bases de datos creadas con los microservicios, se puede solicitar un enlace que se debe incluir en la configuración de la aplicación de Flask (de cada uno de ellos), al igual que si se tuviera la base de datos en local. De esta manera, se pueden ejecutar consultas a las bases de datos desde el propio código.

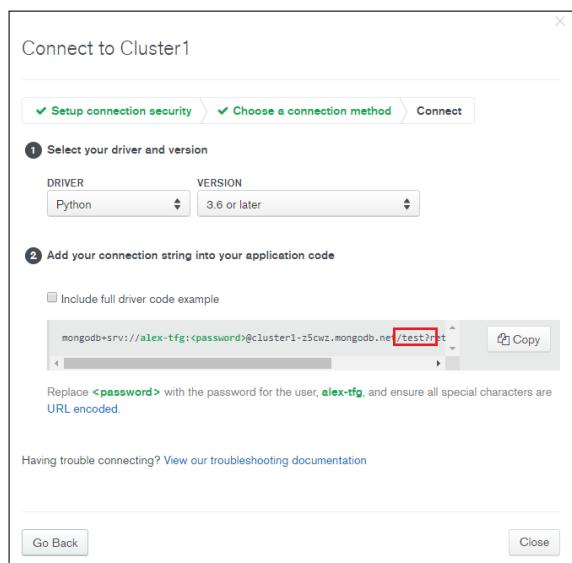


Fig. 10.18: MongoDB Atlas - Enlace para conectar el cluster con la aplicación.

Para conectar cada base de datos, es necesario modificar la parte del enlace marcada con un recuadro en rojo en la imagen anterior. En vez de *test* hay que poner el nombre de la base de datos en cuestión. Para poder conectarla correctamente, hay que crear un usuario en MongoDB Atlas que tenga permisos de lectura y escritura sobre el cluster. Este usuario tendrá asignado un nombre y una contraseña que serán las que incluiremos en el enlace comentado anteriormente (sustituyendo los valores en verde que se pueden ver en la captura anterior). Además, es importante también que modifiquemos la configuración de MongoDB Atlas para que pueda recibir conexiones desde cualquier IP o dominio.

A screenshot of the MongoDB Atlas 'Database Access' page under 'Database Users'. It shows a table with one row: User Name: 'alex-tfg', Authentication Method: 'SCRAM', and MongoDB Roles: 'readWriteAnyDatabase@admin'. There are 'EDIT' and 'DELETE' buttons at the bottom right. A green button at the top right says '+ADD NEW DATABASE USER'.

Fig. 10.19: MongoDB Atlas - Creación de usuario con permisos sobre el cluster.

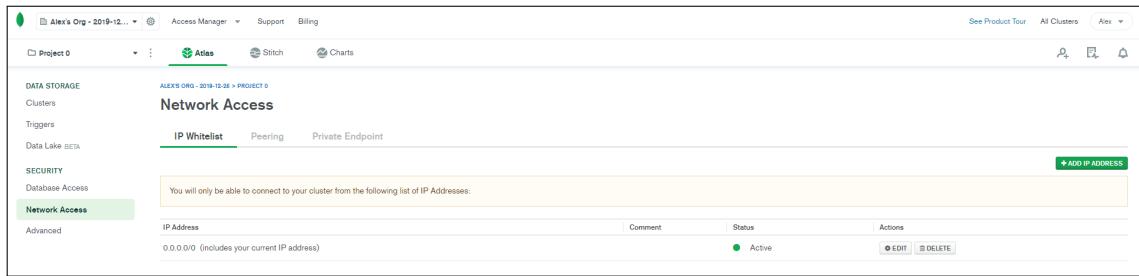


Fig. 10.20: MongoDB Atlas - Modificar acceso de red a MongoDB Atlas.

Una vez que hecho todo esto, podemos proceder con el lanzamiento de los microservicios. A continuación se muestran los 3 comandos que hay que ejecutar (en diferentes consolas) para poner los microservicios en marcha.

Microservicio de usuarios:

```
sudo docker run -e MONGO_USER=<valor> -e MONGO_PASS=<valor> -p 5000:5000 <nombre_imagen>
```

Microservicio de investigadores:

```
sudo docker run -e MONGO_USER=<valor> -e MONGO_PASS=<valor> -e SECRET_KEY=<valor> -p 5001:5001 <nombre_imagen>
```

Microservicio de administradores:

```
sudo docker run -e MONGO_USER=<valor> -e MONGO_PASS=<valor> -e SECRET_KEY=<valor> -e GMAIL_EMAIL=<valor> -e GMAIL_PASS=<valor> -p 5002:5002 <nombre_imagen>
```

En los comandos, se incluyen múltiples variables de entorno a las que hay que asignarles el usuario y contraseña del cluster de MongoDB Atlas, una clave secreta que podríamos generar aleatoriamente con un programa externo, y las credenciales de la cuenta de correo electrónico que será la que envíe notificaciones a los usuarios que utilicen la aplicación (al confirmar el registro de un investigador o cuando un administrador inicie sesión en la aplicación).

Cuando hayamos ejecutado los 3 comandos, tendremos todos los microservicios funcionando en localhost, por lo que ya podremos utilizar la aplicación en su totalidad conectándonos a la página del frontend desde el navegador.

Por último, se incluyen algunos comandos generales de Docker que pueden ser muy útiles para gestionar las imágenes y contenedores que hayamos creado:

- Listar todos los contenedores (operativos y detenidos): `docker ps -aq`
- Detener todos los contenedores: `docker stop $(docker ps -aq)`
- Eliminar todos los contenedores: `docker rm $(docker ps -aq)`
- Eliminar todas las imágenes: `docker rmi $(docker images -q)`

### 10.3. DESPLIEGUE DE LA APLICACIÓN EN LA NUBE

El despliegue de la aplicación en la nube, como se ha dicho anteriormente, se ha llevado a cabo en Heroku. El proceso a seguir es bastante sencillo y no es necesario realizar muchos cambios en los archivos del proyecto.

Primero de todo, es necesario crearse una cuenta en Heroku. Con ella, podremos iniciar sesión, lo que nos permitirá crear aplicaciones en la plataforma que más adelante conectaremos con varios repositorios de Github. A la hora de crear una aplicación en Heroku solo nos pedirá que introduzcamos el nombre de la aplicación y la región en la que nos encontramos.

En este caso, tenemos que crear 4 aplicaciones en Heroku y 4 repositorios en Github (3 para los microservicios y 1 para el frontend). Dentro de cada repositorio habrá que subir todos los archivos referentes a los microservicios y el frontend respectivamente. Además, tendremos que incluir dos archivos en todos ellos para que Heroku sea capaz de desplegarlos correctamente.

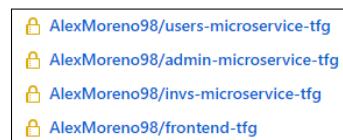


Fig. 10.21: Repositorios de Github para desplegarlos en la nube.

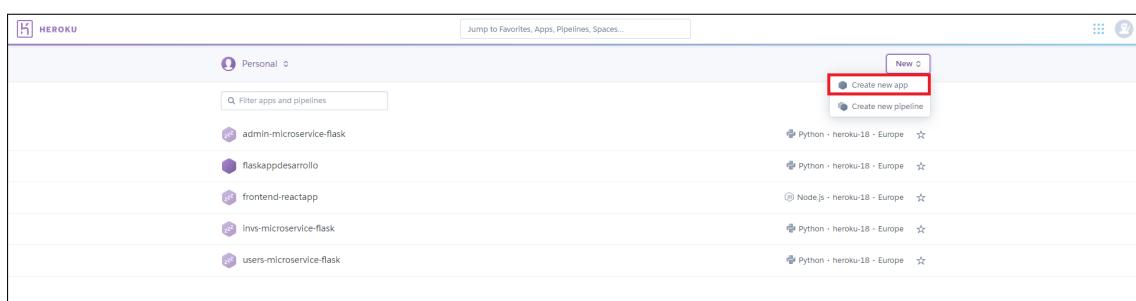


Fig. 10.22: Heroku - Crear una aplicación en la plataforma.

En el caso de los microservicios hay que añadir un archivo *requirements.txt*, en el que se definen las dependencias y paquetes que tiene que instalar Heroku para que funcione bien la aplicación, y un *Procfile*, en el que se especifican los comandos que ejecutará la aplicación en su puesta en marcha. El contenido de estos dos archivos es el mismo para los 3 microservicios, y se puede ver a continuación.

*requirements.txt*:

---

```
bcrypt==3.1.7
cffi==1.14.0
click==7.1.2
cryptography==2.9.2
dnspython==1.16.0
Flask==1.1.2
```

```

Flask-Bcrypt==0.7.1
Flask-Cors==3.0.8
Flask-PyMongo==2.3.0
gunicorn==20.0.4
itsdangerous==1.1.0
Jinja2==2.11.2
MarkupSafe==1.1.1
pycparser==2.20
PyJWT==1.7.1
pymongo==3.10.1
six==1.14.0
Werkzeug==1.0.1

```

*Procfile:*

```
web: gunicorn app:app
```

Para el despliegue de los microservicios se ha empleado otro servidor de producción distinto a *Waitress* llamado *Gunicorn*, ya que tras investigar sobre cómo desplegar aplicaciones Flask en Heroku, la mayoría de las alternativas encontradas lo utilizaban. Incluso en la propia guía publicada por el *Heroku Dev Center* lo usan [44].

Respecto al frontend, también tenemos que incluir en su repositorio de Github dos archivos. Por un lado, un *Procfile*, similar al de los microservicios, y por otro lado, un *package.json* en el que se definen las dependencias que necesita la aplicación para funcionar adecuadamente, similar al *requirements.txt* pero con otra estructura.

*package.json:*

```
{
  "name": "frontend",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^4.2.4",
    "@testing-library/react": "^9.5.0",
    "@testing-library/user-event": "^7.2.1",
    "all": "0.0.0",
    "aws-sdk": "^2.639.0",
    "axios": "^0.19.2",
    "bcrypt": "^4.0.1",
    "bcryptjs": "^2.4.3",
    "bootstrap": "^4.4.1",
    "jquery": "^3.5.0",
    "lodash": "^4.17.15",
    "node-gyp": "^6.1.0",
  }
}
```

```

    "node-pre-gyp": "^0.14.0",
    "popper.js": "^1.16.1",
    "react": "^16.13.1",
    "react-bootstrap": "^1.0.0-beta.17",
    "react-dom": "^16.13.1",
    "react-router-dom": "^5.1.2",
    "react-scripts": "3.4.1",
    "react-test-renderer": "15.4.1",
    "serve": "^11.3.0",
    "typescript": "^2.9.2"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": "react-app"
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}

```

*Procfile:*

```
web: serve -s build
```

El archivo de dependencias es distinto en los repositorios de los microservicios respecto al del frontend, ya que el lenguaje de programación y el framework utilizado es distinto. Para instalar las dependencias necesarias en Python hay que hacerlo a partir de un *requirements.txt*. En Node.js en cambio, a partir de un *package.json*.

Una vez que tengamos los 4 repositorios de Github bien aprovisionados, solo tendríamos que conectar dichos repositorios con las aplicaciones creadas en Heroku previamente. Para ello, hacemos

click encima de una de las aplicaciones, accedemos a la pestaña *Deploy* y en el campo *Deployment method* seleccionamos la opción de Github. Una vez hecho esto, tendremos que autenticarnos con nuestras credenciales de Github, además de seleccionar el repositorio que queremos conectar.

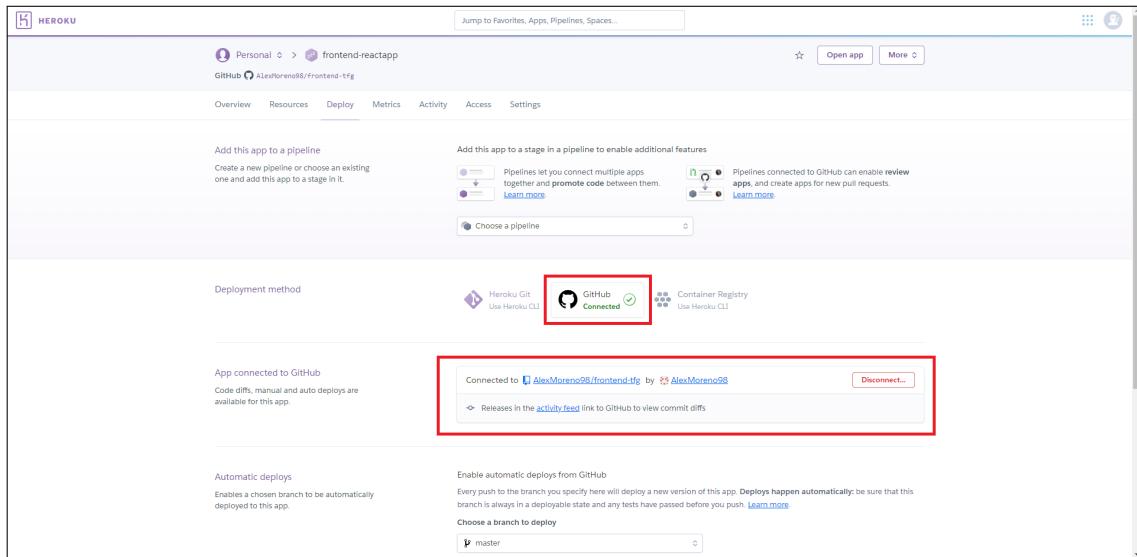


Fig. 10.23: Heroku - Conectar aplicación con un repositorio de Github.

Posteriormente, abajo del todo de la página seleccionaremos la rama del repositorio que queramos desplegar en Heroku, y por último, pulsaremos el botón *Deploy Branch*.

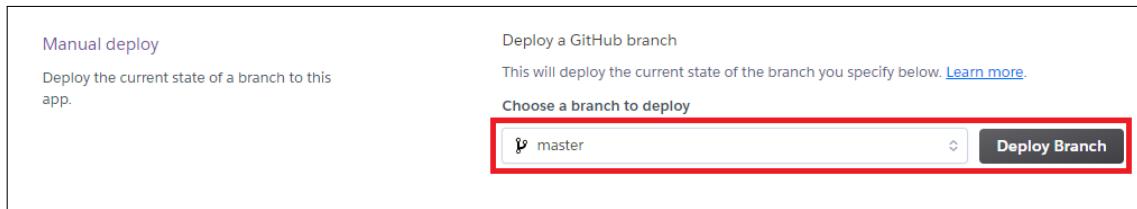


Fig. 10.24: Heroku - Desplegar rama de un repositorio de Github vinculado con Heroku.

Una vez finalizado el proceso, nos aparecerá un botón para abrir la aplicación desplegada en una nueva pestaña del navegador. Si todo ha ido bien, nos cargará la aplicación en cuestión.

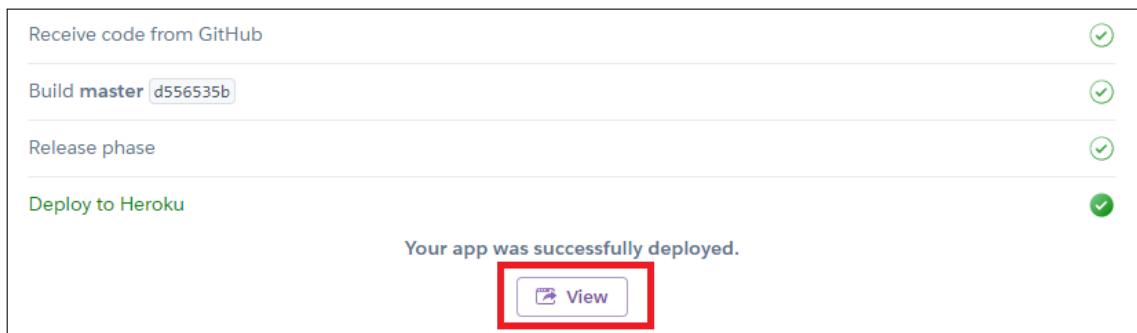


Fig. 10.25: Heroku - Acceder a una aplicación desplegada.



## Capítulo 11

# ANEXO II: DIMENSIÓN ÉTICA DEL PROYECTO

En este anexo, se incluyen varios apartados que representan la dimensión ética del proyecto, es decir, el análisis que en todo proyecto de ingeniería debería hacerse gracias al que se pueden identificar los impactos (positivos y negativos) que puede causar la herramienta o tecnología que se haya desarrollado, así como las carencias éticas que puedan haber surgido durante la realización del proyecto.

El anexo se organiza como se define a continuación. Primero, se cumplimenta mi propio cuestionario de evaluación social de tecnologías en el que se detectan y valoran los impactos que tiene la herramienta desarrollada sobre la sociedad y su entorno. En segundo lugar, se analiza el proyecto en su totalidad detectando e intentando mitigar las carencias éticas que puedan identificarse. Y por último, en tercer lugar, ya que el proyecto ha sido llevado a cabo por mí, se comprueba que se haya cumplido mi propia ética profesional personal durante todo el proceso seguido desde el inicio hasta la finalización del mismo.

### 11.1. APLICACIÓN DE MI CUESTIONARIO DE EVALUACIÓN SOCIAL DE TECNOLOGÍAS

El cuestionario de evaluación social de tecnologías sirve para analizar una tecnología o herramienta con el objetivo de saber si puede considerarse éticamente justificable y correcta. Para ello, se dispone de un cuestionario de más de 150 preguntas que tratan una gran variedad de ámbitos: político, económico, legal, cultural, industrial, tecnocientífico, medioambiental, educación, internacional, regional, salud, filosófico, teológico o religioso, histórico, humanístico, social, ético y militar.

En este apartado se recogen los resultados y conclusiones que se han extraído tras la realización del cuestionario. Se puede acceder al documento donde se encuentran cumplimentadas todas las preguntas que componen el cuestionario visitando el siguiente enlace: [Enlace](#). Para identificar los impactos a partir de las respuestas dadas a las preguntas del cuestionario hay que ir analizando cada pregunta junto con su respuesta. Si consideramos que la respuesta nos indica la presencia de un impacto positivo o negativo lo añadimos a la lista. Cabe destacar que la mayoría de respuestas

## 11. ANEXO II: DIMENSIÓN ÉTICA DEL PROYECTO

son negativas (“No”), lo que implica que no hay impactos positivos ni negativos en lo que respecta al tema tratado en la pregunta en cuestión. Por ejemplo, en la pregunta “¿Promueve la diversidad cultural?” el que la respuesta sea negativa, no quiere decir que la aplicación o tecnología atente contra la diversidad cultural.

A continuación, y tras haber analizado las respuestas del cuestionario se recogen en 2 tablas los impactos detectados, así como la valoración final de la herramienta en cuestión. En la segunda tabla se recogen los impactos identificados. En ella, se pueden incluir tanto impactos positivos como negativos, aunque si se quisiera se podrían recoger solo los negativos. En el apartado “Valoración”, si el impacto es negativo tendrá valores más bajos y si es positivo valores más altos.

En la primera tabla, tras analizar las respuestas del cuestionario y de la segunda tabla se decide si la tecnología es éticamente favorable o desfavorable teniendo en cuenta sobre todo el campo “Valoración”, viendo si hay más impactos positivos que negativos y, valorando si los impactos negativos son aceptables o no respecto a los positivos. Además, en esta segunda tabla se puede detallar algo más el apartado de “Gestión de impactos negativos”.

PROPIUESTA DE RESOLUCIÓN FINAL		
<b>Favorable</b> / Desfavorable		
Ámbito	Impacto negativo	Gestión de impactos negativos
Legal o económico	Compartición de documentos adquiridos ilegalmente	Añadir más seguridad a la hora de subir contenido como documentos, artículos, etc. Por ejemplo, implementando una funcionalidad que solo permita que se publiquen archivos una vez que los administradores de la aplicación les den el visto bueno.
Legal	Compartición de documentos que no son de autoría propia	A la hora de subir un archivo, documentos..., solicitar una prueba que asegure que la persona que está subiendo el documento en cuestión es su autor, o en su defecto, posee permiso del autor para compartirlo. En caso de que el documento sea de acceso libre, no hay problema alguno.
Medioambiental	Gasto excesivo de electricidad	Utilizar los mínimos equipos posibles a la hora de desplegar o poner en marcha el sistema y aprovechar al máximo los recursos disponibles en los equipos de los que se dispone en el momento de tal manera que no pongan a funcionar nuevos equipos si no es completamente necesario.

Tabla 11.2: Valoración final de la herramienta a partir del cuestionario social de evaluación de tecnologías.

Tras la realización, análisis y valoración del cuestionario podemos considerar esta herramienta como **favorable**, ya que analizando tanto impactos positivos como negativos se puede observar como los positivos tienen una mayor relevancia y un mayor efecto que los negativos. Además, lo más probable es que los impactos negativos no ocurran mientras que los positivos sí.

SÍNTESIS DE IMPACTOS							
Identificación de impactos		Análisis de impactos				Conclusión	
Ámbito	Impacto	Probabilidad	Severidad	Tipo (P/N)	Duración	Valoración (0 - 10)	Gestión / Solución
Legal o económico	Compartición de documentos adquiridos ilegalmente	Baja	Media - alta	N	Breve	2	Añadir más seguridad a la hora de subir contenido como documentos.
Legal	Compartición de documentos que no son de autoría propia	Baja	Media	N	Breve	3	Asegurar que la persona que suba un documento tenga permiso del autor.
Medioambiental	Beneficia al medio-ambiente respecto al CO2 (no produce CO2)	Alta	Media - alta	P	Indefinida	10	-
Medioambiental	Gasto excesivo de electricidad	Baja	Baja	N	Indefinida	4	Utilizar los mínimos equipos posibles y aprovechar al máximo los recursos disponibles.
Educativo	Fomenta la compartición de conocimiento (útil para estudiantes)	Alta	Media	P	Indefinida	8	-
Medioambiental	Posibilidad de utilizar equipos o servidores fabricados con componentes reciclados	Media	Media - alta	P	Breve	9	-
Tecnológico	Reutilizar software	Baja - media	Baja - media	P	Larga	7	-

Tabla 11.1: Síntesis de impactos detectados con el cuestionario social de evaluación de tecnologías.

## 11. ANEXO II: DIMENSIÓN ÉTICA DEL PROYECTO

Por último, como aclaración decir que la tabla que recoge todos los impactos identificados debería aparecer antes que la propuesta de resolución final. Sin embargo, el orden de las tablas se ha definido de esta manera ya que se ha considerado como la mejor opción para que el apartado tuviera una buena organización y no se generasen demasiados espacios en blanco.

### 11.2. ANÁLISIS Y VALORACIÓN ÉTICA DEL PROYECTO

En este apartado recojo las conclusiones que he sacado del análisis ético del proyecto. Dicho análisis se centra en varios aspectos que se consideran relevantes, como por ejemplo, los criterios en los que se ha basado para la toma de decisiones durante su ejecución, los criterios para la selección de tecnologías, los costes de todo tipo que haya supuesto el proyecto, los resultados o consecuencias que pueda tener, etc.

En las fases iniciales del proyecto, se ha realizado una planificación y una recopilación de las necesidades, problemáticas, objetivos y requisitos de la aplicación a desarrollar. De esta manera, se han podido definir las características que debería tener el sistema de la forma más precisa posible. En este aspecto, se puede decir que en el proceso de selección de tecnologías y entornos de trabajo, no se disponían de restricciones fijas, por lo que se han tenido en cuenta otros criterios para su elección. Algunos de estos criterios han sido: si eran de código libre, su idoneidad para este proyecto, su importancia en el mercado, etc.

La persona que ha llevado a cabo el proyecto, actúa como experto y no busca un beneficio personal, al contrario, intenta satisfacer las necesidades y objetivos del proyecto por encima de todo. A continuación aparecen algunas cuestiones que prueban esto mismo:

- Tras finalizar la etapa de desarrollo, se hacen pruebas a la aplicación de tal forma que se garantice su buen funcionamiento, rendimiento y seguridad. En caso de que el rendimiento o los resultados no sean los deseados se deja claro que se rediseñará el sistema para así solucionar los problemas que pueda tener.
- Algunos criterios que se han utilizado para la selección de tecnologías son: si son 100 % de código libre, su funcionalidad, interacción y comparación con otras tecnologías, nivel de uso por parte de la comunidad de desarrolladores, sus limitaciones, rendimiento... En definitiva, se ha procurado seleccionar las tecnologías y herramientas que tengan mayor rendimiento y mejores resultados para cumplir con los objetivos del proyecto, teniendo en cuenta también si son de acceso libre o propietario.
- La solución que se describe en la documentación del proyecto es la que se ha considerado como válida y la versión más completa que se ha podido crear.

La bibliografía del proyecto está formada por 40 referencias aproximadamente. Tras echarles un vistazo a los apartados de la memoria que pueden ser considerados como los que mejor referenciados deberían estar, se puede afirmar que todos ellos están referenciados adecuadamente respetando

así los derechos de autor.

Cabe destacar que no he identificado ninguna decisión o acción que podría ser considerada como éticamente incorrecta. Además, en el diálogo que haya podido darse durante el transcurso del proyecto, han estado presentes los tres sujetos y principios primordiales que en todo diálogo profesional deberían darse como son el experto o el principio de beneficencia (desarrollador), el cliente o el principio de autonomía (en este caso, el director del proyecto) y la autoridad competente o el principio de justicia (universidad). Por lo tanto, podemos concluir que el diálogo “profesional” que se ha llevado a cabo en este caso ha sido exitoso y éticamente correcto.

Respecto al cumplimiento de la ley de garantías de derechos digitales [45] se puede decir que no hay indicios que prueben que se incumple alguno de los derechos que la conforman. Aunque es preciso destacar que es complicado comprobar si se cumplen o no, ya que no se detalla demasiado el uso que le da el sistema a los datos con los que trabaja y no trata temas relacionados con los derechos que aparecen en la ley como por ejemplo: los derechos digitales en el ámbito laboral, si se eliminarán o modificarán los datos de un usuario que así lo deseé, la protección de menores, etc. Por lo tanto, se podría concluir que no se dispone de suficiente información acerca de esa cuestión para afirmar o negar rotundamente si se cumple o no la ley de garantías de los derechos digitales. Sería una muy buena práctica que en cualquier proyecto informático se lleve a cabo una reflexión y se tome en consideración esta ley para así desarrollar sistemas que la cumplan en su totalidad, proporcionando datos e información en la documentación del proyecto sobre los pasos seguidos para su cumplimiento.

En relación a los costes del proyecto, decir que no se han tenido costes personales ni materiales excesivos. Por lo tanto, se puede afirmar que se han aprovechado lo máximo posible los recursos disponibles y no se ha contaminado el medio ambiente innecesariamente.

Por último, decir que el hecho de tener una visión ética del proyecto es realmente importante a la hora de desarrollar las distintas funcionalidades del sistema objetivo ya que ayuda a decidir cómo implementarlas de manera que se eviten problemas futuros y se cumplan todos los principios del código ético del desarrollador. Sobre este proyecto, se puede decir que se ha hecho el esfuerzo de analizarlo éticamente de diversas formas (cuestionario social de tecnologías, análisis y valoración ética y mi propia ética profesional personal), y plasmar todas las conclusiones extraídas en este anexo.

### **11.3. APLICACIÓN DE MI PROPIA ÉTICA PROFESIONAL PERSONAL**

En este apartado se incluyen los principios y normas que componen mi propia ética profesional personal que pueden ser aplicables a este proyecto, al igual que una conclusión en la que se valora el nivel de cumplimiento de dichas normas durante todo el transcurso y realización del proyecto.

### 11.3.1. Principios éticos

- **Unión:** es fundamental mantenerse unido tanto con la familia como con los compañeros de trabajo de manera que podamos ayudarnos y apoyarnos mutuamente en momentos o frente a situaciones difíciles. Este principio está muy ligado a otro llamado “Trabajo en equipo”.
- **Honradez:** ser leal y cumplir con el deber en el trabajo buscando un bien común para todas las personas que me rodean (sociedad, familia, amigos, compañeros de trabajo...) sin distinciones (de raza, sexo, religión, cultura...).
- **Solidaridad:** buscar el bien de las demás personas sin buscar un beneficio propio exclusivamente. Lo mismo ocurre en el mundo empresarial, en el que no hay que buscar satisfacer solo las necesidades de la empresa donde se trabaja.
- **Responsabilidad:** ser responsable de mis actos tanto en mi vida personal como profesional, reflexionando sobre las consecuencias positivas o negativas que podrían tener antes de la toma de decisiones. Este principio es realmente importante ya que podemos hacer algo que perjudique muy gravemente a la sociedad actual o a futuras generaciones.
- **Formación continua:** el mundo de la informática está en constante cambio por lo que una persona que se dedique a trabajar en cualquier área relacionada debe ir adquiriendo nuevos conocimientos durante toda su carrera profesional.
- **Preservación del medio ambiente:** cuidar el medio ambiente, contaminando lo menos posible, reciclando, siendo respetuoso con él... en cualquier sitio donde me encuentre (casa, trabajo, calle).
- **Trabajo en equipo:** trabajar en equipo tanto con los componentes de mi familia como con los compañeros de trabajo de manera que podamos afrontar los problemas que surjan rápida y eficazmente, minimizando los efectos negativos que puedan causar. Además, este principio normalmente ayuda a que la actividad que se esté desarrollando se haga mucho mejor que en solitario por lo que se potencian y maximizan los efectos positivos que pueda proporcionar su realización.
- **Lealtad:** actuar de manera respetuosa y serle fiel a los demás, evitando situaciones de traición a toda costa. Lo mismo ocurre con la empresa en la que se trabaje, hay que serle leal siempre y cuando esta, no participe en la realización de actividades ilegales o alejadas de la ética. En tal caso, la lealtad hacia la empresa no debería de darse.
- **Independencia:** evitar que me manipulen otras personas, tomando decisiones y actuando en base a mis principios éticos y morales. Además, en situaciones de disputa hay que mantener la imparcialidad y no dejarse influir por presiones externas, sobornos, chantajes, etc.
- **Dignidad:** actuar siguiendo mis códigos de honor y mi propia ética profesional en el en el trabajo.
- **Sinceridad:** decir la verdad respecto a nuestras capacidades, conocimientos, actos o experiencia, evitando así mentir o realizar publicidad engañosa con el único fin de conseguir beneficio propio.

- **Buena conducta:** procurar siempre tener una conducta apropiada, respetuosa y digna de la situación en la que se esté.
- **Defensa de la propiedad intelectual:** no apropiarse de las ideas, creaciones o aportes realizados por otras personas. Es esencial referenciar toda la información extraída de fuentes externas.
- **Legalidad:** velar por el cumplimiento de todas las leyes y políticas vigentes en el ejercicio profesional, al igual que las normas corporativas. En caso de presenciar un acto que incumpla alguna ley o norma se debe denunciar ante la autoridad competente.
- **Libertad del cliente:** hay que dejarle tomar sus propias decisiones y ofrecerle múltiples soluciones factibles donde poder elegir. De la misma forma, en caso de que el cliente lo vea conveniente hay que permitirle que contrate al profesional que él desee. Además, hay que evitar situaciones de manipulación o en las que se le ofrezca una única solución (que no sea la más adecuada) aprovechando su desconocimiento sobre la materia.
- **Intereses del cliente:** a los clientes hay que ofrecerles las soluciones que más se adecuen a sus necesidades, sin aprovecharse de la situación para venderle peores soluciones (para resolver su problema) por mero beneficio económico o personal. En otras palabras, hay que priorizar los intereses del cliente frente a los propios o los de la organización.

### 11.3.2. Normas de actuación

Cabe destacar que las normas de actuación se han ideado en base a los principios de la sección anterior, y se han dividido en varios ámbitos que tienen relación con la profesión de un ingeniero.

#### 11.3.2.1. La profesión

- Como ingeniero debo cumplir con mi cometido en el trabajo de manera eficaz promoviendo las buenas prácticas que considere que cualquier ingeniero informático debería tener.
- Debo promover y compartir entre las personas de mi alrededor la importancia que tiene la ingeniería informática, así como mis obligaciones y responsabilidades. No solo con mis familiares y amigos, sino también con profesionales, clientes y expertos del campo de la informática.
- Debo ejercer la profesión con pasión intentando disfrutar de todas las experiencias y vivencias que surjan a lo largo de mi carrera profesional siempre que sea posible.
- Debo apoyar a otras personas que se dediquen a la misma profesión que yo, en este caso a la ingeniería informática.
- Debo buscar un equilibrio entre los intereses profesionales y personales, evitando poner uno de ellos por encima del otro.
- Debo conocer todas las leyes vigentes ligadas a la ingeniería informática o a la informática en general de modo que las aplique y cumpla en todo momento. Además, tengo que preocuparme de ir actualizando mis conocimiento sobre ellas ya que con el tiempo surgirán nuevas leyes y se modificarán las ya existentes.

## 11. ANEXO II: DIMENSIÓN ÉTICA DEL PROYECTO

- Debo evitar y denunciar cualquier situación de corrupción.
- Debo demostrar mi disconformidad y denunciar las situaciones en las que no se cumplan las leyes, normativas vigentes o los valores morales y éticos propios de los ingenieros informáticos.
- Es fundamental también que, como ingeniero, me aleje de actividades criminales que llevan a cabo personas que aprovechan sus conocimientos avanzados para piratear, hackear, infectar... máquinas de otras personas con el fin de obtener un beneficio económico. Para minimizar los impactos que estos ataques puedan tener debo ayudar a personas o entidades sin conocimiento informático a proteger y definir medidas de seguridad para sus equipos.
- Tengo que ser competente, honesto, buscar un bien común, no apoyar la tecnocracia ni la dependencia tecnológica y no caer en los errores del paternalismo o la anarquía.
- Tengo que ser fiel a mis principios y pensar independientemente del resto cuando considere que las cosas se están haciendo inadecuadamente, tratando de evitar así cualquier tipo de manipulación.
- Debo de aplicar mi cuestionario de evaluación de tecnologías a cualquier proyecto, herramienta, tecnología o aplicación que desarrolle para así saber si es factible éticamente. Además, tengo que ir mejorando y completando el cuestionario a lo largo de mi carrera profesional con el objetivo de conseguir resultados más precisos y fiables.

### 11.3.2.2. Otros profesionales con los que interactúo

- Como ingeniero debo respetar y tratar justamente las opiniones o metodologías que utilicen otros profesionales sin faltarles al respeto, agredirles o menospreciarles. Si en algún momento quiero expresar mi desacuerdo lo haré con un lenguaje claro, conciso, respetuoso y con razones bien argumentadas.
- Debo mantener una relación cordial y positiva, además de intercambiar información con todos los profesionales o compañeros con los que trabaje con el fin de mejorar nuestro desempeño laboral.
- Debo ser amable, cooperar y no menospreciar a profesionales que se dediquen a otro ámbito diferente al mío.
- Debo estar orgulloso de mis propios méritos, pero nunca debo atribuirme méritos de otros profesionales como si fueran míos, ni interferir en su desarrollo profesional por mero beneficio personal o sentimientos de envidia.
- Debo solicitar y aceptar la ayuda de otros profesionales cuando en un proyecto se requieran conocimientos concretos de los que carezco.
- Debo apoyar a otros profesionales cuando expresen su rechazo a prácticas éticamente inaceptables, ilegales o inadecuadas.

### 11.3.2.3. Los usuarios o interesados en mis servicios profesionales

- Como ingeniero debo cumplir la ley de protección de datos en su totalidad, procurando mantener segura la información de los usuarios que utilicen aplicaciones que haya desarrollado.

Además, al igual que en el subapartado anterior, debo mantener el secreto profesional para con los clientes en caso de disponer de información referente a ellos. Aunque en ciertas circunstancias, se podría incumplir el secreto profesional siempre y cuando esté justificado.

- Debo publicitar mis servicios de forma sincera, es decir, especificando realmente los conocimientos y la educación que poseo evitando inventar o mentir en este aspecto.
- Debo ser capaz de organizar todos los proyectos que esté desarrollando e identificar la carga de trabajo total que puedo soportar, para así no aceptar más trabajos de la cuenta que perjudiquen al desarrollo de los proyectos que actualmente tenga en curso.
- No debo ofrecer servicios profesionales en los que utilice programas o herramientas obtenidas de manera ilegal. Siempre debo comprar la licencia requerida para el uso de este tipo de recursos.
- Debo tener una buena relación con usuarios de tal manera que podamos llevar a cabo las actividades que nos competen de la manera más beneficiosa para todos.
- Como ingeniero tomar decisiones libremente en base a mi experiencia y conocimiento, evitando ser manipulado por clientes, usuarios, proveedores u otras entidades.

#### **11.3.2.4. La sociedad cercana y más amplia**

- Como ingeniero debo ser un profesional que actúe con responsabilidad ante la sociedad.
- Debo rechazar proyectos o actividades que generen impactos negativos que afecten a la sociedad, pongan en peligro la vida, la confidencialidad, el medio ambiente, etc.
- Debo de utilizar los conocimientos que he adquirido durante mi carrera estudiantil y profesional para el bien, es decir, para ayudar a la sociedad y buscar un futuro mejor para las próximas generaciones.
- Debo respetar a todos los miembros de la sociedad en su conjunto, sin distinciones de ningún tipo y sin sentirme superior a ellos. A los grupos o colectivos de la sociedad más desfavorecidos se les podría ayudar o apoyar de alguna forma especial frente a otros grupos que disponen de condiciones normales.
- Debo promover y fomentar los proyectos que tengan como principal objetivo el bien público.
- Debo promover la realización de proyectos desarrollados mediante herramientas o plataformas de libre acceso para que cualquier ciudadano interesado pueda acceder a ellas sin restricciones.
- Si en alguna ocasión tengo que hablar en público, debo dar información real, verdadera y justificada evitando cualquier tipo de engaño, farsa o manipulación social.
- Debo velar porque cualquier grupo o colectivo de personas del mundo pueda acceder a la tecnología cualesquiera que sean su capacidades económicas o físicas, su lugar de procedencia... Además, tengo que tener claro que actualmente no todo el mundo tiene acceso a la tecnología y esto hace que se produzca una mayor distinción entre las sociedades de distintos lugares del mundo.

## 11. ANEXO II: DIMENSIÓN ÉTICA DEL PROYECTO

- Debo procurar que proyectos, aplicaciones o sistemas que desarrolle no sean causa de una destrucción de empleo injustificada.
- Tengo que ser responsable en la toma de decisiones y en la forma de actuar, teniendo en cuenta todas las consecuencias provocadas por mis actos en cualquier ámbito, pero sobre todo en el ámbito social.
- Debo ser consciente de el poder que tiene la tecnociencia sobre la ciudadanía de forma que pueda evitar tomar decisiones que provoquen efectos negativos en ella.

### 11.3.2.5. La naturaleza y el medio ambiente

- Como ingeniero debo intentar siempre que sea posible aprovechar, reutilizar y reciclar materiales, herramientas...
- Debo rechazar proyectos que considere perjudiciales para el medio ambiente y denunciarlos cuando supongan un peligro muy grande para las personas o sus alrededores.
- Tengo que ser responsable para con el medio ambiente y desarrollar soluciones lo más sostenibles posibles.

### 11.3.2.6. Los productos y soluciones desarrolladas

- Como ingeniero debo desarrollar soluciones en el menor tiempo y con el menor coste posible, asumiendo mi errores en el proceso y evitando ocultarle información relevante a la empresa.
- Debo analizar los productos que desarrolle con una visión ética teniendo en mente las distintas consecuencias que pueden tener en la mayor cantidad de ámbitos posibles (económico, político, social, laboral...). De esta forma, puedo identificar sus carencias éticas y decidir si vale la pena continuar con su desarrollo.
- Debo elegir las metodologías de trabajo dependiendo de las tareas que haya que llevar a cabo de manera que se aprovechen al máximo sus ventajas y se consiga el mejor resultado posible.
- Debo escribir una documentación que describa el producto en su totalidad (tal y como es), sin incluir especificaciones o características falsas. Además, sería muy recomendable incluir uno o varios apartado dedicados a la ética ya que es un factor realmente importante a tener en cuenta en cualquier desarrollo, proyecto o trabajo.
- Si realizo pruebas con usuarios debo asegurarme que su información se mantiene oculta y segura. Asimismo, tengo que ofrecerles la posibilidad de abandonar las pruebas en cualquier momento.
- En toda documentación debo añadir todas las referencias utilizadas durante el proceso de desarrollo del producto para así cumplir con los derechos de autor.
- Debo desarrollar soluciones fiables, seguras y eficientes que cumplen con las expectativas y los requisitos acordados con los interesados. Para asegurar que tienen estas características, hay que aplicar pruebas de calidad al producto final.
- Debo mantener informado al director del proyecto durante todo el proceso de desarrollo de un producto o solución.

- Debo describir los sistemas que desarrolle tal y como son, con sus características y funcionalidades reales, procurando no proporcionar información falsa.

#### **11.3.2.7. La ética profesional ingenieril y personal**

- Debo mejorar la forma en la que analizo los posibles impactos que pueda tener un proyecto antes de decidir si participar en él, de tal manera que recoja la mayor cantidad de impactos e información posible.
- Debo procurar cumplir con las normas definidas en mi ética profesional personal en cualquier situación que pueda darse en mi trabajo, y si alguna vez por alguna razón no cumple alguna de ellas, debo aprender de mis errores para no repetirlos. Además, debo actualizar el listado de normas continuamente durante el transcurso de mi carrera profesional, para así disponer de una base ética mucho más completa a la hora de resolver dilemas morales que surjan.
- Debo mejorar como persona, compañero y profesional procurando cumplir con todos los principios y valores que he descrito anteriormente en este anexo. Al igual que con las normas, el listado de valores hay tengo que ir actualizándolo y completándolo a lo largo de mi carrera profesional.
- No debo influenciar a nadie para que realice acciones ilegales, alejadas de la ética o que puedan perjudicarle a él o la sociedad.
- Debo tener presente en todo momento que la tecnología y la ciencia hay que trabajarlas con responsabilidad ya que pueden convertirse en un arma que se vuelva contra nosotros (los seres humanos) y el planeta.

#### **11.3.3. Conclusión valorativa**

Todo ingeniero informático debería disponer de su propio código de actuación en el que basarse a la hora de afrontar cualquier tipo de proyecto. Muchos profesionales le dan poca importancia a la perspectiva ética y no tienen en cuenta conceptos o factores éticos que se puedan aplicar a los proyectos en los que participen.

Respecto a la aplicación de mi ética profesional personal en el proyecto, puedo decir que durante todas las fases que lo componen se han tenido en cuenta tanto los principios como las normas de actuación que se han comentado en los apartados anteriores. De esta forma, se ha podido desarrollar la aplicación y redactar la memoria del proyecto, cumpliendo con la ética en todas sus formas.

Algunos aspectos que vale la pena destacar y que deberían cumplirse sí o sí en todo proyecto serían:

- La obligación de referenciar todos los datos e información extraída de fuentes externas, de tal forma que se cumplan los derechos de autor.
- El análisis de todas las posibles consecuencias e impactos que pueden darse tras la realización de un proyecto, tarea o actividad.
- La protección de la sociedad y el medio ambiente.

## 11. ANEXO II: DIMENSIÓN ÉTICA DEL PROYECTO

- El ser un buen profesional a la hora de interactuar con otros ingenieros o profesionales de cualquier sector.

Es posible que haya cuestiones que no se hayan tenido en cuenta debido a que la ética es un campo muy extenso. Sin embargo, como otro punto de trabajo futuro se podría incluir la tarea de mejorar y completar la dimensión ética del proyecto, así como mi propia ética profesional personal.