

# Monitorización de sistemas con microservicios contenerizados.

**Abel Pérez Guardado**

Grado de Ingeniería Informática  
Java EE

**Albert Grau Perisé**

**Santi Caballe Llobet**

Enero 2022

## **Agradecimientos:**

*Gracias a mi hermano por su saber y ser mi maestro a lo largo de más de 25 años explorando la informática.*

*Gracias a mi compañera y esposa por compartir la alegría cada vez que quedaba un poquito menos para acabar esta aventura.*

*Gracias a mis padres por apoyarme en todo lo que he necesitado para poder estudiar y formarme.*

*Gracias a la UOC por facilitarme y ayudarme a ser cada día un poquito más sabio.*

*“E somentes preservando as nosas enerxías autóctonas, a nosa capacidade creadora, é como poderemos contribuír á civilización universal incorporando a ela as nosas creacións inéditas.” (Rodríguez Castelao, Alfonso Daniel (1944), Sempre en Galiza)*



Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada [3.0 España de Creative Commons](https://creativecommons.org/licenses/by-nc-nd/3.0/es/)

## FICHA DEL TRABAJO FINAL

<b>Título del trabajo:</b>	<i>Monitorización de sistemas con microservicios contenerizados</i>
<b>Nombre del autor:</b>	<i>Abel Pérez Guardado</i>
<b>Nombre del consultor/a:</b>	<i>Albert Grau Perisé</i>
<b>Nombre del PRA:</b>	<i>Santi Caballe Llobet</i>
<b>Fecha de entrega (mm/aaaa):</b>	01/2022
<b>Titulación:</b>	<i>Grado en ingeniería informática</i>
<b>Área del Trabajo Final:</b>	<i>Java EE</i>
<b>Idioma del trabajo:</b>	<i>Castellano</i>
<b>Palabras clave</b>	<i>Spring Boot, React, Docker</i>
<p><b>Resumen del Trabajo (máximo 250 palabras):</b> <i>Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.</i></p> <p>Cada día el número de aplicaciones de software que maneja una empresa son mayores. Estos aplicativos y sistemas deben estar disponibles 24x7x365 y para ello es necesaria una supervisión para poder realizar acciones preventivas. Esto es posible con un aplicativo que unifique su supervisión y sea capaz de monitorizarlos.</p> <p>Se desarrolla un aplicativo fácilmente escalable y tolerante a fallos para la monitorización de aplicaciones y sistemas a partir de los eventos y logs que estos generan. El objetivo principal es ofrecer un panel de administración al personal de administración y un punto común de monitorización al personal de soporte mediante eventos generados con reglas y condiciones.</p> <p>Para la construcción de este proyecto se ha seguido una de desarrollo en cascada [1] con sus diferentes fases: análisis, diseño, codificación, pruebas y despliegue. Durante la fase de codificación se han incorporado metodologías de desarrollo ágil con iteraciones y <i>Github Flow</i> [2] y tecnologías como Spring Boot [3], Maven [4] y Docker [5] para facilitar su construcción y despliegue.</p> <p>El resultado ha sido una aplicación de monitorización con microservicios contenerizada compuesto por 7 imágenes de contenedores Docker [3] cumpliendo con los requisitos de usuario definidos y con con los requisitos generales de escalabilidad y tolerancia a fallos.</p>	

**Abstract (in English, 250 words or less):**

Every day the number of software applications handled by a company are greater. These systems and applications should be available 24x7x365 and it's necessary a supervision to be able to do preventive actions. This is possible with an application that unifies their supervision, and it monitors them.

A scalable and fault-tolerant application is developed, it's monitors events and logs of applications and systems. The main objective is to provide a manage panel to administration staff and to provide a monitoring common point to support staff through events generated with rules and conditions.

To build this project a cascade development methodology [1] has been followed with his different phases: analysis, design, coding, testing and deployment. Agile methodologies like iterations and *Github Flow* [2] have been incorporated during coding phase. Spring Boot [3], Maven [4] and Docker [5] technologies gave been used to make easier building and deploying phases.

As a result, an a containerized monitoring application with microservices has been achieved. The application consist in seven Docker [3] images and it meets defined user requirements and scalability and fault-tolerance requirements.

# Índice

1. Introducción.....	1
1.1 Contexto y justificación del Trabajo.....	1
1.2 Objetivos del Trabajo.....	1
1.3 Enfoque y método seguido.....	3
1.4 Planificación del Trabajo .....	3
1.5 Breve resumen de productos obtenidos .....	5
1.6 Breve descripción de los otros capítulos de la memoria.....	5
2. Resto de capítulos.....	7
2.1 Casos de uso y prototipos .....	7
CUG001 Seguridad en intercambio de mensajería.....	7
CUA0001 Appender Log4j .....	7
CUM0001 Servicio de autenticación .....	8
CUM0002 Guardar línea/s log para un token válido de agente .....	8
CUM0003 Servicio de healthy.....	8
CUM0004 Servicio de shutdown.....	9
CUW0001 Acceso de usuario al sistema.....	9
CUW0002 Listado de últimos agentes con notificaciones .....	10
CUW0003 Listado filtrado de usuarios creados .....	11
CUW0004 Alta y modificación de usuario .....	12
CUW0005 Listado filtrado sistemas creados .....	13
CUW0006 Alta y modificación sistema .....	14
CUW0007 Listado filtrado de agentes creados.....	15
CUW0008 Alta y modificación de agente.....	16
CUW0009 Alta y modificación de regla de agente.....	17
CUW0010 Monitorización de agentes.....	18
CUW0011 Detalle de regla de agente .....	19
2.2 Punto de vista de la información .....	20
2.3 Punto de vista de computacional y de ingeniería .....	21
Capas del sistema .....	21
Capa de microservicios .....	22
Capa de presentación.....	22
Componente de log4j-appender .....	23
Diagrama de casos de uso .....	24
Diagrama de componentes.....	24
Diagrama de componentes detallado .....	25
Diagrama de componentes de microservicio detallado.....	26
Base de datos .....	27
2.4 Punto de vista tecnológico.....	29
Tipos de módulos Maven a usar .....	29
Entorno de desarrollo.....	30
2.5 Detalles importantes en fase de desarrollo .....	30
Uso de GitHub y GitHub Flow .....	30
Uso de Maven.....	31
Uso de Docker y Docker-compose .....	33
Uso de node.js y Yarn.....	35

Eclipse y proyecto Maven multimodular .....	36
2.6 Despliegue del aplicativo .....	39
Compilación y generación de imágenes .....	39
Arranque con Docker-compose .....	42
Prueba de tolerancia a fallos.....	43
2.7 Producto final .....	44
Perfil de agente.....	44
Perfil de administrador .....	45
Gestión de usuarios.....	46
Gestión de sistemas .....	47
Gestión de agentes.....	48
Perfil de soporte.....	50
2.8 Pruebas .....	51
Pruebas manuales .....	52
CUG001 Seguridad en intercambio de mensajes.....	52
CUA001 Appender Log4j.....	52
CUM0001 Servicio de autenticación.....	52
CUM0002 Guardar línea/s log para un token válido de agente .....	52
CUW0001 Servicio de autenticación .....	53
CUW0002 Listado de últimos agentes notificados.....	53
CUW0003 Listado filtrado de usuarios creados.....	53
CUW0004 Alta y modificación de usuario.....	53
CUW0005 Listado filtrado sistemas creados .....	54
CUW0006 Alta y modificación sistema .....	54
CUW0007 Listado agentes creados .....	54
CUW0008 Alta y modificación de agente .....	54
CUW0009 Alta y modificación de regla de agente .....	55
CUW0010 Monitorización de agentes .....	55
CUW0011 Detalle de regla de agente .....	55
Pruebas automatizadas sobre código Java .....	56
Módulo common .....	56
Módulo microservicio de autenticación .....	56
Módulo microservicio de usuarios.....	57
Módulo microservicio de sistemas .....	57
Módulo microservicio de agentes .....	58
Módulo microservicio de log .....	58
3. Conclusiones.....	59
4. Glosario .....	61
5. Bibliografía .....	62
6. Anexos .....	64
6.1 Manual de instalación de aplicativo .....	64
Instalación de Docker-desktop.....	64
Configuración de Docker-compose.....	64
Arranque de microservicios con Docker-compose.....	67

## Lista de figuras

Ilustración 1: Esquema aplicación	2
Ilustración 2: Planificación parte 1	4
Ilustración 3: Planificación parte 2	5
Ilustración 4: Prototipo CUW0001	10
Ilustración 5: Prototipo CUW0002	11
Ilustración 6: Prototipo CUW0003	12
Ilustración 7: Prototipo CUW0004	13
Ilustración 8: Prototipado CUW0004	14
Ilustración 9: Prototipo CUW0006	15
Ilustración 10: Prototipo CUW0007	16
Ilustración 11: Prototipo CUW0008	17
Ilustración 12: Prototipo CUW0009	18
Ilustración 13: Prototipo CUW0010	19
Ilustración 14: Prototipo CUW0011	20
Ilustración 15: Sistema de información V1	20
Ilustración 16: Sistema de información V2	21
Ilustración 17: Esquema de aplicativo React	23
Ilustración 18: Diagrama de casos de uso	24
Ilustración 19: Ampliación de los casos de uso.	24
Ilustración 20: Diagrama de componentes	25
Ilustración 21: Diagrama de componentes V2	25
Ilustración 22: Diagrama de componentes refinado	26
Ilustración 23: Diagrama de componentes refinado V2	26
Ilustración 24: Componente de microservicio detallado	27
Ilustración 25: Diagrama de documentos MongoDB V1	28
Ilustración 26: Diagrama de documentos MongoDB V2	29
Ilustración 27: Log de GitHub.	30
Ilustración 28: Diagrama de GitFlow	31
Ilustración 29: Pom.xml parent	32
Ilustración 30: Pom.xml de módulo con referencia al pom.xml padre	32
Ilustración 31: Fichero mongo.yml	34
Ilustración 32: Arranque de mongoDb con Docker-compose	34
Ilustración 33: Panel de gestión mongo exprés.	35
Ilustración 34: Panel de administración Docker-desktop	35
Ilustración 35: Arranque de servidor con Yarn de manera local	36
Ilustración 36: Pantalla de aplicativo React en servidor Yarn local	36
Ilustración 37: Project explorer de Eclipse	37
Ilustración 38: Creación de nuevo modulo	37
Ilustración 39: Selección de arquetipo	38
Ilustración 40: Modo debug de un microservicio	38
Ilustración 41: Consola de log del microservicio	39
Ilustración 42: Postman consultando el microservicio de autenticación	39
Ilustración 43: Consola de Debian con el proyecto descargado	40
Ilustración 44: Arranque de compilación en Debian	40
Ilustración 45: Diferentes fases de compilación en Debian	41
Ilustración 46: Resultado de la compilación en Debian.	41

Ilustración 47: Imágenes en Debian	42
Ilustración 48: Arranque de aplicativo contenerizado en Debian	42
Ilustración 49: Contenedores inicializados en Debian	43
Ilustración 50: Home del módulo administración en Debian	43
Ilustración 51: Autoarranque de microservicio auth caído	44
Ilustración 52: Log4j.xml con appenders configurados	45
Ilustración 53: Imagen con el log enviado por los agentes.	45
Ilustración 54: Panel de gestión de usuarios	46
Ilustración 55: Pantalla de edición de usuario	47
Ilustración 56: Filtrado de sistemas	47
Ilustración 57: Edición de un sistema	48
Ilustración 58: Filtrado de agentes en el sistema	48
Ilustración 59: Edición de agente 1	49
Ilustración 60: Edición de regla de agente	49
Ilustración 61: Edición de condición de una regla	50
Ilustración 62: Pantalla principal con perfil de soporte	51
Ilustración 63: Gráfica de eventos con perfil de soporte	51
Ilustración 64: Cobertura de módulo common	56
Ilustración 65: Cobertura de módulo de microservicio de autenticación	57
Ilustración 66: Cobertura de módulo de microservicio de usuarios	57
Ilustración 67: Cobertura de módulo de microservicio de sistemas	58
Ilustración 68: Cobertura de módulo de microservicio de agentes	58
Ilustración 69: Cobertura de módulo de microservicio de log	59



# 1. Introducción

## 1.1 Contexto y justificación del Trabajo

Es un hecho palpable que el aumento del número de aplicaciones de software que manejan las empresas es cada día mayor: una página web, un ERP o CRM, sistemas de robotización, etc. Grandes empresas invierten muchos recursos en hacer que estos aplicativos y sistemas estén disponibles 24x7x365, pero no pueden evitar que se produzcan errores. Por desgracia, cuando esto ocurre, suele repercutir negativamente en la empresa.

Una manera de minimizar posibles fallos es disponer de un equipo humano encargado de la supervisión y mantenimiento de dichos sistemas que actúen de manera proactiva evitando que lleguen a producirse errores graves. Para ello, estas personas necesitan un software de monitorización que unifique la lectura de logs, detecte errores, la degradación en los servicios o fallos graves en los sistemas.

El principal problema para satisfacer esta necesidad es disponer de un aplicativo que sea escalable, tolerante a fallos y que sea adaptable a cualquier tipo de sistema ya existente. Si bien existen ofertas de software con estas características, no cubren el 100% de las necesidades, ofrecen características que no son necesarias u obligan a enviar datos sensibles de la empresa a servicios de terceros. Esto provoca que su uso no sea rentable ni adecuado.

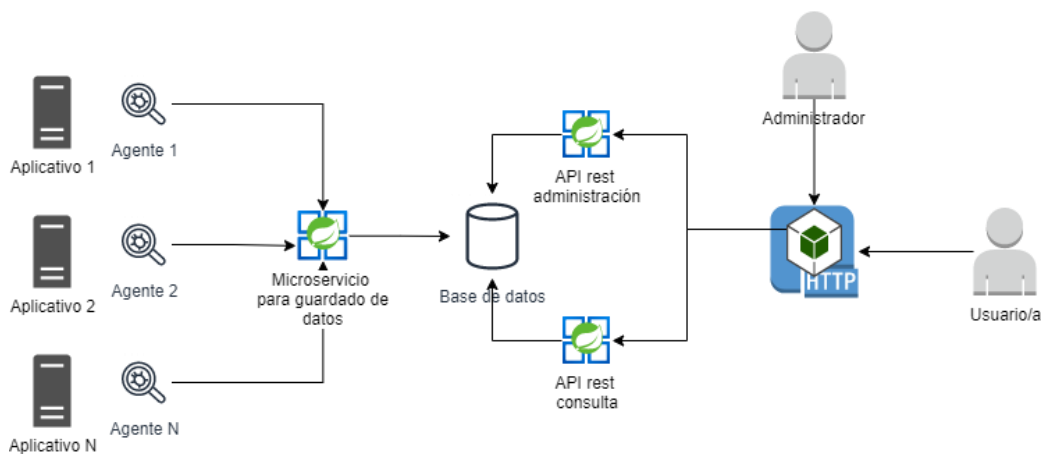
Recurrir a un desarrollo a medida como el que se propone a continuación es la mejor solución dado que permite adaptarse mejor a las necesidades y a los cambios que se puedan proponer a futuro. Además, se asegura de que los datos sensibles no sean tratados por servicios de terceros y los costes de escalabilidad de los sistemas son controlados directamente por el cliente.

## 1.2 Objetivos del Trabajo

El objetivo principal consistirá en la construcción de un sistema de monitorización de aplicaciones basado en microservicios contenerizados dado que ofrece escalabilidad y redundancia. Ser escalable permite adaptarse fácilmente a un n.º elevado de sistemas a monitorizar y, ser redundante facilita disponer de un aplicativo de monitorización preparado contra fallos.

El sistema de monitorización estará compuesto por las siguientes partes:

- Un subsistema de recolección de información de los diferentes aplicativos mediante un agente autorizado. Este agente enviará los eventos de *log* que se produzcan en el sistema que está monitorizando.
- Un subsistema de almacenaje de la información recolectada al que se conectarán los agentes autorizados y se encargará de procesar y guardar los datos.
- Un subsistema de administración que permita la gestión de los usuarios, los agentes y sus reglas de análisis.
- Un subsistema de presentación en tiempo real de alertas y gráficas de medición de los sistemas/aplicativos monitorizados.



**Ilustración 1: Esquema aplicación**

Existirán dos roles principales en el aplicativo. Los **administradores** podrán dar de alta o modificar usuarios o crear o modificar agentes. El alta/modificación de agente implica la generación de las diferentes reglas y condiciones de análisis de log para el sistema pueda generar eventos. Entre las características principales de estas reglas se encuentra:

- Filtrar líneas de log mediante expresiones regulares [4].
- Que el conjunto de líneas cumpla una serie de condiciones, entre otras:
  - Que el evento suceda N veces en un determinado tiempo T
  - Que el valor numérico de la línea a analizar sea mayor que la media en un tiempo T.
  - Que el valor numérico de la línea a analizar sea superior a un valor.
  - Que el valor de la línea contenga un valor en concreto.

Por otro lado, existen el **rol de soporte**, que podrá visualizar de diferentes maneras los eventos generados por el sistema para un determinado sistema o sistemas (agrupación de agentes) en un determinado periodo.

### 1.3 Enfoque y método seguido

Las metodologías más adecuadas para construir un aplicativo hoy en día son algún tipo de metodología ágil o un desarrollo en cascada.

El primero es un desarrollo incremental, basado en historias de usuario en donde cada iteración requiere de un mínimo diseño, análisis, implementación y pruebas. Este tipo de metodología ofrece al cliente disponer del producto mucho antes que otro tipo de metodologías, pero no con su funcionalidad completa. Es más eficiente cuando el cliente no tiene totalmente definido el alcance y se desconocen posibles obstáculos que puedan influir en el desarrollo.

En cambio, el desarrollo en cascada invierte más tiempo en el análisis y diseño antes de comenzar la fase de implementación. Es lineal, es decir, se deben completar las fases previas antes de pasar a la siguiente. Este tipo de metodologías es más eficiente cuando se conoce perfectamente el alcance.

En el desarrollo del aplicativo será usado un desarrollo en cascada dado se es conocedor de todo su alcance, existe un objetivo marcado y consta de una fecha de entrega específica. No obstante, en la fase de implementación se usará el concepto de iteración de metodologías ágiles.

Gracias a usar iteraciones en el desarrollo se podrá obtener un producto mínimo ejecutable al final de cada una de ellas. Esto ofrecerá la oportunidad de probar y ver si lo que se ha analizado y diseñado se adapta a los requisitos o, por lo contrario, es necesario realizar algún cambio para que la solución final sea la más adecuada.

### 1.4 Planificación del Trabajo

El proyecto se desarrollará usando el lenguaje de programación J2EE con una arquitectura de microservicios. Los microservicios se encargarán de la lógica de negocio usando Spring [5] y expondrán un API Rest [6] para la administración y consulta de datos.

Para la capa de presentación se usará React [7] debido a que es más sencillo, ofrece mejor curva de aprendizaje que Angular [8].

Para la capa de persistencia se usará una MongoDB [9].

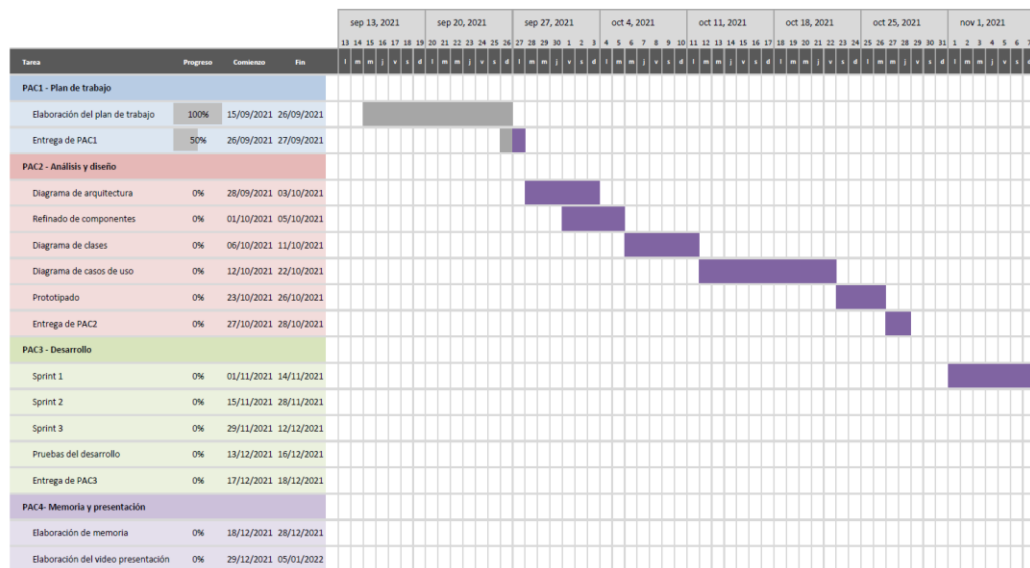
Otros *frameworks* y utilidades a usar:

- Log4j [10] para generación de logs y creación de agente.
- JavaWebToken [11] como sistema de seguridad en la API Rest [6].
- Expresiones regulares para extracción de información de logs [4].

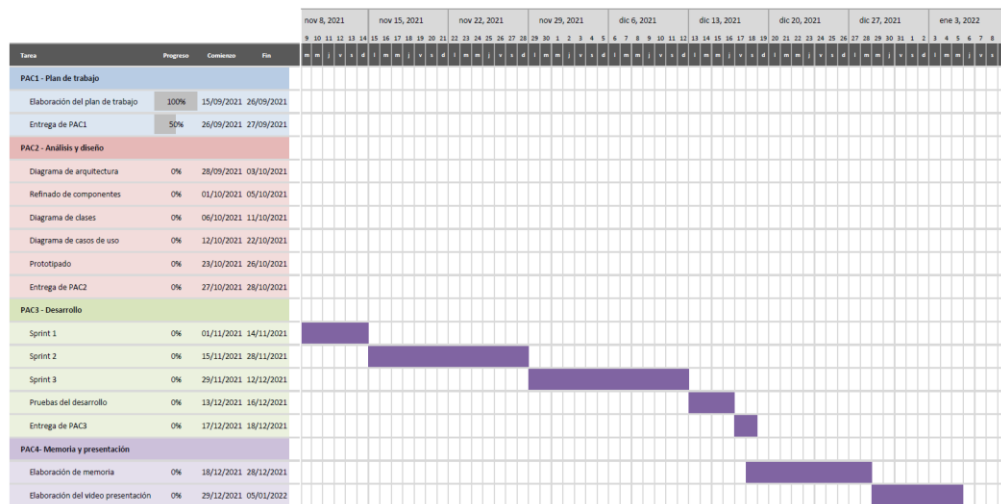
Para su desarrollo se usará:

- Justinmind [12] para la generación de la pantalla del prototipo.
- Drawio [13] para generación de diagramas.
- GitHub [14] como sistema de repositorio de código.
- Eclipse [15] como IDE de programación.
- Maven [16] como herramienta de gestión de proyectos para su compilación y despliegue usando plugins como fabric8 [17].
- VSCode [18] para programación de capa de presentación.
- Docker [3] para creación despliegue de base de datos y prueba de sistemas de microservicios contenerizados.

La planificación estará compuesta por 17 semanas dónde el desarrollo se ha dividido en iteraciones de 2 semanas. El objetivo de la primera iteración es disponer de un aplicativo mínimo que permita definir la estructura de un microservicio y tomarlo como ejemplo para la elaboración del resto. La siguiente iteración se dedicará a la construcción del módulo de administración, mejorando la curva de aprendizaje de React [7] elaborando formularios. La última iteración se dedicará a mejorar todos los formularios creados con lo aprendido, pruebas unitarias y sistema de despliegue automático.



**Ilustración 2: Planificación parte 1**



**Ilustración 3: Planificación parte 2**

## 1.5 Breve resumen de productos obtenidos

Como productos finales se han obtenido los siguientes productos:

- Archivo comprimido con el código fuente.
- Vídeo explicativo de desarrollo.
- Vídeo explicativo de despliegue en Docker mediante “Docker-compose”.
- Memoria final del trabajo.
- Presentación en PowerPoint.
- Vídeo con la presentación del trabajo final.

## 1.6 Breve descripción de los otros capítulos de la memoria

El siguiente capítulo contiene los requisitos de análisis y diseño para la construcción del software de Monitorización de sistemas con el objetivo principal de que sea altamente flexible, escalable, tolerante a fallos, fácil de mantener y que incluya alta cohesión con bajo acoplamiento entre sus diferentes componentes.

En primer lugar, serán definidos los casos de uso con sus prototipos centrándose sobre todo en los objetivos desde un punto de vista de empresa. A continuación, desde un punto de vista de la información se describirá la información a tratar por el sistema con sus diferentes estructuras de datos que darán soporte a los objetivos definidos en el primer punto. Desde un punto de vista computacional se descompondrán el sistema en diferentes componentes que interactuarán entre sí. Con el punto de vista de la ingeniería se especificará la infraestructura para soportar los componentes diseñados. Y, finalmente, desde un punto de vista de la tecnología se especificarán algunos *frameworks*, utilidades y software a usar durante el desarrollo y la puesta en producción.

Además, se amplía el punto de vista tecnológico y se especifican detalles importantes de la fase de desarrollo en cuanto al uso de Git, Maven, Docker y Node.js. Otro punto importante es el despliegue semiautomatizado en un servidor Debian Linux como demostración de cómo se puede desplegar el producto de manera sencilla, rápida como sistema contenerizado. Finalmente, se muestran detalles sobre el producto final, las pruebas manuales y pruebas automáticas realizadas.

## 2. Resto de capítulos

### 2.1 Casos de uso y prototipos

#### CUG001 Seguridad en intercambio de mensajería

CUG001 Seguridad en intercambio de mensajería	
Actor principal	Aplicativo web y agente de log
Nivel de objetivo	General
Precondición	Ninguna
Escenario principal de éxito	<ol style="list-style-type: none"><li>1. Los mensajes intercambiados con la capa de microservicios deben usar JWT [19].</li><li>2. Todo Token generado tendrá una fecha de caducidad no superior 60 segundos.</li><li>3. Se responde con el Token generado y Status code 200 [20].</li></ol>
Escenario alternativo	

#### CUA0001 Appender Log4j

CUA0001 Appender Log4j	
Actor principal	Aplicativo java externo con <i>framework</i> Log4j
Nivel de objetivo	Usuario
Precondición	Ninguna
Escenario principal de éxito	<ol style="list-style-type: none"><li>1. Se configura un nuevo <i>appender</i> en un fichero de configuración de log4j especificando el servidor y el token.</li><li>2. Por cada línea de <i>log</i> que el sistema intente escribir:<ol style="list-style-type: none"><li>a. Si han pasado más de N minutos configurados por parámetro, verificar que el token identificativo sigue siendo válido (CUM0001).</li><li>b. Enviar una petición post a una URL con el token identificativo y la línea de log (CUM0002).</li></ol></li></ol>
Escenario alternativo	<ol style="list-style-type: none"><li>1. Si el sistema devuelve que el token no es válido, el sistema no realizará POST de la información.</li><li>2. Si el sistema remoto no responde, el <i>appender</i> guardará un buffer de N líneas de log en memoria e intentará su reenvío cuando pueda conectarse nuevamente.</li></ol>

#### CUM0001 Servicio de autenticación

<b>CUM0001 Servicio de autenticación</b>	
Actor principal	Aplicativo web y agente de log
Nivel de objetivo	Usuario
Precondición	Ninguna
Escenario principal de éxito	<ol style="list-style-type: none"> <li>1. Se recibe un Token ya generado, un ID de agente o un email/clave encriptada para generar</li> <li>2. Se valida el token, el ID de agente o email/clave encriptada son correctos.</li> <li>3. Se responde con el Token generado y Status code 200 [20].</li> </ol>
Escenario alternativo	<ol style="list-style-type: none"> <li>1. Si se produce un fallo en la validación y se responde con un ERR Status code 401 [21] .</li> </ol>

#### CUM0002 Guardar línea/s log para un token válido de agente

<b>CUM0002 Guardar línea/s log para un token válido de agente</b>	
Actor principal	Appender de Log4j
Nivel de objetivo	Usuario
Precondición	Ninguna
Escenario principal de éxito	<ol style="list-style-type: none"> <li>1. Se recibe una petición POST con un token válido y una o varias de log asociadas al token válido.</li> <li>2. Se procesa la información según las reglas y condiciones definidas para el agente.</li> <li>3. Se persiste la información procesada en la base de datos y el log bruto recibido.</li> <li>4. Se responde a la petición POST con un OK Status code 200 [20].</li> </ol>
Escenario alternativo	<ol style="list-style-type: none"> <li>1. Se recibe una petición POST con un token no válido o vacío y se responde con un ERR Status code 401 [21] .</li> <li>2. No se reciben líneas de log y no se genera información procesada ni se guardan líneas de log.</li> <li>3. Las reglas definidas no generan información procesada y sólo se guardan las líneas de log recibidas.</li> </ol>

#### CUM0003 Servicio de healthy

<b>CUM0003 Sevicio de healthy</b>	
Actor principal	Docker
Nivel de objetivo	Usuario



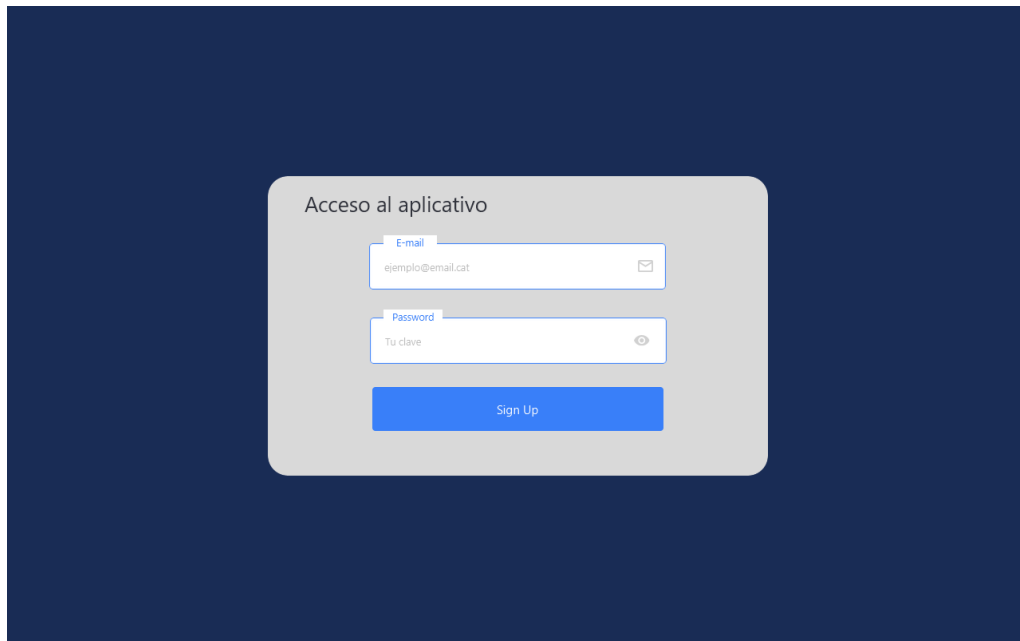
Precondición	Ninguna
Escenario principal de éxito	<ol style="list-style-type: none"> <li>1. Se recibe una petición GET.</li> <li>2. Se responde a la petición con el tiempo en segundos desde su inicio Status code 200 [20].</li> </ol>
Escenario alternativo	<ol style="list-style-type: none"> <li>1. No se responde (timeout) o se responde con ERR Status code 401 [21] .</li> </ol>

#### CUM0004 Servicio de shutdown

<b>CUM0004 Servicio de shutdown</b>	
Actor principal	Docker
Nivel de objetivo	Usuario
Precondición	Ninguna
Escenario principal de éxito	<ol style="list-style-type: none"> <li>1. Se recibe una petición GET.</li> <li>2. Se finaliza el microservicio.</li> </ol>
Escenario alternativo	

#### CUW0001 Acceso de usuario al sistema

<b>CUW0001 Acceso de usuario al sistema</b>	
Actor principal	Administrador y Usuario de soporte
Nivel de objetivo	Usuario
Precondición	Ninguna
Escenario principal de éxito	<ol style="list-style-type: none"> <li>1. La persona introduce su email y una contraseña.</li> <li>2. El sistema verifica el rol del usuario (Administrador o Soporte) y muestra: <ol style="list-style-type: none"> <li>a. Si es administrador: CUW0002</li> <li>b. Si es usuario de soporte: CUW0010</li> </ol> </li> </ol>
Escenario alternativo	<ol style="list-style-type: none"> <li>1. La persona introduce incorrectamente los datos de acceso y el sistema muestra un error de datos de autenticación incorrectos.</li> <li>2. La persona ya está autenticada y no es necesaria la solicitud de su usuario y contraseña mostrando directamente el resumen de los agentes dados de alta.</li> <li>3. Las credenciales introducidas no disponen del nivel del perfil de acceso correcto y muestra un error indicándolo.</li> </ol>



**Ilustración 4: Prototipo CUW0001**

CUW0002 Listado de últimos agentes con notificaciones

<b>CUW0002 Listado de últimos agentes con notificaciones</b>	
Actor principal	Administrador
Nivel de objetivo	Usuario
Precondición	El usuario debe estar autenticado y con perfil de administrador.
Escenario principal de éxito	<ol style="list-style-type: none"> <li>1. El sistema muestra una pantalla principal con un menú para acceder al: <ol style="list-style-type: none"> <li>a. Home (CUW0002).</li> <li>b. Usuarios (CUW0003).</li> <li>c. Sistemas (CUW0005).</li> <li>d. Agentes (CUW0007).</li> </ol> </li> <li>2. El menú "Home" aparece seleccionado.</li> <li>3. Se muestra un listado de los últimos agentes con notificaciones sobre los diferentes sistemas permitiendo acceder a la modificación del sistema / agente directamente.</li> </ol>
Escenario alternativo	<ol style="list-style-type: none"> <li>1. El administrador accede al home, no existen agentes creados y el listado se muestra vacío.</li> </ol>

Home	Usuarios	Sistemas	Agentes	U
Agentes con notificaciones				
Nombre	Sistema	Fecha última notificación	Log recibido	
Agente 1	Sistema 1	01/10/2021 00:10:01	100MB	
Agente 2	Sistema 1	01/10/2021 00:10:01	500MB	
Agente 1	Sistema 2	01/10/2021 00:10:01	400MB	
Agente 2	Sistema 2	01/10/2021 00:10:01	120MB	
Agente 3	Sistema 1	11/10/2021 13:10:01	500MB	
Agente	Sistema 3	NA	NA	

**Ilustración 5:Prototipo CUW0002**

CUW0003 Listado filtrado de usuarios creados

CUW0003 Listado filtrado de usuarios creados	
Actor principal	Administrador
Nivel de objetivo	Usuario
Precondición	El usuario debe estar autenticado y con perfil de administrador.
Escenario principal de éxito	<ol style="list-style-type: none"> <li>El sistema muestra una pantalla principal con un menú para acceder al: <ol style="list-style-type: none"> <li>Home (CUW0002).</li> <li>Usuarios (CUW0003).</li> <li>Sistemas (CUW0005).</li> <li>Agentes (CUW0007).</li> </ol> </li> <li>El menú "Usuarios" aparece seleccionado.</li> <li>Se muestra una pantalla con un filtro y un botón para poder crear un nuevo usuario (CUW004). Los criterios de búsqueda serán: <ol style="list-style-type: none"> <li>Usuarios activos/inactivos.</li> <li>Nombre de usuario.</li> <li>Email de usuario.</li> </ol> </li> <li>El administrador introduce uno o varios criterios de filtro.</li> <li>Se muestra un listado paginado y navegable de usuarios coincidentes con el filtro sobre el que se permite realizar click en cada elemento para editarlo (CUW004).</li> </ol>
Escenario alternativo	<ol style="list-style-type: none"> <li>No existen usuarios según los criterios de filtro y el listado aparece vacío.</li> </ol>

**Ilustración 6:Prototipo CUW0003**

#### CUW0004 Alta y modificación de usuario

CUW0004 Alta y modificación de usuario	
Actor principal	Administrador
Nivel de objetivo	Usuario
Precondición	El usuario debe estar autenticado y con perfil de administrador.
Escenario principal de éxito	<ol style="list-style-type: none"> <li>El sistema muestra una pantalla principal con un menú para acceder al: <ol style="list-style-type: none"> <li>Home (CUW0002).</li> <li>Usuarios (CUW0003).</li> <li>Sistemas (CUW0005).</li> <li>Agentes (CUW0007).</li> </ol> </li> <li>El menú “Agentes” aparece seleccionado.</li> <li>Se muestra un formulario para introducir los datos del nuevo usuario: <ol style="list-style-type: none"> <li>Nombre del usuario.</li> <li>Email del usuario.</li> <li>Clave de usuario.</li> <li>Check de activo.</li> <li>Rol del usuario.</li> </ol> </li> <li>Se pulsa en guardar y se persiste la información.</li> </ol>
Escenario alternativo	<ol style="list-style-type: none"> <li>Los datos introducidos no son válidos y al pulsar guardar se muestra un error indicándolo.</li> </ol>

**Ilustración 7: Prototipo CUW0004**

CUW0005 Listado filtrado sistemas creados

CUW0005 Listado filtrado sistemas creados	
Actor principal	Administrador
Nivel de objetivo	Usuario
Precondición	El usuario debe estar autenticado y con perfil de administrador.
Escenario principal de éxito	<ol style="list-style-type: none"> <li>1. El sistema muestra una pantalla principal con un menú para acceder al: <ol style="list-style-type: none"> <li>a. Home (CUW0002).</li> <li>b. Usuarios (CUW0003).</li> <li>c. Sistemas (CUW0005).</li> <li>d. Agentes (CUW0007).</li> </ol> </li> <li>2. El menú "Sistemas" aparece seleccionado.</li> <li>3. Se muestra una pantalla con un filtro y un botón para poder crear un nuevo sistema (CUW0006) Los criterios de búsqueda serán: <ol style="list-style-type: none"> <li>a. Nombre del sistema.</li> <li>b. País del sistema.</li> <li>c. Activos inactivos.</li> </ol> </li> <li>4. El administrador introduce uno o varios criterios de filtro.</li> <li>5. Se muestra un listado paginado y navegable de sistemas coincidentes con el filtro sobre el cual es posible realizar clic en un elemento y acceder a su modificación (CUW0006).</li> </ol>
Escenario alternativo	<ol style="list-style-type: none"> <li>1. No existen agentes coincidentes con los criterios del filtro y el listado aparece vacío.</li> </ol>

**Ilustración 8: Prototipado CUW0004**

#### CUW0006 Alta y modificación sistema

CUW0006 Alta y modificación sistema	
Actor principal	Administrador
Nivel de objetivo	Usuario
Precondición	El usuario debe estar autenticado y con perfil de administrador.
Escenario principal de éxito	<ol style="list-style-type: none"> <li>El sistema muestra una pantalla principal con un menú para acceder al: <ol style="list-style-type: none"> <li>Home (CUW0002).</li> <li>Usuarios (CUW0003).</li> <li>Sistemas (CUW0005).</li> <li>Agentes (CUW0007).</li> </ol> </li> <li>El menú “Sistemas” aparece seleccionado.</li> <li>Se muestra un formulario con los campos necesarios para dar de alta un sistema: <ol style="list-style-type: none"> <li>Nombre del sistema.</li> <li>Fecha de alta del sistema.</li> <li>País de ubicación del sistema.</li> <li>Check de activo/inactivo.</li> </ol> </li> <li>Se persiste la información del nuevo agente en la base de datos.</li> </ol>
Escenario alternativo	<ol style="list-style-type: none"> <li>Los datos introducidos no son válidos y al pulsar guardar se muestra un error indicándolo.</li> </ol>

**Ilustración 9: Prototipo CUW0006**

CUW0007 Listado filtrado de agentes creados

CUW0007 Listado agentes creados	
Actor principal	Administrador
Nivel de objetivo	Usuario
Precondición	El usuario debe estar autenticado y con perfil de administrador.
Escenario principal de éxito	<ol style="list-style-type: none"> <li>1. El sistema muestra una pantalla principal con un menú para acceder al: <ol style="list-style-type: none"> <li>a. Home (CUW0002).</li> <li>b. Usuarios (CUW0003).</li> <li>c. Sistemas (CUW0005).</li> <li>d. Agentes (CUW0007).</li> </ol> </li> <li>2. El menú “Agentes” aparece seleccionado.</li> <li>3. Se muestra una pantalla con un filtro y un botón para poder crear un nuevo agente (CUW0008) Los criterios de búsqueda serán: <ol style="list-style-type: none"> <li>a. Agentes activos/inactivos</li> <li>b. Nombre del agente.</li> <li>c. Nombre del sistema.</li> </ol> </li> <li>4. El administrador introduce uno o varios criterios de filtro.</li> <li>5. Se muestra un listado paginado y navegable de agentes coincidentes con el filtro sobre el cual es posible realizar clic en un elemento y acceder a su modificación (CUW0008).</li> </ol>
Escenario alternativo	<ol style="list-style-type: none"> <li>1. No existen agentes coincidentes con los criterios del filtro y el listado aparece vacío.</li> </ol>

**Filtro de agentes:**

Nombre:  Sistema:

Activo: ☐ Si ☐ No ☐ Todos

Nombre	Sistema	Notificación	Alertas	Activo
Agente 1	Sistema 1	12/10/2021 14:13	10	Si
Agente 2	Sistema 1	12/10/2021 13:30	5	No

1 2 3 ... 5 ... 10 11 12

**Ilustración 10: Prototipo CUW0007**

CUW0008 Alta y modificación de agente

CUW0008 Alta y modificación de agente	
Actor principal	Administrador
Nivel de objetivo	Usuario
Precondición	El usuario debe estar autenticado y con perfil de administrador.
Escenario principal de éxito	<ol style="list-style-type: none"> <li>El sistema muestra una pantalla principal con un menú para acceder al: <ol style="list-style-type: none"> <li>Home (CUW0002).</li> <li>Usuarios (CUW0003).</li> <li>Sistemas (CUW0005).</li> <li>Agentes (CUW0007).</li> </ol> </li> <li>El menú “Agentes” aparece seleccionado.</li> </ol>



	<ol style="list-style-type: none"> <li>Se muestra un formulario con los campos necesarios para dar de alta un agente: <ol style="list-style-type: none"> <li>Nombre del agente.</li> <li>Fecha de alta del agente.</li> <li>Sistema del agente.</li> <li>Checkbox de activo/inactivo.</li> </ol> </li> <li>Se muestra apartado para gestionar las reglas del agente: <ol style="list-style-type: none"> <li>Se muestran botón para agregar nuevas reglas al agente (CUW0009)</li> <li>Se muestran las reglas existentes en el agente permitiendo la modificación de cada regla (CUW0009).</li> </ol> </li> <li>Se muestra un botón genera un token identificativo aleatorio.</li> <li>Se persiste la información del nuevo agente en la base de datos.</li> </ol>
Escenario alternativo	<ol style="list-style-type: none"> <li>Los datos introducidos no son válidos y al pulsar guardar se muestra un error indicándolo.</li> </ol>

**Ilustración 11: Prototipo CUW0008**

#### CUW0009 Alta y modificación de regla de agente

CUW0009 Alta y modificación de regla de agente	
Actor principal	Administrador
Nivel de objetivo	Usuario
Precondición	El usuario debe estar autenticado y con perfil de administrador.

Escenario principal de éxito	<ol style="list-style-type: none"> <li>Se muestra una ventana modal con el formulario para la creación de una nueva regla: <ol style="list-style-type: none"> <li>Nombre de la regla</li> <li>Combo de activa/inactiva.</li> <li>Expresión regular para extracción de valor de la línea de log.</li> <li>Se muestra un combo para seleccionar si deben coincidir todas o al alguna de las condiciones.</li> <li>Combo para selección de tipo de cálculo: Contar o usar valor directo de la expresión regular.</li> <li>Combo de selección de nivel de severidad.</li> <li>Botón para agregar nueva condición a la regla con sus <ol style="list-style-type: none"> <li>Tipo de comparación.</li> <li>Valor para la comparación.</li> <li>Tiempo para el análisis.</li> </ol> </li> <li>Botón para eliminar condición de la regla.</li> </ol> </li> <li>Se muestra un botón “Añadir” que permite agregar la regla al agente.</li> </ol>
Escenario alternativo	<ol style="list-style-type: none"> <li>Los datos introducidos de la regla no son válidos y al pulsar añadir se muestra un error indicándolo.</li> </ol>

The screenshot shows a web application interface for agent management. The main header includes navigation links: Home, Usuarios, Sistemas, and Agentes. The current view is 'Modificación de agente'. A modal window titled 'Modificación de regla' is open, displaying a form for editing a rule. The form includes fields for 'Nombre' (Regla 1), 'Regexp' (.)(2), 'Tipo cálculo' (Contar), 'Gravedad' (Media), 'Comparación' (Todas), and 'Activo' (radio buttons for Si/No). Below these is a 'Condiciones' table with columns for 'Tipo de comparación', 'Tiempo en segundos', and 'Valor'. The table has two rows: 'Menor que media' with values 300 and 5, and 'Mayor que media' with values 300 and 10. There are 'Nueva condición', 'Guardar', and 'Eliminar' buttons.

**Ilustración 12 Prototipo CUW0009**

CUW0010 Monitorización de agentes

CUW0010 Monitorización de agentes	
Actor principal	Usuario soporte

Nivel de objetivo	Usuario
Precondición	El usuario debe estar autenticado y con perfil de soporte.
Escenario principal de éxito	<ol style="list-style-type: none"> <li>1. Se muestra una pantalla principal en la que se pueden seleccionar los sistemas a monitorizar y el tiempo de monitorización.</li> <li>2. Se muestra un resumen de las alarmas críticas/altas y leves recibidas. <ol style="list-style-type: none"> <li>a. Al pulsar en uno de los resúmenes se muestra debajo las alarmas filtradas, por defecto se muestran las críticas.</li> <li>b. Al pulsar una de las alarmas, se mostrará un detalle (CUW0010)</li> </ol> </li> <li>3. Se muestra un listado de las alarmas recibidas en tiempo real. <ol style="list-style-type: none"> <li>a. Al pulsar una de las alarmas, se mostrará un detalle (CUW0010)</li> </ol> </li> </ol>
Escenario alternativo	



**Ilustración 13: Prototipo CUW0010**

#### CUW0011 Detalle de regla de agente

CUW0011 Detalle de regla de agente	
Actor principal	Usuario soporte
Nivel de objetivo	Usuario
Precondición	El usuario debe estar autenticado y con perfil de soporte.

Escenario principal de éxito	<ol style="list-style-type: none"> <li>1. Se muestra el detalle de una alarma en concreto detallando el agente y el sistema al que pertenece.</li> <li>2. Se muestra una gráfica con los sucesos producidos por la alarma.</li> <li>3. Se muestra el combo de alarmas en tiempo real.</li> </ol>
Escenario alternativo	

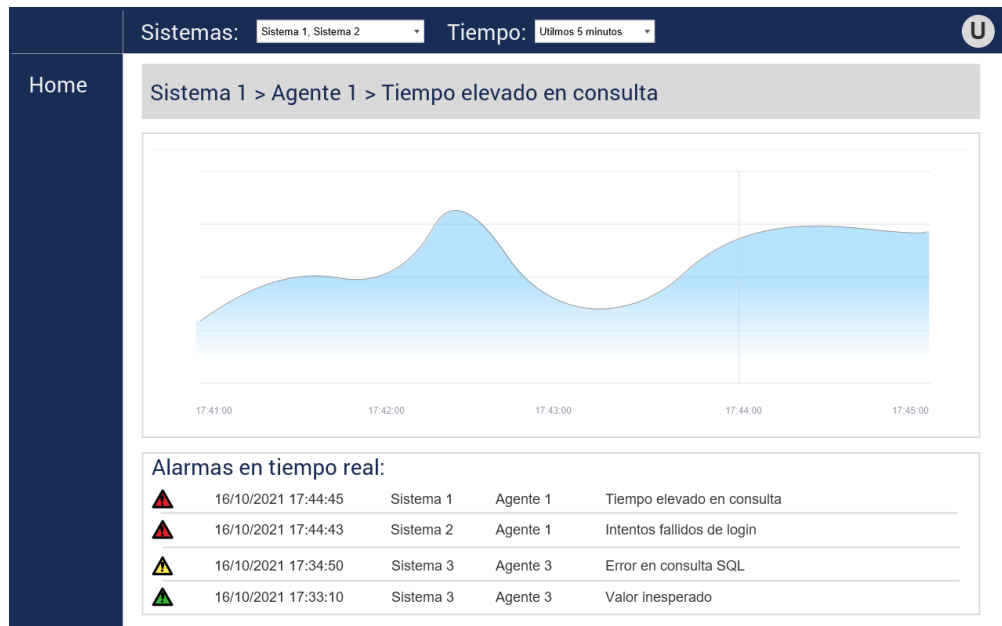


Ilustración 14: Prototipo CUW0011

## 2.2 Punto de vista de la información

Se definen las diferentes estructuras de datos que den soporte a los casos de uso definidos en el anterior punto:

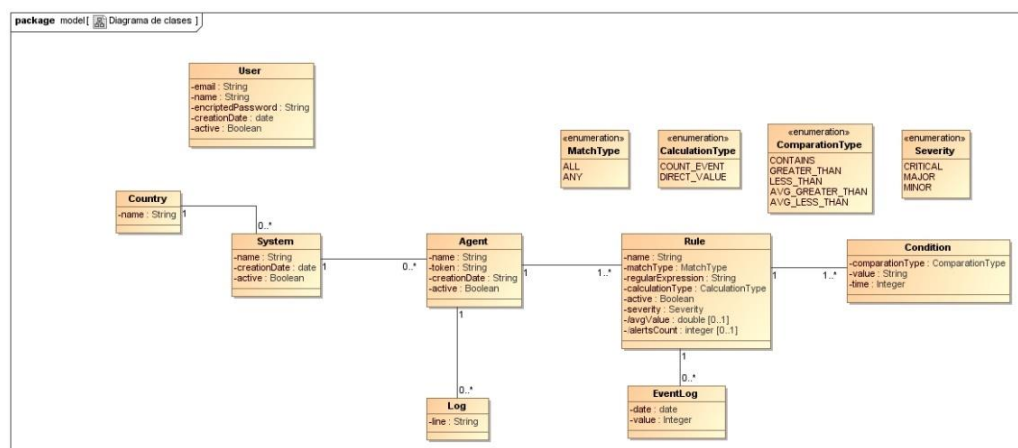
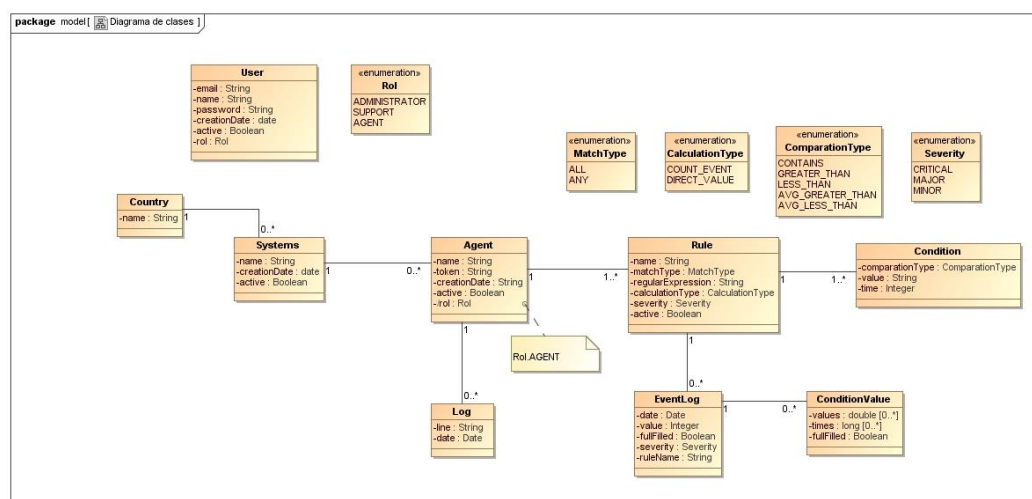


Ilustración 15: Sistema de información V1

A lo largo de la fase de implementación, en las diferentes iteraciones han surgido necesidades que no estaban previstas.

- Necesidad de especificar el Rol que tiene el usuario para diferenciar entre agentes, usuarios administrador y usuarios.
- Se agrega la fecha en la que se crea el log para poder recalculer los eventos.
- Se crea la entidad ConditionValue para almacenar que condiciones y con qué valores se cumplen respecto a una regla de un agente.
- Se modifica la entidad EventLog para saber cuál es la regla, la severidad y si ha sido cumplida totalmente para generar el evento.



**Ilustración 16: Sistema de información V2**

### 2.3 Punto de vista de computacional y de ingeniería

Se opta por una arquitectura basada en microservicios [22] dado que ofrece un sistema escalable horizontalmente [23] para que pueda soportar el aumento/reducción de los sistemas a monitorizar, que no afecte al rendimiento de los subsistemas, que no suponga un coste elevado de ampliación y que se pueda realizar de manera automatizada. Si la arquitectura fuese monolítica, un aumento de envío de logs por parte de los agentes o una ampliación del n.º de sistemas a monitorizar afectaría al resto de funcionalidades (consulta y administración). El coste del escalado sería superior al tener que realizar un mayor aumento de recursos en el sistema para dar soporte a toda la aplicación y, por último, una actualización de recursos de este tipo se tendría que hacer con los sistemas parados.

#### Capas del sistema

El sistema tendrá dos capas principales: capa de microservicios y capa de presentación. Además, existe un componente “cliente” que se conecta

directamente a la capa de microservicios. A continuación, se definirán las dos capas principales:

#### *Capa de microservicios*

Esta capa contendrá todos los microservicios que expondrán una API Rest que será consumida por la capa de presentación. Cada microservicio será diseñado siguiendo una arquitectura de dos capas:

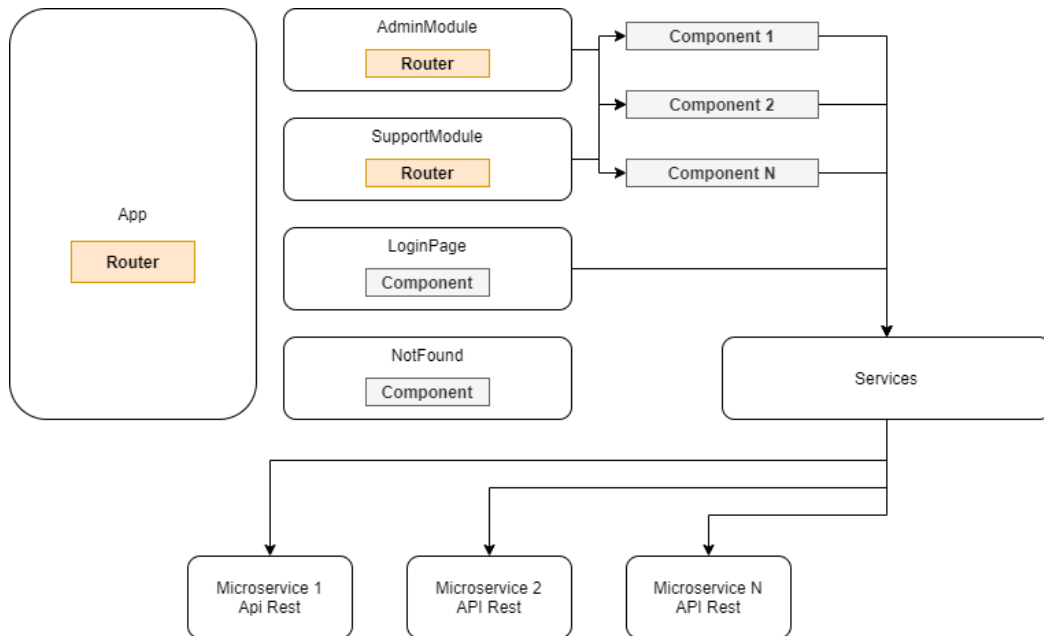
- Capa de modelo: Representa el dominio de la aplicación y será común a todos los microservicios. El rol principal de esta capa será el manejo del flujo de datos desde la base de datos.
- Capa controladora: Gestiona el flujo de información entre el modelo y la capa de presentación del sistema a través de un API Rest.

Para el acceso a datos, los microservicios usarán Spring Data [24] como implementación en la capa de modelo. En la capa controladora se usará el *framework* Spring Boot [5] que facilitará en gran medida el desarrollo y su posterior despliegue en contenedores Docker [3].

#### *Capa de presentación*

La capa de presentación será la encargada de ofrecer una interfaz web al usuario. Esta interfaz de usuario seguirá una arquitectura MVC desarrollada con el *framework* React [7]. Para poder encapsular esta capa en un contenedor Docker, se usará Spring Boot [5] como servidor web y un plugin de Maven Yarn [24] para generar los ficheros JavaScript a partir del código fuente de React [7]:

- Capa de modelo: Representa el dominio de la aplicación, a diferencia de la capa de modelo del microservicio no tendrá acceso a la base de datos, pero si realizará peticiones a la capa controladora de cada microservicio en concreto para obtener los datos.
- Capa controladora: Manejará la lógica de las interacciones que realiza el usuario/a en la capa de vista.
- Capa vista: Esta capa se enfocará en la representación y visualización de los datos obtenidos a través de las capas anteriores.



**Ilustración 17: Esquema de aplicativo React**

En la ilustración se puede ver el esquema de componentes. Se parte de un componente principal llamado “App” que tiene comportamiento Router, este comportamiento indica que es capaz de controlar las rutas de la aplicación. Dependiendo de la ruta especificada, este componente llama a otro componente encargado de gestionar esa ruta específica:

- La ruta /admin/\* la gestiona el componente AdminModule
- La ruta /support/\* la gestiona el componente SupportModule
- La ruta / la gestiona el componente LoginPage
- Para cualquier otra ruta desconocida se gestiona con el componente NotFound.

A su vez, cada componente anterior puede incluir componentes más pequeños para realizar una determinada acción, por ejemplo, un campo de texto con un botón para generar un token.

Cuando un componente requiere la consulta de una determinada API Rest de un microservicio, delega esta lógica en un servicio que es el encargado de establecer la comunicación con el microservicio y, por ejemplo, enviar el token de autenticación para que el microservicio pueda validar la petición.

#### *Componente de log4j-appender*

Es necesario la inclusión de un componente externo que permita generar log contra la capa de microservicios. El componente será un *appender* de log4j que podrá incluirse en un fichero log4j.xml y permitirá enviar las líneas de log al servidor remoto tal y como se ha visto en la Ilustración “Esquema aplicación”.

## Diagrama de casos de uso

Se realiza un pequeño diagrama de los casos de uso con sus 3 actores: Administrador, Agente y Usuario soporte. Se ha establecido una relación entre los casos de uso definidos en el primer apartado respecto este diagrama usando los identificadores, de tal forma que facilite su lectura y comprensión:

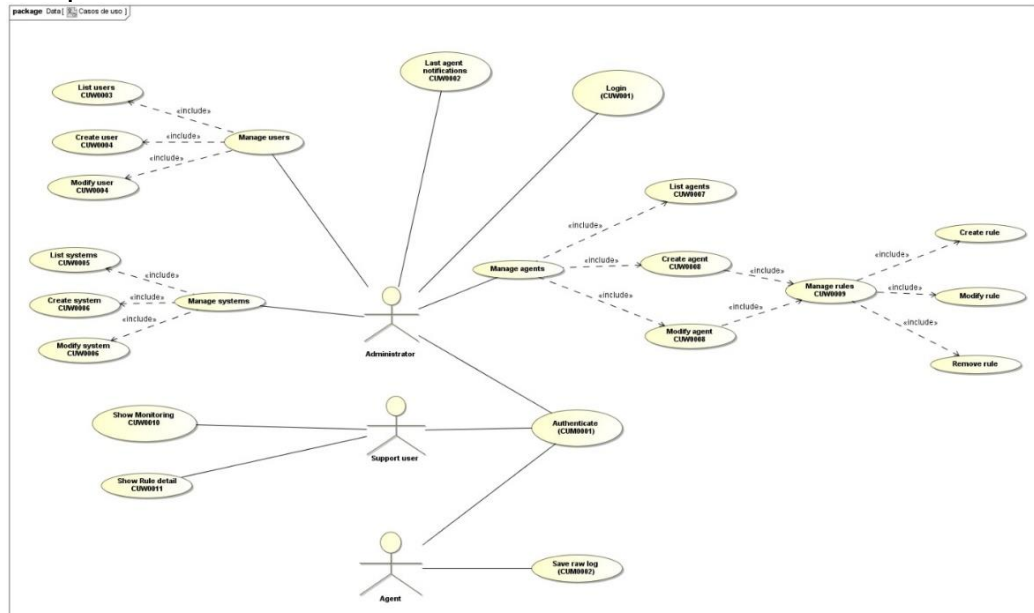


Ilustración 18: Diagrama de casos de uso

Una vez iniciada la fase de desarrollo, y más concretamente en la fase final de pruebas de despliegue se ve la necesidad de implementar dos casos de uso nuevos para tener información sobre el estado de cada uno de los microservicios. Es por ello por lo que se establece un nuevo actor llamado Docker que maneja dos casos de uso:

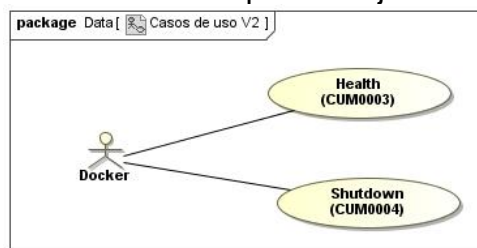
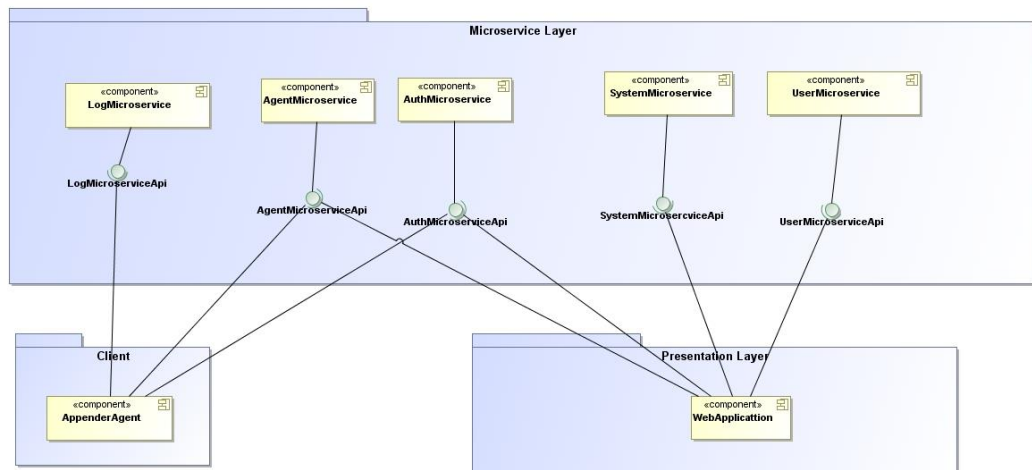


Ilustración 19: Ampliación de los casos de uso.

## Diagrama de componentes

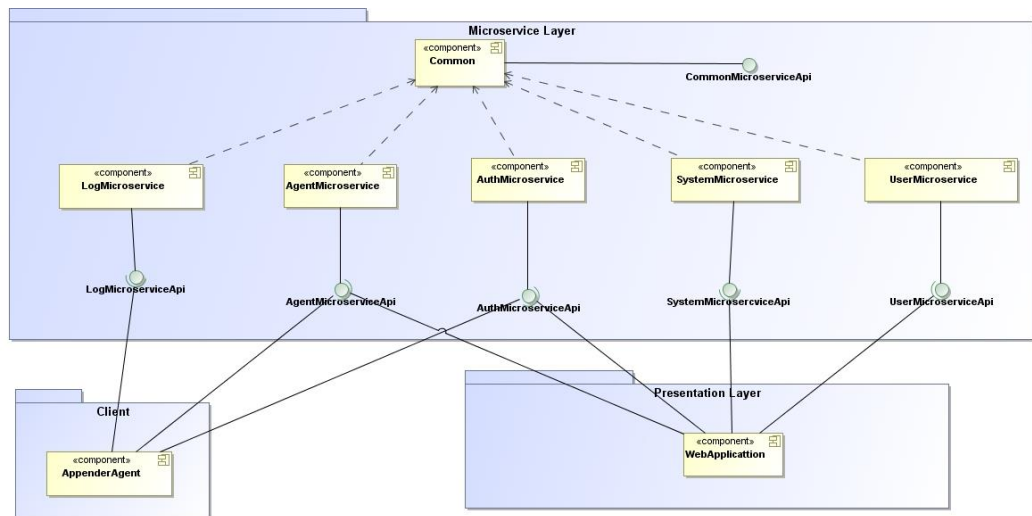
Se especifican los diferentes componentes de las dos capas principales: la capa de microservicios y la capa de presentación. Además, se sitúa el componente "Cliente" que es el software que lee el log en los aplicativos y envía el log a la capa de microservicios.





**Ilustración 20: Diagrama de componentes**

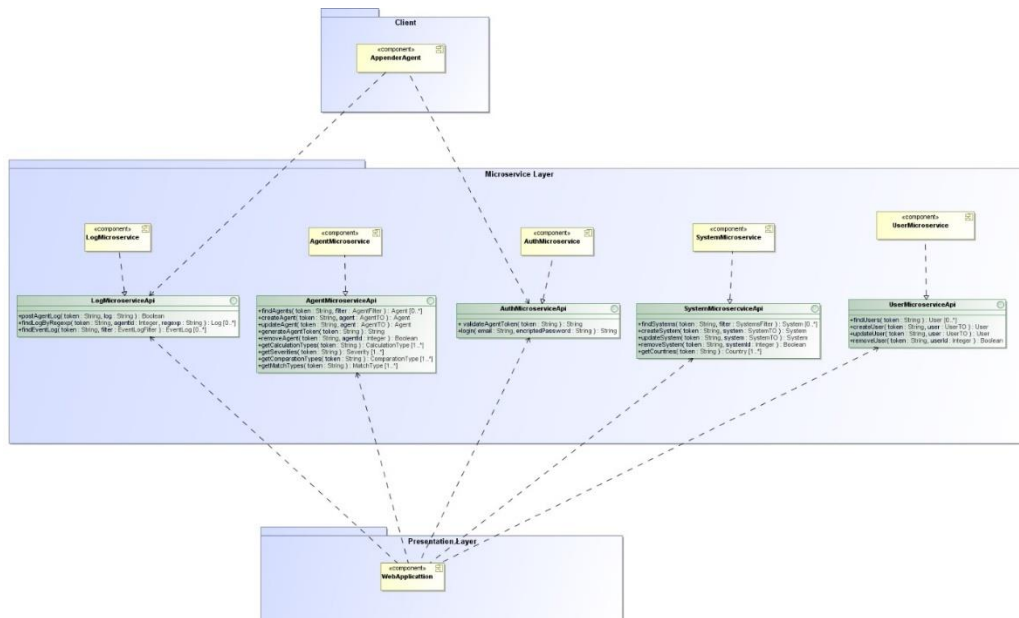
A medida que se desarrollaba la fase de implementación se ha visto que faltaba un componente *common* el cual debería ser usado por todos los microservicios:



**Ilustración 21: Diagrama de componentes V2**

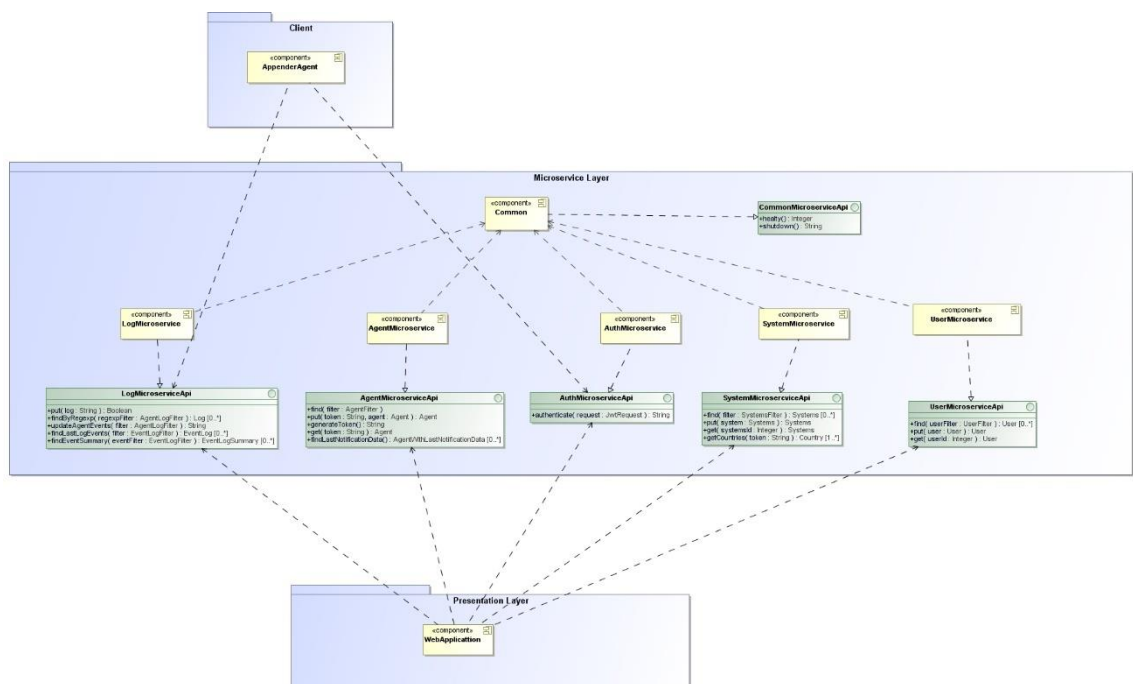
#### Diagrama de componentes detallado

Se definen los diferentes métodos que van a ser llamados desde la capa de presentación y que serán consumidos por el aplicativo web. Básicamente es la definición de la API Rest que será publicada:



**Ilustración 22: Diagrama de componentes refinado**

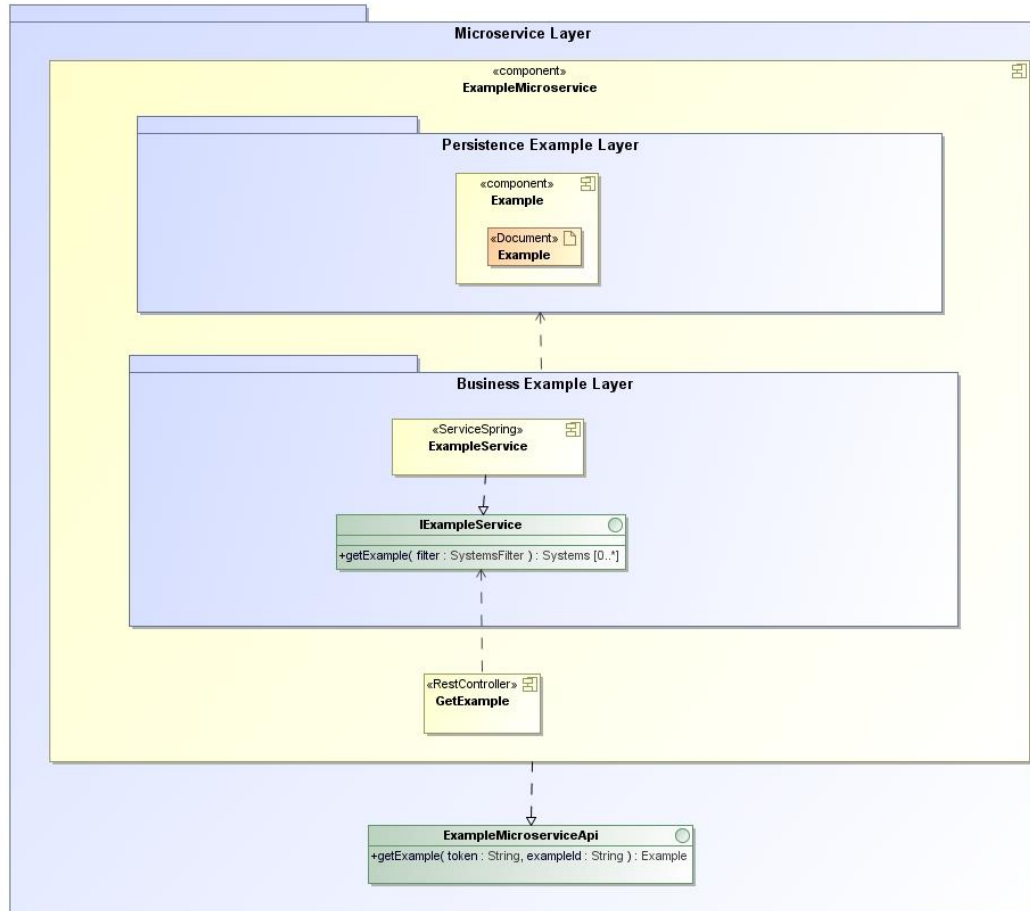
En las diferentes iteraciones de la implementación han surgido cambios sobre algunos de los componentes definidos previamente. Los principales cambios son sobre los métodos finales de la API Rest, se han resumido la mayoría en “put, get y find” y retirado las entidades TO que finalmente no han sido necesarias. También se han retirado la identificación del token dado que es algo que todo microservicio recibe a través de la capa controladora gracias a Spring Security y JWT.



**Ilustración 23: Diagrama de componentes refinado V2**

Diagrama de componentes de microservicio detallado

Cada microservicio tendrá 2 capas: Capa modelo que incluirá la capa de negocio y de persistencia y la capa controladora que será proporcionada por el *framework* Spring Boot [5].

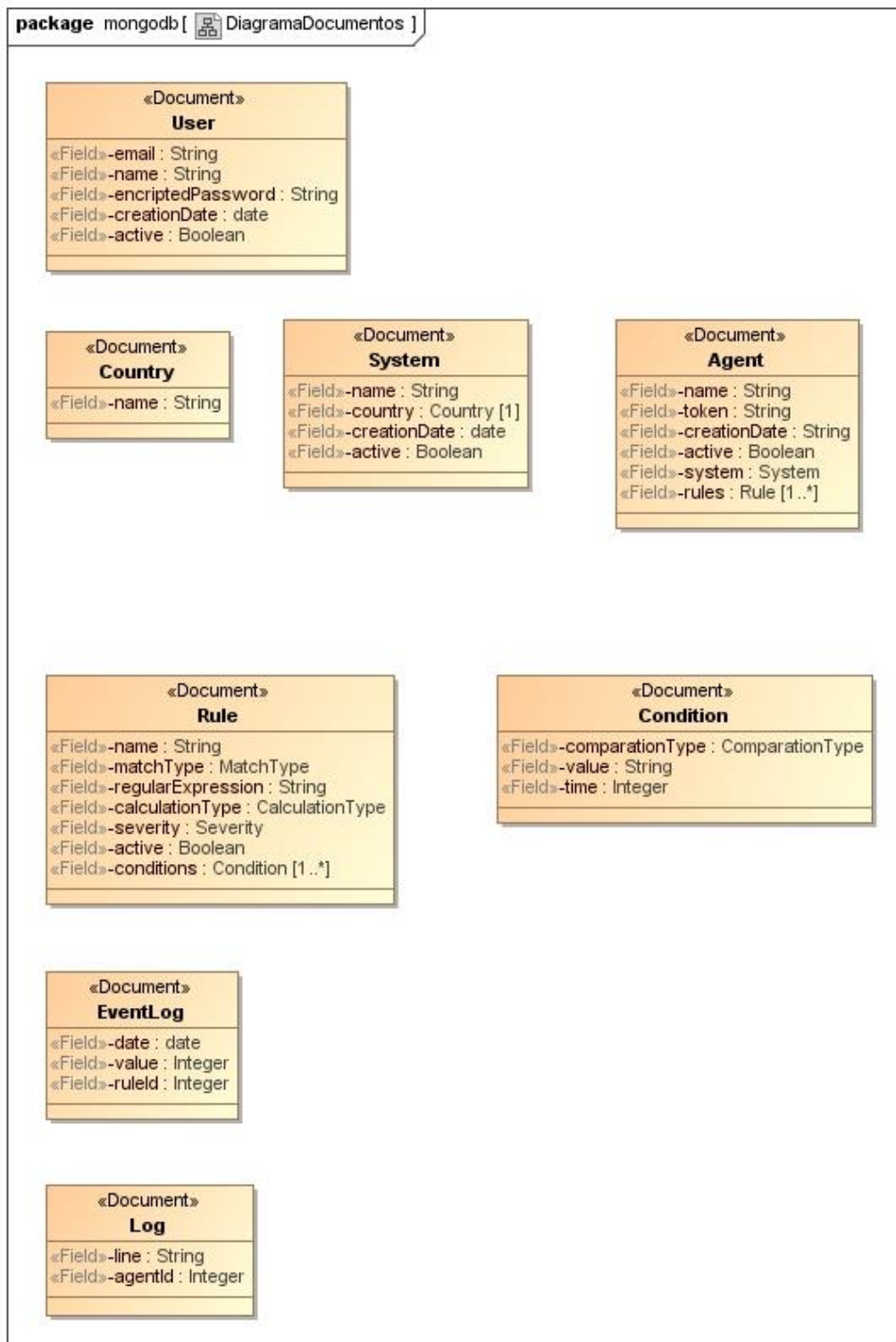


**Ilustración 24: Componente de microservicio detallado**

## Base de datos

Para la base de datos se usará MongoDB que es una base de datos NoSQL [25]. A diferencia de las bases de datos relacionales, una base de datos NoSQL no es necesario definir un esquema, en cambio maneja una estructura de información básica llamada documento [26]. Estos documentos se agrupan en colecciones [27] y no necesariamente los documentos de una colección deben tener el mismo conjunto de campos.

En este proyecto se creará a partir del sistema de información y usando “Spring Data MongoDB” [28] con una estructura de documentos definida de la siguiente manera:



**Ilustración 25: Diagrama de documentos MongoDB V1**

A medida que se han ido completando las iteraciones de la fase de desarrollo se han realizado diferentes cambios respecto al diseño inicial. Principalmente ha sido la eliminación de los documentos: “Condition” y “Rule” porque forman parte del documento “Agent”. También se han agregado campos ID a Log, User, EventLog, Country, Systems.

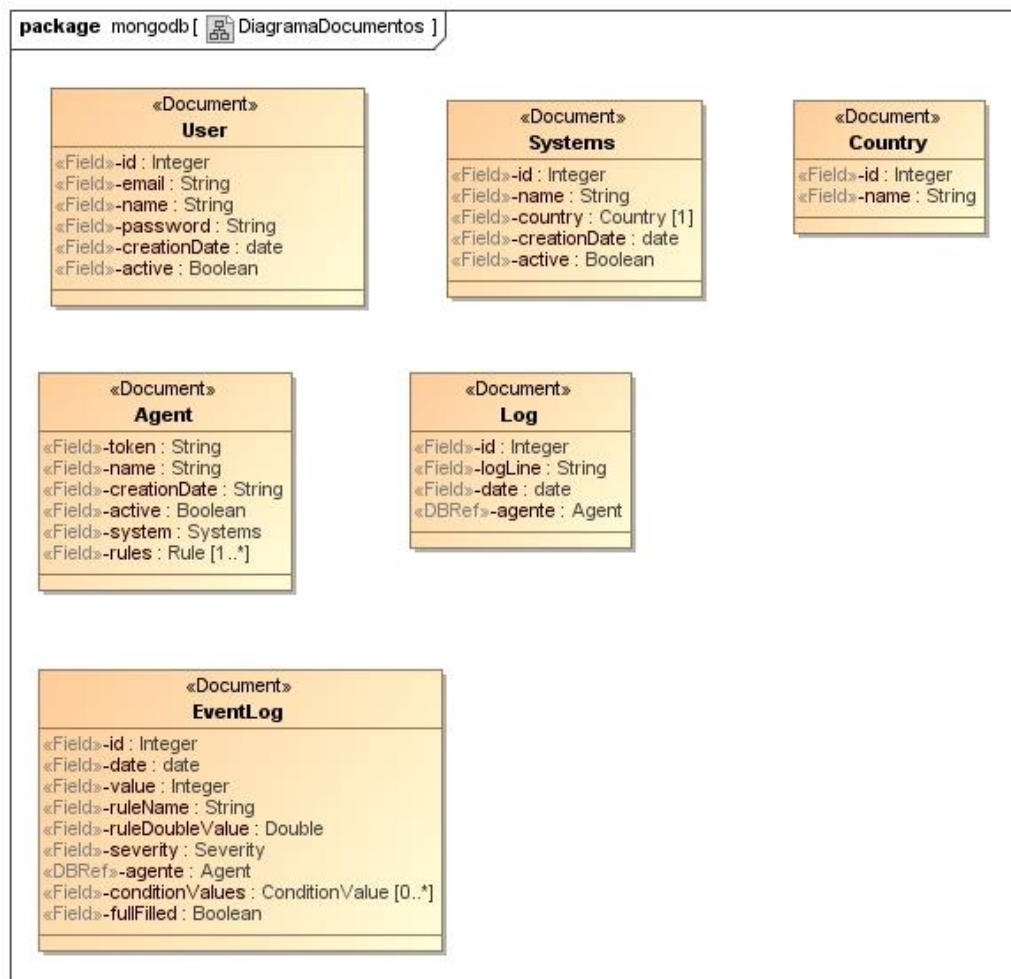


Ilustración 26:Diagrama de documentos MongoDB V2

## 2.4 Punto de vista tecnológico

Para el desarrollo de la arquitectura de microservicios y sus diferentes capas se usará un conjunto de módulos siguiendo arquetipos Maven existentes como base y que tienen como padre un proyecto Maven que los incluye a todos:

Tipos de módulos Maven a usar

- *common*: Módulo Maven con arquetipo Spring Data para la definición de las clases del sistema de información y los documentos. Este proyecto será incluido como dependencia por cada microservicio.
- *microservice-project*: Módulo Maven con arquetipo Spring Boot [5] que incluye como dependencia el *common*. Cada microservicio será un módulo independiente que tienen como padre el proyecto Maven principal.

- *webapp*: Módulo Maven con arquetipo Spring Boot [5] y React [7] para poder construir la capa de presentación y que llamará a cada uno de los servicios.
- *log4j-appender*: Módulo Maven para la generación de librería *appender* de Log4j [10].

## Entorno de desarrollo

Para el desarrollo del proyecto se usarán diferentes herramientas:

- Eclipse como IDE de desarrollo que permita la creación de código, gestión de dependencias y plugins de Maven.
- Maven como gestor de proyectos Java que permita el uso de arquetipos y Plugins como docker-maven-plugin.
- GitHub como repositorio de código para los diferentes proyectos.
- Docker Desktop para la prueba de despliegue de microservicios y base de datos usando contenedores.

## 2.5 Detalles importantes en fase de desarrollo

A continuación, se destacan las cuestiones más importantes de la fase de desarrollo.

### Uso de GitHub y GitHub Flow

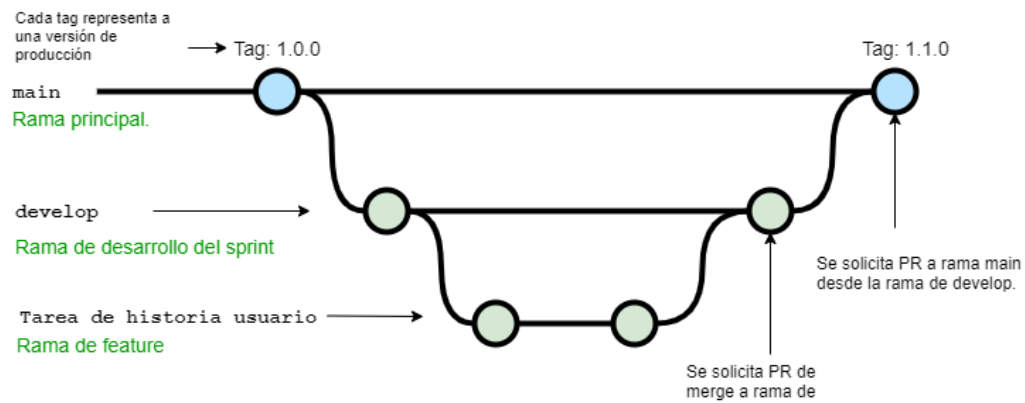
Por una parte, el uso del repositorio de código ubicado en la siguiente URL <https://github.com/aperezgua/tfg-monitor-system> y la metodología de GitHub Flow para gestionarlo. Si bien lo mejor es realizar técnicas como *rebase* en la historia del repositorio en vez de realizar *merges* cuando sólo se cuenta con un desarrollador, se ha querido seguir un esquema como si se perteneciese a un equipo con más personas.

```
* 836720a - (origin/feature/support-webapp, feature/support-webapp) Cambio de valor en datos de test (2021-12-07 09:27:54) <aperezgua>
* c021172 - Finalización de gráfica de eventos. (2021-12-05 21:19:42) <aperezgua>
* 436fc35 - Creando gráfica de regla. (2021-12-05 13:56:51) <aperezgua>
* a3fd7b9 - Diseño de página fullscreen (2021-12-05 13:13:29) <aperezgua>
* 356821f - Sistema de cards en pantalla principal (2021-12-05 11:51:02) <aperezgua>
* aa3c18c - Se implementa seguridad por roles en los controller. (2021-12-05 10:27:20) <aperezgua>
* 9947c19 - Se corrige filtrado de línea de log y parametrización de las reglas (2021-12-03 19:12:16) <aperezgua>
* 8cd3312 - Filtrado de eventos por sistemas y ultimos N segundos. (2021-12-03 18:27:26) <aperezgua>
* 3fd1cc0 - Selectores de cabecera de sistemas y tiempo. (2021-12-03 17:40:14) <aperezgua>
* 612c0ac - Select multiple en home de soporte (2021-12-02 22:21:47) <aperezgua>
* 3e254bd - Corrección de eliminación de regla. (2021-12-02 21:44:12) <aperezgua>
* 57b91ec - Consola de eventos en tiempo real (2021-12-02 21:25:42) <aperezgua>
* 3f53663 - Se hace test para probar el regex (2021-12-02 04:50:00) <aperezgua>
* 1f76510 - Eventos generación de eventos verificada. (2021-12-02 03:59:24) <aperezgua>
* 07e4194 - Finalizando servicio de actualización de log cuando se guarda agente. (2021-12-01 11:59:35) <aperezgua>
* 58798e3 - Sistema de actualización de eventos de agente. (2021-12-01 11:41:14) <aperezgua>
* 337b74b - Finalizando procesamiento de eventos de log (2021-11-30 22:35:00) <aperezgua>
* 6f53feb - Comentarios y sistema de procesamiento de eventos de log (2021-11-30 22:25:20) <aperezgua>
* a9311de - Módulo de soporte con Home (2021-11-30 14:12:13) <aperezgua>
/
* b8971ae - Merge pull request #14 from aperezgua/feature/formularios-list (2021-11-30 13:46:27) <aperezgua>
/
* de2b6ed - (origin/feature/formularios-list, feature/formularios-list) Update RuleRegex.jsx (2021-11-30 13:46:06) <aperezgua>
* d5a6ede - Formateo de fechas y error en datos de reglas. (2021-11-30 13:42:11) <aperezgua>
* 792ef7c - Se finaliza modificación de últimos listados (2021-11-30 13:33:00) <aperezgua>
* 7cb2b34 - Se cambia diseño en pantallas de listado (2021-11-30 13:26:19) <aperezgua>
/
* 3ee902c - (tag: 1.1) Merge pull request #13 from aperezgua/develop/sprint-2 (2021-11-29 21:57:00) <aperezgua>
/
* 633ff9b - (origin/develop/sprint-2, develop/sprint-2) Merge pull request #12 from aperezgua/feature/agent-authentication (2021-11-29 21:56:12)
/
* acf8b42 - (origin/feature/agent-authentication, feature/agent-authentication) Sistema de visualización de logs coincidentes con REGEXP en adm
* 5fd0599 - Listado de agentes con últimas notificaciones (2021-11-27 07:47:17) <aperezgua>
* 227cd54 - Sistema de AgentAppender creado con logs aleatorios. (2021-11-27 07:01:15) <aperezgua>
```

Ilustración 27: Log de GitHub.

En este punto se ha de destacar el uso de las nomenclaturas *feature* y *develop* para gestionar las nuevas funcionalidades y la versión en *snapshot* correspondiente. También el uso de las PR (Pull Request) para llevar los cambios de las ramas de *feature* a *develop* y de *develop* a *main*. Esto es importante cuando se trabaja en equipo y con metodologías ágiles de tal forma que todos los cambios realizados queden enlazados de alguna manera a las historias de usuario que se hayan definido. Además, el uso de PR permite que otras personas validen el código que hayas subido con el fin de evitar posibles errores, bugs o tener una retroalimentación cuando trabajas en equipos con diferentes niveles de experiencia.

#### Ejemplo de diagrama de GitFlow:



#### Ilustración 28: Diagrama de GitFlow

Otra cuestión importante en el uso de GitHub es la posibilidad de usar *actions* y validaciones automáticas a la hora de solicitar la PR como serían: cobertura de código, ejecución de plugins de *checkstyle* o *spotbugs* que hagan que el código que se vaya a llevar a la rama destino contenga el menor n.º de errores posible.

#### Uso de Maven

Esta es la herramienta más importante sobre la que se ha desarrollado todo el proyecto. Gracias a ella, se ha podido establecer una jerarquía modular y disponer de un único punto para definir las diferentes versiones usadas de las librerías (spring-boot, log4j, etc.) y plugins (fabric8, yarn, etc.).

Con esta jerarquía se dispone un pom.xml que va a incluir los módulos del proyecto (microservicios, common, appender):



```

tfg-monitor-system/pom.xml x
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
5   ...
6   <modelVersion>4.0.0</modelVersion>
7   ...
8   <parent>
9     <groupId>org.springframework.boot</groupId>
10    <artifactId>spring-boot-starter-parent</artifactId>
11    <version>2.5.5</version>
12    <relativePath/>
13  </parent>
14  ...
15  <groupId>edu.uoc.tfgmonitorsystem</groupId>
16  <artifactId>tfg-monitor-system</artifactId>
17  <version>1.2</version>
18  <packaging>pom</packaging>
19  <name>tfg-monitor-system</name>
20  ...
21  <properties>
22    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
23  </properties>
24  <build>
25    <plugins>
26      <!-- Plugins versions -->
27      <plugin>
28        <groupId>org.apache.maven.plugins</groupId>
29        <artifactId>maven-clean-plugin</artifactId>
30        <version>3.1.0</version>
31      </plugin>
32      <plugin>
33        <groupId>org.apache.maven.plugins</groupId>
34        <artifactId>maven-resources-plugin</artifactId>
35        <version>3.0.2</version>
36      </plugin>
37      <plugin>
38        <groupId>org.apache.maven.plugins</groupId>
39        <artifactId>maven-compiler-plugin</artifactId>
40        <version>3.8.0</version>
41      </plugin>
42      <plugin>
43        <groupId>org.apache.maven.plugins</groupId>
44        <artifactId>maven-surefire-plugin</artifactId>
45        <version>2.22.1</version>
46      </plugin>
47      <plugin>
48        <groupId>org.apache.maven.plugins</groupId>
49        <artifactId>maven-jar-plugin</artifactId>
50        <version>3.0.2</version>
51      </plugin>
52      <plugin>
53        <groupId>org.apache.maven.plugins</groupId>
54        <artifactId>maven-install-plugin</artifactId>
55        <version>2.5.2</version>
56      </plugin>
57      <plugin>
58        <groupId>org.apache.maven.plugins</groupId>
59        <artifactId>maven-deploy-plugin</artifactId>
60        <version>2.8.2</version>
61      </plugin>
62      <plugin>
63        <groupId>org.apache.maven.plugins</groupId>
64        <artifactId>maven-site-plugin</artifactId>
65        <version>3.7.1</version>
66      </plugin>
67      <plugin>
68        <groupId>org.apache.maven.plugins</groupId>
69        <artifactId>maven-project-info-reports-plugin</artifactId>
70        <version>3.0.0</version>
71      </plugin>
72    </plugins>
73  </build>
74 </project>

```

**Ilustración 29: Pom.xml parent**

Cada módulo por separado va a hacer referencia al pom.xml padre para que simplemente tenga declarar las dependencias y plugins a usar en su construcción:

```

tfg-monitor-system/pom.xml common/pom.xml x
1 <?xml version="1.0"?>
2 <project>
3   <modelVersion>4.0.0</modelVersion>
4   <parent>
5     <groupId>edu.uoc.tfgmonitorsystem</groupId>
6     <artifactId>tfg-monitor-system</artifactId>
7     <version>1.2</version>
8     <relativePath>../pom.xml</relativePath>
9   </parent>
10  <artifactId>common</artifactId>
11  <name>common</name>
12  <packaging>jar</packaging>
13  <dependencies>
14    <dependency>
15      <groupId>org.springframework.data</groupId>
16      <artifactId>spring-data-mongodb</artifactId>
17    </dependency>
18    <dependency>
19      <groupId>org.mongodb</groupId>
20      <artifactId>mongodb-driver</artifactId>
21    </dependency>
22  </dependencies>
23 </project>

```

**Ilustración 30: Pom.xml de módulo con referencia al pom.xml padre**

Por un lado, las librerías del aplicativo nos aportan la base para montar todo el sistema, la más importante es Spring Boot [5] dado que ofrece una batería de herramientas para crear de una manera muy sencilla un aplicativo web, en el caso de este trabajo un conjunto de microservicios.



Además, la herramienta Maven se encarga de resolver de manera automática aquellas librerías que necesita Spring Boot [5].

Por otra parte, los plugins de Maven ayudan a realizar un proceso de integración continua [29] es decir: compilar, ejecutar pruebas, construir y empaquetar. Entre los plugins más destacados y que se han usado en este trabajo se destacan:

- maven-compiler-plugin: Encargado de realizar la compilación del aplicativo.
- maven-surefire-plugin: Encargado de realizar las pruebas de integración y agregar información de cobertura al plugin de jaCoCo.
- jacoco-maven-plugin: Plugin encargado de generar los informes de cobertura del aplicativo.
- maven-checkstyle-plugin: Plugin que realiza una verificación del código para determinar si cumple con unas reglas de estilo y codificación estándar.
- spotbugs-maven-plugin: Plugin para buscar posibles bugs, errores o estructuras ineficientes en la codificación.
- docker-maven-plugin: Plugin para crear las imágenes de los contenedores en Docker.

Finalmente, dado que la filosofía de Maven es conseguir la estandarización de las construcciones de software, permite el uso de perfiles para poder configurar despliegues con diferente configuración parametrizada y usar cualquier tipo de IDE de programación existente hoy en día como, por ejemplo: Eclipse, Netbeans o IntelliJ.

#### Uso de Docker y Docker-compose

Hoy en día, el uso de Docker facilita mucho el trabajo de desarrollo de cualquier aplicativo dado que, con un simple fichero y un par de comandos tienes a disposición del desarrollador una amplia variedad de contenedores con software tan diverso como bases de datos: MySQL, Postgres, Mongo, etc.; o servidores de aplicaciones para pruebas como: Jenkins, Artifactory, etc.

Para el desarrollo de este trabajo se ha usado un fichero YAML [30] junto con Docker-compose [31] para disponer de una base de datos MongoDB [9] y un aplicativo Mongo-Express [32] para la visualización y edición de los datos.

```

mongo.yml x
1 version: '3.1'
2
3 services:
4
5   -- mongo:
6     -- image: mongo
7     -- restart: always
8     -- ports:
9       -- 27017:27017
10    -- environment:
11      -- MONGO_INITDB_ROOT_USERNAME: root
12      -- MONGO_INITDB_ROOT_PASSWORD: toor
13      -- MONGO_INITDB_DATABASE: tfg-monitor-system-db
14    -- healthcheck:
15      -- test: echo 'db.runCommand("ping").ok' | mongo localhost:27017/test --quiet 1
16      -- interval: 10s
17      -- timeout: 10s
18      -- retries: 5
19    -- volumes:
20      -- mongo-vol:/data
21      -- mongo-vol:/data/configdb
22      -- mongo-vol:/data/db
23      -- ./mongo-init.js:/docker-entrypoint-initdb.d/mongo-init.js:ro
24
25   -- mongo-express:
26     -- image: mongo-express
27     -- restart: always
28     -- ports:
29       -- 8081:8081
30     -- environment:
31       -- ME_CONFIG_MONGODB_ADMINUSERNAME: root
32       -- ME_CONFIG_MONGODB_ADMINPASSWORD: toor
33       -- ME_CONFIG_MONGODB_URL: mongodb://root:toor@mongo:27017/
34     -- depends_on:
35       -- mongo
36
37 volumes:
38   -- mongo-vol:

```

**Ilustración 31: Fichero mongo.yml**

Gracias a disponer de Docker-desktop (versión de Windows para Docker) y un simple comando de consola se dispone de una base de datos MongoDB en el equipo en tan solo unos segundos:

```

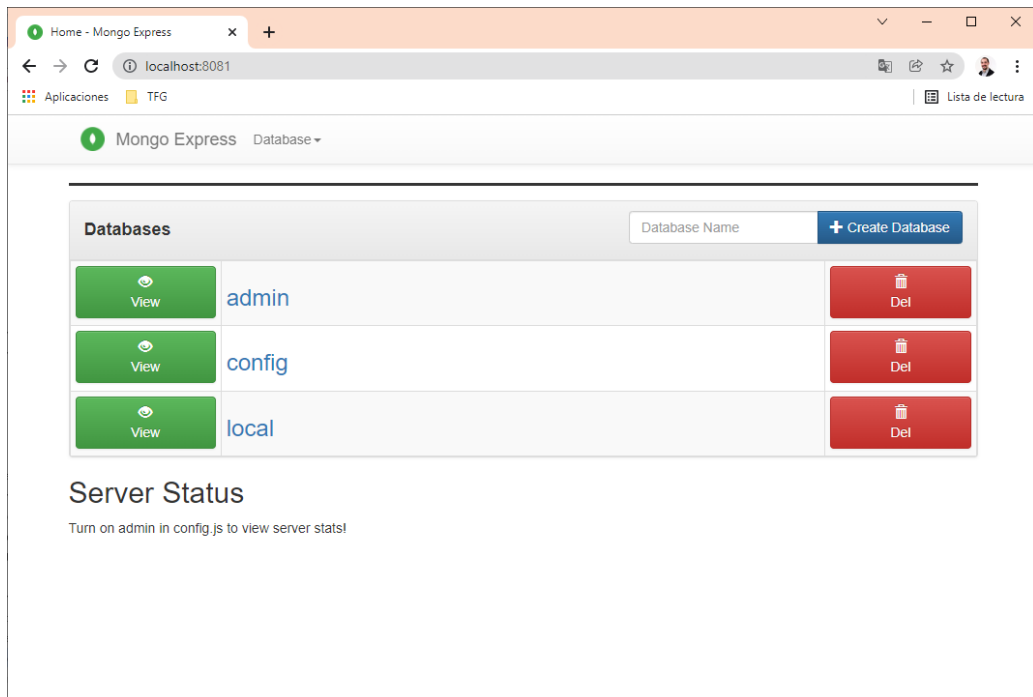
C:\Windows\System32\cmd.exe - docker-compose -f mongo.yml -p mongo-dev up

C:\Proyectos\eclipse-workspace\tfg-monitor-system\docker>docker-compose -f mongo.yml -p mongo-dev up
Creating network "mongo-dev_default" with the default driver
Creating mongo-dev_mongo_1 ... done
Creating mongo-dev_mongo-express_1 ... done
Attaching to mongo-dev_mongo_1, mongo-dev_mongo-express_1
mongo_1 | {"t":{"$date":"2021-12-24T12:13:44.565+00:00"},"s":"I", "c":"CONTROL", "id":23285, "ctx":"-", "msg":
mongo_1 | "Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
mongo_1 | {"t":{"$date":"2021-12-24T12:13:44.567+00:00"},"s":"I", "c":"NETWORK", "id":4915701, "ctx":"-", "msg":
mongo_1 | "Initialized wire specification", "attr":{"spec":{"incomingExternalClient":{"minWireVersion":0,"maxWireVersion":13},"in
mongo_1 | comingInternalClient":{"minWireVersion":0,"maxWireVersion":13},"outgoing":{"minWireVersion":0,"maxWireVersion":13},"isIn
mongo_1 | ternalClient":true}}}
mongo_1 | {"t":{"$date":"2021-12-24T12:13:44.575+00:00"},"s":"W", "c":"ASIO", "id":22601, "ctx":"main", "
mongo_1 | msg":"No TransportLayer configured during NetworkInterface startup"}
mongo_1 | {"t":{"$date":"2021-12-24T12:13:44.576+00:00"},"s":"I", "c":"NETWORK", "id":4648601, "ctx":"main", "
mongo_1 | msg":"Implicit TCP FastOpen unavailable. If TCP FastOpen is required, set tcpFastOpenServer, tcpFastOpenClient, and tcpF
mongo_1 | astOpenQueueSize."}
mongo_1 | {"t":{"$date":"2021-12-24T12:13:44.582+00:00"},"s":"W", "c":"ASIO", "id":22601, "ctx":"main", "
mongo_1 | msg":"No TransportLayer configured during NetworkInterface startup"}
mongo_1 | {"t":{"$date":"2021-12-24T12:13:44.583+00:00"},"s":"I", "c":"REPL", "id":5123008, "ctx":"main", "
mongo_1 | msg":"Successfully registered PrimaryOnlyService", "attr":{"service":"TenantMigrationDonorService", "ns":"config.tenan
mongo_1 | tMigrationDonors"}}
mongo_1 | {"t":{"$date":"2021-12-24T12:13:44.583+00:00"},"s":"I", "c":"REPL", "id":5123008, "ctx":"main", "
mongo_1 | msg":"Successfully registered PrimaryOnlyService", "attr":{"service":"TenantMigrationRecipientService", "ns":"config.tenan
mongo_1 | tMigrationRecipients"}}
mongo_1 | {"t":{"$date":"2021-12-24T12:13:44.583+00:00"},"s":"I", "c":"CONTROL", "id":4615611, "ctx":"initand
mongo_1 | listen", "msg":"MongoDB starting", "attr":{"pid":1, "port":27017, "dbPath":"/data/db", "architecture":"64-bit", "host":"214e62
mongo_1 | aeea37"}}
mongo_1 | {"t":{"$date":"2021-12-24T12:13:44.583+00:00"},"s":"I", "c":"CONTROL", "id":23403, "ctx":"initand
mongo_1 | listen", "msg":"Build Info", "attr":{"buildInfo":{"version":"5.0.3", "gitVersion":"657fea5a61a74d7a79df7aff8e4bcf0bc742b748

```

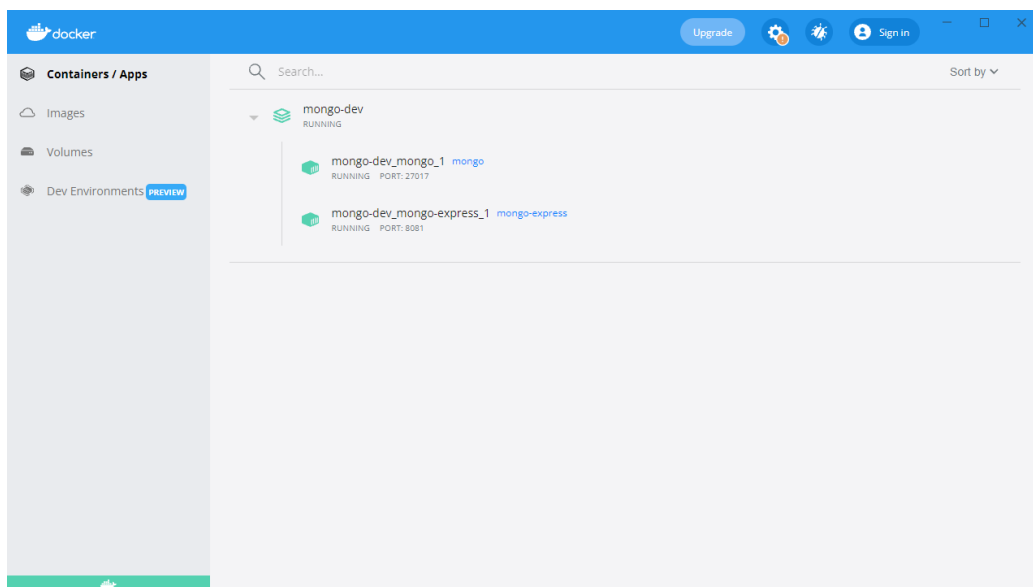
**Ilustración 32: Arranque de mongoDb con Docker-compose**

El Docker-compose se encarga de arrancar dos contenedores, uno con la base de datos de MongoDB y otro contenedor con Mongo Express para poder visualizar datos, modificar o eliminar.



**Ilustración 33: Panel de gestión mongo exprés.**

Desde el panel de gestión de Docker-desktop se puede ver los contenedores arrancados y administrarlos de una forma visual y sencilla:

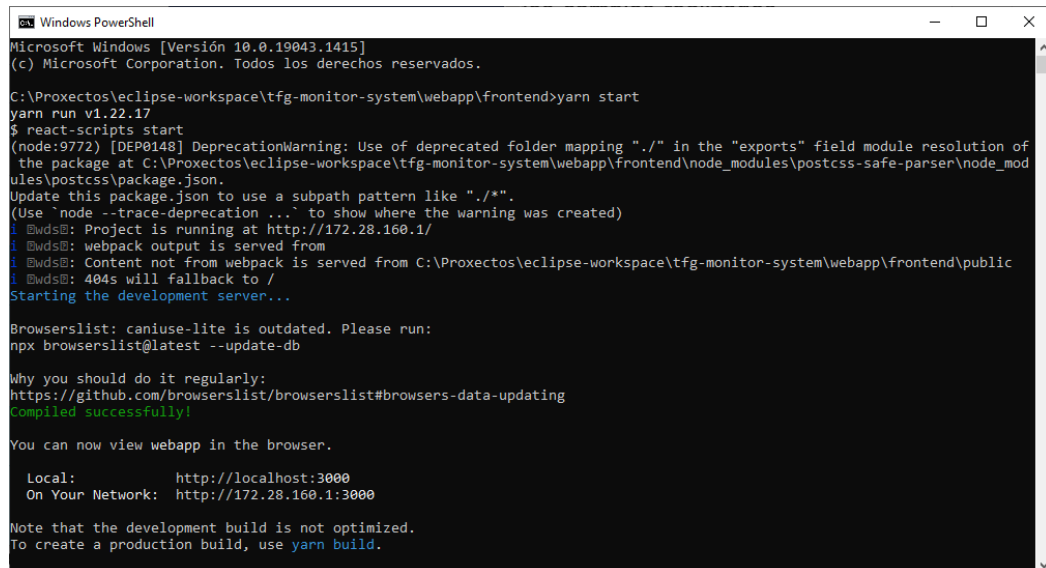


**Ilustración 34: Panel de administración Docker-desktop**

### Uso de node.js y Yarn

Gracias a Maven y el plugin de Yarn [24], el sistema es capaz de construir el aplicativo web basado en React y desplegarlo en un contenedor. No obstante, cuando se está desarrollando realizar esta operativa lleva

tiempo y no sería posible gestionar de manera eficiente la visualización de los cambios realizados en el código. Por este motivo se ha instalado node.js [33] para poder usar Yarn de manera local, disponer de una retroalimentación de posibles errores en el código y poder visualizar de manera inmediata cualquier cambio realizado en los componentes:



```
Windows PowerShell
Microsoft Windows [Versión 10.0.19043.1415]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Proyectos\eclipse-workspace\tfg-monitor-system\webapp\frontend>yarn start
yarn run v1.22.17
$ react-scripts start
(node:9772) [DEP0148] DeprecationWarning: Use of deprecated folder mapping "." in the "exports" field module resolution of
the package at C:\Proyectos\eclipse-workspace\tfg-monitor-system\webapp\frontend\node_modules\postcss-safe-parser\node_mod
ules\postcss\package.json.
Update this package.json to use a subpath pattern like "./*".
(Use `node --trace-deprecation ...` to show where the warning was created)
! @wds@: Project is running at http://172.28.160.1/
! @wds@: webpack output is served from
! @wds@: Content not from webpack is served from C:\Proyectos\eclipse-workspace\tfg-monitor-system\webapp\frontend\public
! @wds@: 404s will fallback to /
Starting the development server...

Browserslist: caniuse-lite is outdated. Please run:
npx browserslist@latest --update-db

Why you should do it regularly:
https://github.com/browserslist/browserslist#browsers-data-updating
Compiled successfully!

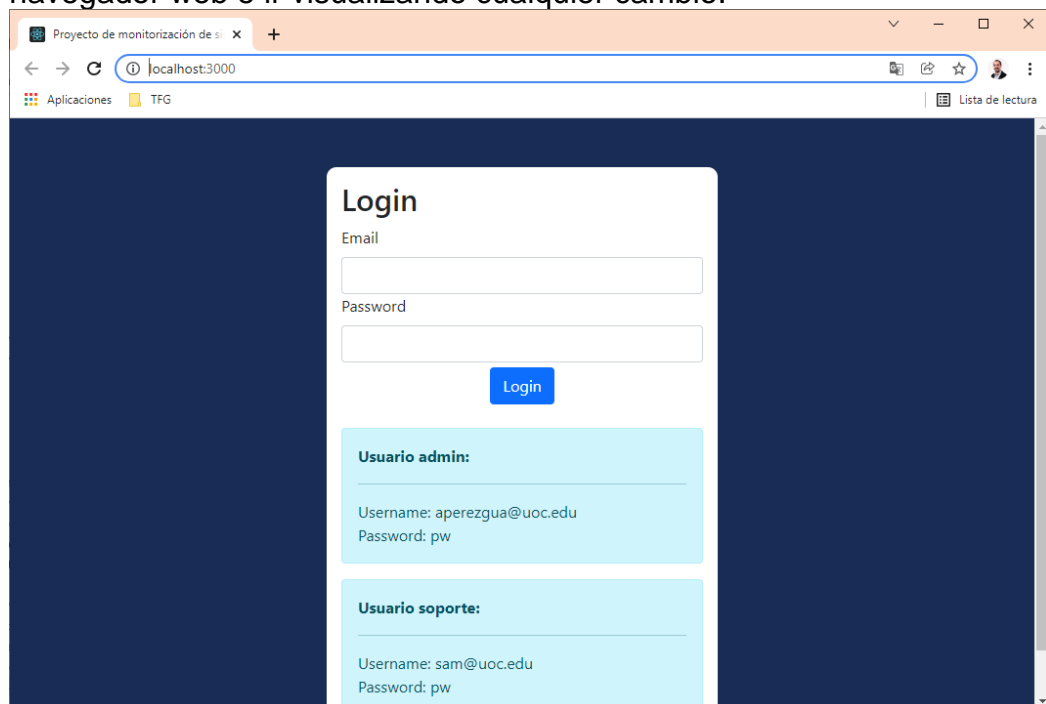
You can now view webapp in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://172.28.160.1:3000

Note that the development build is not optimized.
To create a production build, use yarn build.
```

**Ilustración 35: Arranque de servidor con Yarn de manera local**

Gracias a este servidor, se puede acceder de manera local a través de un navegador web e ir visualizando cualquier cambio:

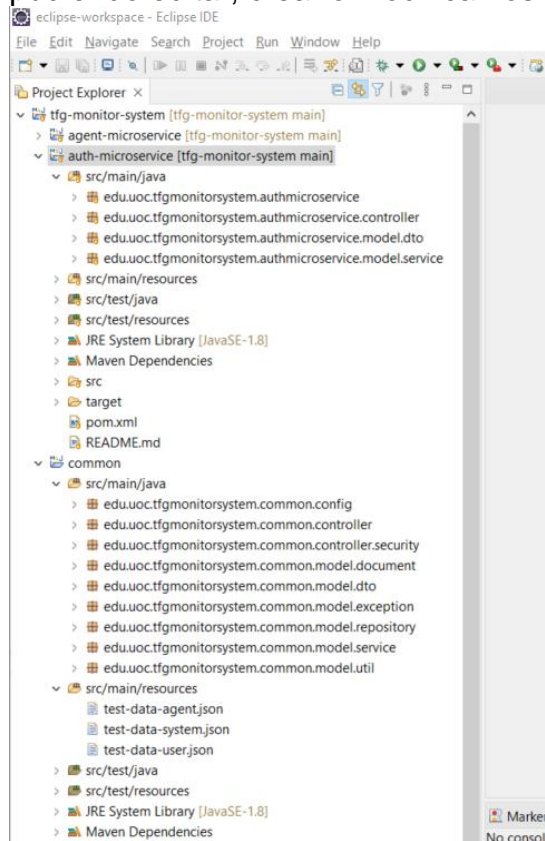


**Ilustración 36: Pantalla de aplicativo React en servidor Yarn local**

Eclipse y proyecto Maven multimodular

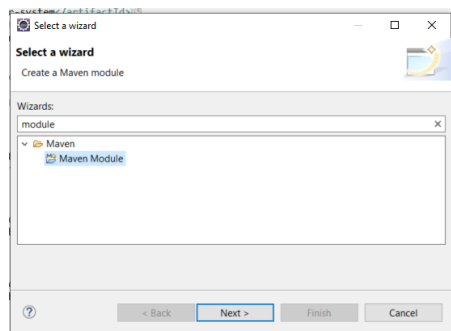
En el desarrollo se ha usado el IDE Eclipse [15] para realizar la programación. En un primer lugar se crea un proyecto Maven

multimodular a partir de una plantilla. Desde Project Explorer se puede acceder fácilmente a cualquier módulo dentro del proyecto principal pudiendo editar, crear o modificar los módulos existentes:



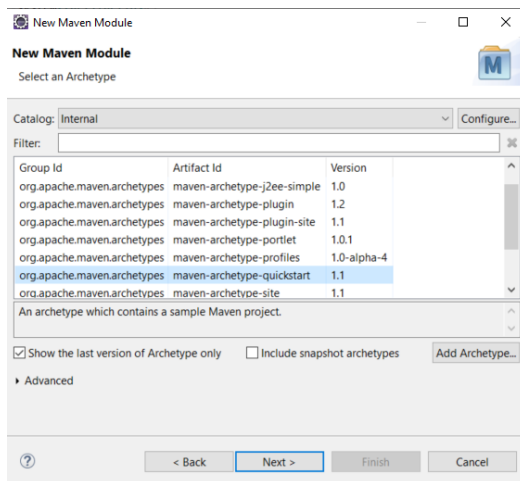
**Ilustración 37: Project explorer de Eclipse**

El desarrollador también puede crear un nuevo módulo a través del menú de eclipse:



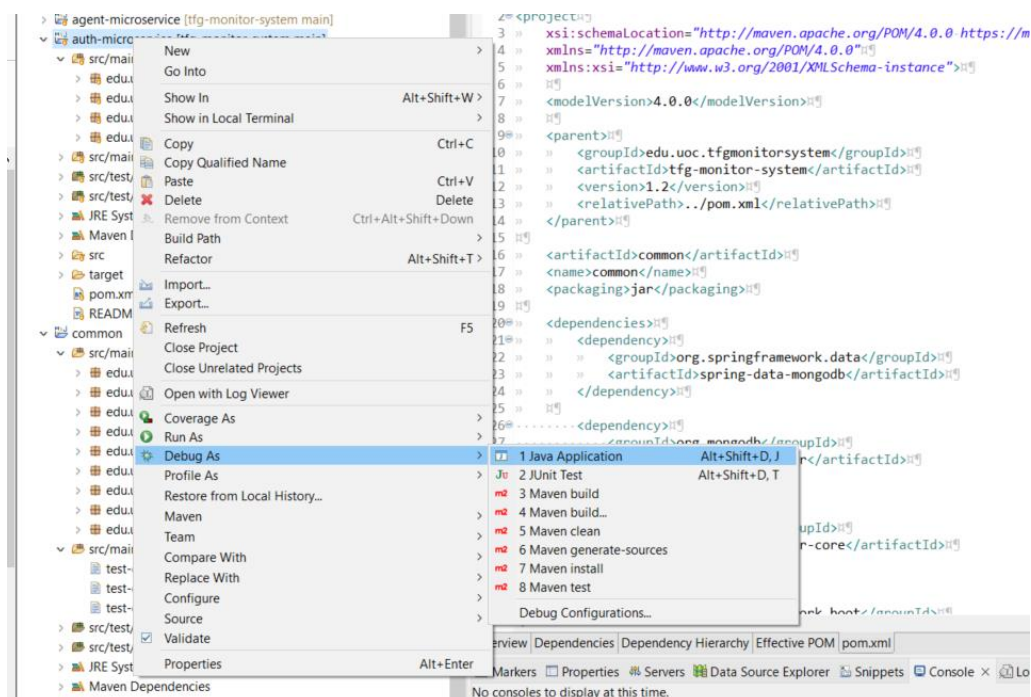
**Ilustración 38: Creación de nuevo modulo**

En este proyecto no se declararon nuevos arquetipos para usar en la creación de un módulo. No obstante, a la hora de seleccionar el arquetipo se elige uno simple y posteriormente se edita manualmente el pom.xml.



**Ilustración 39: Selección de arquetipo**

Otra de las ventajas de usar Eclipse es poder ejecutar cada microservicio en modo *debug* o normal y las pruebas de cobertura de manera muy sencilla. Simplemente con seleccionar el módulo, botón derecho y lanzar una ejecución de la clase principal que contiene el “main”:



**Ilustración 40: Modo debug de un microservicio**

Esta acción lanza el aplicativo y publica el microservicio en el puerto definido en la configuración permitiendo su acceso al aplicativo web otros programas como Postman [34] para verificar su correcto funcionamiento:

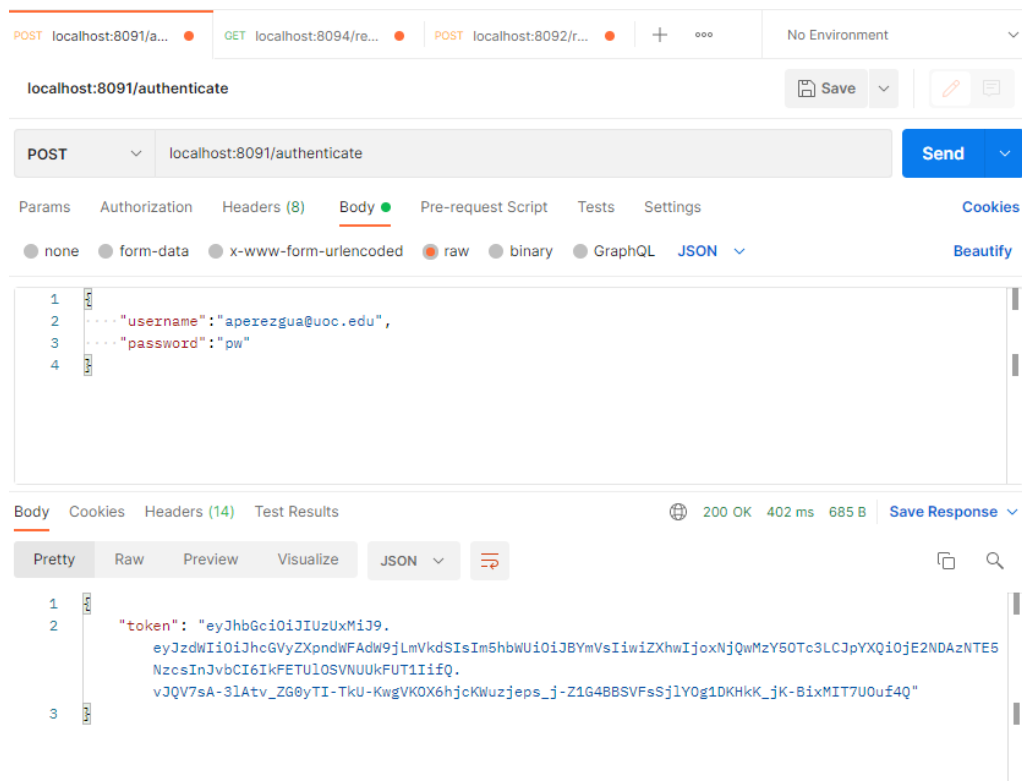
```

Using generated security password: 27c1fc20-41c2-4225-b3e7-28fc258a0260

2021-12-24 14:16:18.746 INFO 11108 --- [main] o.s.s.web.DefaultSecurityFilterChain : Will secure any request with [org.springframework.security.web
2021-12-24 14:16:19.038 INFO 11108 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8091 (http) with context path ''
2021-12-24 14:16:19.049 INFO 11108 --- [main] d.r.i.ResourceReaderRepositoryPopulator : Reading resource: class path resource [test-data-user.json]
2021-12-24 14:16:19.384 INFO 11108 --- [main] org.mongodb.driver.connection : Opened connection [connectionId{localValue:2, serverValue:370}]
2021-12-24 14:16:19.443 INFO 11108 --- [main] d.r.i.ResourceReaderRepositoryPopulator : Reading resource: class path resource [test-data-system.json]
2021-12-24 14:16:19.523 INFO 11108 --- [main] d.r.i.ResourceReaderRepositoryPopulator : Reading resource: class path resource [test-data-agent.json]
2021-12-24 14:16:19.579 INFO 11108 --- [main] e.u.t.a.AuthMicroserviceApp : Started AuthMicroserviceApp in 5.061 seconds (JVM running for
2021-12-24 14:16:19 DEBUG MongoDBConfig:80 - Country: {"id":1,"name":"Espa\u00f1a"}
2021-12-24 14:16:19 DEBUG MongoDBConfig:81 - User: {"active":true,"createdAt":"Tue Nov 16 22:20:47 CET 2021","email":"aperezgua@uoc.edu","id":10001,"name":"Abel"}
2021-12-24 14:16:19 DEBUG MongoDBConfig:82 - Systems: {"active":true,"country":{"id":1,"name":"Espa\u00f1a"},"createdAt":"Tue Nov 16 22:20:47 CET 2021","id":1000}
2021-12-24 14:16:19 DEBUG MongoDBConfig:83 - Agent: {"active":true,"createdAt":"Tue Nov 16 22:20:47 CET 2021","name":"Agente 1","rules":[{"active":true,"agent":n

```

**Ilustración 41: Consola de log del microservicio**



**Ilustración 42: Postman consultando el microservicio de autenticación**

## 2.6 Despliegue del aplicativo

Este aplicativo se puede desplegar en sistemas *cloud* como Amazon Web Services, Google Cloud u otro tipo de sistemas como Kubernetes gracias a su estructura de microservicios containerizados. No obstante, por un motivo de recursos se ha decidido alquilar un pequeño servidor privado virtual con 2GB de Ram para realizar las pruebas de despliegue con Dockers accesible desde Internet a través de la URL <http://vps-1f79b26a.vps.ovh.net:8080/>. A continuación, se describe cómo se ha realizado el despliegue del aplicativo a través de una terminal de Debian Linux.

### Compilación y generación de imágenes

Primeramente, se realiza el clonado del proyecto contra el repositorio de GitHub obteniendo el código fuente:



```
debian@vps-1f79b26a: ~/tfg-monitor-system
debian@vps-1f79b26a:~/tfg-monitor-system$ ls -lsa
total 80
4 drwxr-xr-x 14 debian debian 4096 Dec 14 21:19 .
4 drwxr-xr-x 10 debian debian 4096 Dec 14 21:21 ..
4 drwxr-xr-x 4 debian debian 4096 Dec 14 21:20 agent-microservice
4 drwxr-xr-x 4 debian debian 4096 Dec 14 21:19 auth-microservice
4 drwxr-xr-x 4 debian debian 4096 Dec 14 21:19 common
4 drwxr-xr-x 2 debian debian 4096 Dec 14 21:00 docker
4 drwxr-xr-x 8 debian debian 4096 Dec 24 13:28 .git
4 -rw-r--r-- 1 debian debian 336 Dec 10 12:15 .gitignore
4 drwxr-xr-x 4 debian debian 4096 Dec 14 21:21 log4j-appender
4 drwxr-xr-x 4 debian debian 4096 Dec 14 21:20 log-microservice
16 -rw-r--r-- 1 debian debian 13622 Dec 13 20:28 pom.xml
4 -rw-r--r-- 1 debian debian 3270 Dec 13 20:28 README.md
4 drwxr-xr-x 3 debian debian 4096 Dec 10 12:15 src
4 drwxr-xr-x 4 debian debian 4096 Dec 14 21:20 system-microservice
4 drwxr-xr-x 4 debian debian 4096 Dec 14 21:19 target
4 drwxr-xr-x 4 debian debian 4096 Dec 14 21:20 user-microservice
4 drwxr-xr-x 5 debian debian 4096 Dec 14 21:21 webapp
debian@vps-1f79b26a:~/tfg-monitor-system$
```

Ilustración 43: Consola de Debian con el proyecto descargado

A continuación, se indica a Maven los comandos de compilación y despliegue en Docker usando el perfil *Docker*. Con el siguiente comando:

```
mvn clean install docker:build -P docker
```

El sistema comienza el despliegue realizando las diferentes fases de los plugins para cada módulo: checkstyle, spotbugs, test de integración, generación de cobertura, creación de paquete y construcción de imagen Docker si las fases anteriores han sido satisfactorias:

```
debian@vps-1f79b26a: ~/tfg-monitor-system
debian@vps-1f79b26a:~/tfg-monitor-system$ mvn clean install docker:build -P docker
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by com.google.inject.internal.cglib.core.$ReflectUtils$1 (file:/usr/share/maven/lib/guice.jar) t
o method java.lang.ClassLoader.defineClass(java.lang.String,byte[],int,int,java.security.ProtectionDomain)
WARNING: Please consider reporting this to the maintainers of com.google.inject.internal.cglib.core.$ReflectUtils$1
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
[INFO] Scanning for projects...
[INFO] -----
[INFO] Reactor Build Order:
[INFO]
[INFO] tfg-monitor-system [pom]
[INFO] common [jar]
[INFO] auth-microservice [jar]
[INFO] user-microservice [jar]
[INFO] system-microservice [jar]
[INFO] agent-microservice [jar]
[INFO] log-microservice [jar]
[INFO] webapp [jar]
[INFO] log4j-appender [jar]
[INFO] -----
[INFO] -----< edu.uoc.tfgmonitorsystem:tfg-monitor-system >-----
[INFO] Building tfg-monitor-system 1.2 [1/9]
[INFO] -----[ pom ]-----
[INFO]
[INFO] --- maven-clean-plugin:3.1.0:clean (default-clean) @ tfg-monitor-system ---
[INFO]
[INFO] --- jacoco-maven-plugin:0.8.7:instrument (default-instrument) @ tfg-monitor-system ---
[INFO] Skipping JaCoCo execution due to missing classes directory:/home/debian/tfg-monitor-system/target/classes
[INFO]
[INFO] --- maven-checkstyle-plugin:3.1.2:check (quality-metric-checkstyle) @ tfg-monitor-system ---
[INFO] You have 0 Checkstyle violations.
[INFO]
[INFO] --- jacoco-maven-plugin:0.8.7:restore-instrumented-classes (default-restore-instrumented-classes) @ tfg-monitor-system ---
[INFO]
[INFO] >>> spotbugs-maven-plugin:4.2.3:check (quality-metric-spotbugs) > :spotbugs @ tfg-monitor-system >>>
[INFO]
[INFO] --- spotbugs-maven-plugin:4.2.3:spotbugs (spotbugs) @ tfg-monitor-system ---
debian@vps-1f79b26a:~/tfg-monitor-system$
```

Ilustración 44: Arranque de compilación en Debian



```
debian@vps-1f79b26a: ~/tfg-monitor-system
[INFO] Results:
[INFO]
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] --- maven-checkstyle-plugin:3.1.2:check (quality-metric-checkstyle) @ common ---
[INFO] You have 0 Checkstyle violations.
[INFO]
[INFO] --- jacoco-maven-plugin:0.8.7:restore-instrumented-classes (default-restore-instrumented-classes) @ common ---
[INFO]
[INFO] >>> spotbugs-maven-plugin:4.2.3:check (quality-metric-spotbugs) > :spotbugs @ common >>>
[INFO]
[INFO] --- spotbugs-maven-plugin:4.2.3:spotbugs (spotbugs) @ common ---
[INFO] Fork Value is true
[INFO] Done SpotBugs Analysis....
[INFO]
[INFO] <<< spotbugs-maven-plugin:4.2.3:check (quality-metric-spotbugs) < :spotbugs @ common <<<
[INFO]
[INFO] --- spotbugs-maven-plugin:4.2.3:check (quality-metric-spotbugs) @ common ---
[INFO] BugInstance size is 0
[INFO] Error size is 0
[INFO] No errors/warnings found
[INFO]
[INFO] --- maven-jar-plugin:3.0.2:jar (default-jar) @ common ---
[INFO] Building jar: /home/debian/tfg-monitor-system/common/target/common-1.2.jar
[INFO]
[INFO] --- jacoco-maven-plugin:0.8.7:report (default-report) @ common ---
[INFO] Loading execution data file /home/debian/tfg-monitor-system/common/target/jacoco.exec
[INFO] Analyzed bundle 'common' with 30 classes
[INFO]
[INFO] --- maven-install-plugin:2.5.2:install (default-install) @ common ---
[INFO] Installing /home/debian/tfg-monitor-system/common/target/common-1.2.jar to /home/debian/.m2/repository/edu/uoc/tfgmonitorsys
tem/common/1.2/common-1.2.jar
[INFO] Installing /home/debian/tfg-monitor-system/common/pom.xml to /home/debian/.m2/repository/edu/uoc/tfgmonitorsystem/common/1.2
/common-1.2.pom
[INFO]
[INFO] --- docker-maven-plugin:0.37.0:build (default-cli) @ common ---
[INFO]
```

Ilustración 45: Diferentes fases de compilación en Debian

```
debian@vps-1f79b26a: ~/tfg-monitor-system
[INFO]
[INFO] --- jacoco-maven-plugin:0.8.7:report (default-report) @ log4j-appender ---
[INFO] Skipping JaCoCo execution due to missing execution data file.
[INFO]
[INFO] --- maven-install-plugin:2.5.2:install (default-install) @ log4j-appender ---
[INFO] Installing /home/debian/tfg-monitor-system/log4j-appender/target/log4j-appender-1.2.jar to /home/debian/.m2/repository/edu/u
oc/tfgmonitorsystem/log4j-appender/1.2/log4j-appender-1.2.jar
[INFO] Installing /home/debian/tfg-monitor-system/log4j-appender/pom.xml to /home/debian/.m2/repository/edu/uoc/tfgmonitorsystem/lo
g4j-appender/1.2/log4j-appender-1.2.pom
[INFO]
[INFO] --- docker-maven-plugin:0.37.0:build (default-cli) @ log4j-appender ---
[WARNING] The following patterns were never triggered in this artifact exclusion filter:
o '*:*:*--SNAPSHOT'
[INFO]
[INFO] Copying files to /home/debian/tfg-monitor-system/log4j-appender/target/docker/log4j-appender/1.2/build/deps-release
[INFO] Copying files to /home/debian/tfg-monitor-system/log4j-appender/target/docker/log4j-appender/1.2/build/maven
[INFO] Building tar: /home/debian/tfg-monitor-system/log4j-appender/target/docker/log4j-appender/1.2/tmp/docker-build.tar
[INFO] DOCKER> [log4j-appender:1.2] "log4j-appender": Created docker-build.tar in 1 second
[INFO] DOCKER> [log4j-appender:1.2] "log4j-appender": Built image sha256:9e9dd
[INFO] DOCKER> [log4j-appender:1.2] "log4j-appender": Removed old image sha256:998b0
[INFO]
[INFO] Reactor Summary for tfg-monitor-system 1.2:
[INFO]
[INFO] tfg-monitor-system ..... SUCCESS [ 10.311 s]
[INFO] common ..... SUCCESS [ 29.011 s]
[INFO] auth-microservice ..... SUCCESS [ 34.761 s]
[INFO] user-microservice ..... SUCCESS [ 31.126 s]
[INFO] system-microservice ..... SUCCESS [ 44.589 s]
[INFO] agent-microservice ..... SUCCESS [ 30.023 s]
[INFO] log-microservice ..... SUCCESS [ 47.692 s]
[INFO] webapp ..... SUCCESS [ 46.719 s]
[INFO] log4j-appender ..... SUCCESS [ 13.101 s]
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 04:50 min
[INFO] Finished at: 2021-12-24T14:05:04Z
[INFO]
```

Ilustración 46: Resultado de la compilación en Debian.

Una vez finalizadas las tareas de Maven, se pueden visualizar las diferentes imágenes generadas en el servidor Docker local:

```
debian@vps-1f79b26a: ~/tfg-monitor-system
debian@vps-1f79b26a:~/tfg-monitor-system$ docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
log4j-appender       1.2            a89fdb7550b5   4 minutes ago   646MB
webapp               1.2            adea063353cc   4 minutes ago   662MB
log-microservice     1.2            5f837b0b206c   5 minutes ago   672MB
agent-microservice   1.2            9afeb5e9c552   5 minutes ago   672MB
system-microservice  1.2            f854bd723536   5 minutes ago   672MB
user-microservice    1.2            f724972d0298   5 minutes ago   672MB
auth-microservice    1.2            c9cdeed35bfc   6 minutes ago   672MB
mongo                latest         4253856b2570   5 weeks ago     701MB
java                 8             d23bdf5b1b1b   4 years ago     643MB
debian@vps-1f79b26a:~/tfg-monitor-system$
```

Ilustración 47: Imágenes en Debian

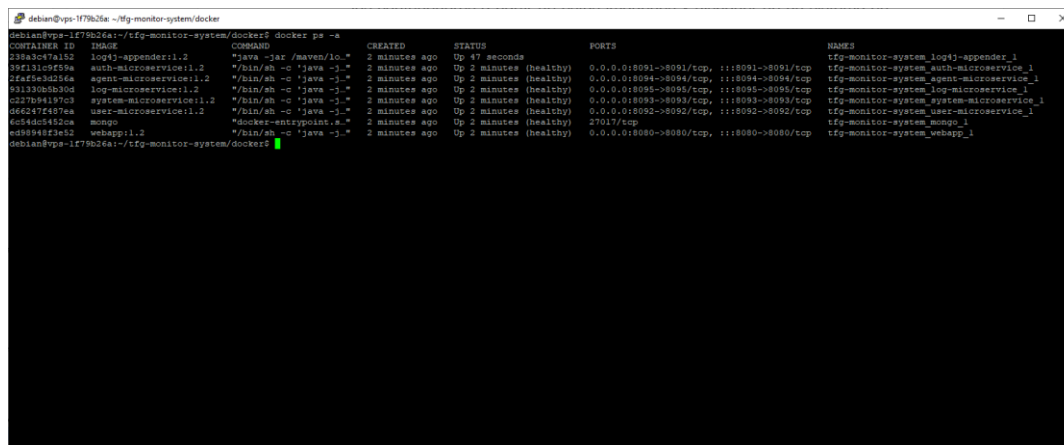
## Arranque con Docker-compose

Se realiza la llamada al Docker-compose usando el fichero *tfg-monitor-system.yml* para el despliegue, de tal manera que cree los contenedores a partir de las imágenes previamente generadas por Maven y disponer de un aplicativo en funcionamiento y publicado en Internet. Para ello se debe ejecutar el siguiente comando:

```
docker-compose -p tfg-monitor-system -f tfg-monitor-system.yml up
```

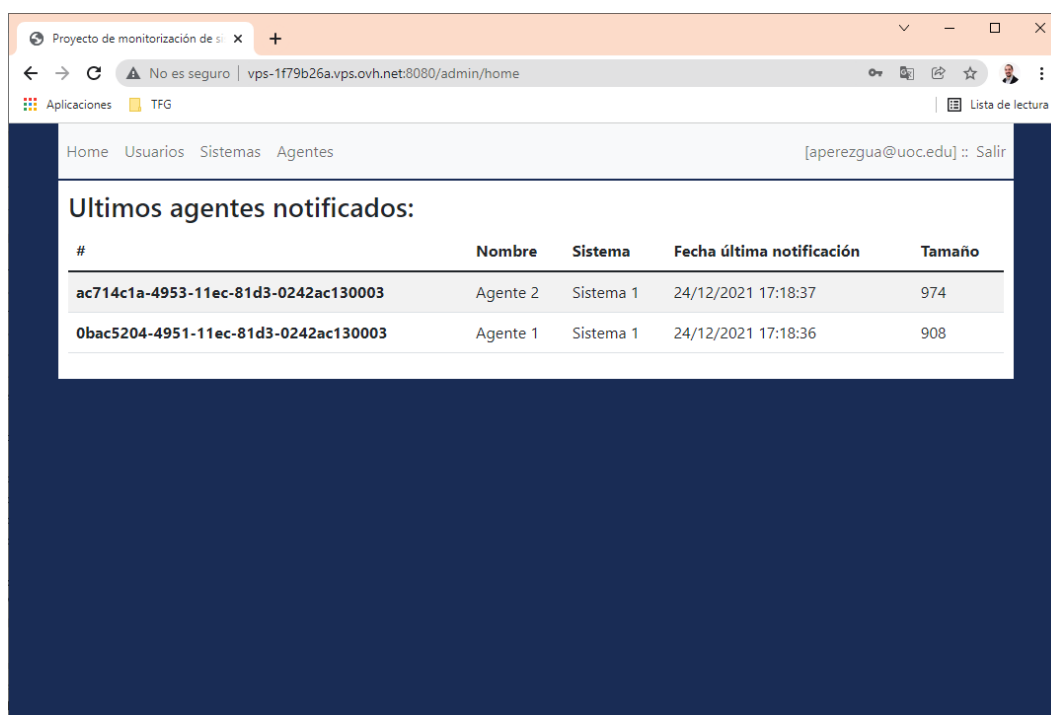
```
debian@vps-1f79b26a: ~/tfg-monitor-system/docker
debian@vps-1f79b26a:~/tfg-monitor-system/docker$ docker-compose -p tfg-monitor-system -f tfg-monitor-system.yml up
Creating volume "tfg-monitor-system_mongo-vol" with default driver
Creating tfg-monitor-system_webapp_1 ... done
Creating tfg-monitor-system_mongo_1 ... done
Creating tfg-monitor-system_user-microservice_1 ... done
Creating tfg-monitor-system_system-microservice_1 ... done
Creating tfg-monitor-system_agent-microservice_1 ... done
Creating tfg-monitor-system_log-microservice_1 ... done
Creating tfg-monitor-system_auth-microservice_1 ... done
Creating tfg-monitor-system_log4j-appender_1 ... done
Attaching to tfg-monitor-system_mongo_1, tfg-monitor-system_webapp_1, tfg-monitor-system_auth-microservice_1, tfg-monitor-system_log-microservice_1, tfg-monitor-system_agent-microservice_1, tfg-monitor-system_user-microservice_1, tfg-monitor-system_system-microservice_1, tfg-monitor-system_log4j-appender_1
mongo_1 | about to fork child process, waiting until server is ready for connections.
mongo_1 | forked process: 30
mongo_1 | {"t":{"$date":"2021-12-24T16:11:15.976+00:00"},"s":"I", "c":"CONTROL", "id":20698, "ctx":"-", "msg":"*** SERVER RESTARTED ***"}
mongo_1 | {"t":{"$date":"2021-12-24T16:11:16.004+00:00"},"s":"I", "c":"NETWORK", "id":4915701, "ctx":"main", "msg":"Initialized wire specification", "attr":{"spec":{"incomingExternalClient":{"minWireVersion":0,"maxWireVersion":13},"incomingInternalClient":{"minWireVersion":0,"maxWireVersion":13},"outgoing":{"minWireVersion":0,"maxWireVersion":13},"isInternalClient":true}}}
mongo_1 | {"t":{"$date":"2021-12-24T16:11:16.008+00:00"},"s":"I", "c":"CONTROL", "id":23285, "ctx":"main", "msg":"Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
mongo_1 | {"t":{"$date":"2021-12-24T16:11:16.010+00:00"},"s":"W", "c":"ASIO", "id":22601, "ctx":"main", "msg":"No TransportLayer configured during NetworkInterface startup"}
mongo_1 | {"t":{"$date":"2021-12-24T16:11:16.011+00:00"},"s":"I", "c":"NETWORK", "id":4648601, "ctx":"main", "msg":"Implicit TCP FastOpen unavailable. If TCP FastOpen is required, set tcpFastOpenServer, tcpFastOpenClient, and tcpFastOpenQueueSize"}
mongo_1 | {"t":{"$date":"2021-12-24T16:11:16.018+00:00"},"s":"W", "c":"ASIO", "id":22601, "ctx":"main", "msg":"No TransportLayer configured during NetworkInterface startup"}
mongo_1 | {"t":{"$date":"2021-12-24T16:11:16.018+00:00"},"s":"I", "c":"REPL", "id":5123008, "ctx":"main", "msg":"Successfully registered PrimaryOnlyService", "attr":{"service":"TenantMigrationDonorService", "ns":"config.tenantMigrationDonors"}}
mongo_1 | {"t":{"$date":"2021-12-24T16:11:16.018+00:00"},"s":"I", "c":"REPL", "id":5123008, "ctx":"main", "msg":"Successfully registered PrimaryOnlyService", "attr":{"service":"TenantMigrationRecipientService", "ns":"config.tenantMigrationRecipients"}}
mongo_1 | {"t":{"$date":"2021-12-24T16:11:16.019+00:00"},"s":"I", "c":"CONTROL", "id":5945603, "ctx":"main", "msg":"Multi threading initialized"}
mongo_1 | {"t":{"$date":"2021-12-24T16:11:16.019+00:00"},"s":"I", "c":"CONTROL", "id":4615611, "ctx":"initandlisten", "msg":"MongoDB starting", "attr":{"pid":30, "port":27017, "dbPath":"/data/db", "architecture":"64-bit", "host":"b32cb76alc52"}}
```

Ilustración 48: Arranque de aplicativo containerizado en Debian



**Ilustración 49: Contenedores inicializados en Debian**

Una vez que los servicios están *healthy* es posible acceder al aplicativo mediante un navegador web consultando la URL <http://vps-1f79b26a.vps.ovh.net:8080/>



**Ilustración 50: Home del módulo administración en Debian**

### Prueba de tolerancia a fallos

Gracias a disponer de un sistema contenerizado, en este caso Docker, es capaz de detectar si uno de los contenedores ha sufrido algún fallo y autorrecuperarse sin necesidad de realizar ninguna acción.

Como se ha visto en la ilustración previa de los contenedores inicializados en Debian, su estado es *healthy*. En el caso del microservicio de autenticación, Docker está consultando cada 10 segundos la llamada a: <http://vps-1f79b26a.vps.ovh.net:8091/health>. Si el microservicio del

contenedor deja de responder o se produce un error Docker vuelve a arrancar el contenedor que ha sufrido un error o parado inesperadamente:

```
debian@vps-1f79b26a:~$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
73108e3e4058	log4j-appender:1.2	"java -jar /maven/lo..."	50 minutes ago	Up About a minute		tfq-monitor-system_log4j-appender_1
d6e7eb70f6c0	user-microservice:1.2	"/bin/sh -c 'java -j..."	50 minutes ago	Up 50 minutes (healthy)	0.0.0.0:8092->8092/tcp, :::8092->8092/tcp	tfq-monitor-system_user-microservice_1
34d47d33699c	system-microservice:1.2	"/bin/sh -c 'java -j..."	50 minutes ago	Up 50 minutes (healthy)	0.0.0.0:8093->8093/tcp, :::8093->8093/tcp	tfq-monitor-system_system-microservice_1
15c63a24f7b	auth-microservice:1.2	"/bin/sh -c 'java -j..."	50 minutes ago	Up 50 minutes (healthy)	0.0.0.0:8091->8091/tcp, :::8091->8091/tcp	tfq-monitor-system_auth-microservice_1
9098c8783cf3	agent-microservice:1.2	"/bin/sh -c 'java -j..."	50 minutes ago	Up 50 minutes (healthy)	0.0.0.0:8094->8094/tcp, :::8094->8094/tcp	tfq-monitor-system_agent-microservice_1
5fed21e80449	log-microservice:1.2	"/bin/sh -c 'java -j..."	50 minutes ago	Up 50 minutes (healthy)	0.0.0.0:8095->8095/tcp, :::8095->8095/tcp	tfq-monitor-system_log-microservice_1
3ed3104d058e	webapp:1.2	"/bin/sh -c 'java -j..."	50 minutes ago	Up 50 minutes (healthy)	0.0.0.0:8080->8080/tcp, :::8080->8080/tcp	tfq-monitor-system_webapp_1
d65632aae701	mongo	"docker-entrypoint.s..."	50 minutes ago	Up 50 minutes (healthy)	27017/tcp	tfq-monitor-system_mongo_1

**Ilustración 51: Autoarranque de microservicio auth caído**

Una vez en estado *healthy* el microservicio de autenticación vuelve a responder a las peticiones.

## 2.7 Producto final

Existen dos maneras diferentes de interactuar con los microservicios publicados. Una es a través de un *appender* de log4j que recoge el log de un aplicativo y lo envía al microservicio de procesamiento de log realizando una autenticación previa y otra es a través de un aplicativo web con React.

### Perfil de agente

El perfil de agente, usado a través del *appender* de log4j, es el encargado de enviar el log al microservicio encargado de procesarlo. Para ello, se ha creado un componente que lanza log aleatoriamente y con el *appender* configurado en log4j.xml es capaz de enviarlo satisfactoriamente.

```

tfg-monitor-system/pom.xml  common/pom.xml  log4j.xml x
1 <!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">
2 <log4j:configuration debug="true" xmlns:log4j="http://jakarta.apache.org/Log4j/">
3
4   <appender name="agentAppender1" class="edu.uoc.tfgmonitorsystem.Log4j.appender.AgentAppender">
5       <param name="authenticationUrl" value="http://@server.host:8091/authenticate" />
6       <param name="putLogUrl" value="http://@server.host:8095/rest/Log/put" />
7       <param name="agentToken" value="ac714c1a-4953-11ec-81d3-0242ac130003" />
8       <layout class="org.apache.Log4j.PatternLayout">
9           <param name="ConversionPattern" value="%d{yyyy-MM-dd HH:mm:ss}.%-5p %c{1}:%L-- %m%n" />
10      </layout>
11   </appender>
12
13   <appender name="agentAppender2" class="edu.uoc.tfgmonitorsystem.Log4j.appender.AgentAppender">
14       <param name="authenticationUrl" value="http://@server.host:8091/authenticate" />
15       <param name="putLogUrl" value="http://@server.host:8095/rest/Log/put" />
16       <param name="agentToken" value="0bac5204-4951-11ec-81d3-0242ac130003" />
17       <layout class="org.apache.Log4j.PatternLayout">
18           <param name="ConversionPattern" value="%d{yyyy-MM-dd HH:mm:ss}.%-5p %c{1}:%L-- %m%n" />
19       </layout>
20   </appender>
21
22   <appender name="console" class="org.apache.Log4j.ConsoleAppender">
23       <param name="Target" value="System.out" />
24       <layout class="org.apache.Log4j.PatternLayout">
25           <param name="ConversionPattern" value="%d{yyyy-MM-dd HH:mm:ss}.%-5p %c{1}:%L-- %m%n" />
26       </layout>
27   </appender>
28
29
30   <logger name="edu.uoc.tfgmonitorsystem.Log4j.app.TestAppender1" additivity="false">
31       <level value="DEBUG" />
32       <appender-ref ref="agentAppender1" />
33   </logger>
34
35   <logger name="edu.uoc.tfgmonitorsystem.Log4j.app.TestAppender2" additivity="false">
36       <level value="DEBUG" />
37       <appender-ref ref="agentAppender2" />
38   </logger>
39

```

**Ilustración 52: Log4j.xml con appenders configurados**

Como se puede observar, el sistema captura el log de 2 clases diferentes y envía las líneas de log generadas gracias a los *appenders* configurados. Cada *appender* configurado requiere una URL de autenticación, un tokenId válido y una URL para el guardado. En la siguiente imagen se puede observar como el log es enviado:

```

debian@vps-1f79b26a: ~
Fri Dec 24 19:13:04 UTC 2021 > Line send correctly: agentTokenId=ac714c1a-4953-11ec-81d3-0242ac130003, logLine=Nivel cri
tico
Fri Dec 24 19:13:04 UTC 2021 > Line send correctly: agentTokenId=ac714c1a-4953-11ec-81d3-0242ac130003, logLine=La ejecu
ción ha tardado 8s
Stop current thread

#####
Worker not run, start it
Start current thread
Fri Dec 24 19:13:05 UTC 2021 > Line send correctly: agentTokenId=0bac5204-4951-11ec-81d3-0242ac130003, logLine=El sistem
a no responde
Fri Dec 24 19:13:05 UTC 2021 > Line send correctly: agentTokenId=0bac5204-4951-11ec-81d3-0242ac130003, logLine=La ejecu
ción ha tardado 2s
Stop current thread

#####
Worker not run, start it
Start current thread
Fri Dec 24 19:13:05 UTC 2021 > Line send correctly: agentTokenId=ac714c1a-4953-11ec-81d3-0242ac130003, logLine=Se ha pro
ducido un error en el sistema
Fri Dec 24 19:13:05 UTC 2021 > Line send correctly: agentTokenId=ac714c1a-4953-11ec-81d3-0242ac130003, logLine=La ejecu
ción ha tardado 1s
Stop current thread
debian@vps-1f79b26a:~$

```

**Ilustración 53: Imagen con el log enviado por los agentes.**

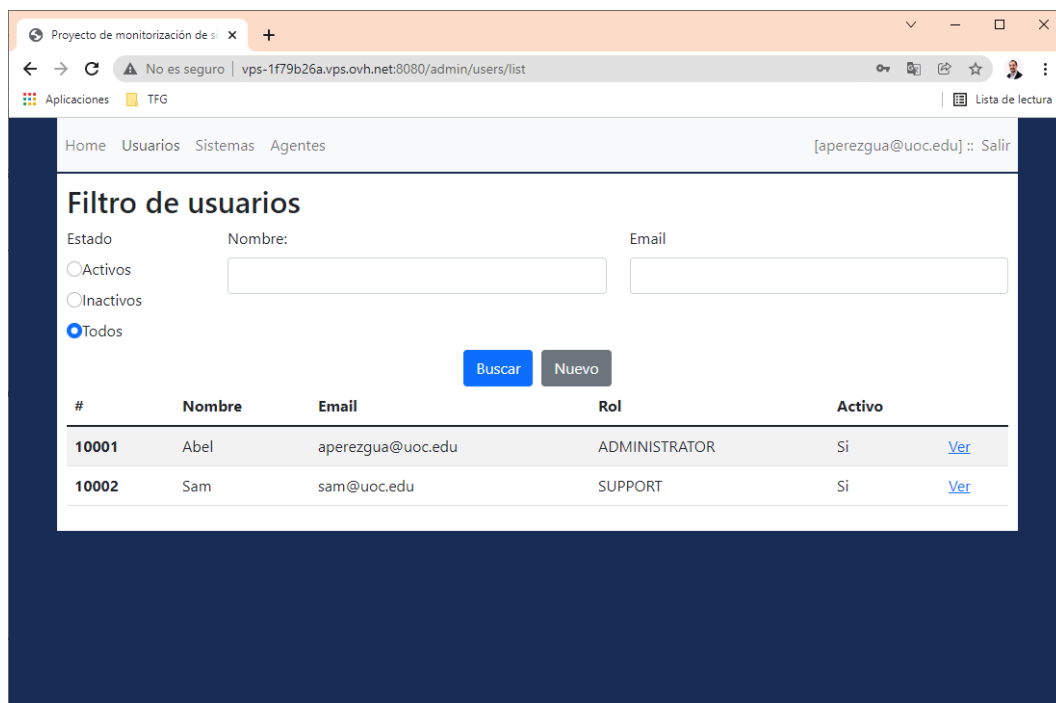
Perfil de administrador

Dispone de un menú desde el que se puede acceder a la administración de usuarios, sistemas (agrupación de agentes) y agentes. También dispone

de una pantalla principal en la que se puede ver un resumen de los agentes activos (ver Ilustración 48).

### *Gestión de usuarios*

La pantalla de gestión de usuarios permite buscar usuarios activos, inactivos o todos, por su nombre o por su email. Además, tiene un botón que permite agregar un nuevo usuario o un enlace para poder editar el usuario.



**Ilustración 54: Panel de gestión de usuarios**

En la pantalla de edición de usuarios se permite cambiar los datos del usuario siendo siempre obligatoria la introducción de la clave:

**Ilustración 55: Pantalla de edición de usuario**

### *Gestión de sistemas*

El aplicativo permite la gestión de sistemas que son una forma de agrupación de los agentes. Un sistema se clasifica en un país determinado y queda habilitado o inhabilitado para que un agente se pueda clasificar en él. Al igual que en gestión de usuarios, se permite su filtrado, creación de nuevos sistemas y edición:

#	Nombre	País	Activo	
10001	Sistema 1	España	Si	<a href="#">Ver</a>
10002	Sistema 2	Portugal	Si	<a href="#">Ver</a>
10003	Sistema 3	Francia	Si	<a href="#">Ver</a>

**Ilustración 56: Filtrado de sistemas**

También, al igual que usuarios, permite su edición:

The screenshot shows a web browser window with the URL `vps-1f79b26a.vps.ovh.net:8080/admin/systems/edit/10001`. The page title is 'Edición de sistema'. It contains a form with the following fields: 'Nombre:' with a text input containing 'Sistema 1'; 'País:' with a dropdown menu showing 'España'; and 'Activo:' with a dropdown menu showing 'Activo'. There is a blue 'Guardar' button at the bottom right of the form. The navigation bar at the top includes 'Home', 'Usuarios', 'Sistemas', and 'Agentes', along with a user profile '[aperezgua@uoc.edu] :: Salir'.

Ilustración 57: Edición de un sistema

### *Gestión de agentes*

Al igual que en la gestión de usuarios y agentes, se permite su filtrado:

The screenshot shows a web browser window with the URL `vps-1f79b26a.vps.ovh.net:8080/admin/agents/list`. The page title is 'Filtro de agentes'. It contains a form with the following fields: 'Estado' with radio buttons for 'Activos', 'Inactivos', and 'Todos' (selected); 'Nombre:' with a text input; and 'Sistema' with a dropdown menu. There are 'Buscar' and 'Nuevo' buttons. Below the form is a table with the following data:

#	Nombre	Sistema	Activo
0bac5204-4951-11ec-81d3-0242ac130003	Agente 1	Sistema 1	Si <a href="#">Ver</a>
ac714c1a-4953-11ec-81d3-0242ac130003	Agente 2	Sistema 1	Si <a href="#">Ver</a>

The navigation bar at the top includes 'Home', 'Usuarios', 'Sistemas', and 'Agentes', along with a user profile '[aperezgua@uoc.edu] :: Salir'.

Ilustración 58: Filtrado de agentes en el sistema

A diferencia de la edición de usuarios y sistemas, la edición de agentes dispone de componentes React más avanzados. En la pantalla principal se



pueden generar tokens de manera aleatoria que identifican al agente, también agregar nuevas reglas o editar las existentes:

Proyecto de monitorización de s... x +

No es seguro | vps-1f79b26a.vps.ovh.net:8080/admin/agents/edit/0bac5204-4951-11ec-81d3-0242ac130003

Aplicaciones TFG

Home Usuarios Sistemas Agentes [aperezgua@uoc.edu] :: Salir

## Edición de agente

Token:

0bac5204-4951-11ec-81d3-0242ac130003 Generar

Nombre:

Agente 1

Sistema:

Sistema 1

Activo:

Activo

#	Nombre	Expresión regular	Nivel	
0	Tiempos elevados	.*[0-9]+s	MAJOR	Ver Eliminar
1	No se encuentra elemento	(NoSuchElementException){1}	MINOR	Ver Eliminar

Guardar

**Ilustración 59: Edición de agente 1**

Al agregar una nueva regla, se pueden establecer los datos básicos y, a medida que se escribe la expresión regular, el sistema filtra el último log recibido. De esta forma se indica a la persona que está creando la regla si existe log que coincide con la expresión:

Proyecto de monitorización de s... x +

No es seguro | vps-1f79b26a.vps.ovh.net:8080/admin/agents/edit/0bac5204-4951-11ec-81d3-0242ac130003

Aplicaciones TFG

Home Us... edu] :: Salir

## Edición de regla

Nombre:

Nueva regla

Activo:

Activo

Regexp:

\*

24/12/2021 20:41:37 - La ejecución ha tardado 3s  
 24/12/2021 20:41:37 - NoSuchElementException  
 24/12/2021 20:41:36 - La ejecución ha tardado 6s  
 24/12/2021 20:41:36 - El sistema no responde  
 24/12/2021 20:41:35 - La ejecución ha tardado 3s  
 24/12/2021 20:41:35 - Password incorrecta  
 24/12/2021 20:41:35 - La ejecución ha tardado 2s  
 24/12/2021 20:41:35 - NoSuchElementException  
 24/12/2021 20:41:35 - La ejecución ha tardado 2s  
 24/12/2021 20:41:35 - Password incorrecta

Tipo de cálculo:

Valor directo

Tipo de coincidencia:

Todas

Nivel:

Baja

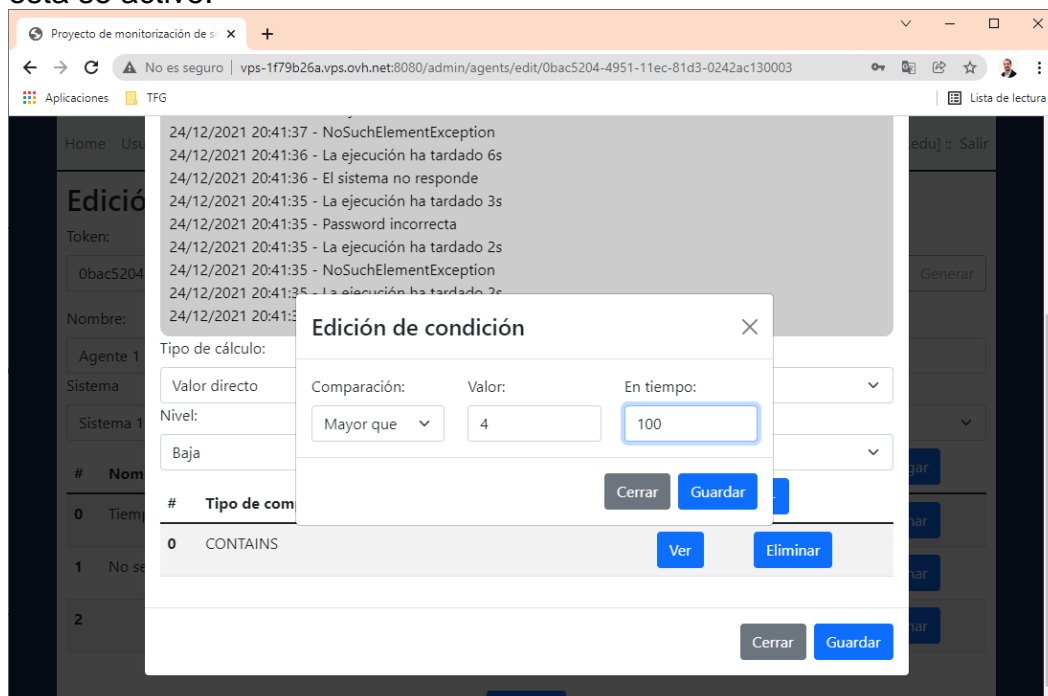
#	Tipo de comparación	Valor	Tiempo
0	Tiem...		
1	No se...		
2			

+

**Ilustración 60: Edición de regla de agente**

Además, se permite elegir el tipo de cálculo a realizar (conteo o valor directo a partir de la expresión regular), si deben coincidir todas las condiciones o alguna y el nivel de criticidad de la regla.

Finalmente, se pueden agregar una o más condiciones a la regla para que esta se active:



**Ilustración 61: Edición de condición de una regla**

### Perfil de soporte

Si el usuario introducido en la pantalla de acceso tiene un perfil de soporte, se accede a un sistema de monitorización de sistemas. En la barra superior se pueden seleccionar ninguno o N sistemas a monitorizar y el periodo de tiempo que se desea. La pantalla mostrará la clasificación de severidad de las incidencias recibidas por parte de los agentes según las reglas creadas y una consola para ver en tiempo real los eventos.

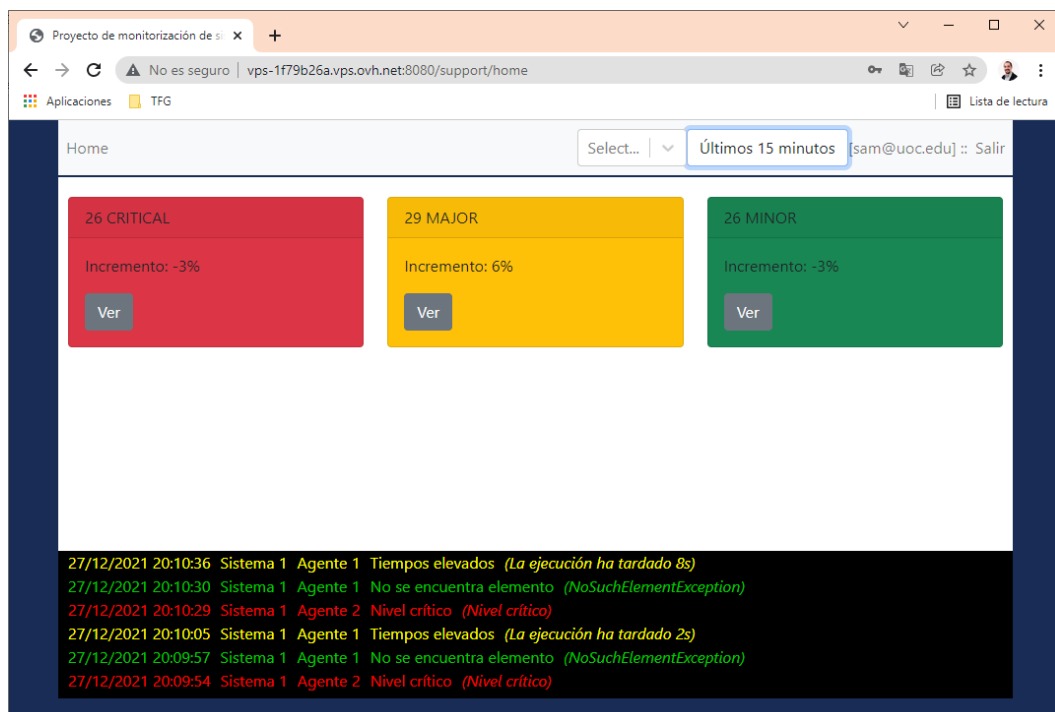


Ilustración 62: Pantalla principal con perfil de soporte

También es posible visualizar los eventos generados de un tipo de severidad en concreto y, si se desea, se puede observar estos eventos en una gráfica:

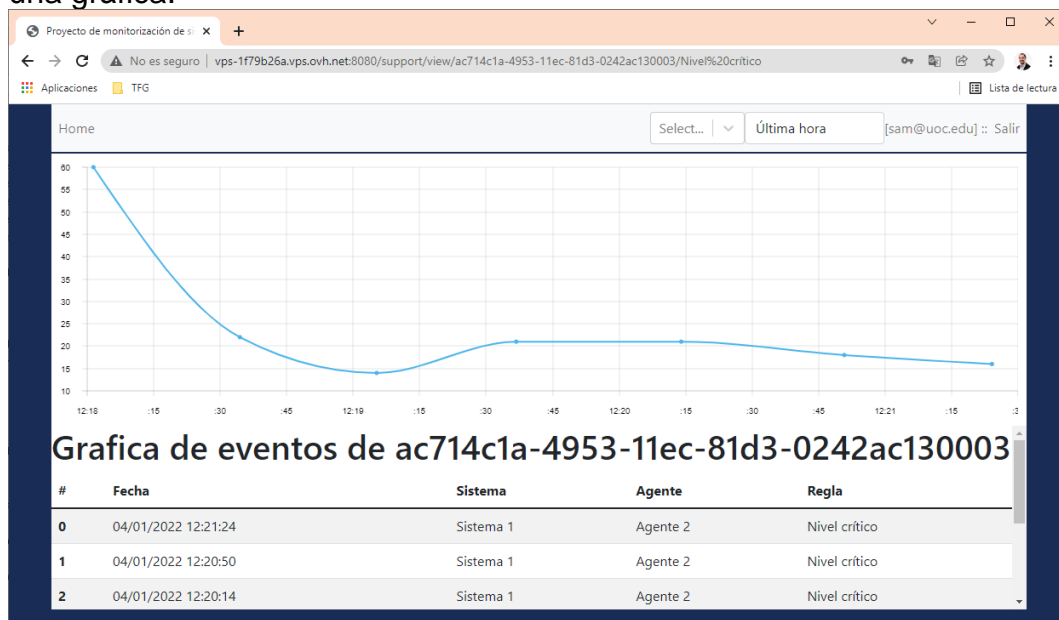


Ilustración 63: Gráfica de eventos con perfil de soporte

## 2.8 Pruebas

El sistema dispone de un conjunto de pruebas automáticas que validan el código fuente y su cobertura en cada compilación. También se han realizado una serie de pruebas manuales para verificar que se han cumplido con los requisitos de los casos de uso definidos.

## Pruebas manuales

El resultado de las pruebas realizadas son 37 pruebas correctas de un total de 41. Esto representa un 92.5% de resultado satisfactorio.

### *CUG001 Seguridad en intercambio de mensajes*

Pruebas satisfactorias: 1/2

N.º	Funcionalidad	Resultado
TCUG01	El token generado es válido según y cumple sistema de firma JWT.	OK
TCUG02	Token se genera con una caducidad de 60 segundos.	KO

### *CUA001 Appender Log4j*

Pruebas satisfactorias: 3/3

N.º	Funcionalidad	Resultado
TCUA01	El <i>appender</i> envía el log al microservicio	OK
TCUA02	El <i>appender</i> guarda el log si no existe conexión con el microservicio	OK
TCUA03	El <i>appender</i> no guarda log si el agente está deshabilitado.	OK

### *CUM0001 Servicio de autenticación*

Pruebas satisfactorias: 2/2

N.º	Funcionalidad	Resultado
TCUM0101	Se intenta autenticar un agente activo y el sistema genera un token correcto	OK
TCUM0102	Se intenta autenticar un agente inactivo y el sistema devuelve un error.	OK

### *CUM0002 Guardar línea/s log para un token válido de agente*

Pruebas satisfactorias: 2/2

N.º	Funcionalidad	Resultado
TCUM0201	Se recibe una línea de log de un agente y coincide con una expresión regular y se genera un evento.	OK

TCUM0202	Se recibe una línea de log de un agente, no coincide con una expresión regular y sólo se guarda.	OK
----------	--	----

*CUW0001 Servicio de autenticación*

Pruebas satisfactorias: 4/5

N.º	Funcionalidad	Resultado
TCUW0101	Se introduce email/clave de usuario activo en el sistema con perfil de administrador y se accede al panel de administración.	OK
TCUW0102	Se introduce email/clave de usuario activo en el sistema con perfil de soporte y se accede al panel de administración.	OK
TCUW0103	Se introduce unas credenciales incorrectas y el sistema devuelve un error	OK
TCUW0104	Se introduce unas credenciales correctas de un usuario inactivo y el sistema devuelve un error.	OK
TCUW0105	Se redirige automáticamente al módulo correspondiente según su perfil estando ya autenticado.	KO

*CUW0002 Listado de últimos agentes notificados*

Pruebas satisfactorias: 1/1

N.º	Funcionalidad	Resultado
TCUW0201	Visualizar últimos agentes notificados ordenados por fecha de notificación.	OK

*CUW0003 Listado filtrado de usuarios creados*

Pruebas satisfactorias: 3/3

N.º	Funcionalidad	Resultado
TCUW0301	Filtrar usuarios	OK
TCUW0302	Crear nuevo usuario desde el listado.	OK
TCUW0303	Visualizar usuario desde el listado.	OK

*CUW0004 Alta y modificación de usuario*

Pruebas satisfactorias: 2/2

N.º	Funcionalidad	Resultado
TCUW0401	Se valida que el email introducido que no exista y que sea válido.	OK
TCUW0402	Se valida que todos los datos sean obligatorios.	OK

*CUW0005 Listado filtrado sistemas creados*

Pruebas satisfactorias: 3/3

N.º	Funcionalidad	Resultado
TCUW0501	Filtrar sistemas.	OK
TCUW0502	Crear nuevo sistema desde el listado.	OK
TCUW0503	Visualizar sistema desde el listado.	OK

*CUW0006 Alta y modificación sistema*

Pruebas satisfactorias: 1/2

N.º	Funcionalidad	Resultado
TCUW0601	Se valida que todos los datos sean obligatorios.	KO. Si no se rellena país, error interno.
TCUW0602	Se guardan los datos del sistema si están todos cubiertos.	OK

*CUW0007 Listado agentes creados*

Pruebas satisfactorias: 3/3

N.º	Funcionalidad	Resultado
TCUW0701	Filtrar agentes.	OK
TCUW0702	Crear nuevo agente desde el listado.	OK
TCUW0703	Visualizar agente desde el listado.	OK

*CUW0008 Alta y modificación de agente*

Pruebas satisfactorias: 3/4

N.º	Funcionalidad	Resultado
TCUW0801	Se valida que todos los datos sean obligatorios.	KO. Se permite guardar agente sin sistema y sin estado de activo/inactivo
TCUW0802	Se permite agregar nueva regla	OK
TCUW0803	Se permite eliminar regla	OK
TCUW0804	Se permite guardar agente con datos cubiertos.	OK

*CUW0009 Alta y modificación de regla de agente*

Pruebas satisfactorias: 5/5

N.º	Funcionalidad	Resultado
TCUW0901	Se valida que todos los datos sean obligatorios.	OK
TCUW0902	A medida que se escribe expresión regular se muestra el log.	OK
TCUW0903	Se permite agregar condición nueva.	OK
TCUW0904	Se permite eliminar condición existente	OK
TCUW0905	Se permite modificar condición existente	OK

*CUW0010 Monitorización de agentes*

Pruebas satisfactorias: 3/3

N.º	Funcionalidad	Resultado
TCUW1001	Se muestra pantalla con los datos relevantes de los eventos según filtro de cabecera.	OK
TCUW1002	Al actualizar el filtro de cabecera se refrescan los datos.	OK
TCUW1003	Se pueden visualizar los eventos según su nivel de criticidad.	OK

*CUW0011 Detalle de regla de agente*

Pruebas satisfactorias: 1/1

N.º	Funcionalidad	Resultado
TCUW1101	Se visualiza gráfica con los datos de un tipo de evento de una regla en concreto de un agente.	OK

## Pruebas automatizadas sobre código Java

Cada microservicio dispone de un sistema automatizado de pruebas y generación de cobertura con el plugin JaCoCo [35]. La cobertura es independiente por cada módulo.

### Módulo common

2 test ejecutados con una cobertura del 1% sobre 1788 líneas de código. Este módulo es incluido como dependencia por el resto, y el porcentaje sólo representa a las pruebas ejecutadas. Si el informe fuese generado en combinación de todas las pruebas de todos los módulos incluyendo esta dependencia, sería mucho más elevada.

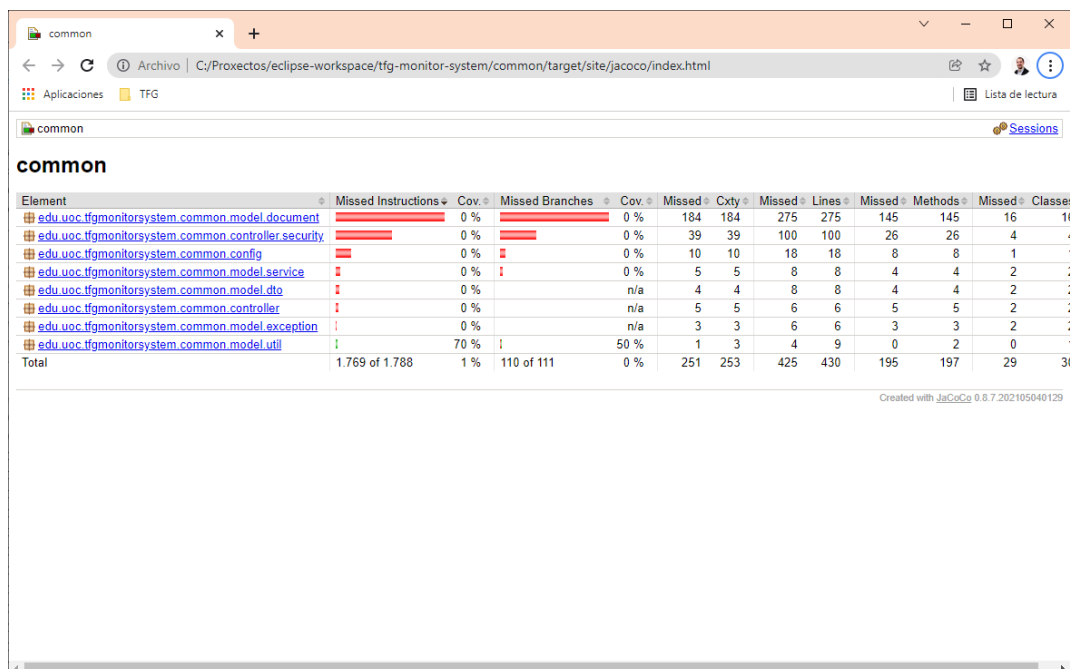
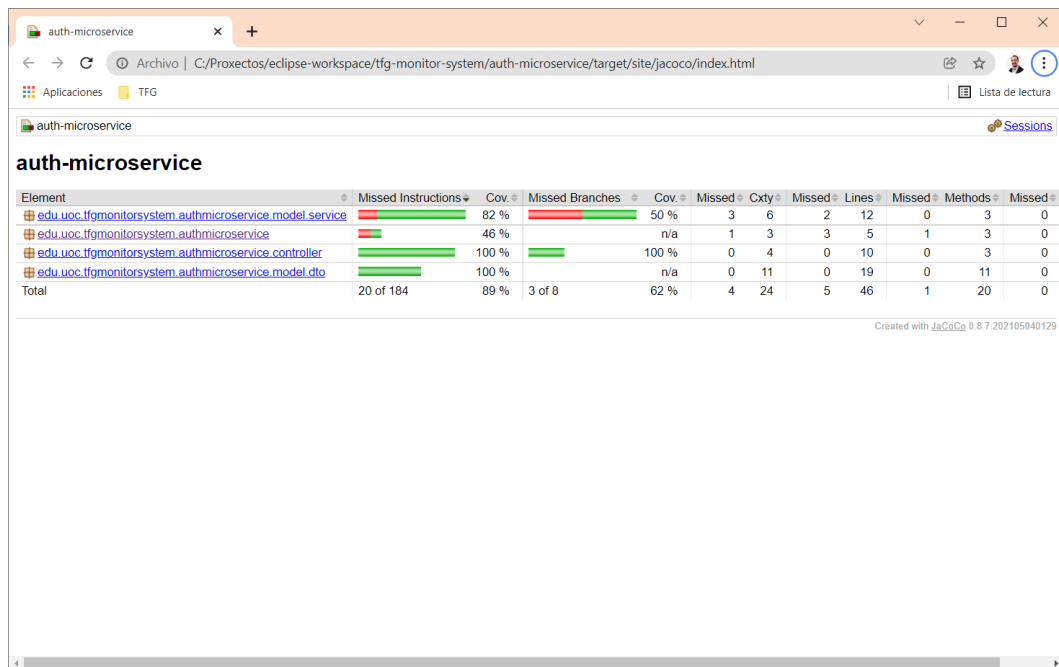


Ilustración 64: Cobertura de módulo common

### Módulo microservicio de autenticación

2 test ejecutados con una cobertura del 89% sobre 184 líneas de código.

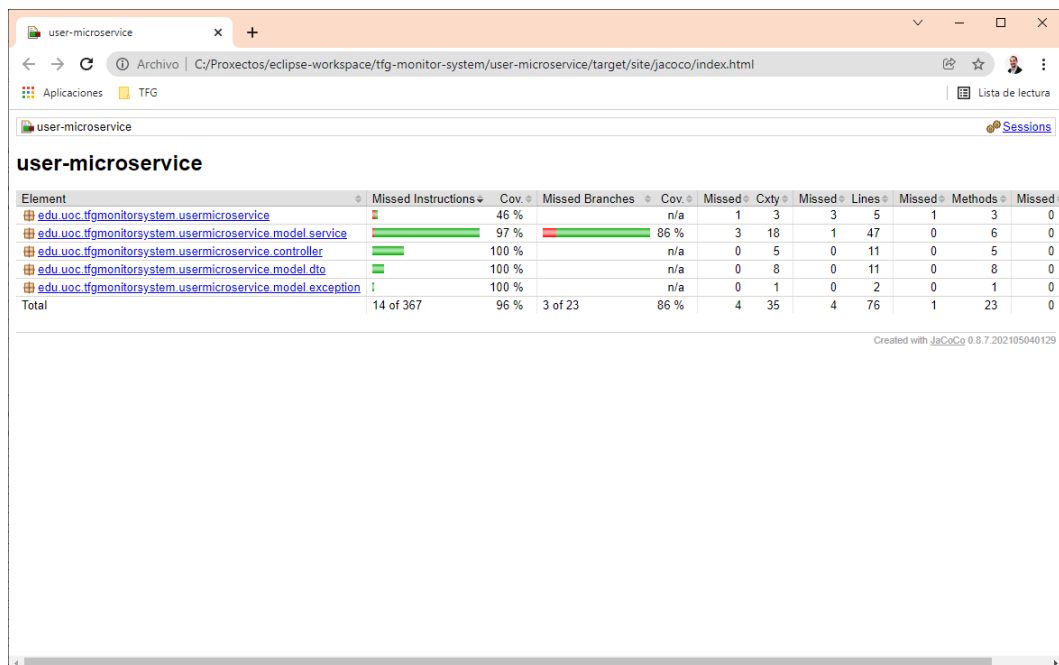




**Ilustración 65: Cobertura de módulo de microservicio de autenticación**

### *Módulo microservicio de usuarios*

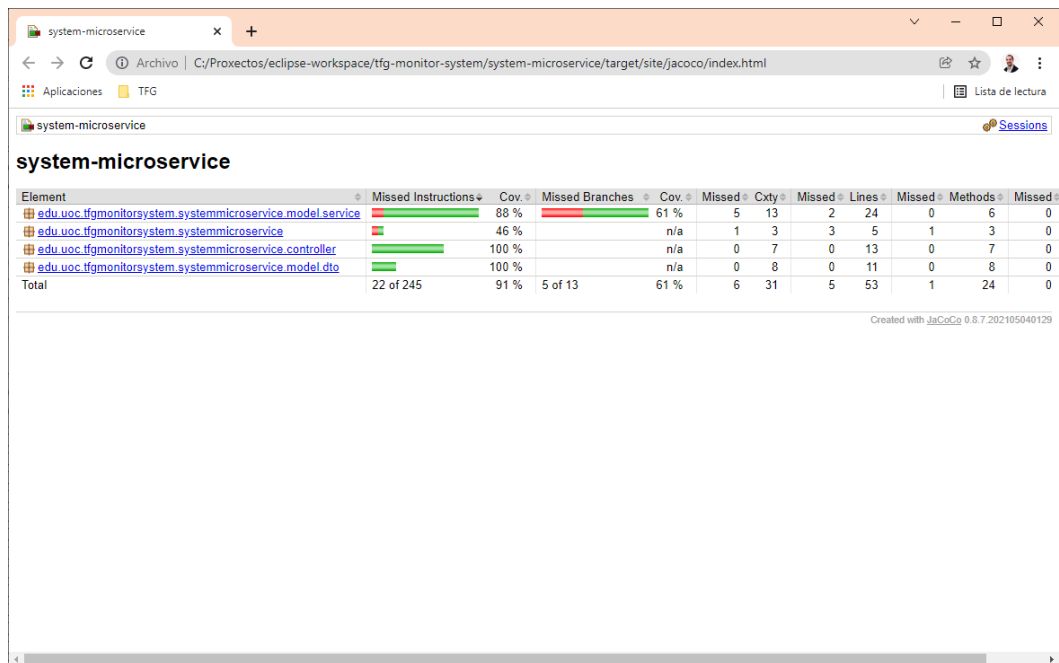
4 test ejecutados con una cobertura del 96% sobre 367 líneas de código.



**Ilustración 66: Cobertura de módulo de microservicio de usuarios**

### *Módulo microservicio de sistemas*

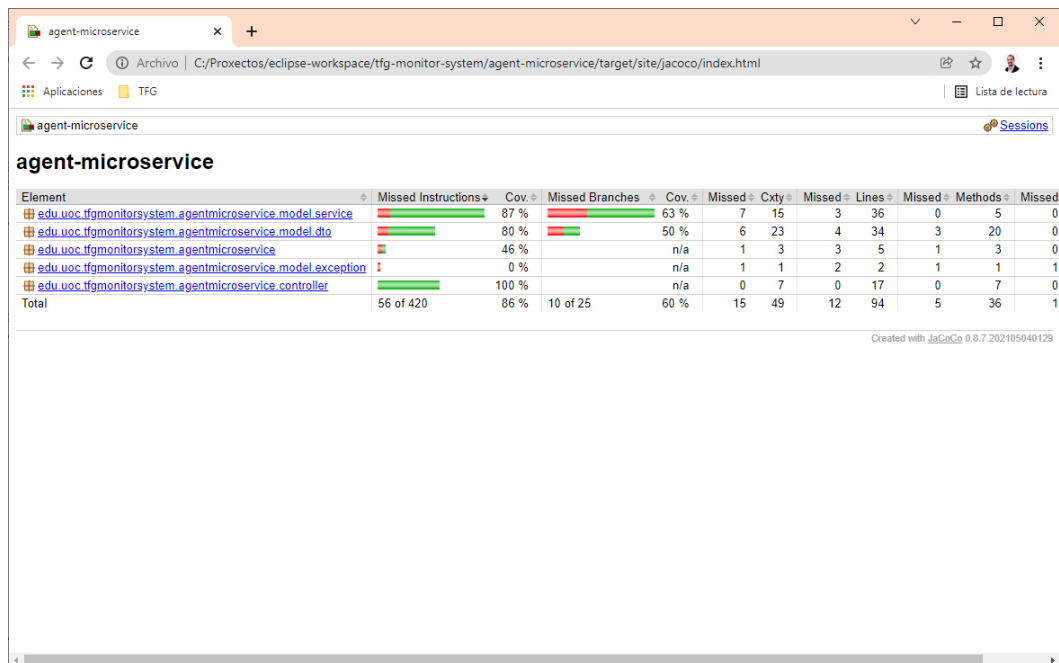
4 test ejecutados con una cobertura del 91% sobre 245 líneas de código.



**Ilustración 67: Cobertura de módulo de microservicio de sistemas**

### *Módulo microservicio de agentes*

5 test ejecutados con una cobertura del 86% sobre 420 líneas



**Ilustración 68: Cobertura de módulo de microservicio de agentes**

### *Módulo microservicio de log*

5 test ejecutados con una cobertura del 71% sobre 1172 líneas

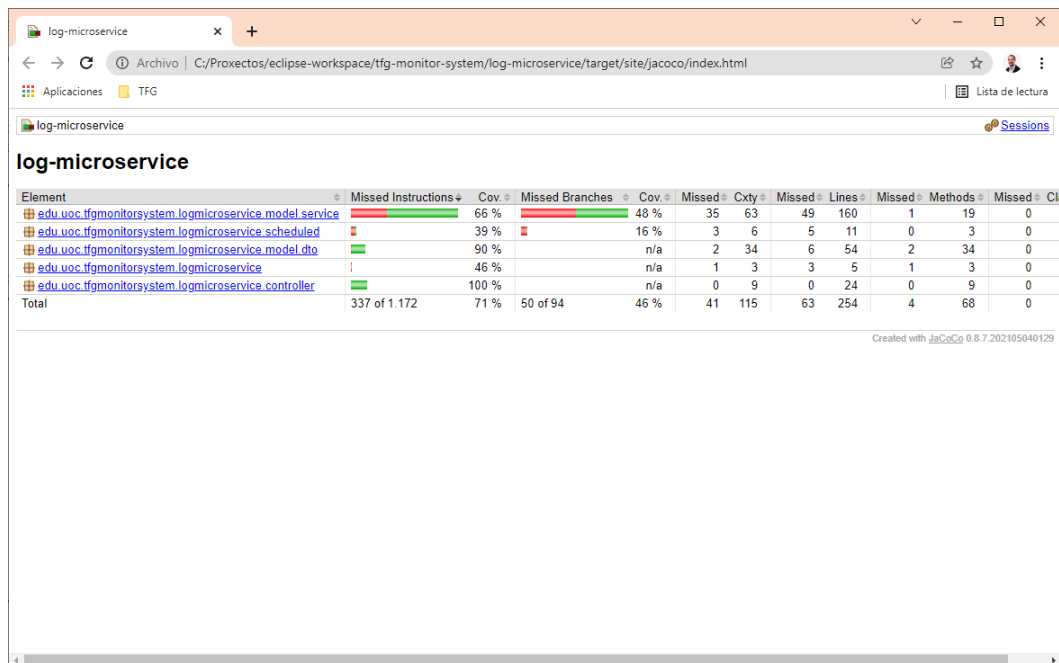


Ilustración 69: Cobertura de módulo de microservicio de log

### 3. Conclusiones

El desarrollo de este proyecto ha supuesto un paso más en la formación del grado de ingeniería de software al tener que adaptar y aplicar los conocimientos que se han adquirido a lo largo del grado como han sido entre otros: diseño de software, aplicaciones distribuidas, patrones de programación o gestión de proyectos. Estos conocimientos han sido necesarios para construir un aplicativo que cumpliera el objetivo principal de ser altamente flexible, escalable, tolerante a fallos y fácil de mantener además de incluir alta cohesión con bajo acoplamiento entre sus diferentes componentes.

Los objetivos principales se han conseguido gracias a la planificación previa, usar la metodología en cascada e incluir metodologías ágiles en la fase de desarrollo. Gracias a esto último, y el uso de las retrospectivas en las iteraciones se ha conseguido corregir parte del diseño y análisis realizado previamente. Sobre todo, estas correcciones, han sido debidas al uso de tecnologías nuevas y sobre las que no se tenía el suficiente conocimiento.

Además, el uso de *Git Flow* ha facilitado la consulta de código y su comparación con versiones estables cuando se realizaba algún cambio importante y el sistema dejaba de funcionar. También el uso de Docker ha sido muy importante, dado que se podía disponer en el equipo local de desarrollo de una base de datos como MongoDB y de la posibilidad de probar el despliegue contenerizado del aplicativo. Es en este punto un ejemplo de cuando se han tenido que aplicar correcciones en el diseño y análisis previos, al realizar las pruebas de despliegue en una de las iteraciones, se ha tenido que incluir una dependencia en los

microservicios al componente *common* para poder definir el sistema *healthy* de Docker.

Otra cuestión que ha ayudado a cumplir los objetivos marcados ha sido el uso de Java como lenguaje de programación, el framework Spring Boot y la herramienta Maven. Gracias a Spring Data se ha podido codificar rápidamente el sistema de información para MongoDB y desarrollar los cinco microservicios: agente, autenticación, sistemas, usuarios y log. Maven ha aportado facilidad y una estandarización a la hora de compilar, empaquetar y construir cada una de las partes del sistema.

Por último y como líneas de trabajo futuras, se deben realizar varias modificaciones. Por un lado, se detecta que para cambiar de MongoDB a otro NoSQL se tendrá que adaptar todos los microservicios y la dependencia de *common*. También se necesita mejorar la seguridad de los sistemas contenerizados y la documentación del API Rest. El siguiente listado contiene las tareas pendientes a realizar:

- Desacoplar Spring Data de cada microservicio creando un sistema de persistencia en el módulo *common*
- Gestionar datos sensibles como son las claves de generación de Tokens usando o las claves de acceso a la NoSQL, por ejemplo, Docker-secrets [36].
- Documentación de microservicios con Swagger [37]

## 4. Glosario

**API:** Application Programming Interface, es un conjunto de definiciones y protocolos para permitir la comunicación entre aplicativos.

**Appender:** Es la parte del programa encargada de hacer llegar los eventos de log a un destino.

**Cloud:** Hace referencia a recursos de computación en Internet repartidos por múltiples localizaciones con un sistema de gestión manejado por el usuario mediante el uso de un API.

**Frontend:** Hace referencia a la capa de visible y con la que interactúa una persona, más concretamente en este proyecto la página web.

**Healthy:** Estado de un contenedor de Docker indicando que está realizando el servicio para el que ha sido arrancando de manera correcta.

**Log:** Se refiere a la información que un aplicativo o sistema produce cuando se ejecuta su código fuente y sirve para analizar si su funcionamiento es el correcto.

**Merge:** Hace referencia a llevar el código de una rama o “branch” hacia otra realizando una mezcla o unión del código nuevo/modificado con el existente.

**Plugin:** Aplicativo que se incluye en otro programa para agregar una nueva funcionalidad.

**PR:** Pull Request, término usado para realizar una petición de integrar las propuestas de cambio de código o nuevo código a una rama del repositorio.

**Rest:** Representational State Transfer es un estilo de arquitectura de software se puede decir que es un sistema para describir una API que usa el protocolo HTTP para su comunicación.

**Snapshot:** Versión en desarrollo de un programa.

## 5. Bibliografía

- [1] «Wikipedia: Desarrollo en cascada,» [En línea]. Available: [https://es.wikipedia.org/wiki/Desarrollo\\_en\\_cascada](https://es.wikipedia.org/wiki/Desarrollo_en_cascada). [Último acceso: 18 12 2021].
- [2] «Github Flow,» [En línea]. Available: <https://docs.github.com/en/get-started/quickstart/github-flow>. [Último acceso: 18 12 2021].
- [3] «Spring Boot,» [En línea]. Available: <https://spring.io/projects/spring-boot>. [Último acceso: 24 12 2021].
- [4] «Maven,» [En línea]. Available: <https://maven.apache.org/>. [Último acceso: 24 12 2021].
- [5] «Docker.com ¿Que es un contenedor?,» [En línea]. Available: <https://www.docker.com/resources/what-container>. [Último acceso: 18 12 2021].
- [6] «Expresiones regulares,» [En línea]. Available: [https://es.wikipedia.org/wiki/Expresi%C3%B3n\\_regular](https://es.wikipedia.org/wiki/Expresi%C3%B3n_regular). [Último acceso: 24 12 2021].
- [7] «Wikipedia: Rest,» [En línea]. Available: [https://es.wikipedia.org/wiki/Transferencia\\_de\\_Estado\\_Representacional](https://es.wikipedia.org/wiki/Transferencia_de_Estado_Representacional). [Último acceso: 19 12 2021].
- [8] «React,» [En línea]. Available: <https://reactjs.org/>. [Último acceso: 24 12 2021].
- [9] «React vs Angular comparison,» [En línea]. Available: <https://pagepro.co/blog/react-vs-angular-comparison/>. [Último acceso: 24 12 2021].
- [10] «MongoDB,» [En línea]. Available: <https://www.mongodb.com/>. [Último acceso: 24 12 2021].
- [11] «Log4j,» [En línea]. Available: <https://logging.apache.org/log4j/2.x/>. [Último acceso: 24 12 2021].
- [12] «JWT,» [En línea]. Available: <https://jwt.io/>. [Último acceso: 24 12 2021].
- [13] «Justmind,» [En línea]. Available: <https://www.justinmind.com/>. [Último acceso: 24 12 2021].
- [14] «DrawIO,» [En línea]. Available: <https://app.diagrams.net/>. [Último acceso: 24 12 2021].
- [15] «GitHub,» [En línea]. Available: <https://github.com/>. [Último acceso: 24 12 2021].
- [16] «Eclipse,» [En línea]. Available: <https://www.eclipse.org/>. [Último acceso: 24 12 2021].
- [17] «Fabric8,» [En línea]. Available: <https://fabric8.io/>. [Último acceso: 24 12 2021].
- [18] «Visual Studio Code,» [En línea]. Available: <https://code.visualstudio.com/>. [Último acceso: 24 12 2021].
- [19] «JSON Web Token RFC7519,» 2015. [En línea]. Available: <https://datatracker.ietf.org/doc/html/rfc7519>.

- [20] «Status code 200,» [En línea]. Available: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/200>. [Último acceso: 24 12 2021].
- [21] «Satus code 401,» [En línea]. Available: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/401>. [Último acceso: 24 12 2021].
- [22] «Arquitectura microservicios, Wikipedia,» [En línea]. Available: [https://es.wikipedia.org/wiki/Arquitectura\\_de\\_microservicios](https://es.wikipedia.org/wiki/Arquitectura_de_microservicios). [Último acceso: 24 12 2021].
- [23] «Escalabilidad horizontal, Wikipedia,» [En línea]. Available: <https://es.wikipedia.org/wiki/Escalabilidad>. [Último acceso: 24 12 2021].
- [24] «Spring Data MongoDB,» [En línea]. Available: <https://spring.io/projects/spring-data-mongodb>. [Último acceso: 24 12 2021].
- [25] «Yarn,» [En línea]. Available: <https://yarnpkg.com/>. [Último acceso: 18 12 2021].
- [26] «NoSQL (Wikipedia),» [En línea]. Available: <https://en.wikipedia.org/wiki/NoSQL>. [Último acceso: 24 12 2021].
- [27] «Document (MongoDB),» [En línea]. Available: <https://docs.mongodb.com/manual/core/document/>. [Último acceso: 24 12 2021].
- [28] «Collections (MongoDB),» [En línea]. Available: <https://docs.mongodb.com/manual/reference/glossary/#std-term-collection>. [Último acceso: 24 12 2021].
- [29] «Wikipedia: Integración continua,» [En línea]. Available: [https://es.wikipedia.org/wiki/Integraci%C3%B3n\\_continua](https://es.wikipedia.org/wiki/Integraci%C3%B3n_continua). [Último acceso: 24 12 2021].
- [30] «Wikipedia: Yaml,» [En línea]. Available: <https://es.wikipedia.org/wiki/YAML>. [Último acceso: 24 12 2021].
- [31] «Docker-compose,» [En línea]. Available: <https://docs.docker.com/compose/>. [Último acceso: 24 12 2021].
- [32] «Mongo-express,» [En línea]. Available: <https://github.com/mongo-express/mongo-express>. [Último acceso: 24 12 2021].
- [33] «Nodejs,» [En línea]. Available: <https://nodejs.org/en/>. [Último acceso: 24 12 2021].
- [34] «Postman,» [En línea]. Available: <https://www.postman.com/>. [Último acceso: 24 12 2021].
- [35] «JaCoCo,» [En línea]. Available: <https://www.eclemma.org/jacoco/>. [Último acceso: 24 12 2021].
- [36] «Docker secrets,» [En línea]. Available: <https://docs.docker.com/engine/swarm/secrets/>. [Último acceso: 03 01 2022].
- [37] «Wikipedia: Swagger,» [En línea]. Available: [https://en.wikipedia.org/wiki/Swagger\\_\(software\)](https://en.wikipedia.org/wiki/Swagger_(software)). [Último acceso: 03 01 2022].

## 6. Anexos

### 6.1 Manual de instalación de aplicativo

Se han publicado las imágenes Docker-hub de los microservicios de la versión 1.2 ubicada en el repositorio público de GitHub <https://github.com/aperezgua/tfg-monitor-system>. Estas imágenes están pensadas para funcionar en un equipo local bajo la URL <http://localhost:8080>

Instalación de Docker-desktop

En máquinas con sistema operativo Windows, instalar Docker-desktop desde la página web <https://www.docker.com/products/docker-desktop>

A continuación, en una carpeta del sistema operativo se deben crear los siguientes dos ficheros y contenido:

Configuración de Docker-compose

#### **aperezgua-tfg-monitor-system.yml**

Se puede descargar formateado y listo para usar desde la siguiente ubicación: <https://raw.githubusercontent.com/aperezgua/tfg-monitor-system/main/docker/aperezgua-tfg-monitor-system.yml>

```
version: '3.3'
```

```
services:
```

```
  mongo:
```

```
    image: mongo
```

```
    restart: always
```

```
    networks:
```

```
      - tfg_network
```

```
    environment:
```

```
      MONGO_INITDB_ROOT_USERNAME: root
```

```
      MONGO_INITDB_ROOT_PASSWORD: toor
```

```
      MONGO_INITDB_DATABASE : tfg-monitor-system-db
```

```
    volumes:
```

```
      - mongo-vol:/data
```

```
      - mongo-vol:/data/configdb
```

```
      - mongo-vol:/data/db
```

```
      - ./mongo-init.js:/docker-entrypoint-initdb.d/mongo-init.js:ro
```

```
    healthcheck:
```

```
      test: ["CMD-SHELL", "echo 'db.runCommand(\"ping\").ok' | mongo Localhost:27017/test --quiet"]
```

```
      interval: 10s
```

```
      timeout: 10s
```

```
      retries: 5
```

```
  auth-microservice:
```

```
    image: aperezgua/auth-microservice:1.2
```

```
    restart: always
```

```
    networks:
```



```

    - tfg_network
  ports:
    - 8091:8091
  depends_on:
    - mongo
  healthcheck:
    test: ["CMD-SHELL", "curl -f http://localhost:8091/health ||
exit 1"]
    interval: 10s
    timeout: 10s
    retries: 5

user-microservice:
  image: aperezgua/user-microservice:1.2
  restart: always
  networks:
    - tfg_network
  ports:
    - 8092:8092
  depends_on:
    - mongo
  healthcheck:
    test: ["CMD-SHELL", "curl -f http://localhost:8092/health ||
exit 1"]
    interval: 10s
    timeout: 10s
    retries: 5

system-microservice:
  image: aperezgua/system-microservice:1.2
  restart: always
  networks:
    - tfg_network
  ports:
    - 8093:8093
  depends_on:
    - mongo
  healthcheck:
    test: ["CMD-SHELL", "curl -f http://localhost:8093/health ||
exit 1"]
    interval: 10s
    timeout: 10s
    retries: 5

agent-microservice:
  image: aperezgua/agent-microservice:1.2
  restart: always
  networks:
    - tfg_network
  ports:
    - 8094:8094
  depends_on:
    - mongo
  healthcheck:
    test: ["CMD-SHELL", "curl -f http://localhost:8094/health ||
exit 1"]
    interval: 10s

```

```
timeout: 10s
retries: 5
```

```
Log-microservice:
  image: aperezgua/Log-microservice:1.2
  restart: always
  networks:
    - tfg_network
  ports:
    - 8095:8095
  depends_on:
    - mongo
  healthcheck:
    test: ["CMD-SHELL", "curl -f http://localhost:8095/health ||
exit 1"]
    interval: 10s
    timeout: 10s
    retries: 5
```

```
webapp:
  image: aperezgua/webapp:1.2
  restart: always
  networks:
    - tfg_network
  ports:
    - 8080:8080
  healthcheck:
    test: ["CMD-SHELL", "curl -f http://localhost:8080/health ||
exit 1"]
    interval: 10s
    timeout: 10s
    retries: 5
```

```
Log4j-appender:
  image: aperezgua/Log4j-appender:1.2
  restart: always
  networks:
    - tfg_network
  depends_on:
    - log-microservice
```

```
networks:
  tfg_network:
    driver: bridge
```

```
volumes:
  mongo-vol:
```

### **mongo-init.js**

Se puede descargar de <https://raw.githubusercontent.com/aperezgua/tfg-monitor-system/main/docker/mongo-init.js>

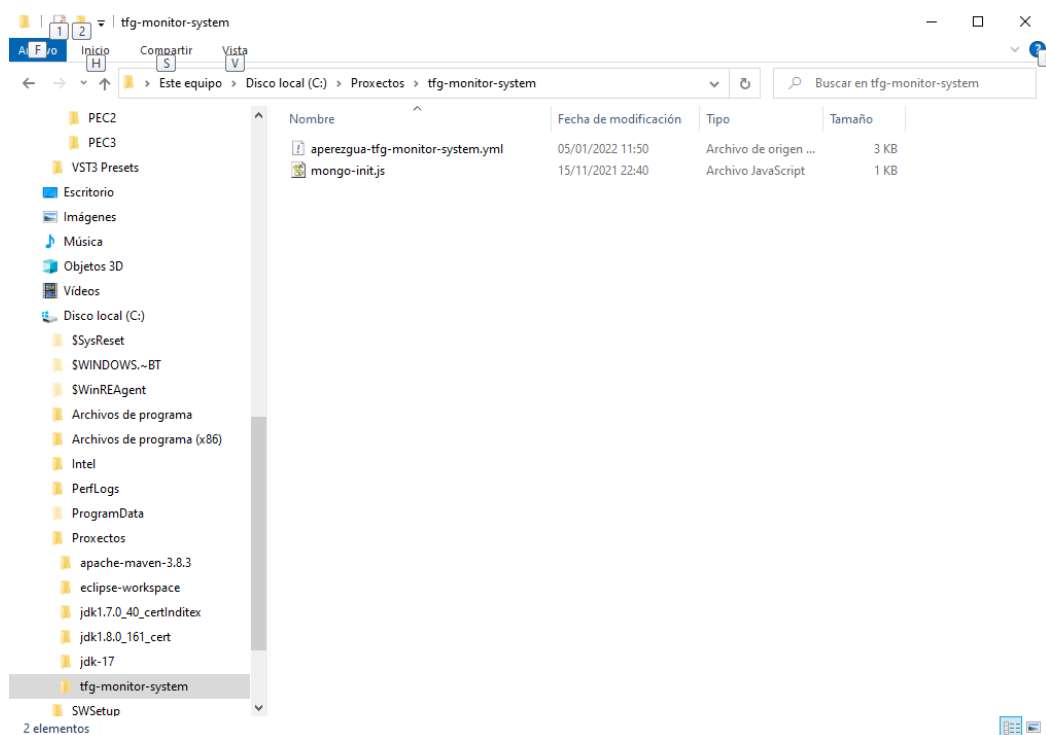
```
db.createUser(
```

```

    {
      user: "tfg-monitor-system-user",
      pwd: "pw",
      roles: [
        {
          role: "readWrite",
          db: "tfg-monitor-system-db"
        }
      ]
    }
  ]
};

```

El resultado final debe ser el siguiente, dos ficheros uno con la configuración de Docker-compose y otro con la configuración de MongoDB:



### Arranque de microservicios con Docker-compose

Lo siguiente es abrir una línea de comandos y ejecutar la siguiente instrucción:

```
docker-compose -p tfg-monitor-system -f aperezgua-tfg-monitor-system.yml up
```

El Sistema comenzará a desplegar en el Docker local todos los microservicios y la base de datos MongoDB:

```
C:\Windows\System32\cmd.exe - docker-compose -p tfg-monitor-system -f aperezgua-tfg-monitor-system.yml up

C:\Proyectos\tfg-monitor-system>docker-compose -p tfg-monitor-system -f aperezgua-tfg-monitor-system.yml up
Creating network "tfg-monitor-system_tfg_network" with driver "bridge"
Pulling mongo (mongo)...
latest: Pulling from library/mongo
7b1a6ab2e44d: Pull complete
90eb44ebc60b: Pull complete
5085b59f2efb: Pull complete
c7499923d022: Pull complete
019496b6c44a: Pull complete
c0df4f407f69: Pull complete
351daa315b6c: Pull complete
a233e6240acc: Pull complete
a3f57d6be64f: Pull complete
dd1b5b345323: Pull complete
Digest: sha256:5be752bc5f2ac4182252d0f15d74df080923aba39700905cb26d9f70f39e9702
Status: Downloaded newer image for mongo:latest
Pulling auth-microservice (aperezgua/auth-microservice:1.2)...
1.2: Pulling from aperezgua/auth-microservice
7448db3b31eb: Downloading [=====] 6.961MB/52.47MB
c36e04fa7939: Downloading [=====] 9.661MB/19.19MB
29e8ef0a3340: Downloading [=====] 11.9MB/43.16MB
a0c934d2565d: Waiting
a360a17c9cab: Waiting
cfcc996af805: Waiting
2cf014724202: Waiting
4bc402a00dfe: Waiting
ec95f0a29742: Waiting
```

```
C:\Windows\System32\cmd.exe - docker-compose -p tfg-monitor-system -f aperezgua-tfg-monitor-system.yml up

3vsMtw
log-microservice_1 | user-agent: Java/1.8.0_111
log-microservice_1 | host: log-microservice:8095
log-microservice_1 | accept: text/html, image/gif, image/jpeg, *, q=.2, */*; q=.2
log-microservice_1 | connection: keep-alive
log-microservice_1 | content-length: 27
log-microservice_1 |
log-microservice_1 | ## PARAMETERS:
log-microservice_1 |
log-microservice_1 | 2022-01-05 11:16:49 DEBUG LogController:80 - lineLog-La ejecución ha tardado 5s
log-microservice_1 | 2022-01-05 11:16:49 DEBUG EventLogService:345 - MATCH: La ejecución ha tardado 5s / .*[0-9]+s -> true
log-microservice_1 | 2022-01-05 11:16:49 DEBUG EventLogService:345 - MATCH: La ejecución ha tardado 5s / (NoSuchElementException){1
log-microservice_1 | } -> false
log-microservice_1 | 2022-01-05 11:16:49 DEBUG EventLogService:381 - FULLFILED : []
log-microservice_1 | 2022-01-05 11:16:49 DEBUG EventLogService:384 - CURRENT : [{"agent":{"active":true,"createdDate":"Tue Nov 16 2
1:20:47 UTC 2021","name":"Agente 1","rules":[{"active":true,"agent":null,"calculationType":"DIRECT_VALUE","conditions":[{"comparationType
pe":"AVG_GREATER_THAN","time":30,"value":4}],matchType":"ANY","name":"Tiempos elevados","regularExpression":".*[0-9]+s","severity":"
MAJOR"}],{"active":true,"agent":null,"calculationType":"COUNT_EVENT","conditions":[{"comparationType":"GREATER_THAN","time":30,"value":
2}],matchType":"ANY","name":"No se encuentra elemento","regularExpression":"(NoSuchElementException){1","severity":"MINOR"}],system
s":{"active":true,"country":{"id":1,"name":"Espa\u00f1a"},"createdDate":"Tue Nov 16 21:20:47 UTC 2021","id":10001,"name":"Sistema 1"},"
token":"0bac5204-4951-11ec-81d3-0242ac130003"},"conditionsValues":[{"fullFilled":false,"times":[1641381408217,1641381409373],"values":[
1.0,1.0]}],"date":null,"fullFilled":false,"id":25,"initDate":"Wed Jan 05 11:09:35 UTC 2022","ruleDoubleValue":null,"ruleName":"No se en
cuentra elemento","severity":"MINOR","value":null}, {"agent":{"active":true,"createdDate":"Tue Nov 16 21:20:47 UTC 2021","name":"Agente
1","rules":[{"active":true,"agent":null,"calculationType":"DIRECT_VALUE","conditions":[{"comparationType":"AVG_GREATER_THAN","time":30
,value":4}],matchType":"ANY","name":"Tiempos elevados","regularExpression":".*[0-9]+s","severity":"MAJOR"}],{"active":true,"agent":n
ull,"calculationType":"COUNT_EVENT","conditions":[{"comparationType":"GREATER_THAN","time":30,"value":2}],matchType":"ANY","name":"N
o se encuentra elemento","regularExpression":"(NoSuchElementException){1","severity":"MINOR"}],systems":{"active":true,"country":{"id
":1,"name":"Espa\u00f1a"},"createdDate":"Tue Nov 16 21:20:47 UTC 2021","id":10001,"name":"Sistema 1"},"token":"0bac5204-4951-11ec-81d3-
0242ac130003"},"conditionsValues":[{"fullFilled":false,"times":[1641381409433],"values":[5.0]}],"date":null,"fullFilled":false,"id":27,
"initDate":"Wed Jan 05 11:16:49 UTC 2022","ruleDoubleValue":null,"ruleName":"Tiempos elevados","severity":"MAJOR","value":null}]
log4j-appender_1 | Wed Jan 05 11:16:49 UTC 2022 > Line send correctly: agentTokenId=0bac5204-4951-11ec-81d3-0242ac130003, logLine
=La ejecución ha tardado 5s
log4j-appender_1 | Stop current thread
```

Una vez finalizado el proceso de descarga de las imágenes, el sistema arrancará y será accesible desde nuestro equipo local:

