UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Migración del portal de la Agencia de Salud y Enfermedades de una Comunidad Autónoma

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Ballesteros García, Jorge

Tutor/a: Valderas Aranda, Pedro José

Cotutor/a externo: BARRASA VENTURA, MIREIA

CURSO ACADÉMICO: 2022/2023

Resumen

En el presente trabajo se realizará la migración de la aplicación APPFINAL, perteneciente al sistema informático de la Agencia de Salud y Enfermedades de una Comunidad Autónoma. En concreto, la migración consiste en la unión de las tres aplicaciones en las que se basan los portales de acceso de la Agencia (cada una con una tecnología diferente) en una nueva llamada APPFINAL. Deberá quedar correctamente configurada en una arquitectura de tres capas: interfaces de usuario, servidores de aplicación y base de datos. Además de la creación, instalación, y configuración de la nueva base de datos, el clúster que contiene el servidor de aplicación y el clúster que contiene la parte de las interfaces, instalaremos 'Elasticsearch' para la recopilación de la información para la propia aplicación. A continuación, se realizará un análisis de los componentes críticos tanto de las propias aplicaciones como de las máquinas que las alojan y se configurarán alarmas (usando las aplicaciones Nagios e Introscope) para que desde el departamento de monitorización puedan seguir su estado en todo momento. Finalmente, se llevará a cabo el primer despliegue de la aplicación y se seguirá la validación en el lado de la empresa para comprobar que el despliegue de la aplicación ha sido correcto y la migración ha terminado de forma exitosa.

Palabras clave: migración; sanidad; portales; Elasticsearch; Nagios; Introscope; LiferayDXP; JBoss; PostgreSQL

Abstract

In this work will be analyzed the migration of the APPFINAL application, belonging to the computer system of the Health and Diseases Agency of an Autonomous Community, will be carried out. Specifically, the migration consists of the union of the three applications on which the Agency's access portals are based (each one with a different technology) in a new one called APPFINAL. It must be correctly configured in a three-tier architecture: user interfaces, application servers and database. In addition to the creation, installation, and configuration of the new database, the cluster containing the application server and the cluster containing the interfaces part, we will install 'Elasticsearch' to collect the information for the application itself. Next, an analysis of the critical components of both the applications themselves and the machines that host them will be performed and alarms will be configured (using the Nagios and Introscope applications) so that the monitoring department can follow their status at all times. Finally, the first deployment of the application will be carried out and the validation on the company side will be followed to verify that the deployment of the application has been correct and the migration has been successfully completed.

Keywords: migration; healthcare; portals; Elasticsearch; Nagios; Introscope; Liferay; JBoss; PostgreSQL

Índice

Índice de figuras

Capítulo 1. Introducción

Capítulo 2. Presentación de la empresa

Capítulo 3. Estado del Arte

Capítulo 4. Análisis del problema

Capítulo 5. Diseño de la solución

Capítulo 6. Desarrollo e implantación de la solución

Capítulo 7. Pruebas

Capítulo 8. Conclusiones y trabajos futuros

1. Introducción

La pandemia del COVID-19 supuso un gran reto para la población mundial. Cada persona, sin importar su procedencia, sexo, etnia o nivel sociocultural se vio obligada a adaptarse a la 'nueva normalidad' que implicaba convivir con la enfermedad. Este virus trajo implícito un cambio radical en nuestros hábitos como la obligatoriedad de usar diariamente mascarillas protectoras, el sometimiento de la población a una vacunación masiva y un confinamiento domiciliario que afectó directamente a la salud y a la vida de las personas, llegando incluso a producirse un cambio en el paradigma laboral con la implantación mayoritaria de la modalidad de teletrabajo.

En el ámbito informático, sin duda alguna, supuso un gran reto. El mundo entero era consciente de la situación existente a nivel mundial, ya que los medios de comunicación proporcionaban datos constantemente de la evolución del virus. Las 'olas de la pandemia', el 'pico de contagios' o 'el crecimiento exponencial del virus' eran algunas de las expresiones que ocupaban el día a día de toda la población, desde la gente de a pie hasta políticos o epidemiólogos, responsables finales de la gestión de la enfermedad a nivel de estado. La mayoría de las decisiones tomadas durante ese periodo venían respaldadas por las estimaciones informáticas sobre la posible evolución de los contagios, una herramienta imprescindible, sin duda, en la lucha contra el SARS-CoV-2.

Este ámbito global de la informática se concreta en este estudio en la Agencia de la Salud y las Enfermedades de una Comunidad Autónoma española, que, por temas de confidencialidad, vamos a denominar en el resto de documento "La Comunidad". Estos centros tuvieron que hacer frente a numerosas dificultades a nivel tecnológico y digital, realizando cambios y adaptaciones en sus aplicaciones informáticas (añadiendo medicamentos, vacunas, modificando bases de datos, portales de acceso…) con la finalidad de brindar una información imprescindible para el ciudadano, tratando de reducir al máximo al mismo tiempo cualquier caída de los sistemas informáticos.

Un ejemplo claro de la necesidad de dichas adaptaciones fue las propias páginas web. En unos pocos días, pasaron de recibir unos cientos de visitas diarias con el objetivo de consultar el portal del paciente, ver las últimas noticias o realizar trámites burocráticos, a ser el principal punto de información de la pandemia, la vacunación, los protocolos y todo lo que la población necesitaba conocer. De hecho, cuando las vacunas estuvieron disponibles, por esta página pasaron en pocas semanas en toda España más de 47 millones de usuarios a fin de comprobar que sus datos personales eran correctos y recibir así la citación de vacunación.

Esto supuso un reto tecnológico tremendo, pues las aplicaciones gestionadas por estas Agencias de la Salud y las Enfermedades debieron hacer frente, entre otras muchas cosas, a una gran carga de accesos, peticiones o inserciones de datos que generaron muchas modificaciones en muy poco tiempo. En las bases de datos hubo que modificar los datos de más de 47 millones de usuarios, así como añadir nueva información sobre citas, vacunas, estado, positivos en COVID, etc… lo que originó una gran escalada de estas, para poder ampliarlas y dar soporte a las necesidades sociales, manteniendo además la disponibilidad del servicio, asunto imprescindible ante la gran cantidad de afectados que trataban de buscar ayuda a diario en las páginas web.

Para la realización de todas esas mejoras y el correcto funcionamiento y mantenimiento de las aplicaciones a nivel general, más allá de la pandemia del COVID-19, necesitan apoyo informático de diferentes empresas externas. Prosodie Ibérica S.L.U., en concreto, dedica por completo uno de sus departamentos denominado GSI a la gestión de los sistemas informáticos de una Agencia de la Salud y las Enfermedades, departamento en el que se ubica el presente trabajo final de grado.

Aunque en este departamento se gestionan una gran cantidad de aplicaciones, en este estudio se hará referencia a tres de ellas, a las que denominaremos ***APP1, APP2 y APP3***, utilizando estos nombres con la intención de mantener la confidencialidad de la información y no revelar datos privados a la comunidad educativa. Ante las nuevas necesidades informáticas surgidas en la pandemia y la desactualización de la tecnología que las soporta, desde GSI se detectó la necesidad de migrar todas ellas a una nueva aplicación que llamaremos ***APP FINAL***, un proceso que se encuentra todavía en pleno desarrollo.

Tras la migración, APP FINAL será una aplicación que adquirirá una gran importancia en la estructura de la Agencia de la Salud y las Enfermedades, ya que dentro de la diversidad de servicios que se ofrecen desde estas entidades (gestión de ambulatorios, farmacias, almacenamiento de datos de vacunación, etc.) será responsable de la gestión de los portales y puntos de acceso de los usuarios a la Agencia. En este sentido, cabe señalar que la arquitectura que poseían las aplicaciones APP1, APP2 y APP3 y la antigüedad y desactualización de algunos de los componentes provocaban que los tiempos de respuesta en las búsquedas de los usuarios fueran demasiado altos, empeorando así el funcionamiento y la imagen de la página para estos.

Para la creación y posterior configuración de APP FINAL, es necesario disponer de las nuevas máquinas sobre las que se construirán los frontales, los servidores de aplicación y las bases de datos, que serán proporcionadas por el equipo de AT (Administración de Tecnología) un departamento dentro de Prosodie S.L.U. dedicado a la administración de los equipos. Una vez estén disponibles, se iniciará la instalación y configuración de los componentes, un proceso que se dividirá en cuatro partes principales.

Inicialmente, se creará y configurará correctamente una base de datos PostgreSQL 14 que permitirá el acceso a toda la información almacenada en ella, desde credenciales de usuarios con sus sesiones hasta la propia información dinámica de la página, aprovechando así las mejoras que puede aportar esta versión sobre la anterior PostgreSQL 8.3. Posteriormente, la base de datos se conectará con los servidores de aplicación de tipo JBOSS con Liferay incorporado (software que unifica las máquinas dentro de un mismo contenedor según sus características comunes). Por último, se pondrán en funcionamiento los frontales y se instalará un sistema llamado Elasticsearch para poder proveer al equipo de soporte de la aplicación de información sobre el funcionamiento de su aplicación.

Al ofrecer un servicio indispensable para la sociedad, estas aplicaciones necesitan estar las veinticuatro horas del día en funcionamiento, los trescientos sesenta y cinco días del año. Por ello, en cada una de ellas se especifican las secciones críticas que podrían afectar al servicio, habiendo un equipo de monitorización que se encarga de las posibles alertas que emerjan de cualquiera de los servidores. Por lo tanto, tras la migración, será necesario analizar los posibles apartados críticos para el servicio y configurar correctamente las herramientas para poder reaccionar rápidamente ante un fallo importante. Nótese que ya había una

configuración de alertas en las tres aplicaciones previas (APP1, APP2 y APP3), pero al migrarlas a APP FINAL, será necesaria una revisión y reconfiguración de estas.

Para finalizar la migración, una vez las máquinas estén configuradas con la base de datos, servidores, frontales y el Elasticsearch, se desplegará la nueva aplicación APP FINAL y se comprobará que da servicio adecuadamente. El despliegue se hará inicialmente en dos entornos de prueba, donde se corregirá cualquier error que pudiera producirse, lanzándolo, una vez corregidos los errores, a la fase de “producción". En todo momento, el equipo de soporte de la nueva aplicación comunicará sus hallazgos y reportará los fallos detectados y, tras finalizar el último despliegue, aportará feedback y confirmará el correcto funcionamiento de ***APP FINAL***.

1.1 Motivación

1.1.1 Motivación Personal

Durante mis prácticas en la empresa Prosodie Ibérica S.L.U. he desempeñado varias funciones. Inicialmente, mi trabajo se basaba, entre otras tareas, en la gestión de alertas, la atención al cliente mediante llamada, email o ticketing, y el seguimiento de las incidencias. Gracias a este proyecto he podido profundizar en la labor que desempeña el departamento de GSI, metiéndome de lleno en procesos y desarrollos en los que como CGR (Centro de Gestión Remota) no habría conocido prácticamente.

En este sentido, creo que el presente estudio me ayudará en mi desarrollo personal y profesional en la empresa. Normalmente, cualquier trabajador que empieza en el proyecto relacionado con la Agencia de la Salud y las Enfermedades suele necesitar un tiempo para instruirse, ponerse al día y desarrollar así su trabajo con normalidad. La migración de la aplicación APP FINAL supone un gran reto para mí, que me permitirá aprender todo lo necesario para llevarla a cabo, además de agilizar mi proceso de adaptación laboral e influir positivamente en mi desarrollo personal.

Cuando fui consciente de que desde las tres aplicaciones que dan lugar a la nueva APP FINAL se gestionaron gestionado, en la época más dura de la pandemia, los portales de acceso de las diferentes Agencias de las Comunidades Autónomas, me di cuenta de la gran importancia que tuvo para la sociedad el correcto funcionamiento de estas y de que mediante esta migración podría formar parte de un proyecto muy interesante y útil socialmente. Comprendí la gran importancia que tiene renovar la arquitectura de las máquinas para tratar de evitar los problemas que ocurrieron con la web (problemas por exceso de carga, reinicios continuados...), que se comentarán en puntos posteriores, y por todo ello me sentí atraído rápidamente por el proyecto.

Individualmente, me considero una persona interesada en el bien común y el hecho de pensar que los usuarios de estos portales posiblemente resulten beneficiados, por ejemplo, por una mejora en la fluidez de la web o en la disminución del número de caídas de la página, es algo que me resulta muy atractivo y de sumo interés. Sin embargo, creo que es necesario no

olvidar que es un proyecto que voy a realizar bajo la supervisión de varios compañeros, no siendo mi intención la de atribuirme méritos que no poseo.

Además, desde mi perspectiva como alumno universitario este estudio supone una fuente de motivación. Después de estudiar durante cuatro años de forma teórica muchos de los conceptos sobre los que trabajaré (bases de datos, servidores, frontales…) y realizar algunas simulaciones prácticas en el laboratorio, me parece muy interesante formar parte de un proyecto real. Por ello, trataré de realizarlo con la mayor eficacia posible para lograr que tanto los grupos de soporte de las aplicaciones involucradas como mis supervisores en Prosodie Ibérica S.A. valoren de forma positiva mi trabajo.

Por último, también me gustaría resaltar que hay algunos conceptos y procedimientos que, lógicamente, yo no conocía antes de iniciar este desarrollo. En las clases teóricas recibidas en la UPV aprendí por encima el concepto de despliegue, entre muchos otros, y aquí tendré que hacer un esfuerzo extra para comprender cómo se realiza correctamente y llevarlo a cabo en APP FINAL junto a mis supervisores. Por ello, el reto considero que es mayor y eso me genera más ganas de lograr los objetivos propuestos.

1.1.2 Motivación Profesional

Las tres aplicaciones que finalmente se van a convertir en APP FINAL empleaban diferentes tecnologías, algunas de ellas bastante obsoletas. Un ejemplo de ello es el tipo de servidor de APP1, inicialmente un Apache Tomcat con una versión de 2002, que se encontraba sometido a una elevadísima carga de trabajo a pesar de su antigüedad. Con la inesperada llegada del COVID-19, quedó patente que para las aplicaciones que dan soporte al sistema sanitario, ante el incremento exponencial en la demanda requerida por parte de los usuarios y la carga que acumuló el sistema durante los primeros días de pandemia era necesario renovar la disposición de los equipos y la tecnología que utilizaban.

Como ejemplo, cabe señalar que la aplicación APP1 en sus inicios tenía una sola máquina que hacía las funciones de base de datos, servidor de aplicación y frontal, detectándose ya, tras su puesta en marcha, la necesidad de ampliar la infraestructura, añadiendo más hardware para poder gestionar la aplicación. Esta situación es similar a la actual, ya que, a pesar de poseer más hardware y mejor tecnología y tener una arquitectura de tres capas mejor soportada, desde hace un tiempo se ha detectado la necesidad de crear una nueva aplicación que mejore las anteriores, y las englobe, APP FINAL.

Actualmente, la carga de trabajo es considerablemente menor para esas aplicaciones, ya que disponemos de una situación sociosanitaria bastante mejor que la de hace dos años. Sin embargo, desde el departamento de monitorización diariamente se detectan demasiadas alertas en estas tres aplicaciones, lo que provoca que se realice un mantenimiento sobre ellas demasiado frecuente. En futuros apartados se explicará más en detalle mediante tablas y de forma gráfica esta situación.

1.2 Objetivos

Para lograr marcar los objetivos del proyecto de forma correcta, se considera necesario describir la situación actual de las tres aplicaciones *(*APP1, APP2 y APP3*)* antes de la migración. Todas ellas tienen una arquitectura de tres capas, diferenciando la base de datos, los servidores de aplicación y los frontales.

La tabla 1 muestra la cantidad de máquinas y las versiones de todos los componentes de las aplicaciones que nos interesan. En ella, la primera columna refleja los diferentes componentes de la arquitectura (base de datos), servidores de aplicación, frontales y Elastic Search. En la segunda, se puede observar la disposición de máquinas, o en algún caso clústeres, y las versiones de los componentes mencionados para APP1, APP2 y APP3. Por último, la tercera columna refleja el estado esperado tras la migración a APP FINAL.

	ANTES DE LA MIGRACIÓN		**DESPUÉS DE LA MIGRACIÓN**	
	Versión	**Número de máquinas**	**Versión**	**Número de máquinas**
Base de Datos	PostgreSQL 8.3	APP1: Una APP2: Una APP3: Una	PostgreSQL 14	Una máquina
Servidores de Aplicación	APP1: Tomcat 2.0 APP2: JBOSS 6.1 APP3: JBOSS 4.2.3	APP1: Una APP2: Una APP3: Dos	JBOSS 7.4	Un clúster (2 Servidores)
Frontales	APP1: Apache 1.3 APP2 y APP3: Apache 2.0	APP1: Una APP2: Tres APP3: Tres	Apache 2.2	Un clúster (3 máquinas)
Elastic Search	Solo en app3 Elasticsearch 5.0.5	APP1: - APP2: - APP3: Una	Elasticsearch 7.14.0	Un clúster (3 máquinas)

Tabla 1. Disposición de máquinas y versiones

De la tabla anterior cabe destacar el cambio de versión de la Base de Datos y la nueva configuración del Elasticsearch. Hace relativamente poco tiempo, se añadió una máquina de

Liferay en APP2 que resultaba en la creación de un clúster en los servidores de aplicación. Con la intención de clarificar la disposición de las máquinas, se adjunta la figura 1.1. Es necesario recalcar que los nombres de las aplicaciones y de los equipos no son los reales por motivos de protección de datos.

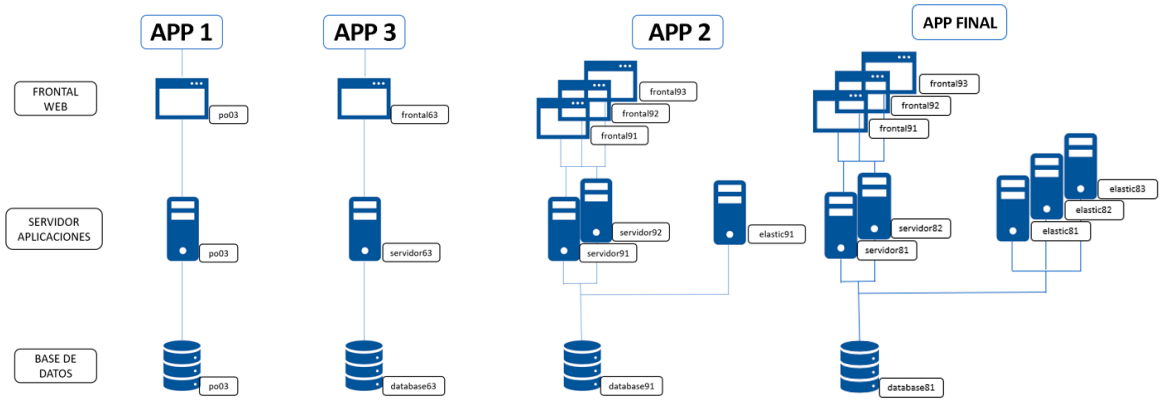

Figura 1.1 Disposición de máquinas y versiones

Una vez analizadas las principales diferencias de versiones y arquitectura entre las tres aplicaciones que soportan los portales de acceso de la Agencia de la Salud y las Enfermedades, y sus diferencias con la nueva aplicación APP FINAL, se puede describir de forma más concreta la finalidad del proyecto.

El **objetivo final** es lograr la instalación y configuración correcta de todos los elementos y la migración completa de APP1, APP2, APP3 a la nueva APP FINAL, haciendo que sobre esta última aplicación recaiga el servicio a los clientes en el entorno de Producción. Al finalizar el proyecto se realizará una comparación entre las alarmas que emerjan en la nueva aplicación con las que aparecieron tanto en un mes estándar del año 2022, como al inicio del COVID-19. Mediante este análisis se podrá concluir lo que ha supuesto este estudio y si ha permitido una mejora sustancial o no.

Para la consecución del fin descrito anteriormente, se van a establecer puntos de control u objetivos a corto plazo, que irán marcando la temporalización de la migración. Principalmente, los checkpoints o puntos de control que se han establecido son ocho:

1- Creación y configuración de la nueva base de datos PostgreSQL en la nueva APP FINAL. Se comprobará que se ha realizado correctamente en caso de que se pueda entrar con el usuario creado y la disposición de los elementos sea la que se había especificado.
2- Configuración y preparación de los Servidores de Aplicación, el Liferay y el Elasticsearch. En esta fase será necesario modificar los servidores de aplicación cambiar contraseñas, gestionar los diferentes nodos, especificar las direcciones de las bases de datos… Se comprobará mediante una validación con la empresa encargada del desarrollo de la aplicación.
3- Enlazar los Servidores de Aplicación y el Liferay a Elasticsearch. Mediante esta configuración se logrará que todos los nodos donde está instalado el Elastic y que forman el clúster se vean entre ellos y estén levantados y que los nodos de los servidores de aplicación (que son otros) donde está la aplicación detecten el clúster entero del Elastic y se comuniquen con él

4- Prueba intermedia de arranque de los JBOSS (SAs) y los Elastic. Se realizará una prueba intermedia que consistirá en comprobar que están levantados todos los componentes modificados anteriormente.
5- Upgrade del Liferay. Tras cambiar la base de datos PostgreSQL, hay aspectos que es necesario modificar, ya que el Liferay va enlazado a la base de datos. Para validarlo y comprobar el correcto funcionamiento se observará el archivo de log tras realizar los cambios y se hablará con el departamento de soporte Liferay, empresa externa que realizará sus pruebas.
6- Modificación de los frontales. Será necesario crear un nuevo virtual host de portales para dar acceso desde fuera a la aplicación. Se revisará que se ha realizado bien mediante accesos a la url y se comprobará que se encuentran los archivos estáticos correspondientes.
7- Despliegue de APP FINAL. Se realizará el despliegue de la aplicación en los entornos de prueba, preproducción y finalmente en producción, momento en el que estarán correctamente migradas las aplicaciones en APP FINAL).

Por último, antes de desplegar la aplicación en producción y que el tráfico recaiga sobre ella, se usarán dos entornos de trabajo que posee la empresa destinados a la realización de pruebas para evitar problemas en el entorno de producción. Por ello, antes de la consecución del objetivo final se deberá desplegar y validar la aplicación de APP FINAL en el entorno de pruebas (TEST) y el de preproducción (PRE). Desde la Agencia deberán aprobar estos despliegues confirmando que la instalación, conexión de componentes y el montaje de *APP FINAL* es correcto.

Posteriormente, tras realizar un análisis de las alarmas a configurar en Introscope y Nagios, se validarán con el equipo de soporte de la aplicación para que den su visto bueno. Esta validación supondrá otro punto de control para poder continuar con la configuración de las herramientas y avanzar al siguiente paso.

1.3 Impacto Esperado

Para evaluar el impacto que tendrá la migración de las aplicaciones a *APP FINAL* hay que tener en cuenta dos puntos de vista atendiendo al perfil profesional de la persona que interactúe con la nueva aplicación. Así, se analizará el punto de vista de un técnico de la empresa y de un usuario promedio al entrar al portal de la Agencia de la Salud y las Enfermedades.

Al acceder a cualquier portal web, un usuario promedio puede detectar dos tipos de fallos de la página. El más común, en caso de tener una gran carga de trabajo, es experimentar retrasos en los tiempos de respuesta, con pantallas que tardan demasiado en cargar o accesos que se realizan en un tiempo mayor de lo esperado. El otro caso engloba problemas más serios, que requieren reinicios de la aplicación y provocan que se tenga que detener el servicio por un tiempo.

El exceso de los tiempos de respuesta en alguna de las tres aplicaciones previas constituye un error muy común, ya que se pueden encontrar alarmas de este tipo a diario en varias ocasiones. Desde Introscope, se ha podido extraer una representación gráfica de las alertas

que salieron en uno de los primeros días de la pandemia de la COVID-19, tal y como muestra la Figura 1.2. Se puede ver que, en sólo 24 horas, en APP1 (una de las tres aplicaciones que van a ser migradas) hubo 44 alertas por excesos de tiempos de respuesta.

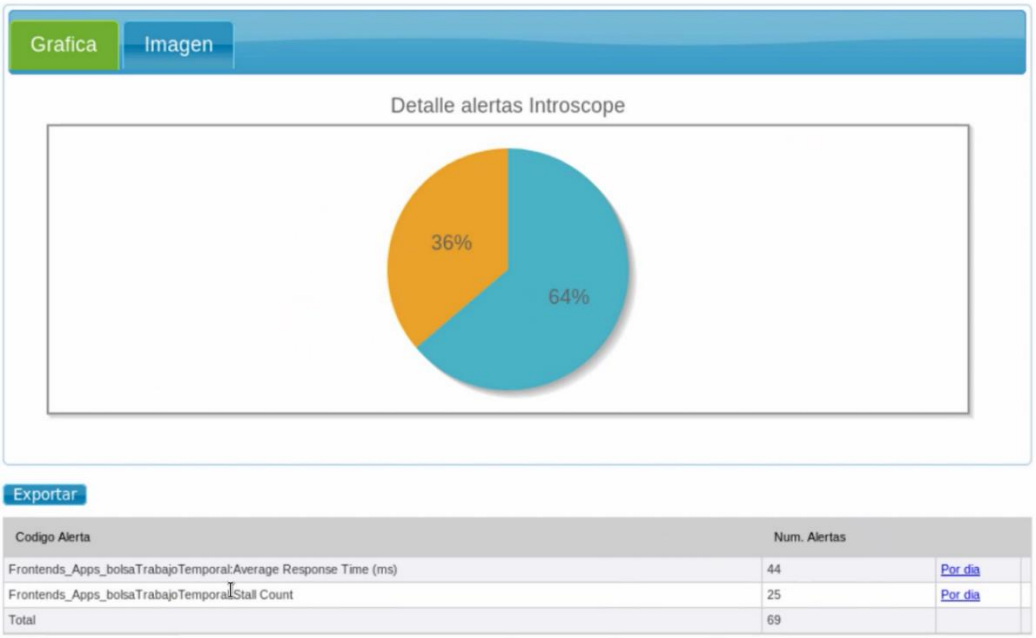

Figura 1.2 Representación de las alertas en APP1 en uno de los primeros días del COVID

Además, vemos que se notificaron también 25 alarmas por Stall Counts (transacciones que superan el umbral de los 30 segundos). Esto ocurre cuando la carga es muy elevada o ha habido un problema con la aplicación y en muchas ocasiones requieren un reinicio (ya que la transacción no se resuelve, se queda enganchado el recurso y no se libera). Como hemos explicado anteriormente, en un momento tan importante tener que reiniciar los portales de los que dependía APP1 (en este caso) tres o cuatro veces al día como mínimo no fue la opción idónea. Sin duda, esto generó un gran impacto negativo para los usuarios.

Esta migración debe suponer un gran paso adelante en cuanto a disponibilidad, disminución de errores y alarmas, mejorando la interacción de los usuarios con la aplicación. Asimismo, también se espera un cambio significativo en el trabajo de los técnicos que interactúan diariamente con la aplicación, quienes dedican gran parte de su tiempo a gestionar las alertas que emergen en las aplicaciones en las que trabajan. Estas alertas deberían disminuir drásticamente con la ampliación de nodos y actualización de las tecnologías tras la migración, con lo que el tiempo que invertía el equipo en revisar las alertas y reiniciar los sistemas, se podría utilizar para otras tareas como mejorar otras aplicaciones críticas que también requieren ser actualizadas a tecnologías mejores y más punteras.

1.4 Metodología

Para el desarrollo de este trabajo, es necesaria una continua comunicación con los grupos de soporte. Cada una de las tres aplicaciones, tanto APP1 como APP2 y APP3 que van a migrarse a APP FINAL, tiene un equipo detrás que forma parte de una empresa externa que se dedica a desarrollar la misma. Por ello, y para que el software que realizan pueda funcionar dentro de la arquitectura y de la distribución de la aplicación que se les da desde Prosodie Ibérica S.L.U., será imprescindible, tras cada paso, establecer una comunicación con ellos para informarles y que realicen las comprobaciones necesarias.

Los checkpoints expuestos como objetivos en la sección 1.2 marcan los pasos descritos para completar la migración, desde la creación o modificación de la base de datos hasta el despliegue final. Es necesario realizar un inciso, ya que en este documento no se va a hacer referencia a la repetición de pruebas o cambios reiterativos que soliciten los equipos de soporte de las empresas externas de forma extraordinaria. En otras palabras, hay ocasiones en las que el equipo de la aplicación pide rehacer un despliegue o cualquier otro proceso sólo para tomar datos estadísticos o planificar una posible reversión, y aunque se nombrarán todos estos pasos y se tendrán en cuenta, no se explicarán tan en detalle como el proceso en sí en caso de tratarse de una repetición de lo anterior.

Para poder llevar a término este proyecto, será necesario un largo proceso de adaptación. La primera fase del trabajo consistirá en estudiar la arquitectura de la empresa en pos de lograr conocer la estructura de máquinas, conceptos y todos los términos necesarios para realizar una migración de tal calibre. En concreto, será imprescindible conocer el funcionamiento del servidor de portales Liferay con JBoss y del Elasticsearch, un motor de búsqueda, lo que requerirá cierto tiempo para poder manejarlo con facilidad, ya que antes de entrar a la empresa el alumno no poseía conocimientos sobre el asunto. Tras esta larga etapa de formación, se procederá a realizar como tal la migración.

1.5 Estructura

El resto del presente estudio se presentará dividido en diferentes secciones. En el siguiente punto, correspondiente al segundo capítulo, se presentará la empresa Prosodie Ibérica S.L.U., su dedicación principal y el proyecto para la Agencia de Salud y las Enfermedades en el que se ubica este trabajo.

En el tercer capítulo, se analizará el estado del arte. En él, se mostrarán como diferentes herramientas comerciales realizan los despliegues, uno de los pasos que incluirá la aplicación de APP1, APP2 y APP3 a APP FINAL. En este apartado, se establecerá una diferencia entre los productos de las diferentes empresas y el despliegue llevado a cabo en Prosodie S.L.U.

Posteriormente, el cuarto capítulo profundizará en el problema que presentaban las aplicaciones que se migrarán y se adjuntará un pequeño análisis de estimación del esfuerzo que costará dicha migración. Después, en el capítulo quinto se profundizará en el diseño de la solución adoptada, analizando los diferentes elementos de la arquitectura y explicando su evolución al modificar los componentes.

Además, en el sexto capítulo se presentará el desarrollo y la implantación de la solución descrita, detallando todos los pasos seguidos para cumplir el objetivo. Finalmente, se comprobará el funcionamiento correcto de APP FINAL mediante un despliegue en el entorno de producción. Tras el despliegue, la aplicación deberá realizar las pruebas pertinentes para comprobar, desde su óptica, que todo va como se espera. En el sexto capítulo se abordarán estas, además de pruebas de carga para comprobar la eficiencia y el número de recursos.

En el sétimo capítulo se mostrarán las pruebas por las que ha pasado la aplicación para poder llegar a la ciudadanía. Posteriormente, en el capítulo octavo se establecerán las conclusiones del estudio en relación con los objetivos planteados. Además, se mostrarán los problemas encontrados, los errores cometidos y una pequeña recapitulación sobre la importancia del proyecto para mi aprendizaje y futuro en la empresa.

Junto con las conclusiones, en el séptimo capítulo se tratarán ciertos aspectos del trabajo que podríamos haber desarrollado con mayor profundidad o cambios que, tras la implementación y vistos los errores cometidos, se podrían haber realizado. A continuación, el capítulo décimo contendrá las referencias consultadas para la correcta enmarcación teórica del estudio.

Por último, se incluirán los anexos, en caso de que sea necesario desarrollar más en profundidad alguna parte del código o ampliar una parte determinada, y un glosario de términos donde se explicarán ciertos conceptos técnicos de este trabajo para facilitar su entendimiento.

2. Presentación de la empresa

El presente capítulo se dedicará a presentar a la empresa Prosodie Ibérica S.L.U., entidad en la que se ha realizado este proyecto. Debido a que la migración que da lugar a este trabajo tuvo su inicio en una práctica en empresa, bajo conocimiento y aprobación del tutor, en la sección siguiente se detallará la historia de esta, su dedicación general y el proyecto en que se enmarca el trabajo, así como el equipo responsable del mismo.

Prosodie Ibérica S.L.U es una empresa constituida el 31/07/2002 con sede en Madrid, que también posee oficinas en Valencia. Su objeto social según eINFORMA (portal web que suministra información comercial, financiera, sectorial y de marketing de empresas a través de Internet) es la prestación de servicios informáticos y de telecomunicaciones, voz y datos, asesoramiento, consultoría y desarrollo de soluciones para centros de atención al cliente.

El propósito de la compañía es ofrecer a las empresas servicios para la gestión 24×7 de las tecnologías de información y comunicaciones, garantizando la continuidad del negocio y las operaciones del día a día. Posee una organización interna estructurada en diversos niveles y grupos de soporte orientada a la prestación de servicios en base a los estándares ITIL, COBIT, PMI, Lean IT, ISO 20000.

En busca de las mejores soluciones para los clientes, Prosodie Ibérica S.L.U. tiene una asociación con algunas de las principales empresas del sector de TI. Algunas de ellas son Alphabet Inc. (creada tras una reestructuración de Google en 2015), Oracle Corporation, Cisco Systems, Hewlett-Packard, VMware Inc., Huawei, o Amazon Web Service [1]. Las cuatro primeras están dentro de las 100 mejores compañías digitales en 2023, según el ranking de la revista Forbes. Este grupo de empresas provee diferentes herramientas para la gestión de usuarios (como Remedy o CA) infraestructuras (como Cisco o Fortinet), servicios Cloud (como AWS o Azure) y automatización (como SaltStack o Stackstorm). Se pueden ver más en la figura 2.1.

Figura 2.1 Empresas asociadas con Prosodie

En la actualidad, además de dar soporte a otras empresas, Prosodie Ibérica S.L.U. ofrece servicios en diversos ámbitos:

- SERVICIOS CLOUD: La cartera de servicios en la nube permite a las empresas hacer de la nube su forma principal de ofrecer y consumir IT. La diferenciación de Prosodie

Ibérica S.L.U. es que proporciona una transformación hacia la nube de forma que se adapta a las necesidades, la velocidad y la agilidad que demandan los clientes.

- CONTACT CENTER: El 'Support Center' o 'Contact Center' es un departamento dentro de Prosodie que ofrece tres servicios disponibles 24x7: Service Desk, donde se atiende a los clientes mediante diferentes vías, CGR (Centro de Gestión Remota) que se encarga de monitorizar las infraestructuras de los clientes y propias y 'Operaciones', encargado de resolver incidencias, peticiones, ejecutar rutinas y diseñar procedimientos.
- AUTOMATIZACIÓN INTELIGENTE: La integración y automatización de servicios y herramientas permite realizar procesos con la mínima intervención del usuario, resolver incidencias o responder ante eventos instantáneamente definiendo reglas y flujos de trabajo.
- SERVICIOS DE DATACENTER: La empresa Prosodie Ibérica S.L.U posee diversos centros físicos que permiten a las empresas clientes establecer su plataforma cerca de los usuarios. Están ubicados en tres sedes: Madrid, Valencia y Francia. Los centros de datos poseen diversas características interesantes como sistemas de alimentación ininterrumpida (SAI), distribución de pasillos calientes y fríos (Cold Aisle Corridors) y sistemas de seguridad con protección contra incendios, detección de líquidos o control de acceso.
- SERVICIOS DE CONTACT CENTER: La herramienta 'Odigo' (solución Contact Center as a Service - CCaaS) permite un contacto telefónico directo entre los técnicos y los clientes, reduciendo el tiempo de espera drásticamente en la resolución de incidencias.
- SOLUCIONES DE TRABAJO EN REMOTO: Presentan a las diversas empresas maneras de crear entornos virtuales de trabajo, tanto en la gestión de la infraestructura como sistemas de conexión y virtualización de servidores, escritorios, etc.

Además de los grupos de soporte y aquellos que permiten ofrecer los servicios indicados anteriormente, desde Prosodie Ibérica S.L.U. se está llevando a cabo un proyecto para una Agencia de la Salud y las Enfermedades. En el primer capítulo de este documento se introdujo el grupo de GSI, encargado de la gestión de sistemas de dicho proyecto. La misión de los técnicos que trabajan en él es gestionar las diferentes aplicaciones y evolucionarlas mediante tecnologías más punteras.

Es necesario e imprescindible resaltar que, al ser un proyecto realizado de forma específica para una Agencia de la Salud y las Enfermedades, el objetivo del mismo no es lograr comercializar un producto ni establecer un servicio estándar que poder ofrecer a otras empresas. Algunos de los programas y herramientas utilizados han sido creados por el equipo técnico específicamente para el proyecto, como un gestor de tickets o una página web que permite organizar todas las máquinas usadas en el proyecto, estructurándolas por aplicaciones involucradas, entornos, dominios, versiones y especificando sus características.

3. Estado del arte

En este proyecto se presentará una solución completa para la migración de frontales, servidores de aplicaciones, base de datos, y todos los componentes de las tres aplicaciones que conforman los portales de acceso de la Agencia de la Salud y las Enfermedades de La Comunidad.

En la actualidad, no hemos encontrado ninguna aplicación web ni dominio en Internet en el que se facilite una herramienta para migrar de forma completa todos los componentes. Por ello, no es posible elaborar una comparación entre una herramienta externa y el procedimiento descrito en la introducción que llevaremos a cabo en este proyecto.

Aun así, sí que se han encontrado ciertas herramientas que permiten automatizar o realizar algunas partes de la migración. En la actualidad, hay diversos programas que permiten realizar despliegues de aplicaciones, y a pesar de no poder compararse con este proyecto de forma integral, analizaremos varias alternativas en pos de establecer un análisis del despliegue que realizaremos desde las 3 aplicaciones originales a APP FINAL.

3.1 Microsoft System Center Configuration Manager

Microsoft ofrece SCCM (System Center Configuration Manager) [2], una herramienta para la distribución de aplicaciones tanto de Microsoft como de terceros a equipos cliente en la red que permite desplegar software propio de Microsoft de forma sencilla.

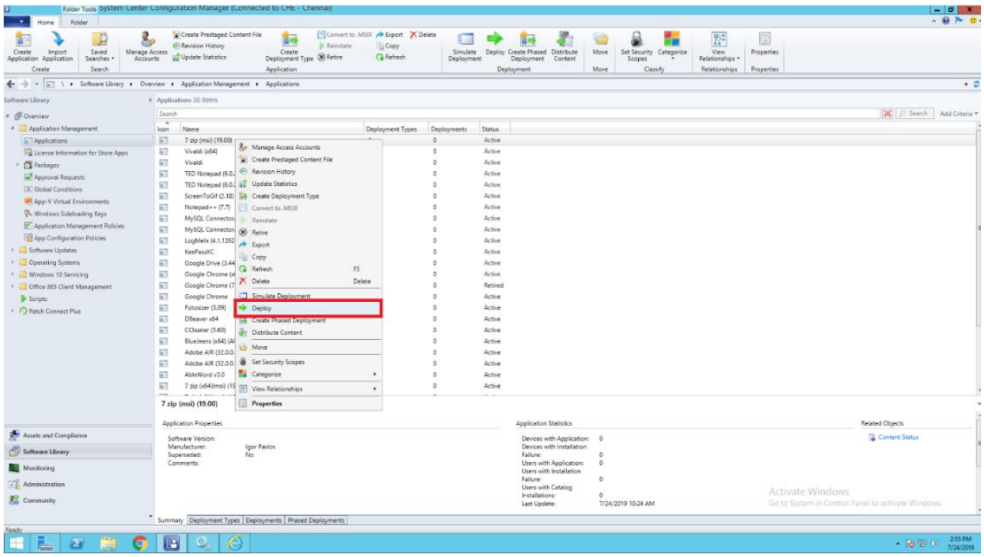

Figura 3.1 Menú del SCCM (System Center Configuration Manager)

Sin embargo, para poder trabajar con aplicaciones de terceros será necesario disponer de otras herramientas, en este caso para describir el comportamiento de un despliegue se usará la consola **Patch Connect Plus,** y los administradores deberán llevar a cabo varias tareas previas como aplicación de certificados de firma de código o creación de paquetes de aplicaciones. A continuación, se describe brevemente el proceso para realizar un despliegue con SCCM:

1. Configuración de Patch Connect Plus: se configurará esta consola que simplifica la implementación con una biblioteca de más de 300 aplicaciones.
2. Sincronización de los archivos binarios de la aplicación con su servidor SCCM: mediante una configuración de los ajustes de SCCM (figura 3.1). Además, se creará una ruta compartida para los binarios de software, que limpiará los elementos desactualizados cada 60/90 días.
3. Comprobar aplicaciones para desplegar: Una vez esté todo correctamente configurado, en la figura inferior , vemos el elemento "Overview/Application Management/Applications" en el menú del lado izquierdo. Pulsando ahí, se puede ver el listado de aplicaciones para desplegar.
4. Al pulsar sobre la aplicación solicitada, se abrirá el "Asistente de Implementación de Software". En él, se podrá desplegar, seleccionando una colección de equipos clientes y el punto de distribución (nodo que va a reproducir la aplicación en los clientes).
5. A continuación, se especificará una fecha para realizar la implementación, y se configurará el asistente para notificar en el momento del fin del despliegue.

Mediante el apartado resumen del asistente de despliegues, se puede ver un resumen en detalle de todos los elementos configurados. Tras esto, la aplicación estará en los equipos objetivo. En comparación con otras herramientas similares del sector, SCCM ofrece diversas ventajas.

- Permite programar despliegues a cierta hora sin necesidad de intervención del usuario.
- Implementa el 'Software Metering', técnica que permite conocer quién utiliza una aplicación y de qué forma, únicamente lanzando una consulta.
- Proporciona reportes con información sobre el sistema, como la tarjeta gráfica, información sobre despliegues de aplicaciones, espacio en disco y la información que se desee.

3.2 PDQ Deploy

PDQ Deploy [3] es una herramienta de implementación software creada para entornos Windows que permite automatizar la distribución de aplicaciones en un entorno, y posee una versión gratuita y otra de pago. Para ser utilizada, deberá ser instalada en uno de los servidores presentes en el sistema, momento en el que usando las credenciales del administrador del sistema se le deberán otorgar los permisos para desplegar aplicaciones. Una vez esté la conexión configurada, aparecerá una ventana inicial como la mostrada en la figura 3. Desde ahí se podrá proceder al despliegue de una aplicación, tarea para la que se deberán seguir los siguientes pasos.

1- Pulsaremos "New Package" en el menú (figura 3.2) superior para crear un paquete con un .exe que se desplegará posteriormente. Tras ponerle nombre permitirá especificar un procedimiento mediante "steps" o pasos que describirán acciones que se ejecutarán sobre el paquete: instalación, abrir una interfaz de línea de comandos, powershell, escanear, copiar ficheros...

2- Al seleccionar la opción "instalar", necesaria para desplegar, se deberá situar una ubicación del ejecutable de la aplicación a instalar (normalmente un directorio compartido). Así, se generará un paquete con un instalador interno. Para desplegarlo, se pulsará el botón derecho > Deploy Once. Permite realizar despliegues sobre el

propio equipo o un "PDQ Inventory" entre otras opciones. Una vez elegida/s la/s máquina indicada/s, comenzará el despliegue sobre ella/s.

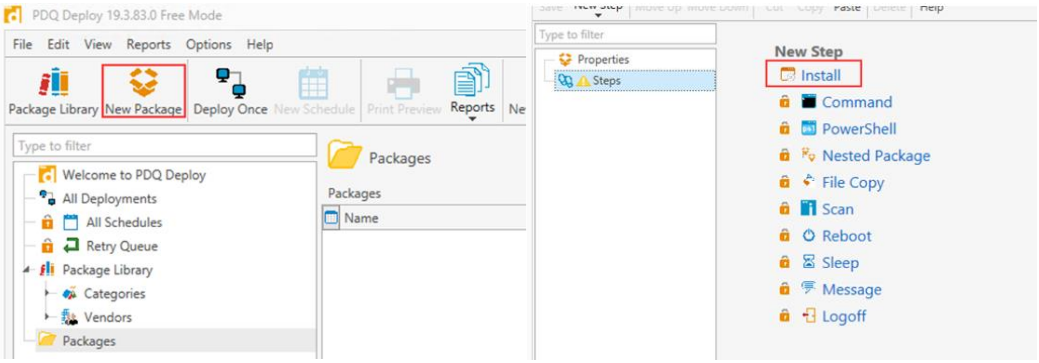

Figura 3.2 Menú principal de PDQ Deploy y menú de 'Steps'

Es un software que permite realizar despliegues de forma muy sencilla, pero únicamente soporta implementaciones sobre aplicaciones de Windows (ficheros .exe o .msi). Asimismo, a diferencia de otros productos con finalidad similar la versión gratuita sólo dura catorce días y el precio de la herramienta es relativamente elevado, alrededor de 1.300$ al año por administrador.

3.3 Octopus Deploy

Octopus Deploy [4] es un servidor que permite automatizar el despliegue de aplicaciones ASP. NET, Java, NodeJS y diferentes tipos de Scripts. Tiene dos componentes, una web centralizada llamada 'Octopus Server' y agentes localizados en cada uno de los endopints o 'Tentacles'.

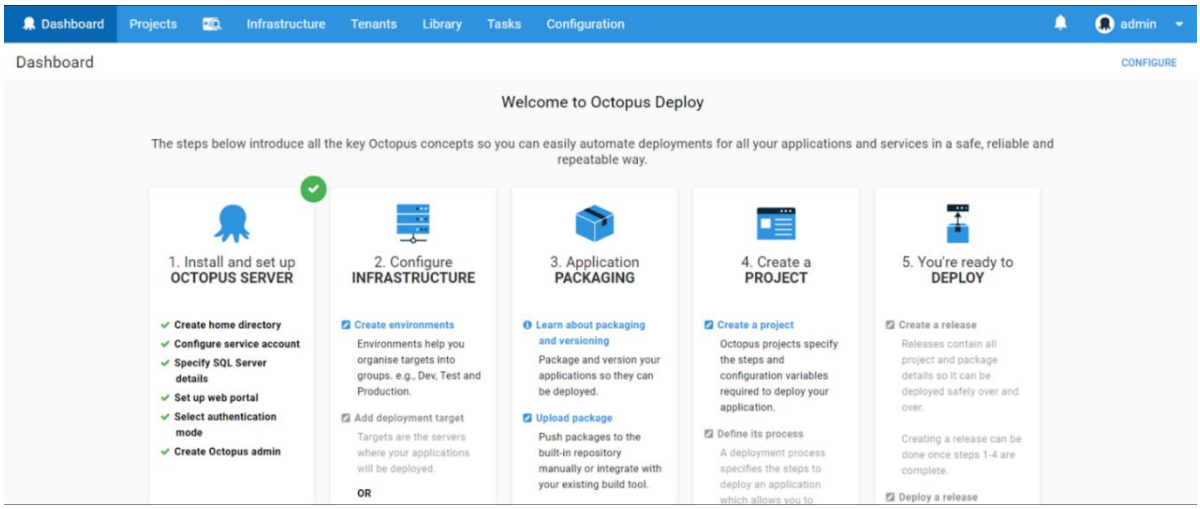

Figura 3.3 Menú principal Octopus Deploy con los pasos del despliegue

Para explicar cómo realizar un despliegue mediante la herramienta Octopus Deploy es necesario comenzar por la descarga y montaje del 'Octopus Server'. Al tratarse de opciones de configuración sobre el producto y no ser relevantes en cuanto al despliegue, vamos a obviar el desarrollo completo de ese punto. En segundo lugar, en un despliegue mediante esta herramienta es necesario identificar el entorno, es decir, los nodos o servicios sobre los que vamos a realizar el despliegue. Para ello, en la ventana inicial del cliente de Octopus se

seleccionará el paso 2 “Configurar infraestructura”, y se seleccionará el tipo de objetivo en el menú (Aplicación de Azure, Kubernetes, Conexiones SSH, Tentacles, Paquetes sin conexión…).

El tercer paso para realizar el despliegue es crear el proyecto a desplegar o empaquetar las aplicaciones que se instalarán en los endpoints. En este proceso, el producto permite definir el “proceso de implementación”, un conjunto de pasos que determinarán las acciones que realizará el software. Así mismo, en este cliente se pueden encontrar plantillas con los pasos del despliegue ya definidos.

Finalmente, las acciones realizadas para el despliegue dependen del entorno seleccionado, pero en caso de querer desplegar en nodos ‘Tentacles’, simplemente es necesario configurar uno de los pasos anteriores, llamado “enviar paquetes a Octopus”. En las opciones emergentes del lado derecho, se especifica el proyecto seleccionado y el servidor de implementación de Octopus. Desde ahí, y con la nueva versión del proyecto creada, en la opción “projects” del menú principal únicamente hay que pulsar sobre el botón “create release”. Deberá ir a los nodos objetivos y podrá ver desplegada la aplicación seleccionada.

3.4 EMCO Remote Installer

EMCO Remote Installer [5] es una herramienta de implementación software que permite instalar, desinstalar y reparar software Windows de forma remota a través de una red de área local.

Para realizar un despliegue con esta herramienta, el primer paso será escanear la red para encontrar máquinas en las que poder desplegar las aplicaciones. El producto de EMCO permite seleccionar los equipos por rangos de IPs (manualmente) o desde un archivo de texto.

Posteriormente, se deberá especificar el archivo de instalación. El programa permite una Instalación rápida, donde simplemente hace falta seleccionar el fichero y las máquinas objetivo. Otra posible solución que ofrece la herramienta es trabajar con paquetes con formato MSI (Windows Installer). A la hora de desplegar diferentes versiones de un producto, esta última opción es la recomendada, ya que permite definir tareas y guardarlas, para sobre ellas aplicar cambios al desplegar futuras versiones.

Para finalizar, al crear o especificar el archivo de instalación, el asistente solicita indicar la versión de equipos objetivo (32 o 64 bits), a pesar de que en caso de no especificar nada el instalador remoto seleccionará el paquete correcto para cada cliente. Como última opción se seleccionará (o no) la opción del reinicio tras el despliegue.

Respecto a otras herramientas similares, el instalador remoto de EMCO posee ciertas ventajas:

- Inventario de Software: es un programa que permite conocer la versión del software instalado en cada uno de los equipos de la red.
- Destinos de implementación flexibles: permite definir condiciones para ir modificando los destinos de implementación de forma dinámica (por ejemplo, todos los equipos de la misma unidad organizativa).
- Despliegues programados: permite configurar implementaciones automáticas y especificar despliegues periódicos.

4. Análisis del problema

Las aplicaciones APP1, APP2 y APP3 eran las encargadas de soportar los portales de acceso web a la Agencia de la Salud y las Enfermedades de La Comunidad. Como ya se explicó de forma introductoria en el capítulo uno de esta memoria, la pandemia del COVID-19 trajo consigo un crecimiento exponencial de la carga de peticiones que recibían estas tres aplicaciones. El aumento en las solicitudes de acceso al portal mostró ciertas vulnerabilidades en el sistema. Ante este problema, la empresa Prosodie Ibérica S.L.U y su grupo de trabajo dedicado a este proyecto, GSI, se vio en la obligación de seguir dando servicio y trabajar de la mejor forma posible tratando de que el usuario no se viera afectado en su navegación.

No influir en la experiencia de los ciudadanos fue realmente difícil, ya que las regulaciones sanitarias y los cambios sociales que implicó la pandemia provocaron que en el ámbito sanitario tuvieran que ser modificadas una grandísima cantidad de aplicaciones. Indudablemente ese no era el momento adecuado para pensar en un cambio radical en el software y el hardware de las aplicaciones encargadas de los accesos web, sino para tratar de adecuar un sistema acostumbrado a recibir miles de peticiones diarias a otro que, en poco más de diez días, pasó a recibir decenas y cientos de miles de ellas. En ese momento, se vio que era necesaria la ampliación y actualización de bases de datos, servidores de aplicaciones y todos los elementos sometidos a estos cambios continuos.

Por otro lado, ciertos componentes de las aplicaciones previas a la migración estaban relativamente obsoletos, y con el paso del tiempo cada vez funcionaban peor, por lo que era más difícil adecuar el resto del sistema informático a ellos. Esta razón fue otra parte importante del porqué de este trabajo. Ante la necesidad descubierta en la pandemia, cuando los accesos y la carga disminuyeron notablemente a principios de 2022, comenzó este proyecto con el objetivo de actualizar hardware y software.

Desde la Agencia de la Salud y las Enfermedades se acordó con la empresa Prosodie Ibérica S.L.U que, con el objetivo de no pasar por lo mismo y poder mejorar la experiencia de los ciudadanos en los portales de acceso, se agruparían las tres aplicaciones que daban soporte hasta ese momento (APP1, APP2 y APP3) en APPFINAL. Esta sería otra aplicación con diversas evoluciones, entre las que a grandes rasgos se podrían destacar nuevos elementos, un incremento en el número de máquinas y actualización de versiones en ciertos componentes de la arquitectura.

Situación durante el COVID

A continuación, se profundizará en la situación crítica que vivieron las aplicaciones en tiempos de COVID, inicialmente evaluando como conjunto las alertas que surgieron en el año 2020 y concretando posteriormente en el mes de abril de ese año. Tras observar el funcionamiento de los sistemas a través de las alarmas que surgieron, se comparará con el estado de estos en el año 2022, y en concreto en el mes de octubre. Por último, se comentará la obsolescencia de los componentes en vista de las alertas que aparecen en APP1 y se mostrará porque es necesario realizar la migración.

A lo largo del año 2020, el sistema informático de Prosodie Ibérica S.L.U. en el proyecto de la Agencia de la Salud y las Enfermedades registró 66084 alertas, representadas en la figura

4.1. De estas, 40614 fueron provocadas por fallos en las aplicaciones, mientras que el resto emergieron por otros componentes: errores en las bases de datos, sobrecargas en las unidades centrales de procesamiento (CPUs)... En ese momento, los sistemas informáticos de La Agencia contenían más de 120 aplicaciones diferentes, y sólo entre APP1, APP2 y APP3 (un 2.5% de las aplicaciones totales) acumulaban más del 16% de las alertas, como se puede ver representado en la figura 2. En concreto, la más castigada durante ese período anual fue indudablemente APP3, con más de 5000 alertas en 365 días. Toda esta información se puede ver con mayor claridad en la gráfica 4.2.

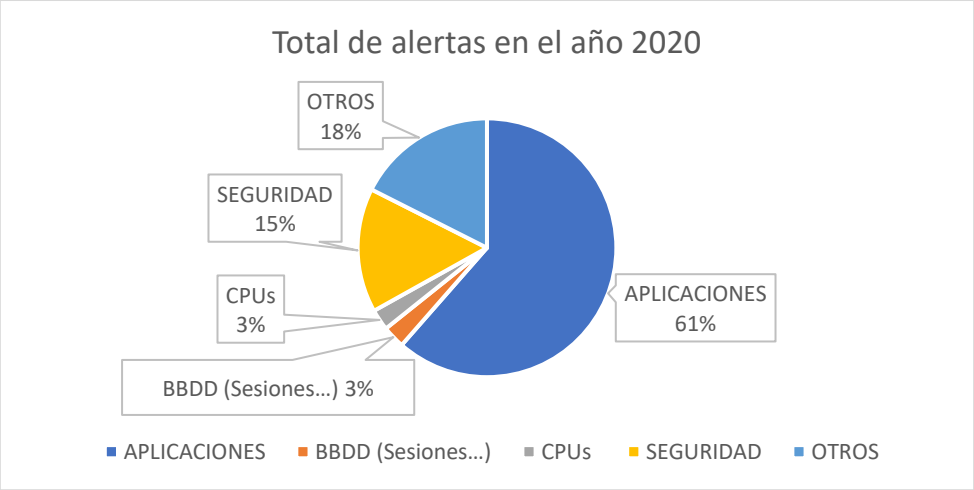

Figura 4.1 Total de alertas registradas en el sistema en el año 2020.

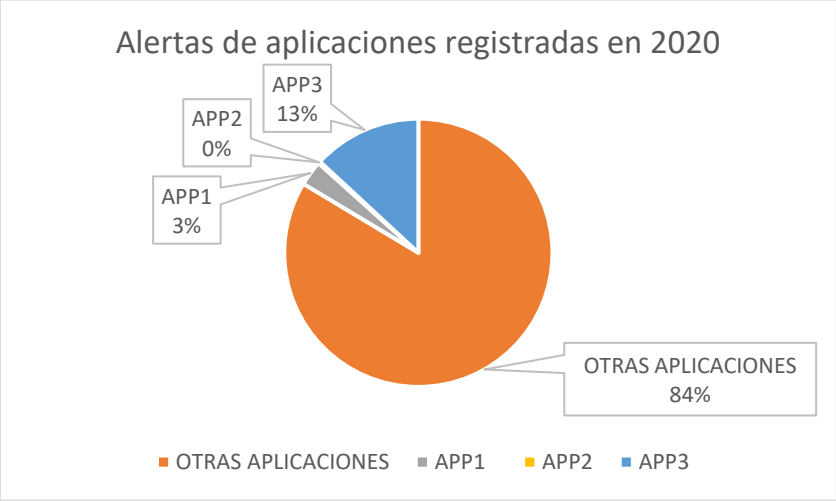

Figura 4.2 Distribución de alertas de aplicaciones en el sistema en el año 2020.

Antes de analizar más a fondo APP3 durante ese año y concretarlo en el mes de abril, el más crítico en cuanto a número de alertas y reinicios necesarios, es necesario explicar las tres alertas más importantes que puede notificar una aplicación, según están configuradas en la herramienta sobre la que recae la monitorización, Introscope. Son:

- **Stall Counts**: Describe una situación en la que una transacción no se ha resuelto en un período de tiempo determinado por los técnicos y no ha liberado los recursos

que necesita para llevarse a cabo. La herramienta Introscope, empleada para la monitorización, permite especificar cada cuántos segundos se toman datos y cuántos periodos consecutivos son necesarios para la aparición de la alerta. En este caso, en APP3, se actualiza el estado de las transacciones cada 15 segundos y la alerta emerge si la transacción no finaliza en un tiempo menor a dos minutos (ocho periodos).
En el caso de los Stall Counts se especifica un umbral de '1', por encima de él emergerá la alerta. En caso de ser una única transacción aislada la que se ha quedado bloqueando el recurso no saltará.

- **Average Response Time**: Describe el tiempo medio de respuesta que han tenido las peticiones registradas en un periodo de 15 segundos. Sin duda, es un buen reflejo de la carga que está soportando un servidor en un momento determinado. En caso de detectar en las gráficas un 'pico' por encima del umbral de forma aislada, es posible que haya alguna petición que está causando serios problemas lo que provoca que la media aumente drásticamente. Sin embargo, si el estado de retraso en las respuestas persiste durante un gran periodo de tiempo (algunos minutos), es posible que haya transacciones no resueltas, sesiones bloqueantes en bases de datos o ciertos elementos activos que afectan directamente al servidor.

  El umbral de la alerta es un segundo, y en caso de que la media del tiempo que tarda el servidor en resolver las peticiones que recibe sea mayor a un segundo durante ocho periodos (un total de dos minutos), saltará la alerta y deberá ser notificada.

- **GC (Tiempo empleado en el Garbage Collector)**: El Garbage Colector (o recolector de basura) es una función de recuperación de memoria integrada en lenguajes de programación como C# y Java que libera espacio de memoria por objetos que el programa ya no necesita. De esta forma, el programa puede otorgar ese espacio de memoria a otros objetos de forma dinámica.

  La alerta relacionada con el GC describe el porcentaje de tiempo que una aplicación está empleando el recolector de basura en los últimos 15 minutos. Al igual que en los Stall Counts, el estado se actualiza cada 15 segundos, pero en este caso, con un único periodo por encima del umbral, la alerta se deberá notificar haciendo que el equipo encargado de monitorizar notifique el problema.

  De este modo, se permite un tiempo menor al 10%, es decir, no se antoja preocupante que la aplicación emplee 90 segundos de los últimos 900 en el colector. Esta alerta es crítica, y por ello está configurada para detectarse lo más rápidamente posible.

En la mayoría de las ocasiones en las que aparece una alerta de GC, es necesario realizar un reinicio completo de la aplicación, ocasionando una pequeña pérdida de servicio. Esto se debe a que si el Recolector de Basura no es capaz de administrar la memoria de forma dinámica y el servidor se queda lleno al 100%, se perderán datos cruciales como logs, información de los errores y se sobrescribirán algunas de las peticiones más antiguas realizadas al servidor.

En caso de tener transacciones sin liberar recursos de memoria (Stall Counts), se monitorizará el sistema comprobando sesiones bloqueantes, solicitudes pendientes, ocupaciones de discos y demás parámetros de la aplicación que puedan afectar a que la transacción detenida no finalice. Aun así, también pueden generar retrasos en las respuestas a peticiones de los usuarios, llegando en un caso extremo a un reinicio completo de la aplicación, siempre y cuando haya afectación directa al servicio y sean varias las transacciones afectadas.

Las alarmas de tiempo de respuesta elevado suelen venir ocasionadas por otros factores. Estos pueden ser diversos, y el procedimiento en caso de encontrarse con una de ellas es llegar al foco de la cuestión y resolverlo. No requerirán un reinicio, pero en numerosas ocasiones implican un periodo grande de tiempo para el técnico, ya que se deberán revisar si hay sesiones bloqueantes en base de datos, solicitudes pendientes… hasta dar con el problema. En caso de persistir y bajo petición del cliente, se podría reiniciar la aplicación.

Por último, antes de continuar hay que tener en cuenta un factor determinante a la hora de evaluar la afectación que tienen las alertas encontradas en un periodo de tiempo, la franja horaria en la que salen. Las alertas que se comentarán a continuación se notifican, en su mayoría, en un horario de máximo servicio. Esto supone un problema adicional, ya que, si el técnico se ve obligado a aplicar un reinicio en hora punta (franjas del día en que más usuarios están enviando solicitudes a los servidores de aplicación), los usuarios afectados serán muchos más que en horas de menor uso, de ahí que los reinicios programados sean siempre a partir de las 15:00h.

A pesar de esto, es necesario recalcar que en el caso de APP1, APP2 y APP3 en concreto, siempre que se requiera realizar alguna acción sobre los servidores habrá afectación a algún usuario, ya que los portales de acceso de la Agencia de la Salud y las Enfermedades deben estar disponibles para la población las 24 horas de los 365 días del año.

Dentro de las tres aplicaciones que se van a migrar, APP3 fue indudablemente la que más sufrió durante la pandemia en comparación a su carga habitual. La tabla mostrada en la figura 3 muestra todas las alertas que surgieron en esta aplicación durante el año 2020, un total de 5226. De estas, (que suponen un 13% de las más de 40.000 alarmas que surgieron ese año), salieron una cantidad de alertas de GC próxima a 300, lo que aproximadamente implicaría un reinicio (al menos) cada día de ese año. En estos datos no se tuvieron en cuenta los Stall Counts (4812, una media de 13.2 diarios) y las demás alertas mostradas en la figura 3.

TIPO DE ALERTA	NÚMERO DE ALERTAS
Stall Counts en Frontends.	4812
GC Monitor: Porcentaje de Tiempo en GC en los últimos 15 minutos.	279
Tiempo de respuesta medio en ms (ms)	107
Otros	28

Tabla 3. Distribución de las alertas en APP3 en 2020.

Los datos de esta tabla muestran que era necesaria una modificación de la estructura de las aplicaciones que gestionaran los portales de acceso a la Agencia. En un momento de máxima necesidad no era sostenible realizar cinco, seis o incluso diez reinicios diarios, porque la experiencia de los usuarios se pudo ver claramente afectada. Si las más de 120 aplicaciones de los sistemas de GSI hubieran funcionado del mismo modo, ese año el sistema habría reportado más de 600.000 alertas, número desorbitado e insostenible.

En concreto, el estado de alarma e inicio de la reclusión en los hogares en España tuvo lugar el día 14 de marzo de 2020. Para entender más a fondo lo que supuso el COVID-19 en APP3, se van a mostrar a continuación los datos del mes de abril, primer mes completo desde el inicio del confinamiento domiciliario. En la figura 4.3 comprobamos que aparecieron más de 485 alertas, con un pico claro el día 13 de abril, momento de mayor acceso a los portales de 'La Agencia' por parte de la ciudadanía. Además, vemos que 60 de estas fueron producidas por tiempo en el Recolector de Basura, lo que ocasionó una gran cantidad de reinicios (al igual que hemos comentado antes a nivel anual).

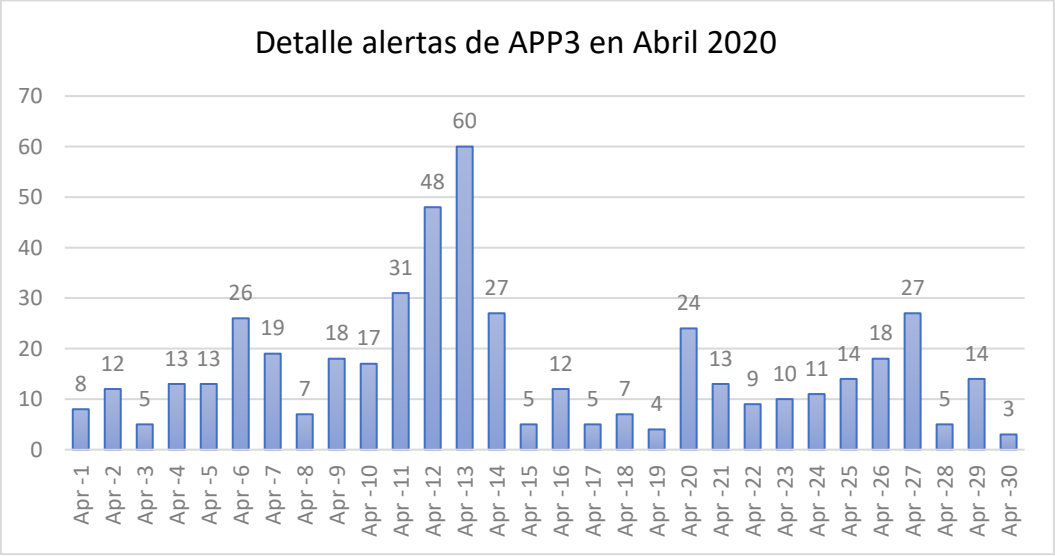

Figura 4.3 Distribución de alertas de APP3 en abril de 2020.

En las figuras 4.4 y 4.5 se pueden ver diferenciadas las alertas de Stall Counts y GC de ese mes. La estructura de las gráficas es completamente diferente. En cuanto a las transacciones bloqueadas, suelen estar entre 5-20 diariamente, mostrando la gran cantidad de peticiones diarias que tenían que soportar los servidores ininterrumpidamente. Sin embargo, en el GC, vemos que los días 04, 05, 06, 14 y 29 de abril hubo siete o más alertas, lo cual significa que prácticamente no se pudo dar servicio durante dos horas consecutivas sin tener una pérdida, aunque fuera mínima. Indudablemente esto era necesario mejorarlo.

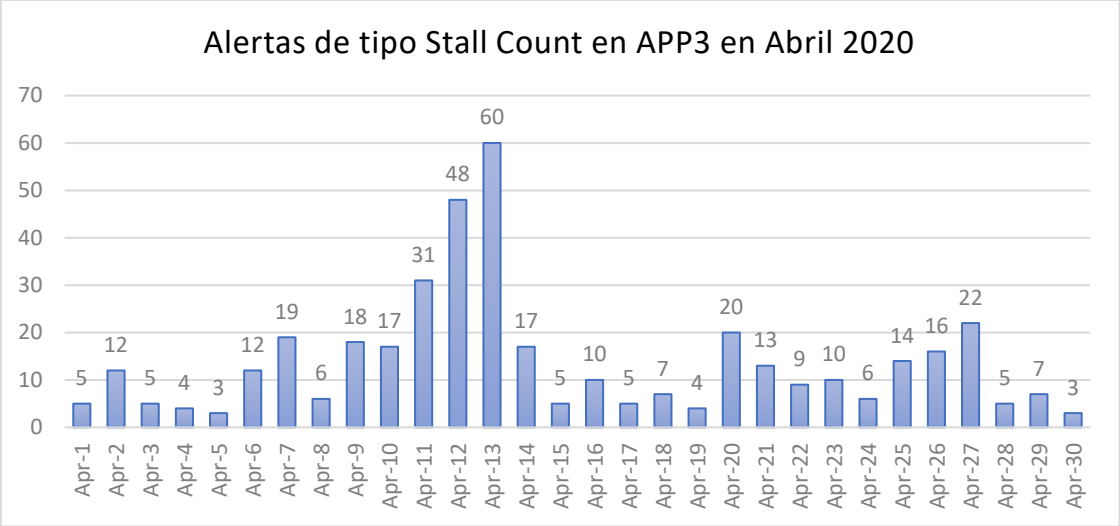

Figura 4.4 Distribución de alertas de Stall Counts de APP3 en abril de 2020.

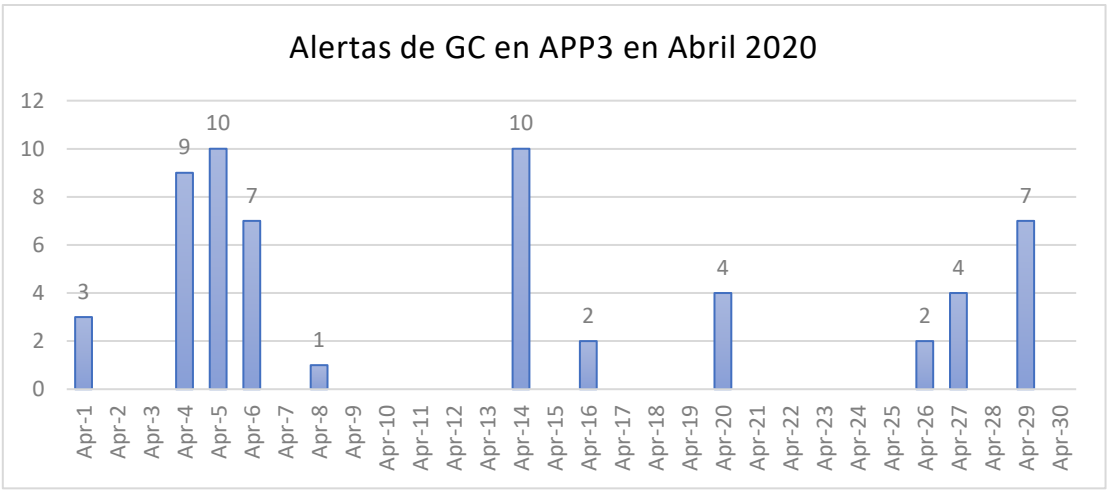

Figura 4.5 Distribución de alertas de GC de APP3 en abril de 2020.

Situación normal fuera de la pandemia

En el año 2022 podemos ver que se mantuvo la cantidad de alertas que salieron de APP1, APP2 y APP3. En este caso, el sistema registró una suma de 110474 alertas, pero en portales fueron 7087 de un total de 84188 alertas (8.4%) generadas en las aplicaciones, tal y como podemos comprobar en la figura 4.6. Prácticamente, suponen la mitad de porcentaje con relación a las que salieron en estas tres aplicaciones en el año 2020. Sin embargo, y pese a ver que las más de 7000 son similares a las del año que supuso el inicio de la pandemia, la situación es radicalmente diferente. En esta sección analizaremos a fondo las alertas de APP3 y también de APP1, que en sólo dos años ha mostrado la obsolescencia de sus componentes y sobre la que recaen la mayoría de las alertas.

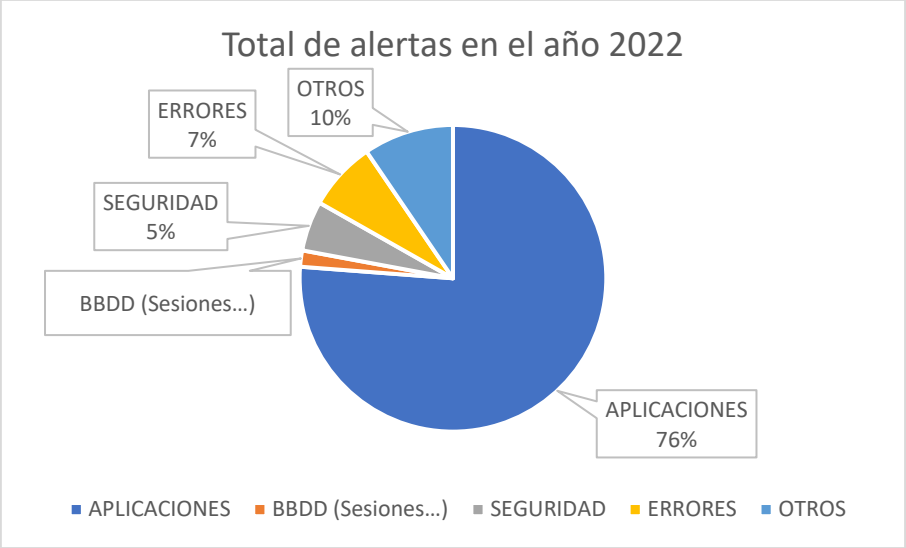

Figura 4.6 Distribución de alertas de aplicaciones en el sistema en el año 2022

El año pasado, APP3 registró sólo 282 alertas, por 569 de APP2. Más de un 85% de las notificaciones en los portales de acceso en este periodo vinieron originadas por APP1 (6236). Esta es una aplicación con un servidor Tomcat que está claramente desactualizada y se trató de evolucionar a APP3, pero todavía permanece en uso. El descenso en las alarmas que generaron las otras dos aplicaciones muestra que, en un periodo normal, estas aplicaciones también sufren reinicios, peticiones bloqueadas y tardan cierto tiempo en responder a algunas solicitudes, pero de forma mucho menor en comparación al año 2020.

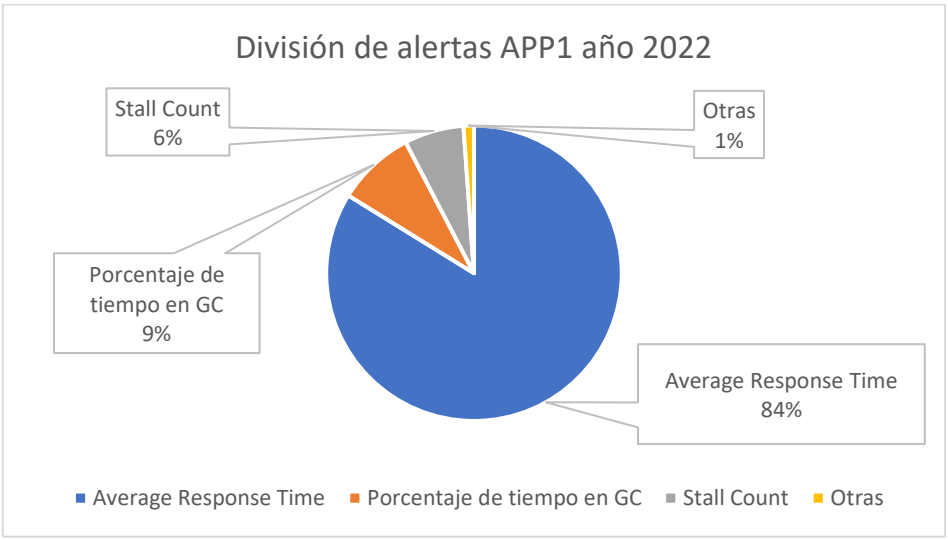

Figura 4.7 Distribución de alertas de APP1 en abril de 2022.

El último análisis de las alertas que realizaremos en este capítulo es sobre APP1, detallado en la figura 4.7. De las 7087 alertas que aparecieron el año 2022, más de 6000 fueron sobre esta aplicación. Evaluando en detalle las diferentes alertas posibles, vemos que 5227 son de tiempo de respuesta elevado. Como hemos expuesto en la página anterior, los componentes que estaban empezando a quedar obsoletos se hicieron notar, mostrando que cada año son menos capaces de gestionar correctamente todas las peticiones que reciben y de ahí la subida clara de 1238 en 2020 a 6236 en 2022, un incremento de un 503%.

Ahora, ya hemos analizado a fondo las dos principales causas que motivan este proyecto: la desactualización de ciertos componentes y la necesidad de mejorar y ampliar los sistemas informáticos ante una pandemia como el COVID-19, que parecía ficción hace unos años, pero ha demostrado ser una realidad. Aprovechando los cambios, se incluirán nuevos componentes en la arquitectura, como veremos en el capítulo.

4.1 Análisis de Riesgos

El desarrollo del presente proyecto va ligado a la Agencia de la Salud y las Enfermedades. Desde esa agencia, hay un equipo de trabajo que se encarga de realizar las pruebas y aportar a GSI retroalimentación sobre los despliegues, las migraciones y los cambios en el sistema informático. En este caso, al migrar las tres aplicaciones, antes de aplicar los cambios en el entorno de producción, se deberá pasar por diferentes filtros y entornos.

Hay una primera fase correspondiente al entorno de TEST donde se implantarán los primeros cambios, se creará la base de datos nueva… En cada uno de los pasos será necesario recibir la validación del equipo de soporte de La Agencia, para poder continuar con el siguiente paso. Al finalizar la migración en ese entorno, se pasará a otro intermedio, el entorno de PREPRODUCCIÓN (PRE). El proceso en 'PRE' será el mismo que en el primer entorno de pruebas, y al finalizar sí que se pasará a producción.

Aun así, un mal funcionamiento en el entorno de producción que tenga una afectación prolongada en el servicio ocasionaría problemas para los usuarios, no permitiéndoles entrar en la Agencia de la Salud y las Enfermedades de forma digital. Por ello, existe un gran riesgo en caso de configurar incorrectamente algún componente o realizar un mal despliegue en producción. El problema que esto ocasionaría es mucho mayor que un despliegue incorrecto de una versión de aplicación cualquiera, ya que ahí es posible volver a una versión de reversión de esa misma aplicación, pero en la migración sería más complicado tener que volver a poner en funcionamiento APP1, APP2 y APP3, ya que hablaríamos de reactivar la arquitectura entera de tres aplicaciones diferentes.

4.2 Análisis de la seguridad y el marco legal

Al igual que se ha explicado en el apartado anterior acerca del análisis de riesgos, al estar ubicada la nueva aplicación dentro del sistema informático de la Agencia de la Salud y las Enfermedades, y sustituir a otras tres aplicaciones similares, no nos adentraremos en la seguridad ni la protección de los datos en este estudio, ya que los nodos en los que situaremos la base de datos, los frontales y los servidores de aplicación estarán bajo la misma protección que cualquiera de las otras más de 120 aplicaciones que se gestionan desde el equipo de trabajo.

En este mismo sentido, desde GSI hay ciertas políticas sobre las contraseñas, directivas de grupo y configuración de los sistemas que tienen acceso a la infraestructura de La Agencia. Al ser igual que en todos los otros nodos y aplicaciones, tampoco supondrá un problema el tratamiento de los datos personales de los usuarios que visiten a diario los portales de acceso. Obviamente, es un asunto crítico y se debe preservar toda esa información, pero el cumplimiento de este fin no tiene relación directa con la migración en sí. GSI (grupo en el que tiene lugar el proyecto), no es responsable de las directivas que gestionan la seguridad ni del

tratamiento de los datos, por lo que se ha decidido no dedicar más líneas en el desarrollo de este trabajo a ese aspecto.

4.3 Solución propuesta

En vista de la desactualización de ciertos componentes de la arquitectura presente antes de la migración, y los problemas de carga que surgieron con la pandemia, se presentaron diversas soluciones para tratar de actualizar las aplicaciones presentes y lograr un mejor funcionamiento de estas.

Inicialmente, las aplicaciones encargadas de los portales eran APP1 y APP2, y al ver que tecnológicamente no estaban preparadas para ser sometidas a altas cargas de trabajo, se decidió incluir APP3 en la infraestructura. Esta "nueva" aplicación incluiría los servicios de las dos primeras, pero actualizando algunos de los componentes, como el servidor de aplicación tomcat de APP1 por un JBOSS en APP3. Esta solución no permitió sustituir las dos aplicaciones más antiguas por la nueva en su totalidad, ya que, hasta ahora, se han tenido que seguir utilizando al mismo tiempo APP1, APP2 y APP3.

Tras la pandemia del COVID-19, se ha optado por otra solución para resolver el problema que suponen los portales de acceso. La Agencia de la Salud y las Enfermedades junto al equipo de GSI optó por la creación de una cuarta aplicación, denominada APP FINAL, que sí debe sustituir a las otras tres. Esta, tendrá componentes más actualizados y mayor cantidad de nodos, para poder balancear la carga de peticiones y permitir un funcionamiento mejor para los usuarios y los técnicos encargados de su mantenimiento.

APPFINAL contará con tres frontales, dos servidores de aplicación un nodo de base de datos y tres nodos de Elastic Search, que proveerá al equipo de soporte información acerca de su aplicación. Esta es la solución que se va a poner en práctica en este estudio, y tras la configuración y el testeo de esta en los entornos de pruebas evaluaremos si la decisión tomada ha sido correcta o, al igual que pasó con APP3, no permite dejar de utilizar las aplicaciones anteriores.

5. Diseño de la Solución

Tras analizar la situación actual de APP1, APP2 y APP3 y exponer los motivos por los que se realizará la migración, en este capítulo se pretende profundizar en el diseño escogido para APP FINAL. En esta sección se expondrá la arquitectura del sistema, compuesta por diversas capas en las que se ubicarán las diferentes máquinas que formarán parte de los portales de acceso a La Agencia de Salud y las Enfermedades de La Comunidad, así como la función que tomará cada una de ellas dentro de esta arquitectura.

Por otro lado, en este capítulo se abordarán las características de APP FINAL que pretenden solucionar los problemas comentados anteriormente, y que la diferenciarán de sus antecesoras. Para entender correctamente la evolución que supondrá esta migración en cada uno de los componentes arquitectónicos, se expondrán en detalle las diferentes versiones de los mismos, realizando una pequeña comparación con sus homólogos en las aplicaciones previas a la migración y evaluando las ventajas que aportarán al nuevo sistema.

5.1 Arquitectura del sistema

Dentro del proyecto de GSI, todas las aplicaciones poseen una arquitectura de tres niveles. Este concepto, según la empresa IBM, describe una arquitectura de aplicaciones de software consolidada que organiza las aplicaciones en tres niveles informáticos, lógicos y físicos: el nivel de presentación, o la interfaz de usuario; el nivel de aplicación, donde se procesan los datos, y el nivel de datos, donde se almacenan y gestionan los datos asociados a la aplicación. [6]

La característica que hace especial a la arquitectura dividida en tres capas es que los componentes únicamente interactúan con su capa inmediatamente superior e inferior. Es decir, desde un frontal (nodo perteneciente a la capa de presentación que recibe una petición de un usuario) no se podrá acceder a la base de datos, únicamente podrá interactuar con un servidor de aplicación (perteneciente al nivel de aplicación que contiene la parte lógica).

Para comprender más en profundidad el funcionamiento de este sistema, se describirá el recorrido de una solicitud genérica por parte del cliente en la figura 5.1. Inicialmente, el cliente enviará una petición HTTP sobre un objeto de la base de datos (del tipo GET, por ejemplo). El frontal es el nodo encargado de recibir esta petición por parte del cliente y comunicársela al servidor de aplicaciones, para que la gestione y envíe a la capa de base de datos. Tras pasar por el servidor de aplicaciones (SA), son los nodos con acceso directo a las bases de datos quienes podrán extraer el objeto solicitado de la misma.

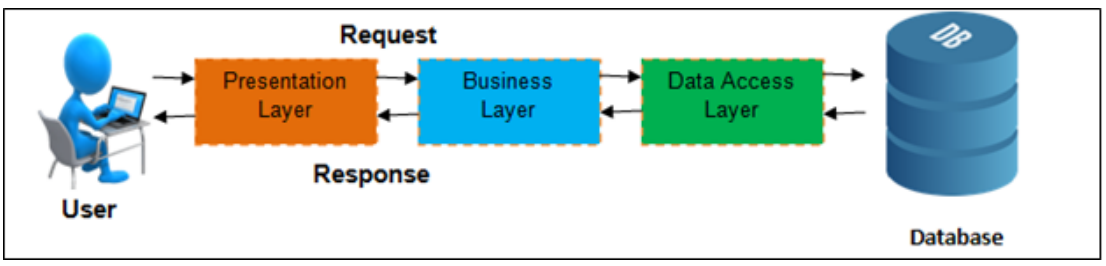

Figura 5.1 Funcionamiento de una petición en una arquitectura de tres capas.

Esta arquitectura dividida en tres niveles o capas posee ciertas ventajas que hacen de ella la mejor solución entre las diferentes alternativas. A continuación, se van a presentar algunas de ellas con el objetivo de comprender el por qué en el equipo de GSI se ha decidido escoger esta arquitectura.

- Fácil configuración y pruebas: Permite que la actualización de los componentes se realice de forma independiente. En caso de detectar un problema de conectividad (fallo que tiene que solucionar el equipo con asiduidad) desde el exterior con la base de datos, si estuviera todo en el mismo nodo, sería muy complicado descubrir cuál es y solucionarlo rápidamente. Sin embargo, teniendo tres capas, se puede comprobar la conectividad de una con la siguiente mediante un simple comando, descubriendo qué nodo tiene el problema y arreglándolo únicamente cambiando la configuración de uno de los tres componentes.
- Fácil de escalar y desarrollar: En caso de querer incluir un nodo más en la capa de presentación, que actúe como frontal y recoja las peticiones de los usuarios, al ser independiente cada uno de los niveles, basta con incluirlo en el fichero de configuración de los servidores de aplicación, mientras que el resto de máquinas (incluso los propios frontales) no necesitarían ser conocedoras del cambio. De este modo, se podría escalar con mayor facilidad que un sistema bicapa o incluso monocapa.
- Separación de responsabilidades: como hemos explicado en los puntos anteriores, cada uno de los niveles desarrollará una tarea diferente.
- Seguridad: Los diferentes nodos que hay en cada uno de los niveles están estructurados dentro de una subred diferente (además de cortafuegos y otros componentes), y al aislarlos se reduce drásticamente la posibilidad de realizar ataques sobre ellos.

Tras exponer las ventajas que supone este modelo de tres capas, es necesario recalcar que su montaje supone un mayor desembolso económico y un trabajo adicional respecto a uno de dos o una única capa. En caso de situar un servidor de aplicaciones en la misma máquina que un frontal o el nodo de base de datos, no es necesario interconectar los diferentes componentes a nivel de red, ya que se podrán gestionar las peticiones dentro del mismo equipo. Sin embargo, será necesario diseñar una pequeña red para poder conectar entre sí los diferentes componentes (por ejemplo servidores de aplicación con frontales), para permitir una comunicación segura y directa entre los equipos.

Además de las tres capas básicas, que ya se han comentado en este capítulo, que identifican a los frontales, servidores de aplicación y el nodo en el que se aloja la base de datos, en Portales habrá algunos componentes adicionales. Uno de ellos está ubicado en la capa de presentación, a la que llegan de forma directa las peticiones de los usuarios , y permite gestionarlas con la mayor eficiencia posible. Este nuevo elemento recibe el nombre de balanceador de carga o '**Load Balancer'**. Un balanceador de carga es una herramienta que direcciona a un cliente al servidor web que se encuentre con mayor disponibilidad entre los que cuentan con el mismo contenido. El objetivo de añadir uno de ellos al sistema de los portales de acceso es lograr que uno de los frontales no se sature debido a la gran cantidad de peticiones que reciben.

Lo más recurrente a la hora de configurar un balanceador de carga es dividir de forma equitativa las peticiones de los usuarios entre los diferentes frontales (y a su vez en los servidores de aplicación) . Debido a que este elemento es externo a la migración de las

aplicaciones APP1, APP2 y APP3, ya que únicamente es necesario cambiar en él las direcciones de los nodos que realizarán la función de frontales, no se incluirá como tal en este proyecto, pero parece interesante remarcar que este 'load balancer' será un factor importante en una aplicación de alta disponibilidad.

Con respecto a las tres aplicaciones que se usaban antes de la migración, APP FINAL tendrá un elemento diferenciador que permitirá mejorar los servidores de aplicación, ya que incluirá tres nodos en los que estará instalado Elastic Search. **Elastic Search** [7] es un motor de búsqueda y analítica distribuido, gratuito y abierto para todos los tipos de datos, incluidos textuales, numéricos, geoespaciales, estructurados y no estructurados. El clúster en el que irán ubicados los tres nodos con el Elastic se comunicará con los dos servidores de aplicación, con el cometido de recopilar datos sobre los accesos, peticiones, problemas y cualquier aspecto que desee conocer el equipo de soporte. Se espera que entre otros aspectos, Elastic suponga una mejora sustancial en el acceso y análisis a ficheros de log, monitorización del rendimiento de los servidores y extracción de estadísticas sobre las visitas y navegación dentro del sitio web.

En el futuro, la inclusión de Elastic Search (tecnología reciente que vio la luz en el año 2010), podría derivar en la inclusión de otros componentes del Elastic Stack, conjunto de herramientas abiertas y gratuitas para una mejor visualización y almacenamiento de los datos extraídos por este componente. Al ser la primera aplicación del sistema informático de La Agencia de la Salud y las Enfermedades que incluye esta tecnología, se evaluará su rendimiento unos meses después de la puesta en marcha en producción, y se analizará si ha sido una decisión correcta y si se deben seguir dando pasos en esa dirección hacia el futuro tanto dentro como fuera de APP FINAL.

5.1.2 Otras posibles Arquitecturas

La principal alternativa a una estructura de tres capas o niveles es la arquitectura de dos niveles. En ella, los equipos encargados de recibir las peticiones de los clientes o frontales estarían ubicados dentro del propio servidor de aplicaciones, recayendo sobre este el doble de trabajo, pero ahorrando uno de los componentes y la interconexión con los demás. Así, una arquitectura de dos capas constaría de un nivel de base de datos y otro que se comunica con el cliente y contendría la lógica de la aplicación.

Una solución alternativa, pero muy poco realista y por ello no tenida en cuenta en la creación de Portales, es construir una única capa dentro del sistema. Este sistema almacenaría la base de datos, los servidores de aplicación y los frontales o capa de presentación en el mismo nodo. De hecho, esta arquitectura es la presente en APP1, pero ante sus claras desventajas, se tuvo que montar APP2 para apoyarla (esta ya con tres capas). Una aplicación casera, en la que se instala un gestor de bases de datos en tu propio sistema, puede ser un ejemplo de aplicación de una única capa, pero debido a la importancia de Portales y la necesidad de que sea escalable y seguro ante cualquier tipo de fallo, fue una opción que ni siquiera se planteó.

El modelo que sólo consta de dos capas está pensado para aplicaciones no tan robustas, donde merece la pena reducir los componentes y las conexiones entre ellos y en las que un

mismo nodo puede desempeñar las funciones de servidor de aplicación y de frontal sin problema. Sin embargo, se ha elegido la de 3 capas porque en este caso concreto, al ser los portales de acceso de La Agencia un elemento crítico, con miles de accesos cada hora y una gran cantidad de datos a almacenar y tratar, el equipo decidió dotarlo de la infraestructura necesaria que garantizara un servicio ininterrumpido y robusto ante posibles fisuras.

5.2 Diseño detallado

En esta sección se incidirá en los diferentes componentes que componen PORTALES. A diferencia de las páginas anteriores, correspondientes al capítulo 4.1, ahora se analizará más a fondo la configuración y características principales de la base de datos, los servidores de aplicación, los frontales y el Elastic Search. No sólo se expondrá lo imprescindible para la migración, sino que se realizará un análisis completo de los aspectos más relevantes de su funcionamiento para tratar de lograr una mejor comprensión. Aquí no se incluirán las características sobre versiones ni las mejoras implementadas a través de la evolución del producto, pero sí se detallará la comunicación entre los elementos y la configuración a nivel de creación y distribución de ficheros.

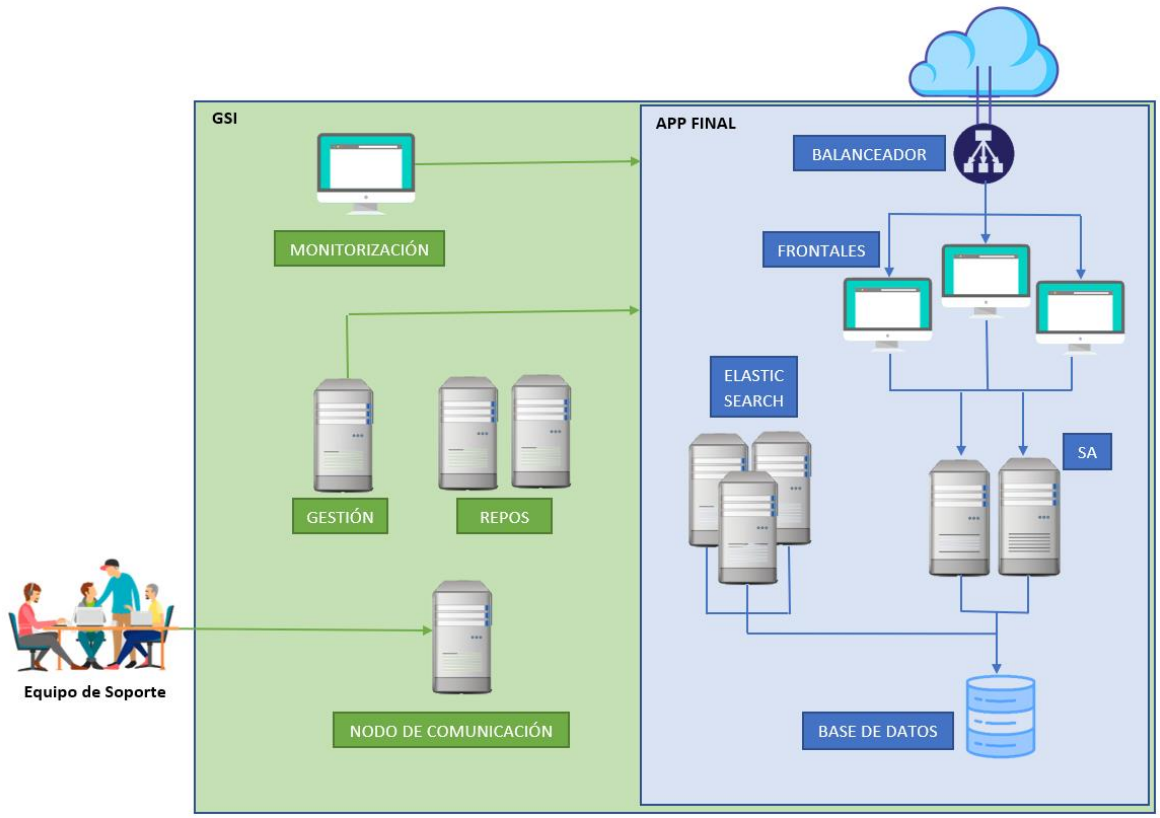

Figura 5.2 Composición de los equipos involucrados en la migración.

En la figura 5.2 vemos una representación de los equipos que tienen algún tipo de relación con APP FINAL, tanto los que conforman la aplicación en cualquiera de las tres capas, como los nodos de gestión y almacenamiento empleados para mantener un correcto estado del sistema. La parte derecha de la ilustración, con color de fondo azul muestra todos los equipos pertenecientes a la aplicación que dará soporte a los portales de acceso de La Agencia de la Salud y las Enfermedades.

De arriba a abajo, en la zona azul se pueden observar todos los componentes que intervienen en el procesamiento de una petición desde que entra al sistema hasta que llega a la base de datos y consulta el objeto deseado. Inicialmente, la solicitud atraviesa un balanceador de carga , que, al analizar la carga actual de cada uno de los tres frontales, enviará la solicitud al que tenga más procesos libres para atenderla. Tras pasar por el frontal, uno de los dos servidores de aplicación recogerá la petición y por último, accederá al nodo de base de datos, para posteriormente hacer el recorrido inverso hasta llegar al cliente de nuevo.

Posteriormente se explicará a fondo el funcionamiento de cada uno de ellos, y también se dedicarán unas líneas a los componentes internos de GSI necesarios para realizar la migración y mantener la nueva aplicación: un nodo destinado a monitorizar el sistema, dos que hacen la función de repositorios, un nodo de gestión y otro de comunicación. A este último puede acceder el equipo de soporte para intercambiar información con ellos.

Cabe resaltar también que el flujo de la información expuesto anteriormente se produce cuando es necesario acceder a un objeto que tiene almacenado la base de datos (como la lista de pacientes que han visitado un centro médico determinado o los facultativos que poseen convenios con diferentes compañías aseguradoras). Sin embargo, si un recurso web como una imagen o un fichero .pdf está alojado en el servidor de aplicaciones, no será necesario acceder a la base de datos para devolverlo al cliente. Es decir, que el último paso dentro de la figura 1 no se producirá.

5.2.1 Capa de presentación. Clúster con los frontales.

Quizá los frontales son el elemento de APP FINAL más similar a cualquiera de las otras aplicaciones pertenecientes al sistema de GSI. Desde que una petición sale del sistema del cliente, el primer elemento dentro de “los portales” al que llega son los balanceadores de carga. Estos, como bien se explicará en este mismo capítulo, monitorizan a tiempo real la ocupación de los frontales y les asignan las peticiones que van llegando a aquellos que poseen hilos (threads) libres para poder resolver la solicitud del cliente.

En este punto, es necesario hacer un breve inciso para conocer como son las comunicaciones internas entre todos los componentes del sistema. A pesar de estar situados en una sección destinada a los frontales, vamos a explicar como se envían las peticiones de los balanceadores a los frontales, de cada frontal a un servidor de aplicaciones y de estos, a la base de datos. El elemento que quiere trasladar la solicitud está dedicando un proceso (o hilo, proceso ligero encargado de completar una pequeña secuencia de tareas) a leer e interpretar la petición del cliente, y redirigirla al siguiente elemento en el flujo de información.

Cuando este proceso termina de leerlo, el elemento (pongamos como ejemplo el balanceador) va a intercambiar mensajes con los tres frontales que forman parte de APP FINAL, preguntándoles cuántos hilos de tipo workers (se designa de esta forma a los subprocesos destinados a resolver las solicitudes) tienen disponibles. Al recibir las respuestas de los tres nodos, evalúa a cuál de ellos debe enviar la petición, y cuando lo hace, el proceso encargado de la misma en el balanceador finaliza su cometido. El mismo hilo puede “revivir” y seguir resolviendo solicitudes, hasta un máximo que viene delimitado por configuración. Este límite irá reduciéndose en cada una de las peticiones HTTP y al llegar a 0, se creará otro hilo de tipo worker para que pueda continuar resolviendo peticiones. En caso de encontrar

los tres frontales saturados, es decir, con todos los workers ocupados resolviendo peticiones, el balanceador quedará en espera y cuando sea notificado de que un nodo ha liberado un hilo, le enviará la petición.

Este proceso se repite tanto entre el balanceador y un frontal, como entre cualquiera de los frontales y un servidor de aplicaciones o entre uno de los servidores y el nodo de base de datos, con la pequeña diferencia que en la base de datos el tratamiento de las peticiones tras terminar con una por parte de los hilos no es exactamente así.

Para comprobar el funcionamiento del balanceador de carga, se han hecho varias peticiones consecutivas del mismo objeto separadas por un minuto de tiempo. El objetivo es descargar un documento .png de los portales de acceso de la Agencia de la Salud y las Enfermedades, y comprobar en los registros de los tres frontales cuál ha sido el nodo que ha recibido la petición para comprobar el funcionamiento del balanceador. El comando lanzado se puede observar en la figura 5.3, junto a la hora de ejecución para poder compararla con los logs.

```
[2023-04-05 08:17:09]

wget https://URL_AGENCIA/documents/1339000/atragantamiento-heimlich.png/6f(...)84
```

Figura 5.3 Comprobación de funcionamiento del balanceador de carga (I)

Tras la descarga de la imagen, se comprobó cuál de los tres frontales recibió la petición mediante el comando de la figura 5.4, que permite mostrar de todo el fichero de logs únicamente las peticiones en que encuentra el comando especificado tras el “grep”.

```
less access_20230405.log | grep /documents/1339310/2119388/atragantamiento-maniobra-heimlich.png
```

Figura 5.4 Comprobación de funcionamiento del balanceador de carga (II)

Así, de un fichero de más de 180MB se consiguió extraer las figuras 5.5,5.6 y 5.7, en las que se pueden ver los registros de los tres frontales. Al aplicar un sencillo análisis, se puede observar que las primeras tres peticiones recayeron sobre el segundo frontal, que las atendió correctamente (al devolver estas un 200 OK). Sin embargo, un minuto después, el primero de los frontales fue el que dio una respuesta a la petición de descarga del objeto, al estar menos sobrecargado en ese momento puntual. En las cuatro peticiones, el tercero de los frontales no tuvo participación.

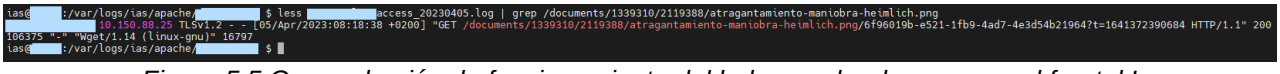

Figura 5.5 Comprobación de funcionamiento del balanceador de carga en el frontal I

Figura 5.6 Comprobación de funcionamiento del balanceador de carga en el frontal II

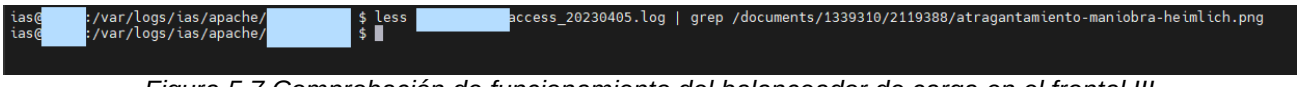

Figura 5.7 Comprobación de funcionamiento del balanceador de carga en el frontal III

De forma aclaratoria, es importante destacar que se ha eliminado el nombre real de la nueva aplicación y de la máquina en la que se alojan los frontales a petición de la empresa en las figuras 4,5,6, para tratar de mantener este documento lo más anonimizado posible. Sin embargo, en las peticiones se ve claramente que la hora coincide con la de la petición y el objeto solicitado es el mismo.

Una vez expuesta la interacción entre los elementos que componen la arquitectura de tres capas ya presentada, se procede a analizar más a fondo el frontal. Con relación al servicio prestado ante una petición de un cliente, el frontal es un componente completamente transparente, es decir, que no va a enviar respuesta alguna y prácticamente sólo redirige el mensaje al servidor de aplicaciones. De hecho, por esta función, es por lo que se podría calificar a los frontales del sistema y de APP FINAL como reverse proxies. Un proxy inverso [8], según Will Reese (2008), es un servidor que de forma transparente entrega las peticiones a otro servidor, en este caso los servidores de aplicación.

En cuanto a la configuración del elemento propiamente dicha, hay ciertos elementos que se consideran de interés. En primer lugar, hay que recalcar que los portales de acceso que se van a migrar se componen de muchas de las páginas web disponibles en La Agencia de la Salud y las Enfermedades de La Comunidad. Para ello, es necesario tener un fichero de configuración en el que se especifique el virtual host que atenderá la petición. Según IBM, un host virtual, o vhost, es una entidad de configuración que permite que una sola máquina host se asemeje a varias máquinas host. En este caso, tres frontales sostendrán más de veinte portales de acceso. Esta configuración está especificada en el fichero vhosts.conf (disponible parcialmente en la figura 5.8).

```
ias@frontal1:/opi/aplic/apache/appfinal/conf $ less vhosts/vhost_appfinal.conf
<VirtualHost *:80>
 ServerAdmin Admin1
 ServerName Name1
 RewriteEngine On
 RewriteRule (.*) https://UrlPrincipal$1
</VirtualHost>

<VirtualHost *:80>
  ServerAdmin gsi@LaAgencia.es
  ServerName Nombre_servidor
  ServerAlias Alias1
  …
  ServerAlias Alias20

  ErrorLog "|/(ruta_logs) /appfinal/appfinal_error_%Y%m%d.log 86400"
  CustomLog "|/(ruta_logs) /appfinal/appfinal_access_%Y%m%d.log 86400" combined-vhost

  DocumentRoot "/opi/applic/apache/appfinal/htdocs/appfinal"
</VirtualHost>
```

Figura 5.8 Configuración de diferentes hosts virtuales en el mismo frontal

En la figura 5.8 podemos observar tres directivas, que explicaremos a continuación siguiendo la documentación de Apache [9]. Mediante estas se escoge el virtual host que dará respuesta

a una solicitud determinada. En el caso de APPFINAL esta elección se basará en encontrar el mejor <VirtualHost> coincidente, es decir, el más específico en cuanto a dirección IP y puerto utilizados por la solicitud. En caso de haber más de un host virtual que contenga la mejor combinación de dirección y puerto, se compararán las directivas ServerName y ServerAlias con el nombre del servidor presente en la solicitud. Además de Server Alias, ServerName y <VirtualHost>, otra directiva importante es DocumentRoot. Esta especifica en qué parte del sistema de archivos se puede encontrar el contenido del host seleccionado.

La configuración explicada anteriormente es muy importante en los frontales, ya que permite conocer donde se encuentran los ficheros estáticos correspondientes a la capa de presentación. Realmente, la mayoría de estáticos, como documentos, imágenes o forms están en el servidor, pero siguiendo la ruta especificada en DocumentRoot podemos encontrar objetos como el index.html o el logo de la nueva aplicación. Se puede comprobar en la figura 5.9.

```
ias@frontal1:/opi/aplic/apache/appfinal/htdocs/appfinal $ ls -lhrt
-rw-r----- 1 ias dba 14K Feb 15 09:04 logo.jpg
-rw-r----- 1 ias dba 361 Feb 15 09:04 index.html
(…)
ias@frontal1:/opi/aplic/apache/ appfinal /htdocs/ appfinal $
```

Figura 5.9 Ubicación de ficheros estáticos en los frontales

Por último, cabe destacar el más importante de los ficheros ubicados en el frontal. Este es sin duda el access_log, que contiene un registro de todas las peticiones que atiende el frontal en un día determinado. Las figuras 4, 5 y 6, mostradas para explicar el funcionamiento del balanceador de carga anteriormente, han sido el resultado de un análisis del fichero access.log. Las peticiones tienen un formato similar, y en ellas se puede ver el método HTTP seleccionado, el objeto al que va relacionado y datos del cliente como el modelo de dispositivo en el que realiza la búsqueda o el navegador que ha empleado para realizar la búsqueda.

5.2.2 Capa de negocio. Clúster con los servidores. Liferay y Elasticsearch

Tras los frontales, una solicitud del usuario llega a los servidores de aplicación. Son estos los que contienen la parte lógica y devolverán al usuario el objeto que requiera. En caso de ser un fichero, un documento pdf, o una imagen almacenada en ellos, no tendrá la necesidad de hacer una consulta a la base de datos. Sin embargo, si el cliente desea, por ejemplo, un listado de los medicamentos que han prescrito en el último mes, el servidor de aplicaciones que esté dando servicio a esa solicitud deberá acceder a la base de datos PostgreSQL y, mediante una petición, recibirá los datos necesarios que podrá devolver al frontal, y finalmente llegarán al cliente.

A diferencia de los frontales, donde sí resulta de sumo interés tener un registro de cada una de las solicitudes que reciben, el equipo de soporte de la aplicación no desea configurar los servidores de aplicación para mantener ese mismo fichero de acceso. Esto resulta lógico, puesto que más allá de ver cuál de los servidores recibe cada una de las peticiones, no aportaría una información extra, ya que las mismas peticiones que pasan por los frontales

deben llegar a los servidores para ser resueltas. Debido a esto, no se podrá observar en un fichero de log el paso entre componentes de la petición realizada anteriormente.

Al igual que se explicó en el punto 4.2.1 de esta memoria, la gestión entre los componentes para enviarse una solicitud se hace mediante un pool de threads, un conjunto de hilos en el frontal espera encontrar "workers" (otro conjunto de hilos dentro de un servidor de aplicación) libres, y cuando los encuentra se les asigna la petición correspondiente.

Servidor de portales. Liferay DXP

En la capa del servidor de aplicaciones hay un componente que todavía no se ha mencionado en este capítulo y que supone una mejora entre las tres aplicaciones previas a la migración y APP FINAL. Se trata de Liferay DXP [10], un portal de gestión de contenidos de código abierto escrito en Java, orientado a la construcción de sitios Web, portales verticales, intranets, etc. Está basado en los estándares JSR-168, JSR-286, JSR-170 y CMIS. Se ha escogido Liferay DXP, ya que es uno de los servidores de portales más destacados, y permite una gran adaptabilidad al ser un contenedor de portlets. Un portlet es un módulo web reutilizable que proporciona acceso a contenido, aplicaciones y otros recursos en la web que puede ejecutarse en un servidor de aplicaciones.

El funcionamiento de un portlet es muy simple: procesa peticiones y genera respuestas, en las que devuelve contenido (por ejemplo, HTML, XHTML) para su visualización en los navegadores. De las muchas diferencias que tiene con respecto a otro tipo de aplicaciones web como los servlets vamos a destacar dos. Por un lado, un portlet se ejecuta en una parte de la página web, por lo que el código es independiente al del resto de la página, y un servlet renderiza una página web completa. Por otro lado, los portlets se ejecutan sobre un servidor de portales (en este caso Liferay DXP), por lo que pueden beneficiarse del soporte que este servidor les aporta en la gestión de usuarios, permisos, sesiones, gestión de páginas y puede centrarse en la funcionalidad principal. Esto supone que el desarrollo de un portlet sea más sencillo que una aplicación web independiente. [11]

Desde el punto de vista del usuario, no habrá una notable diferencia a la hora de ver el contenido, ya que un portlet no será más que una ventana en un sitio del portal que proporciona información a un servicio específico. A nivel de desarrollo, son módulos web conectables diseñados para ejecutarse dentro de un contenedor de Portlets, función que en este caso hace Liferay DXP. Sin embargo, no entraremos más a nivel de desarrollo, puesto que el objetivo de este proyecto es realizar una migración y no desarrollar uno o varios portlets.

Dentro de los servidores de portales, Liferay DXP es uno de los más punteros. Comparado con su principal competidor, Apache Pluto [12], incorpora más portlets con más funcionalidades, lo que lo hace más completo y adaptable a la hora de realizar el diseño del portal de acceso. Liferay incluye más de 60 portlets que ofrecen funcionalidades diferentes, como foros, blogs, wiki, calendario, encuestas, anuncios, integración de contenido de sistemas externos… que permiten construir el sitio web de la forma que se desee.

Para poder dar los servicios implementados en los diferentes portlets, es necesario montar Liferay DXP sobre un servidor de aplicaciones, en este caso un JBOSS, del que hablaremos posteriormente. Dentro del servidor, hay un directorio en el que se recogen diariamente los logs de Liferay, ficheros en los que podemos encontrar los errores existentes en alguna de las peticiones para informar al equipo de soporte de la aplicación. Un ejemplo de los errores que se pueden ver en el fichero de log de liferay aparece en la figura 9, en este caso ante la pérdida de conectividad con la base de datos.

Ejemplo de error en logs de Liferay DXP:

```
2023-03-27 11:48:23.531 ERROR [default task-1168][SiteNavigationMenuEditPortletToolbarContributor:171] Unable to set edit site navigation menu to menu item

com.liferay.portal.kernel.exception.SystemException: java.sql.SQLException: javax.resource.ResourceException: IJ000453: Unable to get managed connection for java:jboss/datasources/appfinal

Caused by: java.sql.SQLException: javax.resource.ResourceException: IJ000453: Unable to get managed connection for java:jboss/datasources/appfinal
```

Figura 5.10 Fragmento de fichero de log de Liferay DXP

En la figura 5.10 se muestra un fragmento de uno de esos ficheros del log del Liferay DXP tras la instalación en APPFINAL, con el objetivo de comprobar algunas de las tareas que desempeña el servidor de portales. En este caso se observan errores en la gestión de conexiones con la base de datos, pero también son frecuentes los problemas en la resolución de peticiones o a la hora de construir un menú de navegación.

Servidor de aplicaciones JBoss

Dejando de lado el servidor de portales Liferay DXP, este componente es necesario construirlo sobre un servidor de aplicaciones, en este caso un Jboss [13], servidor web escrito en Java que facilita a las empresas la gestión y la administración de los entornos web de cualquier tamaño. Para ver la configuración del servidor, dentro de APP FINAL podemos encontrar un fichero server.log, en el que están definidas varias de sus propiedades dentro del nodo en el que se encuentran (nombre del servidor, ruta del directorio destinado a almacenar los logs, directorio desde el que se realizan los despliegues o el directorio base).

Además de registros sobre peticiones, o ficheros de propiedades, dentro de los servidores de aplicación destacan también los ficheros estáticos como documentos o imágenes. En muchas otras aplicaciones estos son visibles dentro del servidor, pero en APP FINAL aparecen encriptados, por lo que se ha decidido no mostrar su ubicación en el sistema, ya que verlos encriptados no aportaría información de utilidad.

Elastic Search como motor de búsqueda y recolector de datos.

Como ya comentamos al principio de este capítulo, en APP FINAL se ha decidido incorporar Elastic Search como nuevo componente, cuyo cometido era la recolección de datos de la aplicación entre otras muchas funciones. Además de proporcionar información acerca de los accesos realizados, los errores y guardar el estado del sistema ante cualquier fallo,

Elasticsearch [14] es un motor de búsqueda basado en Lucene [15] que va de la mano con Liferay DXP desde la versión 7.0, que permite explotar al máximo las capacidades del servidor de portales.

Antes de la versión 7.0 de Liferay DXP, se usaba el motor de búsqueda Lucene, que se ejecutaba en la misma JVM del portal. Con la evolución tecnológica, y la aparición del Big Data, cada vez los resultados de una búsqueda son más numerosos y Elastic Search ha permitido a Liferay implementar un sistema de búsqueda externo basado en texto capaz de descargar al portal de esa tarea, agilizando así el comportamiento del servidor de portales.

5.2.3 Capa de datos. Nodo con la base de datos PostgreSQL.

El sistema de base de datos elegido para APPFINAL es PostgreSQL, un potente sistema de base de datos relacional por objetos de código abierto que utiliza y amplía el lenguaje SQL combinado con numerosas funciones, que almacenan y escalan con seguridad las cargas de trabajo de datos más complicadas. Además, permite definir sus propios tipos de datos, crear funciones personalizadas e incluso escribir código de diferentes lenguajes de programación sin tener que recompilar su base de datos.

La base de datos es consultada mediante peticiones desde fuera del sistema pasando por los balanceadores de carga, frontales y servidores de aplicación. Sin embargo, la modificación de sus tablas, alteración de permisos de usuarios y tareas que puedan implicar un cambio en alguno de sus datos la deben realizar técnicos dentro del equipo de GSI. Al ser un elemento crucial en el sistema, y no poder tolerar un fallo que causara una pérdida en el servicio, el equipo dispone de tres bases de datos con los mismos datos (anonimizados) en las que se ejecutan las consultas SQL que solicita el equipo de la aplicación para alterar algún elemento de la misma.

En este estudio no es posible mostrar los elementos internos de la base de datos, como usuarios, tablas, restricciones y demás información, ya que son confidenciales. Sin embargo, en la figura 5.11 se detalla el patrón de lanzamiento de una consulta SQL (lenguaje de computación para trabajar con conjuntos de datos y las relaciones entre ellos).

Ejemplo de consulta en la base de datos:

```
psql -d database -e -f consulta.sql -U usuario > consulta.log 2> consulta.err
```

Figura 5.11 Ejemplo de ejecución de fichero .sql sobre la base de datos

Mediante la opción -d se especifica la base de datos, -f indica el fichero que se lanzará contra la bdd indicada, que debe estar en el mismo directorio de lanzamiento, con el -U asociamos un usuario con sus permisos a la ejecución y finalmente indicamos donde se sitúan los ficheros creados al finalizar la ejecución, tanto el .log como el .err, donde irán los errores.

5.2.4 Elementos externos

Además de estos componentes, es importante conocer otros elementos del sistema, externos a la aplicación presentes en el sistema informático de la Agencia de la Salud y las Enfermedades de La Comunidad. Estos permiten realizar conexiones a la base de datos, revisar el estado de todos los nodos y almacenar las diferentes versiones de la aplicación, así como realizar despliegues y enviar y recibir ficheros al equipo de soporte de la aplicación de forma segura.

Nodo de comunicación

En primer lugar, el equipo de técnicos de 'La Comunidad' posee un nodo, que denominaremos Nodo de Comunicación al cuál también tiene acceso el equipo de soporte de Portales. Es ahí donde el equipo de PORTALES dejará cualquier fichero que implique una interacción con su aplicación, desde accesos a la base de datos y ejecuciones de SQLs hasta nuevos ficheros de configuración con cambios en redirecciones de URLs, modificación de contraseñas para accesos limitados o cualquier elemento presente en la aplicación.

A su vez, en esa máquina existe un directorio llamado 'docs' en el cual se comparten los ficheros que se requiera en cada caso. Un ejemplo muy recurrente de su uso es la compartición de ficheros de log. El equipo de PORTALES puede solicitar conocer que ha ocurrido en el sistema, al informarle de la detección de Stall Counts en el mismo. En ese caso, se puede extraer el fichero de log del servidor de aplicación correspondiente para enviarlo a la aplicación y que puedan revisar dónde está el error para poder modificar el método que lo causa. Simplemente, habría que extraerlo del nodo que hace de SA y depositarlo en 'docs', dentro del nodo de comunicación.

Repositorios de propiedades y versiones

En segundo lugar, hay otros dos nodos que también contendrán ficheros relacionados con Portales. Estos son los repositorios de propiedades y de versiones. En el primero de ellos, se almacena una copia tanto de las propiedades actuales que tienen el servidor de aplicaciones y el frontal, como de cada una de las modificaciones realizadas desde la primera versión. Esto permite revertir los cambios al modificar un fichero y detectar un error y tener un backup para no perder la configuración en caso de desastre y pérdida de la misma en el nodo que da servicio.

En el segundo no se almacenan propiedades, sino las versiones de la aplicación que han sido desplegadas, como su propio nombre indica. Cada una de ellas contiene los ficheros estáticos correspondientes, ya sean del frontal o el servidor, una o varias consultas que modifican la base de datos y un fichero EAR. Un EAR (Archivo de empresa) es un archivo comprimido que contiene las bibliotecas, los enterprise beans (componente JAVA que se puede combinar con otros recursos para crear aplicaciones JAVA) y los archivos JAR que la aplicación necesita para su despliegue.

Máquinas de gestión

Una vez expuestas algunas de las máquinas externas a los portales que tienen influencia directa en el día a día de la aplicación, es necesario destacar el medio de transferencia de

ficheros entre el nodo de comunicación, los repositorios de propiedades y versiones y los propios nodos que contienen los componentes de la aplicación. Se trata de las máquinas de gestión, donde se modifican los ficheros y se aplican los cambios necesarios para no sobrecargar los nodos que deben dar un servicio eficiente en PRODUCCIÓN.

Entre todas ellas hay una especial. Además de funcionar como intermediaria entre las diferentes aplicaciones, posee scripts para agilizar tareas del trabajo diario. Desde ella se podrían revisar si hay sesiones bloqueantes en la base de datos de APP FINAL, comprobar el estado de ocupación del servidor de aplicaciones y subir los ficheros de una versión determinada almacenados en el repositorio de versiones al frontal que toque. En conclusión, gracias a las máquinas de gestión se pueden realizar cambios de forma simple y transparente en varios de los componentes que forman parte de la aplicación.

Máquinas de monitorización

En la parte final de esta memoria, se dedicará un pequeño espacio a configurar las alertas que la nueva aplicación tendrá en Introscope, una herramienta introducida anteriormente que permite monitorizar el estado de APP FINAL y el resto de las aplicaciones del equipo de GSI. Gracias a dos máquinas que tienen instalada esta herramienta se puede saber prácticamente de forma inmediata el estado de ocupación, bloqueo, riesgo de caída por GC o sesiones activas, entre otras cosas. En este sentido, estas dos máquinas encargadas de la monitorización también tendrán una participación activa en la nueva aplicación.

Cabe recalcar que los nodos nombrados en esta subsección no formarán parte de la migración. Funcionan correctamente mientras coexisten APP1, APP2 y APP3, y en el futuro deberán hacer lo mismo cuando se introduzca APP FINAL. Esto es así, porque no dependen de una u otra aplicación, sino que son elementos externos a ellas que todas usan de una forma u otra, y tienen un gran peso e importancia para permitir que el sistema funcione correctamente. De ahí que se haya considerado importante dedicarles un fragmento dentro de este capítulo.

5.3 Tecnología utilizada

Tras explicar la funcionalidad y configuración de los componentes que formarán APP FINAL, se dedicará este espacio a analizar las distintas versiones de estos y disertar un poco más acerca de ellos. Inicialmente, se expondrá el por qué se han elegido unas versiones determinadas y no otras para los distintos elementos, y en la parte final se analizarán las principales diferencias y mejoras que se podrán obtener al actualizar los componentes de los frontales, servidores y base de datos, tras la inclusión del Liferay DXP como servidor de portales y Elastic Search.

Para un equipo como GSI, las garantías en los componentes escogidos en una migración como esta constituyen quizá el aspecto más importante. Si las versiones de los componentes de APP FINAL iban a mantenerse tras la migración por una década, la prioridad para el equipo fue saber cuáles de las versiones ofrecidas por los componentes iban a tener un equipo de soporte asociado durante ese periodo.

En este sentido, la piedra angular en la decisión fueron los servidores de portales Liferay DXP. Tras una conversación con el soporte de la herramienta, se acordó la compra de la licencia de la misma para usarla en APP FINAL siempre y cuando la versión que se instaurara en los servidores tuviera personal disponible para ayudar, en cualquier caso. Por otro lado, a la hora de escoger la versión de un componente, no siempre la más reciente es la mejor elección. En muchos casos, las versiones que han visto la luz hace pocos meses poseen ciertos bugs o errores que se van descubriendo con su uso. Por esto mismo, la elección de una versión un poco menos actualizada, pero más estable y que dé más garantías, suele ser la opción escogida en GSI, ya que no se podría asumir un cambio repentino en la arquitectura del sistema por un bug descubierto mientras está en ejecución.

Desde Liferay DXP comunicaron que la versión que se debía implantar para contar con apoyo y soporte durante un largo periodo de tiempo era el Liferay DXP 7.4, disponible a partir de octubre de 2021. A pesar de ser compatible con la mayoría de los servidores de aplicaciones (WebLogic, Jboss, Geronimo…) y bases de datos (MySQL, Oracle, SQL Server, PostgresSQL), desde el equipo de Liferay comunicaron cuáles de ellos podían encajar mejor con la versión escogida.

En la primera aplicación que soportó los portales de acceso de la Agencia de la Salud y las Enfermedades, APP1, la solución elegida en cuanto al servidor de aplicaciones fue un Apache Tomcat 2.0 [16]. Con la evolución de la tecnología y el aumento en la carga de peticiones que tenían que soportar, este Tomcat se cambió por un JBOSS en APP2 y APP3. Desde el equipo de Liferay comunicaron a GSI que se podía montar un Jboss, en la línea de lo que se venía haciendo en las aplicaciones anteriores, pero en este caso la mejor versión debía ser un Red Hat JBoss Enterprise Application Platform 7.4 [17] (que salió a mediados de 2021).

En relación con la base de datos, desde el equipo de soporte indicaron que se podía seguir con PostgreSQL, que era el producto que había en APP1 y APP2, pero se decidió actualizar a la versión 14. [18] En la nueva aplicación, se debían juntar las bases de datos de las tres aplicaciones previas en una única máquina. Siguiendo la misma política a la hora de escoger, la versión que se implementará en los frontales es el HTTP Apache 2.2. Como se ha indicado en este mismo capítulo, al actualizar el Liferay por encima de la versión 7.0, se consideró necesario incluir un motor de búsqueda externo, que ha sido Elastic Search, conectado al servidor Jboss y empleado también para la recolección de datos, y que se actualizará de la versión 5.5 desplegada en APP3 a la 7.14.1 en APP FINAL.

6. Desarrollo e implantación de la solución

En este capítulo se profundizará en la migración de APP1, APP2 y APP3 a APP FINAL. Es importante destacar que, debido a la importancia de que los portales de acceso se mantengan disponibles, desde el equipo de GSI se ha decidido no eliminar ni modificar ninguna de las tres aplicaciones previas hasta varios meses después de completar la migración. En caso de detectar un problema en producción al configurar los nuevos portales, sería necesario redireccionar el balanceador de carga y el reverse proxy para que apuntase a los frontales de las aplicaciones previas, de manera que mientras se solucionaran los problemas en APP FINAL, estas pudieran mantener el servicio activo. Por ello, en este capítulo no se va a incluir el vaciado y apagado de las máquinas que sostenían los portales de acceso antes de migrar las aplicaciones, ya que supondría un desastre hacerlo y arriesgarse a que no funcionara APP FINAL.

Antes de exponer paso a paso el trabajo realizado, es importante recalcar que en la estructura de GSI hay tres entornos a disposición del equipo para tratar de llegar con un producto perfectamente depurado y optimizado al entorno de producción. En TEST y PREPRODUCCIÓN, nombres de los entornos previos, se detectarán los fallos en cada uno de los componentes, solicitando al equipo de soporte las modificaciones necesarias en ellos para lograr que tras desplegar APP FINAL en producción no tengan que sufrir modificaciones más drásticas. Todos los pasos descritos en este capítulo se han llevado a cabo en cada uno de los tres entornos mencionados, pero para evitar la repetición innecesaria de información en esta memoria, únicamente se establecerá una diferencia entre ellos a la hora de desplegar la aplicación. Esto es debido a que, como se verá en la sección específica, un despliegue de estas características en producción es realmente diferente al mismo en los demás entornos.

La Agencia de la Salud y las Enfermedades posee diversos equipos que participan en la migración de los portales de acceso. La creación y actualización de los componentes, su despliegue y las acciones más importantes las lleva a cabo el equipo de GSI, y esas son las que se reflejarán en este capítulo, mientras que hay otras tareas como el montaje de los equipos o la programación de la propia aplicación las realizan otras entidades que también trabajan para La Agencia. En este sentido, para poder comenzar con la migración, se solicitó al grupo encargado de administración de los equipos la creación de las máquinas sobre las que se configurarán los diversos componentes. Así, se requirió la creación de una máquina sobre la que estuviera instalada la versión 14 de PostgreSQL, dos para crear sobre ellas los servidores de aplicación; tres más, que serían sobre las que se instalaría Elasticsearch; y otras tres para crear sobre ellas los frontales.

Una vez se han expuesto los tres entornos utilizados para asegurarse de que la aplicación queda correctamente desplegada en producción y analizados los equipos de los que dispone GSI para la configuración de la nueva aplicación, ya es posible empezar a desarrollar la migración en sí misma. La migración, que se divide en los ocho pasos mostrados a continuación. Dentro de cada uno se ejecutarán pruebas para comprobar que es posible avanzar al siguiente paso, haciéndose también pruebas de carga previas al despliegue y evaluando el uso por parte de los usuarios tras el mismo.

6.1 Análisis de Riesgos

El primer paso de la migración consiste en la creación de una base de datos PostgreSQL 14. Como se ha indicado en la introducción de este capítulo, se parte de una máquina ya creada, database81, con el paquete de PostgreSQL 14 ya instalado. Esta sección se dedicará a la creación y configuración de la base de datos, así como la creación de la estructura de datos necesaria para realizar la migración de datos de la aplicación APP3 a APP FINAL.

Inicialmente, como se puede ver en la figura 6.1, se creará una variable de entorno postgres.env. Las variables se utilizan para controlar el comportamiento predeterminado de un entorno de bases de datos, incluido el comportamiento de autorización, migración tras error y red. Para ello, se crea un fichero postgres.env definiendo diferentes variables para localizar en *PATH* el fichero de inicialización de la base de datos, en *LD_LIBRARY_PATH* las librerías que necesita PostgreSQL y en *PGDATA* el directorio en el que irán los ficheros que constituyen la base de datos. La última línea de la figura 6.1 establece el valor que se ha otorgado a las diferentes variables, y para ello usa el comando *export VAR_1 VAR_2* siendo VAR_N las variables empleadas.

```
[postgres] database81:/export/home/ias $ less postgres.env

PS1="[postgres] `hostname`:"'$PWD $ '

PATH=/usr/pgsql-14/bin:$PATH

MANPATH=/usr/pgsql-14/share/man:$MANPATH

LD_LIBRARY_PATH=/usr/pgsql-14/lib:/usr/lib:/usr/local/lib

PGDATA=/datos/db/pg_system14

export PGDATA LD_LIBRARY_PATH MANPATH PATH
```

Figura 6.1. Creación de la variable de entorno postgres.env

A continuación, se planteó como optimizar el arranque y la parada de la base de datos de la forma más rápida y eficiente posible, permitiendo hacer reinicios simplemente mediante la ejecución de un comando dentro de database81. Para ello, se ha creado un script postgres.sh, que además de las funcionalidades mencionadas, permite consultar el estatus de la bdd, recargarla o reiniciarla. El script, admite un parámetro, que puede ser una de las siguientes cadenas de caracteres: start, stop, restart, reload o estatus y al ser ejecutado, aprovecha la utilidad de PostgreSQL pg_ctl para actuar sobre el elemento de datos y efectuar la opción escogida sobre este. En la figura 6.2 se facilita un fragmento de ese script, en concreto la parada de la base de datos.

```
case "$1" in

(…) stop)

        echo -e "\n"

        /usr/pgsql-14/bin/pg_ctl -D /datos/db/pg_system14 -m fast stop

        echo -e "\n"

        ;;
```

Figura 6.2. Fragmento de la parada de la BDD en el script postgres.sh

Una vez se creó el script, se comprobó que funcionaba parando, arrancando y reiniciando la base de datos, así como consultando el estatus. En tercer lugar, se creó la base de datos tal y como indica la documentación oficial de PostgreSQL. [19] Para ello, usa el comando *createdb altdb,* ya que *altdb* fue el nombre seleccionado. A continuación, se comprobó que la base de datos estaba correctamente creada y se procedió a acceder.

Posteriormente se creó el directorio donde se ubicarán los ficheros de log, en los que la base de datos volcará los errores en las ejecuciones de las consultas y los detalles más importantes. Dado que el GSI está en continuo contacto con el equipo de la aplicación, y para llenar lo menos posible database81, se decidió almacenar únicamente los registros de los últimos siete días. Para ello, se generaron siete ficheros con el nombre de los días de la semana, siendo el primero postgresql-Lunes.log y así sucesivamente.

En este momento, una vez configurado el nodo de database81 (creando los directorios de log, estableciendo la variable de entorno y la ubicación de base de datos y creando el script), se procedió a crear diversos roles en la base de datos. Para esta tarea se siguió el manual de administración de PostgreSQL, que incluye las principales actividades de administración sobre una base de datos de este tipo. Dado que el equipo de APP FINAL es quien ejecutará las consultas y actualizaciones en la base de datos, se crearon los tres roles que ellos solicitaron. Además de *us_ptl*, *us_ptl_operativo* y *us_ptl_url* se creó un usuario *ias* al que se le otorgaron los permisos de súper usuario, que permiten crear roles y crear ficheros dentro de la misma base de datos.

A los roles *us_ptl_operativo* y *us_ptl_url* no se les quiere permitir la conexión a la base de datos. Para ello fue necesario establecer una conexión como súper usuario a la base de datos y alterar sus permisos. La figura 6.3 muestra cómo se han realizado algunas de las modificaciones mencionadas, con un comentario asociado. Además, en la parte inferior de la misma se comprueba la lista de usuarios donde se puede ver la correcta creación y asignación de permisos mencionada anteriormente.

```
altdb=# create role us_ptl_url login password ****; --creación del rol us_ptl_url
altdb=# alter role ias createrole; --se otorga permiso para crear roles a ias.
altdb=# alter role us_ptl_url nologin; --no se permite la conexión a us_ptl_url


-- Comprobación de cambios en la lista de roles.
                          List of roles
     Role name      |               Attributes                          | Member of
--------------------+---------------------------------------------------+-----------
 ias                | Superuser, Create role, Create DB                 | {}
us_ptl_url          | Cannot login                                      | {}
```

Figura 6.3 Algunas modificaciones en la creación de roles en la base de datos

A continuación, se creó el esquema, entidad en base de datos que contendría las tablas y datos de APP FINAL. A la hora de crearlo, fue necesario especificar la entidad de seguridad en base de datos que poseerá el esquema, que en este caso sería el rol us_ptl. Después, se trató de comprobar que con el usuario us_ptl era posible conectarse a la base de datos, y con us*_ptl_operativo* y *us_ptl_url no.* Para nuestra sorpresa, no era posible establecer una conexión con el *rol us_ptl*, devolviendo la base de datos el siguiente error: *"psql: error: FATAL: Peer authentication failed for user 'us_ptl'".*

Tras investigar la fuente del error en diversas páginas web, se observó que era necesario permitir que desde el fichero de configuración de la base de datos que el usuario *us_ptl* pueda conectarse a ella. Para ello, se accedió al fichero *pg_hba.conf.* Este fichero, que controla la autenticación de los usuarios, se inicializa al arrancar la base de datos y está almacenado en el directorio de datos del clúster de bdd.

Según la documentación de PostgreSQL, el formato general del archivo pg_hba.conf es un conjunto de registros, uno por línea. Cada registro especifica un tipo de conexión, un rango de direcciones IP de cliente (si es relevante para el tipo de conexión), un nombre de base de datos, un nombre de usuario y el método de autenticación que se utilizará para las conexiones que coincidan con estos parámetros. El primer registro que coincida con el tipo de conexión, la dirección del cliente, la base de datos solicitada y el nombre de usuario se utiliza para realizar la autenticación. [20]

Una vez se analizó cómo funcionaba la autenticación en una base de datos PostgreSQL, se decidió permitir al usuario ias (recuerden que tenía permisos de súper usuario) entrar en la base de datos una vez se había autenticado al entrar en database81. Sin embargo, al usuario *us_ptl*, se le permitirá la entrada solicitando una autenticación previa. Los cambios aplicados al fichero pg_hba.conf por defecto se muestran en la figura 6.4.

```
[postgres] database81:/datos/db/pg_system14 $ diff pg_hba.conf pg_hba.conf.original

< local   all             ias                                     trust

< local   all             us_ptl                                md5
```

Figura 6.4. Modificaciones en el fichero pg_hba.conf (I)

Tras la creación de los usuarios (roles) en la base de datos y el esquema asociado a ellos, se procedió a la creación de los tablespaces. Un tablespace define el espacio lógico que pertenece a un usuario y que contiene todas las tablas de su esquema de datos. En el espacio físico, el tablespace está formado por ficheros que componen la base de datos, llamados datafiles. Para poder crear el tablespace, se crea un directorio que contendrá los datafiles, otorgándole los permisos necesarios.

Se siguió la documentación de PostgreSQL para crear un nuevo tablespace llamado *tb_ptl*, especificando la ubicación de los datafiles. Posteriormente, hubo que otorgar permisos al usuario que quisiéramos emplear para modificar este espacio de datos. Se eligieron los usuarios *ias* y *us_ptl.* En la figura 6.5, se puede ver cómo se creó el tablespace y los permisos de los usuarios *ias* y *us_ptl* sobre este objeto.

```
altdb=# create tablespace tb_ptl location '/datos/db/pgdatos/ptl';

CREATE TABLESPACE
altdb=# \db+
(…) – se otorgan los permisos
List of tablespaces

   Name      |  Owner   |   Location   |  Access privileges  | Options |  Size   | Description

-------------+----------+--------------+---------------------+---------+---------+-------------

   tb_ptl    | ias      | /datos/…/ptl | ias=C/ias      +    |         | 0 bytes |
   -                                   | us_ptl=C/ias        |         |         |

   (…) -- tablespaces creados por defecto al inicializar la bdd
```

Figura 6.5. Modificaciones en el fichero pg_hba.conf (II)

Tras la creación y configuración de la base de datos, se logró establecer una estructura inicial y los roles necesarios para realizar sobre ella la migración de la base de datos de APP3 a APP FINAL. Cabe destacar que, para poder trasladar la base de datos era necesario configurar el esquema, los ficheros de datos y los usuarios en database81.

6.2 Configuración y preparación de los SAs y el Elasticsearch.

Tras la configuración de la base de datos, el siguiente paso de la migración fue configurar los servidores de aplicación y los nodos en los que se ubicaría el Elasticsearch. Las máquinas en las que irían ubicados los servidores de aplicación con el Liferay DXP 7.4 incorporado recibirían el alias de server1 y server2, y las que tuvieran el Elastic configurado serían elastic1, elastic2 y elastic3.

Antes de entrar en configuración, es importante destacar el estado de las máquinas cuando llegaron al equipo de GSI. El entorno que se recibió para los servidores de aplicación estaba compuesto por dos máquinas con el servidor Red Hat Jboss EAP 7.4 con el entorno de ejecución de Java 11. [21] En los SAs era necesario instalar el servidor de portales Liferay DXP 7.4 sobre el Jboss y modificar la configuración estableciendo variables de entorno, propiedades, descriptores y eliminando configuraciones que vienen por defecto al realizar la instalación de JBOSS y Liferay.

Por otro lado, en los tres nodos de Elasticsearch fue necesario configurarlo en clúster para aprovechar la funcionalidad de replicación en clúster, incluida en Elastic en la versión 6.7.0, que permite replicar los datos de un nodo a otro. Esta evolución aporta una serie de ventajas como permitir una alta disponibilidad o una mayor capacidad de recuperación ante un desastre.

6.2.1 Instalación y configuración de Liferay.

Para la instalación de Liferay DXP 7.4 en los servidores de aplicaciones, se facilitó al equipo de GSI un archivo con extensión .war. Según indica la documentación de Liferay, este archivo .war contiene la versión de Liferay seleccionada. Por ello, para desplegar este producto hubo que descomprimirlo y ubicarlo en la ruta *'JBOSS_HOME'/appfinal/deployment/ROOT.war/* en los tres nodos.

Tras ubicar correctamente el nuevo archivo y antes de desplegarlo, fue necesario copiar los archivos de dependencia para Liferay y todos los plugins al cargador de clases del servidor de aplicaciones global en el servidor de producción, en el Jboss. Para instalar los plugins necesarios, el equipo de appfinal dejó una carpeta comprimida en el nodo de comunicación denominada *Dependencias de OSGi.* Fue necesario copiar ese archivo .zip y descomprimirlo en el directorio *<Liferay-home>/osgi.* En el caso de appfinal, el directorio en el cual se ubicaron estas dependencias es *"/opt/app/jboss/osgi/marketplace".* En la figura 6.6, podemos ver cómo quedó el directorio tras descomprimir el .war.

```
[jboss_eap7.4] ias@server1:/opt/app/jboss/osgi/marketplace $ ls -lhrt

total 753M
-rw-r--r-- 1 ias dba  39K Aug  2 14:53 Liferay Collaboration - Translation - API.lpkg
-rw-r--r-- 1 ias dba 123K Aug  2 14:53 Liferay Collaboration - Subscription - Impl.lpkg
-rw-r--r-- 1 ias dba  24K Aug  2 14:53 Liferay Collaboration - Subscription - API.lpkg
(…)
```

Figura 6.6. Directorio tras descomprimir las dependencias de Liferay DXP 7.4

En tercer lugar, es necesario asegurar que hay conectividad desde los nodos del servidor de aplicaciones donde tenemos el Jboss y Liferay al nodo donde se ubica el datasource. Un datasource o fuente de datos es un objeto que representa un repositorio de datos, en este caso database81. Para conseguir comunicar el servidor de aplicaciones con la base de datos se tuvo que instalar dentro del JBOSS el controlador de PostgreSQL. En vista de esto, se procedió a copiar el .jar en un nuevo directorio, donde según la documentación se debe situar el controlador de una base de datos PostgreSQL. La ruta es:

/JBOSS_HOME/modules/org/postgres/main. En la figura 6.7 se facilita el directorio mencionado.

```
[jboss_eap7.4] ias@server1:/JBOSS_HOME/modules/org/postgres/main $ ls

module.xml postgresql-42.3.4.jar
```

Figura 6.7. Directorio donde se ubica el controlador

El otro fichero presente en el directorio representado en la figura 6.7 es el archivo *module.xml*. Un módulo es una colección empaquetada de clases y otros recursos, junto con la especificación de lo que este módulo importa y exporta de/a otros módulos. Jboss identifica diferentes tipos de módulos, a veces llamados módulos estáticos y dinámicos. Los módulos estáticos están predefinidos en el directorio EAP_HOME/modules/ del servidor de aplicaciones. Cada subdirectorio representa un módulo y define un subdirectorio main/ que contiene un archivo de configuración (module.xml) y cualquier archivo JAR necesario. En el module.xml que se va a modificar, se añadirán dependencias, como se ve en la figura 6.8.

```
[jboss_eap7.4] ias@server1:/JBOSS_HOME/modules/org/postgres/main $ less module.xml

<?xml version="1.0"?>
<module xmlns="urn:jboss:module:1.0" name="com.liferay.portal">
<resources> <resource-root path="postgresql.jar" /> </resources>
<dependencies>
<module name="javax.api" /> <module name="javax.mail.api" /> <module
name="javax.servlet.api"/> - <module name="javax.servlet.jsp.api" /> <module
name="javax.transaction.api" />
</dependencies>
```

Figura 6.8. Archivo module.xml donde se definen las dependencias con la base de datos

Gracias a la configuración del directorio module, se le permitiría al Jboss utilizar el controlador de PostgreSQL para poder conectarse a la base de datos. A continuación, se modificó el fichero de configuración de la nueva aplicación *appfinal.xml*, añadiendo toda la información que necesita la aplicación para poder, a través de jboss y el driver que acabamos de instalar, abrir conexiones contra la base de datos. Dentro de *appfinal.xml*, el primer apartado que se modificó fue el de datasource, insertando los datos que se han configurado al crear la base de datos (nombre de BD en el servidor de nombres de dominio (DNS), el puerto abierto para la conexión y el nombre de la base de datos).

De esta forma, estaremos indicando a la aplicación cuál es su fuente de datos, y a qué host y mediante qué puerto deberá realizar las consultas cuando necesite acceder a ella. Por otro lado, fue necesario indicar el nombre del controlador que habíamos configurado anteriormente, ya que, mediante ese driver, el usuario propietario de los esquemas de la aplicación podrá realizar las consultas que desee. Para agilizar la comunicación y permitir

que la conexión se haga de forma automática, se introduce también la contraseña del usuario propietario (en este caso *us_ptl*), para que pueda abrir las conexiones de forma automática.

```
<datasources>
	<datasource jndi-name="java:jboss/datasources/appfinal" (…) >
		<connection-url>jdbc:postgresql://database81:pto/lportal</connection-url>
		<security>
			<user-name>us_ptl</user-name>
			<password>****</password>
		</security>
	</datasource>
	<drivers>
		<drivername="postgresql" module="com.liferay.portal (…) </driver>
	</drivers>
</datasources>
```

Figura 6.9. Fragmento del fichero de configuración appfinal.xml

Para evitar mostrar el fichero completo, ya que contiene bastantes líneas, se ha seleccionado en la figura 6.9 lo más significativo. La cadena que define el datasource es la que está dentro de la etiqueta <connection-url>, siendo database81 el nombre DNS del datasource y el puerto 5432 el que se mantiene activo esperando la comunicación por parte del Jboss. Mediante las etiquetas <security>, se define el usuario y la contraseña de conexión y en <drivers>, se ubica el nombre del controlador asociado, además de otros datos que no se han considerado de sumo interés.

Por último, hubo que modificar el fichero appfinal-ext.properties. Para realizar cualquier operación sobre base de datos desde el lenguaje Java, se usa la API JDBC (Java Database Connectivity). Según la documentación de Liferay, con el objetivo de proporcionar un mecanismo para que los programas Java se conecten a las bases de datos, es necesario definir fuente de datos JDBC, ya que, en caso contrario, appfinal intentará crear un origen de datos JDBC. [22] La variable *jdbc.default.jndi.name* es la que era necesario modificar, y se le asignó el valor del datasource creado anteriormente *java:Jboss_HOME/datasources/appfinal.*

6.2.2 Cambio de configuración en el JBoss.

Además de los cambios realizados en los ficheros anteriores, en el Jboss de cada uno de los servidores de aplicación fue necesario hacer algunos cambios para acoger la nueva aplicación APPFINAL. En esta sección se tratará el establecimiento de variables de entorno, la especificación de propiedades y descriptores y la eliminación de configuraciones por defecto al instalar JBOSS y Liferay innecesarias.

En primer lugar, fue necesario crear variables de entorno, al igual que se hizo en database81. El uso de estas no solo permite al sistema operativo conocer qué versión de Java se usará por defecto y su ubicación dentro del sistema de directorios del servidor, sino que además permitirá conocer la versión del Jboss instalada y utilizada en APPFINAL. Por ello, se configuró un fichero jboss_eap7.4.env, del que se puede ver un fragmento en la figura 6.10. Destacan las variables JAVA_HOME y JBOSS_HOME que especifican las rutas donde está instalado el JDK (en este caso la versión 11) y el Jboss EAP 7.4 respectivamente. El resto de las variables son comunes a las explicadas en la creación de la variable entorno que se creó en la base de datos; PS1 indica el cambio de la consola añadiendo la cadena de la variable antes de la ruta actual, y el export para establecer el valor de las diferentes variables.

```
[jboss_eap7.4] ias@server1:/export/home/ias $ less jboss_eap7.4.env
JAVA_HOME=/opt/app/java/jdk-11.0/
JBOSS_HOME=/opt/app/jboss/jboss-eap-7.4/
PS1="[jboss_eap7.4] `logname`@`hostname`:"'$PWD $ '
(…)
export JAVA_HOME JBOSS_HOME JBOSS_USER JBOSS_CONSOLE_LOG PATH PS1
```

Figura 6.10. Creación de variable de entorno jboss_eap7.4 en el Jboss

A continuación, se especificarán propiedades y descriptores para tener el Jboss y Liferay configurados y funcionando en clúster atendiendo a las necesidades definidas por el equipo de appfinal para poder acoger su aplicación. Para configurar el Jboss y que pueda correr sobre él Liferay DXP, se deberá modificar el *standalone.xml.*

Antes de entrar en los parámetros modificados, es necesario indicar que a partir de la versión 6 de Jboss, Red Hat incorporó dos modos de trabajo, standalone y domain. Standalone es el modo de trabajo donde tenemos una instancia de servidor independiente trabajando en un host, donde la configuración y el despliegue de servicios y aplicaciones es único por instancia. En el modo domain habrá un controlador del dominio. Se identifican unos procesos JVM llamados Host-Controllers que son responsables de leer la configuración dada por el Domain-Controller y transferirla a las instancias de servidor que gestionen. De ese modo, se logra que esas instancias de servidor tengan configurados los mismos servicios y desplegadas las mismas aplicaciones.

Para que el Jboss permita ejecutar sobre él Liferay DXP, la documentación de Liferay mostró que se debían realizar algunos cambios en el fichero *standalone.xml,* que se renombró por *appfinal.xml*. [23]

En primer lugar, se configurará el contenedor de portlets, modificando el elemento que describe el funcionamiento del contenedor *<jsp-config>* estableciendo los atributos *development, source y* target-vm. Según la documentación de Jboss, el primero de ellos indica si está activado el modo desarrollo, el segundo la versión de JDK con la que son compatibles los ficheros con el código fuente, y el tercero la versión de JDK con la que son compatibles los ficheros de clases. [24] En la figura 6.11 se puede ver la configuración añadida.

```
<jsp-config development="true" source-vm="1.8" target-vm="1.8" />
```

Figura 6.11 Configuración del contenedor de portlets

En segundo lugar, para asegurarse de que el servidor de aplicaciones recibe los parámetros de cualquier solicitud con la codificación UTF-8 hay que insertar dos propiedades en el fichero appfinal.xml. [25] Mediante estas, mostradas en la figura 6.12, se consigue un mecanismo para traducir caracteres no imprimibles o especiales a un formato universalmente aceptado por servidores web y navegadores, en este caso el formato UTF-8. Son necesarias dos, porque la primera de ellas hace referencia a la codificación de la URI (Identificador de Recursos Uniforme), y la segunda a la codificación de los caracteres presentes en el cuerpo de la solicitud.

```
<system-properties>
    <property name="org.apache.catalina.connector.URI_ENCODING" value="UTF-8" />
    <property   name="org.apache.catalina.connector.USE_BODY_ENCODING_FOR_QUERY_STRING"
value="true" />
</system-properties>
```

Figura 6.12. Configuración de la codificación de caracteres en appfinal.xml

En tercer lugar, es necesario establecer un filtro en los ficheros de log. Esto es necesario, porque hay dos avisos que, de no eliminar, aparecerían continuamente y no permitirían ver los registros relevantes que informen sobre algún error en el Jboss. El primero de ellos, bajo el código *WFLYSRV0059,* indica que las dependencias del recurso .jar no están en la ruta por defecto. [26] El segundo, WFLYEE0007, indica que hay components opcionales que no se han instalado, pero en ningún caso son críticos en el correcto funcionamiento del Jboss. En la figura 6.13 se puede ver como se filtra el fichero de log mostrando los mensajes que no coincidan con ninguno de esos errores.

```
<filter-spec value="not(any(match(&amp;quot;WFLYSRV0059&amp;quot;),match(&amp;quot;WFLYEE0007&amp;quot;)))" />
```

Figura 6.13. Filtrado de los logs en el fichero appfinal.xml

En cuarto lugar, dentro del appfinal.xml, hay un elemento importante que es necesario configurar en este punto, el *deployment scanner*. Su función es monitorizar un directorio en busca de nuevos archivos y desplegarlos en el servidor. Usando la documentación de Jboss, se consideró necesario definir ciertos atributos en el elemento que define el escáner de despliegues: el directorio que va a monitorizar, usando las variables *relative-to* (path completo del directorio a escanear) y *path* (nombre de la carpeta objetivo); la frecuencia de revisión del directorio en busca de nuevos ficheros para ser desplegados, mediante *scan-interval*, y el tiempo máximo que puede tomar un despliegue antes de ser cancelado (*deployment-timeout)*. Esta configuración es visible en la figura 6.14

```
<deployment-scanner   deployment-timeout="600"   path="deployments"   relative-
to="jboss.server.base.dir"     scan-interval="5000"     runtime-failure-causes-
rollback="${jboss.deployment.scanner.rollback.on.failure:false}"/>
```

Figura 6.14. Configuración del escáner de despliegues

Por último, fue necesario agregar el dominio de seguridad JAAS de Liferay. Java Authentication and Authorization Service (JAAS) es una API de seguridad que consiste en un conjunto de paquetes Java diseñados para la autenticación y autorización de usuarios. [27] El dominio de seguridad se basa en uno o más módulos de login (login modules), que autentifican y/o autorizan al usuario actual contra un determinado almacén (security datastore) que contiene los passwords y/o roles de los usuarios. En este caso, el único módulo de login que se añadiría al sistema de seguridad de appfinal sería el de Liferay, como se puede ver en la figura 6.15.

```
<security-domain name="PortalRealm">
    <authentication>
        <login-module code="com.liferay.portal.security.jaas.PortalLoginModule"
flag="required" />
    </authentication>
</security-domain>
```

Figura 6.15. Módulo de seguridad JAAS de Liferay

Adicionalmente, se eliminaron los elementos y mensajes de bienvenida del servidor, ya que se considera configuración innecesaria que viene por defecto cuando se realiza la instalación de JBOSS y Liferay.

6.2.3 Configuración de la Java Virtual Machine (JVM).

Tras la modificación del fichero appfinal.xml, es necesario revisar el script de configuración de arranque *appfinal.conf*, para modificar ciertos parámetros de la máquina virtual Java (JVM). Fue necesario identificar la variable *USER_JAVA_OPTS* y adaptar los parámetros de memoria a la memoria disponible en el entorno de PRO. Inicialmente, se quisieron poner 16 GB, pero desde el equipo de APP FINAL comunicaron que no era posible, debían ser como máximo 8 GB, por lo que se adoptó esta opción.

Las modificaciones que se realizan respecto al fichero que viene por defecto son la especificación de la memoria predeterminada (como se ha comentado anteriormente), el establecimiento de la codificación en UTF-8, de la zona horaria del usuario y de la pila de protocolos preferida. La etiqueta *-Djava.net.preferIPv4Stack* en este caso permite sólo conexiones que usen el protocolo IPv4. El JDK iniciado con esa opción puede conectarse a hosts IPv4-only e IPv4-or-IPv6, no puede conectarse a hosts IPv6-only. Otras propiedades que se pueden ver en la figura 6.16 junto con las mencionadas son la ruta del fichero de propiedades, en el que se configuran las conexiones del servidor al exterior, o la localización y contraseña del almacén de certificados de confianza (o *TrustStore*).

```
ias@server1 # less appfinal.conf
# parámetros de memoria expresados en MEGAS
MEM="12288"
# parámetros de usuario
USER_JAVA_OPTS="(…) -Djava.net.preferIPv4Stack=true
-Dexternal-properties=/etc/app/conf/appfinal/appfinal/appfinal-ext.properties
-Djavax.net.ssl.trustStore=/etc/app/cert/almacenesCA/cacerts
-Djavax.net.ssl.trustStorePassword=***
-Dlog4j2.formatMsgNoLookups=true
-Dfile.encoding=UTF-8 -Djava.locale.providers=JRE,COMPAT,CLDR -Duser.timezone=Europe/Madrid
-Djgroups.bind_addr=server1fvi01  -Djgroups.tcpping.initial_hosts=  server1fvi01  [port1],
server2fvi01 [port2] -Djboss.as.management.blocking.timeout=900"
(END)
```

Figura 6.16 Configuración JVM modificando el script de arranque

6.2.4 Configuraciones varias. Appfinal-ext.properties y document_library

Además de los documentos ya mencionados, fue necesario realizar modificaciones en otros dos ficheros. Hubo que añadir algunas líneas al portal-ext.properties de Liferay, uno de los archivos que más información nos proporciona de las capacidades de Liferay. En él se ubican diversas configuraciones como el pool de conexiones, el datasource donde se conectará el servidor, las rutas de los controladores (o drivers) de los diversos sistemas de bases de datos que pueden conectarse al SA, el protocolo ligero de acceso a directorios (LDAP), etc.

Como se puede comprobar en la figura 6.17, el primer elemento que se incorporó al fichero fue la ruta del datasource. A continuación, se configuró Liferay DXP 7.4 para que requiera que las nuevas cuentas verifiquen sus direcciones de correo electrónico a través del fichero portal.property y se especificó el path del fichero *home* de Liferay y del directorio al cual, mediante Liferay, se depliegan los elementos previamente situados en el */deploy*, el */osgi*. Por último, se modificó la zona horaria, el protocolo https y se añadió la posibilidad de hacer redirecciones a URLs siempre y cuando se encontraran dentro del dominio de La Agencia de la Salud y las Enfermedades, *domain.es.*

```
jdbc.default.jndi.name=java:jboss/datasources/appfinal          # path del datasource.
company.security.strangers.verify=false                         # No verificar cuentas.
liferay.home=/JBOSS_HOME/appfinal/liferay  # Path de liferay.home
module.framework.base.dir=/opt/app/jboss/osgi                   # Path del directorio osgi
web.server.protocol=https
company.default.time.zone=Europe/Paris
company.default.locale=es_ES
redirect.url.security.mode=domain                # Permitir redirecciones en el dominio
redirect.url.domains.allowed=*.agencia.es        # Permitir redirecciones en *.domain.es
```

Figura 6.17. Líneas añadidas al appfinal-ext properties

Por defecto, Liferay DXP utiliza una opción de almacén de biblioteca de documentos llamada Simple File Store para almacenar documentos y archivos multimedia en un sistema de archivos (local o montado). La carpeta raíz predeterminada del almacén es [Liferay Home]/data/document_library. Se trata de es una implementación simple de almacenamiento de archivos utilizando una carpeta local. . Así, dentro de */osgi/configs* se creará un fichero en

el que se defina la document_library de la siguiente forma: *rootDir="/datos/global/app/appfinal/document_library.* Al tratarse de un entorno en el que hay más de un servidor de aplicaciones, esta ruta deberá ser accesible desde todos los nodos.

6.2.5 Configuración de Elastic en clúster

Appfinal posee tres máquinas sobre las que está instalado Elasticsearch, que en este estudio recibirán los sobrenombres de elastic1, elastic2 y elastic3, deben ser configuradas individualmente para poder ser enlazadas posteriormente a los servidores de aplicación y Liferay DXP. Cabe recordar que elastic es un motor de búsqueda que va de la mano con Liferay DXP, y para permitir que operen correctamente es necesario establecer una vinculación entre el servidor en el que está instalado el servidor de portales y el motor de búsqueda.

Una vez se tuvo Elasticsearch instalado, desde el equipo de appfinal situaron en el nodo de comunicación los plugins necesarios para la configuración de elastic en Liferay (complementos al código de la aplicación con el que se añade la funcionalidad mencionada) y se copiaron las carpetas comprimidas a */opt/app/elasticsearch/elasticsearch-7.14.0/bin/ .* Desde ahí, se instalaron los plugins como se puede ver en la figura 6.18.

```
bin/elasticsearch-plugin install file:/path/…/analysis-icu-7.14.0.zip.
```

Figura 6.18. Instalación de plugins de Liferay en uno de los nodos de elastic

Elasticsearch utiliza un almacenamiento de memoria del tipo mmapfs, almacenando el índice de fragmentos en el sistema de archivos mapeando un archivo en memoria (mmap). El mapeo de memoria utiliza una porción del espacio de direcciones de memoria virtual en el proceso igual al tamaño del archivo que está siendo mapeado. Es probable que los límites predeterminados del sistema operativo en los recuentos mmap sean demasiado bajos, lo que puede dar lugar a excepciones de falta de memoria. Por ello, fue necesario realizar una ampliación de la memoria virtual usada por Elasticsearch añadiendo la siguiente línea *vm.max_map_count= 262144* al fichero /etc/*sysctl.conf.*

Por otro lado, un descriptor de fichero es un número entero sin signo utilizado por un proceso para identificar un fichero abierto. Elasticsearch utiliza muchos de ellos y quedarse sin *file descriptors* puede ser desastroso y conducir a la pérdida de datos. Por ello, se realizó una modificación respecto a la configuración establecida por defecto, aumentando a 65535 el número de descriptores de archivos abiertos.

6.3 Enlazado de los SAs y Liferay a Elasticsearch

En esta sección se describirá el proceso llevado a cabo para lograr configurar Elastic con los entornos de Liferay. Inicialmente, se expondrá la configuración que es necesaria en cada uno de los tres definiendo el concepto de nodo maestro y su importancia en elastic. A continuación, se mostrará la modificación llevada a cabo en el fichero ElasticsearchConfiguration.config de los servidores para añadir las tres direcciones IP de los

nodos de Elastic. De este modo, será posible que Liferay DXP 7.4 se comunique con cualquiera de los tres nodos desde uno de los servidores de aplicación.

En appfinal hay tres instancias de Elasticsearch, y cada una corresponde a un nodo. Un conjunto de nodos conectados se denomina clúster, y en la aplicación destino de la migración este estará formado por tres nodos, todos arrancados al mismo tiempo. Cada uno conoce al resto y puede reenviar las peticiones de los clientes al nodo apropiado. Además, un nodo puede cumplir diferentes roles, que se asignarán en el fichero *elasticsearch.yml* (como se mostrará a continuación), entre los que destacan el nodo maestro, de datos, de ingestión, de aprendizaje automático (en caso de usar las características de machine learning), de transformación… En este caso, inicialmente se permitirá a los tres asumir el rol de nodo maestro, siendo responsable el primero que arranca de las acciones ligeras en todo el clúster, como la creación o eliminación de un índice, el seguimiento de los nodos que forman parte del clúster y la decisión de qué fragmentos asignar a qué nodos. Además, los nodos maestros deben tener un directorio *path.data* cuyo contenido persista a través de los reinicios, al igual que los nodos de datos, porque aquí es donde se almacenan los metadatos del clúster. Los metadatos del clúster describen cómo leer los datos almacenados en las máquinas que contienen los datos, por lo que, si se pierden los metadatos almacenados, los nodos de datos no se pueden leer. [28]

En los tres nodos del clúster se accedió a /opt/app/elasticsearch/elasticsearch-7.14.0/config/elasticsearch.yml y se modificó el fichero para ajustar las direcciones IP y los nombres de los nodos al entorno de PRO. Además, se especificó el nodo que realizaría las funciones de maestro y el número de nodos del clúster sin contar en el que se realiza la configuración, mediante la propiedad *discovery.zen.minimum_master_nodes.* Esta configuración se puede ver en la figura 6.19.

```
[elastic] ias@elastic1:/path/elasticsearch-7.14.0/config $ less elasticsearch.yml
# Direcciones IP de los nodos en el entorno de PRO
discovery.zen.ping.unicast.hosts: ["10.172.2.1", "10.172.2.2", "10.172.2.3"]
#
# Especificación de los nodos maestros (inicialmente todos)
cluster.initial_master_nodes: ["elastic1", "elastic2", "elastic3"]
discovery.zen.minimum_master_nodes: 2
```

Figura 6.19. Configuración de los nodos de Elasticsearch mediante el elasticsearch.yml

Posteriormente, en cada uno de los servidores de aplicaciones sobre los que estaba instalado Liferay DXP 7.4, fue necesaria la configuración de Elasticsearch. Este proceso se basa en indicar a server1 y server2 la ubicación de elastic1, elastic2 y elastic3, mediante su dirección IP y el número del puerto que tienen abierto para realizar dichas comunicaciones. En *com.liferay.portal.search.elasticsearch7.configuration.ElasticsearchConfiguration.config,* fichero ubicado en la ruta <*liferay_home*>*/osgi/config/*, fue necesario sustituir las IPs y asignar un nombre al clúster. Además, mediante la línea *indexNamePrefix="liferay-"*, establece un valor de cadena que se utilizará como prefijo para el nombre del índice de búsqueda en Elastic. Esta nueva configuración dentro de uno de los servidores es visible en la figura 6.20.

```
less com.liferay.portal.search.elasticsearch7.configuration.ElasticsearchConfiguration.config
networkHostAddresses=[\"http://10.172.2.1:9200", \ "http://10.172.2.2:9200",\ "http://10.172.2.3:9200"\]
indexNamePrefix="liferay-"
clusterName="LiferayElasticsearchCluster"
com.liferay.portal.search.elasticsearch7.configuration.ElasticsearchConfiguration.config (END)
```

Figura 6.20 Configuración de los nodos de Elasticsearch desde el SA

6.4 Prueba de arranque de los Jboss (SAs) y los Elastic

El objetivo de esta sección es realizar una prueba de arranque de los servidores y de los tres nodos que contienen instalado el Elasticsearch. En ella se mostrarán los cambios realizados en tres de los ficheros alojados en los SAs para configurar los servidores de aplicación en clúster, y por último, se levantarán server1, server2 y los tres nodos del Elastic, pero sin contener información de la aplicación ni poblar la base de datos. Esta comprobación final resulta útil para asegurar que los componentes ya configurados funcionan correctamente.

Liferay DXP puede servir desde los sitios web más pequeños hasta los más grandes. De fábrica, está configurado de forma óptima para un entorno de servidor único. Si un servidor no es suficiente para atender las necesidades de alto tráfico de una aplicación en un momento determinado, es adaptable más de uno. Por ello, la instalación de Liferay DXP 7.4 en un entorno clúster es igual que en un entorno donde únicamente hay un nodo. En el caso de appfinal, una vez instalados por separado Elastic y Liferay en modo clúster, se podrán configurar en este caso utilizando el modo Unicast. Los tres ficheros que se modificarán son *appfinal.conf, tcp.xml y appfinal-ext.properties.*

Modificación de appfinal.conf

En este fichero, únicamente hay que añadir los nodos que forman el clúster en los parámetros JVM del arranque en todos los nodos en los que se ubique Liferay DXP 7.4, es decir, en server1 y server2. Además de esto, se indicó en cuál de los nodos del clúster se está editando el fichero, especificando el nodo actual. Para las dos modificaciones, mostradas en la figura 6.21, se emplearon los elementos XML *Djgroups.bind_addr y Djgroups.tcpping.initial_hosts.* Estos parámetros hacen referencia al framework *JGroups,* que proporciona servicios que permiten la comunicación entre nodos del clúster. Está construido sobre una pila de protocolos de comunicación de red que proporcionan servicios de transporte, descubrimiento, fiabilidad y detección de fallos, y gestión de miembros del clúster. [29]

```
# Parámetros de usuario
-Djgroups.bind_addr=server1   #nodo en el que se aplica la configuración
-Djgroups.tcpping.initial_hosts=server1fvi01[7800],server2fvi01[7800]" #todos los SAs
```

Figura 6.21 Modificación en appfinal.conf para la configuración en clúster

Modificación de appfinal.ext-properties

Al igual que en el fichero appfinal.conf, será necesario modificar el fichero appfinal.ext-portales en los dos servidores de aplicación. A la configuración realizada en la sección anterior, mostrada en la figura 6.17, se añadirán cuatro líneas (representadas en la figura 6.22). En este caso, con la propiedad *cluster.link* se habilitó la caché distribuida entre los diferentes nodos Liferay DXP. Fue necesario también añadir en esta sección la ruta de los ficheros xml de configuración del clúster unicast, que se analizarán a continuación.

```
cluster.link.autodetect.address=10.172.7.22:9200
cluster.link.enabled=true
cluster.link.channel.properties.control=/etc/app/conf/appfinal/appfinal/tcp.xml
cluster.link.channel.properties.transport.0=/etc/app/conf/appfinal/appfinal/tcp.xml
```

Figura 6.22 Modificación en appfinal-ext.properties para la configuración en clúster

Modificación de tcp.xml

Este fichero es el que contiene la configuración real de los nodos del clúster. Para que Liferay lo identificase correctamente, se situó en cada uno de los servidores en la ruta especificada en el fichero appfinal-ext.properties. Dentro de la configuración completa, se destacarán algunas de las líneas más interesantes para tratar de explicar cómo afectan en el funcionamiento del servidor de portales. Al ser muy extenso, se van a comentar las líneas más relevantes de la configuración, pero no se mostrará completo en este documento.

En primer lugar, se debe cambiar el puerto en el que se comunica el descubrimiento en todos los nodos que no sean el primero, para evitar la colisión de puertos TCP, de ahí que aparezca la línea siguiente *bind_port="7800"*. En segundo lugar, los nodos del clúster irán comprobando si el resto 'están vivos', o si por cualquier razón hubieran caído. Esto se configura en la línea *<FD timeout="3000" max_tries="3" />,* donde se especifican tres segundos entre cada prueba de vida y un total de tres intentos fallidos para detectar una pérdida del servicio en el otro nodo.

Otra de las líneas interesantes es esta: *<pbcast.NAKACK2 use_mcast_xmit="false" discard_delivered_msgs="true"/>*. El protocolo NAKACK se utiliza para los mensajes multidifusión. En este protocolo, cada mensaje se etiqueta con un número de secuencia. El receptor hace un seguimiento de los números de secuencia y entrega los mensajes en orden. Cuando se detecta un hueco en el número de secuencia, el receptor pide al emisor que retransmita el mensaje que falta. El protocolo NAKACK se configura como el subelemento pbcast.NAKACK. Mediante el primero de los atributos, se especifica que el emisor no debe retransmitir el mensaje a todos los nodos del clúster en caso de no llegar correctamente a su destino, y mediante el segundo, se aprueba el descarte de los mensajes que llegan fuera de orden. También se hace uso de pbcast.STABLE y pbcast.GMS, para gestionar la recogida de basura (mensajes ya vistos por todos los miembros) y las altas o bajas de algunos miembros en la estructura en clúster.

Una vez configurado el modo clúster en los servidores de aplicación, fue el momento de realizar un borrado de los ficheros temporales de los dos nodos de liferay y la carpeta */opt/app/elasticsearch/elasticsearch-7.17.2/data/nodes* de los nodos en los que se ubica Elasticsearch. Al realizar una limpieza de los ficheros temporales el administrador se asegura de que no quedan ocupando un espacio de disco innecesario, y de que al iniciar la instancia de Elastic o Liferay no arranca sobre ficheros anteriores con elementos posiblemente desactualizados.

Prueba de arranque

Para arrancar correctamente todos los componentes, se iniciaron primero los nodos de Elasticsearch, comprobando en los logs de arranque que aparecía un cambio de estado en el clúster. La línea "*[elastic1] Cluster health status changed from [RED] to [GREEN]*" indicó que el nodo actual había arrancado correctamente. Al terminar con los tres, se realizó una comprobación mediante la consola del estado del clúster, visible en la figura 6.23.

```
[INFO] [elastic1] Cluster health status changed from [RED] to [GREEN] (…)
```

Figura 6.23 Comprobación del inicio en cluster de los Elastic

Tras comprobar que el inicio de Elastic ha sido satisfactorio, se procedió a arrancar los tres nodos que contenían el servidor de portales Liferay DXP 7.4, mediante el script destinado a ello. En los logs de arranque de los Jboss se encuentra la línea mostrada en la figura 6.24, en la que se pueden ver los dos servidores entre sí, lo que indica que se han arrancado los nodos correctamente en clúster.

```
16:32:41,897 INFO [stdout] (SCR Component Actor) 2023-04-07 16:32:41.896 INFO  [SCR Component Actor][JGroupsReceiver:98] Accepted view [server2-4857|1] (2) [server1-4857, server2-4035]
```

Figura 6.24 Comprobación del inicio en clúster de los servidores de aplicación

De forma adicional, se ha accedido a la URL de uno de los servidores sobre el que está Liferay DXP incorporado. Al comprobar que la imagen de la sección inferior derecha carga correctamente, se confirma que el arranque ha sido correcto. Además, se hizo un login mediante el usuario de pruebas que se indica en la configuración de Liferay, y se comprueba que permite el acceso al portal web. Esta imagen aclarativa se puede observar en la figura 6.25

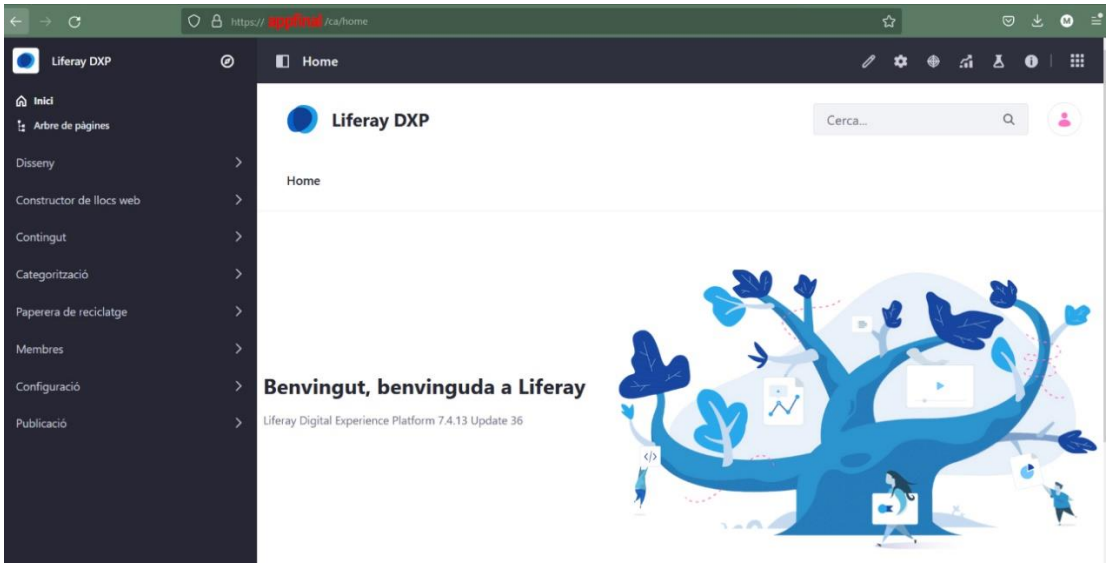

Figura 6.25 Comprobación gráfica del inicio de los servidores de aplicación

En la sección de configuración/búsquedas, se pudo comprobar también de forma gráfica el correcto arranque de Liferay. Para ello, únicamente fue necesario comprobar el estado "GREEN", que indica que arrancó correctamente, y que se identificaban los tres nodos dentro del clúster. Por decisión de la empresa, se omitirá el nombre de los nodos en los que se ha alojado Elastic modificándolos también en la captura de pantalla representada en la figura 6.26 de ahí que "elastic1, elastic2 y elastic3" aparezcan con otra tipografía en la imagen (como appfinal en la URL).

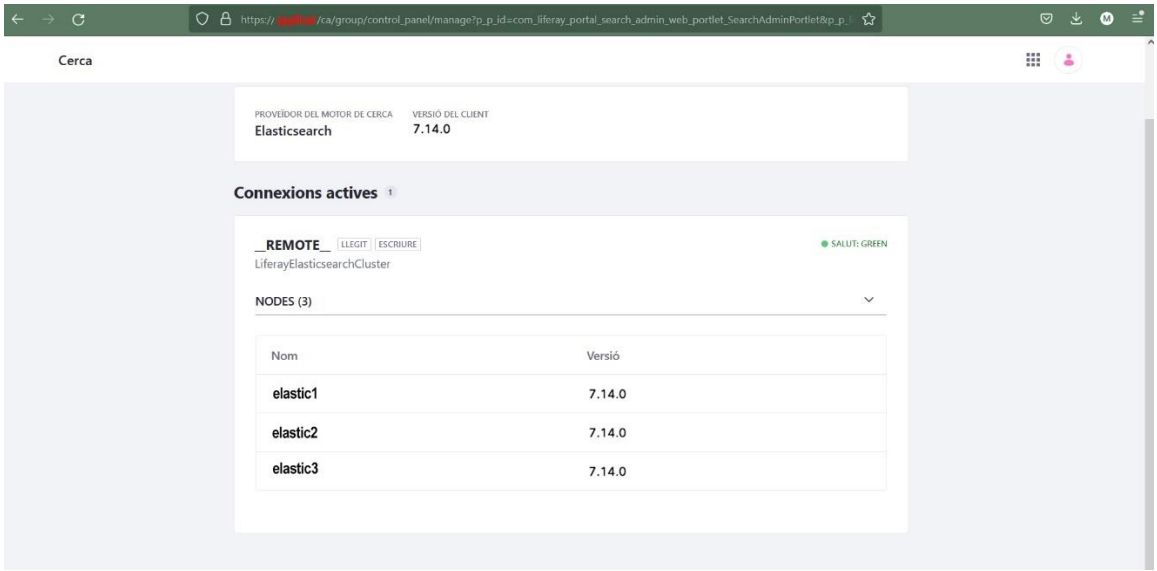

Figura 6.26 Comprobación gráfica del inicio de los servidores de Elastic

6.5 Upgrade de Liferay

Como parte de la preparación de datos para el nuevo entorno, se deben copiar los datos de la base de datos y la carpeta donde se almacenan las imágenes y documentos de los contenidos, la biblioteca de documentos (document library). Tras el montaje de database81, nodo de base de datos, en este mismo documento se relató cómo se crearon los usuarios *us_ptl*, *us_ptl_operativo* y *us_ptl_url*, la variable de entorno, se otorgaron permisos, y se creó una base de datos inicial para comprobar la conexión.

En esta sección se describirán las diferentes tareas que hubo que realizar para lanzar un proceso de upgrade de base de datos para la actualización de Liferay DXP de la versión 7.1 a 7.4. Inicialmente, se parará el Jboss mediante un script y se ejecutará otro para preparar la base de datos de APP3 y ser migrada a APP FINAL. Posteriormente, se copiará la biblioteca de documentos y se realizará la copia de base de datos borrando los esquemas que se crearon inicialmente en la creación de la base de datos. Finalmente, se preparará la base de datos para el upgrade, se realizará el mismo mediante un script y se realizarán algunas tareas post-upgrade.

En primer lugar, se deben parar los servidores para todo el proceso, acción para la que se usará un script kill_appfinal situado en el directorio bin del Jboss. Al tratarse de un entorno configurado en clúster, fue necesario parar ambos nodos (server1 y server2) para no modificar nada mientras alguien puede estar realizando pruebas, pero fue suficiente con lanzar el proceso de Upgrade de Liferay en uno de ellos.

Antes de realizar una copia de la base de datos de APP3 para importarla en la base de datos de APP FINAL, se lanzó un script con el objetivo de limpiar registros huérfanos en la base de datos. Un registro huérfano es aquel que hace referencia a otro registro que no existe como, por ejemplo, un registro de un pedido que hace referencia a un registro de un cliente que ha sido borrado de la base de datos. La idea era llevar a la nueva aplicación la base de datos con el espacio necesario, pero sin filas innecesarias en la tabla.

En ese momento, una vez hecha la limpieza, fue momento de realizar la copia de la base de datos de APP3 para el upgrade. La vieja versión se debió restaurar en el mismo esquema borrando previamente la base de datos creada en el momento de la instalación del entorno en APPFINAL.

Copia de la base de datos para el upgrade

Para lograr una copia de la base de datos de APP3, con el objetivo de trasladarla a la nueva aplicación, se utilizó la utilidad *pg_dump* que facilita PostgreSQL. Según la documentación oficial, este comando extrae una base de datos PostgreSQL en un fichero script u otro fichero de archivo, haciendo una copia de seguridad consistente a pesar de estar en uso. A continuación, en esa copia de la base de datos, resultó necesario modificar todas las apariciones de la cadena ‘us_app3’ por ‘us_ptl’, usuario creado en la nueva base de datos que poseerá acceso a los objetos existentes en database81. Para ello, se empleó la sentencia mostrada en la figura 6.27, haciendo uso de la sentencia Sed. La expresión básica de Sed es sed 's/busca/reemplazo/X' y lo que hace es buscar "busca" y reemplazarlo con "reemplazo"

utilizando los flags descritos en X. Si este flag es g, quiere decir que lo haga tantas veces como pueda.

```
/*Dump de la base de datos sobre app3 en export_app3_20230212*/
ias@bdapp3 # nohup pg_dump -d db -n us_app3 > export_app3_20230212.sql &
/*Reemplazo del usuario us_app3 por us_ptl*/
ias@bdapp3 # sed 's/us_app3/us_ptl/g' export_app3_20230212.sql > export_appfinal_20230212.sql
```

Figura 6.27. Copia de la base de datos de APP3 (I)

Tras esto, en database81 se borró el esquema asociado al usuario us_ptl, creado en el inicio de la base de datos para realizar pruebas. A continuación, se realiza la importación de la copia de la base de datos de APP3, con el usuario modificado (mediante el fichero export_appfinal_20230212.sql), y por último se cambió la password del usuario us_ptl, ya que, en caso de no hacerlo, en el fichero mencionado tendrá la contraseña de us_app3.

Copia del document library para el upgrade

Además de realizar una importación de la base de datos de APP3, Liferay funciona de una manera particular. Es necesario también copiar la biblioteca de documentos, o *document library*, que está en el servidor de aplicaciones y proporciona un mecanismo para almacenar archivos en línea utilizando el mismo tipo de estructura que se utiliza para almacenar archivos localmente. [30] Para guardar la consistencia de los datos, se debe realizar este proceso al mismo tiempo que el export/import de la base de datos descrita anteriormente. Para realizar esta copia, se usó el comando scp document_library_20230212.tar.gz ias@server1:/datos/global/app/appfinal/ desde la máquina de gestión, y una vez ubicado en el servidor de aplicaciones de appfinal, se descomprimió ubicándolo en el directorio correspondiente.

Actualización de Liferay

Una vez copiada la base de datos e importada la biblioteca de documentos, la fase de preparación de los datos había finalizado. El siguiente paso consistía en la actualización de Liferay a la última versión, en este caso la ‘u36’, acción para la que se emplearía la herramienta de Liferay Database Upgrade Tool, un programa cliente para actualizar bases de datos de Liferay DXP y Liferay Portal offline. Para actualizar Liferay DXP se realizaron una serie de pasos, con el objetivo de eliminar y copiar las carpetas del núcleo de Liferay DXP y borrar los ficheros temporales.

Los tres objetos que se renovaron fueron el directorio /osgi/Marketplace, el fichero ROOT.war y la herramienta necesaria para las actualizaciones mencionada anteriormente, Portal Tool DB Upgrade. Para ello, se eliminaron los mismos de su ubicación actual en los dos servidores de appfinal, se movió del nodo de comunicación la nueva versión facilitada por el equipo de la aplicación y se descomprimieron los archivos en el directorio en el que estaban anteriormente. Por último, se eliminó el contenido de cuatro directorios donde se almacena información temporal, que puede contener ficheros muy grandes que no sean necesarios al actualizar algunos de los componentes. Desde el directorio home de Liferay, se eliminó el

contenido de las carpetas '/osgi/state', 'temp' y 'work' y desde el directorio home del Jboss, se hizo lo propio con la información almacenada en '/standalone/tmp/*'. Se considera importante recalcar la necesidad de eliminar el contenido, y no los directorios, ya que Liferay, en el arranque necesita localizar estas ubicaciones para depositar los ficheros temporales, y en caso de no hacerlo, no podrá iniciarse correctamente.

A pesar de estar ya en la versión 7.4, se actualizó Liferay DXP con la intención de seguir los procedimientos establecidos, y porque no suponía una gran carga de trabajo. Sin embargo, al haber instalado ya el 7-4 en server1 y server2, no resultaba imprescindible situar los componentes en la última versión.

Preparación de la herramienta DB Upgrade y el proceso de Upgrade

En la sección anterior se mencionó la herramienta empleada para realizar el upgrade de la base de datos, Portal Tool DB Upgrade, con la que fue necesario realizar diversas acciones para preparar el entorno: configurar las properties del portal-tools-db-upgrade-client, crear archivos .config en algunas rutas y copiar el theme y el fichero de licencia.

En primer lugar, dentro de la ruta <*liferay_home*>*/tools/portal-tools-db-upgrade-client* existen unos properties que se deben de configurar. Estos son el controlador de la base de datos (jdbc.default.driverClassName), y la url, el usuario y la contraseña de la misma. Dentro de ese fichero es muy sencillo realizar esa modificación, ya que por defecto vienen comentados los parámetros necesarios para cada uno de los sistemas de gestión de base de datos que se deseen conectar, en este caso PostgreSQL.

A continuación, se hizo lo mismo, en el Servidor de Aplicaciones. El archivo que se modificó venía por defecto en la instalación, y al igual que en el caso anterior, se parte de una plantilla para rellenar según el tipo de sistema de gestión de base de datos y servidor de aplicación de la que se disponga. Como lo aportó la aplicación, la parte del SA venía por defecto, por lo que se añadieron los mismos parámetros que antes (url, usuario y contraseña).

En segundo lugar, fue necesario crear dos ficheros en la ruta <*liferay_home*>*osgi**configs*. El primero va en relación con el gestor del estado del índice, y en él se debe incluir la línea *indexReadOnly*=true, mediante la que se suspenden todas las operaciones de indexación y escritura en el motor de búsqueda. El segundo consiste en añadir la ruta principal de almacenamiento en la base de datos. Para ello, se crea el fichero *FileSystemStoreConfiguration.config* en y se añade el directorio raíz mediante la etiqueta rootdir, poniendo como valor el directorio en el que se encuentra el document library.

Ejecución del script para la actualización

Una vez completados los pasos previos, se procedió a lanzar el script db_upgrade, ubicado en el directorio <liferay_home>/tools/portal-tools-db-upgrade-client. Después del lanzamiento, apareció una pregunta solicitando la confirmación de la ejecución del script, y al confirmar, rápidamente se detectó que algo no había funcionado bien. El terminal mostraba el error reflejado en la figura 6.28 indicando que no se podía encontrar la clase DBUpgrader.

Tras investigar la posible causa del mismo y cómo solucionarlo, se comprobó que en el fichero app-server.properties no se había modificado la cadena de texto “standalone” por appfinal, como también se puede comprobar en la figura. Esto generaba que no encontrara la ruta especificada anteriormente.

Figura 6.28 Error al lanzar el db_upgrade, script para la actualización de la base de datos

En vista del mensaje de error, se modificaron todas las apariciones de esta cadena en el fichero afectado, y se volvió a lanzar el script. Para comprobar el correcto funcionamiento y finalización del mismo, fue necesario analizar el fichero donde se almacenan los registros del script, el log. En la figura 6.29 se puede comprobar que esta vez funcionó como era de esperar, ya que se mostró el mensaje “completed Liferay core upgrade process in 139 seconds”.

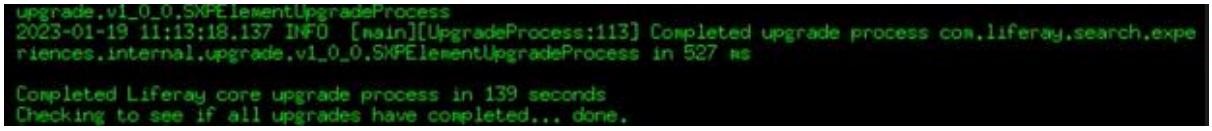

Figura 6.29. Correcta finalización del script de Upgrade

La última tarea en esta fase consistía en enviar mediante el nodo de comunicación al equipo de APPFINAL los logs del script, para que validaran que todo estaba en orden. Simplemente se realizó un scp -r del directorio logs de la ruta en la que se ejecutó el script a un nodo de gestión y de ahí, al de comunicaciones.

Tareas post-upgrade

Tras la ejecución del script, lo primero que tuvimos que realizar fue reactivar la indexación de contenidos, eliminando el fichero que gestionaba el estado del índice creado anteriormente. Además de eso, se comprobó que el fichero de propiedades appfinal-ext.properties contenía la configuración anterior, y se añadieron cuatro líneas mediante las que se especificaba que CKEditor sería el editor WYSIWYG por defecto.

Además, se volvieron a eliminar los ficheros temporales, para evitar almacenar grandes volúmenes de información sobre el script que no sería necesaria. Por último, se arrancó el Jboss, haciendo uso de un script de arranque.

6.6 Modificación de los frontales.

Los frontales son otro elemento que fue necesario modificar, al ser ellos quienes reciben las peticiones de los clientes y deben redirigir sus búsquedas al servidor adecuado. Como gestionan un gran número de portales de acceso, se deben configurar todas las URLs que atenderán los tres frontales en el fichero *vhost_appfinal.conf*. Este concepto recibe el nombre de virtual host, o servidor virtual, una forma de alojamiento web que permite que varias páginas web puedan funcionar en una misma máquina [31]. En este caso, el vhost se basa en nombres de dominio, donde una sola dirección IP (la de uno de los frontales) puede estar dando servicio a varias páginas web diferentes al mismo tiempo.

Para configurar virtual hosts basados en nombres de dominio, es necesario crear un bloque xml <VirtualHost> para cada uno de los hosts de los que se quiera dar servicio. Dentro de ese bloque, habrá al menos un '*ServerName',* directiva para designar el host al que se presta el servicio y una directiva *DocumentRoot* para indicar en qué parte del sistema de archivos se encuentra el contenido de ese host. Además, se puede incorporar la directiva *ServerAlias*, que indica otros nombres de dominio que pueden ser accedidos dentro de ese mismo vhost. En la figura 6.30 se ha facilitado un pequeño fragmento del fichero *vhost_appfinal.conf.*

```
ias@frontal1:/opt/app/apache/appfinal/conf/vhosts $ less vhost_appfinal.conf
<VirtualHost *:80>
  ServerAdmin gsi@LaAgencia.es
  ServerName appfinal.domain.es
  ServerAlias portal1.domain.es
  ServerAlias portal2.domain.es
  ServerAlias portal3.domain.es
  ServerAlias portal4.domain.es
  (…)

  ErrorLog "|/usr/sbin/rotatelogs /LOGS_HOME/appfinal/appfinal_error_%Y%m%d.log 86400"
  DocumentRoot "/opt/app/apache/appfinal/htdocs/appfinal"
```

Figura 6.30 Modificación de los vhosts en los frontales

Por otro lado, estos nuevos vhosts tienen que quedar registrados en la base de datos de database81. Es importante remarcar que la base de datos en APPFINAL posee el mismo esquema que en APP3, pero modificando al usuario propietario del mismo. Sin embargo, los vhosts tenían en su nombre una terminación ".app3" y en el nuevo nodo de base de datos, se decidió sustituir por ".appfinal". Para ello, se lanzaron diferentes órdenes UPDATE desde el nodo de base de datos. En la figura 6.31 se pueden ver dos de ellas junto con la cadena "UPDATE 1", que muestra la correcta finalización de la transacción anterior.

```
altdb=# UPDATE us_ptl.virtualhost SET hostname = 'portalcsappfinal.domain.es' WHERE
hostname = 'portalcsapp3.domain.es';
UPDATE 1
altdb=# UPDATE us_ptl.virtualhost SET hostname = 'coronaviruscsappfinal.domain.es'
WHERE hostname = 'coronaviruscsapp3.domain.es';
UPDATE 1
```

Figura 6.31 Modificación de los vhosts en base de datos

6.7 Despliegue de APP FINAL en los entornos de prueba

Una vez realizada toda la configuración, fue el momento de realizar el despliegue de la primera versión de la aplicación. Para ello, el equipo de la aplicación facilitó un fichero .war mediante el nodo de comunicación con las clases Java y demás objetos que querían desplegar sobre la infraestructura que se había preparado. Esta sección reflejará el primer despliegue en el entorno de TEST, usado para realizar pruebas sobre él. En producción son necesarias algunas acciones adicionales, además de un horario determinado, que se expondrán con detalle en la siguiente sección, y parece conveniente dedicar este espacio a exponer cómo se realiza un despliegue genérico en un Jboss, en este caso en APP FINAL.

Al no estar dando servicio, el primer despliegue se puedo realizar en ambos nodos al mismo tiempo, pero en todos los demás se deben completar las acciones en uno de los dos nodos y luego repetirlas en el segundo. Los pasos seguidos en el primer despliegue consistieron en parar el servidor, eliminar ficheros temporales, eliminar el .war anterior, depositar en el directorio *deploy* el .war facilitado, asegurarse de que elastic está levantado correctamente y, por último, volver a arrancar el Jboss.

1- Parada del Jboss. Para realizar esta acción se hizo uso del script de parada/arranque mencionado anteriormente, y se comprobó el estado de parada evaluando si existía algún proceso llamado “appfinal” mediante el comando *ps -fea | grep -i appfinal* . Al ver que el proceso lleva 0 segundos activo, se comprobó que la parada funcionó correctamente. Véase la figura 6.32 para más detalle.

```
#Ejecución del script de parada
[jboss_eap7.4] ias@server1:/JBOSS_HOME/bin $ ./K10appfinal
#Comprobación de la detención
[jboss_eap7.4] ias@server1:/JBOSS_HOME/bin $ ps -fea | grep -i appfinal
ias 79698 58475 0 15:12 pts/0 00:00:00 grep --color=auto -i appfinal
```

Figura 6.32. Parada del Jboss y comprobación del proceso

2- Limpieza de los ficheros temporales. Como se ha reiterado anteriormente en este mismo capítulo, la limpieza de directorios donde se almacenan ficheros temporales es una acción necesaria e importante, para evitar la acumulación de grandes volúmenes de datos sin motivo.
3- Dejar caer en la carpeta deploy del jboss el v1.war. En cualquier despliegue de appfinal en un Jboss, el directorio deploy de Liferay es donde se depositan los ficheros a desplegar. Si hubiera una versión anterior, se debería haber eliminado la misma antes de desplegar, pero al ser la primera no es necesario llevar a cabo esa acción. Para realizar esta tarea, usando el nodo de gestión se ha enviado el fichero dejado por la empresa en el nodo de comunicación a la ubicación destino. En la figura 6.33 se puede comprobar la correcta copia del fichero.

```
#Se sitúa el fichero en el directorio de despliegues
[jboss_eap7.4] ias@server1:/JBOSS_HOME/appfinal/liferay/deploy $ ls -lhrt
total 5.5M
-rw-r--r-- 1 ias gro 5.5M Mar 7 15:14 v1.war
[jboss_eap7.4] ias@server1:/JBOSS_HOME/appfinal/liferay/deploy $
```

Figura 6.33. Se sitúa el .war en el directorio de despliegues

4- Se comprueba que el Elastic está levantado. Fue necesario entrar en los nodos en los que estaba instalado Elasticsearch y simplemente comprobar si existía algún proceso activo en el sistema con la cadena “elastic”. En la figura 6.34 se comprueba el estado de arranque en uno de los tres nodos.

```
#Comprobación de que el elasticsearch estaba levantado en elastic1
ias@elastic1 # ps -ef | grep elastic
ias 54379 1 1 2023 ? 1-00:57:39 /JAVA_HOME/jdk-11.0/bin/java -Xshare:auto (…)
```

Figura 6.34. Se comprueba que elastic está levantado en uno de los nodos

5- Arranque del jboss. Mediante el script de arranque ./S10appfinal se inició el jboss. Al entrar al directorio donde se sitúan los elementos a desplegar, se comprobó que no estaba el .war de la primera versión, lo que indicaba que el despliegue debía haber ido bien.

Para comprobarlo, se ejecutó la existencia de un proceso del mismo modo que en la figura 31. Al ver que el proceso estaba arrancado, se revisó el nohup.out, fichero que almacena los logs del arranque. En la figura 35 se ha facilitado la línea que muestra que se han arrancado los servicios que gestiona el jboss (hay algunos pasivos que no aparecen en esa línea, de ahí que no ponga 628 de 628).

```
ias@server1 # less nohup.out | grep -i servicios
16:33:48,848 INFO  [org.jboss.as] (Controller Boot Thread) WFLYSRV0025: JBoss EAP
7.4.0.GA (WildFly Core 15.0.2.Final-redhat-00001) inició en 115738ms - Se iniciaron 428
de 628 servicios (347 servicios son diferidos, pasivos o por demanda).
```

Figura 6.35 Se comprueba que elastic está levantado en uno de los nodos

6.8 Despliegue de APP FINAL en producción

El primer despliegue de la aplicación en producción fue sin duda el momento más importante de todo el desarrollo. En la sección 5.7 se han detallado los pasos que se han de seguir en cualquiera de los despliegues que se realicen en la aplicación en cualquiera de los entornos. Lo que diferencia al primer despliegue en producción, es que además de las acciones mencionadas anteriormente, tras realizar las pruebas pertinentes y comprobar que todos los elementos están funcionando correctamente tanto desde el equipo de soporte de APPFINAL como desde GSI, se deben aplicar redirecciones en los frontales para abrir el portal al público en general. Para este proceso, fue necesario acordar un horario determinado con la Agencia de la Salud y las Enfermedades con el objetivo de influir lo menos posible en el servicio.

En concreto, fue necesario realizar un cambio de agujas para que la antigua www.domain.es dejase de mostrarse al público, pasando a ser wwwobsoleta.domain.es y que la actual www.domainappfinal.es sea la que se muestra, renombrándola como www.domain.es. Las modificaciones de los DNS de estos dominios no es una tarea que realice el equipo de GSI, pero antes de esta modificación que suponía mostrar al público al fin APPFINAL, se realizaron diversas acciones que se explicarán a continuación. En primer lugar, al cambiar algunas de las URLs en APPFINAL respecto de APP3, fue necesario revisar el fichero vhost_appfinal.conf. Para realizar las redirecciones de las URLs afectadas, en este fichero dentro de cada uno de los tres frontales se añadieron directivas RedirectMatch como las que se pueden ver en la figura 6.36.

```
ias@wbs81:/APACHE_HOME/conf/vhosts $ less vhost_appfinal_domain.conf
(…)
RedirectMatch 301 "^/web_estatica/index_es.html$" "/"
RedirectMatch 301 "^/web_estatica/listas_medicos_es.html$" "/es/web/listas-de-medicos"
RedirectMatch 301 "^/web_estatica/portal _paciente_es.html$" "/es/web/ciudadanos"
(…)
RedirectMatch "^/dominioapp-apple$" https://apps.apple.com/es/app/domain/id1191802694
RedirectMatch "^/seve/campus-virtual$" https://seve.domain.es/web/guest/campus-virtual
RedirectMatch "^/seve/itinerarios$" "https://seve.domain.es/web/guest/itinerarios"
(…)
#Se muestra una redirección de múltiples páginas a una sola
RedirectMatch 301 ^/publicaciones/(.*)$ /web/fondo-aplicaciones_appfinal
```

Figura 6.36. Se comprueba que elastic está levantado en uno de los nodos

En Apache, la forma de aplicar una redirección de forma permanentes es usar *RedirectMatch 301* y en caso de querer mostrar otra página de forma temporal se debería modificar ese 301 por 302. El formato básico de una línea es “RedirectMatch 301 /directorio-viejo/ http://www.dominio.com/directorio-nuevo/”. Esta, se puede complicar un poco más, en caso de querer redirigir múltiples páginas a una sola, como vemos también en la última línea de la figura 35. En ella se especifica que todas las líneas que comiencen por la cadena ‘publicaciones/’ se deben redirigir a ‘/web/fondo-editorial-y-documental’, siguiendo el formato “RedirectMatch 301 ^/viejo-directorio/(.*)$ http://www.dominio.com/nuevo-directorio”.

Tras aplicar un conjunto de redirecciones que solicitó el equipo de soporte de APP FINAL, se cambiaron los DNS de los dominios actuales y obsoletos. Al mismo tiempo, esperando que todo funcionara correctamente, se modificó el balanceador, que también actúa como reverse

proxy, para que comenzara a dar servicio a las peticiones de los usuarios. Durante los primeros minutos en funcionamiento, fue necesario realizar múltiples cambios aplicando redirecciones a URLs que solicitaba el equipo de la aplicación. Recibían llamadas informando de que algunas páginas no estaban dando servicio, debido a que no las habían incluido en las redirecciones previas al cambio en los DNS ni en el reverse proxy que daba acceso a la ciudadanía de La Comunidad a los nuevos portales de acceso. Alterando el fichero mostrado en la figura 35, se aplicaron y se resolvió el problema.

El despliegue en el entorno de producción fue satisfactorio. El procedimiento habitual (parada del jboss, eliminación de ficheros temporales, despliegue del war, comprobación de elastic y arranque del jboss) se realizó sin problema, ya que había sido probado y depurado gracias a los entornos de prueba. Tras los cambios aplicados, la actualización de las redirecciones supuso un esfuerzo extra, ya que de las más de 80 existentes en APP3, hubo que seleccionar y aplicar las necesarias hasta que, aparentemente, todos los recursos estaban disponibles para aquel que efectuara una petición sobre ellos. A pesar de esto, en próximos capítulos se analizará el comportamiento de la nueva aplicación un mes después de su instauración, comprobando su buen funcionamiento y evaluando de forma global el trabajo realizado. Ante la aprobación del equipo de APPFINAL, se dio por finalizada la migración de APP1, APP2 y APP3 de forma exitosa.

7. Pruebas

Como se ha mencionado en anteriores capítulos, en GSI todas las aplicaciones se encuentran en diferentes entornos. APPFINAL está disponible en TEST, PREPRODUCCIÓN y PRODUCCIÓN, siendo este último el que da servicio a los usuarios. Los otros dos los emplea el equipo de GSI y el equipo de soporte de la empresa para probar nuevas versiones, realizar los cambios pertinentes y actualizar los componentes para mantener el entorno de producción lo más estable posible. En este capítulo se describirán las diferentes pruebas realizadas sobre la aplicación APPFINAL una vez se desplegó en los entornos previos a salir al público, así como los problemas que se encontraron.

Dentro de la Agencia de la Salud y las Enfermedades de la Comunidad hay diferentes equipos, además de GSI, que participaron en la migración expuesta en este estudio como, por ejemplo, una unidad de comunicaciones encargada de la conectividad a nivel físico entre la Agencia y las diferentes entidades que la conforman, como centros de salud, hospitales, o farmacias; un grupo encargado de dar soporte a los clientes, el equipo de monitorización o el grupo destinado a realizar las pruebas de las aplicaciones.

En cada uno de los pasos descritos en el capítulo anterior se realizaron pruebas intermedias que sirvieron como puntos de control en la migración de APP FINAL. Inicialmente, tras la creación y configuración de la base de datos, se comprobó que se podía acceder a esta, y se revisaron los roles y el esquema recién creado, comprobando que el procedimiento seguido había sido el correcto. A continuación, tras configurar los SAs, los nodos que tenían instalado Elasticsearch y el servidor de portales Liferay DXP, se realizó otra prueba importante. Cuando se arrancó Liferay, se comprobó que el estado del servidor en la consola web era el adecuado, y que la configuración de ciertos elementos mediante el terminal había sido la idónea. Ocurrió lo mismo con Elasticsearch, al arrancar la consola de uno de ellos en el navegador, se comprobó que la configuración permitía el arranque, y este era capaz de identificar al resto de nodos, lo que mostró que la configuración en clúster era totalmente operativa.

En la actualización de Liferay DXP y la modificación de los frontales, la validación de los cambios fue desarrollada por equipos externos a GSI. Al ser un producto llevado a cabo por una empresa externa, desde el equipo de APP FINAL comprobaron que los frontales mostraban todos los elementos esperados, y desde soporte Liferay revisaron la configuración. Con el despliegue en todos los entornos ocurrió algo muy similar, tanto desde el equipo de la aplicación como desde GSI se revisó que el contenido que habían proporcionado era correcto y la funcionalidad era la esperada. De hecho, en PRODUCCIÓN, se reservaron un par de días desde el primer despliegue hasta la redirección de los frontales, momento en el que se iba a empezar a prestar servicio, para comprobar que todo estaba en orden.

Dado que las prácticas del alumno, y su posterior colaboración en Prosodie S.L.U. no han tenido lugar en el equipo de pruebas, no se puede exponer al detalle cómo se han llevado a cabo las mismas. Sin embargo, para la redacción de este capítulo desde el grupo de trabajo que testeó APP FINAL han facilitado información y explicado cómo se realizaron las pruebas para asegurar que cuando la aplicación se abriera al público iba a tener el menor número de errores posibles.

Ahora sí, dentro de las acciones realizadas por el equipo de pruebas, es necesario diferenciar dos tipos de verificaciones. En el primer entorno o TEST se realizarían las pruebas funcionales, basadas en comprobar el correcto funcionamiento de la aplicación (creación de usuarios, muestra del cuadro de médicos, acceso a ficheros con listas de medicamentos…) y en el segundo entorno o PRE se realizarían las pruebas no funcionales o pruebas de carga, en las que se sometería el sistema a una gran cantidad de peticiones para comprobar el funcionamiento y la robustez de los diferentes elementos de la arquitectura.

Pruebas funcionales

En el entorno de TEST, el primero en el que se despliegan las aplicaciones, la base de datos no es exactamente igual a la de preproducción ni a la de producción. Los usuarios, roles, esquemas, y demás objetos que conforman la aplicación sí que están presentes, pero hay una cantidad de datos mucho menor, en su mayoría anónimos y diferentes a los de producción. El hecho de que la base de datos sea mucho menos densa explica perfectamente por qué ahí se realizan las pruebas funcionales y no las pruebas de carga, ya que estas últimas no tendrían validez si el entorno posee una décima parte de las tablas de la BD de producción.

En las pruebas funcionales, el equipo encargado de desarrollar la aplicación, externo a La Agencia de la Salud y las Enfermedades, empieza por comunicar los casos de uso en los que han basado la construcción de la aplicación. Un caso de uso, según IBM, es un artefacto que define una secuencia de acciones que da lugar a un resultado de valor observable. Este conjunto de acciones es el que el equipo de pruebas deberá validar, comprobando que se ejecuten de forma correcta. Un ejemplo de caso de uso puede ser el de añadir un medicamento nuevo. Para ello, se supone que habría un formulario, con ciertos requerimientos en alguno de sus campos, y si se cumplimenta de forma correcta, generaría una nueva entrada en la base de datos con esa información. En ese caso hipotético, desde el equipo de pruebas se comprobará que cumple los requerimientos necesarios y que la aplicación da un servicio acorde al que se pretende dar.

Una vez *pruebas* extraiga los resultados, al tratarse del primero de los entornos, realizará un informe, basándose en la corrección o incorrección de los casos de uso planteados por la empresa externa, y en caso de encontrar algún fallo, el equipo de APP FINAL debería subir una nueva versión que los solucionara. Una vez completadas y validadas estas pruebas en la aplicación que gestionaba los portales de acceso, el despliegue en el primero de los entornos se da por finalizado. Es necesario remarcar que APP1, APP2 y APP3 eran idénticas funcionalmente a APP FINAL, por lo que en este caso estas pruebas fueron un mero trámite. A pesar de actualizar componentes, modificar versiones y cambiar elementos de la arquitectura, el código que desarrollaron para los diferentes casos de uso simplemente lo rescataron de las aplicaciones anteriores.

Pruebas no funcionales

Al finalizar el despliegue en el segundo de los entornos, el equipo de pruebas establece el filtro más importante para que una aplicación pueda ser visible para el público, que sea capaz de adaptarse a una gran cantidad de peticiones sin dejar de dar servicio y sin perder prestaciones. En este caso, la base de datos de las aplicaciones sí que es idéntica a la de PRODUCCIÓN, lo que permitió evaluar el comportamiento de APP FINAL ante una carga elevada. Teniendo claro que los diferentes casos de uso que planteó la aplicación están programados correctamente, a grandes rasgos se probará cada uno de ellos de forma masiva, y se compararán los resultados con los de versiones anteriores, para concluir si el despliegue en PRO es viable o si son necesarios cambios en la aplicación para continuar.

El equipo de pruebas dispone de una infraestructura determinada para testear las diferentes aplicaciones formadas por una máquina central que es la que controla todo el proceso, y otras cinco a disposición de esta. Desde la principal se gestiona el lanzamiento de peticiones abriendo diferentes clientes en cada una de las otras máquinas. En caso de estar diseñada para 200 clientes de forma concurrente, desde la máquina central se abrirían 40 navegadores en cada una de las otras cinco y se ejecutaría la misma petición o actualización, midiendo los tiempos de respuesta y estableciendo una media entre ellos.

En la figura 7.1 se puede ver el esquema de los equipos de pruebas y su interacción con los equipos configurados en capítulos anteriores en GSI. Los primeros poseen una máquina central, como se puede ver en la parte superior de la figura, que emplea cinco clientes para abrir n navegadores en cada uno. La cantidad de clientes que se abrirá en cada máquina variará en función de las peticiones que debe poder atender la aplicación para cada uno de los casos de uso probados en los testeos funcionales. Además, en la figura se comprueba la interacción entre estos equipos y los de GSI. Las peticiones de estos frontales llegan al balanceador de carga conectado con los frontales, que las llevarán al SA para poder dar servicio a estas.

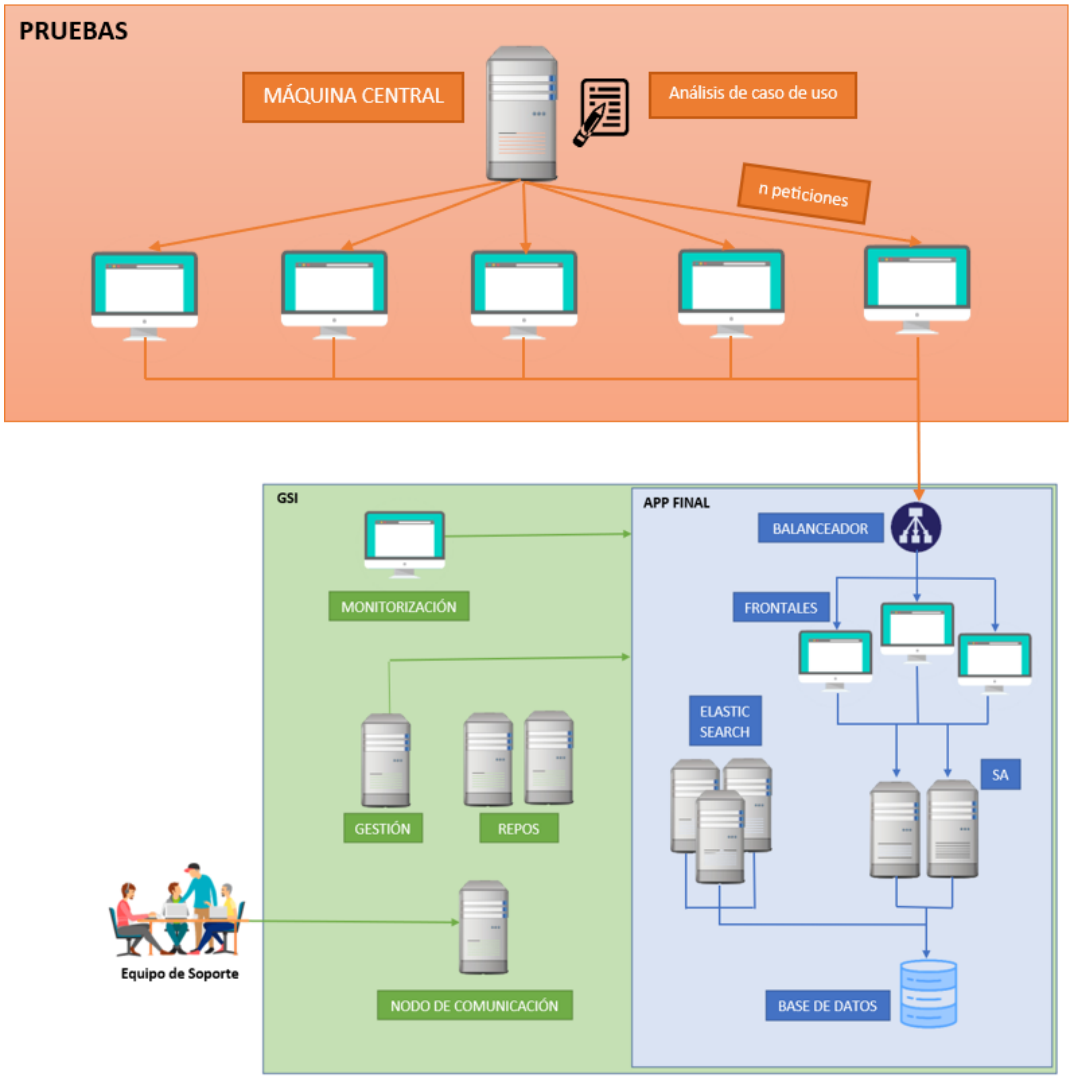

Figura 7.1.Esquema de GSI y el equipo de pruebas único.

En las pruebas funcionales se comprobó si los diferentes casos de uso que resolvía la aplicación estaban correctamente implementados. En las pruebas de carga se seleccionan los casos de uso uno por uno y en el nodo central se graba el procedimiento para realizarlos sobre la aplicación (por ejemplo entrar en una página, visitar un enlace, rellenar un formulario y hacer clic sobre el botón “enviar”). Así, este nodo será capaz de trasladar las acciones necesarias a las cinco máquinas que actuarán como clientes y estas podrán resolver las peticiones de forma automática. El aspecto más interesante de esta disposición es que permite la realización de tantas pruebas concurrentes como sea necesario para cada uno de los casos de uso (igual la aplicación está diseñada para permitir consultar un listado de médicos a 1.000 personas al mismo tiempo, pero sólo permite actualizar la base de datos a la vez a diez usuarios).

Para comprender mejor el propósito de las pruebas de carga, es necesario comprender el concepto de línea base. En el primer despliegue de una aplicación se prueban todos los casos de uso y se evalúa el retardo medio de las peticiones para diversos números de usuarios, y junto al equipo de la aplicación se establece una duración aceptable para cada uno de los casos. En el caso de APP FINAL, la línea base ya estaba establecida con APP1, APP2 y APP3, pero en el primer despliegue en PRE se comprobó que los tiempos de respuesta

medios eran claramente inferiores a los que tenían de antes de la migración. Esto mostraba una evolución positiva respecto a las aplicaciones previas a la migración, aspecto que permitió el paso de la aplicación al entorno de producción.

La línea base es la clave a la hora de evaluar el rendimiento de una nueva versión de una aplicación. Si las peticiones de un caso de uso se resolvían de media en 15 milisegundos y pasan a necesitar 200, es necesario establecer una comunicación con el equipo encargado de desarrollar la aplicación y evaluar si esa diferencia va acorde a una modificación del código de la aplicación. En el caso de APP FINAL, simplemente se les comunicaron los tiempos de respuesta y se validaron las pruebas, ya que en ninguno de los casos de uso que propuso la aplicación se detectaron retardos preocupantes en la resolución de peticiones.

El equipo de GSI participa indirectamente en estos testeos, y es que a pesar de no conocer el procedimiento ni evaluar los diferentes casos de uso (es tarea del equipo de pruebas), las peticiones de las cinco máquinas clientes serán lanzadas sobre los frontales, servidores de aplicación y bases de datos montadas en GSI. De este modo, en caso de un fallo de conexión entre dos componentes, como un frontal y uno de los SAs, desde el equipo de pruebas detectan este error y lo notifican, permitiendo depurar también las comunicaciones entre los equipos y gestión de las peticiones antes de pasar a producción.

En el caso de APP FINAL, como se ha detallado a lo largo de este capítulo, las pruebas fueron satisfactorias. La base de datos estaba correctamente configurada y los servidores de aplicación resolvían las peticiones de los “falsos clientes” sin denotar un gran retardo en los tiempos de respuesta. Además, en las pruebas funcionales se comprobó que los diferentes casos de uso se resolvían correctamente. Tras el éxito en las pruebas funcionales y las pruebas de carga, el equipo de soporte de APP FINAL aprobó el paso al entorno de PRO.

8. Conclusiones y trabajos futuros

8.1 Despliegue de APP FINAL en producción

En esta sección se evaluarán objetivamente los resultados de la migración, comprobando si se cumplieron los objetivos planteados al inicio del estudio y dando una visión global sobre el funcionamiento de APP FINAL en comparación a APP1, APP2 y APP3. El objetivo final del proyecto, que describe el capítulo 1, es el siguiente: "lograr la instalación y configuración correcta de todos los elementos y la migración completa de ***APP1, APP2, APP3*** a la nueva ***APP FINAL***, haciendo que sobre esta última aplicación recaiga el servicio a los clientes en el entorno de Producción".

Tras el proceso, se comprueba que la aplicación está dando servicio a la ciudadanía, y para llegar a ese estado ha sido necesaria la configuración correcta de cada uno de los elementos que conforman la arquitectura de esta. Además, se ha comprobado su eficacia haciendo un pequeño análisis de la cantidad de solicitudes que recaen sobre APP FINAL y los problemas de funcionamiento que ha registrado desde que está activa. Mediante un filtrado de los ficheros de log en cada uno de los frontales, se comprobó que diariamente cada uno de ellos atiende más de un millón de peticiones. La aplicación, que está dando servicio a más de tres millones de clientes a diario, únicamente ha necesitado dos reinicios en un mes, desde que está en funcionamiento. Este dato es muy positivo, pues las aplicaciones previas a la migración registraban un reinicio prácticamente cada día, tal y como se puede ver en el capítulo dedicado al análisis.

De los dos reinicios mencionados, uno ha sido por un cambio de configuración para ampliar el pool de la base de datos y poder aumentar el número de conexiones concurrentes a petición del equipo de soporte y el otro para realizar un nuevo despliegue sobre la aplicación. No han surgido Stall Counts preocupantes ni alertas por exceso de tiempo empleando el recolector de basura o Garbage Collector.

En la figura 2 se aprecia una comparación de las alertas que saltaron en la aplicación APP3 en abril de 2020 con respecto a las que surgieron en APP FINAL en abril de 2023. A simple vista se puede apreciar cómo la migración ha eliminado el 97.4% de las alertas que saltaron en el servidor de la aplicación en el que se alojan los portales de acceso de La Agencia de la Salud y las Enfermedades. En el mes de abril del presente año el total de alertas ha sido de 13, mientras que, en el año de la pandemia en ese mismo mes, el total fue de 485. Claramente la migración ha cumplido con el objetivo de minimizar las alarmas que han aparecido, eliminando drásticamente la aparición de Stall Counts y reduciendo el tiempo que los servidores emplean el recolector de basura (GC).

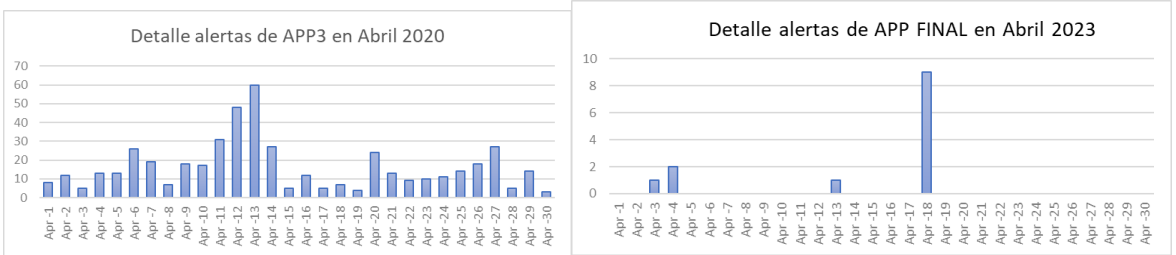

Figura 8.1 Comparación de las alertas de APP3 en abril 2020 y APP FINAL en abril 2023.

Desde el equipo de GSI no se han podido extraer datos sobre las peticiones diarias que recibían los portales de acceso de La Agencia en 2020, pero ante la emergencia social que se vivió durante la pandemia del COVID-19, se supone que sería un número mucho mayor que en la actualidad. Aun así, aunque fueran diez veces más que las actuales (30 millones diarias), las alertas proporcionales seguirían siendo el cuádruple que las registradas hoy en día.

Los objetivos a corto plazo que se propusieron al inicio del desarrollo se han cumplido también, ya que eran puntos de control que marcaron el desarrollo de la migración. Por ejemplo, sin tener montada y configurada la base de datos no se podía configurar la misma en los ficheros del JBOSS ni del servidor de portales Liferay DXP. Así, en el capítulo 5 se puede comprobar paso a paso cómo se completó la creación y configuración de la base de datos, los servidores de aplicación, el Liferay y el Elasticsearch, la conexión entre los SAs y los Elastic comprobada mediante una prueba de arranque… y de igual modo todos los objetivos a corto plazo propuestos inicialmente.

A pesar de superar los checkpoints propuestos inicialmente de forma exitosa, la parte final de este desarrollo no se ha podido realizar por falta de tiempo. Al terminar el despliegue de la primera versión de APP FINAL, se estableció una comunicación con el equipo de soporte para actualizar la monitorización de las alarmas en Introscope y Nagios, con el objetivo de adaptarlas al nuevo código proporcionado por ellos. Sin embargo, desde la aplicación, se decidió que no era necesario hacerlo con rapidez, puesto que las alarmas que monitorizaban APP1, APP2 y APP3 se adaptaban adecuadamente al nuevo código. Por ello, se dejó como un tema pendiente para el futuro y no ha podido estar descrito en este estudio, puesto que no se ha realizado todavía. Aun así, se trata de una parte posterior a la migración y su no resolución en el momento de desarrollar la memoria no es razón suficiente como para no concluir en que ha sido un éxito.

A lo largo del desarrollo de la migración hubo ciertos aspectos que supusieron un problema para la misma. En general, la planificación de los pasos a seguir fue algo sencillo, ya que se hizo uso de la documentación facilitada por las diferentes herramientas y el equipo de la aplicación estableció una guía a seguir. Sin embargo, fue muy laborioso identificar todas las variables que era necesario modificar en cada uno de los archivos y analizar su funcionamiento, para poder entender el comportamiento de la aplicación en cada uno de los casos. Por ejemplo, inicialmente no se logró configurar en clúster el Elasticsearch, y al realizar una primera prueba de arranque se comprobó que uno de los nodos no detectaba al resto. Tras una revisión de la configuración y una extensa búsqueda en la web, se encontró la solución y se repitió la prueba tras modificar la configuración, resultando ahora sí, satisfactoria.

Realmente, lo más difícil con diferencia ha sido el aprendizaje por parte del alumno de tantas tecnologías y de su funcionamiento interno. No sólo ha supuesto conocer Elasticsearch, Liferay DXP, PostgreSQL o el resto de las herramientas, sino que además ha sido necesario comprender cómo están configuradas y qué importancia tiene cada una de ellas dentro de la arquitectura de la aplicación. En el grado universitario, el alumno aprendió el concepto de base de datos, era conocedor de la función que desempeña un servidor o del frontend con los frontales, pero ni siquiera conocía conceptos como servidor de portales ni motor de

búsqueda, por lo que los tuvo que aprender con anterioridad a este desarrollo para ser capaz de configurarlos posteriormente.

Además, también ha resultado de sumo interés para el alumno el proceso por el que debe pasar una aplicación desde que se escribe el código de esta hasta que se publica para el uso del público general. Ha sido sorprendente conocer que existen entornos de prueba, se aplican testeos funcionales y no funcionales, descubrir la infraestructura a nivel de equipos y el número de personas que hay detrás trabajando para que funcione, así como la necesidad de una comunicación estrecha y continua entre todos los equipos para llevar el desarrollo a buen puerto.

Este estudio ha sido de sumo interés profesional para el alumno. Al llegar a la empresa, no había tratado nada de lo que se ha desarrollado y presentado en esta memoria, pero gracias a la realización de este trabajo, no sólo se han adquirido conceptos teóricos, manejo y destreza con las tecnologías relacionadas, sino que se ha podido aprender cuál era la forma de trabajar en Prosodie S.L.U. y evolucionar como analista informático. Ha sido muy útil, ya que ha supuesto un proceso acelerado de aprendizaje, y en tres meses se ha evolucionado de no saber hacer un despliegue a poder completar una migración de una aplicación tan importante como APP FINAL, bajo supervisión del resto del equipo, por supuesto. La empresa ha valorado muy positivamente el trabajo del alumno en el proyecto, y fuera de él, y gracias a los conocimientos adquiridos con el proyecto y a la destreza a la hora de desempeñar su tarea, ha decidido contratar al alumno de cara al futuro.

Personalmente, ha sido un gran reto. Como alumno, no imaginé a principio del curso académico 2022-23 que podría realizar algo de tal importancia en una empresa. Además, en el grado pude comprender teóricamente muchos conceptos, pero ha sido muy gratificante aplicarlos en la práctica y entender cómo funcionan realmente los portales de acceso de La Agencia de la Salud y las Enfermedades de La Comunidad. A nivel conceptual, ha sido un proceso bastante difícil, ya que no conocía la mayor parte de las herramientas con las que he trabajado y he tenido que adaptarme no sólo a ellas, sino que, además, ha sido necesario acostumbrarme al trabajo en la empresa y conocer cómo funcionaba todo allí. El inicio fue un poco caótico, pero a estas alturas puedo afirmar que, personalmente, he crecido mucho.

8.2 Relación del trabajo realizado con los estudios cursados

Durante la carrera el alumnado estudia desde una perspectiva general la Ingeniería Informática. En diversas asignaturas se analiza la arquitectura de tres capas, y cómo conectar a nivel de programación la capa de datos con las bases de datos, la capa lógica con los servidores de aplicación y la capa de presentación con los frontales. En este caso, el estudio no se ha centrado en el desarrollo del código, sino en la comunicación a nivel de configuración de ficheros y de gestión del tráfico. Esto ha otorgado otra visión al alumno y ha cumplimentado los estudios cursados en este ámbito, enriqueciendo claramente el conocimiento acerca del tema.

Sin embargo, el trabajo tiene una relación directa con la rama estudiada por el alumno, ‘Tecnologías de la Información’. En ella una parte de los contenidos son asignaturas en las que se explica el funcionamiento de las bases de datos, y en este estudio se ha configurado una base de datos y se ha realizado una migración de una máquina a otra, importando y exportando esquemas, creando los tablespaces… Por otro lado, se establecen ficheros de

log en todas las máquinas, en los que se reflejan todas las transacciones realizadas sobre la base de datos, accesos a la aplicación en los frontales y resolución de peticiones en los SAs. En estas asignaturas de bases de datos se estudió el concepto de log y su utilidad. El conocimiento aprendido fue útil para poder realizar la configuración y localizar la información acerca de arranques, peticiones o transacciones con facilidad.

Las máquinas que se tratan en el estudio son máquinas virtuales alojadas en máquinas físicas situadas en un centro de procesamiento de datos de La Agencia. El concepto de máquina virtual y su tratamiento se estudió en 'Redes de Área Local', por lo que la comprensión del funcionamiento y la abstracción de las máquinas respecto de dispositivos físicos fueron conceptos ya interiorizados. Por el contrario, en este estudio no se han tratado aspectos como la conexión cableada entre los componentes o con los diferentes centros que pertenecen a la Agencia (hospitales, farmacias, ambulatorios…), ya que no es una tarea que se realice desde GSI y no tenía relación directa con la migración, pero sí ha sido imprescindible comprender la conexión entre los diferentes elementos en APP FINAL.

El funcionamiento de un pool de conexiones fue un concepto estudiado en 'Fundamentos de Sistemas Operativos', y resultó sencillo refrescar ese conocimiento cuando fue necesario para entender cómo se interconectaban frontales, SAs y la base de datos. Además, en esa misma asignatura se estudió la interfaz de usuario de línea de comandos *bash* (The Bourne Again Shell), mediante la cuál se realizó todo el proyecto, y principal herramienta para el equipo de GSI.

Además de los aspectos mencionados, en cada uno de los pasos realizados a lo largo del proyecto se han necesitado conceptos estudiados a lo largo del grado, ya sea de bases de datos, acerca de configuración de servidores o parámetros de memoria de una máquina más propios de la rama de Sistemas de la Información. A la hora de configurar cualquier fichero es necesario comprender las variables que se van a modificar y su significado, y el haber cursado el grado me ha permitido comprender con rapidez qué estaba realizando en cuanto he visto los manuales de las herramientas o explicaciones acerca de ellos.

8.3 Trabajos futuros

Como se ha destacado en este mismo capítulo, en la migración han faltado por configurar las herramientas de monitorización como son Introscope y Nagios. Sin duda, en los próximos meses se realizará una revisión de las alertas por si es necesario realizar alguna modificación en ellas para APP FINAL respecto APP1, APP2 y APP3.

En vista de los satisfactorios resultados de la migración, en la empresa se evaluará la migración de algunas de las aplicaciones que cuentan con servidores antiguos como el Apache Tomcat que tenía APP1 u otros sistemas de gestión de bases de datos que no sea PostgreSQL, que es la alternativa que se está tomando en las aplicaciones configuradas en los últimos tiempos en La Agencia.

Realmente, más allá de los aspectos mencionados en los párrafos anteriores, no hay intención dentro de La Agencia de realizar una modificación en APP FINAL tras esta migración. Fue un proyecto cerrado, y tras haberse cumplido los objetivos y APP FINAL estar disponible para el público, en los próximos años se analizará si es necesario modificar alguno de los elementos sustituyéndolos por algo más novedoso, pero por ahora no será así.

9. Referencias bibliográficas

[1] Partners – Prosodie. Prosodie – Prosodie ofrece a las empresas servicios para la gestión 24×7 de las tecnologías de información y comunicaciones. [en línea]. [sin fecha] [consultado el 10 de Febrero de 2023]. Disponible en: https://www.prosodie.es/?page_id=3868

[2] SANDBU, Marius. Microsoft System Center Configuration Manager. Packt Publishing Ltd, 2013.URL

[3] pdq deploy: instalación, configuración y despliegue de software - .:Blog Virtualizado:. .:Blog Virtualizado:. [en línea]. [sin fecha] [consultado el 12 de febrero de 2023]. Disponible en: https://blogvirtualizado.com/pdq-deploy-instalacion-configuracion-y-despliegue-de-software/

[4] FENTON, Steve. Exploring Octopus Deploy. 2015.

[5] How to Deploy Application with EMCO Remote Installer. TechSupport [en línea]. [sin fecha] [consultado el 12 de Febrero de 2023]. Disponible en: https://www.techsupportpk.com/2016/10/how-to-deploy-application-with-emco-remote-installer.html

[6] What is Three-Tier Architecture | IBM. IBM - Deutschland | IBM [en línea]. [sin fecha] [consultado el 12 de Febrero de 2023]. Disponible en: https://www.ibm.com/topics/three-tier-architecture

[7] GORMLEY, Clinton; TONG, Zachary. Elasticsearch: the definitive guide: a distributed real-time search and analytics engine. " O'Reilly Media, Inc.", 2015.

[8] REESE, Will. Nginx: the high-performance web server and reverse proxy. Linux Journal, 2008, vol. 2008, no 173, p. 2.

[9] Name-based Virtual Host Support - Apache HTTP Server Version 2.4. Welcome! - The Apache HTTP Server Project [en línea]. [sin fecha] [consultado el 24 de febrero de 2023]. Disponible en: https://httpd.apache.org/docs/current/vhosts/name-based.html

[10] DXP. Liferay Learn [en línea]. [sin fecha] [consultado el 24 de febrero de 2023]. Disponible en: https://learn.liferay.com/en/w/dxp/index

[11] DIAZ, Oscar; RODRIGUEZ, J. Portlets as web components: An introduction. Journal of Universal Computer Science, 2004, vol. 10, no 4, p. 454-472.

[12] Pluto – Welcome to Pluto. The Apache Portals Site – Apache Portals [en línea]. [sin fecha] [consultado el 24 de febrero de 2023]. Disponible en: https://portals.apache.org/pluto/

[13] JAMAE, Javid; JOHNSON, Peter. JBoss in action: configuring the JBoss application server. Simon and Schuster, 2008.

[14] KUC, Rafal; ROGOZINSKI, Marek. Elasticsearch server. Packt Publishing Ltd, 2013.

[15] Welcome to Apache Lucene. Apache Lucene - Welcome to Apache Lucene [en línea]. [sin fecha] [consultado el 5 de marzo de 2023]. Disponible en: https://lucene.apache.org/

[16] Apache Tomcat Maven Plugin - About Apache Tomcat Maven Plugin. Apache Tomcat® - Welcome! [en línea]. [sin fecha] [consultado el 5 de marzo de 2023]. Disponible en: https://tomcat.apache.org/maven-plugin-2.0/

[17] Product Documentation for Red Hat JBoss Enterprise Application Platform 7.4 | Red Hat Customer Portal. Red Hat Customer Portal [en línea]. [sin fecha] [consultado el 5 de marzo de 2023]. Disponible en: https://access.redhat.com/documentation/en-us/red_hat_jboss_enterprise_application_platform/7.4

[18] PostgreSQL 14.7 Documentation. PostgreSQL Documentation [en línea]. [sin fecha] [consultado el 15 de marzo de 2023]. Disponible en: https://www.postgresql.org/docs/14/index.html

[19] 1.3. Creating a Database. PostgreSQL Documentation [en línea]. [sin fecha] [consultado el 17 de marzo de 2023]. Disponible en: https://www.postgresql.org/docs/14/tutorial-createdb.html

[20] 21.1. The pg_hba.conf File. PostgreSQL Documentation [en línea]. [sin fecha] [consultado el 23 de marzo de 2023]. Disponible en: https://www.postgresql.org/docs/current/auth-pg-hba-conf.html

[21] JDK 11 Releases. JDK Builds from Oracle [en línea]. [sin fecha] [consultado el 23 de marzo de 2023]. Disponible en: https://jdk.java.net/11/

[22] IBM Documentation. IBM - Deutschland | IBM [en línea]. [sin fecha] [consultado el 23 de marzo de 2023]. Disponible en: https://www.ibm.com/docs/en/atlas-policy-suite/6.0.3?topic=suite-data-source-definitions

[23] Installing Liferay DXP on JBoss EAP. Liferay Help Center [en línea]. [sin fecha] [consultado el 28 de marzo de 2023]. Disponible en: https://help.liferay.com/hc/es/articles/360028810012-Installing-Liferay-DXP-on-JBoss-EAP

[24] JBoss Web Configuration Reference - The jsp-configuration element. docs.jboss.org [en línea]. [sin fecha] [consultado el 28 de marzo de 2023]. Disponible en: https://docs.jboss.org/jbossweb/7.0.x/config/jsp.html

[25] Chapter 16. Internationalization, localization and themes. docs.jboss.org [en línea]. [sin fecha] [consultado el 28 de marzo de 2023]. Disponible en: https://docs.jboss.org/seam/2.3.0.CR1/reference/html/i18n.html

[26] Configuring Messaging Red Hat JBoss Enterprise Application Platform 7.1 | Red Hat Customer Portal. Red Hat Customer Portal [en línea]. [sin fecha] [consultado el 3 de abril de 2023]. Disponible en:https://access.redhat.com/documentation/en-us/red_hat_jboss_enterprise_application_platform/7.1/html-single/configuring_messaging/index

[27] 18.2. Java Authentication and Authorization Service (JAAS) JBoss Enterprise Application Platform 6.2 | Red Hat Customer Portal. Red Hat Customer Portal [en línea]. [sin fecha]

[consultado el 3 de abril de 2023]. Disponible en: https://access.redhat.com/documentation/en-us/jboss_enterprise_application_platform/6.2/html/security_guide/java_authentication_and_authorization_service_jaas

[28] ROSENBERG, Jake. Data Discoverability in Science Gateways at Scale using Elasticsearch Cluster Architecture. En Practice and Experience in Advanced Research Computing. 2022. p. 1-3.

[29] SEZOV, Richard. Liferay Administrator's Guide. Lulu. com, 2008.

[30] Managing Documents and Media. Liferay Help Center [en línea]. [sin fecha] [consultado el 3 de abril de 2023]. Disponible en: https://help.liferay.com/hc/en-us/articles/360017876132-Managing-Documents-and-Media#:~:text=Liferay%20DXP's%20Documents%20and%20Media,use%20to%20store%20files%20locally

[31] SILEIKA, Rytis. Maintaining a List of Virtual Hosts in an Apache Configuration File. Pro Python System Administration, 2010, p. 137-157.

Anexo: Objetivos de Desarrollo Sostenible (ODS)

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	**Alto**	**Medio**	**Bajo**	**No Procede**
ODS 1. Fin de la pobreza.				X
ODS 2. Hambre cero.				X
ODS 3. Salud y bienestar.	X			
ODS 4. Educación de calidad.				X
ODS 5. Igualdad de género.				X
ODS 6. Agua limpia y saneamiento.				X
ODS 7. Energía asequible y no contaminante.				X
ODS 8. Trabajo decente y crecimiento económico.				X
ODS 9. Industria, innovación e infraestructuras.	X			
ODS 10. Reducción de las desigualdades.		X		
ODS 11. Ciudades y comunidades sostenibles.				X
ODS 12. Producción y consumo responsables.				X
ODS 13. Acción por el clima.				X
ODS 14. Vida submarina.				X
ODS 15. Vida de ecosistemas terrestres.				X
ODS 16. Paz, justicia e instituciones sólidas.		X		
ODS 17. Alianzas para lograr objetivos.				X

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

Dentro de los Objetivos de Desarrollo Sostenible, el más relacionado con el presente estudio es, indudablemente, el tercero *'Salud y bienestar'*. A pesar de ser un desarrollo informático, las aplicaciones que se migran son los portales de acceso a la web de La Agencia de la Salud y las Enfermedades de la Comunidad. Hace varias décadas no eran tan populares, pero debido a la evolución de la tecnología, hoy en día las aplicaciones informáticas como APP FINAL son el principal punto de información para la ciudadanía acerca de cualquier tema sanitario. Como se explica en capítulos anteriores, el objetivo del estudio es mejorar los

diferentes componentes de la arquitectura para tratar de dar un servicio robusto e ininterrumpido. Como se explicó en las conclusiones del proyecto, el número de reinicios que ha sufrido *APP FINAL* tras la migración ha sido un 97% menos que con las tres aplicaciones anteriores. Esto quiere decir que, al año, la sociedad de La Comunidad sufrirá un 97% menos de pérdida en el servicio de los portales sanitarios, convirtiéndolos prácticamente en un servicio disponible ininterrumpidamente durante todo el año.

Por otro lado, se considera que este proyecto posee relación con el noveno objetivo 'Industria, Innovación e Infraestructuras'. Quizá es el más común en un desarrollo informático, pero en este caso con más razón. Se evoluciona una aplicación construyendo nuevas máquinas, instalando y configurando software de calidad para lograr prestar a la población un mejor servicio.

Otros dos objetivos de desarrollo sostenible con los que se puede relacionar el presente estudio son el décimo 'Reducción de las Desigualdades' y el decimosexto 'Paz, justicia e instituciones sólidas'. El primero de los citados se ve reflejado en el trabajo, ya que con la evolución de los componentes de la aplicación se busca permitir que el conjunto de la ciudadanía con acceso a internet, sin importar las facilidades de las que dispongan, pueda tener una buena experiencia de usuario a la hora de acceder a los recursos de la web. El segundo de ellos tiene cabida en el desarrollo también, ya que el estudio se ha realizado para una Agencia de la Salud y las Enfermedades de una Comunidad, siendo una institución que tiene un gran peso a nivel social.

Fuera de los ODS mencionados, el estudio no tiene una relación directa con ninguno de los demás. Al mejorar la salud de la población de forma indirecta, contribuye a mejorar la vida de los ciudadanos, pero no se considera que "acabe con la pobreza", "disminuya el hambre" o influya de alguna forma en la educación. El alumno que lo ha desarrollado ha conseguido mejorar su situación laboral, logrando un trabajo decente al final de las prácticas curriculares y con él un crecimiento económico, pero no se considera que haya afectado en este objetivo a nivel social más allá del individuo mencionado. En este mismo sentido, la aplicación distingue a los diferentes usuarios que entran en la web, sin importar su género. A pesar de esto se considera que el estudio no va enfocado en lograr la igualdad, y por ello no se ha destacado en la tabla anterior.