UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Desarrollo dirigido por datos: Un ejemplo práctico con ChessLeague

Especialización en Ingeniería del Software

Santiago Del Rey Juárez

Director: Silverio Juan Martínez Fernández
Codirector: Antonio Salmerón Cerdán

21 de junio de 2021

Resumen

Hoy en día cualquier producto de software genera una gran cantidad de datos tanto de manera explícita como implícita, los cuales se pueden aprovechar de muchas formas. Un ejemplo claro es su análisis para extraer información sobre como se está utilizando una aplicación o porque están ocurriendo según qué problemas. El uso de estos datos para guiar el desarrollo de un producto es una práctica cada vez más extendida, a la cual se la conoce como desarrollo dirigido por datos.

En este proyecto estudiamos la utilidad de las redes bayesianas como herramienta de análisis para guiar el desarrollo de un proyecto software. Para ello, hemos creado un caso de estudio donde buscamos mejorar el juego en línea ChessLeague utilizando redes bayesianas como herramienta para analizar *software logs*. En este recopilamos feedback explícito e implícito de distintas fuentes. Este feedback se visualiza de manera exploratoria con *dashboards* y se usa para crear tres modelos con redes bayesianas que nos sirven para analizar el funcionamiento de la aplicación. Gracias a esto, hemos sido capaces de detectar múltiples puntos de mejora dentro del juego, que van desde problemas de rendimiento hasta errores que ocurren de forma recurrente en la aplicación, los cuales se han implementado como tarjetas en Scrum.

Viendo estos resultados consideramos que el uso de redes bayesianas dentro del desarrollo dirigido por datos tiene un gran potencial como herramienta de análisis de *software logs*, y animamos al resto de la comunidad a seguir profundizando en sus posibles aplicaciones.

Finalmente, concluimos que el desarrollo dirigido por datos resulta muy útil a la hora de tomar decisiones sobre el camino que ha de seguir un producto de software. Además, este enfoque nos proporciona una mejor visión de la aplicación, permitiéndonos por ejemplo detectar fallos o puntos débiles que de otra manera seguirían ocultos.

Resum

Avui dia qualsevol producte de programari genera una gran quantitat de dades tant de manera explícita com implícita, els quals es poden aprofitar de moltes formes. Un exemple clar és la seva anàlisi per a extreure informació sobre com s'està utilitzant una aplicació o perquè estan ocorrent segons quins problemes. L'ús d'aquestes dades per a guiar el desenvolupament d'un producte és una pràctica cada vegada més estesa, a la qual se la coneix com a desenvolupament dirigit per dades.

En aquest projecte estudiem la utilitat de les xarxes bayesianes com a eina d'anàlisi per a guiar el desenvolupament d'un projecte de software. Per a això, hem creat un cas d'estudi on busquem millorar el joc en línia Chess-League utilitzant xarxes bayesianes com a eina per a analitzar *software logs*. En aquest recopilem feedback explícit i implícit de diferents fonts. Aquest feedback es visualitza de manera exploratòria amb *dashboards* i s'usa per a crear tres models amb xarxes bayesianes que ens serveixen per a analitzar el funcionament de l'aplicació. Gràcies a això, hem estat capaços de detectar múltiples punts de millora dins del joc, que van des de problemes de rendiment fins a errors que ocorren de manera recurrent en l'aplicació, els quals s'han implementat com a targetes en Scrum.

Veient aquests resultats considerem que l'ús de xarxes bayesianes dins del desenvolupament dirigit per dades té un gran potencial com a eina d'anàlisi de *software logs*, i animem a la resta de la comunitat a continuar aprofundint en les seves possibles aplicacions.

Finalment, concloem que el desenvolupament dirigit per dades resulta molt útil a l'hora de prendre decisions sobre el camí que ha de seguir un producte de programari. A més, aquest enfocament ens proporciona una millor visió de l'aplicació, permetent-nos per exemple detectar fallades o punts febles que d'una altra manera continuarien ocults.

Abstract

Nowadays any software product generates a large amount of data both explicitly and implicitly, which can be leveraged in many ways. A clear example is its analysis to extract information about how an application is being used or why problems are occurring. Using this data to guide product development is an increasingly widespread practice, known as data-driven development.

In this project, we study the usefulness of Bayesian networks as an analysis tool to guide the development of a software project. For this, we have created a case study where we seek to improve the online game ChessLeague using Bayesian networks as a tool to analyse software logs. In this, we collect explicit and implicit feedback from different sources. This feedback is visualized exploratorily with dashboards and is used to create three models with Bayesian networks that serve to analyse the application's operation. As a result, we have been able to detect multiple enhancement points within the game, ranging from performance problems to errors that occur repeatedly in the application, which have been implemented as cards in Scrum.

Seeing these results, we consider that the use of Bayesian networks within data-directed development has great potential as a software logs analysis tool, and we encourage the rest of the community to continue to deepen its possible applications.

Finally, we conclude that data-driven development is very useful in making decisions on the way a software product should follow. In addition, this approach gives us a better insight into the application, allowing us, for example, to detect faults or weaknesses that would otherwise remain hidden.

Agradecimientos

Quisiera aprovechar para expresar mi agradecimiento a todas las personas que durante estos 4 años, de forma desinteresada, me han ayudado a desarrollarme tanto intelectual como personalmente.

A Silverio Juan Martínez, uno de los co-directores del proyecto, por haberme dado la oportunidad de trabajar en una aplicación a la que dedicó tanto tiempo en el pasado, y por su gran ayuda durante la elaboración del proyecto.

A Antonio Salmerón, co-director de este trabajo, por toda su ayuda y consejos que me han servido como guía a lo largo de todo el proyecto.

A mis padres, por su apoyo incondicional durante todos los años de la carrera.

A mis compañeros de clase, en especial a Albert, Enric, Marc, Miguel y Xavier por lo que me han enseñado durante estos años y por hacer este trayecto más agradable.

A mis profesores de la universidad, los cuales me han enseñado los conocimientos necesarios para completar este trabajo.

A mis amigos y familia, que me siempre me han apoyado y han estado ahí en los buenos y en los malos momentos.

Y a la universidad, por haberme concedido una beca IniRec y haber financiado parcialmente este proyecto.

Índice general

Índice de figuras

Índice de tablas

Índice de código fuente

1. Introducción

En la sociedad actual el uso de todo tipo de aplicaciones software está interiorizado en nuestro día a día desde que nos levantamos hasta que nos acostamos. Si nos paramos a pensar, utilizamos todo tipo de software a lo largo de nuestro día, ya sea para ver un vídeo mientras vamos al trabajo, enviar un mensaje o leer las noticias.

A pesar de estar utilizando de manera continua estas aplicaciones muchos no nos damos cuenta de la magnitud de trabajo y personas implicadas en el mantenimiento y la constante mejora de dichas aplicaciones.

Otro aspecto importante y que ha estado tomando relevancia a lo largo de los últimos años es el gran valor que aportan los datos que generamos al utilizar todas esas aplicaciones. Todos conocemos ejemplos como el de Google, que analiza las páginas que visitamos para ofrecernos anuncios y sugerencias personalizadas [1].

¿Entonces, si los datos que generamos son tan valiosos y aportan tanta información no se podrían utilizar para hacer más fácil el desarrollo y mantenimiento de las aplicaciones? Lo cierto es que la gran mayoría de empresas ya recopilan información de sus aplicaciones para detectar fallos y solucionarlos. ¿Pero no sería más interesante obtener datos para ver cómo se utiliza dicho software y ver cómo mejorarlo a partir de ahí?

En este proyecto buscamos mejorar una aplicación web llamada ChessLeague mediante la recopilación y el análisis de diferentes datos obtenidos dentro de esta, con el objetivo de comprobar la utilidad que tiene esta información sobre el proceso de desarrollo de una aplicación que ya se encuentra en producción.

1.1. Qué es ChessLeague

ChessLeague es un juego en línea de ajedrez que simula competiciones por equipos donde los jugadores gestionan su propio club. Los usuarios han de realizar tareas como: la compra-venta de ajedrecistas, el fichaje de entrenadores o gestionar la alineación del equipo antes de los encuentros, con el objetivo de acumular puntos durante las diferentes jornadas y llegar así a ganar la liga.

ChessLeague se desarrolló en el año 2010 como un trabajo de fin de grado por el actual director de este proyecto Silverio Juan Martínez Fernández y dirigido por el actual codirector de este trabajo Antonio Salmerón Cerdán [2]. El juego se mantuvo en funcionamiento por un período de 2 años, alcanzando una base de unos 2000 jugadores.

Figura 1.1: Página principal de ChessLeague. Fuente: www.chessleague.es.

1.2. Contenido de la memoria

Esta memoria consiste en un Trabajo de Final de Grado perteneciente al Grado en Ingeniería Informática impartido por la Facultad de Informática de Barcelona.

En esta se recogen los pasos necesarios para llevar a cabo el desarrollo de una aplicación software utilizando distintos análisis de datos de diversas fuentes. En los siguientes apartados se detallan las distintas etapas de obtención y análisis de los datos, así como del desarrollo.

La memoria comienza con una introducción, donde se describe el anteproyecto. Le sigue el capítulo 2, donde se introducen conceptos previos y se describe el estado del arte y las partes implicadas. A continuación, en los capítulos 3 y 4 se detallan el alcance del proyecto y la metodología utilizada durante este. En los capítulos 5 y 6 se describe la planificación temporal del trabajo y se elabora un presupuesto detallado. El capítulo 7 muestra los pasos seguidos para obtener los datos de uso utilizados posteriormente. En el capítulo 8 se describen las fuentes de datos utilizadas en el proyecto junto con los resultados de un análisis preliminar. A continuación, el capítulo 9 detalla el proceso seguido para generar los distintos modelos para analizar los datos. Acto seguido, en el capítulo 10 se muestran los cambios realizados en ChessLeague a partir de los datos obtenidos. En el capítulo 11 se detallan los resultados que se desprenden de la gestión del proyecto. Finalmente, el capítulo 12 concluye el documento con conclusiones, integración de conocimientos y trabajo futuro.

1.3. Partes del proyecto

Este proyecto consta de tres partes bien diferenciadas las cuales se pueden ver en la Figura 1.2.

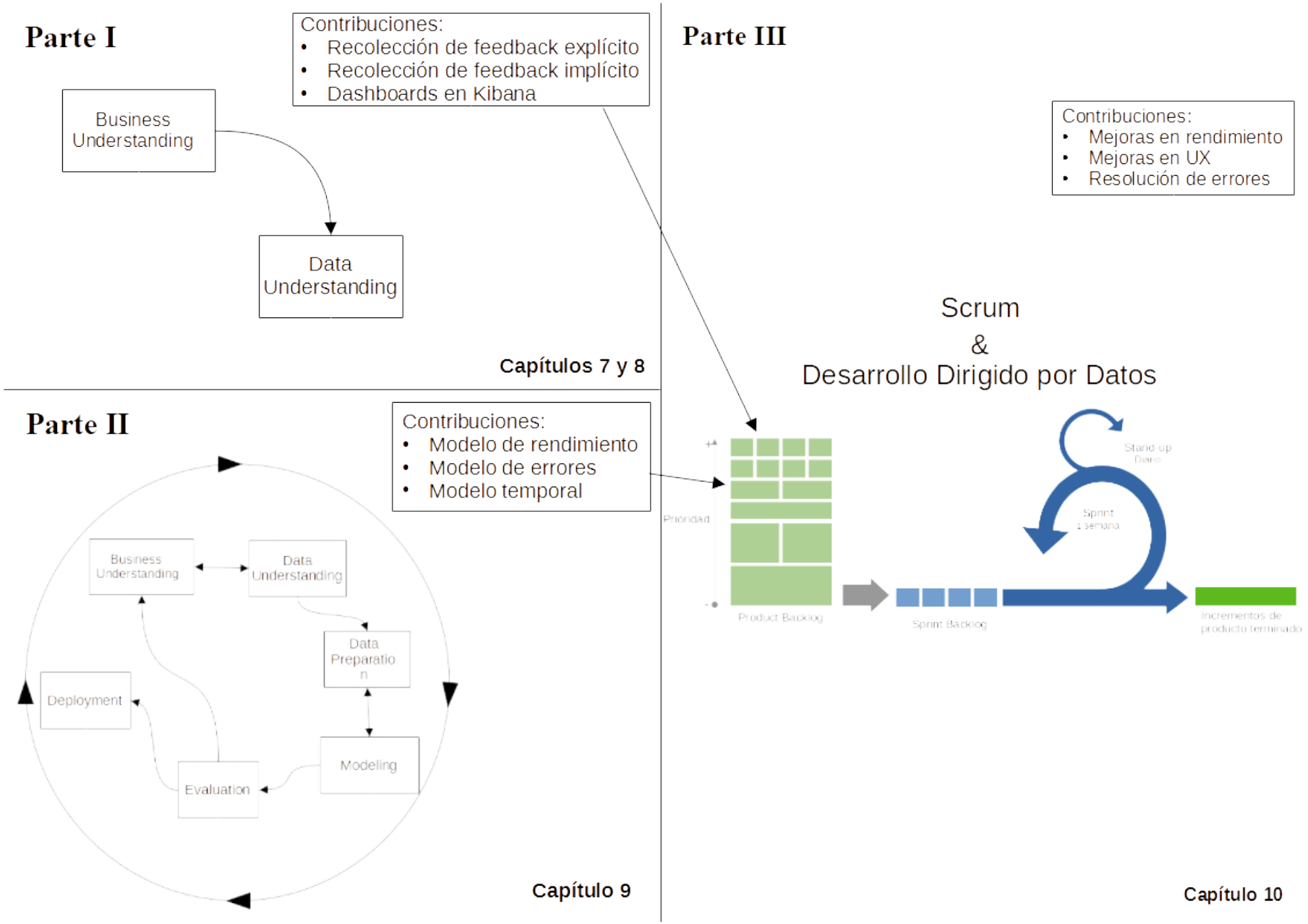

Figura 1.2: Partes del proyecto. Fuente: Elaboración propia

1.3.1. Parte I

La primera parte se lleva a cabo durante los capítulos 7 y 8. Estos se corresponden con las etapas de Business Understanding y Data Understanding de la metodología CRISP-DM.

En concreto, el capítulo 7 se corresponde con el Business Understanding. En este capítulo se crea la infraestructura necesaria para recopilar los datos de uso, para ello hemos de analizar los objetivos de negocio de este proyecto. Como resultado, finalizamos este capítulo con una serie de preguntas que queremos resolver con este trabajo y con toda la infraestructura necesaria para recopilar los datos.

La segunda etapa de esta parte es el Data Understanding, que se corresponde con el capítulo 8 de la memoria. En este capítulo, se analizan las distintas fuentes de datos de las que disponemos, elaborando una descripción de los distintos tipos de datos que contiene cada una. Con esto pretendemos hacernos una idea de como hemos de manipular los datos para conseguir los objetivos establecidos anteriormente, o de si necesitamos nuevos datos. Una vez sabemos con qué tipos de datos estamos trabajando, pasamos a utilizar Kibana para elaborar distintas visualizaciones que nos aporten información sobre el estado de los datos.

1.3.2. Parte II

Una vez terminamos la parte anterior pasamos a la parte de creación de los modelos, descrita en el capítulo 9. En esta se ha vuelto a repetir la metodología CRISP-DM, pero en este caso hemos realizado el ciclo completo.

Este capítulo se centra especialmente en las etapas de Data Preparation, Modeling y Evaluation. En este se muestra el proceso que se ha seguido durante las distintas iteraciones para limpiar los datos de posibles valores que puedan introducir ruido en el análisis, así como las nuevas variables introducidas. Una vez explicada la etapa de Data Preparation, pasamos a explicar las etapas de Modeling y Evaluation para cada uno de los tres modelos generados. En estas se muestra el proceso seguido para la elaboración de los distintos modelos, y después se hace una descripción detallada de cada uno de ellos y de sus posibles aplicaciones.

Como resultado de esta parte tenemos la creación de tres redes bayesianas. La primera nos permite analizar el rendimiento de la aplicación, la segunda sirve para analizar el comportamiento de los errores producidos en el servidor y la tercera para analizar el comportamiento de la aplicación en distintos instantes temporales.

1.3.3. Parte III

La última parte se corresponde con el capítulo 10 de la memoria. En esta se utiliza el desarrollo dirigido por datos, el cual se alimenta de las dos partes anteriores, siguiendo la metodología de trabajo Scrum. Esta parte se inicia con la elaboración de un Product Backlog (PB, de sus siglas en inglés) a partir de todos los datos que hemos obtenido, desde el feedback explícito generado con la primera parte hasta los modelos creados en la segunda parte. Esto nos da como resultado un PB con 27 tarjetas en el momento de redactar esta memoria. Finalmente, se

detalla cada uno de los sprints realizados. Enumerando las tareas realizadas en cada uno de ellos y describiendo las principales modificaciones introducidas.

Como resultado de esta última parte tenemos una versión mejorada de Chess-League. Esta incluye optimizaciones en rendimiento y tratamiento de errores, así como nuevas funcionalidades y cambios en la interfaz de usuario.

2. Contexto

En este capítulo se describen una serie de conocimientos previos necesarios. A continuación, se presentan las partes implicadas y terminamos describiendo el estado del arte.

2.1. Conceptos previos

Para entender correctamente este trabajo el lector ha de estar familiarizado con los siguientes conceptos.

Software analytics

Se entiende como *software analytics* el proceso por el cual se busca obtener información de valor a partir de los distintos artefactos generados durante el desarrollo de software así como a partir de los implicados en dicho desarrollo, con el objetivo final de ayudar en las tareas de desarrollo.

Existen diversas definiciones para este concepto dependiendo del enfoque de los datos a analizar, pero todas coinciden en el mismo objetivo, que es obtener información que ayude al desarrollo final del producto. Entre algunas de las definiciones podemos encontrar las siguientes:

- *"Software analytics aims to obtain insightful and actionable information from software artifacts that help practitioners accomplish tasks related to software development, systems, and users."* [3], donde se centra en el análisis aplicado a los artefactos que componen un software.

- *"Software analytics is analytics on software data for managers and software engineers with the aim of empowering software development individuals and teams to gain and share insight form their data to make better decisions"* [4], donde centra el foco tanto en los artefactos del software como en las actividades que llevan acabo los desarrolladores y equipos involucrados en el desarrollo.

Desarrollo dirigido por datos

Tal y como su nombre indica, el desarrollo dirigido por datos o *data-driven development* es un enfoque que se basa en la obtención y análisis de datos para dirigir el camino de desarrollo de un producto. A diferencia del tradicional desarrollo dirigido por objetivos, donde las metas y nuevas características que van dando valor al producto se originan a partir de nuevas ideas que van apareciendo a medida que el producto evoluciona, sin saber con certeza si lo que se está desarrollando va a aportar realmente valor añadido o si por el contrario quedará en el olvido en cuestión de meses, el desarrollo dirigido por datos busca obtener toda la información posible, tanto de los artefactos creados durante el desarrollo como de los propios usuarios del producto, y tras analizar toda esta información obtener una idea de en qué dirección se dirige el proyecto de manera que se puedan establecer unas metas y objetivos que se alineen con el estado real del producto.

Mining Software Repositories

El campo de la minería de repositorios de software (MSR, de sus siglas en inglés) se centra en analizar la gran variedad de datos que se generar en dichos repositorios de software para descubrir información interesante y útil sobre los sistemas y proyectos software [5].

Anualmente se celebra una conferencia donde los investigadores de este campo pueden presentar sus descubrimientos o los conjuntos de datos que han creado para que puedan ser utilizados en otros proyectos. Además, se realizan varias presentaciones de investigadores invitados al igual que varios talleres relacionados con el campo del MSR.

Deuda técnica

El término de deuda técnica en el mundo del desarrollo software fue acuñado por primera vez por Ward Cunningham hace aproximadamente tres décadas para explicar a los *stakeholders* sin conocimientos técnicos la necesidad de aplicar lo que hoy en día conocemos como *refactoring* [6].

Podemos considerar la deuda técnica como algo similar a la deuda financiera, esta deuda surge de la tendencia que tiene el desarrollo de software en crear código de baja calidad o en obviar pruebas de calidad, entre otros, a causa de un calendario ajustado o por falta de presupuesto, por ejemplo. Todo esto acaba finalmente llevando a un incremento en la dificultad y el tiempo de desarrollo, ya que se acaba teniendo que lidiar con errores o código mal estructurado que en un principio se podría haber evitado.

Feedback explícito e implícito

Dentro de una aplicación software podemos diferenciar entre dos tipos de feedback [7].

Por un lado, tenemos el feedback explícito, el cual entendemos como aquel que es proporcionado directamente por los usuarios del sistema. Este tipo de feedback puede venir en múltiples formatos, desde texto hasta imágenes, grabaciones de audio, vídeos u otros tipos. De igual manera, las fuentes desde donde se obtiene dicha información pueden ser variadas, yendo desde foros hasta redes sociales (en nuestro caso Facebook o Twitter) o sistemas de tickets.

Por el otro lado, tenemos el feedback implícito, el cual entendemos como aquel que se obtiene de recopilar información del uso del sistema mientras es utilizado por los usuarios. La principal diferencia con el anterior, es que los datos vienen de los usuarios sin que haya una comunicación explícita por su parte, pero con su consentimiento explícito.

Dentro del feedback implícito podemos distinguir distintos tipos, de los cuales en este proyecto nos centraremos en dos. El primero es la calidad del servicio (QoS, de sus siglas en inglés), el cual incluye atributos como el tiempo de respuesta, disponibilidad y seguridad, lo que puede proporcionar información útil para entender que partes del sistema necesitan ser mejoradas.

El segundo tipo son los datos de uso. Estos incluyen las páginas visitadas por un usuario, interacciones con la interfaz de usuario, etc. Estos datos pueden resultar útiles para detectar patrones de uso que pueden servir para descubrir funcionalidades en desuso o para reorganizar la interfaz de usuario para extraer el mayor valor posible a esta.

Modelos en grafo

Los modelos en grafo o modelos grafo probabilísticos (PGM, de sus siglas en inglés) son modelos estadísticos que expresan una estructura de dependencia condicional entre múltiples variables aleatorias. Estas representaciones se encuentran entre los campos de la estadística y la computación, utilizando conceptos de la teoría de la probabilidad, algoritmos de grafos, *machine learning*, y más [8]-[10].

Existen multitud de PGM, pero uno de los más conocidos y, por lo tanto, más utilizados son las redes bayesianas.

Una red bayesiana es un grafo acíclico dirigido (DAG, de sus siglas en inglés) con nodos representado variables observables o latentes del modelo, y las aristas representan dependencias condicionales [8], [11].

Dentro de las redes bayesianas tenemos tres tipos de conexiones fundamentales, que determinan como se transmite la evidencia a través de una variable [12].

Conexiones en serie

A tiene influencia sobre B, el cual a su vez, tiene influencia sobre C. Por otra parte, si se conoce el estado de B, la transmisión se bloquea, entonces A y C se vuelven independientes. Se dice que A y C están *d-separados* dado B.

En resumen, la evidencia puede transmitirse a través de una conexión en serie a menos que el estado de la variable en la conexión sea conocido.

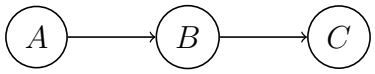

Figura 2.1: Conexión en serie. Fuente: Elaboración propia

Conexiones divergentes

La evidencia sobre cada hijo de B puede pasar a través de este, hacia cualquier otro hijo a menos que el estado de B sea conocido, en cuyo caso bloquea el camino, y todos sus hijos pasan a ser independientes.

Por lo tanto, la evidencia puede transmitirse a través de una conexión divergente salvo que la variable en la conexión esté instanciada.

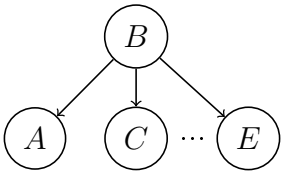

Figura 2.2: Conexión divergente. Fuente: Elaboración propia

Conexiones convergentes

A puede influir sobre cada uno de sus padres, pero si no se conoce nada sobre A, entonces los padres $B, C, ..., E$, son independientes. Sin embargo, cualquier evidencia sobre A, convertirá a sus padres en condicionalmente dependientes dado A.

En resumen, la evidencia puede transmitirse a través de una conexión convergente si la variable en la conexión o alguno de sus descendientes ha recibido evidencia.

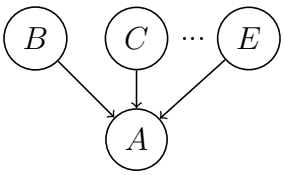

Figura 2.3: Conexión convergente. Fuente: Elaboración propia

2.2. Partes implicadas

Pese a ser un proyecto de tamaño reducido, éste implica a distintos grupos, ya sea de manera directa o indirecta, dependiendo del tipo de beneficio que extraigan de él. A continuación se listan los diferentes actores implicados, *stakeholders* en inglés, ya sean personas interesadas que se beneficiarán, personas afectadas o participantes en el desarrollo:

Usuarios

Los usuarios son las personas que utilizan ChessLeague desde cualquier dispositivo. Estos son los principales beneficiados del proyecto, así como la principal fuente de datos.

Roles:

- Utilizar la aplicación.
- Publicitar la aplicación e invitar a amigos.

Objetivos:

- Disfrutar del juego sin encontrarse con errores.

Creador de ChessLeague

El creador de ChessLeague es Silverio Juan Martínez Fernández.

Roles:

- Proporcionar acceso al código fuente de la aplicación.
- Determinar que funcionalidades hay que incorporar o modificar.

Objetivos:

- Mejorar la calidad de la aplicación.
- Incrementar la popularidad de la aplicación y su base de usuarios.

ANW Hosting

ANW Hosting[1]es la compañía de *hosting* que aloja la aplicación ChessLeague.

Roles:

- Ofrecer una plataforma de *hosting* donde poder desplegar la aplicación.
- Ofrecer acceso a una base de datos donde poder almacenar la información de la aplicación.
- Dar soporte en caso de algún fallo al desplegar la aplicación.

Objetivos:

- Aumentar sus ganancias con la subscripción a su servicio.

Directores del proyecto

Son los profesores encargados de supervisar el proyecto y procurar que vaya en buena dirección.

Roles:

- Supervisar la correcta elaboración del proyecto.
- Asesorar sobre los distintos temas tratados en el trabajo.

Objetivos:

- Asegurar que se cumplen los objetivos establecidos por la rúbrica de evaluación.
- Procurar que el trabajo obtenga la máxima nota posible.

Desarrollador

Es la persona encargada de desarrollar este proyecto. Dado que el proyecto es de carácter individual, la misma persona interpreta todos los roles: analista de datos, programador, tester, etc.

Roles:

- Elaborar la memoria y el software que forma este proyecto cumpliendo los estándares de calidad.
- Seguir las indicaciones de los tutores i aplicarlas en el trabajo.

Objetivos:

- Demostrar que se tienen los conocimientos esperados de un Ingeniero del Software aplicando los conocimientos obtenidos durante el transcurso del grado.
- Obtener y asimilar nuevos conocimientos para la correcta elaboración de este proyecto.
- Obtener la máxima nota posible.

2.3. Estado del arte

Durante el proceso de desarrollo del software se generan varios tipos de datos de manera natural, como por ejemplo el código fuente, informes de errores o tests de calidad. A su vez, a medida que crece el uso de servicios software gracias a su fácil acceso hoy en día, nos encontramos con que cada vez disponemos de más y más datos a nuestro alcance que nos dan información del estado de un programa en tiempo de ejecución.

[1] https://www.anw.es

Durante la última década el uso de todos estos datos aplicados al desarrollo de software ha ido incrementando su relevancia. Esto ha dado origen a nuevos planteamientos a la hora de desarrollar software que pretenden aprovechar al máximo la información que se puede obtener de dichos datos, con el objetivo de facilitar y mejorar el proceso de desarrollo.

En este trabajo nos centramos en la aplicación de *software analytics* en un entorno de desarrollo ágil. Su utilización en este entorno se ha discutido en un gran número de trabajos, y tal y como se indica en [13] ha habido un incremento en el interés que este tema suscita en la comunidad científica. Como fruto de este interés han surgido una gran cantidad de investigaciones que cubren un amplio abanico de temas, como las posibles aplicaciones que puede tener el uso de *software analytics* en el desarrollo de software [4], [14] o propuestas de nuevos procedimientos y técnicas [15], entre otros.

Dentro de la gran cantidad de áreas que engloba el *software analytics*, nos enfocaremos en el uso de datos basados en registros para la monitorización de una aplicación. A pesar de que este tema ya ha sido explorado por una cantidad considerable de trabajos, como se menciona en [16], no encontramos que se hayan publicado ejemplos prácticos del uso de redes bayesianas como modelos para evaluar el rendimiento y comportamiento de una aplicación en producción.

Por este motivo, creemos que con este trabajo podemos aportar un ejemplo del uso de PGMs en una aplicación a pequeña escala que está en un entorno en producción. Además, en este proyecto nos centramos en la obtención y el análisis de información obtenida de los usuarios, ya que actualmente la gran mayoría de estudios, como [17]-[19], se centran en los datos de desarrollo. Con esto pretendemos demostrar la utilidad y el beneficio que puede aportar generar información que nos permita entender cómo los usuarios utilizan un producto software, así como ofrecer una serie de lecciones aprendidas durante el transcurso del proyecto.

3. Alcance del proyecto

En este capitulo se presenta el problema a resolver junto con los objetivos principales del proyecto. Finalmente se discuten los posibles riesgos y obstáculos que pueden aparecer durante la elaboración de este trabajo.

3.1. Problema a resolver

A pesar de que existen una gran variedad de estudios en *software analytics* y *data-driven development*, no consideramos que haya un gran número de éstos que se centren en el impacto que tienen ambos enfoques dentro del propio proceso de desarrollo en la práctica. Otro aspecto a tener en cuenta, es que la mayoría de dichos estudios se centran en los propios datos de desarrollo dejando de lado los datos que se pueden obtener de los usuarios, ya sea por la dificultad de acceder a éstos debido a las políticas de privacidad o por otros motivos.

En este proyecto buscamos aportar nueva información en los aspectos mencionados en el párrafo anterior, aprovechando que disponemos de una mayor libertad para trabajar con datos tanto de desarrollo como de los propios usuarios. Por lo tanto, en este trabajo se construyen tres PGMs utilizando redes bayesianas con el objetivo de entender qué elementos llevan a una pérdida de rendimiento en la aplicación web y cómo los usuarios están usando la aplicación.

3.2. Objetivos

El objetivo principal de este proyecto es mejorar el juego en linea ChessLeague, tanto en rendimiento como en usabilidad, mediante el análisis de datos y así poder observar las ventajas e inconvenientes del desarrollo dirigido por datos en la práctica. Para conseguir esto hemos creado tres grupos de sub-objetivos, que se corresponden con las tres partes que forman este proyecto.

Recolección y visualización de los datos

- **SO1:** Crear un sistema de recolección de feedback explícito.
- **SO2:** Crear un sistema de recolección de feedback implícito.
- **SO3:** Crear un *dashboard* con Kibana para visualizar los datos.

Análisis de software logs con Redes Bayesianas

- **SO4:** Crear un PGM para medir el rendimiento de la aplicación.
- **SO5:** Crear un PGM para analizar el comportamiento de los errores.
- **SO6:** Crear un PGM para analizar el comportamiento de la aplicación a lo largo del tiempo.

Desarrollo dirigido por datos

- **SO7:** Incrementar en un 20 % las tareas del *backlog* provenientes del análisis de los datos.
- **SO8:** Mejorar la detección de errores en la aplicación.

3.3. Riesgos y obstáculos

A lo largo del proyecto pueden surgir diversos riesgos y obstáculos que pueden afectar al desarrollo del proyecto con los que hemos de lidiar.

- **Fecha de entrega:** Existe una fecha de entrega establecida a priori, lo cual deja poco margen de maniobra en caso de imprevistos, por lo tanto, hay que realizar una buena planificación de las tareas que se van a realizar.
- **Inexperiencia en el campo:** Es la primera vez que el autor del trabajo investiga en las áreas de *software analytics* y *data-driven development*, así como es la primera vez que realiza tareas de análisis de datos pertenecientes al campo de la ciencia de datos.
- **Disponibilidad del servicio de *hosting*:** La aplicación web se encuentra desplegada en un servicio de *hosting* web ajeno al autor, por lo que en caso de caída del servidor o suspensión por mantenimiento la disponibilidad de la aplicación depende totalmente de la empresa de *hosting*.
- **Base de usuarios insuficiente:** Si no existe una base de usuarios lo suficiente mente grande para generar una cantidad de datos aceptable el análisis de datos puede no resultar concluyente, lo que implicaría una mayor dificultad a la hora de seguir un desarrollo dirigido por datos.
- **Corrupción de la base de datos:** Es posible que por algún error en la propia aplicación o por otros motivos la base de datos quede corrompida inutilizando todo el sistema.

4. Metodología y rigor

En este capítulo se presenta la metodología de trabajo que se ha utilizado para la realización del proyecto, junto con las principales herramientas utilizadas.

4.1. Metodología

En el mundo del desarrollo software existen una gran variedad de metodologías de trabajo, cada una ideada para ser aplicada en distintas situaciones dependiendo del problema a afrontar. Dado que el proyecto consta de tres partes bien diferenciadas, las dos primeras dedicadas al análisis de datos y la tercera al desarrollo, se ha tomado la decisión de aplicar dos metodologías de trabajo distintas. A lo largo de la primera parte del proyecto se utiliza parcialmente *CRISP-DM* [20], que esta orientada a problemas de análisis de datos, en concreto las fases de *Business Understanding* y *Data Understanding*. En la segunda parte, se vuelve a utilizar *CRISP-DM*, pero ahora completando el ciclo. Finalmente, durante la parte final del proyecto, orientada al desarrollo, se sigue una metodología *Agile* [21], en concreto el marco de trabajo *Scrum* [22], [23].

La motivación de usar dos metodologías de trabajo distintas de manera simultánea viene dada por la necesidad de usar una metodología orientada a resolver un problema del campo de la ciencia de datos, ya que ese es el área al que pertenece el principal objetivo de este trabajo, así como la necesidad de encontrar una metodología que aporte flexibilidad y una cierta rapidez a la hora de generar artefactos que incrementen el valor del producto en un corto periodo de tiempo.

4.2. Validación de los resultados

Para determinar si se han alcanzado los objetivos con éxito una vez finalizado el proyecto se ha de establecer un método para la validación de los resultados. En este caso, se considera que los sub-objetivos de **recolección y visualización de los datos** se han cumplido si en el momento de la finalización del proyecto se han creado métodos para la recolección de feedback explícito e implícito, además de disponer de un *dashboard* en Kibana. Los sub-objetivos de **análisis de software logs con redes bayesianas** se considerarán alcanzados si para la fecha límite de este proyecto se han creado tres modelos en grafo funcionales que permitan evaluar el rendimiento de la aplicación, averiguar donde se originan los errores y comprender como se utiliza la aplicación. Finalmente, el sub-objetivo **SO7** se

validará contabilizando las tarjetas creadas en Jira que provengan del análisis de datos, y el sub-objetivo **SO8** se considerará completado si somos capaces de detectar nuevos errores gracias a Kibana y los modelos creados.

4.3. Herramientas

Con el objetivo de facilitar tanto la aplicación de la metodología *Scrum* como el mismo desarrollo del proyecto se utilizarán diversas herramientas.

Gestión del proyecto

- **Jira:** Jira [24] es un software de gestión de proyectos e incidencias gratuito. Éste tiene funcionalidades como la creación de épicas e historias de usuario o el seguimiento de incidencias, además de disponer de integraciones con los principales gestores de repositorios como Bitbucket, GitHub o GitLab.

Sistema de Control de Versiones

- **Git:** Git [25] es un sistema de control de versiones gratuito y de código abierto, que permite el desarrollado de proyectos software de una manera cooperativa facilitando la gestión de versiones con el uso de repositorios de código y ramas, entre otras muchas funcionalidades.

- **GitHub:** GitHub [26] es uno de los gestores de repositorios Git más utilizados en este momento, ya que además de gestionar los repositorios ofrece otras funcionalidades como la integración y despliegue continuo y el control de errores, entre otros.

Fuentes de datos

- **Registros de uso:** ChessLeague dispone de 2 tipos de registros internos, el primero es generado por el propio servidor y registra las trazas de los errores y toda la información que tiene como destino la consola. El segundo tipo se ha creado manualmente y registra la actividad de los usuarios dentro de la aplicación.

- **Sugerencias:** Los usuarios pueden aportar sugerencias para mejorar la aplicación o reportar errores. Estas sugerencias pueden provenir tanto de la propia aplicación como de las redes sociales como Facebook y Twitter.

- **Google Analytics:** Google Analytics [27] es un servicio web de análisis ofrecido por Google que permite rastrear y analizar el tráfico dentro de una aplicación web.

Tratamiento de los datos

- **Python:** Python [28] es un lenguaje interpretado de alto nivel cuya filosofía se enfatiza en la legibilidad del código. Éste es uno de los lenguajes más utilizados en el campo de la ciencia de datos debido a su fácil curva de aprendizaje así como por la gran cantidad de librerías disponibles para este campo.

- **R:** R [29], [30] es un entorno de desarrollo gratuito para la computación estadística y la generación de gráficos. Éste es otro de los lenguajes más utilizados en el campo de la ciencia de datos para el desarrollo de software estadístico y el análisis de datos.

- **Logstash:** Logstash [31] es una herramienta gratuita y *open-source* para el procesamiento de datos que obtiene dichos datos de múltiples fuentes, los transforma y los envía a otras herramientas para su posterior uso.

- **Kibana:** Kibana [31] es una herramienta gratuita y *open-source* que permite la visualización de datos de Elasticsearch en *dashboards*.

- **Hugin:** Hugin [32] es una herramienta para la manipulación y visualización de redes bayesianas.

4.4. Legislación

Puesto que este proyecto recoge y analiza datos extraídos del uso de la aplicación, hay que seguir una serie de leyes para asegurar que no se incurre en ningún tipo de violación de la privacidad de los usuarios, así como asegurar el correcto nivel de protección para estos datos. En concreto, este proyecto se atiene a las vigentes RGPD [33] y LOPDGDD [34]. Aun así, hay que destacar que actualmente la única información de carácter sensible que se almacena son el nombre y apellidos de los usuarios, los cuales son únicamente visibles para el propio usuario y para los administradores de la base de datos.

5. Gestión del proyecto

En este capítulo se hace un análisis de las tareas a realizar durante el proyecto y de los recursos necesarios. Finalmente se analizan los riesgos vistos en la seccion 3.3 y como han de gestionarse.

5.1. Descripción de las tareas

Este proyecto tiene una duración aproximada de cuatro meses y medio, iniciando el 23 de febrero con la gestión del proyecto y finalizando a finales del mes de junio, aproximadamente el día 25. Por lo tanto el proyecto tiene una duración de 18 semanas, de las cuales hay 82 días lectivos y 44 no lectivos. Considerando una carga de trabajo media de 30 horas semanales, el proyecto constará de 540 horas de trabajo aproximadamente.

5.1.1. Tareas

Gestión del proyecto

- **GP1 – Contextualización y alcance (25h):** Definición del alcance del proyecto en el contexto de su estudio. Indicando el objetivo del proyecto, su justificación y la metodología a seguir.
- **GP2 – Planificación temporal (9h):** Planificación del proyecto, detallando las tareas y los recursos necesarios, así como la gestión de riesgos.
- **GP3 – Presupuesto y sostenibilidad (10h):** Elaboración del presupuesto del proyecto y un análisis de sostenibilidad.
- **GP4 – Integración del documento final (19h):** Elaboración de un documento integrando los documentos anteriores corrigiendo posibles errores o actualizando la información.
- **R – Reuniones de control (18h):** Reuniones de control con los tutores del proyecto para ver que éste se desarrolla de manera correcta. Se distribuyen a lo largo del proyecto de manera que hay una reunión semanal de una hora.

Visualización y análisis preliminar de los datos

- **VA1 – Obtención de los datos (18h):** Obtención de los registros de uso, tanto los generados automáticamente como los creados manualmente, los datos generados por Google Analytics y puesta en marcha de SonarCloud.
- **VA2 – *Business understanding* y *Data understanding* (9h):** Elaboración de los objetivos de negocio y los criterios de éxito del proyecto, junto con una descripción de las principales fuentes de datos y de las variables a utilizar en el estudio.
- **VA3 – Selección de los datos (15h):** Selección de los datos a utilizar para el análisis, eliminando tanto atributos como registros no relevantes para el estudio.
- **VA4 – Limpieza de los datos (15h):** Mejora de la calidad de los datos a través de distintas técnicas de manipulación de datos como: la selección de subconjuntos de datos o la inserción de valores por defecto.
- **VA5 – Construcción de datos (20h):** Creación de nuevos datos mediante la aplicación de distintas técnicas como: obtención de atributos derivados, la creación de nuevos registros, o la transformación de atributos existentes.
- **VA6 – Integración de los datos (10h):** Integración de las distintas fuentes de información en una o varias tablas.
- **VA7 – Aprender a utilizar Logstash y Kibana (6h):** Aprender a utilizar las herramientas Logstash y Kibana para su posterior utilización.
- **VA8 – Análisis inicial de los datos (20h):** Creación de un *dashboard* en Kibana para visualizar los datos de uso de la aplicación.

Modelado de los datos

- **MD1 – Familiarización con las redes bayesianas y PGMs (30h):** Investigar los principales conceptos y el funcionamiento de las redes bayesianas y los PGMs.
- **MD2 – Aprender R (18h):** Aprender el lenguaje de programación R, debido a su necesidad para la realización de algunos análisis.
- **MD3 – Construcción de un modelo para la predicción del rendimiento (60h):** Construcción de un modelo con redes bayesianas para entender/predecir del rendimiento de la aplicación.
- **MD4 – Construcción de dos modelos para la detección de comportamientos anómalos (60h):** Construcción de dos PGMs para la detección de comportamientos anómalos en la aplicación.

Implementación de mejoras

- **IM1 – Generación de un *product backlog* en Jira (6h):** Creación de tareas en el *product backlog* de Jira basadas en el análisis de los datos.
- **IM2 - Definición de completo (6h):** Elaboración de la lista de condiciones para dar por completada una tarea del *backlog*.
- **IM3 - Implementación de mejoras (80h):** Implementación de las tareas del *product backlog*.

Elaboración de la memoria

- **EM1 – Documentación de la fase de visualización y análisis preliminar de los datos (30h):** Creación de la documentación perteneciente a la fase del estudio de los datos.
- **EM2 – Documentación de la fase del modelado de los datos (24h):** Creación de la documentación perteneciente a la fase del modelado de los datos.
- **EM3 – Documentación de la fase de implementación de mejoras (18h):** Creación de la documentación perteneciente a la fase de implementación de mejoras.
- **EM4 – Preparación de la defensa del proyecto (24h):** Preparación de la presentación final del proyecto.

5.1.2. Recursos

A continuación se presentan los recursos necesarios para llevar a cabo el proyecto.

Recursos humanos

Siendo que este proyecto es un Trabajo de Fin de Grado, hay únicamente tres personas implicadas en su desarrollo que interpretan diferentes roles dependiendo de la tarea. Por lo tanto solo se consideran como recursos humanos a los directamente implicados que son:

- **[S] Santiago Del Rey Juárez:** Autor del proyecto y desarrollador del mismo que interpreta los roles de: jefe de proyecto, *business analyst*, *data scientist*, desarrollador y tester.
- **[SJ] Silverio Juan Martínez Fernández:** Director del proyecto que supervisa el desarrollo del proyecto y el análisis de datos; e interpreta el rol de *product owner* y en ocasiones el de *data scientist* para ofrecer consejos y resolver dudas.

- **[A] Antonio Salmeron Cerdán:** Codirector del proyecto que supervisa las fases de análisis de los datos e interpreta el rol de *product owner* y en ocasiones el de *data scientist* para ofrecer consejos y resolver dudas.

A continuación se da una breve descripción de los roles:

- **[JP] Jefe de proyecto:** Es el encargado de identificar las necesidades del cliente y los objetivos de negocio, además de gestionar al equipo para cumplir con esos objetivos.
- **[PO] *Product owner*:** Es la persona encargada de representar los intereses de los *stakeholders*, por lo tanto, es el encargado de maximizar el valor del producto.
- **[BA] *Business analyst*:** Es el encargado de analizar el dominio y documentar sus procesos, productos, servicios y sistemas para mejorarlos a través del análisis de datos.
- **[DS] *Data scientist*:** Es el encargado de extraer conocimientos e ideas de datos estructurados y no estructurados, y aplicar conocimientos e ideas prácticas a partir de los datos en una amplia gama de dominios.
- **[D] Desarrollador:** Es el encargado de escribir el código necesario para crear las nuevas funcionalidades y para solucionar posibles errores en el software.
- **[T] Tester:** Es el encargado de escribir los tests necesarios para asegurar el correcto funcionamiento del software.

Recursos materiales

Para el correcto desarrollo del proyecto se necesitan una serie de recursos materiales indispensables como: un ordenador, un monitor, un teclado, un ratón, etc. Además se necesitan varios recursos específicos que se indican a continuación.

- **[ANW] ANW Hosting:** Plataforma donde desplegar la aplicación web.
- **[AT] Atenea:** Canal de comunicación con el tutor de GEP y fuente de información para la fase de gestión de proyecto.
- **[GA] Google Analytics:** Herramienta para la generación de datos de uso de la aplicación.
- **[GIT] Git, GitHub:** Herramientas para el control de versiones y repositorios.
- **[J] Jira:** Herramienta para gestionar las tareas del proyecto.
- **[JR] JabRef:** Herramienta para la gestión de referencias.
- **[K] Kibana:** Herramienta para la visualización de gráficos.

- **[LO] LibreOffice:** Herramienta para la redacción preliminar de la documentación.
- **[LS] Logstash:** Herramienta para el procesamiento de registos.
- **[MEET] Google Meet:** Herramienta para llevar a cabo las reuniones de control.
- **[NB] Netbeans IDE:** IDE para el desarrollo de la aplicación web en Java y JavaScript.
- **[PC] PyCharm IDE:** IDE para la preparación de los datos en Python.
- **[PL] ProjectLibre:** Herramienta para la elaboración del diagrama de Gantt.
- **[RS] RStudio:** IDE para la el análisis de los datos en R.
- **[H] Hugin:** Herramienta para la manipulación de redes bayesianas.
- **[TEX] Texmaker:** Herramienta para la redacción final de la documentación.
- **[WEB] Microsoft Edge, Google Chrome:** Navegadores para buscar información y para probar el funcionamiento de la aplicación web.

5.1.3. Resumen de las tareas

A continuación se puede ver un resumen de las tareas en la tabla 5.1.

Tabla 5.1: Tabla resumen de las tareas. Fuente: Elaboración propia

ID	TAREA	DURACIÓN (h)	DEPENDENCIAS	ROLES	RECURSOS	
					HUMANOS	MATERIALES
GP1	Contextualización y alcance	25	-	JP	S	WEB, AT, TEX, LO, JR
GP2	Planificación temporal	9	GP1	JP	S	WEB, AT, TEX, LO, JR, PL
GP3	Presupuesto y sostenibilidad	10	GP2	JP	S	WEB, AT, TEX, LO, JR
GP4	Integración del documento final	19	GP3	JP	S	WEB, AT, TEX, LO, JR
R	Reuniones de control	18	-	JP, PO	S, SJ, A	WEB, MEET
VA1	Obtención de los datos	18	-	D	S	WEB, NB, GA, GIT
VA2	*Business understanding* y *Data understanding*	9	VA1	BA, DS	S	WEB, TEX, LO
VA3	Selección de los datos	15	VA2	DS	S	WEB, PC, GIT, ANW, GA
VA4	Limpieza de los datos	15	VA3	DS	S	WEB, PC, GIT
VA5	Construcción de datos	20	VA4	DS	S	WEB, PC, GIT
VA6	Integración de los datos	10	VA5	DS	S	WEB, PC, GIT
VA7	Aprender a utilizar Logstash y Kibana	6	-	DS	S	WEB, LS, K
VA8	Análisis inicial de los datos	20	VA7	DS	S	WEB, LS, K, ANW
MD1	Familiarización con las redes bayesianas y los modelos en grafo	30	-	DS	S	WEB
MD2	Aprender R	18	-	DS	S	WEB, RS

Continua en la página siguiente

Tabla 5.1 – *Continuación de la página anterior*

ID	TAREA	DURACIÓN (h)	DEPENDENCIAS	ROLES	RECURSOS	
					HUMANOS	MATERIALES
MD3	Construcción de un modelo para la predicción del rendimiento	60	VA6, MD1,MD2	DS	S	WEB, RS, H, GIT, J
MD4	Construcción de un modelo para la detección de comportamientos anómalos	60	VA6, MD1, MD2	DS	S	WEB, RS, H, GIT, J
IM1	Generación de un *product backlog* en Jira	6	MD3, MD4	JP	S	WEB, GIT, J
IM2	Definición de completo	6	-	JP	S	WEB, TEX, LO
IM3	Implementación de mejoras	80	IM1, IM2	D, T	S	WEB, NB, GIT, J, ANW
EM1	Documentación de la fase de visualización y análisis preliminar de los datos	30	-	BA, DS	S	WEB, TEX, LO
EM2	Documentación de la fase del modelado de los datos	24	-	DS	S	WEB, TEX, LO
EM3	Documentación de la fase de implementación de mejoras	18	-	D, T	S	WEB, TEX, LO
EM4	Preparación de la defensa del proyecto	24	-	JP	S	WEB, TEX, LO

5.2. Gantt

En la figura 5.1 se puede ver la planificación del proyecto. En ella se puede ver información como los cuatro hitos del proyecto indicados con un rombo, los roles que intervienen en cada tarea y las tareas que se consideran críticas en color rojo.

	Nombre	Inicio	Terminado
1	**Gestión del proyecto**	**23/02/21 10:00**	**19/03/21 18:00**
2	GP1	23/02/21 10:00	2/03/21 18:00
3	GP2	2/03/21 11:00	8/03/21 11:00
4	GP3	8/03/21 10:00	12/03/21 18:00
5	GP4	15/03/21 10:00	19/03/21 18:00
6	**Visualización y análisis preliminar de los datos**	**8/03/21 10:00**	**14/04/21 18:00**
7	VA1	8/03/21 10:00	19/03/21 18:00
8	VA2	22/03/21 10:00	26/03/21 18:00
9	VA3	29/03/21 10:00	31/03/21 13:00
10	VA4	31/03/21 13:00	2/04/21 16:00
11	VA5	2/04/21 16:00	7/04/21 17:48
12	VA6	7/04/21 17:48	9/04/21 16:00
13	VA7	12/04/21 9:00	12/04/21 18:00
14	VA8	13/04/21 10:00	14/04/21 18:00
15	Dashboard en Kibana	14/04/21 18:00	14/04/21 17:00
16	**Modelado de los datos**	**12/04/21 10:00**	**21/05/21 18:00**
17	MD1	12/04/21 10:00	16/04/21 18:00
18	MD2	19/04/21 10:00	23/04/21 18:00
19	MD3	26/04/21 10:00	7/05/21 18:00
20	Modelo para la predicción del rendimiento	7/05/21 18:00	7/05/21 17:00
21	MD4	10/05/21 10:00	21/05/21 18:00
22	Modelo de detección de anomalías	21/05/21 18:00	21/05/21 17:00
23	**Implementación de mejoras**	**21/05/21 10:00**	**14/06/21 18:00**
24	IM1	21/05/21 10:00	21/05/21 18:00
25	IM2	24/05/21 10:00	25/05/21 18:00
26	IM3	26/05/21 10:00	14/06/21 18:00
27	Mejora de la aplicación	14/06/21 18:00	14/06/21 17:00
28	**Elaboración de la memória**	**22/03/21 10:00**	**25/06/21 18:00**
29	EM1	22/03/21 10:00	23/04/21 16:00
30	EM2	26/04/21 10:00	28/05/21 18:00
31	EM3	31/05/21 10:00	18/06/21 18:00
32	EM4	21/06/21 10:00	25/06/21 18:00
33	**Reuniones de control**	**5/03/21 10:00**	**25/06/21 18:00**
34	R	5/03/21 10:00	5/03/21 18:00
35	R	12/03/21 10:00	12/03/21 18:00
36	R	19/03/21 10:00	19/03/21 18:00
37	R	26/03/21 10:00	26/03/21 18:00
38	R	9/04/21 9:00	9/04/21 18:00
39	R	16/04/21 9:00	16/04/21 18:00
40	R	23/04/21 9:00	23/04/21 18:00
41	R	30/04/21 9:00	30/04/21 18:00
42	R	7/05/21 9:00	7/05/21 18:00
43	R	14/05/21 9:00	14/05/21 18:00
44	R	21/05/21 9:00	21/05/21 18:00
45	R	28/05/21 9:00	28/05/21 18:00
46	R	4/06/21 9:00	4/06/21 18:00
47	R	11/06/21 9:00	11/06/21 18:00
48	R	18/06/21 9:00	18/06/21 18:00
49	R	25/06/21 9:00	25/06/21 18:00

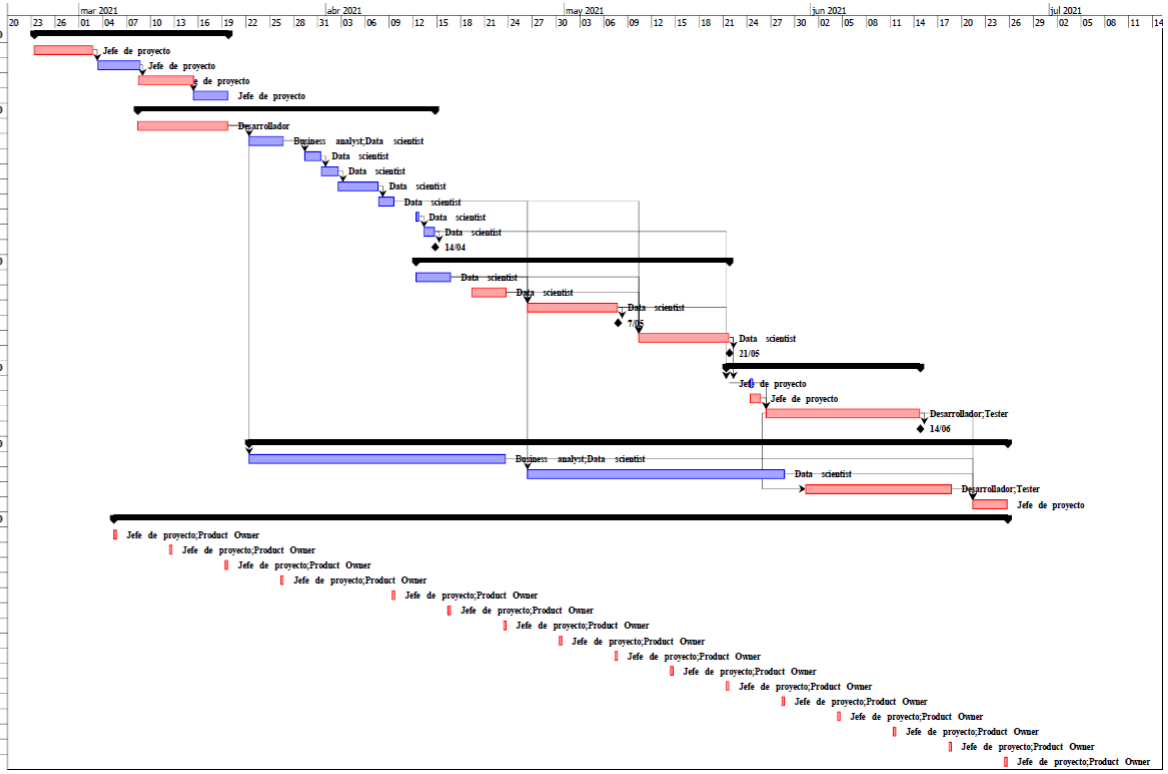

Figura 5.1: Diagrama de Gantt. Fuente: Elaboración propia

5.3. Gestión de riesgos

Durante el transcurso del proyecto pueden surgir diversos imprevistos que afecten negativamente el desarrollo de éste. En esta sección se discuten las soluciones a los potenciales riesgos presentados en la sección 3.3.

Fecha de entrega

Siendo que el proyecto tiene una fecha de entrega fija antes de empezarlo puede llevar a problemas de planificación debido a una mala estimación de la duración de las tareas. Esto puede ocasionar desplazamiento de tareas con respecto al plan original.

- **Impacto:** Alto
- **Probabilidad:** Media
- **Solución:** Revaluar la duración de las tareas y reorganizar el plan de proyecto respecto a éstas. Para esto habría que crear una nueva tarea de 8h aproximadamente para reorganizar el proyecto. En caso de no disponer de suficiente tiempo pese a la re-planificación, habría que considerar recortar el alcance del proyecto, reduciendo la cantidad de modelos a realizar a uno.

Inexperiencia en el campo

Dado que el campo de la ciencia de datos es nuevo para el autor, la búsqueda de información de terminología y herramientas para la realización del análisis puede consumir más tiempo del previsto. Adicionalmente, la realización de tareas relacionadas con este campo puede ser más lenta de lo esperado.

- **Impacto:** Medio
- **Probabilidad:** Media
- **Solución:** Se ha sobre-estimado la duración de las tareas relacionadas con el análisis y tratamiento de los datos en respuesta a la inexperiencia del autor. En caso de requerir más horas de dedicación, habría que re-planificar el plan de proyecto y transferir horas de otras tareas como la de implementación de mejoras.

Disponibilidad del servicio de *hosting*

Dado que la aplicación web se despliega en un servicio web sobre el que el autor no tiene control total, es posible que su disponibilidad se vea afectada por tareas de mantenimiento o problemas con el servicio, lo que puede llevar a un período de tiempo en el que la aplicación no sea accesible. Esto implicaría no tener acceso a los datos generados por el servidor.

- **Impacto:** Medio

- **Probabilidad:** Baja
- **Solución:** Siendo que es un servicio contratado y se depende totalmente de la empresa que administra dicho servicio, no hay otra opción que esperar a que la compañía lo reanude. Para mitigar el impacto sobre el proyecto semanalmente se realiza una copia de los registros en un ordenador local, de manera que sea posible seguir trabajando en el análisis de datos pese a no tener acceso al servidor.

Base de usuarios insuficiente

Dado que la aplicación ha estado varios años inactiva ha perdido la gran mayoría de sus usuarios originales, por lo que hay que volver a darle visibilidad y atraer nuevos. Este puede ser un proceso lento y, teniendo en cuenta que el proyecto tiene una duración limitada, puede ocurrir que no se incremente la cantidad de usuarios lo suficientemente rápido, lo que afectaría a la cantidad de datos generados.

- **Impacto:** Alto
- **Probabilidad:** Media
- **Solución:** Para mitigar la falta de datos con los que analizar se pueden aplicar diversas técnicas de manipulación, como *Synthetic Minority Oversampling Technique (SMOTE)*, para generar nuevos datos a partir de los existentes. Ésto provocaría un incremento de aproximadamente 12h en la fase de visualización y análisis preliminar de los datos.

Corrupción de la base de datos

Es posible que tras añadir cambios dentro de la aplicación o debido a algún error no conocido por los desarrolladores se corrompa la base de datos, dejando inservible la aplicación y perdiendo los datos almacenados.

- **Impacto:** Alto
- **Probabilidad:** Baja
- **Solución:** Para prevenir una pérdida total de los datos se ha implementado un sistema de copias de seguridad por el cual se realiza una copia de seguridad incremental semanalmente y una copia de seguridad total mensualmente, de esta forma si se corrompe la base de datos se podrá restaurar su estado al de la semana anterior al fallo.

6. Gestión económica

En este capítulo se realiza un presupuesto del trabajo para determinar el coste que tendría en caso de realizarse realmente.

6.1. Identificación y estimación de los costes

En esta sección se presenta un estudio de los costes que conlleva el desarrollo del proyecto. Por lo tanto, se identifican y se realiza una estimación de los costes de personal, junto con los gastos directos e indirectos.

6.1.1. Recursos humanos

Antes de definir los costes de personal por tarea hay que definir el salario de cada uno de los roles que se desempeñan en el proyecto, definidos en la sección 5.1.2. Para definir el coste medio de cada rol se ha utilizado el portal GlassDoor[1], el cual proporciona el salario medio anual de los diferentes roles. A partir de éste salario anual se ha computado el salario por hora estableciendo que se trabajan de media 1944 horas al año [35] y se le ha añadido el coste de la Seguridad Social [36].

Rol	**Sueldo/año (bruto)**	**Sueldo/hora (bruto)**	**Salario/hora (bruto) + Seguridad Social (x1.3)**
Jefe de proyecto	38.909,00 €	20,01 €	26,02 €
Product owner	45.000,00 €	23,15 €	30,09 €
Junior business analyst	24.682,00 €	12,70 €	16,51 €
Junior data scientist	30.312,00 €	15,59 €	20,27 €
Desarrollador junior	23.432,00 €	12,05 €	15,67 €
Tester	30.082,00 €	15,47 €	20,12 €

Tabla 6.1: Coste por hora de cada rol. Fuente: Elaboración propia

Una vez computados los costes de cada rol (ver tabla 6.1), se pasa a determinar los costes asociados a cada una de las tareas definidas en la sección 5.1.1 (ver tabla 6.2). Con esto obtenemos una estimación del coste de personal por actividad (CPA).

[1] https://www.glassdoor.es

ID	Duración (h)	Horas/Rol (h)						Coste (€)
		JP	PO	BA	DS	D	T	
GP1	25	25	0	0	0	0	0	650,48
GP2	9	9	0	0	0	0	0	234,17
GP3	10	10	0	0	0	0	0	260,19
GP4	19	19	0	0	0	0	0	494,37
R	18	18	18	0	0	0	0	1.010,02
VA1	18	0	0	0	0	18	0	282,05
VA2	9	0	0	4	5	0	0	167,37
VA3	15	0	0	0	15	0	0	304,06
VA4	15	0	0	0	15	0	0	304,06
VA5	20	0	0	0	20	0	0	405,41
VA6	10	0	0	0	10	0	0	202,70
VA7	6	0	0	0	6	0	0	121,62
VA8	20	0	0	0	20	0	0	405,41
MD1	30	0	0	0	30	0	0	608,11
MD2	18	0	0	0	18	0	0	364,87
MD3	60	0	0	0	60	0	0	1.216,22
MD4	60	0	0	0	60	0	0	1.216,22
IM1	6	6	0	0	0	0	0	156,12
IM2	6	6	0	0	0	0	0	156,12
IM3	80	0	0	0	0	60	20	1.342,50
EM1	30	0	0	10	20	0	0	570,46
EM2	24	0	0	0	24	0	0	486,49
EM3	18	0	0	0	0	18	0	282,05
EM4	24	24	0	0	0	0	0	624,47
TOTAL	550	117	18	14	303	96	20	11.865,54

Tabla 6.2: Coste estimado para cada tarea. Fuente: Elaboración propia

6.1.2. Costes generales

Para describir de manera detallada el coste del proyecto también hay que tener en cuenta los recursos materiales definidos en la sección 5.1.2. En esta sección se detallan los costes derivados del *harware* y los costes indirectos.

Dado que todos los recursos software que se utilizan para la realización del proyecto son gratuitos no aparecerán en la descripción de los costes.

Hardware

Para poder llevar a cabo el proyecto se necesitan un conjunto de materiales *hardware*. Teniendo en cuenta que en un año hay unas 1944 horas laborales (243 días laborales) y estimando una vida útil de 4 años por recurso, podemos calcular

la amortización como:

$$Amortización = \frac{Coste\ hardware}{Vida\ útil * Días\ de\ trabajo * Horas\ al\ día} * Horas\ de\ uso \quad (6.1)$$

Hardware	**Coste (€)**	**Vida útil (años)**	**Amortización (€)**
Ordenador de sobremesa por piezas	1.368,74	4	129,08
Monitor ACER X193HQ	25,00	4	2,36
Monitor ACER B223WL	69,00	4	6,51
Ratón Mars Gaming	15,00	4	1,41
Teclado Glorious GMMK TKL	102,66	4	9,68
Servidor en ANW[2]	188,75	1	65,16
TOTAL	1.769,15	-	214,20

Tabla 6.3: Amortización del hardware usado. Fuente: Elaboración propia

Software

Además de los materiales hardware, también tenemos que tener en cuenta el *software* a utilizar en el proyecto. En este caso, todo el *software* utilizado es gratuito a excepción de la licencia de Hugin. Por lo tanto, calculamos la amortización de igual manera que con el *hardware* (ver Equación 6.1).

Software	**Coste (€)**	**Vida útil (años)**	**Amortización (€)**
Hugin	79,00	0,17	58,09
TOTAL	79,00	-	58,09

Tabla 6.4: Amortización del software usado. Fuente: Elaboración propia

Costes indirectos

De la misma manera que se han tenido en cuenta los costes materiales, también se contabilizan los costes indirectos derivados del uso de los equipos informáticos. Puesto que el proyecto se lleva a cabo desde el domicilio del autor y no hay necesidad de desplazamientos, los únicos costes que se computan son: el coste del internet (6.2) y de electricidad (6.3). Para el consumo eléctrico se asume que un ordenador de sobremesa consume 250 W y que un monitor estándar consume 80 W [37], en este caso se tiene en cuenta el uso de dos monitores.

$$Amortización\ internet = Coste\ internet * (5\ meses) * (\frac{22\ días}{30}) * (\frac{6\ horas}{24}) \quad (6.2)$$

$$Amortización\ electricidad = Coste\ kWh * Cosumo\ medio * Horas\ de\ uso \quad (6.3)$$

[2]Para el cálculo de la amortización se ha tenido en cuenta que el servidor está funcionando las 24 horas del día durante todo el año.

Recurso	Coste (€)	Amortización (€)
Electricidad	0,17 [38]	37,74
Internet	30,00 [39]	22,92
TOTAL	30,17	60,66

Tabla 6.5: Amortización de los costes indirectos. Fuente: Elaboración propia

Coste general del proyecto

En la tabla 6.6 se puede ver un resumen del coste general (CG) del proyecto, donde se juntan la amortización de los recursos *hardware* y los costes indirectos.

Descripción	Coste (€)
Coste *hardware*	214,20
Coste *software*	58,09
Costes indirectos	60,66
TOTAL	332,95

Tabla 6.6: Resumen del coste general del proyecto. Fuente: Elaboración propia

6.1.3. Desviaciones del presupuesto

Con tal de disponer de un margen de maniobra para afrontar posibles imprevistos y riesgos, se ha de crear un fondo de contingencia en caso de imprevistos y se ha de hacer un estudio de los riesgos del proyecto y su coste asociado.

ara el cálculo del fondo de contingencia se ha aplicado un 10 % de margen al PCA y un margen del 5 % al CG, ya que el primero es mas susceptible a sufrir imprevistos debido a una mala planificación del proyecto.

l coste asignado a los posibles riesgos del proyecto y el uso de un plan alternativo se puede ver en la tabla 6.7.

Riesgo	Probabilidad	Duración (h)	Coste estimado (€)	Coste (€)
Fecha de entrega	25,00 %	10	260,19	65,05
Inexperiencia en el campo	20,00 %	30	608,11	121,62
Disponibilidad del servicio de *hosting*	5,00 %	6	94,02	4,70
Base de usuarios insuficiente	50,00 %	12	243,24	121,62
Corrupción de la base de datos	5,00 %	6	94,02	4,70
TOTAL	-	-	1.299,58	317,69

Tabla 6.7: Coste de los riesgos. Fuente: Elaboración propia

6.1.4. Coste final del proyecto

En la tabla 6.8 se puede ver un resumen del coste final del proyecto juntando los costes definidos en las secciones anteriores y añadiendo el 21 % de IVA.

Tipo de coste	Coste (€)
CPA	11.865,54
CG	332,95
Contingencia	1.203,20
Imprevistos	317,69
Subtotal	13.719,39
Impuestos	2.881,07
TOTAL	16.600,46

Tabla 6.8: Coste total del proyecto. Fuente: Elaboración propia

6.2. Control de gestión

A continuación se detalla cómo se controlan las desviaciones en el presupuesto del proyecto debido a imprevistos. El primer paso es contabilizar las horas reales utilizadas para llevar a cabo cada tarea, de este modo podemos calcular las posibles desviaciones sobre el presupuesto y ver si nos encontramos dentro del cálculo inicial.

Para determinar la desviación de cada uno de los elementos computables en el presupuesto se utilizarán las siguientes fórmulas.

- Desvío de mano de obra por tarea en precio

$$(coste\ estimado - coste\ real) * consumo\ horas\ real$$

- Desvío de mano de obra por tarea en consumo

$$(consumo\ horas\ estimado - consumo\ horas\ real) * coste\ estimado$$

- Desvío de recursos materiales en precio

$$(coste\ estimado - coste\ real) * consumo\ horas\ real$$

- Desvío de recursos materiales en consumo

$$(consumo\ horas\ estimado - consumo\ horas\ real) * coste\ estimado$$

- Desvío total en mano de obra

$$total\ coste\ estimado\ mano\ de\ obra - total\ coste\ real\ mano\ de\ obra$$

- Desvío total en recursos materiales

$$total\ coste\ estimado\ recursos\ materiales - total\ coste\ real\ recursos\ materiales$$

- Desvío total de los imprevistos

$$total\ coste\ estimado\ imprevistos - total\ coste\ real\ imprevistos$$

- Desvío total del proyecto

  $$\textit{total coste presupuestado del proyecto} - \textit{total coste real del proyecto}$$

7. Obtención de los datos de uso

Dado que este trabajo se centra en el análisis de datos obtenidos de la aplicación, una de las mejores fuentes para obtener datos es la propia aplicación. Para poder extraer dicha información y sacarle el máximo provecho hemos utilizado dos métodos, uno explícito y uno implícito.

7.1. Página de sugerencias

El método de obtención de datos de manera explícita se ha llevado a cabo mediante la creación de una página de sugerencias dentro de la aplicación. En ella los usuarios disponen de un desplegable, donde pueden elegir si quieren notificar un error o si quieren sugerir alguna mejora, y de un área de texto donde pueden escribir lo que consideren oportuno.

Para gestionar las sugerencias, los administradores disponen de una página donde aparecen listadas. En esta página se puede ver la siguiente información de las sugerencias: el ID, la fuente, el tipo, la fecha de creación, el mensaje y el estado. Además, desde esta misma página pueden crear, borrar o modificar sugerencias.

Se ha decidido añadir el campo fuente para especificar el origen de la sugerencia, ya que podemos encontrarnos con que haya usuarios que notifiquen fallos o valoren la aplicación a través de redes sociales o correos electrónicos.

7.2. Registros de uso

El método de obtención de datos de manera indirecta se basa en la recopilación de datos utilizando diversos puntos de registro a lo largo de la aplicación con el objetivo de generar un registro de las actividades que realizan los usuarios dentro de la aplicación. A continuación se detallan los pasos para crear dicho registro.

7.2.1. Primeros pasos

Antes de empezar a implementar el *logger* hay que decidir que datos concretos queremos obtener y que información queremos extraer de estos, ya que si no se realiza un planteamiento adecuado antes de crear los registros podemos encontrarnos con que los datos que estamos obteniendo no nos aportan tipo de información. Por este motivo, uno de los pasos más importantes a realizar antes de escribir código

es definir una serie de objetivos o preguntas que queremos responder a partir de los datos obtenidos de la aplicación.

Preguntas a responder

En este proyecto nos centramos en como los usuarios interaccionan con aplicación, así como el funcionamiento interno de ésta. Por lo tanto, dividiremos las preguntas a responder en dos categorías.

1. Preguntas a responder sobre los usuarios y las funcionalidades
    a) ¿Cuánto tardan en cargar las páginas?
    b) ¿Cuánto rato están los usuarios en una página determinada?
    c) ¿Cuáles son las páginas más utilizadas por los usuarios?
    d) ¿Cuál es el flujo de utilización de la aplicación?
    e) ¿Qué días hay más afluencia de usuarios?
    f) ¿Qué acciones son más comunes entre los usuarios?
    g) ¿Cuánto tiempo pasan los usuarios dentro de una sesión?
    h) ¿Cuántas veces entran los usuarios a la aplicación?
    i) ¿Cómo evoluciona la cantidad de usuarios?
    j) ¿Qué idiomas son los más utilizados por los usuarios?
    k) ¿Qué tipos de dispositivos son los más utilizados para acceder a la aplicación?
2. Preguntas a responder sobre el funcionamiento del juego
    a) ¿Qué páginas suelen dar un tipo de error en concreto (e.g. 404, 500)?
    b) ¿Qué errores son más comunes?
    c) ¿Qué páginas fallan más frecuentemente?

Definición de las variables

Una ver hemos definidas correctamente las preguntas a resolver pasamos a elaborar la lista de variables que recopilaremos dentro de la aplicación, y que nos ayudarán a responder nuestras preguntas. Dado que las preguntas pertenecientes al segundo grupo las podemos responder gracias a los registros que nos proporciona el propio servidor, nos centramos en las del primer grupo, que hacen referencia a como se utiliza la aplicación.

Con tal de no sobrecargar en exceso los registros y para no recopilar variables innecesarias, hemos optado por un número reducido de variables detalladas a continuación. Las cuales nos proporcionan la toda la información necesaria, ya sea directamente o al aplicarles algún tipo de manipulación.

Variables

- **Timestamp:** Es la marca de tiempo de la entrada, ya que la mayoría de las preguntas tienen un factor temporal, como pueden ser el tiempo de carga o el tiempo de uso.
- **Idioma:** Es el idioma con el que se está visualizando la página actual. Con esta variable podemos ver que idiomas son los más utilizados y hacernos una idea de en qué regiones es más popular la aplicación.
- **ID usuario:** Es un identificador de usuario, en este caso el nombre de usuario, lo que nos permite analizar los datos a nivel de usuario. Extrayendo información como los usuarios diarios o la evolución del número de usuarios de la aplicación.
- **ID de sesión:** Es un identificador único para cada sesión, el cual se regenera al abrir una nueva ventana del navegador. Con este ID podemos tener una granularidad aún más fina que con el ID de usuario, lo que nos permite analizar patrones de comportamiento en distintas sesiones.
- **Página actual:** Es el nombre de la página en la que se ha registrado la acción. Esta variable combinada con algunas de las demás nos proporciona información muy valiosa, como puede ser el tiempo de carga de las páginas, la popularidad que tiene cada una e incluso nos permite trazar el flujo de páginas que siguen los usuarios al utilizar la aplicación.
- **Acción:** Es la acción que se está llevando acabo en el momento del registro. Entendemos como acción cualquier evento que ocurra dentro de una página. Los más comunes serán el inicio y la finalización de la carga de una página, pero también pueden haber eventos propios de cada una de las páginas, como el pujar por un jugador o contratar un entrenador.

7.2.2. Implementación de los registros

Una vez hemos terminado de definir los objetivos y conocemos los datos que pensamos registrar pasamos a la implementación de los registros. Para ello existen multitud de *frameworks* que facilitan la creación de registros personalizados, como por ejemplo Logback[1] o Log4j[2].

Para este proyecto hemos utilizado Logback como herramienta para generar los registros y SLF4J[3] que es una librería que proporciona una abstracción común para varios *frameworks* de creación de registros en Java, lo que permite cambiar fácilmente entre ellos sin tener que modificar el código. El principal motivo por el cual se ha optado por estas librerías es su fácil puesta en marcha, ya que su configuración es bastante simple así como su utilización.

[1] http://logback.qos.ch/
[2] https://logging.apache.org/log4j/2.x/
[3] http://www.slf4j.org

Configuración

Antes de empezar a registrar los datos hay que realizar ciertas tareas de configuración para que todo funcione correctamente.

En el caso de SLF4J solamente se necesita incluir el JAR correspondiente a la última versión de la API, que en el momento de escribir este trabajo es `slf4j-api-1.7.30.jar`, al *classpath*. Como SLF4J está implementado para trabajar por defecto con Logback no necesitamos nada más que los JARs necesarios para utilizar este último y SLF4J se conectará automáticamente con Logback.

Para empezar a utilizar Logback hay que añadir al *classpath* los JARs correspondientes a las últimas versiones estables de `logback-core.jar` y `lobgack-classic.jar`. Una vez tenemos Logback en el proyecto podemos empezar a utilizarlo con la configuración por defecto, que envía los registros a la consola. En nuestro caso queremos que los registros se vayan añadiendo a un fichero de forma que podamos analizar los datos en los siguientes pasos. Para esto necesitamos crear un fichero llamado logback.xml y añadirlo al classpath (ver Apéndice A.1).

En este caso la configuración es muy simple, únicamente especificamos la ubicación y el nombre del fichero donde queremos que se guarden los registros, el tipo de *appender* que queremos utilizar, en este caso un FileAppender, y el patrón que tendrán las entradas de los registros, que lo hemos mantenido como una línea en formato CSV para facilitar su manipulación más adelante.

Implementación

Una vez ya tenemos toda la configuración realizada pasamos a implementar el código necesario para registrar las acciones que se llevan a cabo dentro de la aplicación.

El primer paso es crear una clase que utilizaremos como *logger* personalizado, de esta manera si necesitamos añadir nuevas variables que registrar o necesitamos aplicar alguna transformación al formato de los registros únicamente tendremos que cambiar la implementación de esta clase, en vez de tener que ir página por página cambiando el código. Esta clase no contiene nada más que una serie de métodos para registrar distintas acciones que se pueden realizar dentro de una página, como pueden ser el inicio y la finalización de la carga de esta (ver Apéndice A.2).

Ahora que ya tenemos nuestra clase para realizar los registros solo queda capturar las acciones que consideremos relevantes. Puesto que en este proyecto nos interesa la interacción de los usuarios con las distintas páginas de la aplicación hemos colocado diversos puntos de registro en todas las páginas registrando eventos como la realización de una puja o el despido de un jugador.

8. Visualización y análisis preliminar

En este capítulo se describe el proceso que se ha seguido para la obtención, manipulación y posterior visualización de los datos.

8.1. Origen de los datos implícitos

Para poder realizar un análisis completo de la aplicación y así poder obtener una mejor visión de todo el ecosistema que esta conforma, hay que recopilar datos de diversas fuentes que nos permitan entender que está sucediendo desde distintos puntos de vista. Por este motivo en este proyecto se utilizan datos obtenidos de cuatro fuentes distintas.

8.1.1. ChessLeague

La primera fuente de información se obtiene a partir de los registros generados dentro de la propia aplicación descritos en la sección 7.2.1

Puesto que en este caso somos nosotros los que generamos los registros tenemos mucha más libertad a la hora de decidir que datos queremos obtener y cómo. En nuestro caso hemos considerado que la información que nos permitirá responder las preguntas propuestas en la sección 7.2.1 es la siguiente: la marca de tiempo, el nombre de usuario, el ID de la sesión, el nombre de la página y la acción que se lleva a cabo. El resto de datos necesarios para el análisis se pueden obtener aplicando varias transformaciones a la información previamente mencionada. En la Tabla 8.1 se pueden ver con más detalle.

Estos registros se han mantenido simples y sin un exceso de información con el objetivo de que sea más fácil su utilización y manipulación en etapas posteriores del proyecto.

Variable	Descripción	Tipo	# Niveles	Valores
Timestamp	Instante de tiempo en el que se registra la acción	Timestamp	-	-
Idioma	Idioma en el que se visualiza la página	Categórica nominal	4	es, en, fr, de
Nombre de usuario	Usuario que realiza la acción	String	-	-
ID de sesión	ID de la sesión en la que se realiza la acción	String	-	-
Página actual	Nombre de la página donde se realiza la acción	Categórica nominal	47	Bid, Cron, Help, Play, Team, Start, index, Invite, League...
Acción	La acción que se lleva acabo	String	-	-

Tabla 8.1: Detalles de las variables de interés de los registros de uso.
Fuente: Elaboración propia

8.1.2. Glassfish

El propio servidor Glassfish nos proporciona de manera automática los registros que genera durante ejecución y también se pueden activar los registros de acceso desde su consola de administración. Esto nos permite obtener información valiosa sobre lo que está sucediendo con la aplicación a nivel de funcionamiento y ver que tipo de incidencias están ocurriendo. Como ya se ha mencionado Glassfish nos proporciona dos tipos de registros, los cuales se detallan a continuación.

Registros del servidor

Estos registros nos proporcionan diversos datos de la actividad dentro del servidor, como pueden ser errores o mensajes provenientes de la aplicación. En este caso, se ha hecho una selección de los datos a utilizar, ya que no toda la información que proporciona cada entrada nos resulta relevante para el estudio. En la Tabla 8.2 se pueden ver los datos relevantes en detalle.

Variable	Descripción	Tipo	# Niveles	Valores
Timestamp	Instante de tiempo en el que se registró la entrada	Timestamp	-	-
Severidad	Severidad de la incidencia	Categórica ordinal	3	SEVERE, WARNING, INFO
Mensaje	Detalle de la incidencia	String	-	-
Nombre del hilo	Nombre del hilo donde ocurre la incidencia	String	-	-
Error	Tipo de error que ha ocurrido	String	-	-
Clase	Clase donde se ha originado el error	String	-	-
Método	Método que ha provocado el error	String	-	-

Tabla 8.2: Detalles de las variables de interés de los registros del servidor.
Fuente: Elaboración propia

Registros de acceso

Estos registros nos proporcionan información sobre las peticiones HTTP que recibe el servidor y los recursos que se acceden. En este caso los datos que nos aportan información son la petición HTTP, el recurso accedido, el código de respuesta y el tamaño en bytes de la respuesta. En la Tabla 8.3 se pueden ver los datos relevantes en detalle.

Variable	Descripción	Tipo	# Niveles	Valores
Timestamp	Instante de tiempo en el que se registró el acceso	Timestamp	-	-
Petición	Tipo de petición HTTP	Categórica nominal	2	GET, POST
Recurso	Recurso implicado en la petición	String	-	-
Código respuesta	Código de respuesta enviado por el servidor	Categórica nominal	4	200, 302, 404, 500
Tamaño respuesta	Tamaño de la respuesta en bytes	Integer	-	-

Tabla 8.3: Detalles de las variables de interés de los registros de acceso al servidor. Fuente: Elaboración propia

8.1.3. Google Analytics

La última fuente de datos que utilizamos en este proyecto proviene de Google Analytics (GA). Esta utiliza Google Tag Manager (GTM) para obtener una serie de datos propios de la aplicación, junto con el resto de datos que ya recopila GA por defecto. En este caso, los datos que recopilamos a parte de los que vienen por defecto son: nombre de usuario, id de sesión y el instante en que se accede a una página. Además, debido a que GA recopila una gran cantidad de información que no nos es relevante para el estudio hemos hecho una selección de los datos a utilizar. En la Tabla 8.4 se pueden ver los datos relevantes en detalle.

Variable	Descripción	Tipo	# Niveles	Valores
ID Usuario	Usuario que realiza la acción	String	-	-
ID Sesión	ID de la sesión en la que se realiza la acción	String	-	-
Timestamp	Instante de tiempo en el que se registra la acción	Timestamp	-	-
Tipo de usuario	Tipo de usuario	Categórica nominal	2	New Visitor, Returning Visitor
País	País de la conexión	Categórica nominal	12	Spain, Portugal, Germany...
Fuente	Origen del acceso a la aplicación	Categórica nominal	7	(direct), google, duckduckgo.com, lm.facebook.com...
Navegador	Navegador utilizado	Categórica nominal	10	Chrome, Firefox, Safari, Edge, Samsung Internet...
Versión Navegador	Número de versión del navegador utilizado	String	-	-
Tipo Dispositivo	Tipo de dispositivo utilizado	Categórica nominal	3	desktop, mobile, tablet
Ruta página	Ruta de la página donde se realiza la acción	String	-	-
Ruta página anterior	Ruta de la página anterior a la actual	String	-	-
Días desde última sesión	Días desde la última conexión	Integer	-	-
Visitas página	Número de visitas a la página	Integer	-	-
Tiempo en página	Tiempo utilizado en la página	Double	-	-
Tiempo de carga	Tiempo de carga de la página en milisegundos	Integer	-	-

Tabla 8.4: Detalles de las variables de interés de Google Analytics.
Fuente: Elaboración propia

8.2. Visualización de los datos

Una vez tenemos todos los datos podemos pasar a realizar una primera visualización de los datos sin aplicarles ningún tipo de tratamiento. De esta manera nos podemos hacer una idea general de la información de la que disponemos y así planificar nuestros próximos pasos.

Para realizar esta visualización se ha utilizado Elastic Stack, que está compuesto por Logstash para la ingesta de los datos, Elasticsearch para almacenarlos y Kibana para su visualización (ver Figura 8.1).

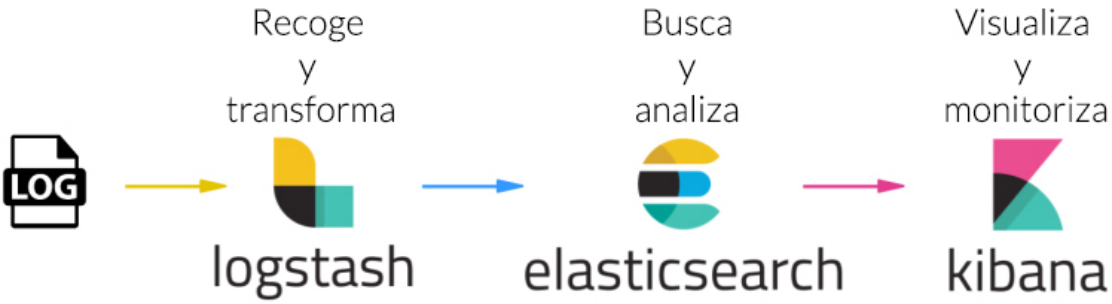

Figura 8.1: Elastic stack. Fuente: [40]

8.2.1. Ingesta y almacenamiento

Con el objetivo de automatizar el almacenamiento de los datos y de extraer la información disponible en cada uno de los registros se ha configurado Logstash de manera que observe continuamente los archivos de registro y que en el momento que detecte un cambio lo envíe a la base de datos que tiene Elasticsearch y al fichero CSV correspondiente a esa fuente de datos.

Dado que tenemos cuatro fuentes de datos distintas y cada una de ellas tiene un formato distinto se han creado cuatro *pipelines*, de manera que cada una de ellas extraiga los datos pertinentes y los almacene (ver Apéndice A.3).

8.2.2. Visualización

Una vez tenemos los datos almacenados en Elasticsearch pasamos a utilizar Kibana para manipular la información y generar así diversas visualizaciones. Esta herramienta proporciona una interfaz amigable e interactiva que facilita la elaboración de las visualizaciones, así como el administrar los datos de Elasticsearch.

En nuestro caso hemos generado cuatro *dashboards*, uno para cada una de las fuentes de datos:

1. Visualización de los datos de uso
2. Visualización de los registros del servidor
3. Visualización de los registros de acceso
4. Visualización de los datos de Google Analytics

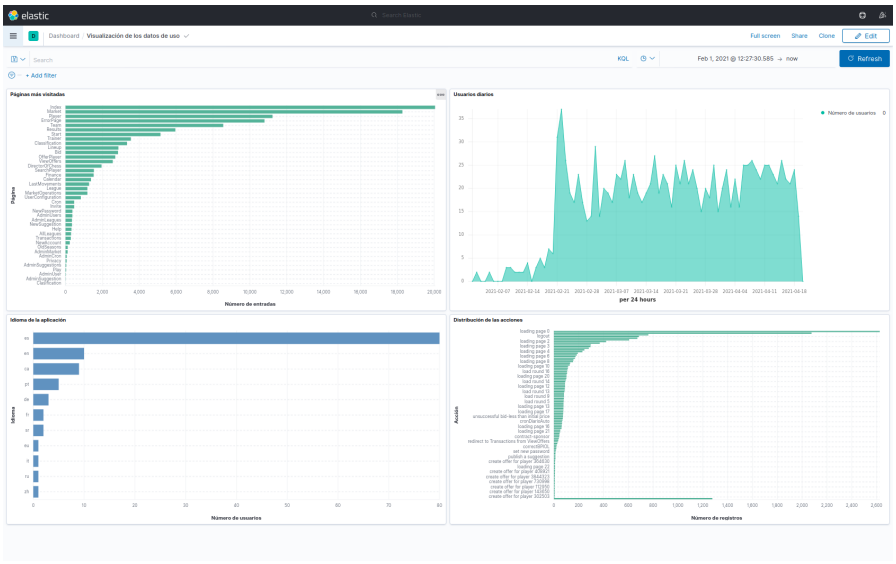

Figura 8.2: Visualización de los datos de uso. Fuente: Elaboración propia

La Figura 8.2 presenta un *dashboard* con la visualización de los datos de uso más relevantes durante el periodo de tiempo seleccionado.

La visualización de la esquina superior izquierda muestra la cantidad de visitas que tiene cada una de las páginas. A su derecha, encontramos la evolución temporal con la cantidad de usuarios diarios. En la parte inferior izquierda podemos encontrar los idiomas más utilizados dentro de la aplicación, y a su lado vemos la distribución de las acciones más comunes que se realizan dentro de la aplicación. Estas visualizaciones se pueden filtrar seleccionando elementos dentro de las mismas. Por ejemplo, acabamos de añadir una nueva página a la aplicación y queremos saber si los usuarios la están utilizando y si se está popularizando o no. En este caso, bastaría con seleccionar la página dentro de la visualización de las páginas más visitadas y automáticamente veríamos el resto de visualizaciones centradas en esta página.

Este *dashboard* (Figura 8.2) puede ayudar a los administradores a ver la popularidad de la aplicación, así como entender el valor que aportan cada una de las páginas a la aplicación.

En nuestro caso el *dashboard* nos ayuda a responder las preguntas 3, 5, 6, 9 y 10 sobre los usuarios y las funcionalidades, enumeradas en la sección 7.2.1. Lo que nos puede servir para determinar que páginas no se están utilizando e intentar mejorarlas o incluso considerar eliminarlas. También nos puede dar una idea de la popularidad de la aplicación y ver si los cambios que realizamos son bien recibidos, así como detectar qué días solemos tener mayor número de accesos y estudiar que causa esos incrementos de manera que lo podamos extender al resto de la semana.

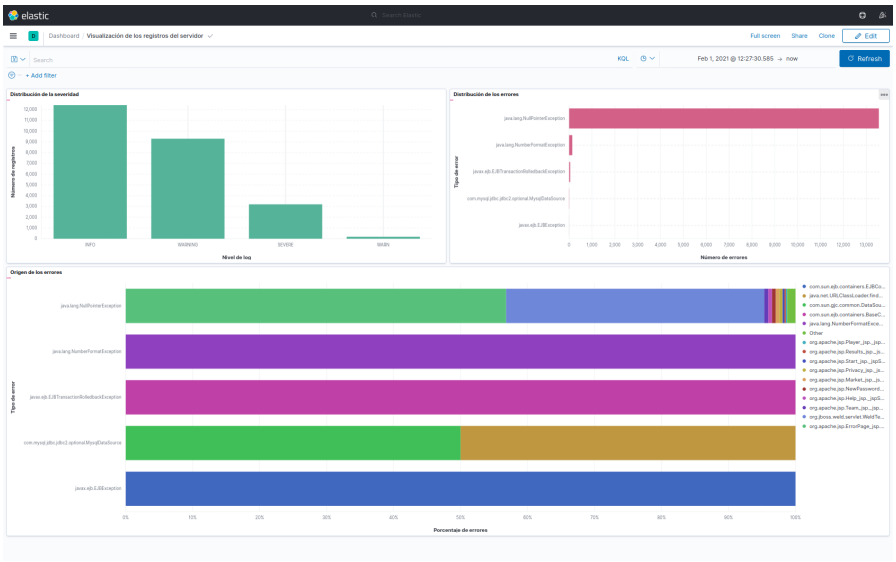

Figura 8.3: Visualización de los registros del servidor. Fuente: Elaboración propia

La Figura 8.3 muestra el *dashboard* con los errores más comunes que ocurren dentro del servidor. En la parte superior de este podemos ver la distribución de los niveles de registro y que tipo de errores ocurren dentro de la aplicación. En la zona inferior se muestran las principales causas de las excepciones.

Este *dashboard* se puede utilizar para detectar las principales clases donde se originan las excepciones, de manera que sea más sencillo buscar sus posibles causas dentro del código.

En este caso en concreto nos permite responder la pregunta 2 sobre el funcionamiento del juego, que hace referencia a los errores más comunes. Que nos indica que la gran mayoría de los errores que tenemos pertenecen a la clase *NullPointerException* seguido por *NumberFormatException*, lo que nos facilita el trabajo a la hora de detectar los errores y su origen, y por tanto reduce el tiempo que tardamos en encontrarlos y solucionarlos.

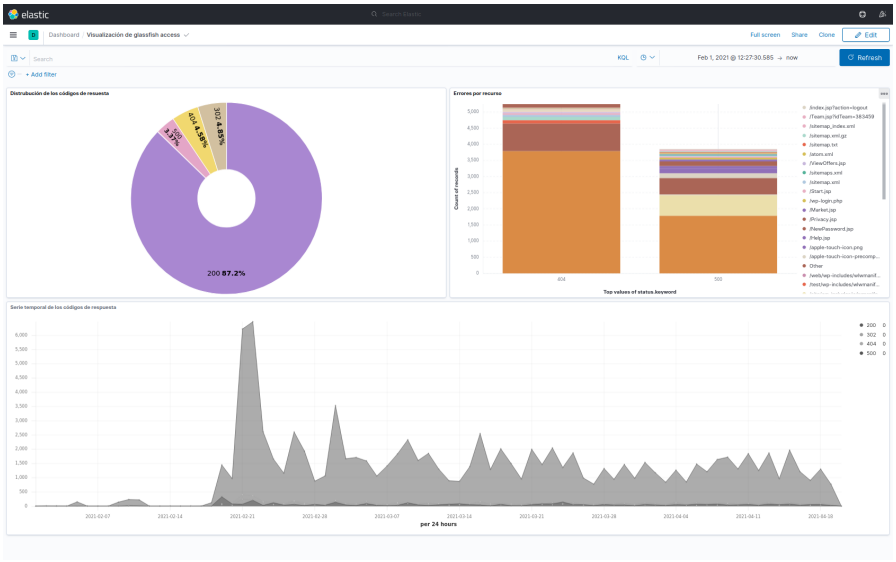

Figura 8.4: Visualización de los registros de acceso. Fuente: Elaboración propia

La Figura 8.4 presenta un *dashboard* con la visualización de los datos de acceso al servidor.

En la parte superior izquierda se muestra la distribución de los diferentes códigos de respuesta obtenidos. A su derecha, podemos ver que recursos están siendo solicitados para cada código de error. Finalmente, en la parte inferior podemos ver una evolución temporal de cada uno de los códigos de respuesta.

Este *dashboard* puede ser útil para que los desarrolladores detecten puntos de acceso a la web que puedan estar ocasionando problemas o para detectar recursos que no son accesibles, por ejemplo.

En este proyecto lo utilizamos para responder a las preguntas 1 y 3 sobre el funcionamiento del juego, enumeradas en la sección 7.2.1. Con la ayuda de este *dashboard* somos capaces de detectar recursos no accesibles para los usuarios ya sea por un error interno o por que no existen. Un ejemplo claro de esto último es uno de los favicons de la aplicación que es el que acapara más de la mitad de las respuestas con código 404, en la franja temporal observada en la Figura 8.4. Esto nos permite conocer rápidamente el origen de estos códigos de error y solucionarlos más rápidamente.

Finalmente, la Figura 8.5 muestra un resumen de los datos extraídos de Google Analytics.

En la parte superior de este podemos ver la evolución del número de usuarios diarios y la media de días entre sesiones. Justo debajo, podemos ver el tiempo medio de uso de cada una de las páginas, lo que nos permite ver cuales de ellas son

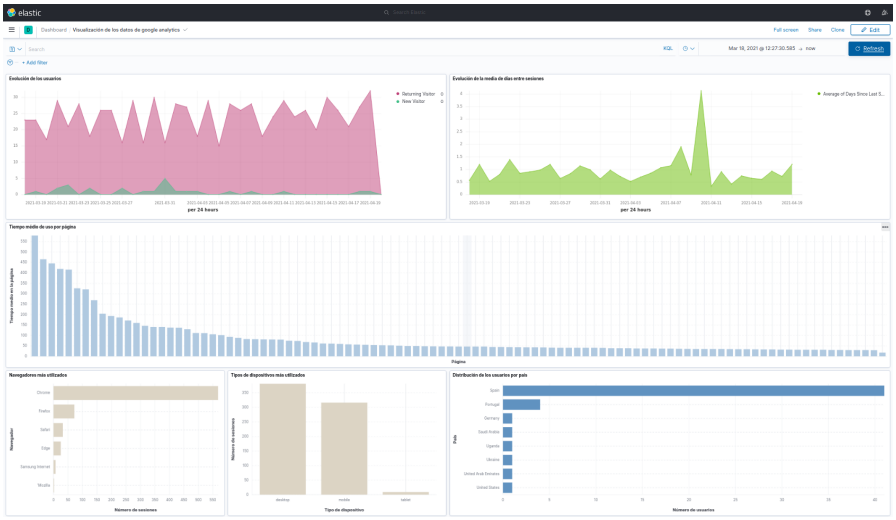

Figura 8.5: Visualización de los datos de Google Analytics. Fuente: Elaboración propia

las más utilizadas. En la parte inferior del *dashboard* podemos ver que navegadores y dispositivos se están utilizando para acceder a la aplicación, así como de que países se conectan los usuarios.

Este *dashboard* resulta especialmente útil para hacerse una idea general del estado de la aplicación en cuanto a uso y popularidad se refiere. Por ejemplo, permite ver fácilmente si esta está incrementando su base de usuarios o si por el contrario va perdiendo popularidad. También permite ver qué dispositivos son los más utilizados, permitiendo así optimizar la aplicación para dichos dispositivos y mejorar así su usabilidad.

En nuestro caso este *dashboard* se utiliza para responder a las preguntas 3, 5, 9 y 11 sobre los usuarios y las funcionalidades, descritas en la sección 7.2.1. Pese a que algunas de las preguntas ya las podemos responder con el primer *dashboard*, nos sirve para comparar los datos obtenidos directamente de la aplicación y de los ofrecidos por GA, lo que nos permite evaluar la fiabilidad de éstos últimos. La aportación más importante que nos ofrece este *dashboard* es la visualización del tipo de dispositivo utilizado para acceder a la aplicación, ya que si el número de dispositivos móviles y tables empieza a ser considerable sería un gran incentivo a la hora de decidir si ampliar la aplicación, que actualmente sólo dispone de versión web, con una versión para dispositivos Android y/o iOS.

Las principales contribuciones de la parte 1 han sido (ver Figura 8.6):

- Creación de una página de sugerencias.
- Creación de un sistema para registrar datos de uso.
- Creación de un sistema para recopilar datos de Google Analytics.
- Despliegue de Elastic Stack para coleccionar *logs* sobre el uso de la aplicación desde diversas fuentes de información.
- Creación de cuatro *dashboards* para visualizar la información coleccionada.

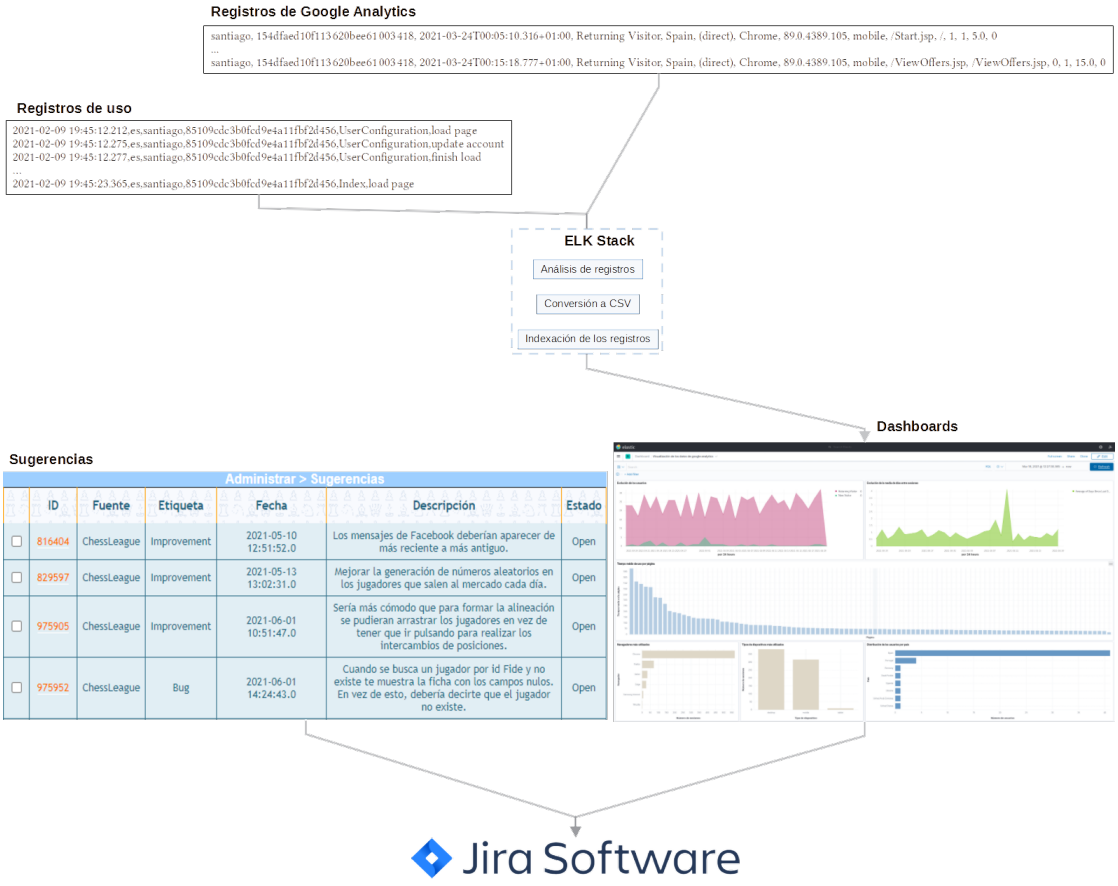

Figura 8.6: Contribuciones de la parte 1. Fuente: Elaboración propia

9. Redes bayesianas para analizar software logs

En este capítulo se describe el proceso utilizado para la creación de un conjunto de modelos que tienen como objetivo ayudarnos a mejorar la aplicación. Para cumplir esta meta hemos decidido establecer tres sub-objetivos, que se corresponden con la creación de los modelos siguientes:

Modelo de rendimiento

Con este modelo, pretendemos obtener una visión interna sobre como afectan distintas variables, como el tipo de usuario o la página, sobre el tiempo de carga de las distintas páginas.

Modelo de errores

Con este modelo, pretendemos entender qué clases son más propensas a generar errores, así como analizar si estos se originan en la aplicación o son por causas ajenas a nosotros.

Modelo temporal

Con este modelo, queremos ser capaces de analizar como influyen las distintas variables a lo largo del tiempo. Lo que nos permitirá entender cosas como en que orden se suelen acceder las páginas y su impacto en el tiempo de acceso.

9.1. Preparación de los datos

Antes de empezar a crear los distintos modelos necesitamos tratar los datos en crudo, para poder extraer el máximo de información posible. Para ello hemos creado una serie de *scripts* en Python [28], que han ido evolucionando tras distintas iteraciones.

9.1.1. Primera iteración

En esta primera iteración nos centramos en extraer información sobre los datos de uso y en añadir el país, dispositivo y navegador a dicho conjunto.

Limpieza de los datos

El primer paso para obtener un conjunto de variables que podamos utilizar para el análisis es limpiar los datos de posibles duplicados o valores erróneos.

El conjunto de los datos de uso es el que requiere de más tratamiento. Para limpiarlo hemos seguido los siguientes pasos:

1. Eliminar las filas duplicadas.
2. Pasar todos los nombres de usuario a minúsculas. Ya que detectamos que un usuario puede acceder con su nombre de usuario en mayúsculas o minúsculas.
3. Substituir los usuarios desconocidos de una sesión por el usuario real.
4. Substituir los idiomas desconocidos en una sesión por el idioma seleccionado.
5. Reordenar los datos de manera ascendente basándonos en la marca de tiempo.

En el caso de los conjuntos obtenidos del servidor Glassfish y Google Analytics lo único que hemos hecho ha sido eliminar las filas duplicadas y reordenarlas de manera ascendente a partir de la marca de tiempo.

Creación de variables

Una vez tenemos limpios los datos, pasamos a crear nuevas variables a partir de la información de la que disponemos.

La generación de nuevas variables se ha llevado a cabo sobre los datos de uso, de los que hemos extraído la información siguiente:

1. Día de la semana.
2. Tiempo de carga de la página.
3. Tiempo en la página.
4. Duración de la acción.

Integración de los datos

El último paso de la preparación de los datos es la integración de las distintas tablas en una sola que nos proporcione la información necesaria para crear los modelos.

Los primeros conjuntos que integramos son los datos de uso y los de Google Analytics. Con esta integración, lo que pretendemos es añadir información sobre el país, el navegador y el dispositivo a los datos de uso. Para ello seleccionamos dichas variables del conjunto de GA y eliminamos los duplicados. Finalmente integramos los datos con un *left join*, para no perder filas en los datos de uso, basándonos en el ID de sesión.

9.1.2. Segunda iteración

Durante la segunda iteración nos centramos en extraer variables que consideramos que podrían afectar al tiempo de carga de las páginas.

Creación de variables

Esta vez la creación de nuevas variables se ha llevado a cabo sobre los datos de uso y del servidor. En lo que respecta al primero, hemos extraído la información siguiente:

1. Número de peticiones cada 30 segundos.
2. Número de conexiones semanales por usuario.

En cuanto a los datos del servidor, hemos generado otro conjunto de datos donde pivotamos las variables a partir del tipo de error y luego calculamos el número de errores en intervalos de 5 segundos. Esta agrupación se debe a que no vemos factible asociar cada error en el servidor con una entrada de los registros de uso, ya que el error puede ocurrir varios segundos después del acceso o puede haber dos entradas de distintos usuarios en instantes muy cercanos. Por lo tanto, no podemos saber con certeza a que fila pertenece cada error.

Integración de los datos

Esta vez vamos a integrar el conjunto de datos generado en la fase de integración de la primera iteración con la nueva tabla que contiene los datos de los errores. Para ello juntamos las dos tablas a partir de la marca de tiempo más cercana con una tolerancia de 5 segundos hacia adelante. Una vez tenemos la integración, propagamos el número de errores hacia las filas siguientes hasta encontrar un valor no nulo. De esta manera podemos aproximar el número de errores que están ocurriendo en cada una de las entradas de la tabla.

Finalmente, renombramos las variables con una alta variabilidad, como el usuario o algunas acciones, para agruparlas en grupos más grandes.

9.1.3. Tercera iteración

Esta tercera iteración es la más corta, ya que su único objetivo es generar un índice que nos permita identificar el orden de las entradas para cada una de las sesiones, de manera que podamos generar el modelo temporal.

9.2. Modelo de rendimiento

El objetivo de este primer modelo es poder analizar el efecto que tienen las distintas variables sobre el tiempo de carga. De esta manera esperamos obtener una vista interna de que factores ralentizan la aplicación y así poder solucionarlos más fácilmente.

9.2.1. Primera iteración

Durante la primera iteración se ha creado una red bayesiana utilizando el método Hill-Climbing. Para ello, primero hemos tenido que discretizar las variables numéricas. En este caso, hay que destacar tres de ellas: tiempo de carga, tiempo en la página y duración de la acción. Estas variables han sido discretizadas definiendo los intervalos mostrados en las tablas siguientes.

	optimo	bajo	medio	alto
Tiempo de carga (s)	$< 0,5$	< 2	< 5	$>= 5$

Tabla 9.1: Intervalos del tiempo de carga. Fuente: Elaboración propia

	muy bajo	bajo	medio	alto
Tiempo en la página (s)	< 20	< 60	< 120	$>= 120$

Tabla 9.2: Intervalos del tiempo en la página. Fuente: Elaboración propia

	optimo	bajo	medio	alto
Duración de una acción	$< 0,5$	< 2	< 5	$>= 5$

Tabla 9.3: Intervalos de la duración de una acción. Fuente: Elaboración propia

Una vez tenemos todas las variables en el formato correcto utilizamos la librería de R bnlearn[1]. Con la ayuda de esta librería aprendemos la estructura del modelo, la cual se puede ver en la Figura 9.1.

Tal y como podemos ver en el grafo la variable `idioma` forma una conexión divergente con las variables: `pais`, `usuario`, `dispositivo` y `pagina`. Lo que nos indica que estas están *d-separadas* dado `idioma`.

Si ahora nos fijamos en el grupo de nodos derecho, vemos como las variables `dia` y `navegador` están *d-separadas* dado `usuario`. Si seguimos bajando por `navegador` vemos como se ha formado una conexión en serie de `usuario` a `dispositivo`, lo que da lugar a una conexión convergente. Estos enlaces lo que provocan es que `navegador` e `idioma` estén d-conectados aunque tengamos instanciado `usuario`, siempre y cuando dispongamos de evidencia de `dispositivo` o de cualquiera de sus descendientes. Por último podemos ver como `id_sesion` y `pais` son independientes dado `dispositivo`.

[1] https://www.bnlearn.com

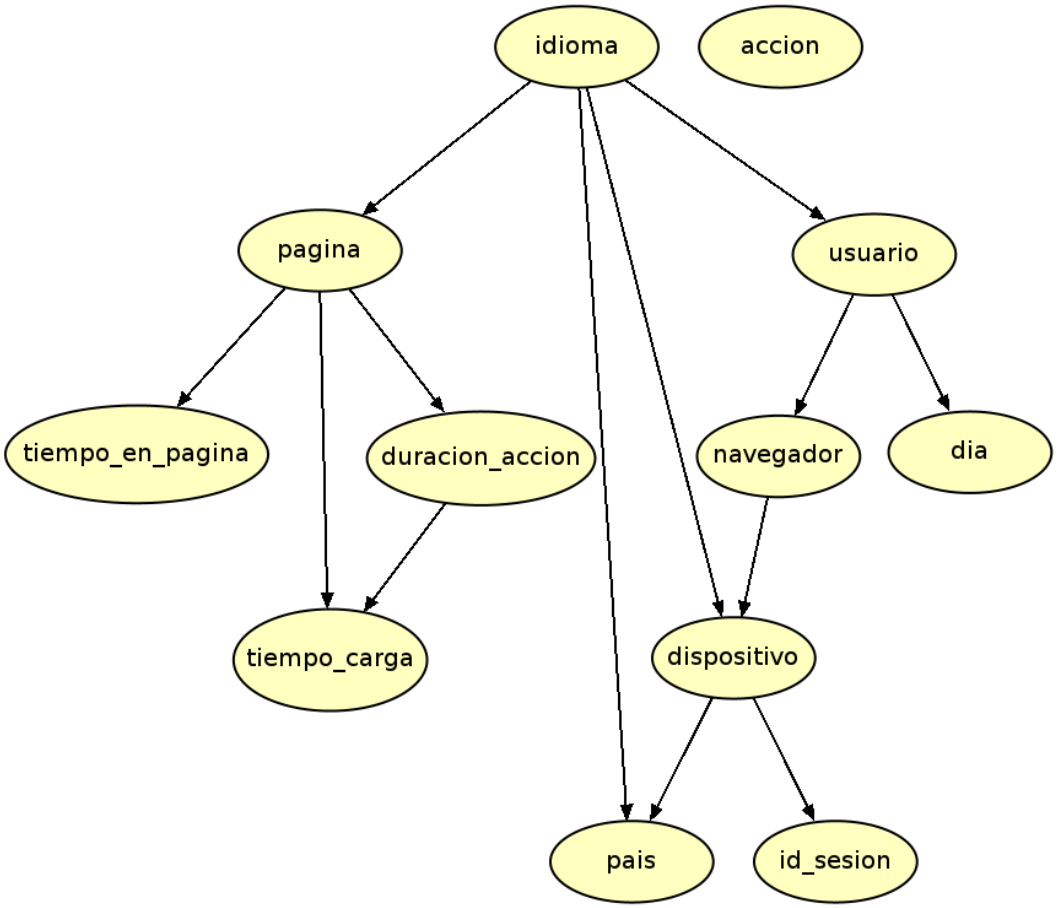

Figura 9.1: Primer modelo de rendimiento. Fuente: Elaboración propia

Si centramos nuestra atención en el grupo de la izquierda, vemos como `pagina` forma una conexión divergente, lo que da lugar a que `duracion_accion` y `tiempo_en_pagina` sean condicionalmente independientes dado `pagina`. Y de igual manera, `tiempo_carga` y `tiempo_en_pagina` son independientes dado `pagina`. En el caso de `duracion_accion` y `tiempo_carga` no existe *d-separación* entre ellas dado que existe una conexión de dependencia directa entre ambas.

Como podemos ver en general las conexiones son bastante intuitivas y forman caminos que podemos pensar como naturales. Aun así, quisiéramos destacar varios aspectos que nos han sorprendido en este primer modelo.

El primero de ellos, y más notable, es que contrario a nuestro pensamiento inicial, la variable `accion` no tiene ningún tipo de influencia sobre el resto de variables. Sospechamos que se debe a que sus valores son altamente variables, ya que representan acciones muy específicas como comprar un jugador en concreto. Por ello hemos decidido agrupar las acciones con una alta variabilidad en valores más generales en la siguiente iteración.

El segundo punto a destacar es la ausencia de una conexión entre las variables `usuario` y `pagina`, ya que en un primer momento pensamos que existiría algún tipo de dependencia entre estas dos variables, dado que entendemos que cada jugador utiliza la aplicación de manera distinta. Por ello, hemos decidido agrupar

los usuarios en dos grupos: administradores y no administradores. Ya que realmente no nos interesa el usuario en concreto, sino su rol.

Finalmente, detectamos que las variables que representan medidas temporales no están distribuidas de forma balanceada, ya que la mayoría de valores se encuentran por debajo del primer límite, y no representan correctamente el estado de la aplicación.

9.2.2. Segunda iteración

Una vez hemos terminado la segunda iteración de la fase de preparación de los datos volvemos a generar el modelo con la nueva información.

Para ello hemos redefinido los intervalos de las variables: `tiempo_carga` y `duración_accion`. Para definir los límites hemos utilizado los valores de la mediana y el tercer cuartil de cada una de las variables. A continuación se puede ver como han quedado los intervalos de cada una de las variables.

Min.	1er Cu.	Mediana	Media	3er Cu.	Max.
0,0000	0,0100	0,0580	0,4503	0,1130	1691,5410

	optimo	bajo	medio	alto
Tiempo de carga (s)	$< 0,05$	$< 0,1$	< 1	$>= 1$

Tabla 9.4: Distribución del tiempo de carga. Fuente: Elaboración propia

Min.	1er Cu.	Mediana	Media	3er Cu.	Max.
0,0000	0,0000	0,0400	8,1014	0,0900	116007,1

	optimo	bajo	medio	alto
Duración de una acción	$< 0,04$	$< 0,08$	$< 0,8$	$>= 0,8$

Tabla 9.5: Distribución de la duración de una acción. Fuente: Elaboración propia

Aparte de la discretización de las variables anteriores, se han añadido dos nuevas. Una para el número de errores (ver Tabla 9.6) y otra para el número de sesiones semanales por usuario (ver Tabla 9.7), la cual se ha utilizado para establecer el tipo de usuario.

	ninguno	bajo	medio	alto
Número de errores	< 1	< 2	< 3	$>= 3$

Tabla 9.6: Intervalos del número de errores. Fuente: Elaboración propia

Una vez tenemos todos los datos en el formato correcto pasamos a crear el nuevo modelo, el cual se puede ver en la Figura 9.2.

Como podemos ver la estructura ha cambiado considerablemente con respecto al primer modelo. Aunque muchas de las dependencias del primero se siguen manteniendo, como la relación entre `duracion_accion` y `tiempo_carga`, o la cadena

	ocasional	regular	activo	muy activo
Número de sesiones/semana	< 5	< 8	< 15	$>= 15$

Tabla 9.7: Intervalos del número de sesiones. Fuente: Elaboración propia

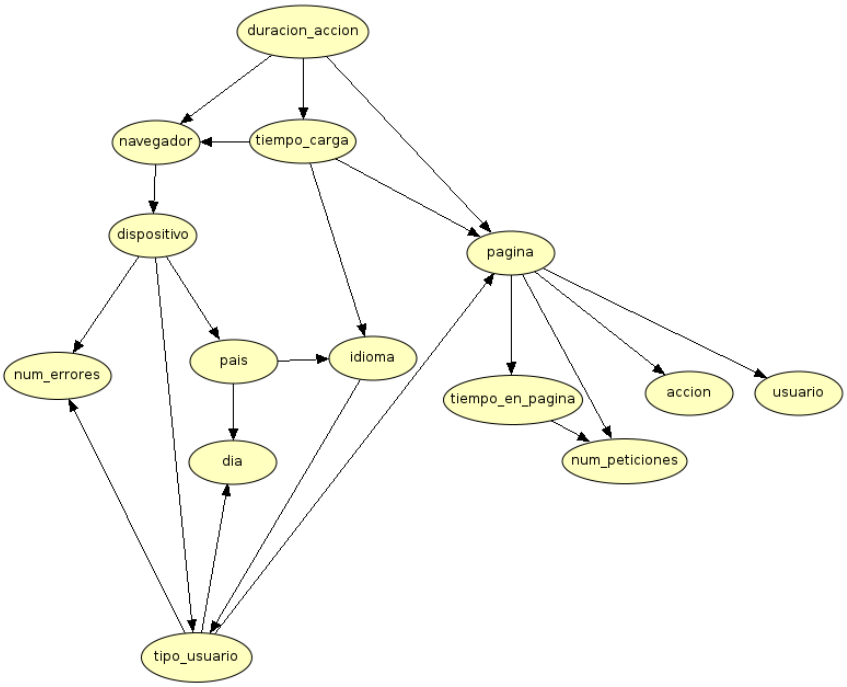

Figura 9.2: Modelo de rendimiento. Fuente: Elaboración propia

entre `navegador`, `dispositivo` y `pais`. En este nuevo modelo vemos como sí que existe una dependencia entre `pagina` y `accion`. Y de igual manera, observamos como ahora `pagina` sí que influye sobre `usuario` y viceversa.

En cuanto a la propagación de la información, ya podemos intuir que se ha vuelto mucho más compleja que antes. Por lo que es muy complicado discernir variables independientes entre sí a simple vista. Las únicas variables que podemos asegurar que son independientes del resto son: `accion` y `usuario`, si tenemos evidencia de `pagina`.

Finalmente, para analizarlo en profundidad utilizamos Hugin. En la Figura 9.3 podemos ver como si establecemos una probabilidad del 100 % de que el tiempo de carga sea alto, las páginas más probables de sufrir de este bajo rendimiento sean: Cron, Start, ViewOffers, LastMovements y Market.

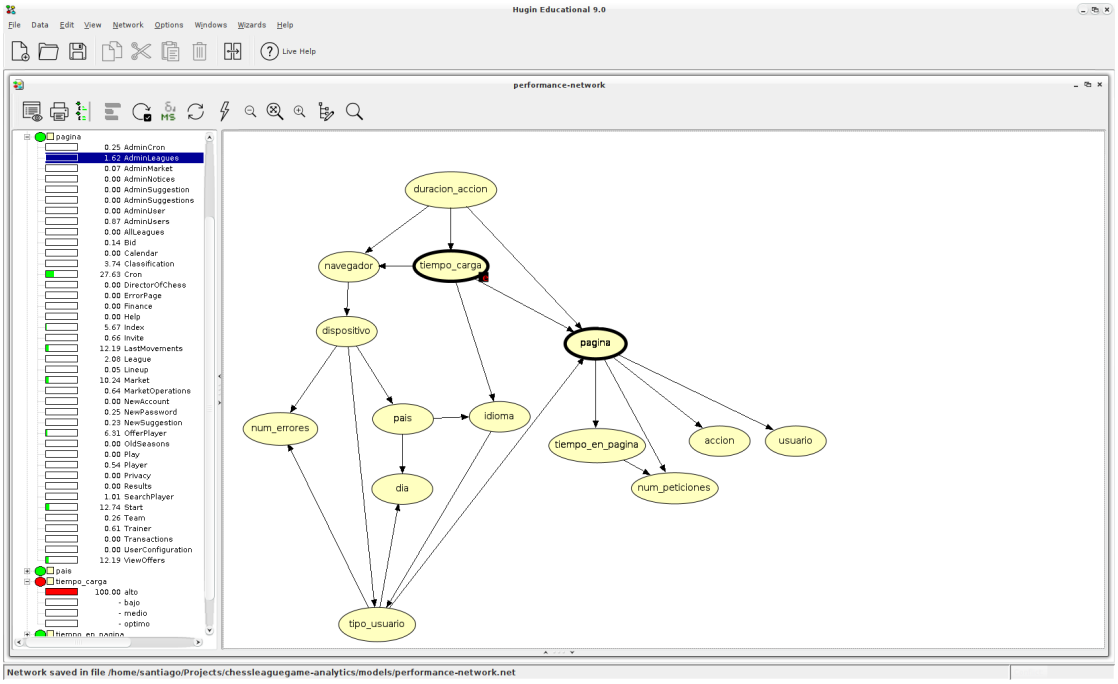

Figura 9.3: Probabilidades de que una página tenga un tiempo de carga alto. Fuente: Elaboración propia

Dado que nuestro objetivo es analizar como se ve afectado el rendimiento de la aplicación, existe un tipo de red bayesiana que se ajusta más a nuestras necesidades. Esta se conoce como naive bayes, la cual se caracteriza por ser una red donde la variable de interés se conecta con el resto formando una conexión divergente, como se ve en la Figura 9.4. Esto nos permite observar como afectan los distintos parámetros al rendimiento.

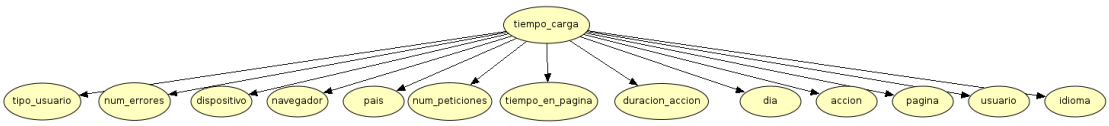

Figura 9.4: Naive bayes tomando como variable de interés el tiempo de carga. Fuente: Elaboración propia

Al analizar este modelo con Hugin, podemos jugar con las probabilidades de las distintas variables y ver las como distintas combinaciones afectan al rendimiento.

En las Figuras 9.5 y 9.6 podemos ver un ejemplo, donde ocurre algo curioso. En la primera de ellas podemos ver como si tenemos un alto número de errores y peticiones la probabilidad de que la página LastMovements tenga una pérdida de rendimiento es bastante notable. Lo más sorprendente en este caso, es que si el número de errores se mantiene alto, pero las peticiones bajan a un máximo de 3, la disminución del rendimiento de esta página es aún más probable. Como se puede ver en la segunda figura.

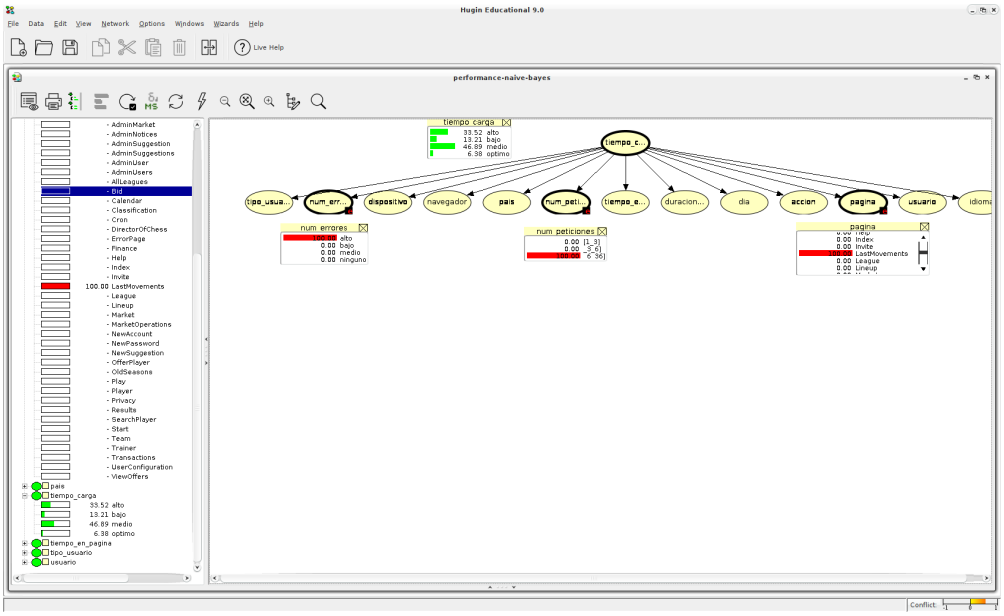

Figura 9.5: Visualización en Hugin del rendimiento de la página LastMovements cuando tenemos un alto número de errores y peticiones. Fuente: Elaboración propia

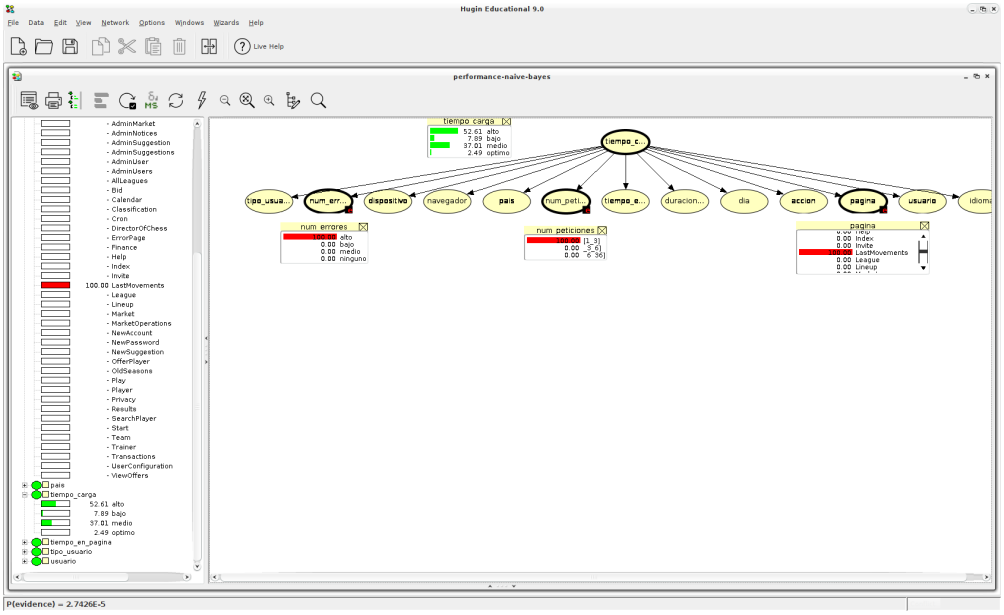

Figura 9.6: Visualización en Hugin del rendimiento de la página LastMovements cuando tenemos un alto número de errores y un bajo número de peticiones. Fuente: Elaboración propia

9.3. Modelo de errores

El objetivo de este modelo es analizar el comportamiento de los errores dentro del servidor, de manera que nos permita localizar las clases donde se originan dichos fallos y su causa principal. Para ello hemos utilizado los datos del servidor y hemos seleccionado las variables siguientes: error, clase, método, severidad y nombre del hilo.

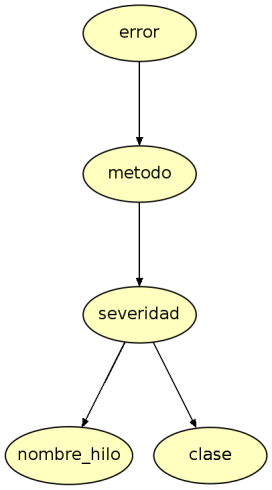

Figura 9.7: Modelo para el comportamiento de los errores. Fuente: Elaboración propia

Como podemos ver en la Figura 9.7 existe una relación de dependencia entre el error y el método donde este se origina, como cabría esperar. Una cosa interesante es que el método no influye de manera directa sobre la clase donde se produce el error, si no que este condiciona la severidad del error, y es esta la que nos aporta información sobre la clase que lanza el error.

En cuanto al flujo de información, podemos ver como claramente las variables `error` y `severidad` están *d-separadas* por `metodo`, lo que nos indica que una vez tenemos evidencia sobre la función que lanza la excepción estas dos variables se tornan independientes. De igual forma, podemos asegurar que las variables `clase` y `nombre_hilo` son condicionalmente independientes dado `severidad`.

Si exploramos el modelo con Hugin, podemos manipular las probabilidades para extraer más información del modelo. Por ejemplo, en la Figura 9.8 vemos como si fijamos la severidad al grado de SEVERE, la probabilidad de que el error sea lazado por el AutoDeployer es del 100 %, lo que nos sugiere que ahora mismo no tenemos errores de tal severidad que provengan de la implementación de la aplicación. Otro dato que nos proporciona, es que el 100 % de las veces el error que ocurre está relacionado con un problema con la base de datos.

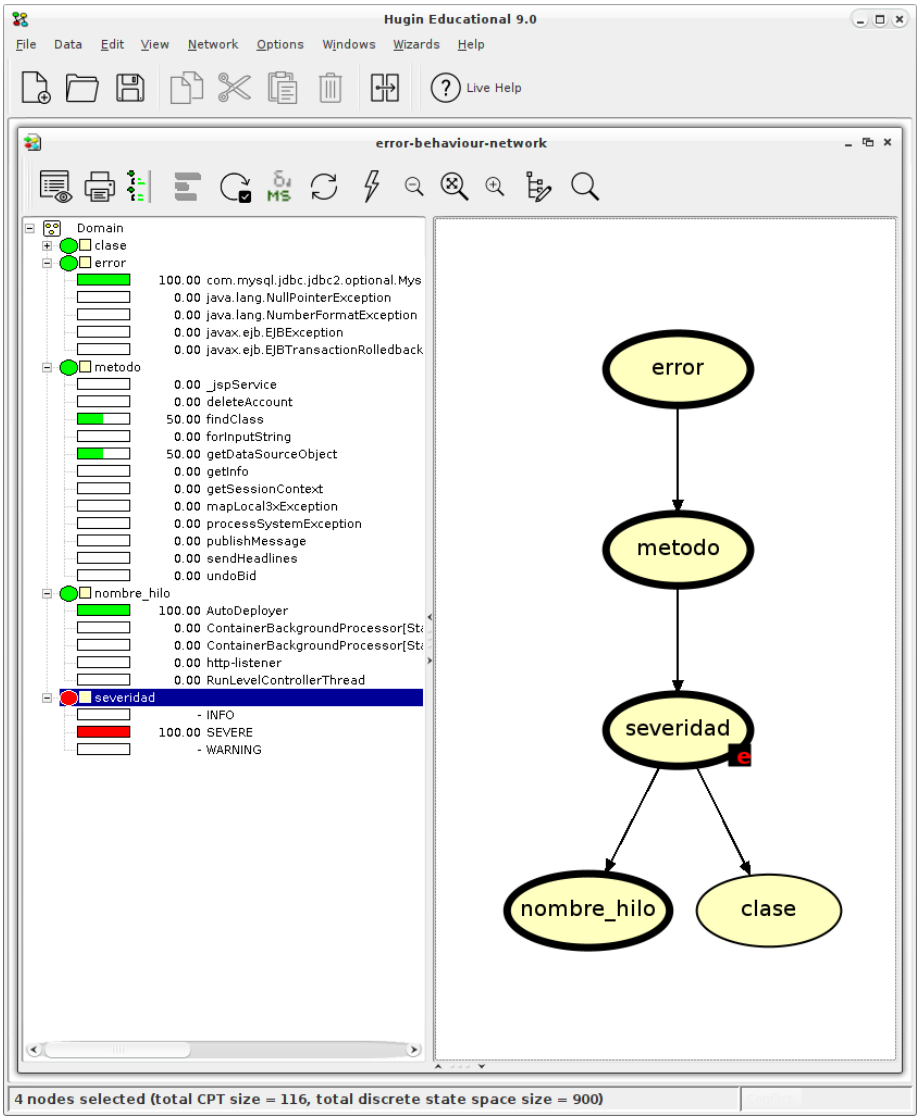

Figura 9.8: Visualización en Hugin del modelo para errores. Fuente: Elaboración propia

9.4. Modelo temporal

El objetivo de este modelo es analizar el comportamiento de los usuarios dentro de la aplicación a lo largo de varios instantes temporales. Con esto pretendemos hacernos una idea del flujo que siguen los jugadores dentro de la aplicación. De esta manera queremos entender mejor como se está utilizando la aplicación y ver si hay comportamientos particulares que nos lleven a errores o a pérdidas de rendimiento.

Para la realización de este modelo hemos utilizado los datos obtenidos de los registros de uso. En concreto, las variables utilizadas han sido: la página, el tiempo de carga, el tiempo en la página y el número de sesiones por semana. El principal motivo para elegir este subconjunto de variables es para tener un modelo que se pueda entender a simple vista y que contenga los datos que nos permitan alcanzar el objetivo establecido para el modelo.

El primer paso para la realización del modelo ha sido la discretización de las variables numéricas. Para ello hemos establecido los mismos intervalos que en la sección 9.2.2.

A continuación pasamos a crear las distintas variables que representan los distintos instantes de tiempo. Dado que el número de accesos en cada sesión no es homogéneo, hemos optado por considerar todo el conjunto como una sola sesión y replicar esos datos para representar múltiples instantes. Para ello, hemos utilizamos la librería de R dbnR[2].

El proceso que se utiliza es muy simple. Lo único que necesitamos es indicar la cantidad de instantes temporales queremos tener y la librería se encarga de replicar los datos tantas veces como instantes le indiquemos. Hay que notar que para que cada momento no sea idéntico al anterior, a cada nueva réplica se le elimina la última entrada y a su predecesora la primera.

Debido a esta implementación, para que los accesos queden correctamente ordenados hemos tenido que reordenar el conjunto de datos por ID de sesión, lo que nos asegura que no tendremos accesos mezclados entre sesiones, y luego por orden descendente de acceso, lo que nos asegura que se ordenan de manera correcta al replicar las columnas. En las Tablas 9.8 y 9.9 podemos ver un ejemplo del resultado que obtenemos al usar esta librería.

orden_acceso
6
5
4
3
2
1

Tabla 9.8: Ejemplo de una tabla antes de crear los instantes temporales.
Fuente: Elaboración propia

orden_acceso_t_0	orden_acceso_t_1	orden_acceso_t_2
4	5	6
3	4	5
2	3	4
1	2	3

Tabla 9.9: Ejemplo de una tabla después de crear los instantes temporales.
Fuente: Elaboración propia

Finalmente, aprendemos la red utilizando de nuevo el método Hill-Climbing. Lo que nos da como resultado el modelo de la Figura 9.9.

[2] https://github.com/dkesada/dbnR

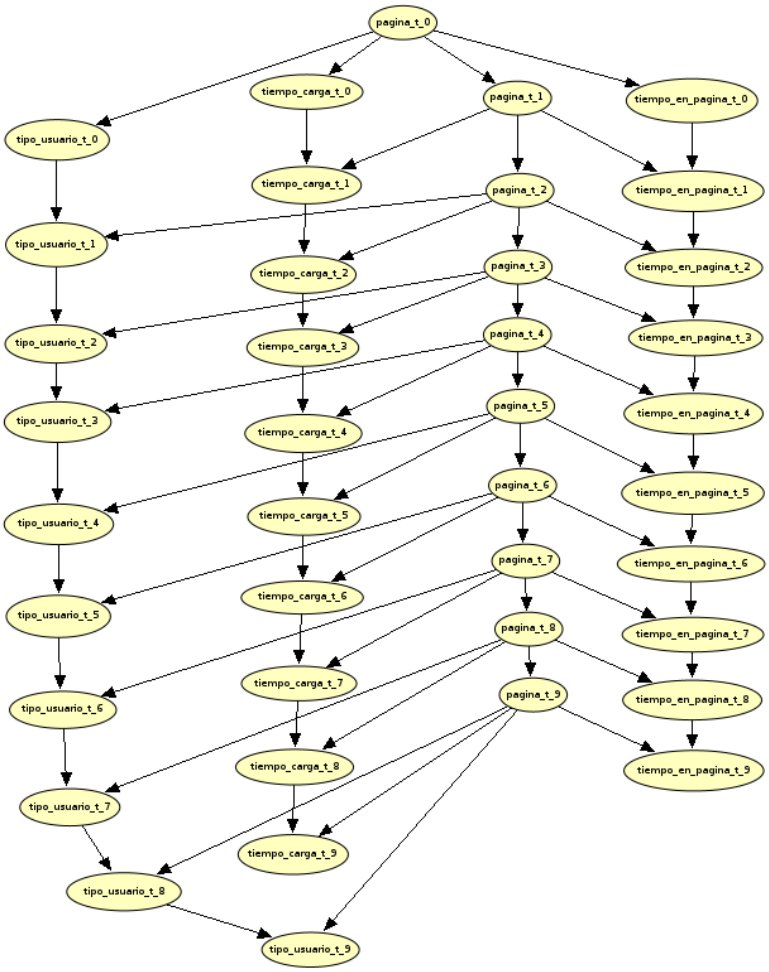

Figura 9.9: Modelo temporal. Fuente: Elaboración propia

Si nos fijamos ahora en el modelo, este tiene una estructura bastante clara formada por los tres principales tipos de conexiones vistos en la sección 2.1.

Por un lado, vemos como todas las variables pertenecientes al mismo tipo siguen una conexión en serie en el orden natural.

Por otro lado, vemos como las páginas en los distintos instantes temporales forman una conexión divergente con el resto de variables en ese instante menos para el tipo de usuario, que generalmente se conecta al del instante anterior. Como excepción a esto último vemos como para el instante T0 la página define el tipo de usuario en T0 y en cambio la página en T1 tiene conexión con el tipo de usuario.

Las dos conexiones explicadas anteriormente dan lugar a una tercera conexión, en este caso convergente. Donde cada una de las variables, excepto la página, a partir de T1 hacen de nodo intermedio conectado la misma variable en el instante anterior con la página que hace de nexo en la conexión divergente.

Para extraer el máximo de este modelo lo analizamos con Hugin, donde una vez empezamos a manipular las probabilidades de las variables podemos extraer más información del modelo.

Un ejemplo es comprobar que páginas son más probables de ser accedidas nada más un usuario entra al juego. Para ello fijamos la página en T0 a Index y en T1 a

Start, que son las dos páginas que se han de visitar para entrar a la aplicación. Una vez hemos fijado estas variables, podemos ver como la página de resultados es la más probable de ser accedida a continuación, seguida del mercado y la alineación. En la Figura 9.10 podemos ver como queda representado en Hugin.

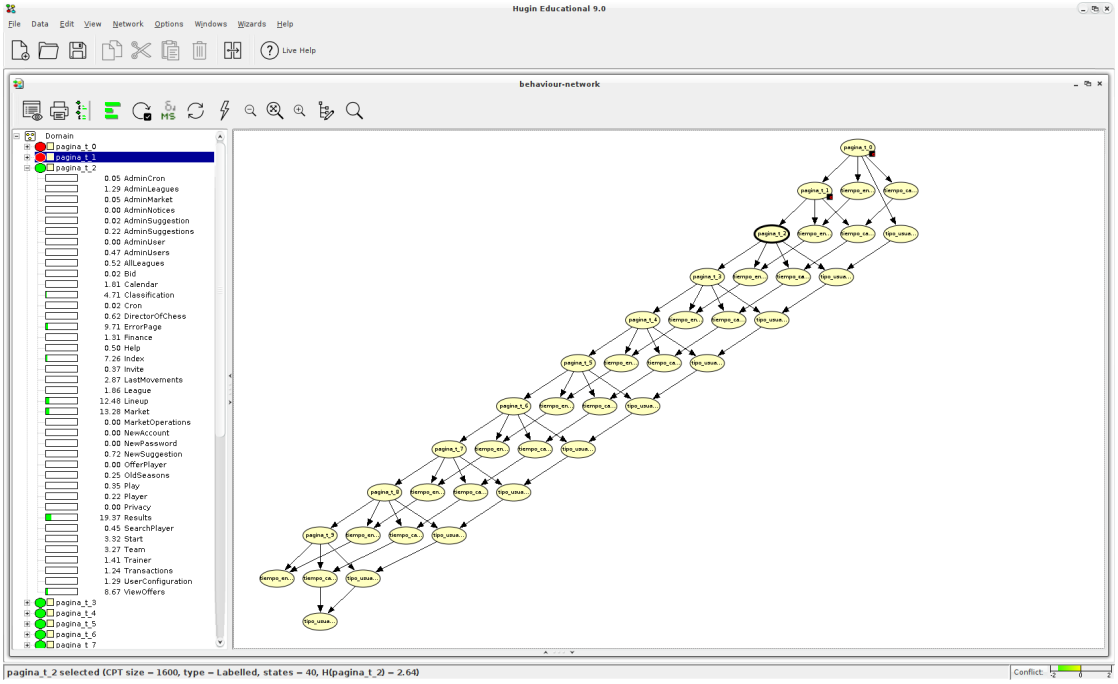

Figura 9.10: Visualización con Hugin de las páginas más visitadas al acceder a Chessleague. Fuente: Elaboración propia

Aplicando diferentes modificaciones en las probabilidades también podemos averiguar que secuencia de páginas acaban llevándonos a la página de error. Lo que nos puede ayudar a detectar errores en el código que hasta ahora han pasado desapercibidos.

Las principales contribuciones de la parte 2 han sido:

- Creación de tres redes bayesianas para el análisis del rendimiento, el análisis de los errores y el análisis del comportamiento de la aplicación.
- Identificar los siguientes puntos de mejora en ChessLeague:
  - Las páginas con mayor probabilidad de sufrir una bajada de rendimiento son: Start, Cron, ViewOffers, Market y LastMovements.
  - El error más común dentro de la aplicación es NullPointerException, el cual se origina, generalmente, en la página de error.
  - Las primeras páginas que los usuarios suelen acceder nada más entrar en la aplicación son la de resultados y el mercado. Por lo que se podría incorporar parte de la información de esas páginas a la página de inicio a modo de previsualización.

10. Desarrollo dirigido por datos

En este capítulo de describen las tareas realizadas durante la fase de desarrollo del proyecto utilizando los datos obtenidos en las fases anteriores.

10.1. Product Backlog

Tal y como indica la metodología Scrum, antes de empezar el primer sprint, el equipo tiene que elaborar un Product Backlog, es decir, una lista de nuevas características o tareas a realizar ordenadas por prioridad. Generalmente, esta lista se obtiene a partir del análisis de requisitos y las historias de usuario que resultan de este. En el caso de este proyecto se han analizado los resultados del estudio realizado en los capítulos anteriores y se han creado las tarjetas a partir de estos.

Para diferenciar de dónde proviene cada una de las tareas se ha optado por crear cinco épicas distintas que se definen a continuación junto con cada una de sus tarjetas. Las épicas, detalladas junto a sus tarjetas a continuación son:

10.1.1. Mejoras basándonos en las visualizaciones en Kibana

Estas tarjetas se han creado después de analizar los resultados obtenidos de la creación de los distintos dashboards, presentados en el capítulo 8, los cuales nos muestran como tenemos un alto número de peticiones que se topan con recursos no disponibles.

Añadir banderas faltantes

Como usuario
Quiero poder ver los iconos de las banderas de cada federación
Con el objetivo de mejorar la experiencia de usuario dentro del juego.

Criterios de aceptación:

- Ha de existir una imagen .gif para todas las banderas que aparecen en el juego.

Esfuerzo estimado: 1

Añadir un favicon a la aplicación

Como administrador
Quiero disponer de un favicon en la aplicación
Con el objetivo de que esta se asocie a un icono y sea reconocible más fácilmente.

Criterios de aceptación:

- El favicon ha de ser visible en los principales navegadores y dispositivos.
- El favicon ha de ser visible en el modo oscuro de los navegadores.

Esfuerzo estimado: 1

10.1.2. Mejoras basándonos en el modelo de rendimiento

Estas tareas se han creado después de analizar el modelo de rendimiento presentado en el capítulo 9, con el que hemos sido capaces de detectar los principales lugares con pérdidas de rendimiento.

Optimizar el rendimiento de la página de inicio

Como desarrollador
Quiero poder mejorar el rendimiento de la página Start
Con el objetivo de mejorar la experiencia de usuario dentro del juego.

Criterios de aceptación:

- Se ha de reducir el tiempo de carga en un 15 % con respecto al actual.

Esfuerzo estimado: 3

Optimizar el rendimiento de la página Market

Como desarrollador
Quiero poder mejorar el rendimiento de la página Market
Con el objetivo de mejorar la experiencia de usuario dentro del juego.

Criterios de aceptación:

- Se ha de reducir el tiempo de carga en un 15 % con respecto al actual.
- Se ha de ha de mejorar la paginación de las ofertas.

Esfuerzo estimado: 3

Optimizar el rendimiento de la página ViewOffers

Como desarrollador
Quiero poder mejorar el rendimiento de la página ViewOffers
Con el objetivo de mejorar la experiencia de usuario dentro del juego.

Criterios de aceptación:

- Se ha de reducir el tiempo de carga en un 15 % con respecto al actual.

Esfuerzo estimado: 3

Optimizar el rendimiento de la página LastMovements

Como desarrollador
Quiero poder mejorar el rendimiento de la página LastMovements
Con el objetivo de mejorar la experiencia de usuario dentro del juego.

Criterios de aceptación:

- Se ha de reducir el tiempo de carga en un 15 % con respecto al actual.

Esfuerzo estimado: 3

Optimizar el rendimiento del cron diario

Como desarrollador
Quiero poder mejorar el rendimiento del cron diario
Con el objetivo de reducir el tiempo para computar los movimientos del mercado.

Criterios de aceptación:

- El tiempo para completar el cron diario se reduce en un 15 %.

Esfuerzo estimado: 5

Optimizar el rendimiento del cron de jornada

Como desarrollador
Quiero poder mejorar el rendimiento del cron de jornada
Con el objetivo de reducir el tiempo para calcular los resultados de las jornadas.

Criterios de aceptación:

- El tiempo para completar el cron de jornada se reduce en un 15 %.

Esfuerzo estimado: 5

10.1.3. Mejoras basándonos en el modelo de errores

Estas tarjetas se han creado después de analizar el modelo de errores descrito en el capítulo 9, con el que hemos averiguado el origen de las principales excepciones que ocurren en la aplicación.

Resolver los NullPointerException originados en ErrorPage

Como desarrollador
Quiero reducir el número de errores originados en la página de error
Con el objetivo de mejorar la calidad de la aplicación.

Criterios de aceptación:

- Se han de reducir en un 70 % el número de `NullPointerException` dentro de la ErrorPage.

Esfuerzo estimado: 5

Analizar el origen de los NumberFormatException

Como desarrollador
Quiero poder saber donde se originan los `NumberFormatException`
Con el objetivo de poder evitar dichos errores.

Criterios de aceptación:

- Se han de reducir en un 70 % el número de `NumberFormatException` dentro de la aplicación.

Esfuerzo estimado: 3

10.1.4. Mejoras basándonos en el modelo temporal

Estas tarjetas se han creado después de analizar el modelo temporal presentado en el capítulo 9, con el que hemos descubierto comportamientos anómalos, así como las páginas que mas se visitan.

Investigar como se llega a la página de error desde la página OfferPlayer

Como desarrollador
Quiero poder saber que secuencia de acciones realizadas desde OfferPlayer llevan a la página de error
Con el objetivo de poder arreglar errores aún no detectados.

Criterios de aceptación:

- Se ha de detectar como mínimo una secuencia de acciones que lleve a la página de error desde OfferPlayer.

Esfuerzo estimado: 5

Añadir resultados de la última ronda a la página de inicio

Como usuario
Quiero poder ver los resultados de mi última jornada en la página de inicio
Con el objetivo de ver como ha ido última jornada sin necesidad de navegar a la página de resultados.

Criterios de aceptación:

- Se ha de mostrar una tabla con los resultados de la última jornada del usuario.
- El rendimiento de la página no ha de disminuir más de un 10
- La tabla ha de estar disponible en todos los idiomas ofrecidos.

Esfuerzo estimado: 3

10.1.5. Mejoras basándonos en las sugerencias

Estas tarjetas se han creado después de analizar las sugerencias creadas por los usuarios de ChessLeague.

Bajar a 100 el número de jugadores que salen a la venta y que tengan más de 1700 de ELO

Como usuario
Quiero poder pujar por jugadores con más de 1700 de ELO
Con el objetivo de mejorar la calidad de mi equipo.

Criterios de aceptación:

- Cada día han de salir 100 jugadores nuevos al mercado.
- No puede haber más de 200 jugadores ofrecidos por el sistema en el mercado.
- Los jugadores ofrecidos por el sistema han de tener un ELO mayor o igual a 1700.

Esfuerzo estimado: 1

Cambiar la frecuencia de las jornadas

Como usuario
Quiero que las jornadas se computen más frecuentemente
Con el objetivo de que el juego sea más dinámico y las temporadas más cortas.

Criterios de aceptación:

- Las jornadas se han de jugar los martes, jueves, viernes, sábados y domingos.

Esfuerzo estimado: 1

Notificar a los usuarios cuando otro jugador puja por el mismo ajedrecista

Como usuario
Quiero poder ser notificado cuando alguien supere mi puja por un ajedrecista
Con el objetivo de poder volver a pujar en caso de estar interesado y no perder el jugador.

Criterios de aceptación:

- Se ha de informar al usuario de la posibilidad de recibir notificaciones mediante un checkbox.
- Si el navegador no soporta el uso de las notificaciones, el checkbox a de estar oculto.
- Se ha de solicitar permiso a los usuarios para enviarles notificaciones.
- Si el usuario está suscrito a las notificaciones en un navegador, se le ha de suscribir en el resto de navegadores si este lo permite.
- Si el usuario elimina su suscripción en un navegador, se han de eliminar todas las suscripciones del resto de navegadores.
- El sistema de notificaciones ha de funcionar en los principales navegadores.
- El mensaje de las notificaciones ha de estar disponible en todos los idiomas ofrecidos en la aplicación.

Esfuerzo estimado: 5

Mejorar la generación de jugadores aleatorios del mercado

Como desarrollador
Quiero poder mejorar la aleatoriedad de los jugadores puestos a la venta por el sistema
Con el objetivo de ofrecer una mejor experiencia de juego a los usuarios.

Criterios de aceptación:

- La nacionalidad de los jugadores no se ha de acumular en una en concreto.
- Se han de poner en venta jugadores con distintos rangos de ELO, sin centrarse en una franja en particular.

Esfuerzo estimado: 5

Mostrar un mensaje al usuario cuando se ha cambiado la contraseña

Como usuario
Quiero ser avisado cuando se ha cambiado mi contraseña
Con el objetivo de saber si esta se ha cambiado correctamente.

Criterios de aceptación:

- Se ha de mostrar un mensaje de alerta confirmando el cambio de la contraseña.
- El mensaje mostrado ha de estar en el idioma seleccionado por el usuario.

Esfuerzo estimado: 1

Ordenar los comentarios de Facebook de la página de inicio por fecha descendente

Como usuario
Quiero poder ver los comentarios de Facebook en la página de inicio ordenados cronológicamente de manera descendente
Con el objetivo de ver la actividad más reciente del resto de usuarios.

Criterios de aceptación:

- Se han de mostrar los comentarios de los usuarios con cuenta de Facebook de manera que los más recientes salgan primero.

Esfuerzo estimado: 3

Cambiar los iconos de las redes sociales en la landing page

Como administrador
Quiero mostrar los iconos de la landing page sin deformar
Con el objetivo de mejorar la apariencia de la aplicación.

Criterios de aceptación:

- Se han de cambiar los iconos del pie de página en la landing page por unos más actualizados y que no se deformen.

Esfuerzo estimado: 1

Admitir nombres de usuarios que contengan minúsculas, mayúsculas y números

Como usuario
Quiero poder utilizar un nombre de usuario que contenga caracteres alfanuméricos incluyendo mayúsculas y minúsculas
Con el objetivo de tener más libertad en el momento de elegir un nombre de usuario.

Criterios de aceptación:

- El nombre de usuario solamente ha de aceptar letras mayúsculas y minúsculas, y números.
- El nombre de usuario ha de distinguir entre mayúsculas y minúsculas.
- No pueden existir dos cuentas con el mismo nombre de usuario.

Esfuerzo estimado: 5

Mostrar antigüedad de los jugadores en la plantilla

Como usuario
Quiero poder ver la antigüedad de los ajedrecistas de mi plantilla
Con el objetivo de saber si ha pasado el tiempo mínimo para poder venderlos.

Criterios de aceptación:

- Se ha de mostrar una columna adicional donde se indique la antigüedad de los jugadores.
- La antigüedad se ha de mostrar en semanas.

Esfuerzo estimado: 1

Añadir mensajes privados entre jugadores

Como usuario
Quiero poder enviar mensajes privados al resto de jugadores
Con el objetivo de comunicarme con el resto de usuarios dentro de la aplicación.

Criterios de aceptación:

- Solamente los usuarios implicados en la conversación pueden ver los mensajes.
- La longitud máxima de un mensaje ha de ser de 200 caracteres.
- Cuando un usuario recibe un nuevo mensaje se ha de enviar una notificación.

Esfuerzo estimado: 8

Asignación de equipos a nuevos usuarios a ligas donde ya hay jugadores

Como usuario recién registrado
Quiero poder estar en un grupo donde haya jugadores humanos
Con el objetivo de poder competir contra el resto de usuarios.

Criterios de aceptación:

- Se ha de asignar un equipo que pertenezca a un grupo donde ya existan usuarios humanos.
- Se ha de empezar la búsqueda de equipos desde la liga 1 a las inferiores.

Esfuerzo estimado: 8

Añadir filtros al mercado

Como usuario
Quiero poder filtrar los jugadores del mercado
Con el objetivo de poder ver solamente aquellos ajedrecistas que me interesen.

Criterios de aceptación:

- Se ha de poder ordenar por ELO, título, federación, nombre, apellido y precio.
- Se ha de poder filtrar por rango de ELO y precio.
- Se ha de poder filtrar por nombre, apellido, título y federación.

Esfuerzo estimado: 8

Añadir fecha y hora al calendario de las jornadas

Como usuario
Quiero poder ver en el calendario el día y hora en que se jugarán las jornadas
Con el objetivo de saber cuando jugaré la próxima jornada de manera que pueda preparar mi alineación.

Criterios de aceptación:

- Se ha de mostrar la fecha y hora en el formato correcto dependiendo de la configuración del usuario.
- La fecha de una jornada no puede ser anterior a la fecha de inicio de la temporada.
- La fecha de una jornada no puede ser anterior a la de las jornadas previas.

Esfuerzo estimado: 5

10.2. Definition of done

Antes de pasar a implementar las tareas hay que elaborar una lista de condiciones que se han de cumplir para dar por completada una tarea del *backlog*. A continuación se listan las condiciones establecidas en este proyecto.

- La tarea está bien definida y contiene criterios de aceptación.
- Está implementada y pasa todos los tests unitarios que verifican sus criterios de aceptación.

10.3. Sprints

Una vez elaborado el Product Backlog y la definición de completo podemos empezar con los sprints, donde se desarrollarán las tarjetas que se pacten durante el sprint planning. Debido a la limitación de tiempo para la finalización del proyecto se ha decidido que se realizarán tres sprints de una semana cada uno.

10.3.1. Primer sprint

Durante el sprint planning se ha decidido que en esta iteración se realizarán las siguientes tareas:

1. Añadir un favicon a la aplicación
2. Añadir banderas faltantes
3. Cambiar la frecuencia de las jornadas
4. Bajar a 100 el número de jugadores que salen a la venta y que tengan más de 1700 de ELO
5. Notificar a los usuarios cuando otro jugador puja por el mismo ajedrecista

Para las dos primeras tareas únicamente necesitamos buscar las imágenes necesarias e incluirlas en el proyecto. La tercera tarea solamente requiere cambiar la frecuencia con la que se ejecuta el cron de jornada, y añadir los días que se juegan las jornadas en la página de ayuda.

En cuanto a la cuarta tarea, únicamente se ha tenido que modificar una constante dentro de la clase que selecciona los jugadores que salen a la venta, y añadir a la sentencia SQL que obtiene los ajedrecistas de la base de datos la condición de que han de tener más de 1700 de ELO.

Como se puede ver, la implementación de las tareas anteriores ha sido trivial.

Por otro lado, el desarrollo de la última tarea ha requerido la implementación de varias clases para utilizar la Push API[1]. Entre las adiciones más destacadas se encuentran la creación de la tabla SUBSCRIPTIONS, que registra las subscripciones a este servicio de los distintos usuarios, además de la creación de la clase

[1] www.w3.org/TR/push-api/

Figura 10.1: Ejemplo de una notificación. Fuente: Elaboración propia

PushController, la cual gestiona el envío de notificaciones, y de un service worker que se encarga de mostrar las notificaciones en el navegador. En la Figura 10.1 se puede ver el resultado final de esta tarea.

10.3.2. Segundo sprint

Durante el segundo sprint se han realizado las siguientes tareas:

1. Optimizar el rendimiento de la página de inicio
2. Resolver los NullPointerException originados en ErrorPage
3. Investigar como se llega a la página de error desde la página OfferPlayer
4. Ordenar los comentarios de Facebook de la página de inicio por fecha descendente

La realización de la tarea 4 ha sido trivial, ya que únicamente requería de la adición de la opción `data-order-by="reverse_time"` dentro de la etiqueta que muestra los comentarios.

Optimizar el rendimiento de la página de inicio

Una vez realizada la cuarta tarea, pasamos a optimizar la página de inicio. Para ello primero analizamos que se está realizando en esta página que pueda afectar a su rendimiento. Cuando revisamos el código, vemos que la tarea más intensiva en la página se realiza cuando esta solicita todos los avisos que existen, ordenados de manera ascendente por tiempo, y luego muestra los seis últimos en una tabla (ver Código Fuente 10.1).

```
<div class="cuerpoInfo" align="justify">
    <% List<MessageInfo> messages = accountRemote.getNotices();
          if (messages != null) {
    %>
    <table>
        <thead>
            <tr>
                <th>
                    <%=messages_i18n.getString("start.notices")%>
                </th>
            </tr>
        </thead>
        <tbody>
            <% for (int i = messages.size() - 1, j = 0; i >= 0 &&
                    j < 6; i--, j++) {%>
```

```
16              <tr>
17                  <td>
18                      <%=messages.get(i).getDescription()%>
19                      <br/>
20                      <%=messages.get(i).getInfoMessage()%>
21                  </td>
22              </tr>
23              <%}
24            }%>
25          </tbody>
26      </table>
27  </div><br/>
```

Código Fuente 10.1: Versión antigua de la tabla de avisos. Fuente: Elaboración propia

Para reducir el uso de memoria, se ha creado una nueva función que únicamente devuelve los n últimos avisos, siendo n una constante definida en la propia página Start (ver Código Fuente 10.2).

```
1   <div class="cuerpoInfo" align="justify">
2   <% List<MessageInfo> messages =
3       accountRemote.getNotices(NOTICES_DISPLAYED);
4      if (messages != null) {%>
5       <table>
6           <thead>
7               <tr>
8                   <th>
9                       <%=messages_i18n.getString("start.notices")%>
10                  </th>
11              </tr>
12          </thead>
13          <tbody>
14              <% for (MessageInfo message : messages) {%>
15              <tr>
16                  <td>
17                  <%=message.getDescription()%>
18                  <br/>
19                  <%=message.getInfoMessage()%>
20                  </td>
21              </tr>
22              <%}
23      }%>
24          </tbody>
25      </table>
26  </div><br />
```

Código Fuente 10.2: Versión nueva de la tabla de avisos. Fuente: Elaboración propia

Internamente también se ha optimizado la obtención de los avisos. Para ello, se ha creado una nueva consulta que nos devuelve directamente una lista de `MessageInfo`, ya que hasta ahora se obtenían todos los avisos y luego se iteraba sobre ellos para crear otra lista de tipo `MessageInfo` (ver Código Fuente 10.3).

```
SELECT NEW chessleague.util.MessageInfo(m.account, m.idMessage, m.timeM,
↪  m.description, m.isNotice) FROM Message AS m
WHERE m.isNotice = true
ORDER BY m.timeM DESC
```

Código Fuente 10.3: Consulta para obtener las notificaciones. Fuente: Elaboración propia

Pese a que no son cambios excesivamente grandes y que a corto plazo no se ven reflejados en el rendimiento, esperamos que mejoren la escalabilidad de la aplicación a largo plazo, cuando esta tenga que soportar un número considerable de usuarios conectados al mismo tiempo.

Resolver los NullPointerException originados en ErrorPage

Para la realización de esta tarea hemos utilizado Kibana, ya que nos permite filtrar todos los mensajes de error por tipo, mensaje, clase de origen, etc. En la Figura 10.2 se puede ver como el método que provoca la excepción siempre es el mismo `_jspService()`. Además, también se observa que la línea que provoca este error se repite en todos los casos.

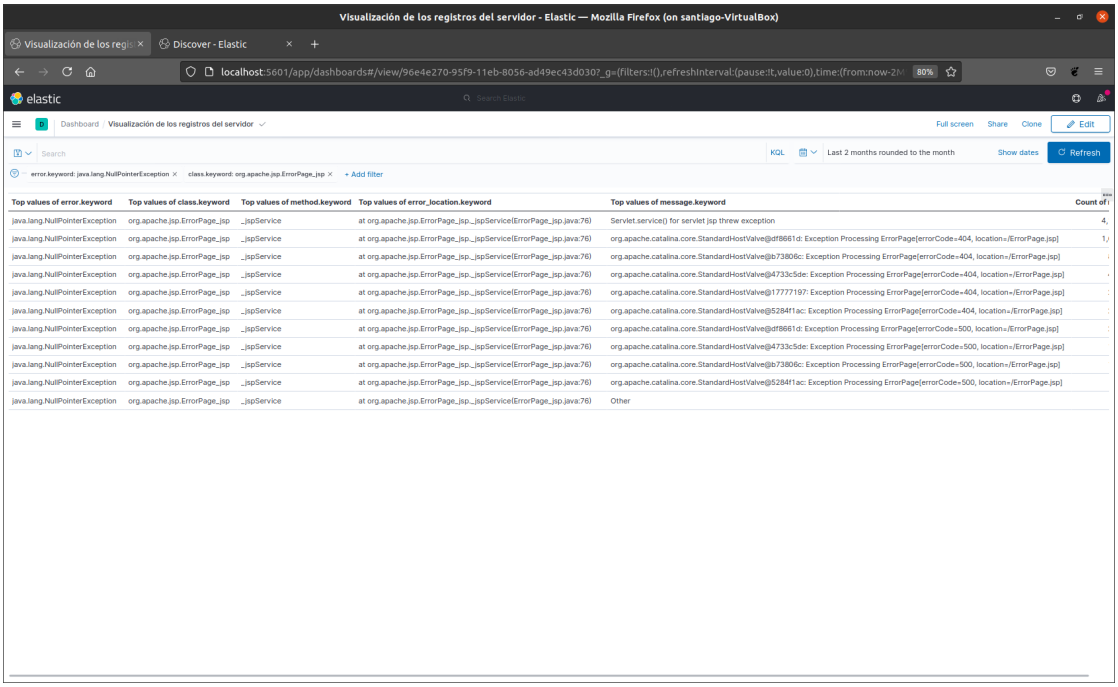

Figura 10.2: Tabla con la principal información de los NullPointerException en ErrorPage. Fuente: Elaboración propia

Si vamos al código y vemos que hace esta línea, podemos ver como lo único que hace es obtener el título de la página del fichero de idiomas correspondiente. En este caso, es muy extraño que se lance este tipo de excepción debido a esta línea. Aún así, se ha cambiado la manera de acceder a esta variable cambiando la

notación actual por una notación *null-safe*. Aún así, no podemos afirmar al cien por cien que esta sea la causa real del problema, ya que no hemos sido capaces de reproducir el error. Por lo tanto, habrá que esperar unos meses para volver a comprobar el estado de estos errores y ver si realmente se han solucionado.

```
<div class="titInfo" align="center">
    <%=messages_i18n.getString("errorpage.title")%>
</div><br/>
```

Código Fuente 10.4: ErrorPage con notación no segura. Fuente: Elaboración propia

```
<div class="titInfo" align="center">
    \${messages_i18n.getString("errorpage.title")}
</div><br/>
```

Código Fuente 10.5: ErrorPage con notación segura. Fuente: Elaboración propia

Investigar como se llega a la página de error desde la página OfferPlayer

La última tarea por realizar está bastante relacionada con la tarea anterior, ya que en esta también analizamos los datos en Kibana, para poder entender que excepciones se lanzan.

Gracias al modelo temporal sabemos que la secuencia de páginas que se visitan desde OfferPlayer hasta ErrorPage (ver Figura 10.3). Con esta información empezamos a buscar en Kibana que errores se lanzan en OfferPlayer o Team.

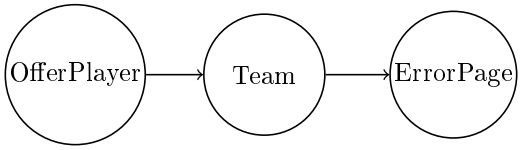

Figura 10.3: Secuencia de páginas que llevan a ErrorPage desde OfferPlayer. Fuente: Elaboración propia

Si nos fijamos en los resultados de la Figura 10.4 podemos ver como se repite el mismo error que en la tarea anterior, pero ahora para las páginas OfferPlayer y Team. En este caso no hemos sido capaces de encontrar el fallo dentro de los JSP, ya que las líneas que indica el error contienen fragmentos de JavaScript para utilizar aplicaciones de terceros como Google Analytics o directamente código fuente no tiene tantas líneas, como ocurre en el caso de Team.jsp que únicamente tiene 269 líneas. De esto extraemos que estos errores pueden deberse a problemas de configuración o a otros factores externos a la implementación.

Aún así, tras hacer pruebas manuales hemos sido capaces de detectar un fallo a la hora de ponerle un precio a un jugador. En este caso, hemos descubierto que la aplicación admite crear ofertas con valores negativos, lo cual no debería ser posible. Para solventar este problema, se ha añadido un filtro en JavaScript para que este campo solo admita números naturales. Además, se ha implementado la verificación de este campo en el lado del servidor, para prevenir los casos en que se consiga evitar el filtro de alguna forma.

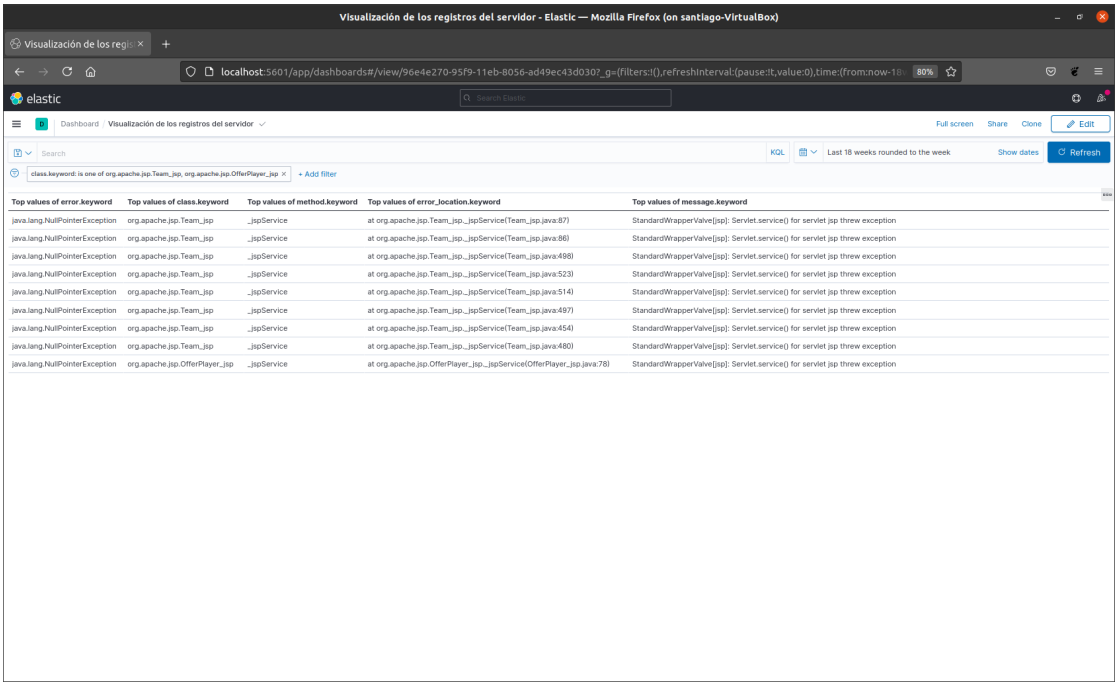

Figura 10.4: Visualización de los errores en las páginas OfferPlayer y Team. Fuente: Elaboración propia

10.3.3. Tercer sprint

La realización de este tercer sprint ha coincidido con la última semana de preparación de la memoria, por lo que se ha decidido incluir tareas con menos carga para dedicar más tiempo a la finalización de este documento. Por lo tanto, durante el tercer sprint se han realizado las siguientes tareas:

1. Mostrar un mensaje al usuario cuando se ha cambiado la contraseña
2. Mejorar la generación de jugadores aleatorios del mercado
3. Analizar el origen de los NumberFormatException
4. Añadir resultados de la última ronda a la página de inicio

La realización de la primera tarea ha sido trivial, únicamente ha requerido de la adición del siguiente código dentro de la página de configuración.

```
<% if (updateResult) { %>
    <p style="text-align: center; font-size: 150%;">
        ${messages_i18n.getString("userconfiguration.updated")}
    </p><br/><br/>
<%}%>
```

Código Fuente 10.6: Fragmento que muestra un mensaje al actualizar los datos de usuario correctamente. Fuente: Elaboración propia

Mejorar la generación de jugadores aleatorios del mercado

La realización de esta tarea ha sido bastante simple. Esto se debe a que el principal problema de la obtención de jugadores de manera aleatoria estaba en la mala generación de los índices aleatorios. El motivo es que hasta ahora se estaba utilizando incorrectamente el generador de números aleatorios de la clase `Random`. Para solucionar este problema hemos cambiado la implementación de la generación de estos índices y hemos pasado a utilizar la clase `Math`, la cual nos gestiona de forma transparente la creación de un generador de números pseudo-aleatorios.

Otro cambio ha sido la cantidad de jugadores solicitados por consulta. Anteriormente se seleccionaban jugadores agrupados en tandas de 20, lo que hacía que se tuvieran que realizar menos consultas a cambio de que la aleatoriedad disminuyera. En esta ocasión, hemos decidido sacrificar algo de rendimiento por más aleatoriedad. De esta manera ahora realizamos una consulta por jugador.

Analizar el origen de los NumberFormatException

Igual que en las tareas de los anteriores sprints donde buscábamos entender por qué se originaban ciertas excepciones, hemos vuelto a utilizar Kibana para analizar este tipo de errores.

Tal y como vemos en la Figura 10.5 el error se origina en la clase `LineupServlet`. También como el error se debe a un intento de convertir un String a Long. Una vez hemos localizado el origen del error, vamos al código y vemos que estas líneas se corresponden con la conversión de los distintos IDs de los jugadores, que se pasan como parámetros en la petición, al tipo Long. Aquí también podemos ver como la única comprobación que se hace sobre estas variables es que no sean cadenas vacías, lo que deja abierta la posibilidad de que contengan caracteres no admitidos.

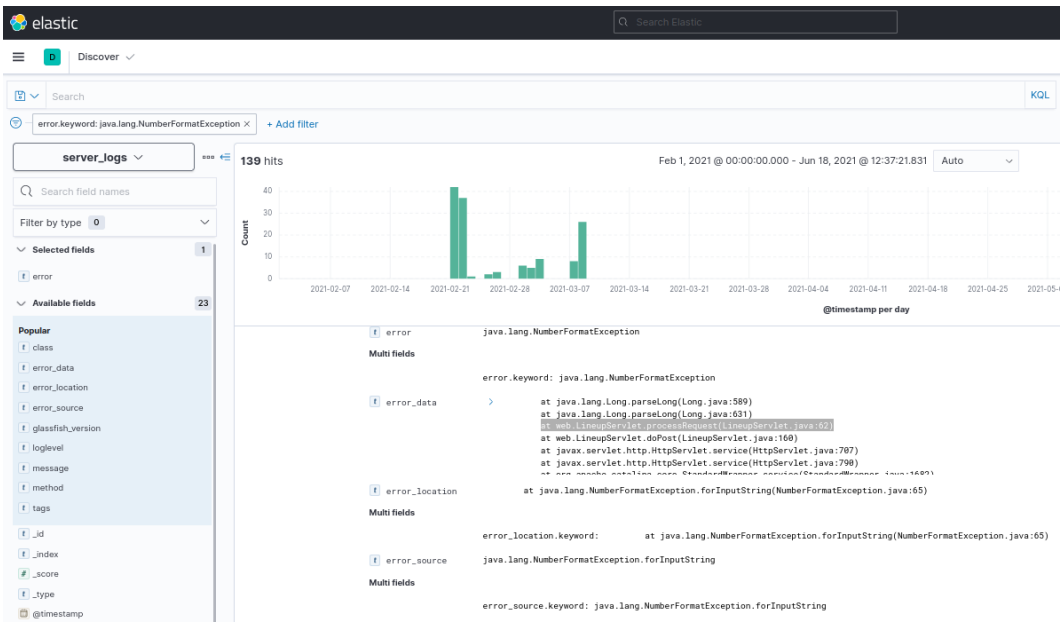

Figura 10.5: Origen de los NumberFormatException visto con Kibana.
Fuente: Elaboración propia

Para solucionar este problema se ha creado la función 10.7, que se encarga de comprobar que los IDs únicamente contienen números y que no están vacíos.

```
private boolean checkIdFormat(String id) {
    if (id != null) {
        return !id.isEmpty() && id.matches("^\\d+$");
    }
    return false;
}
```

Código Fuente 10.7: Función que valida el correcto formato de los IDs. Fuente: Elaboración propia

Añadir resultados de la última ronda a la página de inicio

Para realizar esta tarea se ha tenido que crear una nueva consulta (ver Código Fuente 10.8), en la cual se obtienen todas las partidas que ya ha jugado el usuario ordenadas descendentemente por ronda. Una vez tenemos todas las partidas, nos quedamos con la más reciente, es decir, la primera de la lista.

```
SELECT m FROM MatchGame m JOIN m.homeTeam h JOIN m.visitingTeam v
WHERE m.roundCalendar.isEnded = true AND
    (h.idTeam = :idTeam OR v.idTeam = :idTeam)
ORDER BY m.roundCalendar.round DESC
```

Código Fuente 10.8: Consulta para obtener las partidas jugadas por un usuario. Fuente: Elaboración propia

El paso final es añadir una tabla en la página de resultados donde se muestren las puntuaciones de los ajedrecistas. Para ello, hemos re-aprovechado la tabla de la página de resultados. El resultado final se puede ver en la Figura 10.6.

Las principales contribuciones de la parte 3 han sido:

- Adopción del desarrollo dirigido por datos.
- Mejora de la experiencia de usuario.
- Mejoras en el rendimiento de la aplicación.
- Solucionar errores hasta ahora desconocidos en la aplicación.

¡Bienvenido n ! Está en: Noticias > Ver Mensajes
Hora local: 15:00 (GMT + 01:00 Madrid).

Avisos

Se acaban de computar los movimientos del mercado por parte del sistema. admin, 19.06.2021 a la(s) 10:00
Se ha simulado la ronda 10. admin, 18.06.2021 a la(s) 20:45
Se acaban de computar los movimientos del mercado por parte del sistema. admin, 18.06.2021 a la(s) 13:00
Se ha simulado la ronda 9. admin, 17.06.2021 a la(s) 20:45
Se acaban de computar los movimientos del mercado por parte del sistema. admin, 17.06.2021 a la(s) 20:00
Se acaban de computar los movimientos del mercado por parte del sistema. admin, 16.06.2021 a la(s) 23:00

Resultados de tu última ronda
División: 7, Grupo: 6, Ronda: 10

Match	Color	Local	Elo	Resultado	Visitante	Elo	Color
	♔	Team		5.5-4.5	Team		♚
1	♔	M	1488	0.5-0.5	M	1467	♚
2	♚	N	1474	1-0	R	1424	♔
3	♔	G R	1447	0-1	P	1436	♚
4	♚	F	1446	0.5-0.5	P	1407	♔
5	♔	S	1428	1-0	C	1406	♚
6	♚	S	1311	0-1	S	1391	♔
7	♔	J	1334	0.5-0.5	C	1274	♚
8	♚	Z M	1282	0.5-0.5	B	1293	♔
9	♔	P	1223	0.5-0.5	P	1267	♚
10	♚	A C	1048	1-0	A R	1136	♔

Figura 10.6: Página de inicio tras añadir la tabla con el resultado de la última jornada. Fuente: Elaboración propia

11. Resultados de la gestión del proyecto

En este capítulo se detallan las desviaciones respecto a la planificación inicial y se realiza un análisis de la sostenibilidad del proyecto.

11.1. Resultados de la planificación

Dado que la planificación se llevó a cabo durante el módulo de GEP, la única fase que dispone de datos reales ha sido esta. El resto de fases se han planificado basándonos en estimaciones, lo cual ha llevado a ciertas desviaciones respecto a la planificación inicial. En la Tabla 11.1 se puede ver la diferencia entre las horas estimadas y las reales de cada fase. A continuación se detallan los motivos de las desviaciones para cada una de las fases.

Fase	Duración estimada (h)	Duración real (h)	Diferencia en horas
GEP	63	63	0
Visualización y análisis	113	82	-31
Creación de los modelos	168	91	-77
Implementación de mejoras	92	80	-12
Elaboración de la memoria	96	127	31
TOTAL	532	443	-89

Tabla 11.1: Resumen de las horas dedicadas a las distintas fases del proyecto. Fuente: Elaboración propia

Visualización y análisis

En esta fase uno de los principales cambios fue la decisión de alterar el orden de las distintas etapas y empezar por el análisis preliminar de los datos utilizando Elastic Stack. Pese a que esta etapa ha llevado algo más de tiempo del esperado, ha servido para entender mejor los datos de los que disponíamos. Lo cual ha agilizado de manera considerable el resto de etapas de esta fase. Otro de los aspectos que más a influido, ha sido que los datos no han requerido de tanto tratamiento como estaba previsto en un principio, lo que también ha servido para acortar el tiempo dedicado a esta fase.

Creación de los modelos

Dada la inexperiencia del autor en este campo, inicialmente pensamos que el esfuerzo requerido para esta fase iba a ser considerable, pero la familiarización con los modelos en grafo y su elaboración ha sido mucho más rápida de lo esperado.

Implementación de mejoras

Esta fase se retrasó una semana con respecto a la planificación original, debido a la necesidad de dedicar una semana ha terminar la documentación para el hito de seguimiento. Debido a esto, la carga de trabajo durante la tercera iteración se ha visto reducida con el fin de dedicar más tiempo a la finalización de esta memoria.

Elaboración de la memoria

La elaboración de la memoria ha sido la única fase del trabajo que ha superado la cantidad de horas estimadas inicialmente. Aún así, consideramos que la dedicación extra no ha supuesto ningún impedimento para el resto de tareas, sino que se ha utilizado el tiempo sobrante de las fases anteriores para pulir este documento.

Tal y como se observa en la Tabla 11.1, la dedicación real ha sido de 443 horas con una desviación total de -89 horas con respecto a la planificación inicial, es decir, menos horas que las estimadas en un primer momento.

11.2. Resultados de la gestión económica

A partir de las desviaciones temporales detalladas en la sección anterior, se ha determinado si el proyecto se encuentra dentro del presupuesto planteado al inicio.

En la Tabla 11.2 se resume el coste final del proyecto junto con las desviaciones de cada parte.

Tipo de coste	Coste estimado (€)	Coste real (€)	Desviaciones (€)
CPA	11.865,54	9.967,17	1.898,37
CG	332,95	240,93	92,02
Contingencia	1.203,20	0,00	1.203,20
Imprevistos	317,69	0,00	317,69
Subtotal	13.719,39	10.208,10	3.511,29
Impuestos	2.881,07	2.143,70	737,37
TOTAL	16.600,46	12.351,80	4.248,66

Tabla 11.2: Coste final del proyecto. Fuente: Elaboración propia

Como podemos ver, tenemos una desviación positiva sobre el coste del proyecto, lo cual indica que estamos dentro del presupuesto que se calculó al comienzo de este. Esta diferencia es debida a la reducción en horas totales dedicadas al proyecto, lo que nos ha permitido reducir el coste de personal en 1.898,37 €. Además de no haber necesitado los 1.520,89 € dedicados a contingencia e imprevistos. Para un desglose detallado de los costes de personal finales ver el Apéndice B.1.

11.3. Sostenibilidad

A continuación se realiza una reflexión sobre el impacto sostenible que tiene el trabajo.

11.3.1. Autoevaluación

Al empezar un proyecto de ingeniería se han de tener en cuenta una multitud de factores que pueden afectar al proyecto y otros muchos que pueden resultar afectados por éste. Uno de esos factores es la sostenibilidad del propio proyecto, algo que puede pasar desapercibido para los ingenieros más nóveles, e incluso a algunos con más experiencia.

Si bien es cierto que en el transcurso del grado hemos ido recibiendo, a través de diversas asignaturas, algunas pinceladas sobre sostenibilidad y el impacto que tiene el campo de la ingeniería informática sobre ésta, encuentro que todavía no dispongo de un conocimiento suficiente en la materia. Hasta el momento de la realización de este informe desconocía la existencia de las tres dimensiones en que se puede dividir y medir la sostenibilidad: económica, ambiental y social.

Con las que me encuentro más familiarizado son la ambiental y la social, que desde mi punto de vista son las que tienen más impacto mediático y resultan más visibles para todos aquellos que no estamos muy familiarizados con el tema. Además, estas dos dimensiones son en las que más se han centrado en la universidad, en mi opinión. Nos han explicado el impacto ambiental y social que tienen los residuos generados por el *hardware* obsoleto o la obtención de minerales como el coltán, utilizado para la fabricación de microprocesadores.

Por otro lado, la dimensión económica es la que menos conozco ya que no he tenido ocasión de participar en ningún proyecto donde se analizase, hasta ahora. Los únicos conocimientos que poseo son los básicos que he obtenido durante el grado, o incluso fuera de éste, como el hecho de administrar de manera correcta las horas de trabajo para cumplir un plazo de entrega o el conocer conceptos como: amortizar o rentabilidad.

11.3.2. Dimensión ambiental

Después de analizar el impacto ambiental de este proyecto, hemos concluido que no se puede mejorar más este aspecto. Esto se debe a que el trabajo se ha llevado a cabo con la cantidad mínima de recursos, utilizando únicamente un servidor externo y el ordenador de sobremesa utilizado para el desarrollo. Por lo tanto, consideramos que no se podría realizar este proyecto con menos recursos.

Además, durante la vida útil del proyecto se estará utilizando de manera continua el servidor donde se aloja la aplicación. Esto implica que no tenemos poder sobre el impacto que tendrá el consumo eléctrico de este servidor, ya que está gestionado por la empresa contratada.

De todas formas, el proyecto pretende facilitar las tareas de desarrollo, por lo que se espera que el tiempo dedicado a estas tareas se reduzca, disminuyendo así el consumo de los recursos necesarios para llevarlas a cabo.

Finalmente, no hay que olvidar que ahora mismo la aplicación no tiene una base de usuarios excesibamente grande. Pero en caso de que su popularidad fuera creciendo, podría requerir del uso de múltiples servidores, lo que empeoraría su huella ecológica.

11.3.3. Dimensión económica

Se ha realizado un informe detallado de los costes implicados en el proyecto, tanto de los recursos humanos como materiales, como se ha explicado en las secciones previas de este documento.

Como se ha mencionado en el apartado anterior, el proyecto se ha llevado a cabo con la mínima cantidad de recursos necesarios. Por lo tanto, el coste durante la vida útil de este viene dado principalmente por el precio del servicio de *hosting*, el cual se podría reducir buscando alternativas más económicas. También hay que tener en cuenta, que pese a que en el presupuesto se han tenido en cuenta los costes de personal, en la actualidad el trabajo se está llevando de manera voluntaria, por lo que nos ahorramos este coste.

11.3.4. Dimensión social

A nivel personal, este proyecto me ha permitido ponerme en los pies de un administrador de una aplicación web. Con ello, he descubierto las dificultades y la cantidad de trabajo que conlleva gestionar este tipo de aplicaciones. Además, también me ha permitido dar mis primeros pasos en el mundo de la investigación académica.

En cuanto al impacto social, este proyecto beneficiará principalmente a tres colectivos. Entre ellos los principales beneficiarios son los usuarios, ya que el objetivo final del trabajo es mejorar ChessLeague para ofrecer una mejor experiencia al usuario. El segundo son los propios desarrolladores de la aplicación, ya que gracias al análisis de los datos se facilitarán las tareas de desarrollo y definición de objetivos. Finalmente tenemos a la comunidad científica, ya que pretendemos publicar los resultados obtenidos del uso de los modelos creados en el proyecto. De esta manera esperamos que investigaciones futuras se puedan beneficiar de nuestro trabajo.

Para concluir, hay que tener en cuenta que este proyecto utiliza un juego en línea. Por lo tanto, no hay que olvidar que este tipo de software puede crear adicción a sus usuarios, sobre todo a los más jóvenes, como ya sucede con muchos otros videojuegos.

12. Conclusiones

Para acabar el proyecto, veremos las conclusiones del proyecto, como encaja este dentro de la especialidad y trabajo futuro que se puede realizar.

12.1. Conclusiones del proyecto

Con la creciente cantidad de datos que las compañías manipulan constantemente, hay una necesidad de buscar herramientas y metodologías para sacarle el mayor provecho a toda esta información. En este proyecto, hemos llevado a cabo un caso de estudio con una aplicación a pequeña escala para comprobar la utilidad de las redes bayesianas como herramienta de análisis.

Para ello hemos creado tres redes bayesianas para analizar *software logs*, que se han construido utilizando el feedback implícito que la aplicación. Con estas hemos sido capaces de ver como se comportan los errores que ocurren en la aplicación. También hemos visto como afectan diversas variables al rendimiento de la aplicación y como se comporta la aplicación a lo largo del tiempo. Con esto, hemos conseguido completar todos los sub-objetivos que nos definimos en la sección 3.2. Por lo que consideramos que los resultados son muy satisfactorios.

Viendo los buenos resultados que nos ha dado el uso de redes bayesianas en este proyecto, creemos que son una herramienta muy flexible y con mucho potencial que puede aportar un gran valor a la Ingeniería del Software.

Pese a la dificultad que conlleva la preparación previa a la creación de los modelos y a no existir, en la actualidad, un *workflow* estándar para integrarlos en el proceso de desarrollo, consideramos que los beneficios que nos aportan son superiores. Como hemos visto en este proyecto, no requerimos de grandes modelos para extraer información útil de ellos. Otro de sus puntos fuertes, es la posibilidad de manipular las probabilidades dentro del modelo para simular diferentes situaciones. Esto nos permite predecir como se comportará una aplicación bajo ciertas circunstancias, cosa que puede resultar de extrema utilidad en el campo de la Ingeniería del Software.

Otro aspecto que ha sido un éxito ha sido la adopción del desarrollo dirigido por datos. A lo largo de la última parte del proyecto, hemos visto como la utilización del feedback explícito, recopilado en la primera parte, y del feedback implícito de la segunda parte, nos han proporcionado la información necesaria para comprender los puntos débiles y fuertes de la aplicación. De esta manera, se ha facilitado la elaboración de una lista de tareas para solucionar problemas críticos e introducir mejoras que incrementarán el valor de la aplicación.

Finalmente, destacar que este trabajo servirá como punto de partida para la creación de un artículo científico que se propondrá para una conferencia (p.ej. PROFES 2021).

12.2. Integración de conocimientos

Con el objetivo de demostrar la integración de conocimientos de distintas disciplinas dentro del proyecto, a continuación se listan las principales asignaturas del grado que han servido como base para la realización de este trabajo.

Bases de Datos (BD)

La aplicación utiliza una base de datos MariaDB para almacenar toda la información generada. Por lo tanto, todas las consultas que se han optimizado dentro de la aplicación se han hecho utilizando los conocimientos obtenidos en la asignatura.

Conceptos para Bases de Datos Especializadas (CBDE)

En el proyecto se utiliza la herramienta Elasticsearch, la cual funciona de manera muy similar a una *document store* distribuida. Por lo tanto, para entender como funciona y poder gestionarla correctamente se necesitan los conocimientos obtenidos en esta materia.

Sistemas Operativos (SO)

El ecosistema del proyecto ha requerido de la creación de *scripts*, y de la configuración manual de Elastic Stack. Para ello se han utilizado los conocimientos sobre los sistemas basados en Linux explicados en la asignatura.

Matemáticas 1 (M1)

En este proyecto centramos la mayor parte de nuestra atención en el uso de modelos en grafo. Por lo tanto, para entenderlos correctamente se requieren los conceptos básicos sobre teoría de grafos impartidos en esta materia.

Probabilidad y Estadística (PE)

Para la creación de los modelos y su posterior análisis se requieren conocimientos de probabilidad y del lenguaje de programación R. Ambos enseñados en el transcurso de la asignatura.

Proyectos de Programación (PROP)

La aplicación ChessLeague esta desarrollada en Java, lenguaje que se explica en la asignatura.

Gestión de Proyectos de Software (GPS)

Pese a que esta materia se centraba en la gestión de un equipo de desarrollo, se han utilizado algunas de las metodologías de trabajo y herramientas explicadas. Como pueden ser el uso del *Kanban* y del *Product Backlog*.

Proyecto de Ingeniería del Software (PES)

Ha servido como base para aprender a gestionar un proyecto de pequeña envergadura. Esto ha facilitado en gran medida la realización de este trabajo.

Software Libre y Desarrollo Social (SLDS)

En esta asignatura se nos introduce a los editores de texto basados en LaTeX, lo cual ha servido en gran medida para la realización de esta memoria.

Aparte de los conocimientos obtenidos directamente del grado, también se han aplicado conceptos, adquiridos de manera autodidacta, propios del campo de la ciencia de datos. Como son el uso de Python para el tratamiento de los datos, la metodología de trabajo CRISP-DM o la utilización de diversas librerías de R para la creación de redes bayesianas.

12.3. Justificación de las competencias

En el siguiente apartado se describe como se han cumplido las competencias técnicas asociadas al proyecto.

CES1.1: Desarrollar, mantener y evaluar sistemas y servicios software complejos y/o críticos. [En profundidad]

El objetivo principal del proyecto ha sido desarrollar un conjunto de soluciones para mejorar una aplicación web, mantener el código ya existente a medida que el proyecto ha ido avanzando. Además se han tenido que evaluar las diferentes alternativas disponibles para cada una de las fases del proyecto. Además, no solo se han tenido que implementar nuevas funcionalidades, sino que también ha habido que mantener todo el ecosistema de aplicaciones utilizadas para la obtención de los datos. Dado que el proyecto se ha desarrollado con éxito, consideramos que esta competencia se ha asumido en profundidad.

CES1.2: Dar solución a problemas de integración en función de las estrategias, de los estándares y de las tecnologías disponibles. [Un poco]

Uno de los procesos más importantes para el correcto funcionamiento del proyecto es la integración de las distintas fuentes de datos. Para ello se ha tenido que configurar la integración de las distintas herramientas utilizadas para la ingesta de

datos. Además, se han elaborado varios scripts en Python para la correcta integración de los datos. Por lo tanto, consideramos que la competencia se ha asumido de manera satisfactoria.

CES1.6: Administrar bases de datos (CIS4.3). [Bastante]

La aplicación utiliza una base de datos MariaDB, por lo que ha habido que gestionar la realización de copias de seguridad. Por otro lado, se han tenido que estudiar las diferentes consultas SQL que se realizan dentro de la aplicación, de manera que estas estén optimizadas.

CES1.7: Controlar la calidad y diseñar pruebas en la producción de software. [En profundidad]

Uno de los principales objetivos dentro del proyecto era la creación de varios modelos para asesorar la calidad del software. Como resultado, hemos generado tres modelos distintos, los cuales nos han permitido detectar problemas de calidad en varios aspectos de la aplicación, ya fueran errores o pérdidas de rendimiento. Además, asegurar la calidad del código, se han elaborado un conjunto de tests unitarios con el objetivo de verificar el correcto funcionamiento de la aplicación. Por lo tanto, consideramos que se ha alcanzado esta competencia de manera satisfactoria.

CES1.9: Demostrar comprensión en la gestión y gobierno de los sistemas software. [Un poco]

El proyecto está formado por un ecosistema de aplicaciones comprendido por ChessLeague, el servicio de hosting ANW y Elastic Stack. Por lo tanto, a lo largo del proyecto se ha tenido que gestionar, configurar y mantener cada una de las aplicaciones. De este modo, consideramos que se ha cumplido satisfactoriamente con esta competencia.

CES2.1: Definir y gestionar los requisitos de un sistema software. [Bastante]

Durante la segunda fase del proyecto, que se corresponde con la implementación de mejoras en ChessLeague, se han tenido que analizar los resultados del estudio de los datos y, posteriormente, definir los requisitos para implementar dichas mejoras. Adicionalmente, se ha tenido que gestionar el Product Backlog para cada una de las iteraciones, identificando a que requisitos darles prioridad.

CES2.2: Diseñar soluciones apropiadas en uno o más dominios de aplicación, utilizando métodos de ingeniería del software que integren aspectos éticos, sociales, legales y económicos. [Un poco]

Este proyecto se ha desarrollado siguiendo los estándares éticos y sociales dentro del ámbito de la ingeniería del software y la ciencia de datos. Además, se han seguido las pautas establecidas por las vigentes leyes de protección de datos a nivel nacional. En cuanto al aspecto económico, se ha procurado que el presupuesto del

trabajo se ajuste a las necesidades del proyecto sin incurrir en gastos innecesarios. Por lo tanto, determinamos que esta competencia se ha alcanzado de manera satisfactoria.

12.4. Trabajo futuro

Pese a que el proyecto ha terminado de manera satisfactoria y se han alcanzado los principales objetivos de este, hay ciertos puntos que con más tiempo se podrían haber implementado/mejorado.

Actualización de tecnologías en ChessLeague

Actualmente ChessLeague está implementada sobre GlassFish 4.1.2, que es una versión bastante antigua. Además de utilizar arquitecturas y frameworks que empiezan a quedar obsoletos. Por esto, uno de los principales puntos a mejorar sería la actualización de toda la aplicación, cosa que facilitaría su desarrollo en un futuro.

Integración de los datos de uso y del servidor

Debido a la manera en que el servidor registra su actividad ahora mismo no hay forma posible de trazar una relación directa entre una entrada en estos registros con una entrada en los registros de uso. Esto dificulta la integración de los dos conjuntos de datos. Por ello, habría que implementar un *logger* para registrar que usuario está provocando los errores, o añadir algún mecanismo que nos permita crear esta relación.

Mejora del modelo temporal

Actualmente el modelo temporal cumple su función y nos ha dado buenos resultados. Aún así, con más tiempo se podría elaborar un modelo que fuera capaz de distinguir entre sesiones, ya que actualmente el modelo se genera considerando que todos los registros pertenecen a una misma sesión debido a la heterogeneidad de estos.

12.5. Conclusiones personales

Como cierre para este proyecto, quisiera terminar de una forma algo más informal y explicar que ha significado este trabajo para mí.

Este proyecto ha sido una gran oportunidad para trabajar en un ambiente de investigación, algo que es totalmente nuevo para mí. Esto me ha proporcionado la oportunidad de contribuir al mundo de la Ingeniería del Software, cosa que ya me resulta asombrosa. Otra de las cosas que me ha ofrecido este proyecto a sido la oportunidad de asistir a la conferencia MSR, donde he podido presenciar gran cantidad de nuevas investigaciones e ideas que han despertado mi interés. Además,

esta experiencia me ha hecho reflexionar sobre el camino que quiero tomar en el futuro, entrando en juego la posibilidad de dedicarme a la investigación.

Este trabajo también me ha servido para introducirme en el campo de la ciencia de datos. Esto ha supuesto un reto para mí, dada mi inexperiencia en esta área. Aun así, el tener que investigar formas de manipular los datos y como generar nuevos ha sido un proceso muy entretenido e instructivo. Esto ha hecho que despierte un interés por este campo, que espero seguir desarrollando con el tiempo.

Por último, quisiera agradecer la oportunidad que se me ha dado de contribuir a la aplicación ChessLeague.

Bibliografía

[1] *How Google uses information from sites or apps that use our services – Privacy & Terms – Google*, en-US. dirección: https://policies.google.com/technologies/partner-sites?hl=en-US (visitado 21-06-2021).

[2] S. Martínez-Fernández, *Chessleague [Manuscrito] : juego en línea de simulación de ligas de ajedrez/ Silverio Juan Martínez Fernández*, spa, 2010. dirección: https://indaga.ual.es/permalink/34CBUA_UAL/1fi96lk/alma991001193459704991.

[3] D. Zhang, S. Han, Y. Dang, J.-G. Lou, H. Zhang y T. Xie, «Software Analytics in Practice», *IEEE Software*, vol. 30, n.º 5, págs. 30-37, sep. de 2013. DOI: 10.1109/ms.2013.94. dirección: https://ieeexplore.ieee.org/abstract/document/6559957.

[4] R. P. L. Buse y T. Zimmermann, «Information needs for software development analytics», en *2012 34th International Conference on Software Engineering (ICSE)*, ISSN: 1558-1225, jun. de 2012, págs. 987-996. DOI: 10.1109/ICSE.2012.6227122. dirección: https://ieeexplore.ieee.org/abstract/document/6227122.

[5] *Mining Software Repositories*, en-US. dirección: http://www.msrconf.org/ (visitado 16-03-2021).

[6] P. Kruchten, R. L. Nord e I. Ozkaya, «Technical Debt: From Metaphor to Theory and Practice», *IEEE Software*, vol. 29, n.º 6, págs. 18-21, nov. de 2012, ISSN: 1937-4194. DOI: 10.1109/MS.2012.167.

[7] X. Franch, «Data-Driven Requirements Engineering: A Guided Tour», en *Evaluation of Novel Approaches to Software Engineering*, R. Ali, H. Kaindl y L. A. Maciaszek, eds., Cham: Springer International Publishing, 2021, págs. 83-105, ISBN: 978-3-030-70006-5.

[8] B. Holländer, *Introduction to Probabilistic Graphical Models*, en, feb. de 2020. dirección: https://towardsdatascience.com/introduction-to-probabilistic-graphical-models-b8e0bf459812 (visitado 16-03-2021).

[9] V. Chugh, *Introduction to Probabilistic Graphical Models*, en, ago. de 2020. dirección: https://towardsdatascience.com/introduction-to-probabilistic-graphical-models-7d2c0b4bef19 (visitado 16-03-2021).

[10] *Graphical model*, en, Page Version ID: 1006029636, feb. de 2021. dirección: https://en.wikipedia.org/w/index.php?title=Graphical_model&oldid=1006029636 (visitado 16-03-2021).

[11] *Bayesian network*, en, Page Version ID: 1011105360, mar. de 2021. dirección: https://en.wikipedia.org/w/index.php?title=Bayesian_network&oldid=1011105360 (visitado 16-03-2021).

[12] R. Nagarajan, M. Scutari y S. Lèbre, *Bayesian Networks in R: with Applications in Systems Biology*, en, ép. Use R! New York: Springer-Verlag, 2013, ISBN: 9781461464457. DOI: 10.1007/978-1-4614-6446-4. dirección: https://www.springer.com/gp/book/9781461464457 (visitado 27-05-2021).

[13] K. Biesialska, X. Franch y V. Muntés-Mulero, «Big Data analytics in Agile software development: A systematic mapping study», en, *Information and Software Technology*, vol. 132, pág. 106 448, abr. de 2021, ISSN: 0950-5849. DOI: 10.1016/j.infsof.2020.106448. dirección: https://www.sciencedirect.com/science/article/pii/S0950584920301981 (visitado 25-02-2021).

[14] D. Zhang y T. Xie, «Software analytics and its application in practice», en, en T. Menzies, L. Williams y T. Zimmermann, eds., Boston: Morgan Kaufmann, ene. de 2016, págs. 7-11, ISBN: 9780128042069. DOI: 10.1016/B978-0-12-804206-9.00002-7. dirección: https://www.sciencedirect.com/science/article/pii/B9780128042069000027 (visitado 24-02-2021).

[15] P. Pospieszny, B. Czarnacka-Chrobot y A. Kobylinski, «An effective approach for software project effort and duration estimation with machine learning algorithms», en, *Journal of Systems and Software*, vol. 137, págs. 184-196, mar. de 2018, ISSN: 0164-1212. DOI: 10.1016/j.jss.2017.11.066. dirección: https://www.sciencedirect.com/science/article/pii/S0164121217302947 (visitado 16-03-2021).

[16] J. Cândido, M. Aniche y A. v. Deursen, «Log-based software monitoring: a systematic mapping study», en, *PeerJ Computer Science*, vol. 7, e489, mayo de 2021, ISSN: 2376-5992. DOI: 10.7717/peerj-cs.489. dirección: https://peerj.com/articles/cs-489 (visitado 19-05-2021).

[17] G. Balogh, G. Antal, Á. Beszédes, L. Vidács, T. Gyimóthy y Á. Z. Végh, «Identifying wasted effort in the field via developer interaction data», en *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, sep. de 2015, págs. 391-400. DOI: 10.1109/ICSM.2015.7332490.

[18] M. Conoscenti, V. Besner, A. Vetrò y D. M. Fernández, «Combining data analytics and developers feedback for identifying reasons of inaccurate estimations in agile software development», en, *Journal of Systems and Software*, vol. 156, págs. 126-135, oct. de 2019, ISSN: 0164-1212. DOI: 10.1016/j.jss.2019.06.075. dirección: https://www.sciencedirect.com/science/article/pii/S0164121219301372 (visitado 16-03-2021).

[19] S. Martínez-Fernández, A. M. Vollmer, A. Jedlitschka y col., «Continuously Assessing and Improving Software Quality With Software Analytics Tools: A Case Study», *IEEE Access*, vol. 7, págs. 68 219-68 239, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2917403. dirección: https://ieeexplore.ieee.org/document/8717997.

[20] I. Rodrigues, *CRISP-DM methodology leader in data mining and big data*, en, feb. de 2020. dirección: https://towardsdatascience.com/crisp-dm-methodology-leader-in-data-mining-and-big-data-467efd3d3781 (visitado 28-02-2021).

[21] *Principios del Manifesto Ágil*, es. dirección: https://agilemanifesto.org/iso/es/principles.html (visitado 28-02-2021).

[22] *What is Scrum?*, en. dirección: https://www.scrum.org/resources/what-is-scrum (visitado 28-02-2021).

[23] *The Scrum Primer*, en. dirección: https://www.infoq.com/minibooks/Scrum_Primer/ (visitado 18-06-2021).

[24] Atlassian, *Jira | Issue & Project Tracking Software*, en. dirección: https://www.atlassian.com/software/jira (visitado 28-02-2021).

[25] *Git*, en. dirección: https://git-scm.com/ (visitado 28-02-2021).

[26] *Build software better, together*, en. dirección: https://github.com (visitado 28-02-2021).

[27] *Herramientas y servicios de análisis para su empresa - Google Analytics*, es. dirección: https://marketingplatform.google.com/intl/es/about/analytics/ (visitado 02-03-2021).

[28] *Python*, en. dirección: https://www.python.org/ (visitado 28-02-2021).

[29] *R: The R Project for Statistical Computing*, en. dirección: https://www.r-project.org/ (visitado 28-02-2021).

[30] *R (programming language)*, en, Page Version ID: 1006068491, feb. de 2021. dirección: https://en.wikipedia.org/w/index.php?title=R_(programming_language)&oldid=1006068491 (visitado 28-02-2021).

[31] *Logstash: Recopila, parsea y transforma logs*, es-mx. dirección: https://www.elastic.co/es/logstash (visitado 15-03-2021).

[32] *Hugin Expert | Hugin Expert*, en-US. dirección: https://www.hugin.com/ (visitado 23-05-2021).

[33] *Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo, de 27 de abril de 2016, relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos y por el que se deroga la Directiva 95/46/CE (Reglamento general de protección de datos)*, abr. de 2016. dirección: https://www.boe.es/doue/2016/119/L00001-00088.pdf.

[34] *Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales*. dirección: https://www.boe.es/buscar/pdf/2018/BOE-A-2018-16673-consolidado.pdf.

[35] *Horas y jornada anual máxima de trabajo efectivo, según Estatuto de los Trabajadores*, es. dirección: https://www.elplural.com/economia/asesoria-laboral-ugt/jornada-maxima-de-trabajo-efectivo-y-promedio-computo-anual_120611102 (visitado 13-03-2021).

[36] *El coste de un trabajador para la empresa [+fórmula] - Factorial*, es-ES, mar. de 2020. dirección: https://factorialhr.es/blog/coste-empresa-trabajador/ (visitado 13-03-2021).

[37] *Power Management Statistics: Information Technology - Northwestern University*, en. dirección: https://www.it.northwestern.edu/hardware/eco/stats.html (visitado 14-03-2021).

[38] *Precio del kWh de Iberdrola*, es. dirección: https://tarifasgasluz.com/comercializadoras/iberdrola/precio-kwh (visitado 14-03-2021).

[39] *DIGI | Fibra Óptica Simétrica | Con DIGI #AhorroSeguro*, es. dirección: https://www.digimobil.es/ (visitado 14-03-2021).

[40] «Las herramientas Elastic - documentación de ManualKibanaOCDS - latest», es, dirección: https://manualkibanaocds.readthedocs.io/es/latest/C2/Seccion1.html (visitado 25-04-2021).

A. Apéndice 1: Código fuente

A.1. Configuración de Logback

```xml
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <property name="LOGS_HOME" value="path-to-your-logs-home" />

  <appender name="FILE" class="ch.qos.logback.core.FileAppender">
      <file>${LOGS_HOME}/log-file-name.log</file>

      <encoder>
          <pattern>%d{YYYY-MM-dd HH:mm:ss.SSS},%msg%n</pattern>
      </encoder>
  </appender>

  <root level="debug">
    <appender-ref ref="FILE" />
  </root>
</configuration>
```

A.2. *Logger* personalizado

```java
package chessleague.util;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
 *
 * @author santiago
 */
public class ActionLogger {
    public static final String LOAD = "load page";
    public static final String FINISH_LOAD = "finish load";
    public static final String REDIRECT_TO = "redirect to ";
    public static final String EXIT = "exit app";
```

```
15
16      private final Logger logger =
        ↪ LoggerFactory.getLogger("ActionLogger");
17
18      public ActionLogger() {
19      }
20
21      public void traceLoadPage(String language, String userId,
        ↪ String sessionId, String currentLocation) {
22          traceAction(language, userId, sessionId, currentLocation,
            ↪ LOAD);
23      }
24
25      public void traceFinishLoadPage(String language, String userId,
        ↪ String sessionId, String currentLocation) {
26          traceAction(language, userId, sessionId, currentLocation,
            ↪ FINISH_LOAD);
27      }
28
29      public void traceExitApp(String language, String userId, String
        ↪ sessionId, String currentLocation) {
30          traceAction(language, userId, sessionId, currentLocation,
            ↪ EXIT);
31      }
32
33      public void traceRedirectTo(String language, String userId,
        ↪ String sessionId, String currentLocation, String
        ↪ destination) {
34          traceAction(language, userId, sessionId, currentLocation,
            ↪ REDIRECT_TO + destination + " from " +
            ↪ currentLocation);
35      }
36
37      public void traceAction(String language, String userId, String
        ↪ sessionId, String currentLocation, String action) {
38          logger.info("{},{},{},{},{}", language, userId, sessionId,
            ↪ currentLocation, action);
39      }
40
41  }
```

A.3. Configuración de Logstash

Pipeline de los datos de uso

```
input {
    file {
        path => "pathtologs/usersActivity.log"
        start_position => "beginning"
    }
}

filter {
    csv {
        id => "chessleague_csv_filter"
        separator => ","
        columns => ["logdate", "language", "username", "sesion_id",
        ↪  "current_page", "action"]
        remove_field => ["message", "host", "path", "column7"]
    }

    date {
        match => [ "logdate", "YYYY-MM-dd HH:mm:ss.SSS" ]
    }
}

output {
    csv {
        id => "usage-data-csv-output"
            path => "outputpath/usage-data.csv"
            fields => ["@timestamp", "language", "username",
            ↪  "sesion_id", "current_page", "action"]
    }

    elasticsearch {
        hosts => "localhost:9200"
        index => "user_logs"
    }
}
```

Pipeline de los datos del servidor

```
input {
    file {
        path => "pathtologs/server.log*"
        # Merge lines not starting with [date] up to the previous
        ↪  line.
```

```
5           codec => multiline {
6               pattern => "^\[%{TIMESTAMP_ISO8601}\]"
7               negate => true
8               what => "previous"
9           }
10          start_position => "beginning"
11      }
12  }
13
14  filter {
15      grok {
16          match => { "message" =>[
17              "^\[%{TIMESTAMP_ISO8601:logdate}\] \[%{DATA:version}\]
                ↪  \[%{LOGLEVEL:loglevel}\] \[%{DATA:data}\]
                ↪  \[%{DATA:log_class}\] \[tid:
                ↪  _ThreadID=%{NUMBER:thread_id:int}
                ↪  _ThreadName=%{DATA:thread_name}\] \[timeMillis:
                ↪  %{NUMBER:timeMillis:int}\] \[levelValue:
                ↪  %{NUMBER:levelValue:int}\] \[\[\n
                ↪  %{DATA:message}\n.*:
                ↪  %{JAVACLASS:error}\n%{JAVASTACKTRACEPART:
                ↪  error_location}\n%{DATA:error_data}\]\]$",
18              "^\[%{TIMESTAMP_ISO8601:logdate}\] \[%{DATA:version}\]
                ↪  \[%{LOGLEVEL:loglevel}\] \[%{DATA:data}\]
                ↪  \[%{DATA:log_class}\] \[tid:
                ↪  _ThreadID=%{NUMBER:thread_id:int}
                ↪  _ThreadName=%{DATA:thread_name}\] \[timeMillis:
                ↪  %{NUMBER:timeMillis:int}\] \[levelValue:
                ↪  %{NUMBER:levelValue:int}\] \[\[\n
                ↪  %{DATA:message}\n%{JAVACLASS:
                ↪  error}\n%{JAVASTACKTRACEPART:
                ↪  error_location}\n%{DATA:error_data}\]\]$",
19              "^\[%{TIMESTAMP_ISO8601:logdate}\] \[%{DATA:version}\]
                ↪  \[%{LOGLEVEL:loglevel}\] \[%{DATA:data}\]
                ↪  \[%{DATA:log_class}\] \[tid:
                ↪  _ThreadID=%{NUMBER:thread_id:int}
                ↪  _ThreadName=%{DATA:thread_name}\] \[timeMillis:
                ↪  %{NUMBER:timeMillis:int}\] \[levelValue:
                ↪  %{NUMBER:levelValue:int}\] \[\[\n
                ↪  %{DATA:message}\n%{JAVACLASS:
                ↪  error}.*\"\n%{JAVASTACKTRACEPART:
                ↪  error_location}\n%{DATA:error_data}\]\]$",
```

```
20              "^\[%{TIMESTAMP_ISO8601:logdate}\] \[%{DATA:version}\]
                ↪  \[%{LOGLEVEL:loglevel}\] \[%{DATA:data}\]
                ↪  \[%{DATA:log_class}\] \[tid:
                ↪  _ThreadID=%{NUMBER:thread_id:int}
                ↪  _ThreadName=%{DATA:thread_name}\] \[timeMillis:
                ↪  %{NUMBER:timeMillis:int}\] \[levelValue:
                ↪  %{NUMBER:levelValue:int}\] \[\[\n
                ↪  %{GREEDYDATA:message}\]\]$"
21          ] }
22          overwrite => ["message"]
23      }
24
25      if "version" != "" {
26          mutate {
27              split => { "version" => " " }
28              add_field => { "glassfish_version" => "%{[version][1]}"
                ↪  }
29              remove_field => [ "version", "host", "path" ]
30          }
31      }
32
33      if "" in [class] {
34          mutate {
35              add_field => { "error_source" => "%{class}.%{method}" }
36          }
37      }
38
39      date {
40          match => [ "logdate", "ISO8601" ]
41      }
42  }
43
44  output {
45      elasticsearch {
46          hosts => "localhost:9200"
47          index => "server_logs-%{+YYYY-MM}"
48      }
49
50      csv {
51          id => "output-server-csv"
52          path => "outputpath/glassfish-server.csv"
53          fields => ["@timestamp", "glassfish_version", "loglevel",
                ↪  "log_class", "thread_id", "thread_name", "timeMillis",
                ↪  "levelValue", "message", "error", "class", "method"]
54      }
55  }
```

Pipeline de los datos de acceso

```
input {
    file {
        path => "pathtologs/server_access_log.*.txt"
        start_position => "beginning"
   }
}

filter {
    csv {
        id => "glassfish-access-csv-filter"
        separator => " "
        columns => ["client_name", "auth_user_name", "datetime",
        ↪  "request", "status", "response_length"]
        convert => {
            "response_length" => integer
        }
    }

    mutate {
        remove_field => [ "message", "host", "path", "column7" ]
        split => { "request" => " " }
        add_field => { "request_op" => "%{[request][0]}" }
        add_field => { "resource" => "%{[request][1]}" }
        add_field => { "http_version" => "%{[request][2]}" }
        remove_field => [ "request" ]
    }

    date {
        match => ["datetime", "dd/MMM/YYYY:HH:mm:ss Z"]
    }
}

output {
    csv {
        id => "output-access-csv"
        path => "outputpath/glassfish-access.csv"
        fields => ["client_name", "auth_user_name", "@timestamp",
        ↪  "request_op", "resource", "http_version", "status",
        ↪  "response_length"]
    }

    elasticsearch {
        hosts => "localhost:9200"
        index => "access_logs"
```

```
    }
}
```

Pipeline de los datos de google analytics

```
input {
    file {
        path => "pathtologs/ga_report.csv"
        start_position => "beginning"
    }
}

filter {
    csv {
        id => "ga_csv_filter"
        separator => ", "
        columns => ["User ID", "Session ID", "Timestamp", "User
        ↪  Type", "Country", "Source", "Browser", "Browser
        ↪  Version", "Device Category", "Page Path", "Previous
        ↪  Page Path", "Days Since Last Session", "Page Views",
        ↪  "Time On Page", "Page Load Time"]
        skip_header => true

        convert => {
            "Days Since Last Session" => integer
            "Page Views" => integer
            "Time On Page" => float
            "Page Load Time" => integer
        }
        remove_field => ["message", "host", "path"]
    }

    date {
        match => [ "Timestamp", "YYYY-MM-dd'T'HH:mm:ss.SSSZZ" ]
    }
}

output {
    csv {
        id => "ga-csv-output"
        path => "outputpath/google-analytics.csv"
        fields => ["User ID", "Session ID", "@timestamp", "User
        ↪  Type", "Country", "Source", "Browser", "Browser
        ↪  Version", "Device Category", "Page Path", "Previous
        ↪  Page Path", "Days Since Last Session", "Page Views",
        ↪  "Time On Page", "Page Load Time"]
```

```
34      }
35
36      elasticsearch {
37          hosts => "localhost:9200"
38          index => "ga_logs"
39      }
40  }
```

B. Apéndice 2: Costes del proyecto

B.1. Desglose del coste de personal real

ID	Duración real (h)	Horas/Rol reales (h)						Coste estimado (€)	Coste real (€)	Desvío total (€)
		JP	PO	BA	DS	D	T			
GP1	25	25	0	0	0	0	0	650,48	650,48	0,00
GP2	9	9	0	0	0	0	0	234,17	234,17	0,00
GP3	10	10	0	0	0	0	0	260,19	260,19	0,00
GP4	19	19	0	0	0	0	0	494,37	494,37	0,00
R	18	18	18	0	0	0	0	1.010,02	1.010,02	0,00
VA1	13	0	0	0	0	13	0	282,05	203,70	78,35
VA2	9	0	0	4	5	0	0	167,37	167,37	0,00
VA3	3	0	0	0	3	0	0	304,06	60,81	243,24
VA4	8	0	0	0	8	0	0	304,06	162,16	141,89
VA5	15	0	0	0	15	0	0	405,41	304,06	101,35
VA6	6	0	0	0	6	0	0	202,70	121,62	81,08
VA7	6	0	0	0	6	0	0	121,62	121,62	0,00
VA8	22	0	0	0	22	0	0	405,41	445,95	-40,54
MD1	26	0	0	0	26	0	0	608,11	527,03	81,08
MD2	13	0	0	0	13	0	0	364,87	263,51	101,35
MD3	30	0	0	0	30	0	0	1.216,22	608,1	608,111
MD4	22	0	0	0	22	0	0	1.216,22	445,9	770,275
IM1	3	3	0	0	0	0	0	156,12	78,06	78,06
IM2	3	3	0	0	0	0	0	156,12	78,06	78,06
IM3	74	0	0	0	0	54	20	1.342,50	1.248	94,02,49
EM1	26	0	0	10	16	0	0	570,46	489,38	81,08
EM2	35	0	0	0	35	0	0	486,49	709,46	-222,97
EM3	42	0	0	0	0	42	0	282,05	658,12	-376,07
EM4	24	24	0	0	0	0	0	624,47	624,47	0,00
TOTAL	461	111	18	14	207	109	20	11.865,54	9.967,17	1.898,37

Tabla B.1: Coste real para cada tarea. Fuente: Elaboración propia