Facultad de Ciencias

MAADLE Y PREZ: PROCESADO DEL LOG DE EVENTOS DE MOODLE PARA LA GENERACIÓN DE INFORMES DE LA ACTIVIDAD DE LOS USUARIOS

(Moodle's Events Log Processing for the Generation of Users' Activity Reports)

Trabajo de Fin de Grado
para acceder al

GRADO EN INGENIERÍA INFORMÁTICA

Autor: Brian Sal Sarria

Director: Pablo Sánchez Barreiro

Co-Director: Alfonso de la Vega Ruiz

Junio – 2020

Contenido

Lista de Figuras

Lista de Listados

Agradecimientos

En primer lugar, me gustaría expresar mi más sincero agradecimiento a Alfonso de la Vega y a Pablo Sánchez, los directores del trabajo, por permitirme trabajar con ellos y por las incansables horas de reuniones y correos electrónicos intercambiados. Su buen hacer ha sido fundamental para el prosperar del proyecto.

En segundo lugar, también me gustaría agradecer a Ruth Villalón su participación y buena voluntad. Su ayuda desinteresada ha sido de gran utilidad en todo momento.

Por último, agradecer desde aquí a mi familia y amigos por aguantarme y apoyarme durante el proceso, sobre todo por aguantarme. También disculparme por las horas que he dedicado a este trabajo en lugar de a ellos. Parte del mérito es suyo.

Resumen

En la actualidad, gran parte de la actividad desarrollada en casi cualquier sector está apoyada en sistemas informáticos. Generalmente, estos sistemas registran multitud de datos sobre cómo se interactúa con ellos. Si estos datos se procesan y analizan adecuadamente pueden llegar a proporcionar información muy útil, que sirva para mejorar el propio sistema que los recogió o, incluso, el sector al que sirven.

El sector de la educación es un ejemplo de lo comentado. Actualmente, dentro del ámbito educativo es muy común la utilización de *LMSs* (*Learning Management Systems*) como *Moodle*, *Chamilo* o *Evolcampus*, como apoyo en los procesos de enseñanza-aprendizaje, permitiendo la publicación de material docente, su consulta o la creación de actividades con sus respectivas entregas, entre otras muchas posibilidades.

Estos *LMSs* guardan datos sobre el uso que se hace de ellos en un fichero de log. De esta forma, el docente con acceso a tal fichero puede analizarlo para averiguar, por ejemplo, qué recurso ha sido el más visitado, qué alumno ha interactuado más con el curso, o en qué momentos se usa con más frecuencia la plataforma, entre otras muchas opciones.

Por desgracia, estos ficheros log suelen contener una cantidad ingente de datos en un formato tan pobre que hace muy tediosa, y en algunos casos casi directamente imposible, la extracción de información útil de ellos de forma simple y directa. Esto provoca que los datos almacenados en tales ficheros log normalmente se infrautilicen o no se utilicen en absoluto.

Con el fin de solucionar este problema, este Trabajo de Fin de Grado se plantea dos objetivos. El primero de ellos consiste en el desarrollo de una librería, denominada *Maadle*, que permita a los programadores acceder a los datos del log de *Moodle* de una forma cómoda y sencilla. El segundo consiste en desarrollar una aplicación, denominada *Prez*, que genere informes con distintos indicadores sobre la utilización de esta plataforma de enseñanza. Se ha seleccionado *Moodle* como *LMS* por ser esta la plataforma oficial de la Universidad de Cantabria.

Palabras Clave

Enseñanza virtual, Inteligencia de Negocio, Moodle, Procesado de Datos

Abstract

Currently, much of the activity carried out in almost any sector is supported by computer systems. Generally, these systems record a huge amount of data on how users interact with them. If these data are processed and analyzed, they can end up providing very useful information, that may serve to improve the system that retrieves them or, even, the sector they serve.

The educational sector is an example of what has been said. Currently, within the educational field, the use of *LMSs* (*Learning Management Systems*) such as *Moodle*, *Chamilo* or *Evolcampus* is very common as support in teaching-learning processes, publication of teaching material, consultation or creation of activities with their respective deliveries, among many other possibilities.

These *LMSs* save data about the use that is made of them in a log file. In this way, the teacher with access to such file can analyze it to find, for example, which resource has been the most visited, which student has interacted the most with the course, or when the platform is most frequently used, among many other options.

Unfortunately, these logs often contain a huge amount of data in such a poor format that makes it very tedious, and in some cases almost directly impossible, to extract useful information from them simply and directly. This causes the data stored in those log files to be routinely underused or not used at all.

In order to solve this problem, this Final Degree Project has two objectives. The first one of them is to develop a library, *Maadle*, that may allow developers to access *Moodle* log data in a comfortable and simple way. The second one is to develop an application, *Prez*, which generates reports with different indicators on the use of this teaching platform. *Moodle* has been selected as the *LMS* to work with, as it is the official platform of the University of Cantabria.

Keywords

Busines Intelligence, data processing, Moodle, virtual learning

1. Introducción, Objetivos y Metodología

1.1. Introducción

En la actualidad, gran parte de las actividades desarrolladas en casi cualquier sector de la sociedad están sustentadas por sistemas informáticos. Los *ERP* (*Enterprise Resource Planning)* que hay en casi todas las empresas no son más que sistemas informáticos desarrollados con el fin de mejorar la gestión empresarial (Demian Abrego Almazán [4]), (Dmitrij Lipaj [5]). Otro ejemplo de esto son los *CRM* (*Client Relationship Management*), sistemas informáticos orientados a gestionar las relaciones con los clientes de una empresa. El sector de la enseñanza no es ajeno a lo comentado. Las plataformas *LMS* (*Learning Management System*) (Long [9]) como *Moodle*[1], *Chamilo*[2] o *Evolcampus*[3], son sistemas informáticos que se encuentran presentes hoy en día en muchos centros de enseñanza. Según un estudio realizado en 2009 por la Universidad de Murcia, con el objetivo de realizar un análisis comparativo del uso de campus virtuales en las universidades españolas, 62 de 74 universidades encuestadas utilizaban *Moodle* como plataforma de enseñanza (Espinosa [7]). *Moodle* también es la plataforma utilizada por la Universidad de Cantabria.

Moodle, entre muchas otras cosas, pone a disposición de los docentes un registro con las acciones de los usuarios con los distintos cursos de la plataforma. Estas acciones van desde la visualicación de las calificaciones hasta la selección de un fichero de texto o la apertura de una carpeta. Este registro es lo que se conoce como *fichero log*.

El análisis de los datos de estos ficheros log podría llegar a ser muy útil, ya que este podría permitir conocer, por ejemplo, cuáles son los recursos más utilizados de un curso de *Moodle*, quiénes son los usuarios que más han participado o en qué momentos del curso se ha participado más, entre otras muchas cosas. Es decir, el análisis de los datos de estos ficheros log podría permitir conocer el proceso de enseñanza-aprendizaje de un curso en mayor profundidad, y gracias a ello mejorarlo.

El problema con los ficheros log mencionados es que su utilidad, sin un tratamiento previo, es casi inexistente. La cantidad y el formato de los datos que contienen hacen que extraer información a partir únicamente de su revisión en crudo sea una tarea muy laboriosa y complicada (Sven F. Crone [12]).

Este Trabajo de Fin de Grado busca solucionar este problema mediante la siguiente estrategia. El objetivo último de este Trabajo de Fin de Grado fue el desarrollo de una aplicación, denominada *Prez*, para la generación de informes compuestos por una serie de indicadores visuales que permitan a los docentes analizar con facilidad la interacción de los alumnos con sus cursos. Esta aplicación se apoya en una librería, a la que se denomina *Maadle*, cuyo objetivo será el de facilitar a los programadores el acceso a los datos contenidos en los ficheros de log de *Moodle*. La idea tras esta separación es que *Maadle* pueda ser reutilizada en otros proyectos de *Moodle*. Por tanto, el desarrollo de esta librería *Maadle* se convirtió en un segundo objetivo de este Trabajo Fin de Grado.

[1] https://moodle.org
[2] https://chamilo.org
[3] https://evolmind.com

1.2. Motivación y Objetivos

Todos los usuarios de *Moodle* con rol de profesor tienen acceso a los ficheros log de los cursos que imparten. Estos ficheros log pueden descargarse desde la plataforma y ser manejados por los profesores como elementos independientes de la misma. Estos ficheros se encuentran en formato *CSV*, un formato en el que los datos se representan en forma de tabla, separando las columnas mediante comas y las filas mediante saltos de línea. En el caso del log de *Moodle*, cada una de las filas representa una acción que un usuario ha realizado en el curso de *Moodle*. En lo que resta de documento, se hace referencia a estas acciones como eventos generados por el usuario. Las columnas del log almacenan los datos recogidos de cada uno de estos eventos, datos que incluyen la fecha y la hora a la que se generó el evento, el nombre completo del usuario que lo desencadenó, el elemento con el que interactuó, entre otros, como muestra el *Listado 1*.

Existe un problema asociado a estos ficheros de log, y es que estos contienen tantos datos y están tan pobremente estructurados que hacen casi imposible el extraer información útil mediante su revisión en crudo. Dependiendo de ciertas variables, como el número de recursos, el número de participantes del curso y la participación de estos en el mismo, el número de eventos registrados en el log puede ascender a las decenas de miles, como es el caso de la asignatura Métodos de Desarrollo en el año 2019-2020 con 15.468 eventos, o Bases de Datos en el año 2018-2019 con 47.172 eventos. Por otro lado, algunas columnas aparecen en otros idiomas e incluyen identificadores que rara vez tendrán interés para el docente y que complican la comprensión del fichero. En el *Listado 1* se muestra cómo se registrarían cinco eventos en el fichero log.

```
1 "23/12/2019 16:48"," Turing, Alan",-,"Carpeta:
  Exámenes",Carpeta,"Módulo de curso visto","The user with id
  '1' viewed the 'folder' activity with course module id
  '5002'.",web,0.0.0.0
2 "30/10/2019 17:25"," Turing, Alan",-,"Foro: Noticias de
  clase",Foro,"Tema visto","The user with id '1' has viewed the
  discussion with id '1001' in the forum with course module id
  '5000'.",web,0.0.0.0
3 "9/09/2019 11:42","BYRON, ADA",-,"Carpeta:
  Exámenes",Carpeta,"Módulo de curso visto","The user with id
  '1008' viewed the 'folder' activity with course module id
  '5002'.",web,0.0.0.3
4 "17/07/2019 12:46","RUSSELL, BERTRAND",-,"Curso: G000 -
  Curso de Testing - Curso 2018-2019",Sistema,"Curso
  visto","The user with id '1018' viewed the course with id
  '5010'.",web,0.0.0.12
5 "3/05/2019 10:59","VON NEUMANN, JOHN",-,"Carpeta: Entrega
  inicial",Carpeta,"Módulo de curso visto","The user with id
  '1015' viewed the 'folder' activity with course module id
  '5016'.",web,0.0.0.16
```

Listado 1: Ejemplo de log de *Moodle* con cinco eventos registrados

Por ejemplo, la primera fila de esos eventos recoge que el día 23 de diciembre de 2019 a las 16:48, el usuario *Alan Turing* entró a la carpeta 'Exámenes' a través de la web de *Moodle*. Además de esto, la fila proporciona una serie de identificadores internos de *Moodle* tanto para el usuario como para la carpeta, que en este caso serían 1 y 5002 respectivamente, y también indica la dirección IP del usuario que generó el evento.

Habiendo visto cómo se recogen los eventos en el log, se hace evidente la dificultad que alberga el interpretar esos datos de forma directa. Si un docente quisiera, por ejemplo, conocer cuál es el recurso de *Moodle* más utilizado por los alumnos, tendría que enfrentarse a decenas o incluso cientos de miles de líneas como las recién expuestas. Para poder extraer la información indicada, en primer lugar, el profesor tendría que encontrar y eliminar todas las filas de eventos generados por docentes u otros usuarios con roles que no interesen, como los administradores. Una vez hecho esto, tendría que ir ubicando los distintos recursos y contar el número de eventos en los que aparecen. Con el número de eventos en los que aparecen calculado, el docente habría de encontrar cuál es el que tiene más ocurrencias, en una lista de decenas o cientos de recursos.

Para aliviar estos problemas, se plantean dos objetivos al respecto, tal como se ha comentado. El primero de ellos es la creación de una librería, denominada *Maadle*[4], que facilite a programadores el acceso a los datos almacenados en los logs de *Moodle*.

Para probar la potencia y adecuación de *Maadle*, se creará además *Prez*[5], una aplicación que, haciendo uso de esta librería, genere informes con indicadores sobre el desarrollo de un curso de *Moodle*. Esta aplicación permitirá extraer información del fichero log sin necesidad de que el docente desenmarañe los datos del mismo. La información proporcionada por *Prez* se espera que sea de ayuda para la mejora continua de los procesos de enseñanza-aprendizaje.

1.3. Alcance del proyecto

En esta sección se hace un breve repaso de las distintas fases del ciclo de vida de desarrollo software, comentando cuál es el alcance en cada una de ellas con relación al proyecto desarrollado en este Trabajo de Fin de Grado. La fase de mantenimiento no se describe puesto que, al no haberse puesto en producción el sistema, dicha fase no ha dado comienzo aún.

Antes de comenzar con el desarrollo software, hubo una etapa de toma de contacto con el dominio del problema. En esta comenzó la formación en las herramientas a utilizar durante el desarrollo del software y el establecimiento de la forma de trabajo. Tras esto, dio comienzo el ciclo de vida de desarrollo.

En primer lugar, en lo que respecta a la fase de requisitos, se partió de unas reuniones iniciales entre los directores y el autor del trabajo. En estas reuniones se consiguió conformar una idea inicial de cuál iba a ser la visión del proyecto, así como su contexto y sus fuentes de requisitos, como se relata en el Capítulo 2.

[4] *Maadle* proviene de la mezcla del término inglés *mad* y la palabra *Moodle.*
[5] *Prez* proviene de la aclamada serie de David Simon *The Wire.* Es el apodo que se le da al personaje Roland Pryzbylewski, detective con especial interés en el estudio de los datos.

Tras estas reuniones iniciales, el director del trabajo organizó una reunión con una docente del área de educación que serviría como fuente principal de requisitos. Tal reunión, permitió la definición más formal de unos requisitos del proyecto.

La arquitectura de la aplicación fue proporcionada por los directores del trabajo al comienzo del mismo. Por un lado, habría una librería que permitiese el tratamiento de los datos recogidos por los logs de *Moodle*, así como el de otros datos externos relacionados que pudieran considerarse de interés. Ese aspecto lo cumpliría *Maadle*. Por otro lado, habría una generación de informes como resultado del análisis de los logs de *Moodle*. Ese aspecto lo cumpliría *Prez*.

En tercer lugar, en lo referente a la fase de implementación, su desarrollo se hizo siguiendo la metodología ágil *Scrum*, que se comenta en la sección siguiente. La infraestructura de desarrollo estaba organizada en torno a un sistema de control de versiones, un sistema de integración continua y un sistema para el control de la calidad del código fuente.

En cuarto y último lugar, en lo concerniente a las pruebas y el despliegue es crucial tener en cuenta tanto la metodología de trabajo que se siguió, como la arquitectura del proyecto. De acuerdo con la metodología ágil seguida, se hizo una verificación continua del correcto funcionamiento de la herramienta. Más concretamente, dentro del proyecto se desarrollaron pruebas de aceptación (pruebas con el usuario final, tal como requería la metodología ágil utilizada) y las unitarias (pruebas a nivel de función programática). No se realizaron pruebas de integración ya que, dadas las características de la arquitectura del proyecto, estas aportaban poco valor.

Por último, en lo concerniente al despliegue, se consideró fundamental proteger y controlar el acceso a los ficheros de log de *Moodle*, ya que estos contienen datos personales de los alumnos. Por tanto, se optó por hacer correr la aplicación de forma local en el computador de cada profesor, evitando que los ficheros log quedasen alojados en servidores externos que pudiesen dar lugar a agujeros de seguridad.

1.4. Metodología e Infraestructura de Desarrollo

En esta sección se comenta de forma detallada cuál fue la metodología seguida para el desarrollo del proyecto, así como la infraestructura de desarrollo. El proyecto objeto de esta memoria se desarrolló siguiendo la metodología ágil *Scrum* (Ken Schwaber [8]), con ciertas modificaciones. Tal metodología fue gestionada mediante la herramienta *Scrumdesk*[6].

Conforme a esta metodología, en primer lugar, como resultado de la reunión con la docente mencionada en la sección anterior, se extrajeron una serie de requisitos. Estos requisitos se introdujeron en *Scrumdesk* en forma de historias de usuario, conformando lo que se conoce como *product backlog*.

En segundo lugar, se estableció la duración de los *sprints*. En *Scrum* los *sprints* son las iteraciones en las que se divide el desarrollo de un proyecto. En este caso, se decidió

[6] https://www.scrumdesk.com

que esta duración fuese de una semana. De esta forma, cada semana habría una versión nueva del software.

A partir de este punto, comenzó un proceso iterativo en el que se repitieron las siguientes acciones. Semanalmente, directores y desarrollador se reunían. En estas reuniones, por un lado, se comentaba el desarrollo del *sprint* anterior, así como los resultados del mismo, evaluando si estos eran satisfactorios o no. Y, por otro lado, se elegían los requisitos a tratar en el siguiente *sprint* generando así lo que se conoce en *Scrum* como *sprint backlog*. Tras esto, se definían los criterios de aceptación de los requisitos y se dividía su desarrollo en tareas que ir gestionando para controlar adecuadamente el trabajo pendiente y el realizado.

Este proceso iterativo es el marcado por *Scrum*, con la diferencia de no llevar a cabo las *Daily Scrum Meetings*, *Sprint Retrospective* y *Sprint Burndown Chart*, técnicas que carecían de sentido en un trabajo desarrollado por un solo individuo y no por un equipo de desarrollo software.

En resumen, la metodología de trabajo seguida fue una adaptación de la metodología ágil *Scrum*. Cada semana, se evaluaba el trabajo hecho en la anterior y se acordaba el que se realizaría en la posterior. De esta forma, cada semana se generaba una versión nueva del software, con la documentación pertinente añadida, las pruebas realizadas y un ejecutable nuevo totalmente funcional.

En lo que respecta a la infraestructura de desarrollo, cabe tratar cinco aspectos: (1) la gestión general del proyecto, (2) los *frameworks* utilizados, (3) el sistema de control de versiones, (3) el sistema de integración continua y (4) el control de la calidad.

En primer lugar, para la gestión general del proyecto se utilizó *Scrumdesk*. Como se indicaba previamente, esta herramienta permite la gestión de proyectos que siguen la metodología ágil *Scrum*. En esta herramienta se especificaron los requisitos y se organizó el trabajo semanal.

En segundo lugar, en lo relativo a los *frameworks* utilizados para la implementación del sistema, tanto para *Maadle* como para *Prez*, se hizo uso de *Python*, por solicitud expresa de los directores del Trabajo de Fin de Grado. En concreto, para *Maadle* se hizo uso de la librería para análisis de datos *Pandas*[7] y para *Prez* se hizo uso del *framework* para desarrollo de informes *Dash*[8].

En tercer lugar, para el control de versiones se utilizó *Git* (Scott Chacon [11]) creando un repositorio en *Github*[9], que se gestionó conforme al esquema de ramificación *Gitflow* (Driessen [6]).

En cuarto lugar, como sistema de integración continua se utilizó *Travis CI*[10] (Travis-CI [13]). Para ello se creó un repositorio enlazado con el de *Github*. De esta forma, cada *commit* realizado en *GitHub* desencadenaba las acciones deseadas en *Travis CI* de forma automática.

[7] https://pandas.pydata.org
[8] https://dash.plotly.com
[9] https://github.com/brian8sal/LibreriaMoodleAnalisis
[10] https://travis-ci.com/brian8sal/LibreriaMoodleAnalisis

En quinto y último lugar, para el control de la calidad se utilizó *Sonarcloud*[11]. Para ello, se creó una organización y dentro de ella se definió un proyecto enlazado con *Travis CI* y con *GitHub*, de manera que se analizase el código de las distintas ramas del proyecto conforme se iba subiendo código al repositorio y superando las pruebas.

2. Requisitos

2.1. Introducción

El presente capítulo recoge cómo se desarrolló la etapa de requisitos del proyecto. Las fases que se describen en él siguen el proceso impartido en la asignatura *Ingeniería de Requisitos*, que puede verse representado en la *Figura 1*.

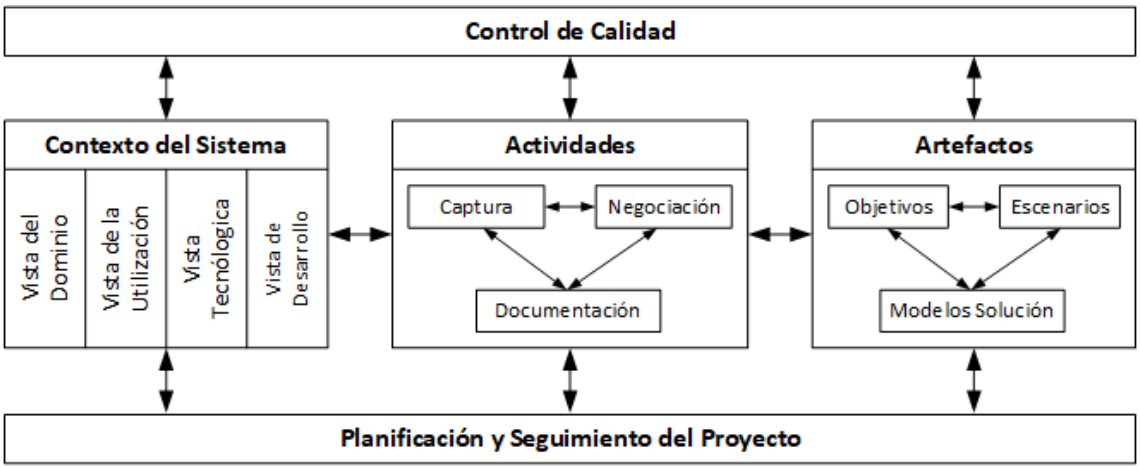

Figura 1: Proceso de ingeniería de requisitos (Pohl [10])

De forma previa a las fases que pueden verse en la *Figura 1*, por ser fieles al proceso, se definió la visión del sistema, es decir, la especificación de sus aspiraciones y objetivos. Tal como se ha comentado en el capítulo anterior, el objetivo de este proyecto es la mejora continua de los procesos de enseñanza-aprendizaje. Y a este objetivo subyacen otros dos. El primero de ellos consiste en la creación de una librería (*Maadle*) que facilite la recogida de los datos almacenados en los ficheros log de *Moodle* para su posterior tratamiento. El segundo de ellos consiste en crear una aplicación (*Prez*) que proporcione una serie de indicadores que permitan conocer el desarrollo de un curso de *Moodle*.

De esta forma, la *visión del sistema*, o el objetivo último del mismo, podría definirse como la mejora continua de los procesos de enseñanza-aprendizaje, mediante la creación de dos herramientas: una librería para la recogida de datos de los ficheros log y una aplicación para la visualización de indicadores que permitan conocer el desarrollo de un curso de *Moodle*.

[11] https://sonarcloud.io/dashboard?id=brian8sal_LibreriaMoodleAnalisis

Tras esto, como puede verse en la *Figura 1*, el proceso tiene una fase inicial en la que se conoce y analiza el contexto al que pertenece el sistema. Este conocimiento del contexto se obtiene desde cuatro vistas a las cuales se dedican las tres siguientes secciones: la vista del dominio, la de utilización, la tecnológica y la de desarrollo, tratándose estas dos últimas en este documento de forma conjunta.

En la *vista del dominio* se busca identificar los elementos, eventos y objetos a los que hará falta atender o tener en cuenta a la hora de definir los requisitos. En la *vista de utilización* se busca identificar a los actores que puede afectar el desarrollo del proyecto. No solo a los usuarios potenciales del sistema desarrollado, sino a todos aquellos que se vean inmersos en el dominio del mismo, y puedan guardar alguna relación con dicho sistema. Y, por último, para terminar con el contexto, se dedica una misma sección a las *vistas de tecnología* y *de desarrollo*. En estas se señalan elementos importantes de la infraestructura de despliegue, así como posibles restricciones a los métodos de desarrollo del proyecto.

Tras la definición del contexto del sistema se pasa a la identificación de las fuentes y captura de requisitos, a lo que se dedicará una sección. En esta sección se hace un breve repaso a cuáles fueron las fuentes de requisitos principales en el proyecto y las formas en que se extrajeron de estas los requisitos. Tras esto, se le dedica una sección a la especificación de requisitos, incluyendo tanto los requisitos funcionales como no funcionales.

2.2. Contexto del Proyecto

2.2.1. Dominio del Sistema

En esta subsección se señalan los elementos que pueden ser relevantes para el proyecto, así como las leyes y restricciones que afectarán al desarrollo del mismo.

El proyecto se ve inmerso en el contexto de los *LMS* (*Learning Management System*) y más concretamente, de *Moodle* (*Modular Object-Oriented Dynamic Learning Environment*). *Moodle* funciona mediante una serie de cursos que no son más que conjuntos de actividades y recursos. Estas actividades y recursos tienen una alta variedad y pueden ser combinados de varias formas, permitiendo así dar diferentes grados de soporte a un proceso de enseñanza-aprendizaje.

Los usuarios de *Moodle* pueden tener diferentes roles, teniendo cada rol, como es de esperar, acceso a distintas funcionalidades. Dentro del conjunto de roles, que puede ser muy variado, los más habituales en cada instalación de *Moodle* son los de alumno, profesor o docente, y administrador. El profesor es el encargado de gestionar las actividades y recursos del curso, mientras que los alumnos solo pueden interactuar con ellos, sin posibilidad de modificarlos. Además, *Moodle* guarda los eventos generados por sus usuarios en el sistema en unos ficheros log.

Este proyecto se centra en los mencionados ficheros log, que pueden descargarse desde la sección de informes de *Moodle*. Tales ficheros log recogen, por cada evento y en el formato mostrado en el *Listado 1*, los siguientes datos, los cuales se explican con ayuda del ejemplo mostrado en el *Listado 2*.

- *Hora*: fecha y hora en la que se produjo el evento. En el caso del evento del *Listado 2* sería *13/02/2019 14:05*.
- *Nombre completo del usuario*: apellidos y nombre del usuario que generó el evento. En el caso del evento del *Listado 2* sería *Bertrand Russell*.
- *Usuario afectado*: apellidos y nombre del usuario al que afecto la acción del evento, si lo hubiera. Un ejemplo de usuario afectado es aquel que recibe una calificación. En el caso del *Listado 2*, el evento no afecta a ningún otro usuario, por lo que en el log se registra como '-'.
- *Contexto del evento:* nombre del recurso/actividad sobre el que se desarrolló la acción del evento. En el caso del evento del *Listado 2* sería *Tarea: Entrega del Proceso de Construcción*.
- *Componente:* componente al que pertenece el recurso/actividad sobre el que se desarrolló la acción del evento. Los componentes permiten a *Moodle* clasificar los distintos elementos de los cursos. En el caso del *Listado 2* el componente sería *Tarea* y otros ejemplos de componente son *foro*, *carpeta*, entre otros.
- *Nombre evento:* describe el tipo de acción realizada. En el caso del evento del *Listado 2* sería *Se ha visualizado el estado de la entrega*.
- *Descripción:* descripción detallada de la acción con identificadores internos del usuario y el recurso/actividad inmersos en el evento.
- *Origen:* forma de acceso a *Moodle* incurrida en la acción del evento. Si se accede a través de la aplicación móvil será *ws*, pues funciona con un servicio web, si no, será *web*.
- *Dirección IP:* dirección IP del dispositivo del usuario que generó el evento.

De esta forma, el registro de un evento cumplimentaría los datos comentados como en el ejemplo siguiente:

```
1. "13/02/2019 14:05","RUSSELL, BERTRAND",-,"Tarea: Entrega del
   Proceso De Construcción",Tarea,"Se ha visualizado el estado
   de la entrega.","The user with id '1018' has viewed the
   submission status page for the assignment with course module
   id '19892'.",web,40.50.52.104
```

Listado 2: Ejemplo de registro de un evento en el fichero log de *Moodle*

En resumen, tal como se muestra en la *Figura 2*, en un curso de *Moodle* se tienen, (1) una serie de recursos y actividades que dan soporte a un proceso de enseñanza-aprendizaje, (2) unos usuarios que entre otras cosas crean o interactúan con tales elementos, y (3) unas calificaciones que evalúan el rendimiento de tales usuarios sobre las mencionadas actividades. Además, *Moodle* mantiene unos registros que almacenan datos sobre los distintos eventos generados en un curso dado a lo largo de su desarrollo.

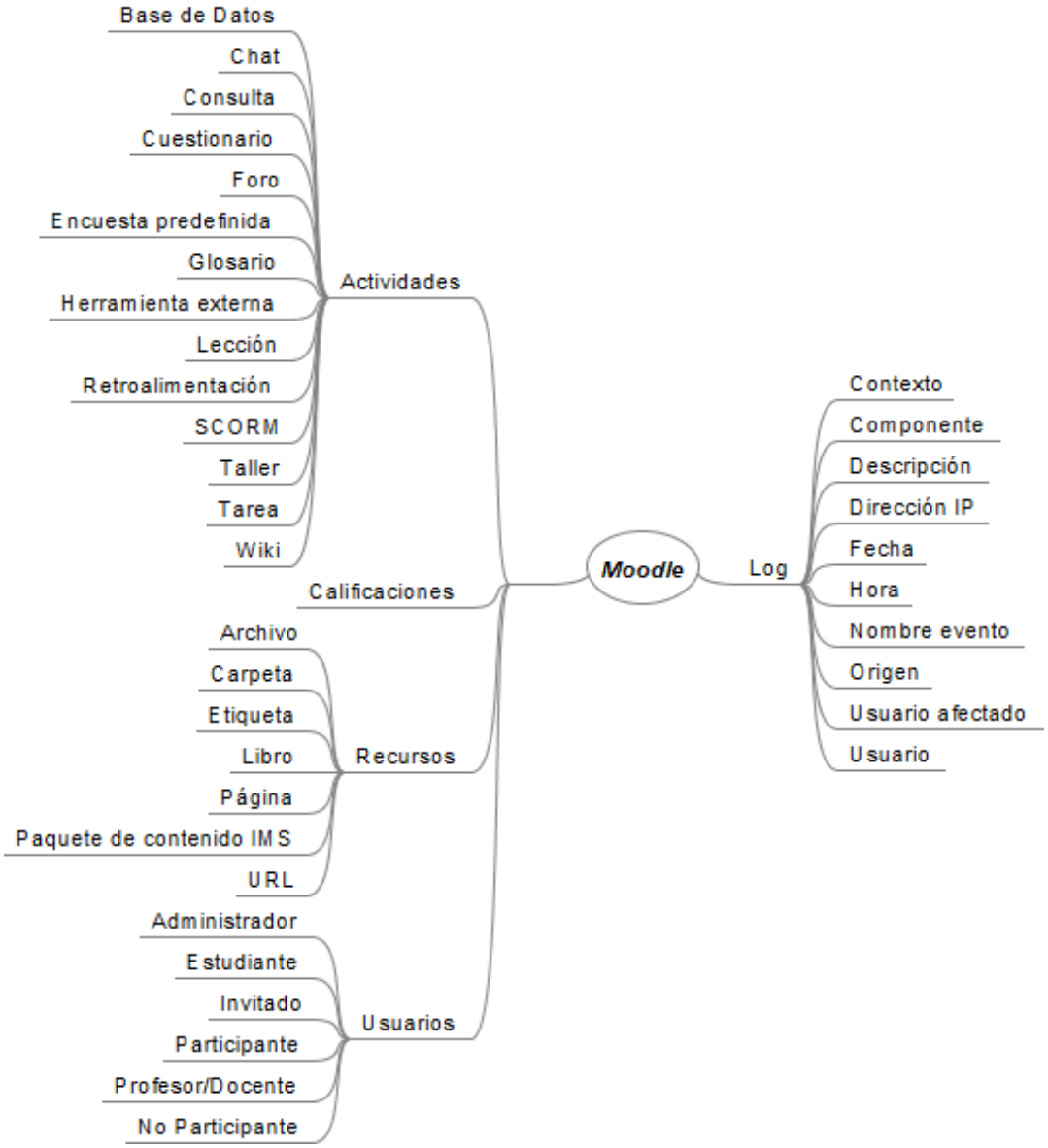

Figura 2: Mapa mental del dominio de *Maadle/Prez*

Dentro del dominio, por último, conviene hacer mención de las leyes de relevancia en el contexto del sistema. Por la naturaleza del sistema, se ha de tener en cuenta la *Ley Orgánica de Protección de Datos Personales y garantía de los derechos digitales*. Los ficheros log de los que se hace uso en el proyecto contienen datos personales de los usuarios de *Moodle*, de ahí la necesidad de tener en cuenta la citada ley. Además, el Comité de Seguridad de la Información de la Universidad de Cantabria aprobó el 13 de mayo de 2019 un documento que detallaba la aplicación de la mencionada ley a la propia universidad. El documento puede encontrarse bajo el titulo *Normativa reguladora del ejercicio de los derechos de acceso, rectificación, supresión, limitación, oposición y portabilidad de los datos de carácter personal recogidos en tratamientos de la Universidad de Cantabria*, siendo importante tenerlo también en cuenta a lo largo del proyecto.

2.2.2. Actores y Utilización

En esta sección se recogen los actores que de una forma u otra pudieran guardar relación con el sistema. Adicionalmente, como ejemplo, se añaden tablas con el análisis completo del perfil de dos de estos actores en el *Apéndice A.*

El primer actor identificado es todo aquel docente de la Universidad de Cantabria que haga uso de *Moodle*. Este actor sería el principal usuario de *Prez*, siendo su objetivo poder analizar fácilmente el uso que se ha hecho de *Moodle* durante el desarrollo del curso académico, de cara a mejorar sus procesos de enseñanza-aprendizaje.

El segundo actor identificado sería cualquier científico de datos con experiencia previa en el tratamiento de los ficheros log de *Moodle*. Este actor sería el principal usuario potencial de *Maadle*. El objetivo de este actor con *Maadle* es el disponer de una serie de funciones que faciliten el tratamiento de los ficheros log de *Moodle* desde el ámbito de la programación y el análisis de datos.

El tercer actor identificado lo constituyen los alumnos de la Universidad de Cantabria que sean usuarios de *Moodle*. Este actor no supone un usuario potencial de *Prez* ni de *Maadle*, pero no por ello deja de ser un actor interesante para la extracción de requisitos, ya que los eventos contenidos en los ficheros log de *Moodle* se generan a partir de sus acciones. Además, podrían estar interesados en el tratamiento que se hace de los datos generados por ellos, pudiendo, por ejemplo, querer que no se utilicen sus datos para determinados fines.

Como cuarto actor identificado se considera a los miembros del Servicio Informática de la Universidad de Cantabria. De nuevo, este actor no supone un usuario potencial de ninguna de las herramientas desarrolladas, pero conviene tenerlo en cuenta al ser el organismo que pudiera gestionar la distribución del software desarrollado en el proyecto y el responsable de la plataforma *Moodle* de la cual se extraen los datos a analizar.

El quinto actor identificado lo constituyen el rector de la Universidad de Cantabria y el decano de la Facultad de Ciencias. Al igual que antes, no supone un usuario potencial de las herramientas, pero pudieran ser relevantes durante la extracción de requisitos al ser una figura importante en el mundo de la enseñanza en el que se ve inmerso el proyecto, que seguramente estará interesado por saber qué sucede en las instituciones que gobiernan.

2.2.3. Tecnología y Desarrollo

En esta sección se recoge cierta información importante sobre la infraestructura tecnológica donde desplegar el sistema y algunos comentarios sobre posibles restricciones a sus métodos de desarrollo.

En primer lugar, *Maadle* debía ser desarrollado haciendo uso de *Python* y la librería *Pandas*. Dicha condición surge por petición de los directores del proyecto, con el fin de ser útil para sus investigaciones en este ámbito (Alfonso de la Vega Ruiz [1]), (Alfonso de la Vega Ruiz [2]). Asimismo, debía ser desarrollado para la versión 3 de *Moodle*, pues es la utilizada a nivel institucional por la Universidad de Cantabria.

A la hora de considerar el desarrollo del proyecto como un plugin de *Moodle*, deben tenerse en cuenta las fuertes restricciones y la compleja burocracia impuestas por los Servicios de Informática de la Universidad de Cantabria para la ejecución de esta tarea. Además, un segundo factor a tener en cuenta es que los plugin de *Moodle* deben desarrollarse en *PHP*, mientras que *Maadle*, tal como se ha comentado anteriormente, debido a otros intereses, debía desarrollarse en *Python*, por lo que podrían surgir serios problemas de interoperatividad en caso de decantarse por esta opción.

2.3. Fuentes y Captura

En esta sección se recogen las principales fuentes de requisitos del sistema y se comenta cómo se desarrolló la captura de requisitos.

En primer lugar, la principal fuente de requisitos del proyecto fue el actor docente de la Universidad de Cantabria descrito en la sección previa. Como representante principal de dicha fuente se tuvo a Ruth Villalón Molina, docente e investigadora del Departamento de Educación de la Universidad de Cantabria, adscrita al área de Psicología Evolutiva y de la Educación. Esta profesora había trabajado previamente con los directores de este Trabajo Fin de Grado y estaba altamente interesada en los resultados del mismo, por lo que accedió con facilidad a colaborar con su desarrollo, hecho que se le agradece sinceramente desde estas páginas. Su participación consistió en dos entrevistas. Una inicial, de la que se extrajo el grueso de los requisitos del sistema. Y una, con el sistema ya avanzado, con el fin de comprobar si se cumplían los objetivos propuestos, y donde podían, además, proponerse nuevos requisitos.

En la primera entrevista Ruth proporcionó una serie de propuestas iniciales que fueron comentadas y refinadas hasta su posterior especificación en forma de requisitos. Por otro lado, la segunda entrevista se desarrolló con una demostración del funcionamiento de la aplicación y una serie de comentarios posteriores sobre el cumplimiento de lo planteado.

Para complementar la visión de Ruth del proyecto, se comenzó a desarrollar un cuestionario a difundir entre más docentes. Sin embargo, la crisis sanitaria de la COVID-19 interrumpió tal proceso. Las urgencias por transitar de un modelo de docencia presencial a un modelo de docencia virtual, más todas las dificultades asociadas a la citada crisis sanitaria, generaron una sobrecarga de trabajo en los cuerpos docentes que hacía poco recomendable solicitarles la realización de cualquier tarea adicional, por mínima que fuera, como la de completar el mencionado cuestionario. Por dicho motivo, se optó por renunciar al mismo.

En segundo lugar, otra fuente de requisitos para el proyecto fueron los propios ficheros log, así como la documentación de *Moodle*. Estas son fuentes de las que se extrajeron requisitos a partir de su mera revisión. Por ejemplo, de la revisión de los ficheros log se pudieron extraer condiciones a tener en cuenta a la hora de definir nuevas métricas, así como métricas en sí mismas que normalmente acababan siendo requisitos.

2.4. Especificación de Requisitos

2.4.1. Objetivos

Los objetivos que aquí se recogen son resultado del análisis de la información obtenida en la reunión inicial con Ruth Villalón Molina. Parte de dichos objetivos se muestran, a modo de ejemplo, en la *Figura 3*, en forma de diagrama de objetivos *GRL* (*Goal-oriented Requirements Language)* (University of Toronto [14]).

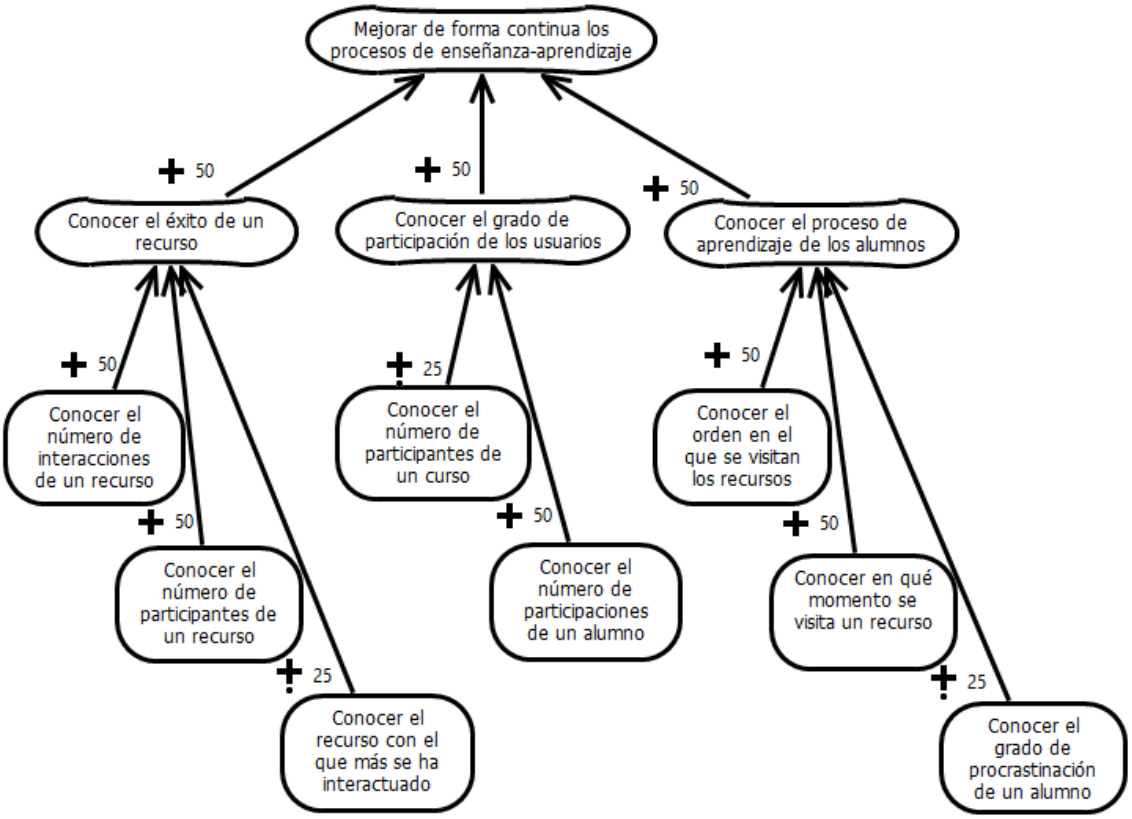

Figura 3: Diagrama de objetivos

En el diagrama de objetivos de la *Figura 3*, pueden distinguirse tres niveles. El primero de ellos corresponde al objetivo de más alto nivel del sistema: *mejorar de forma continua los procesos de enseñanza-aprendizaje*. Este sería el objetivo último del sistema, y, para su cumplimiento se definen otros tres objetivos más concretos: (1) *conocer el éxito de un recurso*, (2) *conocer el grado de participación de los usuarios* y (3) *conocer el proceso de aprendizaje de los alumnos*. A su vez, el cumplimiento de los objetivos del segundo nivel se ve favorecida por el cumplimiento de los del tercer nivel, que en este caso serían las métricas.

Por ejemplo, con el fin de cumplir el objetivo *mejorar de forma continua los procesos de enseñanza-aprendizaje*, se trata de cumplir el objetivo *conocer el éxito de un recurso*, cuyo cumplimiento sería favorecido por la métrica, u objetivo de tercer nivel, *conocer el número de participantes de un recurso*.

2.4.2. Requisitos Funcionales

De nuevo, los requisitos funcionales del sistema que aquí se recogen provienen del análisis de la información resultante de la reunión inicial con Ruth Villalón Molina. Estos requisitos se especificaron como *historias de usuario* (Cohn [3]), por ser esta la técnica de especificación de requisitos habitual de la metodología *Scrum*, la metodología ágil utilizada para el desarrollo de este proyecto.

Los requisitos capturados se organizaron en tres historias *épicas*. La primera historia épica hace referencia a métricas relacionadas con el curso o asignatura de *Moodle*, la segunda a los recursos y la tercera a los participantes.

La historia épica del análisis de un curso hace referencia a una asignatura concreta. Esta historia épica concibe todo lo relativo a un curso de *Moodle*, incluyendo a los usuarios y a los recursos pertenecientes a tal curso. Un ejemplo de historia de usuario perteneciente a esta historia épica sería el indicar el número total de eventos de un curso.

La historia épica del análisis de los recursos hace referencia a los elementos que conforman un curso. Esto es, los elementos de un curso con los que los usuarios de *Moodle* interactúan al utilizar la plataforma. Un ejemplo de historia de usuario perteneciente a esta historia épica sería el indicar el número de participantes que han interactuado con un recurso concreto.

Por último, la historia épica del análisis de los participantes hace referencia a los usuarios de *Moodle*, que pueden ser docentes o estudiantes. Un ejemplo de historia de usuario perteneciente a esta historia épica sería el indicar el participante con más eventos del curso.

A modo de ejemplo, para ilustrar el aspecto que tenían estas historias de usuario, se incluye la especificación de una de ellas, más concretamente, la de *Conocer Eventos por Recurso*.

Yo, como usuario, quiero poder conocer el número total de veces que los usuarios han interactuado con un recurso concreto, de manera que pueda saber fácilmente cuáles son los recursos más populares.

Inicialmente, conformando el primer conjunto de requisitos, se identificaron 34 historias de usuario.

2.4.3. Requisitos No Funcionales

En lo que respecta a los requisitos no funcionales, se revisó la taxonomía propuesta en el estándar ISO 25010. Dentro de ese estándar, *Maadle* no debía cumplir con requisitos no funcionales especiales más allá de los habituales para la mayoría de categorías. Por ejemplo, con respecto al rendimiento, el sistema debía ser igual de rápido que cualquier sistema actual, admitiéndose incluso ligeras demoras en casos de cursos o métricas muy pesadas. Por ello, no se analizaron en profundidad requisitos para estas categorías.

No obstante, sí había una categoría que merecía especial atención, que era la de seguridad, y dentro de esa categoría, su categoría de confidencialidad.

Como se ha comentado anteriormente, *Maadle* y *Prez,* al trabajar con los ficheros log de *Moodle,* tratan con datos personales de los usuarios de los cursos, tanto alumnos como docentes. De esta forma, el sistema debería salvaguardar la confidencialidad de los datos que trata tanto durante su desarrollo como en su futuro despliegue y utilización, asegurando que solo puedan acceder a ellos los docentes involucrados en cada curso.

3. Arquitectura

3.1. Diseño Arquitectónico

En esta sección se recoge el diseño arquitectónico del sistema. La base de tal diseño vino propuesta por los directores del proyecto. Por un lado, el sistema debía tener una librería, *Maadle*, que se encargase de recoger y procesar los datos de los ficheros log de *Moodle*. Por otro lado, el sistema debía tener una segunda capa, *Prez*, que permitiera visualizar ciertos indicadores generados con *Maadle*. En la medida de lo posible, desde *Prez* no se debía hacer más que representar lo recibido de *Maadle*.

Maadle, con el fin de ser útil para otros proyectos de este ámbito, debía estar implementado en *Python*, y debía ser compatible con la librería *Pandas*, una librería *Python* ampliamente utilizada para el procesamiento y gestión de datos. Ambas cuestiones fueron expresamente solicitadas por los directores del proyecto. *Pandas* está estructurada en base a dos tipos de datos fundamentales, que son *Series* y *DataFrames*. Los *Series* son *arrays* unidimensionales capaces de contener cualquier tipo de dato, y los *DataFrames* son matrices, también capaces de contener cualquier tipo de dato. Por tanto, para que los resultados devueltos por *Maadle* pudiesen manipularse fácilmente desde código *Pandas*, las funciones de *Maadle* debían devolver objetos de tipo *Series* o *DataFrame*.

De esta forma, *Prez* debía encargarse únicamente de representar los *DataFrames* o *Series* que le proporcionase *Maadle*. Existen diferentes aplicaciones que permiten representar datos a modo de informe, para su fácil digestión. Dentro de la comunidad de *Python*, *Dash* es una de las aplicaciones más populares. Esta librería permite construir aplicaciones web para mostrar informes de análisis de datos mediante la utilización de una serie de potentes componentes reutilizables y personalizables, como pueden ser gráficos de barras, mapas de calor, gráficos circulares, entre muchos otros. Estos componentes reciben un *DataFrame* de *Pandas* y, tras una adecuada configuración, muestran gráficos que permiten al usuario visualizar e interactuar con la información contenida en los *DataFrames*. De esta manera, el programador puede centrarse en la producción de los *DataFrames* a visualizar, encargándose *Dash* de aspectos más pesados de la visualización de estos *DataFrames*, como puede ser el del escalado de los datos para adecuarlos al espacio disponible. Para construir estos *DataFrames*, *Maadle* procesaría un fichero log de *Moodle*. *Moodle* proporciona estos ficheros log en diferentes formatos, entre ellos, *CSV*. En este proyecto se optó por trabajar con este formato por su compatibilidad con *Pandas*.

Para analizar la viabilidad de esta arquitectura, y para tener una base sobre la cual poder empezar a desarrollar *sprints*, se creó un *walking skeleton* de la aplicación. El *walking skeleton* construido mostraba una serie de métricas, seleccionadas expresamente para analizar la versatilidad y potencia de *Dash* ante diferentes escenarios. De esta forma, el *walking skeleton* mostraba: (1) un indicador textual, que reflejaba el número total de eventos registrados por el log durante el curso; (2) un indicador gráfico, que mostraba el número de eventos registrados en cada mes del curso en forma de gráfico de barras; y, (3) un indicador configurable o interactivo, que mostraba el número registrado de eventos por día, dentro de un rango de fechas configurable por el usuario.

El *walking skeleton* permitió analizar la viabilidad de la arquitectura, así como el funcionamiento de las herramientas que integraba. En concreto, permitió analizar el funcionamiento de *Dash*.

Esta arquitectura inicial, la compuesta en el *walking skeleton*, es la que puede observarse en la *Figura 4*, dentro del rectángulo *Core*. Tal arquitectura se fue adaptando a las necesidades del proyecto.

La primera modificación surgió de la decisión de no utilizar ningún servidor web externo para correr *Dash*. Tal como se ha comentado anteriormente, *Dash* genera unas aplicaciones que deben correr en un servidor web. Por tanto, existían dos opciones para ejecutar *Prez*. Utilizar un proveedor de servidores web externo, como *Heroku* o *Amazon Web Services*, o ejecutar la aplicación en un servidor web local instalado en la computadora de cada usuario. Con el objetivo de proteger la privacidad de los ficheros de log, se optó por la segunda opción.

La segunda adaptación surgió de la necesidad de proporcionar al sistema ciertos datos adicionales para el análisis del curso que no podían ser encontrados en el fichero log de *Moodle*, como por ejemplo qué alumnos estaban de Erasmus, o el tema al que pertenecía cada recurso. Estos datos adicionales se proporcionaban al sistema a través de un fichero *Excel* y una copia de seguridad del curso extraída de la plataforma *Moodle*. Estas copias de seguridad son archivos comprimidos en formato *MBZ*, un formato interno de *Moodle*, que contienen ficheros *XML* con información útil, que por ejemplo, permite saber el tema al que pertenece cada recurso del log.

La tercera y última adaptación surge de la necesidad de facilitar al usuario la utilización del sistema. Con este fin, se creó el componente *Prez starter*, que proporcionaría una interfaz gráfica con la que el usuario podría interactuar para realizar la configuración inicial del sistema, seleccionando, por ejemplo, el fichero log del curso a analizar. Además, desde esta interfaz, se arrancaría el servidor local en el cual correría *Prez.*

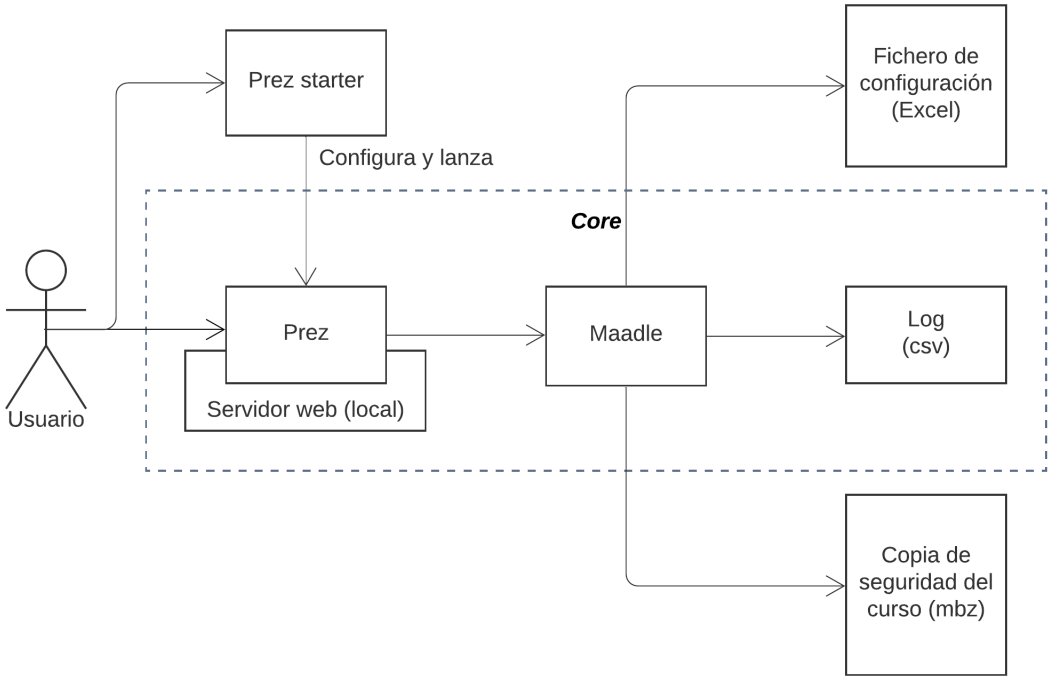

Figura 4: Diseño arquitectónico del sistema

En las siguientes secciones, se describen cada uno de los componentes de la arquitectura.

3.2. *Maadle*

Maadle es el componente que se encarga de facilitar el acceso a los datos de los ficheros log de *Moodle*. Para ello, haciendo uso de *Pandas*, los preprocesa y los carga en memoria. Además, proporciona un conjunto de funciones que realizan una serie de agrupaciones y cálculos sobre los datos con el fin de proporcionar cierta información de interés. De esta forma, *Maadle* estructura el log convirtiéndolo en un *DataFrame* de *Pandas*, añadiendo una serie de columnas y limpiando una serie de filas, como se comentará más adelante, con el fin de facilitar la extracción de datos del mismo. A su vez, proporciona una serie de funciones que permiten la extracción de diferentes indicadores que permiten analizar el desarrollo y rendimiento de un determinado curso académico. Las funciones que contiene *Maadle* pueden verse en el *Listado 3*.

```
1.  __init__(self, name, path, config, backup)
2.  create_config(self, config)
3.  create_data_frame(self, name, path) -> DataFrame
4.  create_data_frame_file_fame(self, name) -> DataFrame
5.  add_ID_user_column(self) -> DataFrame
6.  add_ID_resource_column(self) -> DataFrame
7.  delete_columns(self, columns) -> DataFrame
8.  delete_by_ID(self, idList) -> DataFrame
9.  change_hora_type(self) -> DataFrame
10. add_month_day_hour_columns(self) -> DataFrame
11. create_dynamic_session_id(self) -> DataFrame
12. course_structure(self, backup) -> DataFrame
```

```
13.  between_dates(self, initial, final) -> DataFrame
14.  number_of_sessions_by_user(self) -> DataFrame
15.  num_events(self) -> int
16.  num_teachers(self) -> int
17.  num_participants_per_subject(self) -> int
18.  num_participants_nonparticipants(self) -> DataFrame
19.  list_nonparticipant(self) -> Series
20.  num_events_per_participant(self) -> DataFrame
21.  events_per_month(self) -> DataFrame
22.  events_per_week(self) -> DataFrame
23.  events_per_day(self) -> DataFrame
24.  events_per_hour(self) -> pd.DataFrame
25.  events_per_resource(self) -> DataFrame
26.  participants_per_resource(self) -> DataFrame
27.  resources_by_number_of_events(self, min, max) -> DataFrame
28.  events_between_dates(self, initial, final) -> DataFrame
29.  events_per_day_per_user(self, usuario) -> DataFrame
30.  events_per_day_per_resource(self, resource) -> DataFrame
31.  sessions_matrix(self) -> DataFrame
```

Listado 3*: API Maadle*

En *Maadle* se pueden distinguir dos tipos de funciones, aquellas que sirven para estructurar el fichero log de *Moodle* (líneas 1-13) y aquellas que sirven para explotar los datos que el fichero log contiene (líneas 14-31).

Las funciones que estructuran el fichero log de *Moodle* se encargan de cargarlo en memoria, así como de aplicar un preprocesado, cuyos principales objetivos son la organización y la limpieza de los datos en crudo. Adicionalmente, este preprocesado permite añadir nuevas columnas a partir de datos del log. Un ejemplo de esto es *add_month_day_hour_columns*, que a partir del procesado de la columna *Hora* del log añade tres nuevas columnas al *DataFrame* que lo representa: una con el mes, otra con el día y otra con la hora del evento. También hay funciones que añaden columnas a partir de datos externos, como es el caso de *course_structure*, que añade una columna con el tema al que pertenece cada recurso. Además, también hay funciones que se encargan de cambiar el tipo de ciertas columnas, como *change_hora_type*, que cambia el tipo de dato de la columna *Hora* para que pase a ser un tipo fecha. Por otro lado, la limpieza que realizan estas funciones consiste en la eliminación de ciertas filas, aspecto del que se encarga *delete_by_ID*, que permite eliminar todas las filas referentes a un usuario determinado. Esto permite, por ejemplo, eliminar todos los eventos generados por el administrador de *Moodle*, o por alumnos que, aún estando de Erasmus, han accedido puntualmente al curso, generando algunas interacciones espurias. Estas funciones son utilizadas por el constructor para conseguir una estructura con la que puedan trabajar más fácilmente el resto de funciones.

Por otro lado, las funciones que se encargan de explotar el contenido del fichero log lo que hacen es organizar los datos bajo distintas agrupaciones y cálculos teniendo como resultado datos que representan métricas de interés. Un ejemplo de esto es *resources_by_number_of_events* que se encarga de recoger aquellos recursos que se hayan visto inmersos en cierto rango de eventos, devolviendo tal información en forma de *DataFrame*. Otro es el caso de *sessions_matrix*, que se encarga de construir una matriz de relación de los distintos recursos, permitiendo conocer qué parejas o conjuntos de recursos se tienden a utilizar en la misma sesión.

3.3. *Prez*

Prez es el componente que se encarga, haciendo uso de *Dash*, de visualizar una serie de indicadores a partir de los *DataFrames* provenientes de *Maadle*. A modo de ejemplo, en la *Figura 5* figura el mapa de calor de las relaciones entre los recursos y las sesiones, uno de los diversos indicadores mostrados por *Prez* para facilitar el análisis del desarrollo de un curso.

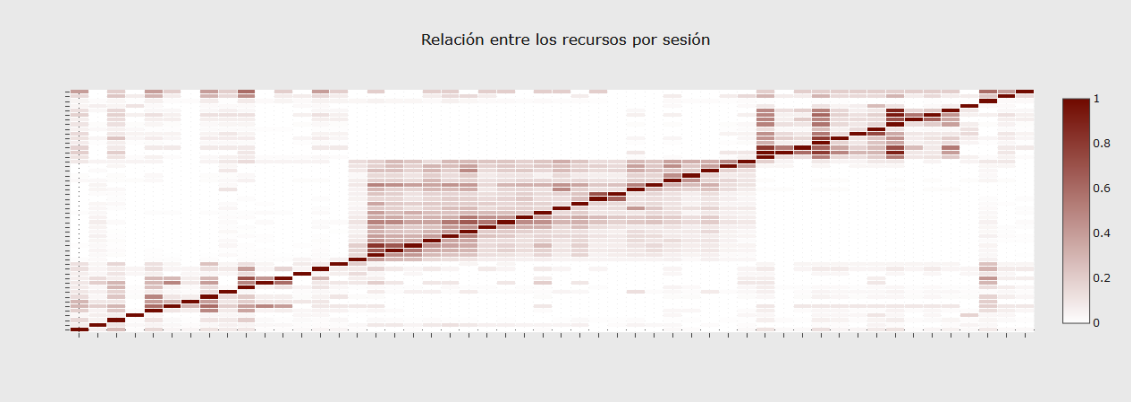

Figura 5: Gráfico de *Prez* 'Relación entre los recursos por sesión'

Tal gráfico es un elemento *Dash* de tipo *heatmap*, que para ser visualizado correctamente requiere cierta configuración. Esta configuración se recoge en *Prez*. *Dash* proporciona una estructura básica en la que, para visualizar un gráfico, han de proporcionarse, como mínimo, los datos a mostrar en cada eje. En la visualización generada, *Dash* proporciona por defecto una serie de herramientas que pueden verse en la *Figura 6*. Estas herramientas permiten:

1. Descargar una imagen del gráfico en formato *PNG*.
2. Hacer zoom en cualquier parte del gráfico.
3. Desplazarse por el gráfico.
4. Hacer *zoom in* general en el gráfico.
5. Hacer *zoom out* general en el gráfico.
6. Autoescalar el gráfico.
7. Reestablecer el estado inicial del gráfico.
8. Mostrar una traza de correspondencia entre los datos de todos los ejes del gráfico.
9. Mostrar los datos de los ejes del gráfico en una misma etiqueta o en varias.

Todas estas potentes funcionalidades son automáticamente generadas por *Prez*.

Figura 6: Herramientas *Dash* para interactuar con los gráficos

Si bien *Dash* se encarga de la visualización del gráfico, si se quiere que la visualización sea adecuada, es necesaria la configuración del correspondiente *widget* en *Prez*, más allá de la indicación de los datos a representar. Desde *Prez*, ha de configurarse el texto que mostrarán las etiquetas del gráfico, los valores que representan, los colores a utilizar, el

tratamiento del texto que acompaña a cada eje, entre otras cosas. Dependiendo del grado de personalización que se quiera alcanzar, esta configuración puede ir desde la simple especificación de unas etiquetas a mostrar en cada eje, a la definición de complejos *callbacks* que permitan definir un color adecuado para cada elemento del gráfico. En el Capítulo 4 se muestran algunos ejemplos de estos *callbacks* de configuración.

En resumen, *Prez* se estructura en torno a la definición y configuración de una serie de *widgets* de *Dash* para la representación de datos. Estos *widgets* permiten visualizar diversos indicadores a partir de las estructuras de datos proporcionadas por *Maadle*.

3.4. Ficheros de Configuración

En esta sección se comentan los ficheros de configuración que contempla el sistema: los ficheros *Excel* y las copias de seguridad *MBZ* de los cursos de *Moodle*. Ambos ficheros se hicieron necesarios en determinados puntos del desarrollo del sistema, con el fin de cumplir ciertas funcionalidades que aquí se comentan.

En cierto punto del desarrollo del sistema, como se comentaba previamente, surgió la necesidad de adaptar la concepción inicial de *Maadle* para que pudiese recibir otros datos complementarios al propio fichero log. En concreto, esta necesidad surgió al querer mostrar en el análisis de *Prez* un indicador con el número de usuarios de un curso de *Moodle* que no hubiesen participado en él, así como una lista de sus nombres. Los ficheros log recogen los datos de todos los eventos generados por los usuarios con un curso, pudiéndose saber quiénes son todos los que han participado. Sin embargo, lo que no se podía saber es quiénes no lo habían hecho, ni tampoco su número.

Para ponerle remedio a esta situación y poder implementar ese indicador, se decidió definir un fichero de configuración externo, en formato *Excel*, en el que el usuario pudiera proporcionar datos que el propio log no contuviera. Se decidió que dichos ficheros estuviesen en formato *Excel* por ser este un formato que resulta familiar a sus potenciales usuarios y que es además fácil de manipular por nuestra aplicación. Para facilitar el trabajo al usuario, la estructura inicial de estos ficheros *Excel* la crearía el propio sistema, de manera que el usuario solo tenga que incorporar la información extra que quisiese añadir en determinadas celdas de estas hojas de cálculo.

Una vez creados estos ficheros de configuración, siguieron surgiendo usos que se le podrían dar para aprovechar su generación. De esta forma, se decidió crear dos hojas dentro del fichero para así organizarlo mejor, una para los datos relativos a los usuarios y otra para los datos relativos a los recursos. A continuación, en la *Figura 7* y en la *Figura 8* se muestra un ejemplo de fichero de configuración, para facilitar la comprensión de su utilidad.

1	Nombre completo del usuario	Excluido
2	BADDAGE, CHARLES	
3	BEREZIN, EVELYN	
4	BERNERS LEE, TIM	
5	BRIN, SERGEY	
6	BYRON, ADA	
7	HOPPER, GRACE	
8	LAMARR, HEDY	
9	NASH, JOHN	
10	RITCHIE, DENNIS	
11	RUSSEL, BERTRAND	
12	Stallman, Richard	X
13	Turing, Alan	X
14	VON NEUMANN, JOHN	

Figura 7: Hoja 'Usuarios' de un fichero de configuración

Como puede verse en la *Figura 7*, en la hoja de los usuarios del fichero de configuración hay dos columnas. La primera de ellas contiene el nombre completo de todos los usuarios que aparecen en el fichero log y que, por lo tanto, han participado en el curso de *Moodle*. De querer conocer datos como el número de los que no han participado, el usuario debe incluir los nombre de aquellos que no aparezcan. En el caso de la Universidad de Cantabria, esta inclusión es sencilla, ya que el formato de los nombres en *Moodle* es el mismo que el utilizado en el Campus Virtual, por lo que para incluir el total de nombres bastaría con directamente copiar la lista del campus en el fichero *Excel*. La segunda columna (*Excluido*), permite eliminar del análisis de *Prez* los datos relativos a cualquier usuario. Esto puede ser útil, por ejemplo, para eliminar los datos relativos a alumnos que estén fuera realizando un intercambio Erasmus, pero hayan entrado puntualmente en el curso.

1	Contexto del evento	Alias	Excluido
2	Foro: Noticias de clase	Foro: Noticias de clase	X
3	Carpeta: Exámenes	Carpeta: Exámenes	
4	Tarea: Entrega pregunta 4 del examen: Desarrollo de Sisemas Inteligentes	Tarea: Entrega pregunta 4 del examen: Desarrollo de Sisemas Inteligentes	
5	Carpeta: Recursos del Alumnado	Carpeta: Recursos del Alumnado	
6	Carpeta: Papeleo	Carpeta: Papeleo	
7	Carpeta: Recursos del Alumnado	Carpeta: Recursos del Alumnado	
8	Tarea: Entrega del Proceso MSI de MÉTRICA v3 modelado en SPEM	Tarea: Entrega del Proceso MSI de MÉTRICA v3 modelado en SPEM	
9	Foro: Dudas Generales	Foro: Dudas Generales	X
10	Página: Enlaces de Interés	Página: Enlaces de Interés	

Figura 8: Hoja 'Recursos' de un fichero de configuración

En la segunda hoja del fichero de configuración, la ilustrada en la *Figura 8*, se encuentran los datos relativos a los recursos del curso de *Moodle*. En ella tenemos tres columnas: la primera de ellas contiene los nombres de los recursos del curso de *Moodle* junto a su tipo. La segunda permite al usuario establecer alias a los recursos. Los nombres proporcionados a los recursos en *Moodle* pueden ser muy largos o incluso estar repetidos, por ejemplo, es muy común encontrar múltiples recursos con el nombre *transparencias* o *apuntes*. De esta forma, esta columna permitirá al usuario cambiar tales nombres para que, al visualizarlo en la interfaz gráfica, le sea más fácil distinguirlos. La tercera y última columna permite al usuario excluir ciertos recursos del análisis de *Prez* de la misma forma que en la hoja previa con los usuarios. Por ejemplo, al usuario pueden no interesarle los eventos en los que se vean inmersos los foros del

curso. En este caso simplemente tendría que marcarlos en tal columna, y su información no aparecería.

Por otro lado, también se ofrece la posibilidad de proporcionar a *Prez* una copia de seguridad del curso de *Moodle*, en formato *MBZ*. En esta copia de seguridad pueden encontrarse muchos ficheros útiles, el que se trató en el trabajo fue *moodle_backup.xml*. Con él, el sistema podría conocer cuál es la estructura del curso y conocer los temas a los que pertenece cada recurso. De esta forma, en la interfaz gráfica de la aplicación, cuando se muestren listas de recursos, se pueden ordenar por temas, lo cual facilita la comprensión de ciertas métricas, como veremos más adelante.

3.5. *Prez starter*

Como se comentaba previamente, en cierto punto del desarrollo surgió la necesidad de crear una interfaz gráfica para proporcionar los ficheros de configuración comentados en la Sección 3.4. Con ese fin, de desarrolló *Prez starter (PrezGUI)*.

Prez starter es el componente que se encarga, haciendo uso de *Tkinter*[12], una librería *Python* para el desarrollo de aplicaciones de escritorio, de mostrar una interfaz gráfica que permita al usuario proporcionar, de manera sencilla, los ficheros necesarios para ejecutar *Prez*. De esta forma, este componente se estructura en una serie de ventanas que permiten al usuario utilizar el explorador de archivos para seleccionar el fichero log, su fichero *Excel* de configuración y la copia de seguridad del curso de *Moodle* a analizar. Con ellos proporcionados, *Prez* es lanzado para mostrar el reporte del curso.

3.6. Flujo de Ejecución

Una vez descritos los componentes que conforman la arquitectura, cabe describir cuál es el flujo de ejecución que los atraviesa para que el usuario pueda ejecutar la aplicación y tener disponible el reporte del curso.

El flujo comienza cuando el usuario inicia la aplicación. Una vez lanzada, se ejecuta *Prez starter*, en las ventanas mostradas por este, el usuario selecciona el fichero log, el fichero *Excel* de configuración y la copia de seguridad del curso de *Moodle* a analizar. Tras esto, desde *Prez* se proporcionan a *Maadle* tales ficheros para que este se encargue de extraer y estructurar los datos contenidos en ellos. Una vez hecho esto, se establece una comunicación entre *Prez* y *Maadle* en la que *Prez* hace uso de funciones de *Maadle* para que este le retorne *DataFrames* y *Series* que visualizar. Una vez que *Maadle* ha proporcionado todos los datos, *Prez* crea un servidor local y abre un navegador web en el que el usuario podrá visualizar e interactuar con el reporte realizado.

[12] https://wiki.python.org/moin/TkInter

4. Implementación

4.1. *Maadle*

En esta sección se describe la implementación de *Maadle*. Tal como se ha comentado anteriormente, *Maadle* es una clase que hace uso de la librería de *Python* para el análisis de datos *Pandas*. Para facilitar el acceso a los datos del log, y evitar además continuos accesos al fichero, antes de manipular un log, *Maadle* lo preprocesa, limpiando ciertos campos, y lo carga en memoria. Esta tarea la realiza el constructor de *Maadle*.

Concretamente, el constructor se encarga de crear un *DataFrame* con columnas para el identificador del usuario, el identificador del recurso, el identificador del tema al que pertenece el recurso, la sesión a la que pertenece el evento, la hora, el día y el mes en el que se produjo. Asimismo, el constructor se encarga de procesar cada fila del log y de convertirla a una fila del *DataFrame*. Además, el constructor crea el fichero de configuración del curso de *Moodle*, y, mediante su procesado, elimina los datos relativos a los usuarios y recursos que no se deseen y establece los alias para los recursos.

Para la estructuración de la que se encarga el constructor, en la que un *DataFrame* contiene los datos del fichero log, se hace uso de varias funciones. Varias de estas funciones, en algunos casos hacen uso de expresiones regulares para añadir columnas de interés al *DataFrame.* Este es el caso de *add_ID_resource_column*, que se encarga de procesar el campo *Descripción* extrayendo, mediante una expresión regular, el identificador de cada recurso. Por ejemplo, del evento con *Descripción "The user with id '1018' viewed the 'folder' activity with course module id '5014'."* la expresión regular (*“with course module id\s\'(\d*)\'\.”)* extraería el identificador del recurso: *5014*. Otras funciones, se encargan de procesar los datos de ciertas columnas para generar nuevos datos que conformarán nuevas columnas. Un ejemplo de esto es *create_dynamic_session_id*. Esta función permite estimar la sesión a la que pertenece cada evento. El fichero log generado por *Moodle* no contiene ninguna información acerca del inicio y fin de las sesiones de sus usuarios. Por este motivo, si se quiere conocer ese dato, solamente puede hacerse mediante estimaciones. De estas estimaciones se encarga la función mencionada, que asocia los eventos relativos a un usuario que tienen lugar en un mismo umbral temporal dentro de una misma sesión. Así, esta función añade una nueva columna en la que se relaciona cada evento con un identificador por el par usuario-número de sesión.

Por otro lado, están las funciones que explotan los datos del log una vez ha sido estructurado por el constructor. Estas funciones se encargan de explotar el contenido del log en base a agrupaciones y cálculos con los datos que el log alberga. A modo de ejemplo, se incluye a continuación la implementación de la función *events_per_day_per_resource*. Esta función se encarga de obtener el número de eventos por día generados por todos los usuarios de un curso con un recurso concreto.

```
1.   def events_per_day_per_resource(self, resource) ->pd.DataFrame:
2.       """
3.       Summary line.
4.
5.       Calcula el número de eventos de un recurso concreto por día
         del log.
6.
7.       Parameters
8.       ----------
9.       resource : String
10.          ID del recurso a analizar.
11.
12.      Returns
13.      -------
14.      DataFrame
15.          DataFrame con el nombre del evento, su recurso y las
                 fechas en las que se interactuó con él.
16.
17.      """
18.      df = self.dataframe[[FECHA_HORA, CONTEXTO, ID_RECURSO]]
19.      df = df[df[ID_RECURSO] == resource]
20.      result = df[FECHA_HORA].groupby(df.Hora.dt.strftime('%Y-%m-
             %d')).agg('count')
21.      resultdf = pd.DataFrame(data=result.values,
              index=result.index, columns=[NUM_EVENTOS])
22.      resultdf['Fecha'] = resultdf.index
23.      resultdf['Fecha'] = pd.to_datetime(resultdf['Fecha'])
24.      resultdf.reset_index(drop=True, inplace=True)
25.      return resultdf
```

Listado 4: Función *events_per_day_per_resource*

La función, que puede verse en el *Listado 4*, recibe como parámetro el identificador del recurso a analizar. En primer lugar, se selecciona del *DataFrame* original las columnas con la fecha y la hora, el contexto e identificador del recurso, y con esas columnas, se construye un nuevo *DataFrame* (línea 18). A continuación, se conservan solo las filas del *DataFrame* que recogen eventos relacionados con el recurso pasado como parámetro (línea 19). A continuación, se agrupan las filas del *DataFrame* por la fecha y la hora, y son contadas todas las filas que se producen en el mismo día (línea 21). Tras esto, se crea un *DataFrame* con la fecha y el número de eventos que se generaron con el recurso en ella (línea 21). Para terminar, se estructura el *DataFrame* creado, para que los datos contenidos se organicen en dos columnas, sin que se utilice ninguna de ellas como índice (líneas 22-24), y se retorna como resultado de la función.

La mayoría de funciones de *Maadle* tienen un comportamiento similar al descrito previamente. Su implementación puede consultarse en el repositorio de la herramienta[13].

[13] https://github.com/brian8sal/LibreriaMoodleAnalisis

4.2. *Prez*

Para ilustrar los elementos que se definen en *Prez*, la *Figura 9* muestra la visualización de un indicador de la métrica descrita en la sección previa, relativa a los eventos por día sobre un determinado recurso. En primer lugar, el usuario debe seleccionar el recurso de interés sobre el que quiere obtener la información de los eventos registrados por día. En la figura, el recurso seleccionado es "Carpeta: Documentos Administrativos".

En la gráfica generada para el recurso, el eje de las abscisas representa los diferentes días del curso donde pudieron registrarse elementos, mientras que el eje de las ordenadas muestra el número total de eventos acumulados para cada día.

Figura 9: Gráfico de *Prez* 'Eventos por día por recurso'

En el *Listado 5* puede verse la implementación del gráfico mostrado en la *Figura 9*.

```
1.   html.Div(
2.      children=[
3.          html.Div(
4.              children='Seleccione un recurso ',
5.              style={
6.                    'textAlign': 'left',
7.                    'color': colors['text'],
8.                    'font-size': '20px'},
9.           ),
10.          dcc.Dropdown(
11.             id='resources-dropdown',
12.             options=[{'label': i, 'value': j} for i, j in
13.                 zip(prezz.dataframe_recursos[Maadle.ALIAS],
                     prezz.dataframe_recursos[Maadle.ID_RECURSO])
14.                                 ],
15.              searchable=True,
16.              placeholder="Seleccione a un recurso",
17.              value=prezz.dataframe_recursos[Maadle.ID_RECURSO].
                 unique()[0]
18.          ),
19.          dcc.Graph(id='graph-events-per-day-per-resource')
20.     ], style={'background': colors['background']}
21.  ),
```

Listado 5: Código para representar el resultado de *events_per_day_per_resource*

En primer lugar, se incluyen los elementos a representar en el *layout* del objeto *Dash* creado. En este caso, tenemos un *div* (líneas 1-21) que marca una sección formada por sus tres hijos: un *div* con texto (líneas 3-9), un *dropdown* (líneas 10-18) y un gráfico *Dash* (línea 19).

El primer hijo (líneas 3-9) no será más que un texto que figurará encima del *dropdown*, con un estilo determinado, este hijo muestra el texto "Seleccione un recurso". El segundo hijo es un selector de tipo *dropdown* (líneas 10-18) cuyas etiquetas serán los nombres de los recursos y cuyos valores serán los identificadores correspondientes a tales recursos. Además de poder usar este selector para elegir el recurso de interés, es posible realizar búsquedas de texto (útiles en los casos en los cursos con muchos recursos). El selector por defecto tiene seleccionado uno de los recursos, para que aparezca una gráfica con datos desde el comienzo. El tercer hijo (línea 19) es el gráfico que será actualizado por el mencionado selector, por ello en este punto solamente se define el identificador de dicho gráfico.

Dash permite crear elementos interactivos gestionados mediante *callbacks*. El identificador *'graph-events-per-day-per-resource'* de la gráfica previamente mencionada sirve para indicar dónde colocar la salida de la función que puede verse en el *Listado 6*.

```
1.   @app.callback(
2.       dash.dependencies.Output('graph-events-per-day-per-
         resource', 'figure'),
3.       [dash.dependencies.Input('resources-dropdown', 'value')])
4.    def update_output(value):
5.       dfaux7 = prezz.events_per_day_per_resource(value)
6.       return {
7.           'data': [
8.               {'x': dfaux7[FECHA],
9.                'y': dfaux7[Maadle.NUM_EVENTOS],
10.               'type': 'bar',
11.               'marker': {
12.                   'color': colors['uc_color']
13.               }
14.               },
15.          ],
16.          'layout': {
17.              'title': 'Eventos por día por recurso',
18.              'yaxis': {
19.                  'title': 'Número de Eventos',
20.              },
21.              'xaxis': {
22.                  'title': 'Fecha',
23.              },
24.              'plot_bgcolor': colors['background'],
25.              'paper_bgcolor': colors['background'],
26.              'font': {
27.                  'color': colors['text']
28.              }
29.          },
30.      }
```

Listado 6: Función para controlar la representación del resultado de *events_per_day_per_resource*

A la hora de definir un *callback* en *Dash* han de definirse las entradas y las salidas a las que responder. En este caso, la entrada es el valor que tome el *dropdown* (línea 3), mientras que la salida es el gráfico cuyo identificador (*'graph-events-per-day-per-resource'*) se señalaba previamente (línea 2). Una vez definidas las entradas y salidas se puede definir la función en sí misma, que recibirá el valor del *dropdown*, es decir el identificador del recurso. En este caso, como se comentó previamente, *Prez* se debe encargar únicamente de la representación, de esta forma, recibe los datos que representará ya preparados para su visualización (línea 5). Con los datos a representar en un *DataFrame* lo que se hace es definir su representación, en este caso en forma de gráfico de barras con la fecha en el eje de las abscisas y el número de eventos en el de ordenadas (líneas 7-10). El resto de las líneas hacen referencia a aspectos del estilo, como el color de las barras (líneas 11-13), para lo cual se usaron los colores propios de la Universidad de Cantabria o el título a mostrar en los ejes (líneas 17-23), entre otras cuestiones.

4.3. Interfaz Gráfica

La interfaz gráfica de configuración de *Prez* permite al usuario proporcionar de una manera cómoda tanto el fichero de log a analizar como los ficheros de configuración adicionales que quiera utilizar. Para la implementación de la interfaz gráfica se hizo uso de la librería de *Python Tkinter*. Se decidió separar el proceso de configuración de *Prez* en cuatro pasos. Por cada uno de ellos, hay una ventana implementada en *Tkinter*, las cuales se comentan a continuación.

La primera ventana permite seleccionar el fichero log del curso de *Moodle* a analizar. La segunda, permite crear un nuevo fichero *Excel* de configuración, con los datos por defecto, como, por ejemplo, los nombres de los alumnos que aparecen en el log. La tercera, permite seleccionar un fichero de configuración preexistente. Y, por último, la cuarta, permite al usuario seleccionar la copia de seguridad del curso de *Moodle*.

El diseño de estas ventanas consiste en un texto para guiar al usuario a lo largo de los cuatro pasos y una serie de botones para ir seleccionando ficheros o continuando al siguiente paso. Un ejemplo de esta interfaz puede verse en la *Figura 10*, donde se muestra la cuarta de estas ventanas, la referente a seleccionar una copia de seguridad del curso a analizar.

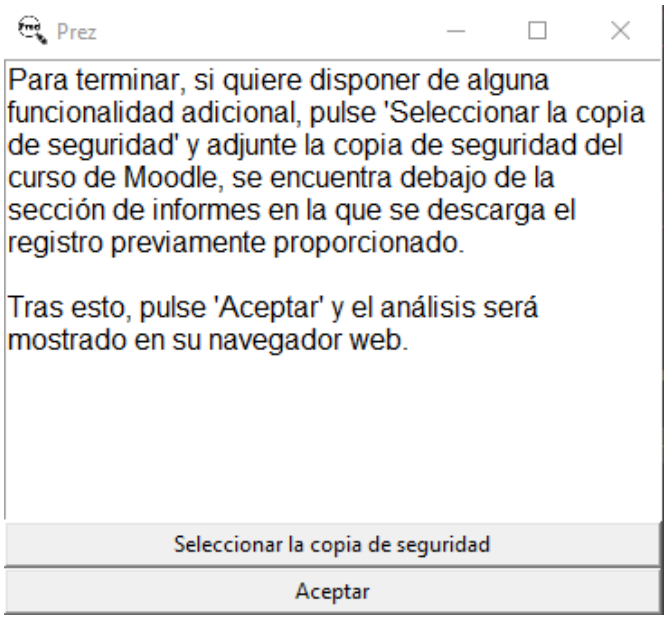

Figura 10: Ventana de la interfaz gráfica de *Prez*

Para ilustrar la implementación, a continuación, en el *Listado 7* se incluye el código relativo a la ventana mostrada en la *Figura 10*.

```
1.  windowBackup = Tk()
2.  windowBackup.title("Prez")
3.  windowBackup.iconbitmap("assets/favicon.ico")
4.
5.  mensaje_backup = Text(windowBackup, wrap=WORD, width=40,
     height=14, font='Helvetica')
6.  mensaje_backup.insert(INSERT,
7.                        "Para terminar, si quiere disponer de
                           alguna funcionalidad adicional, pulse "
8.                        "'Seleccionar la copia de seguridad' y
                           adjunte la copia "
9.                        "de seguridad del curso de Moodle, se
                           encuentra debajo de la sección de
                           informes en la que se "
10.                       "descarga el registro previamente
                           proporcionado. \n \n"
11.                       "Tras esto, pulse 'Aceptar' y el análisis
                           será mostrado en su navegador web.")
12.
13. mensaje_backup.config(state=DISABLED)
14. btn_backup = Button(windowBackup, text="Seleccionar la copia de
     seguridad", command=clicked_btn_backup)
15. btn_accept = Button(windowBackup, text="Aceptar",
     command=clicked_btn_accept)
16. windowBackup.protocol("WM_DELETE_WINDOW", on_closing)
17.
18. mensaje_backup.pack(fill=X)
19. btn_backup.pack(fill=X)
20. btn_accept.pack(fill=X)
21.
22. windowBackup.mainloop()
```

Listado 7: Código de la última ventana de configuración

La tres primeras líneas se encargan de crear la ventana (línea 1) y proporcionarle el título (línea 2), además del icono que figurará al lado de este (línea 3). Tras esto, se crea un texto y se indican detalles como su tamaño y fuente (línea 5). En las siguientes líneas se incluye el texto que indicará al usuario lo que debe hacer (líneas 6-11).

La ventana cuenta con dos botones (líneas 14-15). El primero de ellos causa la ejecución de la función *clicked_btn_button*, que abre el explorador de archivos y permite seleccionar ficheros con formato *MBZ* (el correspondiente a las copias de seguridad de *Moodle*). El segundo desencadena la ejecución de la función *clicked_btn_accept*, lo cual provoca que se cierre la ventana, para posteriormente abrir el navegador web con el análisis realizado. Asimismo, se indica qué debe hacer la interfaz cuando se intenta cerrar la ventana (línea 16). En este caso, se lanza una ventana emergente preguntado al usuario si realmente quiere cancelar el análisis. Para terminar, las últimas instrucciones disponen los elementos comentados en la ventana creada.

4.4. Acceso a Ficheros

Otro aspecto relevante de la implementación del sistema es el tratamiento de los ficheros. El sistema contempla el trato con tres tipos de fichero: (1) el fichero log del curso de *Moodle* a analizar; (2) el fichero *Excel* de configuración creado por *Maadle;* y (3) la copia de seguridad del mencionado curso.

El fichero log del curso de *Moodle* se encuentra en formato *CSV*. Para su tratamiento, simplemente ha de obtenerse un *DataFrame* de ese *CSV*, mediante la función *read_csv* de *Pandas*. Para ello *Maadle* ofrece dos funciones, una en la que se proporciona el nombre del fichero y crea el *DataFrame* si el *CSV* y *Maadle* se encuentran en el mismo directorio. Esta función también puede recibir la ruta completa para crear el *DataFrame*. Y otra, en la que se proporciona el nombre del fichero y el directorio en el que se encuentra. Una vez obtenido el *DataFrame* del fichero log, puede empezar a tratarse con *Pandas* y por lo tanto con *Maadle* y *Prez*.

Por otro lado, el fichero *Excel* de configuración se crea desde el propio *Maadle*. Para ello, se utiliza la clase de *Pandas ExcelWriter*, que permite convertir los *DataFrame* de usuarios y recursos a hojas del fichero *Excel*. Una vez hecho esto, se configura el tamaño de las columnas del fichero de configuración para que se adapten al tamaño de la cadena de texto más larga, haciéndolo más legible para el usuario. Por otro lado, en la hoja de recursos del usuario se oculta la columna del identificador de los recursos, pues no es de interés para el usuario. Este proceso se ve implementado en el código mostrado en el *Listado 8*.

```
1.   if not os.path.isfile(config):
2.       with pd.ExcelWriter(config) as writer:
3.           self.dataframe_usuarios.to_excel(writer,
              sheet_name='Usuarios', index=False)
4.           self.dataframe_recursos.to_excel(writer,
              sheet_name='Recursos', index=False)
5.           writer.sheets['Usuarios'].set_column('A:A',
              self.dataframe_usuarios[NOMBRE_USUARIO].map(len).max()
              )
6.           writer.sheets['Recursos'].set_column('A:C',
              self.dataframe_recursos[CONTEXTO].map(len).max())
7.           writer.sheets['Recursos'].set_column('B:B',
              options={'hidden': True})
```

Listado 8: Código para la creación de los ficheros *Excel* de configuración

Por último, en lo relativo al tratamiento de la copia de seguridad de *Moodle*, esta se encuentra en formato *MBZ*, un formato propio de *Moodle* que sirve para comprimir archivos. Por el momento, de tal copia de seguridad se utiliza solo el fichero *moodle_backup.xml* del que se extraen los temas a los que pertenece cada recurso, permitiendo así distinguir mejor los recursos a la hora de visualizarlos

Para poder extraer los temas a los que pertenece cada recurso primero hay que extraer desde *Maadle* el fichero *XML* mencionado del fichero comprimido *MBZ* y luego hay que procesarlo. Para ello, *Maadle* implementa la función *course_structure* recogida en el *Listado 9*.

```
1.   def course_structure(self, backup):
2.       """
3.       Summary line.
4.
5.       Añade la sección a la que pertenece cada recurso dentro del
         curso de Moodle.
6.
7.       Parameters
8.       ----------
9.       str
10.          Ruta de la copia de seguridad del curso de Moodle en
              formato mbz
11.
12.      Returns
13.      -------
14.      DataFrame
15.          DataFrame con una columna columna con la sección a la
              que pertenece el recurso.
16.
17.      """
18.
         tarfile.open(backup).extract(member='moodle_backup.xml')
19.      tree = ET.parse('moodle_backup.xml')
20.      root = tree.getroot()
21.      df = pd.DataFrame(columns=[SECCION, ID_RECURSO, 'Recurso'])
22.          for activity in
                root.findall('information/contents/activities/activ
                ity'):
23.              df = df.append(
```

```
24.                     pd.Series(
25.                         [activity.find('sectionid').text,
                             activity.find('moduleid').text,
                             activity.find('title').text],
26.                         index=[SECCION, ID_RECURSO, 'Recurso']),
                             ignore_index=True)
27.             id_curso = ''
28.             for activity in
                    root.findall('information/contents/course'):
29.                id_curso = activity.find('courseid').text
30.             dfaux = self.dataframe.copy()
31.             dfaux[SECCION] = id_curso
32.             for activity, section in
                 zip(df[ID_RECURSO],df[SECCION]):
33.                 dfaux.loc[dfaux[ID_RECURSO] == float(activity)] =
                     dfaux.loc[
34.                     dfaux[ID_RECURSO] ==
                         float(activity)].astype(str).replace(id_curso,
                         section)
35.             os.remove("moodle_backup.xml")
36.             return dfaux
```

Listado 9: Función para procesar *moodle_backup.xml* para extraer los temas

Esta función se utiliza en el constructor de *Maadle* para añadir al *DataFrame* del fichero log una columna con el identificador del tema al que pertenece cada recurso. Esto *Prez* lo utiliza para distinguir mejor en los gráficos a los recursos. Como ejemplo, se proporciona el gráfico de la *Figura 11*, que muestra el número de eventos que se han realizado sobre cada recurso a lo largo de todo el curso. Para poder visualizar mejor los recursos por bloques, los recursos se agrupan y ordenan por temas, apareciendo arriba los del tema 1, luego los del tema 2, y así sucesivamente. Además, para poder distinguir mejor los temas como bloques, se va alternando el color en el cual se muestran las barras asociadas a sus recursos. Así, si un tema se muestra en el color verde corporativo de la Universidad de Cantabria, el siguiente se muestra en su color complementario. Por otro lado, como puede verse, las etiquetas de los recursos están ocultas, dado el alto número de recursos, era la mejor opción. Para poder ver el recurso al que pertenece cada elemento del gráfico el usuario deberá colocar el cursor encima del mismo.

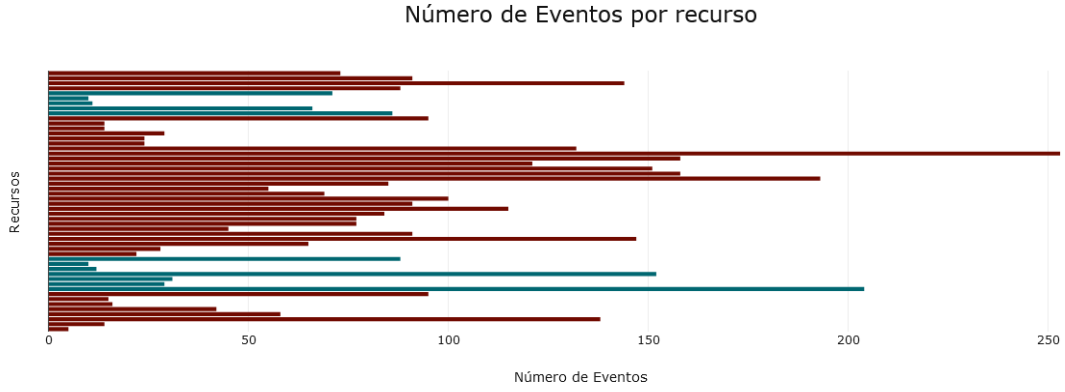

Figura 11: Gráfico de *Prez* 'Número de eventos por recurso'

5. Pruebas y Gestión de la Calidad

5.1. Introducción

En este capítulo se describen las pruebas realizadas, así como la gestión de la calidad seguida durante el desarrollo del proyecto, comentando en ambos casos las herramientas escogidas para ello.

Este capítulo contiene tres secciones. En la primera de ellas se tratan las pruebas unitarias, desarrolladas con el fin de asegurar el correcto funcionamiento de *Maadle*. En la segunda, se tratan las pruebas de aceptación, cuyo objetivo es validar el cumplimiento de los requisitos, permitiendo así comprobar el correcto funcionamiento de *Prez*. Por último, se dedica una sección a comentar cómo se llevó a cabo la gestión de la calidad en el proyecto, indicando la herramienta que se utilizó para ello, así como sus principales utilidades.

5.2. Pruebas Unitarias

En este proyecto se desarrollaron pruebas unitarias para asegurar el correcto comportamiento de las funciones que componen la librería *Maadle,* que se encuentran definidas en una única clase *Python.* Esta, al ser la clase que se encarga del tratamiento de datos, es la única que merece el desarrollo de este tipo de pruebas, pues *Prez* o la interfaz de configuración únicamente representan lo procesado por *Maadle*, siendo más recomendable realizar para tales clases pruebas de aceptación. Además, como interfaz gráfica, *Prez* no tiene muchos elementos interactivos cuya lógica deba verificarse mediante pruebas unitarias.

Antes de comenzar con las pruebas, se hacía necesario tener unos ficheros log y de configuración que pudiesen servir para ejecutarlas. Dado que estos ficheros se iban a alojar en los servidores públicos de *Git*, *Travis* y *SonarCloud*, no podían ser ficheros con datos de alumnos reales, por lo que se optó por crear un fichero log artificial con datos sintéticos, así como ficheros de configuración acordes a esos datos. Los ficheros mencionados tenían que cumplir dos condiciones: tener los suficientes datos como para que las pruebas fuesen realistas, y, aun así, ser manejables para facilitar una correcta implementación de las pruebas. Con este fin se creó el fichero *TestingLog99Rows.csv* con ficheros *Excel* de configuración. Con los elementos asociados creados, se comenzó el desarrollo de las pruebas unitarias.

Las pruebas unitarias diseñadas pueden encontrarse en la clase *Test_Maadle*. Fueron desarrolladas haciendo uso de la librería de *Python, unittest*. Esta clase, *Test_Maadle*, recoge 28 pruebas que aseguran una cobertura de *Maadle* del 89%, correspondiendo el porcentaje restante, principalmente, a la función *course_structure.* La implementación de sus pruebas implicaría la creación de una copia de seguridad de *Moodle* que respetase la privacidad de los datos de estas, de manera que dicha copia pudiese alojar en *Git*, *Travis CI* y *SonarCloud* de manera pública. La creación de una copia de seguridad de *Moodle* con datos sintéticos implica un muy alto coste en horas de trabajo, por lo que esta función se probó de manera manual de forma exclusivamente local,

renunciando por el momento a la implementación de pruebas automatizadas que puedan incorporarse a un entorno de integración continua.

El desarrollo de las pruebas unitarias consistió en implementar una prueba nueva con cada función desarrollada, siempre que fuese posible. Para hacerlo, generalmente se calculaba cuál era el resultado de la función a probar en base a una revisión del fichero log creado y se comparaba con el que generaba la propia función.

Un ejemplo de esto es la prueba de la función *sessions_matrix*, función encargada de generar una matriz de relación con el uso de los recursos en las distintas sesiones del curso de *Moodle*. Para comprobar su correcto funcionamiento, se realizó a mano el cálculo de la matriz, se comprobó que era el deseado y luego se implementó la correspondiente prueba. El código de esta prueba puede verse en la *Figura 12*.

```
1.   def test_sessions_matrix(self):
2.      result_mat = prueba99Rows.sessions_matrix().values.tolist()
3.      expected_mat = [[1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
4.                      [0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
5.                      [0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0],
6.                      [0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0],
7.                      [0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0],
8.                      [0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0],
9.                      [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0],
10.                     [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0]]
11.     self.assertEqual(result_mat, expected_mat)
```

Figura 12: Prueba de *sessions_matrix*

Otro ejemplo distinto es la prueba de la función *list_nonparticipant*, función que debía devolver una lista con los nombres de todos los usuarios de *Moodle* que no hubieran participado en la plataforma. Para comprobar el correcto funcionamiento de la función hubo que tener en cuenta, por la lógica de la misma, los casos límite. En concreto, el caso límite en esta función es que todos hubieran participado, ya que de ser así no habría nada que mostrar. En la función este caso límite se trató indicando que todos habían participado, aspecto que debía tenerse en cuenta en la prueba. La implementación de dicha prueba se encuentra a continuación, en el *Listado 10*.

```
1.   def test_list_nonparticipant(self):
2.       self.assertTrue((prueba99Rows.list_nonparticipant())[
3.                   Maadle.NOMBRE_USUARIO][0] == 'Hollerith, Herman')
4.       self.assertTrue((prueba99Rows.list_nonparticipant())[
5.                   Maadle.NOMBRE_USUARIO][1] == 'ZUSE, KOURAND')
6.       self.assertTrue((prueba99Rows.list_nonparticipant())[
7.                   Maadle.NOMBRE_USUARIO][2] == 'MAUCHLY, JOHN')
8.       self.assertTrue((prueba99Rows.list_nonparticipant())[
9.                   Maadle.NOMBRE_USUARIO][3] == 'LARSON, EARL')
10.      self.assertTrue('TODOS HAN PARTICIPADO' in
                     prueba99RowsTodosUsuarios.list_nonparticipant().
                     columns)
```

Listado 10: Prueba de *list_nonparticipant*

Estas pruebas se ejecutaban localmente con frecuencia, y, por otro lado, al encontrarse en un entorno con integración continua, en este caso *Travis CI*, también se ejecutaban con cada integración, concretamente, cada vez que se registraba un cambio en el sistema de control de versiones. Para lograr esto, hizo falta introducir un *script* en el fichero de configuración *.travis.yml* indicando las librerías necesarias para ejecutar *Maadle*, así como el directorio y nombre de la clase de prueba con indicaciones para recoger también la cobertura.

5.3. Pruebas de Aceptación

Para comprobar el correcto funcionamiento tanto de *Prez* como de su interfaz de configuración, implementada en *PrezGUI*, se realizaron pruebas de aceptación. Estas pruebas se hacen para validar si se satisfacen realmente las necesidades del usuario y se cumplen los requisitos establecidos.

Durante el desarrollo del proyecto, estas pruebas se realizaron conforme al siguiente procedimiento: al comienzo de cada *sprint*, tras seleccionar los elementos que iban a ser tratados en dicho *sprint*, se especificaba cuál debía ser el resultado a cumplir para considerarlos satisfactorios, tal como establece la metodología *Scrum*. Esta especificación se hacía en forma de *pruebas de aceptación*, a modo de ejemplo, la *Figura 13* muestra las pruebas de aceptación pertenecientes a la historia de usuario que concebía la creación de la última ventana de la interfaz de configuración de *Prez* en la que se solicita la copia de seguridad del curso.

1. Ejecución correcta con copia de seguridad.

a. El usuario ejecuta Prez.
b. Se muestra al usuario una ventana en la que se le pide que seleccione el fichero log del curso a analizar y pulse 'Siguiente'.
c. El usuario selecciona el fichero log y pulsa 'Siguiente'.
d. Se muestra al usuario una ventana en la que se le pide que cree un fichero de configuración o pulse 'Saltar este paso'.
e. El usuario pulsa 'Saltar este paso'.
f. Se muestra al usuario una ventana en la que se le pide que seleccione el fichero de configuración del fichero log seleccionado y pulse 'Siguiente'.
g. El usuario selecciona el fichero de configuración y pulsa 'Siguiente'.
h. Se muestra al usuario una ventana en la que se le pide que seleccione la copia de seguridad del curso de Moodle a analizar y pulse 'Aceptar', o de no querer proporcionar la copia de seguridad que pulse directamente 'Aceptar'.
i. El usuario selecciona la copia de seguridad y pulsa 'Aceptar'.
j. Se muestra al usuario el análisis del curso en un navegador web.
k. En las gráficas de barras, los recursos pertenecientes a una sección par utilizan el color corporativo de la UC, los perteneciente a una sección impar lo utilizan del color inverso.

2. Ejecución correcta sin copia de seguridad.

a. El usuario ejecuta Prez.
b. Se muestra al usuario una ventana en la que se le pide que seleccione el fichero log del curso a analizar y pulse 'Siguiente'.
c. El usuario selecciona el fichero log y pulsa 'Siguiente'.
d. Se muestra al usuario una ventana en la que se le pide que cree un fichero de configuración o pulse 'Saltar este paso'.
e. El usuario pulsa 'Saltar este paso'.
f. Se muestra al usuario una ventana en la que se le pide que seleccione el fichero de configuración del fichero log seleccionado y pulse 'Siguiente'.
g. El usuario selecciona el fichero de configuración y pulsa 'Siguiente'.
h. Se muestra al usuario una ventana en la que se le pide que seleccione la copia de seguridad del curso de Moodle a analizar y pulse 'Aceptar', o de no querer proporcionar la copia de seguridad que pulse directamente 'Aceptar'.
i. El usuario pulsa 'Aceptar'.
j. Se muestra al usuario el análisis en un navegador web.
k. En las gráficas de barras todos los recursos utilizan el color corporativo de la UC.

**Las pruebas podrán ser realizadas creando un fichero Excel de configuración nuevo, en lugar de proporcionando uno preexistente, esto no debería afectar a la funcionalidad tratada.*

Figura 13: Pruebas de aceptación de la última ventana de la interfaz de configuración de *Prez*

Al acabar cada *sprint*, tal como establece la metodología *Scrum*, se realizaba una reunión con los directores del proyecto en la que se comprobaba si lo desarrollado de ajustaba a lo especificado en cada prueba de aceptación. De haber cualquier desviación con respeto a lo especificado se generarían *tickets* de mantenimiento que serían trabajados en futuros *sprints*. Este esquema de trabajo permitía asegurar semanalmente que el sistema cumpliese lo especificado.

5.4. Gestión de la Calidad

Para el desarrollo de este Trabajo de Fin de Grado, se creó un proyecto en *SonarCloud* que, sincronizado con *Travis CI*, permitió hacer un control de su calidad. *SonarCloud* proporciona una serie de indicadores sobre el código de los proyectos que analiza. Estos indicadores contemplan la fiabilidad, la seguridad, la mantenibilidad, la cobertura de las pruebas y el código duplicado.

SonarCloud permite establecer ciertos márgenes en esos indicadores para poder controlar la calidad de un proyecto de forma uniforme durante su desarrollo. En este caso, los márgenes establecidos para la calidad del código del sistema fueron los que pueden verse en la *Figura 14*. Estos márgenes fueron definidos por los directores del proyecto, estableciendo la calidad de la mantenibilidad, la fiabilidad y la seguridad en sus valores máximos, el porcentaje de código duplicado en un valor casi despreciable y la cobertura adaptada a la concepción de desarrollar pruebas unitarias para *Maadle* y no para *Prez*.

Metric	Operator	Value	Edit	Delete
Coverage	is less than	60.0%		
Duplicated Lines (%)	is greater than	3.0%		
Maintainability Rating	is worse than	A		
Reliability Rating	is worse than	A		
Security Rating	is worse than	A		

Figura 14: Márgenes de calidad de *SonarCloud*

Tales indicadores se actualizaban con cada nueva integración, y si cualquiera de los indicadores no se encontraba en los márgenes establecidos, antes de terminar el *sprint* se debían tomar las acciones pertinentes para volver a estar dentro de ellos. Aun así, de forma regular, se hacía una revisión de la calidad del código y se implementaban cambios con independencia de que la calidad se considerase ya aceptable.

Un ejemplo de las incidencias que *SonarCloud* permitió detectar, solucionar, y por lo tanto mejorando así la calidad del código, es la existencia de cadenas de caracteres repetidas en numerosas ocasiones para hacer referencia a las distintas columnas del *DataFrame* que representaba el fichero log del curso de *Moodle*. La solución de esta incidencia consistió en la creación de constantes para los nombres de las distintas columnas del *DataFrame*, y la reutilización de estas constantes en las diferentes clases del sistema (*Maadle*, *Prez*, *Test_Maadle*).

Siguiendo con rigor la revisión de los indicadores de *SonarCloud* semanalmente, se logró obtener mantener la calidad del código a los niveles que refleja la *Figura 15*. En ella puede verse como el código final no contiene ningún *bug* que pueda comprometer la fiabilidad del código. Lo mismo sucede con la seguridad, en la que no se registran vulnerabilidades ni puntos calientes. En lo que respecta a la mantenibilidad del código, tan solo se encontró una incidencia, la cual hace referencia a la complejidad cognitiva de la función *sessions_matrix*. Esta función se encarga de calcular la matriz de relación de los distintos recursos en base a su coincidencia en las sesiones, de esta forma, se hace ineludible una complejidad cognitiva alta, teniendo que recorrer todas sesiones del curso y por cada una de ellas todos los recursos, realizando comprobaciones para saber qué recursos se encuentran en cada sesión y con qué otros recursos aparecen. En lo que respecta a la cobertura del código, se registra un 71.1% en lugar de un 89% por incluir el porcentaje de cobertura de *Prez*, su interfaz de configuración, así como el *CSS* que proporciona el estilo a la interfaz del análisis de *Prez*. Por último, el porcentaje de código duplicado se encuentra en su valor óptimo, del 0%.

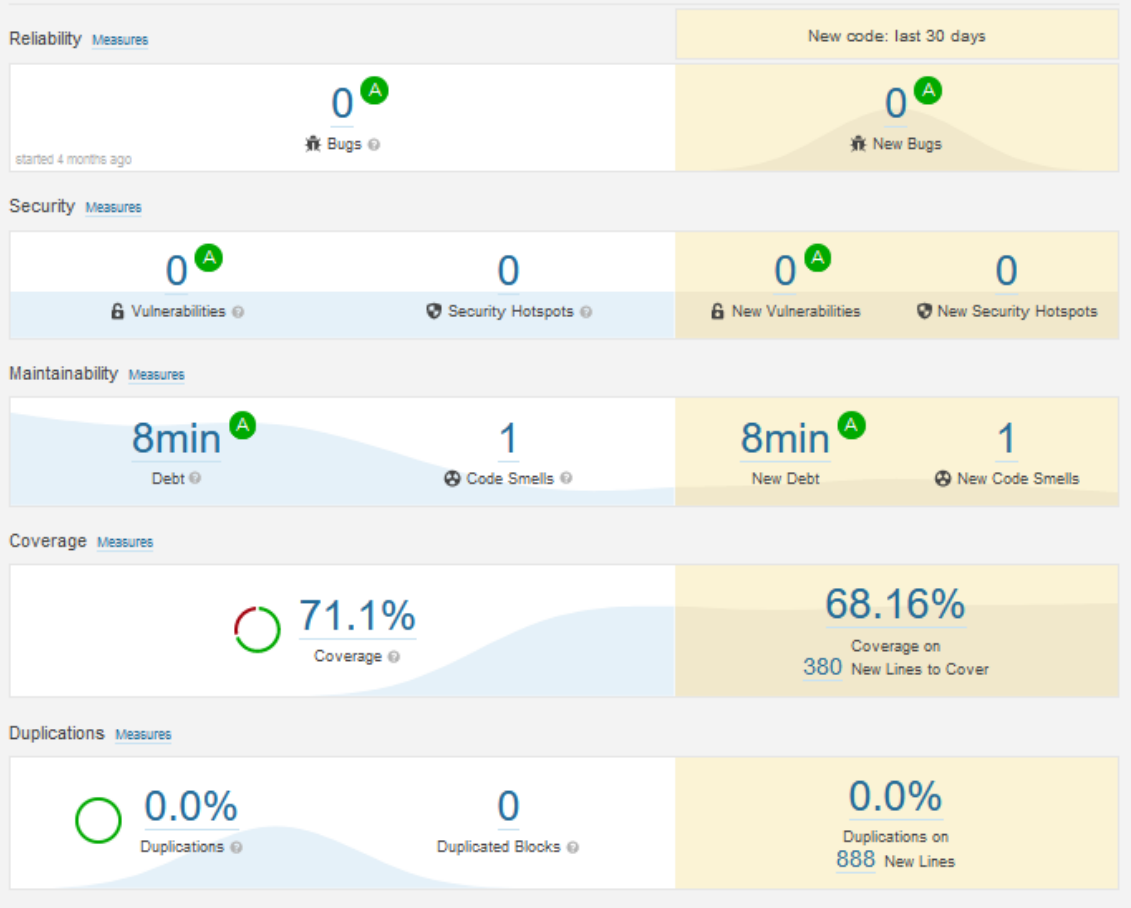

Figura 15: Reporte final de *SonarCloud*

6. Despliegue

A la hora de diseñar el despliegue de la aplicación, se consideraron tres aspectos fundamentales: (1) la complejidad de incluir un nuevo plugin en el *Moodle* oficial de la Universidad de Cantabria, debido a las múltiples restricciones impuestas por los responsables de su gestión; (2) la privacidad de los datos a tratar; y, (3) el perfil de los usuarios a los que va dirigido.

En primer lugar, hay que considerar que, la forma más natural de desplegar el sistema podría parecer en un principio la de desarrollar un plugin de *Moodle*. Sin embargo, la aparente coherencia de desarrollar el sistema como plugin de *Moodle* deja fuera problemáticas como la dificultad burocrática de incluir un plugin de forma institucional o la compatibilidad con herramientas que también sirviesen para alcanzar los otros objetivos planteados en el proyecto. De manera más concreta, de haberse realizado el proyecto como plugin de *Moodle* se tendría que haber desarrollado en *PHP*, y como se ha comentado en apartados anteriores, *Maadle* debía desarrollarse en *Python*, con la librería *Pandas*, con el fin de ser una ayuda para los trabajados de los directores de este Trabajo de Fin de Grado. Adicionalmente, la creación de un plugin de *Moodle* añade una complejidad accidental a la ya de por si intrínseca al proyecto realizado, así como de hacer necesaria la comprobación de que el plugin creado puede coexistir en un curso con el resto de plugins habilitados y disponibles actualmente en la universidad. En

definitiva, por aportar más problemas que soluciones, se decidió optar por no crear un plugin de *Moodle* sino una aplicación independiente.

En segundo lugar, tal como se ha comentado anteriormente, se debía velar por la confidencialidad de los datos procesados por la aplicación, lo que condiciona mucho las opciones de despliegue. Es este el motivo por el cual se decidió no desplegar el sistema en un servidor remoto, al ser siempre mucho más seguro tener sólo los datos de los alumnos de manera local en el computador de cada profesor.

En tercer y último lugar, el despliegue está orientado al perfil de los usuarios a los que va dirigido el sistema. Ciertas formas de despliegue pueden llegar a presentar ciertos problemas, por ejemplo, de usabilidad, que hagan que el usuario final no sepa utilizar el sistema, o no lo haga con comodidad. Un ejemplo de esto sería distribuir la herramienta en forma de ficheros *Python* obligando al usuario a saber ejecutarlos.

Teniendo en cuenta todos estos factores, se decidió que el sistema debía proporcionar una aplicación web que se ejecutase en un servidor local, ejecutado en el computador de cada profesor. Para facilitar el arranque y configuración del servidor web local, y de la aplicación que en él se ejecutaba, se optó por crear una aplicación de escritorio que mostrase al usuario una serie de cuadros de diálogos, o *wizards*, que le permitiesen seleccionar tanto el fichero de log a analizar, como los ficheros de configuración a utilizar. Las secciones 3.5 y 4.3 proporcionan más información de la aplicación desarrollada.

Para crear esta estrategia de despliegue, se hizo uso de dos herramientas de *Python*, *cx_Freeze* y *Waitress*. Por un lado, *cx_Freeze* permite generar ejecutables a partir de ficheros *Python* y, por otro lado, *Waitress* permite trabajar con servidores desde *Python*.

Utilizando estas herramientas, se creó un instalador de *Windows*, en formato *MSI*. Este instalador podría distribuirse dejando al usuario la única responsabilidad de instalarlo en su equipo, sin necesidad de ejecutar ningún comando, teniendo que elegir, únicamente, el directorio de instalación. De esta forma, el usuario puede tener un acceso directo a la aplicación, permitiéndole ejecutarla directamente desde su escritorio.

Una vez instalada la aplicación en el equipo del usuario final, el último paso del despliegue pasa por los manuales. Por la posible complejidad que conlleva para sus usuarios el uso de la aplicación, se realizó un videotutorial en el que se comenta cuáles son los pasos que se han de seguir para hacer uso de la aplicación de forma exitosa.

En este videotutorial se comenta cuál es el proceso de instalación de la aplicación, así como los distintos pasos que tiene que seguir el usuario para poder ver el análisis completo de un curso de *Moodle*, comentando también todas las opciones a las que tiene acceso.

En resumen, el despliegue del sistema desarrollado en este Trabajo de Fin de Grado pasa por la generación de un instalador para la aplicación desarrollada, que se ejecuta como una aplicación web dentro de un servidor local, y cuyo funcionamiento se explica, paso a paso, en un videotutorial que sus futuros usuarios pueden revisar a la hora de hacer uso de la aplicación.

7. Conclusiones y Trabajos Futuros

7.1. Conclusiones

Este Trabajo de Fin de Grado ha presentado *Maadle* y *Prez*, un sistema que en su conjunto busca mejorar de forma continua los procesos de enseñanza-aprendizaje mediante el análisis de los ficheros log de *Moodle*. La dificultad a la hora de interpretar estos ficheros hace que normalmente caigan en desuso, perdiéndose con ellos información que podría ser muy útil para los responsables de los cursos.

El Trabajo Fin de Grado se puede dividir en dos subproyectos relacionados pero independientes. Por un lado, *Maadle* es una librería que busca simplificar el acceso a los datos contenidos en los ficheros log de *Moodle*, ayudando así a facilitar su tratamiento desde el ámbito del análisis de datos o ciencia de datos. Por otro lado, *Prez* es una aplicación que proporciona a los docentes una serie de indicadores que permiten analizar el desarrollo de cierto curso de *Moodle*.

En el desarrollo de este Trabajo de Fin de Grado, se abarcaron todas las fases del ciclo de vida del desarrollo software siguiendo una adaptación de la metodología ágil *Scrum*. Incluyendo una fase previa de formación en las tecnologías a utilizar, una fase de requisitos en la que se analizó el dominio a tratar y se identificaron las fuentes de las que luego se extrajeron los requisitos. También se pasó por una fase en la que se diseñó la arquitectura del sistema. Asimismo, se desarrolló una fase de implementación en la que se vieron inmersos varios aspectos interesantes, como el procesado de datos, la generación de indicadores de rendimiento, la creación de interfaces, el desarrollo de hojas de estilo, entre muchas otras cosas. La fase de implementación se vio acompañada de una fase de pruebas. Y, por último, también se pasó por una fase de despliegue. Todas estas fases, fueron desarrolladas siguiendo un proceso iterativo, en un entorno de trabajo estructurado, conformado por un sistema de control de versiones, integración continua y control de la calidad.

A título personal, este Trabajo de Fin de Grado, fuera de las lecciones aprendidas inherentes al trabajo dentro de un dominio determinado, me ha permitido conocer con mayor fidelidad la forma en la que se deben afrontar proyectos de mayor calado. Durante su desarrollo he podido interiorizar conceptos, que por el momento no conocía en la práctica, así como aprender nuevos con mucha relevancia de cara al futuro laboral.

Ejemplos de lo aprendido hay muchos, he tenido la oportunidad de profundizar en ciertos lenguajes como *Python*, descubriendo la potencia de algunas de sus librerías; como *Pandas* o como *Dash*, revelando por otro lado la importancia de los conceptos elementales, más allá de las particularidades de cada lenguaje. Por otro lado, he podido empezar a conocer los fundamentos del trato con futuros clientes alejados del mundo del software. También he podido comprobar la importancia de tener un entorno de trabajo bien estructurado, que permita la consistencia en el manejo de versiones, la integración continua y la gestión de la calidad, aspectos sin los que un proyecto software difícilmente saldrá adelante con éxito. En resumen, son muchas las lecciones que me llevo de este proyecto, convirtiéndose así, sin duda, en uno de los más enriquecedores en los que me he visto inmerso durante mi vida académica.

7.2. Trabajos Futuros

Dándose por cumplidos los objetivos planteados para este Trabajo de Fin de Grado, quedan varias líneas de trabajo interesantes descubiertas a lo largo de su desarrollo. En esta sección se listan varias de esas líneas que podrían ser trabajadas en un futuro:

- Ampliación de las funcionalidades de los ficheros de configuración. Por el momento, los ficheros de configuración cumplen todos los objetivos para los cuales fueron planteados. Sin embargo, podrían dar más de sí incorporando en ellos nuevas funcionalidades, como la posibilidad de distinguir los recursos en grupos definidos por el usuario para mostrarlos con cierta organización en el reporte de *Prez.*

- Incorporación de las calificaciones en los reportes de *Prez*. La incorporación de las calificaciones dentro de los ficheros de configuración abriría las puertas a multitud de métricas interesantes y variantes nuevas.

- Automatización de la descarga de los ficheros de *Moodle* mediante vías seguras. Actualmente, es el usuario el que debe descargar los ficheros log de *Moodle*, así como las copias de seguridad de los cursos, para ejecutar los análisis del sistema. Sería interesante que esta descarga y posterior utilización de los ficheros se hiciesen desde *Prez* accediendo para ello a *Moodle*.

- Explotación más profunda de las copias de seguridad de *Moodle*. Por el momento, solo se hace uso de uno de los ficheros que proporcionan las copias de seguridad de *Moodle*. Sería interesante investigar más detenidamente los posibles usos de ese fichero concreto y del resto de ficheros contenidos en la copia de seguridad.

- Ampliación de las funciones proporcionadas por *Maadle*. En este proyecto se contemplan un conjunto de funciones de *Maadle* que dan soporte a las métricas hasta el momento implementadas en *Prez*, así como a algunas otras. Una línea interesante de trabajo sería seguir haciendo crecer la librería.

Bibliografía

[1] Alfonso de la Vega Ruiz, D. G. (2015). Towards a DSL for Educational Data Mining. En *Languages, Applications and Technologies* (págs. 79-90). doi:10.1007/978-3-319-27653-3_8

[2] Alfonso de la Vega Ruiz, D. G. (2018). FLANDM: a development framework of domain-specific languages for data mining democratisation. *Computer Languages, Systems and Structures, 54*, 316-336. doi:10.1016/j.cl.2018.07.002

[3] Cohn, M. (2004). *User Stories Applied: For Agile Software Development.* Addison-Wesley Professional.

[4] Demian Abrego Almazán, Y. S. (01 de 2015). Influence of information systems on organizational results. *Contaduría y Administración, 62*(2), 321-338. doi:10.1016/j.cya.2017.03.001

[5] Dmitrij Lipaj, V. D. (4 de 2013). Influence of Information Systems on Business Performance. *Mokslas – Lietuvos Ateitis / Science – Future of Lithuania, 5*(1), 38-45. doi:10.3846/mla.2013.06

[6] Driessen, V. (5 de Marzo de 2020). *nvie.* Obtenido de A succesful fit branching model: https://nvie.com/posts/a-successful-git-branching-model/

[7] Espinosa, M. P. (01 de 2009). *Plataformas de Campus Virtual con Herramientas de Software Libre: Análisis comparativo de la situación actual en las universidades españolas.* Informe del Proyecto EA-2008-0257 de la Secretaría de estado de Universidades e Investigación. Obtenido de https://www.um.es/campusvirtuales/informe.html

[8] Ken Schwaber, J. S. (2017). *The Scrum Guide - The Definitive Guide to Scrum: The Rules of the Game.*

[9] Long, P. D. (2004). Learning Management Systems (LMS). En P. D. Long, *Encyclopedia of distributed learning* (págs. 291-293). SAGE Publications. doi:4135/9781412950596.n99

[10] Pohl, K. (2010). *Requirements Engineering: Fundamentals, Principles, and Techniques.* Springer Publishing Company, Incorporated.

[11] Scott Chacon, B. S. (2014). *Pro Git.* Apress.

[12] Sven F. Crone, S. L. (9 de 2006). The impact of preprocessing on data mining: An evaluation of classifier sensitivity in direct marketing. *European Journal of Operational Research, 173*(3), 781 - 800. doi:10.1016/j.ejor.2005.07.023

[13] Travis-CI. (s.f.). *Travis Documentation*. Obtenido de https://docs.travis-ci.com/

[14] University of Toronto. (2000). *GRL - Goal-oriented Requirement Language*. Obtenido de http://www.cs.toronto.edu/km/GRL/

Apéndice A: Actores y Utilización

Tabla 1: Actor con rol Docente con experiencia en *Moodle*

Rol	Docente con experiencia en *Moodle*.
Tipo	Actor primario.
Edad	Sin especificar.
Habilidades Inf.	Este usuario perteneciente a la docencia tiene a *Moodle* como una herramienta fundamental para su enseñanza y la ha utilizado previamente. De esta forma, puede esperarse de él cierto conocimiento sobre el funcionamiento de *Moodle* y cierto desenvolvimiento con entornos virtuales en general.
Objetivos	El objetivo de este usuario al hacer uso de la herramienta es conocer la forma en la que usan sus alumnos la plataforma: quiénes son los que más la utilizan, en qué periodos de tiempo la plataforma se utiliza más, cuáles son los recursos más utilizados, entre otras cosas.
Modo de Acceso	Accederá a la herramienta en su puesto de trabajo, ya sea en un ordenador portátil o en uno de sobremesa.
Frecuencia	Sin especificar por el momento.
Ejemplo Actor	Como ejemplo de actor se añade a la docente e investigadora Ruth Villalón Molina, docente del Departamento de Educación de la Universidad de Cantabria. Se podrá contactar con ella a través de su email ruth.villalon@unican.es.

Tabla 2: Actor con rol Científico de datos con experiencia en *Moodle.*

Rol	Científico de datos con experiencia en *Moodle*.
Tipo	Actor primario.
Edad	Sin especificar.
Habilidades Inf.	Este usuario tiene amplia experiencia en el tratamiento de datos provenientes de *Moodle*, pudiéndose esperar de él habilidades informáticas altas.
Objetivos	El objetivo de este usuario al hacer uso de la herramienta es tener una serie de funciones disponibles que le faciliten el análisis programático de los ficheros log de *Moodle*.
Modo de Acceso	Accederá a la herramienta en su puesto de trabajo, ya sea en un ordenador portátil o en uno de sobremesa.
Frecuencia	Sin especificar por el momento.
Ejemplo Actor	Como ejemplo de actor se añade a Diego García Saiz, docente e investigador del Departamento de Ingeniería Informática y Electrónica de la Universidad de Cantabria. Se podrá contactar con ella a través de su email diego.garcia@unican.es.