UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Herramienta de visualización de registros de Moodle

Óscar García de Lara Parreño
Tutor: Alejandro Sierra Urrecho

Mayo 2018

Herramienta de visualización de registros de Moodle

AUTOR: Óscar García de Lara Parreño
TUTOR: Alejandro Sierra Urrecho

Dpto. Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Mayo de 2018

Resumen

En esta última década despuntan los COMA, Cursos Online Masivos y Abiertos. Con estos curso se trata de democratizar el conocimiento para que llegue a todo el mundo. Debido a lo anterior, estos cursos se imparten gratuitamente y por internet.

Las universidades presenciales se suman a esta iniciativa y también desarrollan sus propios COMA, siendo estos un extensión de su oferta docente. Pero las universidades no se quedan atrás en el uso de herramientas en línea para complementar las clases presenciales.

Una de estas herramientas en línea es Moodle. Esta plataforma de gestión permite ofrecer a los estudiantes distintas acciones de forma remota. Estas acciones son: obtener material, realizar entregas y examinar a través de cuestionarios de manera sencilla y estructurada. Sin embargo todas estas acciones de los alumnos, en la plataforma, deja rastros que están siendo ignorados y desaprovechados. Estas trazas se almacenan en forma de registros en un fichero.

El propósito de este Trabajo de Fin de Grado es dotar de utilidad a los rastros de los estudiantes. Para ello se desarrolla Graph Log Education, una herramienta que transforma el fichero de registros en una representación gráfica del mismo. Adicionalmente, la herramienta permite el registro de los profesores para la creación de cursos, que son una representación del curso de Moodle.

Graph Log Education proporciona las siguientes graficas de cada curso: eventos/día, eventos/hora, eventos/semana, uso del material, tiempo dedicado/día y tiempo de realización/cuestionario. Además esta información es desglosada para cada estudiante, salvo el uso del material.

La herramienta se desarrolla con el framework Django conjuntamente con una base de datos SQL. Esto permite que la herramienta se use de forma localhost o disponerla en un servidor para su utilización en línea. Adicionalmente tener persistencia de los datos procesados.

Palabras clave

Django, aplicación web, Moodle, logs, visualización.

Abstract

In the last decade, the MOOCs, *Massive Open On-line Course*, have been a highlight. The aim of these courses is to democratize knowledge so that it reaches the whole world. Due to the above, these courses are offered free of charge and over the Internet.

The face-to-face universities join this initiative and also develop their own MOOCs, which is an extension of their teaching offer. But universities are not lagging behind in using online tools to complement face-to-face classes.

One of these online tools is Moodle. This management platform allows to offer students different actions remotely. These actions are: obtaining material, making deliveries and examining through questionnaires in a simple and structured way. However, all these actions of the students, on the platform, leave traces that are being ignored and wasted. These traces are stored as records in a file.

The purpose of this Bachelor Thesis is to make the student's tracks useful. For this purpose, Graph Log Education is developed, a tool that transforms the records file into a graphic representation of it. Additionally, the tool allows the registration of teachers for the creation of courses, which are a representation of the Moodle course.

Graph Log Education provides the following graphs for each course: events/day, events/hour, events/week, material usage, time spent/day and time spent/questionnaire. In addition, this information is broken down for each student, except for the use of the material.

The tool is developed with the Django framework in conjunction with an SQL database. This allows the tool to be used localhost or arranged on a server for online use. Additionally have persistence of the processed data.

Keywords

Django, web application, Moodle, Logs, display.

Agradecimientos

Todos los que van salir mencionados ya saben que soy un hombre de pocas palabras para estas cosas. En primer lugar quería agradecer a mi familia todo el apoyo que me ha dado estos años y en general en toda mi vida.

A mis amigos, Silvia, Patri y Ángel por todos esos momentos vividos en la carrera y que por fin voy a cerrar esta etapa como ya habéis hecho vosotros. También agradecer a Mónica por estos últimos años de amistad y risas. Y no olvidarme de mis amigas Elvira y Gema que llegaron el año pasado y espero que para muchísimo tiempo.

Y por último agradecer a mi tutor, Alejandro, por haberme ofrecido este TFG y haberme ayudado en todo lo posible.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

1 Introducción

En este capítulo se explican los motivos por los cuales se realiza este Trabajo Fin de Grado, lo objetivos propuestos y cómo está estructurada la memoria.

1.1 Motivación

Durante los últimos años han proliferado cursos online masivos y abiertos, COMA o en inglés MOOC, en estos el peso de la docencia cae en la interacción de los alumnos y profesores a través de la plataforma online, ya sea consultando documentos, vídeos; realizando ejercicios o evaluaciones. Del mismo modo, las universidades presenciales también se están apuntando a este tipo de cursos como complemento a sus propios grados y masters que ya ofertan en la actualidad. [1] [2]

No todos los cursos online son COMA, ya que estos se rigen por el principio de que la educación debe ser gratuita y abierta para todo el mundo, por eso la necesidad de que sean online para poder llegar a más gente. También han crecido mucho las páginas que ofertan cursos de pago.

Durante las últimas décadas las universidades han visto el potencial de Internet para la docencia. Esto provoca la inclusión de nuevas tecnologías como son el correo electrónico y las páginas web. Debido a lo anterior se instauran plataformas online para la docencia, como Moodle [3], que se vuelven indispensables para el día a día. Similarmente a las plataformas COMA, estas plataformas online permiten el uso para compartir material, entregar trabajos o realizar cuestionarios a los estudiantes. Estos últimos, al hacer uso de la plataforma, están dejando rastros de su interacción con la plataforma que actualmente se está desaprovechando.

La motivación de este proyecto es el desarrollo de una herramienta de visualización que, a través de un fichero de logs que se puede descargar desde el curso de Moodle, permita visualizar esta información de una forma más clara y gráfica. Lo que ayuda a los profesores, que son el usuario objetivo de la herramienta, a realizar un análisis del propio curso ayudados de toda la información que se pierde al no poder visualizarla de una forma clara.

1.2 Objetivos

Este Trabajo Fin de Grado tiene de objetivo el diseñar y desarrollar Graph Log Education, una herramienta que permite visualizar la información oculta que generan los alumnos de una forma clara. Contará con los siguientes objetivos:

- **Gestión de usuarios:** La herramienta soportará tener más de un usuario, esto se implementará con un apartado de registro y autentificación.

- **Gestión de cursos:** Los cursos se añadirán, actualizarán y eliminarán a petición del usuario.

- **Curso:** Los cursos se repartirán la información en dos partes:

    - **General:** Se mostrará la información teniendo en cuenta a todos los alumnos, y se permitirá comparación con otros cursos.

- **Estudiantes:** Se permitirá mostrar la información de un solo estudiante.

- **Gráficas:** La información que se mostrará del curso será:

  - **Eventos/día:** Los eventos generados por los estudiantes en un día, esto se dará como cálculo global y para cada estudiante.

  - **Eventos/semana:** Los eventos generados por estudiantes en cada semana, se dará como cálculo global y para cada estudiante.

  - **Eventos/hora:** Los eventos generados por estudiantes en las horas del día, se dará como cálculo global y para cada estudiante.

  - **Tiempo dedicado/día:** Se estimará el tiempo que pasa el estudiante en la plataforma en cada día, se dará como cálculo global y para cada estudiante.

  - **Tiempo medio/prueba:** El tiempo que tardan en completar las distintas pruebas de evaluación, se dará la media para el global y el valor para cada estudiante.

  - **Porcentaje de uso/material:** Se mostrará el porcentaje de uso del material proporcionado a los estudiantes.

Como objetivo complementario, Graph Log Education se pondrá a prueba en un caso de uso real, con datos proporcionados de un curso de Programación I Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- El segundo capítulo contiene el Estado del Arte y se divide en un análisis sobre distintas tecnologías en el mercado y las tecnologías empleadas en el proyecto.
- El tercer capítulo muestra las decisiones de diseño que se toman en el proyecto.
- El capítulo cuarto describe la implementación de ese diseño y las decisiones de desarrollo.
- El capítulo quinto muestra una selección de pruebas y el resultado de uso de la herramienta.
- Y para finalizar, en el sexto capítulo se exponen las conclusiones tras finalizar el proyecto y se presentan ideas para una futura hoja de ruta.

2 Estado del arte

En el presente capítulo se analiza Moodle para encontrar qué herramientas proporciona para visualizar los logs. Adicionalmente, un estudio de las tecnologías en el mercado para desarrollar Graph Log Education.

2.1 Análisis a Moodle

Figura 2-1 Logo Moodle

Antes de comenzar el desarrollo se estudia lo que nos ofrece la propia plataforma Moodle para poder visualizar estos logs. Actualmente no ofrece nativamente ninguna forma de visualizar estos logs, lo más parecido a trabajar con los datos que genera la plataforma son una serie de complementos que se pueden instalar a Moodle. La finalidad de este análisis es comprender qué puntos fuertes o débiles tenían estos complementos.

2.1.1 Analytics (Proyect Inspire)

Este complemento está disponible en el núcleo de Moodle desde la versión 3.4[4], pero en la versión actual de Moodle en la UAM, que es la 3.3, está disponible como complemento aparte [5]. No obstante, esta extensión de moodle se analiza desde la perspectiva de un complemento, ya que se considera la forma más realista de compararlo a una herramienta externa.

Analytics es un módulo que permite crear modelos de aprendizaje automático para predecir qué personas del curso tienen más probabilidades de abandonarlo. A priori no es un complemento que nos muestre los datos de forma visual, pero sí que usa los datos de los alumnos para ayudar a mejorar la calidad del curso, ya que permite saber la tasa de abandono y tomar decisiones a consecuencia.

Estas predicciones necesitan tener datos de cursos anteriores, lo cual es una limitación comprensible ya que tiene que generar un modelo y para ello hay que entrenarlo con datos de cursos anteriores. También permite seleccionar el tiempo de las divisiones por trimestre, mensual, etc.

2.1.2 Graylog Logstore

Este otro complemento exporta los logs a una plataforma de terceros, para ello los transforma en un formato GELF. Este formato es compatible con el framework Graylog [6].

Este framework aporta la posibilidad de hacer un análisis visual del curso mostrando los datos entre distintas métricas, asimismo, permite configurar alertas cuando se producen cierto tipo de eventos.

2.1.3 Conclusiones

Después de ver estos dos tipos de complementos, se concluye que actualmente se desaprovechan los datos generados por los estudiantes y que no sirven como retroalimentación para poder mejorar la docencia. No obstante, con las últimas versiones los desarrolladores de Moodle están siendo conscientes de la importancia de estos datos.

Asimismo, el desarrollo de la herramienta dentro de Moodle tiene la ventaja que todo está dentro de la plataforma, pero el inconveniente de estar desarrollada como un complemento. Debido a lo anterior, este complemento debe ser instalado por el administrador del sistema, cerrando la puerta a poder visualizar los datos si el administrador no accede a la instalación. Adicionalmente, otro inconveniente que se encuentra mientras se busca información es la inestabilidad en la propia API interna, que cambia con frecuencia entre versiones.

Para concluir, la mejor forma de construir Graph Log Education es siendo una herramienta externa a Moodle, ya que no se tienen los inconvenientes antes mencionados y los logs son exportables a un fichero.

2.2 Estudio de tecnologías

Después de analizar lo que Moodle nos ofrece y llegar a la conclusión de que lo mejor es crear una herramienta externa, se realiza un estudio de los distintos frameworks o lenguajes de programación del mercado para crear aplicaciones locales o servicios web que permitan crear Graph Log Education, para ver cuál se puede adaptar más al estilo.

2.2.1 Aplicaciones locales

En esta sección se explican los frameworks para aplicaciones locales:

- **Java Swing:** Es la librería estándar de interfaces de java para crear aplicaciones locales con interfaz gráfica (GUI). Es parte del propio JDK, extiende una librería más antigua llamada AWT y sigue el patrón Modelo Vista Controlador (MVC). Una ventaja del propio lenguaje es que es multiplataforma.

- **Windows Forms:** Microsoft dentro de su plataforma .NET tiene una librería que permite crear aplicaciones locales con interfaz gráfica usando distintos lenguajes, donde más destaca C#. Los Windows Forms también siguen el patrón MVC. Tiene la desventaja que están enfocados para su uso en un ordenador con sistema operativo Windows.

- **Electron:** Es un framework [7] open source desarrollado por GitHub, que permite crear aplicaciones multiplataforma usando los estándares web HTML/CSS/JS, esto es posible ya que empaqueta un navegador web (Chromium) y Node.js. El problema de Electron es el uso elevado de recursos del sistema al no ser aplicaciones nativas.

2.2.2 Servicios Web

En esta sección se detallan los frameworks para una aplicación en línea con una arquitectura cliente/servidor:

- **ASP .NET:** Es un framework [8] de desarrollo web creado y comercializado por Microsoft. Soporta varios lenguajes de programación dentro del entorno .NET, donde vuelve a destacar C#.

  Tiene a su disposición la colección Biblioteca de clases (Framework Class Library), que es un conjunto de clases e interfaces que hacen de base para la creación de una aplicación web, aparte de dar soporte para formatos como XML, lectura y escritura, etc. También sigue el patrón MVC, tradicionalmente su uso estaba pensado para usarlo en el S.O. Windows Server, pero las ultimas políticas aperturistas de Microsoft lo hacen compatible con Linux.

- **Spring Framework:** Es en realidad, una colección de frameworks [9] en Java que están modularizados entre sí, lo que permite una mejor gestión de todo el proyecto. Ya que el núcleo del framework se encarga de instanciar cada parte de una forma transparente, dotándolo de un acoplamiento flexible. Está desarrollado por Pivotal Software.

  Según los módulos activos se puede enfocar la aplicación web de distintas formas, dotándola de API REST, servicios web SOAP, autentificación con seguridad, el uso de varias bases de datos SQL o No-SQL, tener servidores Jakarta EE, anteriormente conocidos como Java EE, embebidos como Tomcat. Asimismo, adoptaron el modelo MVC que no estaba disponible en las versiones iniciales.

- **Laravel:** Es un framework [10] desarrollado en PHP, uno de los lenguajes más populares para la creación de páginas web, pero que falla en su baja modularidad. Esto último es lo que quiere corregir este framework creado por Jack McDade y su desarrollo es por parte de la comunidad.

  Permite una sintaxis más clara, aporta un módulo de ruteo entre las distintas páginas web, tiene módulo de autentificación para facilitar su control tanto con usuarios de la misma página o con sistemas de autentificación externos y el desarrollo con la comunicación con las bases de datos a través de objetos.

- **Django:** Para el desarrollo con el lenguaje Python, un lenguaje que destaca en múltiples campos de la informática también tiene presencia para la creación de aplicaciones web. Existen varios frameworks, uno de los más completos y conocidos es Django, que es desarrollado por la Django Software Foundation.

  Django destaca en que cada aplicación web se compone de un proyecto y que este se subdivide en aplicaciones, donde cada una es una función distinta del proyecto, esto le aporta modularidad. También sigue el patrón MVC, esto hace que cada aplicación del proyecto se subdivida en partes que gestionan cada aspecto de este patrón. Debido a lo anterior el modelo de la aplicación se desarrolla directamente sobre Python, también gestionan las rutas de sus recursos propios y tiene sus propios controladores.

2.2.3 Conclusiones

Después de analizar las aplicaciones en línea y las aplicaciones locales, se concluye que ambas tienen ventajas y desventajas para el desarrollo de la herramienta. La aplicación local es más sencilla de cara a la instalación del usuario, también es más centralizada dando más control al usuario sobre sus datos, pero se pierde la posibilidad de usar una base de datos si queremos mantener esta sencillez.

Por otra parte, la aplicación en línea se puede usar también en modo localhost o dar la posibilidad de instalarlo en un servidor, lo que es muy útil hoy en día ya que usamos múltiples dispositivos, la instalación no es mucho más complicada y permite usar una base de datos para no procesar el mismo fichero varias veces. Asimismo, los usuarios no técnicos están más acostumbrados a usar un navegador web para acceder a distintos servicios. Pero también hay que tener en cuenta otros aspectos como es la seguridad de las conexiones.

No obstante, hacer un servicio web permite el uso de Graph Log Education como un servicio de la institución educativa que puede ser ofrecido a cada profesor de la misma. Por tanto, se considera que hacer una aplicación en línea es la mejor opción.

Finalmente, considerando las distintas opciones que se mencionan para realizar la herramienta como servicio web, se decide usar *Django*. Los motivos son que se adecúa muy bien a las necesidades de la herramienta, su alta modularidad, conocimientos previos del desarrollador sobre este framework y, como se ha referido, el lenguaje Python es usado para múltiples situaciones lo que va a ayudar mucho al procesamiento del fichero.

2.3 Otras tecnologías utilizadas

Como anteriormente se menciona, Django va a ser el pilar donde va a sustentarse la herramienta, la versión elegida va a ser la 2.0.3. Pero no va a ser la única librería para construir la herramienta, en las siguientes secciones se describen las librerías adicionales para el cliente y el servidor, como también donde esta alojada la herramienta.

2.3.1 Front-end

- **Bootstrap con Material Design:** Librería open-source desarrollada por Twitter, que se encarga de dar formato a una web con plantillas, colores, animaciones, etc. Estilos predefinidos que ayudan a dar formato a una web adaptativa. La versión que se usa en este proyecto es la 4.0 pero modificando sus plantillas para dotarle un aspecto Material Desing, que es la guía de diseño para aplicaciones Android.

Figura 2-2 Logo Bootstrap

- **jQuery:** Una librería veterana de JavaScript, necesaria para sacar el mayor partido a Bootstrap. Permite manipular el árbol DOM de la página web de una manera mucho más clara, y simplifica el uso de AJAX para dotar de dinamismo a la página web. También cuenta con múltiples complementos para aumentar sus funcionalidades, como crear un selector de opciones múltiple o con búsqueda.

Figura 2-3 Logo jQuery

- **ChartJS:** Una librería open-source, en JavaScript que permite crear gráficas modernas y flexibles de una manera simple en el lado del cliente. Al ser en JS permite una interacción limpia al usuario, posee ocho tipos de gráficas y usa los últimos estándares de HTML5. Es una parte importante de la herramienta, ya que es la encargada de visualizar los datos al usuario.

Figura 2-4 Logo ChartJS

2.3.2 BackEnd

- **Pandas:** Es una librería open-source escrita en Python, el uso de esta librería te permite leer distintos tipos de fichero hacia una estructura de datos, lo que facilita la manipulación y el análisis del mismo. En el caso de este proyecto, la estructura es muy similar a una tabla SQL, lo que facilita usar métodos de la estructura de una forma aparecida a consultas SQL como filter, group by.

- **PostgreSQL:** Para la base de datos he usado una que implementa el lenguaje SQL, es una base de datos relacional open-source con más de 30 años en desarrollo, y he decidido usar la versión 10.

Figura 2-5 Logo PostgreSQL

2.3.3 Repositorio del proyecto

- **GitHub:** Para el control de versiones del proyecto se utiliza el protocolo git en uno de los servicios que lo implementan, que es GitHub. Este servicio permite tener repositorios públicos o privados. También permite tener una wiki en el mismo repositorio.

  En este caso, Graph Log Education está disponible en un repositorio público con licencia MIT para que pueda ser usado, modificado por cualquier persona o institución.

Figura 2-6 Logo GitHub

3 Diseño

Este capítulo contiene el proceso para el desarrollo de la herramienta, también conocido como ciclo de vida. Asimismo, se detallan las decisiones de diseño para la base de datos y el diagrama de casos de uso para enfatizar las posibilidades de utilización de la herramienta por parte del usuario.

3.1 Ciclo de vida

De los distintos ciclos de vida que puede tener un proyecto, se sigue el de Cascada Iterativa. Este ciclo de vida permite volver hacia atrás en cualquier fase del proyecto para un mayor refinamiento del mismo.

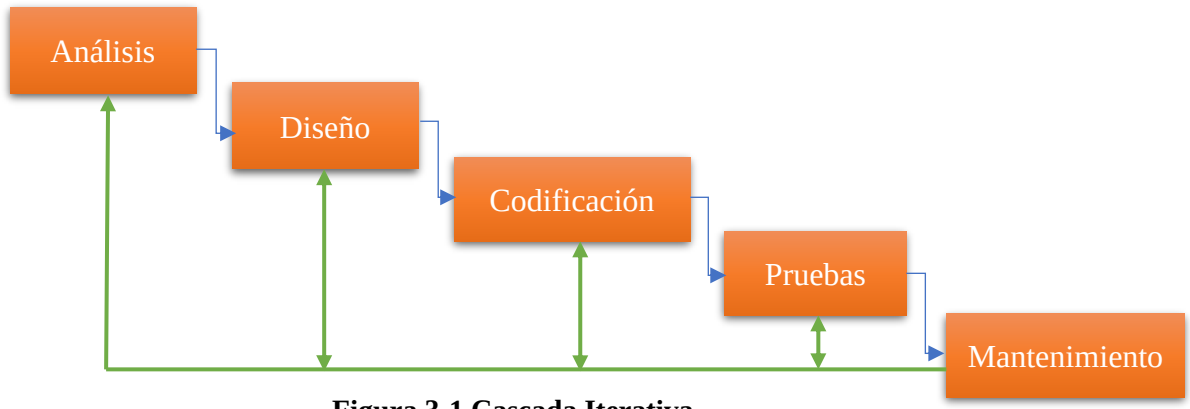


Figura 3-1 Cascada Iterativa

Cascada Iterativa consta de distintas etapas para las tareas del proyecto. Estas etapas son:

1. **Análisis:** En esta fase se definen los requisitos funcionales y no funcionales para cubrir las necesidades de los usuarios finales del proyecto. Así mismo se examina el fichero de log de Moodle para entender qué datos contiene y cuáles pueden ser útiles para poder definir los requisitos correctamente. [Anexo C]

2. **Diseño:** En esta etapa se busca cómo lograr una implementación de los requisitos, para ello se estudian las distintas herramientas posibles, ya mencionadas en el apartado Estado del Arte. También se realiza un diagrama Entidad-Relación de la base de datos, un diagrama de casos de uso y una jerarquía del proyecto.

3. **Codificación:** En esta fase se implementa el diagrama Entidad-Relación en la base de datos y la aplicación web a través de Django.

4. **Pruebas:** En esta etapa se realizan distintas pruebas a lo desarrollado para analizar si el funcionamiento y rendimiento cumplen las directrices de todos los requisitos.

5. **Mantenimiento:** Esta fase forma parte del ciclo de vida, pero no dentro de este proyecto.

3.2 Base de datos

Al estar usando una base de datos relacional, las distintas entidades se relacionan entre sí. Por lo tanto, para poder visualizar esto de una forma clara en las siguientes subsecciones se muestra un diagrama Entidad-Relación y se describe las distintas entidades.

3.2.1 Diagrama Entidad-Relación

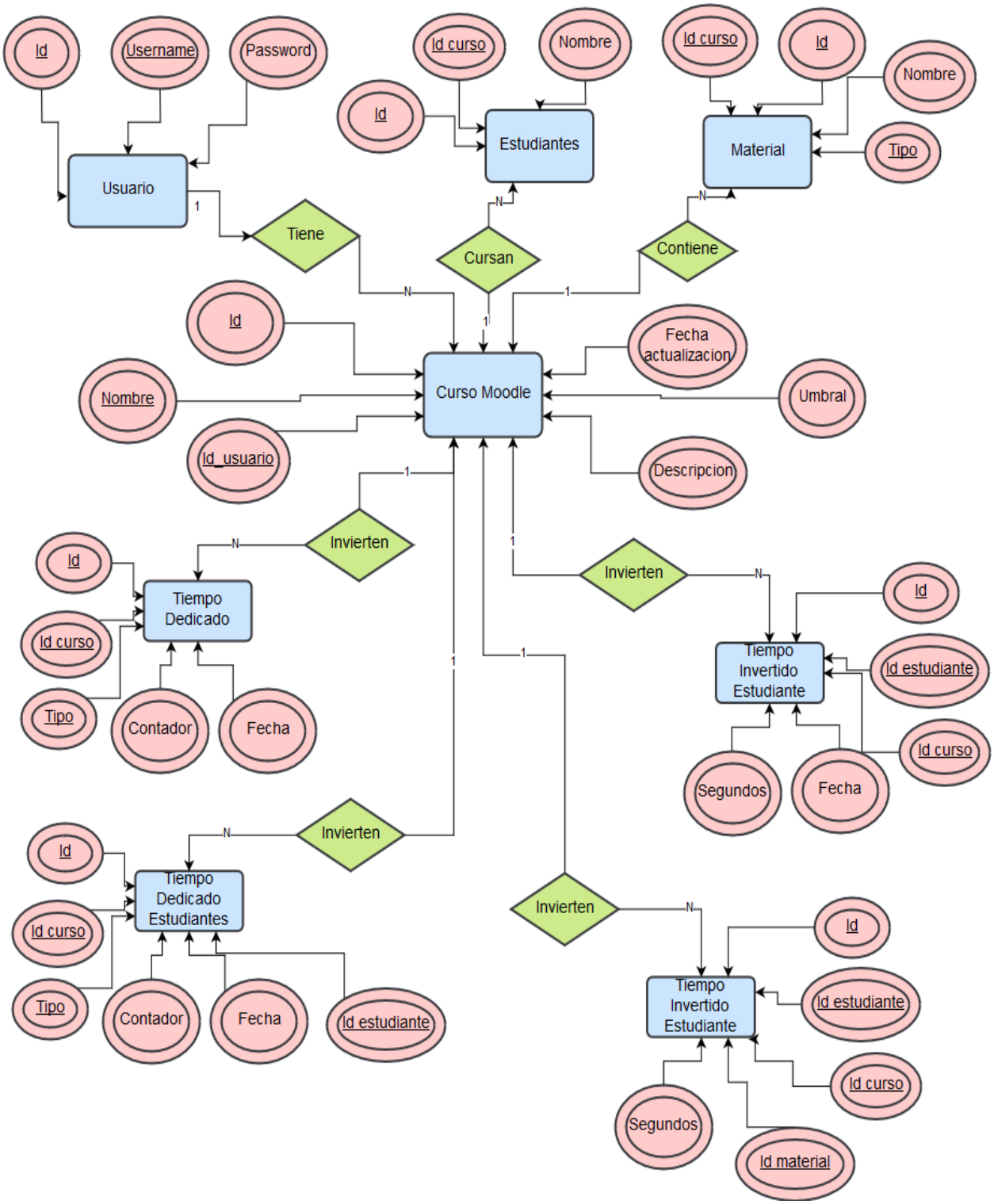


Figura 3-2 Diagrama ER

3.2.2 Entidades y relaciones

Como se puede observar en la Figura 3-2 se muestran las distintas entidades que contiene la base de datos, a continuación se describen:

- **Usuario:** Representa a cada usuario en el sistema, es uno de los componentes base del sistema ya que contiene un username único y su contraseña encriptada para poder acceder al sistema.

- **Curso Moodle:** Un curso representa la información de una asignatura, está relacionado con un usuario, por lo que un mismo usuario no puede tener dos cursos con el mismo nombre. También tiene una descripción, la última fecha que se ha actualizado su información y un valor umbral que son los minutos límite para calcular el tiempo dedicado.

- **Estudiantes:** Se guardan los nombres de los estudiantes de cada curso.

- **Material:** Se guardan los nombres del material de cada curso, podrá ser de dos tipos, Archivos o Cuestionarios. Básicamente se guarda el material al que tienen acceso los estudiantes durante el curso.

- **Tiempo dedicado:** Se guarda el número de eventos producidos por los estudiantes en un curso. También dependiendo del tipo, el campo fecha guarda los datos por día o por hora.

- **Tiempo dedicado estudiante:** Es igual a la entidad anteriormente descrita pero los datos se guardan para cada estudiante.

- **Tiempo invertido estudiante:** La idea de esta entidad es parecida a la de la entidad tiempo dedicado estudiante, pero almacenando los datos solo por día. El dato almacenado en vez del número de eventos que hay del estudiante en ese día, es una estimación del tiempo que ha pasado en la plataforma, para ello se usa el parámetro umbral del curso, si dos eventos seguidos del estudiante pasan con una diferencia menor al umbral, se guarda ese tiempo en segundos.

- **Tiempo invertido material:** Se guarda el tiempo que ha tardado cada estudiante en completar un cuestionario.

3.3 Diagrama de casos de uso

Con el diagrama de casos de uso se muestra visualmente las opciones de utilización de Graph Log Education por el actor. Los actores son el rol de los usuarios al usar la herramienta.

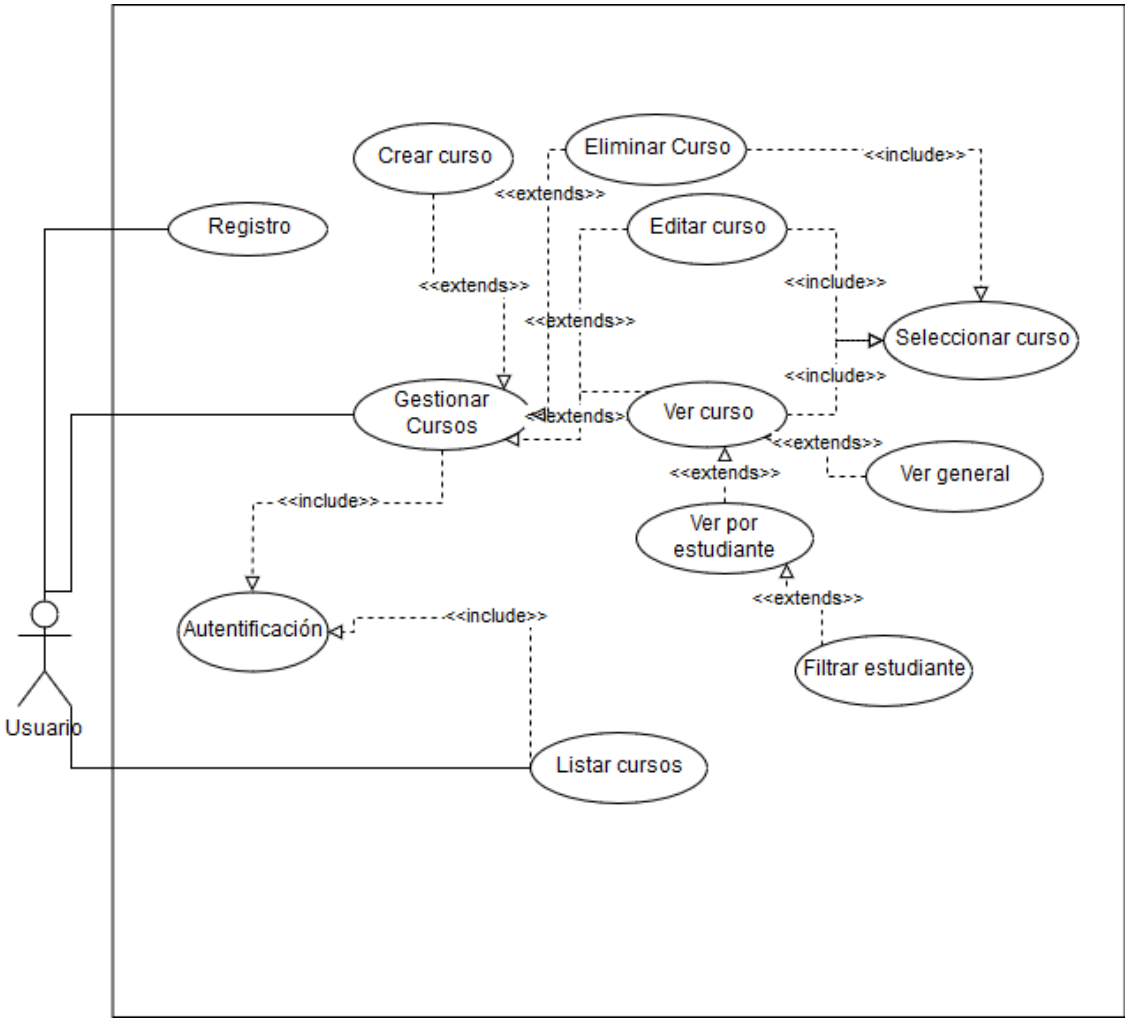


Figura 3-3 Diagrama casos de uso

En la Figura 3-3 se puede observar que la herramienta dispone de un solo actor llamado Usuario, ya que es cualquier persona que tenga en su poder un fichero de logs de Moodle. Asimismo, este actor puede registrarte en la aplicación y para las demás acciones de listar y gestionar los cursos necesita estar autentificado en él sistema.

3.4 Jerarquía del proyecto

Django divide el proyecto en distintas aplicaciones que se encargan de una función específica en la herramienta. A continuación, se muestra visualmente la división en Grap Log Eduacation y se explica lo que contiene cada una de las aplicaciones.

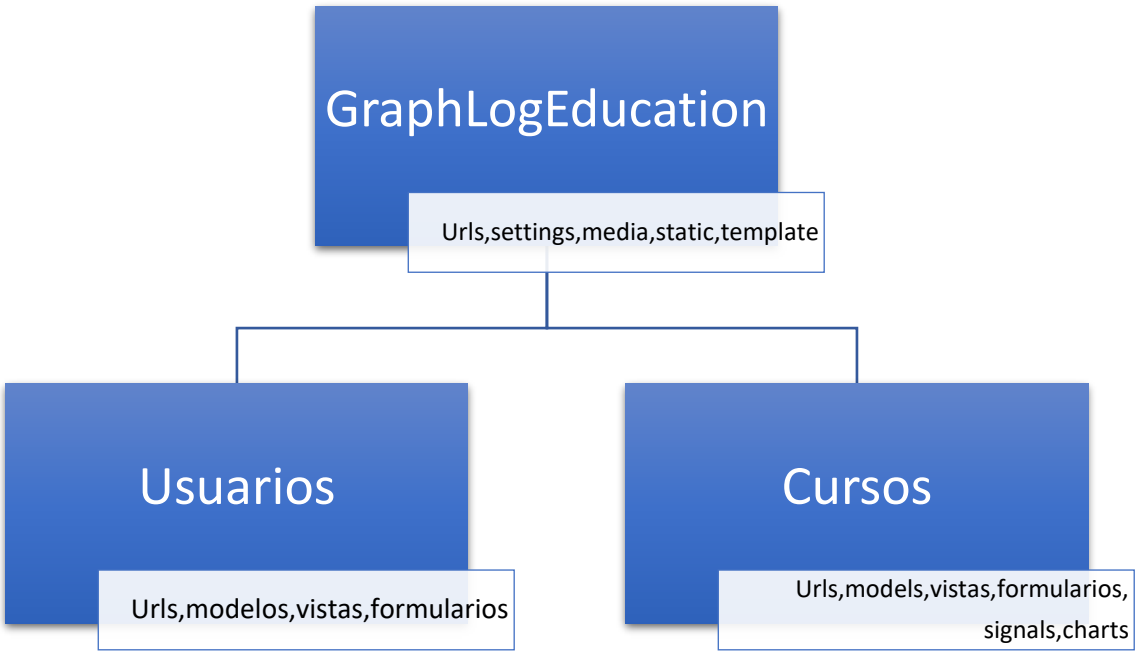


Figura 3-4 Jerarquía del proyecto

El proyecto contiene:

- El enrutamiento general.
- Las opciones de configuración.
- Ficheros CSS, JavaScript, imágenes y archivos que suben los usuarios.
- Las plantillas (template).

Como se aprecia en la Figura 3-4, el proyecto se divide en dos aplicaciones que comparten mayoritariamente los archivos de los modelos, vistas y enrutamiento local. Las dos aplicaciones son:

- **Usuarios:** Se encarga de manejar la gestión de usuarios, como es el registro, la autentificación y cerrar sesión.
- **Cursos:** Se encarga de gestionar los cursos desde su creación, listado, actualización, eliminación. Así mismo contiene el proceso de procesado del fichero y los controladores para generar las gráficas.

4 Desarrollo

En este capítulo se muestra lo relativo a la implementación de la aplicación web, tal como se menciona durante las anteriores secciones el framework elegido es Django. Por lo tanto, se describe la lógica de las distintas aplicaciones que consta el proyecto. Para este desarrollo se elige el IDE Pycharm Professional ya que ofrece una buena integración con el framework.

4.1 MVC, patrón Modelo-Vista-Controlador

Antes de continuar con lo desarrollado se explica un patrón que es muy importante a la hora de implementar Graph Log Education. Este es el patrón MVC que se menciona varias veces durante la sección de Estado del Arte.

Este patrón [11], divide la lógica del negocio y sus datos en tres módulos diferenciados que se comunican entre sí en un orden preestablecido:

Modelo: Se encarga de gestionar la información con la que funciona el programa, para ello acepta o deniega consultas, inserciones, actualizaciones y eliminaciones de la información que tendrán los demás módulos.

Vista: Es la representación de la información que se le muestra al usuario y la forma en la que el usuario puede interactuar con la información.

Controlador: Es el intermediario entre la vista y el modelo. El controlador procesa la interacción del usuario con la información de la vista. Debido a lo anterior dependiendo de la interacción con la vista se encarga de usar el modelo y generar una vista.

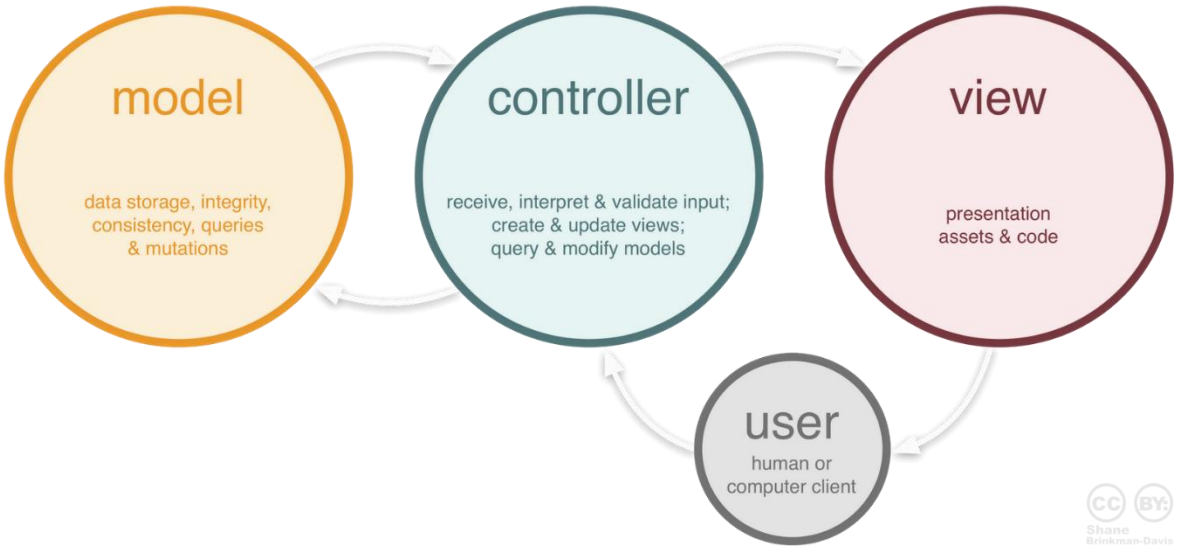


Figura 4-1 Patrón MVC

El uso de este patrón permite una mayor reutilización y legibilidad del código, esto nos da mayor flexibilidad a la hora de programar. El patrón también permite el uso de vistas compuestas, estas vistas contienen otras vistas y cambian a través de una serie de botones.

Tal como se puede ver, este patrón se puede dividir en otros patrones. En definitiva, el patrón está formado por un grupo de patrones como Strategy, Observer, y Composite.

4.1.1 MVC en Django

Una vez definido el patrón, se va a redefinir tal como lo usan los desarrolladores de Django [12] en sus framework. Los desarrolladores cambian los nombres estándar del patrón y puede que cause confusión al continuar la memoria.

- Controlador => Vista (View)
- Vista => Plantilla (Template)

El motivo de estos cambios es la visión del equipo sobre la funcionalidad de cada parte que compone el patrón. Para ellos, Vista es una función callback que define cual es el dato, no cómo se debe mostrar, la diferencia es una forma de apreciación. Y la plantilla es la que se encarga de cómo mostrar este dato.

Si se es más riguroso se puede decir que Django implementa MTV, model template view, pero esto es una forma propia de llamarlo porque la finalidad de MVC sigue siendo la misma.

4.2 URL dispatcher

Django da libertad a la hora de cómo gestionar las urls que se proporcionan a la aplicación. Para la implementación de este proyecto se opta, por dejar que cada aplicación controle sus URLs de forma separada para que así puedan enlazar con sus propias vistas.

```
urlpatterns = [
    path('user/',include('usuarios.urls', namespace="user")),
    path('cursos/', include('cursos.urls', namespace="cursos")),
    path('admin/', admin.site.urls),
    path('', RedirectView.as_view(url=reverse_lazy('cursos:todos'), permanent=False), name='index'),
]
```

Figura 4-2 Ejemplo de url.py

Para que cada aplicación pueda controlar sus urls hay que especificarle un name space único y un fichero urls.py propio. También se toma una de estas urls, donde se muestran todos los cursos, como la de entrada por defecto en Graph Log Education.

4.3 Aplicación Usuarios

En esta sección se explica la implementación de la aplicación Usuarios en la que se subdivide el proyecto, así como los cambios que hay respecto en el diseño inicial pero siempre cumpliendo todos los requisitos.

Para explicar mejor la implementación y hacer una representación práctica del patrón MVC, se va a apoyar en este patrón para modularizar mejor la explicación en las distintas etapas. Para ello las secciones se llaman como el archivo principal que controla estas acciones.

4.3.1 Models

Para implementar el modelo, se opta por implementarlo con la clase AbstractUser, esto da la ventaja de tener ya implementadas ciertas funcionalidades y atributos de esta clase. Estos

atributos son reflejados en la base de datos, como username, contraseña y otros que no se contempla en el diseño de la base de datos como el email, nombre y apellidos.

Aprovechando que se habla de la base de datos en esta sección. Se va a hablar de un aspecto muy importante y que no se es lo suficiente consciente de la importancia de la seguridad de los datos. Una parte vital es como se almacena en base de datos la contraseña.

Para almacenar la contraseña hay varias formas, la más simple y muy insegura es almacenarla tal como llega en texto plano. Otra forma más segura es usar distintos algoritmos de hash, estos algoritmos transforman la cadena de caracteres por otra única para esa cadena. No obstante, este paso solo es posible en una dirección y no es posible volver a sacar la contraseña a través de su hash.

De estos algoritmos hay de varios tipos y se opta por PBKDF2[13] (Password-Based Key Derivation Function 2) este algoritmo utiliza SHA256 junto con un salt, cadena aleatoria, repitiendo este proceso un mínimo de 1000 iteraciones. Una de las razones de optar por este algoritmo es por ya estar implementado en el framework, ya que si se elige hacer uno propio elevaría considerablemente el tiempo de desarrollo y se tendría que someter a todo tipos de pruebas para evitar cualquier vulnerabilidad. Otro de los motivos por elegir este algoritmo y no otro, es por ser un algoritmo que recomienda el NIST (National Institute of Standards and Technology).[14]

```
PASSWORD_HASHERS = [
    'django.contrib.auth.hashers.PBKDF2PasswordHasher',
    'django.contrib.auth.hashers.PBKDF2SHA1PasswordHasher',
]

AUTH_USER_MODEL = 'usuarios.Usuario'
```

Figura 4-3 Opciones de Usuarios

También hay que añadir que este modelo de usuario es puesto por defecto, por si se modifica en un futuro, no se tengan que tocar otras aplicaciones.

4.3.2 Views

Las vistas en esta aplicación se implementan usando las clases de autentificación por defecto que provee Django para la gestión de usuarios. De estas vistas se usan dos, LoginView y LogoutView. LoginView se encarga de autentificar los usuarios, controlando que los datos sean correctos contra la base de datos y creando la sesión para el usuario. Por lo contrario, para cerrar la sesión de los usuarios se usa la vista LogoutView. No obstante, esto no está en el fichero views.py, si no en el de urls.py, que como se menciona anteriormente se encarga de las URLs relacionados con el módulo de Usuarios.

```
urlpatterns = [
    path('login/',auth_views.LoginView.as_view(template_name="usuarios/login.html"),name='login'),
    path('logout/', auth_views.LogoutView.as_view(),name="logout"),
    path('signup/',SignUpView.as_view(),name='signup')
]
```

Figura 4-4 Vistas para un usuario

Lo que sí está en views.py es la vista para crear la clase que permita el registro, para ello se hereda de la clase CreateView, esta clase es genérica para cualquier modelo y está pensada para poder insertar una nueva fila en el modelo de la base de datos a través de un formulario.

```
class SignUpView(CreateView):
    form_class = FormRegister
    success_url = reverse_lazy("user:login")
    template_name ="usuarios/register.html"
```

Figura 4-5 Implementación de CreateView

Asimismo, este formulario por legibilidad del código se separa a otro fichero, forms.py pero sigue siendo parte de la vista. Ya que como bien se dice anteriormente en Django una vista se encarga de definir que se ve, no cómo.

```
class FormRegister(forms.ModelForm):

    val_password = forms.CharField(widget=forms.PasswordInput(),label='Confirma la contraseña')

    class Meta():
        model = Usuario
        fields = ('username','email','password','first_name','last_name')

    def __init__(self,*args,**kargs):
        super(FormRegister,self).__init__(*args,**kargs)
        self.fields['password'].widget = forms.PasswordInput(attrs={'class': 'form-control'})
        for v in self.visible_fields():
            v.field.widget.attrs['class'] = 'form-control'

    def clean(self):
        datos = super(FormRegister,self).clean()
        passw1 = datos.get("password")
        passw2 = datos.get("val_password")

        if passw1 != passw2:
            raise forms.ValidationError({
                'password': forms.ValidationError('No coinciden',code='invalid'),
            })

        return datos

    def save(self, commit=True):
        user = super(FormRegister,self).save(commit)
        user.set_password(user.password)
        if commit:
            user.save()
        return user
```

Figura 4-6 Form de registro

Como se observa en la Figura 4-6 el formulario hereda de la clase ModelForm que te permite elegir de que Modelo quiere ser creado. Esto transforma cada campo de la base de datos en un Widget que serán los inputs de HTML, y que campos quieres mostrar para que el usuario pueda rellenarlos. Asimismo, permite añadir más input como es el caso de otro campo de contraseña para poder validarla.

Por lo tanto, puedes sobrescribir métodos por defecto para hacer nuevas comprobaciones de seguridad como que la contraseña sea la misma en el campo de verificación.

4.3.3 Templete

Para finalizar, el último elemento del patrón MVC, las plantillas. Estas no se guardan en cada aplicación si no en una carpeta general del proyecto. No obstante, esto no hace que no se modularicen, al contrario, permite llegar al punto de tener una plantilla base.

```
{% load staticfiles %}
<!DOCTYPE html>
<html lang="es">
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no" charset="UTF-8">
    <link rel="stylesheet" type="text/css" href="{% static 'css/fontawesome-all.min.css' %}" />
    <link rel="stylesheet" type="text/css" href="{% static 'css/bootstrap.min.css' %}" />
    <link rel="stylesheet" type="text/css" href="{% static 'css/mdb.min.css' %}" />
    <link rel="stylesheet" type="text/css" href="{% static 'css/style.css' %}" />
    <script src="{% static 'js/jquery-3.2.1.min.js' %}"></script>

    {% block jsLoad %}{% endblock %}
    <title>{{ tittle|default:"GraphLogEducation" }}</title>

</head>
<body class="mdb-color blue-grey lighten-4">
<header>
    {% block navBarTop %}{% endblock %}
</header>

<main class="pb-4" role="main">
    {% block contenido %}{% endblock %}
</main>

<footer class="footer page-footer font-small elegant-color pt-2">

    <div class="container-fluid text-center text-md-left">
        <div class="row">

            <div class="col-md-6">
                <h5 class="text-uppercase">Graph Log Education</h5>
                <p>Proyecto de software libre que permite visulizar los logs de Moodle</p>
            </div>
            <div class="col-md-6">
                <h5 class="text-uppercase">Links</h5>
                <ul class="list-unstyled">
                    <li>
                        <a class="btn btn-outline-default btn-sm" href=" https://github.com/yyandrakk/GraphLogEducation"><i class="fab fa-github pr-1"></i> Github</a>
                    </li>
                </ul>
            </div>
        </div>
    </div>
</footer>
</body>
<script src="{% static 'js/popper.min.js' %}"></script>
<script src="{% static 'js/bootstrap.bundle.min.js' %}"></script>
<script src="{% static 'js/mdb.min.js' %}"></script>
{% block jsBody %}{% endblock %}

</html>
```

Figura 4-7 Plantilla base

Como se puede apreciar en la figura 4-7, esta plantilla base incluye los principales archivos de CSS y JavaScript que son necesarios en todas las demás plantillas. Además de tener el pie de página, que es el mismo para todos, también se usa el lenguaje de plantilla de Django que permite reservar huecos para después ser rellenados en las demás plantillas.

Para el caso de las plantillas de Usuarios son parecidas, ya que en ambas se piden datos y se pulsa un botón, así que se va a poner el ejemplo de registrarse.

```
{% extends  "Base/base.html"%}

{% block navBarTop %}

<nav class="navbar navbar-expand-lg bg-dark navbar-dark">
  <a class="navbar-brand" href="{% url "cursos:todos" %}">GraphLog Education</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarDespegableTop" a
    <span class="navbar-toggler-icon"></span>
  </button>

  <div class="collapse navbar-collapse" id="navbarDespegableTop">
    <ul class="navbar-nav mr-auto mt-2 mt-lg-0">
      <li class="nav-item">
        <a class="nav-link" href="{% url "user:login" %}">Autentificación</a>
      </li>
      <li class="nav-item active">
        <a class="nav-link" href="{% url "user:signup" %}">Registro</a>
      </li>
    </ul>
  </div>
</nav>

{% endblock %}

{% block contenido %}
    <section class="mt-3 mdb-color blue-grey lighten-4">
    <div class="container">
    <div class="card mt-4  blue-grey lighten-5">
        <div class="card-header">
            <p class="h4 text-center py-4">Registro</p>
        </div>
        <div class="card-body">
        <form method="POST">
            <div class="md-form">
             <i class="fas fa-user prefix grey-text"></i>
             {{ form.username.label_tag }}

             {{ form.username }}
            </div>
```

Figura 4-8 Parte de la plantilla registro

Como bien se ve en la Figura 4-8, se llama a la plantilla base para poder extenderla, primero extendemos la cabecera, aunque el orden no importa, pero es útil mantener un orden de como va apareciendo en la página. Los {% %} permite indicar una función del lado del servidor que será transformada cuando se crea la página para el cliente.

En la parte de extensión del contenido se crea el Form, para ello se usa {{ }} que indica el uso de variables que proporciona el servidor. Como se menciona en la parte de View se usa el formulario que contiene todos los campos HTML necesarios.

```
{% csrf_token %}
```

Figura 4-9 Token csrf

Por último, destacar el token CSRF [15], es una variable que protege de los ataques de falsificación de peticiones en sitios cruzados (Cross Site Request Forgeries) que explota la confianza del servidor en las peticiones de un usuario.

4.4 Class-based views y Mixin

Antes de continuar explicando la siguiente aplicación, se va a definir algo que sale en la aplicación anterior pero que en Cursos es mucho más importante.

Una vista toma una solicitud y devuelve una respuesta en forma de plantilla, esta vista se puede hacer como una función o como una clase, esta última opción es más recomendada.

Por ello Django proporciona una serie de clases ya predefinidas [16] y en las que se puede aprovechar ya lo implementado o extender nuevas funcionalidades. Para ello vamos a definir para que sirve cada una de las que se usan en este proyecto:

- **CreateView:** Esta clase proporciona la posibilidad de insertar en la base de datos, para ello hay que indicarle el modelo donde se quiere insertar los datos y la plantilla donde se cogen los datos.

- **UpdateView:** Es parecida a la anterior, pero en vez de insertar nuevos datos se actualizan, para ello se le indica también el modelo. Ya que cuando entra en la clase ya sabe cuál es el objeto que tiene que actualizar por el URL Dispatcher.

- **DeleteView:** Se encarga de eliminar los datos de la base de datos, su uso es igual que UpdateView.

- **ListView:** Esta clase es diferente de las anteriores ya que no modifica la base de datos, solamente extrae todos los elementos de un modelo que son pasados a una plantilla, permite sobrescribir la query para coger datos del modelo filtrando el contenido.

- **DetailView:** Se usa después de un ListView ya que permite seleccionar un elemento listado para verlo con más detalle, así mismo necesita el modelo y la plantilla para mostrarlo.

Aparte de las class-based View para crear las vistas, Django permite el uso de Mixin, que son clases que permiten dotar de funcionalidades más concretas a las vistas que se han creado. En el caso de esta herramienta es necesario solamente **LoginRequiredMixin** que obliga a estar autentificado en el sistema para usar una vista.

4.5 Aplicación Cursos

En esta sección se detalla la implementación de la aplicación Cursos. Para ello se sigue el mismo esquema que se usa en la aplicación Usuarios.

4.5.1 Models

Los modelos de la aplicación Cursos controlan la parte del Diagrama de Entidad Relación que están relacionados con Curso Moodle. Por lo tanto, se implementa el mismo Curso Moodle y en las que este modelo es una foreing key de alguna de las otras entidades. Como los modelos se construyen como clases en Django se opta por dividir el Curso Moodle en dos, uno AbstractCurso que es abstracto y otro que hereda que es CursoMoodle.

```python
class AbstractCurso(models.Model):
    profesor = models.ForeignKey(get_user_model(),on_delete = models.CASCADE)
    nombre = models.CharField(max_length=50,null=False,blank=False)
    desc = models.TextField(max_length=200)
    actualizado = models.DateTimeField(auto_now=True)
    slug = models.SlugField()

    class Meta:
        abstract = True


class CursoMoodle(AbstractCurso):
    umbral = models.PositiveSmallIntegerField()
    documento = models.FileField(upload_to=user_directory_path,validators=[FileExtensionValidator(['csv'])])
    procesado = models.BooleanField(default=False)

    class Meta:
        ordering = ["-actualizado"]
        unique_together = ["profesor", "nombre"]
```

Figura 4-10 Modelo del curso

Como se puede apreciar en la Figura 4-10, en AbstractCurso se sacan los campos que debe tener cualquier curso y no solo de Moodle, esto es útil para futuras ampliaciones. También se añade un atributo que es el slug field, este atributo sirve para guardar una ruta única al curso cuando se ingrese la URL.

En Curso Moodle se añade lo que, a priori, solo es específico de Moodle como es el umbral y el documento, que este último se le añade un validador para que solo sea de extensión csv. También se añade un campo boolean que sirve para indicar si el curso se sigue procesando en el hilo.

Como se puede ver no he declarado ningún campo id, esto es porque ese campo se añade automáticamente cuando se transforman de las clases Python a la base de datos.

Para evitar repetir exactamente lo mismo, se va a mostrar un ejemplo de clase que implementa una tabla para guardar la información procesada de los ficheros.

```python
class TiempoDedicadoEstudianteCursoMoodle(models.Model):
    DIA_STD = "DS"
    HORA_STD = "HS"
    TYPES_STD = ((DIA_STD,"Dia estudiante"),(HORA_STD,"Hora estudiante"))
    curso = models.ForeignKey(CursoMoodle, on_delete=models.CASCADE)
    contador = models.IntegerField(null=False,blank=False)
    tipo = models.CharField(max_length=2, choices=TYPES_STD, null=False)
    timestamp = models.DateTimeField()
    estudiante = models.ForeignKey(EstudianteCursoMoodle,on_delete=models.CASCADE,blank = True,null=True)

    class Meta:
        unique_together = ["curso", "timestamp", "tipo", "estudiante"]
```

Figura 4-11 Ejemplo de tabla

Lo primero que llama la atención del modelo de la Figura 4-11, son los tres campos de variables de la clase, que no son atributos en la base de datos. Estos tres indican que valores puede tener el campo tipo lo que permite almacenar distinta información procesada pero que comparten estructura.

4.5.2 Views

Esta sección se enfoca por partes, ya que las vistas son muy amplias en funcionalidades, ya sea gestionando los cursos en sí o las peticiones AJAX para obtención de gráficas. Por tanto, primero se va a ver la gestión de Curso Moodle y después las peticiones AJAX.

4.5.2.1 Curso Moodle

En esta sección se enseña las vistas que se encargan de mostrar los cursos, ya sea como listado o en detalle, su creación y modificación. Todas ellas van a usar el Mixin LoginRequiredMixin. Empecemos por como crear un curso:

```
class addCursoView(LoginRequiredMixin, generic.CreateView):
    template_name = "cursos/addCursos.html"
    form_class = FormCursoMoodle
    success_url = reverse_lazy("cursos:todos")

    def form_valid(self, form):
        self.object=form.save(commit=False)
        self.object.profesor = self.request.user
        self.object.slug = unique_slug_generator(slugify(self.object.nombre))
        self.object.save()
        return super().form_valid(form)

    def get_form_kwargs(self):
        kwargs = super(addCursoView, self).get_form_kwargs()
        kwargs['instance'] = self.request.user
        return kwargs
```

Figura 4-12 Vista de crear curso

Se opta por seguir lo explicado en la sección de las Class-based views pero extendiendo la funcionalidad de CreateView para que guarde también el slug único que se forma en cada curso. También se usa un formulario personalizado tal como se hace en Usuarios para que controle que un mismo usuario no pueda tener dos cursos con el mismo nombre.

No se puede terminar de hablar de la creación de un curso sin hablar de un procedimiento muy importante, que es el procesado del fichero que se lanza después de guardar la información.

```python
@receiver(post_save, sender=CursoMoodle)
def creacion_informacion_curso(sender,instance,created, **kwargs):

    if created:
        ProcesarFicheroThread(instance,False).start()
    else:
        update_fields = kwargs.get("update_fields")
        if update_fields!=None and "documento" in update_fields:
            CursoMoodle.objects.filter(id=instance.id).update(procesado=False)
            instance.refresh_from_db()
            EstudianteCursoMoodle.objects.filter(curso=instance).delete()
            MaterialCursoMoodle.objects.filter(curso=instance).delete()
            TiempoDedicadoCursoMoodle.objects.filter(curso=instance).delete()
            TiempoDedicadoEstudianteCursoMoodle.objects.filter(curso=instance).delete()
            TiempoInvertidoMaterialCursoMoodle.objects.filter(curso=instance).delete()
            TiempoInvertidoEstudianteCursoMoodle.objects.filter(curso=instance).delete()
            ProcesarFicheroThread(instance, False).start()
        elif  update_fields!=None and "umbral" in update_fields:
            CursoMoodle.objects.filter(id=instance.id).update(procesado=False)
            instance.refresh_from_db()
            TiempoInvertidoEstudianteCursoMoodle.objects.filter(curso=instance).delete()
            ProcesarFicheroThread(instance, True).start()
```

Figura 4-13 Signals del curso

El procesado se lanza en un signals, esto es la forma que tiene Django de implementar los triggers de SQL a través de código Python. En este caso, se activa después de crear o actualizar la información en la base de datos. Adicionalmente, se comprueba si es una nueva creación o una actualización de algún campo.

La diferencia de crear o actualizar, es que en la creación se procesa todo y en la actualización va por partes, si se ha cambiado el umbral se borra de la tabla de la entidad Tiempo invertido estudiante y se manda procesar solo esa información. En el caso de actualizar el fichero se borra todo y se procesa como si fuera nuevo el curso.

Por defecto, todo se realiza en el mismo hilo que está usando el usuario, pero por las necesidades del procesado y evitar bloqueos se lanza un nuevo hilo.

Ahora que tenemos un curso se va a ver como se obtiene un listado de un curso.

```python
class indexCursoView(LoginRequiredMixin,generic.ListView):
    template_name = "cursos/listCursos.html"
    context_object_name = 'curso_list'
    models = models.CursoMoodle

    def get_queryset(self):
        return self.models.objects.filter(profesor_id=self.request.user.id)
```

Figura 4-14 Vista de listado de cursos

Tal como se explica en la sección anterior, para este caso se usa ListView pero modificando la consulta para que solo devuelva los cursos del usuario autentificado.

Asimismo, se puede actualizar o eliminar estos cursos, para lo que se necesita implementar dos vistas que usen los Class-based views. En el caso de actualizar se hereda de UpdateView.

```
class updateCursoView(LoginRequiredMixin, generic.UpdateView):
    template_name = "cursos/updateCurso.html"
    model = models.CursoMoodle
    form_class = FormUpdateCMoodle
    context_object_name = 'curso'
    success_url = reverse_lazy("cursos:todos")

    def get_initial(self):
        initial = super(updateCursoView, self).get_initial()
        initial['desc'] = self.object.desc
        initial['umbral'] = self.object.umbral
        return initial

    def form_valid(self, form):
        if(form.has_changed() and form.is_valid()):
            form.instance = CursoMoodle.objects.get(id=self.object.id)
            form.save(commit=True)

        return HttpResponseRedirect(self.get_success_url())

    def user_test(self,request,slug):
        return models.CursoMoodle.objects.filter(slug=slug,profesor_id=request.user.id).exists()

    def get_context_data(self, **kwargs):
        context = super(updateCursoView,self).get_context_data()
        return context

    def dispatch(self, request, *args, **kwargs):
        if not self.user_test(request,kwargs.get('slug','')):
            return redirect("cursos:todos")
        return super().dispatch(
            request, *args, **kwargs)
```

Figura 4-15 Vista de actualizar curso

Se extiende la funcionalidad para incorporar una comprobación de seguridad, que verifica que el usuario que llega a la vista es ciertamente el usuario que es el creador del curso. También se obtiene información del curso para pasárselo a la plantilla correspondiente.

Para eliminar un curso, la vista es muy parecida a la de actualizar ya que incorpora la misma comprobación de seguridad, pero se usa la clase DeleteView, también para confirmar la eliminación del curso se introduce una comprobación para evitar errores irreparables. Esta comprobación es un formulario con un campo que pide el nombre del curso para asegurarse que el usuario no se equivoca.

Mostrar un curso en detalle tiene dos partes, una de la que se explica a continuación que es obtener la información del curso y otra es detallada en otra sección tal como se explica al principio de la sección y son las peticiones AJAX.

```python
class detailCursoView(LoginRequiredMixin, generic.DetailView):
    template_name = "cursos/detailCurso.html"
    model = models.CursoMoodle
    context_object_name = 'curso'

    def get_context_data(self, **kwargs):
        context = super(detailCursoView, self).get_context_data()
        curso = context['curso']
        context['estudiantes'] = EstudianteCursoMoodle.objects.filter(curso__id=curso.id).order_by('nombre')
        context['otrosCursos'] = CursoMoodle.objects.exclude(id=curso.id).exclude(procesado=False) \
            .filter(profesor__id=curso.profesor_id).order_by('nombre')
        return context

    def user_test(self, request, slug):
        return models.CursoMoodle.objects.filter(slug=slug, profesor_id=request.user.id).exists()

    def dispatch(self, request, *args, **kwargs):
        if not self.user_test(request, kwargs.get('slug', '')):
            return redirect("cursos:todos")
        return super().dispatch(
            request, *args, **kwargs)
```

Figura 4-16 Vista del detalle del curso

Para mostrar con detalle se usa DetailView pero se extiende la información que se proporciona a la plantilla añadiendo los estudiantes que presenta el curso y otros cursos que tiene el usuario, necesarios para evitar peticiones de más en la plantilla.

4.5.2.2 AJAX

Las peticiones AJAX permiten actualizar dinámicamente el contenido de las páginas sin tener que recargarlas, muy útil para generar la información de las gráficas de la herramienta.

Para ello, se opta en usar la otra forma de poder crear una vista, las funciones, ya que la vista de una petición AJAX está muy contenida en funcionalidad y crearla con una clase no aportará nada. Pero para que sea más modular se separa de estas funciones la obtención de la información del modelo.

```python
def ajaxCharts(request):
    if not request.user.is_authenticated:
        jsonr = {'authenticated': False}
        return JsonResponse(jsonr, mimetype='application/json')

    charts = []
    id = request.GET.get('id', None)
    idsGN = request.GET.get('idsGN', None)
    if id != None and CursoMoodle.objects.filter(pk=id,profesor_id=request.user.id).exists():
        # Grafica dia/#evento
        charts.append(graficaTiempo(id, idsGN=idsGN))
        # Grafica semana/#evento
        charts.append(graficaTiempoSemanal(id, idsGN=idsGN))
        # Grafica hora/#evento
        charts.append(graficaTiempoHora(id))
        # Grafica media contextos
        charts.append(graficaTiempoMedioContexto(id))
        # Grafica tiempo invertido
        charts.append(graficaTiempoInvertido(id))
        #Grafica de uso de archivos
        charts.append(graficaUsoArchivos(id))


    return JsonResponse(charts, safe=False)
```

Ilustración 4-1 Ejemplo de función AJAX

Las dos funciones se encargan de gráficas distintas, una de los datos de forma general del curso y la otra de los datos por un estudiante especifico. En ambas se hace una serie de comprobaciones de seguridad para evitar peticiones malintencionadas, se comprueba que el usuario este autentificado y que la información que quiere sacar es realmente suya.

La otra parte de la vista son las distintas funciones que generan los datos necesarios para generar una gráfica de los distintos modelos. Para ello, se les proporciona el id del curso y dependiendo de la función también, los ids de otros cursos o el id de un estudiante. Estas funciones devuelven un JSON con la estructura que necesita ChartJS, así se puede añadir más gráficas sin preocuparse de controlar en la plantilla.

Debido a lo anterior se puede decir que se rompe el patrón MVC. Ya que se dice como pintar la gráfica al pasar los datos en el formato que necesita una librería, pero en ningún momento se obliga a usarla. Sin embargo al ser un JSON se puede extraer los datos y la plantilla usarlos de otro modo.

4.5.3 Templete

Las plantillas de Curso siguen la misma filosofía de diseño que las plantillas de Usuario, todas extienden de una plantilla base. Como ya se explicó en la sección de Templete de Usuarios cómo se diseña las plantillas con formularios se opta por hablar de las plantillas de ListView y DetailView ya que son las dos que difieren al no tener formularios. Y así ver las decisiones de diseño en estas.

En vez de código que era la manera más clara para ver cómo se estructuran las plantillas, en esta sección se muestra una selección del resultado final para que quede más clara la explicación.

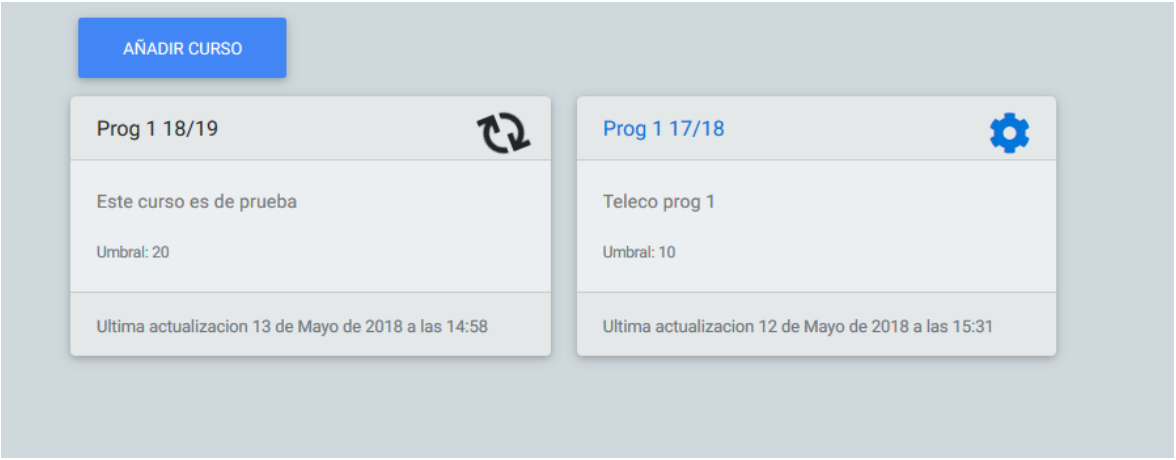


Figura 4-17 Tarjetas de listado de cursos

En ListView se opta por mostrar todos los cursos del usuario, se hayan procesado o no, esto se debe por el atributo procesado del modelo. Cuando es falso los enlaces se desactivan y aparece una animación que simula que se está cargando. O en el caso que sea verdadero se activa el enlace en el nombre que lleva a la vista detallada o un botón de opciones que lleva a la pantalla de editar o eliminar el curso.

También hay que añadir que la pantalla se actualiza automáticamente para mostrar el curso ya procesado y evitar que el usuario tenga que estar refrescando la página para ver si termina de procesarse.

En el caso de DetailView tiene una primera parte que carga la pantalla y otra que carga todas las gráficas que llegan de la petición AJAX. Si se recuerda el patrón MVC permite que las vistas, en nuestro caso las plantillas, sean compuestas que es justamente este caso.

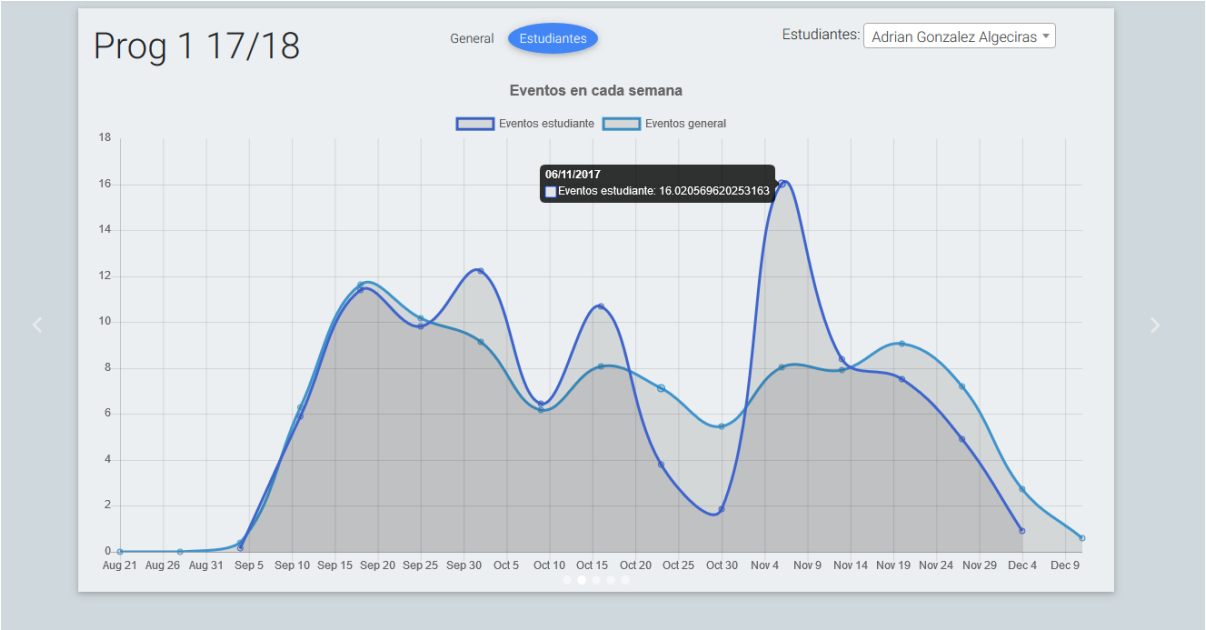


Figura 4-18 Curso en detalle

El curso se divide en dos zonas, una cabecera y otra la de la gráfica. En la primera cuenta con el nombre, dos botones para alternar las vistas y actualizar la segunda zona, y con un selector. Este selector especifica la información que se va a mostrar pudiendo ser un alumno en este caso, o en la parte general otro curso.

La segunda zona son las distintas gráficas que se muestran de una en una, esto se ha preferido así para que se adapte mejor a distintos tamaños de pantalla ya que el objetivo es mostrar datos y estos se deben visualizarse de la forma más óptima posible. Para ir cambiando de gráfica hay dos botones en los laterales para ir pasando entre gráficas.

Para poder lanzar estas peticiones dinámicamente al servidor sin refrescar el cliente se ha usado JQuery para poder lanzar las peticiones AJAX. También mencionar que las gráficas permiten mostrar en detalle pasando el ratón por encima o quitar información si se pulsa sobre la leyenda.

5 Integración, pruebas y resultados

En esta sección se van a detallar por una parte las pruebas realizadas para comprobar el correcto funcionamiento de Graph Log Education y por otra parte mostrar la aplicación en un caso real.

5.1 Pruebas

Para las pruebas se opta por seguir el esquema de caja negra. Estas pruebas están pensadas para no tener en cuenta la implementación sino una entrada que tenga una salida esperada. También el ciclo de vida permite no esperar a terminar toda la codificación para poder realizar las pruebas y se opta por ir haciéndolas según se programan partes de cada aplicación.

Asimismo, debido a la gran similitud entre las pruebas se va a optar por mostrar las más relevantes.

5.1.1 Prueba de la aplicación Usuario

Para la parte de la aplicación Usuario se hace una prueba inicial del modelo implementado desde un script de Python para comprobar que en la base de datos se almacena los datos de forma correcta, sobre todo el campo contraseña que debe estar cifrado.

	id	password	last_login	is_superuser	username
1	1	pbkdf2_sha256$100000$Cn3svis8pM9Q$jl2zEIOQ3TXjxXSp2vVF6R0…	2018-05-12 15:31:41.923737	☐	manager

Figura 5-1 Prueba del modelo de Usuarios

Una vez comprobado que se almacena correctamente todos los datos, se realiza la comprobación de los formularios de registro y de autentificación.

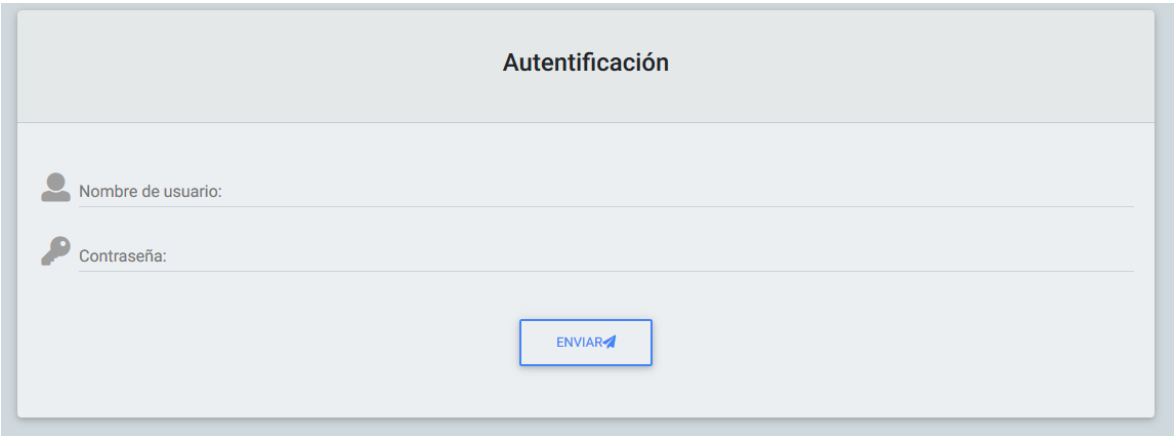


Figura 5-2 Pantalla de autentificación

Para ello en la Figura 5-2 se muestran dos campos que permiten escribir texto, en el primer caso se prueba a insertar un usuario y contraseña inválidos para comprobar si bloquea el acceso.

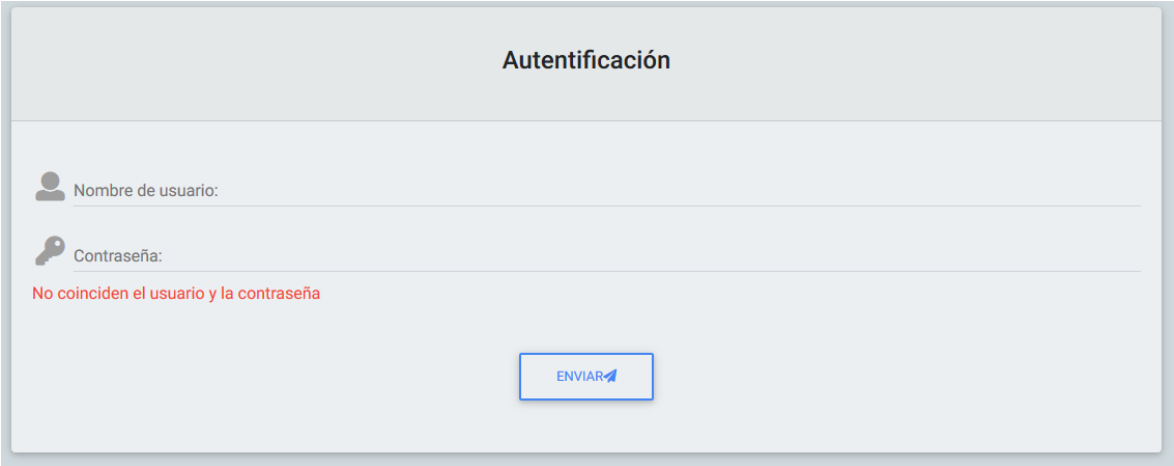
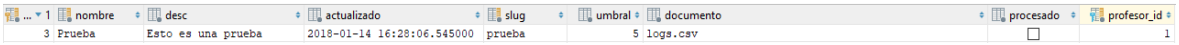


Figura 5-3 Pantalla de autentificación con error

Como se observa la Figura 5-3 se introducen dos datos al azar y la pantalla sigue siendo la misma, pero con un mensaje que indica que existe un error. En el caso de introducir correctamente los datos se redirige a una página de prueba para simular que entra a la lista de cursos ya que la aplicación de cursos no estaba implementada.

5.1.2 Prueba de la aplicación Cursos

Las pruebas de Cursos se realizan por etapas según se desarrolla la aplicación, esto permite evitar efectos colaterales. Lo primero que se prueba como en el caso de Usuarios es la base de datos, para ello se volvió a crear un script Python que crea un objeto en la base de datos para comprobar que tuviera todos los campos correctamente.



... 1	nombre	desc	actualizado	slug	umbral	documento	procesado	profesor_id
3	Prueba	Esto es una prueba	2018-01-14 16:28:06.545000	prueba	5	logs.csv	☐	1

Figura 5-4 Prueba del modelo de CursoMoodle

Una vez comprobado los datos en la base datos, se procede a probar la pantalla de listados de cursos. Esta pantalla como se menciona en la sección 4.5.2.1 utiliza un Mixin en la vista que obliga a estar autentificado. Por lo consiguiente se accede a la aplicación con el formulario de la Figura 5-2 introduciendo datos válidos y se comprueba que se redirecciona a la página de listado de cursos. Una vez en esta página se guarda la URL que da acceso al listado "https://192.168.0.185:8000/cursos/" y se cierra sesión para volver a introducir la URL y comprobar que se redirecciona a la pantalla de autentificación.

Para proseguir con las pruebas se analiza la creación de un curso, pero sin la implementación del signal, esto permite comprobar que el fichero se sube correctamente.

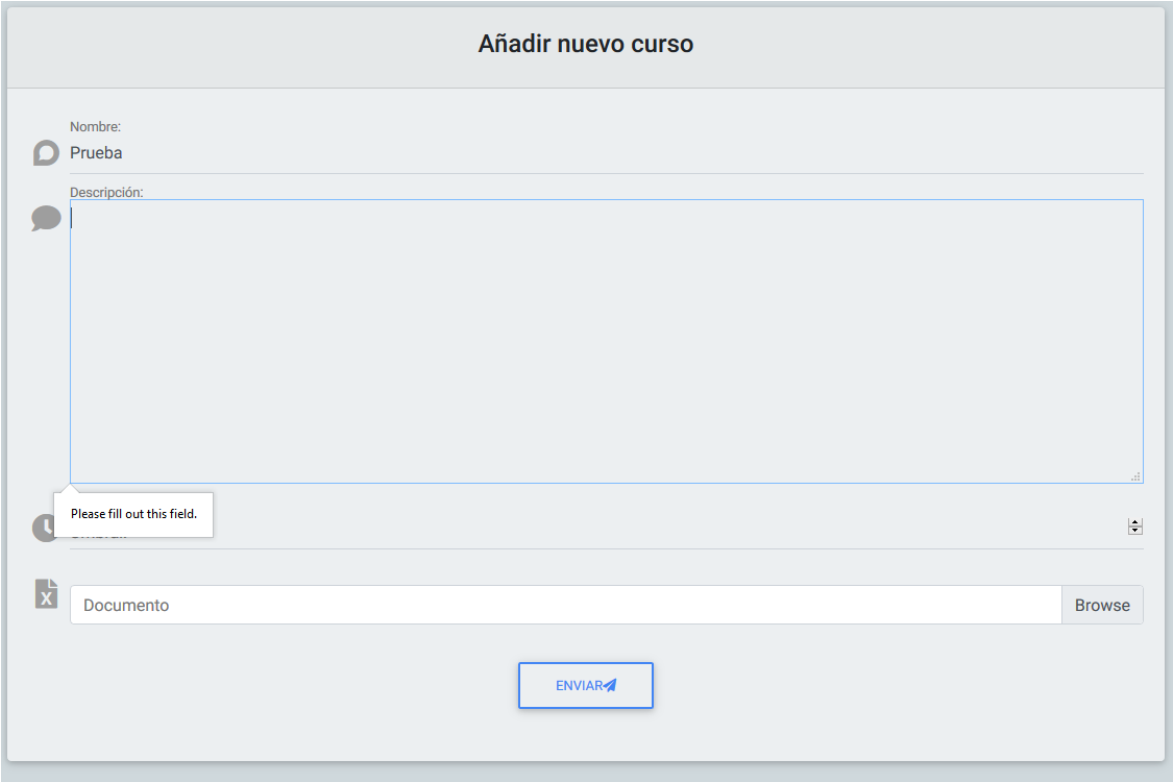


Figura 5-5 Prueba error campo vacio

La primera prueba realizada al formulario, fueron los errores que se controlan en el lado del cliente, esto se realiza con el último estándar de HTML5. Por tanto, el error sale según la configuración local del navegador del usuario. Como se puede apreciar en la Figura 5-5 está en inglés y marca los campos vacíos que son requeridos o si solo aceptan números como el campo umbral.

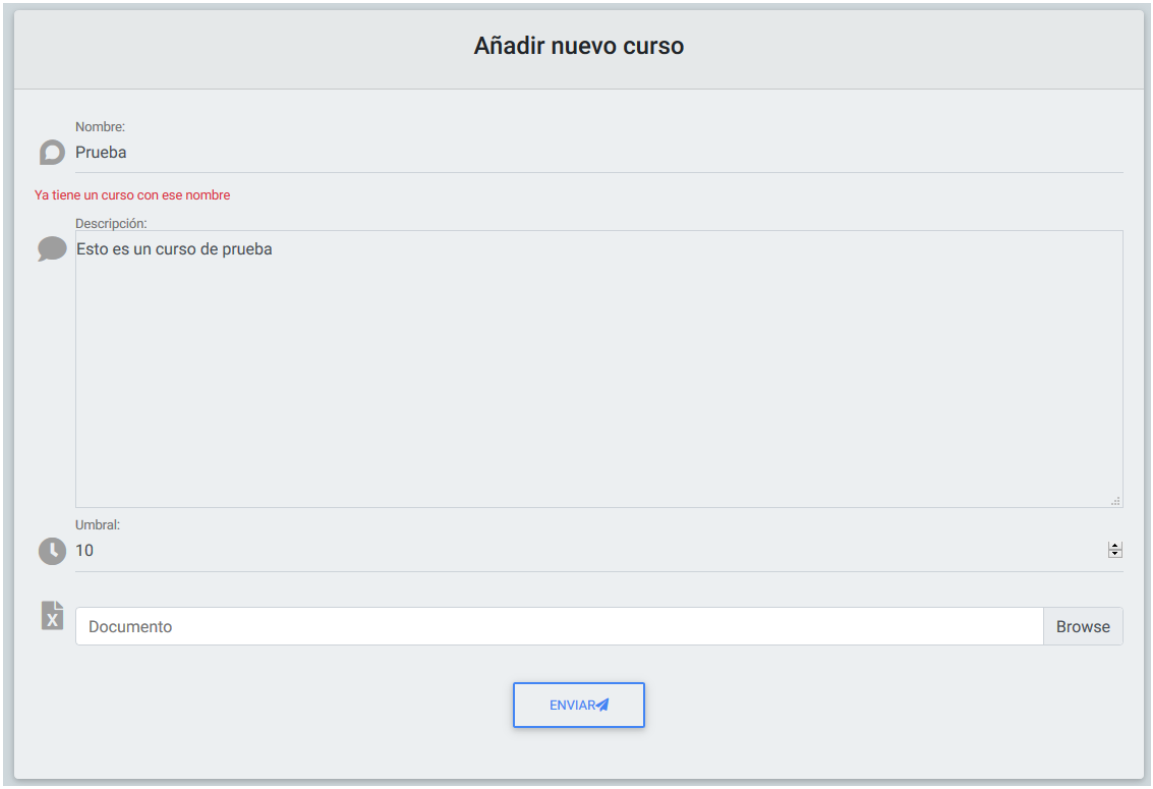


Figura 5-6 Prueba de un curso existente

La segunda prueba se realiza para comprobar los errores que devuelve el servidor. En el caso que se muestra en la Figura 5-6 se crea un curso con un nombre repetido para el usuario. Debido a lo anterior el servidor devuelve el error que ya existe un curso con ese nombre en el campo correspondiente. Así mismo se comprueba si el servidor devuelve el error cuando se intenta subir un fichero que no sea CSV.

Una vez comprobado que el formulario funciona y se inserta los datos correctamente en la base de datos, se prueba la función signals que procesa el fichero. Esto se divide en dos partes: una unitaria que procesa el fichero de forma local con script para comprobar si las distintas consultas que se lanzan con Pandas devuelven el valor esperado; la segunda parte es una prueba de integración que activa el signals cuando se guarda un nuevo curso en la base de datos y se comprueba que los datos de la base de datos son los mismos que los devueltos con el script.

```
['7.3 Makefile.', 'Calificaciones (después de la prueba 5)',
 Reserva dinámica de memoria.', 'La función strtok', '7.4 Dis
 do-while.', 'Argumentos de main', '5.6 Punteros y tablas.',
 información.', '1.1 Edición, compilación y enlazado de un pr
 igualdad.', '3.6 Tablas de estructuras.', 'Grupo A', '3.5 Es
 Funciones con argumentos.', '5.1 Funciones sin argumentos.',
 lenguaje.']

Process finished with exit code 0
```

Figura 5-7 Extracto del script

48	7.1 Archivos de cabecera.	AR	1
33	7.2 Proyectos con más de un archivo.	AR	1
8	7.3 Makefile.	AR	1
50	7.4 Diseño descendente.	AR	1
28	Argumentos de main	AR	1
35	Calificaciones	AR	1
10	Calificaciones (después de la cuarta prueba)	AR	1
13	Calificaciones (después de la prueba 5)	AR	1

Figura 5-8 Extracto de la DB

También se realizan pruebas para comprobar la correcta generación de las gráficas. Para ello, se opta por seguir la misma estrategia, hacer comparaciones cruzadas de lo que se muestra en la pantalla con los datos que se almacenan en la base de datos.

5.2 Resultados

El objetivo de esta sección es mostrar el potencial de la herramienta y el uso que se le puede dar como complemento a un análisis de un curso. Esto quiere decir que en esta sección no se analiza el curso, ya que esa tarea solo pueden realizarla satisfactoriamente profesores que han impartido esa asignatura.

Los datos a visualizar corresponden al curso de Programación I del Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación. Primero se va a empezar comentando la visión general del curso. Para ponerse en contexto, esta asignatura solo tiene turno de mañana y las clases de teoría son los lunes y martes, las prácticas son miércoles y jueves. Los datos extraídos terminan a principios de diciembre lo que es positivo para ver la utilidad de la herramienta no es solo para el final, sino también durante el desarrollo del curso.

5.2.1 Resultados generales

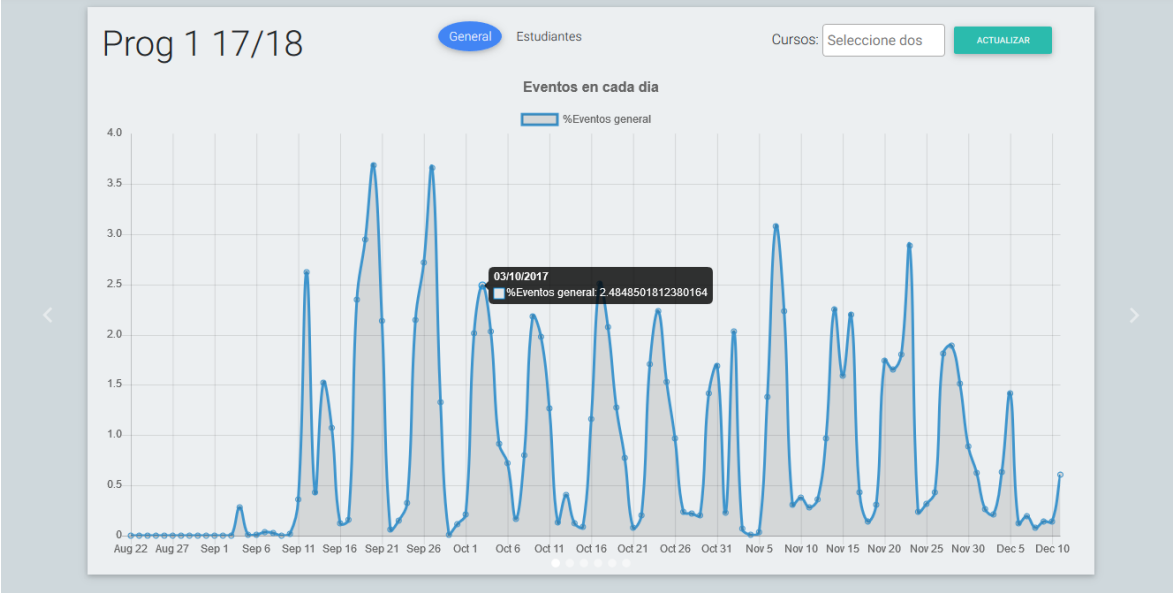


Figura 5-9 Gráfica de eventos por día

La Figura 5-9 muestra los datos por día. Estos indican el porcentaje de eventos que se producen cada día, el 3 de octubre corresponde a un martes lo que tiene sentido teniendo en cuenta el día que se imparte la asignatura, con esta gráfica se puede llegar a la conclusión que los estudiantes tienden a dedicar más tiempo en horario lectivo.

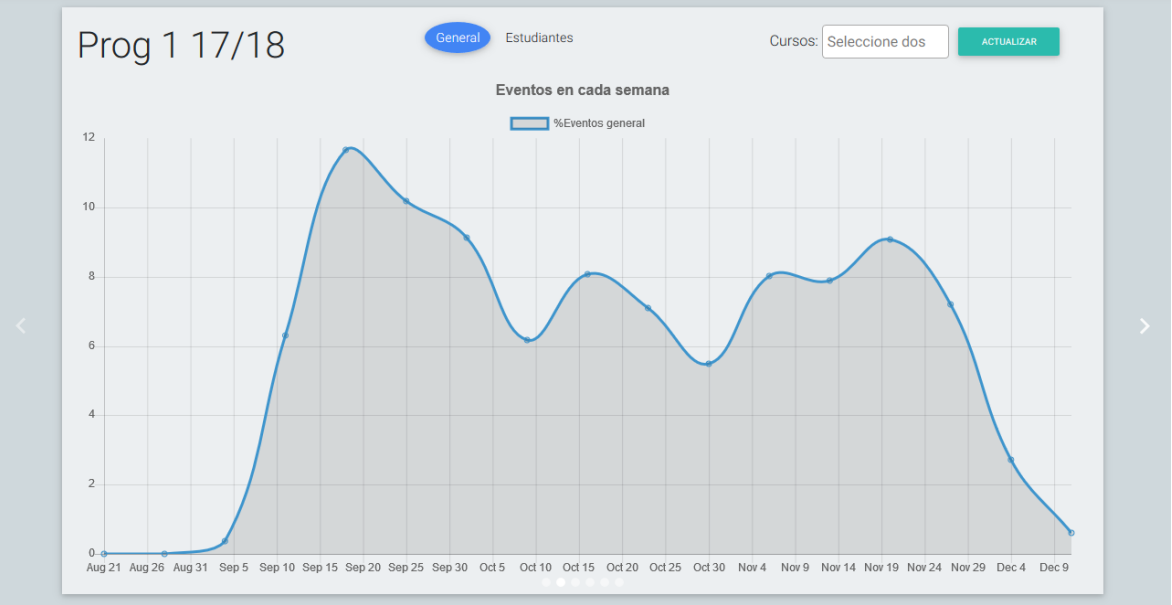


Figura 5-10 Gráfica de eventos semanal

Con la gráfica de la Figura 5-10 se visualiza el porcentaje de eventos por semana. Como se puede apreciar se muestra una tendencia descendente, pero termina estabilizándose. Estando los picos al principio pueden indicar la novedad de la asignatura y, como es natural, va descendiendo ya sea por carga de trabajo de la asignatura no necesite tanta dedicación. Pero cuando se produce un mínimo en la gráfica puede indicar una mayor carga de otras asignaturas que provoca que los estudiantes dediquen más tiempo a las otras asignaturas. Debido a lo anterior, se produce un pico de eventos la semana siguiente para compensar la falta de dedicación.

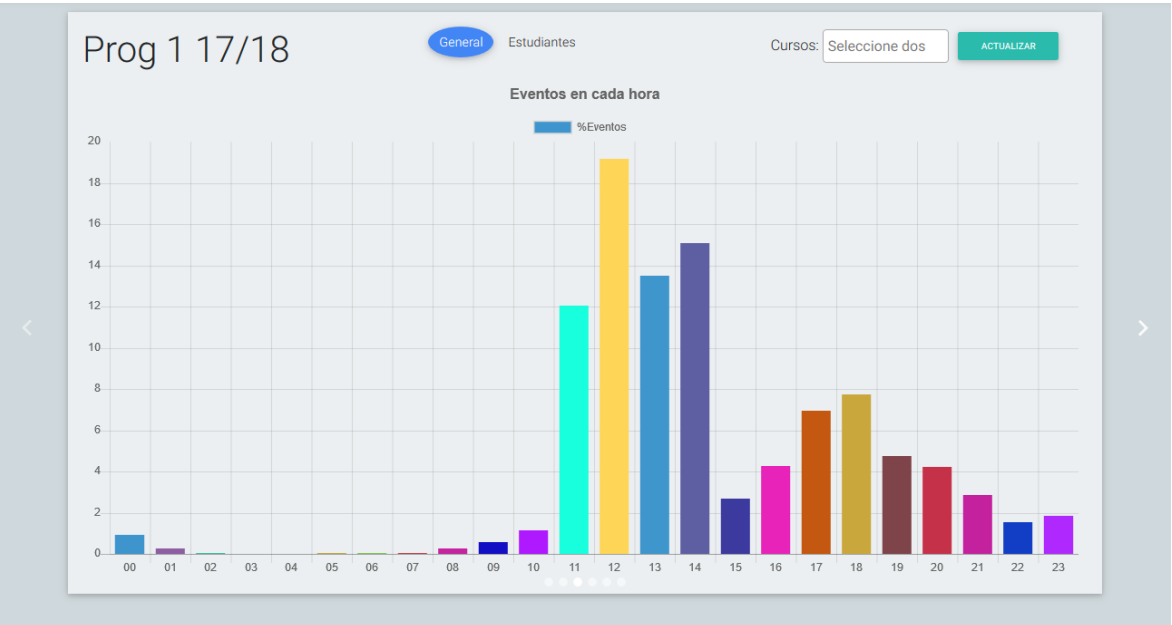


Figura 5-11 Gráfica de eventos por horas

La Figura 5-11 corresponde a la gráfica de evento por hora. Se observa que las cuatro horas principales de estudio están dentro del horario de clase, y que fuera de ella también se trabaja en la asignatura mayoritariamente por las tardes, pero también de madrugada que es lo que más sorprende siendo clases del turno de mañana.

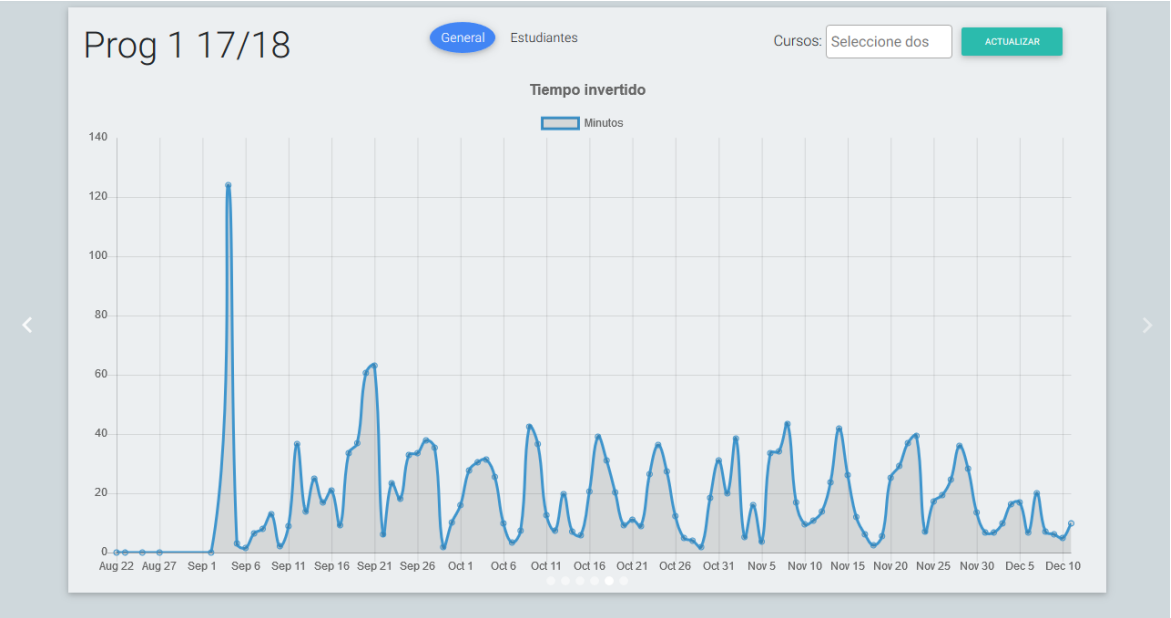


Figura 5-12 Gráfica de tiempo invertido

Con la gráfica de la Figura 5-12 no se trata de sacar el número de eventos que hay en un día, si no cuantos son suficientemente seguidos para indicar que se encuentra activo en la plataforma. El umbral para considerar que dos eventos son seguidos se establece para esta prueba a los diez minutos. De esta gráfica se puede llegar a la conclusión que el trabajo real sigue la tendencia de la Figura 5-9 al ser mayor en los días de clase, pero con un descenso en los otros días no es tan pronunciado.

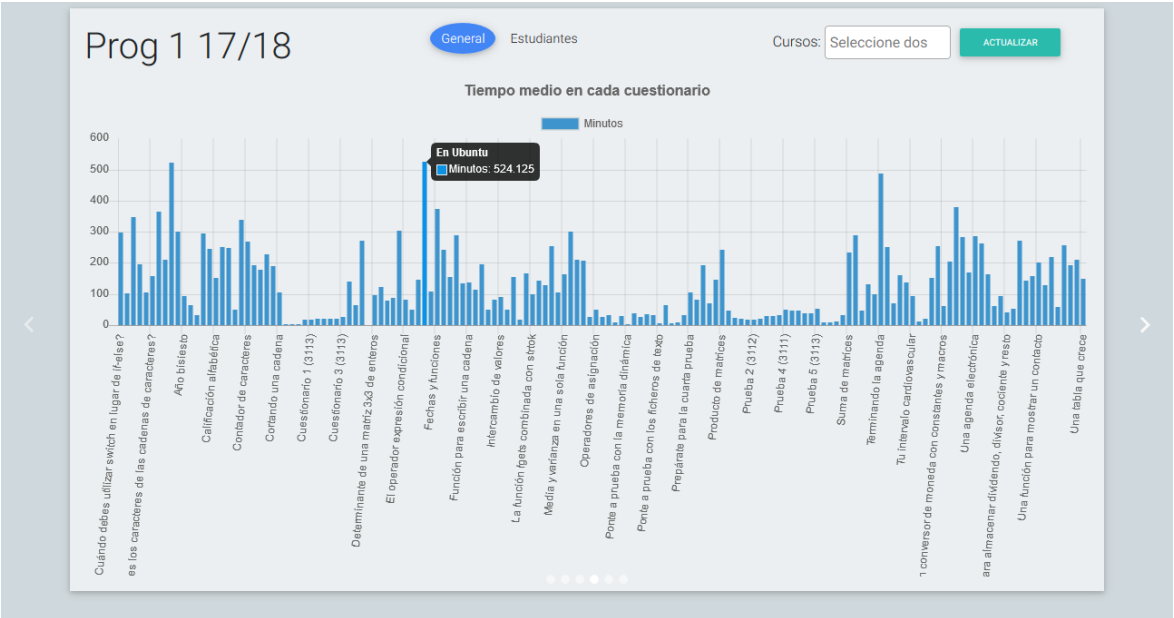


Figura 5-13 Gráfica en tiempo medio

La Figura 5-13 marca el tiempo medio en resolver los distintos cuestionarios. Estos se deben analizar de forma individual, ya que el tiempo que permanece abierto un cuestionario no tiene por qué ser el mismo.

La información que ofrece esta gráfica es muy importante ya que si un cuestionario como por ejemplo “En Ubuntu” ha tardado de media ocho horas, que es mucho tiempo, puede indicar que el concepto no ha quedado suficientemente claro. Esto provoca que a la hora de enfrentarse al cuestionario, los estudiantes decidan dejarlo para más adelante para así revisar la teoría, preguntar a sus compañeros o solicitar una tutoría. Por este motivo resulta muy útil, poder darse cuenta de puntos flojos o menos claros de la materia.

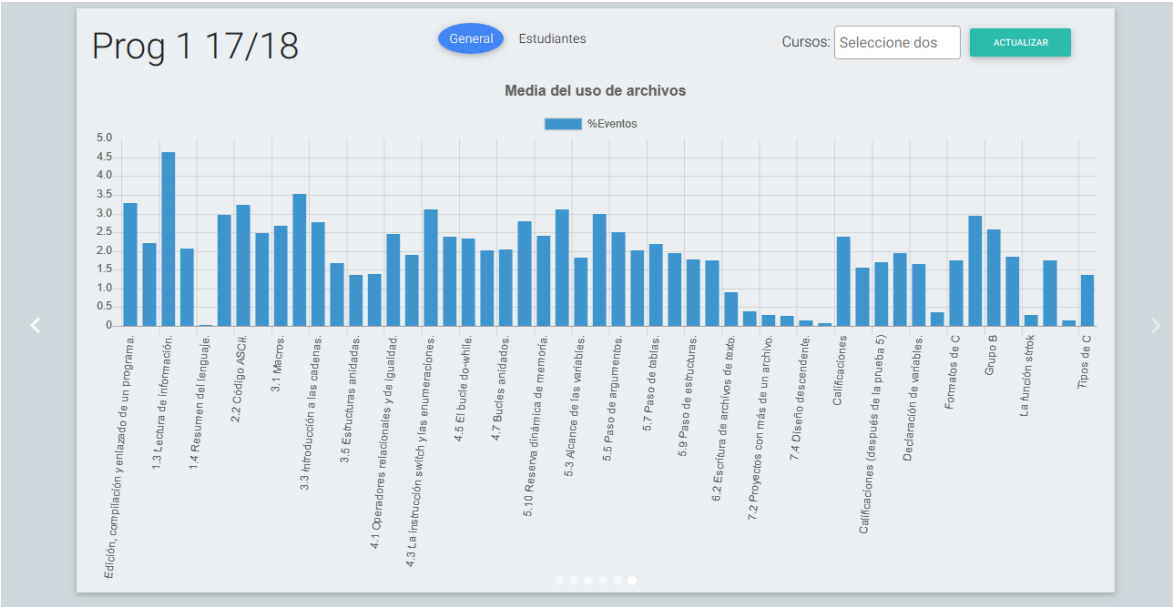


Figura 5-14 Gráfica de uso de archivos

Para finalizar la Figura 5-14 muestra la gráfica de uso medio de archivos. Esta gráfica también aporta mucha información, en este caso podemos destacar que el Resumen del primer tema es muy poco visitado en comparación con los otros archivos del tema. Por tanto, se puede llegar a la conclusión de que es poco útil para los estudiantes un resumen y prefieren estudiar el temario entero.

5.2.2 Resultados de un estudiante

En esta segunda sección de resultados, se muestra el potencial de la herramienta mediante graficas que muestran los datos de un estudiante. Se exponen las gráficas anteriores salvo la de evento/hora al no considerarse relevante y el uso de archivos al no implementarse para los estudiantes.

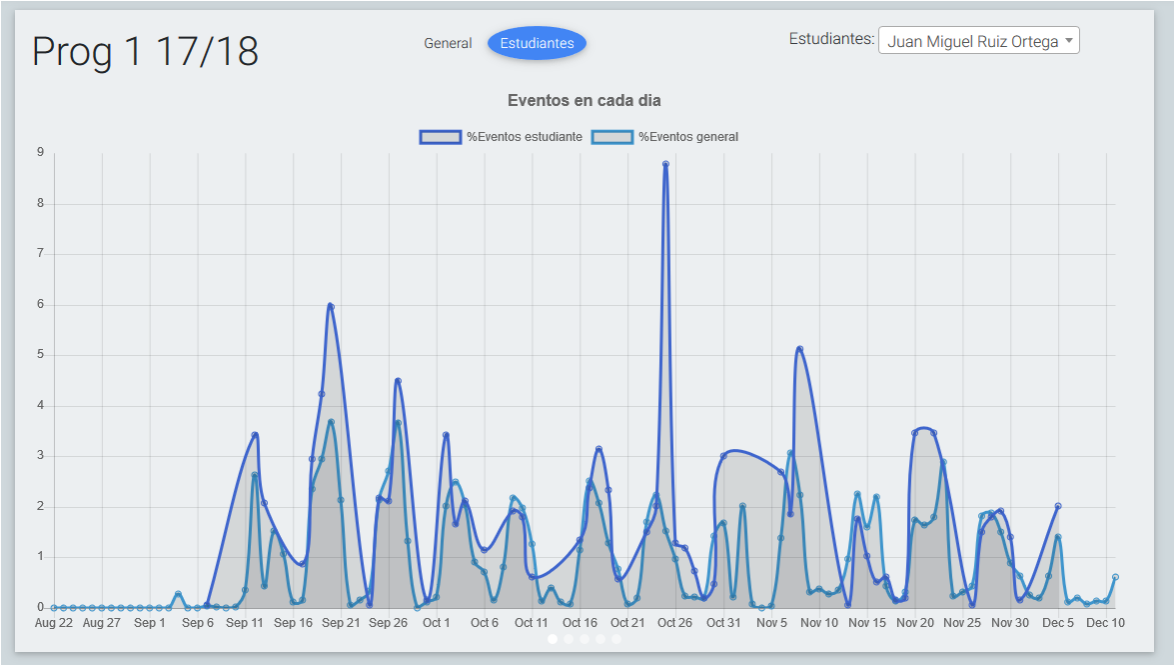


Figura 5-15 Gráfica estudiante eventos/dia

En la Figura 5-15 se muestra los datos tanto del estudiante como en general del curso. Lo útil de disponer de los dos datos juntos, es que permite ver si el estudiante sigue el mismo patrón de comportamiento que el curso, pero los datos más concluyentes se pueden sacar con la gráfica de eventos semanal.

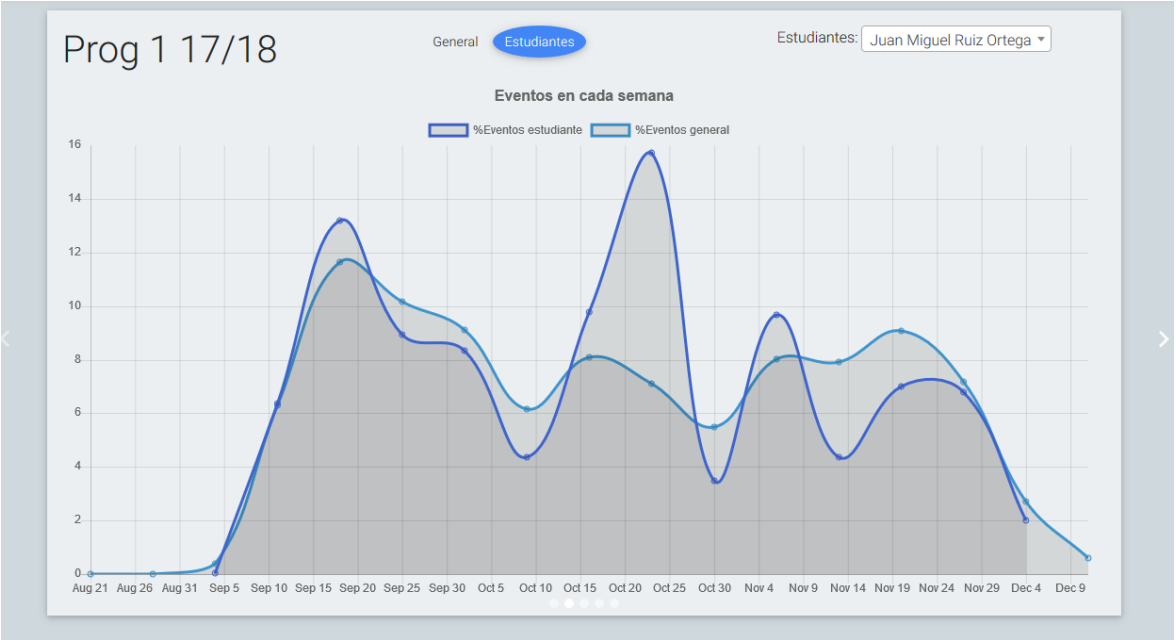


Figura 5-16 Gráfica estudiante eventos/semana

De la Figura 5-16 se puede llegar a la conclusión que el estudiante sigue la tendencia del curso hasta mediados de octubre que duplica la media del curso. Esto puede indicar un sobresfuerzo del alumno en la asignatura porque se había quedado descolgado o por una planificación propia del estudiante por una mayor carga de trabajo de otras asignaturas.

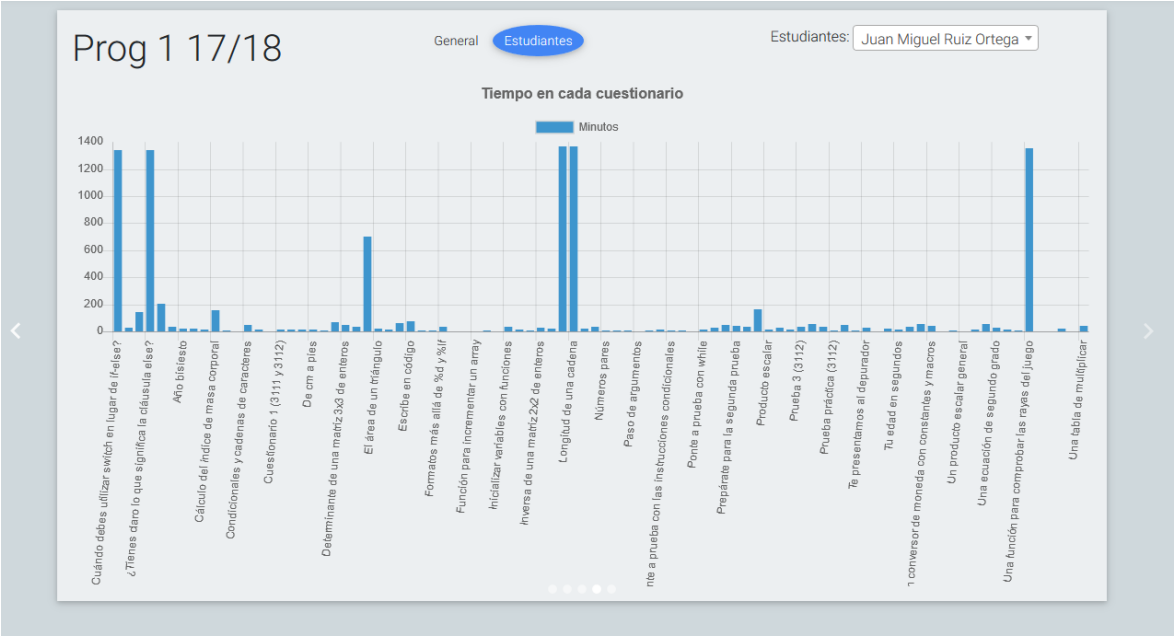


Figura 5-17 Gráfica estudiante tiempor/cuestionario

La Figura 5-17 aporta información muy importante ya que se puede observar los cuestionarios que más cuestan al estudiante. No obstante, esto puede reorientar una posible tutoría de una duda a explicar un conocimiento base que de primeras puede darse por sabido. En este caso se puede llegar a esa conclusión con las dos barras del medio sobre dos cuestionarios sobre cadenas.

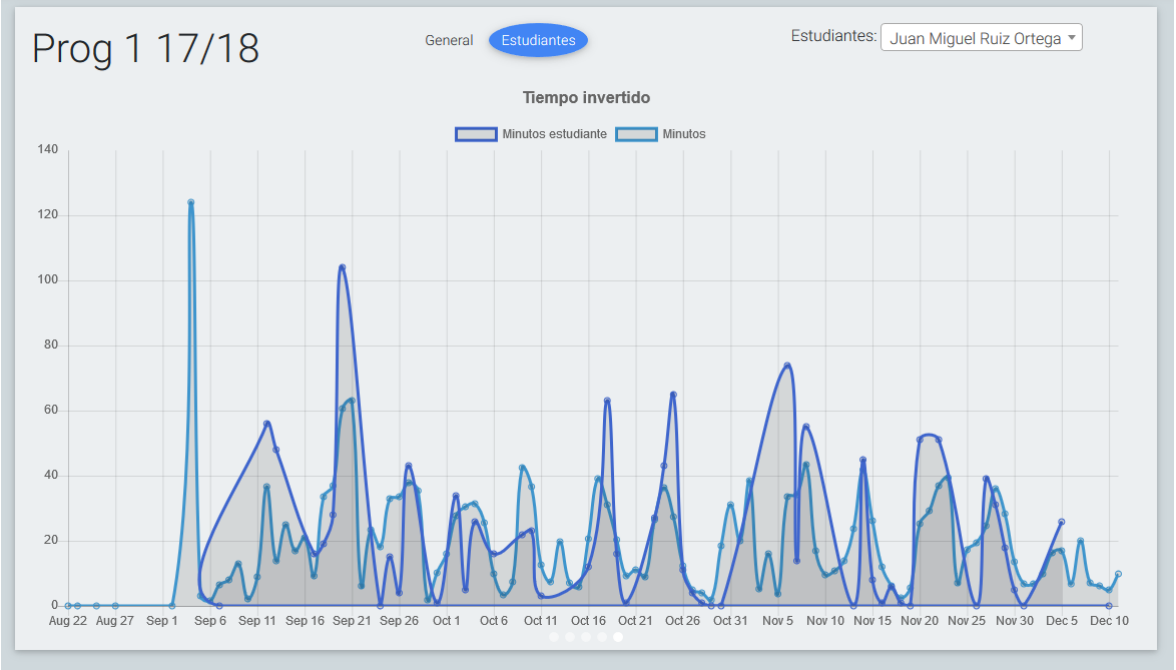


Figura 5-18 Gráfica estudiante tiempo dedicado/día

De la Figura 5-18 se llega a la conclusión que no hay una correlación entre los eventos que provoca el estudiante en la plataforma y el tiempo dedicado realmente. Para esto hay que ver el pico del 25 de octubre entre esta gráfica y la Figura 5-15.

6 Conclusiones y trabajo futuro

En este capítulo se detallan las conclusiones del trabajo desarrollado y las distintas formas de continuar este Trabajo de Fin de Grado.

6.1 Conclusiones

Tras realizar el proyecto se pueden obtener unas conclusiones positivas ya que se ha logrado el principal objetivo de crear una herramienta para visualizar los logs de Moodle. Adicionalmente, se consigue que esta visualización sea realmente útil como complemento para analizar el desarrollo de un curso.

Otra de las conclusiones que se obtiene después de realizar Graph Log Education, es el contraste de una herramienta de análisis frente a una de visualización. En las herramientas de análisis se espera obtener una respuesta, en contraste con una herramienta de visualización que provoca más preguntas. Las nuevas preguntas que se obtienen al usar la herramienta durante el análisis permiten un mayor refinamiento de los resultados del mismo, mejorando la calidad docente para futuros cursos.

A nivel del trabajo desarrollado, me ha servido para crecer como desarrollador al enfrentarme a un proyecto real desde principio a fin, asimismo destacar la experiencia obtenida tanto en la parte de servidor con Django y en el lado del cliente con Bootstrap.

6.2 Trabajo futuro

El trabajo futuro de la herramienta puede ser muy amplio. Uno de estos futuros sería el uso de la herramienta para realizar un análisis de las distintas asignaturas durante un cuatrimestre para encontrar correlaciones en los patrones de las gráficas que provocan las distintas asignaturas entre sí.

Otro de los trabajos futuros sería ampliar la herramienta con nuevas funcionalidades:

- Añadir otro idioma como el inglés.
- Añadir la posibilidad de filtrar los cursos que se muestran.
- Crear la posibilidad de filtrar la fecha en algunas gráficas.
- Ampliar la aplicación de Usuarios, dotándole de más funcionalidades como recuperar contraseña o borrar cuenta.
- Añadir la posibilidad de subir el fichero con las notas para cruzar los datos.
- Investigar nuevas gráficas.

Y para concluir esta sección, se puede enfocar el futuro trabajo en especializar la herramienta para una institución educativa, por ejemplo, la UAM. Esto tendría dos partes: la instalación en un servidor de la institución y modificar la aplicación de Usuarios para que permita la autentificación con las claves de acceso de la propia institución.

Referencias

[1] UAMx, Edx y MOOCs, web https://www.uam.es/ss/Satellite/es/1242677588563/subHomeServicio/MOOCs_de_la_UAM_en_edX.htm

[2] "De cómo la formación online y gratuita cambiará el futuro de la Universidad", web http://www.elmundo.es/f5/campus/2017/05/31/592db90de5fdea4f2e8b45e0.html

[3] Moodle Web official, web https://moodle.org/?lang=es

[4] Tool inspire, web https://moodle.org/plugins/tool_inspire

[5] Funcionalidad Analytics, web https://docs.moodle.org/34/en/Analytics

[6] Complemento logstore graylog, web https://moodle.org/plugins/logstore_graylog

[7] Electron, web https://electronjs.org/

[8] ASP.NET, web https://docs.microsoft.com/es-es/aspnet/

[9] Spring Framework, web https://spring.io/

[10] Laravel, web https://laravel.com/docs/5.6

[11] Erich Gamma, Richard Helm, Ralph Johnson, John Vissides "Patrones de diseño" Addison Wesley

[12] FAQ de Dango, web https://docs.djangoproject.com/en/2.0/faq/general/

[13] RFC 2898, web https://www.ietf.org/rfc/rfc2898.txt

[14] NIST https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf

[15] CSRF https://docs.djangoproject.com/en/2.0/ref/csrf/

[16] Class-View https://docs.djangoproject.com/en/2.0/topics/class-based-views/

Glosario

API	Application Programming Interface
CSS	Cascading Stylesheets
CSV	Comma-Separated Values
HTML	HyperText Markup Language
JSON	JavaScript Object Notation
MOOC	Massive Open On-line Course
MVC	Modelo Vista Controlador
REST	Representational state transfer
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
XML	eXtensible Markup Language

Anexos

A Manual de instalación en Ubuntu

Prerrequisitos

Para poder usar Graph Log Education se necesita tener instalado Python 3, esto viene de serie en las últimas versiones de Ubuntu, pero se recomienda usar un gestor como es Anaconda https://www.anaconda.com/download/.

Una vez instalado el gestor, se ejecuta el entorno grafico "Anaconda Navigator" que permite crear un Enviroment para todos los paquetes de la herramienta. Debido a lo anterior en partes posteriores de este manual se hace referencia al Enviroment por el nombre de TFG.

Otro requisito previo es tener instalado un sistema de base de datos SQL. Se recomienda usar PostgreSQL https://www.postgresql.org/download/ por ser la opción configurada en esta herramienta. No obstante si ya dispone de un sistema de base de datos SQL, puede usarlo si es compatible con Django, https://docs.djangoproject.com/en/2.0/ref/databases/ cambiando la configuración de la herramienta que se indica en la sección *Opciones y ejecución* de este manual.

Descarga

Para descargar el código fuente se visita el repositorio del proyecto, https://github.com/Yyandrakk/GraphLogEducation/releases donde se debe descarga la versión 1.0 "Primera Versión". Una vez descargado se descomprime el código fuente en una carpeta.

En esa carpeta se abre una terminal y se ejecutan los siguientes comandos:

$ source activate TFG
(TFG) $ pip install -r requirements.txt

El primer comando activa el Enviromente y el segundo descargara todos los paquetes necesarios para el funcionamiento de la herramienta.

Opciones y ejecución

Antes de poder ejecutar nada hay que cambiar las opciones de Graph Log Education para poder conectarnos a la base de datos antes instalada y la Secret Key. Pero antes de modificar la configuración hay que tener en cuenta la visibilidad del código que vamos a modificar.

```
155
156 DATABASES['default']={'ENGINE':'django.db.backends.postgresql',
157     'NAME': os.environ.get('DB_DB'),
158     'USER': os.environ.get('DB_U'),
159     'PASSWORD': os.environ.get('DB_P'),
160     'HOST': os.environ.get('DB_H'),
161     'PORT': '5432',
162 }
163
164
165 SECRET_KEY = os.environ.get('SECRET_KEY')
166
```

Figura: Opciones a cambiar

En el caso de que el código con los cambios, vaya a alojarse en un servidor de acceso público se recomienda el uso de variables de entorno o si el servidor es de acceso es privado, se pueden poner directamente.

Para modificar las opciones se abre el fichero GraphLogEducation/settings.py, donde se localizara las variables DATABASES y SECRET_KEY como se puede observar en la Figura: Opciones a cambiar.

La variable DATABASES se rellena con los datos de instalación de PostgreSQL o del gestor de base de datos SQL que tenga instalado.

Para obtener una nueva secret Key usaremos la terminal:

(TFG) $ python
>> from django.core.management.utils import get_random_secret_key
>> print(get_random_secret_key())

Se copia la combinación de caracteres que sale en pantalla y se asigna a la variable SECRET_KEY.

Por último, se ejecutan los siguientes comandos:

(TFG) $ python manage.py makemigrations
(TFG) $ python manage.py migrate
(TFG) $ python manage.py runsslserver {IP de la maquina}

Este último comando nos da un enlace para poder ingresar en el navegador.

B Manual del usuario

Para empezar a usar la aplicación hay que acceder a la URL donde se está ejecutando el servidor, ya sea en modo localhost o en un servidor remoto. Se recomienda usar un ordenador o una Tablet para visualizar los contenidos correctamente

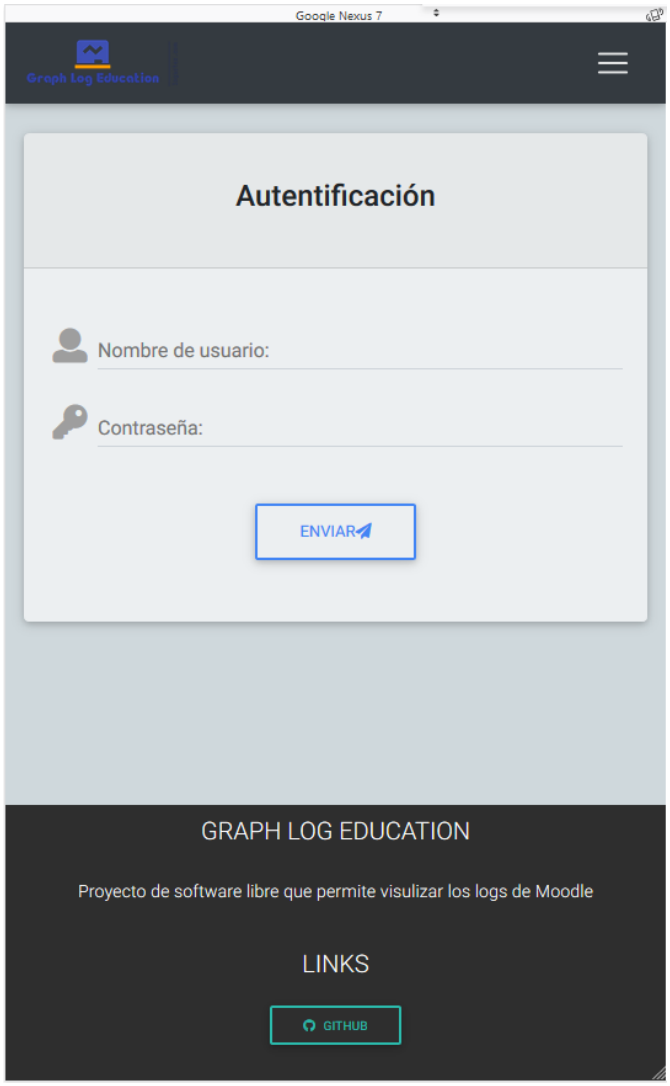


Figura: Modo Tablet

Si la pantalla es demasiado pequeña, los contenidos se adaptan. Los botones de la barra de navegación se ocultan y se muestran pulsando el botón “Hamburguesa”. Además, en el pie de página se dispone de un link al GitHub del proyecto para comentar dudas, reportar bugs y contribuir en el proyecto.

Registro y autentificación

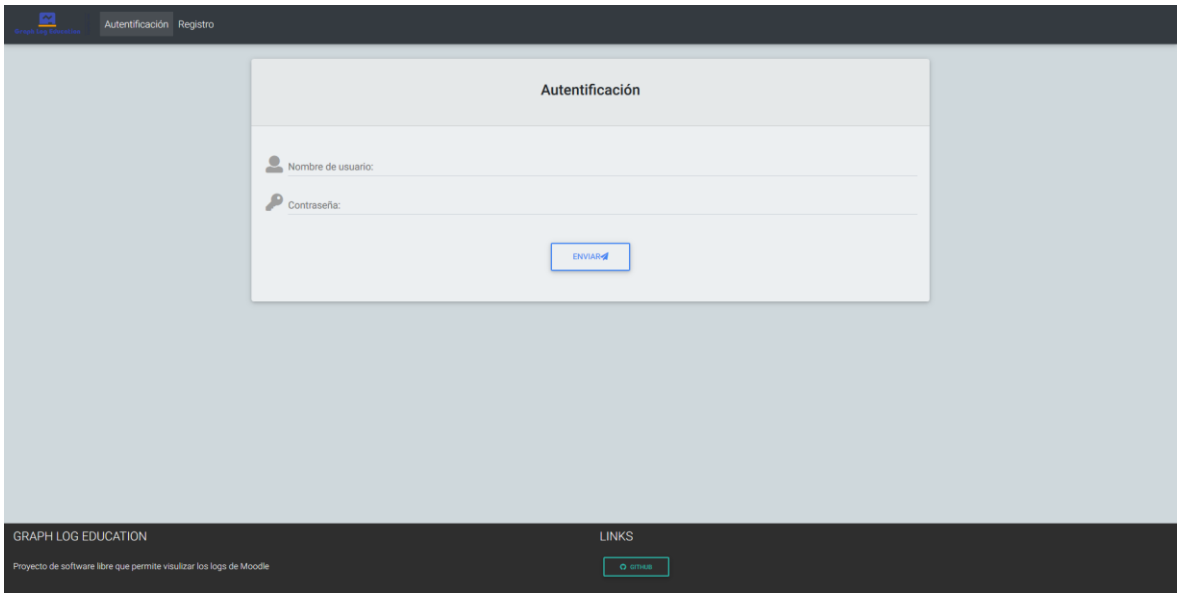


Figura: Pantalla de autentificación

Al acceder por primera vez, se muestra la pantalla de autentificación pero antes tendremos que registrarnos. Para ello pulsaremos el botón “Registro” de la barra de navegación superior.

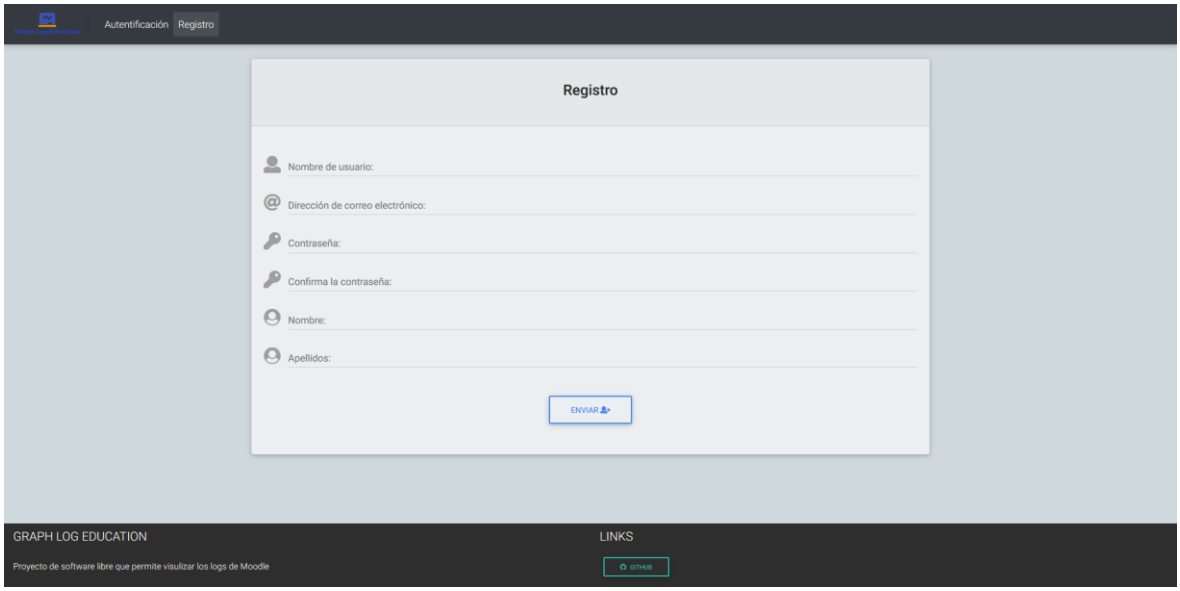


Figura: Pantalla de registro

Se rellenan todos los campos, teniendo en cuenta que la contraseña debe tener como mínimo 10 caracteres y contener letras y números. Una vez dado a registrar se redirecciona a la pantalla de autentificación y se insertan el nombre usuario y la contraseña.

Gestionar los cursos

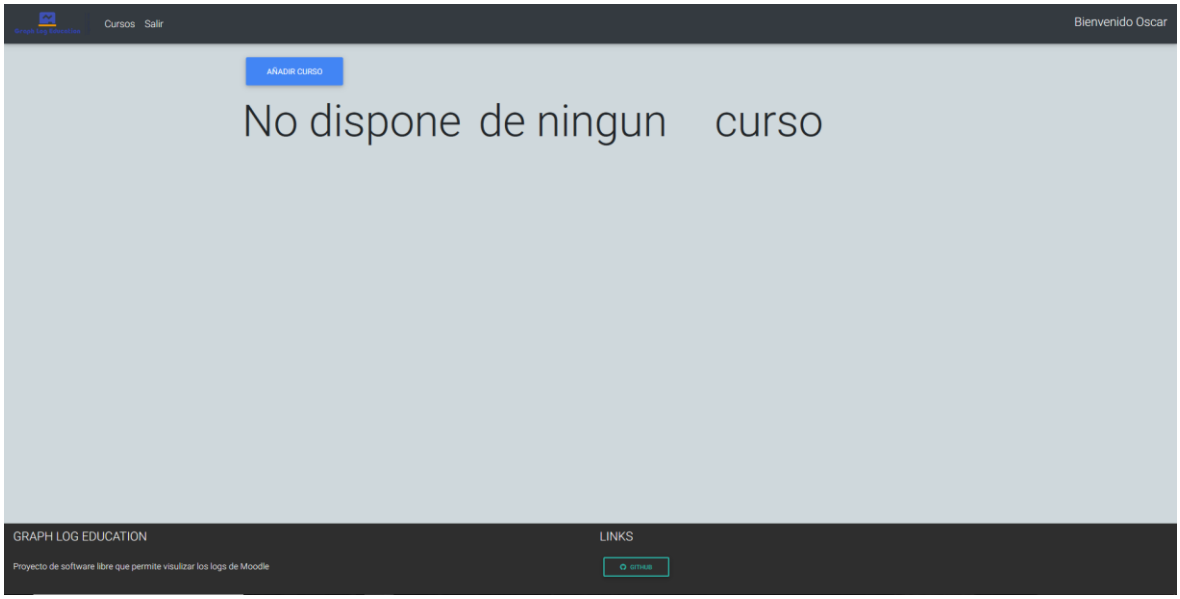


Figura: Pantalla sin cursos

Tras autentificarse, se muestra la pantalla del listado de cursos. En este momento indica que no se tiene ningún curso disponible y se debe pulsar el botón de "Añadir curso".

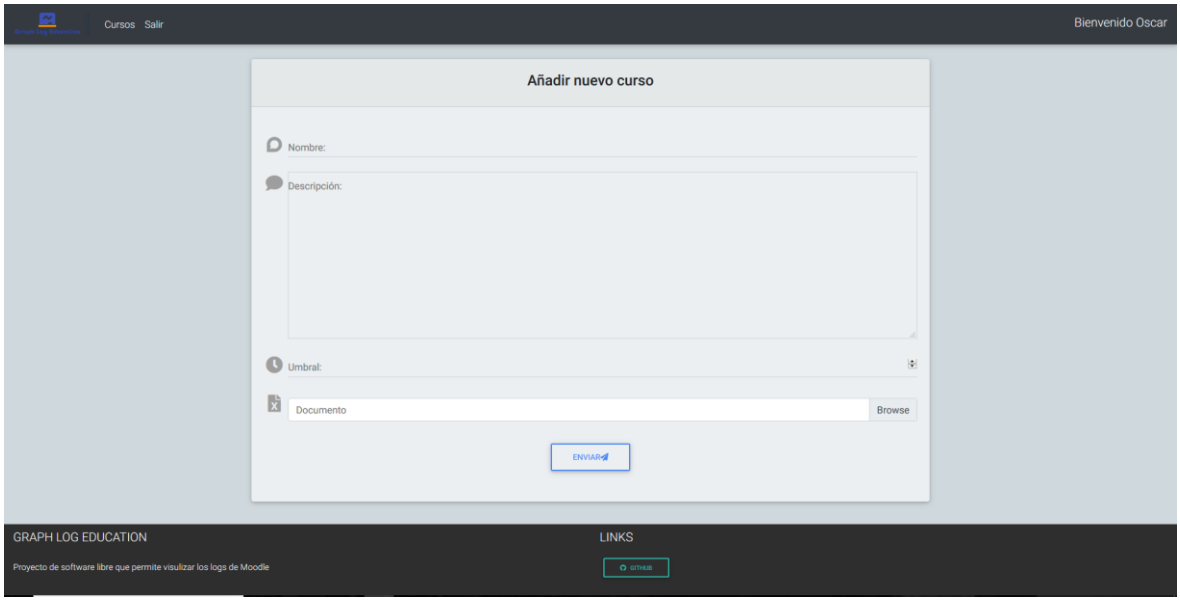


Figura: Pantalla de añadir curso

En la pantalla de añadir un nuevo cursos, se rellenan los siguientes campos:

- **Nombre:** Se recomienda la abreviación de la asignatura y el año que se está realizando.

- **Descripción:** Se puede añadir información de que titulación pertenece el curso, u otros datos de interés.

- **Umbral:** Es el tiempo máximo, en minutos, que puede pasar entre dos eventos, sirve para estimar el tiempo que pasa en Moodle el estudiante. Se recomienda 10 minutos pero puede variarlo para ver cual se ajusta mejor a su curso.

- **Documento:** Logs que se pueden extraer de Moodle del curso, el formato aceptado solamente es el CSV.

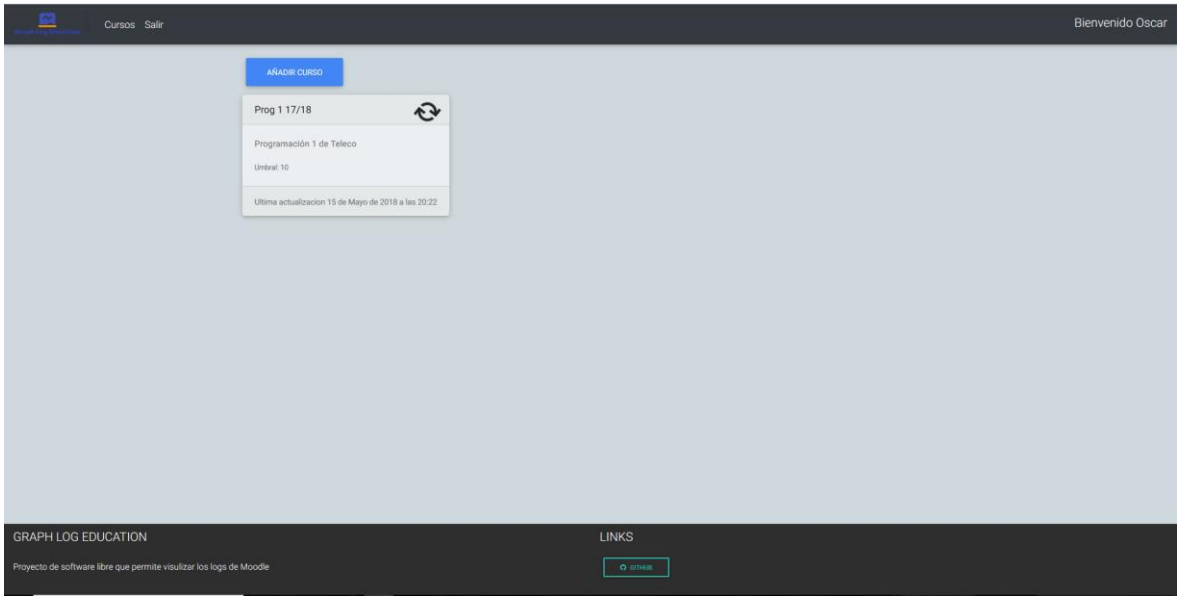


Figura: Pantalla con un curso en proceso

Mientras se procesa el curso, no se puede acceder ni a la información en detalle ni a las opciones del mismo, esto es así mientras la animación de las dos flechas este activada. Durante este proceso se puede añadir más cursos o visualizar ya existentes. No obstante, si se decide esperar en la misma página se recargara automáticamente cuando finalice.

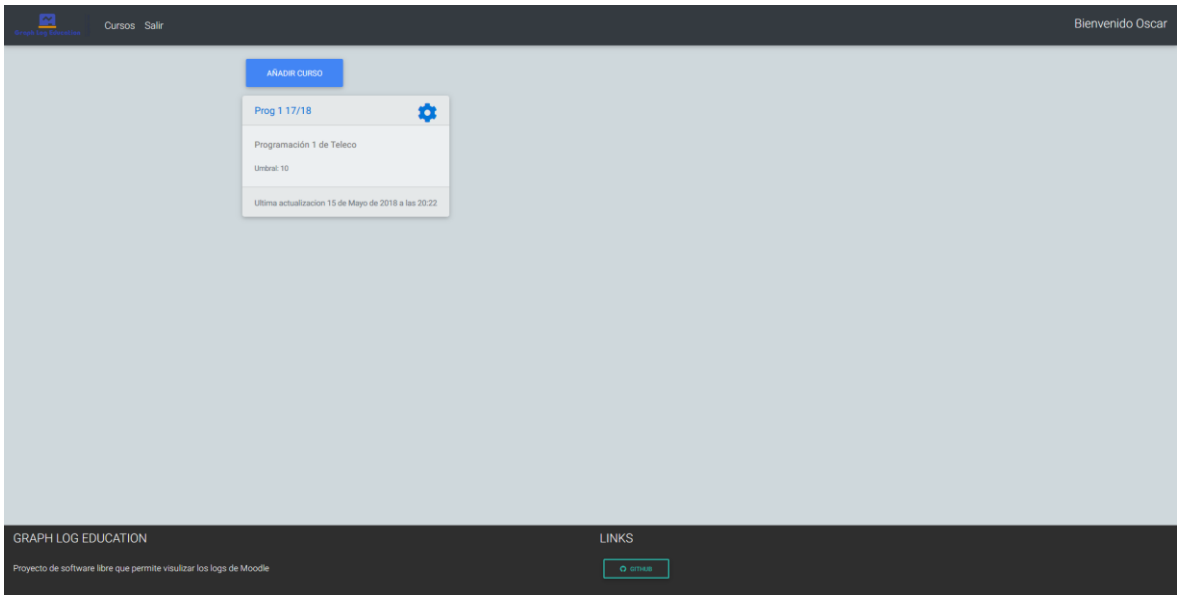


Figura: Pantalla con un curso

Cuando se active el curso, se tiene dos enlaces: uno en el nombre para acceder a la visualización de los logs y otro que es la rueda de opciones que lleva a actualizar o eliminar el curso.

Ver en detalle un curso

Para acceder a visualizar se pulsa el enlace del nombre que muestra las distintas gráficas.

Figura: Pantalla inicial de un curso

En la pantalla inicial se tiene una cabecera que siempre contiene el nombre del curso y dos botones para alternar entre la vista general y por estudiante.

Para cambiar de gráfica hay dos botones flecha en los laterales del cuadrado que contiene a la gráfica. Estos botones hacen que se vaya pasando las gráficas como si de un carrusel se tratase. Las gráficas son interactivas y exponen detalles si se pasa el ratón por encima de los puntos o barras.

Debido a los dos botones, en el caso de la vista general también tiene un desplegable para poder seleccionar hasta dos cursos distintos al actual para mostrar información conjunta en las gráficas de Evento/día y Evento/semana.

Figura: Pantalla del curso en modo estudiante

En el caso del modo estudiantes. El selector cambia para mostrar una lista con todos los estudiantes junto un buscador, en este caso no es necesario dar un botón actualizar para que los datos cambien. Adicionalmente, la información de las distintas gráficas es mostrada conjuntamente con los datos del alumno y los datos generales del curso. No obstante si se quiere no mostrar cualquiera de los dos, solamente hay que pulsar en la leyenda de la gráfica.

Opciones del curso

El curso dispone de dos opciones, actualizar la información y eliminar el curso.

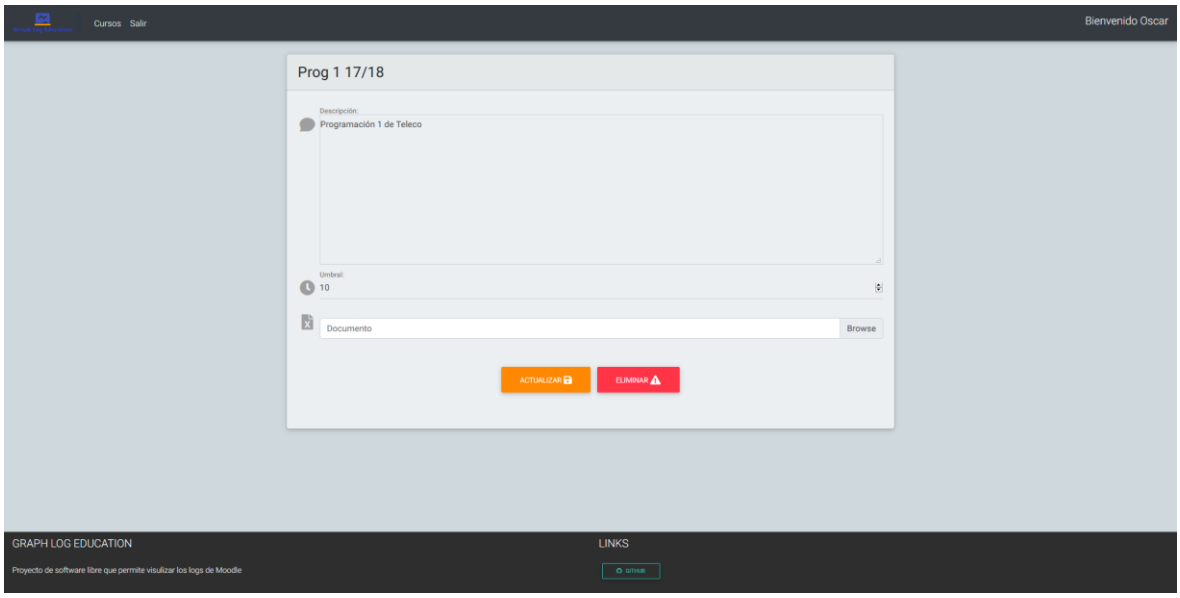


Figura: Pantalla de actualización

Para actualizar un curso, se dispone de los campos: descripción, umbral y documento. Una vez pulsado el botón "Actualizar" se redirige a la pantalla de listado del curso y dependiendo de los campos modificados el curso debe ser procesado parcialmente o totalmente, o no ser necesario. No obstante si se pulsa el botón eliminar se redirecciona a la pantalla de confirmación.

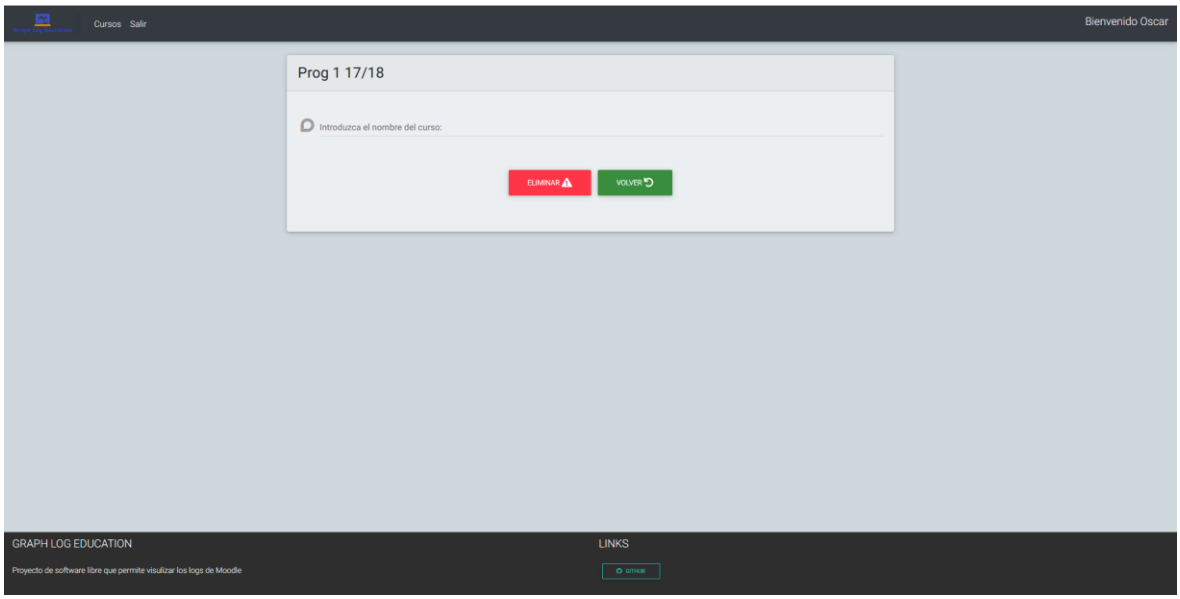


Figura: Pantalla de confirmación

Para confirmar la eliminación del curso, hay que introducir su nombre y dar al botón eliminar. Esta acción quiere evitar un eliminación accidental ya que eliminar un curso es una operación definitiva y se pierde la información.

C Análisis

Requisitos funcionales

RF1: Los usuarios tendrán un apartado para poder registrarse y usar la aplicación.

RF2: Los usuarios registrados podrán autentificarse con los datos proporcionados en el registro para poder acceder a la aplicación.

RF3: Los usuarios autentificados podrán cerrar la sesión.

RF4: Los usuarios autentificados podrán crear nuevos cursos, que solo serán visibles para el usuario creador.

RF5: Los usuarios autentificados podrán ver un listado de los cursos que han creado.

RF6: Los cursos creados por los usuarios contendrán los siguientes campos para ser proporcionados por el usuario: nombre, descripción, parámetro umbral y el fichero de logs de Moodle.

RF7: Los cursos podrán editar la información de descripción, parámetro umbral y el fichero de logs de Moodle.

RF8: Estos cursos podrán ser borrados.

RF9: Al procesar el fichero de un curso se guardarán los datos para poder mostrar las siguiente información:

a. Numero de eventos/día
b. Numero de eventos/semana
c. Numero de eventos/hora
d. Estimación de uso de material /día
e. Tiempo en resolver cada Cuestionario

Esta información también se deberá mostrar para cada estudiante, salvo la d.

RF10: El curso deberá permitir filtrar el estudiante elegido para mostrar su información.

Requisitos no funcionales

RNF1: La aplicación contará con un manual de instalación y uso.

RNF2: La aplicación tendrá una cabecera dinámica como navegación y un pie de página estático.

RNF3: El único idioma disponible será el castellano.

RNF4: Se dispondrá de interfaz gráfica responsive, y podrá ser usada con teclado y ratón.

RNF5: La información indicada en el *RF8*, será mostrada preferentemente por gráficas.

RNF6: El procesado del fichero no será bloqueante, y permitirá a los usuarios seguir con la aplicación mientras se crea un curso nuevo.

RNF7: Se indicará cuando un curso no está disponible porque se está procesando su fichero.

RNF8: El sistema obligará al usuario a introducir una contraseña alfanumérica, con mayúsculas y minúsculas, de un tamaño mínimo de diez.

RNF9: La aplicación se podrá usar en cualquier navegador.

RNF10: El fichero tendrá la extensión CSV.