UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Módulo de Seguimiento de un Tutor Online de C

Carlos García Mañas
Tutor: Alejandro Sierra Urrecho

Junio 2016

Módulo de Seguimiento de un Tutor Online de C

AUTOR: Carlos García Mañas
TUTOR: Alejandro Sierra Urrecho

Dpto. Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2016

Resumen

El objetivo de este TFG es el desarrollo de un módulo encargado de llevar el seguimiento de cada uno de los alumnos de la web para el aprendizaje autónomo del lenguaje de programación C. El módulo proporciona información detallada al profesor sobre el proceso de aprendizaje de cada alumno, permitiendo así ayudar de manera más eficaz a los alumnos que lo necesiten y poniendo a prueba la dificultad de nuevos ejercicios impuestos a los alumnos y así poder mejorar el contenido del curso entre ediciones sucesivas. La aplicación web, en sí, puede ser de gran ayuda a los estudiantes que tengan asignaturas relacionadas con la programación web. No obstante, con este módulo se ayudará también a la gestión de los alumnos por parte del profesor y tener una mayor proximidad a los problemas que surgen a los alumnos.

Para la implementación de este módulo, he tenido que partir del trabajo previamente desarrollado por Felipe Millán Fernández para su TFG llamado "Aplicación web para la enseñanza de C". Estructuralmente hablando, he tenido que modificar principalmente la base de datos, para poder capturar mayor información por estudiante y, después, poder ser analizada. Además se han retocado y añadido nuevas funcionalidades que serán explicadas a lo largo de este documento.

En cuanto al uso de gráficas me he ayudado de la librería de Highcharts JS, que ofrece una cantidad infinita de posibilidades en el trato y visualización de los datos en gráficas. Está librería y su implementación será mostrada más adelante en este mismo documento.

Uno de los objetivos del trabajo, además del educativo, ha sido el estudio y análisis de la aplicación web previamente creada. Me he encontrado con un trabajo realizado en un Framework totalmente desconocido para mí, ya que no se estudia en la carrera, hablo de Laravel. Por tanto, he tenido que estudiarme previamente cómo crear una aplicación web desde cero en Laravel y, después, estudiarme el código realizado por Felipe Millán Fernández, además de estudiarme todo lo relacionado con mi TFG, seguimiento de usuarios en una aplicación web.

Palabras clave

Estadísticas, ingeniería del software, aplicación web, lenguaje de programación, Laravel, Highchart JS, módulo de seguimiento, gráficas de datos.

Agradecimientos:

En primer lugar agradezco a mi familia el apoyo recibido a lo largo de la carrera y a mis padres por hacer posible que pudiese estudiar una carrera que me llena tanto como es la Ingeniería Informática.

En segundo lugar agradezco al tutor de este TFG, Alejandro Sierra Urrecho, el haber querido llevar a cabo este proyecto conmigo y en haber tenido que aguantar mi modo de vida intentando adaptarnos el uno al otro de la mejor manera posible para poder sacar el proyecto de manera satisfactoria.

Por último, y no por ello menos importante, agradezco a Rodolfo, exprofesor de la escuela, todo el apoyo y consejos recibidos por su parte para que pudiera sacar tal carrera e introducirme al mundo laboral. Sin él ahora mismo no estaría en la situación en la que estoy.

No quiero terminar estos agradecimientos sin nombrar a amigos, profesores y alumnos conocidos durante la carrera que han formado parte en ella y que, por tanto, han contribuido a que pudiese finalizarla.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

Hoy en día las empresas realizan un gran esfuerzo económico al seguimiento e intereses de los usuarios en la red, para así poder satisfacer de manera más eficaz y directa lo que el usuario quiere. Esta idea, extrapolada a una aplicación web de educación, ayuda a que el profesor pueda ver las carencias y eficacia de cada alumno. Así, el profesor podrá comprobar que alumnos requieren mayor o menor ayuda y quiénes son los alumnos más aventajados de la clase. Este trabajo de fin de grado forma parte del área de la ingeniería del software y de las tecnologías de la información y computación (departamento de Ingeniería Informática) y se van a exponer las ideas para la creación de un módulo de seguimiento para una aplicación web relacionada con la educación.

1.1 Motivación

Mi principal motivación para la elección de este proyecto de TFG es que he ido estudiando y poniendo en práctica programación orientada a Internet. Esto quiere decir que, independientemente de lo estudiado y trabajado en la carrera, yo he ido profundidad de la programación relacionada con HTML, PHP y JavaScript principalmente. Por ello, veía una gran oportunidad de aprender aún más y contribuir de una manera más desarrollada a si no hubiese centrado mi enfoque de futuro a este campo de la informática, el desarrollo de aplicaciones web.

No obstante, también me veía realmente atraído por la idea presentada por mi tutor de TFG Alejandro Sierra Urrecho de contribuir en el desarrollo de una herramienta que pudiera ayudar a futuras generaciones de informáticos en la introducción al mundo de la programación C, como es el caso.

1.2 Objetivos

Uno de los principales objetivos del proyecto es la obtención y análisis de datos obtenidos de los estudiantes de un curso para que el profesor de la asignatura tenga un mayor control, tanto individualmente por cada alumno como de manera global. Para conseguir este objetivo principal hay que cumplir otros secundarios.

Uno de los objetivos secundarios del proyecto es crear una interfaz más amigable y cómoda para que el usuario se vea atraído por ella y así no haya un rechazo de una asignatura por la plataforma que se use en ella.

Otro objetivo secundario es que los datos obtenidos sean tratados de manera correcta para que, al mostrarlos de manera definitiva al profesor de la asignatura se muestren datos relevantes y no obviedades.

Por último, el profesor ha de tener un entorno que pueda reconocer con facilidad y que no sea complicado su navegación por ella. Para ello se tienen que mostrar las estadísticas, gráficas y datos de una manera atractiva y útil.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Estado del arte:** En este capítulo se verá las diferentes maneras existentes de crear gráficas y cómo debieran ser para este proyecto, y análisis del proyecto heredado (ya que se parte de un trabajo realizado por otro alumno con anterioridad).
- **Diseño:** Este capítulo servirá de guía para lo que se quiere realizar en el proyecto. Se verán aspectos estéticos y técnicos de la aplicación, abarcando así tanto el Front-End como el Back-End.
- **Desarrollo:** En este capítulo se verá cómo se ha desarrollado las diferentes funcionalidades y estéticas explicadas en el capítulo anterior.
- **Integración, pruebas y resultados:** Este capítulo tendrá como fin la explicación de las funcionalidades desarrolladas desde la codificación. Se verán diferentes ejemplos que explicarán de manera general el trabajo realizado. Además también se contemplarán las pruebas y resultados tenidas en cuenta para obtener el resultado final.
- **Conclusiones y trabajo futuro:** En el último capítulo se valorará el trabajo realizado en este proyecto y posibles pasos y mejoras a llevar a cabo en un futuro.

2 Estado del arte

Para cumplir los objetivos establecidos, previamente hay que comparar los sistemas similares que existen en la actualidad. Como el módulo del que trata este TFG no es visible de manera pública en Internet, ya que sirve para la analítica interna de cada web, no me es posible comparar de manera directa con las tecnologías existentes. Por tanto, a continuación explicaré las tecnologías actuales que nos podrían servir para la visualización de los datos recogidos en la aplicación.

2.1 Tecnologías actuales de visualización de datos

Está sección contemplará las posibles maneras de mostrar datos y ejemplos de tecnologías que nos ayudarán a implementarlos. Para mostrar los datos siempre nos ayudaremos de las tecnologías como JavaScript, HTML y PHP, principalmente.

2.1.1 Gráficos estáticos

Se trata de gráficos que no pueden ser alteradas directamente por el usuario y que, por tanto, no permite flexibilidad a la hora de querer centrarte, como en este caso, en un alumno en concreto comparándole con la media de los estudiantes.

Investigando, he llegado a la conclusión de que este tipo de tecnología es poco útil dentro del módulo a crear ya que no permite la interacción directa con las gráficas.

2.1.2 Gráficos dinámicos

¿Qué es un gráfico dinámico? Un gráfico dinámico es el que se utiliza basado en una tabla dinámica, para mostrar gráficamente los datos deseados y manejar la información según se desee. En nuestro caso, la tabla dinámica son las tablas que tendremos que crear en la base de datos ya existente para almacenar toda la información posible y de interés para el profesor, y que posteriormente sea tratada. Más adelante se realizará un estudio para saber cuáles son los datos idóneos a recoger y la manera de tratarlos.

Hoy en día hay múltiples librerías que sirven para mostrar datos de manera dinámica en un gráfico. No obstante, hay que seleccionar la más apropiada para tener una facilidad, tanto para el usuario como para el programador. Si investigas un poco en Internet encontrarás varias librerías que sirven para éste fin, como por ejemplo GraPHPico, AnyChart, pChart, etc.

Finalmente me he decantado por la librería HighCharts JS, que tiene un abanico muy amplio de diferentes maneras de representar los datos y de manera muy estética y funcional para el usuario. Esta librería será explicada con detalle y su implementación y uso en el proyecto más adelante en este mismo documento.

2.2 Análisis del trabajo realizado en Codeps

En esta sección realizaré un análisis del estado inicial de la base de datos, estética y navegación y funcionalidad de la aplicación web creada en un TFG del año pasado por Felipe Millán Fernández. Si se quiere saber más sobre dicho TFG, se puede consultar [1].

2.2.1 Estructura inicial de la base de datos

La base de datos es en cimiento básico de una aplicación web. Si quiero modificar cualquier cosa, primero he de entender cómo funciona y que relaciones hay entre tablas. Por ello, se saca primero el diseño de la base de datos ya implementada.

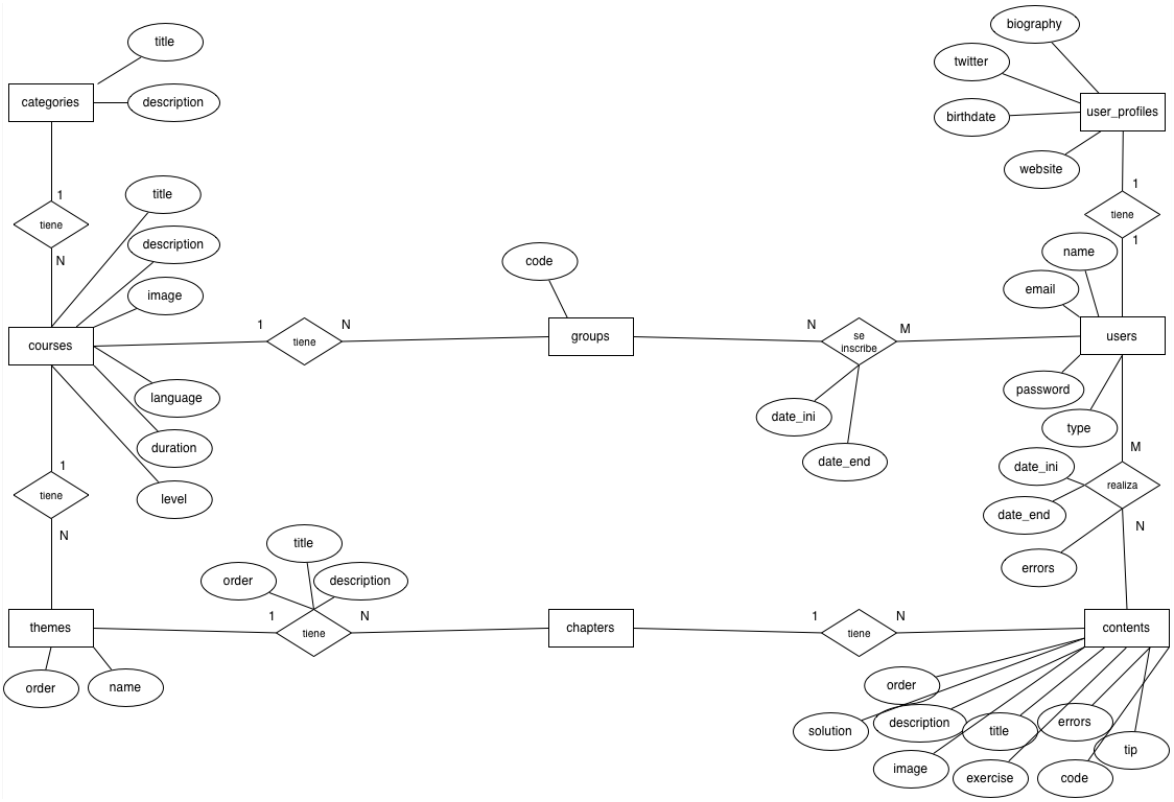

Figura 1. Modelo Entidad-Relación inicial de Codeps

Como se puede observar el la Figura 1, la aplicación Codeps usa una base de datos relacional para almacenar la información de todos los usuarios registrados en el sistema y todos los cursos que hay disponibles. Cada usuario tendrá su perfil, donde se indica si es un administrador/profesor o un estudiante. Cada curso contiene varios grupos a los que los alumnos pueden registrarse. El curso está formado por varios temas, éstos por varios capítulos y a su vez por distintos contenidos. Un contenido es el ejercicio que el alumno tendrá que realizar con una breve explicación y el enunciado en cuestión.

Viendo esto, habrá que retocar la base de datos para guardar datos de los usuarios en relación al curso que estén cursando. Pero antes de hacer esto, hay que saber cuáles son los datos requeridos y diseñar una nueva base de datos partiendo de ésta(Figura 1).

2.2.2 Estética de la página web

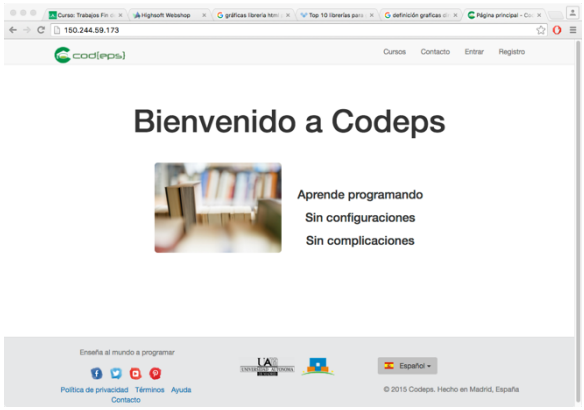

Figura 2. Página principal Codeps (antes)

En la Figura 2 se observa la página inicial de la aplicación web. A primera vista tiene un estilo minimalista pero, para mi gusto, con demasiado blanco. Se echa en falta un fondo principal que acompañe a lo largo de la navegación, ya que este fondo lo comparte por toda la aplicación web.

En la Figura 3 tenemos la selección de curso para verlo, y si da el caso matricularse. Se puede ver el mismo problema del fondo. Además los cursos se muestran alineados a la izquierda, lo que dejaría, como es el caso, mucho hueco libre a la derecha al haber un único curso. Para ello se me ocurre alinear al centro los cursos.

Figura 3. Selección de curso Codeps (antes)

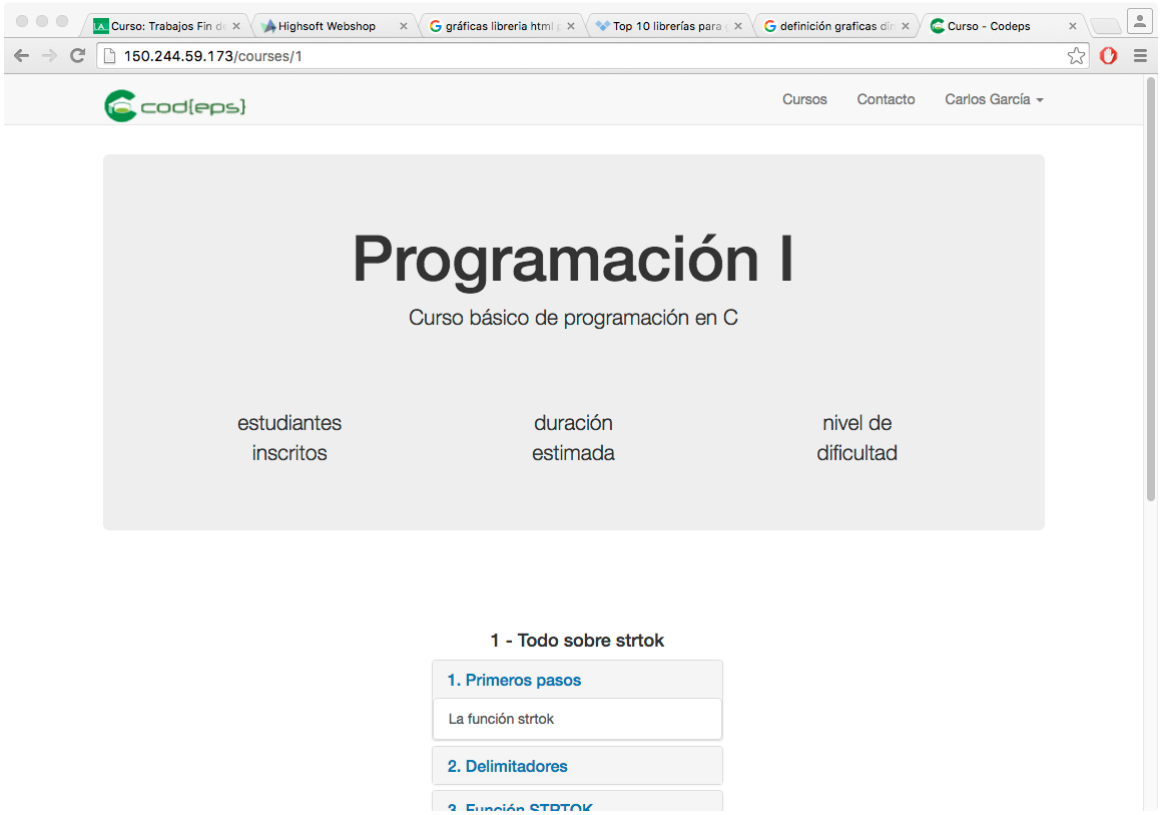

Figura 4. Página de curso Codeps (antes)

En la Figura 4 tenemos la parte superior de un curso donde se muestran los diferentes ejercicios a disposición del alumno para que sean realizados. Como he comentado antes, el fondo deja mucho que desear. Además se observa en la cabecera del curso, en este caso "Programación 1", que muestra títulos de información, pero que no muestra dicha información. También puedes acceder a cualquier ejercicio aunque no hayas hecho el anterior de manera satisfactoria.

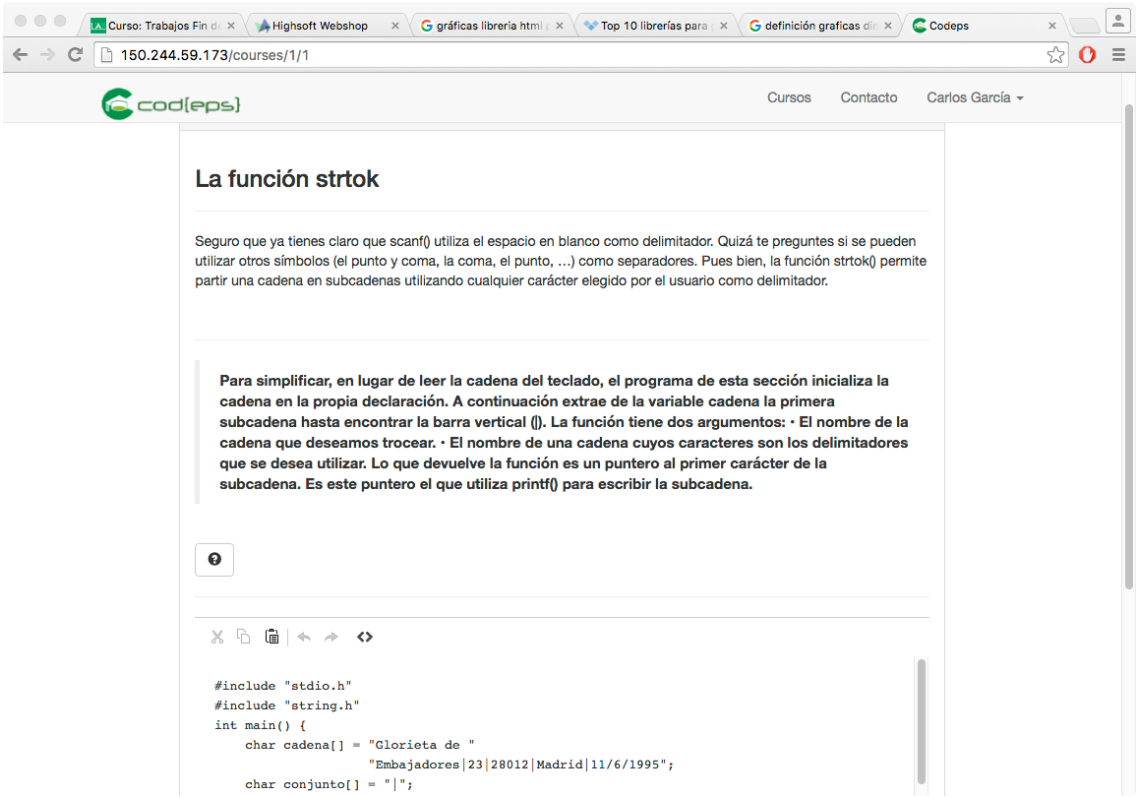

Figura 5. Ejercicio en Codeps (antes)

En la Figura 5 se observa la zona de ejercicio, en concreto el ejercicio “La función strtok”. Esta imagen muestra el título inicial, una breve explicación y el ejercicio en sí. Además tiene un botón de ayuda por si necesita información adicional, al verse el alumno en apuros para resolver el ejercicio. Debajo del todo ya se ve la zona donde implementar el código que se solicita en el ejercicio. Como se puede ver, el alumno requerirá de realizar un scroll para poder realizar el ejercicio, lo que le conllevará a no poder ver parte del enunciado. Además no hay manera posible del volver al curso dentro del entorno, sino que hay que volver atrás con el navegador. También excede de blanco el fondo para mi gusto y además si finalizas el ejercicio de manera exitosa no puedes acceder directamente al siguiente ejercicio.

Todos estos comentarios son en base a la estética de la aplicación web que tengo que modificar y añadir el módulo de seguimiento del alumno. Las páginas web son como la comida, primero ha de entrar por los ojos, y por ello le doy cierta relevancia a este aspecto en mi TFG.

2.2.3 Navegación y funcionalidad por la página web

Para que el usuario de la aplicación web Codeps se sienta cómodo en ella, ésta tiene que ofrecer la mayor simplicidad y facilidad al usuario para navegar por ella. Entonces, una página web ha de ser sencilla e intuitiva.

Por ejemplo, uno de los aspectos a mejorar en esta navegación son la posibilidad de navegar de un ejercicio a otro sin tener que pasar obligatoriamente por el curso, solo ha de haber aprobado el ejercicio anterior. Esto conlleva que hay que realizar, también, un seguimiento del alumno por ejercicio para saber a qué ejercicios puede, o no, acceder.

Para la parte a desarrollar en este TFG, es decir, las páginas relacionadas con el seguimiento de los alumnos por parte del profesor y las funcionalidades añadidas para la recogida de datos del alumno, como se parte desde cero, serán explicadas más adelante.

2.3 Características del servidor web instalado para Codeps

Bueno, hasta ahora sabíamos cómo estaba estructurada la base de datos y como se presenta al cliente la aplicación pero, ¿cómo funciona el servidor que sustenta esta aplicación? El servidor que está instalado es un servidor Apache que usa un Framework llamado Laravel. ¿Qué es **Apache**? ¿Qué es **Framework**? ¿Y **Laravel**?

Pues bien, un **servidor HTTP Apache** es un servidor web HTTP de código abierto, para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.12 y la noción de sitio virtual. Este tipo de servidor tiene múltiples ventajas, como que es modular, de código abierto, multi-plataforma, extensible y muy popular, por lo que se podrá encontrar mucha información en Internet. Si se quiere saber más sobre **servidor HTTP Apache**, se puede consultar [2].

Un **Framework** es un entorno o ambiente de trabajo para desarrollo; dependiendo del lenguaje normalmente integra componentes que facilitan el desarrollo de aplicaciones como el soporte de programa, bibliotecas, plantillas y más. Si se quiere saber más sobre **Framework**, se puede consultar [3].

Laravel es el Framework utilizado en la aplicación web existente. **Laravel** es un Framework de código abierto para desarrollar aplicaciones y servicios web con PHP 5. Su filosofía es desarrollar código PHP de forma elegante y simple, evitando el "código espagueti". Si se quiere saber más sobre **Laravel**, se puede consultar [4].

El código de la aplicación está escrito en **PHP Hypertext Preprocessor**, al ser el lenguaje más y con mayor comunidad de usuarios detrás, que cuenta con una extensa biblioteca de funciones. PHP sirve para crear aplicaciones web dinámicas con acceso a información almacenada en una base de datos y que permite incorporar código en las páginas HTML de forma sencilla. Usa técnicas de programación orientada a objetos y sigue el patrón de diseño modelo vista controlador. La Figura 6 es un ejemplo de cómo funciona la tecnología PHP. Se comporta como un módulo de Apache, que extrae código dentro de las páginas, lo ejecuta en el servidor y envía el resultado al cliente. Éste no puede visualizar el código del programa, solamente su resultado. Si se quiere saber más sobre el modelo vista controlador, se puede consultar [5].

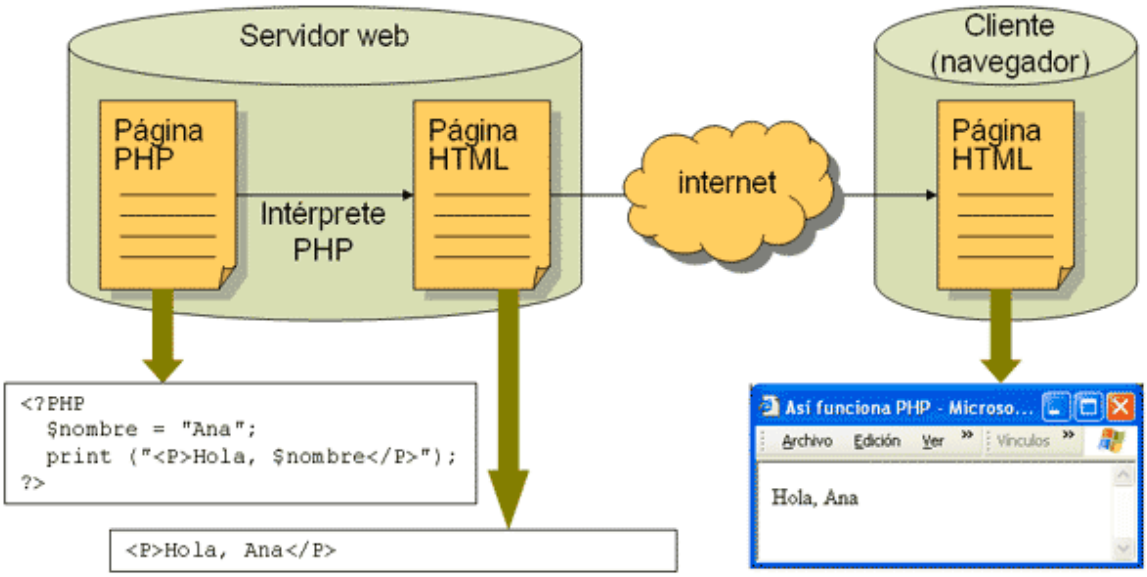

Figura 6. Funcionamiento PHP

Además, la aplicación se soporta en un sistema de bases de datos, **PostgreSQL**. **PostgreSQL** es un sistema de gestión de bases de datos relacional orientado a objetos y libre, publicado bajo la licencia PosgreSQL , similar a la BSD o la MIT. Si se quiere saber más sobre **Postgresql**, se puede consultar [6].

2.3.1 Herramientas usadas para el lado del cliente (Front – End)

Cómo es natural a la hora de programar una aplicación web o, en este caso, una parte de una aplicación, se necesitan varios lenguajes de programación para hacer que tu web sea atractiva y funcional a la vista de una persona sin mucho conocimiento de Internet. Para ello, en este proyecto he usado, principalmente, tres lenguajes de programación claves: **HTML**, **CSS**, **PHP** y **JavaScript**.

¿Qué es **HTML**? **HTML** son las siglas en inglés de lo denominado **HyperText Markup Language** (lenguaje de marcas de hipertexto), hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia del software que conecta con la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código (denominado código **HTML**) para la definición de contenido de una página web, como texto, imágenes, videos, juegos, entre otros. En este proyecto será usado **HTML 5** que presenta ciertas mejoras ante sus versiones predecesoras. El **HTML** nos ayudará para crear nuevas páginas desde cero manteniendo una estructura firme.

El **HTML** es la estructura principal de una página web pero, ¿cómo dejar esa estructura atractiva para el usuario?. Aquí es donde entra el **CSS**. El **CSS** es un lenguaje de programación adherido al **HTML** con el fin de cambiar la estética con que éste es presentado a un usuario. Es decir, el **HTML** puede crear sin problemas una tabla, pero si las líneas de la tabla las quiero más gruesas he de recurrir al **CSS**. **CSS** es comúnmente conocido como la hoja de estilos de **HTML**. El **CSS** hará que nuestras páginas queden vistosas y ordenadas.

Si se quiere saber más sobre **HTML** , se puede consultar [7], y si quieres saber más sobre **CSS** consulta [8].

Tenemos la estética y también la estructura, ¿qué nos falta?. La funcionalidad. Tras tener una página bien estructurada y bonita, hemos de implementar funcionalidades en ella para dar diferentes efectos, hacerla dinámica, etc… Para ello he recurrido a dos lenguajes de programación: **PHP** y **JavaScript**. El **PHP** ya ha sido explicado en el punto anterior de este documento. También aparece aquí porque es un intermediario entre el servidor y el navegador. Si se quiere saber más sobre **PHP** , se puede consultar [9],

El **JavaScript**, hasta este punto, aún no ha sido casi usado en la aplicación web. El **JavaScript** es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Nos será muy útil a la hora de implementar todas las gráficas en el lado del administrador, para así poder ver las estadísticas de los alumnos de una manera funcional y estética. Si se quiere saber más sobre **JavaScript**, se puede consultar [10].

LENGUAJE	**FUNCIONALIDAD**
HTML	ESTRUCTURA WEB
CSS	ESTÉTICA WEB
PHP	CONEXIÓN DIRECTA CON SERVIDOR
JAVASCRIPT	FUNCIONALIDAD WEB

Tabla 1. Lenguajes de programación

Tras ver las tecnologías que están a nuestro alcance y analizar el entorno en el que se va a desarrollar mi proyecto, a continuación se muestra el diseño usado para la creación del módulo de seguimiento de los alumnos.

3 Diseño

3.1 Diseño de la base de datos

3.1.1 Datos necesarios para el módulo de seguimiento

Antes de empezar a retocar la base de datos ya creada, analizaremos los datos que nos interesaría recoger de cada usuario, los datos que hemos de retocar en la base de datos ya creada y, por último, como relacionar los datos entre si de una manera estrictamente conexa, eficaz e intuitiva.

Datos necesarios para realizar el seguimiento de un alumno

- Errores cometidos por cada alumno en cada ejercicio.
- Errores totales por ejercicio dentro de un grupo.
- Tiempo requerido por cada alumno para realizar un ejercicio.
- Cuándo y dónde accede un alumno.
- Historial de respuestas enviadas por cada ejercicio.
- Media de tiempo requerido por ejercicio de un grupo.
- Media de errores por ejercicio de un grupo.
- Nota del alumno por curso.

3.1.2 Base de datos a implementar

A continuación (Figura 7) se muestra el modelo Entidad-Relación de la nueva base de datos. Como se puede observar, se han añadido tablas a la base de datos y también se han modificado algunas de la tablas ya creadas. Para notar dicha diferencia hay que comparar la tabla actual (Figura 7) con la heredada del proyecto anterior (Figura 1).

Se ha creado una tabla nueva para la creación del log por usuario. Esto es un seguimiento del alumno al acceder a la web y navegar por ella. Esta tabla almacena el Id del usuario en cuestión, la fecha en la que se captura el movimiento del alumno, el tipo de movimiento que realiza (envío de respuesta, ver ejercicio, entrada a curso) y, finalmente, la información relacionada con el suceso captado, ampliando así la información antes dada por el tipo de movimiento.

Además, se ha creado otra tabla que almacena la nota del usuario por cada curso que tiene. Esto le servirá tanto a él como para el profesor para ver si su rendimiento es el adecuado. Más adelante se explica la manera de presentarle al alumno su nota, ya que se quiere hacer que la experiencia para el alumno sea de lo más dinámica y entretenida.

Finalmente, se ha modificado la tabla que almacena los ejercicios realizados por cada alumno para que tenga un historial de las respuestas enviadas por cada alumno, por si el profesor desea revisarlas en caso de cualquier duda, el tiempo expedido en el ejercicio y si el alumno ha superado o no el ejercicio.

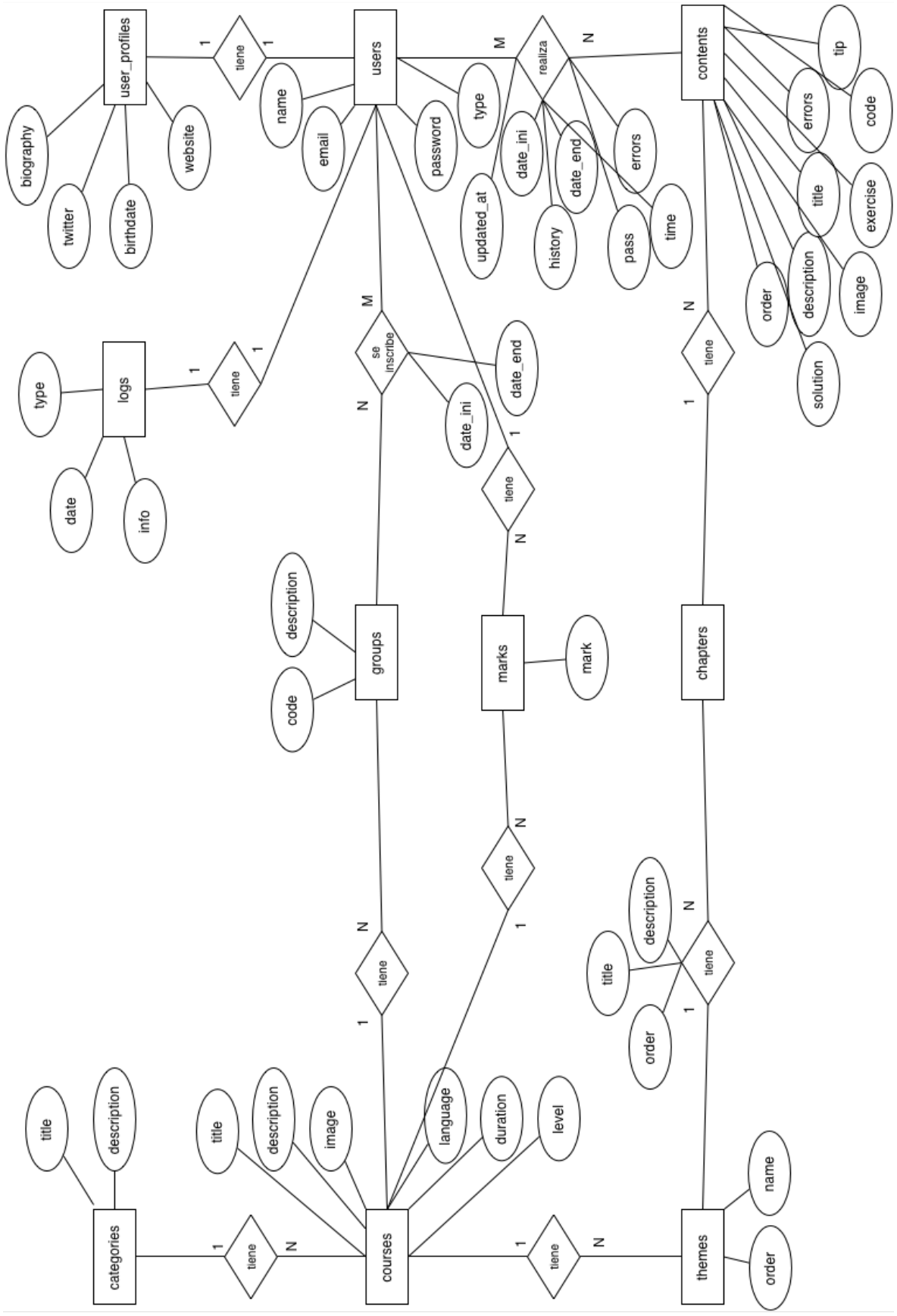

Figura 7. Modelo Entidad-Relación Codeps (Después)

3.2 Diseño estético de la aplicación

A continuación se mostrarán los aspectos estéticos que se cambiarán a lo largo del desarrollo del módulo de seguimiento del alumno. Para la realización de este apartado partimos de la sección 2.2.2 donde analizaba la estética de la web heredada.

3.2.1 Mejoras estéticas

Lo primero que realizaré será colocar un fondo para la página web que acompañe a lo largo de toda la navegación. Después me centraré en mejorar la estética general de la presentación de los cursos y sus ejercicios. El fondo escogido para la web es un fondo pizarra fijo, de tal manera que cuando realices un scroll el fondo se quede quieto mientras los elementos de alrededor se muevan.

La presentación de las imágenes de los diferentes cursos se centrará. Una vez dentro de un curso se eliminarán los elementos superiores relativos a la duración y dificultad del curso ya que no tienen utilidad alguna al no estar aún implementada. Al acceder a un ejercicio, éste será presentado de manera totalmente diferente y añadiéndole funcionalidades para mayor comodidad del usuario. Más adelante, en el desarrollo del proyecto, se podrá ver la nueva distribución de los elementos dentro de la página y las diferencias con antes.

Además, se añadirá en la cabecera de la web la nota de cada alumno, pero no de manera numérica. Para que el alumno relacione los ejercicios como un juego y así sea su resolución menos monótona la nota aparecerá a modo de vidas. Esto quiere decir que cuando el alumno vea diez corazones tendrá un diez y si ve cinco pues tendrá un cinco respectivamente. Obviamente, el profesor verá las notas de los alumnos de manera numérica, ya que a él le interesan más los datos en sí que su presentación.

3.2.2 Diseño del módulo de seguimiento

El diseño estético del módulo de seguimiento del alumno estará dividido, principalmente, en tres páginas diferentes. En estas páginas, del Front-End, tendrán que ser dinámicas y sugerentes para el usuario, en este caso el profesor de un curso.

La primera página será para ver los cursos y grupos de los cuales es profesor el usuario. Aquí se hace el primer filtrado de estadísticas de los alumnos. Por cada grupo de curso puedes elegir entre ver las estadísticas o ver los datos de los alumnos.

La página relativa a las estadísticas te muestran varias gráficas, totalmente dinámicas y actualizadas, donde se verán los datos de cada alumno en función a una media para poder tener una idea generalizada, también, de los alumnos que constituyen el grupo en cuestión. Cada gráfica tendrá la posibilidad de eliminar o incluir alumnos dentro del visionado de la gráfica, para poder centrarte en unos u otros. Además las gráficas podrán ser guardadas en diferentes formatos de manera sencilla, para así poder hacer un estudio en un momento determinado.

La página relacionada con los datos de los alumnos mostrará a la derecha una tabla con la media de errores, en ese curso y grupo, por ejercicio. Esto facilitará la comparación de datos del alumno respecto con el colectivo. A la izquierda habrá una tabla desplegable con los nombres de todos los alumnos inscritos al curso y grupo. Al seleccionar cualquier nombre,

éste desplegará su información relacionada. La información que aparecerá será: en qué ejercicio se encuentra el alumno, avance dentro del curso, errores del alumno en cada ejercicio, la nota y además un botón para poder bajarte los movimientos relacionados con ese alumno, hasta ese momento, en formato *xml*.

3.3 Diseño de las funcionalidades del servidor

A continuación se mostrarán las funcionalidades a desarrollar en la parte del servidor para poder alimentar la base de datos con datos de cada alumno.

3.3.1 Funcionalidades para llevar un registro

Cada alumno, como ya he comentado, es monitorizado en todo momento por la aplicación para que así su profesor sepa en cada momento lo que hace. Esto tiene que realizarse en el Back-End ya que el alumno no se tiene que percatar de que guardan sus movimientos en la aplicación. Es parecido, pero a menor escala, lo que puedes realizar con la herramienta de Google usada para llevar un seguimiento de las personas que entran, salen, etc., dentro de una página web cualquiera. Esta herramienta es conocida como Google Analytics, que sirve para llevar la analítica web de una página.

Para la realización de esto en código PHP habrá incrustaciones de información detallada cada vez que el alumno pulse algún botón o cambie de página dentro del propio entorno de la aplicación. A continuación (Figura 8) se muestra un ejemplo de cómo el programa llevará a cabo el guardado de datos de los movimientos del usuario.

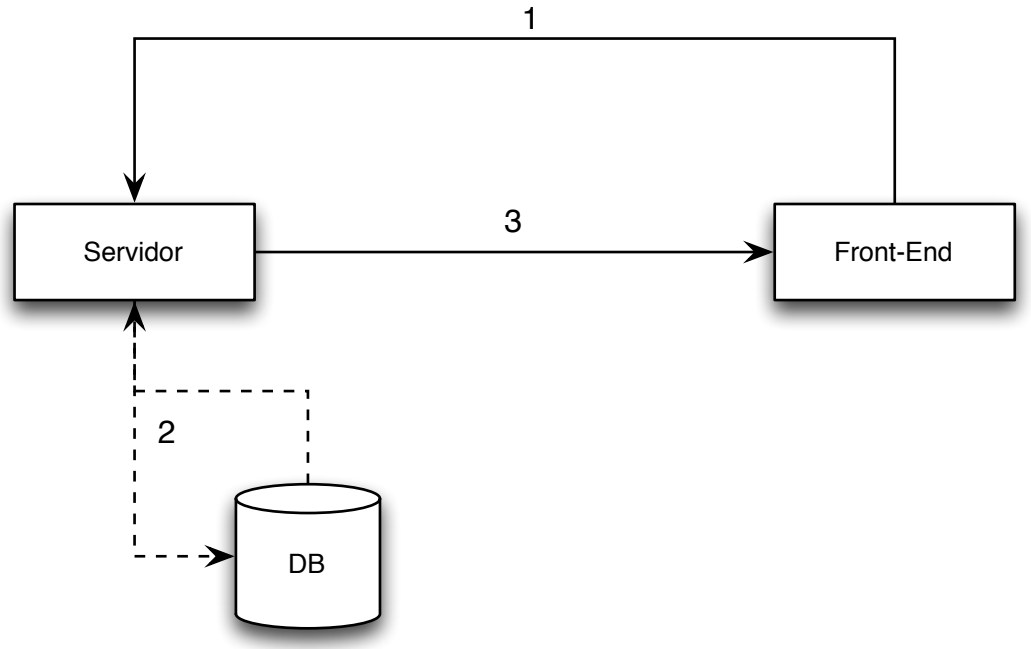

Figura 8. Ejemplo de almacenaje de datos del seguimiento

- Paso 1 : El alumno pulsa, por ejemplo, el botón de enviar una respuesta de un ejercicio. Esto es enviado al servidor para que ejecute lo pertinente al pulsar dicho botón.
- Paso 2 : El servidor recibe la orden y ejecuta el código relacionado con la llamada ejecutada por el alumno. Mientras tanto, el servidor registra el movimiento realizado

por el alumno en la base de datos con todas las características: tipo, información y hora. La base de datos devuelve un OK si todo ha ido bien.

- Paso 3 : El servidor tras recibir el OK de la base de datos envía al usuario (Front-End) la información solicitada. En este caso, al tratarse del envío de un ejercicio, responderá si el ejercicio está o no bien.

3.3.2 Funcionalidades de recopilación de datos de usuario

Este apartado explica el esquema a seguir para recopilar los errores, nota, tiempo invertido y avance del usuario. En resumen, todos los datos que serán usados al mostrar la página de datos de cada usuario.

Para recopilar los errores de cada usuario tendrá cada uno un contador en los ejercicios que envíen respuesta. Este contador aumentará en el caso de que el alumno falle y así podrá saberse la cantidad de fallos que comete por ejercicio y monitorizar en cual tiene mayores problemas.

La nota funciona prácticamente igual que los errores, lo único es que ésta aumentará cuando acierte una respuesta. La nota en un principio está inicializada a 10. Disminuirá según el usuario vaya cometiendo fallos en los ejercicios.

El tiempo invertido es lo que más trabajo lleva de seguir, ya que no se puede saber al 100% cuánto está el usuario invirtiendo en el ejercicio. No obstante, con funciones de JQuery se puede saber cuándo alguien abandona la página y, así, poder calcular de una manera más exacta cuánto tiempo emplea. De todos modos este sistema es aproximativo ya que no garantiza que los resultados devueltos sean 100% exactos.

Finalmente, para llevar el avance de los usuarios cada ejercicio que estén o hayan resuelto llevará un elemento denominado “pass” que indicará si ha realizado o no el ejercicio de manera exitosa.

3.3.3 Funcionalidades de manipulación de datos recopilados

Vale, hasta el momento ya tenemos una idea de cómo recopilar los datos de los alumnos pero, ¿qué haremos con ellos?.

Empezando por los datos de registro, como ya he dicho antes, podrán ser descargados en cualquier momento por el profesor a través de la página que muestra los datos de usuarios inscritos en un curso y grupo. Con lo que, para realizar esto, habrá que hacer una consulta a la base de datos para que nos devuelva los datos que nos interesen de un alumno y cree un documento Excel para que sea enviado al usuario que solicite esos datos, en este caso el profesor encargado del grupo del alumno.

El resto de datos, como errores , tiempos, avance del curso, etc. serán recopilados todos y se hallarán medias y se expondrán los datos ya tratados. Más adelante me explayaré más en el trato de cada uno de los elementos y como adaptarlos a las gráficas de las que dispondrá el módulo de seguimiento.

3.3.4 Funcionalidades para mejorar la aplicación existente

En este apartado se verán las funcionalidades que han de incorporarse o modificarse a la aplicación ya existente para mejorar su dinamismo y funcionalidad.

Por ejemplo, actualmente se podría acceder a cualquier ejercicio del curso inscrito. Lo suyo es que el alumno tenga una cierta limitación y no pueda acceder a un ejercicio sin haber superado el anterior. Para ello, como antes ha sido mencionado en el diseño de la base de datos, cada vez que un alumno supere un ejercicio, ese ejercicio guardado en la base de datos tendrá una variable, previamente inicializada a **FALSE**, denominada "**pass**". Esta variable nos permitirá controlar el acceso a los ejercicios y, además, nos permitirá controlar enl avance de cada usuario, con lo que realizará una doble función.

Además, cuando un alumno accede a un ejercicio dentro de un curso éste no puede volver al curso a no ser que pulse "atrás" en el buscador. Esto es un tanto tedioso y poco dinámico y el alumno se terminaría cansando. Para ello se situará un botón de vuelta al curso.

Lo mismo ocurre cuando terminas exitosamente un ejercicio. Lo suyo es que la propia aplicación te incite a continuar al siguiente ejercicio, pero no es así. Para ello cada vez que se envíe una respuesta se comprobará si esta es correcta, y si es así aparecerá un botón para poder acceder al siguiente ejercicio.

Hasta ahora al enviar un ejercicio solo se guardaba el último envío que había realizado el alumno. Con lo que, si el profesor quería ver el progreso y por qué un alumno había fallado tanto, no lo podría ver. Para ello se implementará un historial de todos los envíos por ejercicio que realice un alumno.

Todas estas implementaciones será detallas más adelante, en este mismo documento.

3.4 Diseño de las gráficas

A continuación se explicará la biblioteca escogida para la creación de las gráficas: **Highcharts JS**.

3.4.1 ¿Qué es Highcharts JS?

Highcharts es una biblioteca de gráficas escrita en JavaScript puro, ofreciendo un modo fácil de añadir una gráfica a su sitio web o el uso de web. **Highcharts** actualmente implementa la línea, la ranura, el área, areaspline, la columna, la barra, la tarta, dispersarse, medidas angulares, arearange, areasplinerange, columnrange, la burbuja, embalar el argumento, barras de error, embudo, cascada y tipos de carta polares. Esto nos ofrece una amplia gama para crear y moldear las gráficas a nuestro antojo. Además, la licencia de Highchart es gratuita en el caso de que su uso sea no comercial. Si quieres conocer más sobre Highcharts, véase [11].

3.4.2 Características de HighCharts JS

Compatibilidad

Funciona en todos los navegadores modernos móviles y de escritorio, incluyendo el iPhone / iPad e Internet Explorer desde la versión 6. En iOS y Android, soporte multitouch proporciona una experiencia de usuario sin fisuras. navegadores estándar SVG utilizan para el procesamiento de gráficos. En el legado se dibujan los gráficos de Internet Explorer usando VML.

Abierto

Una de las características clave de Highchart JS es que se permite descargar el código fuente y hacer sus propios cambios. Esto permite modificaciones personales y una gran flexibilidad.

JavaScript puro

Highchart JS se basa únicamente en las tecnologías de navegación nativas y no requiere plugins del lado del cliente como Flash o Java. Además no es necesario instalar nada en su servidor. Sin PHP o ASP.NET. Highchart JS sólo necesita dos archivos JS a ejecutar: El núcleo highcharts.js y, o bien el jQuery, MooTools o marco Prototipo. Uno de estos marcos es más probable que ya están en uso en su página web.

Numeroso tipos gráficos

Highchart JS admite una línea, spline, área, areaspline, columnas, barras, circular, dispersión, medidores angulares, arearange, areasplinerange, columnrange y tipos de gráficos polares. Muchos de éstos se pueden combinar en un solo gráfico.

Configuración sencilla de sintaxis

Ajuste de las opciones de configuración Highchart JS no requiere conocimientos especiales de programación. Las opciones se dan en una estructura notación de objetos JavaScript, que es básicamente un juego de llaves y valores conectados por dos puntos, separados por comas y agrupados por corchetes.

Etiquetas de información sobre el gráfico

Cuando sitúas el ratón encima del gráfico puede mostrar un texto de sugerencia con información sobre cada punto y la serie. La descripción se deduce que el usuario mueve el puntero del ratón sobre el gráfico, y se han hecho grandes esfuerzos para que se pegue al punto más cercano, así que es fácil de leer un punto que está por debajo de otro punto.

Exportar e imprimir

Con el módulo de exportación activada, los usuarios pueden exportar el gráfico a formato SVG PNG, JPG, PDF o en el clic de un botón, o imprimir el gráfico directamente desde la página web.

Carga de datos externa

Highchart JS toma los datos en una matriz de JavaScript, que se puede definir en el objeto de configuración local, en un archivo separado o incluso en un sitio diferente. Por otra parte, los datos pueden ser manejados a Highchart JS en cualquier forma, y una función de devolución de llamada utilizan para analizar los datos en una matriz.

Figura 9. Logo de Highchart

4 Desarrollo

4.1 Estudio de Laravel y código previamente creado

Antes de desarrollar el módulo de seguimiento he tenido que estudiarme cómo funciona Laravel, que es el Framework usado en la aplicación, y el código heredado en el proyecto. Para llevar a cabo esta parte he recurrido, principalmente, a la memoria del Trabajo de Fin de Grado creado por Felipe Millán el año pasado. Esta memoria explica perfectamente el funcionamiento de Laravel y, además, era necesario para saber la estructuración del proyecto, tanto base de datos como código. Por tanto, todo lo necesario para saber cómo funciona esta aplicación está distribuido entre este documento y el TFG llamado "Aplicación web para la enseñanza de C".

4.2 Cambios estéticos iniciales del entorno web

Antes se ha comentado que hay varios aspectos a mejorar en la estética del entorno web de la aplicación. A continuación veremos cómo realizar los cambios necesarios para crear una web dinámica, bonita y estéticamente asequible para el usuario.

4.2.1 Cambios generales

Como principal problema en la estética es el fondo inexistente que tiene. Esto provoca mucha iluminación y no queda atractivo, a mi gusto. Por ello he seleccionado un fondo que me parece idóneo a la temática de la aplicación. Este fondo tampoco ha de ser llamativo porque, sino, los usuarios se distraerían con él y no se centrarían en la materia. Además el fondo lo he colocado como "fixed". Esto quiere decir que el usuario tendrá la sensación que todo el contenido se desplaza exceptuando el fondo que se mantiene quieto. Este efecto me parece de lo más atractivo y no afecta en nada al entorno. A continuación, se puede ver una imagen de la entrada a la web antes(Figura 10) y después(Figura 11).

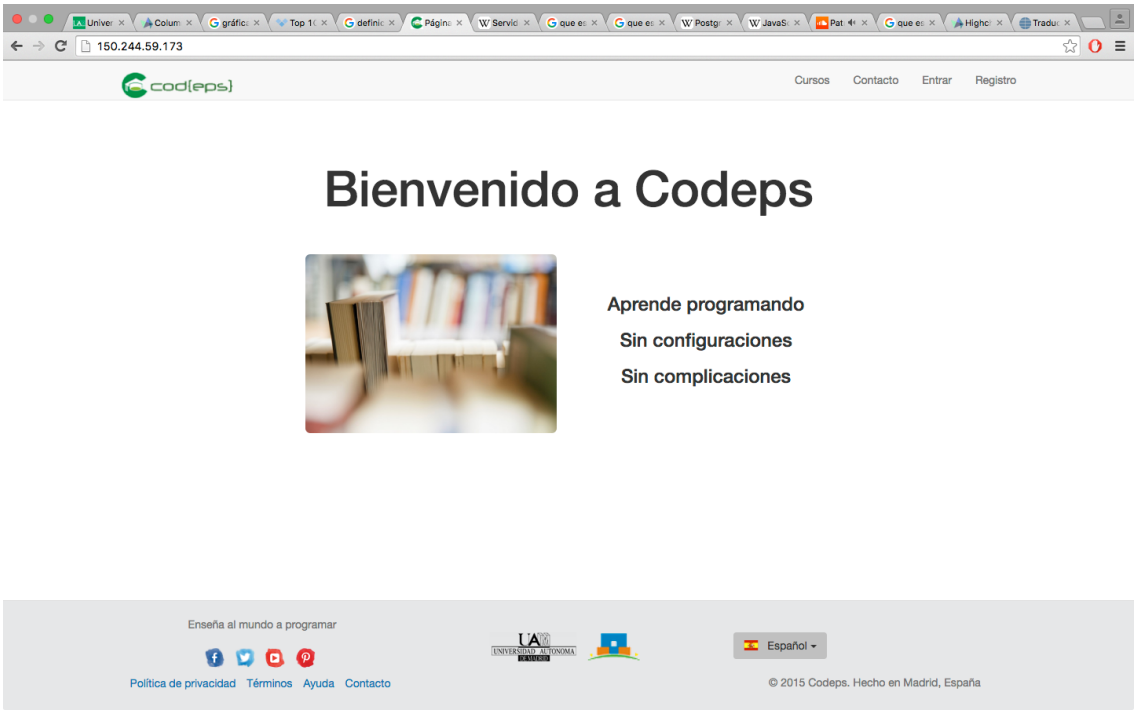

Figura 10. Entrada antes en Codeps

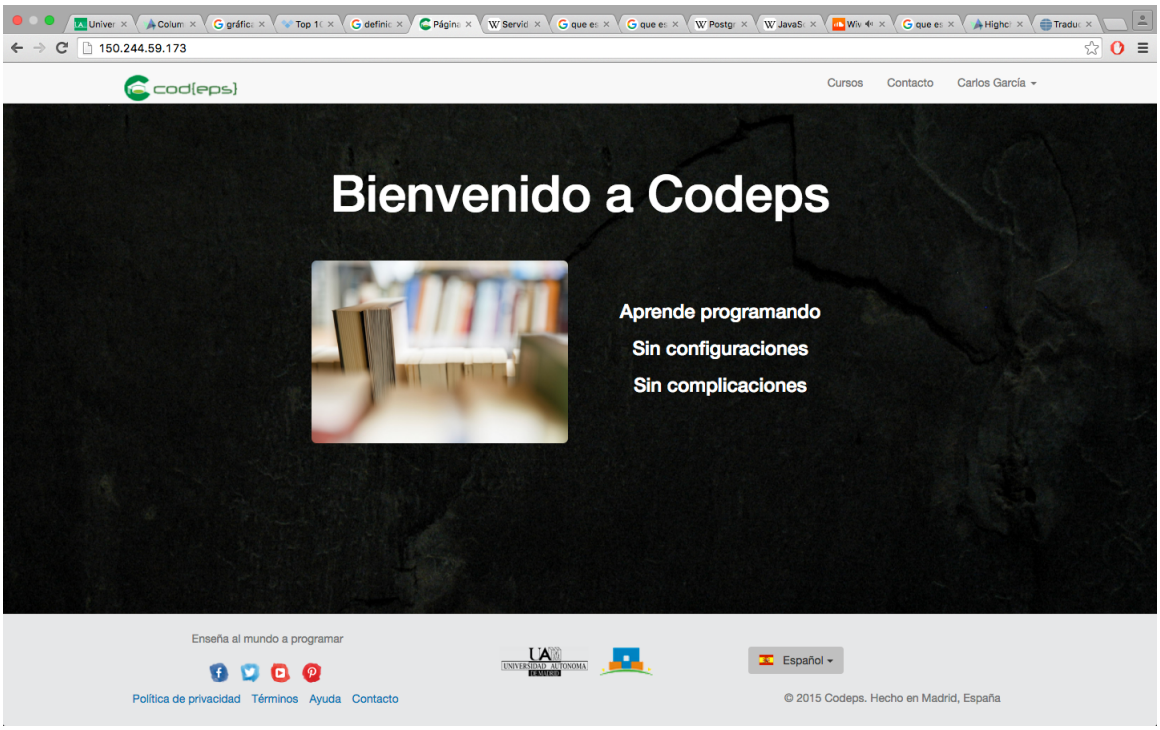

Figura 11. Entrada después en Codeps

Este fondo estará presente a lo largo de toda la web. Además, en la presentación de los cursos disponibles para matricularse se modifica la vista para que estén alineados al centro en vez de a la derecha, lo que hace un presentación más limpia y vistosa para el usuario.

Ya en la página del curso se han eliminado de la cabecera datos que no servían y se ha mejorado la presentación de los ejercicios, ya que antes te aparecía una lista desplegable de los ejercicios por tema ya desplegada menos el primero. Lo suyo es que aparezca la lista desplegable de temas, sin desplegar, y cuando selecciones un tema ya se desplieguen los ejercicios de dicho tema.

4.2.2 Cambios en la estructura del ejercicio

Al acceder a un ejercicio se ha cambiado la forma en que éste es presentado. A continuación se verá el antes (Figura 12) y después(Figura 13) de un ejercicio. Como se podrá apreciar, la estructura ha cambiado totalmente. En la versión anterior se aprecia cómo el ejercicio está estructurado de manera que se vea primero el ejercicio y después la zona de resolución del código. Esto es poco práctico ya que el alumno, en caso de duda y querer revisar lo que le pide el ejercicio, no puede hacerlo según vaya codificando porque ha de hacer un scroll para verlo. Por ello, en la nueva estructuración de la página se ha dividido en una columna el ejercicio y en otra la zona de codificación, basándome en la estructura que sigue CodeAcademy, ya que es bastante acertada.

Además, se ha insertado un botón de vuelta al curso para que sea más dinámica la navegación. Anteriormente si querías volver al curso tenías que darle al navegador. De esta manera ahora se navega en la propia página de manera más intuitiva y directa.

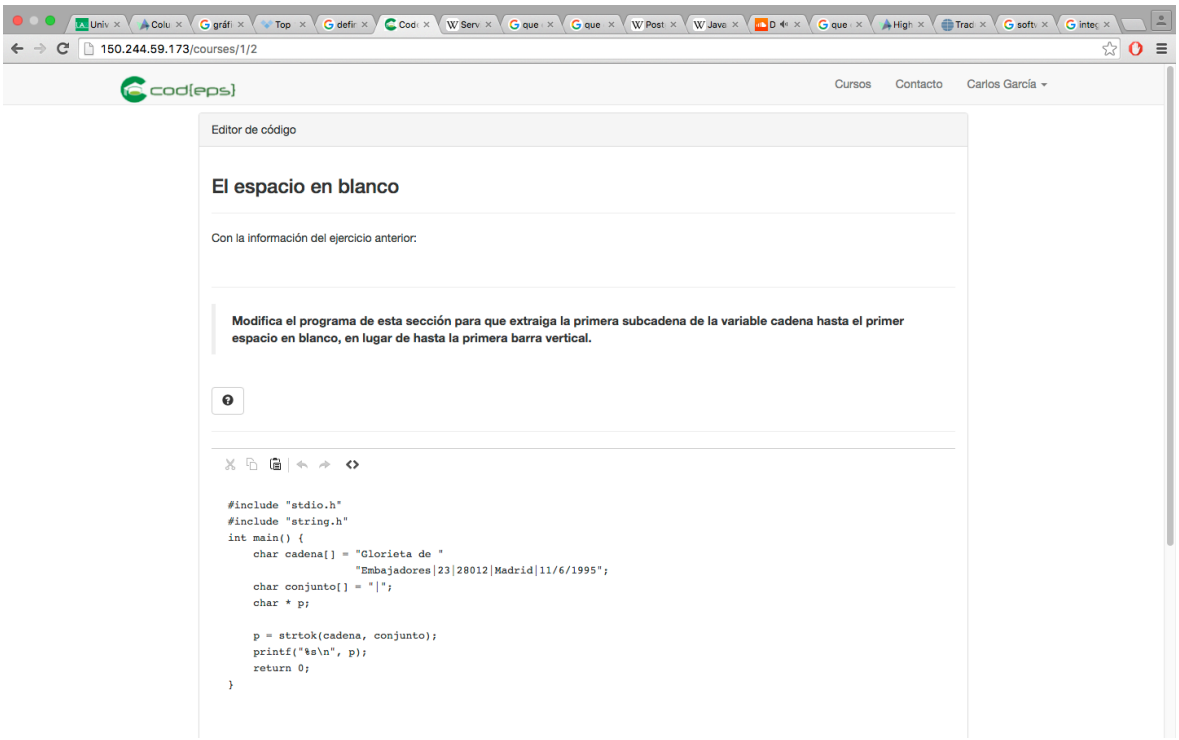

Figura 12. Ejercicio antes de Codeps

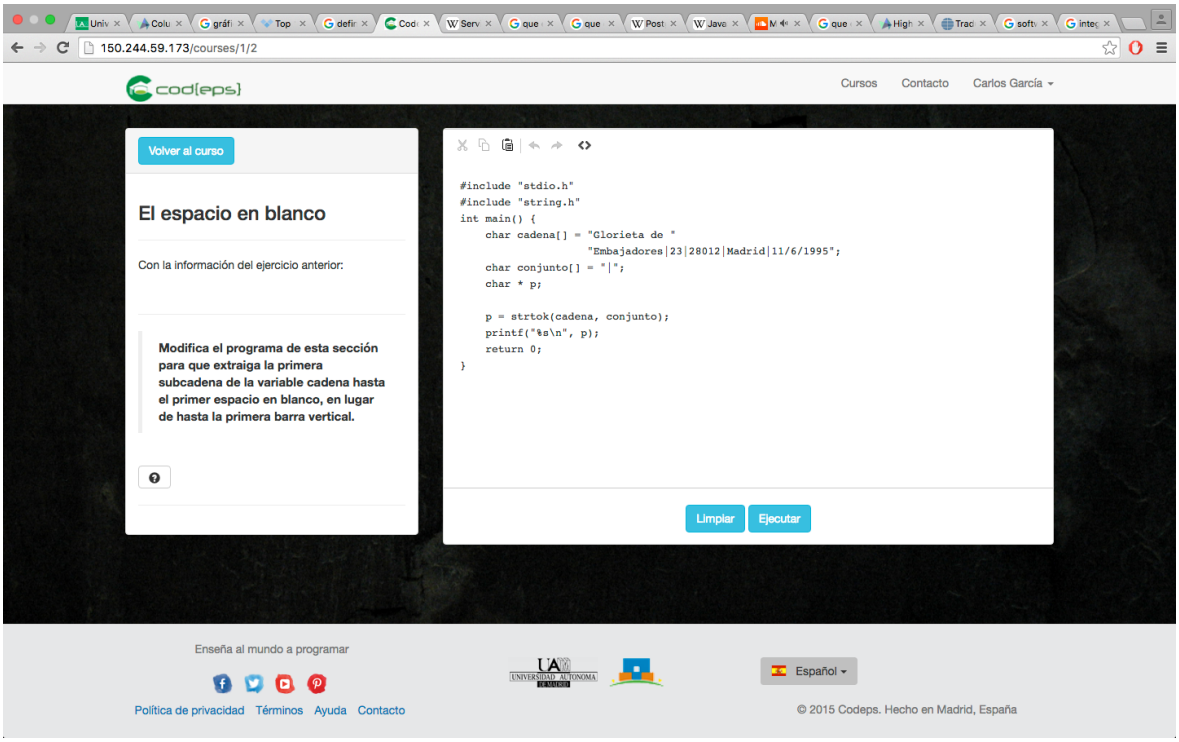

Figura 13. Ejercicio después en Codeps

4.3 Implementación de captura de datos en la aplicación

En esta sección se explicará cómo se ha llevado a cabo la implementación del código para recoger los datos propicios de los alumnos.

4.3.1 Captura de datos para la creación de logs

Una de las funcionalidades que tiene este módulo es la de llevar un seguimiento exhaustivo de lo que hace el alumno en el entorno y llevar un seguimiento del mismo por parte del profesor. Para ello primeramente se ha creado una tabla en la base de datos para que guarde estos datos.

En el modelo entidad-relación (Figura 7) se puede observar la tabla a implementar logs. Además de añadirla a la base de datos, para realizar pruebas iniciales, he tenido que añadir esta tabla al entorno de Laravel, ya que éste tiene una zona para la base de datos e implementación directa, que nos ofrece la posibilidad de crear, borrar, etc. la base de datos que está sujeta a la aplicación.

Más adelante se mostrará la presentación de los datos recogidos hacia el profesor y cómo ha sido desarrollado.

4.3.2 Captura de datos para su análisis

Como hemos visto antes, el profesor de un grupo de un curso podrá ver en línea datos, estadísticas y gráficas de alumnos. Antes de implementar estas páginas hemos de recoger los datos que serán tratados por ellas. Para ello remontamos, otra vez, al modelo entidad-relación (Figura 7) para fijarnos en los datos a almacenar en la base de datos y por ello añadirlos a ésta.

Los errores por alumno ya estaban implementados en la base de datos pero, además, he añadido otro elemento de errores pero que va relacionado al ejercicio en sí y no al alumno. Esto lo he hecho para después poder calcular una media de errores por alumnos de manera más sencilla.

Para llevar un seguimiento de lo que tarda un alumno en la realización de un ejercicio, en un principio, había pensado en reutilizar los atributos “created_at” y “date_end” para calcular el tiempo expedido. Esto no servía porque salían datos poco fiables, ya que no contemplaba la opción de que un alumno se saliese del ejercicio y volviese al él tiempo después, por lo que realmente ese tiempo no habría sido expedido en él. Para solucionar este problema he añadido otro atributo “time” que funcionará a modo de contador de segundos.

He creado también el atributo “pass” que tiene dos funcionalidades principales. La primera es para saber el avance que lleva el usuario dentro de un curso. La segunda es para limitar el acceso a ejercicios por parte del alumno cuando aún no ha acertado el anterior ejercicio. Esto permite al profesor la seguridad de que ningún alumno se salte ningún ejercicio.

Por último se ha añadido una tabla para guardar las notas de los alumnos en función del curso al que están inscritos.

Todos estos cambios de datos están implementados de tal manera que puedan ser usados de manera independiente para varios cursos y así no mezclar datos. Hoy en día en la aplicación solo hay un curso pero uno de los objetivos del proyecto general es que haya más cursos además del de C.

4.4 Creación de las páginas del módulo

Estas páginas a crear estarán únicamente accesibles para el profesor. Por ello lo primero a hacer es crear el acceso de manera adecuada a la zona de estadísticas. Después dividir los cursos y grupos y ya, por último, mostrar las estadísticas o los datos de los alumnos, respectivamente.

4.4.1 Página de acceso a las estadísticas

Lo primero que he hecho es añadir en el menú de los profesores una opción que es "Estadísticas". Esto les permitirá acceder a los cursos y grupos de los cuales son profesores/administradores y ver todos los datos relacionados con los grupos. Para ello simplemente he añadido a la cabecera el siguiente código. Este código está insertado dentro de una condición para que solo sea visible para los profesores.

```
<li><a href="{{ url('/admin/statistics') }}">@lang('app.header_stats')</a></li>
```

Al pulsar en estadísticas serás redireccionado directamente a una página donde aparecerán los cursos y grupos de los cuales eres el profesor. Esto permite hacer un filtrado directamente de cuáles son los cursos y grupos que te interesan. Cada grupo tiene dos opciones. La primera es la de ver los alumnos y la segunda es ver las estadísticas de los alumnos. A continuación (Figura 14) se muestra el resultado final de la página descrita.

Figura 14. Página de selección de estadísticas por grupo

Obviamente, cada vez que se accede a una página distinta, internamente se están realizando una serie de funcionalidades como recogida de datos, filtrar datos de la base de datos, etc.

4.4.2 Página de alumnos

En esta página se muestran los datos por usuario dentro de un grupo de un curso. Primero veremos una imagen con un ejemplo de la página y después la analizaremos.

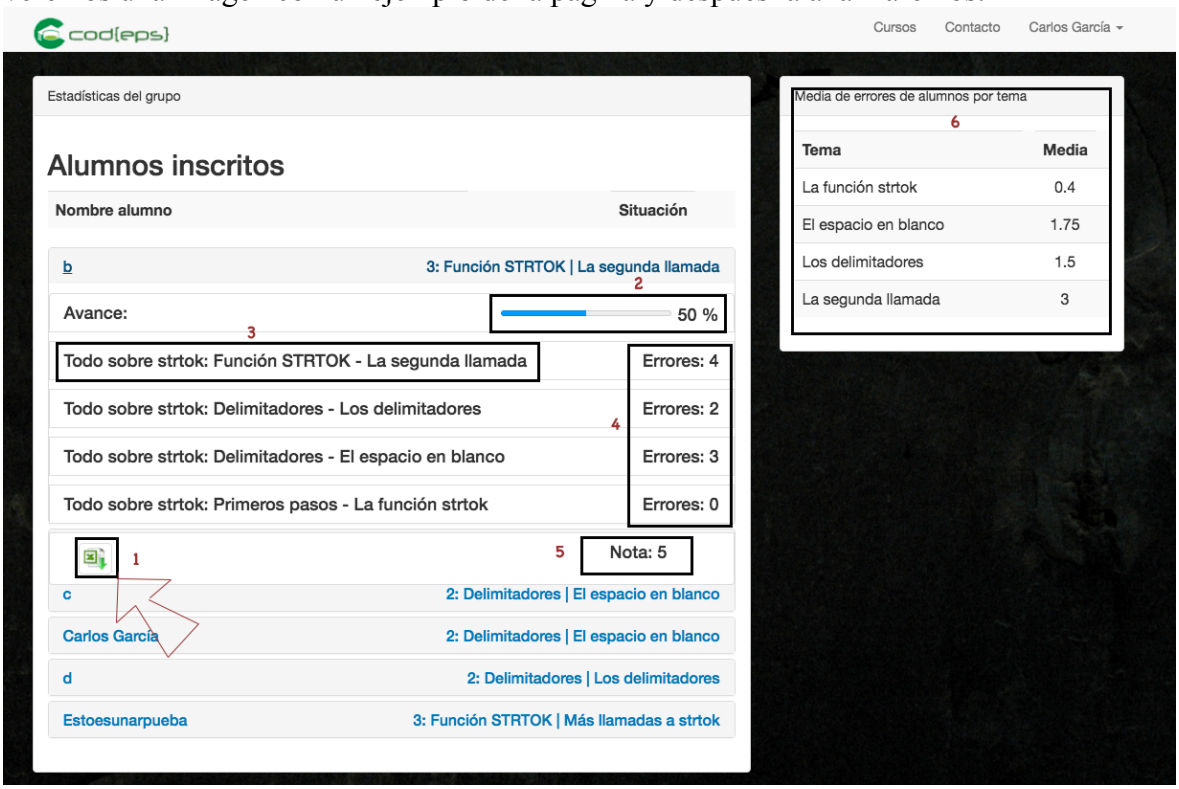

Figura 15. Datos de alumnos en Codeps

Según se puede ver en la imagen (Figura 15), aparece lo que es finalmente la página la cual muestra los datos de los usuarios en un grupo de un curso. La imagen consta de 6 partes fundamentales que serán explicadas a continuación. No obstante, al entrar a esta página no aparece ningún alumno desplegado, como aparece en la imagen, solo aparecen los nombres de los alumnos y en el último ejercicio en el que se encuentran. El profesor puede seleccionar el alumno que desee ver y pulsarlo, y así se le desplegará sus datos relacionados. Esto se ha hecho así para que no ocupe tanto esta página. Vamos a explicar esta página basándonos en los seis puntos señalados en la imagen.

En el **punto 1** se ve un botón con la imagen de descarga de un archivo Excel. Éste botón le servirá al profesor para poder descargarse el log de cada alumno por separado y comprobar qué, cuándo y dónde ha estado el alumno dentro del curso. Más adelante, en las pruebas, se mostrará un ejemplo de dicho archivo Excel. Para rellenar el archivo a descargar he solicitado la información a la tabla "logs" creada anteriormente en la base de datos.

En el **punto 2** se puede apreciar una barra de progreso que muestra el avance del alumno dentro del curso. Así el profesor podrá apreciar cuáles son los alumnos que avanzan más rápido y cuáles necesitan ayuda porque se hayan quedado atascados. Para implementar esta barra me he ayudado del atributo "pass" de la base de datos.

En el **punto 3** aparecen los ejercicios que está o ha realizado el alumno. Primero aparece el nombre del tema al que pertenece, a continuación el capítulo y finalmente el nombre del ejercicio en cuestión.

En el **punto 4** se muestran los errores en cuestión relacionados a los ejercicios mostrados en el **punto 3**. Estos errores son los propios de cada alumno por ejercicio y son almacenados con cada ejercicio en la misma tabla con el atributo "errors", que es inicializado a cero.

En el **punto 5** aparece la nota que lleva el alumno en el global del curso. Esta por supuesto no es la nota que el profesor pondrá finalmente, pero servirá de orientación.

Finalmente, en el **punto 6** se aprecia una tabla con los ejercicios que se han resuelto hasta el momento por, al menos, un alumno y la media de errores hallada tras la media aritmética:

Media errores por ejercicio = (Errores totales del ejercicio/Alumnos superado) * 100

Esta tabla ayudará al profesor a tener en todo momento la vista global de la cantidad de errores por ejercicio que cometen los alumnos y compararlos con el alumno que quiera. Además es una manera directa para ver qué ejercicios tienen poca o demasiada complicación y tomar las medidas necesarias de cara a otro posible curso en el futuro.

En los punto 5 del documento se podrá apreciar la codificación de esta página y las funciones PHP que hay en el servidor para soportar sus funcionalidades.

4.4.3 Página de estadísticas

Tras ver los datos de los alumnos de manera individual el profesor también puede ver esos y otros datos en la página de estadísticas de un grupo de alumnos pertenecientes a un curso. Esto le ayudará a ver de manera más directa cómo los alumnos resuelven los ejercicios. A continuación, presentaré imágenes de la página en cuestión y las iré explicando.

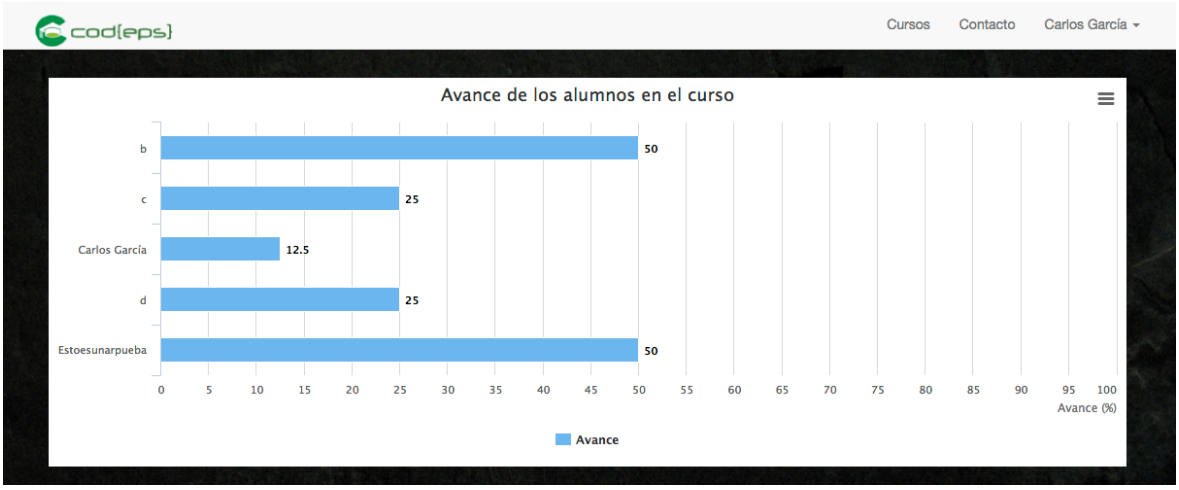

Figura 16. Gráfica de avance de alumnos

En la **figura 16** aparece la primera gráfica que nos encontramos en esta página. Ésta presenta el avance de los alumnos en el curso. De esta manera se puede ver mejor quién es el alumno aventajado y quién necesita un poco de ayuda. Esta y todas las gráficas han sido creadas a raíz de la biblioteca Highcharts JS antes explicada. En el eje X se sitúa, en porcentaje, el avance del alumno y en el eje Y los alumnos que cursan el curso.

Como en esta y en el resto de gráficas, aparecerá un icono de tres rayas en la parte superior izquierda de la tabla. Al pulsarlo despliega un menú para poder descargarte la gráfica en diferentes formatos y la posibilidad de imprimirlo directamente.

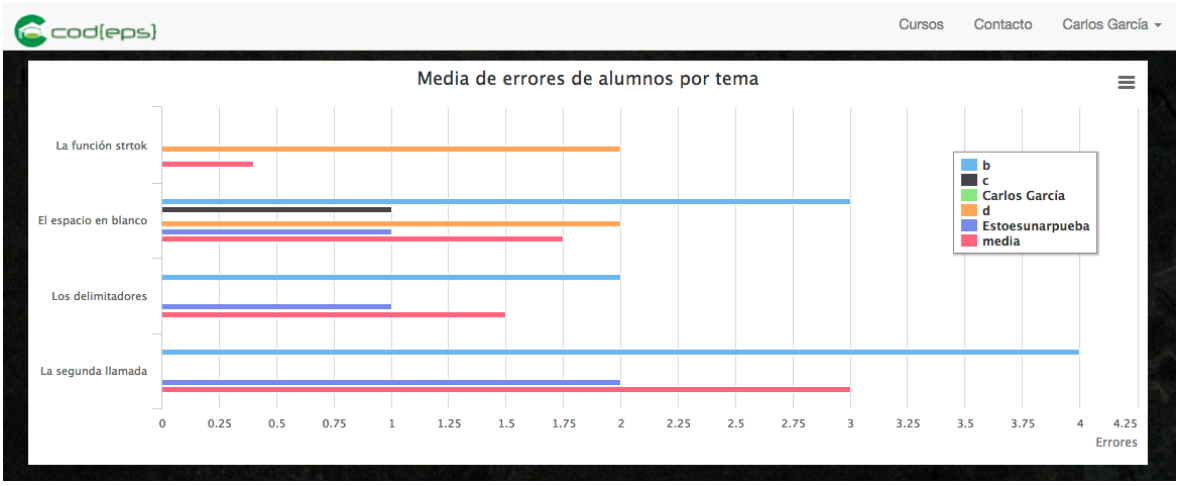

Figura 17. Gráfica de barras horizontales de media errores por alumno

En la **figura 17** se observan los errores que tiene cada alumno por ejercicio, además de la media de errores de todos ellos para tener una referencia. En el eje X se sitúa en número de fallos y en el eje Y los diferentes ejercicios. Cada color de barra representa un alumno que se puede ver en la leyenda situada arriba a la izquierda. Esta, como las siguientes gráficas son dinámicas y puedes ver o dejar de ver un alumno en la gráfica simplemente pinchando en su nombre en la leyenda. Así, si quieres comparar solo un alumno con la media, solo has de eliminar al resto pulsando sobre ellos.

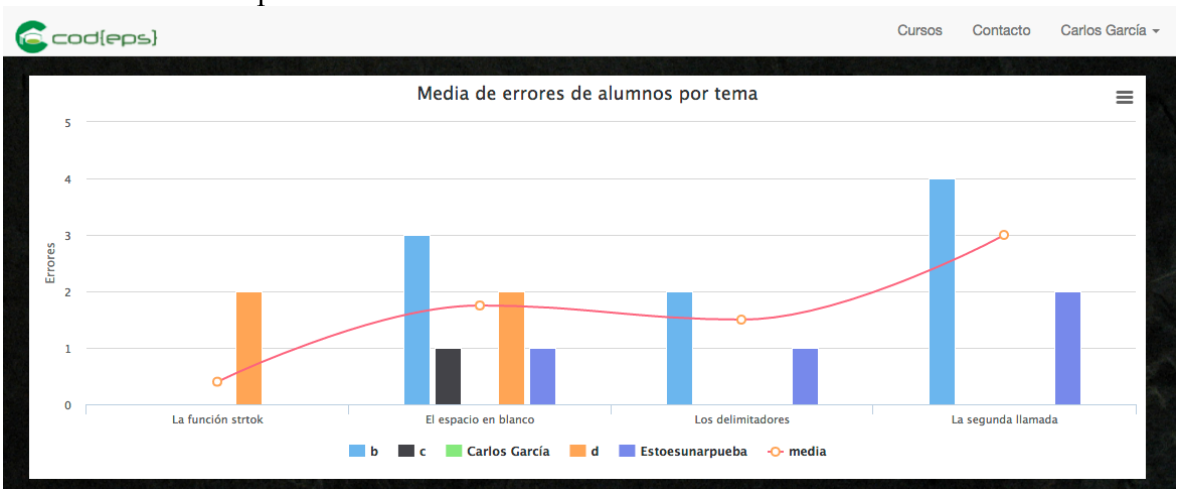

Figura 18. Gráfica de barras verticales con media en línea

Como podemos observar, en la **figura 18** se puede apreciar de una manera más clara y vistosa lo mismo que representa la **figura 17**. En este caso el eje X contiene los diferentes ejercicios y el eje Y el número de errores. Para saber exactamente el valor de cualquier barra solamente hay que situar el ratón encima de él y saldrá una información más detallada. La leyenda de la gráfica, en este caso, aparece en la parte inferior, pero también es seleccionable como la **figura 17.**

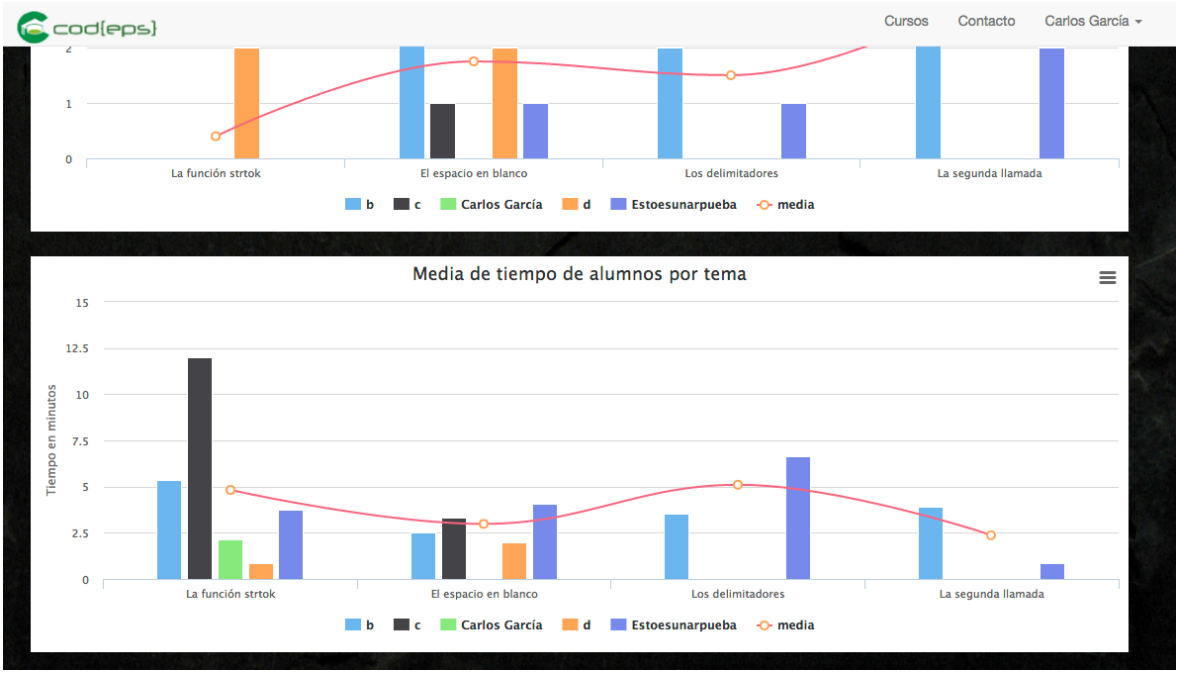

Figura 19. Media de tiempos del alumno empleado por ejercicio

En la última gráfica se representan los tiempos, en minutos, expedidos por cada alumno, y la media de todos ellos, en cada ejercicio. En el eje X están los ejercicios y en el eje Y está el tiempo en minutos. Al igual que en las gráficas anteriores si quieres saber con exactitud el tiempo empleado en algún ejercicio por un alumno en concreto, solo tienes que situar el ratón encima de la barra. La recopilación de estos datos ha sido la más costosa debido a la poca exactitud inicial que había en los datos, y que luego fueron mejorados.

Para la implementación de todas estas gráficas ha sido necesario la codificación, a la vez, de código Javascript con PHP, como se podrá apreciar en el punto 5 del documento. Estos contendrán el código creado para generar estas gráficas.

4.5 Mejoras añadidas

A continuación se explicarán funcionalidades nuevas implementadas, independientemente de la implementación del módulo de seguimiento.

4.5.1 Representación de la nota al alumno

En la cabecera del entorno web se ha añadido, como se podrá ver en la siguiente imagen (Figura 20), la nota del alumno a modo de vidas para hacerlo más amigable.

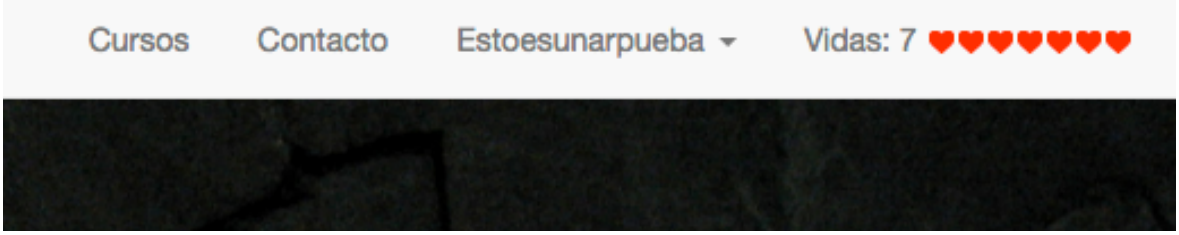

Figura 20. Parte derecha de la cabecera de Codeps

4.5.2 Limitación de acceso a ejercicios

Se ha implementado de manera exitosa la limitación al alumno de no poder acceder a cualquier ejercicio sin haber resuelto antes el anterior. Esto se hace para que el profesor esté seguro de que todos los alumnos siguen el temario.

4.5.3 Mejora en la navegación entre ejercicios

Anteriormente en el entorno web, cuando accedías a un ejercicio no se podía salir de él navegando, solo era posible a través de los botones de atrás del navegador. Como antes he explicado he añadido un botón para volver al curso, pero además si el alumno realiza correctamente un ejercicio, recibe una retroalimentación de parte del servidor con un mensaje de enhorabuena, la salida del código escrito y un botón que permite al alumno continuar directamente al siguiente ejercicio, como aparece en la figura 21.

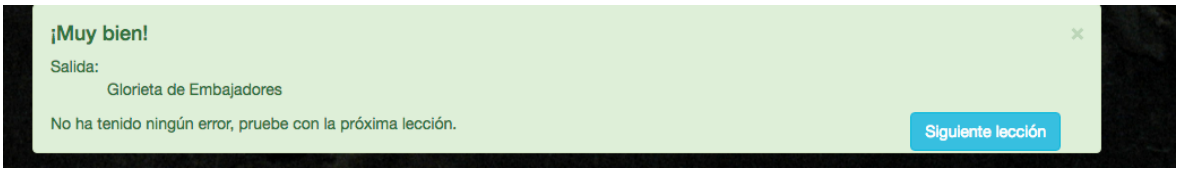

Figura 21. Mensaje al enviar un ejercicio correcto

5 Integración, pruebas y resultados

A continuación, veremos cómo codificar lo explicado en el apartado anterior, las pruebas realizadas en cada una de las funcionalidades implementadas y los resultados obtenidos.

5.1.1 Código en el lado de servidor para recopilar datos

El siguiente código es un ejemplo usado para recoger los datos cada vez que se pulsa un botón, ya sea de acceso a un curso, ejercicio, etc. como para enviar una respuesta a un ejercicio. Cada dato lleva su propio código y es lanzado de maneras distintas, pero el siguiente código es un ejemplo general, en este caso para guardar en log cada vez que el alumno accede a un ejercicio.

Para hacer una **llamada al servidor** se usa:

```
{!! link_to_route('courses.content', 'Siguiente lección', [$course->id, $content->id+1], ['style' => 'float:right', 'class' => 'btn btn-info', 'role' => 'button']) !!}
```

Después en el lado del servidor se trata la ruta y se redirige a la función encargada de devolver una respuesta al lado del cliente. En este caso somos redirigidos a la función que se encarga de guardar la llamada y devolver la página del ejercicio solicitado por el alumno:

```
public function code($courseId, $contentId)
  {
    //registrar el estudio de un contenido
    $user = auth()->user();
    $course = Course::find($courseId);
    $content = Content::find($contentId);
    $aux=$contentId-1;
    $result = \DB::table('content_user')
      ->select('content_user.pass as order')
      ->where('content_user.content_id', '=', $aux)
      ->where('content_user.user_id', '=', $user->id)
      ->get();

    if($contentId>1){
      if(empty($result)){
        return redirect()->back()->withInput();
      }
      foreach ($result as $item) {
        if($item->order != true){
          return redirect()->back()->withInput();
        }
      }
    }
    //hay que asegurarse que el CODE siempre lleve <pre></pre>
    $content->code = strip_tags($content->code);
    $content->code = '<pre>' . $content->code . '</pre>';
    $content->save();

    if(! $user->isContained($contentId))
    {
```

```
            $user->contents()->save($content);
        }

        $date = date('Y-m-d H:i:s');
        $info =  $course->title.' || '.$content->title;
            /*Aquí realizo la inserción de la llamada con el tipo de llamada, el usuario, el tiempo e información adicional.*/
        \DB::insert('insert into logs (user_id, date, type, info) values (?, ?, ?, ?)', [$user->id, $date , 'Ver ejercicio', $info]);

        return view('course.code', compact('course','content'));
    }
```

Este, como ya he dicho, es un ejemplo de cómo recoger la llamada y registrarla en la tabla "logs" creada en la base de datos. Este código está implementado en muchas funciones más del código del servidor para registrar todos los movimientos posibles del alumno pero, como es obvio, no voy a poner todo el código implementado. Ya tenemos datos de los movimientos del usuario, lo siguiente es conseguir los datos en los ejercicios: errores, tiempo de ejecución, etc.

Nos tenemos que situar en la función que es llamada en el lado del servidor cuando un alumno envía una respuesta, en este caso la función se denomina "**compile()**". Esta función recopilará muchos de los datos relacionados con los datos y gráficas que se visualizarán en el lado del profesor. Veamos cómo ha sido implementado:

Primero se comprueba que el alumno tenga vidas para poder enviar una respuesta. Esto es que el alumno haya llegado a una nota inferior a 1.

Comprobación de notas

```
(...)
$mark = \DB::select('select mark as order from marks where user_id = ? and course_id = ?',[$user->id, $courseId]);
        if($mark[0]->order<=0){
            Session::flash('error', 'No te quedan más vida. Por favor solicite al profesor que le reviva para poder continuar con el ejercicio.');
            return redirect()->back()->withInput();
        }
(...)
```

A continuación se guarda en el historial la respuesta si el alumno tiene vidas.

Historial

```
(...)
            //se limpia el código de etiquetas HTML
        $code = strip_tags(\Input::get('code'));
        $code = htmlspecialchars_decode($code);

        //se guarda la respuesta del usuario en el historial y en respuesta
        if(empty($content->pivot->history)){
            $content->pivot->history = $code;
        }else{
```

```
        $content->pivot->history = $content->pivot->history . "\r\n----------------------------
----------------------\r\n" . $code;
    }
    $content->pivot->response = $code;
    $content->pivot->save();
(...)
```

Después se guarda el código en un archivo, se compila y, finalmente se ejecuta. En caso de fallo los errores aumentarán y bajará la nota. En caso de acierto aumentará la nota, se pondrá como que el alumno ha pasado el ejercicio y se devolverá una respuesta positiva al lado del cliente para que pueda continuar con el curso. En cualquiera de los dos casos siempre se guardará el tiempo que ha tardado el alumno entre que se ha cargado la página y ha decidido enviar el resultado. Veamos el código de cada uno de los datos recogidos.

Tiempo invertido en la ejecución del ejercicio

```
\DB::update('update content_user set time = (time+?) where user_id = ? and content_id =
? and pass=FALSE',[$time, $user->id, $contentId]);
```

Aumentar y disminuir la nota del alumno

```
\DB::update('update marks set mark = (mark+1) where user_id = ? and course_id =
?',[$user->id, $courseId]);
\DB::update('update marks set mark = (mark-1) where user_id = ? and course_id =
?',[$user->id, $courseId]);
```

Marcar el ejercicio como resuelto

```
\DB::update('update content_user set pass = true where user_id = ? and
content_id=?',[$user->id, $contentId]);
```

Todos estos datos serán guardados siempre y cuando el alumno no haya superado el ejercicio. Así se evita que maquillar sus resultados y ampliar su nota.

5.1.2 Pruebas y resultados para comprobar el funcionamiento del código anterior

Acabamos de ver el código final para recopilar datos de los alumnos. Pero este no fue el código que en un principio implemente. Para llegar a esto he realizado varias pruebas con errores.

Empezando por las pruebas ejecutadas para probar los logs registrados de los usuarios. Lo primero era crear un alumno ficticio sobre el que se realizaría las pruebas. Este se llamaría Estoesunaprueba. Lo primero fue implementar la recogida de un dato simple que era el que captaba cuándo pulsaba un botón para acceder al curso. Como esto fue exitoso a la primera solo hubo que complicar la llamada a la base de datos para recopilar los datos necesarios de dicha llamada. Para ello se realizaron pruebas directamente en PostgreSQL creando una consulta que realizase lo mismo que quería en el lado del servidor. Una vez conseguido esto, lo pasé al código del servidor y realicé otra prueba. Al ver que el resultado era el querido no me quedó otra que implementar un código relativamente parecido a lo largo de barias funciones llamadas por el lado del cliente y así recopilar un registro adecuado. Según iba implementando una de estas ejecutaba la prueba correspondiente. Todas fueron exitosas, solo hubo algunos problemas de despiste al dejarme algún “;” o alguna referencia.

Las pruebas realmente duras vinieron cuando implementé el código relacionado con los datos de ejercicios resueltos por el alumno, ya que es manejaba un mayor número de variables que controlar. Los principales problemas provinieron de cuándo debía o no guardar unos u otros datos y dónde los guardaba. Pero al fin y al cabo no se trataba de pruebas a gran escala y lo único que debía hacer es ir despacio y con buena letra. Así me pude dar cuenta de que los datos de tiempo guardados en principio no eran exactos del todo y debía modificar la base de datos para que pudiera guardar también un contador de tiempo.

5.2 Código del lado del cliente

A continuación se muestra parte del código implementado en el lado del cliente. Esto quiere decir que el código aquí descrito es accesible y descargable por el usuario al descargarse en su navegador, lo denominado Front-End.

5.2.1 Código añadido para calcular el tiempo de ejecución

Al tener problemas con el cálculo de tiempos al ver que no eran muy exactos, decidí implementar un contador propio en la página de los ejercicios que luego sería enviado junto con la respuesta. Para ello emplee Javascript y JQuery. Este fue el código implementado:

```
<script>
    $(function () {
        $('[data-toggle="popover"]').popover();
        setInterval('count();',1000);
    })

    contador=0;
    function sendCode(){
        $('#code').attr('action', $('#code').attr('action')+'/'+contador);
        $('#code').submit();
    }

    function count(){
        contador++;
    }
</script>
```

También hubo que modificar la llamada al servidor para que enviase también el tiempo y su ruta de manera interna en la aplicación, ya que Laravel trata a las llamadas mediante rutas.

5.2.2 Código para generar la página de datos de los alumnos

Esta es la parte relevante del código de la página que trata y muestra los datos de los alumnos en la página de datos de los alumnos a la que puede acceder el profesor a través de la zona de estadísticas. Al ser muy extensa será puesta en los anexos. Aquí explicaré las funcionalidades del código implementado.

En primer lugar recibe del servidor los datos en objetos y tienen que ser tratados. Entiéndase objetos como paquetes que contienen otros paquetes, que a su vez contienen otros paquetitos con información. Por lo tanto el código ha de desempaquetar de manera ordenada, en este

caso por alumno, la información que contiene. Así se conseguirán datos como avance, errores y la nota. Además se añade un llamada al servidor, personalizada por cada alumno, para poder descargar su log en el curso. En los anexos de este documento aparecerá un ejemplo del archivo descargado de un alumno. A la vez que son ordenados, los datos son mostrados de manera estética y entendible para el profesor. Por último, solo faltaría recopilar la información que es recibida en relación a los errores totales que hay por ejercicio y el número de alumnos que han superado cada ejercicio y calcular la media de errores por ejercicio. Se muestra en una bonita tabla y punto.

5.2.3 Código para generar el archivo Excel del log de un alumno

Este código está implementado en el lado del servidor, pero me parecí adecuado ponerlo aquí. El código se encarga de recopilar y crear un archivo que será después enviado al cliente en modo de descarga. A continuación se muestra uso dicho código.

```
public function downloadStatistics($id_user){
      //$user_id = 32;
      $user = \DB::select('select name as name from users where id= :id', ['id' => $id_user]);
      $result = \DB::select('select * from logs where user_id= :id', ['id' => $id_user]);
      //$user =  User::findOrFail($user_id);
     // $result = [];
      foreach ($result as $item)
      {
         $log[] = array(
            "Usuario" => $user[0]->name,
            "Fecha"=> $item->date,
            "Tipo"=> $item->type,
            "Informacion"=> $item->info,
         );
      }

      $name = $user[0]->name;

      Excel::create($name, function($excel) use($log) {
         $excel->sheet('Logs', function($sheet) use($log) {

            $sheet->fromArray($log);
         });
      })->export('xls');

   }
```

En el anexo habrá un ejemplo del Excel que se puede descargar un profesor.

5.2.4 Código para generar la página de estadísticas de los alumnos

En esta sección analizaremos el código, disponible en los anexos, relativo a la página que muestra las estadísticas de los alumnos por grupo mediante gráficas. Este punto es muy importante ya que se ha tenido que mezclar código PHP con HTML y JavaScript. Como pasa con la página de datos de los alumnos, la página de estadísticas recibe información a modo de objetos del servidor. Estos objetos contienen toda la información necesaria para poder generar las gráficas y que previamente se han creado en la zona del servidor en una función creada también.

Antes de explicar cómo tratar los objetos, tarea que no ha sido nada fácil, veamos como codificar en JavaScript una gráfica de ejemplo. En este caso seleccionaremos la gráfica de media de tiempos que aparece en la Figura 19. A continuación está el código usado para implementar la gráfica:

```
$('#container3').highcharts({
        title: {
            text: 'Media de tiempo de alumnos por ejercicio'
        },yAxis: {
            min: 0,
            title: {
                text: 'Tiempo en minutos',
                align: 'middle'
            },
            labels: {
                overflow: 'justify'
            }
        },
        xAxis: {
            categories: test
        },
        series: dataOutTime
    });
```

El código anterior consta de varios elementos configurables: título, eje X, eje Y, leyenda y la información. Todo excepto la configuración del eje X y la serie a usar es fijo, ya que no nos importa que sea así. En el eje X tenemos una variable que antes ha sido definida, llamada **test**. Veámoslo:

```
<script>
(...)
var test = [];
(...)
<?php
    (...)
    for ($h = 0; $h < count($result2); $h ++){
        echo("\n test[$h]='");
        print_r(json_decode(json_encode($result2[$h]->title), true));
        echo("';");
    }
    (...)
?>
(...)
</script>
```

A parte de la variable, también existe **dataOutTime**, definida con el fin de crear la serie de datos que irá en la gráfica. A continuación se muestra el ejemplo de cómo ha sido definida esta variable antes de ser usada en la gráfica correspondiente.

```
<script>
(...)
var dataOutTime = [];
(...)
<?php
        $h=-1;
        $name="";
        foreach ($result as $item) {
            if($item->order!=$name ){
                if($h>-1){
                    echo("dataOutTime[$h] = dataInTime;");
                }
                $h++;
                echo("j = $h; \n");
                echo("data = [];");
                echo("dataInTime = {type, name, data};");
                echo("dataInTime.type = 'column';");
                echo("\n dataInTime.name='");
                print_r(json_decode(json_encode($item->order), true));
                echo("'; ");
                $name = $item->order;
            }
            if($item->date_end != ''){
                $finalTime = $item->time/60;
                echo("\n dataInTime.data.unshift(");
                print_r($finalTime);
                echo("); ");
            }
        }
        $total = count($result)-1;
        print_r("\n");
        print_r(json_decode(json_encode($result[$total]->order2),true));
        print_r(";");
        echo("dataOutTime[$h] = dataInTime;");;
        echo("data = [];");
        $auxiliar =0;
        foreach($result3 as $item){
            echo("usersTime[$auxiliar] = ");
            print_r(json_decode(json_encode($item->order),true));
            echo(";");
            $auxiliar++;
        }

    ?>
    var mediaTime=[];
    i=0;
    for(var x=0;x < dataOutTime.length ; x++){
```

```
    for(i=0; i < dataOutTime[x].data.length; i++){
        if(x==0){
            mediaTime[i] = 0;
        }
        mediaTime[i] += dataOutTime[x].data[i]/(usersTime[i]);
    }
}
dataInTime={type, name, data};
dataInTime.type="spline";
dataInTime.name="media";
dataInTime.data = mediaTime;
dataInTime.marker = {lineWidth: 2,
          lineColor: Highcharts.getOptions().colors[3],
          fillColor: 'white'};
dataOutTime[j+1] = dataInTime;
```

En el código visto se ve un claro ejemplo de intercalar código JavaScript con PHP. Este código se encarga de extraer toda la información enviada por el servidor a modo de objeto, **result2,** y reordenarla de tal manera que sea aceptada por la biblioteca en la que desea ser incrustada dicha información, es decir, en la biblioteca Highchart JS. Después de eso lo único que hago es hallar la media de tiempos y añadirla al final del objeto ya creado con una serie de parámetros para que se dibuje una línea en vez de una barra.

6 *Conclusiones* y trabajo futuro

6.1 Conclusiones

Este trabajo ha consistido, principalmente en la creación de un módulo de seguimiento del alumno. No obstante, también ha requerido el estudio previo de la aplicación anteriormente creada y del entorno en el que ha sido programada. Además, también he realizado mejoras estéticas y funcionales que no tenían mucho que ver con el TFG, pero que las he realizado a gusto y con el enfoque de que en un futuro no muy lejano este sea un proyecto grande y usable en la propia escuela.

Cabe decir que termino muy contento este proyecto ya que me ha ayudado a conocer una nueva herramienta para crear aplicaciones web con base en PHP. Espero que el módulo de seguimiento del alumno sirva para que el profesor que lo use ayude de una manera más personalizada a los alumnos y, a su vez, que le ayude a él mismo a evaluar y hacer más llevadera la tarea de evaluación de los alumnos.

6.2 Trabajo futuro

Espero que en un futuro cercano se implementen nuevos cursos de programación diferentes a C. Además, también espero que alguien mejore y profundice más aún en el módulo de seguimiento que he creado, ya que a mi parecer aún quedan muchas cosas que hacer, pero que por desgracia no da tiempo al ser un campo en el cuál uno se puede explayar todo lo que quiera. Esto no quiere decir que mi módulo no sirva, solo digo que como todo proyecto de software tendrá sus mejoras y mantenimiento que lo cambiarán a lo largo del tiempo. Si en cualquier momento se decide de comercializar este software se debe pedir la licencia del código de Highcharts.

Los próximos pasos para poner en marcha esta plataforma de aprendizaje han de ser los de trasladar todo el código y configuración a un servidor de verdad que pueda soportar varias visitas a la vez, adquirir una dirección web y establecer las políticas de privacidad que se quieran adoptar en la web, tanto para la política de cookies como para la interna propia, al almacenar datos privados de la gente.

También cabe mejorar la seguridad en la aplicación. He observado que no se guarda la contraseña cifrada en la base datos. Recomendaría un cifrado SHA256 como mínimo para la contraseña. Además habría que cambiar el HTTP por HTTPS que cifra los mensajes usando un cifrado aritmético. Ello provocaría conseguir un certificado de seguridad para la página. Otra vulnerabilidad del entorno es la intrusión de código dañino directamente en el servidor. Recomiendo implementar algún módulo que filtre lo que el servidor recibe del alumno para ejecutar en terminal. Esto puede provocar efectos nefastos en el servidor desde colapso hasta borrado entero del disco.

Referencias

[1] Felipe Millán Fernández, “Trabajo de fin de grado: Aplicación web para la enseñanza de C”

[2] Wikipedia, “Servidor HTTP Apache”, https://es.wikipedia.org/wiki/Servidor_HTTP_Apache.

[3] Cristina M., “Framework”, http://www.nubelo.com/blog/que-son-los-frameworks/.

[4] Wikipedia, “Laravel”, https://es.wikipedia.org/wiki/Laravel.

[5] Wikipedia, “Modelo-vista-controlador”, https://es.wikipedia.org/wiki/Modelo-vista-controlador.

[6] Wikipedia, “PostgreSQL”, https://es.wikipedia.org/wiki/PostgreSQL .

[7] Wikipedia, “HTML”, https://es.wikipedia.org/wiki/HTML.

[8] Wikipedia, “CSS”, https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada.

[9] Wikipedia, “PHP”, https://es.wikipedia.org/wiki/PHP.

[10] Wikipedia, “JavaScript”, https://es.wikipedia.org/wiki/Javascript.

[11] HighCharts, “Highcharts”, http://www.highcharts.com/.

Glosario

Front-End	Interfaz de la aplicación web
Back-End	Motor de la aplicación web
Excel	Extensión usada para hojas de cálculo
HTML	HyperText Markup Language (lenguaje de marcas de hipertexto)
PHP	Lenguaje de programación de uso general de código del lado del servidor
CSS	Hoja de estilo en cascada
Javascript	Lenguaje de programación interpretado
PostgreSQL	Sistema de gestión de bases de datos relacional orientado a objetos y libre
Gráfica	Tipo de representación de datos, generalmente numéricos, mediante recursos gráficos
Servidor	Aplicación en ejecución (software) capaz de atender las peticiones de un cliente y devolverle una respuesta en concordancia.
Logs	Registro de algo.
Media	Valor característico de una serie de datos cuantitativos objeto de estudio que parte del principio del valor esperado, se obtiene a partir de la suma de todos sus valores dividida entre el número de sumandos.
JQuery	Biblioteca de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.
SHA256	La familia SHA (Secure Hash Algorithm, Algoritmo de Hash Seguro) es un sistema de funciones hash criptográficas relacionadas de la Agencia de Seguridad Nacional de los Estados Unidos (NSA) y publicadas por el National Institute of Standards and Technology (NIST).
Cookies	Pequeña información enviada por un sitio web y almacenada en el navegador del usuario, de manera que el sitio web puede consultar la actividad previa del usuario.

Anexos

A Código relativo a la página de datos de los estudiantes

```
@extends('layouts.master')

@section('title')
   Estadísticas - Codeps
@endsection

{{ $i = 0 }}
{{ $j = 0 }}
{{ $k = 0 }}
{{ $name=null }}
{{ $theme=null }}
{{ $id_user=0 }}

@section('content')
   <div class="container">
      <div class="row">
         <div class="col-md-8 ">
            <div class="panel panel-default">

               <div class="panel-heading">
                  @lang('app.statistics_group_title')
               </div>
               <div class="panel-body">
                  @if (empty($result))
                     <div class="alert alert-info alert-dismissible fade in" role="alert">
                        <button type="button" class="close" data-dismiss="alert" aria-
label="Close">
                           <span aria-hidden="true">&times;</span>
                        </button>
                        <h4>@lang('app.warning_no_students')</h4>
                     </div>
                  @else
                     <h2>@lang('app.students')</h2>
                     <table class="table table-striped">
                        <tr>
                           <th>@lang('app.name_label')</th>
                           <th style="float:right; margin-
right:15%;">@lang('app.situation_label')</th>
                        </tr>

                     </table>
                     <div class="panel-group" id="accordion" role="tablist" aria-
multiselectable="true">

                        @foreach ($result as $item)
```

```
<?php
    $valor =0;
    if($name != $item->order){

        $name=$item->order;
        $id=$item->user_id;
        echo("<div class=\"panel panel-default\">
            <div class=\"panel-heading\" role=\"tab\" id=\"heading$i\">
                <a class=\"collapsed\" role=\"button\" data-toggle=\"collapse\" data-parent=\"#accordion\" href=\"#collapse$j\" aria-expanded=\"false\" aria-controls=\"collapse$j\">
                    <h4 class=\"panel-title\"> $item->order
                        <z style=\"float:right;\">
                            $item->order2: $item->title2 |  $item->title
                        </z>
                    </h4>
                </a>
            </div>");


        echo("<div id=\"collapse$j\" class=\"panel-collapse collapse\" role=\"tabpanel\" aria-labelledby=\"heading$i\">");
        $j++;
        $i++;
        foreach ($result5 as $contents) {
            if($contents->id == $id){
                $percent = ($contents->order/$total_content[0]->order)*100;
                echo("<h4 class=\"list-group-item enabled\">Avance:
                    <z style=\"float:right;\">
                        <progress id='progressbar' value='$percent' max='100'></progress>
                        $percent %

                    </z>
                </h4>");
            }
        }
        foreach($result as $item2){
            if($item2->order==$name ){
                echo("<h4 class=\"list-group-item enabled\">$item2->name: $item2->title2 - $item2->title
                    <z style=\"float:right;\">
                        Errores: $item2->errors
                    </z>
                </h4>");

                $valor++;
            }
        }?>
```

```
                        <!--<div class="col-md-12 list-group-item enabled"
style="margin-bottom:25px;">
                            <div id="container3" style="min-width: 310px; height:
400px; margin: 0 auto"></div>
                        </div>-->
                        <h4 class="col-md-12 list-group-item enabled"
style='float:right;'>
                            <a class="col-md-10 " href="{{ route('admin.stats.log',
$item->user_id) }}">
                                <img href="{{ url('/admin/statistics')}}"
src="/img/excel.png" class="img-thumbnail" alt="Ver detalles">
                            </a>
                            @foreach ($result4 as $item4)
                                <?php
                                if($item4->user_id == $item->user_id){
                                    echo("Nota: $item4->order");
                                }
                                ?>
                            @endforeach
                            </h4>
                        <?php
                    echo("</div></div>");
                    }?>



                @endforeach

            </div>
        @endif
    </div>


</div>
</div>


<div class="col-md-4 ">
    <div class="panel panel-default">

        <div class="panel-heading">
            Media de errores de alumnos por tema
        </div>

        <div class="panel-body">
            @if (empty($result))
                <div class="alert alert-info alert-dismissible fade in" role="alert">
                    <button type="button" class="close" data-dismiss="alert" aria-
label="Close">
                        <span aria-hidden="true">&times;</span>
```

```
                    </button>
                    <h4>@lang('app.warning_no_students')</h4>
                </div>
            @else
                <table class="table table-striped">
                    <tr>
                        <th>Tema</th>
                        <th style="float:right; margin-right:15%;">Media</th>
                    </tr>
                    <?php
                        $i=0;
                    ?>
                    @foreach ($result2 as $item2)
                        <?php
                            if($item2->order != ""){
                                $item3 = $result3[$i];
                                $div = $item2->order / $item3->order;
                                echo("<tr><td>$item2->title</td><td                    style=\"text-
align:center;\"> $div</td></tr>");
                            }else{
                                echo("<tr><td>$item2->title</td><td                    style=\"text-
align:center;\"> 0</td></tr>");
                            }
                            $i++;
                        ?>


                    @endforeach

                </table>

                </div>
            @endif
        </div>


    </div>
  </div>

</div>

@endsection
```

B Ejemplo de log descargado de un estudiante

Usuario	Fecha	Tipo	Informacion
Estoesunarpueba	2016-04-27 11:00:42	Entrada a curso	Programación I
Estoesunarpueba	2016-04-27 11:02:03	Entrada a curso	Programación I
Estoesunarpueba	2016-04-27 11:03:18	Ver ejercicio	Programación I || La función strtok
Estoesunarpueba	2016-04-27 11:03:25	Envio respuesta	Programación I || La función strtok
Estoesunarpueba	2016-04-27 11:03:28	Ver ejercicio	Programación I || La función strtok
Estoesunarpueba	2016-04-27 11:03:35	Ver ejercicio	Programación I || El espacio en blanco
Estoesunarpueba	2016-04-27 11:04:35	Entrada a curso	Programación I
Estoesunarpueba	2016-04-27 11:04:40	Entrada a curso	Programación I
Estoesunarpueba	2016-04-27 11:04:44	Ver ejercicio	Programación I || El espacio en blanco
Estoesunarpueba	2016-04-27 11:04:48	Envio respuesta	Programación I || El espacio en blanco
Estoesunarpueba	2016-04-27 11:04:48	Ver ejercicio	Programación I || El espacio en blanco
Estoesunarpueba	2016-04-27 11:05:14	Envio respuesta	Programación I || El espacio en blanco
Estoesunarpueba	2016-04-27 11:05:15	Ver ejercicio	Programación I || El espacio en blanco
Estoesunarpueba	2016-04-27 11:05:17	Ver ejercicio	Programación I || Los delimitadores
Estoesunarpueba	2016-04-28 13:31:31	Entrada a curso	Programación I
Estoesunarpueba	2016-04-28 13:31:38	Ver ejercicio	Programación I || Los delimitadores
Estoesunarpueba	2016-04-28 13:31:42	Envio respuesta	Programación I || Los delimitadores
Estoesunarpueba	2016-04-28 13:31:43	Ver ejercicio	Programación I || Los delimitadores
Estoesunarpueba	2016-04-28 13:31:59	Envio respuesta	Programación I || Los delimitadores
Estoesunarpueba	2016-04-28 13:32:00	Ver ejercicio	Programación I || Los delimitadores
Estoesunarpueba	2016-04-28 13:32:04	Ver ejercicio	Programación I || La segunda llamada
Estoesunarpueba	2016-04-28 13:32:07	Envio respuesta	Programación I || La segunda llamada
Estoesunarpueba	2016-04-28 13:32:08	Ver ejercicio	Programación I || La segunda llamada
Estoesunarpueba	2016-04-28 13:32:09	Envio respuesta	Programación I || La segunda llamada
Estoesunarpueba	2016-04-28 13:32:10	Ver ejercicio	Programación I || La segunda llamada
Estoesunarpueba	2016-04-28 13:33:34	Envio respuesta	Programación I || La segunda llamada
Estoesunarpueba	2016-04-28 13:33:35	Ver ejercicio	Programación I || La segunda llamada
Estoesunarpueba	2016-04-28 13:33:39	Ver ejercicio	Programación I || Más llamadas a strtok

C Código relativo a la página de estadísticas de los estudiantes

```
@extends('layouts.master')

@section('title')
   Estadísticas - Codeps
@endsection

{{ $i = 0 }}
{{ $j = 0 }}
{{ $k = 0 }}
{{ $name=null }}
{{ $theme=null }}
{{ $id_user=0 }}

@section('content')
   <div class="container">

         <div class="col-md-12" style="margin-bottom:25px;">
            <div id="container4" style="min-width: 310px; height: 400px; margin: 0
auto"></div>
         </div>

         <div class="col-md-12" style="margin-bottom:25px;">
            <div id="container" style="min-width: 310px; height: 400px; margin: 0
auto"></div>
         </div>

         <div class="col-md-12" style="margin-bottom:25px;">
            <div id="container2" style="min-width: 310px; height: 400px; margin: 0
auto"></div>
         </div>

         <div class="col-md-12" style="margin-bottom:25px;">
            <div id="container3" style="min-width: 310px; height: 400px; margin: 0
auto"></div>
         </div>


   </div>
    <script type="text/javascript">
       var i = <?php echo($i); ?>;
       var j = 0;
       var test = [];
       var names = [];
       var data = [];
       var progress = {};
       var name = "";
       var dataIn = {};
       var dataInTime = {};
```

```
var type = "";
var dataOut = [];
var dataOutTime = [];
var usersTime = [];

var alumnos = [];
var err = {};
<?php
   $name = "";
   for ($h = 0; $h < count($result2); $h ++){
      echo("\n test[$h]='");
      print_r(json_decode(json_encode($result2[$h]->title), true));
      echo("';");
   }

   $h=-1;
   foreach ($result as $item) {
      $id=$item->user_id;
      if($item->order!=$name ){
         if($h>-1){
            echo("dataOut[$h] = dataIn;");
         }
         $h++;
         echo("j = $h; \n");
         echo("data = [];");
         echo("dataIn = {type, name, data};");
         echo("dataIn.type = 'column';");
         echo("\n dataIn.name ='");
         print_r(json_decode(json_encode($item->order), true));
         echo("'; ");
         echo("names[$h] ='");
         print_r(json_decode(json_encode($item->order), true));
         echo("'; \n");
         $name = $item->order;

      }
      if($item->date_end != ''){
         echo("\n dataIn.data.unshift('");
         print_r(json_decode(json_encode($item->errors), true));
         echo("'); ");
      }
   }
   $total = count($result)-1;
   print_r("\n");
   print_r(json_decode(json_encode($result[$total]->order2),true));
   print_r(";");
   echo("dataOut[$h] = dataIn;");;
   echo("data = [];");

   echo("progress = {data};");
```

```
    $name=null;
    foreach ($result as $item) {
        if($name != $item->order){
            $name=$item->order;
            $id=$item->user_id;
            foreach ($result5 as $contents) {
                if($contents->id == $id){
                    $percent = ($contents->order/$total_content[0]->order)*100;
                    echo("\n progress.data.push($percent);\n");

                }
            }
        }
    }

    echo("data = [];");
    $i=0;
    foreach ($result2 as $item2){
        $item3 = $result3[$i];
        $div = $item2->order / $item3->order;
        echo("data.push($div);");
        $i++;
    }


 ?>
dataIn={type, name, data};
 dataIn.type = "column";
 dataIn.name="media";
 dataIn.data = data;
 dataOut[j+1]= dataIn;
 console.log(progress);

 $(function () {
    $('#container').highcharts({
        chart: {
            type: 'bar'
        },
        title: {
            text: 'Media de errores de alumnos por ejercicio'
        },
        subtitle: {
            text: ''
        },
        xAxis: {
            categories: test,
            title: {
                text: null
            }
        },
```

```
            yAxis: {
                min: 0,
                title: {
                    text: 'Errores',
                    align: 'high'
                },
                labels: {
                    overflow: 'justify'
                }
            },
            tooltip: {
                valueSuffix: ' '
            },
            plotOptions: {
                bar: {
                    dataLabels: {
                        enabled: true
                    }
                }
            },
            legend: {
                layout: 'vertical',
                align: 'right',
                verticalAlign: 'top',
                x: -40,
                y: 80,
                floating: true,
                borderWidth: 1,
                backgroundColor: ((Highcharts.theme && Highcharts.theme.legendBackgroundColor) || '#FFFFFF'),
                shadow: true
            },
            credits: {
                enabled: false
            },
            series: dataOut/*[{
                name: name,
                data: data
            }, {
                name: 'Year 1900',
                data: [1, 5, 2]
            }, {
                name: 'Year 2012',
                data: [1, 4, 0]
            }]*/
        });
        dataIn.type="spline";
        dataIn.marker = {lineWidth: 2,
                lineColor: Highcharts.getOptions().colors[3],
                fillColor: 'white'};
```

```
dataOut[j+1] = dataIn;
$('#container2').highcharts({
    title: {
        text: 'Media de errores de alumnos por ejercicio'
    },
    yAxis: {
        min: 0,
        title: {
            text: 'Errores',
            align: 'middle'
        },
        labels: {
            overflow: 'justify'
        }
    },
    xAxis: {
        categories: test
    },
    series: dataOut/*[{
        type: 'column',
        name: 'Jane',
        data: [3, 2, 1, 3, 4]
    }, {
        type: 'column',
        name: 'John',
        data: [2, 3, 5, 7, 6]
    }, {
        type: 'column',
        name: 'Joe',
        data: [4, 3, 3, 9, 0]
    }, {
        type: 'spline',
        name: 'Average',
        data: [3, 2.67, 3, 6.33, 3.33],
        marker: {
            lineWidth: 2,
            lineColor: Highcharts.getOptions().colors[3],
            fillColor: 'white'
        }
    }]*/
});
<?php
    $h=-1;
    $name="";
    foreach ($result as $item) {
        if($item->order!=$name ){
            if($h>-1){
                echo("dataOutTime[$h] = dataInTime;");
            }
            $h++;
```

```
            echo("j = $h; \n");
            echo("data = [];");
            echo("dataInTime = {type, name, data};");
            echo("dataInTime.type = 'column';");
            echo("\n dataInTime.name='");
            print_r(json_decode(json_encode($item->order), true));
            echo("'; ");
            $name = $item->order;
        }
        if($item->date_end != ''){
            $finalTime = $item->time/60;
            echo("\n dataInTime.data.unshift(");
            print_r($finalTime);
            echo("); ");
        }
    }
    $total = count($result)-1;
    print_r("\n");
    print_r(json_decode(json_encode($result[$total]->order2),true));
    print_r(";");
    echo("dataOutTime[$h] = dataInTime;");;
    echo("data = [];");
    $auxiliar =0;
    foreach($result3 as $item){
        echo("usersTime[$auxiliar] = ");
        print_r(json_decode(json_encode($item->order),true));
        echo(";");
        $auxiliar++;
    }

?>
var mediaTime=[];
i=0;
for(var x=0;x < dataOutTime.length ; x++){
    for(i=0; i < dataOutTime[x].data.length; i++){
        if(x==0){
            mediaTime[i] = 0;
        }
        mediaTime[i] += dataOutTime[x].data[i]/(usersTime[i]);
    }
}
dataInTime={type, name, data};
dataInTime.type="spline";
dataInTime.name="media";
dataInTime.data = mediaTime;
dataInTime.marker = {lineWidth: 2,
            lineColor: Highcharts.getOptions().colors[3],
            fillColor: 'white'};
dataOutTime[j+1] = dataInTime;
```

```
$('#container3').highcharts({
    title: {
        text: 'Media de tiempo de alumnos por ejercicio'
    },yAxis: {
        min: 0,
        title: {
            text: 'Tiempo en minutos',
            align: 'middle'
        },
        labels: {
            overflow: 'justify'
        }
    },
    xAxis: {
        categories: test
    },
    series: dataOutTime
});

progress.name= [];
progress.name = "Avance";
$('#container4').highcharts({
    chart: {
      type: 'bar'
    },
    title: {
        text: 'Avance de los alumnos en el curso'
    },
    xAxis: {
        categories: names,
        title: {
            text: null
        }
    },
    yAxis: {
        min: 0,
        title: {
            text: 'Avance (%)',
            align: 'high'
        },
        max : 100,
        labels: {
            overflow: 'justify'
        }
    },
    legend: {
        text : 'Avance'
    },
    tooltip: {
        valueSuffix: ' %'
```

```
            },
            plotOptions: {
                bar: {
                    dataLabels: {
                        enabled: true
                    }
                }
            },
            credits: {
                enabled: false
            },
            series: [progress]
        });
    });

    </script>

    {!! Html::script('js/highcharts.js') !!}
    {!! Html::script('js/modules/exporting.js') !!}
@endsection
```

D Manual de usuario del módulo de estadísticas

Lo primero es acceder a la zona de estadísticas. Para ello el usuario tiene que ser de tipo "admin", es decir, un profesor. Como aparece en la figura 22, se despliega un menú de la cabecera y aparece la opción "Estadísticas".

Figura 22. Menú desplegable del profesor

Tras pulsar sobre Estadísticas serás redirigido a la página que te muestra los cursos de los cuáles eres profesor y sus respectivos grupos, con la opción de ver los datos y estadísticas de los usuarios inscritos en cada grupo. Esto se ve muy bien en la figura 14. Esta página te da dos opciones: ver alumnos o ver estadísticas.

Si seleccionas el botón "ver alumnos" serás enviado a la página de datos de los alumnos. La figura 15 representa perfectamente esta página. En ella se pueden ver datos de alumnos como nota, errores por ejercicio, tanto de un alumno como la media global, el avance y la posibilidad de descargarte un archivo Excel con el log del alumno seleccionado.

Si seleccionas el botón "ver estadísticas" serás redireccionado a la página donde se visualizan las gráficas, figuras 16-19. En esta página se visualizan datos individuales y globales como tiempos de ejecución de ejercicios, errores de los alumnos por ejercicio y avance del curso de cada alumno. Cada una de las gráficas puede ser descargada en tres formatos: PNG, PDF y JPEG. Además pueden ser directamente impresos desde la página. Las gráficas son totalmente dinámicas y puedes interactuar con ellas. Por ejemplo si quieres retirar un alumno de la gráfica solo has de pulsarlo en la leyenda de la gráfica y si lo que quieres es añadirlo de nuevo tienes que volver a pulsarlo. Esto ayuda mucho en la selección de datos.